UNIVERSIDADE FEDERAL DE VIÇOSA

Automatização do projeto de pontes por meio da estratégia de parametrização: aplicações no cálculo estrutural e em ambiente BIM

Bruno Souza Borges
Magister Scientiae

VIÇOSA - MINAS GERAIS
2025

BRUNO SOUZA BORGES

Automatização do projeto de pontes por meio da estratégia de parametrização: aplicações no cálculo estrutural e em ambiente BIM

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Engenharia Civil, para obtenção do título de *Magister Scientiae*.

Orientador: Diogo Silva de Oliveira

Coorientadores: Kleos M. L. C. Junior
Carol F. Rezende Santos
Jose C. Lopes Ribeiro

VIÇOSA - MINAS GERAIS
2025

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T

B732a 2025

Borges, Bruno Souza, 1999-

Automatização do projeto de pontes por meio da estratégia de parametrização: aplicações no cálculo estrutural e em ambiente BIM / Bruno Souza Borges. – Viçosa, MG, 2025.

1 dissertação eletrônica (533 f.): il. (algumas color.).

Inclui anexos.

Inclui apêndice.

Orientador: Diôgo Silva de Oliveira.

Dissertação (mestrado) - Universidade Federal de Viçosa, Departamento de Engenharia Civil, 2025.

Referências bibliográficas: f. 141-148.

DOI: https://doi.org/10.47328/ufvbbt.2025.493

Modo de acesso: World Wide Web.

1. Pontes - Projetos e construção. 2. Modelagem de informação da construção. 3. Método dos elementos finitos - Processamento de dados. 4. Python (Linguagem de programação de computador). I. Oliveira, Diôgo Silva de, 1988-. II. Universidade Federal de Viçosa. Departamento de Engenharia Civil. Programa de Pós-Graduação em Engenharia Civil. III. Título.

CDD 22. ed. 624.2

Bibliotecário(a) responsável: Euzébio Luiz Pinto CRB-6/3317

BRUNO SOUZA BORGES

Automatização do projeto de pontes por meio da estratégia de parametrização: aplicações no cálculo estrutural e em ambiente BIM

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Engenharia Civil, para obtenção do título de *Magister Scientiae*.

APROVADA: 16 de julho de 2025.

Assentimento:

Bruno Souza Borges
Autor

Diogo Silva de Oliveira
Orientador

Essa dissertação foi assinada digitalmente pelo autor em 04/08/2025 às 18:17:57 e pelo orientador em 05/08/2025 às 09:06:15. As assinaturas têm validade legal, conforme o disposto na Medida Provisória 2.200-2/2001 e na Resolução nº 37/2012 do CONARQ. Para conferir a autenticidade, acesse https://siadoc.ufv.br/validar-documento. No campo 'Código de registro', informe o código **RQAM.PYCI.BQ3M** e clique no botão 'Validar documento'.

AGRADECIMENTOS

Aos meus pais, pelo carinho e incentivo no caminho de meus estudos.

A meu irmão, pelo companheirismo fraterno de sempre.

À Bruna, companheira de vida que sempre apoiou minhas escolhas.

Aos meus amigos pelo incentivo, convivência e conselhos.

Aos professores Diôgo e Carol pelo aprendizado constante e propostas desafiadoras.

Aos amigos do LaBIM, que através das trocas de ideias tornaram a jornada da pesquisa enriquecedora.

À Universidade Federal de Viçosa, pela oportunidade de realizar a pós-graduação.

Ao Departamento Nacional de Infraestrutura de Transportes (DNIT) pela concessão da bolsa de estudos do presente trabalho.

Este trabalho foi realizado com o apoio das seguintes agências de pesquisa brasileiras: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

RESUMO

BORGES, Bruno Souza, M.Sc., Universidade Federal de Viçosa, julho de 2025. **Automatização do projeto de pontes por meio da estratégia de parametrização: aplicações no cálculo estrutural e em ambiente BIM**. Orientador: Diogo Silva de Oliveira. Coorientadores: Kleos Magalhaes Lenz Cesar Junior, Carol Ferreira Rezende Santos e Jose Carlos Lopes Ribeiro.

A complexidade dos projetos de infraestrutura, aliada às exigências normativas de adoção do BIM (Building Information Modeling), demanda soluções mais integradas, padronizadas e automatizadas. Nesta pesquisa, propôs-se uma estratégia de automatização do projeto de pontes, com foco na modelagem paramétrica aplicada tanto à análise estrutural via Método dos Elementos Finitos (MEF) quanto à modelagem BIM. O método envolveu a utilização de linguagem Python para estabelecer comunicação com o CSiBridge por meio de API, possibilitando a geração automatizada de modelos numéricos e a extração dos esforços solicitantes. Paralelamente, foram desenvolvidas rotinas no Dynamo para modelagem paramétrica no Revit com a mesma base de dados de entrada, garantindo coerência entre os modelos. Ao todo, foram gerados 352 modelos numéricos no CSiBridge a partir de mais de 14 mil linhas de código desenvolvidas para organizar, processar e controlar a entrada e saída de dados, além de permitir a execução automatizada das análises estruturais. A extração de esforços viabilizou o dimensionamento automatizado de elementos como longarinas protendidas e metálicas, além da laje do tabuleiro, reduzindo o tempo de projeto e minimizando inconsistências. A abordagem demonstrou-se eficaz para aplicações em projetos de pontes, especialmente naqueles que demandam múltiplas variações geométricas e tipológicas. A integração dos fluxos de trabalho promoveu padronização, rastreabilidade e economia de tempo, contribuindo para a superação de gargalos operacionais típicos de projetos de infraestrutura. Assim, o trabalho representa um avanço no uso coordenado de ferramentas digitais e programação para a modernização do projeto estrutural de pontes.

Palavras-chave: parametrização; bim; dimensionamento; programação; pontes

ABSTRACT

BORGES, Bruno Souza, M.Sc., Universidade Federal de Viçosa, July, 2025. **Bridge design automatization through parametrization strategy: applications in structural analysis and BIM environment**. Adviser: Diogo Silva de Oliveira. Co-advisers: Kleos Magalhaes Lenz Cesar Junior, Carol Ferreira Rezende Santos and Jose Carlos Lopes Ribeiro.

The complexity of infrastructure projects, combined with regulatory requirements for the adoption of Building Information Modeling (BIM), demands more integrated, standardized, and automated solutions. This research proposed a bridge design automation strategy focused on parametric modeling applied both to structural analysis using the Finite Element Method (FEM) and to BIM modeling. The method involved the use of the Python programming language to establish communication with CSiBridge via API, enabling the automated generation of numerical models and the extraction of internal forces. In parallel, routines were developed in Dynamo for parametric modeling in Revit, using the same input database, ensuring coherence between the models. A total of 352 numerical models were generated in CSiBridge, based on over 14,000 lines of code developed to organize, process, and manage data input and output, in addition to enabling automated execution of structural analyses. The soliciting efforts extraction process enabled the automated design of structural elements such as prestressed and steel girders, as well as the deck slab, reducing project time and minimizing inconsistencies. The proposed approach proved to be effective for bridge design applications, especially those requiring multiple geometric and typological variations. The integration of workflows promoted standardization, traceability, and time savings, contributing to the resolution of typical operational bottlenecks in infrastructure projects. Therefore, this work represents an advancement in the coordinated use of digital tools and programming for the modernization of structural bridge design.

Keywords: parametrization; bim; structural design; programming; bridges

SUMÁRIO

1. INTRODUÇÃO 8

1.1 Objetivos 10

1.1.1 Objetivo geral 10

1.1.2 Objetivos específicos 10

1.2 Justificativa 10

1.2.1 O BIM na construção civil: normativas e desafios 11

1.2.2 O cenário acadêmico da aplicação de técnicas de automatização em projetos de pontes 13

1.2.3 Problema de pesquisa 15

2. REVISÃO DE LITERATURA 17

2.1 Modelagem paramétrica 17

2.1.1 Definição geral 17

2.1.2 Potenciais e limitações 20

2.2 O cenário da implementação de tecnologias baseadas em BIM em projetos de infraestrutura 23

2.3 Aplicações da automatização em projetos de pontes 25

2.4 IFC: a abordagem orientada a objetos para a interoperabilidade no BIM 34

2.4.1 Contexto e a linguagem EXPRESS 34

2.4.2 Estrutura do IFC 36

2.4.3 A implementação do IFC em projetos do domínio da infraestrutura 41

2.5 Programação orientada a objetos 42

3. MATERIAIS E MÉTODOS 48

3.1 Definições dos parâmetros de entrada 53

3.2 Automatização da modelagem numérica no CSiBridge 60

3.2.1 Estrutura das planilhas e dos programas elaborados em Python 60

3.2.2 Premissas para a criação dos modelos numéricos 69
3.3 Automatização da modelagem no Revit com o Dynamo 84
3.3.1 Criação de famílias paramétricas 84
3.3.2 Estrutura do grafo desenvolvido 89
4. RESULTADOS E DISCUSSÕES 94
4.1 Modelos numéricos gerados a partir dos códigos desenvolvidos para a conexão API com o CSiBridge utilizando Python 94
4.2 Fluxos de extração de esforços para dimensionamento 99
4.2.1 Longarinas pré-moldadas e protendidas 99
4.2.2 Longarinas metálicas 107
4.2.3 Lajes do tabuleiro 120
4.2.4 Considerações 128
4.3 Modelagem no Revit através do Dynamo 130
4.4 Dificuldades associadas 134
4.4.1 Conexão API com o CSiBridge via Python 134
4.4.2 Conexão com o Revit via Dynamo 135
5. CONCLUSÃO 137
5.1 Geral 137
5.2 Sugestões para trabalhos futuros 139
6. REFERÊNCIAS BIBLIOGRÁFICAS 141
ANEXO A 149
ANEXO B 211
ANEXO C 231
APÊNDICE A 286

1. INTRODUÇÃO

A Modelagem em Elementos Finitos aplicada em pontes permite discretizar sua geometria e definir rigorosamente suas condições de contorno. A adoção de diferentes tipos de elementos finitos (como cascas, placas e barras) e a atribuição de vínculos apropriados possibilitam representar as interações entre diversos elementos estruturais. Essas capacidades, somadas à alta eficiência na determinação de esforços internos tanto em análises estáticas quanto dinâmicas e às sucessivas validações experimentais e de campo consolidaram o Método dos Elementos Finitos (MEF) como ferramenta fundamental na análise estrutural de pontes (BORETTI *et al.*, 2023; CHUNG; SOTELINO, 2006; GIRARDET; BOTON, 2021).

Da mesma forma, com os recentes avanços na entrega de projetos, o *Building Information Modeling* (BIM) vem sendo cada vez mais aceito como padrão na engenharia civil. A criação de um modelo paramétrico nesse ambiente pode permitir o intercâmbio de informações multilateral com outros sistemas, a integração multidisciplinar de projetistas (colaboração) e um ambiente mais consistente de informações em relação aos elementos modelados quando comparado com a elaboração tradicional de projetos. Geralmente, essa última é composta por desenhos e vistas que não se comunicam diretamente, já que a entrega não é um modelo funcionando como um repositório de informações, mas sim desenhos isolados desenvolvidos em ambientes CAD, *Computer Aided Design* (BORRMANN *et al.*, 2018).

Uma das premissas do BIM é a modelagem paramétrica que, baseada em objetos, estrutura os processos para a criação de modelos, sejam eles com fins para análise estrutural ou não. Nessa metodologia, tem-se a modelagem de componentes com representação geométrica e parâmetros que governam seu comportamento. Com isso, tem-se flexibilidade e consistência, já que o modelo é reativo a alterações (SACKS *et al.*, 2018).

Em programas de análise estrutural, a definição dos parâmetros permite a criação de modelos complexos, facilitando a realização de análises detalhadas e precisas que consideram fatores como a interação solo-estrutura, efeitos de segunda

ordem, e não linearidades materiais e geométricas. No ambiente BIM, esses parâmetros podem abranger propriedades físicas dos materiais, dimensões geométricas e estimativas de orçamento. Ajustando esses parâmetros, é viável criar modelos detalhados que consideram interações complexas entre diferentes componentes do projeto. Dessa forma, o comportamento do modelo depende integralmente da inserção e ajuste desses parâmetros (SACKS *et al.*, 2018; TANG *et al.*, 2020).

À medida que a demanda por dados requisitados pelos *software* de modelagem aumenta, uma vez que estão em constante desenvolvimento, surge o desafio de delegar ao usuário a responsabilidade pelo *input* de alguns parâmetros. Deixar essa tarefa a cargo do usuário pode resultar em inconsistências, omissões e erros de entrada, comprometendo a precisão e a confiabilidade dos modelos (FANG *et al.*, 2025; MENG *et al.*, 2023).

A alimentação automática desses parâmetros aumenta a eficiência e minimiza erros em projetos ao empregar técnicas de validação, possibilitando que engenheiros estruturais foquem em aspectos técnicos do projeto, como a análise de resultados e a tomada de decisões. Além disso, a automatização proporciona padronização, reduz a variabilidade e necessidade de retrabalho, contribuindo para projetos mais rápidos e econômicos (SZALAI; AJTAYNÉ KÁROLYFI, 2024).

Neste contexto, a integração de linguagens de programação para estabelecer a conexão entre dois ou mais *software*, conhecida como *Application Programming Interfaces* (API's), emerge como uma alternativa robusta e adaptável para superar as limitações da modelagem manual. O uso de um ambiente de programação para estabelecer essa conexão com *software* de modelagem permite a automatização de tarefas repetitivas, simplificam operações complexas e garantem maior controle sobre os parâmetros dos modelos. Ainda, a automatização por meio de API's possibilita a personalização dos fluxos de trabalho, permitindo que empresas adaptem seus processos de modelagem conforme suas necessidades específicas. Assim, viabilizam-se inovações na maneira como os modelos são desenvolvidos, compartilhados e analisados, além de tornar a extração direta de informações possível (BORETTI *et al.*, 2023; GIRARDET; BOTON, 2021; SARDONE *et al.*, 2022).

Com a constante evolução das distintas tecnologias de computação e inteligência artificial e a crescente demanda por soluções integradas, o intercâmbio de dados entre *software* se torna uma abordagem promissora em projetos de pontes, uma vez que a interoperabilidade aberta ainda não é plena para esse tipo de projeto (BAZÁN *et al.*, 2020; KHOSHAMADI *et al.*, 2023). Diante do exposto, o presente trabalho visa explorar conexões API's para automatizar tanto a modelagem quanto processos de dimensionamento na elaboração de projetos estruturais de pontes.

1.1 Objetivos

1.1.1 Objetivo geral

Este trabalho tem como objetivo geral explorar um método de automatização da modelagem paramétrica de pontes para fins de análise estrutural via Método dos Elementos Finitos e geração de modelos BIM.

1.1.2 Objetivos específicos

a) A criação de modelos para análise estrutural de super e mesoestrutura de pontes através da conexão API com a linguagem de programação *Python* e o *software Computers and Structures, Inc. CSiBridge*;

b) O preenchimento de planilhas a partir da extração de resultados do modelo numérico para automatizar o processo de dimensionamento;

c) Desenvolvimento de códigos no ambiente de programação visual do *Autodesk Dynamo* para a modelagem paramétrica de pontes utilizando a metodologia BIM no ambiente do *Autodesk Revit* com a mesma base de dados considerada no item (a);

1.2 Justificativa

A justificativa desta pesquisa baseia-se na exigência crescente de uso do BIM no Brasil, estabelecida por normas como o Decreto 10.306 de 2020 e a Lei 14.133 de 2021 (BRASIL, 2020a; BRASIL, 2021). Projetos de pontes apresentam dificuldades específicas, como geometrias complexas, fluxos fragmentados e baixa interoperabilidade. Embora o emprego de tecnologias de digitalização tragam ganhos

em eficiência e qualidade, a literatura mostra que há poucos estudos integrando modelagem geométrica, análise estrutural, APIs e programação visual para pontes.

Nesse contexto, a justificativa foi organizada em três tópicos para apresentar, de forma gradual, as razões que sustentam o estudo. Primeiro, aborda-se o cenário normativo, que cria a demanda prática pela adoção do BIM. Depois, analisa-se o estado atual da pesquisa, destacando as lacunas existentes na integração de ferramentas para projetos de pontes. Por fim, delimita-se o problema específico que motivou o desenvolvimento do método proposto, orientado à automação e integração de fluxos paramétricos e analíticos.

1.2.1 O BIM na construção civil: normativas e desafios

A digitalização dos processos que envolvem a construção é uma demanda global, impulsionada por políticas públicas e normativas que visam aumentar a eficiência, reduzir custos e padronizar os processos. A governança brasileira, através do Decreto nº 10.306, instituiu a Estratégia Nacional de Disseminação do BIM, que exige a adoção da metodologia em projetos na esfera pública até 2028, caso considerados de média ou grande relevância para disseminação da tecnologia. Determina-se ainda o uso de formatos abertos (não proprietários) para entrega dos arquivos eletrônicos que compõem projetos de arquitetura e engenharia, estimulando a interoperabilidade (BRASIL, 2020a). Essa diretriz é reforçada pela Lei de Licitações e Contratos Administrativos nº 14.133/2021 que, em seu artigo 19, inciso V, §3º, estipula que:

> [...] Nas licitações de obras e serviços de engenharia e arquitetura, sempre que adequada ao objeto da licitação, será preferencialmente adotada a Modelagem da Informação da Construção (Building Information Modelling - BIM) ou tecnologias e processos integrados similares ou mais avançados que venham a substituí-la (BRASIL, 2021).

Embora o termo “preferencialmente” seja comum em atos administrativos, no Direito ele é comumente interpretado como uma obrigatoriedade. Nessa perspectiva, diversas unidades da federação estabeleceram decretos que promovem a disseminação do BIM, estendendo essa exigência às entidades administrativas estaduais e reforçando a importância do uso de metodologias BIM e formatos abertos

na construção civil (MAESTRO, 2024; MINAS GERAIS, 2021; RIO DE JANEIRO, 2018; RIO GRANDE DO SUL, 2022; SANTA CATARINA, 2021; PARANÁ, 2022).

A padronização na classificação dos objetos em ambiente BIM é fundamental para eliminar ambiguidades na comunicação entre as partes interessadas (*stakeholders)* e enriquecer o modelo com informações para aplicações avançadas que envolvem a metodologia. Inspirada na ISO 19650, a Associação Brasileira de Normas Técnicas (ABNT) implementou iniciativas para estabelecer um sistema de classificação nacional, adaptado às particularidades de idioma, materiais e sistemas construtivos utilizados na indústria brasileira na NBR 15965 (ABNT, 2011). Por meio de códigos definidos nas tabelas de classificação, é possível atribuir identificadores uniformes aos objetos dos modelos, assegurando uma interpretação consistente e facilitando a colaboração entre as partes envolvidas (ABNT, 2011; BLOCH; SACKS, 2018; ISO, 2018).

As iniciativas destacam a importância de superar a histórica fragmentação dos processos de projeto, marcados pelo uso de ferramentas isoladas e fluxos de trabalho desconectados. Essa abordagem tradicional pode comprometer a integridade das informações ao longo de todas as fases de um projeto e tornar o alinhamento dos dados multidisciplinares oneroso, principalmente na ausência de interoperabilidade (BORRMANN *et al.*, 2018)

Esse desafio se torna ainda mais crítico em projetos de infraestrutura, os quais frequentemente carecem de ferramentas computacionais eficazes para a implementação da metodologia BIM, bem como de esquemas de dados padronizados que assegurem a comunicação integrada entre disciplinas. Como consequência, observa-se a adoção de múltiplos *software* especializados, geralmente desprovidos de interoperabilidade, para a modelagem independente de aspectos como traçado, geometria/terraplenagem, arquitetura, análise estrutural, dentre outros. Em vez de um repositório unificado de dados, configura-se um ambiente fragmentado, com informações dispersas e desarticuladas (COSTIN *et al.*, 2018)

Ademais, muitas dessas soluções computacionais foram desenvolvidas sem a consideração adequada às complexidades associadas à integração de múltiplas fontes de dados. Isso resulta em sistemas compartimentalizados (*stovepipe systems*), caracterizados pela limitada interoperabilidade e baixa capacidade de compartilhar

informações com outras plataformas através do uso de formatos de arquivo proprietários para transcrição e tradução. Tal cenário compromete a eficiência na comunicação entre equipes, especialmente em projetos de infraestrutura de transporte, em que a troca contínua de dados é essencial para o progresso coordenado das atividades por conta do caráter disciplinar (COSTIN *et al.*, 2018; GEORGOULA, 2019; PIRDAVANI *et al.*, 2023).

Em um cenário onde a indústria AECO (Arquitetura, Engenharia, Construção e Operação) busca incessantemente otimizar custos, reduzir o desperdício de materiais e acelerar os prazos tanto de projeto quanto análise, a modelagem paramétrica surge como uma resposta promissora para o setor de infraestrutura, especialmente em projetos de pontes. As ferramentas digitais atuais expandem a capacidade de criar geometrias inovadoras e detalhes complexos, mas a digitalização da indústria da infraestrutura ainda enfrenta desafios relacionados a custos iniciais elevados na implementação, falta de interoperabilidade adequada e de fluxos de trabalho eficientes baseados em metodologias BIM (KHOSHAMADI *et al.*, 2023).

Com isso, a pesquisa e aplicação de metodologias BIM, bem como a modelagem paramétrica, no setor de infraestrutura têm se expandido. Esse movimento indica uma tendência clara de adoção dessas tecnologias como elementos fundamentais para o desenvolvimento de projetos mais eficientes, econômicos e sustentáveis (COSTIN *et al.*, 2018).

1.2.2 O cenário acadêmico da aplicação de técnicas de automatização em projetos de pontes

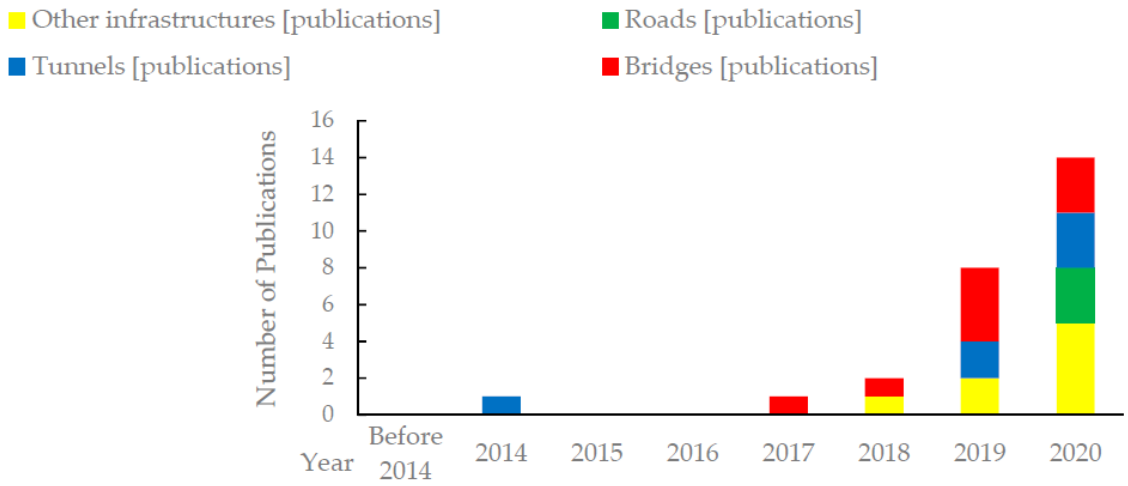
Na base de dados da Scopus foram identificados apenas três estudos que empregam conexão API com o CSiBridge para modelagem estrutural, todos com objetivos muito específicos que limitam sua aplicabilidade em contextos distintos. Cada trabalho incorporou um algoritmo de otimização à modelagem, buscando a configuração ideal da estrutura conforme o tipo de análise (ARELLANO; TOLENTINO; GÓMEZ, 2019; ARTAR; CARBAS, 2023; GARCÍA-SEGURA; YEPES, 2016).

Na revisão de literatura feita por Collao *et al.* (2021) foi estabelecido o estado da arte da aplicação de ferramentas de programação visual (VP) em projetos de infraestrutura BIM e comparando com o cenário de sua aplicação em projetos de edifícios. Foram identificados um total de 153 estudos, dos quais 26 (17%) estavam

associados a projetos de infraestrutura. As primeiras contribuições para ambos os tipos de projetos foram em 2014, e o período de publicações analisado estendeu-se até 2020. O estudo revelou um interesse crescente na aplicação de ferramentas VP em projetos de engenharia civil, com o Dynamo sendo a ferramenta mais utilizada (80,77% das publicações). A maioria das aplicações (80,77%) concentrou-se nas fases de projeto, revelando uma limitação quanto às fases do ciclo de vida da estrutura.

Na Figura 1.1 ilustra-se a distribuição dos estudos revisados de acordo com o tipo de infraestrutura. Observa-se que os projetos de pontes lideram as aplicações de ferramentas de programação visual em projetos de infraestrutura, representando cerca de 34,62% das publicações (9 publicações). Essa predominância pode ser explicada pelas características geométricas complexas das pontes, como pilares, encontros, lajes e vigas, sendo especialmente adequadas para modelagem paramétrica e automatização através de VP.

Figura 1.1 – Distribuição das publicações analisadas pela revisão de Collao *et al.* (2021) até o ano de 2020 segmentadas conforme o tipo de infraestrutura



	Before 2014	2014	2015	2016	2017	2018	2019	2020
Bridges [publications]	0	0	0	0	1	1	4	3
Tunnels [publications]	0	1	0	0	0	0	2	3
Roads [publications]	0	0	0	0	0	0	0	3
Other infrastructures [publications]	0	0	0	0	0	1	2	5

Fonte: Collao *et al* (2021)

Observa-se, portanto, que, embora o número de estudos envolvendo a integração de APIs com o CSiBridge e a aplicação de programação visual em projetos

de infraestrutura ainda seja reduzido, há um interesse crescente por essas abordagens na literatura. Tanto os trabalhos identificados na base Scopus quanto a revisão de Collao *et al.* (2021) evidenciam um campo ainda em estágio inicial, porém com expansão progressiva e relevante potencial de desenvolvimento.

Além disso, a limitada disponibilidade de soluções plenamente interoperáveis para projetos de pontes dificulta a integração eficiente entre a modelagem geométrica e a análise estrutural. Nesse cenário, não foram identificados estudos que abordem de forma integrada a modelagem geométrica, a modelagem analítica e a análise estrutural aplicadas especificamente a projetos de pontes. Tampouco foram encontrados trabalhos que explorem, de maneira conjunta, as ferramentas propostas nesta pesquisa.

1.2.3 Problema de pesquisa

O atual estudo responde as demandas do TED (Termo de Execução Descentralizada) nº 703/2020 na meta de elaboração de um catálogo digital de pontes (BRASIL, 2020b). Nesse catálogo, são estudadas várias tipologias, cada uma com particularidades, como comprimento do vão, largura de seção transversal, dimensões dos elementos estruturais, dentre outras. Para realizar a análise de super e mesoestrutura de todas as tipologias, foi necessária a criação e processamento de vários modelos, tornando a entrada manual de parâmetros em cada instância inviável, além de comprometer a rastreabilidade em eventuais adaptações de modelos existentes.

A integração via API com o *software* de análise estrutural possibilitou a extração automatizada de esforços e o dimensionamento automático de alguns elementos, assegurando maior consistência na definição de suas dimensões no âmbito do catálogo. Paralelamente, a utilização do Dynamo, por meio de programação visual, possibilitou a modelagem automatizada dos elementos no Revit, utilizando os mesmos parâmetros geométricos adotados nos modelos numéricos. Essa abordagem não apenas elimina redundâncias na entrada de dados, mas também reduz a possibilidade de erros de transcrição, além de garantir a coerência geométrica entre o modelo analítico e o modelo BIM.

Neste cenário, a presente pesquisa se justifica para mitigar a dependência em processos manuais nos fluxos de projetos de pontes. Essa dependência pode gerar

retrabalho, inconsistências de dados e comprometem tanto a produtividade quanto a confiabilidade dos resultados. Embora conexões API's com o CSiBridge e o uso do Dynamo ofereçam o potencial de automatizar a geração de geometrias paramétricas e de integrar de forma sistemática fluxos de entrada e saída de dados (*input* e *output*, respectivamente) com linguagens de programação, verifica-se uma escassez de estudos que exploram essas aplicações em pontes.

Com base em todo o exposto, conclui-se que há uma confluência significativa entre os avanços tecnológicos da modelagem paramétrica, a consolidação do BIM como metodologia normativa na construção civil e a crescente complexidade dos projetos de infraestrutura, especialmente no setor de pontes. O MEF, amplamente consolidado como um método fundamental para análises estruturais, aliado à modelagem paramétrica baseada em objetos, oferece um ambiente propício à automatização e à integração de processos fragmentados. O uso de API's surge, nesse cenário, como um meio estratégico para interligar plataformas especializadas, reduzir erros manuais, garantir o intercâmbio de dados e viabilizar fluxos de trabalho mais eficientes e adaptáveis.

Ainda assim, observa-se uma lacuna relevante na literatura e no setor produtivo no que diz respeito à sistematização da aplicação desses métodos em projetos de infraestrutura, especialmente quanto à modelagem paramétrica estruturada por regras e à integração com ferramentas de análise. Esse panorama evidencia a necessidade de pesquisas que explorem o potencial da conexão entre *software* por meio de programação, promovendo soluções customizáveis e escaláveis.

Dessa forma, o presente trabalho visa contribuir de maneira estratégica para a consolidação de um novo paradigma na elaboração de projetos de pontes centrado na automatização e integração coordenada de ferramentas digitais. Ao empregar a modelagem paramétrica e estabelecer uma comunicação estruturada nos *software* por meio de API, a proposta avança na superação de gargalos operacionais que ainda limitam a digitalização plena no setor de infraestrutura.

2. REVISÃO DE LITERATURA

A revisão de literatura aborda os fundamentos teóricos e o estado da arte na integração de tecnologias digitais no projeto de pontes, com ênfase na parametrização como estratégia central de automatização. Inicialmente, serão discutidos os conceitos, princípios e estrutura da modelagem paramétrica, destacando seu papel no BIM para flexibilizar projetos e garantir consistência geométrica e comportamental, bem como suas limitações em geometrias complexas. Em seguida, será analisado o panorama atual da adoção de tecnologias baseadas em BIM no setor de infraestrutura. Na sequência, serão exploradas as aplicações de ferramentas computacionais, como API's, Dynamo e algoritmos de otimização, na modelagem, análise e gestão de pontes, com destaque para integrações entre *software* de análise estrutural e ambientes BIM. Posteriormente, será discutido o esquema de dados IFC (*Industry Foundation Classes*) como padrão aberto para interoperabilidade, abordando sua estrutura orientada a objetos, arquitetura de dados em camadas, extensões recentes e obstáculos técnicos à sua implementação, visando projetos de pontes. Por fim, serão apresentados os princípios da Programação Orientada a Objetos (POO) e sua aplicação na modelagem paramétrica e no desenvolvimento de APIs e fluxos de trabalho customizados.

2.1 Modelagem paramétrica

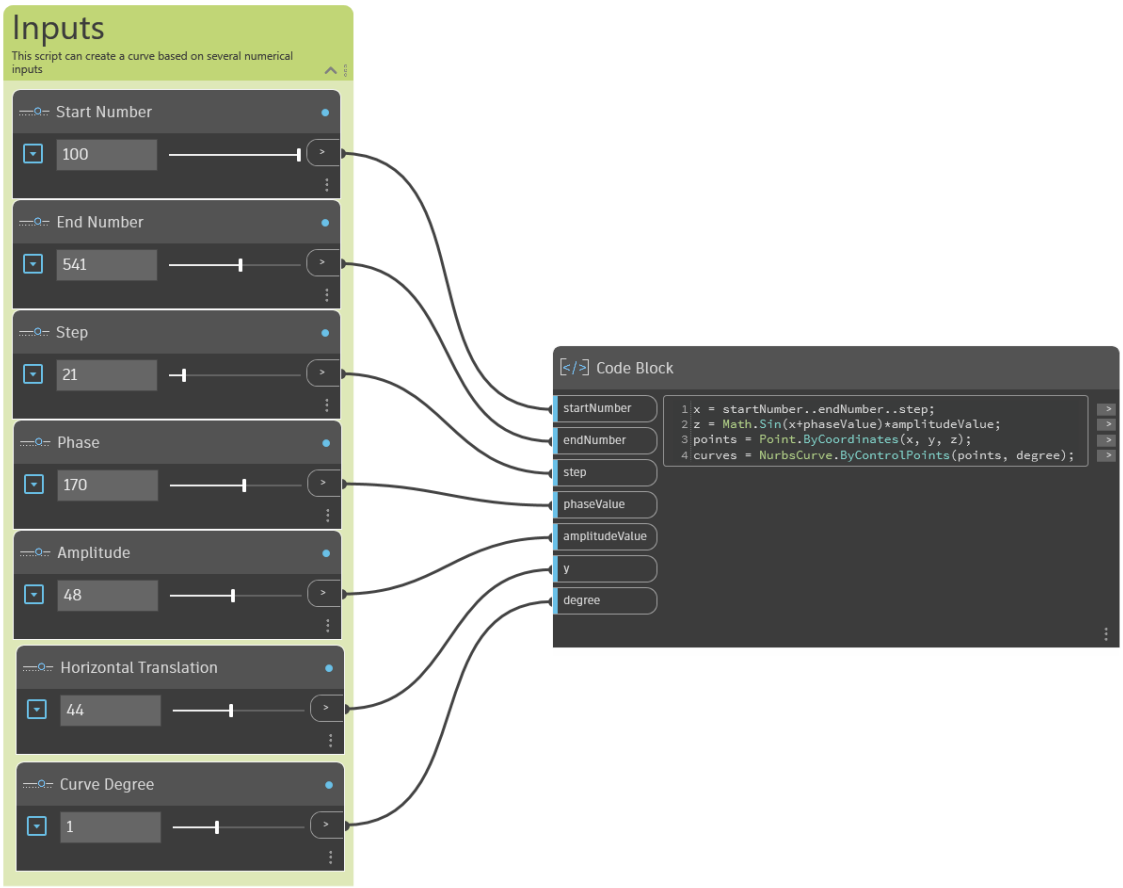
2.1.1 Definição geral

Em um modelo tridimensional gerado por modelagem direta, o usuário ajusta parâmetros como dimensões e posições relativas para obter a geometria desejada. Já em um modelo paramétrico, é possível parametrizar os objetos além da geometria, permitindo que eles não sejam estáticos e que variações nos parâmetros controlem seus dados e comportamentos (GIRARDET; BOTON, 2021). Como esses objetos possuem atributos e comportamentos pré-definidos, bem como relações com outros objetos, há uma analogia com a estruturação da POO.

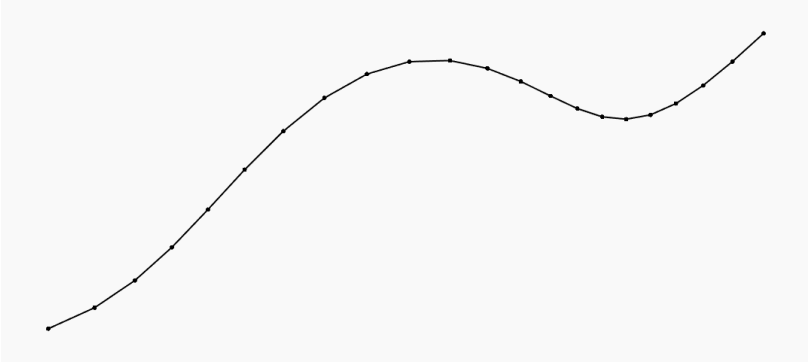
Agboban (2002) se refere a um modelo paramétrico como uma estrutura dupla composta pela especificação do modelo e pelo modelo numérico gerado. Com isso, o último é mais fácil de ser entendido, já que diferentes vistas podem ser extraídas dele,

enquanto o modelo de especificação é mais informativo e maleável, uma vez que contém todas as informações do modelo. Um exemplo disso é uma modelagem de uma curva realizada no Dynamo, com a programação visual sendo a especificação do modelo e o resultado o modelo numérico, conforme ilustrado na Figura 2.1.

Figura 2.1 – Especificação do modelo paramétrico de uma curva no Dynamo por meio de *inputs* e cálculos (a), e respectivo modelo numérico gerado (b)



(a)



(b)

Conforme Woodbury (2010), a modelagem paramétrica é considerada como um sistema de propagação (também conhecido como geometria associativa), ou seja,

uma interconexão de elementos que se relacionam de forma coordenada e que armazenam dados, controlando tanto seu comportamento local quanto global. Por conta disso, é também conhecida como modelagem de restrições, sendo os parâmetros dos objetos regidos por expressões restritivas, que podem envolver operadores matemáticos, funções e até mesmo objetos.

Bouguessa, Forgues e Doré (2013) defendem que um modelo paramétrico deve incluir representação geométrica tridimensional, objetos paramétricos e informação, características primordiais do BIM. Seu objetivo principal é fornecer uma representação virtual de uma estrutura que possa ser entendida pelo usuário, além de fornecer informação suficiente para descrever os elementos que compõem o modelo.

Complementando, conforme Garagnani e Manferdini (2013), a divisão semântica e topologia dos elementos, que define os arranjos e relações, de um modelo paramétrico são recursos que permitem que os objetos sejam “inteligentes”, ou seja, “conscientes” do que são e com quais objetos podem interagir.

Para Sacks *et al.* (2018), objetos paramétricos são constituídos por:

a) Definições geométricas e dados associados, acompanhados de regras;

b) Geometria integrada, sem redundâncias, que garante a consistência entre diferentes visualizações do objeto;

c) Regras paramétricas capazes de ajustar automaticamente os objetos conforme as interações entre seus elementos;

d) Propagação de alterações: modificações em um subcomponente se refletem em todo o objeto;

e) Mecanismos que identificam quando uma mudança compromete a viabilidade da construção do objeto.

Os parâmetros dos objetos possuem categorias definidas; cada combinação de valores desses parâmetros gera uma instância distinta do elemento. Na interface do *software* Revit, a "família" representa um objeto base não instanciável, responsável por definir os parâmetros que regem seu comportamento geral. A partir dela, são gerados os tipos de família, que funcionam como subclasses com valores de parâmetros já definidos e que podem, de fato, ser instanciados pelo usuário no

modelo. As instâncias, por sua vez, são tratadas separadamente, possuindo tanto parâmetros de tipo, herdados do tipo de família e compartilhados por todas as instâncias daquele tipo, quanto parâmetros de instância, que controlam características específicas de cada objeto individualmente no projeto. Embora em muitos casos a relação entre parâmetros e geometria seja direta, ela pode também ser complexa, permitindo gerar formas sofisticadas por meio de lógicas paramétricas. Em essência, todo projeto cujo comportamento e forma dependam de parâmetros configuráveis se enquadra na definição de modelagem paramétrica, independentemente de empregar ou não geometria associativa (CAETANO; SANTOS; LEITÃO, 2020).

Duas abordagens correlatas ao tema são a modelagem generativa e a modelagem algorítmica: a generativa aplica regras e instruções executadas iterativamente com mais autonomia que na modelagem paramétrica até um critério de parada. Isso possibilita a geração de geometrias complexas a partir de *inputs* simples, mas apresentando rastreabilidade limitada devido à opacidade na relação entre código e resultado. Esta não-rastreabilidade é uma das razões pelas quais os métodos de design generativo podem produzir resultados inesperados. Já a modelagem algorítmica, considerada um subtipo da generativa, mantém essa relação explícita, de modo que cada etapa do algoritmo corresponde diretamente a uma transformação geométrica. Por conta disso, oferece maior controle em comparação com outros métodos generativos, facilitando a depuração (*debugging*) e manutenção (CAETANO; SANTOS; LEITÃO, 2020).

2.1.2 Potenciais e limitações

A modelagem paramétrica acelera o processo de projeto ao permitir a exploração de múltiplas soluções sem a necessidade de reconstruir modelos do zero, bastando ajustar determinados parâmetros. A incorporação de regras paramétricas nos objetos contribui para evitar erros de posicionamento e garantir a integridade das informações contidas nos elementos do projeto. Essa abordagem é especialmente relevante na área de infraestrutura, onde as geometrias tendem a ser complexas e os modelos desenvolvidos muitas vezes não podem ser reaproveitados em outros projetos devido à sua especificidade (KHOSHAMADI *et al.*, 2023; GIRARDET; BOTON, 2021). Como um dos pilares tecnológicos do BIM, a modelagem paramétrica é essencial para estabelecer um fluxo de trabalho eficiente ao possibilitar a

reutilização das informações, reduzir erros e otimizar o tempo de execução dos projetos (SACKS *et al.*, 2018).

A criação de vários modelos nas fases iniciais de projeto possibilita a apresentação de soluções mais precisas. Isso permite visualizar e compreender os volumes e as proporções, facilitando a discussão de possíveis alterações de maneira ágil e coordenada entre os envolvidos no projeto (GIRARDET; BOTON, 2021). Com a geração de múltiplos modelos, a avaliação de suas respostas pode ser automatizada conforme o tipo de análise a ser conduzida. Estudos de otimização são frequentemente objeto de estudo, de forma a melhorar o desempenho multidisciplinar do projeto e minimizar desperdícios de materiais. Em projetos estruturais, a modelagem paramétrica é explorada para a elaboração e análise de modelos em elementos finitos (BORETTI *et al.*, 2023; FANG *et al.*, 2025; GAN *et al.*, 2019; SARDONE *et al.*, 2022).

O uso de ferramentas de programação visual, como *Robert McNeel & Associates Grasshopper* e Dynamo, para automatizar a elaboração de modelos paramétricos em *software* BIM torna a modelagem mais acessível ao estabelecer uma camada de abstração entre o usuário e o desenvolvedor. Enquanto linguagens de programação baseadas em texto exigem a escrita de linhas de código que obedecem a uma sintaxe específica, as linguagens visuais possibilitam a manipulação de elementos gráficos, oferecendo maior legibilidade e facilidade de uso. Dessa forma, os códigos gerados nessas plataformas podem ser empregados por profissionais sem conhecimento avançado em programação para a criação de modelos conforme as necessidades dos projetos (VALINEJADSHOUBI *et al.*, 2024).

Diante do exposto, a modelagem paramétrica transcende a mera manipulação geométrica, apresentando-se como uma abordagem inovadora que integra inteligência, flexibilidade e adaptabilidade aos modelos virtuais. Ao incorporar informações, comportamentos e restrições matemáticas, a parametrização possibilita a criação de projetos dinâmicos e inter-relacionados, permitindo análises e otimizações sob múltiplas perspectivas. Em consonância com os princípios da POO, essa metodologia reforça a eficácia do BIM e impulsiona a evolução dos processos de engenharia. Além disso, sua estrutura baseada em regras e interconexões garante

a consistência e integridade dos modelos, evitando redundâncias e assegurando que alterações em subcomponentes sejam refletidas em toda a estrutura.

A modelagem paramétrica, ao alinhar-se a um sistema de propagação de restrições e interações, permite a criação de objetos inteligentes que respondem a mudanças de forma coordenada, otimizando tanto o controle local quanto global dos elementos. A abordagem não apenas aprimora a precisão da representação virtual, mas também proporciona modelos ricos em informação, adaptáveis e eficientes, consolidando-se como uma ferramenta essencial para o desenvolvimento de projetos complexos e inovadores, como frequentemente é o caso em projetos de infraestrutura de transportes.

Apesar disso, algumas limitações da implementação da tecnologia em projetos de pontes podem ser mencionadas (GIRARDET E BOTON, 2021; KHOSHAMADI *et al.*, 2023):

a) Há relatos na dificuldade de se estabelecer a modelagem paramétrica em elementos com dimensões muito reduzidas;

b) Adicionar novos parâmetros para incluir um maior nível de detalhe pode levar a problemas como a complexidade elevada do modelo, dificultando a compreensão e a manipulação de seus dados, podendo comprometer o comportamento local e global dos objetos;

c) Dependendo do resultado de modelagem desejado, pode ser necessário um conjunto abrangente de regras bem definidas antes que os projetistas possam usar o método de forma produtiva;

d) A complexidade de certos elementos, como lajes de transição, aparelhos de apoio e encontros, com formas geométricas raramente idênticas entre diferentes projetos, torna a criação de uma lista de parâmetros quase impossível ou muito trabalhosa, exigindo um número extremamente grande de opções. Nesse mesmo cenário, por conta da grande variedade de tipologias estruturais e configuração de seções transversais possíveis de pontes, uma quantidade elevada de parâmetros teria de ser levantada para atendimento pleno de todas as possíveis geometrias.

2.2 O cenário da implementação de tecnologias baseadas em BIM em projetos de infraestrutura

Khoshamadi *et al.* (2023) conduziram uma revisão sistemática da literatura sobre a aplicação da modelagem paramétrica e generativa em projetos de infraestrutura. A pesquisa inicial identificou 583 publicações, que foram submetidas a um processo de triagem em múltiplas etapas, resultando em 41 artigos selecionados para análise aprofundada. Os resultados evidenciam um interesse crescente na adoção de modelagem paramétrica no setor, especialmente em obras de túneis, pontes e rodovias. Contudo, a análise da literatura revela uma escassez no tema e de estudos que abordem as técnicas de engenharia sob a perspectiva da abordagem paramétrica e generativa. Ou seja, apesar de haver aplicações recorrentes na literatura utilizando-se a tecnologia no ramo da infraestrutura, como por exemplo em ambientes BIM, observa-se um foco limitado nos parâmetros, como restrições paramétricas e as operações de modelagem.

Dessa forma, apesar do surgimento de aplicações pontuais dessas abordagens, persiste uma lacuna significativa quanto à sistematização, avaliação crítica e consolidação do conhecimento relacionado aos métodos e ferramentas no contexto da infraestrutura. Essa deficiência se deve, em parte, à diversidade dos projetos e às variações na forma como a modelagem paramétrica é implementada nesse contexto. Os desafios são amplos e incluem desde a resistência por parte dos profissionais da indústria AEC, que muitas vezes demonstram baixa familiaridade ou relutância em adotar ferramentas modernas que demandam recursos humanos e materiais significativos, até limitações técnicas da própria modelagem. Embora o processo seja reconhecido por sua precisão e rapidez, ainda não é adequado para geometrias com elevado nível de detalhamento. A adição de novos parâmetros para alcançar esse nível de detalhe, embora possível, pode resultar em modelos excessivamente complexos e de difícil interpretação. Além disso, o uso eficiente da modelagem paramétrica exige um conjunto bem definido de regras e diretrizes, o que ainda não é amplamente estabelecido (KHOSHAMADI *et al.*, 2023).

Diversos estudos têm contribuído para consolidar o estado da arte das metodologias BIM no setor de infraestrutura. Entre os principais potenciais da adoção do BIM nesses projetos, tem-se que (ABDAL NOOR; YI, 2018; BRADLEY *et al.*, 2016;

CEPA *et al.*, 2023; CHENG; LU; DENG, 2016; COSTIN *et al.*, 2018; HONGHONG *et al.*, 2023; PIRDAVANI *et al.*, 2023; SALZANO *et al.*, 2023; SHOU *et al.*, 2014):

a) A adoção do BIM, embora demande investimentos iniciais, resulta em economias substanciais ao longo das fases de construção e operação, sendo amplamente vantajosa em comparação aos métodos convencionais. Estimativas de custos mais precisas, aliadas à automação de processos e à melhor coordenação entre disciplinas, contribuem para a redução de prazos e custos dos empreendimentos de infraestrutura. Além disso, o uso do BIM facilita a captura e o armazenamento de dados de forma precisa e abrangente, reduzindo o esforço manual e servindo de base confiável para análises futuras, o que potencializa economias em projetos subsequentes.

b) O BIM aprimora significativamente a gestão da informação ao garantir consistência, rastreabilidade e integridade dos dados ao longo de todo o ciclo de vida do projeto. A tecnologia também permite a avaliação em tempo real de critérios de projeto, a verificação automática de regras e a detecção precoce de falhas, evitando correções onerosas durante a execução do projeto. Sua representação visual detalhada favorece a colaboração entre os envolvidos e a qualidade final do projeto;

c) Quando integrado a ferramentas como a Análise do Ciclo de Vida (LCA, *Lifecyle Assessment*), o BIM permite a gestão de grandes volumes de dados heterogêneos e fornece suporte à tomada de decisões sustentáveis, ao quantificar o impacto ambiental dos componentes desde as fases iniciais do projeto. Isso evidencia seu potencial não apenas para aumentar a eficiência, mas também para promover a sustentabilidade no setor de infraestrutura;

d) Dado que o setor público é o principal proprietário e gestor das infraestruturas viárias e de transporte, ele se apresenta como o maior beneficiário da implementação do BIM em projetos de infraestrutura, especialmente devido à sua capacidade de garantir previsibilidade, controle e transparência na gestão dos ativos públicos;

Da mesma forma, as limitações e lacunas de pesquisa observadas em grande parte dos estudos destacam que:

a) A dificuldade em trocar informações de forma eficiente e sem perda de dados entre diferentes *software*, formatos de arquivo e as diversas partes interessadas ao longo do ciclo de vida do projeto é o obstáculo primordial;

b) A ausência de padrões bem definidos, específicos para projetos de infraestrutura, incluindo esquemas de dados e diretrizes legais, é um desafio recorrente. A falta de um consenso em torno de um padrão de informação da indústria, como IFC, dificulta a implementação consistente do BIM;

c) O investimento inicial em *software*, *hardware* de alto desempenho necessário para lidar com grandes volumes de dados, treinamento de pessoal e desenvolvimento de modelos BIM é considerado dispendioso. Há preocupações sobre o retorno sobre o investimento, especialmente nas fases iniciais da adoção do BIM, com alguns estudos até mostrando impactos negativos nos custos após a primeira implementação;

d) A falta de profissionais com as habilidades e o conhecimento necessários para aplicar o BIM de forma eficaz em projetos de infraestrutura é uma barreira relevante. Isso inclui a necessidade de treinar as partes interessadas e superar a falta de conhecimento sobre as tecnologias e métodos BIM.

Diante dos diversos potenciais e limitações identificados, conclui-se que a aplicação de tecnologias baseadas em BIM nos projetos de infraestrutura representa uma ferramenta promissora para aprimorar a eficiência, precisão e colaboração entre os *stakeholders*. Entretanto, a consolidação de seus benefícios ainda depende da superação de desafios significativos, como a falta de interoperabilidade, a carência de profissionais capacitados e os altos custos iniciais de adoção. Assim, embora os avanços sejam notáveis, a adoção plena do BIM na infraestrutura requer esforços contínuos em capacitação, padronização e desenvolvimento tecnológico.

2.3 Aplicações da automatização em projetos de pontes

A modelagem geométrica de uma superestrutura de ponte geralmente é suficiente utilizando-se varreduras das seções transversais ao longo de seu alinhamento, enquanto a modelagem da mesoestrutura, especialmente os encontros, requer um maior número de operações (Obergrießer, 2017, *apud* Georgoula, 2019).

Apesar disso, deve-se atentar também ao nível de complexidade do projeto no quesito de elementos que o compõem, como uso de pré-lajes, armaduras e ligações entre perfis metálicos. Quanto maior o número de elementos, maior a dificuldade de parametrização das informações, principalmente a geométrica, uma vez que a criação de restrições, quando se trabalha com vários elementos, pode se tornar confusa, muitas vezes conduzindo a um erro na criação do elemento ou até mesmo a uma geometria errada.

De acordo com Abdallah (2017), há três tipos de ferramentas de modelagem:

a) Modeladores baseados em objetos pré-definidos: o acesso aos dados e parâmetros é muito limitado e, em geral, os projetistas possuem acesso limitado aos parâmetros de projeto. Exemplos típicos que se enquadram nessa categoria são Graphisoft Archicad, Revit e Vectorworks;

b) Modeladores semi-restritivos: maior liberdade em relação à categoria anterior, com maior possibilidade de intervir nos parâmetros de projeto. Nessa categoria, citam-se como exemplos Dassault Systèmes Catia, Autodesk 3ds Max, Autodesk AutoCAD e Autodesk Maya;

c) Modeladores livres: acesso quase total aos dados são concedidos ao projetista, com quase total liberdade de composição e parametrização, caso das ferramentas Autodesk Dynamo, David Rutten at Robert McNeel & Associates Grasshopper (plug-in do Rhinoceros 3D) e Bentley Systems GenerativeComponents.

Dessa forma, de acordo com essa classificação, o uso de *software* modeladores livres tende a conduzir a parametrizações mais ricas, uma vez que maior liberdade é dada no acesso às informações e na manipulação desses dados, além de ser ferramentas mais adequadas para modelagens de pontes. Vale notar que são citadas ferramentas de modelagem puramente geométrica, não sendo englobados modelos de análise estrutural de pontes.

O trabalho de Segura e Yepes (2016) concentra-se na otimização multiobjetivo de pontes rodoviárias de concreto protendido tipo viga caixão, visando balancear e encontrar as melhores soluções em termos de custo, emissões de CO2 e fator de segurança geral. A metodologia emprega um Algoritmo de Busca Harmônica

Multiobjetivo (HS) modificado, um algoritmo genético que utiliza estratégias de diversificação e intensificação para explorar o espaço de soluções. O funcionamento de algoritmos genéticos baseia-se no princípio da evolução biológica e busca encontrar um equilíbrio entre os objetivos. O processo envolve a criação aleatória de uma população inicial de "pais", a partir da qual uma população de "descendentes" é gerada através de seleção, cruzada (combinando os parâmetros) e mutada (variação aleatória de determinado parâmetro) (ARELLANO; TOLENTINO; GÓMEZ, 2018).

Foi desenvolvida uma ferramenta computacional para estabelecer a conexão entre o *software* de análise de elementos finitos, CSiBridge, a módulos de verificação de estados limites e ao processo de otimização no Matlab, permitindo um design automático e a busca por soluções ideais em relação à geometria, resistência do concreto, aço passivo e ativo. O programa é organizado em oito módulos: (1) atualização das variáveis de projeto conforme a estratégia do algoritmo; (2) dimensionamento; (3) geração do arquivo de entrada (*.br) para o CSiBridge; (4) execução da análise estrutural no CSiBridge; (5) extração dos resultados via API para o Matlab; (6) cômputo da resistência e verificação dos estados-limites; (7) avaliação das funções objetivo; e (8) obtenção das soluções ótimas. O ciclo se repete até o atendimento dos critérios de parada do algoritmo (SEGURA; YEPES, 2016).

Arellano, Tolentino e Gómez (2018) propuseram uma abordagem de otimização multiobjetivo para resolver o problema da estabilização do deslocamento de mastros em pontes estaiadas de múltiplos vãos sob ação de cargas móveis. A solução investigada é uma configuração não convencional de cabos que se cruzam no centro dos vãos intermediários, formando uma região que funcionam como pontos de fixação para os mastros. O objetivo central foi encontrar o comprimento ótimo de sobreposição de cabos que minimiza o custo do sistema de cabos, minimiza o deslocamento máximo da cabeça do mastro e maximiza a carga móvel na ponte. Para isso, foi empregado o Algoritmo Genético de Classificação por Não Dominância (NSGA II).

Basicamente, no NSGA II, por ser uma classificação de não dominância, uma solução domina a outra quando é melhor ou igual em todos os objetivos e estritamente melhor em pelo menos um. A solução ideal é composta pelo conjunto de pontos em que um objetivo não pode ser melhorado sem prejuízo de outro. Para implementar o

algoritmo de forma automatizada, foi desenvolvido um sistema integrado entre o *MathWorks Matlab* e o CSiBridge por meio de sua API. O estudo de caso corresponde a uma ponte de cinco vãos que totalizam 800 metros e quatro mastros. O processo de otimização segue a mesma estrutura do exposto por Segura e Yepes (2016), sendo o ciclo repetido iterativamente até o atendimento dos critérios de convergência do algoritmo genético (ARELLANO; TOLENTINO; GÓMEZ, 2018).

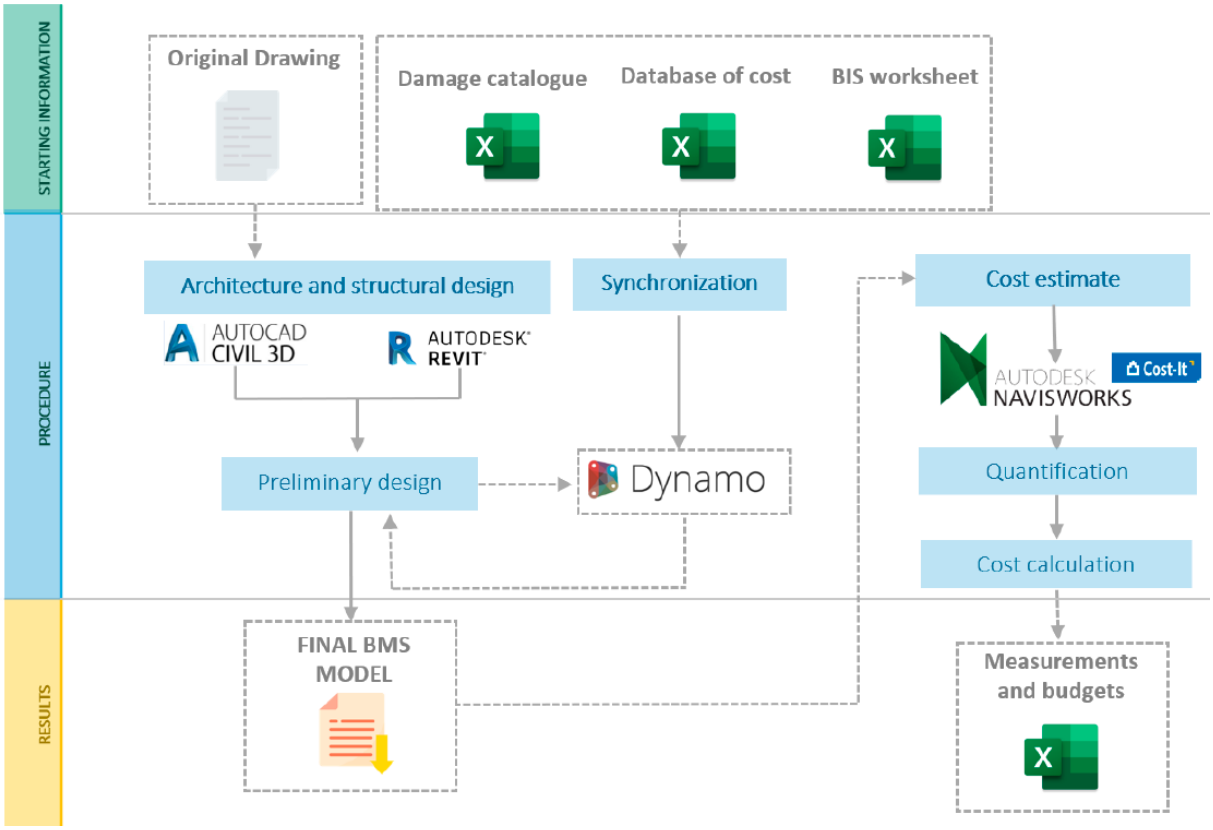
O estudo de Artar e Cabas (2023) propõe uma metodologia focada na otimização do desempenho da resiliência sísmica de pontes treliçadas de aço ao dissipar o máximo de energia sísmica de *input* através do uso de amortecedores de fricção Pall (PFDs – *Pall Friction Dampers*), que absorvem energia por deslizamento e fricção durante um terremoto. A metodologia busca determinar o valor ótimo da carga de deslizamento dos PFDs, essencial para que funcionem sem danificar os elementos estruturais. Para isso, emprega os algoritmos meta-heurísticos *Squirrel Search* (SS) e *Water Strider* (WS), que simultaneamente otimizam os perfis de aço e as cargas de deslizamento dos PFDs. O processo é realizado em várias etapas, incluindo otimização inicial da ponte, análise sob efeitos sísmicos, determinação das cargas de deslizamento ideais para máxima dissipação de energia e uma otimização final do peso total da ponte com os PFDs implementados. O estudo utiliza o SAP2000 para as análises estruturais e o MATLAB para os algoritmos, com comunicação entre eles via API.

Bazán *et al.* (2020) teve como objetivo principal revisar a literatura existente sobre o BIM com foco especial na área de infraestrutura de transporte. A metodologia da pesquisa consistiu em comparar a experiência prática do uso do BIM em projetos de mestrado em Engenharia Civil na Universidade Politécnica de Madrid com a literatura existente, utilizando-se vários *software* comerciais, como Revit, Autodesk Civil 3D, Autodesk Navisworks e Bentley OpenRoads ConceptStation, bem como ferramentas como Dynamo e SketchUp através de quatro estudos de caso.

Focando no primeiro estudo de caso apresentado, o objetivo foi demonstrar a aplicação da metodologia BIM para desenvolver um Sistema de Gerenciamento de Pontes (*Bridge Management System* - BMS) eficiente, crucial dada a vasta quantidade de pontes existentes que necessitam de manutenção. A metodologia empregou um viaduto como modelo, estabelecendo uma conexão entre as fichas de inspeção,

planilhas usadas para registrar patologias e estimar custos, e o modelo BIM. Para isso, um código de programação em Dynamo foi desenvolvido para permitir a troca bilateral de informações entre Excel e Revit, conforme demonstrado na Figura 2.2 (BAZÁN *et al.*, 2020).

Figura 2.2 – Esquema do Sistema de Gerenciamento de Pontes proposto por Bazán *et al.* (2020)



Fonte: Bazán *et al.* (2020)

Essa integração possibilitou a visualização gráfica em 3D do estado da estrutura e a estimativa dos custos de reparo com base nos danos reportados, sendo a visualização do BMS apresentada na Figura 2.3. Os resultados confirmaram que o BIM é viável para atividades de manutenção de pontes, permitindo a detecção de patologias recorrentes, a comparação de custos de manutenção e a otimização de recursos. A principal lacuna é a exigência de modelar pontes já existentes que não foram inicialmente concebidas com BIM, o que limita a aplicabilidade do processo, e a necessidade de tecnologias de Big Data para gerenciar eficientemente o grande volume de informações geradas (BAZÁN *et al.*, 2020).

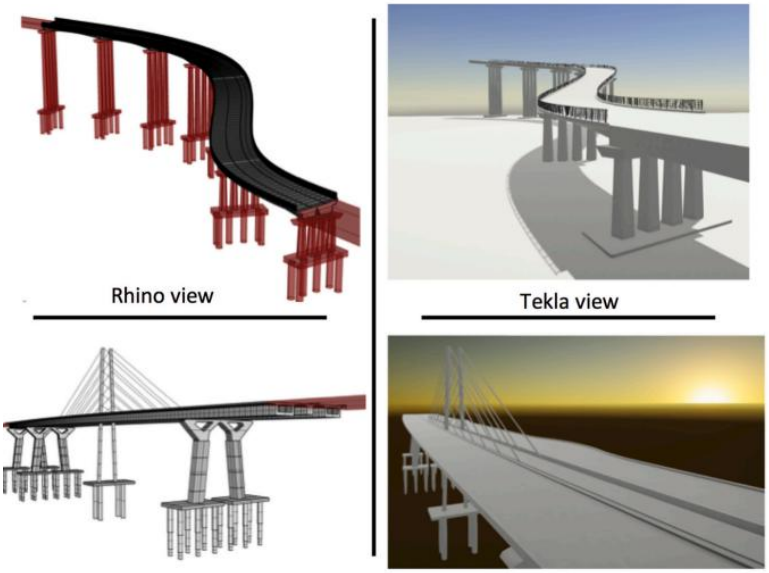
Figura 2.3 – Modelo BIM associado ao Sistema de Gerenciamento de Ponte proposto por Bazán *et al.* (2020) com os parâmetros de danos preenchidos para os encontros e laje do tabuleiro

Fonte: Bazán *et al.* (2020)

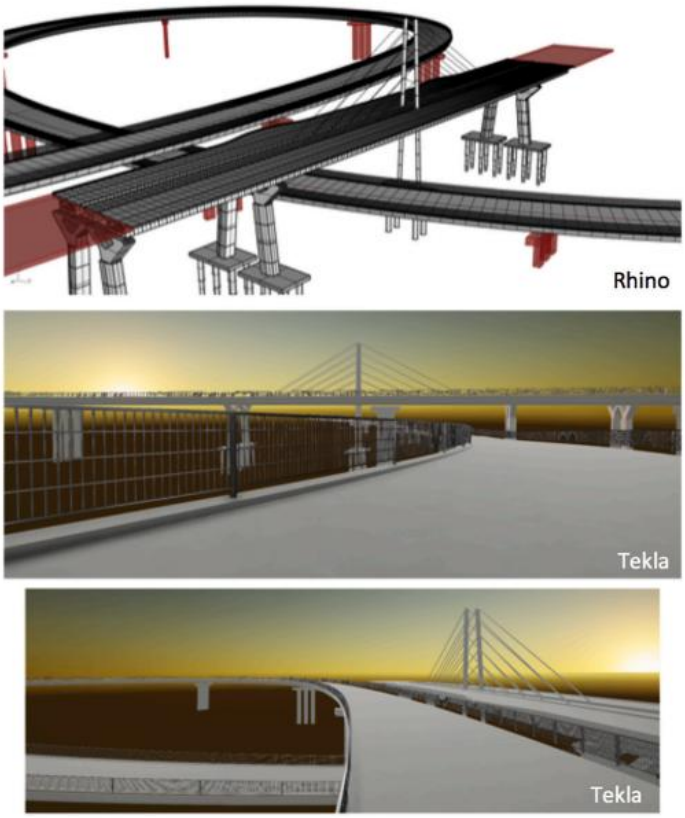
O estudo de Girardet e Boton (2021) teve como objetivo principal desenvolver um arquivo paramétrico único capaz de gerar todos os tipos de pontes, independentemente de sua geometria. A escolha das ferramentas recaiu sobre modeladores livres como o Grasshopper para manipulação de dados e programação, e o Tekla Structures para geração do modelo geométrico, dada a necessidade de lidar com geometrias não padronizadas e a flexibilidade desejada. O processo envolveu a divisão do modelo em subarquivos para diferentes partes da ponte (tabuleiro, colunas, fundações, guarda-corpos) e a criação de um sistema em cascata em que a modificação de um elemento se propaga aos demais.

O arquivo paramétrico se mostrou viável, preciso e altamente adaptável, permitindo uma redução considerável no tempo de modelagem (de semanas para 1-2 dias), mesmo para geometrias complexas. Isso facilita a criação rápida de múltiplos modelos na fase inicial do projeto, melhora a comunicação com o cliente e a integração de diversas disciplinas. No entanto, o arquivo desenvolvido não realiza análises estruturais diretamente e é limitado para níveis de detalhe muito altos. Além disso, a complexidade de muitos parâmetros e a falta de uma interface de usuário mais intuitiva foram apontadas como limitações para futuras melhorias (GIRARDET; BOTON, 2021). Na Figura 2.4 ilustram-se alguns dos modelos gerados através do algoritmo proposto.

Figura 2.4 – Visualizações no Rhino e Tekla, bem como algumas renderizações, de alguns dos modelos gerados a partir do algoritmo proposto por Girardet e Boton (2021)



(a)



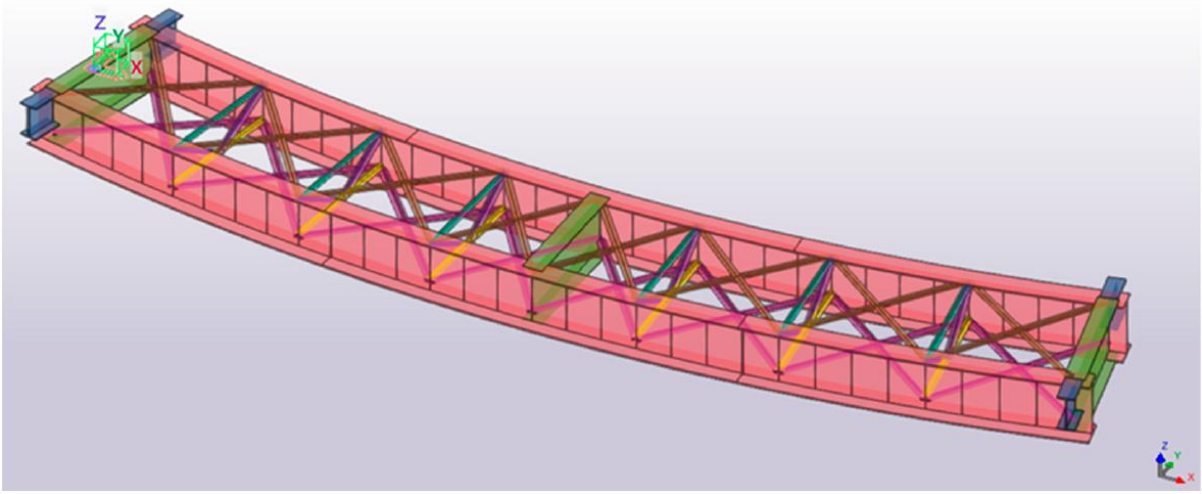
(b)

Fonte: Girardet e Boton (2021)

Boretti *et al.* (2023) visou aprimorar o processo de projeto de pontes, especificamente de pontes mistas de aço e concreto em vigas através de uma

abordagem generativa de forma a aumentar a eficiência, reduzir custos computacionais e esforços de modelagem. A primeira etapa da metodologia consiste no desenvolvimento do modelo paramétrico com foco na geometria, onde são definidas as variáveis de projeto utilizadas em operações geométricas e matemáticas para a geração das seções transversais da ponte. Na segunda etapa, é elaborado o algoritmo visual para modelagem no Tekla (Figura 2.5), incluindo a definição de perfis, materiais, posições e detalhes das ligações estruturais. A terceira etapa corresponde à criação do modelo de elementos finitos, no qual são definidos e dispostos os elementos com suas respectivas seções transversais, materiais, cargas e condições de contorno, utilizando o plug-in Karamba3D no Grasshopper. Em seguida, o modelo gerado é comparado com os resultados obtidos em um *software* secundário, midas Civil, para validação. Por fim, aplica-se um Algoritmo Genético de otimização para determinar as seções transversais das vigas mais eficientes em termos de custo (BORETTI *et al.*, 2023).

Figura 2.5 – Modelo transferido do Rhino3D para o Tekla Structures com a disposição de elementos estruturais nativos deste a partir do algoritmo proposto por Boretti *et al.* (2023)



Fonte: Boretti *et al.* (2023)

Os resultados do estudo demonstram que a metodologia proposta foi de encontro com os objetivos propostos, permitindo explorar e criar projetos de alto potencial com geometrias complexas e encontrar soluções econômicas. O fluxo de trabalho permite a geração de múltiplos *designs* de pontes mistas alterando os valores dos parâmetros, aumentando a flexibilidade do modelo. A qualidade do modelo foi testada e verificada, apresentando boa precisão para a fase de projetos conceituais.

A comunicação direta entre as ferramentas (Rhinoceros, Grasshopper, Karamba3D, Tekla) minimiza o risco de perda de dados, erros e discrepâncias. A estratégia de otimização resultou em uma redução de 3.5% na massa total da estrutura da ponte em um caso simplificado. Entre as lacunas e áreas para desenvolvimento futuro, o estudo menciona a possibilidade de inserir um conjunto maior de parâmetros para modelar mais tipos de pontes, considerando diferentes geometrias, tecnologias construtivas e materiais (BORETTI *et al.*, 2023).

Apesar dos avanços relatados, muitas das abordagens ainda são restritas a determinadas tipologias ou apresentam fluxos segmentados entre modelagem geométrica e análise estrutural. No presente trabalho, avança-se ao propor um fluxo parametrizado capaz de gerar, a partir de uma única base de dados, tanto modelos geométricos desenvolvidos em BIM quanto modelos numéricos robustos em elementos finitos, aplicáveis tanto a pontes pré-moldadas e protendidas de concreto quanto mistas de aço e concreto. Os modelos analíticos gerados permitem análises completas de super e mesoestrutura, abrangendo ainda componentes como laje de transição, parede cortina e alas dos encontros. Além disso, diferentemente dos estudos revisados, este trabalho incorpora rotinas de dimensionamento baseadas nas normas brasileiras, aspecto inédito na literatura, ampliando sua aplicabilidade direta no contexto nacional e fortalecendo sua relevância prática e acadêmica (ABNT, 1982, 2004, 2021, 2023a, 2023b, 2024).

Adicionalmente, este trabalho inova ao estabelecer um fluxo parametrizado, assegurando total consistência entre as representações analítica e geométrica, de forma a eliminar discrepâncias frequentemente observadas no processo de projetar. Alterando-se a base de dados, permite-se gerar rapidamente múltiplas variantes tipológicas com elevada rastreabilidade e padronização, criando-se possibilidades para o desenvolvimento de trabalhos de otimização estrutural através da implementação de algoritmos genéticos, bem como sistemas de manutenção e gestão através da modelagem no ambiente BIM. Tal integração, ainda incipiente na literatura, representa um avanço metodológico especialmente relevante para o setor de projetos de pontes.

Conforme visto, o desenvolvimento de projetos de pontes demanda o uso combinado de diversos *software* especializados, cada um voltado a etapas

específicas, como modelagem geométrica, análise estrutural, otimização e manutenção. Essa diversidade de ferramentas resulta em fluxos de trabalho fragmentados e complexos, exigindo integrações manuais ou personalizadas que podem comprometer a consistência dos dados. Nesse cenário, o esquema de dados IFC se apresenta como uma solução fundamental ao promover a interoperabilidade entre programas da indústria AECO, podendo permitir a troca segura e eficiente de informações ao longo de todo o ciclo de vida do projeto.

2.4 IFC: a abordagem orientada a objetos para a interoperabilidade no BIM

2.4.1 Contexto e a linguagem EXPRESS

Para reduzir o desperdício econômico e melhorar a interoperabilidade por padrão aberto entre modelos digitais na indústria AECO, foi formada a *International Alliance for Interoperability*, sendo composta por desenvolvedores, usuários e autoridades públicas (SACKS *et al.*, 2018). Essa colaboração resultou na criação do esquema de dados orientado a objetos IFC para a representação abrangente e detalhada de modelos digitais, padronizado por meio da ISO 16739 em 2013 (ISO, 2013). A organização posteriormente foi renomeada para buildingSMART, principal entidade responsável pelo desenvolvimento, manutenção e promoção do esquema de dados. O esquema de dados IFC visa, substancialmente, facilitar a troca de informações entre diversos sistemas, reduzindo retrabalhos e garantindo consistência dos dados, haja vista que a indústria da construção é caracterizada por processos muito fragmentados, com variados participantes diferentes e independentes. Com isso, é comum utilizar vários *software* para a elaboração de um projeto, muitas vezes um por disciplina (BORRMANN *et al.*, 2018).

O esquema IFC utiliza a linguagem de modelagem de dados EXPRESS, padronizada pela ISO 10303 (ISO, 1994, 2004) como parte do STEP (*Standard for the Exchange of Product Model Data*). Essa linguagem permite a criação de modelos orientados a objetos nos quais a estrutura de dados é composta por classes denominadas entidades, representando elementos de construção, relações e propriedades. O EXPRESS especifica os atributos dessas classes (como cor, tamanho e forma), seus tipos e as restrições aplicáveis. Além disso, define as relações

entre classes e as limitações numéricas pertinentes a essas interações. (BUILDINGSMART, 2025a).

Em um modelo de dados EXPRESS, uma relação entre um objeto do Tipo A e um objeto do Tipo B é estabelecida atribuindo à entidade do Tipo A um atributo do tipo da entidade B. Uma característica distintiva do padrão EXPRESS é a capacidade de definir explicitamente relações inversas, que representam a mesma associação na direção oposta sem introduzir novas informações. Além disso, o EXPRESS incorpora agregação de tipos de dados, como lista, *array* e *sets*, diretamente na linguagem, facilitando a definição de relações com grupos de objetos. Essa estrutura de tipos abstratos permite a definição de superclasses sem a necessidade de instanciá-las explicitamente. A linguagem EXPRESS permite ainda definir condições algorítmicas por meio de um bloco opcional denominado WHERE, que serve para descrever regras de consistência dos dados. O bloco WHERE contém expressões booleanas que devem ser avaliadas como verdadeiras para que a respectiva instância seja considerada válida (BORRMANN *et al.*, 2018).

O código em linguagem EXPRESS ilustrado na Figura 2.6 define a entidade IfcBeam, que representa as vigas em um modelo de informações de construção. Como subclasse de IfcBuiltElement, IfcBeam herda todos os atributos e comportamentos de suas superclasses. A entidade possui um atributo opcional chamado PredefinedType, do tipo IfcBeamTypeEnum, que é um tipo enumerado restringindo as entradas de dados a valores pré-definidos pela documentação. O bloco WHERE contém restrições para garantir a consistência dos dados. A restrição CorrectPredefinedType assegura que, se o tipo for 'USERDEFINED', o campo ObjectType (atributo herdado de IfcObject) seja preenchido. As condições para o atendimento dessa condição são:

a) Se PredefinedType não está definido (NOT EXISTS), a regra é satisfeita;

b) Se definido e não for USERDEFINED, também é válido;

c) Se for USERDEFINED, exige que ObjectType contenha uma descrição personalizada.

Figura 2.6 - Definição da entidade IfcBeam em EXPRESS conforme a documentação IFC v. 4.3.2

```
ENTITY IfcBeam
 SUBTYPE OF (IfcBuiltElement);
      PredefinedType : OPTIONAL IfcBeamTypeEnum;
 WHERE
      CorrectPredefinedType : NOT(EXISTS(PredefinedType)) OR
 (PredefinedType <> IfcBeamTypeEnum.USERDEFINED) OR
 ((PredefinedType = IfcBeamTypeEnum.USERDEFINED) AND EXISTS
(SELF\IfcObject.ObjectType));
      CorrectTypeAssigned : (SIZEOF(IsTypedBy) = 0) OR
  ('IFC4X3_DEV_9b04d16c.IFCBEAMTYPE' IN
TYPEOF(SELF\IfcObject.IsTypedBy[1].RelatingType));
END_ENTITY;
```

Fonte: Adaptado de BUILDINGSMART (2025c)

Já a outra restrição, CorrectTypeAssigned, garante que, se a viga está associada a um tipo (IsTypedBy), este tipo seja IfcBeamType. Essa condição é atendida caso:

a) Não houver tipo associado (SIZEOF(IsTypedBy) = 0);

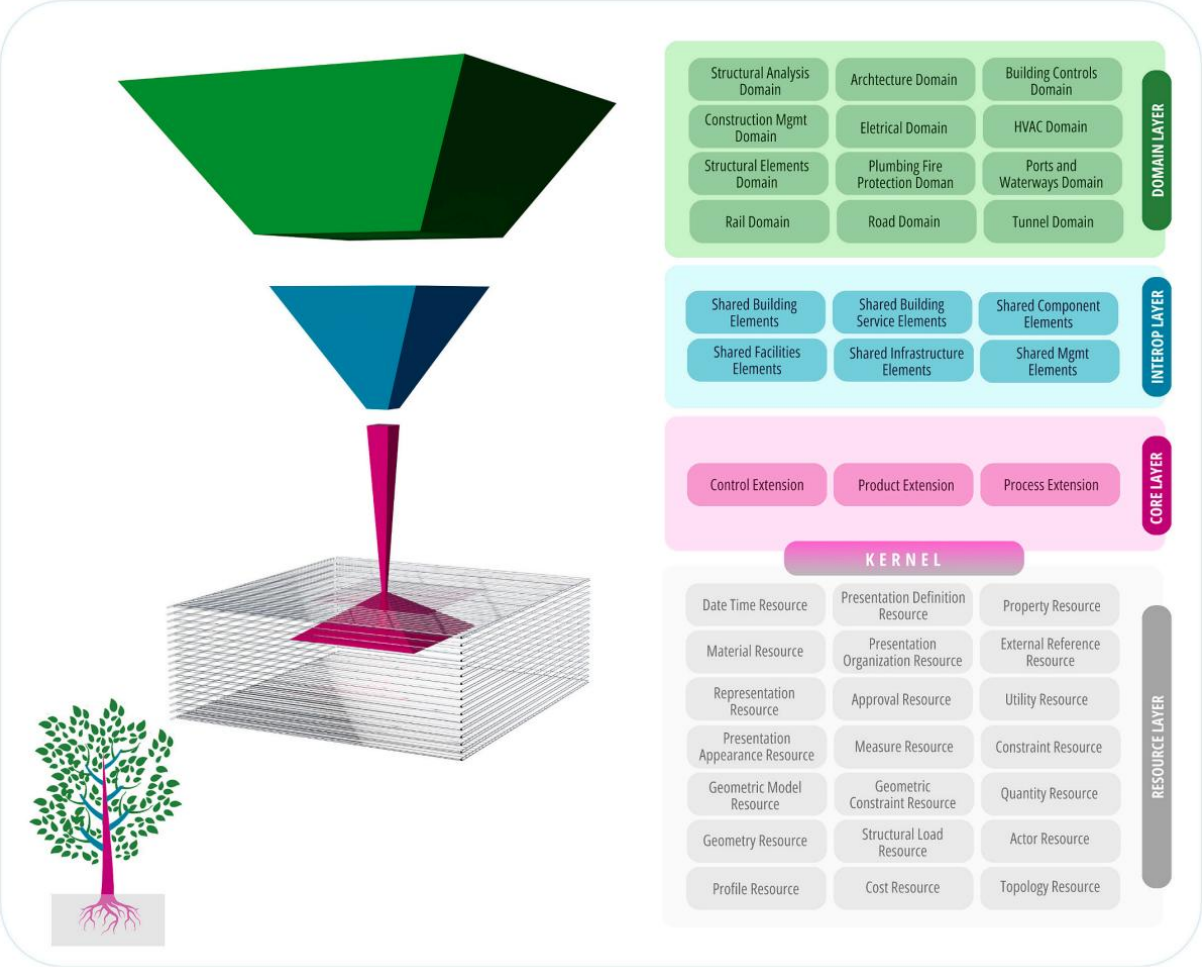
b) Se houver tipo, ele deve ser uma instância de IFCBEAMTYPE (do esquema IFC4X3_DEV).

A linguagem EXPRESS é utilizada para definir o modelo de dados, também conhecido como esquema, não sendo possível descrever instâncias concretas do modelo de dados. Para tanto, vários formatos eletrônicos suportam a codificação do IFC, sendo a mais comum e recomendável o SPF (STEP *Physical File*), que possui extensão .ifc (BUILDINGSMART, 2025b).

2.4.2 Estrutura do IFC

O esquema de dados IFC é extenso e complexo e, para melhorar sua manutenção e extensibilidade, possui uma arquitetura em camadas, conforme ilustrado na Figura 2.7. O princípio geral é de que os elementos das camadas superiores possam referenciar os elementos nas camadas inferiores, mas não vice-versa, garantindo que os elementos do núcleo permaneçam independentes. Sua arquitetura é composta por quatro camadas, respectivamente na ordem hierárquica: (BORRMANN *et al.*, 2018; SÖBKE *et al.*, 2021; BUILDINGSMART, 2025c):

Figura 2.7 – Representação das hierarquias das camadas no esquema de dados IFC



Fonte: Antunes *et al.* (2024)

a) Camada de recursos (*resource layer*), que fornece estruturas de dados básicas que podem ser utilizadas em todo o modelo de dados IFC, sendo a camada mais primitiva da arquitetura. As entidades aqui definidas não podem existir independentemente, e devem ser referenciadas por um objeto. Os principais esquemas de recurso incluem elementos geométricos, elementos para descrição de materiais e autoria e histórico de versão de objetos IFC.

b) Camada central (*core layer*), composta pelo Kernel e mais três extensões, que define classes, relacionamentos e os conceitos fundamentais que podem ser reusados e definidos de forma mais precisa nas entidades das camadas superiores. O Kernel reúne as entidades abstratas essenciais (como IfcRoot, IfcObject, IfcActor, IfcProcess, IfcProduct, IfcProject e IfcRelationship) que formam a base do modelo. Sobre essa fundação, a extensão de produto descreve os objetos físicos e espaciais da edificação (por meio de entidades

subtipo como IfcBuilding, IfcBuildingStorey, IfcSpace, entre outros) e define os relacionamentos que os interligam; a extensão de processo agrupa as entidades destinadas a descrever processos e operações, permitindo estabelecer dependências entre os elementos processuais e associá-los a recursos; e a extensão de controle define as entidades para objetos de controle, como IfcControl e IfcPerformanceHistory, possibilitando sua alocação a elementos físicos.

c) Camada de interoperabilidade (*interoperability layer*), que está localizada logo acima da camada central e funciona como um elo entre o núcleo do modelo de dados e os esquemas específicos de cada domínio. Nessa camada, são definidas entidades derivadas da camada núcleo que podem ser utilizadas em diversos esquemas de aplicação. Como exemplo, tem-se classes essenciais da construção, como IfcWall, IfcBeam e IfcWindow.

d) Camada de domínio (*domain layer*), que é composta por entidades altamente especializadas aplicáveis apenas a um determinado campo, representando os níveis mais detalhados da hierarquia de herança. As entidades dessa camada não podem ser referenciadas por outras camadas ou por esquemas de domínios distintos. Na atual versão oficial do IFC (v. 4.3.2.0), estão definidos domínios de análise estrutura, arquitetura, controle de edifícios, gerenciamento da construção, projetos elétricos, climatização, elementos estruturais, tubulação de proteção a fogo, portos e vias navegáveis, ferrovia e rodovia. O domínio de túneis ainda não foi incluído, mas está previsto para futuras atualizações.

A entidade IfcRoot é o ponto de partida e atua como a raiz da árvore de herança no esquema IFC. Todas as entidades, exceto as da camada de recursos, devem derivar direta ou indiretamente dela. Essa entidade fornece funcionalidades básicas para identificação única de um objeto por meio de um Identificador Global Único (GUID), além de armazenar informações sobre propriedade, origem e histórico de modificações (incluindo autoria e versões anteriores). Cada objeto pode receber uma identificação e uma descrição. A partir de IfcRoot, derivam diretamente três entidades principais, que compõem o próximo nível da hierarquia (BORRMANN *et al.*, 2018):

a) IfcObjectDefinition é uma entidade abstrata que agrupa objetos físicos (como elementos construtivos), objetos espaciais (como aberturas e ambientes) e elementos conceituais (como processos e custos). Além disso, abrange definições sobre os envolvidos no projeto. Seus subtipos são IfcObject (representando objetos individuais no projeto), IfcTypeObject (tipos de objetos) e IfcContext (informações gerais do projeto).

b) IfcRelationship e suas entidades subtipo representam relações entre objetos, separando a semântica das relações dos atributos dos objetos. Assim, propriedades específicas de uma relação podem ser armazenadas diretamente com o objeto relacionado.

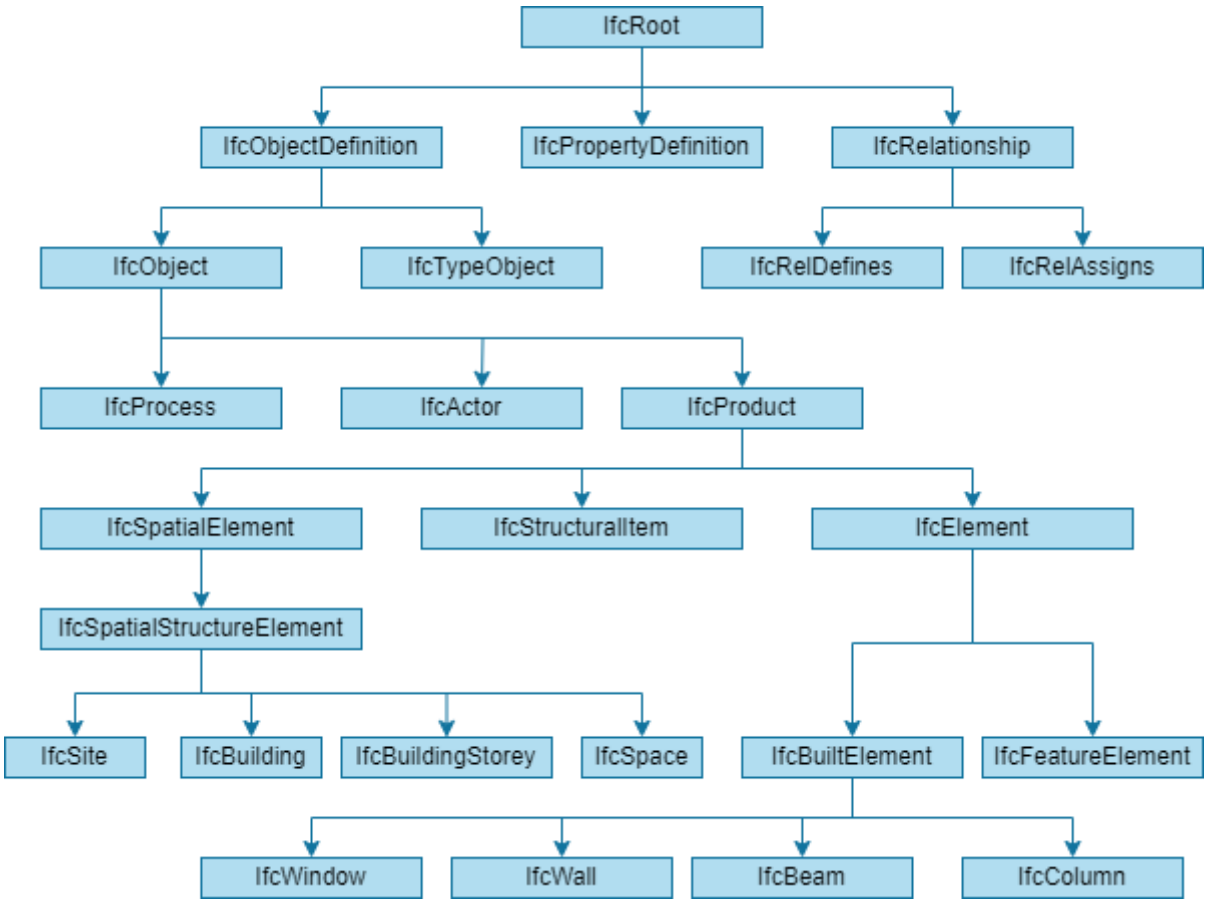
c) IfcPropertyDefinition define propriedades adicionais dos objetos que não estão previamente incluídas no modelo IFC.

As entidades mais importantes das camadas superiores, bem como exemplos de subtipos, estão ilustrados na Figura 2.8. O diagrama evidencia a complexidade da modelagem IFC, onde um objeto real, como uma parede ou pilar, precisa atravessar várias camadas de abstração até ser completamente definido. Partindo da entidade IfcRoot, que garante identificação e histórico, tem-se subtipos como IfcObjectDefinition, IfcProduct e IfcElement, refinando sua classificação até atingir instâncias concretas como IfcWall ou IfcColumn. Essa estrutura detalhada, embora extensa, é essencial para garantir organização e interoperabilidade no setor da construção civil.

Como observado, o esquema de dados IFC pode ficar sobrecarregado de informações devido à multiplicidade de atributos, propriedades e geometrias, podendo exceder o volume de dados desejado para uma troca de dados específica. Por conta disso isso, a buildingSMART introduziu os *Model View Definitions* (MVD) para mitigar a complexidade do modelo de dados, restringindo a quantidade de informações necessárias para cada troca. Os MVD também fundamentam a certificação de compatibilidade de *software* com o padrão IFC (BUILDINGSMART, 2025d).

Apesar de o IFC ser um esquema de dados aberto, há alguns desafios em sua implementação (ABBASNEJAD *et al.*, 2020; BORRMANN *et al.*, 2018):

Figura 2.8 – Exemplo de entidades importantes nas camadas superiores do IFC



Fonte: adaptado de Borrmann *et al.* (2018)

a) O esquema IFC permite múltiplas formas de modelar geometrias tridimensionais, o que exige dos fornecedores de *software* o suporte a todas essas variações e pode resultar em problemas de interoperabilidade se apenas parte delas for implementada;

b) Devido à diversidade característica do esquema de dados (alinhada à demanda da indústria civil), muitos fornecedores de *software* implementam apenas parcialmente suas funcionalidades, o que pode levar a incompatibilidades e perda de dados durante a troca de informações entre diferentes sistemas. Essas limitações comprometem a interoperabilidade e a confiabilidade do esquema IFC, afetando negativamente a confiança dos profissionais na sua adoção;

c) A evolução do IFC depende de longos processos de padronização e da dificuldade de se chegar a acordos comuns entre os envolvidos;

d) O acesso e a manipulação de dados IFC, especialmente quando armazenados no formato STEP, podem ser complexos para programadores;

e) O padrão IFC em si pode ser incompleto para abranger todas as necessidades de informação ao longo do ciclo de vida de um projeto, principalmente para obras do domínio de infraestrutura.

2.4.3 A implementação do IFC em projetos do domínio da infraestrutura.

Para expandir a aplicação do esquema IFC também às obras de infraestrutura, a buildingSMART criou, em 2010, o projeto *Infrastructure Room*, um centro de coordenação para diversos grupos internacionais envolvidos na implementação do IFC nesse setor. Desde então, vários projetos foram conduzidos com esse objetivo (COSTIN *et al.*, 2018).

Na modelagem de rodovias e pontes, o alinhamento geométrico é uma informação essencial para o correto posicionamento dos elementos. Uma das principais conquistas da iniciativa foi a criação de uma entidade capaz de armazenar e trocar dados de alinhamento ao longo de todo o ciclo de vida do projeto, desde a fase de planejamento até a manutenção através da entidade IfcAlignment. Além disso, possibilitou-se a integração das informações de alinhamento a outros aspectos do projeto, como seções transversais e geometrias 3D de elementos construtivos. Como resultado, a entidade foi inserida na versão oficial em 2015 como padrão oficial pela buildingSMART (COSTIN *et al.*, 2018).

Esse projeto serviu de base para desenvolvimentos subsequentes, como o *IFC Road*, voltado para estradas, e o *IFC Bridge*, um módulo específico para pontes. Além disso, houve iniciativas voltadas para túneis e ferrovias. No entanto, a expansão do esquema IFC para diferentes áreas da infraestrutura gerou sobreposição de esforços no desenvolvimento de soluções isoladas para pontes, estradas, túneis e outros elementos. Para resolver essa fragmentação, foi criada a arquitetura de trabalho *IFC Infra Integration Framework* (COSTIN *et al.*, 2018), que visa integrar os diferentes domínios da infraestrutura dentro do esquema IFC. Como resultado, alguns elementos desse setor já foram incorporados, ainda que de forma experimental. Na versão atual do esquema, IFC 4.3.2.0, padronizada pela ISO 16739 (ISO, 2024), foram incorporadas entidades específicas para pontes, além de outras voltadas à infraestrutura.

Apesar dos avanços, os *software* comerciais certificados pela buildingSMART adotam oficialmente apenas o IFC 4 até 2025, que não contempla as extensões para infraestrutura. Além disso, as empresas necessitam de tempo para implementar tais mudanças em seus sistemas. Paralelamente, a integração total do IFC para infraestrutura enfrenta desafios significativos, já que o esquema não foi originalmente concebido para esse fim, evidenciando a falta de interoperabilidade e a necessidade de um formato de intercâmbio neutro padrão. Dessa forma, o IFC permanece como um potencial padrão de informação para a indústria, sendo recomendada a sua utilização em conjunto com outros padrões para aprimorar a interoperabilidade (COSTIN *et al.* 2018, BUILDINGSMART, 2025e).

2.5 Programação orientada a objetos

Na indústria AECO, a modelagem paramétrica orientada a objetos destaca-se como a base tecnológica para o desenvolvimento de modelos digitais em *software* especializados. Neles, os desenvolvedores disponibilizam elementos padronizados com atributos geométricos e não geométricos. Com isso, é mantida a coerência e a precisão dos projetos, já que qualquer ajuste em um parâmetro é automaticamente atualizado em todas as dependências do modelo. Essa metodologia, herdada dos fundamentos da POO, é amplamente empregada no desenvolvimento de *software* em geral (SACKS *et al.*, 2018).

A POO é um dos paradigmas de programação que estrutura todo um sistema em unidades mais gerenciáveis, denominadas objetos, e suas relações. Isso permitiu superar algumas das limitações das linguagens de alto nível nas décadas de 1980 e 1990, que não eram adequadas para o desenvolvimento e a manutenção de programas complexos. Nessa abordagem, os objetos armazenam estado (dados) e comportamento (operações ou métodos). Cada objeto é uma instância de uma classe, que funciona como um molde definindo os atributos e métodos comuns a todos os objetos instanciados. Dessa forma, mesmo que duas instâncias da mesma classe possam manter estados distintos, elas compartilham a mesma estrutura de funcionamento (BUDD, 2001). Desde então, a POO é amplamente utilizada em engenharia de *software* e inteligência artificial, tendo como conceitos centrais a abstração, encapsulamento, herança e polimorfismo (ALSOLAI; ROPER, 2019; BOOCH, 1986; POKKUNURI, 1989; YILMAZ; KARAOGLAN YILMAZ, 2023).

De acordo com Canning *et al.* (1989), cada método do objeto funciona como uma implementação concreta de operações abstratas. Isso é feito realizando o intercâmbio de mensagens (ou requisições) com apenas os argumentos fornecidos pelo usuário para o objeto, que realiza as operações sem fornecer como o método é implementado. Este, por sua vez, pode implementar uma operação manipulando tanto os dados de um objeto, atribuindo novos valores a eles ou requisitando informações, quanto mandando mensagens para o próprio objeto em uma estrutura recursiva. Nesse contexto, interfaces em POO definem um conjunto de métodos que uma classe deve implementar, sem fornecer a implementação desses métodos. Assim, a abstração simplifica e oculta detalhes desnecessários, sendo fundamental na POO. Isso permite que os programadores criem sistemas complexos, além de facilitar a representação de dados e comportamentos de forma que sejam mais fáceis de entender e manipular.

As interfaces podem ser públicas, o que significa que elas são acessíveis a partir de qualquer parte do *software*. Isso é comum em API's (*Application Programming Interfaces*), em que as interfaces públicas permitem que desenvolvedores externos interajam com bibliotecas de *software* sem precisar conhecer os detalhes internos de implementação, garantindo as funcionalidades pré-existentes em novos projetos. A documentação das API's frequentemente inclui diretrizes sobre como usar essas interfaces, ajudando os desenvolvedores a entenderem as restrições e os padrões de uso (MONPERRUS *et al.*, 2011).

A analogia entre uma API e o papel de um garçom em um restaurante ilustra bem sua função como intermediária entre um cliente (usuário) e um sistema (serviço, aplicação ou *hardware*). Assim como o garçom recebe o pedido, o repassa à cozinha e retorna com o prato pronto, a API recebe uma requisição, a direciona ao sistema responsável pelo processamento e retorna a resposta correspondente. O usuário não precisa conhecer a lógica interna do sistema ou serviço com o qual está interagindo; basta utilizar a interface exposta pela API, que encapsula a complexidade do funcionamento interno. Isso permite a abstração dos processos e facilita a integração entre sistemas distintos (CHOPRA *et al.*, 2020). Nesse cenário, os desenvolvedores podem fornecer os recursos oferecidos (cardápio) na forma de documentação para desenvolvedores externos, ou seja, nem sempre quem requisitou o "prato" é o usuário final.

Um *framework* é um conjunto classes, bibliotecas e ferramentas que fornecem a base para o desenvolvimento de aplicações de *software*, fornecendo um *template* no qual os desenvolvedores podem customizar para suas aplicações específicas. Sua estrutura é formada por classes que cooperam entre si e conjuntamente são utilizadas para uma solução reutilizável para um problema. Vários são os *frameworks* disponíveis atualmente, sendo específicos para diferentes linguagens de programação, plataformas e finalidades (BUDD, 2001). Alguns exemplos estão listados a seguir:

a) Desenvolvimento *web*: Django e Ruby on Rails;

b) Aprendizado de máquina: TensorFlow e PyTorch;

c) Desenvolvimento de jogos: Unity e Unreal Engine;

d) Aplicações em Java: Spring e Hibernate;

e) Criação e execução de aplicativos para Windows: .NET *Framework*.

A herança é o mecanismo pelo qual uma classe (subclasse) pode adquirir, de forma automática, os atributos e métodos de outra classe (superclasse), permitindo a especialização e o reuso de código sem redundância. O polimorfismo permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum. Isso é alcançado através dos mecanismos de herança e interfaces, possibilitando que métodos sejam redefinidos em subclasses, facilitando a manutenção do *software* (BUDD, 2001; CARDELLI; WEGNER, 1985). Isso é extremamente vantajoso em linguagens de programação não tipadas, permitindo certa flexibilidade sem comprometer a funcionalidade do sistema como um todo.

Encapsulamento é o princípio que envolve a organização de dados (atributos) e métodos (operações ou comportamentos) dentro de um objeto, estabelecendo interfaces delimitadas. Com isso, ocultam-se os detalhes internos de implementação, permitindo que as interações ocorram exclusivamente por meio de operações externas definidas para os objetos, ou seja, seus métodos. Esse isolamento favorece a abstração de dados, a modularização e o reuso de *software*, minimizando o impacto de mudanças internas sobre o restante do sistema. A introdução da herança na POO pode comprometer os benefícios do encapsulamento, já que subclasses podem

acessar e modificar dados protegidos de suas superclasses, podendo levar a dificuldades na manutenção do código (SNYDER, 1986).

Para evitar relacionamentos complexos devidos à herança em uma hierarquia de classes, a composição surge para permitir que objetos sejam construídos a partir de outros objetos, ocasionando numa redução na interdependência entre classes ou módulos, também chamada de acoplamento. Ao reutilizar comportamentos por meio da composição, é possível modificar ou substituir partes específicas do sistema sem afetar sua estrutura global (ARISHOLM; BRIAND; FØYEN, 2004).

O código apresentado na Figura 2.9, escrito em Python, visa exemplificar as características principais da POO. Nele, é utilizada a biblioteca “abc”, padrão do Python e que permite definir classes abstratas através da herança da classe ABC e do decorador @abstractmethod.

Figura 2.9 - Exemplo de código em linguagem orientada a objetos

```
from abc import ABC, abstractmethod
class ElementoEstrutural(ABC):
    def __init__(self, nome):
        self.nome = nome
    @abstractmethod
    def calcular_tensao(self, carga):
        pass
    def info(self):
        return f"Elemento: {self.nome}"
class Viga(ElementoEstrutural):
    def __init__(self, largura, altura):
        super().__init__("Viga")
        self.largura = largura
        self.altura = altura
    def calcular_tensao(self, carga):
        area = self.largura * self.altura
        return carga / area
class Pilar(ElementoEstrutural):
    def __init__(self, diametro):
        super().__init__("Pilar")
        self.diametro = diametro
    def calcular_tensao(self, carga):
        area = 3.14 * (self.diametro / 2) ** 2
        return carga / area
```

A classe “ElementoEstrutural” atua como uma classe base abstrata e serve como um molde ou interface para as subclasses. Ela define um construtor (“init”) que inicializa o atributo nome, o qual é comum a todos os elementos estruturais. Essa classe declara também o método abstrato “calcular_tensao”, sem fornecer uma implementação, funcionando como um *checklist* que obriga todas as subclasses a implementarem esse método. Além disso, possui o método concreto info, que retorna uma informação descritiva do objeto e que pode ser utilizado pelas subclasses sem necessidade de reimplementação.

As classes Viga e Pilar herdam de “ElementoEstrutural” e demonstram como a herança permite a reutilização de código. Em seus construtores, elas utilizam a função “super()” para chamar o construtor da classe base, garantindo que o atributo nome seja inicializado corretamente com valores fixos, como "Viga" ou "Pilar". Cada subclasse define atributos específicos (por exemplo, largura e altura para a Viga, e diâmetro para o Pilar) e implementa o método “calcular_tensao” conforme a fórmula adequada à sua geometria, realizando o cálculo da tensão dividindo a carga pela área da seção transversal do elemento.

O conceito de polimorfismo é demonstrado quando objetos de tipos diferentes (por exemplo, uma Viga e um Pilar) podem ser tratados de forma uniforme. Apesar de cada subclasse ter sua implementação específica do método “calcular_tensao”, é possível chamar esse método em qualquer objeto do tipo “ElementoEstrutural” sem precisar conhecer a sua implementação interna. Isso permite que, em um laço de repetição (*loop)*, todos os elementos sejam processados da mesma maneira, promovendo flexibilidade e simplicidade no uso do código.

O encapsulamento também é evidenciado, já que cada classe gerencia seus próprios atributos e lógica interna, expondo apenas os métodos necessários para a interação externa, como “info” e “calcular_tensao”. Essa abordagem protege os dados e promove uma separação clara entre a interface pública e a implementação interna dos objetos, facilitando a manutenção e a evolução do código. Um exemplo de uso dessas classes é dado na Figura 2.10, bem como as saídas.

Dessa forma, ao aplicar conceitos como abstração, encapsulamento, herança e polimorfismo na construção de sistemas complexos, a POO promoveu a modularidade, a reutilização de código e a escalabilidade, facilitando tanto a

manutenção quanto o aprimoramento dos processos de automação. Dessa forma, seus princípios não só impulsionaram a evolução da engenharia de *software*, mas também estabeleceram uma base tecnológica capaz de transformar programas e aplicações em ferramentas dinâmicas, inteligentes e integradas, atendendo de forma eficaz às exigências de um mercado cada vez mais inovador e competitivo.

Na indústria AECO, a incorporação dos fundamentos da POO revolucionou o desenvolvimento de modelos digitais. Essa abordagem foi crucial para o avanço das metodologias modernas na engenharia, servindo de base para o desenvolvimento do BIM e o desenvolvimento do esquema de dados IFC, fundamentais para a transformação digital na construção civil.

Figura 2.10 - Exemplo de aplicação do código

```
        viga = Viga(0.3, 0.5) # Viga com largura 0.3 x 0.5 m
        pilar = Pilar(0.4) # Pilar com diâmetro 0.4 m
        carga = 5000 # Exemplo de carga em Newtons
        print(viga.info(), "Tensão:", f'{viga.calcular_tensao(carga):.2f} kN/m²') # Saída:
Elemento: Viga Tensão: 33333.33 kN/m²
        print(pilar.info(), "Tensão:",f'{pilar.calcular_tensao(carga):.2f} kN/m²') # Saída:
Elemento: Pilar Tensão: 39808.92 kN/m²
```

3. MATERIAIS E MÉTODOS

A presente pesquisa foi desenvolvida com foco na automatização de processos de modelagem paramétrica, integrando o uso de linguagens de programação com *software* de análise estrutural e modelagem em ambientes BIM. O método foi estruturado considerando a demanda de elaboração de um catálogo digital de pontes para o DNIT (BRASIL, 2020b), sendo o Revit e o CSiBridge ferramentas já utilizadas dentro do grupo de pesquisa.

Conforme fluxograma apresentado na Figura 3.1, o processo metodológico envolveu quatro etapas principais:

Etapa 1: Desenvolvimento de códigos em Python para a criação de modelos no CSiBridge;

Etapa 2: Extração de resultados da análise para alimentar as planilhas de dimensionamento de longarinas e lajes criadas pelo grupo de pesquisa;

Etapa 3: Criação de código utilizando o Dynamo para a modelagem paramétrica no Revit;

Etapa 4: Validação dos códigos a partir dos esforços obtidos para os modelos gerados no CSiBridge e análise da geometria das pontes geradas pelo Dynamo no Revit.

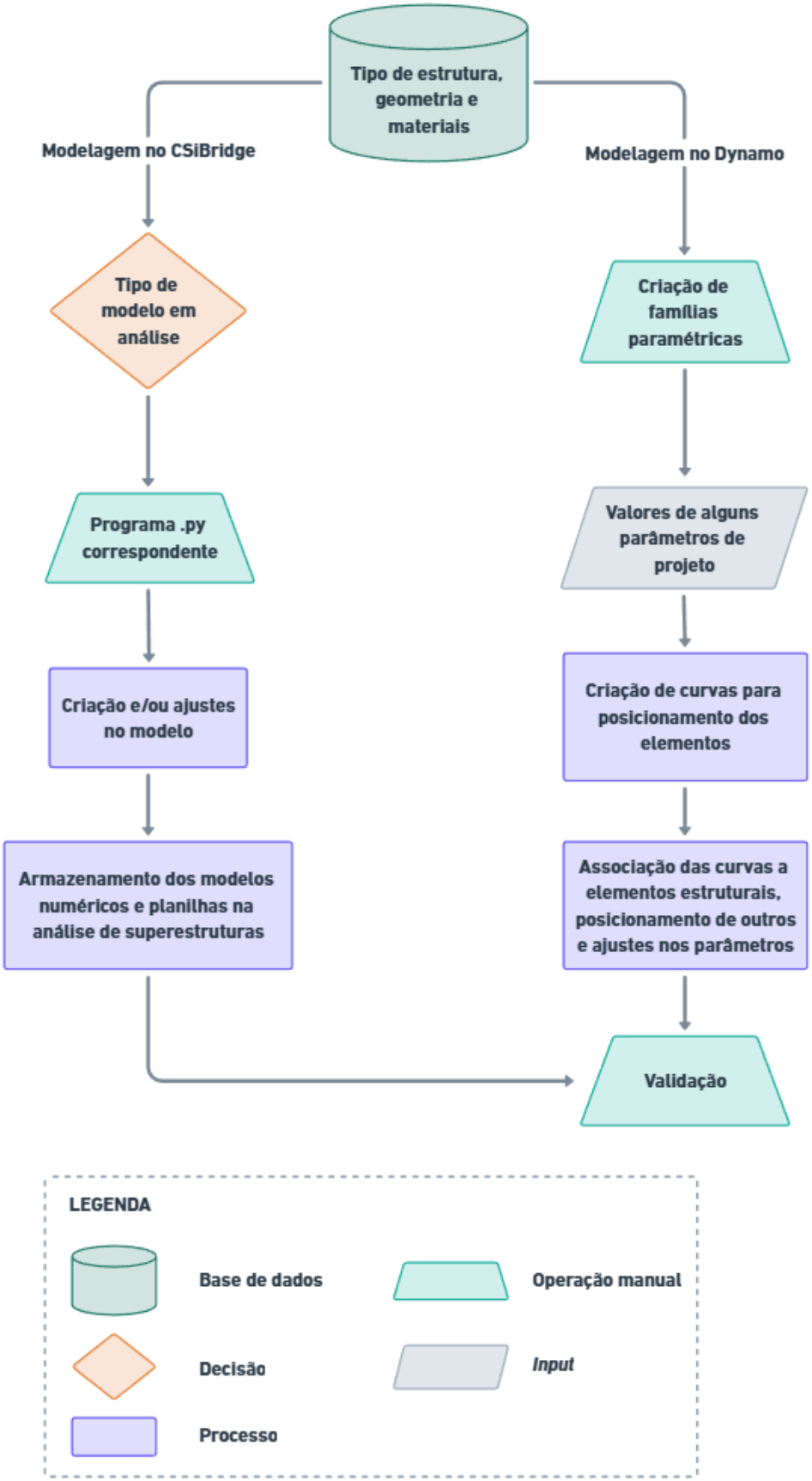
Este estudo utilizou o CSiBridge v. 25.3 por conta da possibilidade da comunicação API com o programa utilizando o Python. O *software* BIM para a modelagem paramétrica escolhido foi o Revit 2025, sendo o Dynamo o responsável pela automatização da criação dos modelos. As ferramentas de programação utilizadas são descritas a seguir:

a) Python (versão 3.12): foi escolhida como a linguagem principal devido à sua versatilidade, simplicidade sintática, ampla compatibilidade com bibliotecas e por permitir a conexão com o CSiBridge;

b) Bibliotecas Python:

- *os* e *sys*: manipulação do sistema operacional e integração de diretórios, bem como módulos necessários para o funcionamento da API;

Figura 3.1 – Fluxograma da metodologia proposta para os processos na automatização de modelagem no CSiBridge e Dynamo



- *pythonnet*: permite a comunicação direta com a API do CSiBridge, com acesso às funcionalidades da plataforma através de métodos e objetos baseados na tecnologia .NET;
- *pandas*: organização e manipulação de dados estruturados em tabelas, facilitando o processo de validação e transferência de informações;
- *numpy*: cálculos matemáticos e manipulação de *arrays*;
- *xlwings*: necessário para comunicação com o *Microsoft Excel 365*, permitindo a troca de informações bilateralmente.

c) Git: utilizado para controle de versão e gerenciamento dos códigos desenvolvidos, garantindo rastreabilidade e consistência ao longo dos projetos desenvolvidos na linguagem Python;

d) Ambiente de desenvolvimento: foi utilizado o *Visual Studio Code* (versão 1.96.4) por possuir extensões que facilitam a programação, a depuração do código e possuir integração com a plataforma Git;

e) Dynamo (versão 2.17): ambiente de programação visual que permite a manipulação de objetos e possui integração com o Revit;

f) *Scripts* em Python dentro do ambiente do Dynamo para manipulação de objetos;

g) *Data*|*Shapes*: biblioteca customizada no Dynamo para requisição de dados.

O uso do Python, Revit, Dynamo e CSiBridge são justificados também por conta de o pesquisador já ter tido contato com essas ferramentas. Além disso, houve disponibilidade de licenças acadêmicas do Revit e CSiBridge no Laboratório de BIM da Universidade Federal de Viçosa.

A comunicação com o CSiBridge foi viabilizada por meio do acesso à biblioteca de vínculo dinâmico (DLL – *Dynamic-Link Library*) fornecida na instalação do programa CSiBridge, que disponibiliza os recursos da API a partir de uma estrutura baseada na plataforma .NET. Embora essas funcionalidades tenham sido originalmente concebidas para serem utilizadas em ambientes .NET (como C# ou VB.NET), a integração foi realizada em Python por meio da biblioteca *pythonnet*, que

permite o carregamento e a importação dessas bibliotecas, tornando possível o acesso às classes e métodos da API a partir de códigos desenvolvidos.

Durante o processo de desenvolvimento, foram consultadas extensivamente as documentações oficiais relativas à linguagem Python e à conexão API com o CSiBridge e SAP2000 (também desenvolvido pela *Computers and Structures Inc.*), localizadas no próprio diretório de instalação do programa. Com isso, visou-se compreender requisitos específicos da integração via API, a estrutura das ferramentas e os métodos disponíveis para manipulação de objetos. Cabe destacar que o CSiBridge originalmente constituía um módulo especializado do SAP2000, voltado à modelagem e análise de pontes, sendo posteriormente desmembrado e consolidado como um *software* independente. Apesar dessa separação, a integração plena com o CSiBridge é realizada com o acesso aos recursos da API através de dois objetos distintos.

O objeto *SapModel*, herdado da estrutura do SAP2000, é responsável pelas definições gerais de análise estrutural, como a criação de casos e combinações de carregamento, aplicação de cargas, definição de materiais e seções transversais. Já o objeto *BridgeModeler* é específico de métodos aplicáveis a pontes criadas pelo modelador paramétrico, englobando a definição da seção do tabuleiro, *layout* da pista (também chamado de *layout line*), faixa de tráfego do veículo (*lane*), aparelhos de apoios, atribuições de transversinas, dentre outros.

Destaca-se que o processo de modelagem adotado neste trabalho utilizou o modelador paramétrico de pontes do CSiBridge, em vez da criação manual de cada componente. Com isso, evitou-se a necessidade de definir individualmente elementos de cascas (*shells*) e barras (*frames*), definição de nós que compartilham deslocamentos, bem como a inserção manual de aparelhos de apoio, travessas e pilares. Além disso, optou-se por aplicar os carregamentos diretamente no objeto de ponte reconhecido pelo programa, em vez de distribuí-los manualmente em cada elemento estrutural.

Para funcionalidades complementares, como organização de dados e automatização de planilhas eletrônicas, foram também utilizadas as documentações das bibliotecas *pandas*, *numpy* e *xlwings*. Durante o desenvolvimento de grafos (códigos elaborados em linguagem de programação visual) no Dynamo, foi também

consultado o *Dynamo Primer*, recurso oficial de aprendizado mantido pela comunidade Dynamo que oferece fluxos de trabalho recomendados.

Como ponto de partida para a implementação da comunicação API com o CSiBridge, foi adotado um código-base disponibilizado pela sua própria documentação, que define a estrutura mínima necessária para acoplar ao programa em execução, instanciar os objetos primitivos da API (*SapModel* e *BridgeModeler*) e estabelecer a comunicação entre o programa em Python e o modelo estrutural. A partir dessa base, foram desenvolvidas extensões e customizações para atender às demandas específicas do fluxo de trabalho desenvolvido (Figura 3.1).

O Dynamo é uma ferramenta disponibilizada tanto como *add-in* nativo no Revit quanto como aplicação independente no *Dynamo Sandbox*. Quando executado dentro do Revit, o Dynamo é carregado na aba *add-ins* e inicia já acoplado à instância do Revit, garantindo que as operações sejam aplicadas diretamente ao documento ativo, sem necessidade de comunicação entre processos distintos.

Internamente, cada nó específico do Revit oferecido na interface gráfica do Dynamo corresponde a uma chamada a métodos encapsulados em código .NET. O usuário vê apenas elementos de alto nível, como "*Wall.ByCurveAndHeight*"; porém, o Dynamo realiza operações externas, abstratas ao usuário, para inserir o componente no modelo. Nem todos os métodos e classes da API estão disponíveis como nós visuais no Dynamo, já que a biblioteca nativa do programa contempla apenas as operações mais comuns. Por exemplo, não é possível utilizar planos de referência para a modelagem de vigas, o que limita o processo à criação de pontes retas. Além disso, outras funcionalidades, como o impedimento da união de elementos de barras com faces que se encontram, tiveram de ser realizadas com *scripts* em Python para serem executados conjuntamente com o grafo desenvolvido no Dynamo.

No presente trabalho, adotou-se a metodologia de vinculação de elementos estruturais a curvas de referência (como elementos de barras) e o posicionamento de instâncias de famílias no modelo, utilizando o Dynamo como ferramenta de modelagem no Revit a partir dos dados paramétricos. Foram utilizadas famílias previamente parametrizadas pelo pesquisador, bem como algumas nativas, com argumentos de geometria e material. Com as duas formas de modelagem empregadas, possibilitou-se a inserção automatizada de componentes como

defensas, encontros e aparelhos de apoio com base em pontos de referência. Essa abordagem explorou a capacidade do Dynamo de processar os parâmetros fornecidos, realizar os cálculos geométricos necessários e, por meio de nós específicos, atribuir as instâncias correspondentes no Revit.

3.1 Definições dos parâmetros de entrada

As seções transversais dos elementos estruturais, bem como sua disposição e quantidade, foram definidas inicialmente com base nas recomendações de otimização propostas por Coelho (2023). Estudos adicionais foram conduzidos a partir dos resultados gerados pelos modelos de cálculo, garantindo ajustes finais. A seleção do tipo de estrutura baseou-se nos requisitos do DNIT e do Manual 698 para a elaboração de um catálogo digital de pontes, que é uma das metas do TED nº 703/2020 (BRASIL, 2020b; DNIT, 1996). Esse catálogo abrange as seguintes características:

- Dois tipos de superestruturas: seção com longarinas pré-moldadas e protendidas e seção mista composta por longarinas metálicas e *stud bolts*. Em ambos os tipos, serão utilizadas pré-lajes e capa de concreto armado;
- Comprimento dos vãos: 20, 25, 30 e 35 m;
- Dimensões do tabuleiro: para cada vão, são projetados tabuleiros que atendem às classes de rodovia I-A (pista dupla) e I-B (pista simples) para o relevo ondulado/montanhoso, sendo avaliada ainda a largura referente ao passeio ou não, conforme indicado na Tabela 3.1 e Tabela 3.2.

Tabela 3.1 - Largura dos tabuleiros das pontes de rodovias da classe I-A com e sem passeio para relevo ondulado/montanhoso

Classe I-A							
Descrição	Guarda Corpo	Passeio	Defensa	Acost. Int.	Acost. Ext.	Faixa	Largura total
Com passeio	0,15	1,50	0,40	1,00	3,00	3,60	13,65
Sem passeio	-	-	0,40	1,00	3,00	3,60	12,00

*Unidades em metro

Tabela 3.2 - Largura dos tabuleiros das pontes de rodovias da classe I-B com e sem passeio para relevo ondulado/montanhoso

Classe I-B						
Descrição	Guarda Corpo	Passeio	Defensa	Acost.	Faixa	Largura total
Com passeio	0,15	1,50	0,40	2,50	3,60	16,30
Sem passeio	-	-	0,40	2,50	3,60	13,00

*Unidades em metro

Dessa forma, são 16 tipologias por tipo de superestrutura, totalizando 32 seções transversais distintas para estudo. As tipologias das pontes foram codificadas internamente segundo uma convenção alfanumérica padronizada, representada por WWW-X-YY-Z. Cada elemento do código indica uma característica específica da ponte, conforme descrito a seguir:

- WWW: tipo de superestrutura, podendo ser PPC para pontes com longarinas pré-moldadas e protendidas de concreto ou MAC para pontes com seção mistas de aço e concreto;
- X: classe de rodovia considerada, podendo ser A para I-A e B para I-B;
- YY: vão da ponte em metros;
- Z: presença de passeio (P) ou ausência de passeio (S).

Apesar das diferenças entre modelos com passeio ou não na largura, essa foi desprezada para o lançamento da carga móvel no modelo numérico, de forma a evitar reforços na estrutura original na eventualidade de um alargamento da ponte. Essa região também não foi contemplada nos modelos elaborados no ambiente do Revit pelo Dynamo.

Foram indicados por meio da Figura 3.2 e Figura 3.3 as principais dimensões da via, bem como os parâmetros dos elementos. A compilação desses parâmetros geométricos referentes a cada tipologia de longarinas pré-moldadas e protendidas se encontra na Tabela 3.3, enquanto as informações adotadas para as pontes mistas de aço e concreto foram descritas por meio da Tabela 3.4. Essas planilhas formam a base de dados principal para a alimentação dos dados tanto nos modelos numéricos elaborados CSiBridge quanto naquelas gerados no ambiente do Revit.

Ainda sobre os parâmetros considerados para as pontes, tem-se que:

- A espessura total da laje do tabuleiro (*h_laje*) foi adotada como 28 cm para todos os casos;

- A espessura das pré-lajes foi adotada como 10 cm (*hPL*). Sua largura de dimensionamento (*bwPL*) é de 100 cm. Esses valores foram utilizados apenas para a alimentação das planilhas de dimensionamento no fluxo com os códigos em Python;

- A quantidade de transversinas (*Ndiaf*) e longarinas (*Nlong*) foram determinadas com base nos estudos de otimização elaborados por Coelho (2023), projetos anteriores e discussões dentro do grupo de pesquisa;

- O balanço dos pilares das travessas extremas (*Distextp*) e o das travessas intermediárias (*Distintp*) foi definido de acordo com estudos prévios para análise do comportamento da travessa. A quantidade de pilares para as travessas intermediárias (*Nintp*) e extremas (*Nextp)* foram definidas de forma análoga. O diâmetro dos pilares (*Dp*) foi convencionado em 100 cm. A altura dos pilares (*Hintp* e *Hextp*) foi considerada como 4, 6 ou 8 metros no modelo numérico, a depender do tipo de análise no CSiBridge. Essa abordagem também é válida no Dynamo.

- O comprimento da laje de transição (*CompLT*) foi definido como o mínimo estipulado pelo anexo K da NBR 7187 (ABNT, 2021). A sua espessura (*EspessuraLT*) foi convencionada em 28 cm e sua largura (*LarguraLT*) tomada igual à do tabuleiro.

- O comprimento das travessas extremas (*Cext*) foi tomado como igual à largura da ponte, enquanto o das travessas internas (*Cint*), caso haja algum apoio intermediário na análise, foi tomado com valores reduzidos em relação à largura do tabuleiro.

- As dimensões introduzidas no perfil metálico são válidas para ambos os perfis das longarinas, bem como para as transversinas. A diferenciação é feita na planilha e no código.

Figura 3.2 – Parâmetros geométricos e respectivos nomes considerados das variáveis para a seção transversal dos tabuleiros das pontes

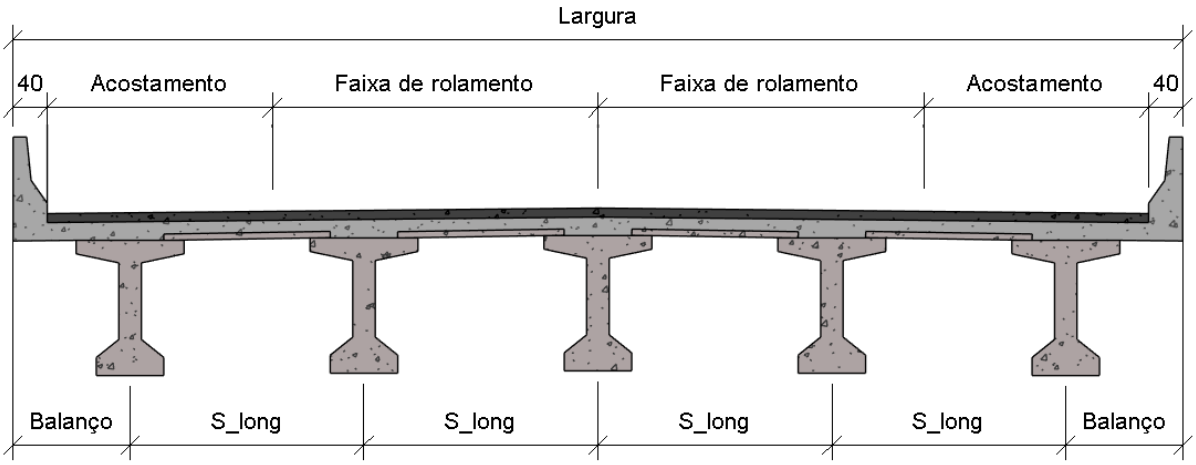
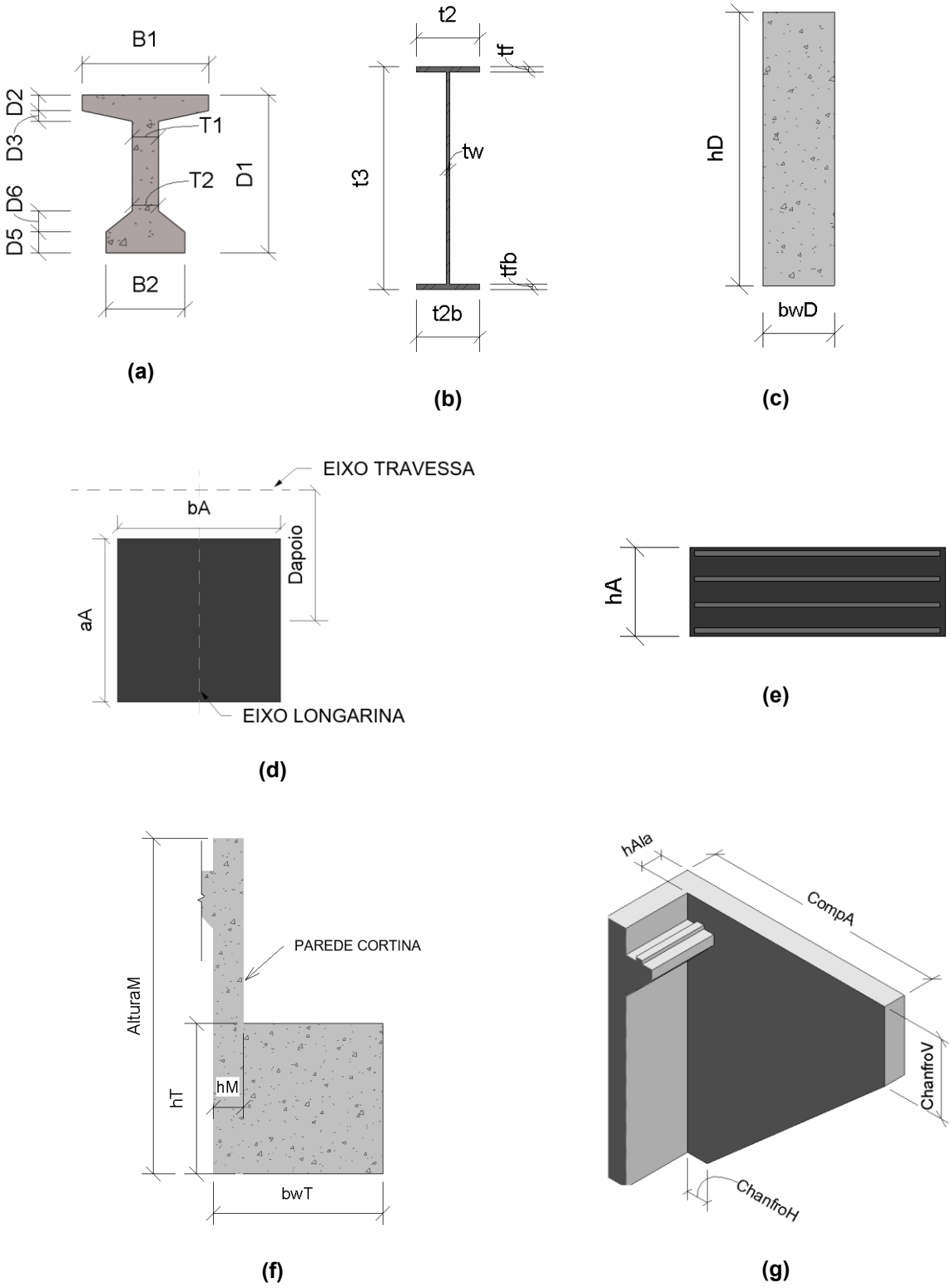


Figura 3.3 – Parâmetros geométricos e respectivos nomes considerados das variáveis para os elementos estruturais que compõem as pontes



Legenda: Dimensões (a) da longarina pré-moldada, (b) perfis metálicos e (c) vigas retangulares de concreto, como transversinas e travessas. Dimensões do aparelho de apoio (d) em planta e (e) elevação. Dimensões das (f) travessas e parede cortina e (g) das alas dos encontros.

Tabela 3.3 - Parâmetros geométricos de cada tipologia das pontes pré-moldadas de concreto (PPC) a serem lidos para a modelagem no CSiBridge e Dynamo

	Tipologia	PPC-A-20-P	PPC-A-25-P	PPC-A-30-P	PPC-A-35-P	PPC-A-20-S	PPC-A-25-S	PPC-A-30-S	PPC-A-35-S	PPC-B-20-P	PPC-B-25-P	PPC-B-30-P	PPC-B-35-P	PPC-B-20-S	PPC-B-25-S	PPC-B-30-S	PPC-B-35-S
SEÇÃO	Vão	2000	2500	3000	3500	2000	2500	3000	3500	2000	2500	3000	3500	2000	2500	3000	3500
	Largura	1365	1365	1365	1365	1200	1200	1200	1200	1630	1630	1630	1630	1300	1300	1300	1300
	Nlong	5	5	5	5	4	4	4	4	5	5	5	5	4	4	4	4
	Balanço	102,5	102,5	102,5	102,5	105	105	105	105	115	115	115	115	110	110	110	110
	S_long	290	290	290	290	330	330	330	330	350	350	350	350	360	360	360	360
LAJE	h_laje	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
LONGARINA	B1	70	80	90	100	70	80	90	100	70	80	90	100	70	80	90	100
	B2	60	65	70	80	60	65	70	80	60	65	70	80	60	65	70	80
	B3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	B4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D1	130	150	180	200	130	150	180	200	130	150	180	200	130	150	180	200
	D2	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	D3	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
	D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D5	25	30	30	30	25	30	30	30	25	30	30	30	25	30	30	30
	D6	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	T1	25	25	30	35	25	25	30	35	25	25	30	35	25	25	30	35
	T2	25	25	30	35	25	25	30	35	25	25	30	35	25	25	30	35
AP. DE APOIO	aA	35	35	40	40	35	35	40	40	35	35	40	40	35	35	40	40
	bA	45	45	60	60	45	45	60	60	45	45	60	60	45	45	60	60
	hA	6	6	8,9	8,9	6	6	8,9	8,9	6	6	8,9	8,9	6	6	8,9	8,9
	nA	4	4	5	5	4	4	5	5	4	4	5	5	4	4	5	5
	Dapoio	60	60	70	70	60	60	70	70	60	60	70	70	60	60	70	70
PRÉ-LAJE	hPL	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
	bwPL	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
DIAFRAGMA	Ndiaf	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	bwD	20	25	30	30	20	25	30	30	20	25	30	30	20	25	30	30
	hD	70	100	140	160	80	100	150	170	80	100	150	170	80	100	150	170
PILAR	Nintp	2	2	2	2	2	2	2	2	3	3	3	3	2	2	2	2
	Nextp	2	2	2	2	2	2	2	2	3	3	3	3	2	2	2	2
	Hintp	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800
	Hextp	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800
	Distintp	250	250	250	250	270	270	270	270	290	290	290	290	290	290	290	290
	Distextp	300	300	300	300	330	330	330	330	350	350	350	350	350	350	350	350
	Dp	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
LAJE DE TRANSIÇÃO	LarguraLT	1365	1365	1365	1365	1200	1200	1200	1200	1630	1630	1630	1630	1300	1300	1300	1300
	CompLT	400	400	400	400	400	400	400	400	400	400	400	400	400	400	400	400
	EspessuraLT	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
TRAVESSA INTERM.	bwTint	205	205	245	245	205	205	245	245	205	205	245	245	205	205	245	245
	hTint	120	120	150	150	120	120	150	150	120	120	150	150	120	120	150	150
	Cint	1260	1260	1260	1260	1090	1090	1090	1090	1500	1500	1500	1500	1180	1180	1180	1180
TRAVESSA EXTREMA	bwText	155	155	175	175	155	155	175	175	155	155	175	175	155	155	175	175
	hText	120	120	150	150	120	120	150	150	120	120	150	150	120	120	150	150
	Cext	1365	1365	1365	1365	1200	1200	1200	1200	1630	1630	1630	1630	1300	1300	1300	1300
MURO	AlturaM	301	321	381,9	401,9	301	321	381,9	401,9	301	321	381,9	401,9	301	321	381,9	401,9
	hM	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
ALA	CompA	250	250	350	350	250	250	350	350	250	250	350	350	250	250	350	350
	hAla	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
	ChanfroV	75	75	100	100	75	75	100	100	75	75	100	100	75	75	100	100
	ChanfroH	30	30	50	50	30	30	50	50	30	30	50	50	30	30	50	50

Legenda: todas as dimensões em centímetros

Tabela 3.4 - Parâmetros geométricos de cada tipologia das pontes mistas de aço e concreto (MAC) a serem lidos para a modelagem no CSiBridge e Dynamo

	Tipologia	MAC-A-20-P	MAC-A-25-P	MAC-A-30-P	MAC-A-35-P	MAC-A-20-S	MAC-A-25-S	MAC-A-30-S	MAC-A-35-S	MAC-B-20-P	MAC-B-25-P	MAC-B-30-P	MAC-B-35-P	MAC-B-20-S	MAC-B-25-S	MAC-B-30-S	MAC-B-35-S
SEÇÃO	Vão	2000	2500	3000	3500	2000	2500	3000	3500	2000	2500	3000	3500	2000	2500	3000	3500
	Largura	1365	1365	1365	1365	1200	1200	1200	1200	1630	1630	1630	1630	1300	1300	1300	1300
	Nlong	5	5	5	5	4	4	4	4	5	5	5	5	4	4	4	4
	Balanço	102,5	102,5	102,5	102,5	105	105	105	105	115	115	115	115	110	110	110	110
LAJE	h_laje (cm)	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
LONGARINA PERFIL 2	t3_2	130	150	180	200	130	150	180	200	130	150	180	200	130	150	180	200
	t2_2	50	50	50	50	50	50	50	50	50	50	55	55	50	50	55	55
	tf_2	1,90	1,90	2,50	2,50	1,90	1,90	3,15	3,15	2,50	2,50	3,15	3,15	2,50	2,50	3,15	3,15
	tw_2	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,60	1,60	1,25	1,25	1,60	1,60
	t2b_2	50	50	55	55	50	50	55	55	50	50	55	55	50	50	55	55
	tfb_2	2,24	2,24	2,50	2,50	2,50	2,50	3,75	3,75	3,75	3,75	5,00	5,00	3,75	3,75	5,00	5,00
	rad_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LONGARINA PERFIL 1	t3_1	130	150	180	200	130	150	180	200	130	150	180	200	130	150	180	200
	t2_1	50	50	50	50	50	50	50	50	50	50	55	55	50	50	55	55
	tf_1	1,90	1,90	2,50	2,50	1,90	1,90	3,15	3,15	2,50	2,50	3,15	3,15	2,50	2,50	3,15	3,15
	tw_1	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,60	1,60	1,25	1,25	1,60	1,60
	t2b_1	50	50	55	55	50	50	60	60	55	60	65	70	55	60	65	70
	tfb_1	2,24	2,24	2,50	2,50	2,50	2,50	3,75	3,75	3,75	3,75	5,00	5,00	3,75	3,75	5,00	5,00
	rad_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AP. DE APOIO	aA	35	35	40	40	35	35	40	40	35	35	40	40	35	35	40	40
	bA	45	45	60	60	45	45	60	60	45	45	60	60	45	45	60	60
	hA	6	6	8,9	8,9	6	6	8,9	8,9	6	6	8,9	8,9	6	6	8,9	8,9
	nA	4	4	5	5	4	4	5	5	4	4	5	5	4	4	5	5
	Dapoio	60	60	70	70	60	60	70	70	60	60	70	70	60	60	70	70
PRÉ-LAJE	hPL	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	bwPL	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Ndiaf		4	5	6	7	4	5	6	7	4	5	6	7	4	5	6	7
DIAFRAGMA EXTREMO	t3diaf_1	97,5	112,5	135	150	97,5	112,5	135	150	97,5	112,5	135	150	97,5	112,5	135	150
	t2diaf_1	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	tfdiaf_1	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25
	twdiaf_1	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60	1,60
	t2bdiaf_1	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	tfbdiaf_1	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25
	raddiaf_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DIAFRAGMA INTERM.	t3diaf_2	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
	t2diaf_2	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
	tfdiaf_2	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6
	twdiaf_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	t2bdiaf_2	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
	tfbdiaf_2	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6
	raddiaf_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PILAR	Nintp	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	Nextp	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	Hintp	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800
	Hextp	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800
	Distintp	230	230	230	230	230	230	230	230	230	230	230	230	230	230	230	230
	Distextp	305	305	305	305	305	305	305	305	305	305	305	305	305	305	305	305
	Dp	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
LAJE DE TRANSIÇÃO	LarguraLT	1365	1365	1365	1365	1200	1200	1200	1200	1630	1630	1630	1630	1300	1300	1300	1300
	CompLT	400	400	400	400	400	400	400	400	400	400	400	400	400	400	400	400
	EspessuraLT	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
TRAVESSA INTERM.	bwTint	205	205	245	245	205	205	245	245	205	205	245	245	205	205	245	245
	hTint	120	120	150	150	120	120	150	150	120	120	150	150	120	120	150	150
	Cint	1260	1260	1260	1260	1090	1090	1090	1090	1500	1500	1500	1500	1180	1180	1180	1180
TRAVESSA EXTREMA	bwText	155	155	175	175	155	155	175	175	155	155	175	175	155	155	175	175
	hText	120	120	150	150	120	120	150	150	120	120	150	150	120	120	150	150
	Cext	1365	1365	1365	1365	1200	1200	1200	1200	1630	1630	1630	1630	1300	1300	1300	1300
MURO	AlturaM	301	321	381,9	401,9	301	321	381,9	401,9	301	321	381,9	401,9	301	321	381,9	401,9
	hM	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
ALA	CompA	250	250	350	350	250	250	350	350	250	250	350	350	250	250	350	350
	hAla	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
	ChanfroV	75	75	100	100	75	75	100	100	75	75	100	100	75	75	100	100
	ChanfroH	30	30	50	50	30	30	50	50	30	30	50	50	30	30	50	50

Legenda: todas as dimensões em centímetros

3.2 Automatização da modelagem numérica no CSiBridge

Esta etapa teve como objetivo principal desenvolver códigos para a criação automatizada de modelos numéricos de pontes no software CSiBridge através do Python. No âmbito do catálogo, visa-se a criação de modelos para a análise de superestrutura e mesoestrutura, bem como de encontros

3.2.1 Estrutura das planilhas e dos programas elaborados em Python

Conforme ilustrado na Figura 3.1, foi necessária a criação de mais de um código para abranger todas as análises realizadas. Isso se deve, principalmente, à dificuldade envolvida na tentativa de consolidar todas as rotinas em um único código. A estruturação de um único código exigiria a implementação de diversos *loops* e condicionais aninhados, o que tornaria o código excessivamente denso e de difícil interpretação.

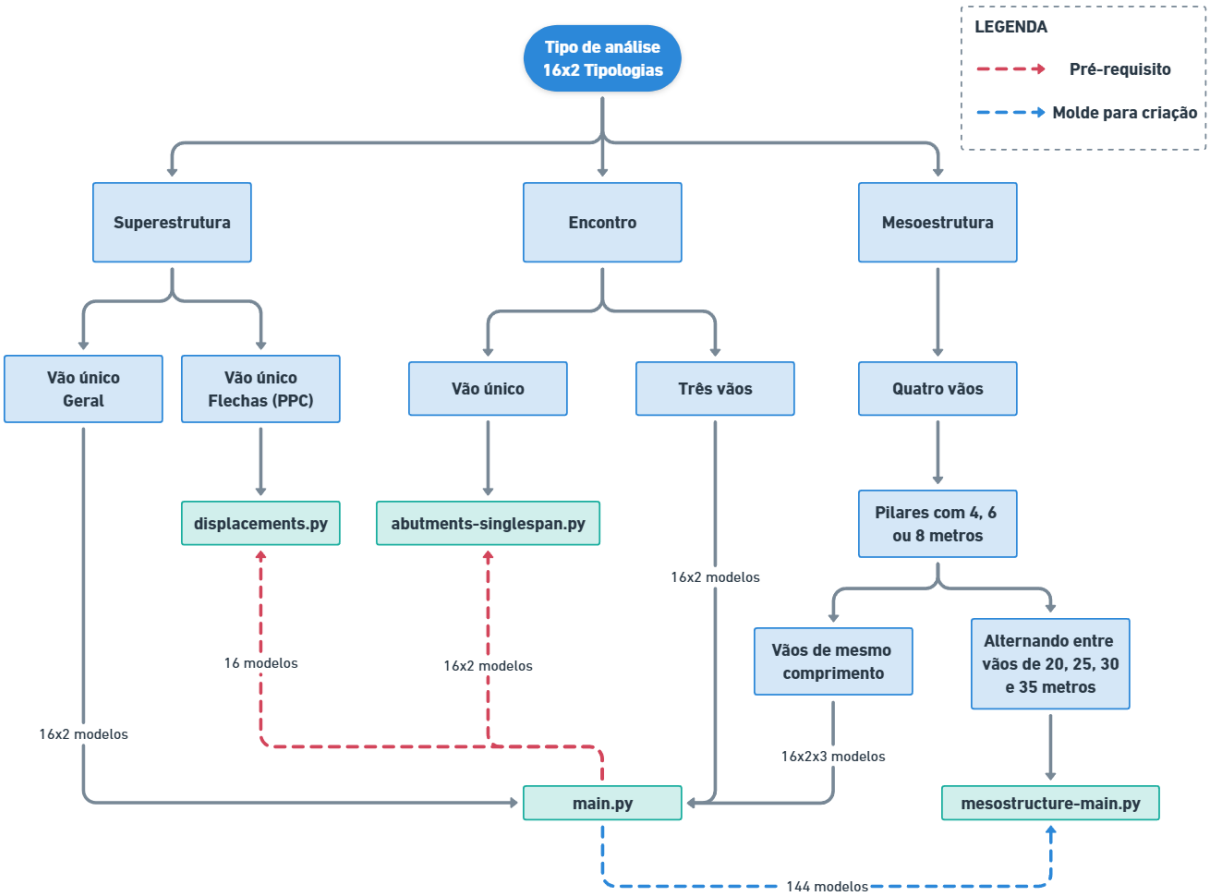
Além disso, esse tipo de estrutura aumenta significativamente o risco de erros lógicos, dificulta o *debugging* e torna a manutenção do código menos eficiente, especialmente à medida que novas funcionalidades são incorporadas. A fragmentação em diferentes códigos permitiu uma abordagem mais modular, facilitando tanto o desenvolvimento quanto a identificação de problemas específicos em cada etapa do processo. A abordagem adotada, bem como o número de modelos gerados para cada análise, foi apresentada por meio de fluxograma (Figura 3.4).

A premissa do catálogo parte da avaliação de peças individuais que podem ser combinadas para formar diferentes tipologias de pontes. Em termos de análise estrutural, em função da estrutura ser biapoiada, a superestrutura foi modelada e calculada como um vão isolado, o que permitiu simplificar as análises sem comprometer a precisão dos resultados.

No que se refere à mesoestrutura, o processo de análise buscou garantir que os elementos de apoio (pilares, travessas ou encontros) absorvam a maior parte dos esforços em comparação aos elementos subjacentes. Esse critério foi fundamental para uma avaliação estrutural precisa, visto que os encontros, por estarem posicionados nas extremidades, recebem uma carga maior do que o pilar oposto ao vão. Para garantir a maior segurança e eficiência no dimensionamento, os encontros foram avaliados tanto para pontes com um único vão quanto para pontes de quatro

vãos, sendo considerada a situação de maior carregamento. Seguindo essa metodologia, os pilares foram avaliados considerando apoios móveis nas extremidades e quatro vãos, com comprimentos iguais e alternados.

Figura 3.4 – Códigos criados em Python para a modelagem no CSiBridge via API referente a cada tipo de análise conduzida nas tipologias PPC e MAC



Para as duas seções transversais analisadas (PPC e MAC), foram desenvolvidos três códigos distintos para cada uma, além da criação de um script específico para a análise de flechas na tipologia PPC. O código principal, "*main*", é responsável pela maior parte das definições e atribuições dos parâmetros no CSiBridge, sendo utilizado direta ou indiretamente em todas as análises, exceto a de mesoestrutura com vãos de comprimento alternados.

Ele é pré-requisito para análises de flechas das longarinas pré-moldadas e protendidas através do código "*displacements*" e para a elaboração de modelos para análise de encontros em um único vão utilizando o código "*abutment-singlesplan*". Esse código, que foi o primeiro elaborado e aprimorado ao longo da pesquisa, serviu

como molde para a criação do código "*mesostructure-main"*, responsável pela criação dos modelos de mesoestrutura com variações nos vãos.

Adicionalmente às planilhas que compilam as informações de geometria referentes à cada tipologia, planilhas auxiliares foram utilizadas para o intercâmbio de informações entre o CSiBridge e códigos desenvolvidos em Python. Como exemplo desse processo, a planilha principal ("*parameters.xlsx*") fornece os dados referentes à geometria da seção transversal da longarina para uma tipologia específica. Tais dados são extraídos via Python para o preenchimento de uma planilha de cálculo das curvas de retração e fluência conforme o Anexo A da NBR 6118 (ABNT, 2023a), além do ganho de rigidez do concreto ao longo do tempo ("*shrinkage-creep-template.xlsx*"). Esta, por sua vez, serve como base para a inserção das respectivas propriedades do material no modelo numérico das pontes da tipologia PPC.

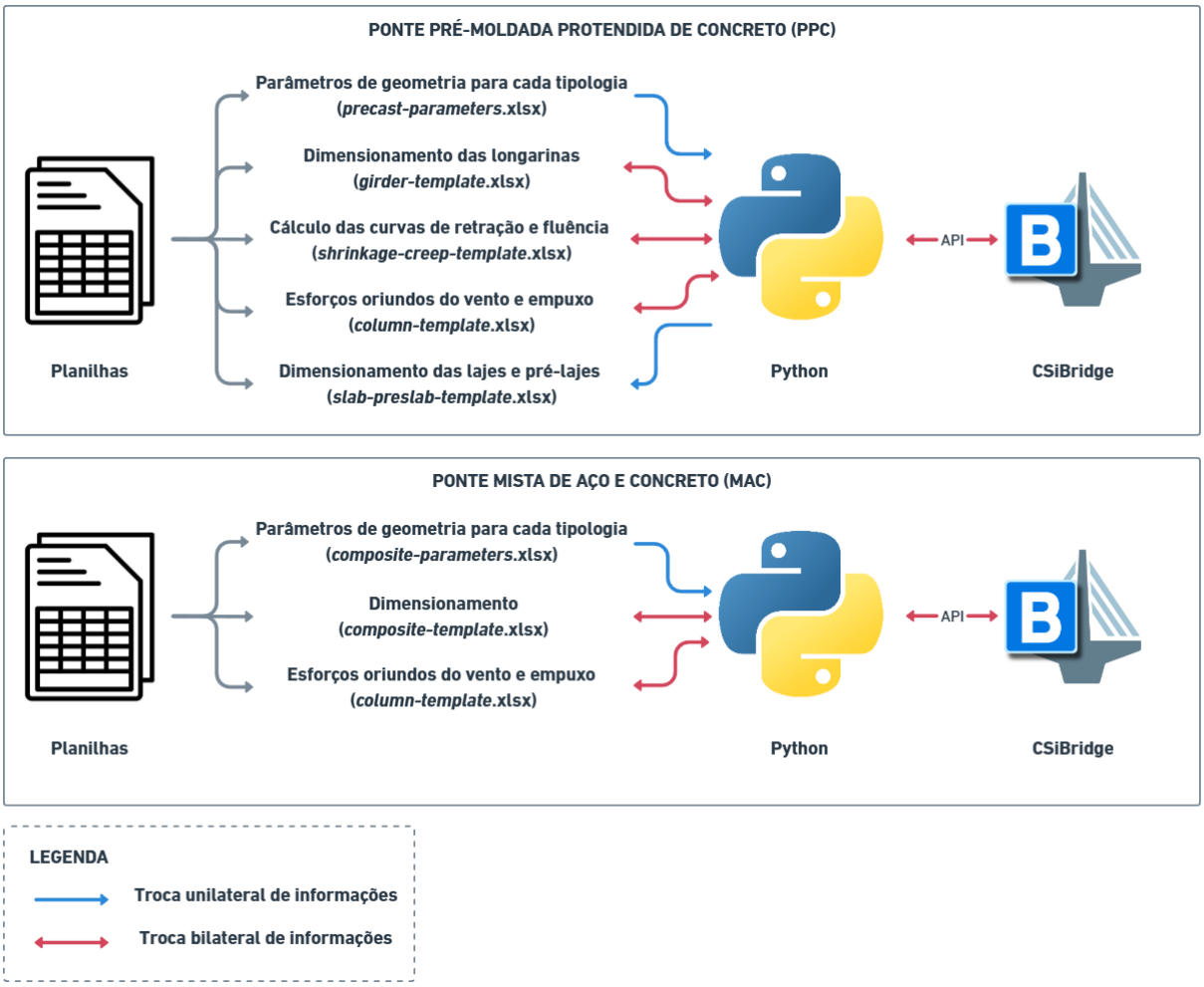
Para viabilizar a inserção dessas propriedades, foi necessário criar um modelo *template* no CSiBridge com o material de concreto previamente definido. Nesse modelo, a opção foi ativada para que as propriedades dependentes do tempo pudessem ser especificadas pelo usuário (caso das pontes da tipologia PPC) ou derivadas de um modelo, caso das tipologias MAC, que utilizou a curva GL 2000 proposta por Gardner e Zhao (1993). Isso se deve à limitação observada na documentação da API, que não apresenta métodos capazes de definir materiais com curvas personalizadas ou pelo modelo GL 2000, funcionalidade disponível na interface gráfica do *software*. Informações constantes entre os diferentes modelos, como o valor da resistência característica do concreto f_{ck} e demais características dependentes, foram integradas diretamente no ambiente do programa, dispensando a necessidade de entrada externa.

A troca de informações entre o Python e as planilhas em Excel se deu por meio da biblioteca *xlwings*. O fluxo de informações estruturado para o código "*main*", ilustrado na Figura 3.5, demonstra que o processo envolveu o uso de várias planilhas como *input*, além das planilhas de dimensionamento que receberam os resultados da análise conduzida no CSiBridge.

Na análise de superestrutura, todas as planilhas indicadas são salvas nas respectivas pastas do modelo, depois de serem alimentadas a partir da planilha

"*parameters.xlsx*" e dos resultados do modelo, caso aplicável. A única diferença para o código "*mesostructure-main*" é a exceção do uso das planilhas de dimensionamento.

Figura 3.5 – Fluxo de informações entre dados de entrada, o código principal "*main*" em Python e o *software* CSiBridge utilizando a comunicação via API



A planilha de dimensionamento das longarinas pré-moldadas de concreto foi inicialmente desenvolvida pelo grupo de pesquisa com o objetivo de calcular uma tipologia específica. Foram realizados ajustes na estrutura da planilha para viabilizar a automatização do processo de dimensionamento. No Anexo A apresenta-se a planilha adotada com os cálculos referentes à tipologia PPC-A-20-P. Nessa planilha, as células em branco correspondem a valores calculados automaticamente ou preenchidos por meio do código principal ("*main*"), enquanto as células em cinza indicam os dados de entrada inseridos manualmente.

A automatização do processo de dimensionamento das longarinas pré-moldadas e protendidas das tipologias PPC envolveu a alimentação automática da

planilha de dimensionamento, incluindo propriedades das seções isoladas e compostas das longarinas, bem como a extração automatizada dos esforços internos obtidos a partir da análise estrutural da superestrutura realizada no CSiBridge. O traçado dos cabos de protensão, o cálculo das perdas e a verificação dos estados-limites foram realizados diretamente na planilha.

Inicialmente, é necessário realizar uma primeira análise estrutural da superestrutura para obtenção dos esforços solicitantes, que são utilizados como entrada na planilha da longarina. Em seguida, o usuário ajusta manualmente a disposição dos cabos de protensão nessa planilha até que os estados-limites relacionados à fissuração considerando protensão limitada (estado-limite de formação de fissuras, ELS-F, e estado-limite de descompressão, ELS-D), bem como o estado-limite no ato da protensão e compressão excessiva na viga e laje, sejam atendidos, gravando-a em seguida.

A partir dela, o código "*displacements*" realiza a leitura dos dados relativos ao traçado dos cabos e os insere no modelo do CSiBridge. Após nova análise, as flechas da longarina são automaticamente extraídas a partir do caso de carregamento definido pelo cronograma de construção e armazenados na mesma planilha, finalizando o processo automático de dimensionamento. A análise de superestrutura das tipologias PPC envolveu também o dimensionamento das lajes e pré-lajes, cujos cálculos, referentes à tipologia PPC-A-20-P, estão apresentados no Anexo B. Elementos mais inerentes ao juízo de projeto, como a definição da bitola das armaduras longitudinais e o espaçamento entre barras, não foram incluídos no processo de automatização, porquanto exigem uma avaliação técnica mais subjetiva e engloba particularidades do detalhamento de armaduras.

De forma análoga ao realizado para a pré-moldada, a planilha de dimensionamento de longarinas metálicas foi adaptada. O processo de automatização das pontes mistas de aço e concreto não englobou todas as verificações necessárias da superestrutura. O código desenvolvido alimenta a planilha quanto a informações da seção transversal e dimensões dos perfis, além de extrair os esforços nas longarinas e diafragmas oriundos da análise de superestrutura. No Anexo C apresenta-se a planilha adotada com os cálculos referentes à tipologia MAC-A-20-P.

O fluxograma de funcionamento do código “*main.py*” está ilustrado na Figura 3.6, enquanto os processos que envolvem o código “*displacements.py*” foram descritos por meio da Figura 3.7, complementando as informações apresentadas pela Figura 3.4 e Figura 3.5.

Figura 3.6 – Fluxograma simplificado que representa os processos do código "*main.py*" conforme o tipo de análise e a ligação com outros códigos também desenvolvidos em Python

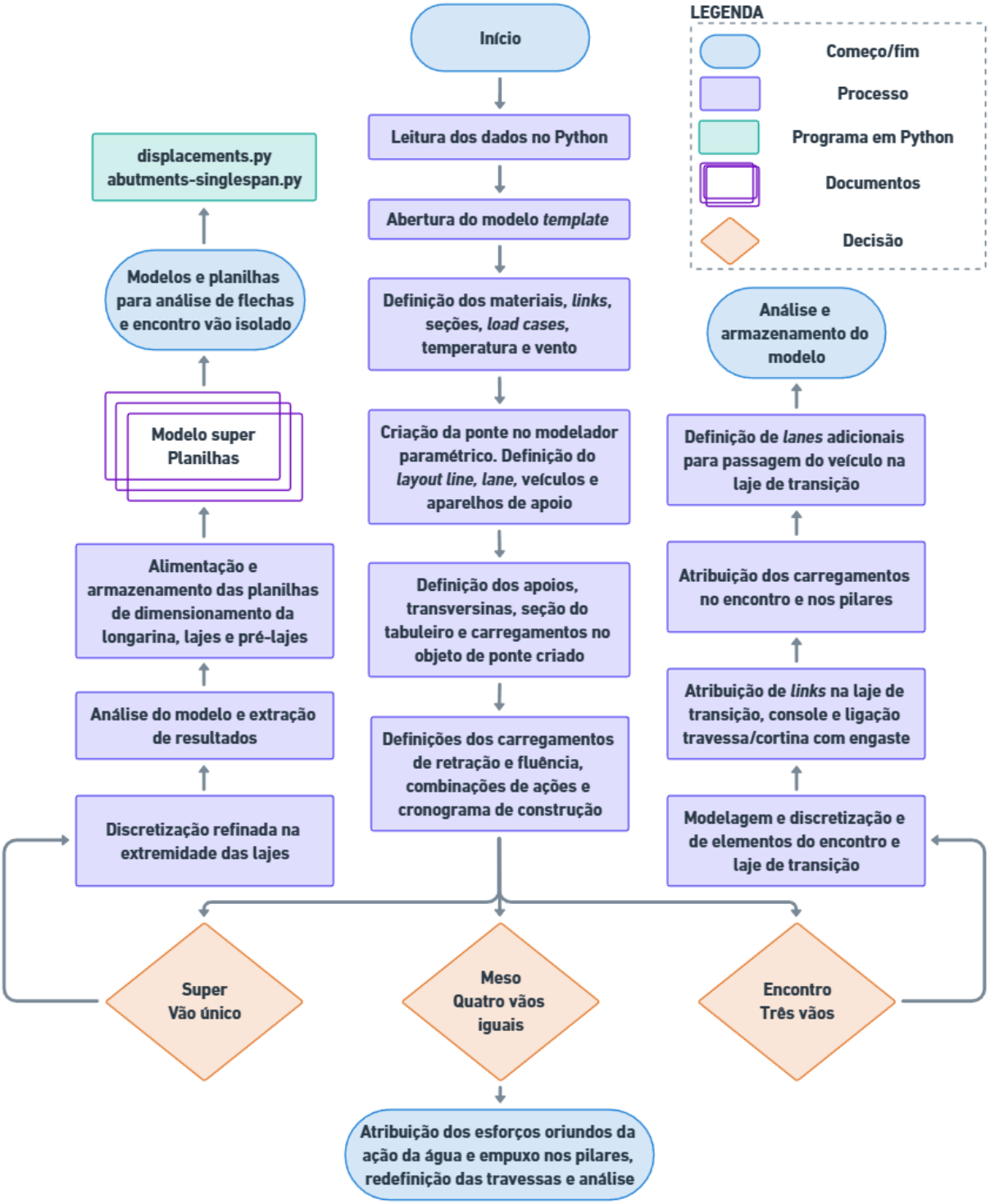
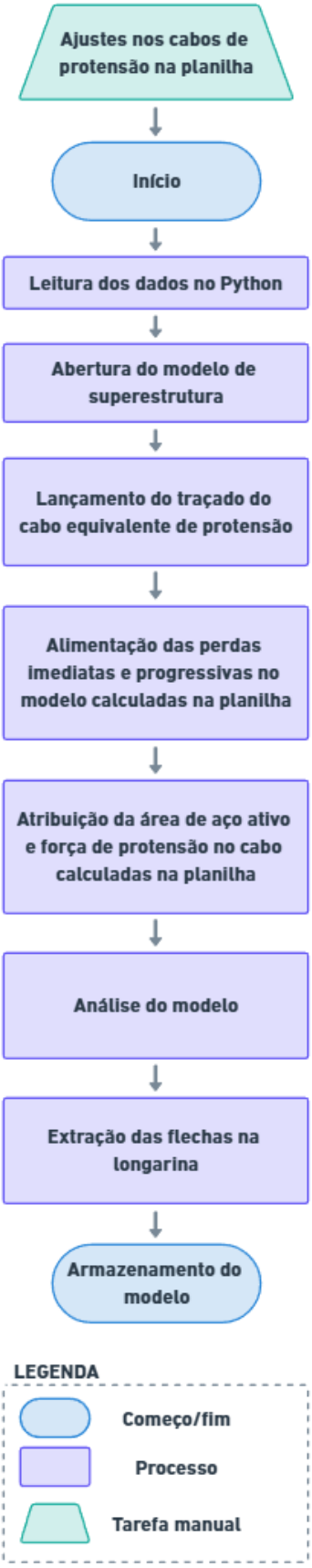


Figura 3.7 – Fluxograma simplificado que representa os processos do código "*displacements.py*" para a entrada de dados dos cabos de protensão no modelo e extração de deslocamentos



Embora tanto os modelos da ponte pré-moldada e mista de aço e concreto tenham tido suas particularidades, ambas seguiram a mesma estrutura de funcionamento ilustrada (Figura 3.4). O código “*abutments-singlesplan.py*” funciona de forma análoga ao “*displacements.py*” (Figura 3.5), partindo da abertura do modelo referente à superestrutura da tipologia e englobando os processos de modelagem e definição das condições de contorno já representados para análise do encontro em três vãos. Já a estrutura do “*mesostructure-main.py*” é muito parecida com o “*main.py*”, sendo realizados apenas alguns ajustes para a estrutura de *loop* englobar variações de vãos e pilares.

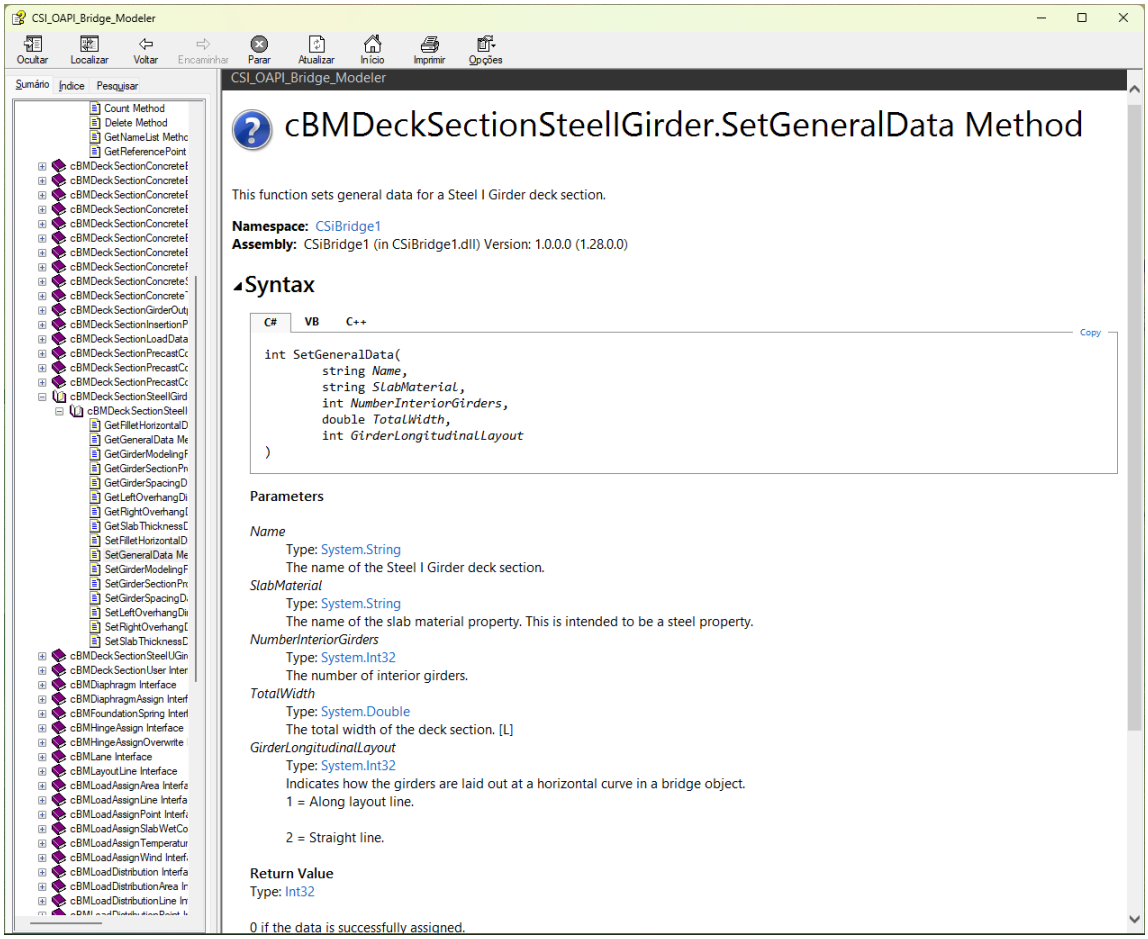
Os códigos foram estruturados com base em laços de repetição sobre uma lista de tipologias, permitindo que todos os modelos correspondentes a uma determinada análise fossem gerados automaticamente em uma única execução. Para cada tipologia iterada, o código realiza a leitura dos dados específicos associados e transcreve essas informações para o ambiente do CSiBridge.

A implementação consiste em uma sequência sistemática de chamadas aos métodos dos objetos *SapModel* e *BridgeModeler*, por meio dos quais são definidos e atribuídos todos os parâmetros necessários à modelagem das pontes. Tomando-se o código base fornecido pela documentação, os pré-requisitos para instanciação desses objetos já é realizada. A documentação da API foi consultada de forma recorrente, garantindo a correta utilização dos recursos disponíveis. Na Figura 3.8 apresentam-se os principais parâmetros exigidos para a chamada do método responsável pela atribuição dos dados da seção das pontes mistas de aço e concreto. Esses parâmetros são:

- “*Name*”: argumento do tipo *string* que define o nome da seção transversal (*Deck Section*) à qual os parâmetros serão atribuídos;
- *“SlabMaterial”*: também do tipo *string*, corresponde ao material a ser atribuído à laje do tabuleiro;
- *“NumerInteriorGirders”*: parâmetro do tipo número inteiro (*int* em Python) que indica a quantidade de longarinas interna
- *“TotalWidth”*: argumento do tipo ponto flutuante (variável do tipo *float* em Python) que representa a largura total do tabuleiro;

- *"GirderLongitudinalLayout"*: parâmetro do tipo inteiro que define o *layout* longitudinal das longarinas, devendo-se utilizar o valor 1 para disposição conforme o *layout line* definido ou 2 para disposição reta.

Figura 3.8 – Método e argumentos necessários para atribuição dos dados referentes à seção transversal das pontes mistas de aço e concreto



Por se tratar de um método específico do modelador paramétrico de pontes, a chamada correspondente no código Python assume a forma BridgeModeler.DeckSection.SteelIGirder.SetGeneralData("Ponte", "C35", 3, 13.85, 1). Conforme também descrito na documentação da API, esse método retorna o valor 0 quando a atribuição dos parâmetros é realizada com sucesso. Isso é válido para todos os métodos de atribuição (*set*). Esse retorno foi particularmente útil durante as etapas de experimentação do código e de *debugging*.

O CSiBridge oferece a possibilidade de definir e atribuir parâmetros diretamente na base de dados do modelo por meio de tabelas estruturadas, em vez de depender exclusivamente da interface gráfica, denominadas *Interactive Database*

Tables. Esse recurso é altamente vantajoso em cenários que envolvem grandes volumes de dados ou modificações em massa, permitindo uma manipulação mais ágil, padronizada e menos propensa a erros manuais. Essas tabelas, bem como as de *output*, são acessíveis por meio da API, permitindo edição direta no modelo e a extração dos resultados de análise.

Todos os códigos elaborados para a conexão pela API com o CSiBridge através do Python se encontram no Apêndice A. Ao todo, foram gerados 352 modelos para todas as análises propostas, distribuídos da seguinte forma:

- 16 para análise de superestrutura para cada tipologia, além dos 16 modelos para a análise de flechas nas tipologias PPC, totalizando 48 modelos;
- 32 para análise de encontros em vão único e três vãos para cada tipologia, totalizando 64 modelos;
- 16 modelos de análise de mesoestrutura com vãos de mesmo comprimento para cada uma das três alturas de pilares em que foram analisadas as variações para cada tipologia, totalizando 96 modelos;
- 72 modelos de análise de mesoestrutura com variações nos comprimentos dos vãos para cada uma das três alturas de pilares em que foram analisadas as variações para cada tipologia, totalizando 144 modelos.

3.2.2 Premissas para a criação dos modelos numéricos

As ações a serem consideradas no cálculo, descritas por meio da Tabela 3.5, são descritas a seguir:

- Peso-próprio da estrutura: foi adotado peso específico de 25 kN/m³ para o concreto armado e 78,5 kN/m³, conforme NBR 6118 (ABNT, 2023a). O CSiBridge já considera esse carregamento conforme as propriedades das seções inseridas nos elementos;
- Pavimentação e recapeamento: conforme o item 7.2.3 da NBR 7187 (ABNT, 2021), foi considerado um peso específico de 24 kN/m³ para o CBUQ (Concreto Betuminoso Usinado à Quente), com espessura de 10 cm, totalizando uma carga adicional de 2,4 kN/m² na laje do tabuleiro. Ainda, de

acordo com o mesmo item, foi considerada uma carga adicional de 2 kN/m² para atender a um possível recapeamento;

- Guarda-rodas: a área da seção transversal das barreiras do tipo *New Jersey*, de 0,203 m², foi determinada conforme o Manual 698, Manual de Projeto de Obras-de-Arte Especiais (DNIT, 1996), resultando em um carregamento de 5,075 kN/m;

- Empuxo de terra: foi calculado considerando peso específico do solo de 18 kN/m³ e ângulo de atrito efetivo de 30º, conforme prescrições da NBR 7187 (ABNT, 2021). A pressão atribuída nos elementos sujeitos a essa ação foi a média. De forma análoga, o empuxo de solo devido a cargas móveis foi considerado de acordo com o item 7.3.5 da NBR 7187 (ABNT, 2021);

- Retração e fluência: definidos conforme o Anexo A da NBR 6118 (ABNT, 2023a) para a geometria de cada tipologia;

- Sobrecarga de construção: 2 kN/m² na laje do tabuleiro referente à mobilização de operários e equipamentos. Esse carregamento foi considerado diretamente nas planilhas de cálculo, não sendo necessário lançá-lo no programa;

- Vento: os esforços oriundos da ação do vento foram calculados conforme as prescrições da NB-2 (ABNT, 1982), considerações de pressões constantes com uma velocidade característica de 45 m/s e como reticulados planos múltiplos, ambas hipóteses preconizadas na NBR 6123 (ABNT, 2023b). Desses três métodos, adotou-se o crítico;

- Temperatura: conforme item 11.4.2 da NBR 6118 (ABNT, 2023a), foi considerado um acréscimo ou redução de temperatura de ± 15ºC em todo o tabuleiro da ponte. Ainda, foi considerada distribuição não uniforme de temperatura ao longo da altura da seção transversal conforme indicado na NBR 7187 (2021);

- Água: os esforços da pressão hidrodinâmica, bem como o acúmulo de detritos, foram determinados conforme as prescrições da NBR 7187 (ABNT, 2021);

- Carga móvel: o veículo de projeto adotado foi o TB-450, conforme especificado na NBR 7188 (ABNT, 2024), que possui 75 kN/roda e carga de multidão de 5 kN/m² em seu entorno. A fim de considerar o efeito dinâmico das cargas móveis, foi calculado o Coeficiente de Impacto Vertical (CIV), Coeficiente de Número de Faixas (CNF) e o Coeficiente de Impacto Adicional (CIA) de acordo com a mesma norma nas planilhas;

- Impacto: de acordo com a NBR 7188 (ABNT, 2024), foi considerado um esforço de 100 kN atuando na extremidade das defensas em um comprimento de 50 cm, sendo considerado para dimensionamento um espraiamento dessa carga por meio de um ângulo de 45º.

Tabela 3.5 - Tabela de carregamentos considerados para os modelos numéricos das pontes consoante recomendações práticas e normas da ABNT

Peso-próprio	γca = 25 kN/m³ e γs = 78,5 kN/m³ (ABNT, 2023a)
Pavimentação	2,4 kN/m² (ABNT, 2021)
Recapeamento	2 kN/m² (ABNT, 2021)
Guarda-rodas	5,075 kN/m (DNIT, 1996)
Empuxo de terra	γsolo = 18 kN/m³, ϕsolo = 30° (ABNT, 2021)
Retração e fluência	Anexo A da NBR 6118 (ABNT, 2023a)
SC construção	2 kN/m²
Vento	ABNT (1982) e ABNT (2023b)
Temperatura	Variação uniforme de ± 15º C (ABNT, 2023a) Distribuição não uniforme (ABNT, 2021)
Água	ABNT (2021)
Veículo	TB-450 ABNT (2024)
Impacto	ABNT (2024)

Como exemplo de lançamento desses carregamentos, ilustra-se o modelo da tipologia PPC-A-20-P de análise de encontro de vão único, cuja malha em elementos finitos e condições de contorno consideradas estão ilustradas na Figura 3.9. Conforme detalhe em (a), foram lançadas molas na superfície dos elementos de área das lajes de transição em uma faixa de comprimento de 2 metros, conforme anexo K da NBR 7187 (ABNT, 2021); para tanto, a discretização desse elemento foi realizada de forma adequada para garantir o comprimento dessa região de apoio elástico. Por conta das informações quanto ao solo em um escopo do catálogo serem de difícil definição, optou-se por adotar uma mola de 5 kgf/cm³ para representar o comportamento do solo. Graficamente, nota-se que o CSiBridge trata o lançamento dessa base elástica como se fossem molas nodais.

De acordo com o detalhe (b) da Figura 3.9, as longarinas são ligadas às travessas por meio de dois *links*: um rígido que liga o topo da longarina até o eixo do aparelho de apoio e outro, efetivamente com a rigidez do aparelho de apoio, que liga esse nó até o eixo da travessa. Em (c), foram definidos *links* rotulados que ligam a laje de transição ao console, também conforme recomendações de cálculo desse elemento conforme anexo K da NBR 7187 (ABNT, 2021). O detalhe (d) mostra que todos os pilares foram considerados engastados na base, enquanto o detalhe (e) ilustra os *links* rígidos considerados para a vinculação da parede cortina à travessa.

Além dessas condições de contorno, o programa também aplica automaticamente restrições (*constraints*) para garantir que todos os nós do tabuleiro interceptados pelas longarinas tenham os mesmos deslocamentos que os nós das próprias longarinas. Na Figura 3.10 até a Figura 3.25, são ilustrados os carregamentos lançados no mesmo modelo (com unidades em kN, m e ºC), e na Figura 3.26 é apresentado o cronograma de construção proposto para essa tipologia de ponte. Ressalta-se que o lançamento dos carregamentos tanto das tipologias PPC e MAC tiveram suas particularidades, porém os carregamentos lançados foram essencialmente os mesmos.

Figura 3.9 – Malha de elementos finitos associada ao modelo numérico da tipologia PPC-A-20-P para análise de encontro de vão único e condições de contorno associadas aos elementos de (a) laje de transição, (b) longarina, (c) console, (d) pilares e (e) travessas e cortina

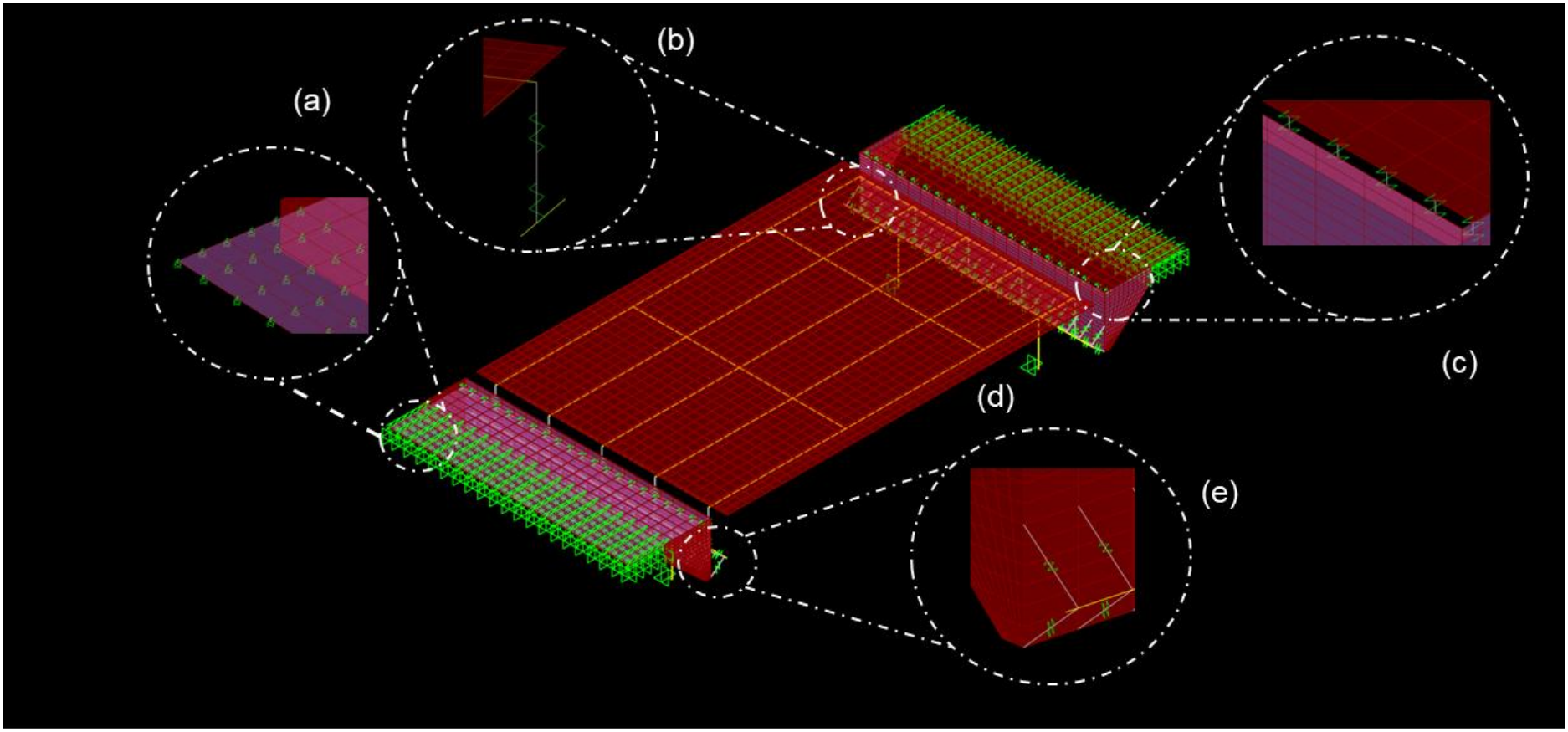


Figura 3.10 – Carregamento (kN/m²) referente à pavimentação no tabuleiro do modelo numérico

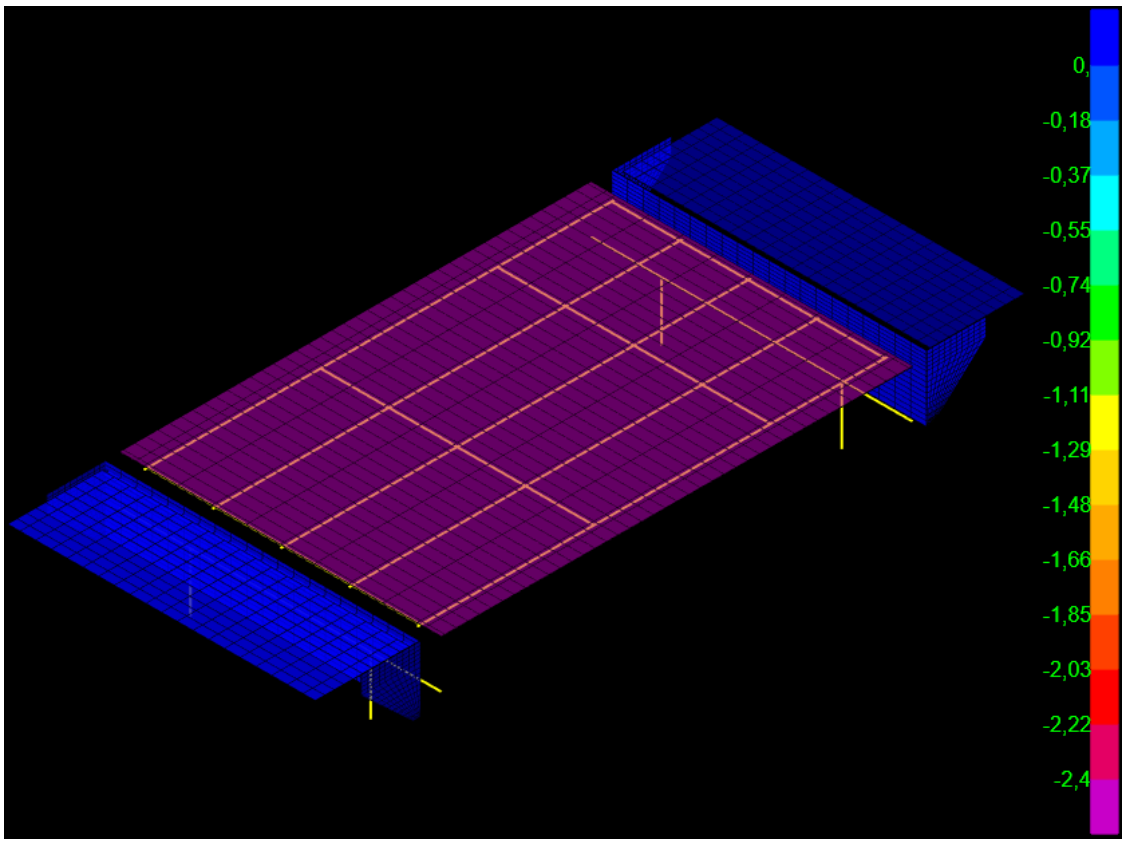


Figura 3.11 – Carregamento (kN/m²) referente ao recapeamento no tabuleiro do modelo numérico

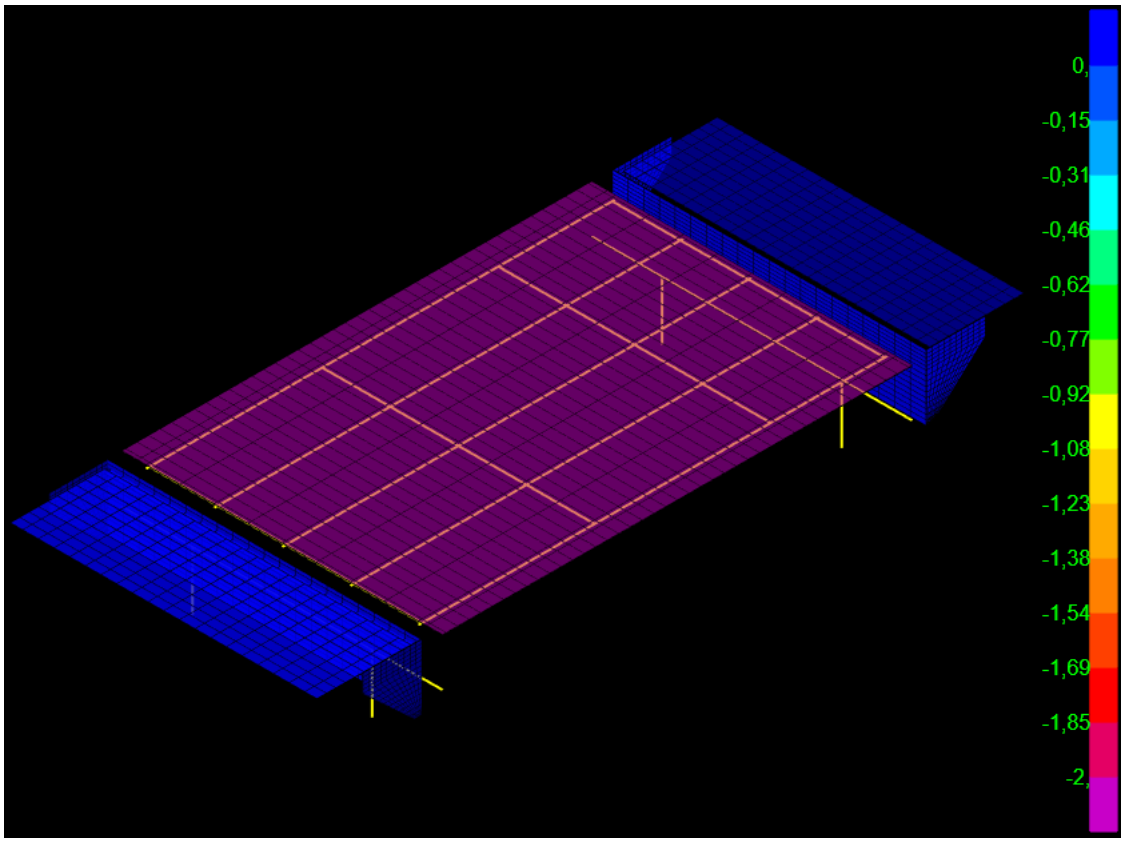


Figura 3.12 – Carregamento (kN/m) referente às defensas no tabuleiro do modelo numérico

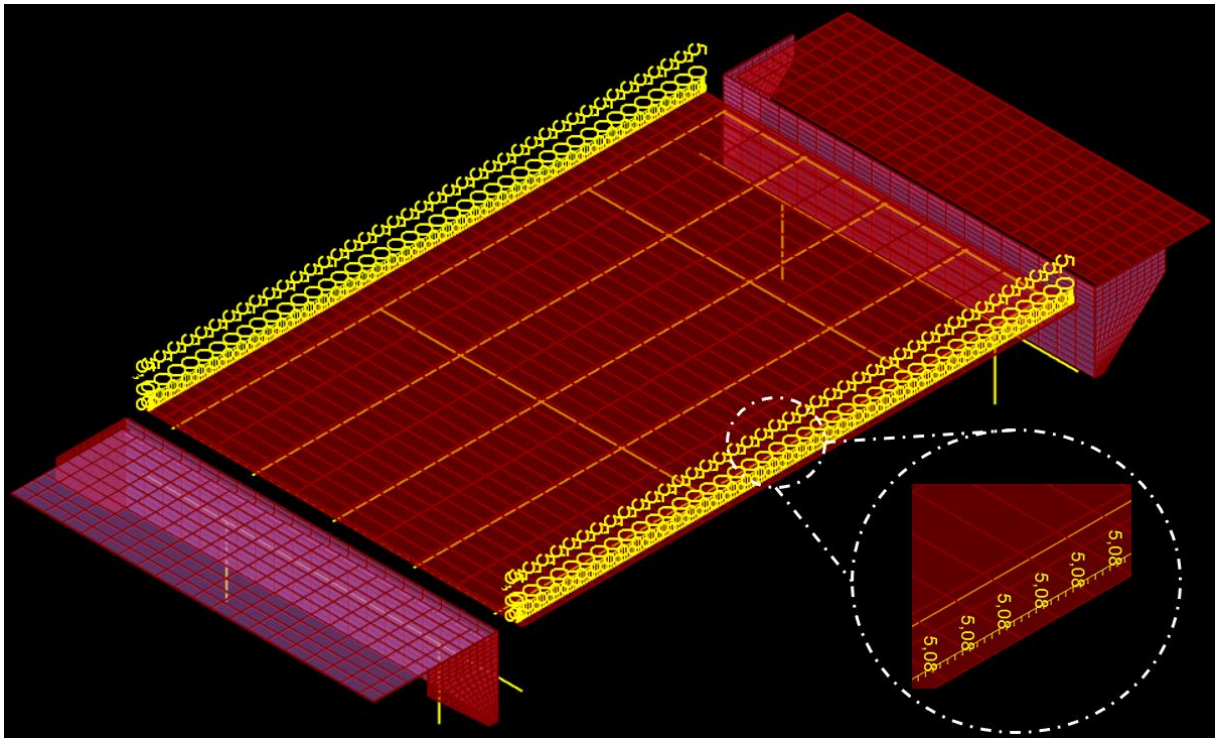


Figura 3.13 – Carregamento (kN/m²) referente ao empuxo de solo no encontro do modelo numérico

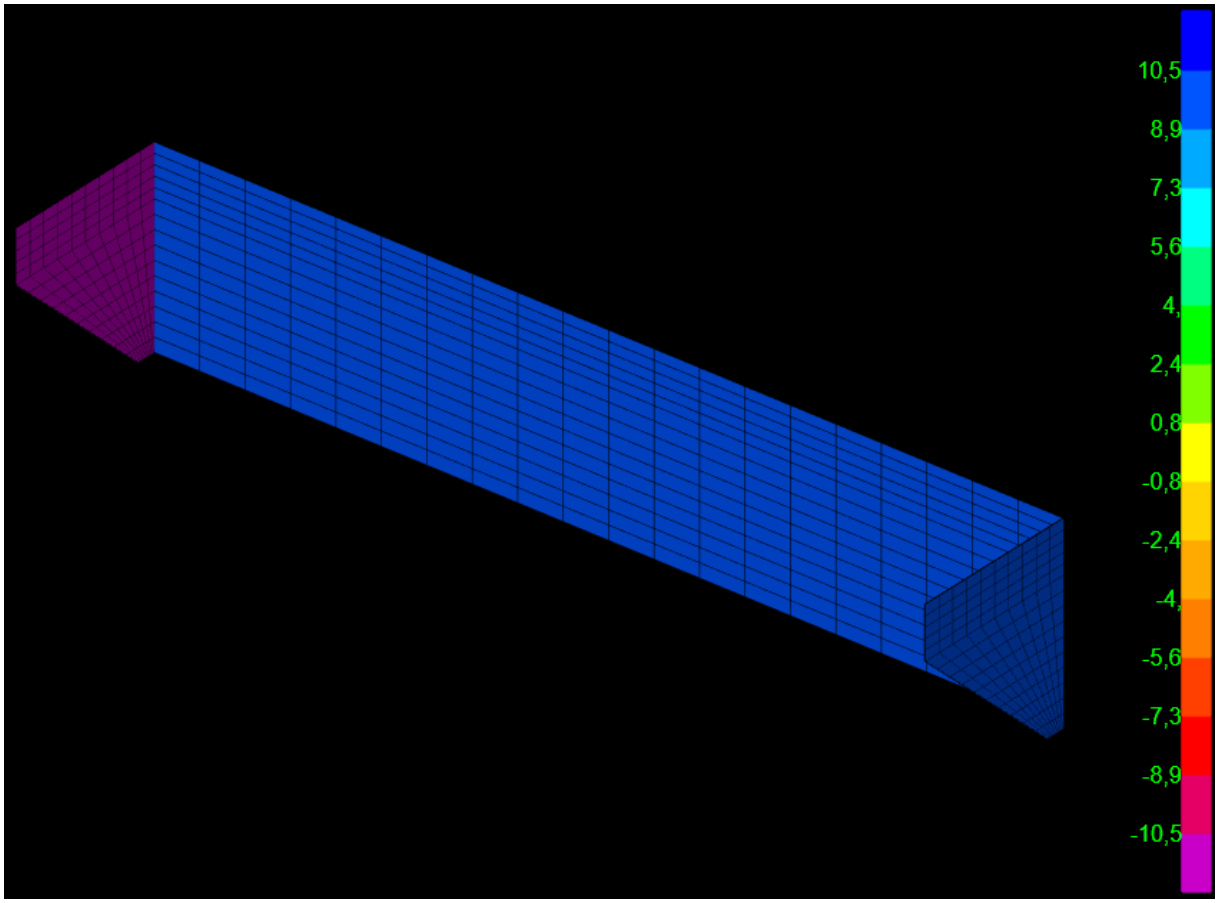


Figura 3.14 – Carregamento (kN/m²) referente ao empuxo de solo devido à carga móvel no encontro do modelo numérico

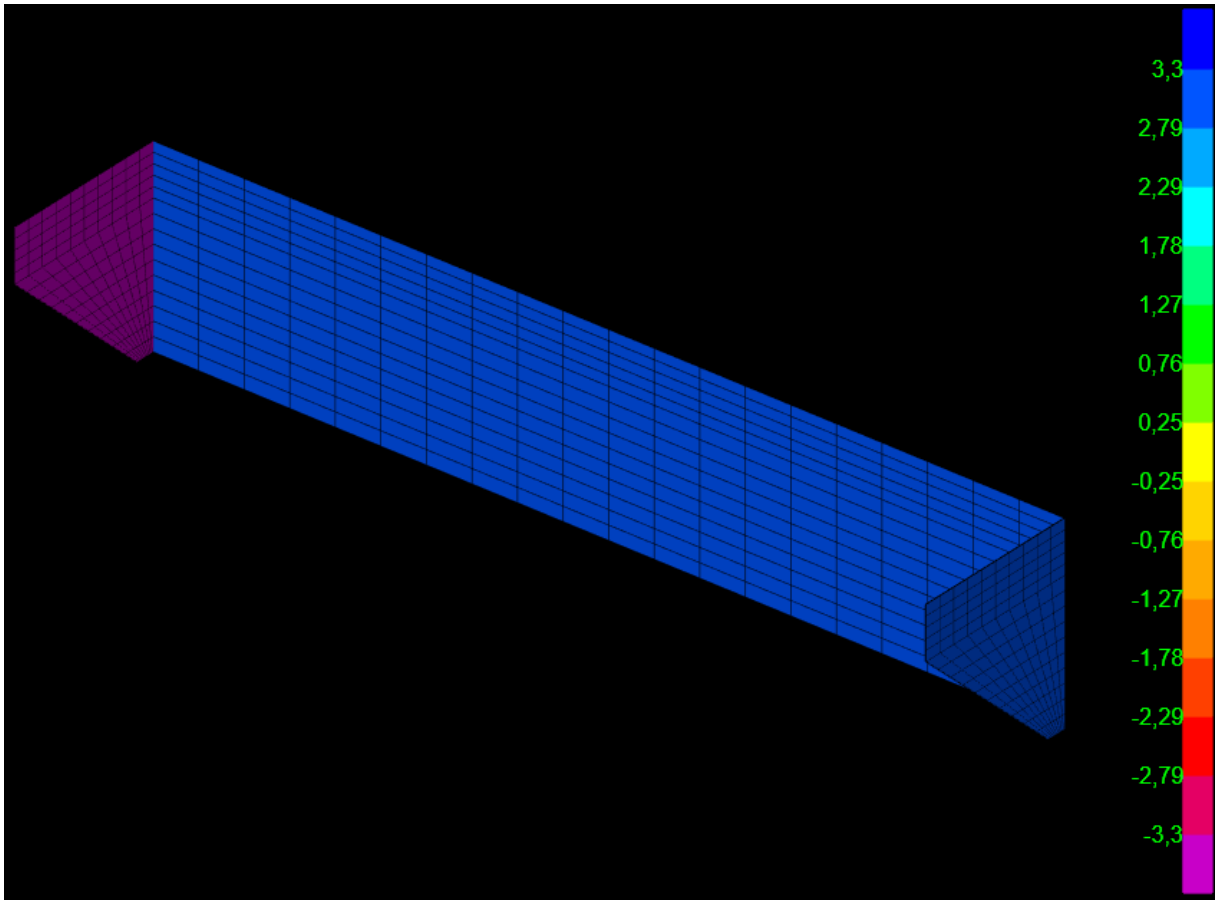


Figura 3.15 – Curva de retração referente ao concreto das longarinas no modelo numérico

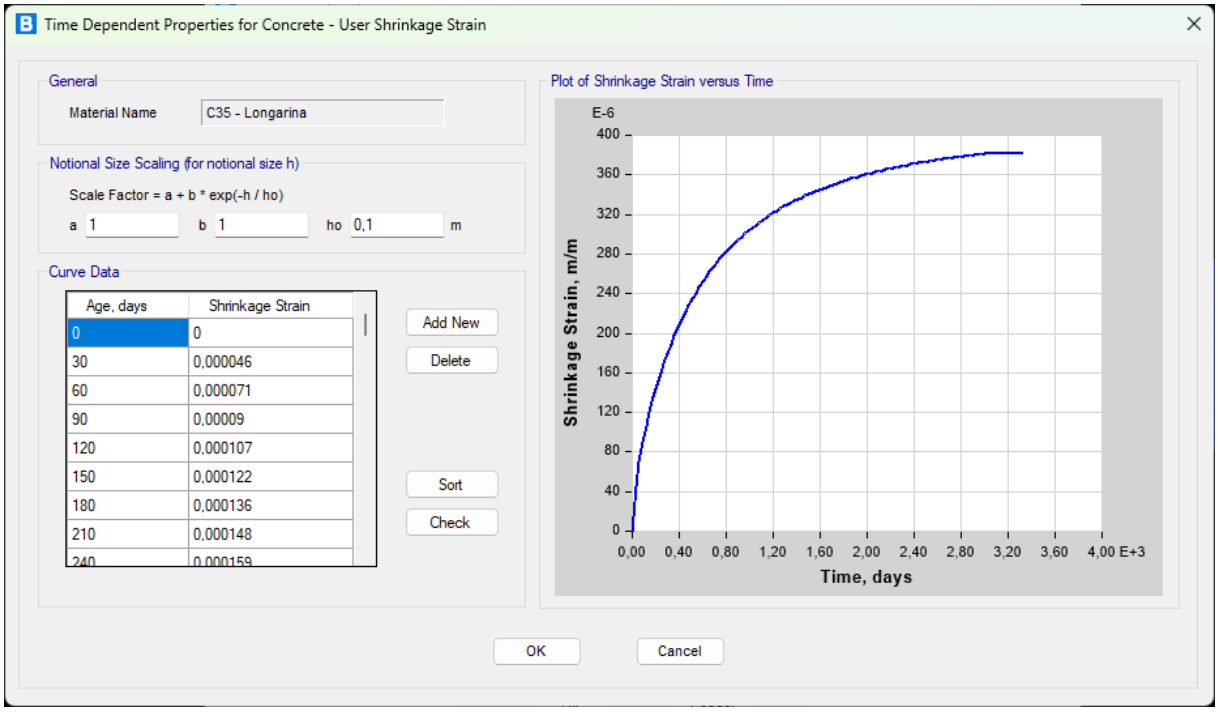


Figura 3.16 – Curva de fluência referente ao concreto das longarinas no modelo numérico

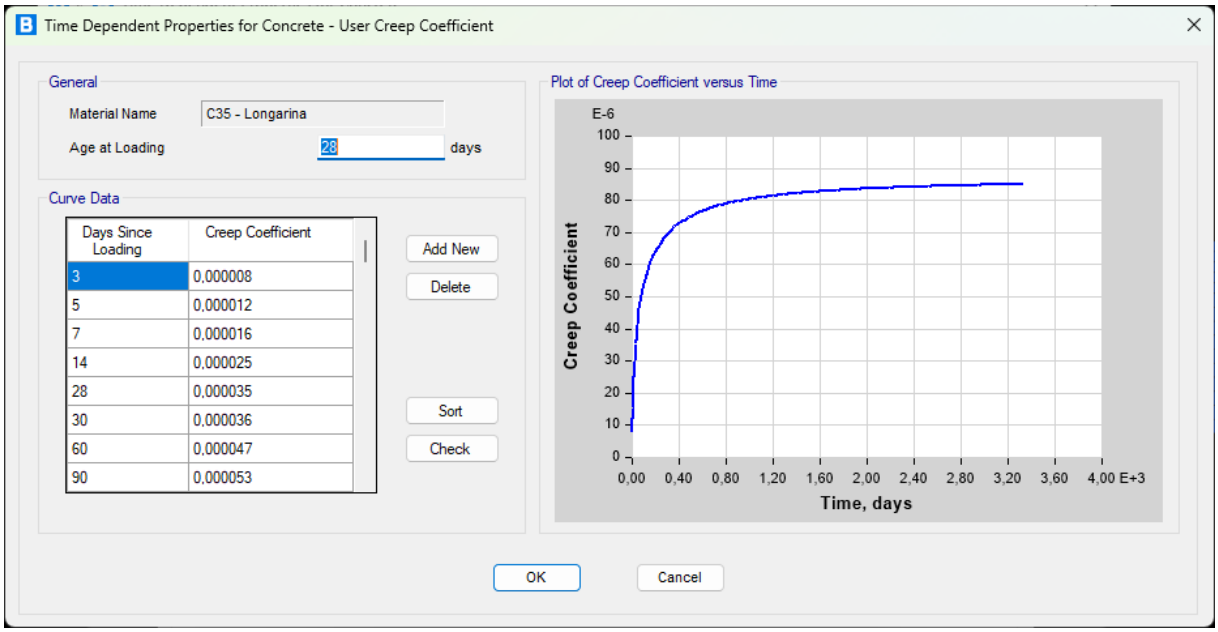


Figura 3.17 – Curva de ganho de rigidez referente ao concreto das longarinas no modelo numérico

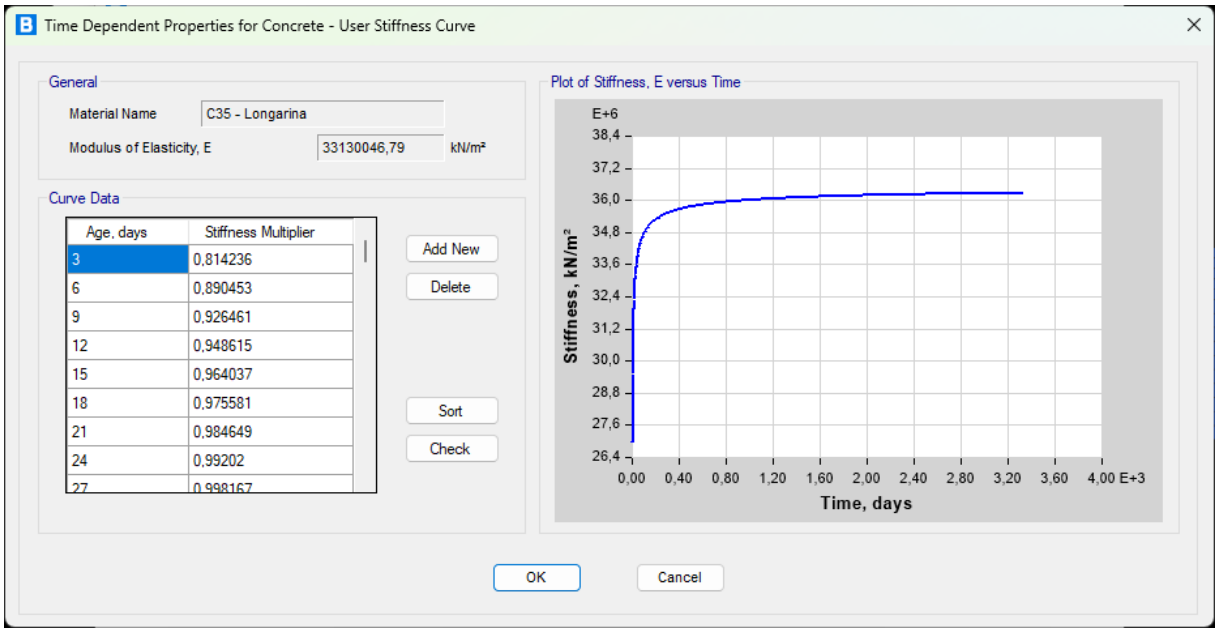


Figura 3.18 – Carregamento (kN/m) referente à ação do vento no tabuleiro do modelo numérico

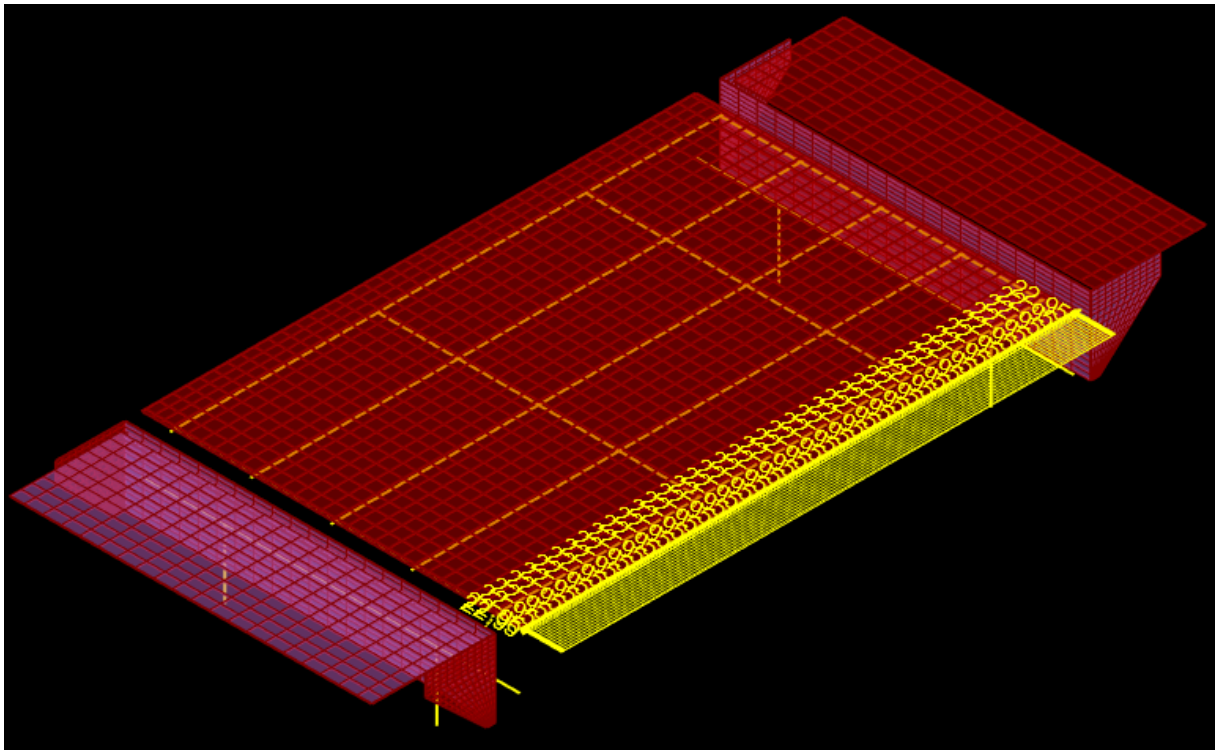


Figura 3.19 - Carregamento (°C) referente à variação uniforme negativa de temperatura no tabuleiro do modelo numérico

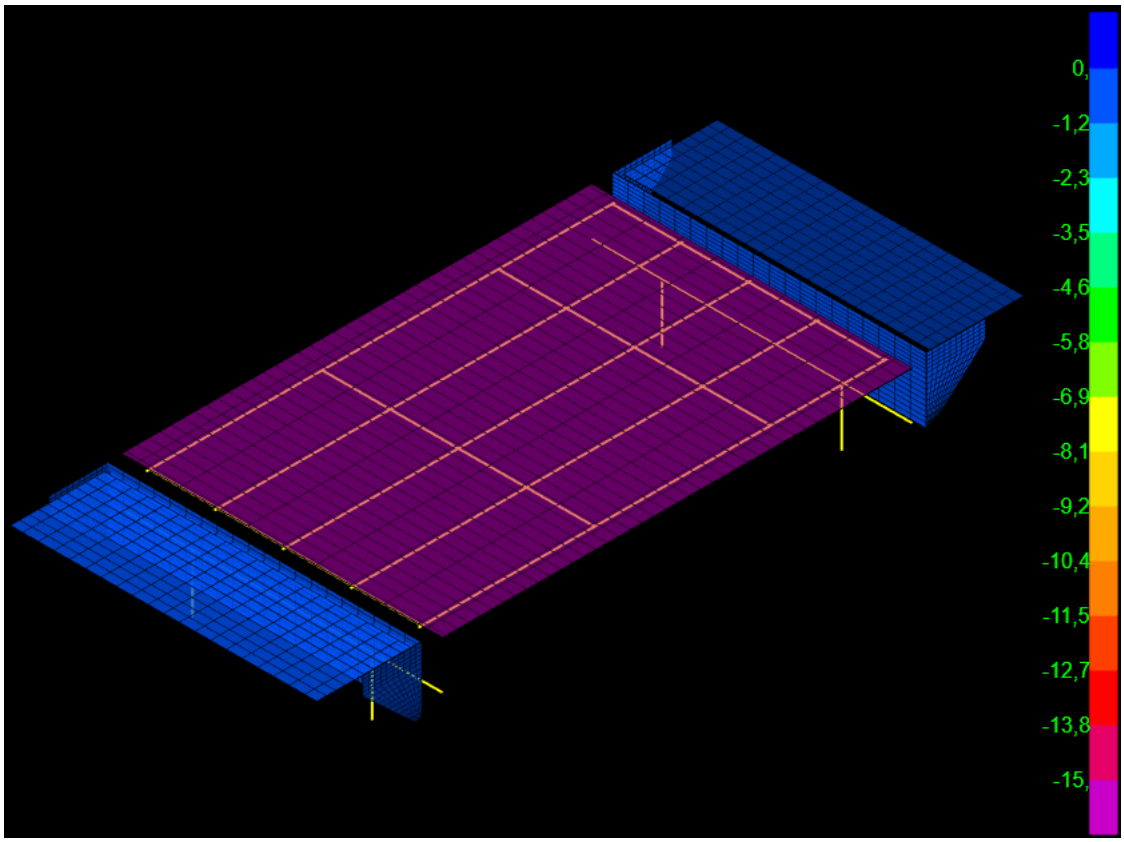


Figura 3.20 – Carregamento (°C) referente à variação uniforme positiva de temperatura no tabuleiro do modelo numérico

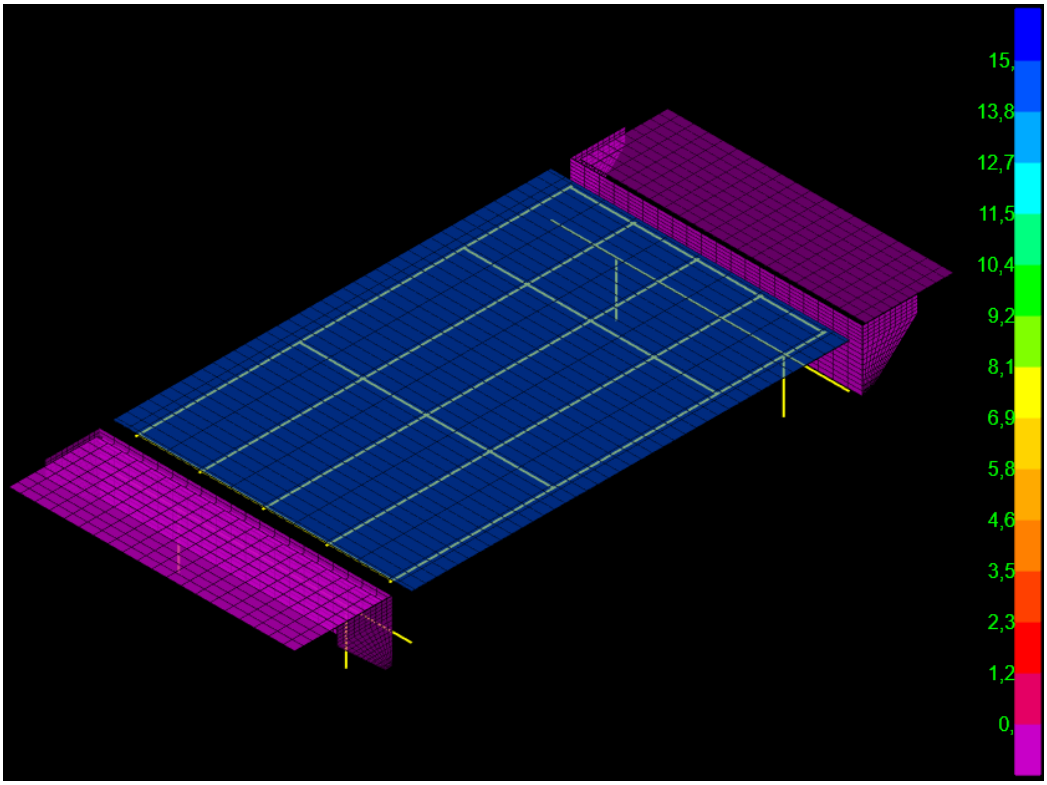


Figura 3.21 – Carregamento (°C) referente à variação não uniforme positiva e negativa de temperatura ao longo da seção da superestrutura do modelo numérico

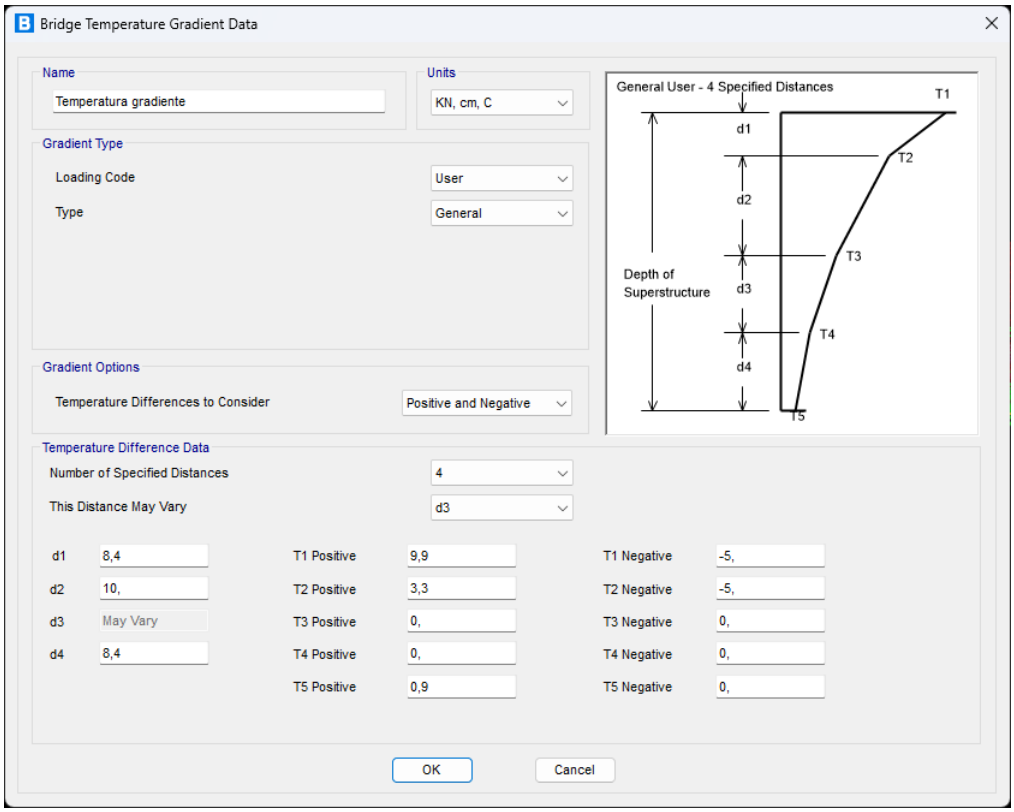


Figura 3.22 – Carregamento (kN/m) nos pórticos de extremidade referente à ação da água no modelo numérico

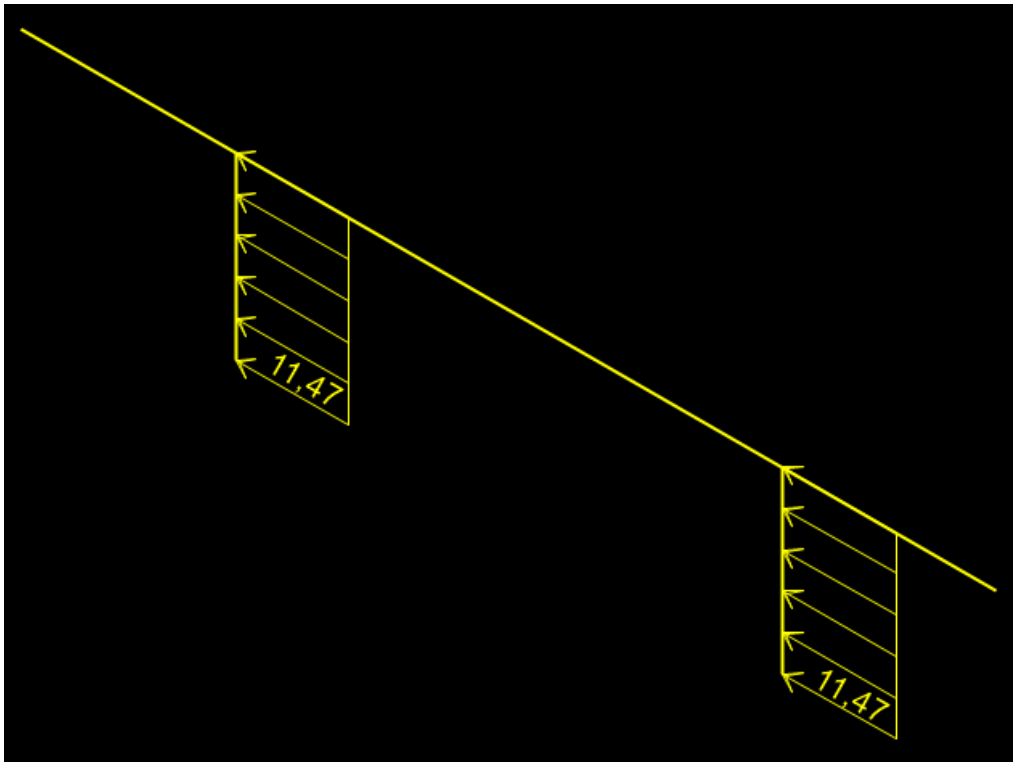


Figura 3.23 – Carregamento (kN/m) nos pórticos de extremidade referente ao empuxo de solo no modelo numérico

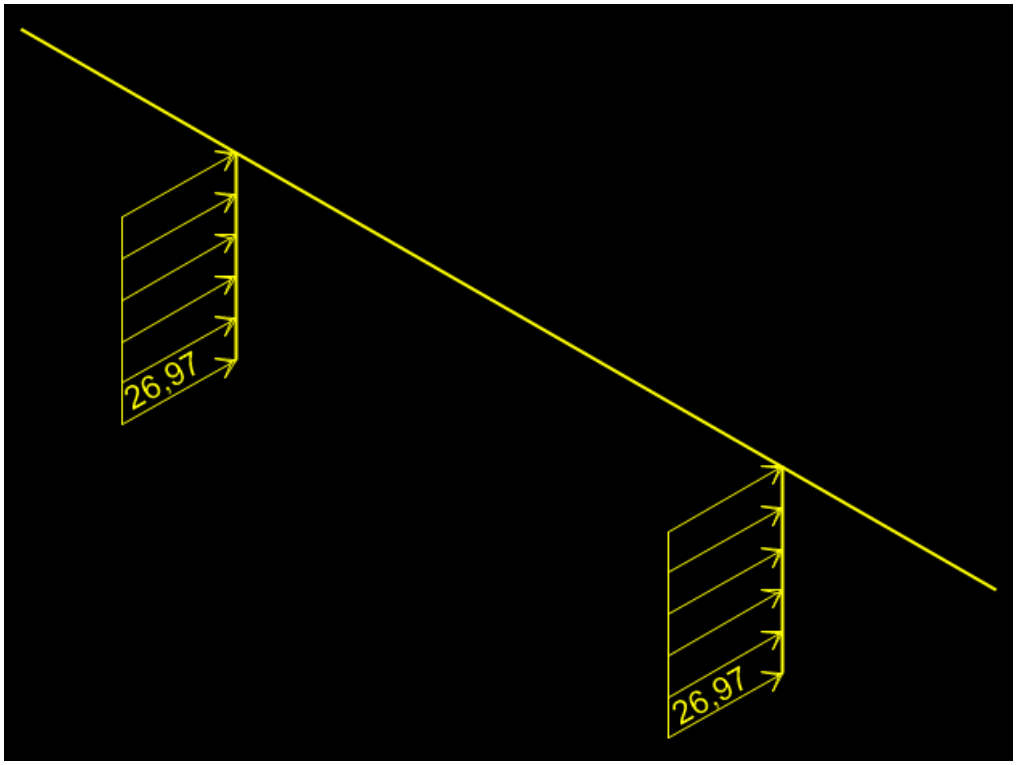


Figura 3.24 – Discretização da *lane* adotada para a passagem do veículo TB-450

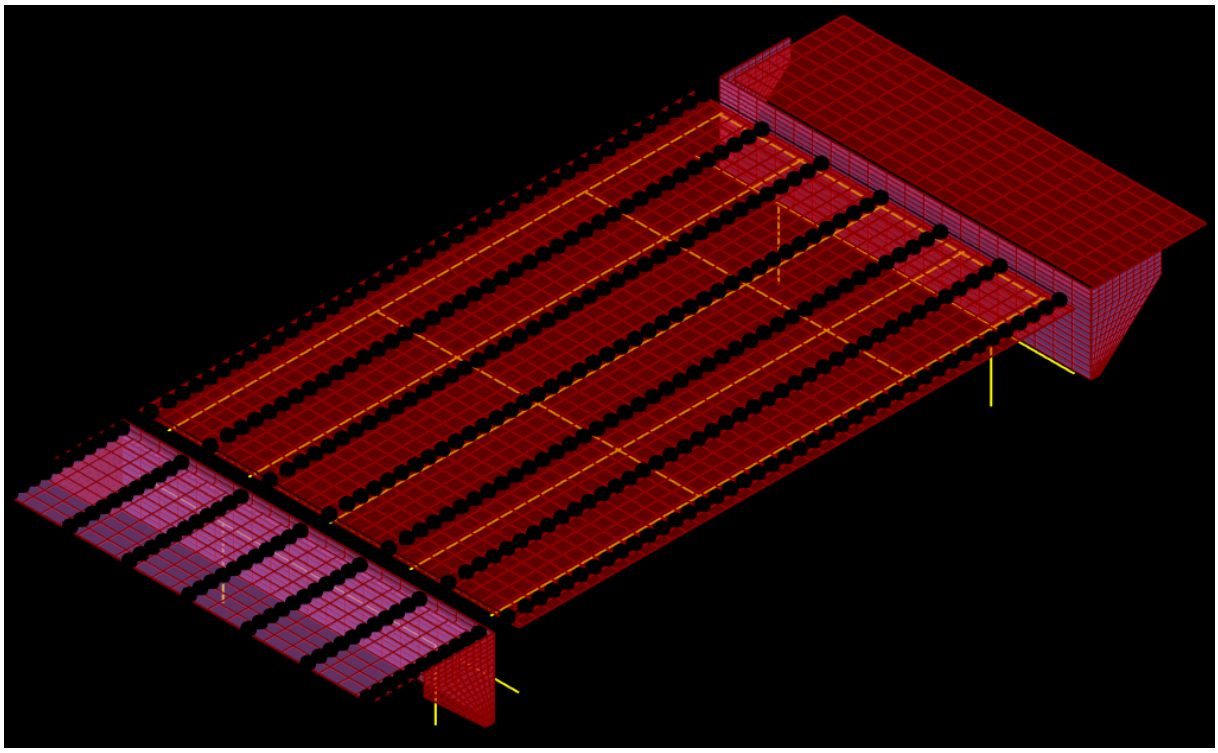


Figura 3.25 – Carregamento (kN e kN.m) referente ao impacto nas alas dos encontros do modelo numérico

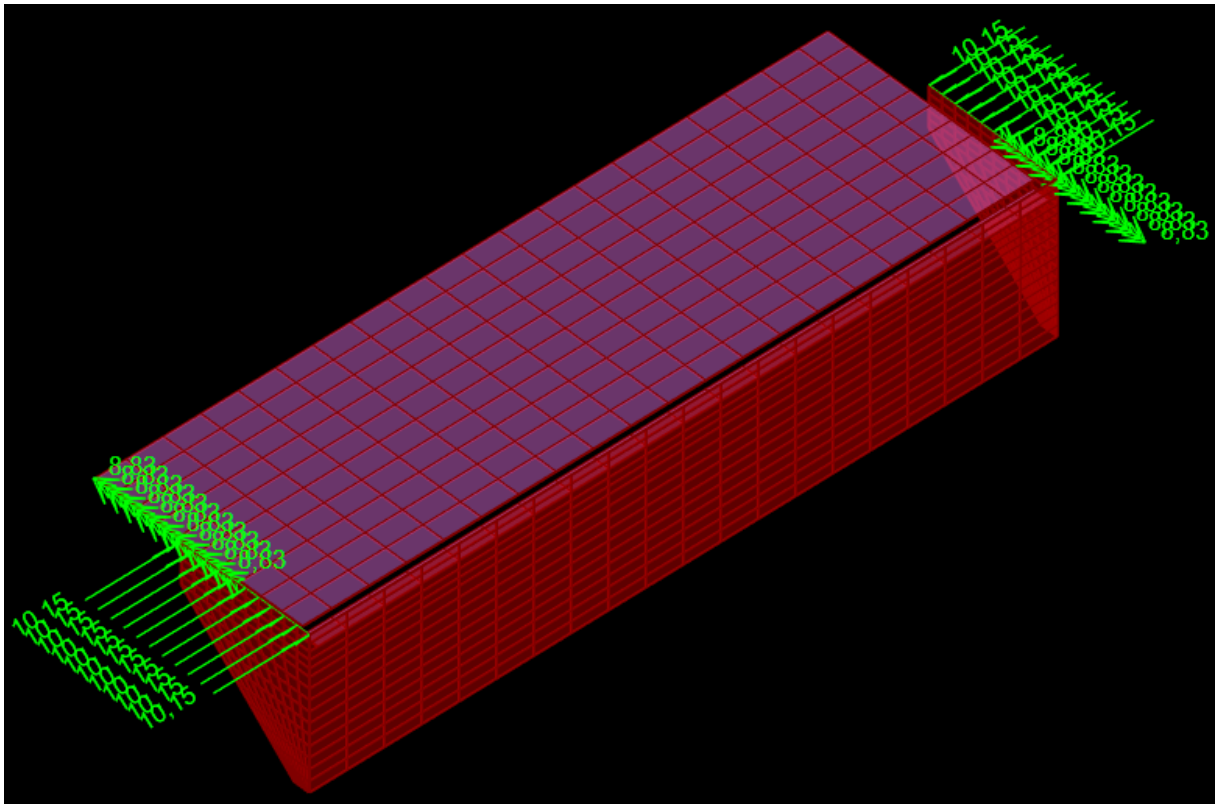
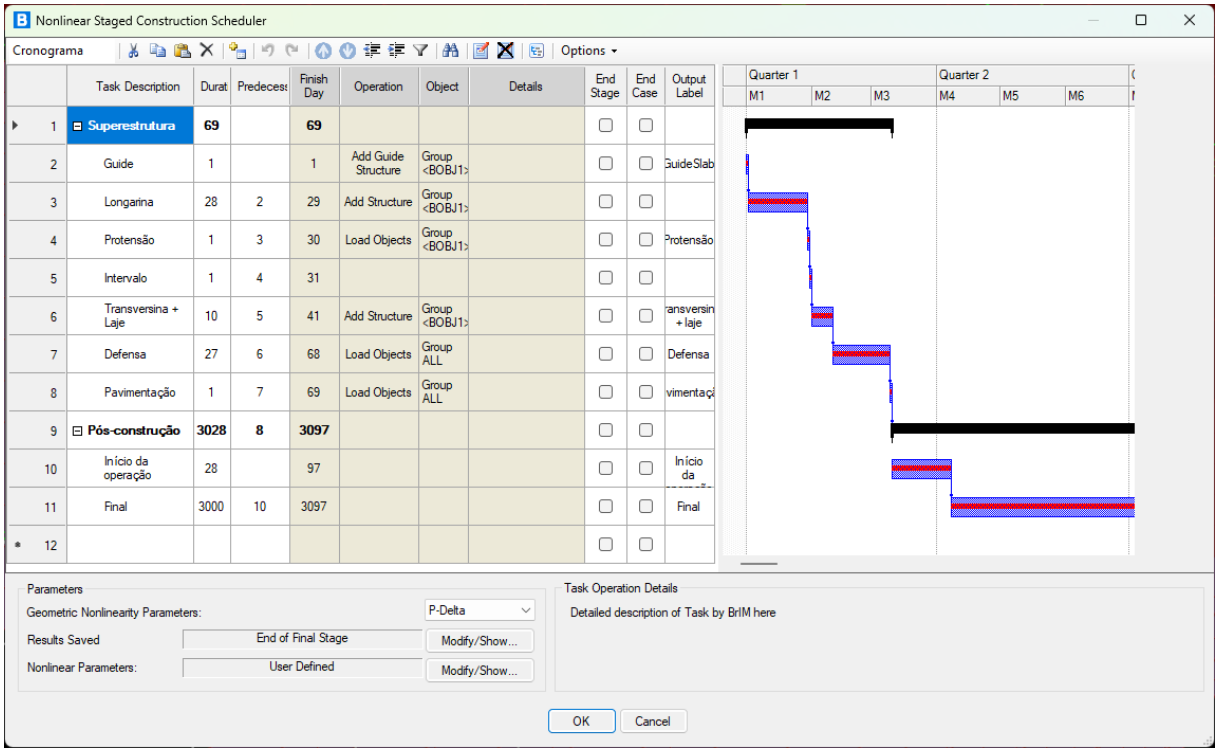


Figura 3.26 – Determinação das etapas do cronograma de construção associado ao modelo numérico para determinação das flechas das longarinas pré-moldadas e protendidas



Os coeficientes de minoração de resistência dos materiais considerados constam na Tabela 3.6, enquanto as combinações de ações consideradas foram definidas de acordo com a NBR 8681 (ABNT, 2004) e NBR 6118 (ABNT, 2023a), conforme equações a seguir e os coeficientes de ponderação da Tabela 3.7 e Tabela 3.8.

- Combinações últimas normais

$$F_d = \sum_{i=1}^{m} (\gamma_{gi} F_{Gi,k}) + \gamma_q \left[F_{Q1,k} + \sum_{j=2}^{n} \psi_{0j,ef} F_{Qj,k} \right]$$

- Combinações últimas excepcionais

$$F_d = \sum_{i=1}^{m} (\gamma_{gi} F_{Gi,k}) + F_{Q,exc} + \gamma_q \sum_{j=1}^{n} \psi_{0j,ef} F_{Qj,k}$$

- Combinações últimas especiais ou de construção

$$F_d = \sum_{i=1}^{m} (\gamma_{gi} F_{Gi,k}) + \gamma_q \left[F_{Q1,k} + \sum_{j=2}^{n} \psi_{0j,ef} F_{Qj,k} \right]$$

- Combinações frequentes de serviço

$$F_d = \sum_{i=1}^{m} F_{Gi,k} + \psi_1 F_{Q1,k} + \sum_{j=2}^{n} \psi_{2j} F_{Qj,k}$$

- Combinações quase permanentes de serviço

$$F_d = \sum_{i=1}^{m} F_{Gi,k} + \sum_{j=1}^{n} \psi_{2j} F_{Qj,k}$$

- Combinações raras de serviço

$$F_d = \sum_{i=1}^{m} F_{Gi,k} + F_{Q1,k} + \sum_{j=2}^{n} \psi_{1j} F_{Qj,k}$$

Tabela 3.6 - Coeficientes de ponderação utilizados para minoração das resistências dos materiais

Combinações	γ_c	γ_s
Normais	1,40	1,15
Especiais ou de construção	1,20	1,15
Excepcionais	1,20	1,00

Fonte: adaptado de NBR 6118 (ABNT, 2023a)

Tabela 3.7 - Coeficientes de ponderação utilizados para majoração das cargas permanentes

Ação/coeficiente	γ_{gELU}	$\gamma_{gfavELU}$	$\gamma_{gE/C}$	$\gamma_{gfavE/C}$	γ_{gEXC}	$\gamma_{gfavEXC}$	γ_{gELS}
Peso-próprio	1,35	1,00	1,25	1,00	1,15	1,00	1,00
Pavimentação	1,35	1,00	1,25	1,00	1,15	1,00	1,00
Recapeamento	1,35	1,00	1,25	1,00	1,15	1,00	1,00
Guarda-rodas	1,35	1,00	1,25	1,00	1,15	1,00	1,00
Empuxo de terra	1,35	1,00	1,25	1,00	1,15	1,00	1,00
Protensão	1,20	0,90	1,20	0,90	1,20	0,90	1,00
Retração e fluência	1,20	0,00	1,20	0,00	0,00	0,00	1,00

Fonte: adaptado de NBR 6118 (ABNT, 2023a) e NBR 8681 (ABNT, 2004)

Tabela 3.8 - Coeficientes de ponderação utilizados para majoração das cargas variáveis

Ação/coeficiente	γ_{qELU}	$\gamma_{qE/C}$	γ_{qEXC}	ψ_0	ψ_1	ψ_{1fad}	ψ_2
SC construção	1,50	1,30	-	0,80	-	-	-
Vento	1,40	1,20	1,00	0,60	0,30	-	0,00
Temperatura	1,20	1,00	1,00	0,60	0,50	-	0,30
Água	1,50	1,30	1,00	0,60	0,50	-	0,30
Veículo	1,50	1,30	1,00	0,70	0,50	*	0,30
Impacto	-	-	1,50	0,70	-	-	-

Legenda: *0,50 para as longarinas, 0,70 para as transversinas e 0,80 para a laje do tabuleiro

Fonte: adaptado de NBR 6118 (ABNT, 2023a) e NBR 8681 (ABNT, 2004)

Destaca-se que todas essas combinações foram realizadas nas planilhas de cálculo. No código, implementaram-se apenas as combinações de carregamento referentes ao estado-limite último nas tipologias de PPC (uma vez que a planilha já implementava as combinações), utilizando-se o recurso "*Range*" do CSiBridge. Esse recurso permite que, dentro de uma combinação composta por múltiplos casos de carga, o *software* avalie automaticamente a contribuição de cada caso, identificando se sua presença ou ausência intensifica a solicitação estrutural. Tal funcionalidade simplifica a geração de envoltórias de esforços em análises que envolvem diversas ações variáveis além da carga móvel.

3.3 Automatização da modelagem no Revit com o Dynamo

Esta etapa tem como objetivo principal desenvolver um grafo no ambiente do Dynamo para criação dos modelos paramétricos de pontes do catálogo digital no Revit de forma automatizada a partir da mesma base de dados utilizada para a modelagem numérica. A lógica de programação implementada no Dynamo, por conta do próprio funcionamento do ambiente, foi definida para a geração de apenas um modelo, a depender da entrada de dados do usuário.

3.3.1 Criação de famílias paramétricas

Como etapa preliminar à modelagem no Revit utilizando o Dynamo, conforme ilustrado na Figura 3.1, foram desenvolvidas famílias paramétricas personalizadas para atender as exigências geométricas de projetos de pontes não contempladas pelas famílias nativas da plataforma. Esse foi o caso de elementos como encontros e aparelhos de apoio, cuja complexidade demandou a criação de famílias específicas.

O processo de criação de famílias no Revit envolve o desenvolvimento de objetos paramétricos que podem ser reutilizados em diferentes contextos de projetos. Esse processo se inicia na seleção de um modelo base (*template*), que define o comportamento e as categorias do objeto a ser modelado. A Autodesk disponibiliza diversos *templates* específicos, como os voltados para elementos estruturais, componentes de mobiliário, aparelhos sanitários, entre outros, permitindo que o usuário escolha aquele mais adequado às características do elemento. Essa definição inicial influencia os planos de referência disponíveis, os parâmetros padrões e as opções de hospedagem da família no ambiente projetual, sendo um passo

fundamental para garantir o comportamento adequado do objeto dentro do modelo BIM.

Posteriormente à inicialização do *template* específico do objeto a ser modelado, tem-se a definição de planos de referência, que atuam como guias para o posicionamento da modelagem a ser realizada. A partir desses planos, são inseridas cotas às quais se associam parâmetros (que podem ser de tipo ou instância), permitindo que essas medidas sejam controladas e modificadas conforme necessário pelo usuário.

A modelagem das geometrias pode ser feita por diferentes ferramentas disponíveis no ambiente de criação de famílias, como extrusão, mesclagem (*blend*), varredura (*sweep*), mescla com varredura (*swept blend*) e formas vazias, possibilitando a representação de formas simples ou complexas conforme a necessidade do componente estrutural ou arquitetônico. Para tornar os elementos mais completos e personalizáveis, também são incluídos parâmetros de material, que permitem ao usuário definir posteriormente o tipo de material aplicado à família no contexto da solução projetual.

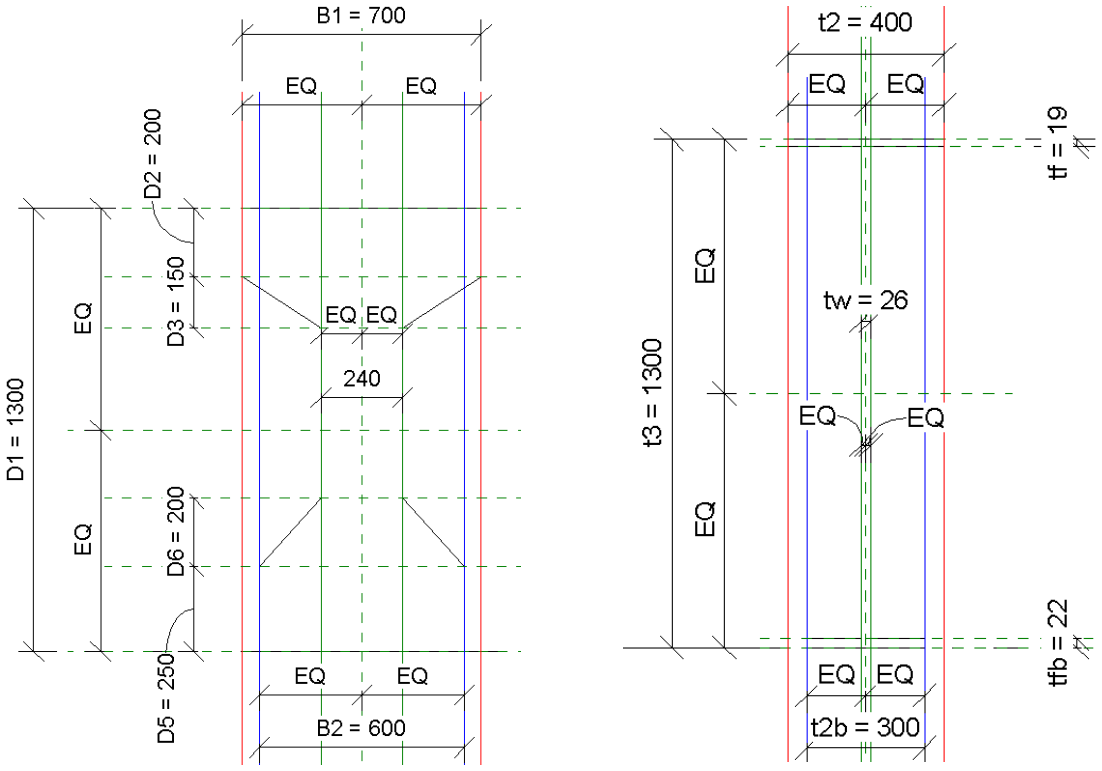
Durante esse processo, houve uma preferência pela criação de parâmetros do tipo "parâmetro de tipo", em vez de parâmetros de instância, para garantir que todas as instâncias de um mesmo tipo de família compartilhem os mesmos valores. Essa abordagem favorece a padronização e o controle global das propriedades dos elementos, além de facilitar ajustes e manutenções futuras, já que qualquer alteração no tipo impacta automaticamente em todas as suas instâncias derivadas.

Adicionalmente, a preferência pela utilização de parâmetros de tipo também se justificou pela própria estruturação do grafo desenvolvido no Dynamo, em que cada componente da ponte, como travessas de extremidade ou intermediárias, transversinas de apoio ou intermediárias, entre outros, foi tratado como um tipo distinto de tipo de família. Essa abordagem tornou o código mais claro e modular, facilitando a associação dos parâmetros aos elementos no Revit, garantindo consistência entre as instâncias e simplificando a manipulação dos elementos no fluxo proposto.

As famílias desenvolvidas para as longarinas seguiram o *template* de quadro estrutural (ou *structural framing*), cujo parâmetro de comprimento já é associado. Isso

possibilita que o comprimento do objeto, ao ser instanciado, seja definido pelo usuário através da definição de dois pontos em projeto. Nesse elemento, utilizou-se a ferramenta de modelagem do tipo extrusão, sendo fornecida a seção transversal parametrizada ao longo do parâmetro comprimento. Por conta disso, a modelagem das vigas foi realizada conforme a definição de curvas no Dynamo e posterior vinculação dessas curvas com objetos no Revit. Os parâmetros lançados para esses elementos seguiram as convenções de nomes já estabelecidas anteriormente, conforme ilustrado na Figura 3.27, enquanto na Figura 3.28 ilustra-se o parâmetro nativo do *template* de famílias do tipo quadro estrutural.

Figura 3.27 – Parâmetros associados em elevação à extrusão de famílias desenvolvidas das longarinas (a) pré-moldadas de concreto e (b) metálicas



Em contraste aos elementos de viga a serem lançados, tem-se elementos que terão parâmetros que controlam toda sua geometria, sendo necessário apenas posicioná-lo através do Dynamo. Como exemplos desses elementos, na Figura 3.29 ilustram-se os parâmetros associados à modelagem dos encontros, enquanto os parâmetros para os aparelhos de apoio constam na Figura 3.30.

Figura 3.28 – Parâmetro nativo associado ao comprimento de vigas em geral que controla o comprimento da extrusão através do *template* de quadro estrutural

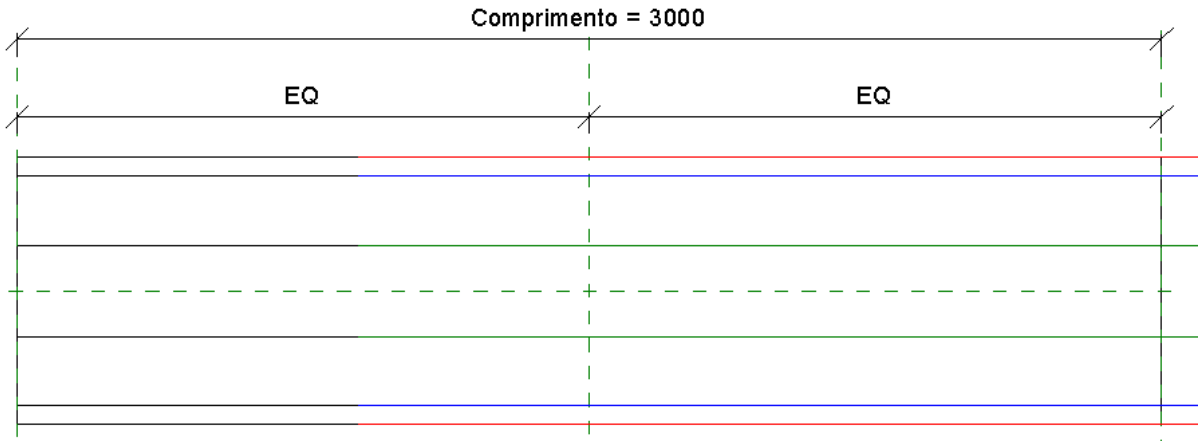
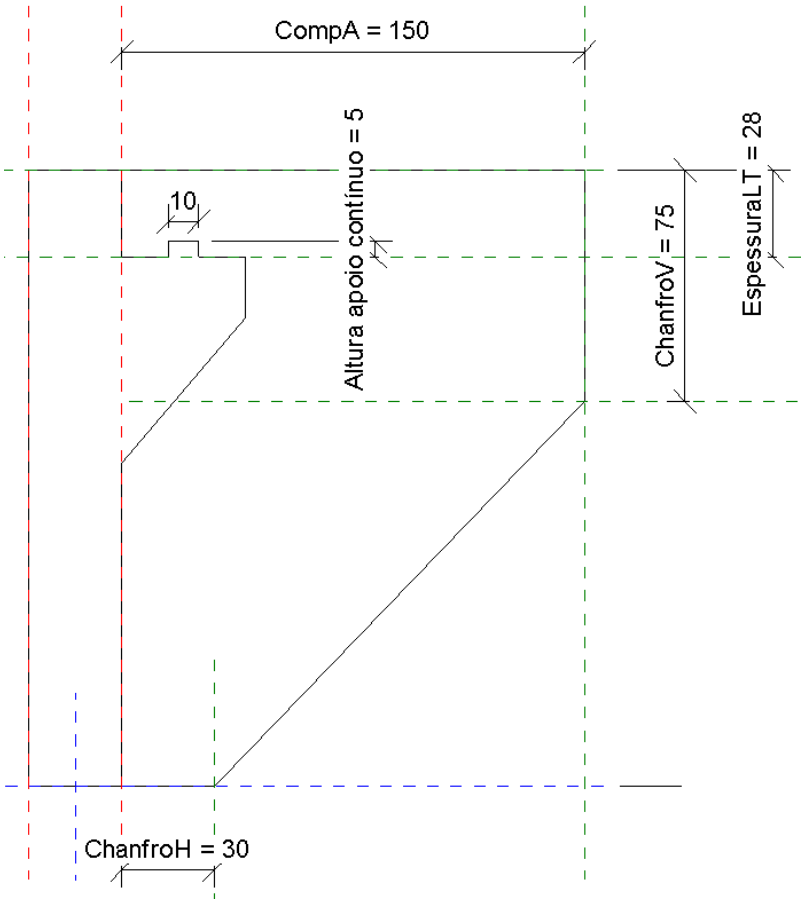
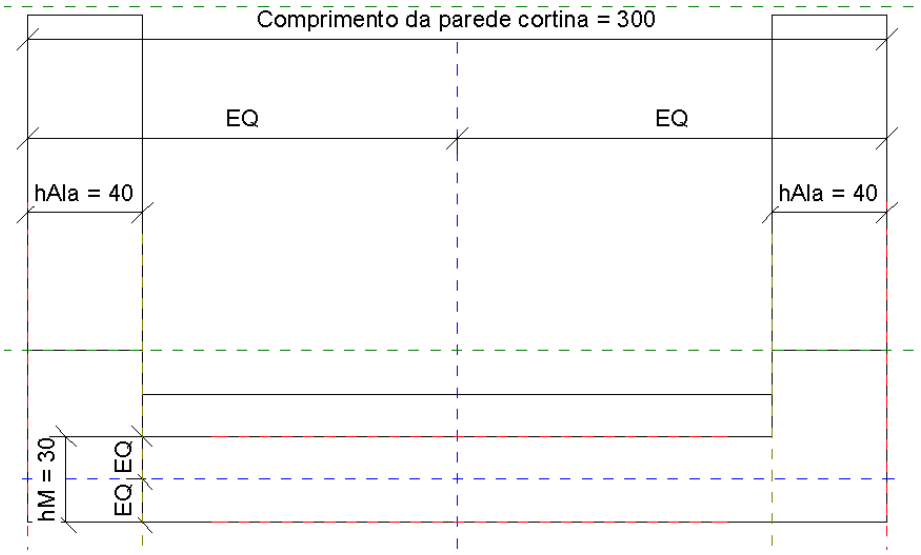


Figura 3.29 – Parâmetros associados em elevação à extrusão de família desenvolvida dos encontros (a) em elevação e (b) planta

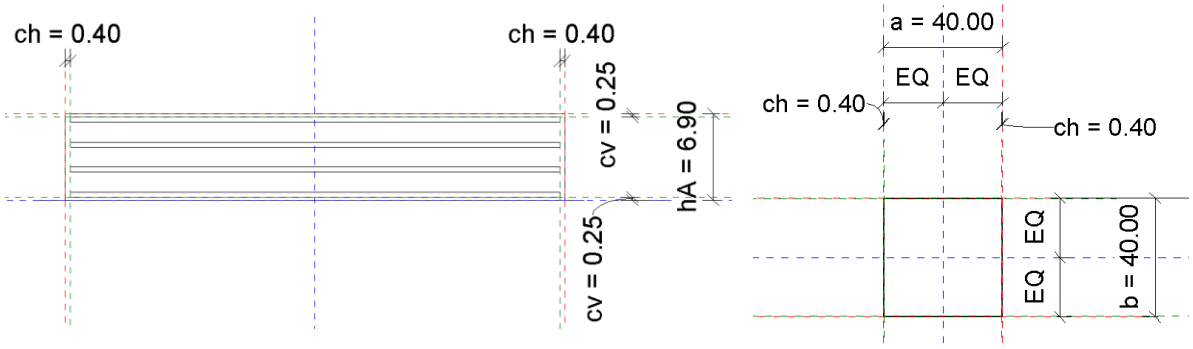


(a)



(b)

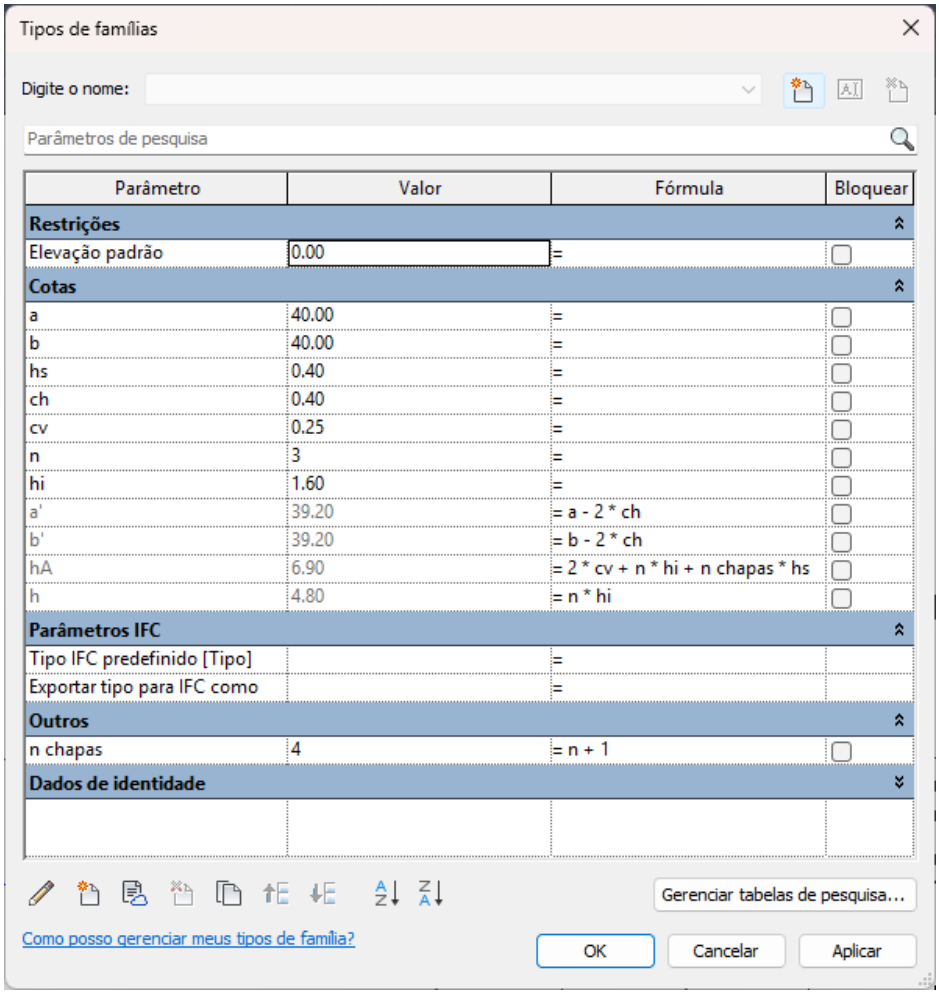
Figura 3.30 – Parâmetros associados em elevação à extrusão de família desenvolvida do aparelho de apoio (a) em elevação e (b) planta



(a) **(b)**

Além dos parâmetros de cálculo já discutidos no item 3.1, relativos ao aparelho de apoio, foram adicionados parâmetros específicos para o controle geométrico de seus componentes, em especial a espessura das camadas de Neoprene e das chapas de aço. Esses parâmetros foram inseridos como variáveis controláveis na família do Revit, permitindo que cada componente do aparelho seja modelado com fidelidade às condições de projeto. O parâmetro de altura total do aparelho de apoio, por sua vez, foi configurado como um parâmetro dependente, sendo calculado automaticamente com base nas espessuras definidas para os demais componentes, conforme ilustrado na Figura 3.31. Vale destacar que, além das famílias desenvolvidas, foram também utilizadas famílias nativas do Revit, como ocorreu com a modelagem das lajes e do pavimento.

Figura 3.31 – Lista de parâmetros e regras paramétricas criados para governarem o comportamento da família de aparelho de apoio



3.3.2 Estrutura do grafo desenvolvido

Conforme apresentado no item 3, a programação visual realizada no Dynamo se dá a partir de nós que, ao serem alimentados com dados de *input,* realizam uma operação específica e fornecem um *output*, sendo conectados por fios. Adicionalmente aos nós nativos do programa, foi necessário o uso de *scripts* em Python para realizar operações não acessíveis diretamente pelo Dynamo e do pacote de nós customizados "*Data*|*Shapes*" para a exibição de interface gráfica para interação com o usuário.

O topo do eixo da longarina da esquerda, em seção transversal, foi considerado como a origem do sistema de coordenadas. A partir dessa origem do sistema de coordenadas que foram realizados os cálculos tanto para a modelagem das curvas quanto posicionamento de elementos sem comprimento definido *a priori*.

Assim como na comunicação via API com o CSiBridge, os valores dos parâmetros de cada tipologia foram importados por meio de planilha, sendo a mesma utilizada para alimentar os modelos numéricos.

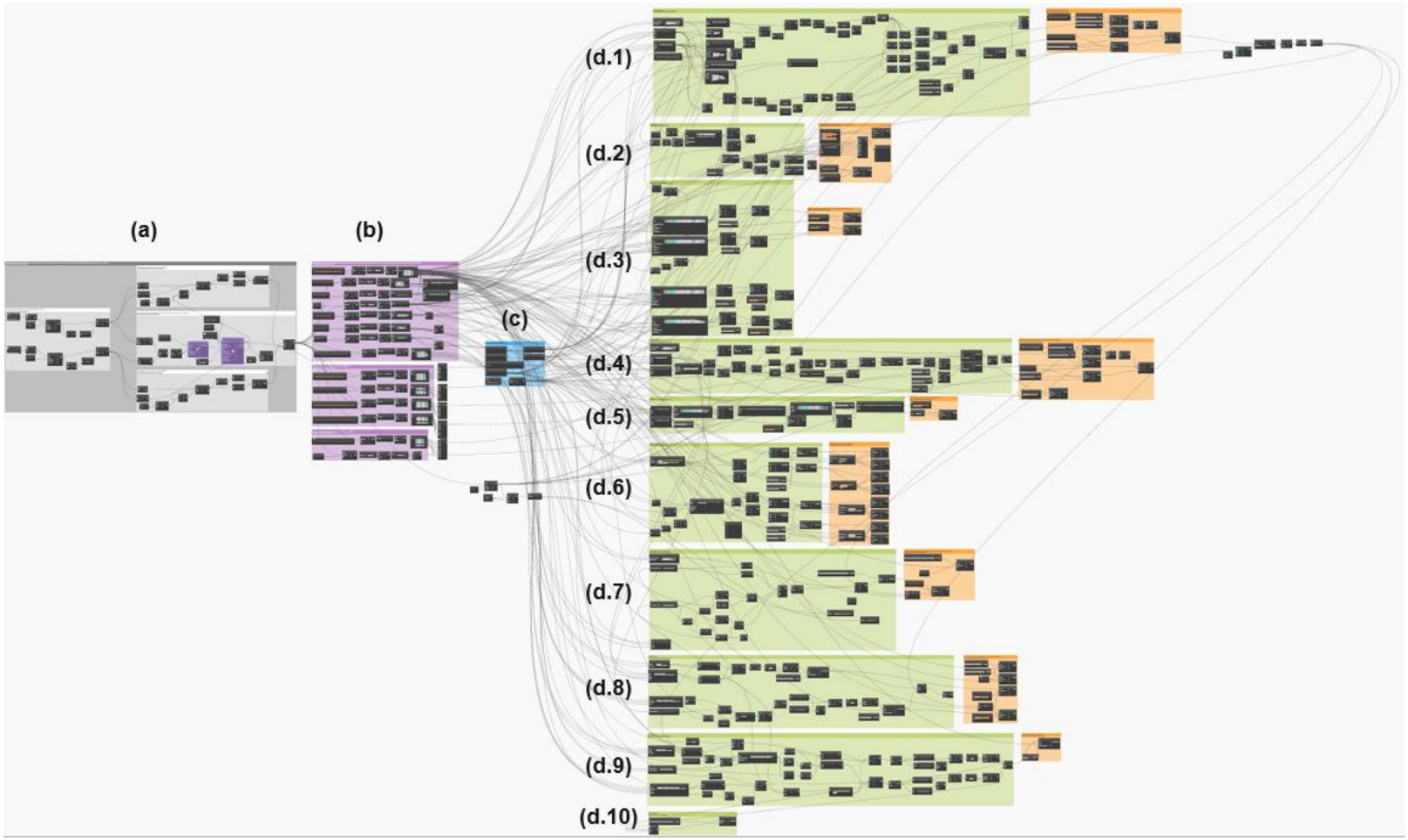
Conforme apresentado na Figura 3.32-(a), o grafo desenvolvido tem início com a leitura dos dados e sua organização em dicionários no Dynamo, que funcionam como estruturas chave-valor, facilitando o manuseio das informações. Em seguida, como ilustrado na Figura 3.33, é solicitado um *input* do usuário para definição da tipologia específica a ser modelada. A partir dessa entrada, é selecionado o dicionário correspondente à tipologia e são aplicadas estruturas condicionais para lidar com as particularidades dos modelos do tipo PPC ou MAC.

A estrutura revelada na Figura 3.32-(b) realiza a obtenção dos valores dos parâmetros no dicionário respectivo à tipologia entrada pelo usuário. A partir deste ponto que são definidas as geometrias da ponte como um todo e de cada elemento estrutural. O grupo de nós da estrutura evidenciada em (c) apresenta parâmetros de projeto adicionais para acrescentar especificidades ao modelo, sendo eles:

- Número de vãos;
- Distância entre o topo das transversinas e as longarinas;
- Distância entre as extremidades das longarinas com o eixo da transversina;
- Espessura da junta.

A partir dessas entradas, é iniciado o processo de posicionamento dos elementos, conforme apresentado na Figura 3.32-(d). Grande parte desse processo envolveu a manipulação dos dados e a criação de curvas para os elementos. Finalizada essa parte, há o posicionamento dos elementos, bem como a associação das curvas geradas a elementos do Revit. Posteriormente, são atribuídos valores aos parâmetros de família relativos aos elementos e controle de elevação (destacado na cor laranja), que são parâmetros de instância, para correto posicionamento. O modelo gerado na Figura 3.34 ilustra a aplicação do grafo para a modelagem da tipologia PPC-A-20-P.

Figura 3.32 – Grafo desenvolvido na plataforma Dynamo para a modelagem paramétrica de pontes a partir dos mesmos parâmetros considerados para os modelos de cálculo no ambiente do Revit



Legenda: em (d), tem-se a criação de criação e curvas ou a obtenção das coordenadas para o posicionamento dos elementos 1 – Longarinas, 2 – Encontros, 3 – Lajes de transição e pavimento sobre elas, 4 – Transversinas, 5 – Lajes do tabuleiro e pavimento, 6 – Barreiras, 7 – Aparelhos de apoio, 8 – Travessas, 9 – Pilares e 10 - Fundação.

Figura 3.33 – Entrada de dados na forma de formulário filtrado com base na lista de tipologias especificadas na Tabela 3.3 e Tabela 3.4

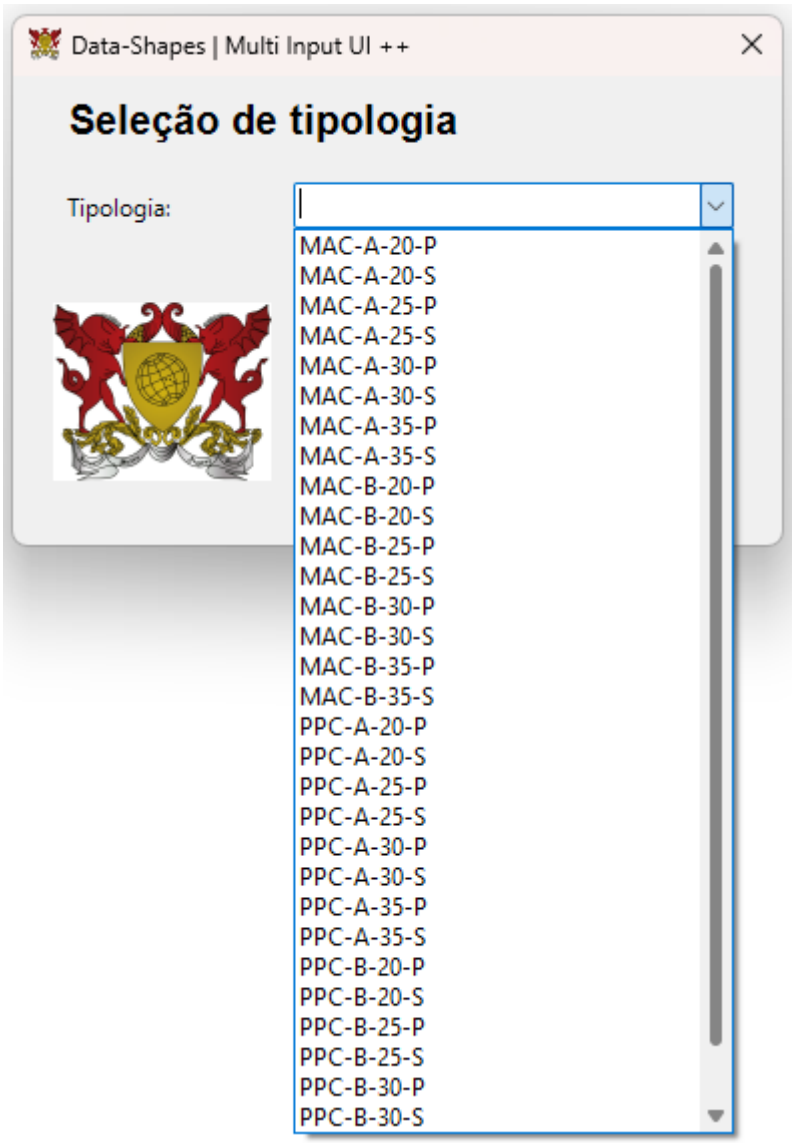
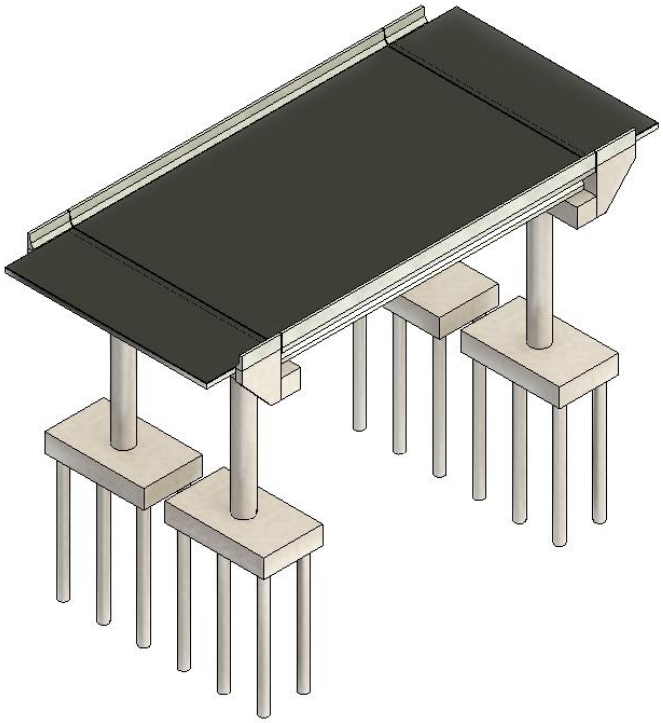
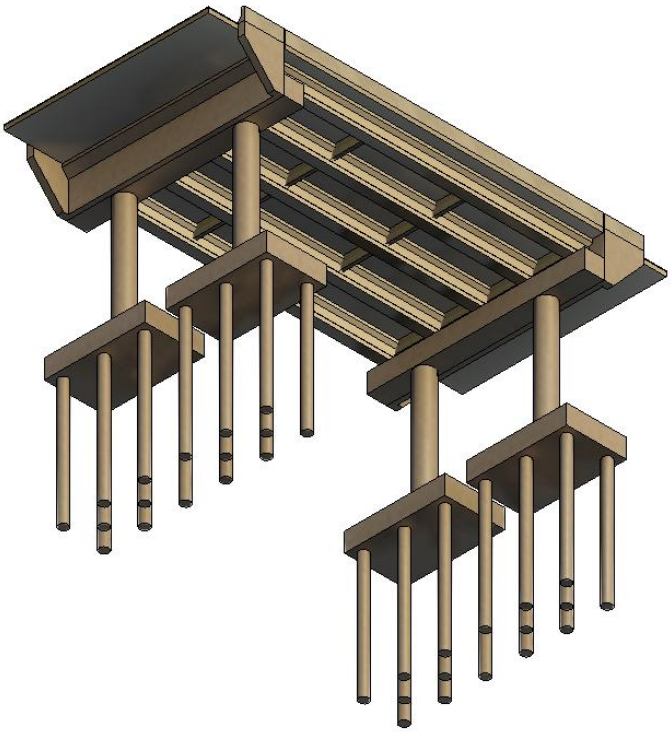


Figura 3.34 – Modelo paramétrico gerado a partir do grafo desenvolvido considerando a tipologia PPC-A-20-P em vista isométrico (superior) e (inferior)



(a)



(b)

4. RESULTADOS E DISCUSSÕES

4.1 Modelos numéricos gerados a partir dos códigos desenvolvidos para a conexão API com o CSiBridge utilizando Python

A fim de exemplificar a automatização da modelagem realizada no CSiBridge por meio da comunicação via API com a linguagem Python, são apresentados os modelos gerados pelos códigos desenvolvidos correspondentes à tipologia PPC-B-30-S (Figura 4.1 a Figura 4.9). O detalhe para a discretização realizada na extremidade do tabuleiro da Figura 4.1 evidencia desencontro de malhas em regiões do modelo em elementos finitos, ocasionado pela necessidade de verificação de esforço cortante da laje próximo à alma e da mesa superior das longarinas. No entanto, trata-se de uma perturbação local e de pequena extensão, cujos efeitos tendem a ser regularizados nas seções seguintes. Para validar o modelo, os esforços nas longarinas e nas lajes foram comparados com os de um modelo sem a discretização refinada, não sendo observadas diferenças significativas para a análise realizada na presente pesquisa. Esse desencontro resultou nas descontinuidades observadas nos diagramas de esforços solicitantes (Figura 4.11 a Figura 4.21).

Figura 4.1 – Modelo numérico da tipologia PPC-B-30-S gerado para análise de superestrutura

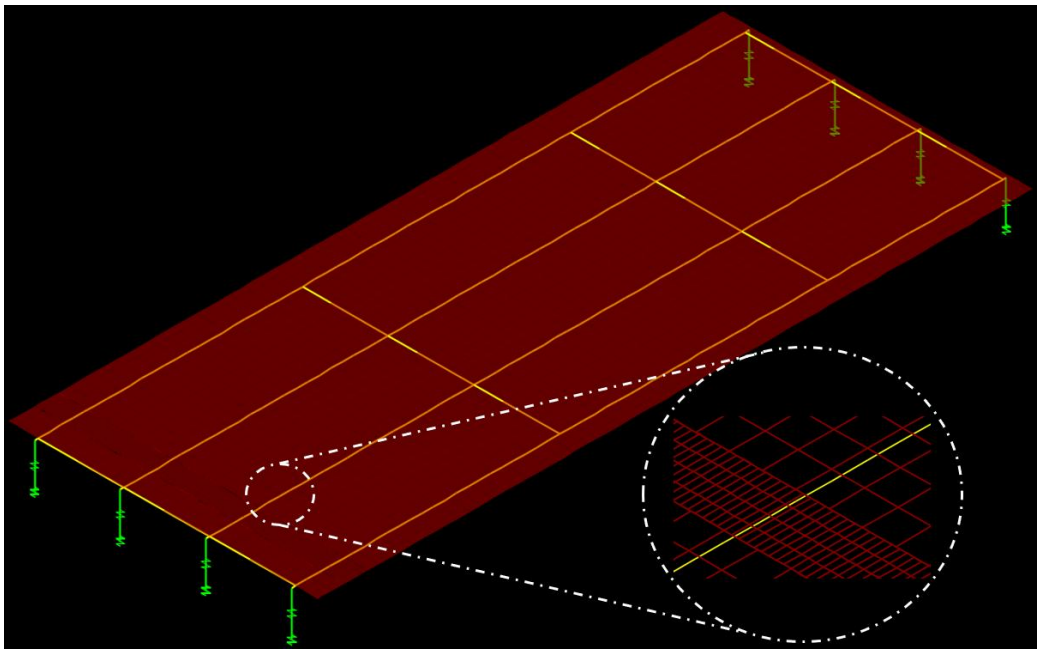


Figura 4.2 – Modelo numérico da tipologia PPC-B-30-S gerado para análise de flechas, com os dados dos cabos de protensão inseridos no modelo

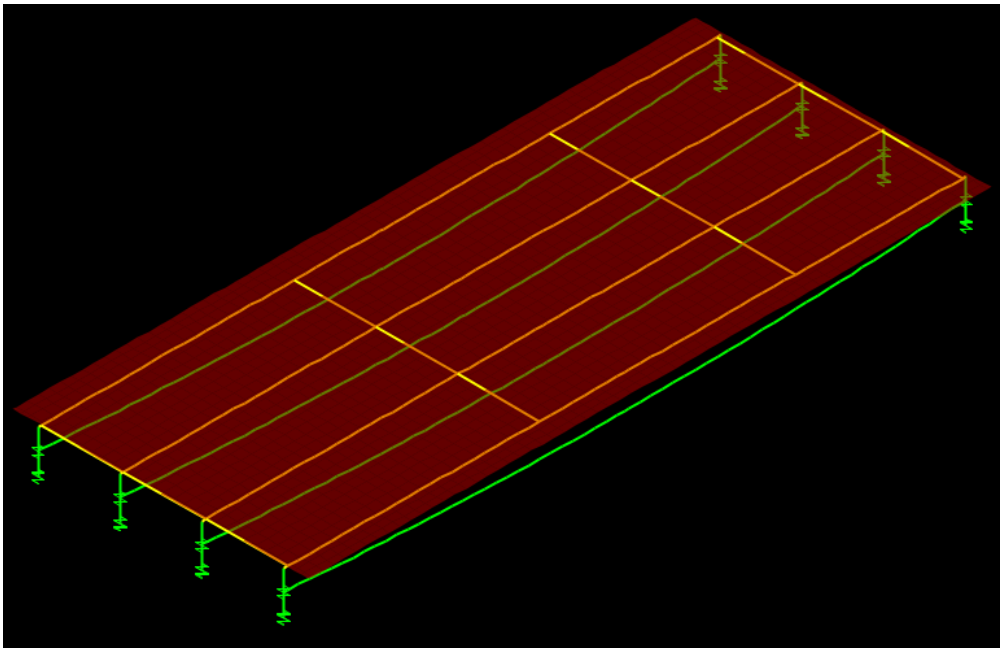


Figura 4.3 – Dados gerais, como posicionamento, força aplicada e área de aço ativo, referentes ao cabo de protensão lançado no modelo da tipologia PPC-B-30-S

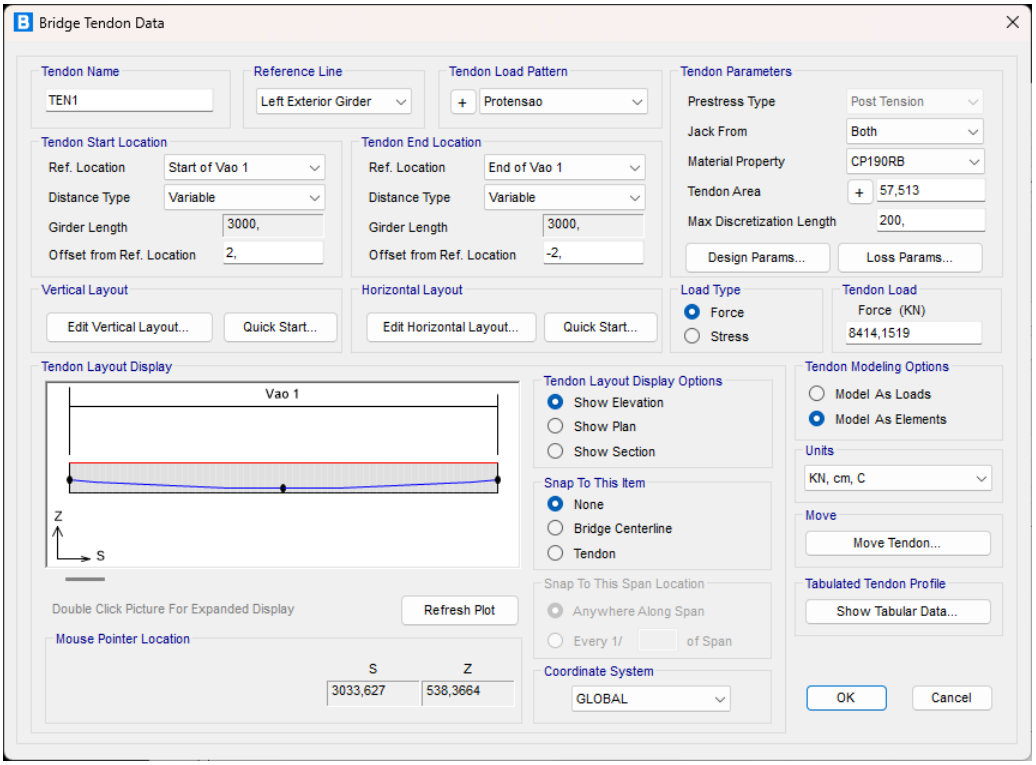


Figura 4.4 – Dados de elevação e tipo de segmento referentes ao cabo de protensão lançado no modelo da tipologia PPC-B-30-S

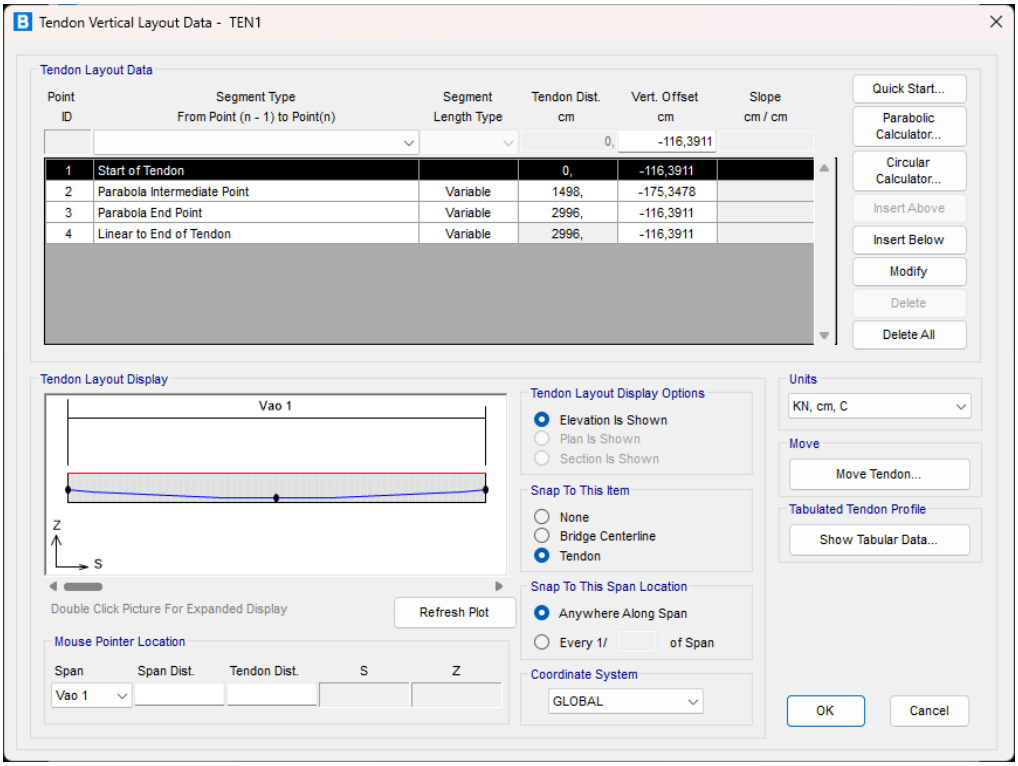


Figura 4.5 – Dados das perdas, tanto imediatas quanto diferidas no tempo, calculadas a partir da planilha, referentes ao cabo de protensão lançado no modelo da tipologia PPC-B-30-S

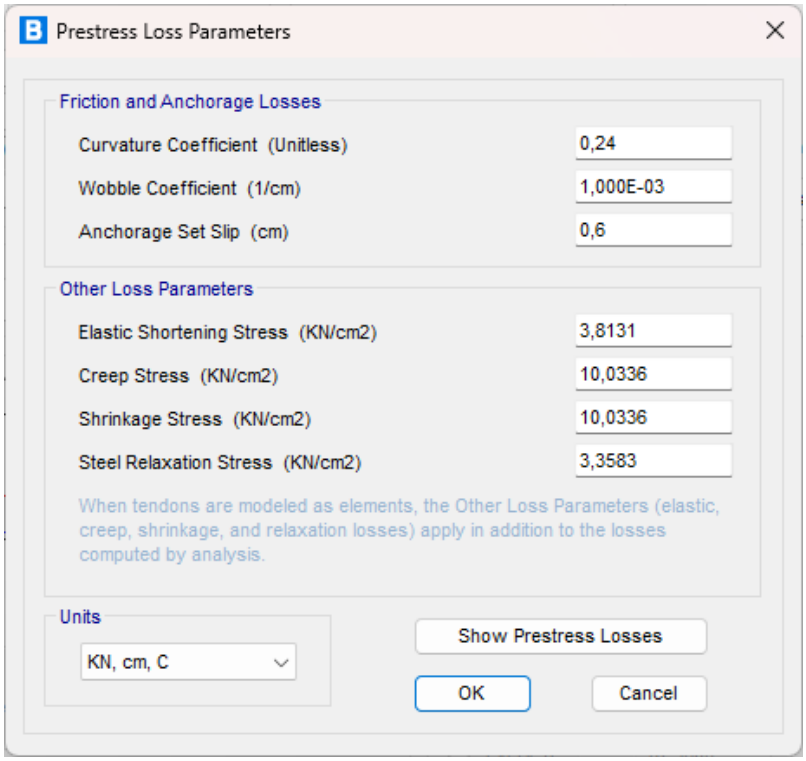


Figura 4.6 – Modelo numérico da tipologia PPC-B-30-S gerado para análise de encontros com vão único

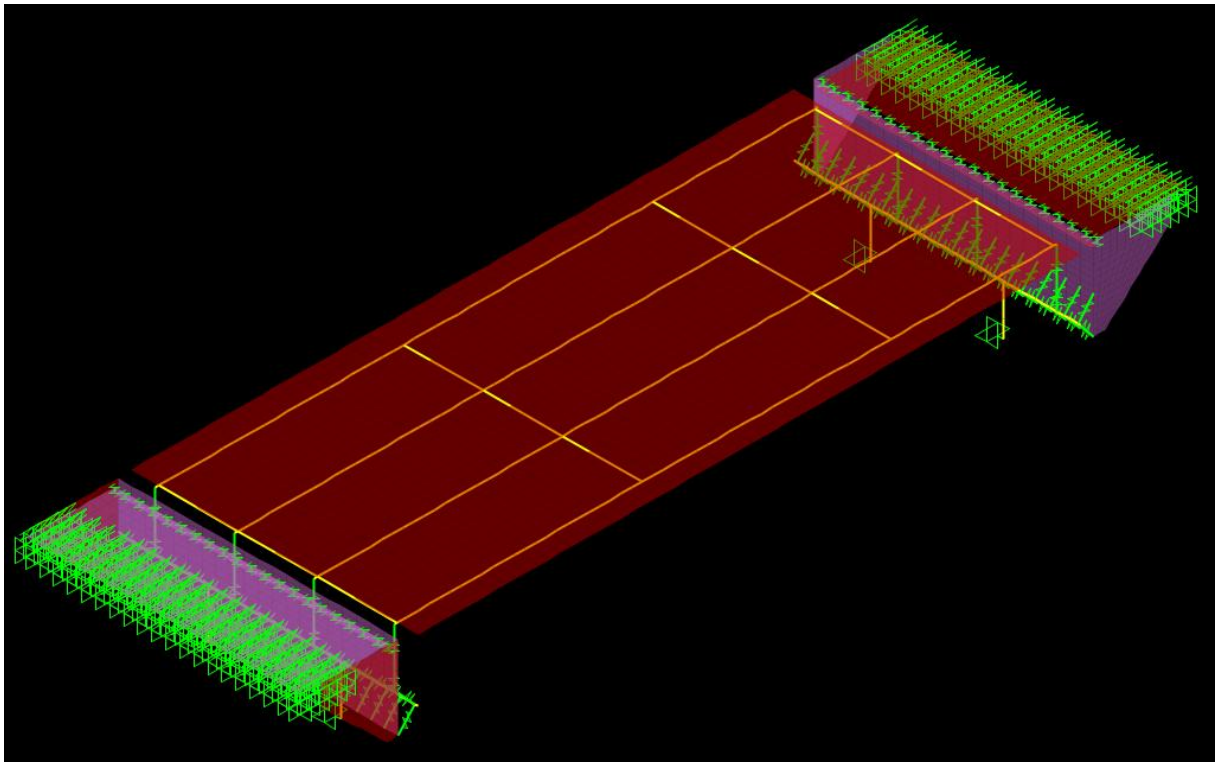


Figura 4.7 – Modelo numérico da tipologia PPC-B-30-S gerado para análise de encontros com três vãos iguais

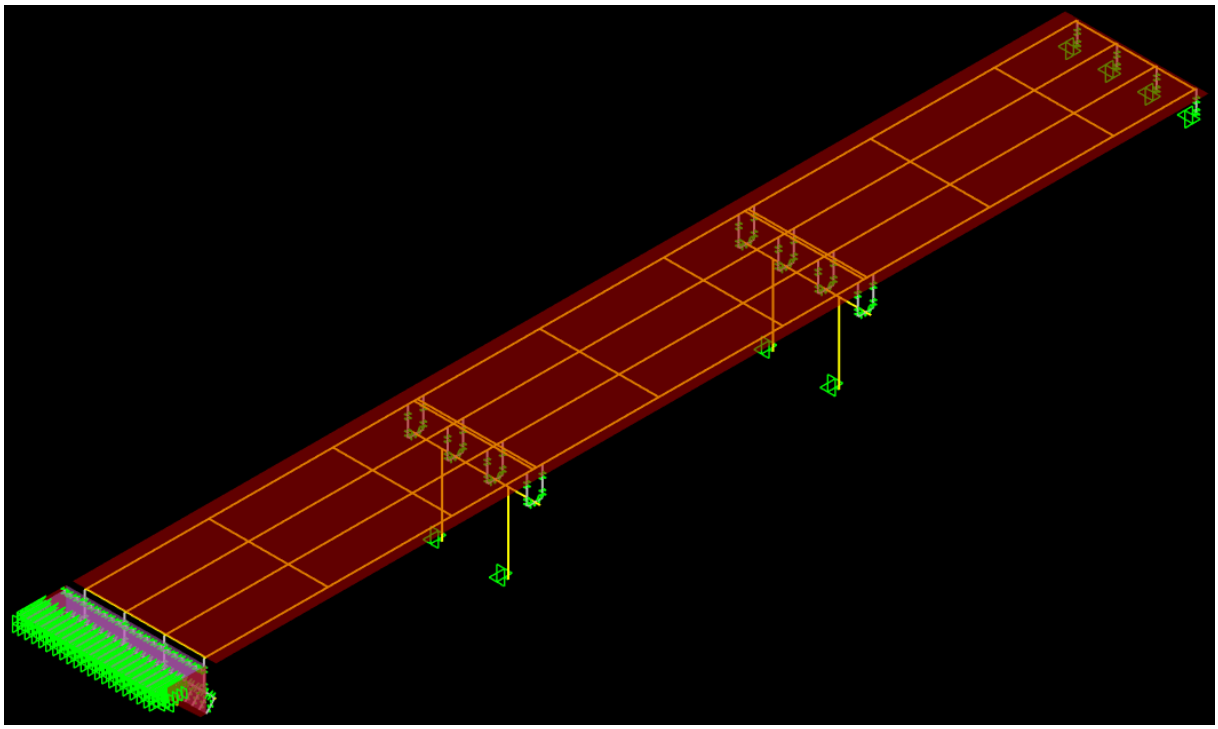


Figura 4.8 – Modelo numérico da tipologia PPC-B-30-S gerado para análise de mesoestrutura com pilares de 4 metros e vãos de 30m e 20m alternados

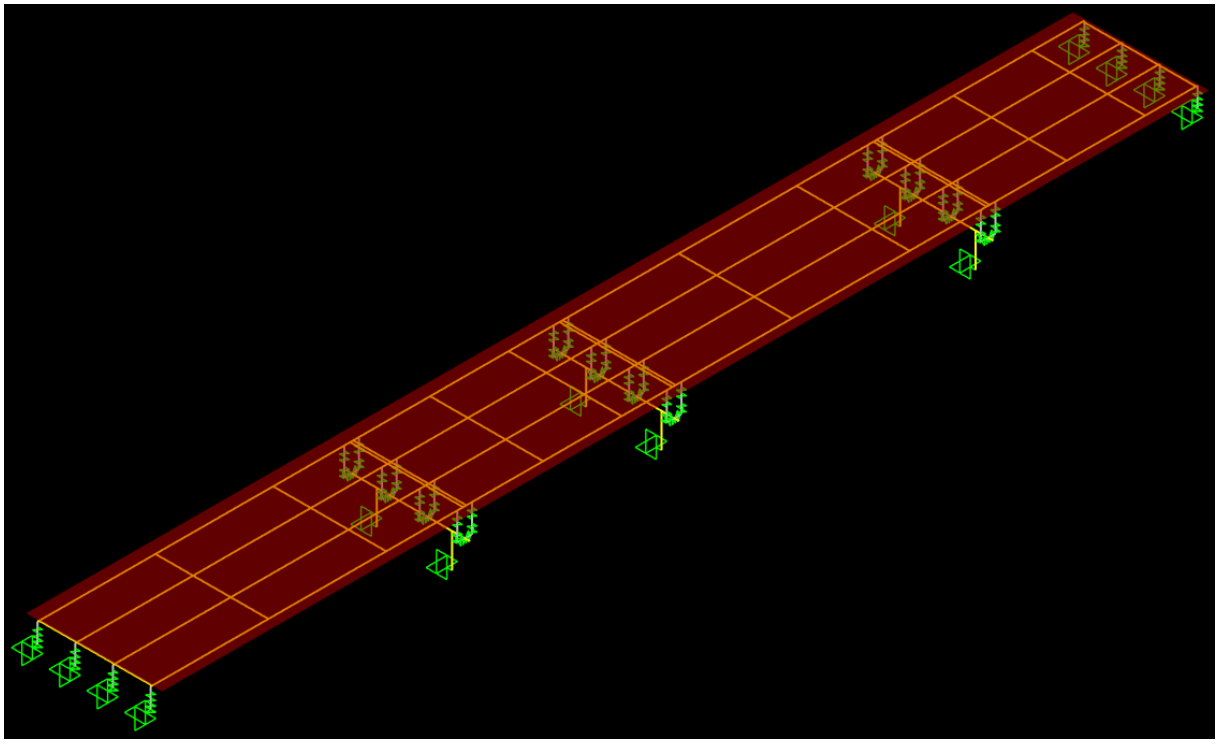
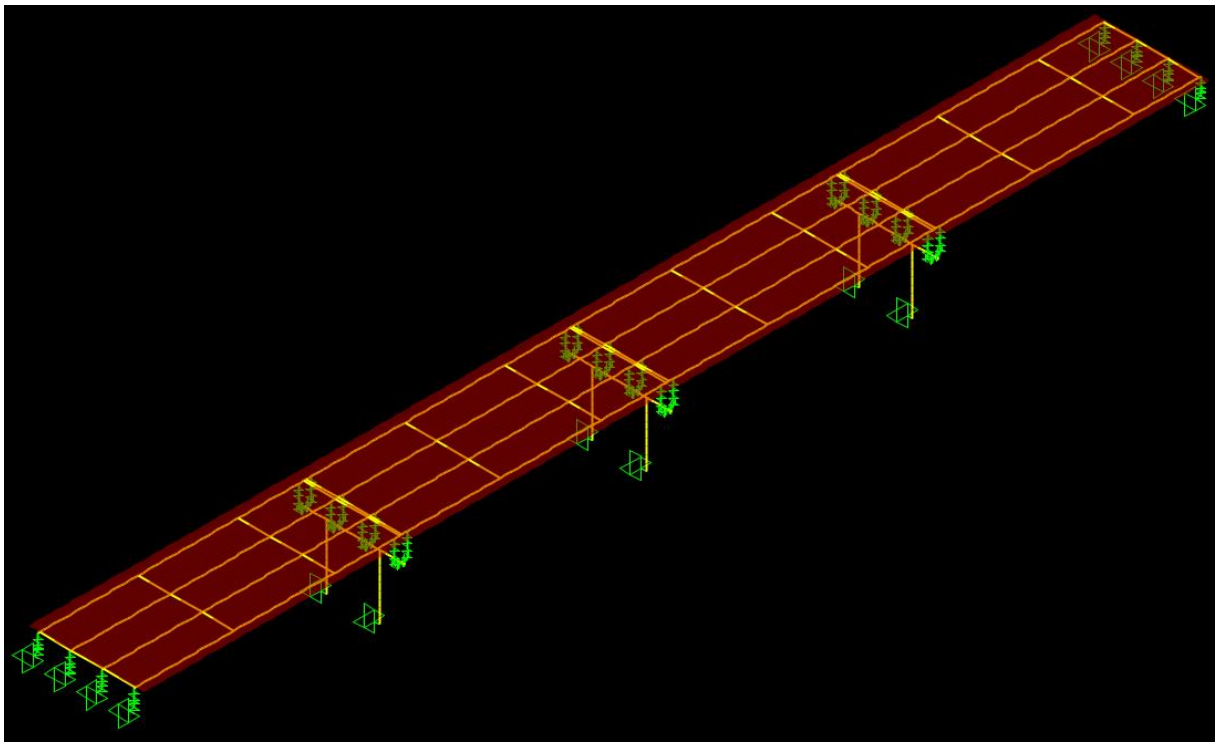


Figura 4.9 – Modelo numérico da tipologia PPC-B-30-S gerado para análise de mesoestrutura com pilares de 8 metros e vãos iguais



Ressalta-se que todos os modelos, juntamente com os resultados das respectivas análises, foram disponibilizados em um ambiente de armazenamento em

nuvem, assegurando o acesso integrado por toda a equipe do grupo de pesquisa, sem a necessidade de reprocessamento. A validação dos resultados foi conduzida pelo grupo de pesquisa por meio da consulta a dimensionamentos de projetos anteriores, de modo a assegurar que os valores obtidos se mantivessem dentro da ordem de grandeza.

4.2 Fluxos de extração de esforços para dimensionamento

4.2.1 Longarinas pré-moldadas e protendidas

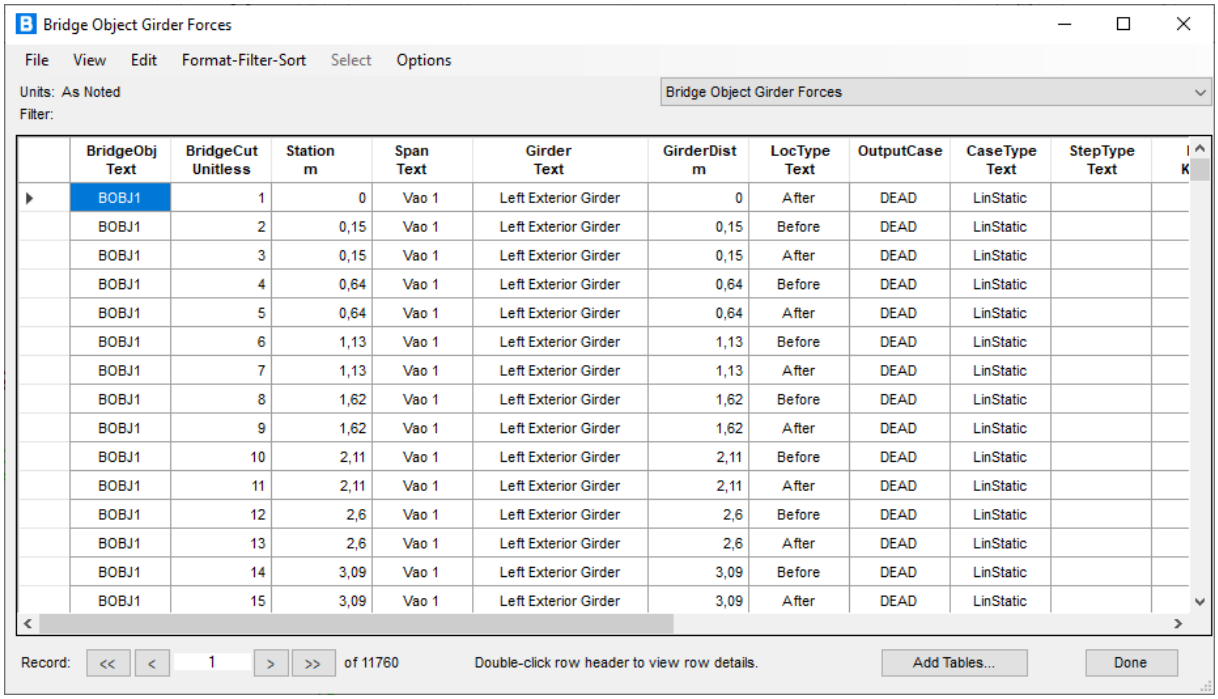
Para a análise dos esforços das longarinas pré-moldadas, foram adotadas as seções correspondentes a L/8, L/4, 3L/8 e L/2, sendo *L* o comprimento das longarinas. Como, em geral, a discretização da malha de elementos finitos não coincidia exatamente com essas posições, foi implementada uma rotina no código capaz de estimar os esforços por interpolação, utilizando os resultados do nó imediatamente anterior e posterior a cada seção definida, no caso de momentos fletores. Para esforços cortantes e de torção, foi considerada uma interpolação para o início do intervalo considerado, onde os esforços são mais acentuados.

Conforme metodologia apresentada no item 3.2.2, as combinações de ações para obtenção dos esforços de cálculo são realizadas por meio de uma planilha, na qual são inseridas as solicitações correspondentes a cada caso de carga. Nesse sentido, foram extraídos os seguintes esforços referentes

- Peso-próprio da estrutura;
- Cargas complementares (defensas, pavimentação e recapeamento);
- Envoltórias de máximos e mínimos associados às cargas móveis.

O CSiBridge disponibiliza os resultados da análise por meio de tabelas; como exemplo, destaca-se a exportação dos esforços em todas as longarinas do modelo, organizados por caso de carga e pelas respectivas seções, conforme apresentado na tabela “*Bridge Object Girder Forces*”, ilustrada na Figura 4.10. Assim, o código envolveu a filtragem dos dados dessa tabela para a extração dos resultados. Vale ressaltar que, no código, foi adotado o critério de extração do esforço solicitante mais crítico em cada seção, o que implica que os resultados podem se referir a diferentes longarinas.

Figura 4.10 – Tabela gerada pelo CSiBridge após a execução da análise para extração dos esforços internos atuantes nas longarinas



Bridge Object Girder Forces

File View Edit Format-Filter-Sort Select Options

Units: As Noted

Filter:

Bridge Object Girder Forces

	BridgeObj Text	BridgeCut Unitless	Station m	Span Text	Girder Text	GirderDist m	LocType Text	OutputCase	CaseType Text	StepType Text
▶	BOBJ1	1	0	Vao 1	Left Exterior Girder	0	After	DEAD	LinStatic	
	BOBJ1	2	0,15	Vao 1	Left Exterior Girder	0,15	Before	DEAD	LinStatic	
	BOBJ1	3	0,15	Vao 1	Left Exterior Girder	0,15	After	DEAD	LinStatic	
	BOBJ1	4	0,64	Vao 1	Left Exterior Girder	0,64	Before	DEAD	LinStatic	
	BOBJ1	5	0,64	Vao 1	Left Exterior Girder	0,64	After	DEAD	LinStatic	
	BOBJ1	6	1,13	Vao 1	Left Exterior Girder	1,13	Before	DEAD	LinStatic	
	BOBJ1	7	1,13	Vao 1	Left Exterior Girder	1,13	After	DEAD	LinStatic	
	BOBJ1	8	1,62	Vao 1	Left Exterior Girder	1,62	Before	DEAD	LinStatic	
	BOBJ1	9	1,62	Vao 1	Left Exterior Girder	1,62	After	DEAD	LinStatic	
	BOBJ1	10	2,11	Vao 1	Left Exterior Girder	2,11	Before	DEAD	LinStatic	
	BOBJ1	11	2,11	Vao 1	Left Exterior Girder	2,11	After	DEAD	LinStatic	
	BOBJ1	12	2,6	Vao 1	Left Exterior Girder	2,6	Before	DEAD	LinStatic	
	BOBJ1	13	2,6	Vao 1	Left Exterior Girder	2,6	After	DEAD	LinStatic	
	BOBJ1	14	3,09	Vao 1	Left Exterior Girder	3,09	Before	DEAD	LinStatic	
	BOBJ1	15	3,09	Vao 1	Left Exterior Girder	3,09	After	DEAD	LinStatic	

Record: << < 1 > >> of 11760 Double-click row header to view row details. Add Tables... Done

Concluído o processo de modelagem da superestrutura pelo código, conforme ilustrado na Figura 3.6, e realizada a análise, foram extraídos os esforços atuantes nas longarinas. Para exemplificar todo o procedimento de dimensionamento, apresentam-se os resultados da tipologia PPC-A-20-P, cuja planilha de dimensionamento preenchida encontra-se no Anexo A.

Os diagramas de esforços solicitantes gerados no CSiBridge estão apresentados na Figura 4.11 a Figura 4.21, enquanto os esforços extraídos pelo código para alimentação da planilha constam na Tabela 4.1. Observa-se uma descontinuidade nos resultados na região onde foi aplicada a discretização refinada da malha da laje. No entanto, como as seções consideradas para a análise dos esforços estão localizadas após essa descontinuidade, além de se dispor de uma estimativa prévia dos esforços sem o refinamento, as respostas foram consideradas satisfatórias para os objetivos da análise. Destaca-se que todos os esforços extraídos consideram a seção composta da longarina com a laje do tabuleiro.

Figura 4.11 – Diagrama de momentos fletores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga de peso próprio (laje, transversinas e longarinas)

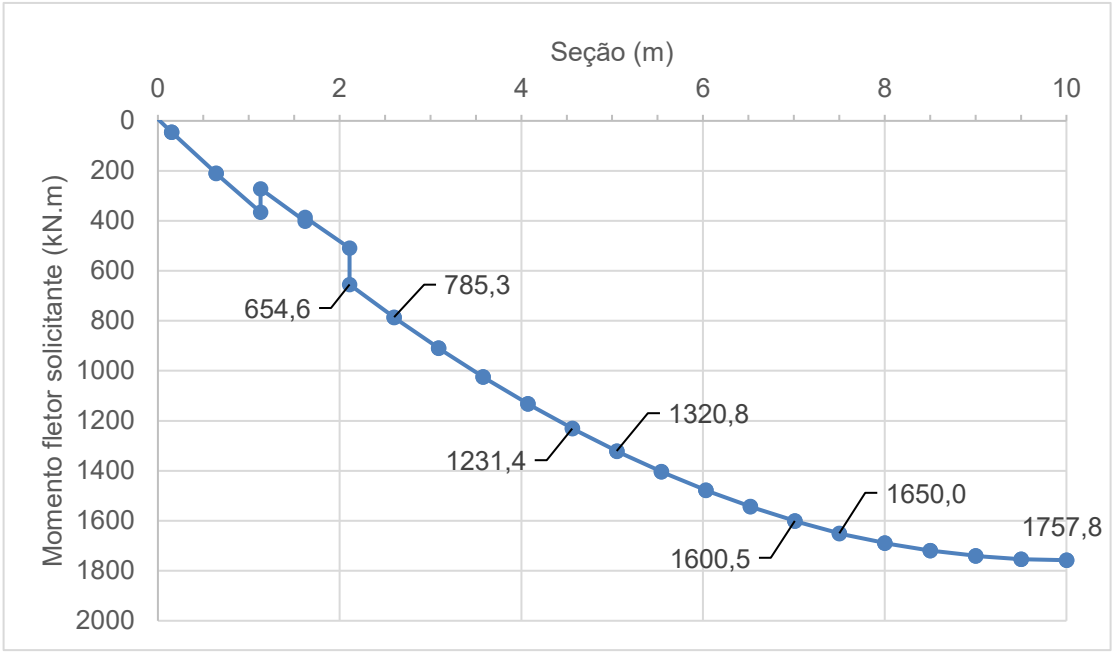


Figura 4.12 – Diagrama de momentos fletores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga de cargas complementares (defensas, pavimentação e recapeamento)

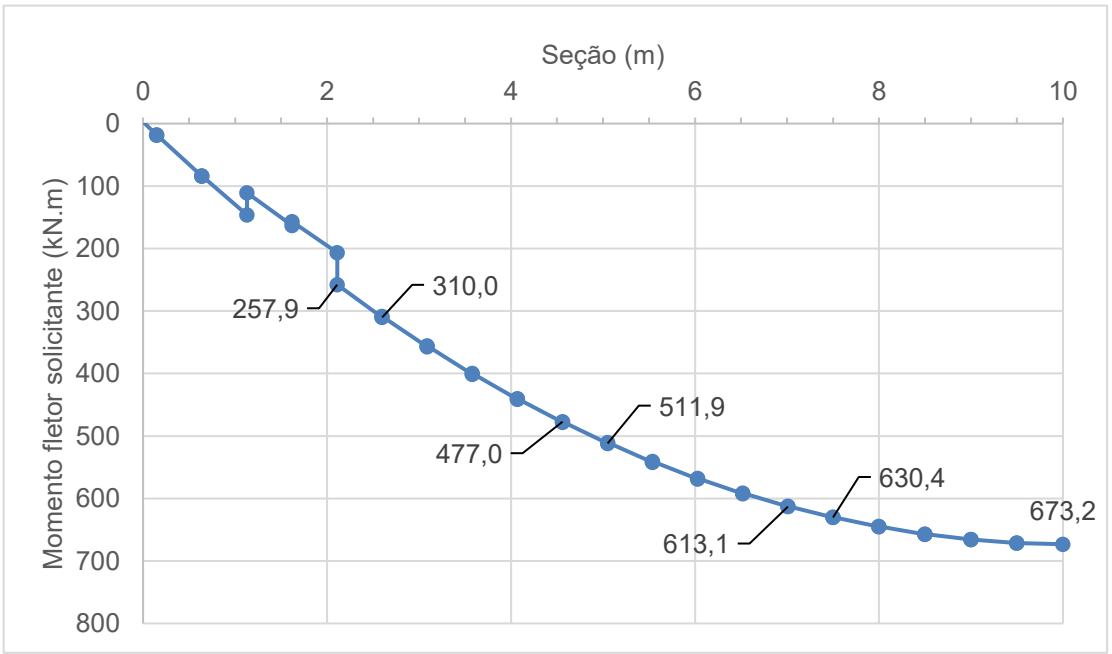


Figura 4.13 – Diagrama de momentos fletores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de máximos)

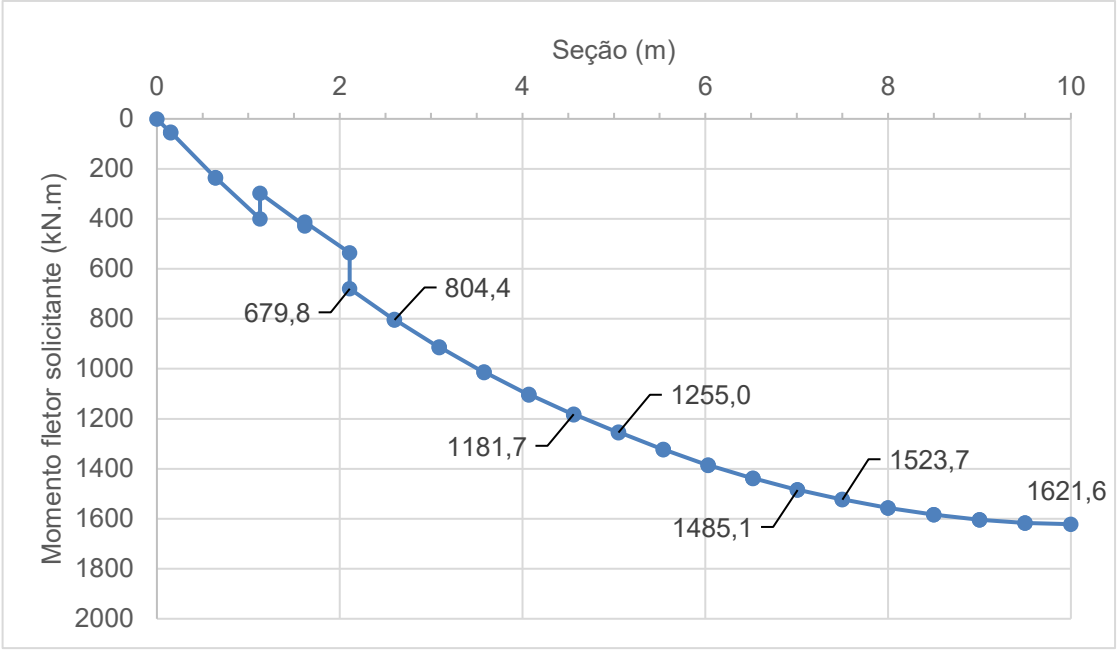


Figura 4.14 – Diagrama de momentos fletores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de mínimos)

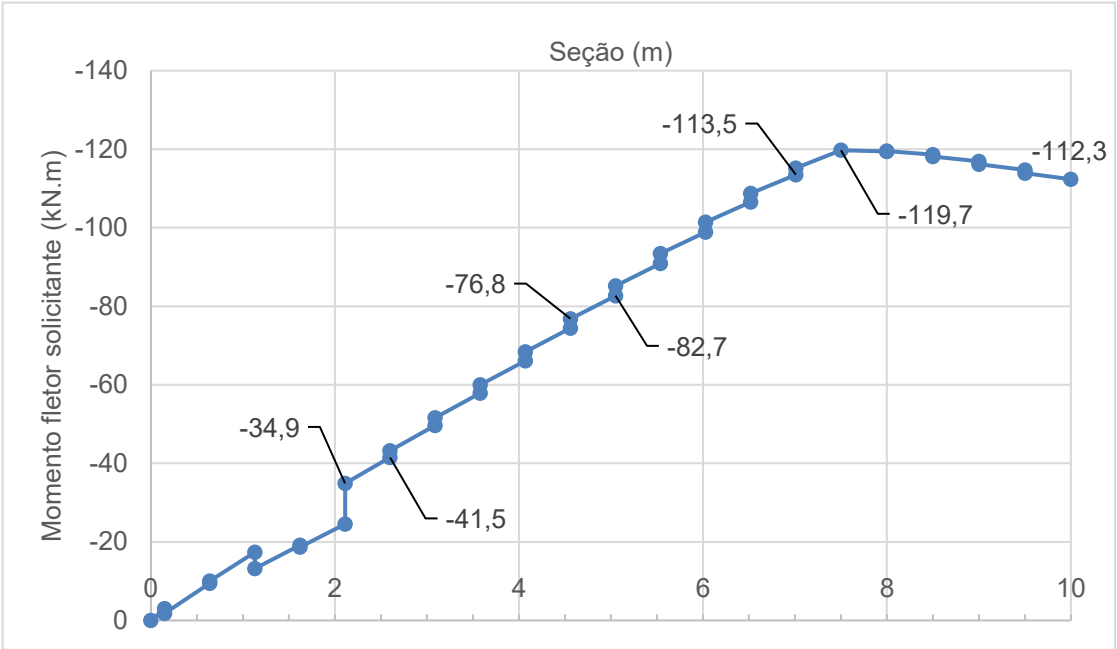


Figura 4.15 – Diagrama de esforços cortantes da longarina pré-moldada utilizado para a extração de esforços para o caso de carga de peso próprio (laje, transversinas e longarinas

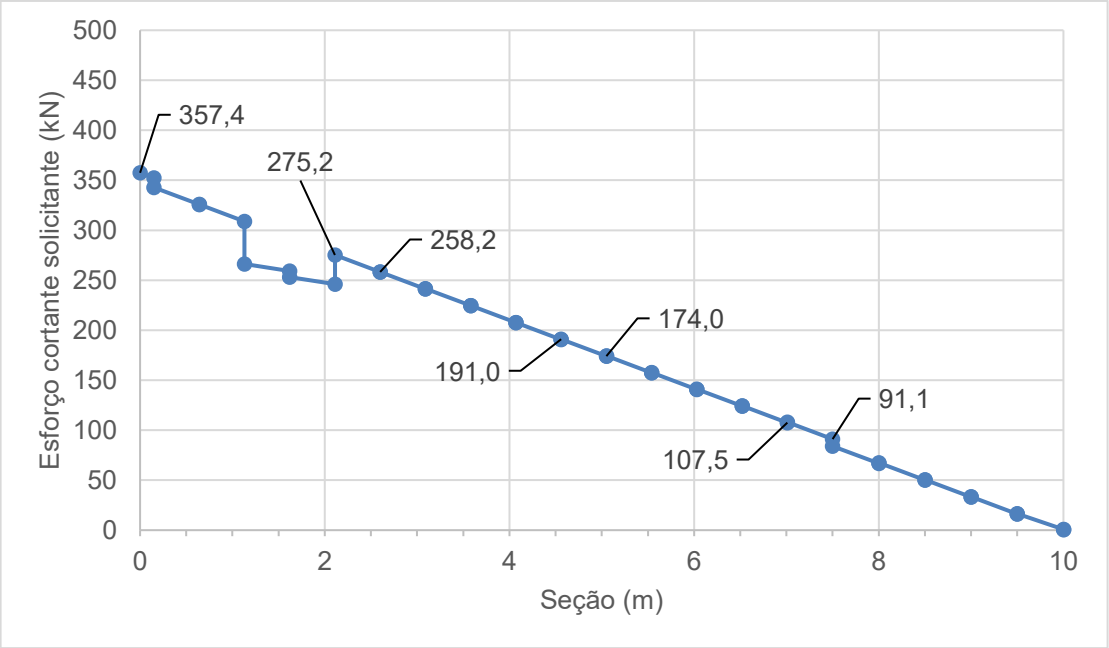


Figura 4.16 – Diagrama de esforços cortantes da longarina pré-moldada utilizado para a extração de esforços para o caso de carga de cargas complementares (defensas, pavimentação e recapeamento)

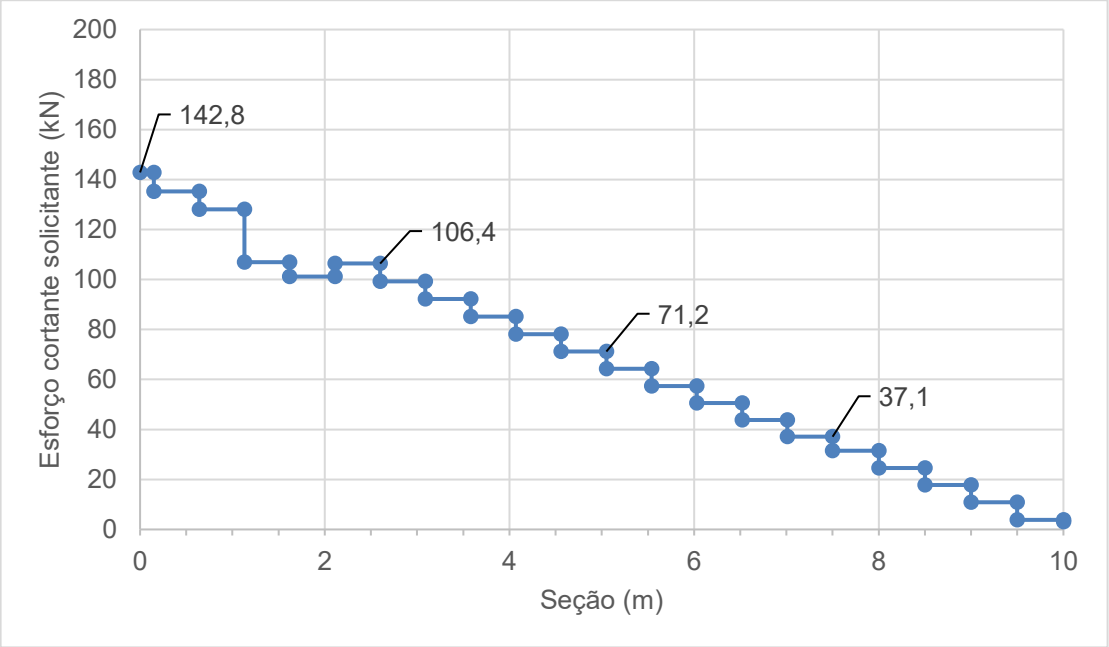


Figura 4.17 – Diagrama de esforços cortantes da longarina pré-moldada utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de máximos)

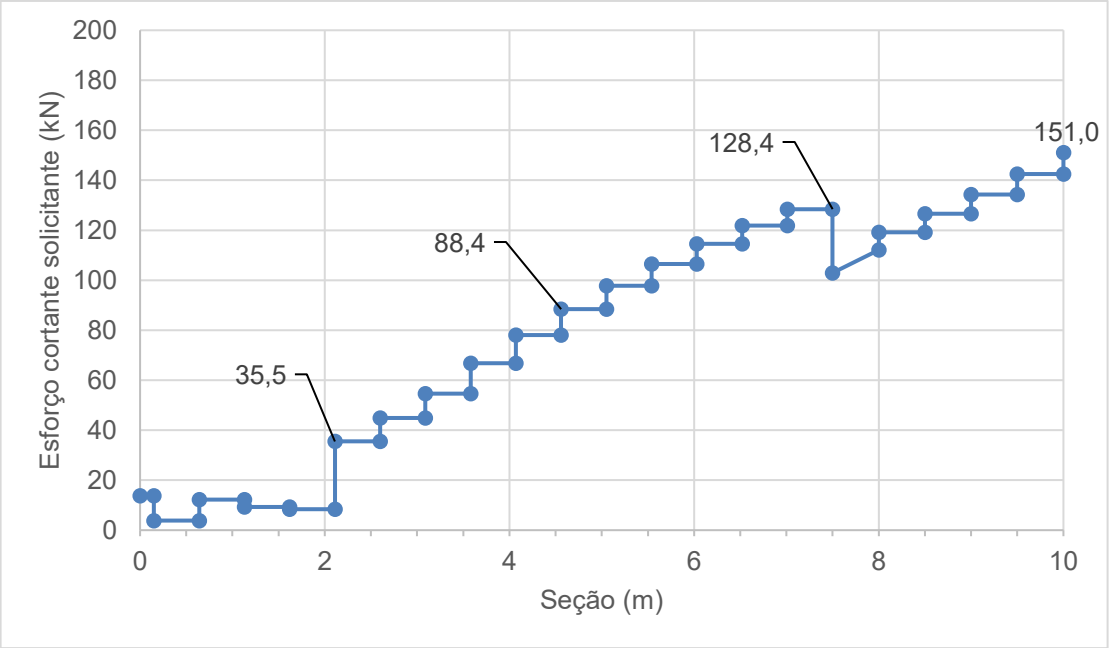


Figura 4.18 – Diagrama de esforços cortantes da longarina pré-moldada utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de mínimos)

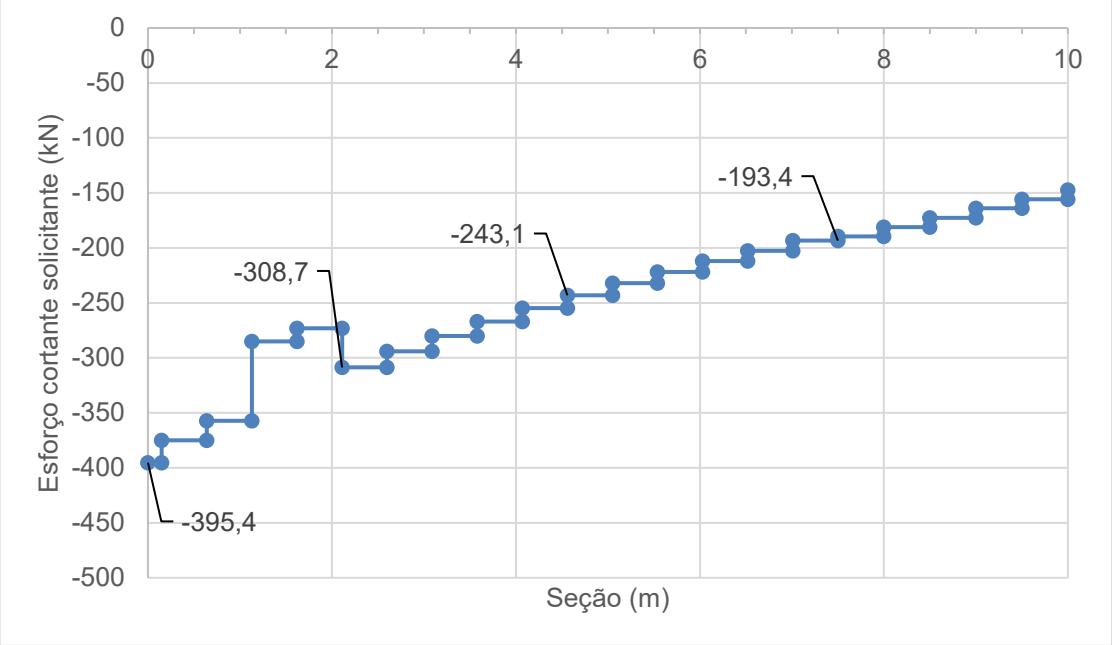


Figura 4.19 – Diagrama de esforços torçores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga de peso próprio (laje, transversinas e longarinas)

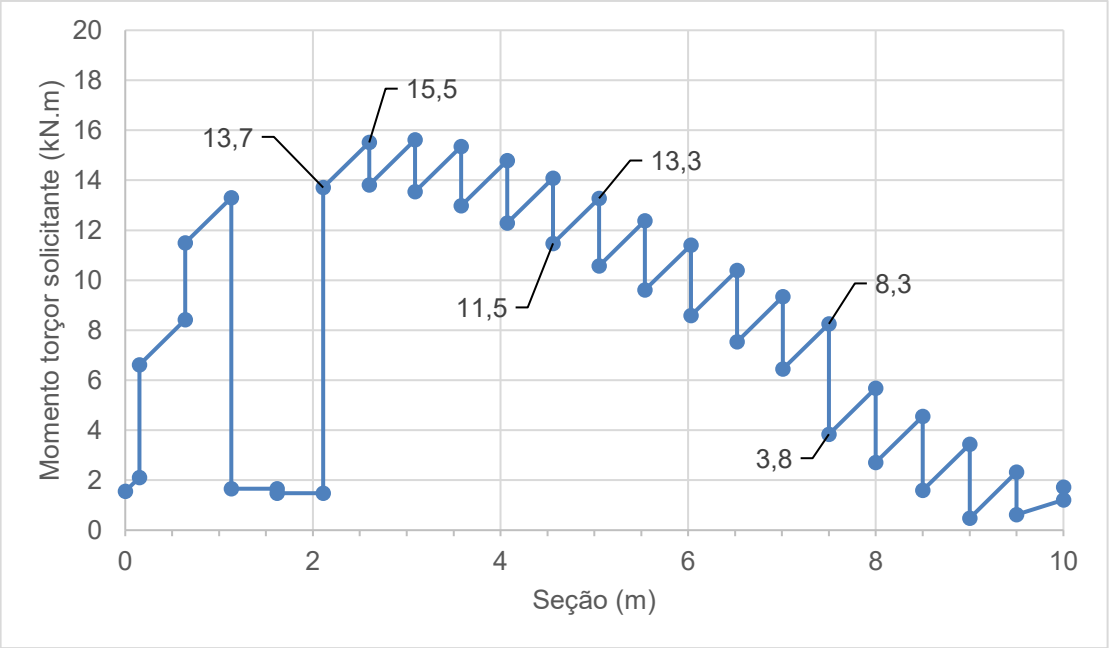


Figura 4.20 – Diagrama de esforços torçores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de máximos)

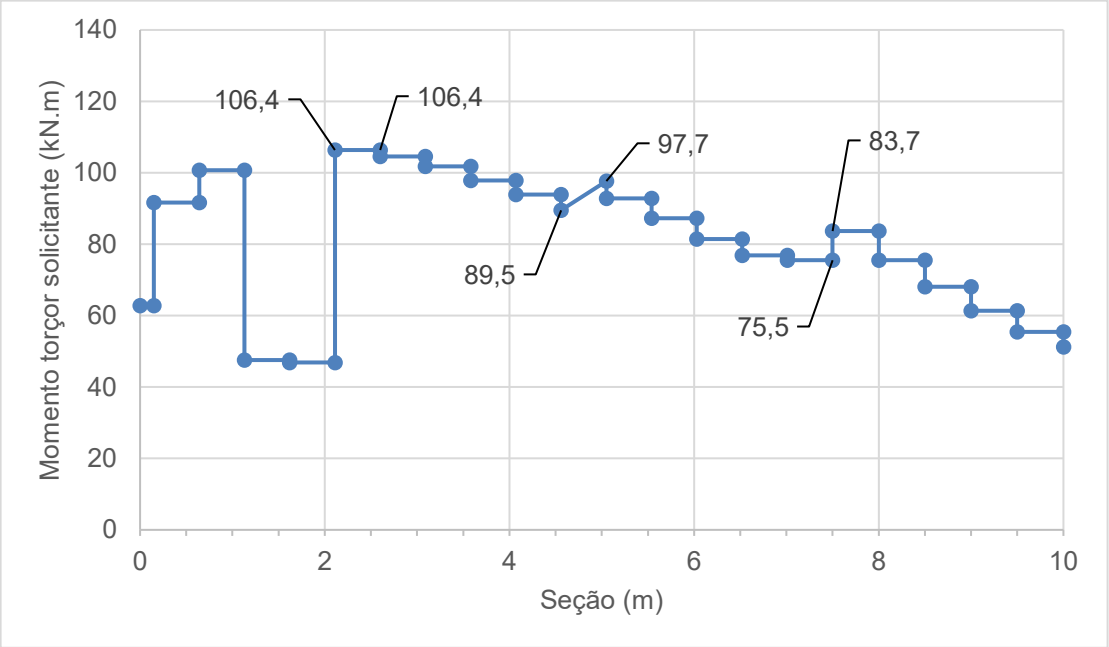


Figura 4.21 – Diagrama de esforços torçores da longarina pré-moldada utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de mínimos)

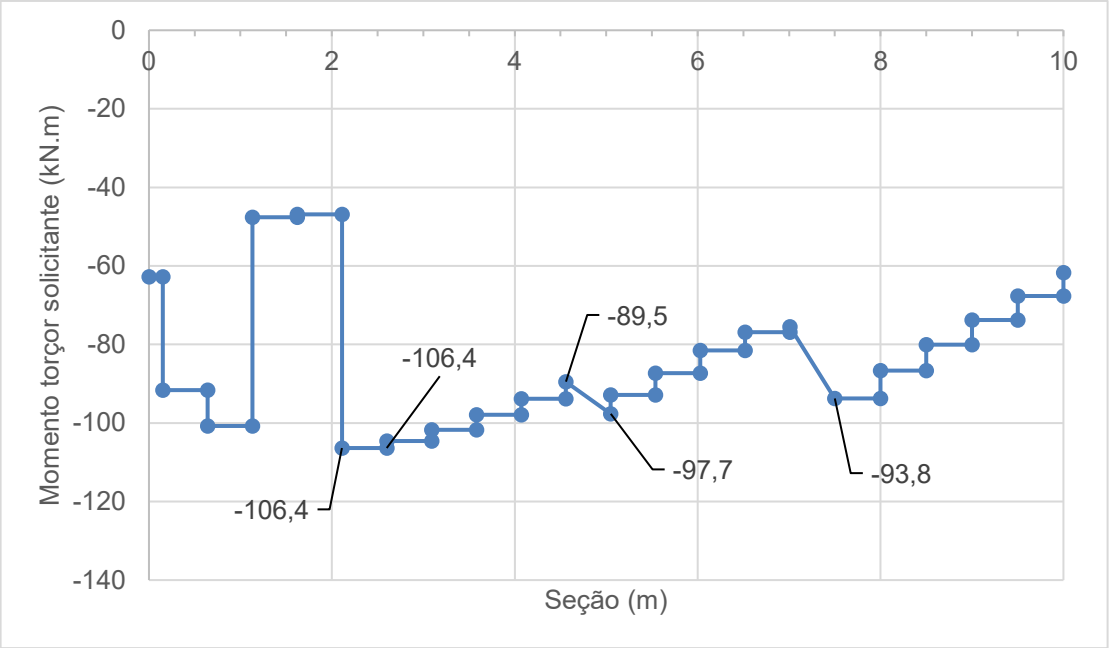


Tabela 4.1 – Tabela de esforços solicitantes das longarinas pré-moldadas alimentada pelo programa "*main.py*" para a tipologia PPC-A-20-P

Seção		S1	S2	S3	S4	S5
x (m)		0,0	2,5	5,0	7,5	10,0
Momentos fletores (kN.m)	Mg1[1] (kNm)	0,0	312,4	535,5	669,4	714,1
	Mg2*[2] (kNm)	0,0	758,6	1311,7	1641,4	1757,8
	Mg2[3] (kNm)	0,0	446,2	776,2	972,0	1043,8
	Mg3[4] (kNm)	0,0	299,4	508,4	630,4	673,2
	Mqc[5] (kNm)	0,0	119,4	204,8	255,9	273,0
	Mq+[6] (kNm)	0,0	779,0	1247,5	1523,7	1621,6
	Mq-[7] (kNm)	0,0	-40,1	-82,1	-119,7	-112,3

Força Cortante (kN)	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
	Vg1 (kN)	142,8	107,1	71,4	35,7
	Vg2* (kN)	357,4	261,7	175,8	90,0
	Vg2 (kN)	214,6	154,6	104,4	54,3
	Vg3 (kN)	142,8	106,4	71,2	37,1
	Vq+ (kN)	35,5	88,4	128,4	151,0
	Vq- (kN)	-395,4	-308,7	-243,1	-193,4

Momento Torçor (kNm)	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
	Tg (kN)	15,1	15,1	13,1	5,1
	Tq+ (kN)	106,4	106,4	97,7	83,7
	Tq- (kN)	-106,4	-106,4	-97,7	-93,8

Legenda: esforço devido ao 1 - peso próprio da longarina, 2 - peso-próprio da laje, transversinas e longarinas, 3 - peso-próprio da laje e transversinas (g1 - g2*), 4 - cargas complementares (defensas, pavimentação e recapeamento), 5 - sobrecarga de construção, 6 - envoltória de máximos do TB-450, 7 – envoltória de mínimos do TB-450

4.2.2 Longarinas metálicas

O processo de extração dos esforços para as longarinas metálicas e diafragmas seguiu os mesmos procedimentos adotados anteriormente. A principal diferença consistiu na utilização da ferramenta "*User Discretization Point*" do CSiBridge, que permite a definição dos pontos exatos onde os esforços devem ser obtidos. Com isso, o *software* ajusta automaticamente a discretização da malha, de modo que os nós coincidam com as seções de interesse, eliminando a necessidade de interpolação. As seções consideradas foram nos centroides dos aparelhos de apoio, região de emendas e metade do comprimento do perfil. Os esforços extraídos foram divididos em duas partes:

- Comportamento do perfil isolado (*Beam*)

- Peso próprio da estrutura metálica (longarinas e diafragmas);
- Peso próprio da laje de concreto;
- Sobrecarga de construção;
- Retração e fluência;
- Temperatura gradiente positivo;
- Temperatura gradiente negativo.

- Comportamento do perfil composto (*Girder*)
  - Cargas complementares (defensas, pavimentação e recapeamento);
  - Envoltórias de máximos e mínimos associados às cargas móveis.

De forma a exemplificar essa extração, serão utilizados os dados da tipologia MAC-A-20-P para os esforços das longarinas metálicas, cujos diagramas de esforços constam na Figura 4.22 a Figura 4.43. No Anexo C apresenta-se a planilha de dimensionamento preenchida com os esforços oriundos da análise, que constam na Tabela 4.2.

Figura 4.22 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de peso próprio da estrutura metálica (longarinas e diafragmas)

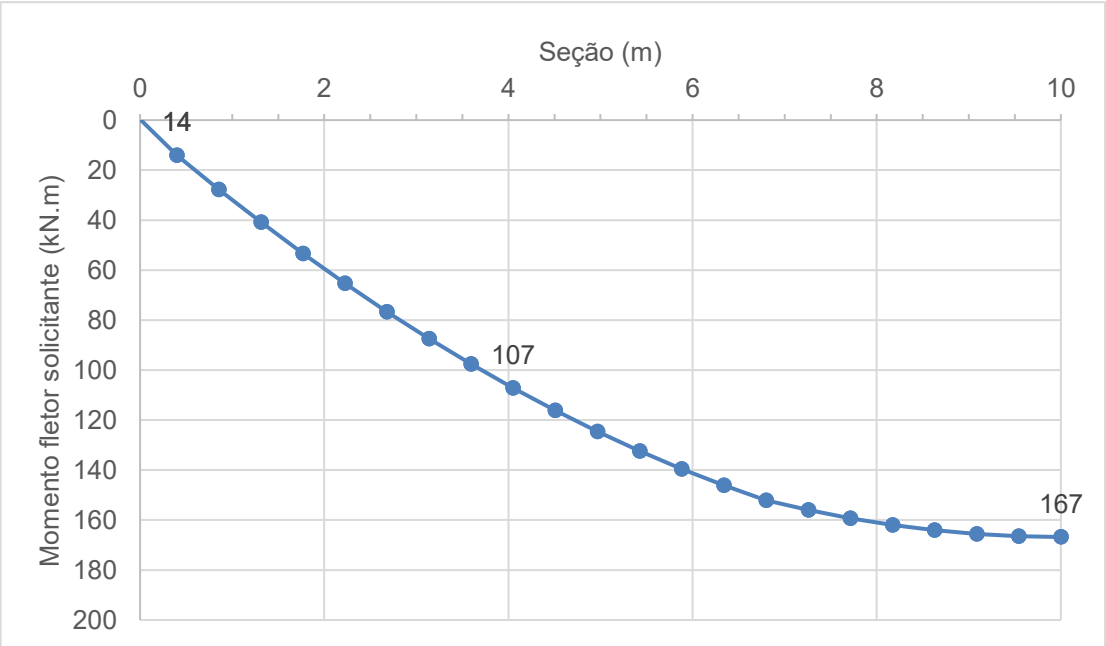


Figura 4.23 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de peso próprio da laje de concreto

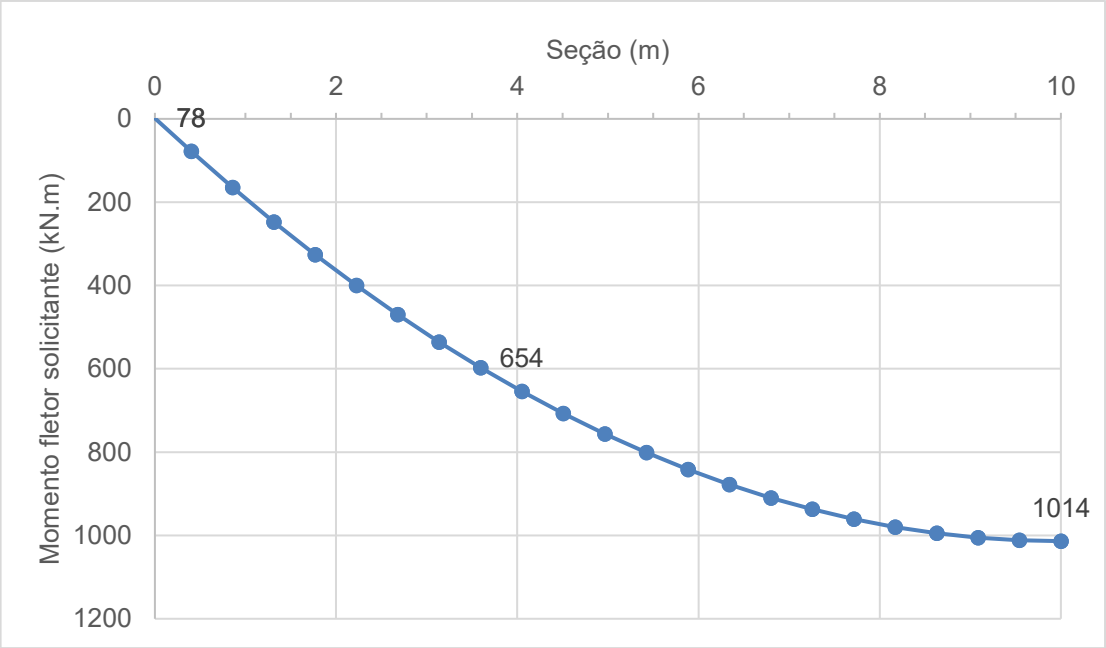


Figura 4.24 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de defensas

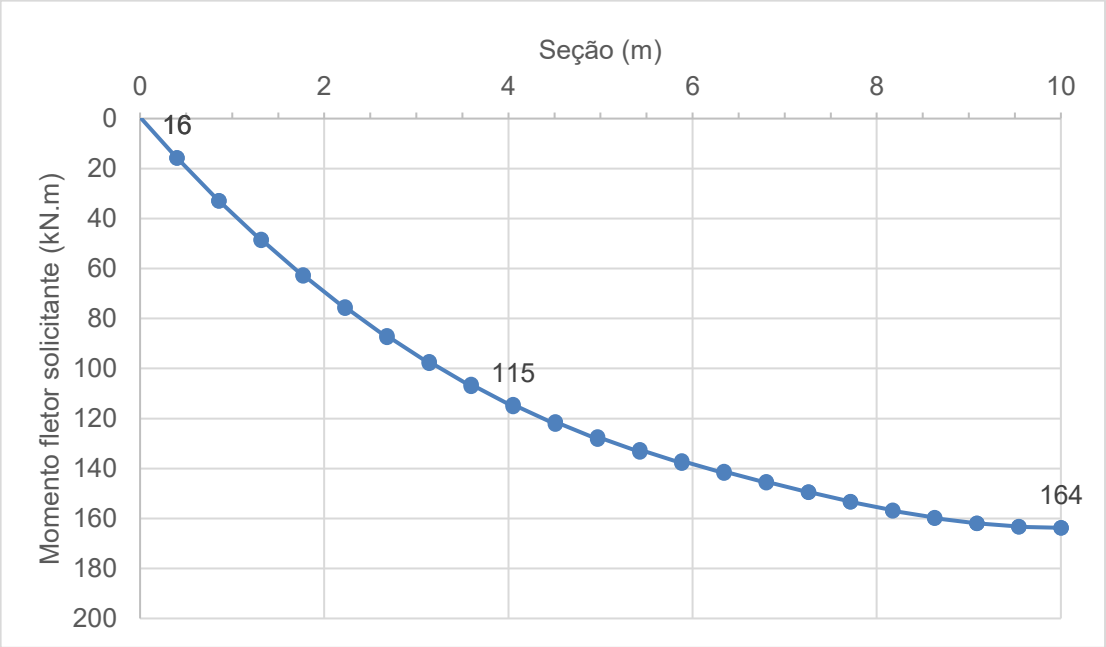


Figura 4.25 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de pavimentação

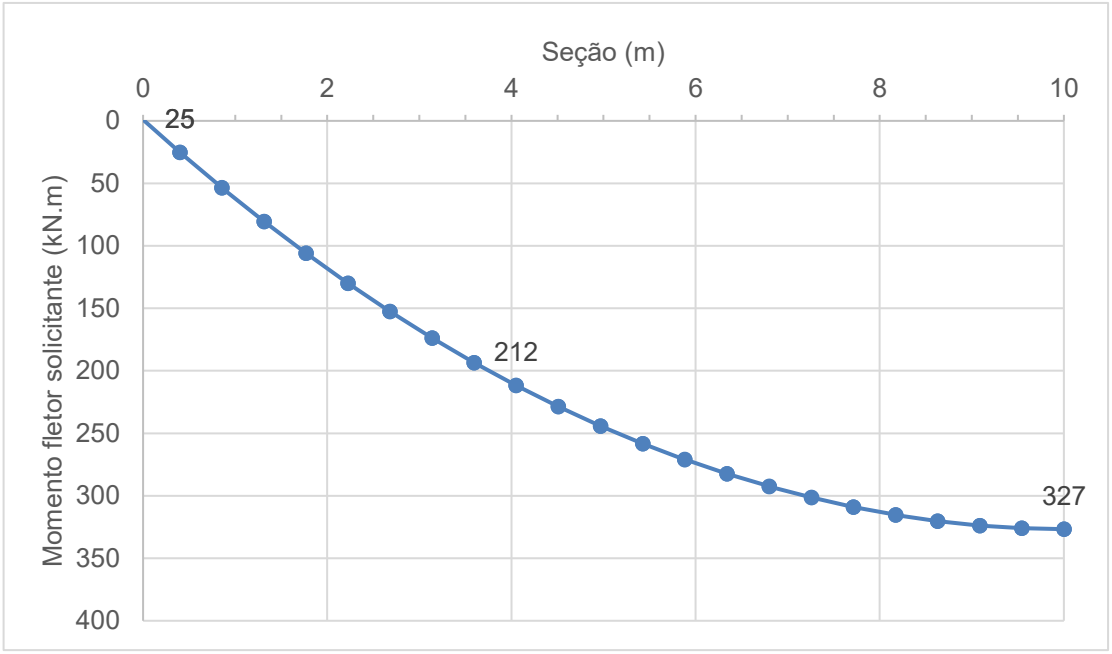


Figura 4.26 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de recapeamento

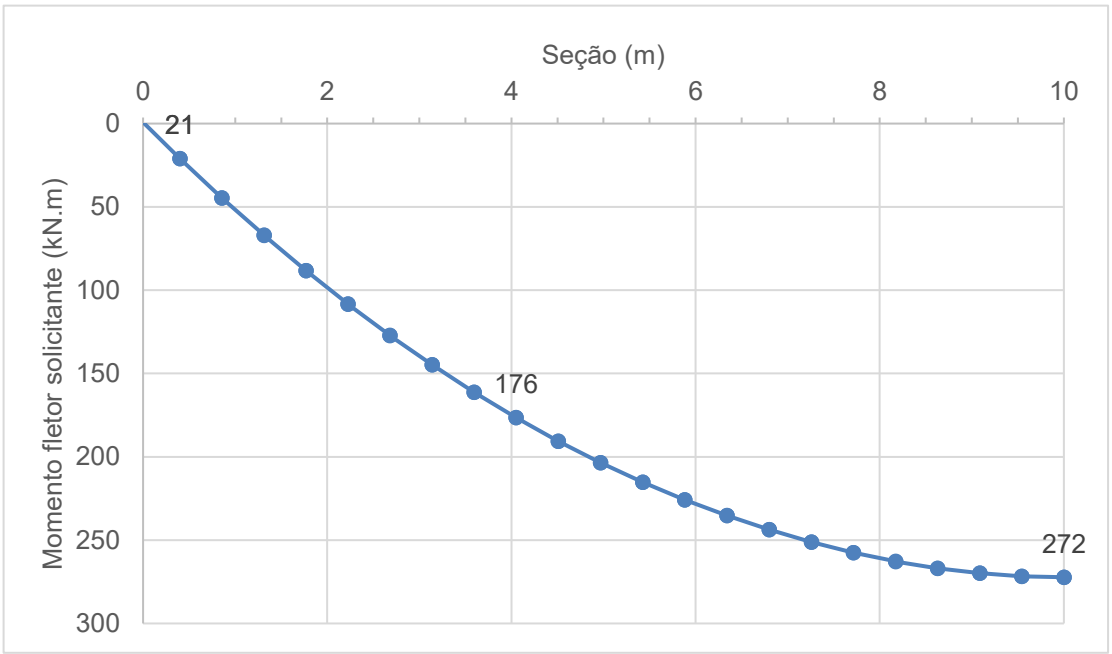


Figura 4.27 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de máximos)

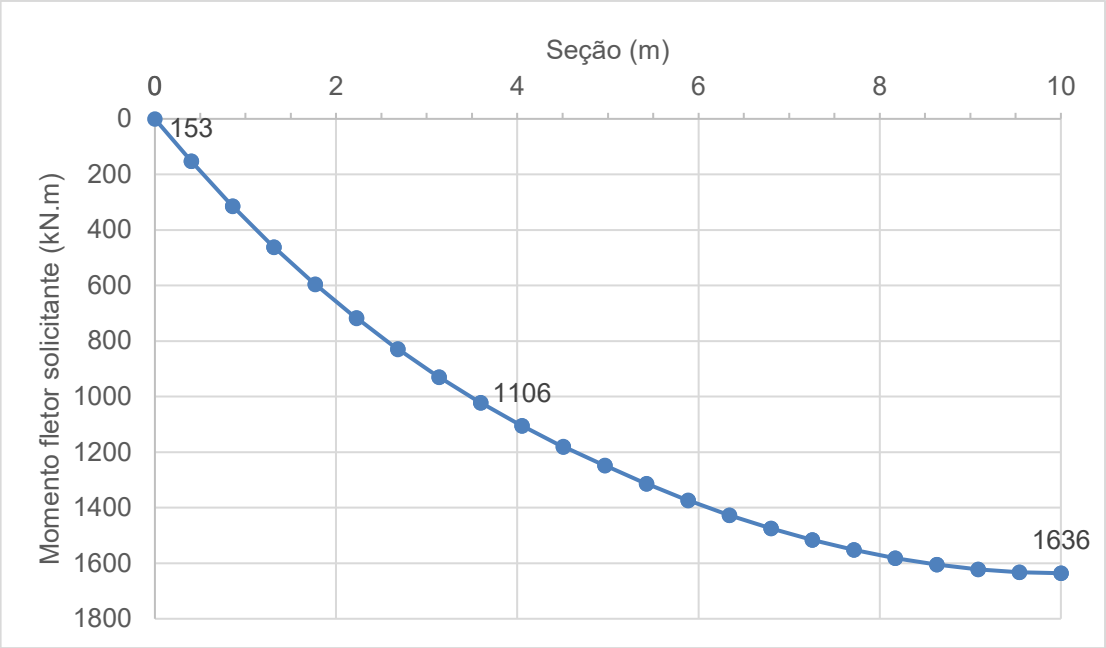


Figura 4.28 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de mínimos)

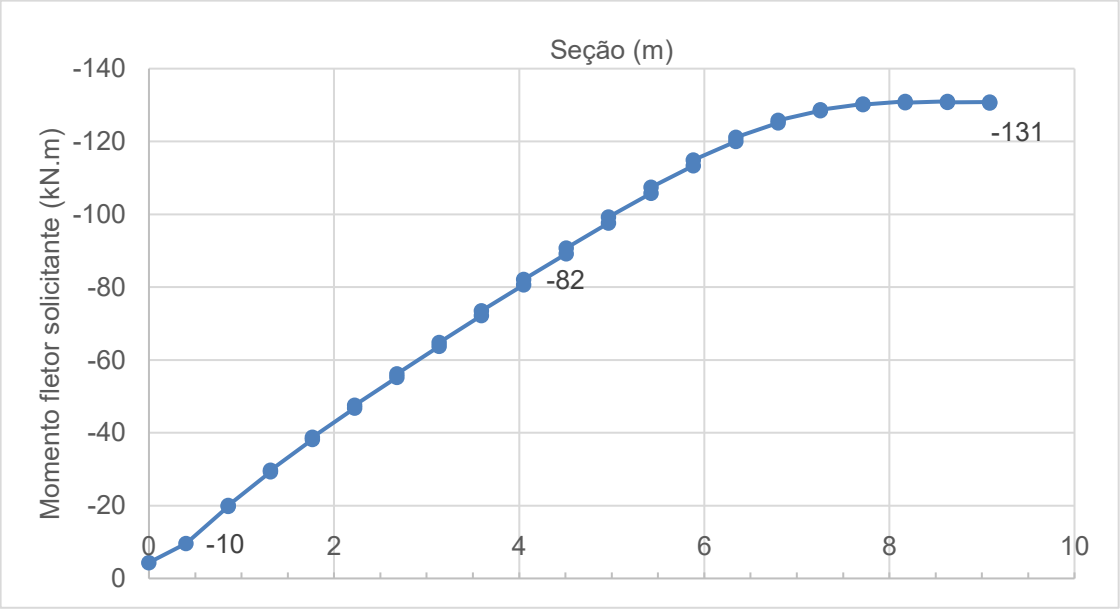


Figura 4.29 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de carga de sobrecarga de construção

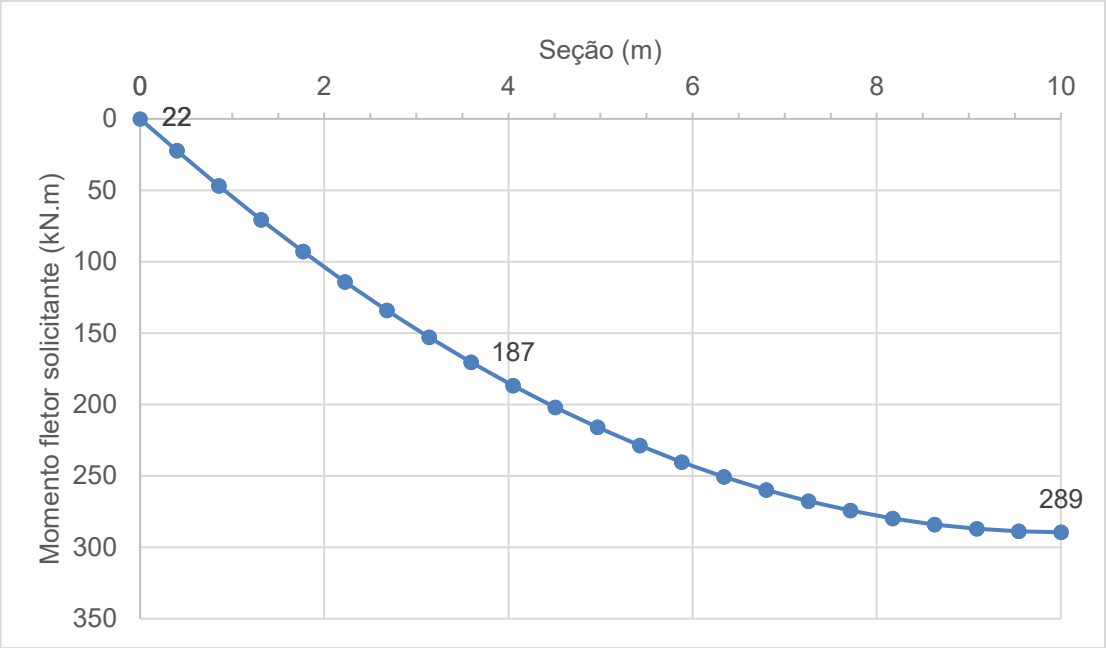


Figura 4.30 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de retração e fluência

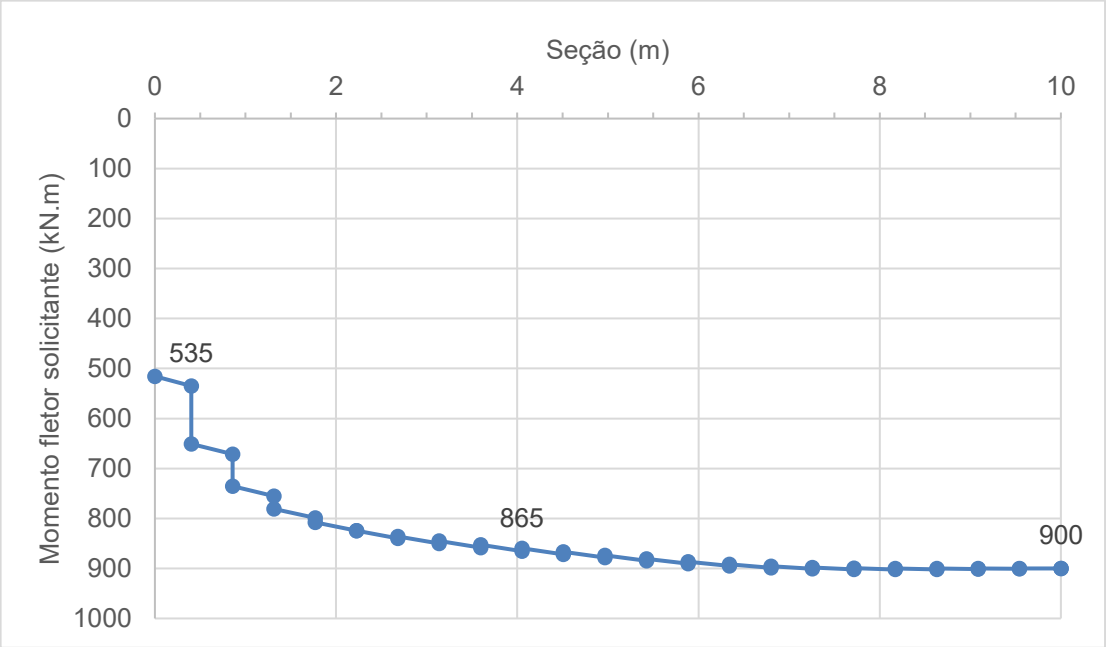


Figura 4.31 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de gradiente de temperatura positivo

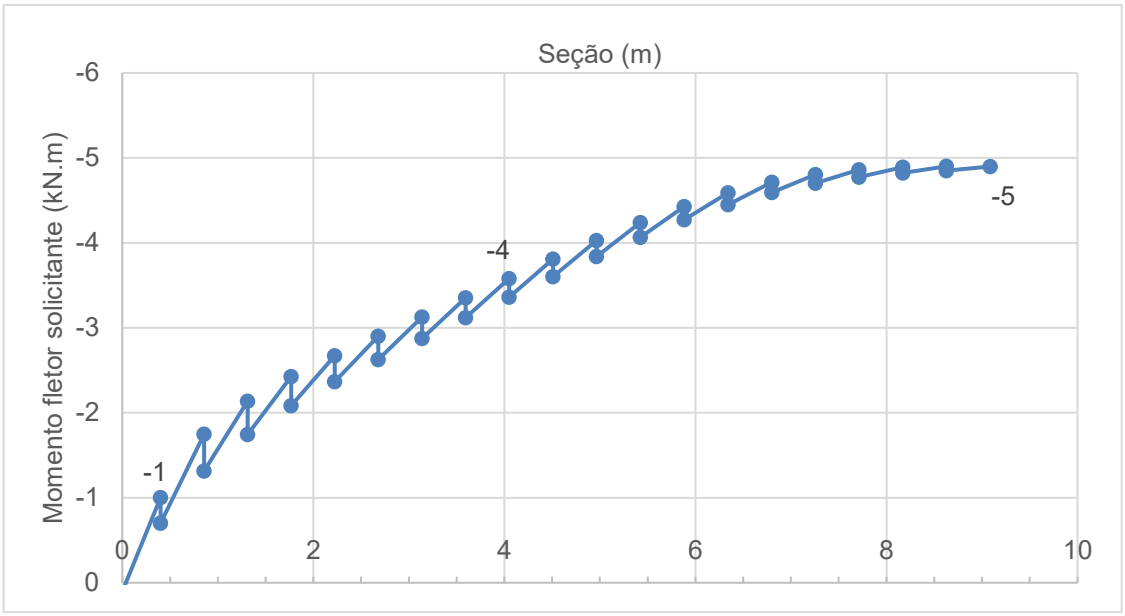


Figura 4.32 – Diagrama de momentos fletores da longarina metálica utilizado para a extração de esforços para o caso de carga de gradiente de temperatura negativo

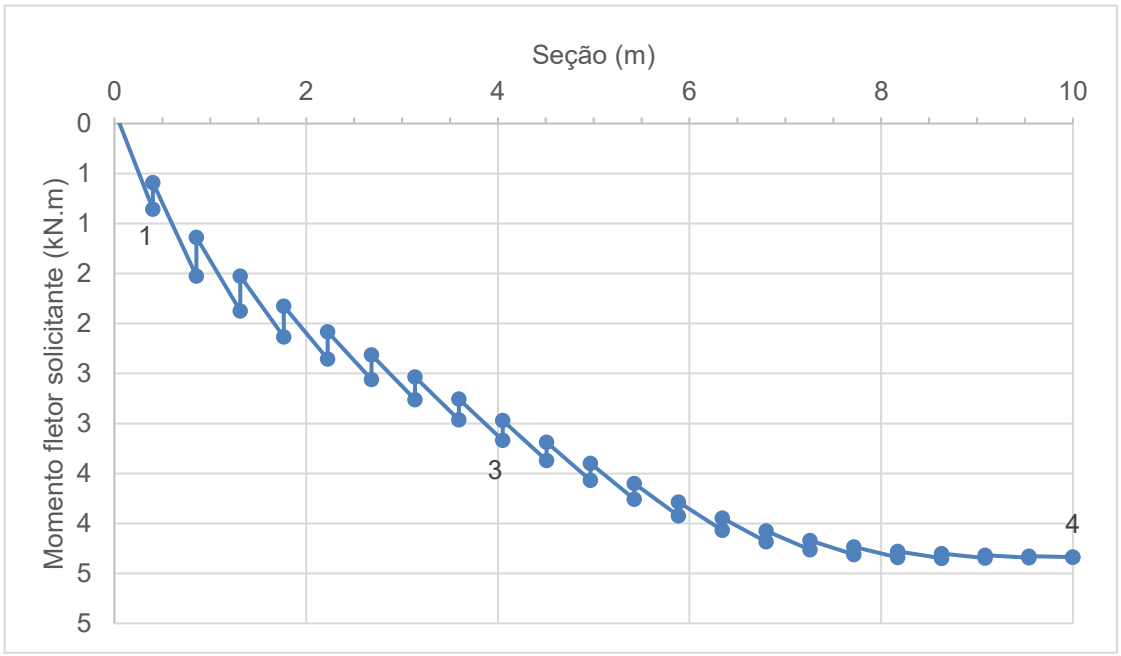


Figura 4.33 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de peso próprio da estrutura metálica (longarinas e diafragmas)

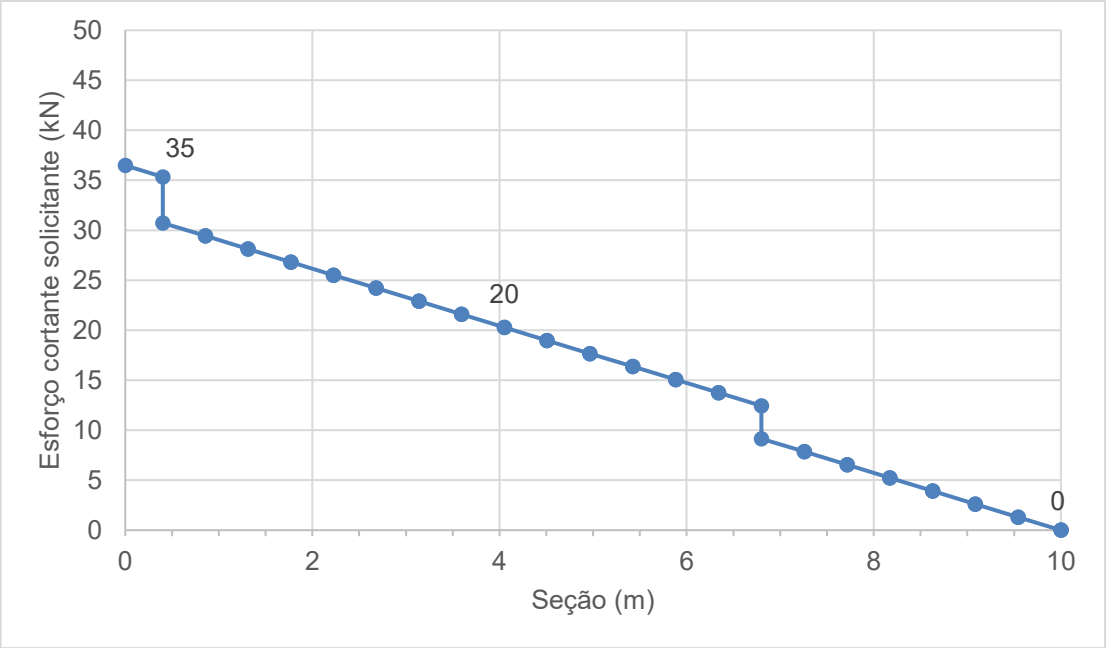


Figura 4.34 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de peso próprio da laje de concreto

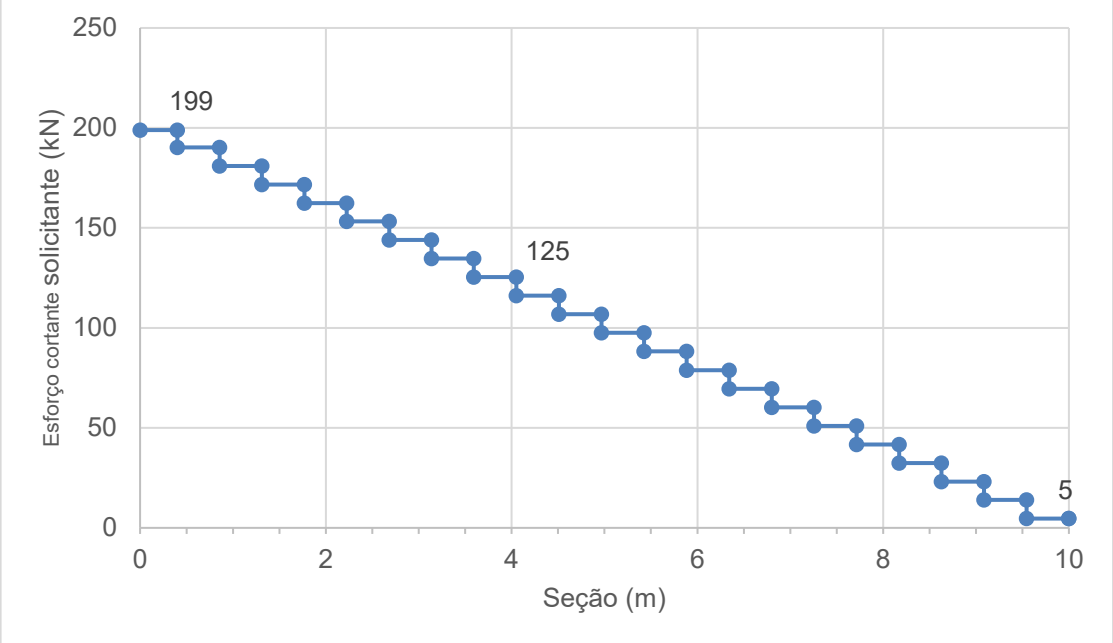


Figura 4.35 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de defensas

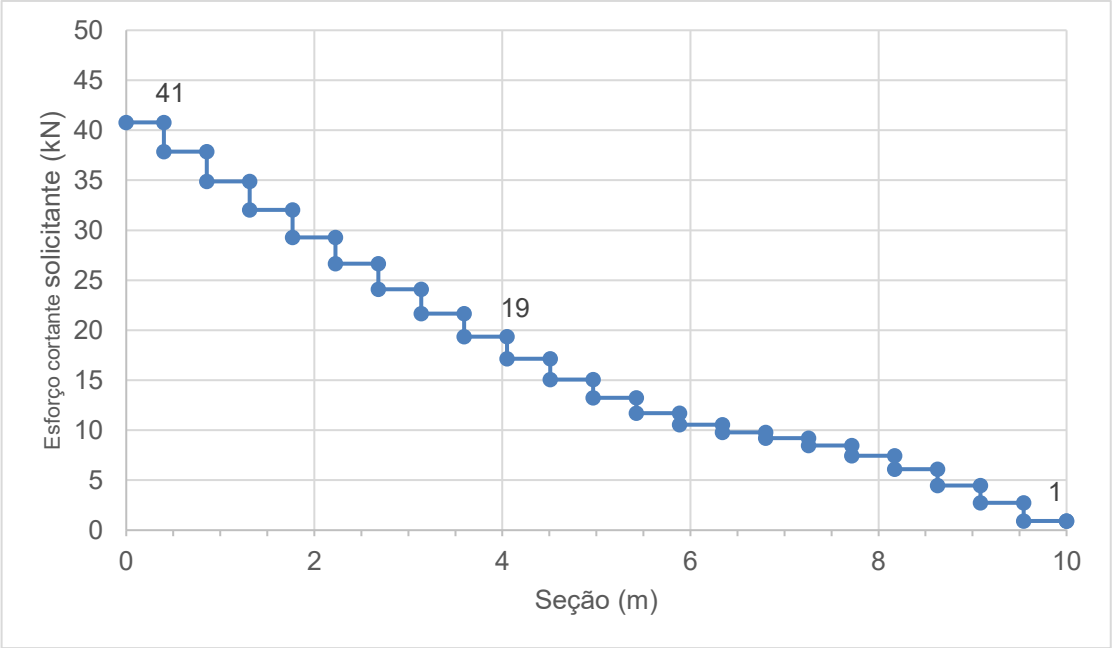


Figura 4.36 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de pavimentação

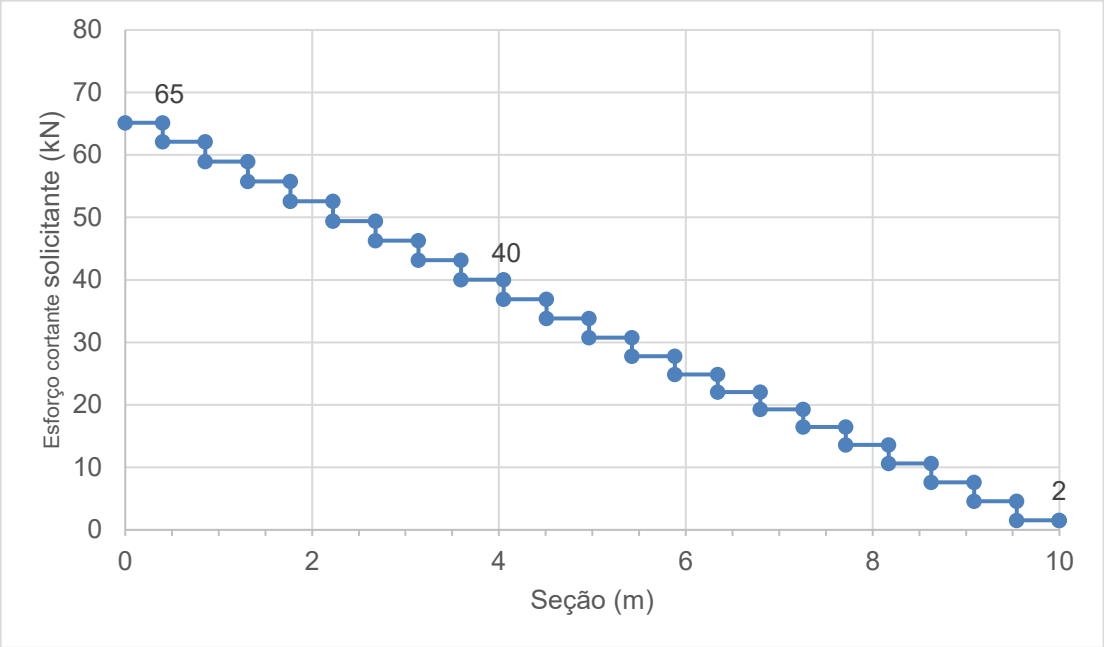


Figura 4.37 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de recapeamento

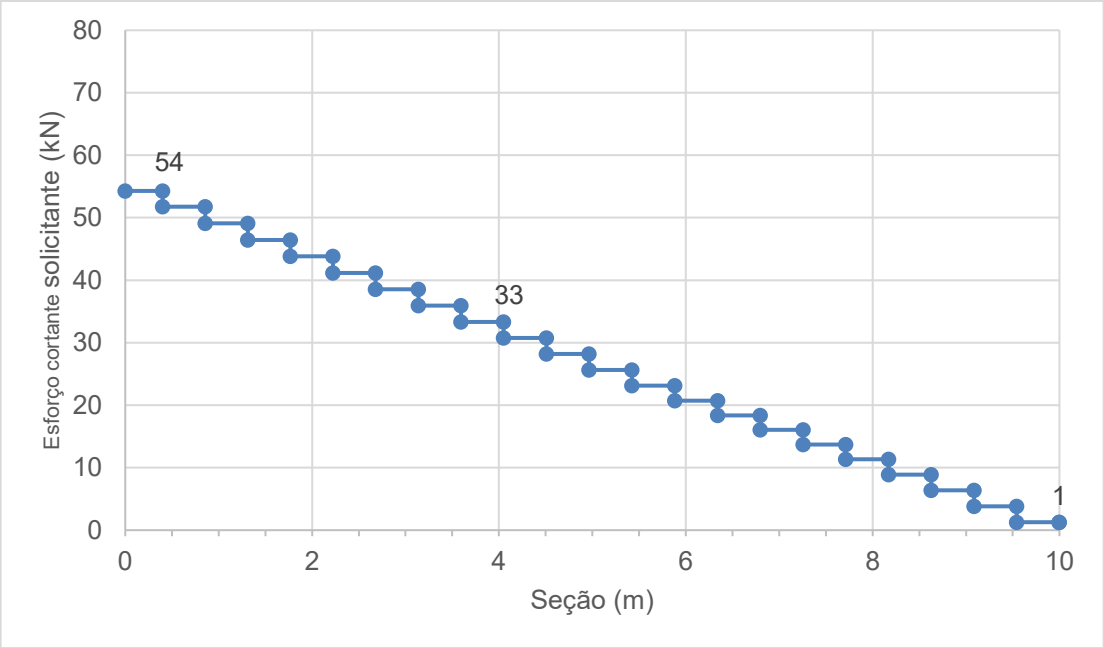


Figura 4.38 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de máximos)

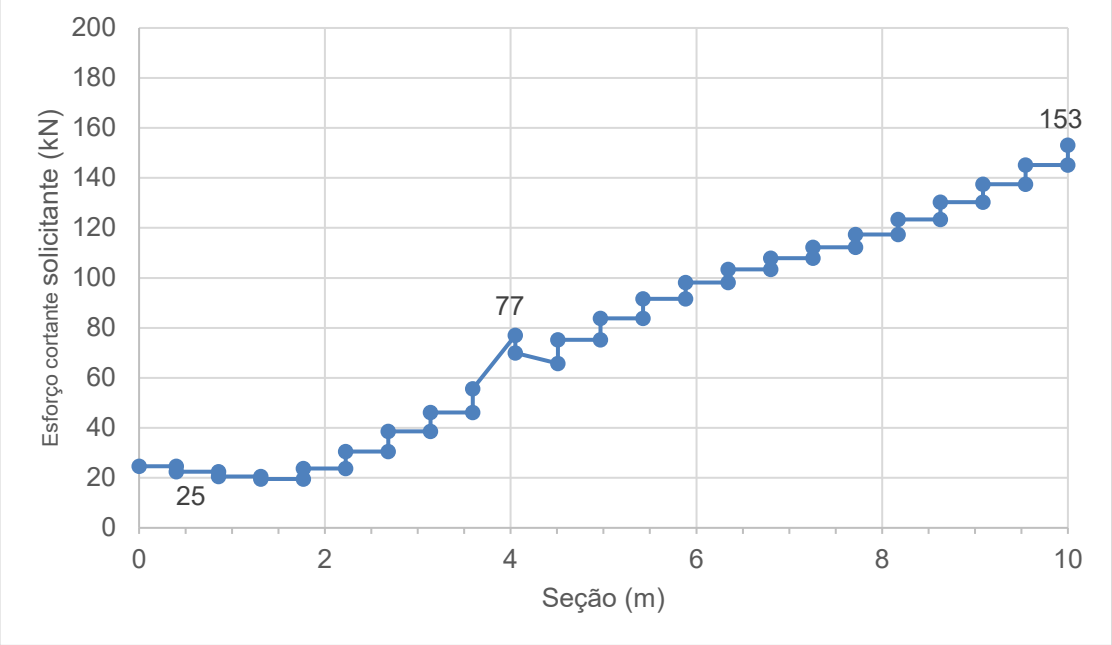


Figura 4.39 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga do veículo TB-450 (envoltória de mínimos)

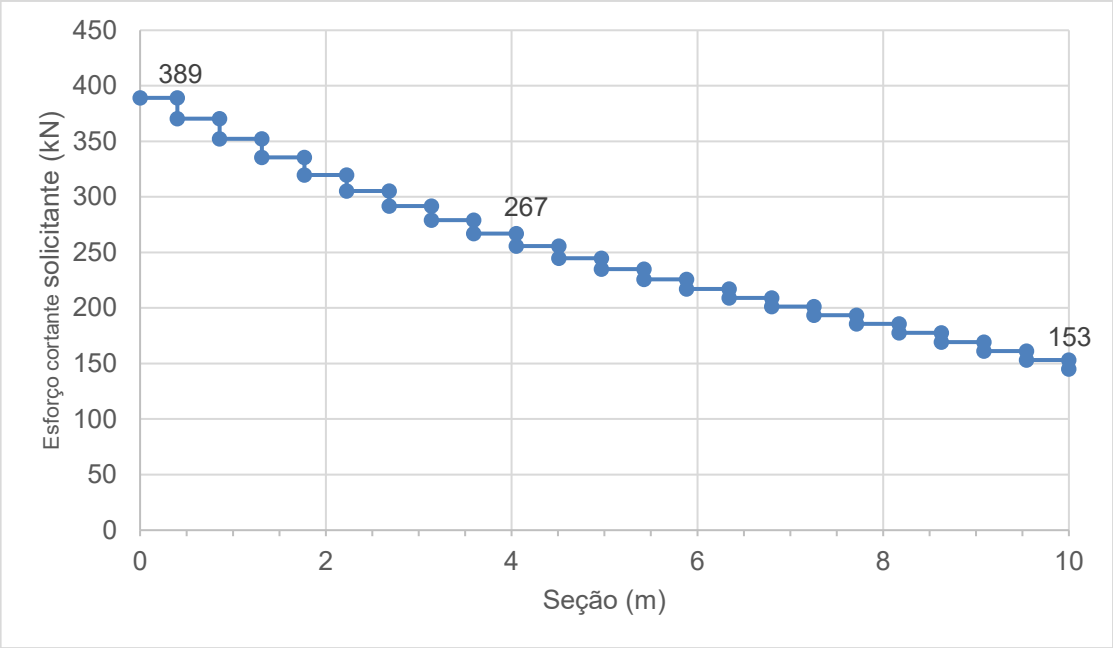


Figura 4.40 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de sobrecarga de construção

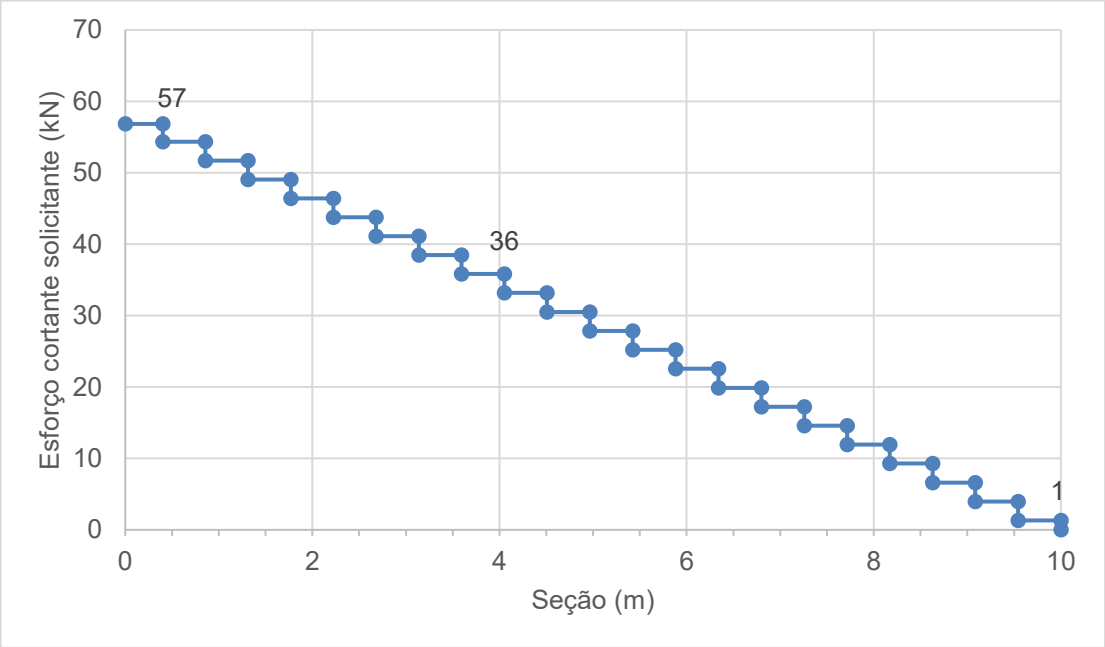


Figura 4.41 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de retração e fluência

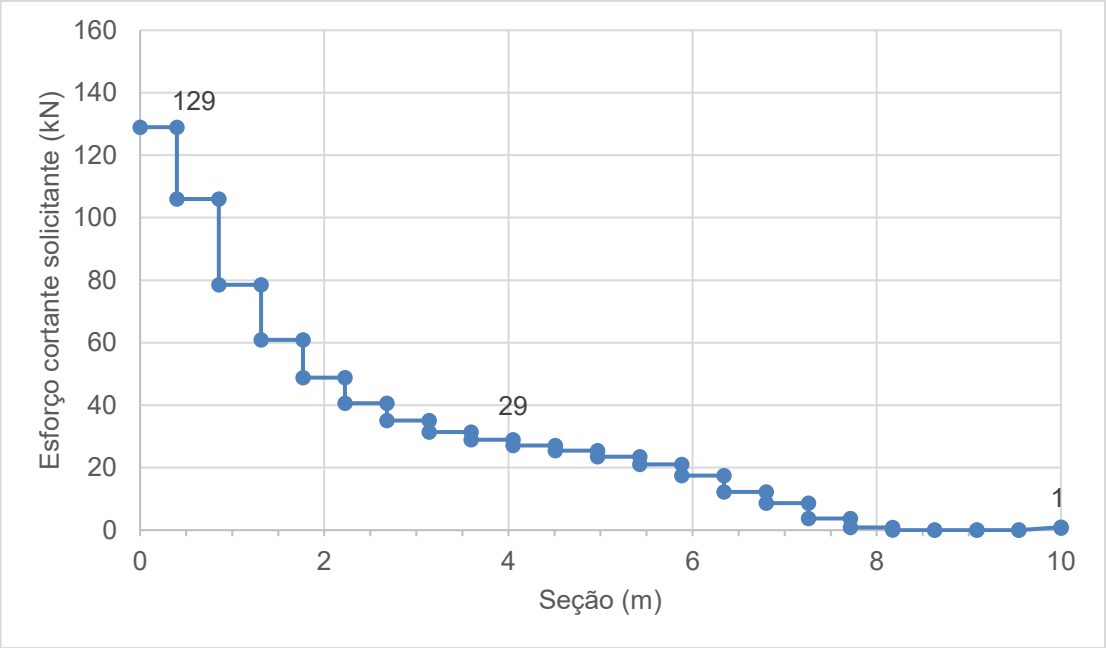


Figura 4.42 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de gradiente de temperatura positivo

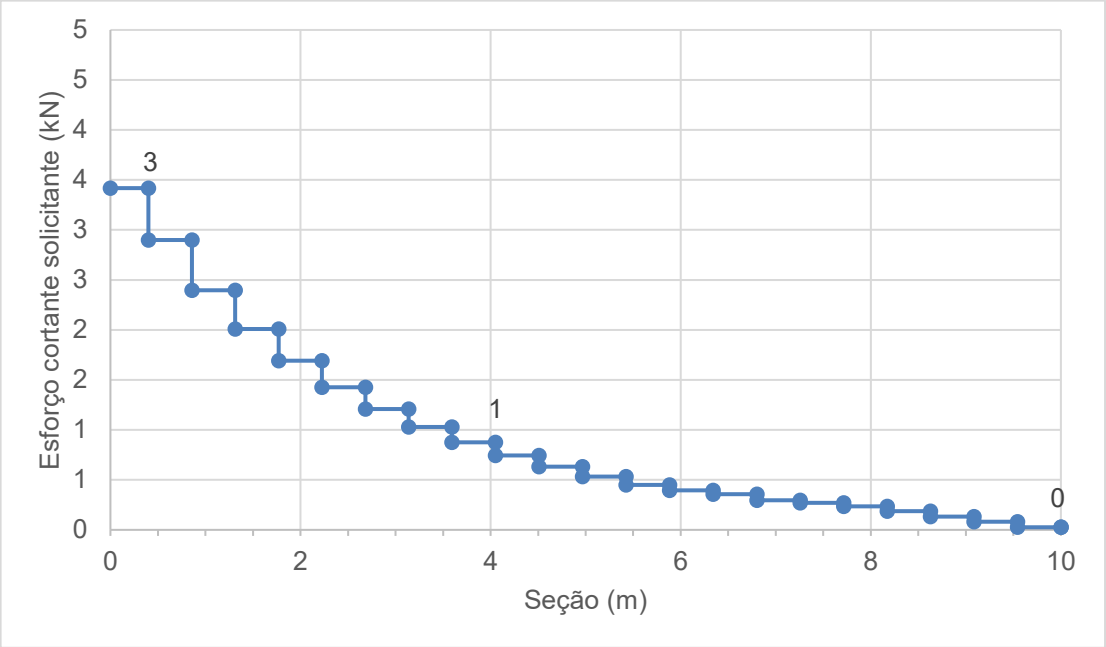


Figura 4.43 – Diagrama de esforços cortantes da longarina metálica utilizado para a extração de esforços para o caso de carga de gradiente de temperatura negativo

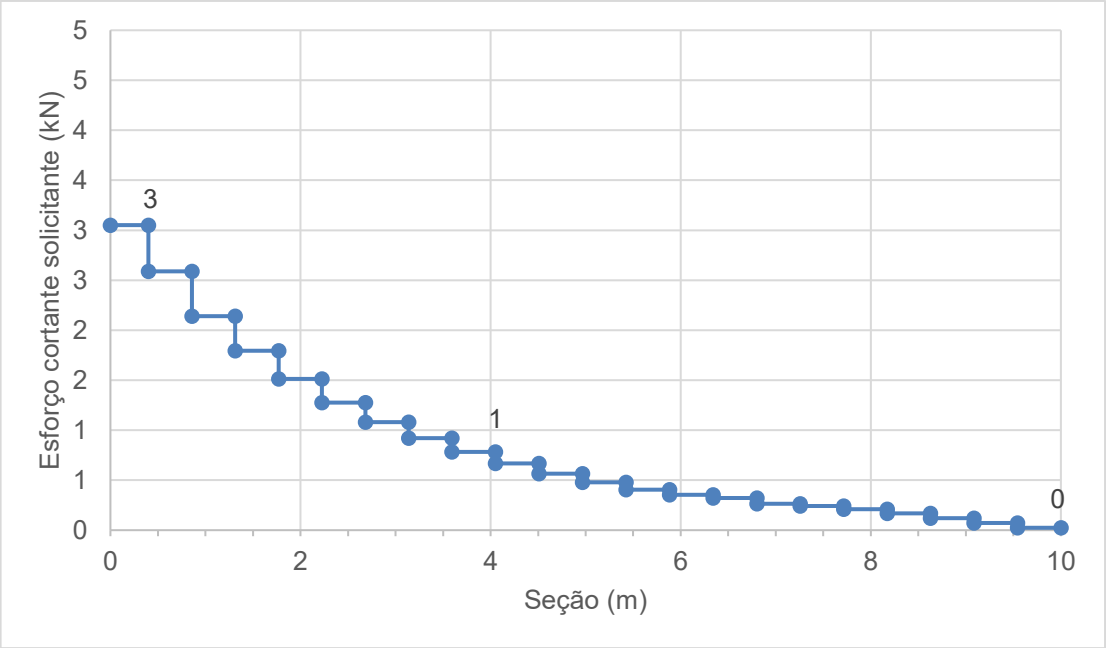


Tabela 4.2 – Tabela de esforços solicitantes das longarinas metálicas alimentada pelo programa "*main.py*" para a tipologia MAC-A-20-P

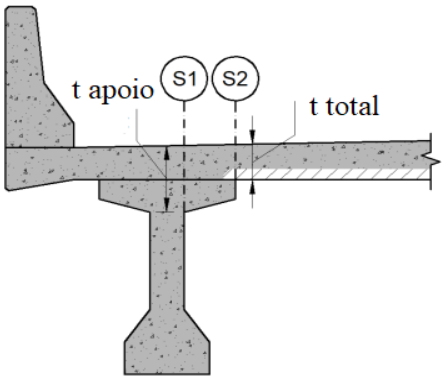
Modelo numérico	Esforços:	M (kNm)			V (kN)		
	Seção:	S0	S1	S2	S0	S1	S2
	x (m):	0,40	4,05	10,00	0,40	4,05	10,00
Beam	G1[1]	14	107	167	35	20	0
Beam	G2[2]	78	654	1014	199	125	5
Girder	G3[3]	16	115	164	41	19	1
Girder	G4[4]	25	212	327	65	40	2
Girder	G5[5]	21	176	272	54	33	1
Girder	qM[6]	153	1106	1636	25	77	153
Girder	qmn[7]	-10	-82	-131	389	267	153
Beam	qc[8]	22	187	289	57	36	1
Beam	R[9]	535	865	900	129	28,9	1
Beam	T+[10]	-1	-4	-5	3	1,0	0
Beam	T-[10]	1	3	4	3	0,9	0

Legenda: esforço devido ao 1 – peso próprio da estrutura metálica (longarinas e diafragmas), 2 – peso próprio da laje de concreto, 3 – defensas, 4 – pavimentação, 5 – recapeamento, 6 – envoltória de máximos do TB-450, 7 – envoltória de mínimos do TB-450, 8 – sobrecarga de construção, 9 – retração e fluência, 10 – gradiente de temperatura positivo (+) e negativo (-)

4.2.3 Lajes do tabuleiro

A discretização refinada dos elementos de área nas regiões de extremidade foi adotada com o objetivo de permitir a análise dos esforços cortantes atuantes no tabuleiro, especificamente nas regiões correspondentes à alma e à mesa superior das longarinas, conforme ilustrado na Figura 4.44. O processo de extração de esforços envolveu o agrupamento dos elementos de área referentes ao centro, balanço e extremidades, tanto de elementos de área para análise da seção S1 quanto S2, onde foram consideradas médias dos esforços em módulo. Os grupos criados para a tipologia PPC-A-20-P estão ilustrados na Figura 4.46 a Figura 4.49.

Figura 4.44 – Divisão das seções de extremidades S1 e S2 para a análise dos esforços cortantes atuantes na laje do tabuleiro



Além da malha inicial definida pelo usuário, o programa aplica uma discretização adicional (*Automatic Area Mesh*), que é gerada com base nos parâmetros estabelecidos pelo usuário, de modo a restringir as dimensões máximas dos elementos finitos resultantes na malha final de análise. É mostrado na Figura 4.45 que o CSiBridge exporta os resultados da análise dos elementos de área em formato tabular, permitindo o acesso individualizado aos valores obtidos em cada nó dos subelementos gerados pela discretização. Para fins de dimensionamento das lajes do tabuleiro, adotou-se como referência o valor máximo dos esforços obtidos nos nós de cada um dos grupos, sendo extraídos os esforços da carga permanente e do veículo.

Ressalta-se que, na interface gráfica do CSiBridge, há a opção de aplicar um recurso de suavização das tensões, o qual é de uso facultativo, porém habilitado por padrão. Esse recurso promove uma distribuição mais contínua dos resultados nos elementos de área, podendo atenuar picos de esforços locais observados na análise numérica, geralmente resultando em valores menores que os fornecidos nas tabelas.

De forma a exemplificar a extração, os momentos fletores atuantes no tabuleiro da tipologia PPC-A-20-P se encontram na Figura 4.50 a Figura 4.58.

Figura 4.45 – Tabela gerada pelo CSiBridge após a execução da análise para extração dos esforços internos atuantes nos nós dos elementos de área

Element Forces - Area Shells

File View Edit Format-Filter-Sort Select Options

Units: As Noted

Filter:

Element Forces - Area Shells

Area Text	AreaElem Text	ShellType Text	Joint Text	OutputCase	CaseType Text	F11 KN/cm	F22 KN/cm	F12 KN/cm	FMax KN/cm	FMin KN/cm
535	535	Shell-Thin	854	Carga perm...	Combination	-1,832	0,2534	1,876	1,357	-2,9356
535	535	Shell-Thin	857	Carga perm...	Combination	-1,4103	0,3377	1,2777	1,0117	-2,0843
535	535	Shell-Thin	858	Carga perm...	Combination	-1,4566	0,1061	1,5085	1,0236	-2,3742
535	535	Shell-Thin	855	Carga perm...	Combination	-1,8783	0,0217	2,1069	1,3829	-3,2395
536	536	Shell-Thin	855	Carga perm...	Combination	-2,2044	-0,0435	1,4329	0,6707	-2,9185
536	536	Shell-Thin	858	Carga perm...	Combination	-2,0528	-0,0132	1,5603	0,831	-2,897
536	536	Shell-Thin	859	Carga perm...	Combination	-2,0181	0,1604	1,6167	1,0206	-2,8783
536	536	Shell-Thin	856	Carga perm...	Combination	-2,1696	0,1301	1,4894	0,8619	-2,9014
537	537	Shell-Thin	857	Carga perm...	Combination	-1,4644	0,0672	0,9205	0,4987	-1,896
537	537	Shell-Thin	860	Carga perm...	Combination	-1,6767	0,0247	1,1288	0,5875	-2,2395
537	537	Shell-Thin	861	Carga perm...	Combination	-1,6678	0,0692	1,7075	1,1163	-2,715
537	537	Shell-Thin	858	Carga perm...	Combination	-1,4555	0,1117	1,4991	1,0196	-2,3634
538	538	Shell-Thin	858	Carga perm...	Combination	-2,0517	-0,0076	1,5508	0,8277	-2,8869
538	538	Shell-Thin	861	Carga perm...	Combination	-1,9491	0,0129	1,4119	0,7512	-2,6874
538	538	Shell-Thin	862	Carga perm...	Combination	-1,9224	0,1465	1,3566	0,818	-2,5939

Record: << < 1 > >> of 160 Double-click row header to view row details. Add Tables... Done

Figura 4.46 – Elementos de área selecionados para a extração dos esforços referentes ao balanço da laje do tabuleiro da tipologia PPC-A-20-P

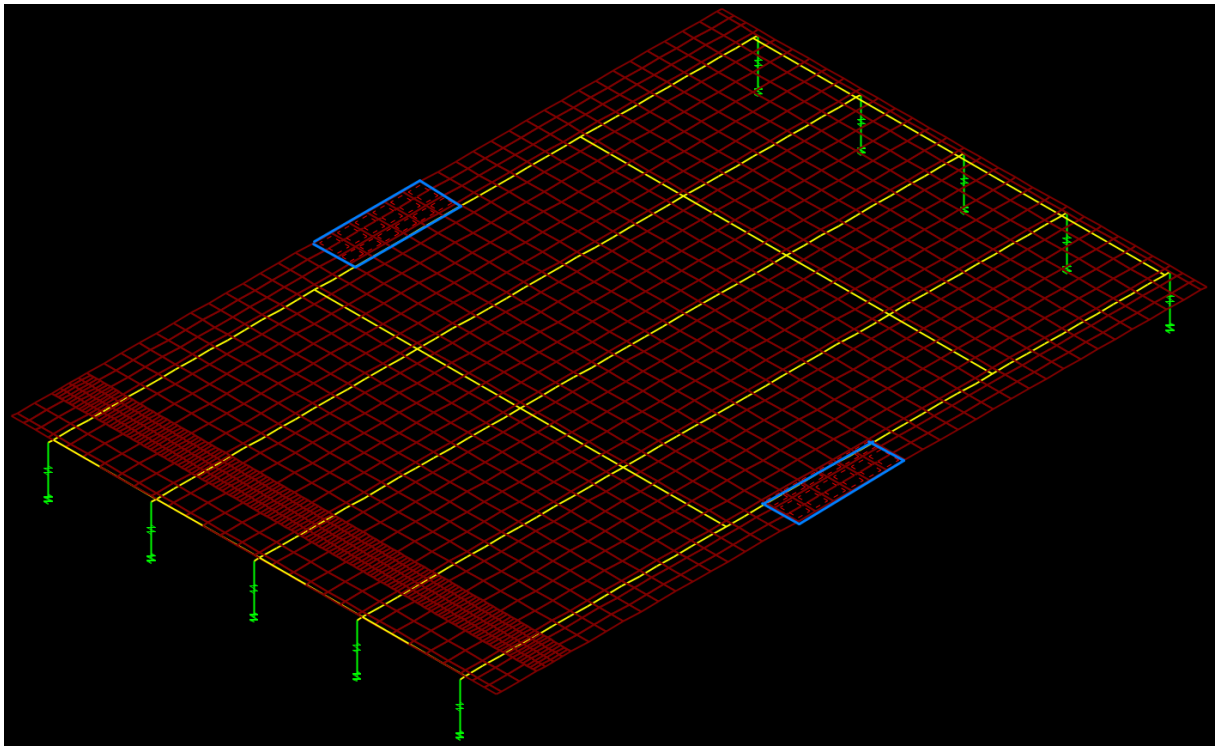


Figura 4.47 – Elementos de área selecionados para a extração dos esforços referentes ao meio do vão laje do tabuleiro da tipologia PPC-A-20-P

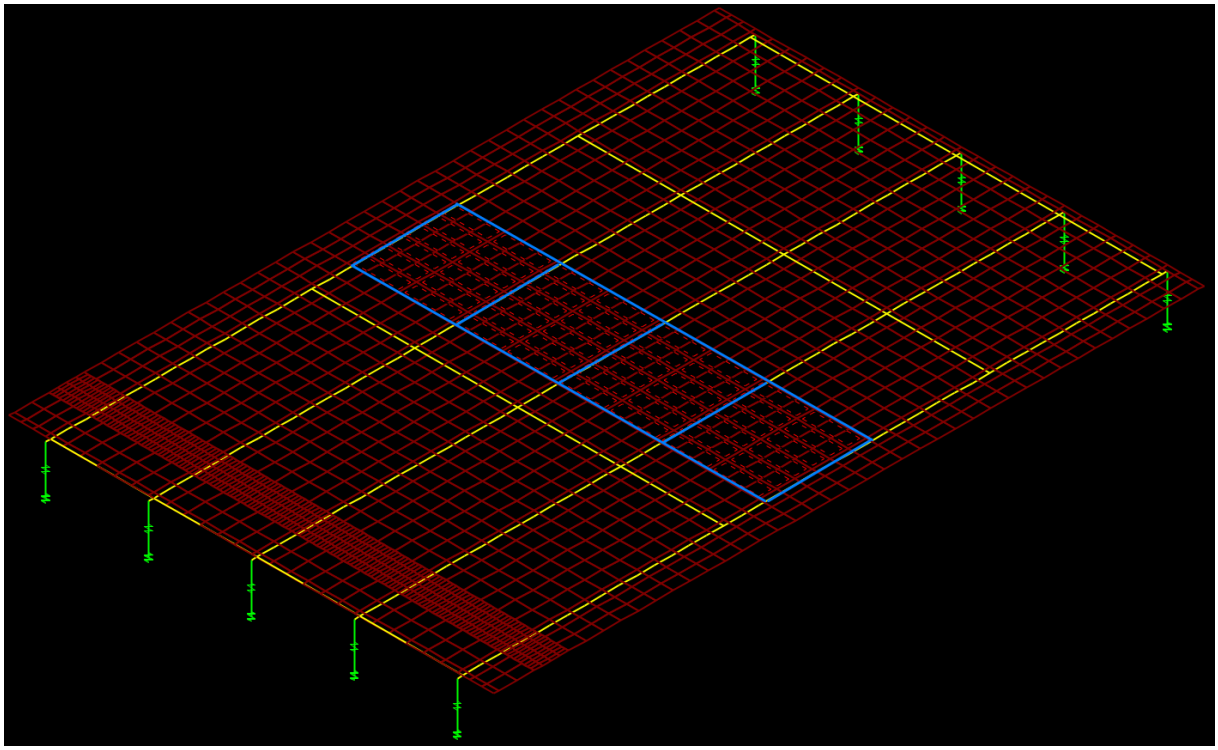


Figura 4.48 – Elementos de área selecionados para a extração dos esforços referentes à região S1 da extremidade da laje do tabuleiro da tipologia PPC-A-20-P

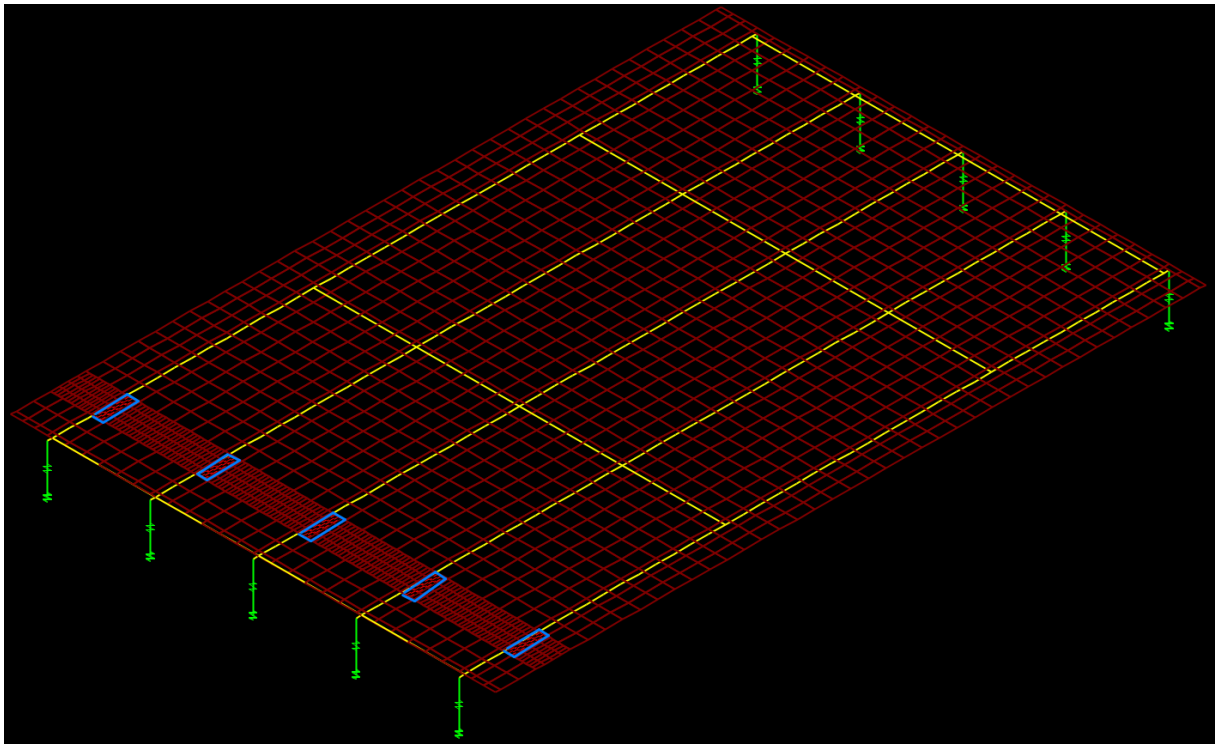


Figura 4.49 – Elementos de área selecionados para a extração dos esforços referentes à região S2 da extremidade da laje do tabuleiro da tipologia PPC-A-20-P

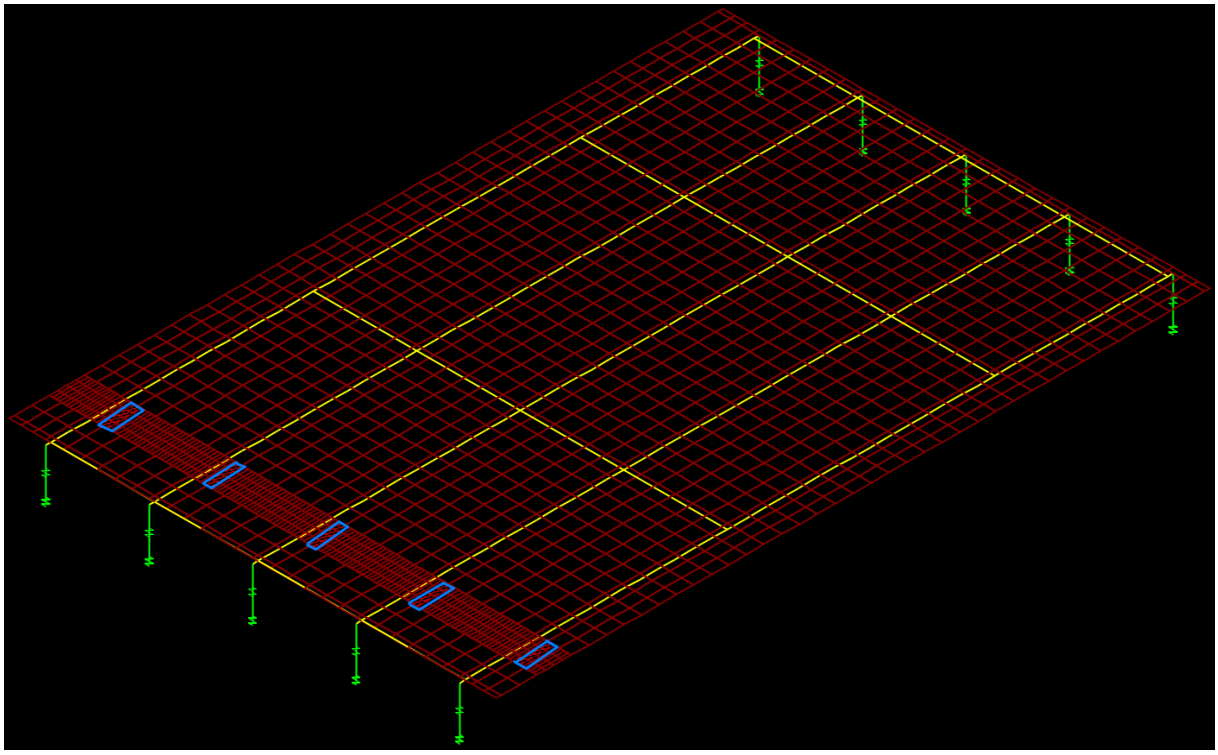


Figura 4.50 – Momentos fletores transversais (kN.m/m) atuantes na laje do tabuleiro para carga permanente

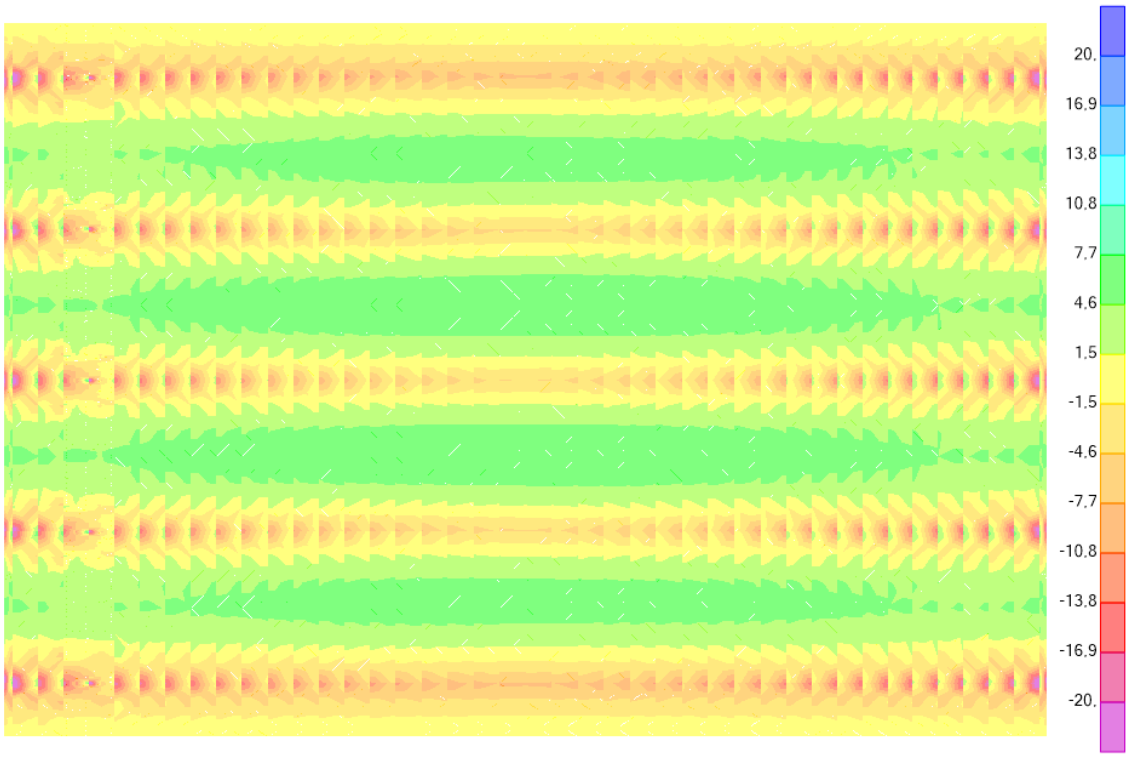


Figura 4.51 – Momentos fletores longitudinais (kN.m/m) atuantes na laje do tabuleiro para carga permanente

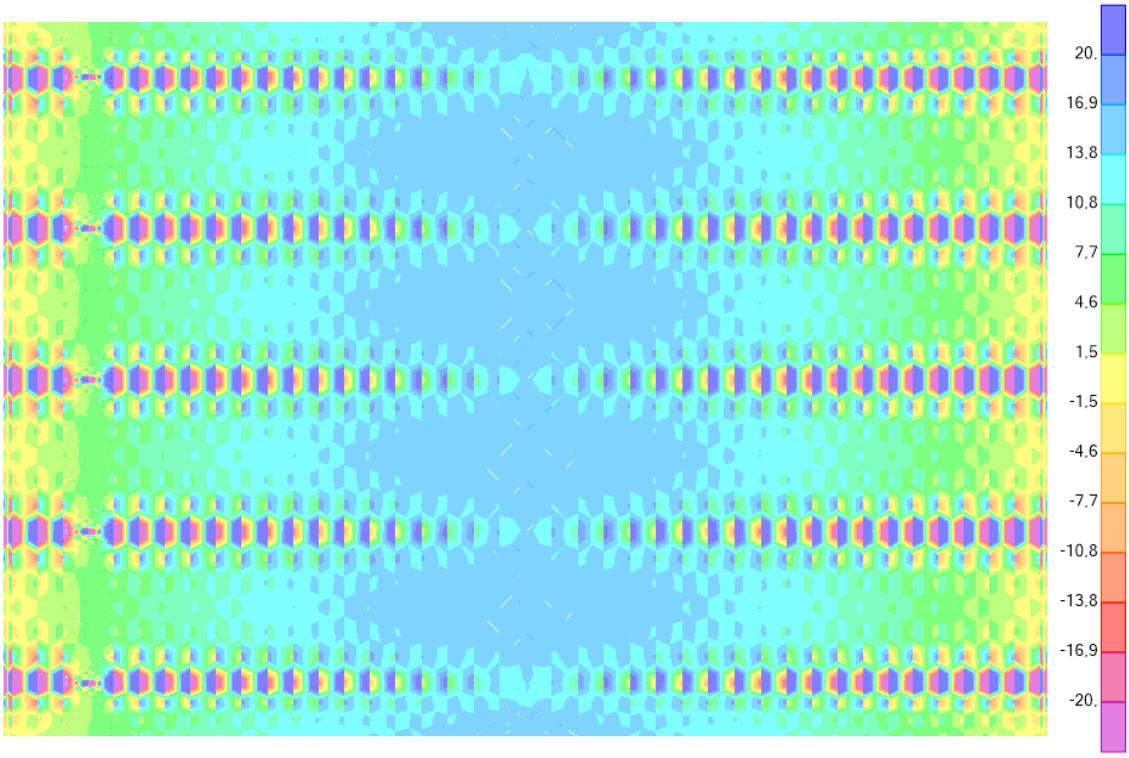


Figura 4.52 – Momentos fletores transversais (kN.m/m) atuantes na laje do tabuleiro na envoltória de máximos para o caso de carga do veículo TB-450

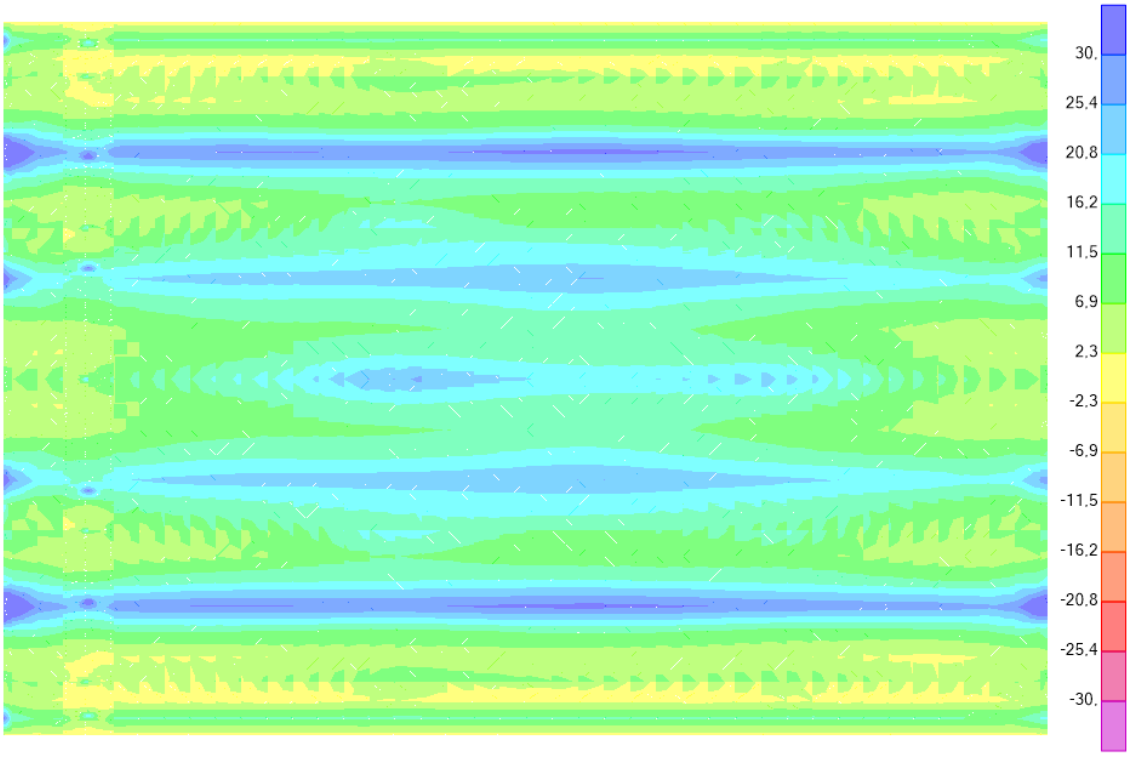


Figura 4.53 – Momentos fletores transversais (kN.m/m) atuantes na laje do tabuleiro na envoltória de mínimos para o caso de carga do veículo TB-450

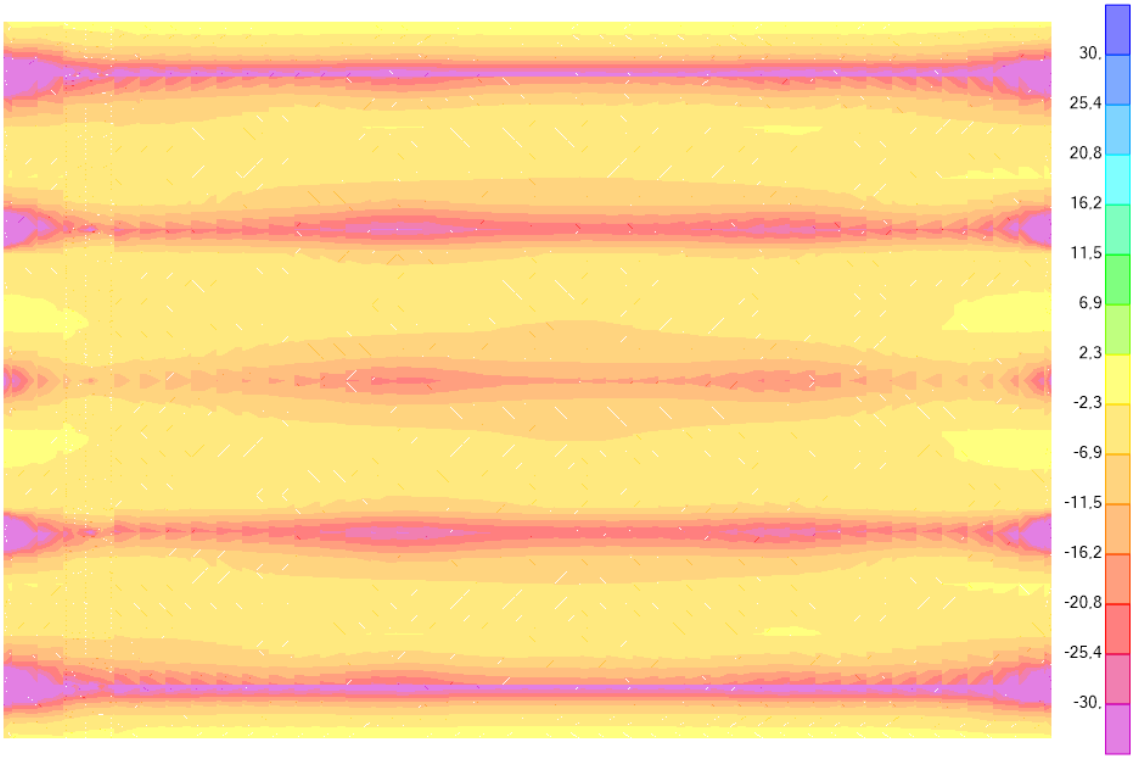


Figura 4.54 – Momentos fletores longitudinais (kN.m/m) atuantes na laje do tabuleiro na envoltória de máximos para o caso de carga do veículo TB-450

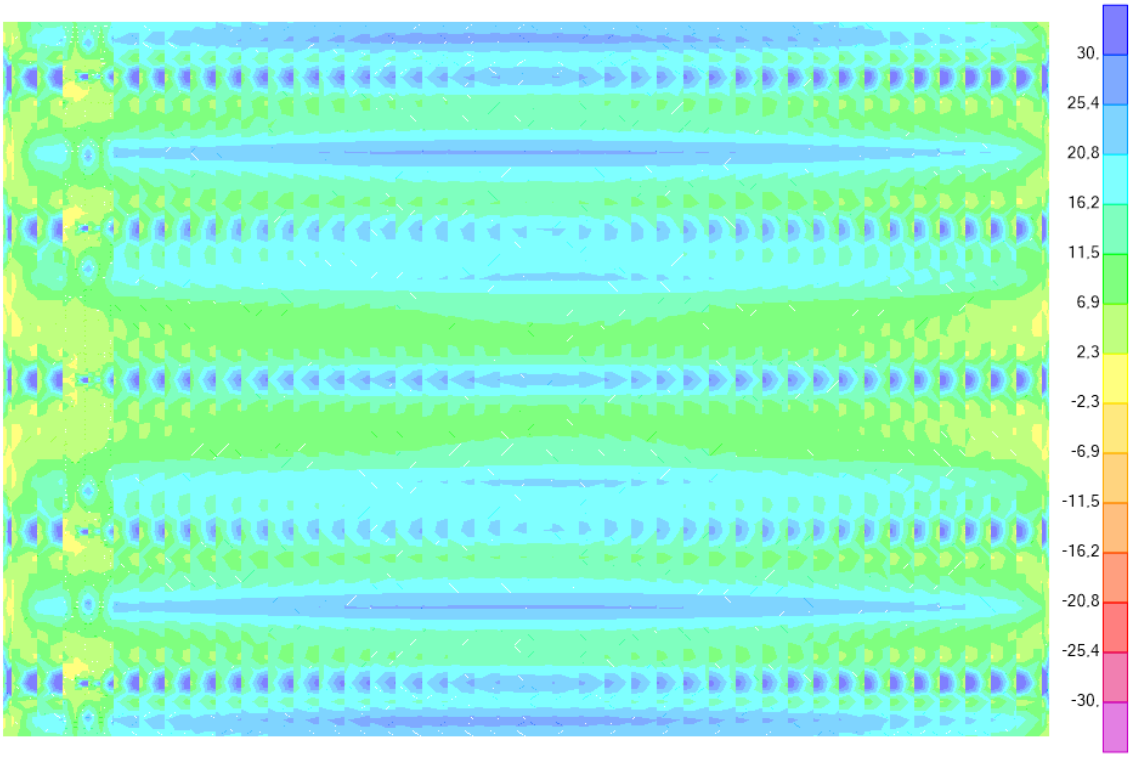


Figura 4.55 – Momentos fletores longitudinais (kN.m/m) atuantes na laje do tabuleiro na envoltória de mínimos para o caso de carga do veículo TB-450

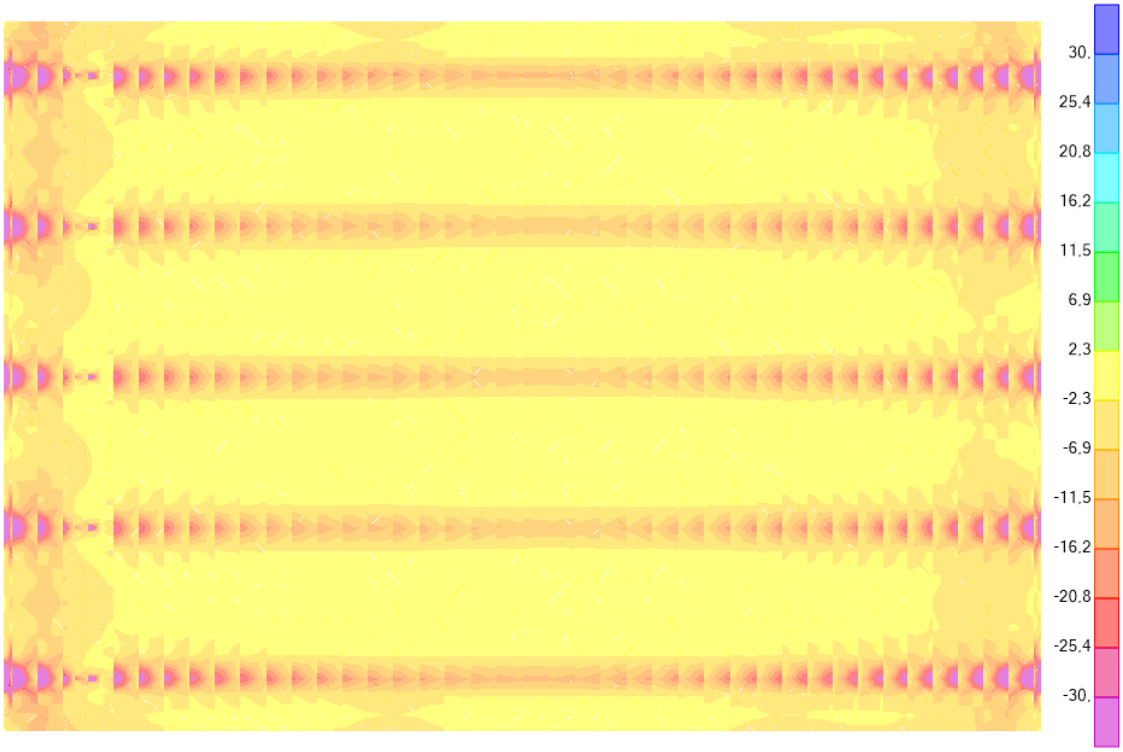


Figura 4.56 – Esforços cortantes verticais (kN/m) atuantes na laje do tabuleiro para carga permanente

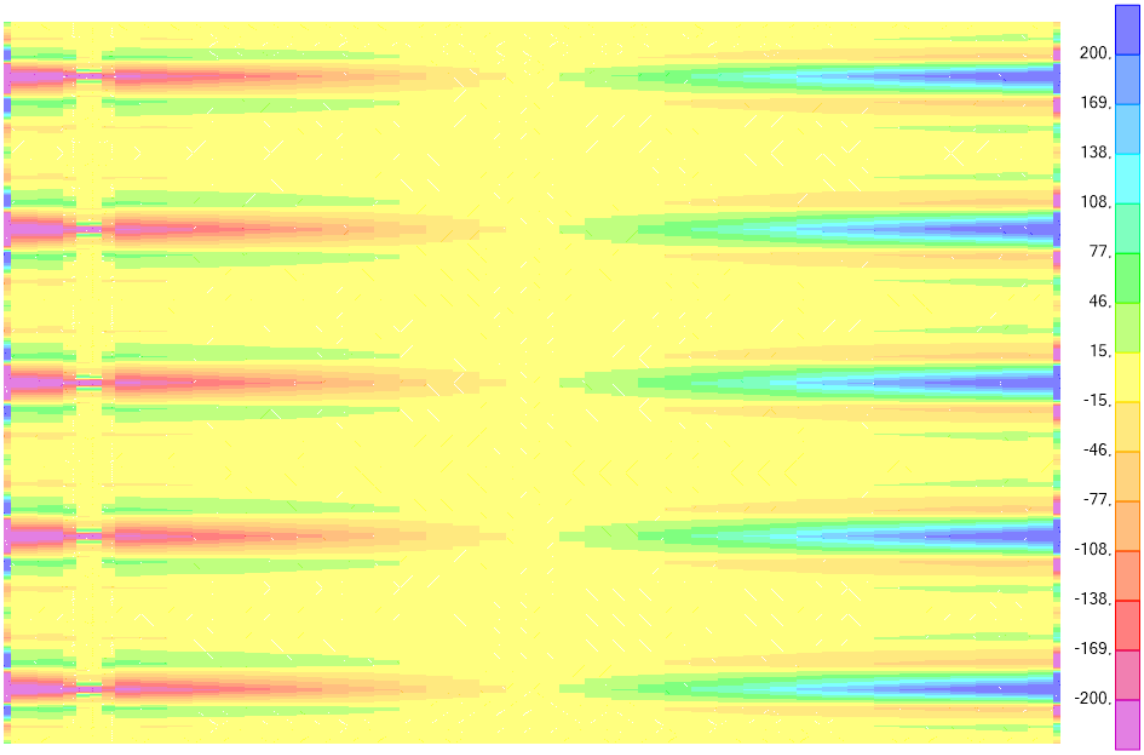


Figura 4.57 – Esforços cortantes verticais (kN/m) atuantes na laje do tabuleiro na envoltória de máximos para o caso de carga do veículo TB-450

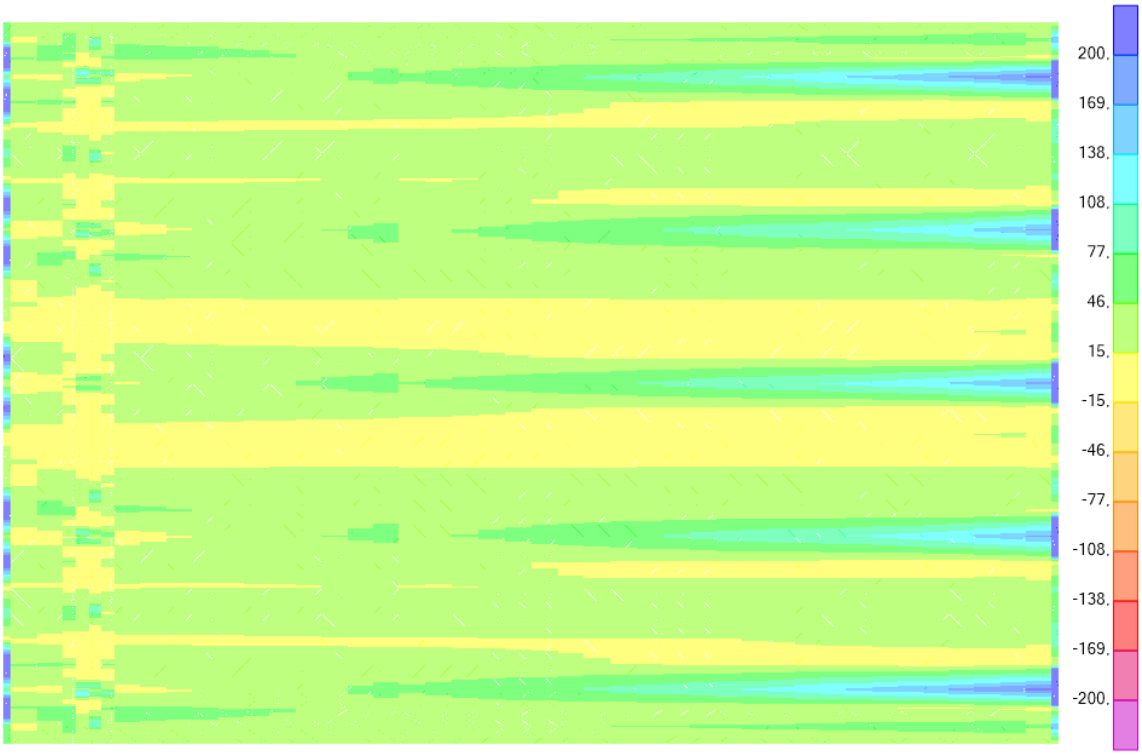
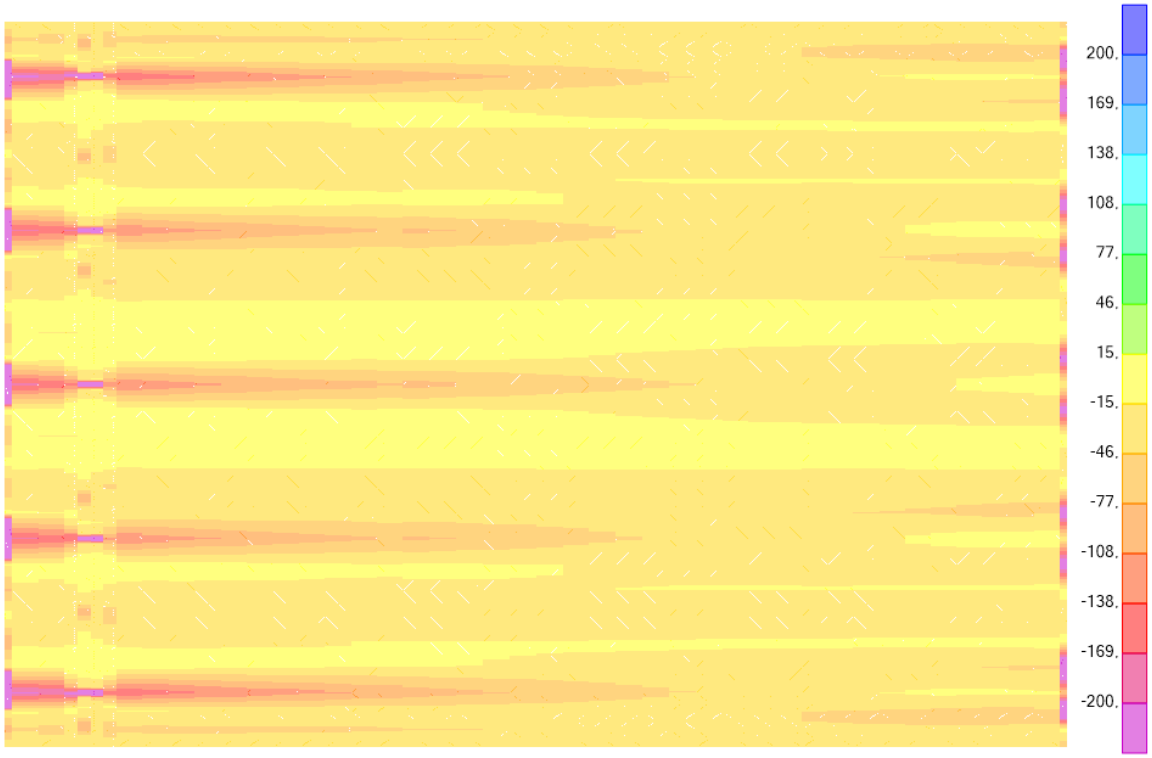


Figura 4.58 – Esforços cortantes verticais (kN/m) atuantes na laje do tabuleiro na envoltória de mínimos para o caso de carga do veículo TB-450



Na Tabela 4.3 apresentam-se os valores alimentados na planilha de dimensionamento da laje do tabuleiro da tipologia PPC-A-20-P.

Tabela 4.3 – Tabela de esforços solicitantes da laje do tabuleiro alimentada pelo programa "*main.py*" para a tipologia PPC-A-20-P

Mpp centro vão	Mgy[1] (kNm/m)	9,47
	Mgx[2] (kNm/m)	21,01
M carga móvel no centro do vão	Mqy+ (kNm/m)	31,71
	Mqy- (kNm/m)	-28,61
M carga móvel no centro do vão	Mqx+ (kNm/m)	28,10
	Mqx- (kNm/m)	-15,60
Mpp balanço	Mgy (kNm/m)	-10,17
M carga móv. Balanço	Mqy+ (kNm/m)	12,38
	Mqy- (kNm/m)	-36,68
M impacto na defensa (efeito no balanço)	Fimp (kN)	100,00
	himp (m)	1,01
	Midy (kNm/m)	-40,08
V permanente S1	Vg1 (kN/m)	28,30
V permanente S2	Vg2 (kN/m)	23,35
V variável S1	Vq1+ (kN/m)	9,81
V variável S2	Vq2+ (kN/m)	22,19

Legenda: 1 –direção transversal, 2 – direção longitudinal

4.2.4 Considerações

A metodologia implementada para a extração automatizada dos esforços estruturais demonstrou desempenho robusto e tecnicamente consistente ao longo das diferentes tipologias analisadas. O programa desenvolvido apresentou-se eficiente na leitura direta das tabelas de resultados geradas pelo CSiBridge, viabilizando a obtenção precisa dos esforços atuantes nas longarinas, conforme os casos de carga estabelecidos.

No caso das longarinas pré-moldadas e protendidas, a extração dos esforços foi realizada por meio de rotinas de interpolação aplicadas às seções de interesse (L/8, L/4, 3L/8 e L/2), uma vez que essas posições, em geral, não coincidiam com os nós da malha de elementos finitos. A estratégia adotada utilizou os nós imediatamente anterior e posterior a cada seção para os momentos fletores e, no caso dos esforços cortantes e de torção, priorizou o valor correspondente ao início do intervalo, onde tais

esforços tendem a ser de maior módulo. Essa interpolação se mostrou satisfatória e coerente com os resultados esperados.

Por outro lado, no caso das longarinas metálicas, a utilização do recurso "*User Discretization Point*" se mostrou superior em termos de precisão. Ao permitir a definição exata das seções onde os esforços devem ser extraídos, esse recurso fez com que o CSiBridge ajustasse automaticamente a malha de modo que os nós coincidissem com os pontos de interesse, eliminando a necessidade de interpolação. Como resultado, obtiveram-se respostas diretas e mais precisas para os esforços.

No que diz respeito à análise dos esforços atuantes na laje do tabuleiro, os valores foram extraídos diretamente dos nós dos elementos de área, uma vez que o recurso de suavização de tensões está disponível apenas na interface gráfica do CSiBridge. Essa limitação impôs uma superestimativa dos esforços extraídos, já que os resultados nodais, por sua natureza pontual e discreta, tendem a apresentar picos que não representam com fidelidade a distribuição média dos esforços ao longo da superfície. Com isso, evidencia-se uma limitação inerente ao processo automatizado de extração, embora não comprometa a viabilidade da metodologia proposta para fins de dimensionamento conservador.

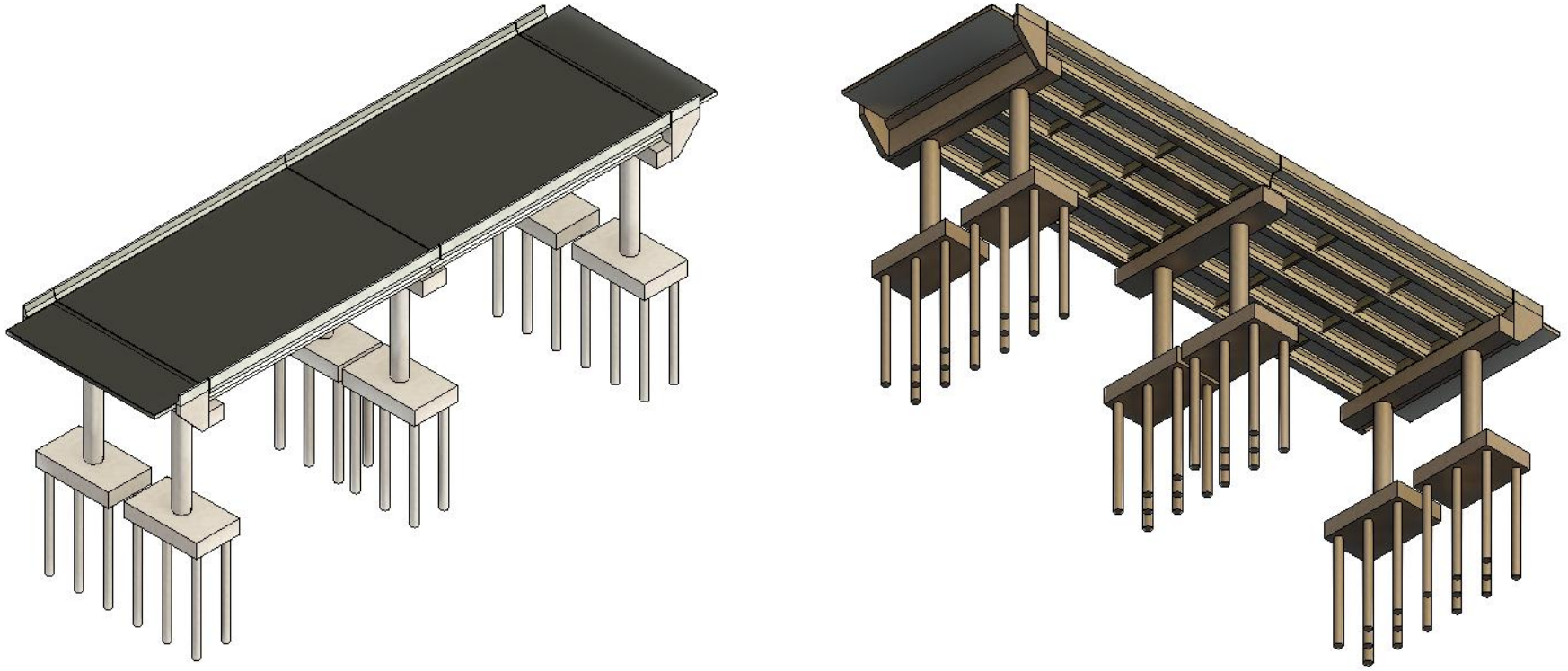
Assim, pode-se concluir que a metodologia adotada para a extração dos esforços demonstrou-se tecnicamente adequada e compatível com os objetivos da pesquisa. A interpolação aplicada às longarinas pré-moldadas forneceu estimativas confiáveis mesmo quando as seções de interesse não coincidiam com os nós da malha, enquanto a utilização do recurso "User Discretization Point" nas longarinas metálicas permitiu obter resultados mais precisos ao alinhar diretamente os nós com as seções críticas, eliminando a necessidade de interpolação.

No caso das lajes, embora a extração tenha sido realizada diretamente nos nós dos elementos de área, em razão da impossibilidade de acessar a suavização de tensões via API, reconhece-se que essa abordagem conduz a uma superestimativa dos esforços, já que os valores nodais tendem a concentrar picos locais. Ainda assim, os resultados extraídos se mostraram úteis para fins de dimensionamento conservador, e o programa atendeu de forma satisfatória à proposta de automatizar a coleta dos dados necessários ao projeto da superestrutura.

4.3 Modelagem no Revit através do Dynamo

Com a execução do grafo desenvolvido no Dynamo (Figura 3.32) para a tipologia PPC-A-20-P, foi possível gerar automaticamente a geometria da ponte correspondente. Esse grafo foi concebido para permitir a modelagem paramétrica de diferentes tipologias de pontes, a partir da variação de parâmetros como comprimento do vão, número de vãos, largura, entre outros. Da Figura 4.59 até a Figura 4.63 apresenta-se a forma final da ponte gerada, evidenciando a flexibilidade e eficiência do grafo na criação de modelos adaptáveis às especificações de projeto.

Figura 4.59 - Vista isométrica (a) superior e (b) inferior do modelo gerado a partir do grafo desenvolvido e dos dados de *input* para a tipologia PPC-A-20-P



(a) **(b)**

Figura 4.60 – Corte longitudinal do modelo gerado a partir do grafo desenvolvido e dos dados de *input* para a tipologia PPC-A-20-P

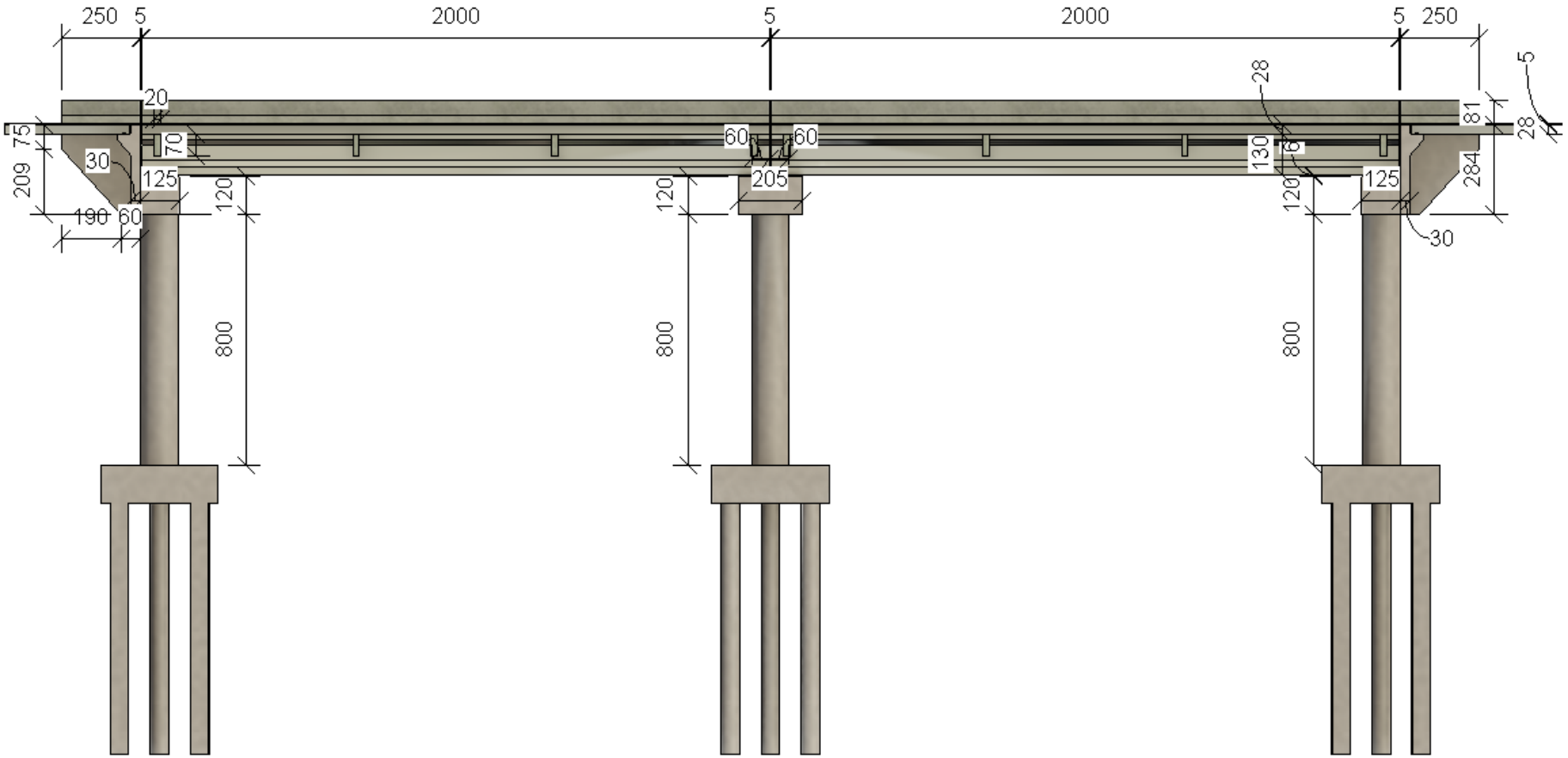


Figura 4.61 – Corte transversal do modelo gerado nos apoios intermediários a partir do grafo desenvolvido e dos dados de *input* para a tipologia PPC-A-20-P

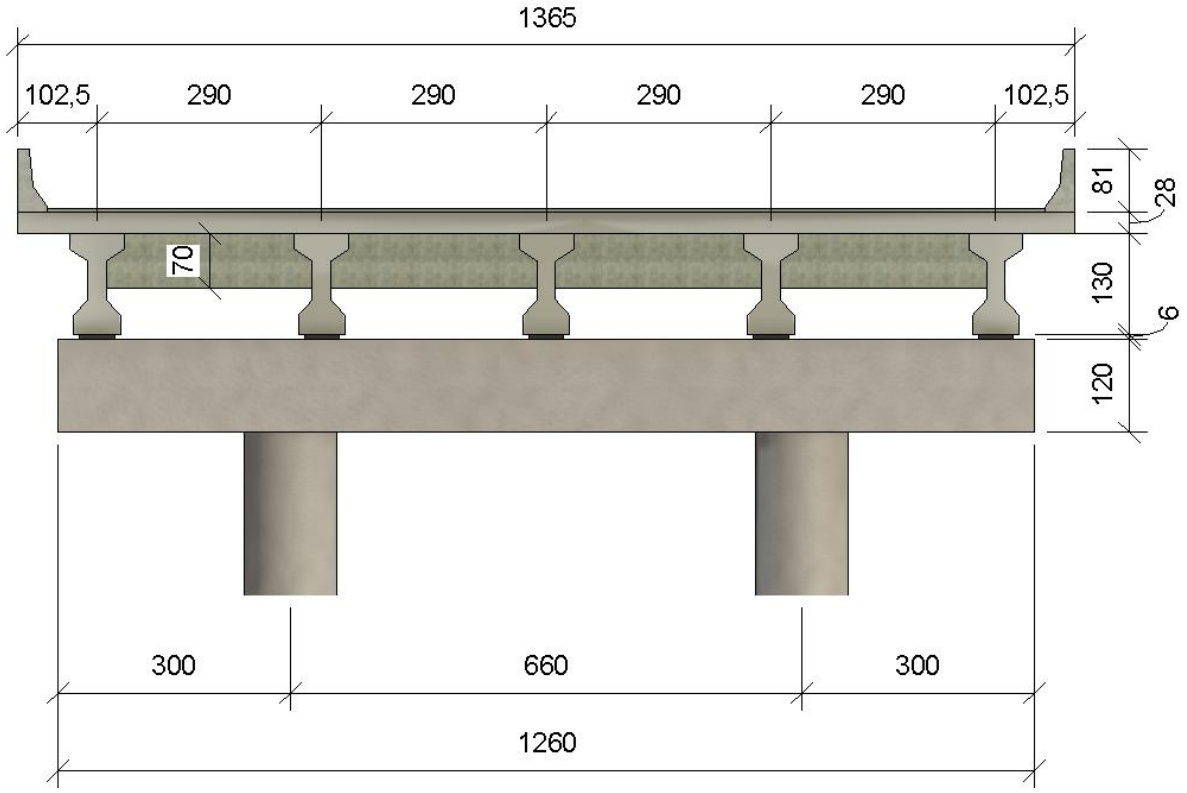


Figura 4.62 – Corte transversal do modelo gerado nos apoios extremos a partir do grafo desenvolvido e dos dados de *input* para a tipologia PPC-A-20-P

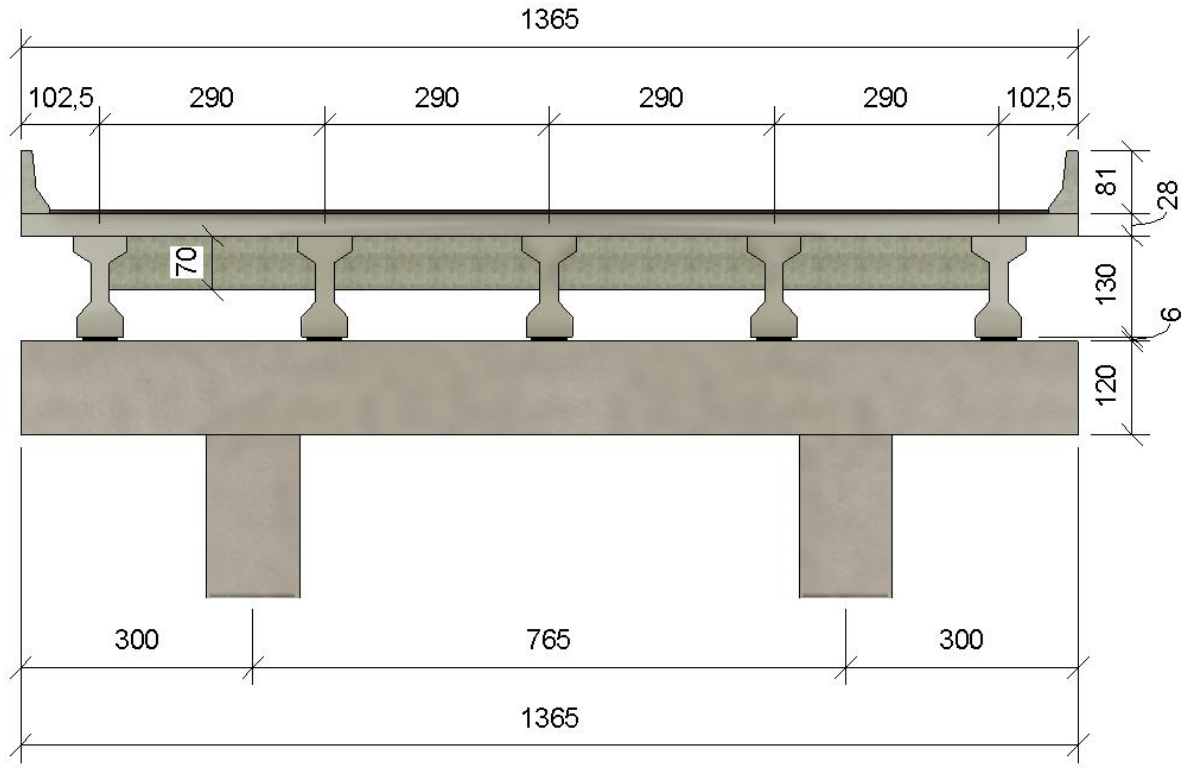
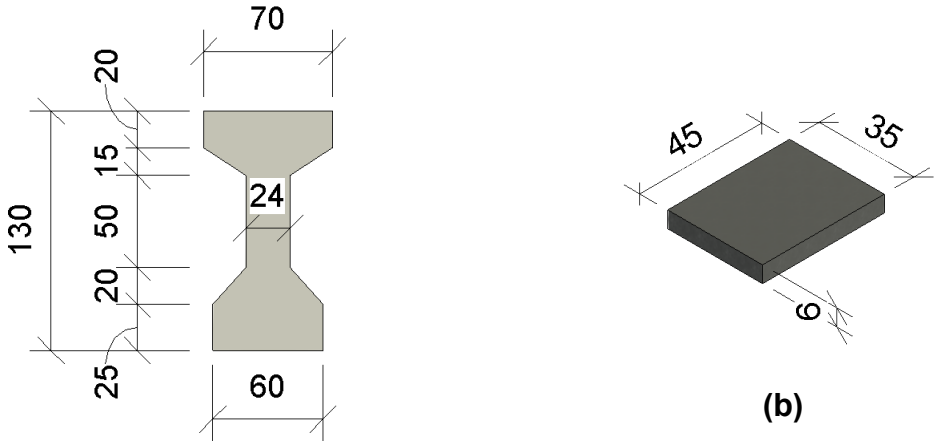


Figura 4.63 – Dimensões (a) das longarinas e (b) aparelhos de apoio do modelo gerado a partir do grafo desenvolvido e dos dados de *input* para a tipologia PPC-A-20-P



4.4 Dificuldades associadas

4.4.1 Conexão API com o CSiBridge via Python

Entre os desafios enfrentados, destaca-se a complexidade inicial de compreensão da conexão com o CSiBridge via API e o esforço requerido para depurar os códigos durante a fase de desenvolvimento. No entanto, essas dificuldades foram mitigadas por uma abordagem iterativa que incorporou melhorias progressivas com base no *feedback* dos testes.

Na etapa de modelagem numérica, a representação dos encontros das pontes exigiu um tratamento específico, realizado manualmente. Como não havia uma ferramenta automatizada previamente implementada para essa parte da estrutura, foi necessário desenvolver rotinas dedicadas para sua modelagem e geração de malha de elementos finitos, garantindo a compatibilidade com os demais elementos do modelo. Esse processo demandou um refinamento criterioso da malha e atenção especial aos vínculos e condições de contorno, de forma a garantir que a resposta estrutural do modelo representasse adequadamente o comportamento real dos encontros na ponte.

Algumas funcionalidades nativas do CSiBridge, como a inserção de dados referentes às curvas de retração, fluência e ganho de resistência do concreto, não puderam ser implementadas diretamente por meio dos métodos previstos na

documentação. Dessa forma, recorreu-se às *Interactive Database Tables*, já descritas anteriormente, para a definição e atribuição dessas propriedades. Ela também foi utilizada para a definição da ligação entre as vigas longarinas metálicas nas seções mistas de aço e concreto, que foram modeladas perfil a perfil.

Adicionalmente, identificou-se a existência de um método relacionado ao cronograma de construção acessível via API, mas ausente na documentação. Diante da dificuldade em localizar informações sobre sua definição e uso, recorreu-se a tentativas e erros. Essa situação evidencia a importância de uma documentação abrangente e atualizada, que inclua todos os recursos efetivamente disponíveis na API, a fim de garantir maior transparência e facilitar o desenvolvimento de soluções automatizadas.

4.4.2 Conexão com o Revit via Dynamo

Durante o desenvolvimento e execução do grafo no ambiente de programação visual do Dynamo, algumas dificuldades foram observadas, especialmente relacionadas ao rastreamento de variáveis e à depuração do código. À medida que o grafo se tornava mais complexo e extenso, o acompanhamento do fluxo de dados entre os nós tornava-se mais difícil, o que impactava diretamente na identificação de inconsistências e contribuía para a lentidão em determinadas etapas de execução.

Além disso, a depuração apresentava limitações, uma vez que o Dynamo executa todo o grafo simultaneamente, dificultando a localização precisa de erros. Em alguns casos, o código apresentava falhas sem mensagens claras, e bastava reiniciar o ambiente para que os erros desaparecessem, o que indicava certa instabilidade e tornava o processo de diagnóstico menos confiável. Essas dificuldades reforçam a necessidade de boas práticas na organização do grafo e na documentação das variáveis, a fim de mitigar os desafios inerentes à programação visual em ambientes complexos.

Ainda, foi constatada uma limitação nas funcionalidades oferecidas pelos nós nativos da plataforma. Embora o Dynamo possibilite a automação de diversas tarefas no Revit, nem todas as operações disponíveis na interface do *software* podem ser executadas diretamente pelos seus nós padrões. Essa limitação impôs desafios específicos à modelagem, especialmente em situações que exigiam maior controle sobre operações geométricas mais refinadas. Para contornar essas restrições, foram

incorporados *scripts* em Python dentro do próprio ambiente do Dynamo, o que possibilitou expandir suas capacidades.

Um exemplo foi a necessidade de desfazer automaticamente a união entre as vigas transversinas, uma operação fundamental para garantir a correta representação geométrica da superestrutura. Com o uso do Python, foi possível acessar funções da API do Revit não disponíveis por meio dos nós tradicionais.

5. CONCLUSÃO

5.1 Geral

Por meio da presente pesquisa, demonstrou-se, com rigor técnico e aplicação prática, a automatização e parametrização de projetos de pontes, ancorada no estado da arte das tecnologias disponíveis e na integração com *software* comerciais de uso disseminado. Apesar de, no domínio da infraestrutura, os programas oferecerem funcionalidades robustas, a interoperabilidade aberta presente neles ainda é limitada, exigindo estratégias adicionais para alcançar fluxos de trabalho integrados.

Nesse contexto, o presente trabalho visou a automatização de projetos de pontes utilizando a parametrização nos processos de criação de modelos numéricos e de modelos BIM a partir da mesma base de dados. O uso de API's mostrou-se como uma solução eficaz e viável para contornar a limitação de interoperabilidade, permitindo o intercâmbio automatizado de dados e a customização de processos que, de outra forma, demandariam alto esforço manual.

A automatização da modelagem numérica por meio da conexão via API com o CSiBridge, utilizando a linguagem Python, possibilitou o desenvolvimento de modelos em elementos finitos detalhados e precisos, servindo como base analítica consistente para análises de super e mesoestrutura de pontes. A integração direta e automatizada entre os resultados dessas análises e os procedimentos de dimensionamento segundo os critérios da norma brasileira constitui um diferencial que, até o momento, não é contemplado por nenhum *software* comercial disponível. A capacidade de realizar o dimensionamento com plena conformidade às exigências normativas nacionais, apoiada por uma infraestrutura computacional validada e automatizada, configura uma contribuição inédita e relevante para o desenvolvimento de projetos estruturais de pontes no Brasil.

O uso do Dynamo no ambiente do Revit demonstrou grande potencial na automatização e parametrização da modelagem geométrica, destacando-se como uma ferramenta eficiente na geração rápida e precisa de modelos baseados em um conjunto de parâmetros definidos. Essa característica o tornou particularmente aplicável à modelagem das tipologias presentes no catálogo digital de pontes, onde a repetição controlada de variações geométricas exigiu consistência e agilidade. A

programação visual no Dynamo permitiu a criação eficiente de modelos paramétricos e, por se tratar de uma mesma base de dados que alimenta tanto o modelo geométrico no Revit quanto o modelo analítico no CSiBridge, garantiu-se a coerência entre eles, facilitando-se eventuais atualizações simultânea nos modelos. No entanto, à medida que os problemas se tornam mais complexos, com geometrias caracterizadas por maior nível de complexidade, múltiplas condicionantes e interdependência lógica, a estrutura em nós dos grafos desenvolvidos no Dynamo tende a crescer exponencialmente, dificultando a manutenção, escalabilidade e interpretação dos fluxos. Nesses casos, o uso de programação textual, como a comunicação via API com o Revit utilizando-se Python, pode se tornar uma abordagem mais robusta e adequada, oferecendo maior controle, clareza e adaptabilidade para lidar com situações complexas de modelagem.

Dentro do escopo do catálogo digital, o presente trabalho englobou o estudo de 32 tipologias de pontes, sendo 16 referentes às pontes com longarinas pré-moldadas e protendidas e 16 com longarinas de aço e *stud bolts* para garantir a conexão com a laje do tabuleiro. Apesar da grande quantidade de comentários e partes que se repetem entre eles, os códigos somam 14050 linhas. Foram analisados ao todo 352 modelos em elementos finitos na aplicação do código visando a modelagem das pontes do catálogo.

Este estudo contribui para o avanço do estado da arte, estabelecendo uma base sólida para pesquisas futuras que possam integrar técnicas de aprendizado de máquina e algoritmos de otimização com modelagem paramétrica. Nesse contexto, o trabalho não apenas confirma a viabilidade da automatização como estratégia para modernizar e racionalizar os fluxos de projeto de pontes, mas também evidencia os limites das ferramentas atuais e a necessidade de combinações estratégicas entre programação visual e textual. O domínio dessas tecnologias e sua aplicação em um fluxo coordenado permitem ganhos expressivos em consistência, eficiência e padronização, aspectos essenciais para a evolução da infraestrutura nacional.

5.2 Sugestões para trabalhos futuros

Com base no método proposto e resultados alcançados, pode-se realizar refinamentos nos códigos e grafo desenvolvidos de forma a torná-los mais escaláveis e aplicáveis a um campo maior de soluções, como:

- Estruturar os códigos desenvolvidos com base nos princípios da POO, organizando-os em classes, métodos e funções reutilizáveis, de modo a otimizar a execução, facilitar a manutenção e permitir a integração em aplicações futuras. Essa abordagem permitiria uma melhor estruturação dos códigos, possibilitando englobar todas as funcionalidades desenvolvidas em um único código. Da mesma forma, recomenda-se modularizar o grafo desenvolvido no Dynamo, convertendo as operações recorrentes em nós personalizados, possibilitando a criação de pacotes de nós reutilizáveis e mais organizados;

- Implementar uma interface gráfica para o usuário ou desenvolver um *plugin* específico no ambiente do CSiBridge com o objetivo de tornar o uso dos códigos mais acessível e intuitivo, inclusive para usuários com menor familiaridade com programação;

- Expandir o escopo de aplicação do sistema para contemplar análises de infraestrutura, incluindo a modelagem automatizada de blocos de fundação, estacas e o lançamento de molas que representem o comportamento do solo com base em dados de campo provenientes de boletins de sondagem SPT (*Standard Penetration Test*);

- Ampliar a automatização do dimensionamento estrutural para outros componentes da ponte, como travessas, pilares, aparelhos de apoio, encontros e lajes de transição, além de elementos específicos de estruturas metálicas, como diafragmas, conectores, enrijecedores, soldas, dentre outros;

- Estender o estudo para outras tipologias estruturais de pontes, como pontes em arco, com seção transversal em caixão, estaiadas e pontes de concreto armado não protendido, bem como para geometrias não usuais, como pontes esconsas ou com inclinação longitudinal acentuada;

- Desenvolver funcionalidades no grafo do Dynamo voltadas ao detalhamento de armaduras e à geração de documentação com o intuito de produzir modelos BIM entregáveis completos. Além disso, incluir a modelagem de elementos complementares, como calçadas e passeios;
- Incorporar técnicas de aprendizado de máquina e outras metodologias de inteligência artificial na estrutura da programação desenvolvida, visando a otimização estrutural por meio de uma análise multiobjetivo dos parâmetros. Essa abordagem permitiria identificar soluções mais eficientes e reforçaria o potencial do sistema como ferramenta de apoio à decisão em projetos de pontes.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ABBASNEJAD, Behzad *et al.* Building Information Modelling (BIM) adoption and implementation enablers in AEC firms: a systematic literature review. **Architectural Engineering and Design Management**, v. 17, n. 5–6, p. 411–433, 16 jul. 2020.

ABDAL NOOR, Bashar; YI, Sirong. Review of BIM literature in construction industry and transportation: meta-analysis. **Construction Innovation**, v. 18, n. 4, p. 433–452, 6 nov. 2018.

ABDALLAH, Yasmine Ben. **Conception architecturale et modélisation paramétr**ique. Dissertação (Mestrado)—Toulouse: ENSA Toulouse, 11 jan. 2017.

AGBOBAN, Dago. **Nomination persistante dans un modèle paramétrique : identification non-ambigue¨ et appariement générique d'entités topologiques**. Tese (Doutorado)—Poitiers: Université de Poitiers, 2002.

ALSOLAI, Hadeel; ROPER, Marc. A systematic literature review of machine learning techniques for software maintainability prediction. **Information and Software Technology**, v. 119, n. C, p. 106214, 1 out. 2019.

ANTUNES, Maria Luisa Ribeiro *et al.* Analysis of IFC interoperability data schema for project representation. **Automation in Construction**, v. 166, p. 105650, 1 out. 2024.

ARELLANO, Hiram; TOLENTINO, Dante; GÓMEZ, Roberto. Optimum Criss Crossing Cables in Multi-span Cable-stayed Bridges using Genetic Algorithms. **KSCE Journal of Civil Engineering**, v. 23, n. 2, p. 719–728, 1 fev. 2019.

ARISHOLM, Erik; BRIAND, Lionel C.; FØYEN, Audun. Dynamic Coupling Measurement for Object-Oriented Software. **IEEE Transactions on Software Engineering**, v. 30, n. 8, p. 491–506, 31 ago. 2004.

ARTAR, Musa; CARBAS, Serdar. Optimizing the seismic resilience performance of steel truss bridges by maximum energy dissipation via friction dampers. **Structures**, v. 58, p. 105508, 1 dez. 2023.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NB-2**: Cálculo e Execução de Pontes de Concreto Armado. Rio de Janeiro, 1982.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 15965-1**. Sistema de classificação da informação da construção. Parte 1: Terminologia e estrutura. Rio de Janeiro: ABNT, 2011.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6118**. Projeto de estruturas de concreto — Procedimento. Rio de Janeiro, 2023a.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6123**: Forças devidas ao vento em edificações. Rio de Janeiro, 2023b.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 7187**: Projeto de pontes, viadutos e passarelas de concreto. Rio de Janeiro, 2021.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 7188**: Carga móvel rodoviária e de pedestres em pontes, viadutos, passarelas e outras estruturas. Rio de Janeiro, 2024.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 8681**: Ações e segurança nas estruturas — Procedimento. Rio de Janeiro, 2004.

BAZÁN, Ángela Moreno *et al.* New perspectives for bim usage in transportation infrastructure projects. **Applied Sciences**, v. 10, n. 20, p. 7072, 12 out. 2020.

BLOCH, Tanya; SACKS, Rafael. Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models. **Automation in Construction**, v. 91, p. 256–272, 1 jul. 2018.

BOOCH, Grady. Object-Oriented Development. **IEEE Transactions on Software Engineering**, v. 12, n. 2, p. 211–221, 28 fev. 1986.

BORETTI, Valentina *et al.* Algorithm-Aided Design for Composite Bridges. **Buildings**, v. 13, n. 4, p. 865, 26 mar. 2023.

BORRMANN, André *et al.* **Building Information Modeling: Technology Foundations and Industry Practice**. 1. ed. Cham: Springer Cham, 2018. v. 1

BOUGUESSA, Abdelhak; FORGUES, Daniel; DORÉ, Sylvie. La complémentarité entre le Building Information Modeling (BIM) et le Product LifeCycle Management (PLM) en passant par le Lean Construction (LC). In: BEDARD, C. *et al.* (org.). **Montreal: IEEE Computer Society**, 1 jun. 2013. Disponível em: <http://www.logistiqueconseil.org/Articles/Methodes-optimisation/3M-muda-mura-muri.htm>

BRADLEY, Alex *et al.* BIM for infrastructure: An overall review and constructor perspective. **Automation in Construction**, v. 71, p. 139–152, 1 nov. 2016.

BRASIL. **Decreto Nº 10.306 , de 02 de abril de 2020**. Estabelece a utilização do Building Information Modeling na execução direta ou indireta de obras e serviços de engenharia realizada pelos órgãos e pelas entidades da administração pública federal

[...]. Brasília, DF: Presidência da República, 2020a. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/decreto/D10306.htm. Acesso em: 23 mar.2025.

BRASIL. Departamento Nacional de Infraestrutura de Transportes. TED nº 703/2020-00. Apoio à implementação da metodologia BIM e desenvolvimento de metodologias e ferramentas para gestão de estruturas de Obras de Arte Especiais - OAEs. **Diário Oficial da União**: seção 3, Brasília, DF, suplemento n. 217, 2020b. Disponível em: https://www.gov.br/dnit/pt-br/acesso-a-informacao/convenios/termos-de-execucao-descentralizada/ted-no-703-2020-00. Acesso em: 22 mar. 2025.

BRASIL. **Lei Nº 14.133, de 01 de abril de 2021**. Lei de Licitações e Contratos Administrativos. Brasília, DF: Presidência da República, 2021. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2021/lei/l14133.htm. Acesso em: 23 mar.2025.

BUDD, Timothy. **An Introduction to Object-Oriented Programming**. 3. ed. Corvallis: Pearson, 2001. v. 1

BUILDINGSMART. **IFC Certification Participants**. Disponível em: < https:// https://technical.buildingsmart.org/services/certification/ifc-certification-participants/>. Acesso em: 9 mar. 2025e.

BUILDINGSMART. **IFC Formats**. Disponível em: < https://technical.buildingsmart.org/standards/ifc/ifc-formats/>. Acesso em: 9 mar. 2025b.

BUILDINGSMART. **IFC Implementation Guidance - buildingSMART Technical**. Disponível em: <https://technical.buildingsmart.org/resources/ifcimplementationguidance/>. Acesso em: 8 mar. 2025a.

BUILDINGSMART. **Introduction - IFC 4.3.2 Documentation**. Disponível em: <https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/content/introduction.htm>. Acesso em: 8 mar. 2025c.

BUILDINGSMART. **Model View Definitions (MVD) – buildingSMART Technical.** Disponível em: <https://technical.buildingsmart.org/standards/ifc/mvd/>. Acesso em: 9 mar. 2025d.

CAETANO, Inês; SANTOS, Luís; LEITÃO, António. Computational design in architecture: Defining parametric, generative, and algorithmic design. **Frontiers of Architectural Research**, v. 9, n. 2, p. 287–300, 1 jun. 2020.

CANNING, Peter S. *et al.* Interfaces for Strongly-Typed Object-Oriented Programming. **ACM SIGPLAN Notices**, v. 24, n. 10, p. 457–467, 1 set. 1989.

CARDELLI, Luca; WEGNER, Peter. On Understanding Types, Data Abstraction, and Polymorphism. **ACM Computing Surveys**, v. 17, n. 4, p. 471–523, 10 dez. 1985.

CEPA, Jorge Jerez *et al.* A Review on the Implementation of the BIM Methodology in the Operation Maintenance and Transport Infrastructure. **Applied Sciences**, v. 13, n. 5, p. 3176, 1 mar. 2023.

CHENG, Jack C. P.; LU, Qiqi; DENG, Yichuan. Analytical review and evaluation of civil information modeling. **Automation in Construction**, v. 67, p. 31–47, 1 jul. 2016.

CHOPRA, Dhurv *et al.* Proton: a transition layer API. **International Journal of Engineering Applied Sciences and Technology**, v. 5, n. 7, p. 303–307, nov. 2020.

CHUNG, Wonseok; SOTELINO, Elisa D. Three-dimensional finite element modeling of composite girder bridges. **Engineering Structures**, v. 28, n. 1, p. 63–71, 1 jan. 2006.

COELHO, Alvaro Carneiro. **Recomendações para otimização da superestrutura de pontes em concreto protendido e pontes mistas de aço e concreto**. Dissertação (Mestrado)—Viçosa: Universidade Federal de Viçosa, 19 dez. 2023.

COLLAO, Jorge *et al.* Bim visual programming tools applications in infrastructure projects: A state-of-the-art review. **Applied Sciences**, v. 11, n. 18, p. 8343, 8 set. 2021.

COSTIN, Aaron *et al.* Building Information Modeling (BIM) for transportation infrastructure – Literature review, applications, challenges, and recommendations. **Automation in Construction**, v. 94, p. 257–281, 1 out. 2018.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES. Diretoria de Desenvolvimento Tecnológico. Divisão de Capacitação Tecnológica. **Manual de Projeto de Obras-de-Arte Especiais** – Rio de Janeiro, 1996. 225p.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES. **Projeto Piloto versão 1.0: Superestrutura**. Produto M3.A2 do TED 00703/2020. Viçosa-MG: Universidade Federal de Viçosa, 2022.

FANG, Yunping *et al.* Scan-to-BIM-to-Sim: Automated reconstruction of digital and simulation models from point clouds with applications on bridges. **Results in Engineering**, v. 25, p. 104289, 1 mar. 2025.

GAN, Vincent J. L. *et al.* Parametric modelling and evolutionary optimization for cost-optimal and low-carbon design of high-rise reinforced concrete buildings. **Advanced Engineering Informatics**, v. 42, p. 100962, 1 out. 2019.

GARAGNANI, Simone; MANFERDINI, Anna Maria. Parametric accuracy: Building information modeling process applied to the cultural heritage preservation. In: **Copernicus**, 13 fev. 2013. Disponível em: <http://www.tcproject.net/pivotx/?e=149>

GARCÍA-SEGURA, Tatiana; YEPES, Víctor. Multiobjective optimization of post-tensioned concrete box-girder road bridges considering cost, CO2 emissions, and safety. **Engineering Structures**, v. 125, p. 325–336, 15 out. 2016.

GARDNER, Noel; ZHAO, John. Creep and Shrinkage Revisited. **Materials Journal**, v. 90, n. 3, p. 236–246, 5 jan. 1993.

GEORGOULA, Vasiliki. **Development of an Autodesk Revit Add-in for the Parametric Modeling of Bridge Abutments for BIM in Infrastructure**. Dissertação (Mestrado)—Munique: Technical University Of Munich, 11 mar. 2019.

GIRARDET, Alexis; BOTON, Conrad. A parametric BIM approach to foster bridge project design and analysis. **Automation in Construction**, v. 126, p. 103679, 1 jun. 2021.

HONGHONG, Song *et al.* Digital twin enhanced BIM to shape full life cycle digital transformation for bridge engineering. **Automation in Construction**, v. 147, p. 104736, 1 mar. 2023.

ISO. **ISO 10303-11**: Industrial automation systems and integration — Product data representation and exchange. Part 11: Description methods: The EXPRESS language reference manual. 2 ed. [S. L.], 2004. 255 p.

ISO. **ISO 10303-31**: Industrial automation systems and integration — Product data representation and exchange. Part 31: Conformance testing methodology and framework: General concepts. 1 ed. [S. L.], 1994. 39 p.

ISO. **ISO 16739**: Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. 1 ed. [S. L.], 2013. 23 p.

ISO. **ISO 16739-1**: Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. Part 1: Data schema. 2 ed. [S. L.], 2024. 496 p.

ISO. **ISO 19650-1**: Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling. Part 1: Concepts and principles. 1 ed. [S. L.], 2018. 34 p.

KHOSHAMADI, Negin *et al.* “Parametric and generative mechanisms for infrastructure projects”. **Automation in Construction**, v. 154, p. 104968, 1 out. 2023.

MAESTRO, M. O BIM além do decreto: normativas sobre BIM em obras públicas. Disponível em: <https://blog.altoqi.com.br/o-bim-alem-do-decreto-normativas-sobre-bim-em-obras-publicas>. Acesso em: 23 mar. 2025.

MENG, Wei *et al.* CBR-RBR fusion based parametric rapid construction method of bridge BIM model. **Advanced Engineering Informatics**, v. 57, n. C, p. 102086, 1 ago. 2023.

MINAS GERAIS. **Decreto Nº 48.146, de 02 de março de 2021**. Dispõe sobre a Estratégia estadual de disseminação do Building Information Modelling – Estratégia BIM-MG e institui o Comitê Gestor da Estratégia BIM-MG. Belo Horizonte, MG: Governador do Estado de Minas Gerais, 2021. Disponível em: https://www.almg.gov.br/legislacao-mineira/texto/DEC/48146/2021/. Acesso em: 23 mar.2025.

MONPERRUS, Martin *et al.* What should developers be aware of? An empirical study on the directives of API documentation. **Empirical Software Engineering**, v. 17, n. 6, p. 703–737, 3 dez. 2011.

OBERGRIESSER, Mathias. **Digitale Werkzeuge zur integrierten Infrastruktur-bauwerksplanung**. 1. ed. Ratisbona: Springer, 2017. v. 1

PARANÁ. **Decreto Nº 10.086, de 17 de janeiro de 2022**. Regulamenta, no âmbito da Administração Pública estadual, direta, autárquica e fundacional do Estado do Paraná, a Lei Nº 14133/2021, que estabelece normas gerais de licitação e contratação para as Administrações Públicas diretas, autárquicas e fundacionais da União, dos Estados, do Distrito Federal e dos Municípios [...]. Curitiba, PR: Governador do Estado do Paraná, 2022. Disponível em: https://www.legisweb.com.br/legislacao/?id=426484. Acesso em: 23 mar.2025.

PIRDAVANI, Ali *et al.* Application of building information modeling (BIM) for transportation infrastructure: a scoping review. In: **Zagreb**: Elsevier, 8 dez. 2023.

POKKUNURI, Bhanu Prasad. Object Oriented Programming. **ACM SIGPLAN Notices**, v. 24, n. 11, p. 96–101, 1 nov. 1989.

RIO DE JANEIRO. **Decreto Nº 46.571, de 24 de outubro de 2018**. Institui sobre a Estratégia estadual de disseminação do Building Information Modelling – BIM. Rio de Janeiro, RJ: Governador do Estado do Rio de Janeiro, 2018. Disponível em: https://leisestaduais.com.br/rj/decreto-n-46471-2018-rio-de-janeiro-institui-a-estrategia-estadual-de-disseminacao-do-building-information-modelling-bim. Acesso em: 23 mar.2025.

RIO GRANDE DO SUL. **Decreto Nº 56.311, de 12 de janeiro de 2022**. Institui a Estratégia Estadual de Fomento e Implantação do Building Information Modeling - BIM. Porto Alegre, RS: Governador do Estado do Rio Grande do Sul, 2022. Disponível em: https://leisestaduais.com.br/rs/decreto-n-56311-2022-rio-grande-do-sul-institui-a-estrategia-estadual-de-fomento-e-implantacao-do-building-information-modeling-bim. Acesso em: 23 mar.2025.

SACKS, Rafael *et al.* **BIM Handbook**. 3. ed. Nova Jersey: John Wiley & Sons, 2018. v. 1

SALZANO, Antonio *et al.* Systematic Literature Review of Open Infrastructure BIM. **Buildings**, v. 13, n. 7, p. 1593, 23 jun. 2023.

SANTA CATARINA. **Decreto Nº 1.370, de 13 de julho de 2021**. Institui a Estratégia Estadual de Implantação e Disseminação do Building Information Modelling em Santa Catarina (Estratégia BIM SC) e o Comitê Técnico da Estratégia BIM SC (CT-BIM SC). Florianópolis, SC: Governador do Estado de Santa Catarina, 2021. Disponível em: https://www.legisweb.com.br/legislacao/?id=417301#:~:text=Institui%20a%20Estratégia%20Estadual%20de,(CT-BIM%20SC). Acesso em: 23 mar.2025.

SARDONE, Laura *et al.* Performative Structural Design Optimization: Generative Algorithm for a Preliminary Study of a Voided Beam. **Applied Sciences**, v. 12, n. 17, p. 8663, 29 ago. 2022.

SHOU, Wenchi *et al.* A Comparative Review of Building Information Modelling Implementation in Building and Infrastructure Industries. **Archives of Computational Methods in Engineering**, v. 22, n. 2, p. 291–308, 1 ago. 2014.

SNYDER, Alan. Encapsulation and Inheritance in Object-Orlented Programming Languages. **ACM SIGPLAN Notices**, v. 21, n. 11, p. 38–45, 1 jun. 1986.

SÖBKE, Heinrich *et al.* An IFC schema extension for BIM-based description of wastewater treatment plants. **Automation in Construction**, v. 129, p. 103777, 1 set. 2021.

SZALAI, Dóra; AJTAYNÉ KÁROLYFI, Kitti. Integration of structural designers' workflows into BIM. **Journal of Building Engineering**, v. 98, p. 111141, 1 dez. 2024.

TANG, Fanlong *et al.* Integrating three-dimensional road design and pavement structure analysis based on BIM. **Automation in Construction**, v. 113, p. 103152, 1 maio 2020.

VALINEJADSHOUBI, Mojtaba *et al.* Automated system for high-accuracy quantity takeoff using BIM. **Automation in Construction**, v. 157, p. 105155, 1 jan. 2024.

WOODBURY, Robert. **Elements of parametric design**. 1. ed. Londres: Routledge, 2010. v. 1

YILMAZ, Ramazan; KARAOGLAN YILMAZ, Fatma Gizem. Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. **Computers in Human Behavior: Artificial Humans**, v. 1, n. 2, p. 100005, 17 jul. 2023.

ANEXO A

Planilha de dimensionamento de longarina pré-moldada protendida

O anexo A contém a planilha de dimensionamento referente às longarinas das pontes pré-moldadas e protendidas de concreto (PPC), desenvolvida dentro do grupo de pesquisa conforme produto do Relatório Técnico M3.A2 (DNIT, 2022). Os dados preenchidos são da tipologia PPC-A-20-P, composta por 5 longarinas, espaçamento entre eixos de longarinas de 290 cm e balanço de 102,5 cm. A altura da longarina adotada é de 130 cm, e a espessura da laje do tabuleiro é de 28 cm. Foram apresentadas na Figura A.1 as dimensões da seção transversal, enquanto as dimensões estipuladas da longarina estão na Figura A.2.

Figura A.1 – Dimensões em cm da seção transversal do tabuleiro da tipologia PPC-A-20-P

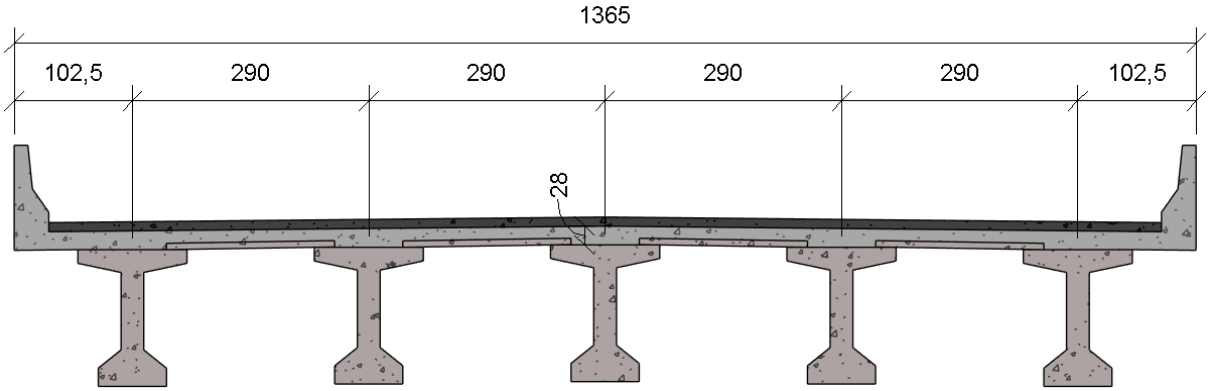
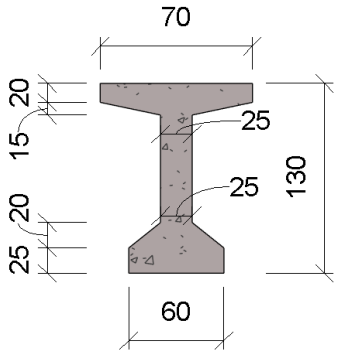


Figura A.2 – Dimensões em cm da seção transversal da longarina da tipologia PPC-A-20-P



Planilha de dimensionamento de longarina pré-moldada protendida

A 1 PROPRIEDADES DOS MATERAIS

A 1.1 CONCRETO

A 1.1.1 IDADE SUPERIOR A 28 DIAS

ABNT NBR 6118:2014	CAAmbiental	CAA II
Itens: 6.4; 7.4, 8.2, 12.3 e 17.3	fck (MPa)	35
$f_{ct,m} = 0,3\ f_{ck}^{2/3}$ $f_{ctk,inf} = 0,7\ f_{ct,m}$ $f_{ctk,sup} = 1,3\ f_{ct,m}$ $E_{ci} = \alpha_E \cdot 5600\ \sqrt{f_{ck}}$ $E_{cs} = \alpha_i \cdot E_{ci}$ $\alpha_i = 0,8 + 0,2 \cdot \frac{f_{ck}}{80} \leq 1,0$ $\alpha_E = 1,0$ para granito e gnaisse $\alpha = 1,2$ para seções T $\sigma_c = 0,85\ f_{cd}\left[1-\left(1-\frac{\varepsilon_c}{\varepsilon_{c2}}\right)^n\right]$ Para $f_{ck} \leq 50$ MPa: n=2 $G_c = \frac{E_{cs}}{2,4}$	fctk,m (MPa)	3,21
	fctk,inf (MPa)	2,247
	fctk,sup (MPa)	4,173
	s (CPV-ARI)	0,20
	α (Seções T)	1,20
	α . fctk,inf (MPa)	2,70
	α_E (agregado)	1,00
	Eci (GPa)	33,13
	αi (Seções T)	0,89
	Ecs (GPa)	29,40
	υ	0,20
	G (GPa)	12,25
	ec2	0,20%
	ecu	0,35%
Coeficiente de dilatação térmica	α (/°C)	0,00001

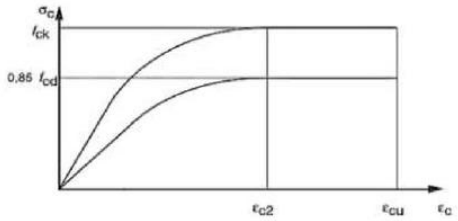


Figura A1 - Gráfico de tensão versus deformação para o concreto

A 1.1.2 IDADE INFERIOR A 28 DIAS

ABNT NBR 6118:2014	Idade j (dias)	14
Itens: 6.4; 7.4, 8.2, 12.3 e 17.3	s (CPV-ARI)	0,2
$f_{ckj} = \beta_1 \cdot f_{ck}$ $\beta_1 = \exp\{s\,[\,1 - (28/t)^{1/2}\,]\}$ $f_{ct,m} = 0,3\ f_{ck}^{2/3}$ $f_{ctk,inf} = 0,7\ f_{ct,m}$ $f_{ctk,sup} = 1,3\ f_{ct,m}$ $E_{ci}(t) = \left[\frac{f_c(t)}{f_c}\right]^{0,5} \cdot E_{ci}$	fckj (MPa)	32
	fctkj,m (MPa)	3,04
	fctkj,inf (MPa)	2,126
	fctkj,sup (MPa)	3,949
	α (Seções T)	1,2
	α . fctk,inf (MPa)	2,55
	α_E (agregado)	1,00
	Ecij (GPa)	31,79
	Ecsj (GPa)	27,02
	αi (Seções T)	0,9
	Ecs (GPa)	27,99

A 1.2 AÇO PRONTESÃO

ABNT NBR 6118:2014	Cordoalha	
Itens: 8.4; 9.6	Ep(GPa)	200,00
$\sigma_{pi}(Pós-tracionadas, RB) \leq \begin{Bmatrix} 0{,}74f_{ptk} \\ 0{,}82f_{pyk} \end{Bmatrix}$	αp28	6,04
	αpj	6,29
	fptk(MPa)	1900
$f_{pyk} = 0{,}90f_{ptk}$, para aços RB	fpyk(MPa)	1710
	σpi (MPa)	1463,0
$\alpha_p = \frac{E_p}{E_{ci}}$	Cordoalha	7 fios
	$\phi_{cordoalha}$ (mm)	12,7
ABNT NBR 7483:2020	Acordoalha (mm²)	100,90
Catálogo Freyssinet 2015	Modelo	12K13
	ϕ_{bainha} (mm)	65
	δ (m)	0,006
Deslizamento cunha Coeficientes de atrito	μ	0,24
	K	0,001
Deformação última	euk	3,50%

Figura A2 - Gráfico de tensão versus deformação para aço ativo

A 1.3 AÇO ARM. PASSIVA

ABNT NBR 6118:2014	CA-50	
Itens: 8.3	fyk (MPa)	500
ABNT NBR 7480:2007	Es (GPa)	210
$\alpha_s = \frac{E_s}{E_{ci}}$	αs28	6,34
	αsj	6,61
	CA-60	
	fyk (MPa)	600
	Es (GPa)	210
Coeficiente de ponderação da resistência	γs	1,15
	γs (Esp. Cons.)	1,15
	γs (Excep.)	1,00

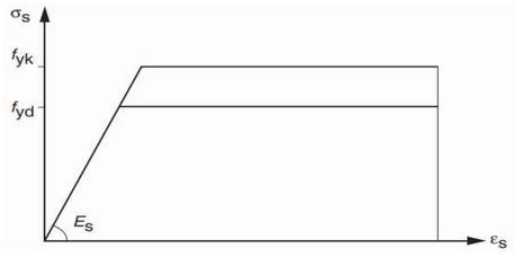


Figura A3 - Gráfico de tensão versus deformação para aço passivo

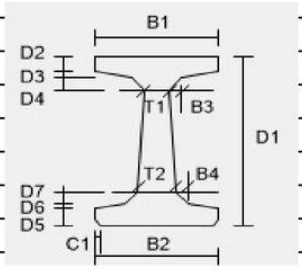
A 2 PROPRIEDADES GEOMETRICAS DAS SEÇÕES TRANSVERSAIS

A 2.1 VÃO DA VIGA

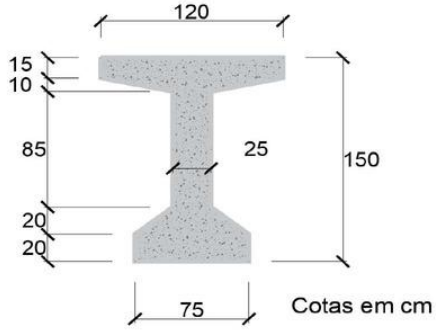
Comprimento da viga	L (m)	20
Espaçamento entre longarinas	a (m)	2,9
Largura colaborante	bf (m)	2,00
Espessura da laje	h (m)	0,28

A 2.2 PROPRIEDADES GEOMÉTRICAS

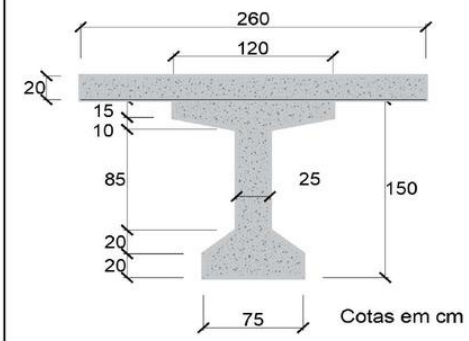
B1 (cm)	70
B2 (cm)	60
B3 (cm)	0
B4 (cm)	0
D1 (cm)	130
D2 (cm)	20
D3 (cm)	15
D4 (cm)	0
D5 (cm)	25
D6 (cm)	20
D7 (cm)	0
T1 (cm)	25
T2 (cm)	25
c1 (cm)	0



A 2.3 SEÇÃO INICIAL

Viga pré-moldada	Ac1 (m²)	0,57125
	Perímetro (m)	4,27
	hlong. (m)	1,30
	halma (m)	0,50
	yi1 (m) (-)	-0,65945
	ys1 (m)	0,64
	Ix1 (m4)	0,107227
	Wi1 (m³) (-)	-0,16
	Ws1 (m³)	0,17
	b_{sup} (m)	0,7
	Iy (m4)	0,0140
	m1 (m)	0,70
	t1 (m)	0,28
	m2 (m)	0,68
	t2 (m)	0,25
	m3 (m)	0,60
Timoshenko (1985) cap.5	t3 (m)	0,35
$J = \frac{1}{3} m t^3$	J1 (m4)	0,0049
	J2 (m4)	0,0035
$J = \frac{1}{3} \sum m_i t_i^3$	J3 (m4)	0,0086
	J (m4)	0,0169

A 2.2 SEÇÃO COMPOSTA

Viga pré-moldada + laje moldada in loco	Ac2 (m²)	1,13125
Os concretos devem ser os mesmos	Perímetro (m)	7,43234
	yi2 (m) (-)	-1,04584
	ys2 (m)	0,534158
	ys2* (m)	0,254158
	Ix2 (m4)	0,283176
	Wi2 (m³) (-)	-0,27
	Ws2 (m³)	0,53
	bw laje (m)	2,00
	t (m) (laje)	0,28
	t_{prel} (m)(pré-laje)	0,08
	$h_{laje+long.}$ (m)	1,58
	b mesa (m)	0,70
	bw alma (m)	0,25

A 3 TRAÇADO DO CABO EQUIVALENTE

A 3.1 ESPAÇAMENTOS VERTICAIS MÍNIMOS

ABNT NBR 6118:2014		Entre cabos	
Item 18.6	$a_v \geq \begin{cases} \emptyset_{bainha} \\ 5cm \end{cases}$	ϕ_{bainha} (mm)	65
		av (mm)	65
Cobrimento Ap		cmín (cm)	3
Diâmetro estribo		$\phi_{estribo}$ (cm)	12,5
d' protensão		d'p (mm)	130
Catálogo Freyssinet 2015		Entre ancoragens	
		E (cm)	27

A 3.2 EXCENTRICIDADES DOS CABOS DE PROTENSÃO

$$y(x) = \frac{t}{c^2}.x^2$$

$$\tan \propto = \frac{dy}{dx} \therefore \tan \propto (x) = \frac{2t}{c^2}.x$$

Seção	S1	S2	S3	S4	S5
x (m)	0	2,5	5	7,5	10
ep 1 cabo(m)	0	-0,232	-0,397	-0,5	-0,53
ep 2 cabo(m)	-0,386	-0,618	-0,783	-0,88	-0,92
e_{CSi} (m)	-0,92	-1,15	-1,32	-1,42	-1,45
d' ep (m)	0,6594	0,428	0,262	0,163	0,13
α	0,105	0,079	0,053	0,026	0,000

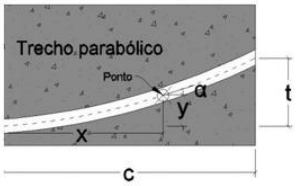


Figura A4 - Esquema de variáveis da parábolola

Figura A5 - Cabo equivalente (seção composta)

A 3.3 EXCENTRICIDADE DA ARMADURA PASSIVA (SEÇÃO COMPOSTA)

ABNT NBR 6118:2014	Seção	S1	S2	S3	S4	S5
Item 7.4	x (m)	0	2,5	5	7,5	10
Cmín As (mm)	e2As (m)	-0,976	-0,976	-0,976	-0,98	-0,98
40,0	d' (m)	0,07	0,07	0,07	0,07	0,07

A 4 ENGROSSAMENTO DA ALMA JUNTO AO APOIO

A 4.1 COMPRIMENTO ENGROSSAMENTO

Altura até 1/2 laje	hviga (m)	1,44
Catálogo Freyssinet (2015)	E (m)	0,27
	D (m)	0,255
Item 9.6.2 NBR 6118:2014	β (rad)	0,588
Inclinação inicial cabo	α (rad)	0,105
Dist. cabo até 1/2 laje	y (m)	0,645
Comprimento mínimo	x (m)	1,23
Comprimento adotado	a adot (m)	1,30

A 4.2 LARGURA DO ENGROSSAMENTO

Cat.Freyssinet	C (m)	0,17
Largura mínima	b mín (m)	0,34
Largura adotada	b adot (m)	0,40

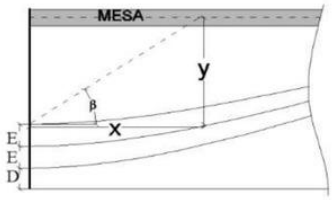


Figura A6 - Esquema das ditâncias x e y para regularização das tensões a partir da inclinação inicial do cabo

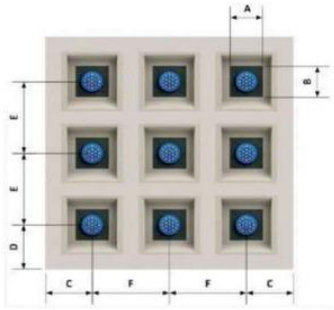


Figura A7 - Ditâncias mínimas ancoragens de acordo com o Catálogo Freyssinet (2015)

Modelo \ Dimensões	A	B	C	D	E	F
	mm	mm	mm	mm	mm	mm
1K13	100	100	80	120	140	140
2K13	120	120	90	135	150	150
4K13	160	120	110	165	180	160
6K13	160	190	120	180	195	225
7K13	160	190	135	203	195	225
9K13	200	200	150	225	230	230
12K13	240	240	170	255	270	270
15K13	270	270	195	293	300	300
19K13	300	300	210	315	330	330
27K13	350	350	235	353	395	395

Figura A8 - Dimensões das ancoragens de acordo com o Catálogo Freyssinet (2015)

A 5 ESFORÇOS SOLICITANTES EM CADA ETAPA CONSTRUTIVA

A 5.1 ETAPAS CONSTRUTIVAS

Etapa		Idade (dias)		Seção	Esforço em cada etapa
1	Protensão	j	14	inicial	Mg1, P0
2	Içamento	x	28	inicial	Mg1, Px
3	Concretagem laje + transv.	w (x+10)	38	inicial	Mg1, Mg2, Mqc,Pw
4	Revestimento +barreiras	z (=w+28)	66	composta	Mg1, Mg2, Mg3, Mqc,Pz
5	Início da operação da ponte	k (=z+28)	94	composta	Mg1, Mg2, Mg3, Mq+-, Pk
6	Final da operação da ponte	∞	∞	composta	Mg1, Mg2, Mg3, Mq+-, P∞

Observação: o içamento será feito com cabos na vertical, será menos crítico que a etapa 3

Mg1	Peso próprio da longarina
Mg2	Concretagem da laje e transversinas
Mg3	Pavimentação e defensas
Mqc	Sobrecarga de equipamentos e pessoas para concretagem da laje
Mq+	Trem tipo, valores máximos
Mq-	Trem tipo, valores mínimos

A 5.2 CARGAS DISTRIBUÍDAS NA LONGARINA

g1 (kN/m)	14,28
g2 (kN/m)	20,30
g3 (kN/m)	13,34
qc (kN/m)	5,46

A 5.3 ESFORÇOS SOLICITANTES SEM COEFICIENTE DE IMPACTO

Seção		S1	S2	S3	S4	S5
x (m)		0,0	2,5	5,0	7,5	10,0
Momentos fletores (kN.m)	Mg1 (kNm)	0,0	312,4	535,5	669,4	714,1
	Mg2* (kNm)	0,0	758,6	1311,7	1641,4	1757,8
	Mg2 (kNm)	0,0	446,2	776,2	972,0	1043,8
	Mg3 (kNm)	0,0	299,4	508,4	630,4	673,2
	Mqc (kNm)	0,0	119,4	204,8	255,9	273,0
	Mq+ (kNm)	0,0	779,0	1247,5	1523,7	1621,6
	Mq- (kNm)	0,0	-40,1	-82,1	-119,7	-112,3

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Força Cortante (kN)	V_{g1} (kN)	142,8	107,1	71,4	35,7
	V_{g2^*} (kN)	357,4	261,7	175,8	90,0
	V_{g2} (kN)	214,6	154,6	104,4	54,3
	V_{g3} (kN)	142,8	106,4	71,2	37,1
	V_{q+} (kN)	35,5	88,4	128,4	151,0
	V_{q-} (kN)	-395,4	-308,7	-243,1	-193,4

Momento Torçor (kNm)	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
	T_g (kN)	15,1	15,1	13,1	5,1
	T_{q+} (kN)	106,4	106,4	97,7	83,7
	T_{q-} (kN)	-106,4	-106,4	-97,7	-93,8

A 5.4 CÁLCULO DO COEFICIENTE DE IMPACTO - CI

ABNT NBR 7188:2013

$$CI = CIV \cdot CNF \cdot CIA$$

sendo

$$CIV = \begin{cases} 1{,}35 & se\ L \leq 10{,}0m \\ 1 + 1{,}06\left(\frac{20}{L+50}\right), & se\ 10 < L \leq 200m \end{cases}$$

$$CNF = 1 - 0{,}05(n-2) \geq 0{,}9$$

$$CIA = \begin{cases} 1{,}25, para\ obras\ em\ concreto\ ou\ mistas \\ 1{,}15, para\ obras\ em\ aço \end{cases}$$

Vão da ponte	L (m)	20
Número de faixas	n	2
Coef. Impacto Vertical	CIV	1,30
Coef. Núm. de faixas	CNF	1
Cosiderar CIA?	CIA	**Não**
Coef. Impacto Adicional	CIA	1
Coef. Impacto	CI	1,30

A 5.5 ESFORÇOS SOLICITANTES COM COEFICIENTE DE IMPACTO

Seção		S1	S2	S3	S4	S5
x (m)		0	2,5	5	7,5	10
Momentos fletores (kN.m)	Mg1 (kNm)	0,0	312,4	535,5	669,4	714,1
	Mg2 (kNm)	0,0	444,1	761,3	951,6	1015,0
	Mg3 (kNm)	0,0	291,8	500,2	625,2	666,9
	Mqc (kNm)	0,0	119,4	204,8	255,9	273,0
	Mq+ (kNm)	0,0	1014,9	1625,4	1985,2	2112,7
	Mq- (kNm)	0,0	-52,3	-107,0	-156,0	-146,4

Força Cortante (kN)	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
	V_{g1} (kN)	142,8	107,1	71,4	35,7
	V_{g2} (kN)	203,0	152,3	101,5	50,8
	V_{g3} (kN)	133,4	100,0	66,7	33,3
	V_{q+} (kN)	46,2	115,2	167,3	196,8
	V_{q-} (kN)	-515,1	-402,2	-316,8	-252,0

Momento Torçor (kNm)	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
	T_g (kN)	15,1	15,1	13,1	5,1
	T_{q+} (kN)	106,4	106,4	106,4	106,4
	T_{q-} (kN)	-106,4	-106,4	-106,4	-106,4

A 6 DETERMINAÇÃO DOS ESFORÇOS DE RETRAÇÃO DIFERENCIAL E VARIAÇÃO DE TEMPERATURA DA LAJE

A 6.1 DADOS

Área laje (mesa colaborante)	Acml (m²)	0,56
Área laje (concr. fresco)	Acml (m²)	0,3
Dist. De Nc ao centróide	z (m)	0,394
Módulo de elasticidade secante	Ecs (GPa)	29,40
Coeficiente de fluência (adotar)	ϕ (∞,z)	1,783
NBR 6118:2014 Item 11.4.2.1: var. não uniforme entre laje e viga	ΔT- (°C)	-5
	ΔT+ (°C)	7
Coeficiente de expansão térmica	α (/°C)	1,00E-05

A 6.2 ESFORÇOS DEVIDO À RETRAÇÃO DA LAJE

Modelo de Moga et al. (2019)

$$N_m = -N_c = \varepsilon_c \cdot E_c \cdot A_c$$

$$M_m = N_m \cdot z_{cm}$$

$$E_c = \frac{n_o}{n_S} E_{cm}$$

$$n_S = n_L$$

EN 1994-1-1:2004 Item 5.4.2.2.

$$n_L = n_0 \cdot (1 + \psi_L \cdot \varphi(t, t_0))$$

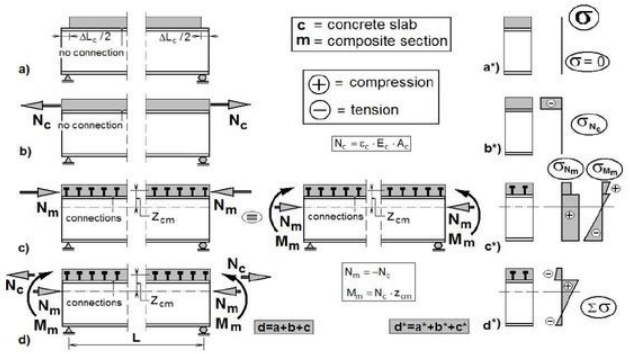


Figura A9 - Modelo de Moga et al. (2019) para consideração da retração

EN 1994-1-1:2004 Item 5.4.2.2.	ψ	0,55
Relação Ecs,viga / Ecs,laje	no	1
Coeficiente de homog. Mecânica	ns	1,981
Módulo com efeito da fluência	Ec (Gpa)	14,85
Retração da laje (adotar)	εc (∞,z)	7,04E-06
Normal tração laje	Nc (kN)	31,4
Normal compressão analítico	Nr (kN)	-31,4
Momento na longarina analítico	Mr(kNm)	12,4
Normal compressão MEF	Nr (kN)	-112,7
Momento na longarina MEF	Mr(kNm)	90,2
Normal na longarina adotado	Nr (kN)	-31,4
Momento na longarina adotado	Mr(kNm)	12,4

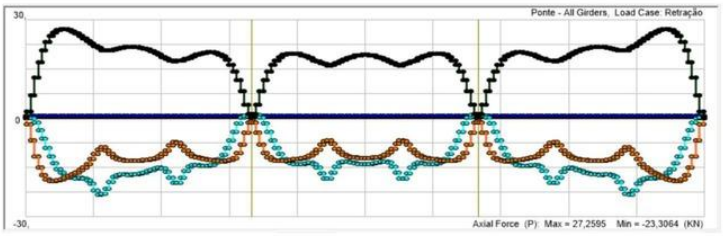


Figura A10 - Força normal nas longarinas devido à retração

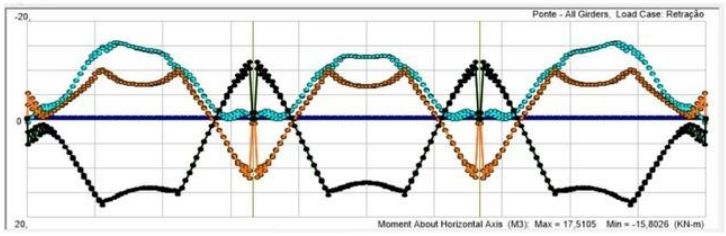


Figura A11 - Momentos fletores nas longarinas devido à retração

A 6.3 ESFORÇOS DEVIDO À RETRAÇÃO TÉRMICA DA LAJE

Modelo de Moga et al. (2019)

$$N_{c.\Delta T} = -N_{m.\Delta T} = -\varepsilon_{c.\Delta T} \cdot E_{cm} \cdot A_c$$

$$M_{m.\Delta T} = N_{m.\Delta T} \cdot z$$

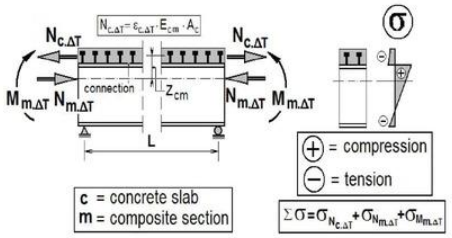


Figura A12 - Modelo de Moga et al. (2019) para consideração da variação de temperatura

A 6.4 ENCOLHIMENTO DA LAJE

Esforços devido à retração térmica	$\varepsilon c,\Delta T$	-0,00005
Normal tração laje	$Nc,\Delta T$ (kN)	823
Normal compressão analítico	$N_{\Delta T}$ (kN)	-823
Momento na longarina analítico	$M_{\Delta T}$(kNm)	324,5
Normal compressão MEF	$N_{\Delta T}$ (kN)	0
Momento na longarina MEF	$M_{\Delta T}$(kNm)	0,2
Normal na longarina adotado	$N_{\Delta T}$ (kN)	-823
Momento na longarina adotado	$M_{\Delta T}$(kNm)	324,5

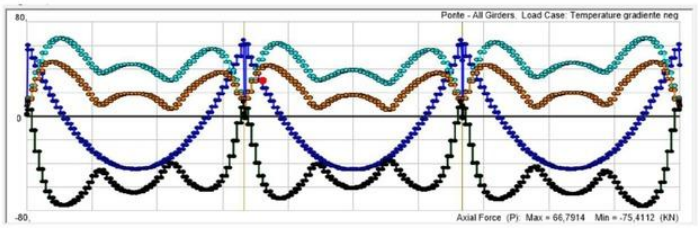


Figura A13 - Força normal nas longarinas devido à variação de temperatura gradiente negativa

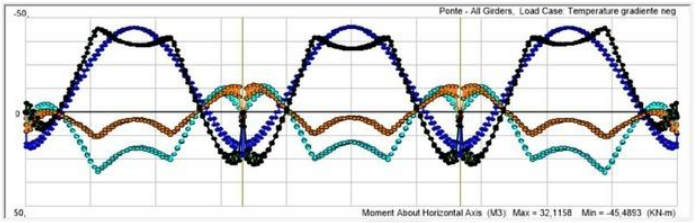


Figura A14 - Momentos fletores nas longarinas devido à variação de temperatura gradiente negativa

A 6.5 DILATAÇÃO DA LAJE

Esforços devido à dilatação térmica	$\varepsilon c,\Delta T$	0,00007
Normal compressão laje	$Nc,\Delta T$ (kN)	-1153
Normal compressão analítico	$N_{\Delta T}$ (kN)	1153
Momento na longarina analítico	$M_{\Delta T}$(kNm)	-454,3
Normal compressão MEF	$N_{\Delta T}$ (kN)	0,244
Momento na longarina MEF	$M_{\Delta T}$(kNm)	-0,2091
Normal na longarina adotado	$N_{\Delta T}$ (kN)	1153
Momento na longarina adotado	$M_{\Delta T}$(kNm)	-454,3

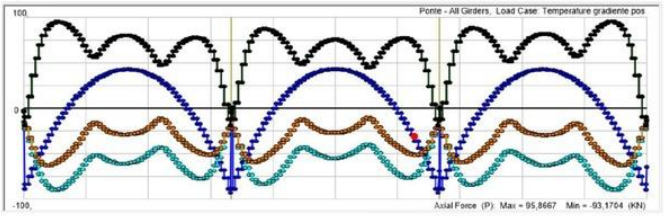


Figura A15 - Força normal nas longarinas devido à variação de temperatura gradiente positiva

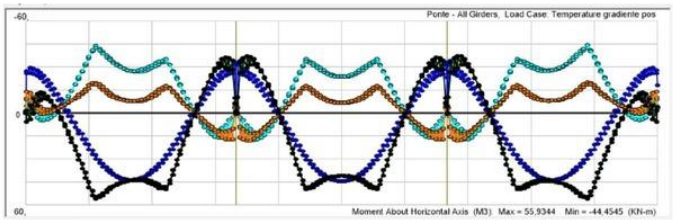


Figura A16 - Momentos fletores nas longarinas devido à variação de temperatura gradiente positiva

A 7 DETERMINAÇÃO DA FORÇA DE PROTENSÃO

A 7.1 ATO DA PROTENSÃO

NBR 6118:2014	σt_ato=	≤ 1,2.fctmj (MPa)	3,64	ELU - Ato	Borda superior viga
Item 17.2.4	σc_ato=	≤ 0,7*fckj (MPa)	-22,55	ELU - Ato	Borda inferior viga

A 7.2 ELS - PROTENSÃO LIMITADA

NBR 6118:2014	σt_serv=	≤ 1,2.fctk,inf (MPa)	2,70	ELS - F	Borda inferior viga
Item 13.4	σt_serv=	≤ 0 (MPa)	0,00	ELS - D	Borda inferior viga
CEB - FIB 1990	σc_serv=	≤ 0,6*fck (MPa)	-21,00	ELS - CE	Borda superior laje

A 7.3 COMBINAÇÃO DE AÇÕES

Combinação Frequente

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + \Psi_1 . F_{q1,k} + \sum_{j=2}^{n} \Psi_{2j} . F_{qj,k}$$

Combinação Quase-permanente

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + \sum_{j=1}^{n} \Psi_{2j} . F_{qj,k}$$

Combinação Rara

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + F_{q1,k} + \sum_{j=2}^{n} \Psi_{1j} . F_{qj,k}$$

A 7.4 ESTIMATIVA DAS PERDAS

Perdas imediatas	10%
Perdas progressivas	20%
Perdas totais	30%

A 7.5 ESTIMATIVA PERDA POR IDADE

Idade (dias)		% Prot. Estimada
j	14	0,90
z	66	0,83
k	94	0,80
∞	10000	0,70

A 7.6 FORÇA DE PROTENSÃO ESTIMADA POR CORDOALHA

Acordoalha (mm²)	100,90
σpi (MPa)	1463
Pi (kN)	147,6
Pj (kN)	132,9
Pz (kN)	122,5
Pk (kN)	118,1
P_∞ (kN)	103,3

A 7.7 CÁLCULO DAS TESÕES NA SEÇÃO COMPOSTA

$$\sigma = \boxed{\frac{P_z}{A_1}} - \frac{(P_z - P_\infty)}{A_2} \boxed{+\frac{P_z \cdot e_{p1}}{W_1}} - \frac{(P_z - P_\infty) \cdot e_{p2}}{W_2} \boxed{-\frac{M_{g1}}{W_1} - \frac{M_{g2}}{W_1}} - \frac{M_{g3}}{W_2} -$$
$$\frac{\psi_i M_q}{W_2} + \boxed{\frac{N_r}{A_2} - \frac{M_r}{W_2}} + \boxed{\frac{\psi_i N_{\Delta T}}{A_2} - \frac{\psi_i M_{\Delta T}}{W_2}} \leq \sigma_t \; ou \; \geq \sigma_c$$

Observações:
A idade z representa o momento de mudança da seção pré-moldada para A seção composta; Para o cálculo da de tensões na altura da laje, as parcelas emolduradas devem ser suprimidas; Os módulos de inércia W devem ser corresposdentes à fibra para o cálculo da tensão; Para o cálculo da de P∞ deve-se isolar o mesmo na equação.

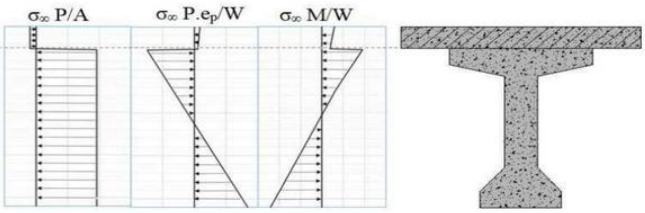


Figura A17 - Perfil de tensões ao longo da seção composta da longarina

A 7.8 CÁLCULO DA FORÇA DE PROTENSÃO

Estratégia: Calcula-se P∞ para ELS-D e ELS-F isolando P∞ na equação.
Depois faz-se as demais verificações

$$P_\infty \geq -\frac{\left(\sigma_t + \frac{M_{g1}}{W_1} + \frac{M_{g2}}{W_1} + \frac{M_{g3}}{W_2} + \frac{\psi_i M_q}{W_2} - \frac{N_r}{A_2} + \frac{M_r}{W_2} - \frac{\psi_i N_{\Delta T}}{A_2} + \frac{\psi_i M_{\Delta T}}{W_2}\right)}{\left(\frac{x}{A_1} - \frac{(x-1)}{A_2} + \frac{x e_{p1}}{W_1} - \frac{(x-1) e_{p2}}{W_2}\right)}$$

A 7.9.1 ELS- D NA BORDA INFERIOR DA VIGA

Comb. Quase Permanente

Comb. Quase. Perm.	Seção:	S5
Requisito	σt ≤ 0 (MPa)	0,00
Dados seção	Ac1 (m²)	0,571
	Ac2 (m²)	1,131
	Wi1 (m³) (-)	-0,163
	Wi2 (m³) (-)	-0,271
Estimativa Pz	x = Pz / P_∞	1,186
Esforços na seção	Mg1 (KNm)	714,06
	Mg2 (KNm)	1015,00
	Mg3 (KNm)	666,90
	Mq+ (KNm)	2112,72
	Nr (kN)	-31
	Mr(kNm)	12
	$N_{\Delta T}$ (kN)	-823
	$M_{\Delta T}$(kNm)	325
Excentricidade da protensão	ep1 (m)	-0,53
	ep2 (m)	-0,92
Tab.6-NBR 8681-2003	ψ2 (q móvel)	0,30
Tab.6-NBR 8681-2003	ψ2 (DT)	0,30
Estimativa P∞	P_∞ (kN) ≥	**-3031,96**

A 7.9.2 ELS- FF BORDA INFERIOR DA VIGA

Comb. Frequente

Comb.Frequente	Seção:	S5
Requisito	σt ≤ α.fctk,inf (MPa)	2,70
Dados seção	Ac1 (m²)	0,571
	Ac2 (m²)	1,131
	Wi1 (m³) (-)	-0,163
	Wi2 (m³) (-)	-0,271
Estimativa Pz	x = Pz / P_∞	1,19
Esforços na seção	Mg1 (KNm)	714,06
	Mg2 (KNm)	1015,00
	Mg3 (KNm)	666,90
	Mq+ (KNm)	2112,72
Excentricidade da protensão	ep1 (m)	-0,53
	ep2 (m)	-0,92
Tab.6-NBR 8681-2003	ψ1 (q móvel)	0,50
	ψ2 (ΔT)	0,30
Estimativa P∞	P_∞ (kN) ≥	**-2811,17**

A 7.9.3 ESCOLHA DA ÁREA DE ARMADURA ATIVA

Catálogo Freyssinet 2015	P_∞ nec. (kN)	**-3032,0**
	Pi (kN)	-4331,4
	N cord, calc	29,3
	$\phi_{cordoalha}$ (mm)	12,7
	Acordoalha (mm²)	100,9
	Ap (cm²), nec	29,61
	N cord, adot	**11**
	n (cabos)	**3**
	Ap (cm²), adot	**33,3**
	Verificação	**OK**

A 7.9.4 PROTENSÃO NAS DATAS A PARTIR DA ARMADURA ADOTADA

	P (n cabos) (kN)
Pi (kN) (macaco)	-4871
Pj (kN)	-4384
Pz (kN)	-4043
Pk (kN)	-3897
P_∞ (kN)	**-3410**

A 7.10 VERIFICAÇÕES NO ATO DA PROTENSÃO

$$\sigma_{ato} = P_j \cdot \left(\frac{1}{A_1} + \frac{e_{p1}}{W_1}\right) - \frac{M_{g1}}{W_1} \leq \bar{\sigma_t} \; ou \geq \bar{\sigma_c}$$

A 7.10.1 ELU - FF NA BORDA SUPERIOR DA VIGA

ABNT NBR 6118:2014	Seção:	S5
Item 17.2.4.3 (fctk,inf mais crítico)	σt_ato ≤ 1,2.fctm (MPa)	3,64
Dados seção pré-moldada	Ac1 (m²)	0,5713
	Ws1 (m³)	0,17
	ep1 (m)	-0,5294
Esforços no ato da protensão	Mg1 (KNm)	714,06
	Pj (kN)	-4384
Coeficientes ponderadores	γf	1,0
	γp	1,1
Verificação	σt (MPa)	2,55
	Verificação	**OK**

A 7.10.2 ELU - CE NA BORDA INFERIOR DA VIGA

ABNT NBR 6118:2014	Seção:	S5
Item 17.2.4.3	σc_ato ≤ 0,7*fck,j (MPa)	-22,55
Dados seção pré-moldada	Ac1 (m²)	0,5713
	Wi1 (m³) (-)	-0,16
	ep1 (m)	-0,5294
Esforços no ato da protensão	Mg1 (KNm)	714,1
	Pj (kN)	-4384,2
Coeficientes ponderadores	γf	1,0
	γp	1,1
Verificação	σc (MPa)	-19,8
	Verificação	**OK**

A 7.11 VERIFICAÇÕES EM SERVIÇO

$$\sigma = \boxed{\frac{P_z}{A_1}} - \frac{(P_z - P_\infty)}{A_2} \boxed{+ \frac{P_z \cdot e_{p1}}{W_1}} - \frac{(P_z - P_\infty) \cdot e_{p2}}{W_2} \boxed{- \frac{M_{g1}}{W_1} - \frac{M_{g2}}{W_1}} - \frac{M_{g3}}{W_2}$$

$$- \frac{\psi_i M_q}{W_2} + \boxed{\frac{N_r}{A_2} - \frac{M_r}{W_2}} + \boxed{\frac{\psi_i N_{\Delta T}}{A_2} - \frac{\psi_i M_{\Delta T}}{W_2}} \leq \sigma_t \; ou \; \geq \sigma_c$$

ELS - Compressão Borda Superior da Laje

	Seção:	S5
Verif. do CEB-FIB 1990 (Comb. Rara) NBR 6118:2014	σc = ≤ 0,6*fck (MPa)	-21,0
Esforços tempo infinito	Pz (kN)	-4043
	P_∞ (kN)	-3410
	Mg3 (KNm)	666,9
	Mq+ (KNm)	2112,7
Dados seção composta (laje)	Ac2 (m²)	1,1313
	ep2 (m)	-0,9158
	Ws2 (m³)	0,53
Verificação	σc (MPa)	-5,78
	Verificação	**OK**

ELS - Compressão Borda Superior da Viga

	Seção:	S5
Verif. do CEB-FIB 1990 (Comb. Rara) NBR 6118:2014 Esforços tempo infinito	σc = ≤ 0,6*fck	-21
	Pz (kN)	-4043,2
	P_∞ (kN)	-3409,9
	Mg1 (KNm)	714,06
	Mg2 (KNm)	1015
	Mg3 (KNm)	666,9
	Mq+ (KNm)	2112,7
	Nr (kN)	-31
	Mr(kNm)	12
	$N_{\Delta T}$ (kN)	-823
	$M_{\Delta T}$(kNm)	325
Tab.6-NBR 8681-2003	ψ1 (ΔT)	0,5
Dados da seção composta no tempo infinito	Ac1 (m²)	0,5713
	Ac2 (m²)	1,1313
	ep1 (m)	-0,5294
	ep2 (m)	-0,9158
	ys1 (m)	0,6406
	Ix1 (m4)	0,1072
	ys2* (m)	0,2542
	Ix2 (m4)	0,2832
Verificação	σc (MPa)	-7,62
	Verificação	**OK**

A 8 CÁLCULO DAS PERDAS IMEDIATAS DE PROTENSÃO

A 8.1 PROTENSÃO INICIALMENTE ADOTADA

Pi (kN)	-4871
Ap (cm²), adot	33,3
σpi (kN/m²)	-1463000

A 8.2 PERDA POR ATRITO CABO/BAINHA

ABNT NBR 6118:2014	Item\ Seção	S1	S2	S3	S4	S5
Item 9.6.3.3.2.2	x (m)	0	2,5	5	7,5	10
$P(x) = Pi.e^{-(\mu.\Sigma\alpha+K.x)}$	α	0,105	0,079	0,053	0,026	0,000
	Σα	0,000	0,026	0,053	0,079	0,105
	K	0,001	0,001	0,001	0,001	0,001
	μ	0,24	0,24	0,24	0,24	0,24
P após perda por atrito	Pμ (kN)	-4871	-4829	-4786	-4744	-4702

A 8.3 PERDA POR ACOMODAÇÃO NA ANCORAGEM

Cholfe e Bonilha (2016) Item 4.3.2.2 (pág 157) Aplica-se apenas a decaimento linear e único trecho curvo, sem trechos retos e x <= Lviga/2.

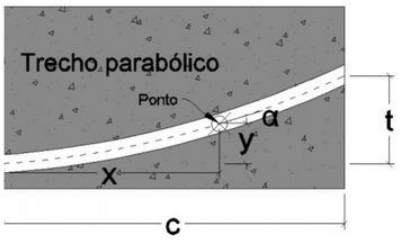


Figura A18 - Esquema de variáveis da parábola do cabo

$$x = \sqrt{\frac{\delta \cdot E_p \cdot A_p}{\tan \alpha}}$$

Se x ≤ L/2:

$$\Delta P_{\delta x=0} = 2 * \tan \alpha * x$$

$$\Delta P_{\delta x} = \Delta P_{\delta x=0} - \tan \alpha * x$$

$$P_{\delta x} = P_{\mu x} - \Delta P_{\delta x}$$

Se x > L/2

$$\Delta P_{x=l/2} = \frac{1}{l/2}\left(\delta \cdot E_p \cdot A_p - \tan \alpha \cdot \left(\frac{l}{2}\right)^2\right)$$

$$\Delta P_{\delta x=0} = 2 * \tan\left(\frac{l}{2}\right) * x + \Delta P_{x=l/2}$$

$$\Delta P_{\delta x} = \Delta P_{\delta x=0} - \tan \alpha * x$$

$$P_{\delta x} = P_{\mu x} - \Delta P_{\delta x}$$

Ep(GPa)	200,00	
δ (m)	0,006	
Ap (cm²), adot	33,3	
L/2 (m)	10,0	
tan α (kN/m)	17,0	
x acom (m)	15,32	Corrigir P_δ conforme valor de x
DP (x=l/2)	169,80	
DP (x=0)	510,2	

Seção	S1	S2	S3	S4	S5
x (m)	0	2,5	5	7,5	10
DPx	510	468	425	383	340
P_δ	-4361	-4404	-4446	-4489	-4531

A 8.4 PERDA POR ENCURTAMENTO ELÁSTICO DO CONCRETO

ABNT NBR 6118:2014	Item\ Seção	S1	S2	S3	S4	S5
Item 9.6.3.3.2.1	x (m)	0	2,5	5	7,5	10
$\Delta\sigma_p = \frac{\alpha_p.(\sigma_{cp}+\sigma_{cg}).(n-1)}{2n}$	αpj	6,29	6,29	6,29	6,29	6,29
	Ac1 (m²)	0,57125	0,57125	0,57125	0,57125	0,571
Tensões no concreto na altura do cabo de protensão	Ix1 (m4)	0,10723	0,107227	0,10723	0,10723	0,107
	n	3	3	3	3	3
	Ap (cm²), adot	33,3	33,3	33,3	33,3	33,3
$\sigma_{cp} = P.(\frac{1}{Ac} + \frac{ep^2}{Ic})$	$P_{(\delta+\mu)}$ (kN)	-4361	-4404	-4446	-4489	-4531
	ep (m)	0	-0,232	-0,397	-0,496	-0,529
$\sigma_{cg} = \frac{Mg1.ep}{Ic}$	Mg1 (kNm)	0	312	536	669	714
	σcp (kN/m²)	-7634	-9912	-14322	-18171	-19778
	σcg (kN/m²)	0	675	1983	3099	3526
$\Delta P_{enc} = \Delta\sigma_p.A_p$	Δσp (kN/m²)	-16012	-19375	-25878	-31613	-34087
	ΔPε (kN)	-53,3	-64,5	-86,2	-105,3	-113,5

A 8.5 PERDAS IMEDIATAS TOTAIS

Item\Seção	S1	S2	S3	S4	S5
x (m)	0	2,5	5	7,5	10
$P_{(\mu)}$ (kN)	-4871,35	-4828,68	-4786,25	-4744,11	-4702,3
$\sigma p_{(\mu)}$ (kN/m²)	-1,5E+06	-1,5E+06	-1,4E+06	-1,4E+06	-1,4E+06
$P_{(\mu+\delta+\varepsilon)}$ (kN)	-4307,8	-4339,2	-4360,1	-4383,5	-4417,8
$\sigma p_{(\mu+\delta+\varepsilon)}$ (kN/m²)	-1,3E+06	-1,3E+06	-1,3E+06	-1,3E+06	-1,3E+06
$\Delta P_{(\mu+\delta+\varepsilon)}$/Pi	11,57%	10,92%	10,50%	10,01%	9,31%

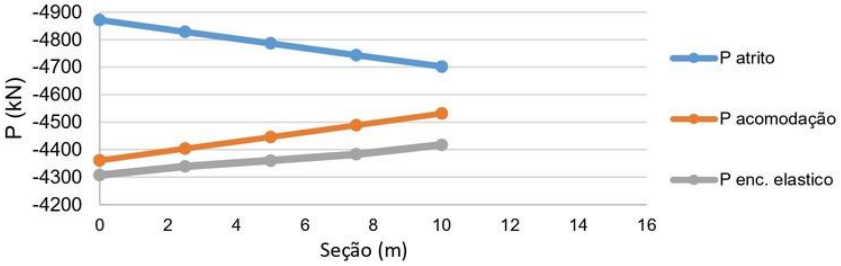


Figura A19 - Força de protensão após perdas imediatas

A 9 CÁLCULO DAS PERDAS PROGRESSIVAS DE PROTENSÃO

De acordo com o ABNT NBR 6118:2014 Item 9.6.3.4.4, para situações em que o carregamento permanente é aplicado parceladamente em idades diferentes deve-se usar o Metodo Geral para considerar a simultaneidade dos efeitos de retração, fluência e relaxação do aço.

A 9.1 MÉTODO GERAL

CEB-FIP (1978) apud Cholfe e Bonilha (2016) (págs. 191 e 205)

$$\Delta\sigma_{p,c+s}(t, t_0) = \frac{[\varepsilon_{cs}(t, t_0) \cdot E_p + \alpha p \cdot \varphi(t, t_0) \cdot (\sigma_{c,P0} + \sigma_{c,g}) + \alpha_P \cdot \Sigma_i [\Delta\sigma_{c,gi} \cdot \varphi(t, t_i)]}{[1 - \alpha_P \cdot (\sigma_{cP0} / \sigma_{P0}) \cdot (1 + \frac{\varphi(t, t_0)}{2})]}$$

Calcula-se a fluência para cada parcela de carregamento permanente acrescida para intervalo de tempo entre a aplicação da carga até a data

A 9.2 DETERMINAÇÃO DA FLUÊNCIA

Anexo A da ABNT NBR 6118:2014

Idade e espessura fictícia

$$t = \alpha \sum_i \frac{T_i + 10}{30} \Delta t_{ef,i} \qquad h_{fic} = \gamma \frac{2A_c}{u_{ar}}$$

$$\gamma = 1 + \exp(-7{,}8 + 0{,}1U)$$

Coeficiente de Fluência

$$\varphi(t, t_0) = \varphi_a + \varphi_{f\infty}[\beta_f(t) - \beta_f(t_0)] + \varphi_{d\infty}\beta_d$$

$$\varphi_a = 0{,}8\left[1 - \frac{f_c(t_0)}{f_c(t_\infty)}\right]$$

$f_{ckj} = \beta_1 \cdot f_{ck}$ $\beta_1 = \exp\{s[1 - (28/t)^{1/2}]\}$ (Item 12.3.3 da ABNT NBR 6118:2014)

$\varphi_{f\infty} = \varphi_{1c} \cdot \varphi_{2c}$

ϕ1c (%) Tabela A.1 (ABNT NBR 6118:2014)

$$\varphi_{2c} = \frac{42 + h_{fic}}{20 + h_{fic}}$$

$$\beta_f(t) = \frac{t^2 + At + B}{t^2 + Ct + D}$$

$$\beta_d(t) = \frac{t - t_0 + 20}{t - t_0 + 70}$$

$A = 42h^3 - 350h^2 + 588h + 113;$

$B = 768h^3 - 3060h^2 + 3234h - 23;$

$C = -200h^3 + 13h^2 + 1090h + 183;$

$D = 7579h^3 - 31916h^2 + 35343h + 1931$

A 9.2.1 COEFICIENTE DE FLUÊNCIA PARA DIFERENTES PARCELAS DE CARREGAMENTO PERMANENTE E INTERVALOS

Idade	∞			w	z		k		
Efeito g	Mg1	Mg2	Mg3	Mg1	Mg1	Mg2	Mg1	Mg2	Mg3
Coef.\ Idades	ϕ(∞,t0)	ϕ(∞,w)	ϕ(∞,z)	ϕ(w,to)	ϕ(z,t0)	ϕ(z,w)	ϕ(k,t0)	ϕ(k,w)	ϕ(k,z)
α	3	3	3	3	3	3	3	3	3
Ti (°C)	25	25	25	25	25	25	25	25	25
ti real	14	38	66	14	14	38	14	38	66
tf real	10000	10000	10000	38	66	66	94	94	94
ti fic	49	133	231	49	49	133	49	133	231
tf fic	10000	10000	10000	133	231	231	329	329	329
s	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2
β1t0=	0,920	1,029	1,072	0,920	0,920	1,029	0,920	1,029	1,072
β1t=	1,209	1,209	1,209	1,029	1,072	1,072	1,095	1,095	1,095
fc(to)/ fc(t∞)	0,762	0,851	0,887	0,895	0,858	0,959	0,841	0,939	0,979
ϕa	0,191	0,119	0,090	0,084	0,113	0,032	0,128	0,048	0,017
Ac(m²)	0,57125	1,13125	1,1313	0,5713	1,1313	1,1313	0,57125	1,131	1,131
u_{ar} (m)	4,27	7,43234	7,4323	4,27	7,4323	7,4323	4,27234	7,432	7,432
U	70	70	70	70	70	70	70	70	70
γ	1,449	1,449	1,449	1,449	1,449	1,449	1,449	1,449	1,449
hfic(m)	0,39	0,44	0,44	0,39	0,44	0,44	0,39	0,44	0,44
ϕ1c (%)	2,50	2,50	2,50	2,50	2,50	2,50	2,50	2,50	2,50
ϕ2c	1,37	1,34	1,34	1,37	1,34	1,34	1,37	1,34	1,34
ϕf∞	3,44	3,36	3,36	3,44	3,36	3,36	3,44	3,36	3,36
A	291	308	308	291	308	308	291	308	308
B	815	874	874	815	874	874	815	874	874
C	596	649	649	596	649	649	596	649	649
D	11276	11962	11962	11276	11962	11962	11276	11962	11962
βf(t)	0,971	0,968	0,968	0,528	0,582	0,582	0,649	0,630	0,630
βf(t0)	0,407	0,513	0,582	0,407	0,398	0,513	0,407	0,513	0,582
ϕd∞	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4
βd(t-to)	0,995	0,995	0,995	0,675	0,802	0,702	0,857	0,812	0,702
ϕ(t,t0)	**2,526**	**2,044**	**1,783**	**0,770**	**1,054**	**0,546**	**1,300**	**0,767**	**0,459**

A 9.3 DETERMINAÇÃO DA RETRAÇÃO

Anexo A da ABNT NBR 6118:2014

$$\varepsilon_{cs}(t, t_0) = \varepsilon_{cs\infty} [\beta_s(t) - \beta_s(t_0)]$$

$$\varepsilon_{cs\infty} = \varepsilon_{1s}\, \varepsilon_{2s}$$

ε_{1s} Tabela A.1

$$\varepsilon_{2s} = \frac{33 + 2h_{fic}}{20,8 + 3h_{fic}}$$

$$\beta_s(t) = \frac{\left(\frac{t}{100}\right)^3 + A\left(\frac{t}{100}\right)^2 + B\left(\frac{t}{100}\right)}{\left(\frac{t}{100}\right)^3 + C\left(\frac{t}{100}\right)^2 + D\left(\frac{t}{100}\right) + E}$$

$A = 40;$

$B = 116h^3 - 282h^2 + 220h - 4,8;$

$C = 2,5h^3 - 8,8h + 40,7;$

$D = -75h^3 + 585h^2 + 496h - 6,8;$

$E = -169h^4 + 88h^3 + 584h^2 - 39h + 0,8;$

A 9.3.1 RETRAÇÃO EM DIFERENTES IDADES

Coef.\Idades	w	z	k	∞
α	1	1	1	1
Ti (°C)	25	25	25	25
to real	14	14	14	14
t final real	38	66	94	10000
to fic	**16**	**16**	**16**	**16**
t fic	**44**	**77**	**110**	**11667**
Ac(m²)	0,5713	1,1313	1,1313	1,1313
u_{ar} (m)	4,27	7,4323	7,4323	7,4323
U	70	70	70	70
γ	1,449	1,449	1,449	1,449
hfic (m)	3,9E-01	4,4E-01	4,4E-01	4,4E-01
ε1s	6,2E-04	6,2E-04	6,2E-04	6,2E-04
ε2s	0,806	0,792	0,792	0,792
εcs inf	5,0E-04	4,9E-04	4,9E-04	4,9E-04
A	40	40	40	40
B	45	47	47	47
C	37	37	37	37
D	269	319	319	319
E	75	98	98	98
Bs(t)	0,138	0,165	0,205	1,004
Bs(to)	0,070	0,058	0,058	0,058
εcs (t,t0)	**3,4E-05**	**5,3E-05**	**7,2E-05**	**4,6E-04**

A 9.4 APLICAÇÃO DO MÉTODO GERAL

$$\Delta\sigma_{p,c+s}(t, t_0) = \frac{[\varepsilon_{cs}(t, t_0) \cdot E_p + \alpha p \cdot \varphi(t, t_0) \cdot (\sigma_{c,P0} + \sigma_{c,g}) + \alpha_P \cdot \Sigma_i [\Delta\sigma_{c,gi} \cdot \varphi(t, t_i)]}{[1 - \alpha_P \cdot (\sigma_{cP0} / \sigma_{P0}) \cdot (1 + \frac{\varphi(t, t_0)}{2})]}$$

$$\sigma_{c,g_i} = \frac{Mg_i \cdot ep}{Ic}$$

$$\sigma_{c,p0} = P.\left(\frac{1}{Ac} + \frac{ep^2}{Ic}\right)$$

A 9.4.1 PERDA DE PROTENSÃO DEVIDO À FLUÊNCIA EM CONJUNTO COM A RETRAÇÃO

	Item\Seção	S1	S2	S3	S4	S5
Dados concreto	Ep(GPa)	2E+08	2E+08	2E+08	2E+08	2E+08
	αp28	6,04	6,04	6,04	6,04	6,04
Retração no intervalo entre to e a idade considerada.	εsc (w,to)	3,4E-05	3,4E-05	3,4E-05	3,4E-05	3,4E-05
	εsc (z,to)	5,3E-05	5,3E-05	5,3E-05	5,3E-05	5,3E-05
	εsc (k,to)	7,2E-05	7,2E-05	7,2E-05	7,2E-05	7,2E-05
	εsc (∞,to)	4,6E-04	4,6E-04	4,6E-04	4,6E-04	4,6E-04
Fluência entre cada etapa de acréscimo de Mgi até o tempo final considerado.	ϕ(¥,t0)	2,526	2,526	2,526	2,526	2,526
	ϕ(¥,w)	2,044	2,044	2,044	2,044	2,044
	ϕ(¥,z)	1,783	1,783	1,783	1,783	1,783
	ϕ(w,to)	0,770	0,770	0,770	0,770	0,770
	ϕ(z,t0)	1,054	1,054	1,054	1,054	1,054
	ϕ(z,w)	0,546	0,546	0,546	0,546	0,546
	ϕ(k,t0)	1,300	1,300	1,300	1,300	1,300
	ϕ(k,w)	0,767	0,767	0,767	0,767	0,767
	ϕ(k,z)	0,459	0,459	0,459	0,459	0,459
Dados da seção	Ac1 (m²)	0,5713	0,5713	0,5713	0,5713	0,57125
	Ic1 (m4)	0,107	0,107	0,107	0,107	0,107
	ep1 (m)	0,00	-0,23	-0,40	-0,50	-0,53
	Ic2 (m4)	0,28	0,28	0,28	0,28	0,28
	ep2 (m)	-0,39	-0,62	-0,78	-0,88	-0,92
Parcelas de Carregamento Permanente	Mg1 (kNm)	0,00	312,4	535,55	669,43	714,063
	Mg2 (kNm)	0	444,06	761,25	951,56	1015
	Mg3 (kNm)	0	291,77	500,18	625,22	666,9
P após Per.	$P_{0\,(imed.)}$ (kN)	-4308	-4339	-4360	-4384	-4418
Tensão no concreto na altura do cabo de prontesão devido a cada efeito em separado	σcp0 (kN/m²)	-7541	-9767	-14044	-17745	-19283
	σcg (kN/m²)	0	675	1983	3099	3526
	σcg2 (kN/m²)	0	969	2106	2966	3283
	σcg3(kN/m²)	0	637	1384	1949	2157
Tensão prot. Inicial	σpi (kN/m²)	1,5E+06	1,5E+06	1,5E+06	1,5E+06	1,5E+06
Perda de protensão devidoà retração e fluência, por idade, por seção	Δσp,cs,w	-39110	-44975	-55590	-64256	-67850
	Δσp,cs,z	-55837	-61386	-73772	-84470	-89109
	Δσp,cs,k	-69998	-74555	-87140	-98335	-103325
	Δσp,cs,∞	-194162	-194898	-208517	-221834	-228280
Perda de protensão porcentagem defido à fluência e retração	Δσp,cs/σpo,w	-2,67%	-3,07%	-3,80%	-4,39%	-4,64%
	Δσp,cs/σpo,z	-3,82%	-4,20%	-5,04%	-5,77%	-6,09%
	Δσp,cs/σpo,k	-4,78%	-5,10%	-5,96%	-6,72%	-7,06%
	Δσp,cs/σpo,∞	-13,27%	-13,32%	-14,25%	-15,16%	-15,60%

A 9.5 PERDA POR RELAXAÇÃO DO AÇO

A 9.5.1 COEFICIENTE DE RELAXAÇÃO PURA

$$\psi(t,t_0) = \psi_{1000}\left(\frac{t-t_0}{41,67}\right)^{0,15}$$

$$\Psi(t, t_0) = \frac{\Delta\sigma_{pr}(t, t_0)}{\sigma_{Pi}}$$

$$\psi(t_\infty, t_0) \cong 2,5\ \psi_{1000}$$

$$\sigma_{Pi} = \sigma_{P0} + \Delta\sigma_{P0}$$

$$\sigma_{P0} = \frac{P_0}{A_p}$$

$$\Delta\sigma_{P0} = \Sigma M_g \cdot \frac{e_p}{I_c} \cdot \alpha_p$$

$$\Delta\sigma_{pr}(\infty) = \Psi_\infty \cdot \sigma_{Pi}$$

Tensão de Protensão após perdas imed.	Item\Seção	S1	S2	S3	S4	S5
	σp0 (kN/m²)	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σp0/fptk	0,681	0,686	0,689	0,693	0,698
Tabela 8.4 NBR 6118	ψ1000	2,27	2,33	2,37	2,41	2,48
Idades de cada etapa	to	14	14	14	14	14
	tw	38	38	38	38	38
	tz	66	66	66	66	66
	tk	94	94	94	94	94
Coeficiente de Relaxação Pura em cada idade em cada seção (porcentagem)	Ψ (w,t0)	2,09	2,15	2,18	2,22	2,28
	Ψ (z,t0)	2,35	2,41	2,45	2,50	2,56
	Ψ (k,t0)	2,50	2,57	2,61	2,66	2,73
	Ψ (∞,t0)	5,68	5,83	5,93	6,04	6,20
Ep/Ep	αp28	6,04	6,04	6,04	6,04	6,04
Tensão inicial de Protensão devido à protensão e à carga permanente	σpi,w (kN/m²)	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σpi,z (kN/m²)	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σpi,k (kN/m²)	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σpi,∞ (kN/m²	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
Variação da tensão de protensão devido à relaxação pura (kN/m²)	Δσp,r,w	-27049	-27872	-28310	-28849	-29803
	Δσp,r,z	-30375	-31160	-31375	-31732	-32692
	Δσp,r,k	-32403	-33141	-33251	-33537	-34518
	Δσp,r,∞	-73456	-75130	-75379	-76029	-78252

A 9.5.2 RELAXAÇÃO RELATIVA

Cholfe e Bonilha (2016) item 4.4.3.2 (págs. 200)

Considera o efeito simultâneo entre retração/fluência/relaxação

$$\Delta\sigma_{pr}(\infty), \text{rel} = \Delta\sigma_{pr}(\infty) \cdot [1 - 2 \cdot (\frac{|\Delta\sigma_P(\infty),_{c+s}|}{\sigma_{Pi}})]$$

	Item\Seção	S1	S2	S3	S4	S5
Variação de tensão devido à relaxação pura por idade e seção.	Δσp,r,w	-27049	-27872	-28310	-28849	-29803
	Δσp,r,z	-30375	-31160	-31375	-31732	-32692
	Δσp,r,k	-32403	-33141	-33251	-33537	-34518
	Δσp,r,∞	-73456	-75130	-75379	-76029	-78252
Variação de tensão devido à retração e fluência por idade e seção.	Δσp,cs,w	-39110	-44975	-55590	-64256	-67850
	Δσp,cs,z	-55837	-61386	-73772	-84470	-89109
	Δσp,cs,k	-69998	-74555	-87140	-98335	-103325
	Δσp,cs,∞	-194162	-194898	-208517	-221834	-228280
Tensão inicial de Protensão devido à protensão e à carga permanente (kN/m²)	σpi,w	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σpi,z	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σpi,k	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
	σpi,∞	-1E+06	-1E+06	-1E+06	-1E+06	-1E+06
Perda devido à relaxação relativa por idade e seção.	Δσp,rel,w	-25413	-25942	-25884	-25992	-26705
	Δσp,rel,z	-27753	-28202	-27760	-27515	-28123
	Δσp,rel,k	-28896	-29309	-28695	-28300	-28867
	Δσp,rel,∞	-51408	-52419	-50668	-49246	-49948
Perda devido à relaxação relativa por idade e seção (pocentagem)	Δσp,rel,w/ σpi	-1,74%	-1,77%	-1,77%	-1,78%	-1,83%
	Δσp,rel,z/ σpi	-1,90%	-1,93%	-1,90%	-1,88%	-1,92%
	Δσp,rel,k/ σpi	-1,98%	-2,00%	-1,96%	-1,93%	-1,97%
	Δσp,rel,∞/ σpi	-3,51%	-3,58%	-3,46%	-3,37%	-3,41%

A 9.6 SOMA DAS PERDAS PROGRESSIVAS

	Item\ Seção	S1	S2	S3	S4	S5
Perdas devido à retração, fluência e relaxação, por idade e por seção. (kN/m²)	Δσ(pr,c,s) w	-64524	-70917	-81474	-90248	-94554
	Δσ(pr,c,s) z	-83590	-89588	-101531	-111985	-117232
	Δσ(pr,c,s) k	-98895	-103864	-115836	-126635	-132192
	Δσ(pr,c,s) ∞	-245570	-247317	-259185	-271080	-278228

A 10 SOMA DAS PERDAS IMEDIATAS

Perdas por Atrito, Delizamento de Ancoratem e Encurtamento Elástico do Concreto. Resumo das perda imediatas em cada seção:

Item\Seção	S1	S2	S3	S4	S5	
$P_{(\mu+\delta+\varepsilon)}$ (kN)	-4308	-4339	-4360	-4384	-4418	
$\sigma p_{(\mu+\delta+\varepsilon)}$ (kN/m²)	-1293758	-1303175	-1309450	-1316495	-1326799	Estimado
$\Delta P_{(\mu+\delta+\varepsilon)}$/Pi	**11,57%**	**10,92%**	**10,50%**	**10,01%**	**9,31%**	**10,00%**

A 10.1 SOMA DAS PERDAS PROGRESSIVAS

Perdas por Fluência, Retração e Relaxação do Aço. Resumo das perdas progressivas por idade e por seção:

Item\Seção	S1	S2	S3	S4	S5
Pi (kN)	-4871	-4871	-4871	-4871	-4871
spi (kN/m²)	-1463000	-1463000	-1463000	-1463000	-1463000
Ap (cm²), adot	33,3	33,3	33,3	33,3	33,3
Δσ(p r, c, s) w	-64524	-70917	-81474	-90248	-94554
Δσ(p r, c, s) z	-83590	-89588	-101531	-111985	-117232
Δσ(p r, c, s) k	-98895	-103864	-115836	-126635	-132192
Δσ(p r, c, s) ∞	-245570	-247317	-259185	-271080	-278228
σ(p r, c, s) w	-1,4E+06	-1,4E+06	-1,4E+06	-1,4E+06	-1,4E+06
σ(p r, c, s) z	-1,4E+06	-1,4E+06	-1,4E+06	-1,4E+06	-1,3E+06
σ(p r, c, s) k	-1,4E+06	-1,4E+06	-1,3E+06	-1,3E+06	-1,3E+06
σ(p r, c, s) ∞	-1,2E+06	-1,2E+06	-1,2E+06	-1,2E+06	-1,2E+06
P(p r, c, s) w	-4657	-4635	-4600	-4571	-4557
P(p r, c, s) z	-4593	-4573	-4533	-4498	-4481
P(p r, c, s) k	-4542	-4526	-4486	-4450	-4431
P(p r, c, s) ∞	-4054	-4048	-4008	-3969	-3945
ΔP(p r, c, s) w/Pi	4,41%	4,85%	5,57%	6,17%	6,46%
ΔP(p r, c, s) z/Pi	5,71%	6,12%	6,94%	7,65%	8,01%
ΔP(p r, c, s) k/Pi	6,76%	7,10%	7,92%	8,66%	9,04%
ΔP(p r, c, s) ∞ /Pi	16,79%	16,90%	17,72%	18,53%	19,02%

A 10.2 SOMA PERDAS TOTAIS

Resumo das perdas de protensão totais por idade e por seção:

Item\Seção	S1	S2	S3	S4	S5	
x (m)	0	2,5	5	7,5	10	
ΔPtot w/Pi	**15,98%**	**15,77%**	**16,06%**	**16,18%**	**15,77%**	Estimado
ΔPtot z /Pi	**17,28%**	**17,05%**	**17,44%**	**17,67%**	**17,32%**	**17,00%**
ΔPtot k /Pi	**18,33%**	**18,02%**	**18,41%**	**18,67%**	**18,35%**	**20,00%**
ΔPtot ∞ /Pi	**28,35%**	**27,83%**	**28,21%**	**28,54%**	**28,33%**	**30,00%**
P0 (kN)	-4308	-4339	-4360	-4384	-4418	
Ptot w (kN)	-4093	-4103	-4089	-4083	-4103	
Ptot z (kN)	-4029	-4041	-4022	-4011	-4027	
Ptot k (kN)	-3979	-3993	-3974	-3962	-3978	
Ptot ∞ (kN)	-3490	-3516	-3497	-3481	-3491	

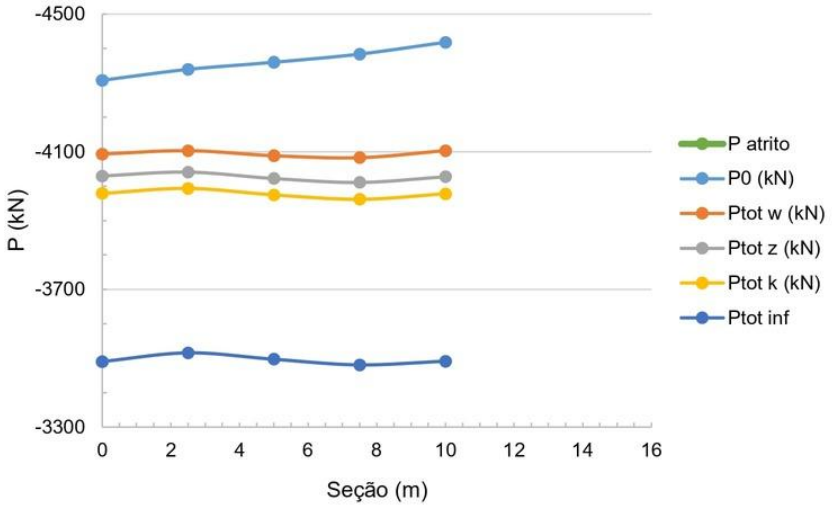


Figura A20 - Força de prontensão após perdas totais por idade

A 11 VERIFICAÇÕES EM ESTADO LIMITE DE SERVIÇO E ESTADO LIMITE ULTIMO - ATO DA PROTENSÃO

A 11.1 ELU - ATO DA PROTENSÃO

NBR 6118:2014	σt_ato=	≤1,2.fctmj (MPa)	3,64	ELU - Ato	Borda superior viga
Item 17.2.4	σc_ato=	≤ 0,7*fckj (MPa)	-22,6	ELU - Ato	Borda inferior viga

A 11.2 ELS - PROTENSÃO LIMITADA

NBR 6118:2014	σt_serv=	≤1,2.fctk,inf (MPa)	2,696	ELS - F	Borda inferior viga
Item 13.4	σt_serv=	≤ 0 (MPa)	0	ELS - D	Borda inferior viga
CEB - FIB 1978	σc_serv=	≤ 0,6*fck (MPa)	-21,0	ELS - CE	Borda superior laje

A 11.3 COMBINAÇÃO DE AÇÕES

Combinação Frequente

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + \Psi_1 . F_{q1,k} + \sum_{j=2}^{n} \Psi_{2j} . F_{qj,k}$$

Combinação Quase-permanente

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + \sum_{j=1}^{n} \Psi_{2j} . F_{qj,k}$$

Combinação Rara

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + F_{q1,k} + \sum_{j=2}^{n} \Psi_{1j} . F_{qj,k}$$

A 11.4 FORÇA DE EFETIVAS PRONTESÃO APÓS PERDAS

	Item/Seção	S1	S2	S3	S4	S5
Força de protensão efetiva por seção.	Ptotj (kN)	-4307,8	-4339,2	-4360,1	-4383,5	-4417,8
	Ptot w (kN)	-4093,0	-4103,0	-4088,8	-4083,0	-4103,0
	Ptot z (kN)	-4029,5	-4040,9	-4022,0	-4010,7	-4027,5
	Ptot k (kN)	-3978,5	-3993,3	-3974,4	-3961,9	-3977,7
	Ptot ∞ (kN)	-3490,2	-3515,7	-3497,1	-3480,9	-3491,4

A 11.5 DADOS

	Seção:	S1	S2	S3	S4	S5
Propriedades geométricas da seção pré-moldada e seção composta	Ac1 (m²)	0,57	0,57	0,57	0,57	0,57
	Ac2 (m²)	1,13	1,13	1,13	1,13	1,13
	Wi1 (m³) (-)	-0,16	-0,16	-0,16	-0,16	-0,16
	Wi2 (m³) (-)	-0,27	-0,27	-0,27	-0,27	-0,27
	Ws1 (m³)	0,167	0,167	0,167	0,167	0,167
	Ws2 (m³)	0,530	0,530	0,530	0,530	0,530
	Ix2 (m4)	0,283176	0,28318	0,28318	0,28318	0,28318
Borda sup. Viga	ys2* (m)	0,2541584	0,25416	0,25416	0,25416	0,25416
Esforços solicitantes	Mg1 (KNm)	0,0	312,4	535,5	669,4	714,1
	Mg2 (KNm)	0,0	444,1	761,3	951,6	1015,0
	Mg3 (KNm)	0,0	291,8	500,2	625,2	666,9
	Mqc (KNm)	0,0	119,4	204,8	255,9	273,0
	Mq+ (KNm)	0,0	1014,9	1625,4	1985,2	2112,7
	Nr (kN)	-31,35	-31,35	-31,35	-31,35	-31,35
	Mr(kNm)	12,36	12,36	12,36	12,36	12,36
	$N_{\Delta T}$ (kN)	-823,28	-823,28	-823,28	-823,28	-823,28
	$M_{\Delta T}$(kNm)	324,50	324,50	324,50	324,50	324,50
Excentricidades da prontensão	ep,eq1 (m)	0,00	-0,23	-0,40	-0,50	-0,53
	ep,eq2 (m)	-0,39	-0,62	-0,78	-0,88	-0,92

A 11.6 CÁLCULO DAS TESÕES NA SEÇÃO COMPOSTA

Em Serviço

$$\sigma = \boxed{\frac{P_z}{A_1}} - \frac{(P_z - P_\infty)}{A_2} \boxed{+\frac{P_z \cdot e_{p1}}{W_1}} - \frac{(P_z - P_\infty) \cdot e_{p2}}{W_2} \boxed{-\frac{M_{g1}}{W_1} - \frac{M_{g2}}{W_1}} - \frac{M_{g3}}{W_2} - \frac{\psi_i M_q}{W_2}$$
$$+ \boxed{\frac{N_r}{A_2} - \frac{M_r}{W_2}} + \boxed{\frac{\psi_i N_{\Delta T}}{A_2} - \frac{\psi_i M_{\Delta T}}{W_2}} \leq \sigma_t \; ou \; \geq \sigma_c$$

No Ato da Protensão

$$\sigma_{ato} = P_0 \left(\frac{1}{A_1} + \frac{e_{p1}}{W_1}\right) - \frac{M_{g1}}{W_1} \leq \bar{\sigma}_t \; ou \geq \bar{\sigma}_c$$

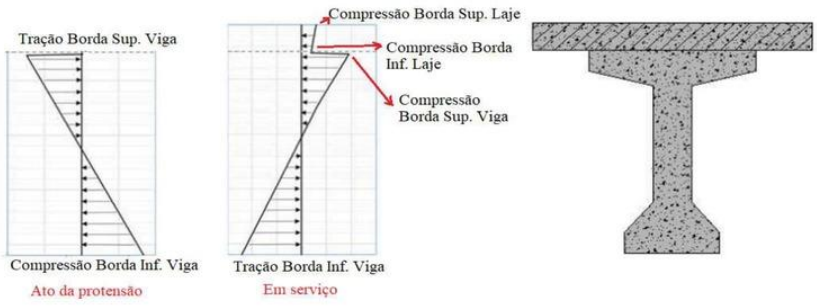


Figura A21 - Pontos de verificação das tensões na seção das longarinas

A 11.6.1 ELS-DESCOMPRESSÃO (COMB. QUASE PERMANENTE) - TRAÇÃO MÁXIMA DA BORDA INFERIOR DA VIGA

	Seção:	S1	S2	S3	S4	S5
Limite Tração	σt ≤ 0 (MPa)	0,00	0,00	0,00	0,00	0,00
Tab. 6 NBR 8681	ψ2 (q)	0,30	0,30	0,30	0,30	0,30
	ψ2 (ΔT)	0,30	0,30	0,30	0,30	0,30
Tensões por idade, por seção (- compressão)	σtj (kN/m²)	-7541	-11856	-14987	-16938	-17727
	σtw (kN/m²)	-7165	-8155	-8790	-9170	-9405
	σtz (kN/m²)	-7036	-7324	-7437	-7486	-7602
	σtk (kN/m²)	-6777	-5666	-4900	-4425	-4355
	σt∞ (kN/m²)	-5648	-4154	-3097	-2432	-2280
Verificação	σtj ≤ 0	OK	OK	OK	OK	OK
	σtw ≤ 0	OK	OK	OK	OK	OK
	σtz ≤ 0	OK	OK	OK	OK	OK
	σtk ≤ 0	OK	OK	OK	OK	OK
	σt∞ ≤ 0	OK	OK	OK	OK	OK

A 11.6.2 ELS-FORMAÇÃO DE FISSURAS (COMB. FREQUENTE) - TRAÇÃO NA BORDA INFERIOR DA VIGA

	Seção:	S1	S2	S3	S4	S5
Limite Tração	σt ≤ 1,2.fctk,inf (kN/m²)	2696,37	2696,37	2696,37	2696,37	2696,37
Tab. 6 NBR 8681	ψ1(q)	0,50	0,50	0,50	0,50	0,50
	ψ2 (ΔT)	0,30	0,30	0,30	0,30	0,30
Tensões por idade, por seção (- compressão)	σtj (kN/m²)	-7541,1	-11856	-14987	-16938	-17727
	σtw (kN/m²)	-7165,0	-7478,1	-7629,4	-7719,7	-7857,8
	σtz (kN/m²)	-7035,9	-7236,0	-7285,5	-7296,7	-7400,6
	σtk (kN/m²)	-6776,9	-4916,5	-3699,7	-2958,4	-2794,4
	σt∞ (kN/m²)	-5648,2	-3404,0	-1896,6	-965,2	-719,8
Verificação	σtj ≤ limite	OK	OK	OK	OK	OK
	σtw ≤ limite	OK	OK	OK	OK	OK
	σtz ≤ limite	OK	OK	OK	OK	OK
	σtk ≤ limite	OK	OK	OK	OK	OK
	σt∞ ≤ limite	OK	OK	OK	OK	OK

A 11.6.3 ELS-COMPRESSÃO EXCESSIVA (COMB. RARA) -COMPRESSÃO NA BORDA SUPERIOR DA VIGA

	Seção:	S1	S2	S3	S4	S5
Limite Compressão	σc = ≤ 0,6*fck (kN/m²)	-21000	-21000	-21000	-21000	-21000
Tab. 6 NBR 8681	ψ1 (ΔT)	0,50	0,50	0,50	0,50	0,50
Tensões por idade, por seção (- compressão)	σcj (kN/m²)	-7541	-3458	-489	1325	1974
	σcw (kN/m²)	-7165	-6738	-6429	-6253	-6165
	σcz (kN/m²)	-7093	-6409	-5918	-5642	-5524
	σck (kN/m²)	-7575	-7707	-7694	-7699	-7681
	σc∞ (kN/m²)	-7312	-7549	-7608	-7655	-7651
Verificação	σcj ≤ limite	OK	OK	OK	OK	OK
	σcw ≤ limite	OK	OK	OK	OK	OK
	σcz ≤ limite	OK	OK	OK	OK	OK
	σck ≤ limite	OK	OK	OK	OK	OK
	σc∞ ≤ limite	OK	OK	OK	OK	OK

A 11.6.4 ELS-COMPRESSÃO EXCESSIVA (COMB. RARA) - COMPRESSÃO NA BORDA SUPERIOR DA LAJE

	Seção:	S1	S2	S3	S4	S5
Limite Compressão	σc = ≤ 0,6*fck (kN/m²)	-21000	-21000	-21000	-21000	-21000
Tensões por idade, por seção (- compressão)	σcz (kN/m²)	0,0	-775,7	-1329,7	-1662,1	-1772,9
	σck (kN/m²)	7,9	-2478,2	-4037,7	-4962,1	-5285,2
	σc∞ (kN/m²)	83,7	-2612,8	-4321,2	-5337,8	-5695,4
Verificação	σcz ≤ limite	OK	OK	OK	OK	OK
	σck ≤ limite	OK	OK	OK	OK	OK
	σc∞ ≤ limite	OK	OK	OK	OK	OK

A 11.6.5 ELS-COMPRESSÃO EXCESSIVA (COMB. RARA) -COMPRESSÃO NA BORDA INFERIOR DA LAJE

	Seção:	S1	S2	S3	S4	S5
Limite Compressão	σc = ≤ 0,6*fck (kN/m²)	-21000	-21000	-21000	-21000	-21000
Tensões por idade, por seção (- compressão)	σcz (kN/m²)	0,0	-261,9	-448,9	-561,2	-598,6
	σck (kN/m²)	27,4	-1157,1	-1899,1	-2338,4	-2491,7
	σc∞ (kN/m²)	289,7	-999,9	-1812,8	-2294,3	-2461,6
Verificação	σcz ≤ limite	OK	OK	OK	OK	OK
	σck ≤ limite	OK	OK	OK	OK	OK
	σc∞ ≤ limite	OK	OK	OK	OK	OK

A 11.7 VERIFICAÇÕES NO ATO DA PROTENSÃO

A 11.7.1 ELU - FORMAÇÃO DE FISSURAS -TRAÇÃO NA BORDA SUPERIOR DA VIGA

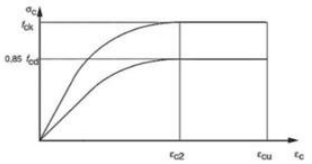
	Seção:	S1	S2	S3	S4	S5
	γp	1,10	1,10	1,10	1,10	1,10
	γf favoravel	0,9	0,9	0,9	0,9	0,9
Limite tração	σt_ato ≤ 1,2.fctm (kN/m²)	3645	3645	3645	3645	3645
Tração na seção no ato da protensão	σt (kN/m²)	-8295	-3430	102	2257	3024
	Verificação	OK	OK	OK	OK	OK

A 11.7.2 ELU - COMPRESSÃO EXCESSIVA -COMPRESSÃO NA BORDA INFERIOR DA VIGA

	Seção:	S1	S2	S3	S4	S5
	γp	1,10	1,10	1,10	1,10	1,10
	γf favoravel	1,00	1,00	1,00	1,00	1,00
Limite compressão	σc_ato ≤ 0,7*fckj (MPa)	-22552	-22552	-22552	-22552	-22552
Compressão na seção no ato da protensão	σc (kN/m²)	-8295	-13234	-16815	-19043	-19939
	Verificação	OK	OK	OK	OK	OK

A 12 DIMENSIONAMENTO DA ARMADURA DE FLEXÃO
A 12.1 PREPARAÇÃO DOS DADOS DOS MATERIAIS
A 12.1.1 CONCRETO

NBR 6118:2014	fck (MPa)	35
Tabela 12.1 NBR	γc	1,4
Resistência concreto	fcd (MPa)	25
	fctk,sup (MPa)	4,17
	Eci (GPa)	33
Deformações limites do concreto	ec2	0,20%
	ecu	0,35%
Ep/Ec	αp28	6,04



$$\sigma_c = 0{,}85\, f_{cd}\left[1-\left(1-\frac{\varepsilon_c}{\varepsilon_{c2}}\right)^n\right]$$

Para $f_{ck} \le 50$ MPa: n=2

Figura A22 - Gráfico de tensão deformação do concreto

A 12.1.2 ARMADURA ATIVA

NBR 6118:2014	Ep(GPa)	200
Itens: 8.4; 9.6	fptk(MPa)	1900
	fpyk(MPa)	1710
Tabela 12.1 NBR	γs	1,15
Resistências limites	fpyd(MPa)	1487
	fptd(MPa)	1652
Deformações limites	epyd	0,74%
	epud	3,50%

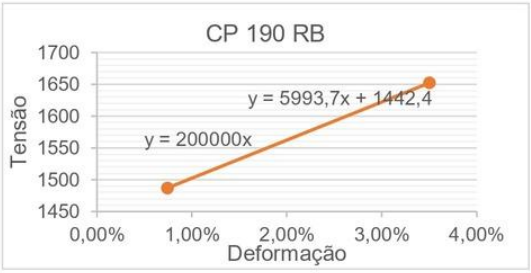


Figura A23 - Gráfico de tensão deformação do aço ativo CP 190 RB

A 12.1.3 ARMADURA PASSIVA

NBR 6118:2014	fyk	500
Tabela 12.1 NBR	γs	1,15
		0,00
Res. Cálculo	fyd	435
		435
Módulo Elast.	Es (GPa)	210
		0
Deformações limites	eyd	0,207%
		1,00%

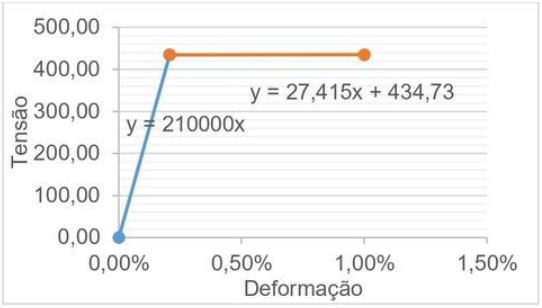


Figura A24 - Gráfico de tensão deformação do aço passivo CA-50

A 12.2 COMBINAÇÃO NORMAL ÚLTIMA

$$F_d = \sum_{i=1}^{m} \gamma_{gi} G_i + \gamma_{q1} Q_1 + \sum_{j=2}^{n} \gamma_{qj} \psi_{oj} Q_j$$

Tab.2 NBR 8681:03	γg	1,35
Tab.5 NBR 8681:03	γq	1,5
Tab.5 NBR 8681:03	γq (ΔT)	1,2
Tabela 11.1 NBR 6118:2014	$\gamma p,fav$	0,9
	$\gamma p,desf$	1,2
Tab. 6 NBR 8681:03	ψ_0 (ΔT)	0,6

A 12.3 DADOS DAS SEÇÕES

	Seção	S1	S2	S3	S4	S5
Mesa superior (laje)	bw laje (m)	2,00	2,00	2,00	2,00	2,00
Mesa superior (viga)	bsup (m)	0,70	0,70	0,70	0,70	0,70
Espessura da laje	t (m) (laje)	0,28	0,28	0,28	0,28	0,28
Altura total seção	h laje+long. (m)	1,58	1,58	1,58	1,58	1,58
Número cabos	n	3,00	3,00	3,00	3,00	3,00
Área seção composta	Ac2 (m²)	1,13125	1,13125	1,13125	1,13125	1,13125
Inércia sec. composta	Ix2 (m4)	0,28318	0,28318	0,283176	0,28318	0,283176
Mód. Inér. Borda Inf.	Wi2 (m³) (-)	-0,27	-0,27	-0,27	-0,27	-0,27
Excentr. Cabo Prot.	ep2 (m)	-0,39	-0,62	-0,78	-0,88	-0,92
Dist. Cabo da borda	d'ep2 (m)	0,66	0,43	0,26	0,16	0,13
Excentr. As Passiva	eAs (m)	-0,9758	-0,97584	-0,97584	-0,97584	-0,97584
Dist. As da borda	d' (m)	0,07	0,07	0,07	0,07	0,07
Alt. Útil (Refer. Ap)	d1 (m)	0,92055	1,15219	1,317639	1,41691	1,45
Alt. Útil (Refer. As)	d2 (m)	1,51	1,51	1,51	1,51	1,51
Protensão infinito	Ptot ∞ (kN)	-3490	-3516	-3497	-3481	-3491
Protensão de cálculo	Pd∞ (kN)	-3141	-3164	-3147	-3133	-3142
Área protensão	Ap (cm²)	33,3	33,3	33,3	33,3	33,3
Momentos fletores atuantes na seção	Mg1 (kNm)	0,00	312,40	535,55	669,43	714,06
	Mg2 (kNm)	0,00	444,06	761,25	951,56	1015,00
	Mg3 (kNm)	0,00	291,77	500,18	625,22	666,90
	Mq+ (kNm)	0,00	1014,92	1625,37	1985,15	2112,72
	Mr(kNm)	12,36	12,36	12,36	12,36	12,36
	$M_{\Delta T}$(kNm)	324,50	324,50	324,50	324,50	324,50
Comb. Normal Última	Msd (kNm)	250,3	3187,8	5114,3	6260,4	6653,9

A 12.4 EQUILÍBRIO DA SEÇÃO

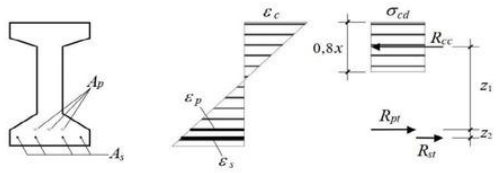


Figura A37 - Equilíbrio da seção de concreto com armadura ativa e passiva

$$M_{sd} = R_{cc} \times z = 0{,}8x \times b_w \times 0{,}85 \times \sigma_{cd}(d - 0{,}4x)$$

$$R_{cc} = R_{pt} + R_{st}$$

A 12.4.1 DOMÍNIOS DE DEFORMAÇÃO

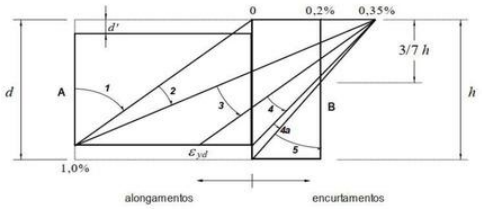


Figura A38 - Domínios de deformação da seção de concreto na flexão

A 12.5 CÁLCULO DE RCC

Aplica-se apenas para linha neutra na laje

	Seção	S1	S2	S3	S4	S5
Linha neutra	x (m) (supor)	0,039	0,13	0,16	0,17	0,172
Verificação	LN	**Laje**	**Laje**	**Laje**	**Laje**	**Laje**
Verificação Domínio 2 ou 3	x/d	0,042	0,113	0,121	0,120	0,119
	Domínio	**2**	**2**	**2**	**2**	**2**
Deformação As	εys	0,010	0,010	0,010	0,010	0,010
Deform. Concreto	εc	0,0003	0,0009	0,0012	0,0013	0,0013
Tensão Concreto	σcd (kN/m²)	5260	15304	17723	18409	18538
Braço alavanca Ap Rcc1	z1 (m)	0,78	1,01	1,18	1,28	1,31
Braço alavanca Ap Rcc2	z1 (m)	0,90	1,10	1,25	1,35	1,38
Rcc1 laje	Rccd1 (kN)	328,3	3183,2	4537,1	5007,1	5101,6
Rcc2 (laje+viga)	Rccd2 (kN)	0,00	0,00	0,00	0,00	0,00
Resultante compressão	Rccd,tot (kN)	328,3	3183,2	4537,1	5007,1	5101,6
Momento resistente	Mrd	256,2	3222,0	5343,0	6393,6	6683,1
Verificação	Mrd >= Msd	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**

A 12.6 VERIFICAÇÃO DA ARMADURA ATIVA

Cholfe e Bonilha (2016)
Item 3.4 Pág 84.

$$P_{nd} = P_d + \alpha_p \, A_p \, \sigma_{cPd}$$

$$\varepsilon_{Pn} = \frac{P_{nd}}{E_p A_P}$$

$$\sigma_{cPd1} = \frac{P_d}{A_{ch}} + \frac{P_d \times e_p^{\,2}}{I_h}$$

$$\varepsilon_{pt} = \varepsilon_p + \varepsilon_{pn}$$

A 12.7 PRÉ-ALONGAMENTO

	Seção	S1	S2	S3	S4	S5
sc na altura de Ap	scpd1 (kN/m²)	-4432,8	-7064,9	-9604,7	-11390,3	-12085,1
P de Neutralização	Pnd (kN)	-3231,4	-3284,3	-3335,35	-3371,24	-3385,21
Pré-alongamento	εpn	0,0049	0,0049	0,0050	0,0051	0,0051

A 12.8 RESULTANTE DE TRAÇÃO DEVIDO À PROTENSÃO

	Seção	S1	S2	S3	S4	S5
Deformação de Flexão	εp	0,0056	0,0073	0,0086	0,0093	0,0096
Deformação Total	εpt	0,0104	0,0123	0,0136	0,0144	0,0146
Tensão da Ap	spt (kN/m²)	1505016	1515868	1523739	1528509	1530117
Result. de Protensão	Rpt	5011,3	5047,4	5073,6	5089,5	5094,8
Verificação	Rpt >= Rcc	**OK**	**OK**	**OK**	**OK**	**Arm. Pas**

A 12.9 CÁLCULO DA ARMADURA PASSIVA

	Seção	S1	S2	S3	S4	S5
Result. de As necess.	Rst (kN)	0,0	0,0	0,0	0,0	6,8
Deformação As	εst	0,0100	0,0100	0,0100	0,0100	0,0100
Tensão da As	σst (kN/m²)	434783	434783	434783	434783	434783
Área necessária	Ast (cm²)	0,0	0,0	0,0	0,0	0,2

A 12.10 ARMADURA MÍNIMA

ABNT NBR 6118:2014
Item 17.3.5.2

$$M_{d,mín} = 0{,}8 . W_0 . f_{ctk,sup}$$

$$x = \frac{d - \sqrt{d^2 - 2M_{d,mín}/0{,}85 . f_{cd} . b_t}}{0{,}8}$$

$$A_{s,mín} = \frac{M_{d,mín}}{f_{yd} . (d - 0{,}4x)}$$

taxa mínima absoluta de 0,15 %

	Seção	S1	S2	S3	S4	S5
Momento mínimo	Md,mín (kNm)	903,9	903,9	903,9	903,9	903,9
Linha neutra	x (m)	0,02	0,02	0,02	0,02	0,02
Armadura mínima	Asmín (cm²)	16,97	16,97	16,97	16,97	16,97

A 12.11 ARMADURA PASSIVA ADOTADA

	Seção	S1	S2	S3	S4	S5
Armadura Necessária	Ast (cm²)	**17,0**	**17,0**	**17,0**	**17,0**	**17,0**
Bitola Adotada	ϕ (mm)	**25**	**25**	**25**	**25**	**25**

A 12.12 ARMADURA DE PELE

Taxa mínima (%) NBR 6118:2014		0,10% Ac,alma
Armadura pele fora do engrossamento	As,pele (cm²)	**1,25**

A 13 DIMENSIONAMENTO DA ARMADURA TRANSVERSAL DE CISALHAMENTO POR TRECHOS DA VIGA

A 13.1 PREPARAÇÃO DOS DADOS

A 13.1.1 DADOS CONCRETO

NBR 6118:2014	fck (MPa)	35,00
	fctk,inf (MPa)	2,25
	fctk,m (MPa)	3,21
Tabela 12.1 NBR	γc	1,40
Rsistência de cálculo	fcd (MPa)	25,00
	fctd (MPa)	1,60

A 13.1.2 DADOS DA ARMADURA ATIVA E PASSIVA

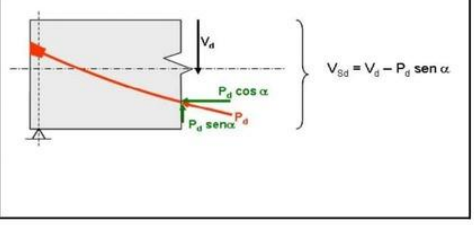
NBR 6118:2014	fyk (MPa)	500,00
	fpyk (MPa)	1710,00
Tabela 12.1 NBR	γs	1,15
Resistência de cálculo	fyd (MPa)	434,78
	fpyd (MPa)	1486,96

A 13.2 COMBINAÇÃO NORMAL ÚLTIMA

$$F_d = \sum_{i=1}^{m} \gamma_{gi} G_i + \gamma_{q1} Q_1 + \sum_{j=2}^{n} \gamma_{qj} \psi_{oj} Q_j$$

Tab.2 NBR 8681:03	γg	1,350
Tab.5 NBR 8681:03	γq	1,500
Tab.11.1 NBR 6118	γp,fav	0,900

A 13.3 CONSIDERAÇÃO DO EFEITO FAVORÁVEL DA PROTENSÃO

ABNT NBR 6118:2014	
Item 17.4.1.2.3	
$A_p . f_{pyd} + A_s . f_{yd} \geq V_{sd}$	
Hanai (2005)	
$V_p = P . \operatorname{sen} \alpha$	
$V_{ef,d} = \gamma_g . V_g + \gamma_q . V_q + \gamma_p . V_p$	

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Efeito Favorável	Considerar Vp	**sim**	**sim**	**sim**	**sim**
Força cortante solicitante no trecho	Vgk (kN)	479,2	359,4	239,6	119,8
	Vqk (kN)	515,1	402,2	316,8	252,0
	Vsd (kN)	1419,5	1088,5	798,6	539,7
Armadura de flexão no trecho	Ap (cm²)	33,30	33,30	33,30	33,30
	As (cm²)	16,97	16,97	16,97	16,97
	Ap*fpyfd+As*fyd	5688,9	5688,9	5688,9	5688,9
	Verificação	**OK**	**OK**	**OK**	**OK**
Inclinação cabo	αcabo (rad)	0,105	0,079	0,053	0,026
Força Protensão	P∞ (kN)	-3490	-3516	-3497	-3481
	Pd∞ (kN)	-3141	-3164	-3147	-3133
Cort. devido a P∞	Vp (kN)	**330,8**	**250,5**	**166,4**	**82,9**
Cortante efetiva	Vsd,ef (kN)	**1088,8**	**838,0**	**632,2**	**456,8**

A 13.4 VERIFICAÇÃO DA BIELA COMPRIMIDA

A 13.4.1 MODELO II

ABNT NBR 6118:2014
Item 17.4.2.3

$$V_{rd2} = 0{,}54 . f_{cd} . \alpha_{v2} . b_w . d . (\operatorname{cotg}\theta + \operatorname{cotg}\alpha) . \operatorname{sen}^2\theta$$

$$b_{w,ef} = b_w - \frac{1}{2}\sum \emptyset \qquad \alpha_{v2} = \left(1 - \frac{f_{ck}}{250}\right)$$

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Inclinação biela	θbiela (graus)	36	36	36	36
Inclinação estribo	αestribo (graus)	90	90	90	90
Largura alma	bw (m)	0,25	0,25	0,25	0,25
Altura total viga	h (m)	1,58	1,58	1,58	1,58
Dist. Ap da Borda Inf	d' prot (m)	0,6594	0,42781	0,262	0,1631
Altura útil seção	d (m)	1,264	1,264	1,318	1,4169
Diâmetro bainha	ø (bainha) (m)	0,07	0,07	0,07	0,07
Largura efet.	bw,efet (m)	0,218	0,218	0,218	0,218
Coeficiente ajuste	αv2	0,86	0,86	0,86	0,86
Resist. Biela	Vrd2 (kN)	1517,8	1517,8	1582,2	1701,4
Verificação	Vrd2 >= Vsd	**OK**	**OK**	**OK**	**OK**

A 13.5 CÁLCULO DA ARMADURA TRANSVERSAL

$V_{Rd3} = V_c + V_{sw}$

$V_c = V_{c1}$ (1+ M_0 / $M_{Sd,máx}$) < 2V_{c1} na flexo-compressão, com:

$V_{c1} = V_{c0}$ quando $V_{Sd} \leq V_{c0}$

$V_{c1} = 0$ quando $V_{Sd} = V_{Rd2}$, interpolando-se linearmente para valores intermediários.

$$V_{c0} = 0{,}6 . f_{ctd} . b_w . d$$

V_{sw} = (A_{sw} / s) 0,9 d f_{ywd} (cotg α + cotg θ) sen α

$$\sigma_c = \frac{M_0}{W_i} = \gamma_p \left(\frac{P_\infty}{A_c} + \frac{P_\infty \cdot e_p}{W_i}\right)$$

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Mecanism. Complem.	Vco (kN)	264,7	264,7	276,0	296,8
Excentric. Prot.	ep (m)	-0,39	-0,62	-0,78	-0,88
Prop. Geométricas Seção	Ac2 (m²)	1,13	1,13	1,13	1,13
	Wi2 (m³) (-)	-0,27	-0,27	-0,27	-0,27
Mom. Anula Compr.	Mo (kNm)	1965,6	2712,8	3219,2	3515,3
Mom. Solic. Trecho	Msd,máx(kNm)	3187,8	5114,3	6260,4	6653,9
Mecanism. Complem.	Vc1 (kN)	74,83	118,57	165,71	217,09
Mecanism. Complem.	Vc (kN)	120,97	181,46	250,92	331,78
Cortante resist. Asw	Vsw (kN)	967,81	656,58	381,28	125,06
Armadura Trasv.	Asw/s (cm²/m)	14,22	9,64	5,37	1,64

A 13.6 ARMADURA MÍNIMA

ABNT NBR 6118:2014
Item 17.4.1.1.1

$$\rho_{s,mín} = \frac{A_{sw,mín}}{s} \geq \frac{0{,}2 f_{ctm} \cdot b_{wm} \cdot \operatorname{sen}\alpha}{f_{ywk}}$$

Espaçamentos mínimos
Item 18.3.3.2

$$\begin{cases} \phi_t \begin{cases} \geq 5\ mm \\ \leq \frac{b_w}{10} \end{cases} \\ s_{t,longitudinal,máx} \begin{cases} \leq 0{,}6d \leq 30\ cm, se\ V_{sd} \leq 0{,}67V_{rd2} \\ \leq 0{,}3d \leq 20\ cm, se\ V_{sd} > 0{,}67V_{rd2} \end{cases} \\ s_{t,transversal,máx} \begin{cases} \leq d \leq 80\ cm, se\ V_{sd} \leq 0{,}2V_{rd2} \\ \leq 0{,}6d \leq 35\ cm, se\ V_{sd} > 0{,}2V_{rd2} \end{cases} \end{cases}$$

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura mínima	Asw,mín/s (cm²/m)	3,21	3,21	3,21	3,21
Armadura necessária	Asw,adot/s (cm²/m)	**14,22**	**9,64**	**5,37**	**3,21**

A 14 DIMENSIONAMENTO DA ARMADURA TRANSVERSAL DE TORÇÃO POR TRECHOS DA VIGA

A 14.1 PREPARAÇÃO DOS DADOS

A 14.1.1 DADOS CONCRETO

NBR 6118:2014	fck (MPa)	35,00
	fctk,inf (MPa)	2,25
	fctk,m (MPa)	3,21
Tabela 12.1 NBR	γc	1,40
Rsistência de cálculo	fcd (MPa)	25,00
	fctd (MPa)	1,60

A 14.1.2 DADOS DA ARMADURA ATIVA E PASSIVA

NBR 6118:2014	fyk (MPa)	500
	fpyk (MPa)	1710
Tabela 12.1 NBR	γs	1,15
Rsistência de cálculo	fyd (MPa)	435

A 14.1.3 COMBINAÇÃO NORMAL ÚLTIMA

$$F_d = \sum_{i=1}^{m} \gamma_{gi} G_i + \gamma_{q1} Q_1 + \sum_{j=2}^{n} \gamma_{qj} \psi_{oj} Q_j$$

Tab.2 NBR 8681:03	γg	1,35
Tab.5 NBR 8681:03	γq	1,5

A 14.1.4 PREPARAÇÃO DOS DADOS GEOMÉTRICOS DA SEÇÃO

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Mesa superior	m1 (m)	0,70	0,70	0,70	0,70
	t1 (m)	0,28	0,28	0,28	0,28
Alma	m2 (m)	0,68	0,68	0,68	0,68
	t2 (m)	0,50	0,25	0,25	0,25
Mesa inferior	m3 (m)	0,60	0,60	0,60	0,60
	t3 (m)	0,35	0,35	0,35	0,35
Inércia Torção mesa sup.	J1 (m4)	0,0049	0,0049	0,0049	0,0049
Inércia Torção alma	J2 (m4)	0,0281	0,0035	0,0035	0,0035
Inércia Torção mesa inf	J3 (m4)	0,0086	0,0086	0,0086	0,0086
Inércia Torção seção	J (m4)	0,0416	0,0169	0,0169	0,0169

ABNT NBR 6118:2014 - Item 17.5.1.4.1

$h_e \leq \frac{A}{u}$ $h_e \geq 2\ c_1$ $h_e = A/u \leq b_w - 2c_1$

Caso A/u resulte menor que $2c_1$

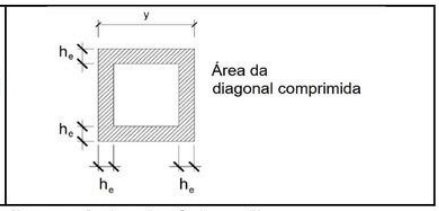


Figura A25 - Definição da seção resistente à torção

d' da armadura extremidade	c1 (m)	0,05	0,05	0,05	0,05
Área mesa sup.	A1 (m²)	0,1925	0,1925	0,1925	0,1925
Perímetro mesa sup.	u1 (m)	1,95	1,95	1,95	1,95
Área alma	A2 (m²)	0,34	0,17	0,17	0,17
Perímetro alma	u2 (m)	2,35	1,85	1,85	1,85
Área mesa inf.	A3 (m²)	0,21	0,21	0,21	0,21
Perímetro inf.	u3 (m)	1,90	1,90	1,90	1,90
Espessura parede mesa	he1 (m)	0,100	0,100	0,100	0,100
Área média mesa sup.	Ae1 (m²)	0,105	0,105	0,105	0,105
Espessura parede alma	he2 (m)	0,100	0,100	0,100	0,100
Área média almo	Ae2 (m²)	0,230	0,086	0,086	0,086
Espessura parede mesa inf.	he3 (m)	0,100	0,100	0,100	0,100
Área média mesa inf.	Ae3 (m²)	0,125	0,125	0,125	0,125

A 14.2 CONSIDERAÇÃO DA PARCELA DE TORÇÃO POR RETÂNGULO DA SEÇÃO

ABNT NBR 6118:2014 - Item 17.5.1.4.2

$$T_{Sdi} = T_{Sd} \frac{a_i^3 b_i}{\Sigma a_i^3 b_i}$$

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Momento torçor solicitante no trecho	Tg (kN)	15,142	15,142	13,092	5,1264
	Tq+ (kN)	106,38	106,38	106,38	106,38
	Tq- (kN)	-106,38	-106,38	-106,38	-106,38
Mom. torçor de cálculo	Tsd (cm²)	180,0	180,0	177,2	166,5
Mom. Torçor mesa sup.	Tsd,1 (cm²)	21,0	51,6	50,8	47,7
Mom. Torçor alma	Tsd,2 (cm²)	121,8	37,4	36,8	34,5
Mom. Torçor mesa inf.	Tsd,3 (cm²)	37,1	91,1	89,7	84,3

A 14.3 VERIFICAÇÃO DA BIELA COMPRIMIDA

$T_{Rd2} = 0{,}50\ \alpha_{v2}\ f_{cd}\ A_e\ h_e\ \text{sen}\ 2\ \theta$ $\qquad \alpha_{v2} = \left(1 - \frac{f_{ck}}{250}\right)$ $\qquad 30° \le \theta \le 45°$

Item 17.5.1.5 - NBR 6118:2014	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Inclinação biela	θbiela (graus)	36	36	36	36
Coeficiente ajuste	αv2	0,86	0,86	0,86	0,86
Resist. biela mesa sup.	Trd2,1 (kN)	107,4	107,4	107,4	107,4
Resist. biela alma	Trd2,2 (kN)	235,1	88,2	88,2	88,2
Resist. biela mesa inf.	Trd2,3 (kN)	127,8	127,8	127,8	127,8
Verificação mesa sup.	Trd2,1 >= Tsd,1	**OK**	**OK**	**OK**	**OK**
Verificação alma	Trd2,2 >= Tsd,2	**OK**	**OK**	**OK**	**OK**
Verificação mesa inf.	Trd2,3 >= Tsd,3	**OK**	**OK**	**OK**	**OK**

A 14.4 CÁLCULO DA ARMADURA TRANSVERSAL

$T_{Rd3} = (A_{90} / s)\ f_{ywd}\ 2A_e\ \text{cotg}\ \theta$	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura mesa sup.	A90/s,1	**1,67**	**4,10**	**4,04**	**3,79**
Armadura alma	A90/s,2	**4,43**	**3,62**	**3,56**	**3,35**
Armadura mesa inf.	A90/s,3	**2,48**	**6,09**	**6,00**	**5,63**

A 14.5 CÁLCULO DA ARMADURA LONGITUDINAL

$T_{Rd4} = (A_{s\ell} / u_e)\ 2A_e\ f_{ywd}\ \text{tg}\ \theta$	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura mesa sup.	Asl/ue,1	**3,17**	**7,77**	**7,65**	**7,19**
Armadura alma	Asl/ue,2	**8,38**	**6,85**	**6,75**	**6,34**
Armadura mesa inf.	Asl/ue,3	**4,70**	**11,54**	**11,36**	**10,67**

A 15 VERIFICAÇÃO DA FADIGA DA ARMADURA LONGITUDINAL E CONCRETO

A 15.1.1 PREPARAÇÃO DOS DADOS

	Seção:	S1	S2	S3	S4	S5
Propriedades geométricas da seção	Ac1 (m²)	0,57	0,57	0,57	0,57	0,57
	Ix1 (m4)	0,11	0,11	0,11	0,11	0,11
	Ac2 (m²)	1,13	1,13	1,13	1,13	1,13
	Ix2 (m4)	0,28	0,28	0,28	0,28	0,28
	ys1 (m)	0,64	0,64	0,64	0,64	0,64
	ys2 (m)	0,53	0,53	0,53	0,53	0,53
	ys2* (m)	0,25	0,25	0,25	0,25	0,25
Execentricidade das armaduras	ep1 (m)	0,00	-0,23	-0,40	-0,50	-0,53
	ep2 (m)	-0,39	-0,62	-0,78	-0,88	-0,92
	eAs (m)	-0,98	-0,98	-0,98	-0,98	-0,98
Força de prontensão nas	Pz (kN)	-4029,50	-4040,88	-4022,0	-4010,7	-4027,5
	P∞ (kN)	-3490,15	-3515,69	-3497,1	-3480,9	-3491,4
Área das armaduras	Ap (cm²)	33,30	33,30	33,30	33,30	33,30
	As (cm²)	16,97	16,97	16,97	16,97	16,97
Bitola das armaduras	ϕs (mm)	25,00	25,00	25,00	25,00	25,00
	ϕp (mm)	12,70	12,70	12,70	12,70	12,70
Esforços solicitantes no tempo infinito	Mg1 (kNm)	0,00	312,40	535,5	669,4	714,1
	Mg2 (kNm)	0,00	444,06	761,3	951,6	1015,0
	Mg3 (kNm)	0,00	291,77	500,2	625,2	666,9
	Mq+ (kNm)	0,00	1014,92	1625,4	1985,2	2112,7
	Mq- (kNm)	0,00	-52,30	-106,97	-155,99	-146,37
	Nr (kN)	-31,35	-31,35	-31,35	-31,35	-31,35
	Mr(kNm)	12,36	12,36	12,36	12,36	12,36
	$N_{\Delta T}$(kNm)	-823,28	-823,28	-823,28	-823,28	-823,28
	$M_{\Delta T}$(kNm)	324,50	324,50	324,50	324,50	324,50

A 15.1.2 DADOS FADIGA

ABNT NBR 6118:2014
Item 23.5
Combinação Frequente

$$F_{d,ser} = \sum_{i=1}^{m} F_{gik} + \psi_1 F_{q1k} + \sum_{j=2}^{n} \psi_{2j} F_{qjk}$$

Tab.7 NBR 8618:2003	ψ1	0,5
Item 23.5.3	γf	1,0
	γc	1,4
	γs	1,0
	ξ	0,4
Tabela 23.2	Δfpd,fad (MPa)	110,0
	Δfsd,fad (MPa)	190,0
Item 23.5.3	αp28	10,0
	αs28	10,0
Item 23.5.4.1	fcd,fad (MPa)	11,3

A 15.2 FADIGA ARMADURA

ABNT NBR 6118:2014
Item 23.5.3

$$\eta_s \cdot \gamma_f \cdot \Delta\sigma_s \leq \Delta f_{ds,fad}$$

$$\eta_s = \frac{1+\frac{A_p}{A_s}}{1+\frac{A_p}{A_s}\sqrt{\xi\frac{\phi_s}{\phi_p}}} \geq 1$$

Verificação no Estádio I (Seção antende ao ELS-F)
Cavalcante (2019) Cap. 3, pág 81

$$\Delta\sigma_p \; ou \; \Delta\sigma_s = \alpha_s \cdot \frac{\psi_1 \Delta M_q \cdot e_s}{I_c}$$

	Seção	S1	S2	S3	S4	S5
Coef. Ajuste. Ader.	ηs	1,08	1,08	1,08	1,08	1,08
Var. Tensão Ap	ηs γf Δσp (MPa)	0,00	12,59	25,90	36,06	39,48
Verificação	Fadiga Ap	**OK**	**OK**	**OK**	**OK**	**OK**
Var. Tensão As	ηs γf Δσs (MPa)	0,00	19,87	32,26	39,87	42,06
	k fadiga	1,00	1,00	1,00	1,00	1,00
	As,fad	16,97	16,97	16,97	16,97	16,97
Verificação	Fadiga As	**OK**	**OK**	**OK**	**OK**	**OK**

A 15.3 FADIGA CONCRETO

ABNT NBR 6118:2014 - Item 23.5.4

$$f_{cd,fad} = 0{,}45\, f_{cd}$$

$$\eta_c \cdot \gamma_f \cdot \sigma_{c,máx} \leq f_{cd,fad}$$

$$\eta_c = \frac{1}{1{,}5 - 0{,}5(|\sigma_{c1}|/|\sigma_{c2}|)}$$

Cálculo das tensões no concreto, no Estádio I

$$\sigma_c = \boxed{\frac{P_z}{A_1}} - \frac{(P_z - P_\infty)}{A_2} \boxed{+\frac{P_z \cdot e_{p1}}{W_1}} - \frac{(P_z - P_\infty) \cdot e_{p2}}{W_2} \boxed{-\frac{M_{g1}}{W_1} - \frac{M_{g2}}{W_1}} - \frac{M_{g3}}{W_2} - \frac{\psi_i M_q}{W_2}$$

σc1 - Menor valor a menos de 30cm da borda sup.
σc2 - Maior valor a menos de 30cm da borda sup.

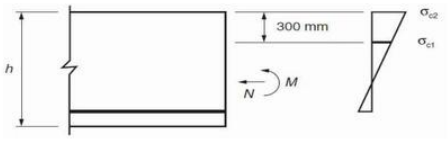


Figura A26 - Altura para considerar as tensões no concreto para ao cálculo de fadiga

A 15.3.1 TENSÕES NO CONCRETO NA FAIXA ALTURA DA LAJE

	Seção	S1	S2	S3	S4	S5
Borda inferior laje	$\sigma c1$(MPa)	0,29	-0,07	-0,31	-0,44	-0,50
Borda superior laje	$\sigma c2$(MPa)	0,08	-0,65	-1,15	-1,45	-1,57
Máxima	$\sigma_{c,máx}$ (MPa)	0,08	-0,65	-1,15	-1,45	-1,57
Coef. Ajuste.	ηc	-4,32	0,69	0,73	0,74	0,75
Verificação	$\eta c\ \gamma f\ \lvert\sigma_{c,máx}\rvert$	-0,36	0,45	0,84	1,07	1,17
	Fadiga σc	**OK**	**OK**	**OK**	**OK**	**OK**

A 15.3.2 TENSÕES NO CONCRETO NA ALTURA DA VIGA

	Seção	S1	S2	S3	S4	S5
Distância de $\sigma c1$ da LN	y $\sigma c1$ (m)	0	0	0	0	0
a 30 cm ou LN viga	$\sigma c1$(MPa)	-6,58	-6,61	-6,58	-6,55	-6,58
Borda super. Viga	$\sigma c2$(MPa)	-6,76	-7,19	-7,41	-7,55	-7,61
Máxima	$\sigma_{c,máx}$(MPa)	-6,76	-7,19	-7,41	-7,55	-7,61
Coef. Ajuste.	ηc	0,99	0,96	0,95	0,94	0,94
Verificação	$\eta c\ \gamma f\ \lvert\sigma_{c,máx}\rvert$	6,67	6,91	7,02	7,08	7,12
	Fadiga σc	**OK**	**OK**	**OK**	**OK**	**OK**

A 16 VERIFICAÇÃO DA FADIGA DA ARMADURA TRANSVERSAL

A 16.1 DIMENSIONAMENTO

A 16.1.1 PREPARAÇÃO DOS DADOS DA SEÇÃO

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Altura útil seção	d (m)	1,264	1,264	1,31764	1,41691
Inclinação Biela	θbiela (graus)	36,00	36,00	36,00	36,00
Inclinação Estribo	αestribo (graus)	90,00	90,00	90,00	90,00
Esforços solicitantes	Vg1 (kN)	142,8125	107,109	71,4063	35,7031
	Vg2 (kN)	203,00	152,25	101,50	50,75
	Vg3 (kN)	133,38	100,035	66,69	33,345
	Vq+ (kN)	46,227977	115,205	167,325	196,759
	Vq- (kN)	-515,0924	-402,24	-316,76	-252,01
Resist. Aço	fyd (MPa)	434,78	434,78	434,78	434,78
V resistido por Asw	Vsw (kN)	967,81	656,58	381,28	125,06
Contrib. Protensão	Vp (kN)	330,76	250,50	166,40	82,90

A 16.1.2 DADOS FADIGA

Item 23.5.3 - ABNT NBR 6118:2014
Correção da Inclinação da Biela Modelo II

$$\mathrm{tg}\,\theta_{corr} = \sqrt{\mathrm{tg}\,\theta} \leq 1$$

$V_{sw} = (A_{sw} / s)\ 0{,}9\ d\ f_{ywd}\ (\mathrm{cotg}\,\alpha + \mathrm{cotg}\,\theta)\ \mathrm{sen}\,\alpha$

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Correção de θ	θcorrigido (graus)	40,44	40,44	40,44	40,44
Correção de Asw	Asw/s,corr (cm²/m)	7,49	7,49	7,19	6,68
Conforme detalhado	Asw/s,adot (cm²/m)	14,22	9,64	7,19	6,68
Redução de 50%Vc	0,5*Vc (kN)	60,49	90,73	125,46	165,89

A 16.2 COMBINAÇÃO FREQUENTE

$$F_{d,ser} = \sum_{i=1}^{m} F_{gik} + \psi_1\ F_{q1k} + \sum_{j=2}^{n} \psi_{2j}\ F_{qjk}$$

Tab.7 NBR 8681:2003	ψ1	0,5
Item 23.5.3	γf	1,0
	γc	1,4
	γs	1,0
Tabela 23.2	Δfs,fad (MPa)	85,0

A 16.3 FADIGA ESTRIBO

Cavalcante (2019) Cap. 3, pág 85

$$\sigma_{sw} = \frac{(V - 0{,}5V_c)}{0{,}9d} \cdot \frac{s}{A_{sw}}$$

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Contantes máximas e mínimas	Vsmáx (kN)	502,31	417,00	323,26	218,18
	Vsmín (kN)	479,19	359,39	239,60	119,80
Contantes efetivas	Vsmáx,efet (kN)	171,54	166,50	156,86	135,27
	Vsmín,efet (kN)	148,43	108,90	73,19	36,89
Tensões máximas e mínimas stribos	σsw,máx (MPa)	68,67	69,06	36,83	0,00
	σsw,mín (MPa)	54,38	16,56	0,00	0,00
Variação de tensões	Δσsw (MPa)	14,29	52,50	36,83	0,00
	k fadiga	1,00	1,00	1,00	1,00
	Asw/s,fad (cm²/m)	14,22	9,64	7,19	6,68
Verificação	Fadiga Asw	**Ok**	**Ok**	**Ok**	**Ok**

A 17 VERIFICAÇÃO DA DEFORMAÇÃO EXCESSIVA

A 17.1 COMBINAÇÃO QUASE PERMANENTE

$$F_{d,ser} = \sum_{i=1}^{m} F_{gi,k} + \sum_{j=1}^{n} \Psi_{2j} . F_{qj,k}$$

A 17.3 FLECHA DIFERIDA MEF

ABNT NBR 6118:2014 Item 17.3.2.1.3

Flechas obtidas pelo modelo em elementos finitos

$f_\infty = f_o(1+\phi)$	Flecha\ Seção	S5
f (cronograma)	fg1 (ato) (mm)	-22,047
f (carga móvel)	fq (mm)	-6,18
Tab. 6 NBR 8618:03	ψ2	0,30
Vão Viga	L (m)	20
Tab. 13.3 - NBR 6118	flim (L/250) (mm)	-80
f total infinito	ftot,∞ (mm)	-23,90
Verificação	Verif. f (ato)	**OK**

A 18 ARMADURA DE CISALHAMENTO LAJE/VIGA

A 18.1 DADOS

A 18.1.1 PREPARAÇÃO DOS DADOS

Resis. Aço	fyk (MPa)	500,00
Tab. 12.1 NBR 6118	γs	1,15
Resis. Cálculo Aço	fyd (MPa)	434,78
Resis. Compr. Concr.	fck (MPa)	35,00
Resis. Traç. Concr.	fctk,inf (MPa)	2,25
Tab. 12.1 NBR 6118	γc	1,40
Resis. Cálc. Compr.	fcd (MPa)	25,00
Resis. Calc. Traç.	fctd (MPa)	1,60
Momentos fletores atuantes na seção do centro do vão	Mg1 (kNm)	714,06
	Mg2 (kNm)	1015,00
	Mg3 (kNm)	666,90
	Mq+ (kNm)	2112,72
	$N_{\Delta T+}$ (kN)	1152,59

A 18.1.2 COMBINAÇÃO NORMAL ÚLTIMA

$$F_d = \sum_{i=1}^{m} \gamma_{gi} G_i + \gamma_{q1} Q_1 + \sum_{j=2}^{n} \gamma_{qj} \psi_{oj} Q_j$$

Tab.2 NBR 8681:03	γg	1,35
Tab.5 NBR 8681:03	γq	1,50
Tab.5 NBR 8681:03	γq (ΔT)	1,20
Tabela 11.1 NBR 6118:2014	γp,fav	0,90
	γp,desf	1,20
Tab. 6 NBR 8681:03	ψ_0 (ΔT)	0,60

A 18.2 VERIFICAÇÃO DA TENSÃO DE CISALHAMENTO

ABNT NBR 9062:2017
Item 6.3

$$\tau_{sd} \leq \beta_s \frac{f_{yd} A_s}{bs} + \beta_c f_{ctd} < 0,25 f_{cd}$$

$$\tau_{sd} = \frac{F_{md}}{a_v \times b}$$

Modelo de El Debs (2017)

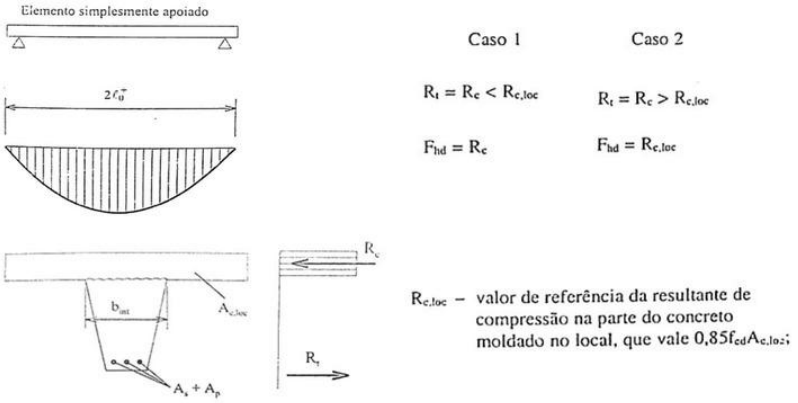


Figura A27 - Modelo para consideração da força de cisalhamento na interface, de acordo com El Debs (2017)

A 18.3 CÁLCULO DA ARMADURA DE COSTURA

	Trecho	S4-S5
Mesa colab. Laje	b,laje (m)	2,00
Interface laje/viga	b, interf. (m)	0,70
Espessura laje	t (m) (laje)	0,28
Área estribo cort.	As1 (cm²/m)	10,03
Área estribo suplementar	As2 (cm²/m)	9,20
Taxa arm. Costura	ρ (%)	0,27
Braço de alavanca	z (m)	1,38
Cortante devido a Mg e Mq	Rccd,tot (kN)	4636,28
Cortante devido a ΔT+	$N_{\Delta T+,d}$ (kN)	829,87
Comb. Normal Última	Fhd	5466,14
Altura Linha Neutra	xLN (m)	0,172
Dist.Entre Mom.Nulo	av (m)	20,00
Cisalham. Solic.	τsd (kN/m²)	780,88
Tabela 10 NBR 9062:2017	βs	0,22
	βc	0,37
Cisalham. Resistente	τrd (kN/m²)	869,18
Verificação	τrd > τsd	**OK**

A 19 ESTABILIDADE LATERAL DE VIGA PRÉ-MOLDADA

Modelo de El Debs (2017) pág 209 e Timoshenko (1961) pág 267

$$q_{crit} = k\frac{\sqrt{(E_{cs}I_y)(G_cJ_c)}}{l^3}$$

A 19.1 PREPARAÇÃO DOS DADOS

Prop. Concreto	Ecs (GPa)	29,40
	Gc (GPa)	12,25
Prof. Aço CA 50	fyd (MPa)	434,78
Inércia Flexão Lat.	Iy (m^4)	0,014004
Inércia Torção	Jc (m4)	0,016943
Vão viga	L (m)	20,00
Altura da longarina PM	h (m)	1,30
Largura mesa comprimida	bf (m)	0,70
Peso próprio	qpp (kN/m)	14,28
PP concr. Laje	glaje (kN/m)	20,30
Carga utiliz. Concr.	qsobr. (kN/m)	5,46

A 19.2 VERIFICAÇÃO SIMPLIFICADA

ABNT NBR 9062:2017 item 6.1

$\frac{L}{b_f} \le 50$ $\frac{L \times h}{(b_f)^2} \le 500$	L/bf	28,57143
	Verificação	**OK**
	Lh/(bf)²	53,1
	Verifcação	**OK**

A 19.3 COMBINAÇÃO DE CONSTRUÇÃO

ABNT NBR 8681:2003
Item 5.1.3.2

$$F_d = \sum_{i=1}^{m} \gamma_{gi} F_{Gi,k} + \gamma_q \left[F_{Q1,k} + \sum_{j=2}^{n} \psi_{0j,ef} F_{Qj,k} \right]$$

ABNT NBR 9062:2017
Amplificação dinâmica Item 5.3.2

$$g_{eq,d} \ge \gamma_f . \beta_a . g_k$$

Tab. 2 e 5 NBR 8681	γg	1,25
	γq	1,30
Item 5.3.2 NBR 9062	βa	1,40
	γf	1,30
Carga cálculo	qd (kN/m)	58,46

A 19.4 DETERMINAÇÃO DA CARGA DISTRIBUÍDA CRÍTICA

El Debs (2017)	k	28,80
pág.209	q crit (kN/m)	1052,49
qcrit ≥ qd	Verificação	**OK**

A 19.5 ALÇA INTERNA DE IÇAMENTO - CABO VERTICAL

Item 5.3.2 NBR 9062	βa	3,00
$F_{v,iça.} = \gamma_f \beta_a \frac{L \cdot g_{pp}}{2}$	$F_{iça}$ (kN)	556,97
	Ast,iça (cm²)	12,81

A 20 ARMADURAS DE FRETAGEM NA REGIÃO DE ANCORAGEM

Modelo indicado no Cap. 7 do livro Técnicas de armar as estruturas de concreto (Fusco,2013)

A 20.1 DADOS

Comprim. de regularização	a (m)	1,30
largura equivalente da seção	b (m)	0,44
Resist. De cálculo Aço CA-50	fyd (kN/m²)	434783
exc. protensão na seção de a	ep1 (m)	-0,1287
Protensão inicial sem perdas	P0 (kN)	-4.331
Número de ancoragens	n (cabos)	3,00
Força em cada ancoragem	Fc0 (kN)	-1.444
Área bruta da seção	Ac1 (m²)	0,57125
Módulo de flexão inferior	Wi1 (m³) (-)	-0,163
Módulo de flexão superior	Ws1 (m³)	0,167
Coef. Ponderador Protensão	γp	1,10

A 20.2 CÁLCULO DAS TENSÕES NA ZONA DE REGULARIZAÇÃO

$$\sigma_{ato} = P_0 \cdot \left(\frac{1}{A_1} + \frac{e_{p1}}{W_1}\right)$$

Tensão na borda superior da seção	σ_{cs} (kN/m²)	-4677
Tensão na borda inferior da seção	σ_{ci} (kN/m²)	-12112

A 20.3 FORÇA DE FISSURAÇÃO SUPERFICIAL (CABO CRÍTICO = CABO SUPERIOR)

$$A_{st0} := \frac{R_{t0}}{f_{yd}}$$

$$R_{t0} := \left[0.04 + 0.2 \cdot \left(\frac{d_1 - d_2}{d_1 + d_2}\right)^3\right] \cdot F_{c0}$$

Distância até a borda mais afastada	d1 (m)	0,92945
Distância até a borda mais próxima	d2' (m)	0,371
Resultante de tração superficial	Rt0 (kN)	88,77
Área de aço - fretagem superficial	Ast0 (cm²)	**2,04**

A 20.4 FORÇA DE FENDILHAMENTO NOS BLOCOS DE ANCORAGEM (CABO CRÍTICO = INTERMEDIÁRIO)

$$A_{st1} := \frac{R_{t1}}{f_{yd}}$$

$$R_{t1} := 0.3 \cdot \left(1 - \frac{a_0}{a}\right) \cdot F_{c0}$$

Dimensão da placa de ancoragem	a0 (m)	0,30
Resultante de tração de fendilhamento	Rt1 (kN)	366,50
Área de aço - fretagem na ancoragem	Ast1 (cm²)	**8,43**

A 20.5 FORÇA TRANSVERSAL DE EQUILÍBRIO

$$R_{cy} := \frac{b \cdot a}{2} \cdot (\sigma_s + \sigma_i)$$

$$e_y := a - \frac{a}{3} \cdot \left(\frac{2 \cdot \sigma_i + \sigma_s}{\sigma_s + \sigma_i} \right)$$

$$M_y := F_{c0} \cdot (d_1 + d_2 + d_3) - R_{cy} \cdot e_y$$

$$A_{st2} := \frac{R_{t2}}{f_{yd}}$$

Altura da ancoragem superior	d1 (m)	0,92945
Altura da ancoragem intermediária	d2 (m)	0,65945
Altura da ancoragem inferior	d3 (m)	0,38945
Resultante de compressão horizontal	Rcy (kN)	4795
Excentridicade da resultante de equilíbrio	ey (m)	0,55
Momento de equilíbrio	My (kNm)	485,05
Resultante de tração vertical de equilíbrio	Rt2 (kN)	820,85
Área de aço - armadura de equilíbrio	Ast2 (cm²)	**18,88**

A 21 RESUMO DOS DETALHAMENTOS

A 21.1 ARMADURAS INICIALMENTE ADOTADAS EM SEPARADO

A 20.1.1 ARMADURA LONGITUDINAL DE FLEXÃO

	Seção	S1	S2	S3	S4	S5
Armadura Necessária	Ast (cm²)	17,0	17,0	17,0	17,0	17,0

Armadura longitudinal de pele

Armadura pele fora do engrossamento	As,pele (cm²)	1,3

A 21.1.2 ARMADURA LONGITUDINAL DE TORÇÃO

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura mesa sup.	Asl/ue,1	3,2	7,8	7,7	7,2
Armadura alma	Asl/ue,2	8,4	6,9	6,7	6,3
Armadura mesa inf.	Asl/ue,3	4,7	11,5	11,4	10,7

A 21.1.3 ARMADURA TRANSVERSAL DE TORÇÃO

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura mesa sup.	A90/s,1	1,7	4,1	4,0	3,8
Armadura alma	A90/s,2	4,4	3,6	3,6	3,3
Armadura mesa inf.	A90/s,3	2,5	6,1	6,0	5,6

A 21.1.4 ARMADURA TRANSVERSAL DE CISALHAMENTO

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura necessária	Asw,adot/s (cm²/m)	14,2	9,6	7,2	6,7

A 21.2 DETALHAMENTOS FINAIS: ARMADURAS AGRUPADAS

A 21.2.1 ARMADURAS LONGITUDINAIS

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura de flexão + torção na mesa inferior	Asl,f+t,m.i. (cm²)	19,3	22,7	22,6	22,3
	ϕ (mm)	25	25	25	25
	n barras	4	5	5	5
Armadura de torção no perímetro da mesa inferior	Asl,t,m.i. tor (cm²)	2,4	5,8	5,7	5,3
	ϕ (mm)	12,5	12,5	12,5	12,5
	n barras	2	5	5	5
Armadura de torção + pele na alma	Asl,p+t,alma (cm²)	5,4	3,4	3,4	3,2
	ϕ (mm)	12,5	12,5	12,5	12,5
	n barras	5	3	3	3
Armadura de torção no perímetro da mensa superior	Asl,t,m.s. (cm²)	3,2	7,8	7,7	7,2
	ϕ (mm)	12,5	12,5	12,5	12,5
	n barras	3	7	7	6

A 21.2.2 ARMADURAS TRANSVERSAIS

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura transversal cortante	Asw/s,cort (cm²/m)	18,64	13,26	10,75	10,03
	n ramos	2	2	2	2
	ϕ (mm)	12,5	12,5	12,5	12,5
	s (cm)	14	19	23	25
Armadura transversal torção mesa superior	Ast/s,m.s. (cm²/m)	1,67	4,10	4,04	3,79
	ϕ (mm)	10	10	10	10
	s (cm)	14	19	i pequ.!	i pequ.!
Armadura transversal torção mesa inferior	Ast/s,m.i. (cm²/m)	2,48	6,09	6,00	5,63
	ϕ (mm)	12,5	12,5	12,5	12,5
	s (cm)	14	19	i pequ.!	i pequ.!

A 21.2.3 ARMADURA DE COSTURA INTERFACE LAJE/VIGA

	Trecho	S1-S2	S2-S3	S3-S4	S4-S5
Armadura adicional de costura da laje, já considerando costura dos estribos de cortante	As,cos/s (cm²/m)	9,2	9,2	9,2	9,2
	n ramos	2	2	2	2
	ϕ (mm)	12,5	12,5	12,5	12,5
	s (cm)	14	19	23	25

A 21.2.4 ESQUEMA DE DETALHAMENTO ARMADURA PASSIVA

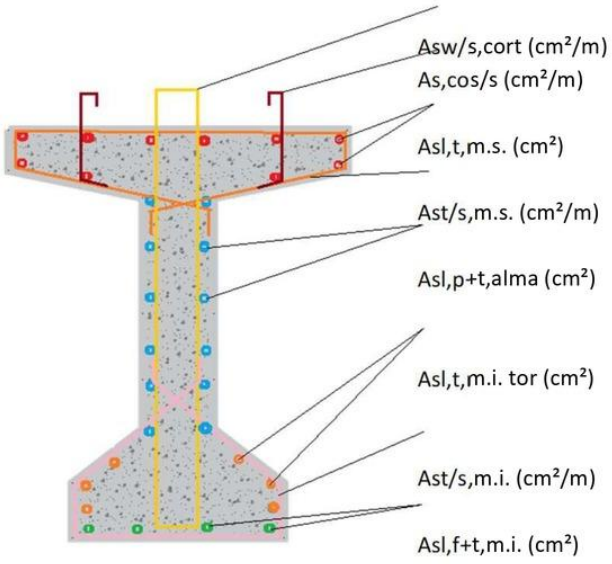
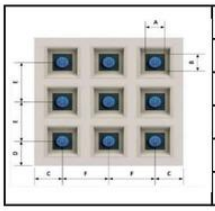
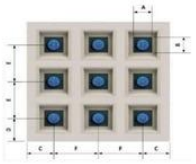


Figura A28 - Esquema de detalhamento das aramduras da longarina

A 21.3 ARMADURA ATIVA



Pi,total (kN)	4331,4
ϕ cordoalha (mm)	12,7
N cord, adot	11
n (cabos)	3
Ap (cm²), adot	33,3
Pi/ cabo (kN)	1443,8



	ϕ bainha (m)	0,065				
	av (m)	0,065				
	E (m)	0,270				
	hlong. (m)	1,30				
	y cg (m)	0,65945				
	Seção	S1	S2	S3	S4	S5
	x (m)	0	2,5	5	7,5	10
Cabo central	exc. cabo equiv. (m)	0	-0,232	-0,397	-0,496	-0,529
	y cabo central (m)	0,659	0,428	0,262	0,163	0,130
	α (rad)	0,105	0,079	0,053	0,026	0,000
Cabo Superior	y cabo superior (m)	0,929	0,580	0,330	0,180	0,130
	α (rad)	0,159	0,119	0,080	0,040	0,000
Cabo Inferior	y cabo inferior (m)	0,389	0,276	0,195	0,146	0,130
	α (rad)	0,048	0,152	0,154	0,156	0,157

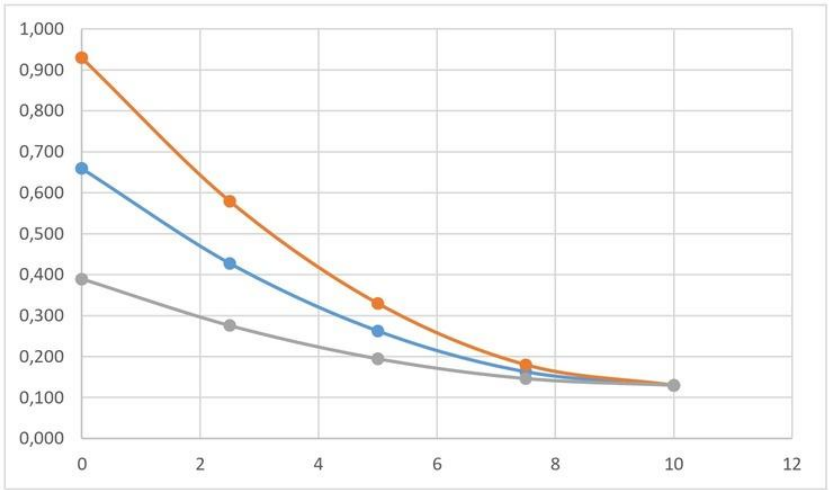


Figura A29 - Esquema de traçado dos cabos

A 21.4 ALONGAMENTO NO ATO DA PRONTENSÃO

Cálculo do alongamento do cabo no ato da prontesão

Item\Seção		S1	S2	S3	S4	S5
Seções	x (m)	0	2,5	5	7,5	10
Comprimento por Trecho	$s_{Cabo,sup.}$ (m)	3,03	4,04	4,04	4,04	0
	$s_{Cabo,interm.}$ (m)	3,01	4,02	4,02	4,02	0
	$s_{Cabo,inf.}$ (m)	3,00	4,00	4,00	4,00	0
Protensão com perda por atrito	$P_{(\mu)}$ (kN)	1624	1610	1595	1581	1567
	$\sigma p_{(\mu)}$ (kN/m²)	1463	1450	1437	1425	1412
	Ep(GPa)	200	200	200	200	200
Alongamento por Trecho	$s_{Cabo,sup.}$ (mm)	22,16	29,29	29,04	28,78	0
	$s_{Cabo,interm.}$ (mm)	22,02	29,15	28,89	28,64	0
	$s_{Cabo,inf.}$ (mm)	21,95	29,00	28,75	28,50	0
Along. Total de Cada Lado	$s_{Cabo,sup.}$ (mm)	**109,28**				
	$s_{Cabo,interm.}$ (mm)	**108,70**				
	$s_{Cabo,inf.}$ (mm)	**108,19**				
Along. Total Para os dois lados	$s_{Cabo,sup.}$ (mm)	**218,55**				
	$s_{Cabo,interm.}$ (mm)	**217,40**				
	$s_{Cabo,inf.}$ (mm)	**216,39**				
Comprimento bainhas	$s_{Cabo,sup.}$ (m)	**30,3**				
	$s_{Cabo,interm.}$ (m)	**30,1**				
	$s_{Cabo,inf.}$ (m)	**30,0**				
Comprimento cabos (1m excedente)	$s_{Cabo,sup.}$ (m)	**32,3**				
	$s_{Cabo,interm.}$ (m)	**32,1**				
	$s_{Cabo,inf.}$ (m)	**32,0**				

A 21.4 ARMADURAS DE FRETAGEM - REGIÃO DE ANCORAGEM DA PROTENSÃO

Armadura de fretagem superficial

Armadura de fretagem na superfície abaixa da placa de ancoragem (malha com espaçamento igual nas duas direções)	Ast0 (cm²)	2,04
	ϕ (mm)	10,0
	n ramos	3
	a1 (cm)	27,0
	s (cm)	13,00

Armadura de fretagem ancoragem

Armadura de fretagem no comp. 2/3 de a1 (armadura em camadas, = espaçamento nas 2 direções em cada camada)	Ast1 (cm²)	8,43
	ϕ (mm)	10,0
	n camadas	5
	n ramos	3
	2/3 a1 (cm)	18,0
	s entre ramos (cm)	13,0
	s entre camadas(cm)	4,0

Armadura de equilíbrio

Armadura vertical em forma de estribos que não até o topo da laje.	Ast2 (cm²)	18,88
	ϕ (mm)	16
	n ramos	2
	n camadas	5
	a/2 (cm)	65,00
	s (cm)	16,0

Armadura de pele no engrossamento

Armadura horizontal na face do engrossamento	Ac/alma (m²)	0,78
	As/pele (cm²)	7,8
	ϕ (mm)	10
	s (cm)	10,0

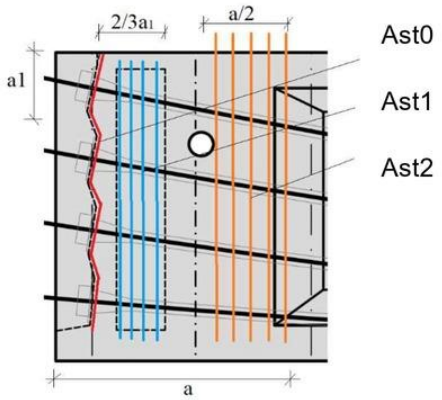


Figura A30 - Disposição das armaduras de fretagem

A 21.5 ARMADURAS INTERNAS DE IÇAMENTO

Alças internas de dois ramos para içamento por furo	Ast,iça (cm²)	12,81
	ϕ (mm)	20
	n alças de 2 ramos	3

ANEXO B

Planilha de dimensionamento de laje e pré-laje

O anexo B contém a planilha de dimensionamento referente às lajes e pré-lajes das pontes pré-moldadas e protendidas de concreto (PPC), desenvolvida dentro do grupo de pesquisa conforme produto do Relatório Técnico M3.A2 (DNIT, 2022). Os dados preenchidos são da tipologia PPC-A-20-P.

Planilha de dimensionamento da laje e pré-laje

A 1 PROPRIEDADES DOS MATERAIS

A 1.1 CONCRETO

ABNT NBR 6118:2014	CAAmbiental	CAA III
Itens: 6.4; 7.4 e 8.2, 17.3	fck (MPa)	35
$f_{ct,m} = 0,3\ f_{ck}^{2/3}$	fctm (MPa)	3,21
$f_{ctk,inf} = 0,7\ f_{ct,m}$	fctk,inf (MPa)	2,247
$f_{ctk,sup} = 1,3\ f_{ct,m}$	fctk,sup (MPa)	4,173
$E_{ci} = \alpha_E \cdot 5600\sqrt{f_{ck}}$	α_E (agregado)	1,00
$E_{cs} = \alpha_i \cdot E_{ci}$ $\quad \alpha_i = 0,8 + 0,2 \cdot \frac{f_{ck}}{80} \leq 1,0$	Eci (GPa)	33,13
$\alpha_E = 1,0$ para granito e gnaisse	αi	0,9
$\sigma_c = 0,85\ f_{cd}\left[1-\left(1-\frac{\varepsilon_c}{\varepsilon_{c2}}\right)^n\right]$ Para $f_{ck} \leq 50$ MPa: n=2	Ecs (GPa)	29,40
	ec2	0,20%
	ecu	0,35%

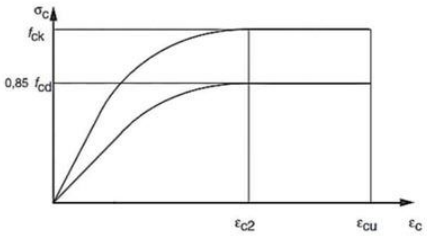


Figura A1 - Comportamento do concreto na compressão

A 1.2 AÇO ARM. PASSIVA

ABNT NBR 6118:2014	CA-50	
Itens: 8.3	fyk (MPa)	500
ABNT NBR 7480:2007	Es (GPa)	210
$\alpha_s = \frac{E_s}{E_{ci}}$	αs	6,34
	CA-60	
	fyk (MPa)	600
	Es (GPa)	210

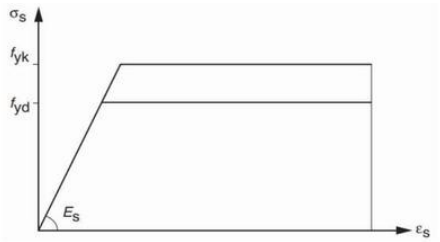


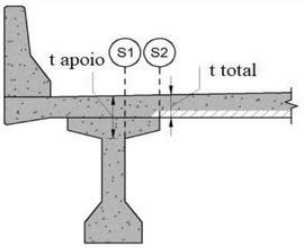
Figura A2 - Comportamento do Aço passivo

A 2 DIMENSÕES DA LAJE

Tabela 7.2 NBR 6118:2014

Cobrimento	c (m)	0,025

A 2.1 LAJE CENTRAL



DADOS		
Espessura total laje	t_{laje} (m)	0,28
Espessura no apoio	t_{apoio} (m)	0,63
Vão livre	$L_{vão}$ (m)	2,20
Vão teórico	$L_{vão,t}$ (m)	2,37

Item 14.6.2.4 NBR 6118:2014

$$\ell_{ef} = \ell_0 + a_1 + a_2$$

$$a_1 \, e \, a_2 \leq \frac{b_{apoio}}{2} \, ou \, 0{,}3 t_{laje}$$

A 2.2 LAJE BALANÇO

Espessura total	t_{bal} (m)	0,28
Vão livre	L_{bal} (m)	0,68
Vão teórico	$L_{bal,t}$ (m)	0,76

A 2.3 DEFENSA

Altura defensa	h def (m)	0,87

A 2.4 PRÉ-LAJES

Espessura pré-laje	t prel (m)	0,08
Largura pré-laje	bw,prel (m)	1,00
Vão livre pré-laje	L_{prel} (m)	2,20
Vão teórico pré-laje	$L_{prel,t}$ (m)	2,28

3 ESFORÇOS SOLICITANTES

3.1 COEFICIENTE DE IMPACTO

ABNT NBR 7188:2013

$$CI = CIV \cdot CNF \cdot CIA$$

sendo

$$CIV = \begin{cases} 1{,}35 & se\ L \leq 10{,}0m \\ 1 + 1{,}06\left(\frac{20}{L+50}\right), & se\ 10 < L \leq 200m \end{cases}$$

$$CNF = 1 - 0{,}05(n-2) \geq 0{,}9$$

$$CIA = \begin{cases} 1{,}25, para\ obras\ em\ concreto\ ou\ mistas \\ 1{,}15, para\ obras\ em\ aço \end{cases}$$

Vão da ponte	L (m)	20
Número de faixas	n	2
Coef. Impacto Vertical	CIV	1,30
Coef. Número de faixas	CNF	1,00
Cosiderar CIA?	CIA	não
Coef. Impacto Adicional	CIA	1,00
Coef. Impacto	CI	1,30

3.2 MOMENTOS FLETORES

x - direção longitudinal

y - direção transversal

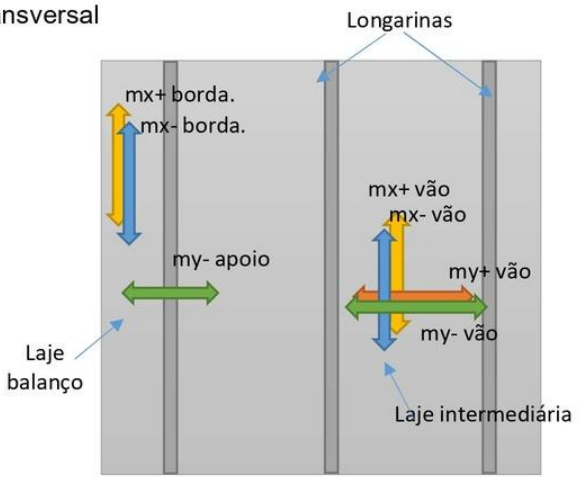


Figura A3 - Esquema de esforços na laje

3.3 SEÇÃO LAJE CENTRO DO VÃO

Mpp centro vão	Mgy (kNm/m)	9,47
	Mgx (kNm/m)	21,01

Consideração do CI		Sem CI	Com CI
M carga móvel no centro do vão	Mqy+ (kNm/m)	31,71	41,31
	Mqy- (kNm/m)	-28,61	-37,27
M carga móvel no centro do vão	Mqx+ (kNm/m)	28,10	36,62
	Mqx- (kNm/m)	-15,60	-20,32

3.4 SEÇÃO APOIO LAJE EM BALANÇO

ABNT NBR 7187:2013 Item 5.2.3

Mpp apoio	Mgy (kNm/m)	-10,17

Consideração do CI		Sem CI	Com CI
M carga móv. Apoio	Mqy+ (kNm/m)	12,38	16,13
	Mqy- (kNm/m)	-36,68	-47,79

M impacto na defensa (efeito no apoio)	Fimp (kN)	100,00
	himp (m)	1,01
	Midy (kNm/m)	-40,08

3.5 SEÇÃO EXTREMIDADE DA LAJE

Mpp borda	Mgx (kNm/m)	21,01

Consideração do CI		Sem CI	Com CI
M carga móv. Borda	Mqx+ (kNm/m)	28,10	36,62
	Mqx- (kNm/m)	-15,60	-20,32

3.6 FORÇA CORTANTE

V permanente S1	Vg1 (kN/m)	28,30
V permanente S2	Vg2 (kN/m)	23,35

Consideração do CI		Sem CI	Com CI
V variável S1	Vq1+ (kN/m)	9,81	12,78
V variável S2	Vq2+ (kN/m)	22,19	28,91

3.7 PRÉ-LAJE

Sobrecarga utilização	$q_{util.}$ (kN/m²)	2,00
Espessura pré-laje	t prel (m)	0,08
Largura pré-laje	bw,prel (m)	1,00
Vão teórico	Lprel,t (m)	2,28
Espessura Laje	t laje (m)	0,28
Peso Concreto fresco	$g_{concr.}$ (kN/m³)	25,00
Peso próprio	g_{pp} (kN/m)	2,00
Concreto fresco	gcf (kN/m)	5,00
Sobrecarga utilização	$q_{util.}$ (kN/m)	2,00
Momento Peso próprio	M_{pp} (kNm)	1,30
Momento Concr. Fresco	M_{cf} (kNm)	3,25
Momento q utilização	M_{util} (kNm)	1,30
Cortante Peso próprio	V_{pp} (kNm)	2,28
Cortante Concr. Fresco	V_{cf} (kNm)	5,70
Cortante q utilização	V_{util} (kNm)	2,28

A 4 DIMENSIONAMENTO ARMADURAS ELU

A 4.1 DADOS

A 4.2 PREPARAÇÃO DOS DADO DOS MATERIAIS

A 4.2.1 CONCRETO

NBR 6118:2014	fck (MPa)	35
Tabela 12.1 NBR	γc	1,4
Resistência concreto	fcd (MPa)	25
Deformações limites do concreto	ec2	0,200%
	ecu	0,350%

A 4.2.2 ARMADURA PASSIVA

NBR 6118:2014	fyk	500
Tabela 12.1 NBR	γs	1,15
Res. Cálculo	fyd	435
Módulo Elast.	Es (MPa)	210
Deformações limite	εyd	0,207%
	εsu	1,000%

A 4.3 COMBINAÇÃO NORMAL ÚLTIMA

$$F_d = \sum_{i=1}^{m} \gamma_{gi} G_i + \gamma_{q1} Q_1 + \sum_{j=2}^{n} \gamma_{qj} \psi_{oj} Q_j$$

Tab.2 NBR 8681:0	γg (desfavorável	1,35
Tab.2 NBR 8681:0	γg (favorável)	1,00
Tab.5 NBR 8681:0	γq	1,50

A 4.4 COMBINAÇÃO EXCEPCIONAL

$$F_d = \sum_{i=1}^{m} \gamma_{gi} F_{Gi,k} + F_{Q,exc} + \gamma_q \sum_{j=1}^{n} \psi_{0j,ef} F_{Qj,k}$$

Tab.2 NBR 8681:03	γg	1,1
Tab.5 NBR 8681:03	γq	1,0
Tab.5 NBR 8681:03	ψ0ef	0,7

A 4.5 DADOS DA SEÇÃO

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Mom. Carga Perm.	Mg (kNm/m)	9,47	21,01	9,47	21,01	-10,17	21,01	21,01
Mom. Carga Variável	Mq+ (kNm/m)	41,31	36,62				36,62	0,00
Mom. Carga Variável	Mq- (kNm/m)			-37,27	-20,32	-36,68		-20,32
Mom. Impacto Defensa	Mq,imp. (kNm/m)					-40,08		
Msd (valor absoluto)	Msd	**74,75**	**83,28**	**46,44**	**0,00**	**76,95**	**83,28**	**9,48**
Largura faixa laje	bw (m)	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Espessura laje	t laje (m)	0,28	0,28	0,28	0,28	0,28	0,28	0,28
Cobrimento	c (m)	0,025	0,025	0,025	0,025	0,025	0,025	0,025
Espessura pré-laje	t prel (m)	0,08	0,08	0,08	0,08	0,08	0,08	0,08
Altura armadura	d' (m)	0,032	0,112	0,032	0,032	0,032	0,032	0,032
Altura útil laje	d (m)	0,25	0,17	0,25	0,25	0,25	0,25	0,25

A 4.6 CÁLCULO DA ARMADURA

$$M_{sd} = R_{cc} \cdot z = 0{,}8x \cdot b_w \cdot 0{,}85 \cdot \sigma_{cd}(d - 0{,}4x)$$

$$R_{cc} = R_{st}$$

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Linha neutra	x (m) (supor	0,03	0,04	0,04	0,04	0,04	0,04	0,04
Verificação Dom. 2 ou 3	x/d	0,121	0,238	0,161	0,161	0,161	0,161	0,161
	Domínio	**2**	**2**	**2**	**2**	**2**	**2**	**2**
Deformação As	εys	0,010	0,010	0,010	0,010	0,010	0,010	1,010
Deform. Concreto	εc	0,0014	0,0031	0,0019	0,002	0,0019	0,0019	0,194
Tensão Concreto	σcd (kN/m²)	19182	21250	21219	######	21219	21219	21250
Braço de alavanca As	z (m)	0,24	0,15	0,23	0,23	0,23	0,23	0,23
Resultante compressão	Rccd (kN)	460,4	680,0	679,0	678,99	679,0	679,0	680,0
Momento resistente	Mrd	108,6	103,4	157,5	157,53	157,5	157,5	157,8
Verificação	Mrd >= Msd	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**

A 4.6.1 CÁLCULO DA ARMADURA

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Result. de As necess.	Rst (kN)	460,38	680,00	678,99	678,99	678,99	678,99	680,00
Tensão da As	σst (kN/m²)	434783	434783	434783	434783	434783	434783	434783
Área necessária	As (cm²/m)	10,59	15,64	15,62	15,62	15,62	15,62	15,64

A 4.6.2 ARMADURA MÍNIMA

ABNT NBR 6118:2014	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Tabela 17.3 e 19.1	ρmín	0,164	0,164	0,164	0,164	0,164	0,164	0,164
	ρs	0,110	0,110	0,110	0,110	0,164	0,110	0,110
	As,mín (cm²	3,08	3,08	3,08	3,08	4,59	3,08	3,08

A 4.6.3 ARMADURA ADOTADA

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Armadura Necessária	As (cm²)	10,59	15,64	15,62	15,62	15,62	15,62	15,64
Bitola Adotada	ϕ (mm)	12,50	12,50	12,50	12,50	12,50	12,50	12,50
Espaçamento calculado	s (mínimo)	11,50	7,50	7,50	7,50	7,50	7,50	7,50
Espaçamento adotado	s (adot)	7,50	7,50	7,50	7,50	7,50	7,50	7,50
Armadura adotada	As,efet (cm²/m)	16,36	16,36	16,36	16,36	16,36	16,36	16,36

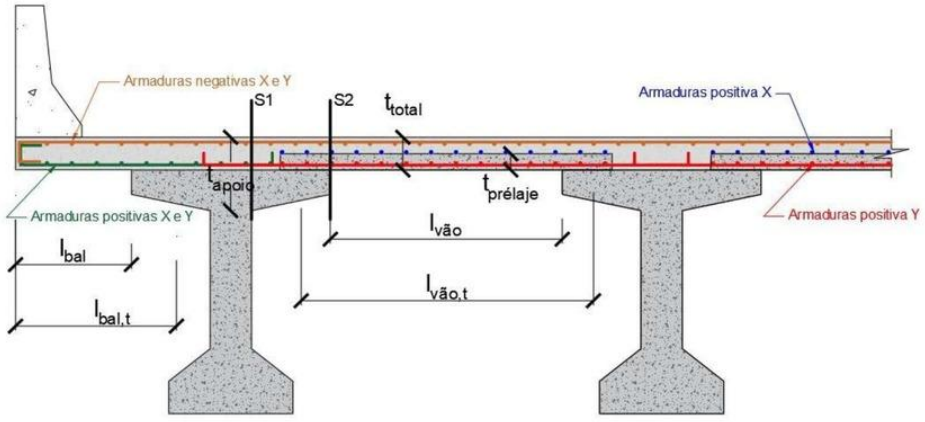


Figura A19 - Dimensões e esquema de detalhamento das lajes

A 5 VERIFICAÇÃO DA FORÇA CORTANTE

A 5.1 DADOS CONCRETO

Resis. Tração Concreto	fctk,inf (MPa)	2,25
Tabela12.1 NBR 6118	γc	1,4

A 5.2 COMBINAÇÃO NORMAL ÚLTIMA

$$F_d = \sum_{i=1}^{m} \gamma_{gi} G_i + \gamma_{q1} Q_1 + \sum_{j=2}^{n} \gamma_{qj} \psi_{oj} Q_j$$

Tab.2 NBR 8681:03	γg	1,35
Tab.5 NBR 8681:03	γq	1,5

A 5.3 DADOS SEÇÃO

	Seção	Seção 1	Seção 2
Cortante Carga Perm.	Vg (kN/m)	28,3	23,4
Cortante Carga Var.	Vq+ (kN/m)	12,8	28,9
Largura faixa Laje	bw (m)	1	1
Espessura Laje	t (laje)	0,63	0,28
Altura útil laje	d (m)	0,50	0,25
Armadura flexão	As (cm²/m)	16,36	16,36

A 5.4 VERIFICAÇÃO

Item 19.4 da ABNT NBR 6118:2014

$$V_{Rd1} = [\tau_{Rd}\, k\, (1{,}2 + 40\, \rho_1) + 0{,}15\, \sigma_{cp}]\, b_w d$$

$$V_{Sd} \leq V_{Rd1}$$

$$\tau_{Rd} = 0{,}25\, f_{ctd}$$

$$f_{ctd} = f_{ctk,inf} / \gamma_c$$

$$\rho_1 = \frac{A_{s1}}{b_w d}, \text{não maior que } |0{,}02|$$

$$\sigma_{cp} = N_{Sd} / A_c$$

$$k = |\, 1{,}6 - d\,|$$

	Seção	Seção 1	Seção 2
Cortante Solic. Cálculo	Vsd (kN/m)	**57,37**	**74,89**
Tensão Cis. Resistente	τrd (kN/m²)	401,25	401,25
Taxa armadura flexão	ρ1 (m²/m²)	0,0032	0,0066
Coeficiente item 19.4	k	1,10	1,35
Contante Resist. Cálculo	Vrd1 (kN/m)	**294,75**	**196,95**
Vrd1 ≥ Vsd	Verificação	**OK**	**OK**

A 6 VERIFICAÇÃO FADIGA NA LAJE

A 6.1 DADOS FADIGA

ABNT NBR 6118:2014
Item 23.5
Combinação Frequente

$$F_{d,ser} = \sum_{i=1}^{m} F_{gik} + \psi_1 F_{q1k} + \sum_{j=2}^{n} \psi_{2j} F_{qjk}$$

Tab.7 NBR 8681:200	ψ1	0,80
Item 23.5.3	γc	1,40
	γf	1,00
Tabela 23.2	Δfsd,fad (MPa)	190,0
Item 23.5.3	αs	10,00
Item 23.5.4.1	fcd,fad (MPa)	11,25

A 6.2 FADIGA ARMADURA

ABNT NBR 6118:2014
Item 23.5.3

$$\gamma_f \cdot \Delta\sigma_s \leq \Delta f_{ds,fad}$$

Verificação no Estádio II (Seção Fissurada)
Cavalcante (2019) Cap. 3, pág 81

$$x_{II} = \frac{-\alpha_e A_{s,ef} + \sqrt{(\alpha_e A_{s,ef})^2 + 2 b_w \alpha_e A_{s,ef} d}}{b_w}$$

$$I_{II} = \frac{b_w {x_{II}}^3}{3} + \alpha_e A_{s,ef} (d - x_{II})^2$$

$$\Delta\sigma_s = \alpha_s \cdot \psi_1 \frac{\Delta M_q \cdot (d - x_{II})}{I_{II}}$$

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Armadura flexão	As (m²/m)	1,6E-03	1,6E-03	1,6E-03	1,6E-03	1,6E-03	1,6E-03	1,6E-03
Altura útil	d (m)	0,248	0,168	0,25	0,25	0,248	0,248	0,248
Largura de referência	bw (m)	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Linha Neutra Estadio II	x_{II} (m)	0,075	0,060	0,08	0,08	0,075	0,075	0,075
Inércia Estádio II	I_{II} (m4)	6,3E-04	2,6E-04	0,00	0,00	6,3E-04	6,3E-04	6,3E-04
Momento C. Perman.	Mg (kNm/m)	9,47	21,01	9,47	21,01	-10,17	21,01	21,01
Momento Máximo	Mq+ (kNm/m)	41,31	36,62	41,31	36,62	12,38	36,62	36,62
Momento Mínimo	Mq- (kNm/m)	-37,27	-20,32	-37,27	-20,32	-36,68	-20,32	-20,32
Var. Tensão As	γf Δσs (MPa)	**172,34**	**187,91**	**172,34**	**124,87**	**107,61**	**124,87**	**124,87**
Verificação	Fadiga As	**OK**	**OK**	**OK**	**OK**	**OK**	**OK**	**OK**

A 6.3 FADIGA CONCRETO

ABNT NBR 6118:2014
Item 23.5.4

$f_{cd,fad} = 0{,}45\ f_{cd}$

$\eta_c \cdot \gamma_f \cdot \sigma_{c,máx} \leq f_{cd,fad}$

$$\eta_c = \frac{1}{1{,}5 - 0{,}5(|\sigma_{c1}|/|\sigma_{c2}|)}$$

Cálculo das tensões no concreto, no Estádio I

$$\sigma_{c,máx} = \frac{(M_g + \psi_1 \cdot M_{q,máx}) \cdot x_{II}}{I_{II}}$$

σc1 - Tensão de compressão na LN (menor valor a menos de 30cm da borda sup
σc1 - Tensão de compressão na borda superior da laje (maior valor a menos de 30cm da borda sup.)

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Tensões no concreto p/ Combinação Frequente	σc1(MPa)	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	σc2(MPa)	-5,07	-11,40	-5,07	-6,00	0,03	-6,00	-6,00
	σc,máx(MPa)	-5,07	-11,40	-5,07	-6,00	0,00	-6,00	-6,00
Coef. Ajuste.	ηc	0,67	0,67	0,67	0,67	0,67	0,67	0,67
Verificação	ηc γf |σc,máx|	3,38	7,60	3,38	4,00	0,00	4,00	4,00
	Fadiga σc	**OK**	**OK**	**OK**	**OK**	**OK**	**OK**	**OK**

A 7 VERIFICAÇÃO ABERTURA DA FISSURAS

A 7.1 DADOS

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
Linha Neutra Estadio II	x_{II} (m)	0,075	0,060	0,075	0,075	0,075	0,075	0,075
Inércia Estádio II	I_{II} (m4)	6,3E-04	2,6E-04	6,3E-04	6,3E-04	6,3E-04	6,3E-04	6,3E-04
Momento C. Perman.	Mg (kNm/m)	9,47	21,01	9,47	21,01	-10,17	21,01	21,01
Momento C. Móvel	Mq (kNm/m)	41,31	36,62	-37,27	-20,32	-36,68	36,62	-20,32
Bitola armadura	ϕ (mm)	12,50	12,50	12,50	12,50	12,50	12,50	12,50
Armadura flexão	As,efet (cm²/m)	16,36	16,36	16,36	16,36	16,36	16,36	16,36
Item 9.3.2 NBR 6118	$\eta 1$	2,25	2,25	2,25	2,25	2,25	2,25	2,25
Cobrimento	c (m)	0,025	0,025	0,025	0,025	0,025	0,025	0,025
Altura útil	d (m)	0,248	0,168	0,248	0,248	0,248	0,248	0,248
Lagura faixa laje	bw (m)	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Resis. tração concreto	fctm (MPa)	3,21	3,21	3,21	3,21	3,21	3,21	3,21
Módulo Armadura	Es (GPa)	210	210	210	210	210	210	210

Combinação Frequente

$$F_{d,uti} = \sum_{i=1}^{m} F_{Gi,k} + \psi_1 F_{Q1,k} + \sum_{j=2}^{n} \psi_{2j} F_{Qj,k}$$

onde

Tab.6 NBR 8681:2003	$\psi 1$	0,5

A 7.2 CÁLCULO DA ABERTURA DE FISSURAS

ABNT NBR 6118:2014
Item 17.3.3.2

$$w_k = \frac{\phi_i}{12,5\eta_1} \frac{\sigma_{si}}{E_{si}} \frac{3\sigma_{si}}{f_{ctm}}$$

$$w_k = \frac{\phi_i}{12,5\eta_1} \frac{\sigma_{si}}{E_{si}} \left(\frac{4}{\rho_{ri}} + 45 \right)$$

$$\sigma_{si} = \alpha_s \cdot \frac{\left(M_g + \psi_1 M_q\right) \cdot (d - x_{II})}{I_{II}}$$

	Seção	Vão y (+)	Vão x (+)	Vão y (-)	Vão x (-)	Eng.bal y (-)	Borda x (+)	Borda x (-)
It. 17.3.3.2: Es/Ec	αs	15,00	15,00	15,00	15,00	16,00	17,00	18,00
Tensão Amadura	σsi (kN/m²)	123875	243272	37694	44591	125075	183214	53509
Área a 7,5ϕ de As	Acri (m2)	0,125	0,125	0,125	0,125	0,125	0,125	0,125
Taxa de As em Acri	ρri (m2/m2)	0,0131	0,0131	0,0131	0,0131	0,0131	0,0131	0,0131
Fissur. não sistemática	wk1 (mm)	0,03	0,12	0,00	0,00	0,03	0,07	0,01
Fissuração sistemática	wk2 (mm)	0,09	0,18	0,03	0,03	0,09	0,14	0,04
Tab.13.4 NBR 6118/14	wk,lim (mm)	0,30	0,30	0,30	0,30	0,30	0,30	0,30
Menor entre wk1 e wk2	Verificação	OK	OK	OK	OK	OK	OK	OK

A 8 DIMENSIONAMENTO ARMADURAS ELU - PRÉ-LAJE

A 8.1 DADOS

A 8.2 PREPARAÇÃO DOS DADO DOS MATERIAIS

A 8.2.1 CONCRETO

NBR 6118:2014	fck (MPa)	35,0
Tabela 12.1 NBR	γc	1,40
Resistência concreto	fcd (MPa)	25,0
Deformações limites do concreto	ec2	0,20%
	ecu	0,35%

A 8.2.2 ARMADURA PASSIVA

NBR 6118:2014	fyk	500,0
Tabela 12.1 NBR	γs	1,15
Res. Cálculo	fyd	434,78
Módulo Elast.	Es (MPa)	210,00
Deformações limites	εyd	0,207%
	εsu	1,000%

A 8.3 COMBINAÇÃO NORMAL ÚLTIMA - CONSTRUÇÃO

$$F_d = \sum_{i=1}^{m} \gamma_{gi} F_{Gi,k} + \gamma_q \left[F_{Q1,k} + \sum_{j=2}^{n} \psi_{0j,ef} F_{Qj,k} \right]$$

Tab.2 NBR 8681:03	γg	1,25
Tab.5 NBR 8681:03	γq	1,30

A 8.4 DADOS DA SEÇÃO

Momento Peso próprio	Mpp (kNm)	1,30
Momento Concr. Fresco	Mcf (kNm)	3,25
Momento q utilização	Mutil (kNm)	1,30
Momento de Cálculo	Msd (kNm)	**7,54**
Largura pré-laje	bw,prel (m)	1,00
Espessura pré-laje	t prel (m)	0,08
Cobrimento	c (m)	0,025
Altura armadura	d' (m)	0,031
Altura útil laje	d (m)	0,049

A 8.5 VERIFICAÇÃO DA RESULTANTE DE COMPRESSÃO

$$M_{sd} = R_{cc} \cdot z = 0{,}8x \cdot b_w \cdot 0{,}85 \cdot \sigma_{cd}(d - 0{,}4x)$$

$$R_{cc} = R_{st}$$

Domínio 3	x/d limite	0,45
Linha Neutra máxima Domínio 2 ou 3	x (m)	0,022
Deformação As	εys	0,010
4.6 Cálculo da armadura	εc	0,0082
Tensão Concreto	σcd (kN/m²)	21250
Braço de alavanca As	z (m)	0,040
Resultante compressão	Rccd (kN)	372,94
Momento resistente	Mrd	**14,91**
Verificação	Mrd >= Msd	**Ok**

A 8.6 VERIFICAÇÃO DA ARMADURA

Result. de As necess.	Rst (kN)	372,94
Tensão da As	σst (kN/m²)	434783
Área necessária	As (cm²)	8,58
As existente	As,exist (cm²)	16,36
As ≥ As existente	Verificação	**OK**

A 9 VERIFICAÇÃO DA FORÇA CORTANTE - PRÉ-LAJE

A 9.1 DADOS CONCRETO

Resis. Tração Concreto	fctk,inf (MPa)	2,25
Tabela12.1 NBR 6118	γc	1,40

A 9.2 COMBINAÇÃO NORMAL ÚLTIMA - CONSTRUÇÃO

$$F_d = \sum_{i=1}^{m} \gamma_{gi} F_{Gi,k} + \gamma_q \left[F_{Q1,k} + \sum_{j=2}^{n} \psi_{0j,ef} F_{Qj,k} \right]$$

Tab.2 NBR 8681:03	γg	1,25
Tab.5 NBR 8681:03	γq	1,30

A 9.3 DADOS SEÇÃO

Cortante Peso próprio	Vpp (kNm)	2,28
Cortante Concr. Fresco	Vcf (kNm)	5,70
Cortante q utilização	Vutil (kNm)	2,28
Largura Pré-Laje	bw,prel (m)	1,00
Espessura Pré-Laje	t prel	0,08
Altura útil Pré-Laje	d (m)	0,05
Armadura flexão	As (cm²)	16,36

A 9.4 VERIFICAÇÃO

Item 19.4 da ABNT NBR 6118:2014

$V_{Rd1} = [\tau_{Rd}\, k\, (1{,}2 + 40\, \rho_1) + 0{,}15\, \sigma_{cp}]\, b_w d$

$V_{Sd} \leq V_{Rd1}$

$\tau_{Rd} = 0{,}25\, f_{ctd}$

$f_{ctd} = f_{ctk,inf} / \gamma_c$

$\rho_1 = \frac{A_{s1}}{b_w d}$, não maior que $|0{,}02|$

$\sigma_{cp} = N_{Sd} / A_c$

$k = |\, 1{,}6 - d\,|$

Cortante Solic. Cálculo	Vsd (kN)	**13,22**
Tensão Cis. Resistente	τrd (kN/m²)	401,25
Taxa armadura flexão	ρ1 (m²/m²)	0,0336
Coeficiente item 19.4	k	1,55
Contante Resist. Cálculo	Vrd1 (kN)	**77,15**
Vrd1 ≥ Vsd	Verificação	**OK**

A 10 VERIFICAÇÃO ABERTURA DA FISSURAS PRÉ-LAJE

A 10.1 DADOS

Momento Peso próprio	Mpp (kNm)	1,30
Momento Concr. Fresco	Mcf (kNm)	3,25
Momento q utilização	Mutil (kNm)	1,30
Espessura pré-laje	t prel (m)	0,08
Altura útil	d (m)	0,049
Lagura faixa laje	bw (m)	1,00
Item 9.3.2 NBR 6118	$\eta 1$	2,25
Cobrimento	c (m)	0,025
Armadura flexão	As	16,36
Bitola armadura	ϕ (mm)	12,50
Resis. tração concreto	fctm (MPa)	3,21
Módulo Armadura	Es (GPa)	210,0

A 10.1.1 COMBINAÇÃO FREQUENTE

$$F_{d,uti} = \sum_{i=1}^{m} F_{Gi,k} + \psi_1 F_{Q1,k} + \sum_{j=2}^{n} \psi_{2j} F_{Qj,k}$$

Tab.6 NBR 8681:2003	$\psi 1$	0,50

A 10.2 CÁLCULO DA ABERTURA DE FISSURAS

Item 17.3.3.2 da ABNT NBR 6118:2014

$$w_k = \frac{\phi_i}{12,5\eta_1} \frac{\sigma_{si}}{E_{si}} \frac{3\sigma_{si}}{f_{ctm}}$$

$$w_k = \frac{\phi_i}{12,5\eta_1} \frac{\sigma_{si}}{E_{si}} \left(\frac{4}{\rho_{ri}} + 45 \right)$$

$$\sigma_{si} = \alpha_s \cdot \frac{(M_g + \psi_1 M_q) \cdot (d - x_{II})}{I_{II}}$$

Verificação no Estádio II (Seção Fissurada)
Cavalcante (2019) Cap. 3, pág 81

$$x_{II} = \frac{-\alpha_e A_{s,ef} + \sqrt{(\alpha_e A_{s,ef})^2 + 2 b_w \alpha_e A_{s,ef} d}}{b_w}$$

$$I_{II} = \frac{b_w {x_{II}}^3}{3} + \alpha_e A_{s,ef} (d - x_{II})^2$$

Item 17.3.3.2: Es/Ec	αs	15
Linha Neutra Estadio II	x_{II} (m)	0,030
Inércia Estádio II	I_{II} (m4)	1,8E-05
Tensão Amadura	σsi (kN/m²)	82120
Área a 7,5ϕ de As	Acri (m²)	0,050
Taxa de As em Acri	ρri (m²/m²)	0,033
Fissur. não sistemática	wk1 (mm)	0,013
Fissuração sistemática	wk2 (mm)	0,029
Tab.13.4 NBR 6118/14	wk,lim (mm)	0,300
Menor entre wk1 e wk2	Verificação	**OK**

ANEXO C

Planilha de dimensionamento de longarina metálica

O anexo C contém a planilha de dimensionamento referente às longarinas das pontes mistas de aço e concreto (MAC), desenvolvida dentro do grupo de pesquisa conforme produto do Relatório Técnico M3.A2 (DNIT, 2022). Os dados preenchidos são da tipologia MAC-A-20-P, composta por 5 longarinas, espaçamento entre eixos de longarinas de 290 cm e balanço de 102,5 cm. A altura da longarina adotada é de 130 cm, e a espessura da laje do tabuleiro é de 28 cm. As dimensões da seção transversal constam na Figura C.1, enquanto as dimensões estipuladas da longarina estão na Figura C.2.

Figura C.1 - Dimensões em cm da seção transversal do tabuleiro da tipologia MAC-A-20-P

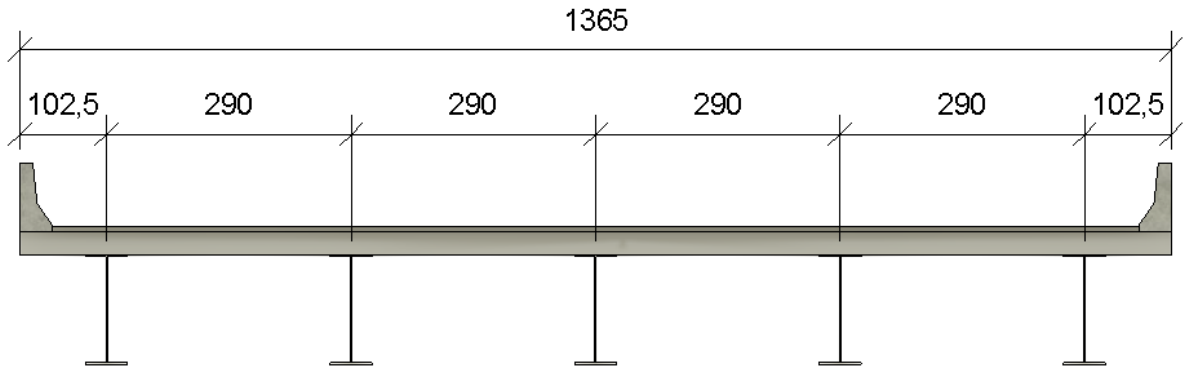
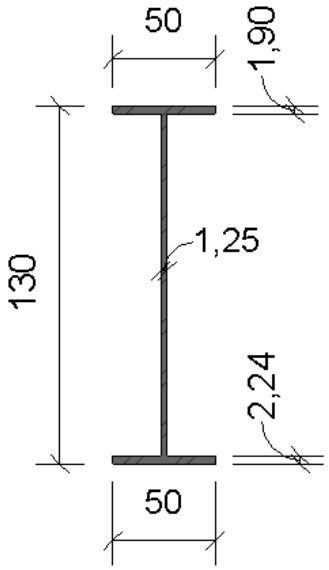


Figura C.2 - Dimensões em cm da seção transversal da longarina da tipologia MAC-A-20-P



Planilha de dimensionamento de longarina metálica

1. CONSIDERAÇÕES INICIAIS

Nesse documento é apresentado o dimensionamento de uma ponte mista de aço e concreto de eixo reto sobre múltiplas longarinas. O tabuleiro é composto por pré-lajes e uma capa de concreto moldada in loco.

Na planilha, as células em cinza correspondem aos parâmetros de entrada fornecidos pelo usuário e, as células em branco, os resultados dos cálculos efetuados.

1.1. SEÇÃO TRANSVERSAL

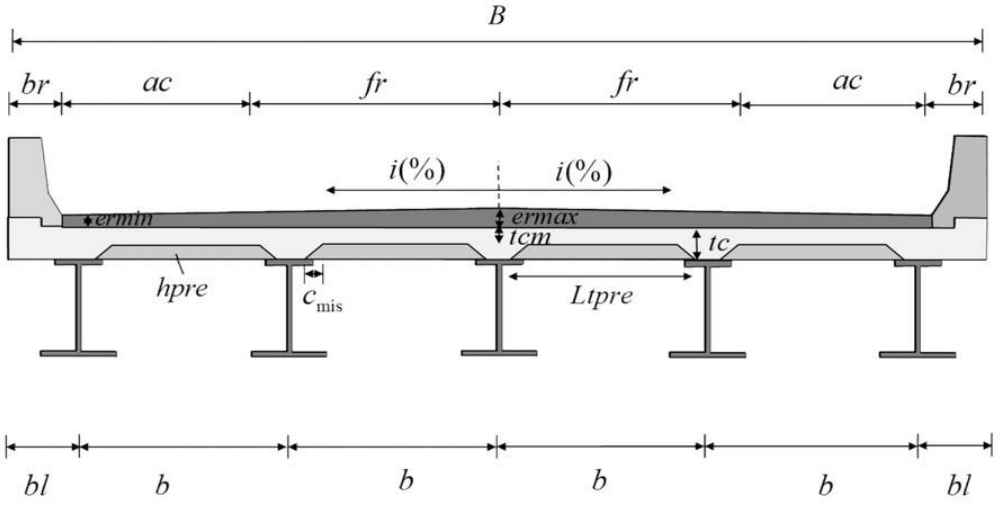


Figura A.1.1. Seção transversal da ponte.

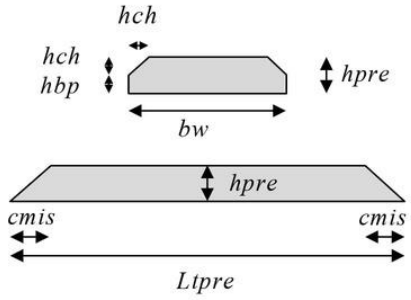


Figura A.1.2. Geometria da pré-laje.

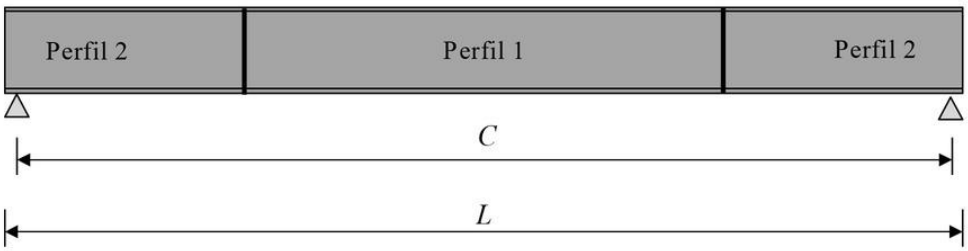


Figura A.1.3. Vista Longitudinal da longarina.

Geometria tabuleiro	Classe ponte	Classe IA com passeio	
	Comprimento do vão	L (m):	20
	Número de faixas	nf:	2
	Largura da faixa de rolamento	fr (m):	3,6
	Largura do acostamento externo	ac (m):	2,5
	Largura do acostamento interno	aci (m):	1,5
	Quantidade de passeios	qp:	1
	Largura do passeio	Lp (m):	1,5
	Número de longarinas	NL:	5
	Guarda corpo (m)	Gp (m):	0,15
	Vão entre longarinas (1.1 < b < 4.9 m ou 2.5 < b < 3 m item 5.5.1 [1])	b (m):	2,9
	Largura da ponte	B (m):	13,65
	Comprimento do balanço	bl (m):	1,03
	Relação bl/b	bl/b	0,35
	Espessura total da laje>(18;(S+3)/30;15) cm AASHTO 9.7.1.1/ NBR7187 9.1.1	tc (m):	0,280
	Camada de concreto adicionada in-loco (m):	tcm (m):	0,180
Pavimento	Inclinação do pavimento	i (%):	2,0
	Espessura mínima do revestimento asfáltico	ermin (m)	0,150
	Espessura máxima do revestimento asfáltico	ermax (m	0,272
	Espessura média do revestimento asfáltico	er (m):	0,211
Pré-laje	Altura da pré laje de concreto	hpre(m):	0,100
	Espessura mísula da prélaje de concreto	cmís (m):	0,35
	Altura chamfro pré-laje	hch (m):	0,025
	Altura base pré-laje	hbp (m):	0,05
	Largura da pré-laje de concreto	bw (m):	1,00
Barreiras	Área das barreiras do tipo New Jersey	Ab (m²):	0,225
	Largura das barreiras do tipo New Jersey	lbr (m):	0,4
	Altura das barreiras do tipo New Jersey	hbr (m):	0,87

1.2. ASPECTOS DO PROJETO

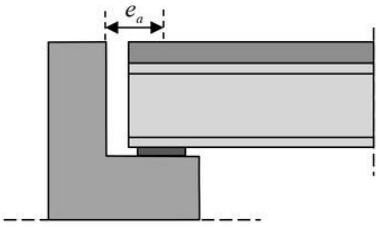


Figura A.1.4. Extremidade da ponte.

Distância entre o início da ponte e o centro do apoio	ea (m):	0,400
Comprimento junta	ej (m):	0,05
Vão efetivo C' = C -2ea (NBR 16694 - 7.1)	C (m):	19,2
[1] Comprimento padrão das chapas (12 - 0.1 = 11.9 m)	$C_{máx}$ (m):	11,9

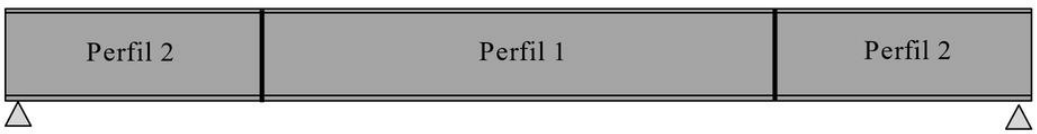


Figura A.1.5. Disposição dos perfis na longarina.

- Distribuição perfis longarina:

Número de trechos	NT:	3
Qt de perfil na longarina	Qt P1:	1
Qt de perfil na longarina	Qt P2:	2
Comprimento do perfil 1	Lp1 (m):	11,90
Comprimento do perfil 2	Lp2 (m):	4,05

Quantidade de emendas	Ne:	1
Posição da emenda 1	S1 (m):	4,05
Trecho contínuo	-	-
Posição da emenda 2	S2 (m):	16,0
Comp das talas na região de emendas (S1)	C_{t1} (mm):	730
Comp das talas na região de emendas (S2)	C_{t2} (mm):	730

- Posição dos diafragmas:

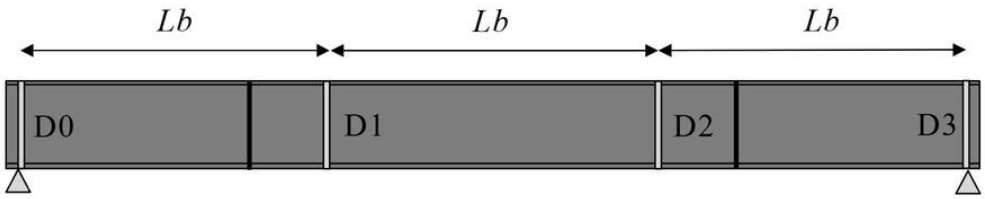


Figura A.1.6. Posição dos diafragmas e enrijecedores.

Quantidade de diafragmas	Qt:	4
Distância entre diafragmas	Lb (mm):	6400
Posição diafragma D0	D0 (mm):	400
Posição diafragma D1	D1 (mm):	6800
Posição diafragma D2	D2 (mm):	13200
Posição diafragma D3	D3 (mm):	19600
Posição diafragma D4	D4 (mm):	-
Posição diafragma D5	D5 (mm):	-
Posição diafragma D6	D6 (mm):	-
Posição diafragma D7	D7 (mm):	-
Posição diafragma D8	D8 (mm):	-

Espaçamento escoramento balanço	Se (m):	1000

Seção diafragma apoio:	PS 975 x 398
Seção diafragmas intermediário:	PS 800 x 335

- Enrijecedores transversais de apoio:

Quantidade de enrijecedores de apoio	ne:	2
Distância entre enrijecedores de apoio	de (mm):	180
Espessura dos enrijecedores de apoio	te (mm):	16
Largura dos enrijecedores de apoio	be (mm):	190

- Enrijecedores transversais intermediários:

Lb/5	Lb/4	Lb/3	Lb/2
1280	1600	2133,33	3200

Distância entre enrijecedores D0-D1	a1 (mm):	-
Distância entre enrijecedores D1-D2	a2 (mm):	-
Distância entre enrijecedores D2-D3	a3 (mm):	-
Espessura dos enrijecedores transversais	eet (mm):	12,5
Largura dos enrijecedores transversais	let (mm):	-

- Classe de agressividade ambiental:

Classificação do ambiente (ISO 9223)	Amb:	C1
(S) - perfis pintados / (N) perfis não-pintado	Proteção:	s
Sobre-espessura para cada face exposta NBR	∆esp (mm):	0

Classe de agressividade ambiental	CAA:	II
Cobrimento face inferior das lajes	cl (mm):	30
Cobrimento face superior das lajes	cl2 (mm):	30
Cobrimento pré-laje superior e inferior	cp (mm):	30

- Coeficiente de curta ou longa duração:

Coeficiente carga de curta duração	$\eta 0$ =	6,8	$\eta 0 = \frac{E}{Ecs}$
Coeficiente carga de longa duração NBR 8800:2008 (item O.1.2.1)	ηf=	20,4	$\eta f = 3 \cdot \eta 0$

1.3. MATERIAIS

- Capa de concreto moldada in loco:

$$E_{cs} = 0.85 E_{ci}$$

$$E_{ci} = 5600\sqrt{f_{ck}}$$

Resistência à compressão característica do concreto fck ≥ 30MPa (NBR16694 - 4.5.1)	fck (MPa):	35
Resistência média à compressão do concreto aos 28 dias	fcm (MPa):	35
Módulo de elasticidade (NBR8800 - 4.5.3.1)	Eci (MPa):	33130
Módulo de elastasticidade secante (NBR8800 - 4.5.3.1)	Ecs (MPa):	29403
Peso específico do concreto armado (NBR16694 - 6.1.1)	γc(kN/m³):	25
Coeficiente ponderação da resistência	γc:	1,4
Coeficiente de dilatação do concreto	αc:	9,1E-06

- Pré-laje de concreto:

Resistência à compressão característica da pré-laje fck > 30MPa (NBR16694 - 4.5.1)	fck (MPa):	35
Módulo de elasticidade (NBR8800 - 4.5.3.1)	Eci (MPa):	33130
Módulo de elastasticidade secante (NBR8800 - 4.5.3.1)	Ecs (MPa):	29403
Módulo de elasticidade das armaduras	Ea (MPa):	210000

- Aço estrutural: longarinas

Aço resistente à corrosão	Aço:	A588 G50
Resistência ao escoamento	fy (MPa):	345
Resistência à ruptura	fu (MPa):	480
Módulo de elasticidade	E (MPa):	200000
Poisson	υ:	0,3
Módulo de elasticidade transversal	G (MPa):	76923
Peso específico do aço	γa (kN/m³):	78,6
Coeficiente ponderação do aço	γa1:	1,15
Coeficiente ponderação do aço	γa2:	1,35

- Aço estrutural: diafragmas

Aço resistente à corrosão	Aço:	A588 G50
Densidade	γa (kN/m³):	78,6
Tensão de escoamento	fy (MPa):	345
Tensão de ruptura	fu (Mpa):	480
Módulo de elasticidade	E (MPa):	200000

- Aço estrutural: chapas emendas

Aço resistente à corrosão		A588 G50
Resistência ao escoamento do aço	fy (MPa):	345
Resistência última do aço	fu (MPa):	480

- Conectores de cisalhamento:

NBR16694 item 4.3	Tipo:	Stud Bolt
	Aço:	A108-G1020
Tensão de escoamento	fysb (MPa):	345
Tensão de ruptura	fusb (MPa):	415
Altura do conector	hsb (mm):	100
Diâmetro do conector	dsb (mm):	22
Área do conector	Asb (mm2):	380
Alongamento mínimo	A (mm):	50
efeito grupo (NBR8800 O4.2.1.2)	Rg:	1
Posição dos conectores	Rp:	1

- Enrijecedores:

Aço resistente à corrosão	Aço:	A588
Tensão de escoamento	fysb (MPa):	345
Tensão de ruptura	fusb (MPa):	485
Módulo de elasticidade	Ee (MPa):	200000
Espessura dos enrijecedores	tet (mm):	12,5
Largura dos enrijecedores	bet-ad (mm):	-

- Solda:

Tipo/Classe	Tipo:	ASW E70
Resistência à ruptura do material	f_w (MPa):	485
Coeficiente de ponderação da solda	γ_w:	1,35

- Armadura passiva:

Tipo:		CA-50
Tensão de escoamento	fysb (MPa):	500
Tensão de ruptura	fusb (MPa):	650
Módulo de elasticidade	Ep (MPa):	210000
Índice de conformação da armadura	η_i:	2,25
Coeficiente de ponderação	γ_s:	1,15

- Revestimento asfáltico:

Peso específico (NBR16694 - 6.1.2)	γ_r (kN/m³):	24

- Parafuso estrutural:

Material		ASTM A325
Resistência ao escoamento do aço	fy (MPa):	635
Resistência última do aço	fu (MPa):	825
Diâmetro parafuso mesa superior	dps (mm):	19
Diâmetro parafuso mesa inferior	dpi (mm):	19
Diâmetro parafuso alma	dpa (mm):	22,225

- Aparelho de apoio:

Largura	a (mm):	300
Comprimento	b(mm):	300
Espessura total	h (mm):	73
Espessura da camada de elastômetro	ece (mm):	12
Quantidade de camadas de elastômetro	Qt camadas:	4
Espessura das chapas de aço	eca (mm):	4
Quantidade de chapas de aço	Qt chapas:	5
Esp. total camadas de elastômetro	hap (mm):	48
Módulo de elasticidade transversal	G (MPa):	1
Rigidez k do aparelho de apoio paracargas dinâmicas	k (kN/m):	1233

1.3. LITERATURA

ABNT

- NBR5884:2013 - Perfil I estrutural de aço soldado por arco elétrico - Requisitos gerais
- NBR6118:2014 - Projeto de estruturas de concreto - Procedimento
- NBR6120:2019 - Cargas para cálculo de estruturas de edificações
- NBR6123:1988: - Forças devido ao vento em edificações
- NBR7188:2013 Carga móvel rodoviária e de pedestres em pontes, viadutos, passarelas
- NBR8681:2003 - Ações e segurança em estruturas - Procedimento
- NBR 8800:2008 - Projeto de estruturas de aço e estruturas mistas de aço e concreto de edifícios
- NBR15980:2011 - Perfis laminados de aço para uso estrutural - Dimensões e tolerâncias
- NBR 16694:2020 - Projeto de pontes rodoviárias de aço e mistas de aço e concreto

NORMAS INTERNACIONAIS

- AASHTO (2014) - Bridges design specifications. 7º ed, 2014.
- FHWA-NCI-15-047 (2015) - Load and Resistance Factor Design (LRFD) for Highway Brigdes Superestructures. Reference Manual.
- AISC (2013) - Steel Manual 13th Edition. 2013.
- AWS D1.5(2015) - Brigdes welding code. 2015.
- EUROCODE 4-2. Design of composite steel and concrete structures: General rules and rules for bridges. 2004.
- ASTM A193 B7 - Specification for high tensile bolts, screws, stud bolts, threads rods, anchor bolts and other externally threaded fasteners. 2020.

DIVERSOS

- Catai, Evandro. Análise dos efeitos de retração e fluência em vigas mistas. Dissertação de mestrado. Escola de Engenharia de São Carlos. Universidade de São paulo. 2005, 166p.
- FWHA-NHI-15-044 - Engineering for Structural Stability in Bridge Construction (2015)
- Ionnayis, V; Aristidis, I (2014). Design of Steel-Concrete composite bridges to Eurocode. CC Press. 1ed, 2014, 473p.
- Lebet et al (2015). Steel Bridges: Conceptual and structural design of steel and steel-concrete composite bridges. 1ed, 2013,576p.
- CENTRO BRASILEIRO DA CONSTRUÇÃO EM AÇO (CBCA). “Pontes e viadutos em vigas mistas” – 2ª edição, 2021
- Moga et al. Concrete shrinkage stresses and thermal effects in the composite girders according to Eurocodes. Green Design and Sustainable Archicture. 2016.
- Álbum de pontes semi-permanentes. DNIT. 2023.

A 2. AÇÕES

As ações consideradas estão descritas na NBR 16694 (2022) item 6.

A 2.1. AÇÕES PERMANENTES

A 2.1.1 AÇÃO VERTICAL: PESO DA ESTRUTURA METÁLICA

O carregamento atuante sobre as longarinas na fase de içamento foi estimado a partir do peso específico do aço e da seção tranversal da longarina.

O carregamento teórico nas fases construtiva e de utilização foi obtido a partir da seção transversal das longarinas, admitindo o acréscimo de 10%, a fim de considerar o peso próprio das transverinas e dos demais elementos de aço. A carga adicional devido aos escoramentos (Pesc) usados nos balanços foi admitida com valor de 0,2 kN/m² (distribuída sobre o tabuleiro).

- Etapa 0:

$$G_{1C} = \frac{1,05 \cdot (n_1 \cdot A_1 \cdot L_1 + n_2 \cdot A_2 \cdot L_2) \cdot \gamma_a}{L}$$

Carregamento na fase de içamento	G1C (kN/m):	3,0

- Etapas 1 e 2: construtiva e utilização

$$G_1 = \frac{1,1 \cdot (n_1 \cdot A_1 \cdot L_1 + n_2 \cdot A_2 \cdot L_2) \cdot \gamma_a}{B} + \frac{0,2 \cdot B}{NL}$$

Carregamento devido aos escoramentos	Pesc (kN/m²):	0,2
Peso próprio (elementos + escoramentos)	G1 (kN/m):	3,7
Peso por m² calculado de estrutura metálica	g1 (kN/m²):	1,4
Peso por m² de estrutura metálica (manual CBCA, 2021)	Pest (kN/m²):	1,65
→ peso calculado < peso estimado		

A 2.1.2 AÇÃO VERTICAL: PESO DA LAJE DE CONCRETO

O peso próprio da laje de concreto foi obtido considerando as espessuras da capa moldada in loco e a das lajes premoldadas e o peso específico do concreto.

Peso por m² da laje de concreto	g2 (kN/m²):	7,00	$g_2 = \gamma_c \cdot t$
Peso por m da laje de concreto	G2 (kN/m):	19,1	$G_2 = \frac{(\gamma_c \cdot t_c \cdot B)}{NL}$

A 2.1.3 AÇÃO VERTICAL: PESO DAS DEFENSAS

O peso próprio das defensas do tipo New Jersey por m² foi obtido a partir do peso específico do concreto, da área da seção transversal e da largura das barreiras.

PP das barreiras do tipo New Jersey	g3 (kN/m²):	0,4	$g_3 = \frac{\gamma_c \cdot A_b}{l_{br}}$
PP das barreiras do tipo New Jersey por longarina	G3 (kN/m):	2,3	$G_3 = \frac{2 \cdot \gamma_c \cdot A_b}{NL}$
PP das barreiras do tipo New Jersey por m	G3 (kN/m):	5,6	$G_3 = \gamma_c \cdot A_b$

A 2.1.4 AÇÃO VERTICAL: PESO DO REVESTIMENTO ASFÁLTICO

O peso próprio do revestimento por m² foi obtido a partir do peso específico (γr) e da espessura (er) do revestimento asfáltico.

PP da pavimentação	g4 (kN/m²):	5,1	$g_4 = \gamma_r \cdot e_r$
PP da pavimentação	G4 (kN/m):	13,8	$G_4 = \frac{\gamma_r \cdot e_r \cdot B}{NL}$

A 2.1.5 AÇÃO VERTICAL: PESO DO RECAPEAMENTO

A NBR16694 (2020) item 6.1.2 recomenda a consideração de uma carga de 2,0 kN/m² a fim de considerar um possível recapeamento.

Carga devido recapeamento (NBR 16694 - 6.1.2)	g5 (kN/m²):	2,00
Carga devido recapeamento (NBR 16694 - 6.1.2)	G5 (kN/m):	5,14

A 2.1.6 AÇÃO HORIZONTAL: RETRAÇÃO E FLUÊNCIA

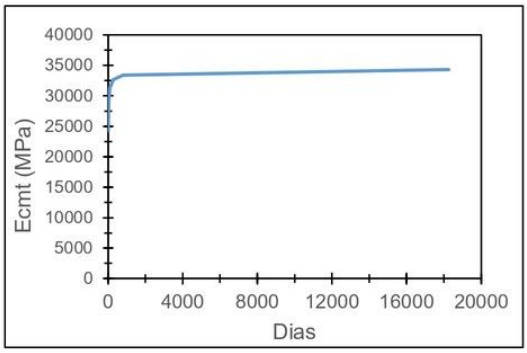
O modelo GL2000 foi utilizado para obtenção da deformação total devido à retração e do coeficiente de fluência da capa de concreto.

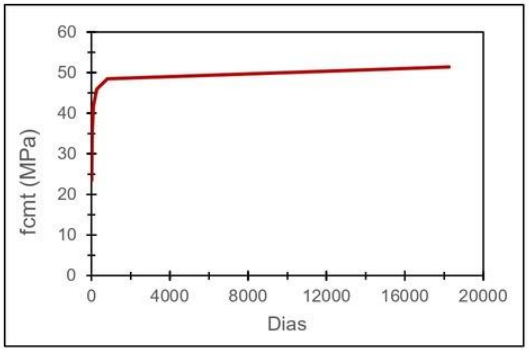
Deformação de retração e coeficiente de fluência:

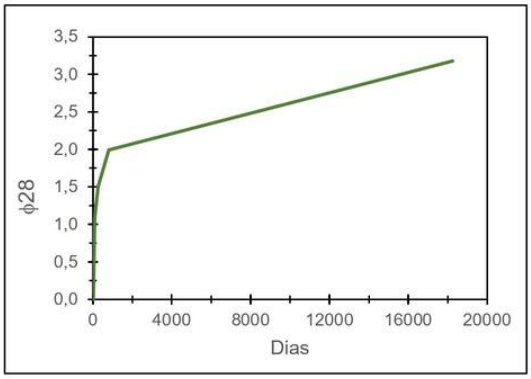
Idade de aplicação do carregamento	t0 (dias):	30
Tempo (anos):	t	50
Umidade relativa (%)	w	50
Dias de cura úmida	Início da retração:	7
Relação volume /superfície	V/S:	135
Indicar o tipo de cimento (I - CPI ou CPII)/ (II - CPIII ou CPIV)/(III - CPV)	Tipo de cimento:	II
Coeficiente de ajuste em função de cura	k :	0,75
Coeficiente de ajuste em função do tipo de cimento	s :	0,40
Coeficiente de ajuste da resistência do concreto	βe(h):	1,007
Resistência à tração do concreto	fcmt0 (MPa):	35,5
Módulo de elasticidade do concreto	Ecmt0 (MPa)	29403
Coeficiente de fluência	Φ (tr)	0,95
Coeficiente de ajuste em função da umidade relativa (%)	β(h):	0,93
Deformação devido à deformação lenta	εshu (mm/mm):	0,0006

No modelo numérico, não foi admitido o ganho de resistência do concreto com o tempo.

t (dias)	βe(h)	fcmt (MPa)	Ecmt (MPa)	β(t-tr)	εsh (mm/mm)	ϕ28 (t,t0)	J(t,t0)
7	0,8	23,5	24328	0,00	0,00000	-	-
30	1,0	35,5	29112	0,10	0,00006	0,00	0,00003
90	1,1	41,8	31292	0,19	0,00011	1,09	0,00007
270	1,1	45,9	32633	0,33	0,00019	1,50	0,00009
810	1,2	48,5	33437	0,52	0,00030	1,99	0,00010
18250	1,2	51,4	34329	0,94	0,00055	3,18	0,00014







• em 50 anos:

εsh (mm/mm):	5,5E-04
ϕ28 (t,t0):	3,18

$$f_{cmt} = \beta_e^2 \cdot f_{cm28}$$

$$E_{cmt} = 3500 + 4300\sqrt{f_{cmt}}$$

$$\beta_e^2 = \exp\left[\frac{s}{2}\cdot\left(1-\sqrt{\frac{28}{t}}\right)\right]$$

$$\varepsilon_{sh} = \varepsilon_{shu}\cdot\beta(h)\cdot\beta(t-t_r)$$

$$\varepsilon_{shu} = 900k\cdot\left(\frac{30}{f_{cm28}}\right)^{0,5}\cdot 10^{-6}$$

$$\beta(h) = (1-1{,}18\cdot h^4)$$

$$\beta(t-t_r)=\left[\frac{(t-t_r)}{(t-t_r)+0.12\cdot(V/S)^2}\right]^{0,5}$$

$$J(t,t_0)=\frac{1}{E_{cmt0}}+\frac{\varphi_{28}\cdot(t,t_0)}{E_{cm28}}$$

$$\varphi_{28}(t,t_0)=\Phi(t_r)\cdot\left[2\cdot\frac{(t-t_0)^{0,3}}{(t-t_0)^{0,3}+14}+\left(\frac{7}{t_0}\right)^{0,5}\cdot\left(\frac{(t-t_0)}{(t-t_0)+7}\right)^{0,5}+2,5\cdot\left(1-1,086h^{0,2}\right)\cdot\left(\frac{(t-t_0)}{(t-t_0)+0,12\cdot(V/S)^2}\right)^{0,5}\right]$$

$$\Phi(t_r)=\begin{cases}1 & t_0=t_r\\ \left[1-\left(\frac{(t_0-t_r)}{(t_0-t_r)+0,12\cdot(V/S)^2}\right)^{0,5}\right]^{0,5} & t_0>t_r\end{cases}$$

O modelo utilizado para a consideração do efeito da retração e fluência nos esforços solicitantes foi o proposto por Monga et al (2016) e será apresentado no item de verificação da seção mista. Outros modelos para consideração da retração, deformação lenta e variação de temperatura nas vigas mistas podem ser encontrados em Catai (2005). Para aplicação do modelo apresentado em Moga et al (2016), é necessário definir o coeficiente de fluência, a deformação total de retração e o coeficiente de retração, os quais foram obtidos segundo as recomendações apresentadas nos itens 5.4.2.2 do EN4-2, 8.2.11 da ABNT NBR 6118 (2014).

$$\eta r=\eta 0\cdot(1+0,55\cdot\varphi r)$$

Coeficiente de fluência (NBR 6118 8.2.11)	φ(t,to):	2,00
Coeficiente de fluência calculado GL2000	φ(t,to):	3,18
Coeficiente de fluência adotado	φ(t,to):	3,18
Deformação total de retração (NBR 6118 Anexo A ou item 8.2.11)	εsh:	0,00025
Deformação total de retração calculado GL2000	εsh:	5,5E-04
Deformação total de retração adotado	εsh:	5,5E-04
Coeficiente de retração EN4 5.4.2.2	ηr:	18,69
Coeficiente de curta duração NBR 8800 O.1.2.1	η0:	6,80

A 2.2. AÇÕES VARIÁVEIS

A 2.2.1. AÇÃO VERTICAL: CARGA MÓVEL

Carga móvel	P (kN):	75
Carga móvel	Q (kN/m²):	5

$$CIV = \begin{cases} 1,35 \text{ , se } C \le 10\,\text{m} \\ 1 + 1,06 \cdot \left(\dfrac{20}{C + 50} \right), \text{ se } 10\,\text{m} < C \le 200\,\text{m} \end{cases}$$

$$CNF = 1 - 0,05 \cdot (nf - 2) \ge 0,9$$

Coef. de impacto vertical (NBR 7188 - 5.1.2.1)	CIV:	1,31
Coef. de número de faixas (NBR 7188 - 5.1.2.2)	CNF:	1,00
Coef. de impacto adicional (NBR 7188 - 5.1.2.3)	CIA:	1,25

A 2.2.2. AÇÃO VERTICAL: SOBRECARGA PEDETRES

Carga dos passeios (NBR 7188 item 5.1.1). Caso exista passeio adotar 3kN/m²	qp (kN/m²):	0

A 2.2.3. AÇÃO VERTICAL: SOBRECARGA DE CONSTRUÇÃO

Sobrecarga de construção (NBR 16694 - 6.2.2)	qc (kN/m²):	2
	Qc (kN/m):	5,46

A 2.2.4. AÇÃO HORIZONTAL: FRENAGEM E ACELERAÇÃO

$$H_f = 0,25 \cdot B_{ef} \cdot L \cdot CNF \ge 135\,\text{kN}$$

$$H_f = 0,25 \left(B - 3 - 2b_r \right) \cdot \left(C\text{'} - 6 \right) \cdot CNF \ge 135\,\text{kN}$$

Força horizontal (NBR 7188 - 5.2.1)	qf (kN/m²):	0,25
	Hf (kN):	135
	Hx (kN/longarina):	27

A frenagem foi inserida no modelo numérico como uma carga horizontal distribuída de 0.25 kN/m² ao longo do tabuleiro.

A 2.2.5. AÇÃO HORIZONTAL: VARIAÇÃO DE TEMPERATURA

Variação de temperatura encurtamento	ΔTen(°C):	-5,00
Variação de temperatura dilatação:	ΔTdi(°C):	12,00
Temperatura uniforme positiva	T+ (°C):	15,00
Temperatura uniforme negativa	T- (°C):	-15,00

A 2.2.6. AÇÃO HORIZONTAL: VENTO

$$V_k = V_0 \cdot S_1 \cdot S_2 \cdot S_3$$

$$q = 0.613 \cdot V_k^2$$

$$F_{an} = C_{an} \cdot q \cdot k$$

$$F_a = C_a \cdot q$$

Velocidade básica do vento (NBR6123 - 5.1)	V0 (m/s):	49
Fator topográfico (NBR6123 - 5.2)	S1:	1
Fator rugosidade (NBR6123 - 5.3)	S2:	0,85
Fator estatítico (NBR6123 - 5.4)	S3:	1,1
Velocidade característica do vento (NBR6123-4.2)	Vk (m/s):	45,8
Pressão dinâmica do vento (NBR6123 - 4.2)	q (kN/m2):	1,29
Fator redução (C/d) (NBR6123 tab11)	K:	0,73
Coeficiente de força (NBR6123 tab.12)	Ca:	2,05
Índice de área exposta (NBR6123 - 7.5 - pag 27)	Ø:	1,00
Relação comprimento/altura longarinas	b/h:	2,23
Fator proteção (NBR6123 fig.8 pag 29)	η:	0,41
Reticulados planos múltiplos (NBR 6123-7.6.1-pag 27)	Can:	5,43
Força de arrasto de N reticulados (NBR6123-7.6.2)	F_{an}(kN/m²):	5,10

A ação devido ao vento foi imposta no modelo numérico como uma carga distribuída ao longo do comprimento da ponte e como um momento fletor aplicado no nível da laje, conforme o esquema ilustrado abaixo.

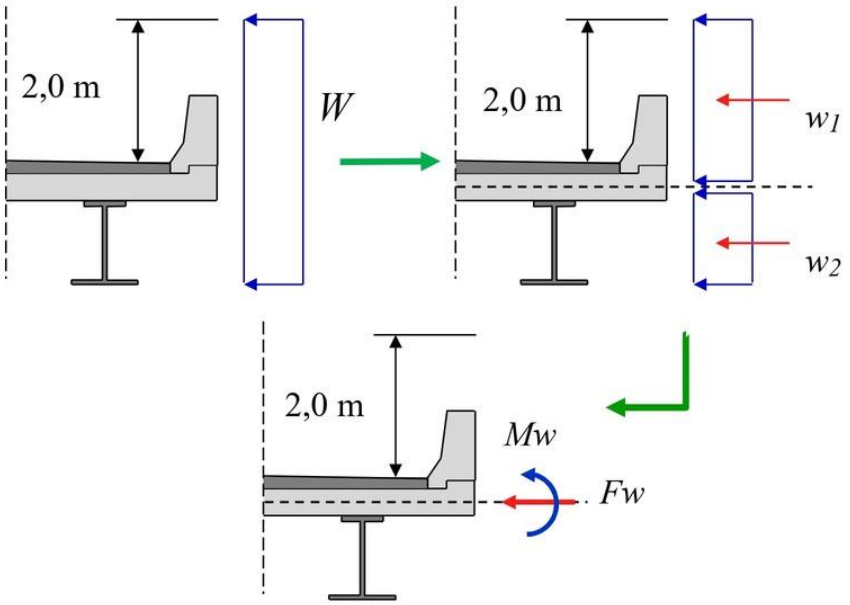


Figura A.2.1. Simplificação adotada no modelo numérico para simulação do vento no tabuleiro.

Força aplicada no nível da laje no modelo numérico	Fw (kN/m):	18,2
Momento aplicado no nível da laje no modelo numérico	Mw (kNm/m):	6,4

A 2.3. CICLOS DE FADIGA

$$VMDP = VMD \cdot tp$$

$$MDPF = VMDP \cdot p$$

$$n = \begin{cases} 1 & \text{se} \leq 12 \text{ m} \\ 2 & \text{se} > 12 \text{ m} \end{cases}$$

$$p = \begin{cases} 1 & se, nf = 1 \\ 0.85 & se, nf = 2 \\ 0.8 & se, nf \geq 3 \end{cases}$$

$$Nc = 365 \cdot VUP \cdot MDPF \cdot n$$

Vida útil de projeto para a fadiga (NBR 16694 A.4)	VUP (anos):	75
Classificação segundo o DNIT	Classe rodovia:	II
Consultar PNCT do DNIT para determinar o VMD (c/dia)	VMD (c/dia):	1000
tráfego pesado (AASHTO - tab.C3.6.1.4.2-1 pag3-32)	tp:	0,15
Volume médio diário de tráfego pesado	VMDP (/dia):	150
tráfego pesado/faixa (AASHTO-T.3.6.1.4.2-1 pag3-32)	p:	0,85
Média diária de tráfego pesado/faixa	MDPF (/dia):	128
Ciclos/passagem (AASHTO tab.6.6.1.2.5-2 pag6-50)	n:	1
Número de ciclos para fadiga em 75 anos	Nc:	3,5E+06
Número de ciclos segundo a NBR16694	N:	2,0E+06

A 2.4. FLECHA LIMITE

Flecha limite	Passagem de pedestres	Δlim(mm):	L/1000
	Carga móvel com impacto	Δlim(mm):	L/800

A 2.5. COMBINAÇÃO DE AÇÕES

As combinações de ações admitidas no dimensionamento das longarinas estão apresentadas na NBR16694 (2020) item 6.4.

A 2.5.1. ESTADOS LIMITES ÚLTIMOS (ELU): COMBINAÇÕES ÚLTIMAS DE CONSTRUÇÃO

$$F_d = \sum_{i=1}^{m} (\gamma_{g1} \cdot F_{Gi,k}) + \gamma_{q1} \cdot F_{qi,k} + \sum_{j=2}^{n} (\gamma_{qj} \cdot \psi_{0j,ef} \cdot F_{Qj,k})$$

A 2.5.2. ESTADOS LIMITES ÚLTIMOS (ELU): COMBINAÇÕES ÚLTIMAS NORMAIS

$$F_d = \sum_{i=1}^{m} (\gamma_{g1} \cdot F_{Gi,k}) + \gamma_{q1} \cdot F_{qi,k} + \sum_{j=2}^{n} (\gamma_{qj} \cdot \psi_{0j} \cdot F_{Qj,k})$$

ψ0W :	0,60
ψ0T :	0,60
ψ0M :	0,70

A 2.5.3. ESTADOS LIMITES ÚLTIMOS (ELU): COMBINAÇÃO ÚLTIMA EXCEPCIONAL

$$F_{d,ser} = \sum \gamma_G \cdot F_{gi,k} + F_{Q,exc} + \gamma_g \cdot \sum \psi_0 \cdot F_{qj,k}$$

Coeficiente ponderador vento	ψ0W :	0,60
Coeficiente ponderador temperatura	ψ0T :	0,60
Coeficiente ponderador carga móvel	ψ0M :	0,70

A 2.5.4. ESTADOS LIMITES DE SERVIÇO (ELS): COMBINAÇÃO FREQUENTE DE SERVIÇO

$$F_d = \sum_{i=1}^{m} F_{Gi,k} + \psi_1 \cdot F_{q1,k} + \sum_{j=2}^{n} (\psi_{2j} \cdot F_{Qj,k})$$

Coeficiente ponderador vento	ψ1W=	0,30	ψ2W=	0,00
Coeficiente ponderador temperatura	ψ1t =	0,50	ψ2t =	0,30
Coeficiente ponderador carga móvel	ψ1m =	0,50	ψ2m =	0,30

A 2.5.5. ESTADOS LIMITES DE SERVIÇO (ELS): COMBINAÇÃO RARA DE SERVIÇO

$$F_d = \sum_{i=1}^{m} F_{Gi,k} + F_{q1,k} + \sum_{j=2}^{n} (\psi_{1j} \cdot F_{Qj,k})$$

Coeficiente ponderador vento	ψ1W=	0,30
Coeficiente ponderador temperatura	ψ1t =	0,50
Coeficiente ponderador carga móvel	ψ1m =	0,50

A 2.5.6. ESTADOS LIMITES DE SERVIÇO (ELS): COMBINAÇÃO QUASE-PERMANENTE DE SERVIÇO

$$F_{d,ser} = \sum F_{gi,k} + \sum \psi_{2j} \cdot F_{qj,k}$$

Coeficiente ponderador vento	ψ2 W=	0,00
Coeficiente ponderador temperatura	ψ2t =	0,30
Coeficiente ponderador carga móvel	ψ2m =	0,30

A 2.5.7. COMBINAÇÃO FREQUENTE DE FADIGA

- Conectores de cisalhamento (AASHTO - 6.10.10.2):

$\Delta fad = \psi_{1,fad} \cdot qM$	ψ1fad =	1,50	MDPF ≥ 960
	ψ1fad =	0,75	MDPF < 960

$$\psi_{1,fad} = \begin{cases} 1{,}5 & para\ MDPF\ \geq 960\ caminhões\ por\ dia \\ 0{,}75 & para\ MDPF < 960\ caminhões\ por\ dia \end{cases}$$

- Cordão de Solda - Tensões normal e cisalhante no metal base:

A verificação do cordão de solda é descrito no anexo A da NBR 16694 nos itens A2.1 e A2.2 e deve ser conduzida considerado apenas os efeitos decorrentes das cargas permanentes e móvel. Não deve ser considerados os efeitos decorrentes da variação da temperatura (AASHTO tab3.4.1.1).

$$F_{d,fad} = \sum_{i=1}^{n} F_{Gi,k} + \psi_1 \cdot \sum_{j=1}^{n} F_{Qj,k}$$

$\tau \leq$	0.66 fy	Tensões normais
	0.40 fy	Tensões de cisalhamento

A 3. PROPRIEDADES GEOMÉTRICAS

A 3.1. SEÇÃO TRANSVERSAL DOS PERFIS DE AÇO

Serão utilizados dois perfis ao longo do vão, conforme ilustrado na Figura A.3.1.

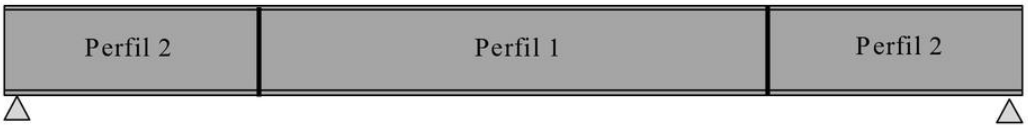
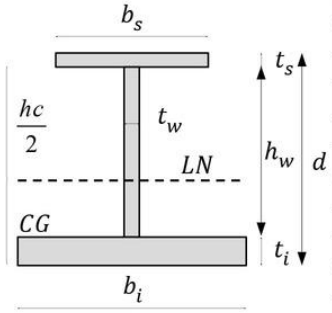


Figura A3.1. Disposição dos perfis de aço na longarina.

Dimensões:

Caso sejam adotadas duas ou mais seções transversais, por questões de construtibilidade, é recomendado variar apenas as larguras das mesas superior e inferior dos perfis. Para o caso de perfis não pintados, é necessário descontar, das espessuras das chapas, as sobre-espessuras necessárias à formação da pátina.

Δtw (mm):	0	(0) pintados / (sobrespessura) não-pintados



Dimensão	Perfil 1	Perfil 2
tw (mm):	12,5	12,5
d (mm):	1300	1300
ts (mm):	19,0	19,0
ti (mm):	22,4	22,4
bs (mm):	500	500
bi (mm):	500	500
*tw' (mm):	12,5	12,5
*ts' (mm):	19,0	19,0
*ti' (mm):	22,4	22,4
h (mm):	1259	1259

* dimensões descontando a sobre-espessura necessária à formação da pátina.

Critérios					P1	P2	Verif.
ABNT NBR 16694 (2020) item 7.3		tw (mm)	≥	8	12,5	12,5	ok
Recomendação prática	$\frac{L}{30} \leq b_s \leq \frac{C}{15}$	d (mm)	≥	640	1300	1300	ok
		d (mm)	≤	1333	1300	1300	ok
Recomendação prática	$(d + t_c) \geq \frac{L}{25}$	H (mm)	≥	768	1580	1580	ok
Recomendação prática		ts (mm)	≥	19	19	19	ok
AASHTO (2014) item C6.10.3.4-1	$b_s \geq \frac{L_i}{85}$	bs (mm)	≥	225,9	500	500	ok
NBR 16694 (2020) item 12.1	$\frac{h_w}{t_w} \leq 150$	hw/tw	≤	150	100,7	100,7	ok
NBR 16694 (2020) item 12.1	$\frac{b_f}{t_f} \leq 24$	bs/2ts	≤	12	13,2	13,2	não ok
		bi/2ti	≤	12	11,2	11,2	ok
NBR 16694 (2020) item 12.1	$b_f \geq \frac{h_w}{6}$	bs (mm)	≥	210	500	500	ok
		bi (mm)	≥	210	500	500	ok
NBR 16694 (2020) item 12.1	$t_f \geq 1{,}1 \cdot t_w$	ts (mm)	≥	14	19	19	ok
		ti (mm)	≥	14	22,4	22,4	ok
LRFD (2014) item 6.3.4.4.3	$t_f \leq 76\ mm$	tf (mm)	≤	76	22,4	22,4	ok

A 3.2. PROPRIEDADES GEOMÉTRICAS DO PERFIL DE AÇO

Propriedades geométricas		P1	P2
Área do perfil	A (mm2):	3,6E+04	3,6E+04
Peso do perfil	Peso (kN/m):	2,9	2,9
Peso do perfil	Peso (kN/m2):	1,0	1,0
CG da seção	CG (mm):	621,4	621,4
Inércia em torno do eixo x	Ix (mm4):	1,1E+10	1,1E+10
Módulo de resistência superior	Ws (mm3):	1,5E+07	1,5E+07
Módulo de resistência inferior	Wi (mm3):	1,7E+07	1,7E+07
Raio de giração em torno dex	rx (mm):	5,4E+02	5,4E+02
Inércia em torno de y	Iy (mm4):	4,3E+08	4,3E+08
Módulo de resistência em y	Wy (mm4):	1,7E+06	1,7E+06
Raio de giração em torno dey	ry (mm):	108,8	108,8
LNP em relação a mesa inf.	Ypl (mm):	583,7	583,7
Módulo de resistência plástico	Zx (mm3):	1,8E+07	1,8E+07
Constante de torção da seção transversal	J (mm4):	3,8E+06	3,8E+06
NBR8800 G2.2 Nota2	Cw:	1,8E+14	1,8E+14

Peso médio + 10% (kN/m²):	1,15	Peso estimado com base na seção transversal
Peso estimado (kN/m²):	1,65	Peso estimado com base no CBCA
Diferença	30,1%	

A 3.2.1. PROPRIEDADES: MESA SUPERIOR COMPRIMIDA

Propriedades geométricas		P1	P2
2x a altura da alma comprimida a partir do CG	hc (mm):	1319	1319
2x a altura da alma comprimida a partir do Ypl	hp (mm):	1,4E+03	1,4E+03
Inércia em Y mesa comprimida	Iyc (mm⁴):	2,0E+08	2,0E+08
Inércia em Y mesa tracionada	Iyt (mm⁴):	2,3E+08	2,3E+08
Área comprimida (mesa + 1/3 alma)	Ac (mm²):	12248	12248
Raio de giração (mesa + 1/3 alma)	ryT (mm):	127	127
NBR8800 - tab.G.1 (nota 9)	Iyc/Iyt:	0,85	0,85
$1/9 \le I_{yc}/I_{yt} \le 9$	Verificação:	**ok**	**ok**
Área da mesa superior	As (mm²):	9500	9500
Área da mesa inferior	Ai (mm²):	11200	11200
$(A_{fmenor} + t_w \cdot h_w)/A_{fmaior} \ge 1$	As/(Ai + Aa):	2,3	2,3
NBR8800 - tab.G.1 (nota 9)	Verificação:	**ok**	
Centroíde do T comprimido	dT (mm):	511	511
Momento de inércia do T comprimido	IdT (mm^4):	8,1E+08	8,1E+08
Área do T comprimido	AdT (mm²):	17745	17745
Raio de giração em torno de x do T comprimido	rxc (mm):	213	213
Raio de giração em torno de y do T comprimido	ryc (mm):	105,6	105,6

- Classificação perfil (NBR8800 tabG.1):

Perfil I monossimétrico (NBR 8800 - tab.G.1)

Perfil:	P1	P2
λr (alma):	137	137
λ:	106	106
λp (alma):	77,5	77,5

Condição perfil:	P1 - Alma não esbelta (NBR8800 AnexoG)
	P2 - Alma não esbelta (NBR8800 AnexoG)

$$\lambda = \frac{h_c}{t_w}$$

$$\lambda = \frac{\frac{h_c}{h_p} \cdot \sqrt{\frac{E}{f_y}}}{\left(0{,}54\frac{M_{pl}}{M_r} - 0{,}09\right)} \leq \lambda_r$$

$$\lambda_r = 5{,}7 \cdot \sqrt{\frac{E}{f_y}}$$

$$M_r = f_y \cdot W$$

$$M_{pl} = Z \cdot f_y$$

A 3.2.2. PROPRIEDADES: MESA SUPERIOR TRACIONADA

Propriedades geométricas			P1	P2
Módulo elástico superior (tracionado)		Ws (mm3):	1,5E+07	1,5E+07
Módulo elástico inferior (comprimido)		Wi (mm3):	1,7E+07	1,7E+07
2x a altura da alma comprimida a partir do CG		hc' (mm):	1198	1198
2x a altura da alma comprimida a partir do Ypl		hp' (mm):	1123	1123
Área comprimida (mesa + 1/3 alma)		Ac' (mm²):	13696	13696
Raio de giração área comp (mesa inf+1/3alma)		ryT' (mm):	131	131
Área comprimida (mesa + alma)		Acc' (mm²):	18687	18687
Raio de giração área comp (mesa inf+alma)		ryc' (mm):	112	112
Inércia em Y mesa comprimida		Iyc' (mm⁴):	2,3E+08	2,3E+08
Inércia em Y mesa tracionada		Iyt' (mm⁴):	2,0E+08	2,0E+08
NBR8800 - tab.G.1	$1/9 \leq I_{yc}/I_{yt} \leq 9$	Iyc'/Iyt':	1,18	1,18
		Verificação	**ok**	**ok**

- Classificação perfil (NBR8800 tabG.1):

Perfil I monossimétrico (NBR 8800 - tab.G.1)

Propriedades	P1	P2
Perfil:	MS	MS
Mpl (kNm):	6256	6256
Mr (kNm):	5346	5346
λp (alma):	87,5	87,5
λr (alma):	137	137
λ:	96	96

MS - Monosimétrico

Condição perfil:	P1 - Alma não esbelta (NBR8800 AnexoG)
	P2 - Alma não esbelta (NBR8800 AnexoG)

$$\lambda = \frac{h_c}{t_w}$$

$$\lambda = \frac{\frac{h_c}{h_p} \cdot \sqrt{\frac{E}{f_y}}}{\left(0{,}54\frac{M_{pl}}{M_r} - 0{,}09\right)} \leq \lambda_r$$

$$\lambda_r = 5{,}7 \cdot \sqrt{\frac{E}{f_y}}$$

$$M_r = f_y \cdot W$$

$$M_{pl} = Z \cdot f_y$$

A 3.3. LARGURA EFETIVA

A largura efetiva (bef) foi calculada segundo as recomendações apresentadas no item O.2.2 da NBR8800:2008, em que bef é dada como o menor dos seguintes valores:

$b_{ef} \leq \begin{cases} 2 \cdot C/8 \\ 2 \cdot b/2 \\ bl + b/2 \end{cases}$	bef (mm) ≤	2400
		2900
		2475
	bef (mm):	2400

A 3.4. PROPRIEDADES GEOMÉTRICAS DA SEÇÃO MISTA

Segundo o item O.1.1.2 da NBR8800 (2008), seções com h/tw < 3.76 · raiz(E/fy) podem ser dimensionadas usando as propriedades plásticas da seção mista. Para relações h/tw > 3,76 · raiz(E/fy) devem ser utilizadas as propriedades elásticas da seção mista.

$$\frac{h}{t_w} \leq 3{,}76 \cdot \sqrt{\frac{E}{f_y}} \quad \rightarrow \text{propriedades plásticas}$$

$$\frac{h}{t_w} > 3{,}76 \cdot \sqrt{\frac{E}{f_y}} \quad \rightarrow \text{propriedades elásticas}$$

ID	P1	P2
h (mm):	1259	1259

As propriedades geométricas da seção foram cálculadas com base nos coeficientes de carga de curta e longa duração, indicados na NBR8800 O.1.2.1. Para cálculo das propriedades da seção transversal devido à retração e variação de temperatura foram utilizados os coeficientes recomendados na EN 4 item 5.4.2.2.

Coeficiente de homogeneização para carga de curta duração	$\eta 0$:	6,8
Coeficiente de homogeneização para carga de longa duração	ηL:	20,4
Coeficiente de fluência (NBR6118 8.2.11)	φr:	3,18
Deformação de retração (NBR6118 8.2.11)	εr:	0,00055
Coeficiente de homogeneização para retração	ηr:	18,7

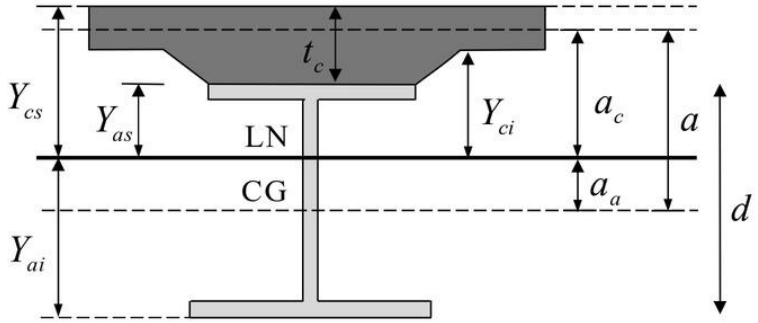


Figura A.3.2. Seção transversal esquemática.

Dimensão	Propriedades elásticas da seção					
	Curta Duração		Longa Duração		Retração	
	P1	P2	P1	P2	P1	P2
Yai (mm):	1219	1219	1010	1010	1028	1028
aa (mm):	598	598	389	389	407	407
Ycs (mm):	361	361	570	570	552	552
Yci (mm):	181	181	290	290	272	272
Yas (mm):	81	81	290	290	272	272
ac (mm):	221	221	430	430	412	412

Propriedade	LN no P	LN no P	LN no P	LN no P	LN no P	LN no P
Asm (mm²):	135226	135226	69364	69364	72385	72385
IxM (mm⁴):	2,9E+10	2,9E+10	2,2E+10	2,2E+10	2,3E+10	2,3E+10
Wcs (mm³):	8,0E+07	8,0E+07	3,9E+07	3,9E+07	4,1E+07	4,1E+07
Wci (mm³):	1,6E+08	1,6E+08	7,7E+07	7,7E+07	8,4E+07	8,4E+07
Was (mm³):	3,6E+08	3,6E+08	7,7E+07	7,7E+07	8,4E+07	8,4E+07
Wai (mm³):	2,4E+07	2,4E+07	2,2E+07	2,2E+07	2,2E+07	2,2E+07

Como as pré-lajes de concreto permanecerão incorporadas no tabuleiro Yci será igual a Yas.

$$\eta 0 = \frac{E}{Ecs}$$

$$\eta L = 3\eta 0$$

$$\eta r = \eta 0 \cdot (1 + 0{,}55 \cdot \varphi(t, t0))$$

$$aa = Yai - CG$$

$$Ycs = tc + h - \text{Yai}$$

$$Yai = \left[\frac{\frac{bef}{\eta} \cdot tc \cdot \left(h + \frac{tc}{2}\right) + A \cdot CG}{\frac{bef}{\eta} \cdot tc + A}\right]$$

$$Yci = h - \text{Yai}$$

$$Yas = d - \text{Yai}$$

$$ac = \frac{tc}{2} + d - Yai$$

$$Asm = tw \cdot h + bs \cdot ts + bi \cdot ti + \frac{bef \cdot tc}{\eta}$$

$$Ixt(i) \leq \begin{cases} Ix + A \cdot aa^2 + \dfrac{bef}{\eta} \cdot \dfrac{tc^3}{12} + \dfrac{bef \cdot tc \cdot ac^2}{\eta} & \square \quad , se\ Yai \ \leq d \\ Ix + A \cdot aa^2 + \dfrac{bef}{\eta} \cdot \dfrac{tc^3}{12} + \dfrac{bef \cdot tc}{\eta} \cdot (d + \dfrac{tc}{2} - Yai)^2 & se\ Yai > d \end{cases}$$

$$Wcs = \frac{Ixtr}{Ycs}$$

$$Wci = \frac{Ixtr}{Yci}$$

$$Was = \frac{Ixtr}{Yas}$$

$$Wai = \frac{Ixtr}{Yai}$$

ID	η0	ηL	ηr
IxM	IxM0	IMXL	IxMr
Wcs	Wcs0	WcsL	Wcsr
Wci	Wci0	WciL	Wcir
Was	Was0	WasL	Wasr
Wai	Wai0	WaiL	Wair

A 4. ESFORÇOS SOLICITANTES

Os esforços solicitantes nas longarinas foram obtidos a partir do modelo numérico. Para fins de segurança, não foi admitido o aumento da resistência do concreto com o tempo.

A 4.1. RESUMO DOS ESFORÇOS NUMÉRICOS

- Esforços sem ponderação: longarina

Modelo numérico	Esforços:	M (kNm)			V (kN)		
	Seção:	S0	S1	S2	S0	S1	S2
	x (m):	0,40	4,05	10,00	0,40	4,05	10,00
Beam	G1	14	107	167	35	20	0
Beam	G2	78	654	1014	199	125	5
Girder	G3	16	115	164	41	19	1
Girder	G4	25	212	327	65	40	2
Girder	G5	21	176	272	54	33	1
Girder	qM	153	1106	1636	25	77	153
Girder	qmn	-10	-82	-131	389	267	153
Beam	qc	22	187	289	57	36	1
Beam	R	535	865	900	129	28,9	1
Beam	T+	-1	-4	-5	3	1,0	0
Beam	T-	1	3	4	3	0,9	0

- Esforços nos diafragmas:

ID Modelo Numéric o	Esforços:	Q (kN)							
	Seção:	D0	D1	D2	D3	D4	D5	D6	D7
	x (m):	0,40	6,80	13,2	19,6	-	-	-	-
Beam	G1	35,3	12,4	9,2	30,7	-	-	-	-
Beam	G2	198,9	69,6	60,3	190,2	-	-	-	-
Girder	G3	40,8	9,8	9,8	40,8	-	-	-	-
Girder	G4	65,1	22,0	22,0	65,1	-	-	-	-
Girder	G5	54,3	18,4	18,4	54,3	-	-	-	-
Girder	qM	24,6	108,3	209,1	389,1	-	-	-	-
Girder	qmn	389,1	209,1	108,3	24,6	-	-	-	-
Beam	qc	56,8	19,9	17,2	54,4	-	-	-	-
Beam	R	129,0	14,3	11,0	105,9	-	-	-	-
Beam	T+	3,4	0,6	0,5	2,9	-	-	-	-
Beam	T-	3,1	0,5	0,4	2,6	-	-	-	-

A 4.2. ESFORÇOS FINAIS NUMÉRICOS COM CARGA MÓVEL PONDERADA:

Coef. de impacto vertical	CIV:	1,31
Coef. de número de faixas	CNF:	1,00
Coef. de impacto adicional	CIA:	1,25
Considerar CIA no dimensionamento?		N
Coeficiente de impacto	CI:	1,31

- Apoio e Emendas:

Esforços:		M (kN/m)			Q (kN)		
Seção:		S0	S1	S2	S0	S1	S2
Posição	x (m):	0,40	4,1	10,0	0,40	4,1	10,0
PP - estrutura metálica	G1	14	107	167	35	20	0
PP - da laje de concreto	G2	78	654	1014	199	125	6
Peso das defensas	G3	16	115	164	41	19	1
PP - revestimento asfáltico	G4	25	212	327	65	40	2
PP - recapeamento	G5	21	176	272	54	33	2
Carga móvel	qM	249	1445	2137	32	101	200
Carga móvel	qmn	-13	-107	-170	508	349	200
Sobrecarga de construção	qc	22	20	289	57	36	1
Retração	R	535	865	900	129	29	1
Temperatura	T+	-1	-4	-5	3	1	0
Temperatura	T-	1	3	4	3	1	0

- Esforços nos diafragmas:

ID Modelo Numérico	Esforços:	Q (kN)							
	Seção:	D0	D1	D2	D3	D4	D5	D6	D7
	x (m):	0,40	6,80	13,20	19,60	-	-	-	-
Beam	G1	35	12	9	31	-	-	-	-
Beam	G2	199	70	60	190	-	-	-	-
Girder	G3	41	10	10	41	-	-	-	-
Girder	G4	65	22	22	65	-	-	-	-
Girder	G5	54	18	18	54	-	-	-	-
Girder	qM	32	141	273	508	-	-	-	-
Girder	qmn	508	273	141	32	-	-	-	-
Beam	qc	57	20	17	54	-	-	-	-
Beam	R	129	14	11	106	-	-	-	-
Beam	T+	3	1	0	3	-	-	-	-
Beam	T-	3	1	0	3	-	-	-	-

A 5. VERIFICAÇÃO DO PERFIL

A 5.1. ETAPA 0: IÇAMENTO

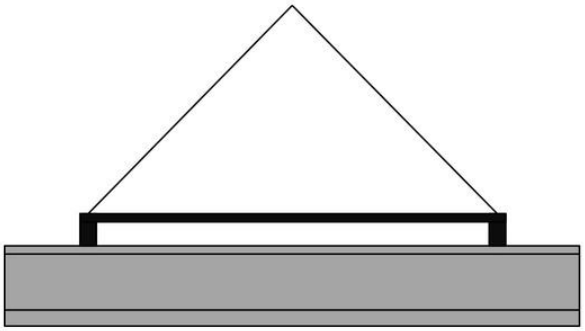


FIGURA A5.1. Caso 1 - Içamento: viga isolada sem contraventamento.

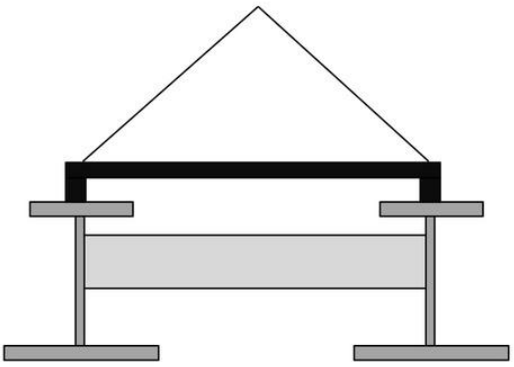


FIGURA A5.2. Caso 2 - Içamento: viga com contraventamento.

A 5.1.1. ESFORÇOS SOLICITANTES

COMB:	1.15G1c

Coeficiente ponderador da carga permanente	$\gamma g1$:	1,15
(1) Sem contraventamento / (2) Com contraventamento	Caso:	2
Posição do içamento: recomendado L/4 (FHWA-15-044)	S (m):	5,00
Inclinação do cabo em relação ao eixo da longarina	α:	90°
Espaçamentoentrecontraventamentos	Lb (mm):	6400

Propriedades geométricas	Perfil	P1	P2
Posição da seção avaliada	x (m):	5,0	4,1
Momento devido peso próprio em L/4	Mz (kNm):	-38	-28
Mesa superior tracionada	$\lvert\sigma xs\text{-}sd\rvert$ (MPa):	2,8	2,1
Mesa inferior comprimida	$\lvert\sigma xi\text{-}sd\rvert$ (MPa):	2,6	1,9

→ P2: Mesa superior tracionada

$$M_{zm} = \frac{g_{1i} \cdot L^2}{32} \qquad \sigma_{xi-sd} = 1{,}15 \cdot \frac{M_z}{W_{i(i)}} \qquad \sigma_{xs-sd} = 1{,}15 \cdot \frac{M_z}{W_{s(i)}}$$

A 5.1.2. CÁLCULO DO MOMENTO RESISTENTE DE CÁLCULO (MRD): NBR8800 - ANEXO G

γa1:	1,1

A) ELU: FLAMBAGEM LATERAL POR TORÇÃO (FLT) - NBR8800 G2.1:

$$M_{Rd} = \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda \le \lambda_p$$

$$M_{Rd} = \frac{C_b}{\gamma_{a1}} \cdot \left[M_{pl} - (M_{pl} - M_r) \cdot \frac{\lambda - \lambda_p}{\lambda_r - \lambda_p} \right] \le \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda_p < \lambda \le \lambda_r$$

$$M_{Rd} = \frac{M_{cr}}{\gamma_{a1}} \le \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda > \lambda_r$$

$$\lambda = \frac{L_b}{r_{yc}}$$

$$\lambda_p = 1{,}76 \sqrt{\frac{E}{F_y}}$$

$$\lambda_r = \frac{1{,}38 \sqrt{I_y J}}{r_{yc} \cdot J \cdot \beta_1} \cdot \sqrt{\beta_2 + \sqrt{\beta_2^2 + 27 \cdot C_w \cdot \beta_1^2}}$$

$$\beta_1 = \frac{(f_y - \sigma_r) \cdot W_c}{EJ}$$

$$\beta_2 = 5{,}2 \cdot \beta_1 \cdot \beta_3 + 1$$

$$\beta_3 = 0{,}45 \cdot (d - \frac{t_{fs} + t_{fi}}{2}) \cdot (\frac{\alpha_y - 1}{\alpha_y + 1})$$

$$\alpha_y = \frac{I_{yc}}{I_{yt}}$$

$$C_w = \frac{(d - \frac{t_{fs} + t_{fi}}{2})^2}{12} \cdot (\frac{t_{fi} \cdot b_{fi}^3 \cdot t_{fs} \cdot b_{fs}^3}{t_{fi} \cdot b_{fi}^3 + t_{fs} \cdot b_{fs}^3})$$

$$M_{cr} = \frac{C_b \cdot \pi^2 \cdot E \cdot I_y}{L_b^2} \cdot \left[\beta_3 + \sqrt{\beta_3^2 + \frac{C_w}{I_y} \cdot (1 + 0{,}039 \cdot \frac{J \cdot L_b^2}{C_w}} \right]$$

$$M_r = (f_y - \sigma_r) \cdot W_c \le f_y \cdot W_t$$

$$M_{pl} = Z \cdot f_y$$

Fator de modificação do diagrama de momento	Cb:	1
Módulo de elasticidade do aço	E (MPa):	200000
Resistência ao escoamento do aço	fy (MPa):	345
Comprimento destravado	Lb (mm):	6400,0
Coeficiente ponderador	γa1:	1,1

Perfil		P1	P2
Const. de empenamento da seção transversal	Cw:	1,8E+14	1,8E+14
Inércia em y Tcompr/ inércia em y Ttracionado	αy:	1,18	1,18
Coeficientes β	β3:	47	47
Coeficientes β	β1:	0,005	0,005
Coeficientes β	β2:	2,31	2,31

Parâmetro de esbeltez corresp. à plastificação	λp:	42,4	42,4
Parâmetro de esbeltez corresp. ao escoamento	λr:	110,2	110,2
Parâmetro de esbeltez	λ:	57,3	57,3

Condição de esbeltez:		$\lambda p < \lambda < \lambda$	$\lambda p < \lambda < \lambda r$
Momento fletor flambagem elástica	Mcr (kNm):	14500	14500
Momento fletor início do escoamento	Mr (kNm):	4087	4087
Momento fletor de plastificação	Mpl (kNm):	6256	6256

$\lambda p < \lambda < \lambda r$	Mrd1 (kNm):	5780	5780
$\lambda > \lambda r$	Mrd2 (kNm):	6256	6256
Momento fletor resistente	**Mrd (kNm):**	**5255**	**5255**

B) ELU: FLAMBAGEM LOCAL DA MESA (FLM) - NBR8800 G2.2:

$$M_{Rd} = \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda \le \lambda_p$$

$$M_{Rd} = \frac{1}{\gamma_{a1}} \cdot \left[M_{pl} - (M_{pl} - M_r) \cdot \frac{\lambda - \lambda_p}{\lambda_r - \lambda_p} \right] \qquad \text{se } \lambda_p < \lambda \le \lambda_r$$

$$M_{Rd} = \frac{M_{cr}}{\gamma_{a1}} \qquad \text{se } \lambda > \lambda_r$$

$$\lambda = \frac{b}{t}$$

$$\lambda_p = 0,38 \sqrt{\frac{E}{F_y}}$$

$$\lambda_r = 0,95 \cdot \sqrt{\frac{E}{(f_y - \sigma_r)/k_c}}$$

$$k_c = \frac{4}{\sqrt{h/t_w}} \qquad \text{em que } 0,35 \le k_c \le 0,76$$

$$\sigma_r = 0,3 \cdot f_y$$

$$M_{cr} = \frac{0,90 \cdot E \cdot k_c \cdot W_c}{\lambda^2}$$

$$M_r = (f_y - \sigma_r) \cdot W_c$$

$$M_{pl} = Z \cdot f_y$$

Perfil		P1	P2
Parâmetro de esbeltez de plastificação	λp:	9,1	9,1
Coeficiente	kc:	0,40	0,40
Parâmetro de esbeltez - escoamento	λr:	17,3	17,3
Parâmetro de esbeltez	λ:	11,2	11,2

Condição:		λp < λ < λr	λp < λ < λr
Momento flambagem elástica	Mcr (kNm):	9748	9748
Momento de início de escoamento	Mr (kNm):	4087	4087
Momento de plastificação da seção trans	Mpl (kNm):	6256	6256
Momento resistente	**Mrd (kNm):**	**5199**	**5199**

C) ELU: FLAMBAGEM LOCAL DA ALMA - NBR8800 G2.2:

$$M_{Rd} = \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda \le \lambda_p$$

$$M_{Rd} = \frac{1}{\gamma_{a1}} \cdot \left[M_{pl} - (M_{pl} - M_r) \cdot \frac{\lambda - \lambda_p}{\lambda_r - \lambda_p} \right] \qquad \text{se } \lambda_p < \lambda \le \lambda_r$$

Consultar anexo H $\qquad$ se $\lambda > \lambda_r$

Perfil		P1	P2
Esbeltez de início do escoamento	λr:	137,2	137,2
Esbeltez de plastificação	λp:	87,5	87,5
Esbeltez	λ:	95,8	95,8
Momento de plastificação	Mpl (kNm):	6256	6256
Momento de início de escoamento	Mr (kNm):	5346	5346
Verificação	Condição:	λp < λ < λr	λp < λ < λr
Momento resistente	**Mrd (kNm):**	**5548**	**5548**

D) MOMENTO RESISTENTE DE CÁLCULO (MRD):

Perfil		P1	P2
FLT	Mrd (kNm):	5255	5255
FLM	Mrd (kNm):	5199	5199
FLA	Mrd (kNm):	5548	5548
Mrd (kNm):		**5199**	**5199**

E) VERIFICAÇÃO

Perfil	P1	P2
x (mm):	5,00	4,05
Msd/Mrd:	0,01	0,01
Verificação	**OK**	

A 5.1.3. CISALHAMENTO ALMA (NBR8800 5.4.3.1)

$$V_{Rd} = \begin{cases} \dfrac{V_{pl}}{\gamma_{a1}} & \text{se } \lambda \le \lambda_p \\ \dfrac{\lambda_p \cdot V_{pl}}{\lambda \cdot \gamma_{a1}} & \text{se } \lambda_p < \lambda \le \\ 1{,}24 \cdot \left(\dfrac{\lambda_p}{\lambda}\right)^2 \cdot \dfrac{V_{pl}}{\gamma_{a1}} & \text{se } \lambda > \lambda_r \end{cases}$$

$$V_{pl} = 0{,}6 \cdot A_w \cdot f_y$$

$$\lambda = \frac{h}{t_w}$$

$$\lambda_p = 1{,}10 \cdot \sqrt{\frac{k_v \cdot E}{f_y}}$$

$$\lambda_r = 1{,}37 \cdot \sqrt{\frac{k_v \cdot E}{f_y}}$$

$$A_w = d \cdot t_w$$

$$k_v = \begin{cases} 5 \text{ almas sem enrijecedores transversais}, \dfrac{a}{h} > 3 \text{ ou a/h} > \left[\dfrac{260}{(h/t_w)}\right]^2 \\ 5 + \dfrac{5}{\left(\dfrac{a}{h}\right)^2} \end{cases}$$

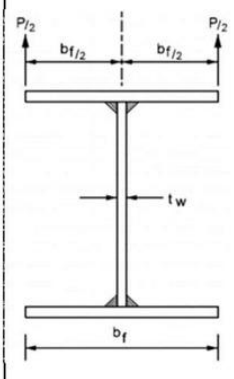
Carga concentrada no P1	Vsd (kN):	34,6
Distância entre enrijecedores	a (mm):	0
NBR8800 5.4.3.1.1	kv:	5,0
NBR8800 5.4.3.1.2	Vpl (kN):	3364

Perfil		P1
Esbeltez de plastificação	λp:	59,2
Esbeltez de início do escoamento	λr:	73,8
Esbeltez	λ:	100,7
Força cortante	Vrd (kN):	1312
Verificação	**OK - Vrd > Vsd**	

A 5.1.4. FORÇAS CONCENTRADAS NO IÇAMENTO

- Flexão local da mesa (NBR8800 5.7.2):

$$P_{Rd} \ge P_{sd}$$



$$\frac{P_{sd}}{2} = \gamma_{g1} \cdot \frac{g_1 \cdot L}{2 \cdot 2}$$

$$P_{Rd} = \begin{cases} P_{Rd} = \dfrac{6{,}25 \cdot t_f^2 \cdot f_y}{\gamma_{a1}} & \text{se } S \ge 10 \cdot t_f \\ P_{Rd} = \dfrac{6{,}25 \cdot t_f^2 \cdot f_y}{2 \cdot \gamma_{a1}} & \text{se } S < 10 \cdot t_f \end{cases}$$

Distância entre a extremidade e o ponto de aplicação da força	S (m):	5
Carga concentrada resistente	Prd (kN):	707,0
Carga concentrada solicitante	Psd (kN):	34,6
Verificação Prd > Psd	**OK**	

- Escoamento local da alma (NBR8800 5.7.3):

$$F_{Rd} = \begin{cases} F_{Rd} = \dfrac{1{,}10 \cdot (5 \cdot k + l_n) \cdot f_y \cdot t_w}{\gamma_{a1}} & \text{se } S \geq d \\ F_{Rd} = \dfrac{1{,}10 \cdot (2{,}5 \cdot k + l_n) \cdot f_y \cdot t_w}{\gamma_{a1}} & \text{se } S < d \end{cases}$$

Distância entre a borda e o ponto de aplicação da força	S (m):	5
Comprimento da força na direção longitudinal da viga	ln (mm):	0
Espessura da mesa que recebe o carregamento	k (mm):	19
Força resistente	Frd (kN):	205
Força solicitante de cálculo	Fsd (kN):	34,6
Verificação (Frd ≥ Fsd)	**OK**	

A 5.2. ETAPA 1: CONCRETAGEM

A 5.2.1. ESFORÇOS SOLICITANTES

Combinação última de construção	$F_d = \sum_{i=1}^{m} (\gamma_{g1} \cdot F_{Gi,k}) + \gamma_{q1} \cdot F_{qi,k} + \sum_{j=2}^{n} (\gamma_{qj} \cdot \psi_{0j,ef} \cdot F_{Qj,k})$

COMB2:	1.15G1 + 1.25G2 + 1.3qc + 1.2 · 0.6W
COMB3:	1.15G1 + 1.25G2 + 1.3 · 0.7qc + 1.2W

Coeficiente ponderador das ações permanentes	γG1:	1,15
	γG2:	1,25
Coeficiente ponderador sobrecarga de construção	γqc:	1,3
Coeficiente ponderador ação do vento	γW:	1,2
Coeficiente ponderador sobrecarga de construção	ψqc:	0,7
Coeficiente ponderador ação do vento	ψ0W:	0,6

A 5.2.1.1. AÇÃO VERTICAL

Travamento	(1) Apenas com os diafragmas		1
	(2) travamento intermediário		
Distância entre travamentos		Lb (mm):	6400
		Lbi (mm):	6400

Ok

Esforços solicitantes carregamento vertical								
	S0 - P2: x = 0,4 m		S1 - P2: x = 4,05 m			S2 - P1: x = 10 m		
Ações	Q (kN)	τa (MPa)	M (kNm)	σsa (MPa)	σai (MPa)	M (kNm)	σsa (MPa)	σai (MPa)
G1	35	2,2	107	6,9	6,3	167	10,8	9,9
G2	199	12,6	654	42,2	38,7	1014	65,4	59,9
qc	57	3,6	20	1,3	1,2	289	18,7	17,1

Posição	x (m) =	10,0	x (m) =	4,1
Perfil	P1		P2	
Combinações	COMB1	COMB2	COMB1	COMB2
σsa-sd (MPa):	118,4	111,2	62,4	61,9
σai-sd (MPa):	108,4	101,8	57,1	56,7

A 5.2.1.2. AÇÃO HORIZONTAL

Durante a cura do concreto, o vento e a carga vertical que atua nos balanços irão provocar ações laterais nas longarinas de extremidade e devem ser consideradas na verificação dessas.

I) Balanços

Os efeitos da torção nas longarinas de extremidade, devido às cargas que atuam nos balanços, foram considerados segundo as prescrições apontadas na AASHTO item 6.10.8.3.

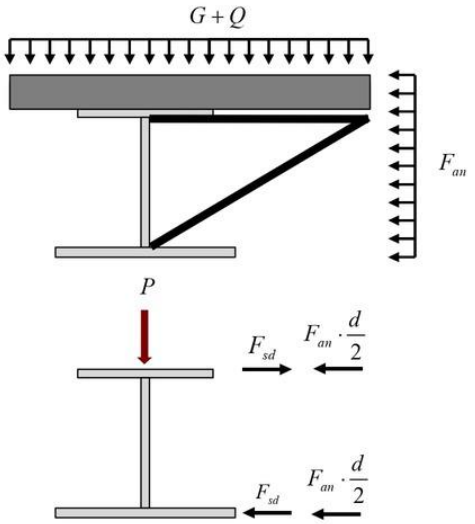


Figura A5.4. Carga nos balanços.

Perfil		P1	P2
Altura da alma	h(mm):	1259	1259
Módulo elástico do flange superior	Wyfs (mm3):	5,7E+05	5,7E+05
Módulo elástico do flange inferior	Wyfi (mm3):	9,4E+05	9,4E+05
Raio de giração da seção comprimida	ryT (mm):	127	127

Comprimento destravado	Lb (m):	6,40
Espaçamento entre a estrutura de apoio	Sd (mm)	1000

Força horizontal nas mesas		
Devido ao concreto fresco nos balanços	Fh-G2 (kN/m):	2,83
Devido à sobrecarga de construção	Fh-qc (kN/m):	0,81
Devido ao vento	Fw (kN/m):	3,31

$$F_{h-G2}(kN/m) = \left(G_2 \cdot \frac{b_l^2}{2}\right) \cdot \frac{1}{h}$$

$$F_{h-qc}(kN/m) = \left(q_c \cdot \frac{b_l^2}{2}\right) \cdot \frac{1}{h}$$

$$F_w(kN/m) = F_{an} \cdot \frac{d}{2}$$

$$\text{COMB1: } F_{h-sd}(kN/m) = \gamma_{G2} \cdot F_{h-G2} + \gamma_{qc} \cdot F_{h-qc} + \gamma_w \cdot \psi_w \cdot F_w$$

$$\text{COMB2: } F_{h-sd}(kN/m) = \gamma_{G2} \cdot F_{h-G2} + \gamma_{qc} \cdot \psi_c \cdot F_{h-qc} + \gamma_w \cdot F_w$$

Ação	G2	qc	W
COMB1:	1,25	1,3	0,72
COMB2:	1,25	0,91	1,2

Mesa superior					
Posição		x (m) =	10,0	x (m) =	4,1
Perfil		P1		P2	
Combinações		COMB1	COMB2	COMB1	COMB2
Força horizontal na mesa sup de cálculo	Fh-sd (kN/m):	2,20	0,30	2,20	0,30
Momento em torno de y mesa superior	Mys-sd (kNm):	9,0	1,2	9,0	1,2
Tensão normal mesa superior	σls1-sd (MPa):	15,8	2,1	15,8	2,1

Mesa inferior					
Posição		x (m) =	10,0	x (m) =	4,1
Perfil		P1		P2	
Combinações		COMB1	COMB2	COMB1	COMB2
Força horizontal na mesa inf de cálculo	Fh-sd (kN/m):	6,97	8,25	6,97	8,25
Momento em torno de y mesa inferior	Mys-sd (kNm):	28,6	33,8	28,6	33,8
Tensão normal mesa inferior	σli1-sd (MPa):	30,5	36,1	30,5	36,1

Verificação		
	σls1-sd ≤ 0.6fy	**OK: σls1-sd ≤ 0.6fy**
	σli1-sd ≤ 0.6fy	**OK: σli1-sd ≤ 0.6fy**

$$M_{y-sd}(kNm) = \frac{F_{h-sd} \cdot L_b^2}{10}$$

$$\sigma_{l-sd}(kN/m^2) = \frac{M_{y-sd}}{W_{y-fs}}$$

$$\sigma_{li1-sd}(kN/m^2) = \frac{M_{y-sd}}{W_{y-fi}}$$

II) Verificação dos efeitos de 2º ordem:

Segundo a AASHTO, caso o comprimento destravado de uma longarina seja superior ao comprimento limite para consideração das tensões de 1 ordem, deve-se ponderar os esforços solicitantes de 1 ordem a fim de considerar os efeitos de 2º ordem. A ponderação é feita a partir do fator βl.

□ se $Lp \geq Lb$:

$\beta l = 1$

□ se $Lp < Lb$:

$$\begin{cases} \dfrac{0,85}{(1-\sigma_{sd}/\sigma_{cr})} \leq 1 \quad \rightarrow \quad \beta l = 1 \\ \dfrac{0,85}{(1-\sigma_{sd}/\sigma_{cr})} > 1 \quad \rightarrow \quad \beta l = \dfrac{0,85}{(1-\sigma_{sd}/\sigma_{cr})} \end{cases}$$

$$L_p = r_t \cdot \sqrt{\frac{E}{f_y}}$$

$$L_b \leq 1.2 \cdot L_p \cdot \sqrt{\frac{C_b \cdot 1}{\sigma / f_y}}$$

$$r_t = \sqrt{\frac{I_{yT}}{A_T}}$$

Perfil		P1	P2
Comprimento limite para σ = fy	Lp (mm):	3061	3061
Comprimento destravado limite para 1° ordem	Lb (mm):	8636	6269
1° Verificação Lb		não ok	
2° Verificação Lb	Lblim < Lb - não ok: amplificar tensão normal		

Posição		x (m) =	10,0	x (m) =	4,1
Perfil		P1		P2	
Combinação		COMB1	COMB2	COMB1	COMB2
Tensão critíca flambagem	σcr (MPa):	778,8		778,8	
Fator de amplificação	βl:	1,0	1,0	1,0	1,0
Tensão normal flange superior	σls1-sd (MPa):	15,8	2,1	15,8	2,1
Tensão normal flange inferior	σli1-sd (MPa):	30,5	36,1	30,6	36,1

Verificação	σls1-sd ≤ 0.6fy	ok: σls1-sd ≤ 0.6fy
	σli1-sd ≤ 0.6fy	ok: σli1-sd ≤ 0.6fy

A 5.2.2. CÁLCULO DO MOMENTO SOLICITANTE - MSD (NBR8800 - ANEXO G)

$$\sigma_{sf-sd} = \sigma_{sd} + \sigma_{ls-sd}$$

$$\sigma_{if-sd} = \sigma_{id} + \sigma_{li-sd}$$

Posição	x =	10 m	x =	4,05 m
Perfil	P1		P2	
Combinação	COMB1	COMB2	COMB1	COMB2
σsf-sd (MPa):	134	113	78	64
σif-sd (MPa):	139	138	88	93

A 5.2.3. CÁLCULO DO MOMENTO RESISTENTE DE CÁLCULO - MRD (NBR8800 - ANEXO G)

γa1:	1,1

A) ELU: FLAMBAGEM LOCAL DA ALMA (FLA) - NBR8800 G2.2:

$$\lambda = \frac{h_c}{t_w}$$

$$\lambda_p = \frac{\frac{h_c}{t_w} \cdot \sqrt{\frac{E}{F_y}}}{(0,54 \cdot \frac{M_{pl}}{M_r} - 0,09)^2} \leq \lambda_r$$

$$\lambda_r = 5,70 \cdot \sqrt{\frac{E}{f_y}}$$

$$M_{Rd} = \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda \leq \lambda_p$$

$$M_{Rd} = \frac{1}{\gamma_{a1}} \cdot \left[M_{pl} - (M_{pl} - M_r) \cdot \frac{\lambda - \lambda_p}{\lambda_r - \lambda_p} \right] \qquad \text{se } \lambda_p < \lambda \leq \lambda_r$$

Anexo H: NBR8800:2008 $\qquad$ se $\lambda > \lambda_r$

Perfil		P1	P2
Parâmetro de esbeltez de escoamento	λr:	137	137
Parâmetro de esbeltez de plastificação	λp:	78	78
Parâmetro de esbeltez	λ:	106	106
Momento de início de escoamento	Mr (kNm):	5346	5346
Momento de plastificação	Mpl (kNm):	6256	6256
Momento resistente	Mrd (kNm):	5688	5688

B) ELU: FLAMBAGEM LATERAL POR TORÇÃO (FLT) - NBR8800 G2.1:

Fator modificação diagrama momento	Cb:	1
Comprimento destravado	Lb (mm):	6400
Módulo de elasticidade do aço	E (MPa):	200000
Tensão de escomento do aço	fy (MPa):	345
Módulo de elasticidade transversal do aço	G (MPa):	7,7E+04
Coef.ponderador da resistencia do aço	γa1:	1,1

Perfil		P1	P2
Inércia Tcomprimido/Ttracionado	Iyc/Iyt:	0,85	0,85
Coeficientes β	B1:	0,0049	0,0049
Coeficientes β	B2:	-0,2	-0,2
Coeficientes β	B3:	-47	-47
Constante de empenamento transversal	Cw:	1,8E+14	1,8E+14

Esbeltez de plastificação	λp:	42,4	42,4
Esbeltez de início de escoamento	λr:	113	113
Esbeltez	λ:	60,6	60,6

Perfil		P1	P2
Momento de escoamento	Mr (kNm):	3742	3742
Momento de plastificação	Mpl (kNm):	6256	6256
Momento de flambagem elástica	Mcr (kNm):	12534	12534
Condição:		λp<λ<λr	λp<λ<λr
Momento resistente	**Mrd (kNm):**	**5102**	**5102**

$$M_{Rd} = \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda \le \lambda_p$$

$$M_{Rd} = \frac{C_b}{\gamma_{a1}} \cdot \left[M_{pl} - (M_{pl} - M_r) \cdot \frac{\lambda - \lambda_p}{\lambda_r - \lambda_p} \right] \le \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda_p < \lambda \le \lambda_r$$

$$M_{Rd} = \frac{M_{cr}}{\gamma_{a1}} \le \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda > \lambda_r$$

$$\lambda = \frac{L_b}{r_{yc}}$$

$$\lambda_p = 1,76 \sqrt{\frac{E}{F_y}}$$

$$\lambda_r = \frac{1,38\sqrt{I_y J}}{r_{yc} \cdot J \cdot \beta_1} \cdot \sqrt{\beta_2 + \sqrt{\beta_2^2 + 27 \cdot C_w \cdot \beta_1^2}}$$

$$\beta_1 = \frac{(f_y - \sigma_r) \cdot W_c}{EJ}$$

$$\beta_2 = 5,2 \cdot \beta_1 \cdot \beta_3 + 1$$

$$\beta_3 = 0,45 \cdot (d - \frac{t_{fs} + t_{fi}}{2}) \cdot (\frac{\alpha_y - 1}{\alpha_y + 1})$$

$$\alpha_y = \frac{I_{yc}}{I_{yt}}$$

$$C_w = \frac{(d - \frac{t_{fs} + t_{fi}}{2})^2}{12} \cdot (\frac{t_{fi} \cdot b_{fi}^3 \cdot t_{fs} \cdot b_{fs}^3}{t_{fi} \cdot b_{fi}^3 + t_{fs} \cdot b_{fs}^3})$$

$$M_{cr} = \frac{C_b \cdot \pi^2 \cdot E \cdot I_y}{L_b^2} \cdot \left[\beta_3 + \sqrt{\beta_3^2 + \frac{C_w}{I_y} \cdot (1 + 0,039 \cdot \frac{J \cdot L_b^2}{C_w}} \right]$$

$$M_r = (f_y - \sigma_r) \cdot W_c \leq f_y \cdot W_t$$

$$M_{pl} = Z \cdot f_y$$

C) ELU: FLAMBAGEM LOCAL DA MESA (FLM) - NBR8800 G2.2:

Perfil		P1	P2
Anexo F item F2.2 - 0.35 < kc < 0.76	kc:	0,40	0,40
Esbeltez de plastificação	λp:	9,1	9,1
Esbeltez de início de escoamento	λr:	17	17
Esbeltez	λ:	13,2	13,2

Perfil		P1	P2
Momento de escoamento	Mr (kNm):	3742	3742
Momento de flambagem elástica	Mcr (kNm):	6422	6422
Momento de plastificação	Mpl (kNm):	6256	6256
Momento resistente	Mrd (kNm):	**4558**	**4558**

$$M_{Rd} = \frac{M_{pl}}{\gamma_{a1}} \qquad \text{se } \lambda \leq \lambda_p$$

$$M_{Rd} = \frac{1}{\gamma_{a1}} \cdot \left[M_{pl} - (M_{pl} - M_r) \cdot \frac{\lambda - \lambda_p}{\lambda_r - \lambda_p} \right] \qquad \text{se } \lambda_p < \lambda \leq \lambda_r$$

$$M_{Rd} = \frac{M_{cr}}{\gamma_{a1}} \qquad \text{se } \lambda > \lambda_r$$

$$\lambda = \frac{b}{t}$$

$$\lambda_p = 0,38 \sqrt{\frac{E}{F_y}}$$

$$\lambda_r = 0,95 \cdot \sqrt{\frac{E}{(f_y - \sigma_r)/k_c}}$$

$$k_c = \frac{4}{\sqrt{h/t_w}} \qquad \text{em que } 0,35 \le k_c \le 0,76$$

$$\sigma_r = 0,3 \cdot f_y$$

$$M_{cr} = \frac{0,90 \cdot E \cdot k_c \cdot W_c}{\lambda^2}$$

$$M_r = (f_y - \sigma_r) \cdot W_c$$

$$M_{pl} = Z \cdot f_y$$

D) MOMENTO RESISTENTE DE CÁLCULO (MRD):

Perfil		P1	P2
Momento flambagem local da alma	FLA →	5688	5688
Flambagem lateral com torção	FLT →	5102	5102
Flambagem da mesa	FLM →	4558	4558
Momento resistente de cálculo	**Mrd (kNm):**	**4558**	**4558**

E) VERIFICAÇÃO:

Perfil	P1		P2	
Distância	x = 10 m		x = 4,05 m	
Combinação	COMB1	COMB2	COMB1	COMB2
σs-sd / σs-rd:	0,46	0,39	0,27	0,22
σi-sd / σi-rd:	0,52	0,51	0,33	0,34
Verificação	OK			

A 5.2.4. FLAMBAGEM DA ALMA POR COMPRESSÃO (NBR8800 ITEM 5.7.6)

$$F_{Rd} = \frac{24 \cdot t_w^3}{\gamma_{a1} \cdot h} \cdot \sqrt{E \cdot f_y}$$

Combinação		COMB2	COMB3
Força concentrada resistente	Frd (kN):	140,6	140,6
Força concentrada solicitante	Fsd (kN):	363,2	341,0
Relação força resistente/concentrada	Fsd/Frd:	2,58	2,43
Verificação	**Não ok: colocar enrijecedores nos apoios**		

O dimensionamento dos enrijecedores dos apoios é apresentado no ITEM 9.

A 5.2.5. CISALHAMENTO ALMA (NBR8800 5.4.3.1)

Perfil		P1	P2
Distância entre enrijecedores	a (mm):	0	0
NBR8800 5.4.3.1.1	kv:	5,0	5,0
NBR8800 5.4.3.1.2	Vpl (kN):	3363,8	3363,8

Perfil		P1	P2
Esbeltez de plastificação	λp:	59,2	59,2
Esbeltez de início de escoamento	λr:	73,8	73,8
Esbeltez	λ:	100,7	100,7

Força cortante resistente	V_{rd} (kN):	1311,8	1311,8
Força cortante solicitante	V_{sd} (kN):	363,2	363,2
Relação força solicitante/resistente	V_{sd}/V_{rd}:	0,28	0,28
Verificação		**OK - Vrd > Vsd**	

A 5.2.6. ENRUGAMENTO DA ALMA

A variável ln corresponde ao comprimento da força aplicada, na direção longitudinal da longarina, ou, para a reação de apoio, o comprimento do aparelho de apoio.

$$F_{sd} = 1,15G1 + 1,25G2 + 1,3G3$$

$$F_{Rd} = \begin{cases} \dfrac{0,66 \cdot t_w^2}{\gamma_{a1}} \cdot \left[1 + 3 \cdot \left(\dfrac{l_n}{d} \right) \cdot \left(\dfrac{t_w}{t_f} \right)^{1,5} \right] \cdot \sqrt{\dfrac{E \cdot f_y \cdot t_f}{t_w}} & \text{se } S \geq \dfrac{d}{2} \\ \dfrac{0,33 \cdot t_w^2}{\gamma_{a1}} \cdot \left[1 + 3 \cdot \left(\dfrac{l_n}{d} \right) \cdot \left(\dfrac{t_w}{t_f} \right)^{1,5} \right] \cdot \sqrt{\dfrac{E \cdot f_y \cdot t_f}{t_w}} & \text{se } S < \dfrac{d}{2} \text{ e } \dfrac{l_n}{d} \leq 2 \\ \dfrac{0,33 \cdot t_w^2}{\gamma_{a1}} \cdot \left[1 + \left(\dfrac{4 \cdot l_n}{d} - 0,2 \right) \cdot \left(\dfrac{t_w}{t_f} \right)^{1,5} \right] \cdot \sqrt{\dfrac{E \cdot f_y \cdot t_f}{t_w}} & \text{se } S < \dfrac{d}{2} \text{ e } \dfrac{l_n}{d} > 2 \end{cases}$$

ln (mm):	0

Distância entre o ponto de aplicação da força e a extremidade da viga	S (mm):	400
Altura do perfil	d (mm):	1300
Espessura da mesa carregada	tf (mm):	19
Espessura da alma	tw (mm):	12,5
Coeficiente ponderador	γa1:	1,1

Força resistente	Frd (kN):	480,1
Força solicitante	Fsd (kN):	363,2
Força solicitante/resistente	Fsd/Frd	0,8
Verificação	**OK**	

A 5.2.7. FLAMBAGEM LATERAL DA ALMA

$$\frac{h/t_w}{l/b_f} > 2,3 \text{ ou } 1,7$$

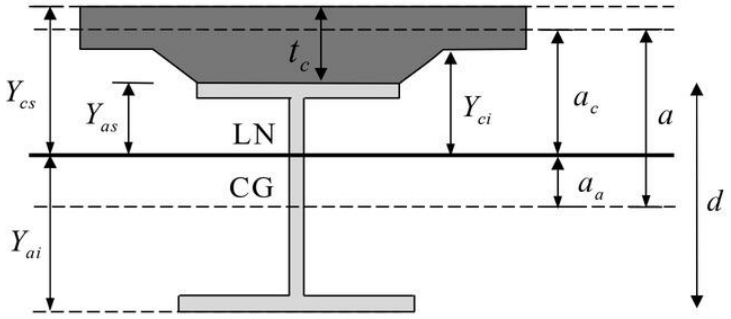
Relação altura/espessura da alma	h/tw:	100,7
Relação comprimento/largura da menor mesa	l/bf:	12,8
Relação	(h/tw) / (l/bf):	7,9
(h/tw) / (l/bf) > 2.3 → Não é necessário verificar a flambagem lateral		

A 6. VERIFICAÇÃO FASE 2: UTILIZAÇÃO

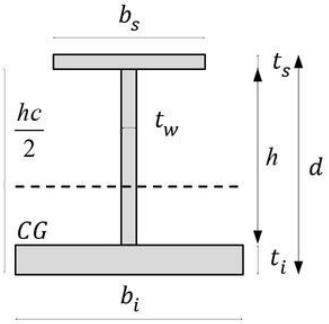
A 6.1. DADOS

Perfil		P1	P2
Classificação da seção do perfil de aço:		Monosimétrico	Mono-simétrico
Esbeltez de plastificação	λp:	90,5	90,5
Esbeltez de escoamento	λr:	137,2	137,2
Relação altura/espessura da alma	h/tw:	100,7	100,7
Verificação		**λp ≤ λ ≤ λr**	**λp ≤ λ ≤ λr**

Vigas mistas semi-compactas - Dimensionamento NBR8800 Anexo O - item O.1.1.2: propriedades elásticas da seção mista



Seção	Mista - CD		Mista - LD		Retração	
Perfil	P1	P2	P1	P2	P1	P2
Coeficientes	η0 :	6,8	ηL :	20	ηr :	18,7
Yai (mm):	1219	1219	1010	1010	1028	1028
aa (mm):	598	598	3,9E+02	389	407	407
Ycs (mm):	361	361	570	570	552	552
Yci (mm):	181	181	290	290	272	272
Yas (mm):	81	81	290	290	272	272
ac (mm):	221	221	430	430	412	412
Asm (mm²):	1,4E+05	1,4E+05	6,9E+04	6,9E+04	7,2E+04	7,2E+04
IxM (mm⁴):	2,9E+10	2,9E+10	2,2E+10	2,2E+10	2,3E+10	2,3E+10
Wcs (mm³):	8,0E+07	8,0E+07	3,9E+07	3,9E+07	4,1E+07	4,1E+07
Wci (mm³):	1,6E+08	1,6E+08	7,7E+07	7,7E+07	8,4E+07	8,4E+07
Was (mm³):	3,6E+08	3,6E+08	7,7E+07	7,7E+07	8,4E+07	8,4E+07
Wai (mm³):	2,4E+07	2,4E+07	2,2E+07	2,2E+07	2,2E+07	2,2E+07



Perfil	P1	P2
d (mm):	1300	1300
A (mm²):	36433	36433
CG (mm):	621	621
Ix (mm4):	1,1E+10	1,1E+10
Ws (mm³):	1,5E+07	1,5E+07
Wi (mm³):	1,7E+07	1,7E+07
h (mm):	1259	1259
tw' (mm):	12,5	12,5
bs (mm):	500	500
bi (mm):	500	500
*ts' (mm):	19	19
*ti' (mm):	22,4	22,4

- Coeficientes ponderadores das ações:

Ações		γ
Peso próprio da estrutura metálica	G1	1,25
Peso próprio da laje de concreto	G2	1,30
Peso próprio das defensas	G3	1,35
Peso próprio da pavimentação	G4	1,35
Peso próprio do recapeamento	G5	1,35
Carga móvel	qM	1,50
Retração	R	1,20
Temperatura	T	1,20
Vento	W	1,20

Coeficiente ponderador ação do vento	ψ0W :	0,6
Coeficiente ponderador variação de temperatura	ψ0T :	0,6
Coeficiente ponderador ação do vento	ψ0M :	0,7
Coeficiente ponderador resistência do aço	γa1:	1,10
Coeficiente poderador resistência do concreto	γc1:	1,40

A 6.2. TENSÃO RESISTENTE

Resistência à compressão do concreto	fck(MPa):	35
Resistência ao escoamento do aço	fy (MPa):	345

(A) TRAÇÃO E COMPRESSÃO NAS MESAS (NBR8800 - 4.8.2):

$$\sigma_{rd} = \frac{f_y}{\gamma_{a1}}$$

B) CISALHAMENTO NA ALMA DO PERFIL (NBR8800 - 5.4.3):

$$\tau_{rd} = \frac{0,6 \cdot f_y}{\gamma_{a1}}$$

C) COMPRESSÃO NA LAJE DE CONCRETO (NBR8800 - O1.1.3):

$$f_{cd} = \frac{0,85 \cdot f_{ck}}{\gamma_c}$$

Verificação	Tensão resistente (MPa)		
	Perfil:	P1	P2
Mesa superior	σrds (MPa):	313,6	313,6
Mesa inferior	σrdi (MPa):	313,6	313,6
Cisalhamento da alma	τrd (MPa):	188,2	188,2
Face superior laje	fcd (MPa):	21,3	21,3

A 6.3. RETRAÇÃO E FLUÊNCIA

O efeito da retração na seção mista foi estimado a partir do modelo de Moga et al (2016). Para fins de comparação, os esforços devido à fluência e retração também foram obtidos via modelo numérico.

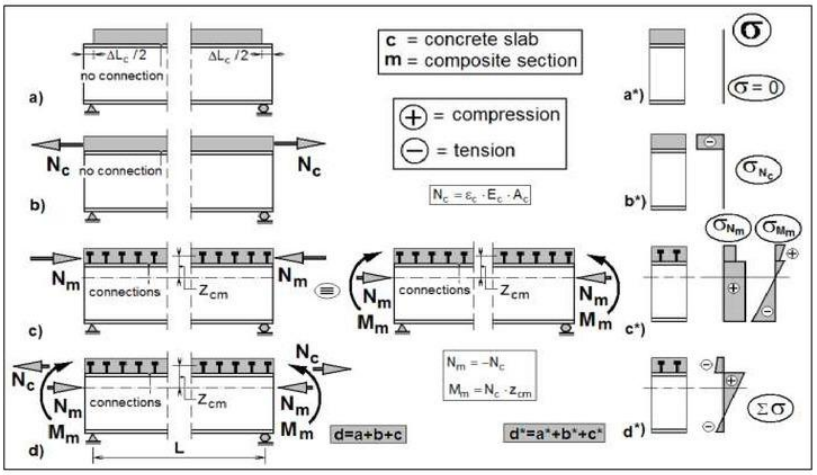


Figura A6.1. Modelo descrito em Moga et al (2016).

Coeficiente de fluência	φr :	3,18
Deformação total de retração	εc :	5,5E-04
Módulo de elasticidade secante aos 28 dias	Ecs (MPa):	29403
Coeficiente de retração EN4 5.4.2.2	ηr :	18,7
Coeficiente de curta duração NBR 8800 O.1.2.1	η0 :	6,8
Esp. camada de concreto adicionada in loco	tcm (mm):	0,180

$$N_c = -N_m = \varepsilon_c \cdot E_{cs} \cdot b_c \cdot t_c \cdot \frac{\eta_0}{\eta_r}$$

$$M_{m(r)i} = N_m \cdot a_{c(r)i}$$

$$\sigma_c = \frac{N_{c(r)i}}{A_{ci}} + \frac{1}{\eta_r} \cdot \left(\frac{N_{m(r)i}}{A_{m(r)i}} + \frac{M_{m(r)i} \cdot z}{I_{m(r)i}}\right)$$

$$\sigma_a = \frac{N_{m(r)i}}{A_{m(r)i}} + \frac{M_{m(r)i} \cdot z}{I_{m(r)i}}$$

ou:

$$\sigma_{cs} = \varepsilon_r \cdot E_a \cdot \frac{A_a}{\eta_r \cdot A_{m(\eta_r)}} \cdot \left[1 - \frac{A_{m(\eta_r)} \cdot a_{a(\eta_r)}}{W^s_{c(\eta_r)}}\right]$$

$$\sigma_{ci} = \varepsilon_r \cdot E_a \cdot \frac{A_a}{\eta_r \cdot A_{m(\eta_r)}} \cdot \left[1 - \frac{A_{m(\eta_r)} \cdot a_{a(\eta_r)}}{W^i_{c(\eta_r)}}\right]$$

$$\sigma_{as} = -\varepsilon_r \cdot E_a \cdot \frac{A_c}{\eta_r \cdot A_{m(\eta_r)}} \cdot \left[1 + \frac{A_{m(\eta_r)} \cdot a_{c(\eta_r)}}{W^s_{a(\eta_r)}}\right]$$

$$\sigma_{ai} = -\varepsilon_r \cdot E_a \cdot \frac{A_c}{\eta_r \cdot A_{m(\eta_r)}} \cdot \left[1 - \frac{A_{m(\eta_r)} \cdot a_{c(\eta_r)}}{W^s_{a(\eta_r)}}\right]$$

Esforços		Analítico		Numérico	Ana/Num
		P1	P2	P1/P2	
Força devido à retração (slab)	Nm (kN):	2528		1145	121%
Momento devido à retração (Beam)	Mm (kNm):	1041	1041	901	16%

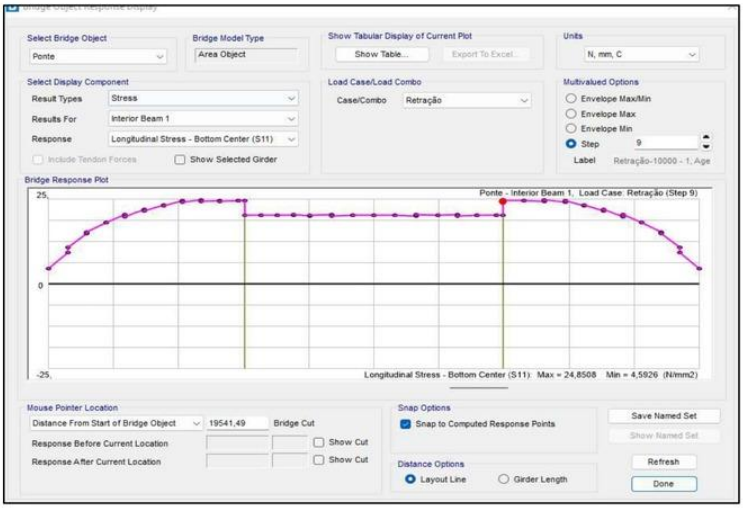


Figura A6.2. Distribuição de tensões numéricas ao longo do comprimento da longarina.

Tensão	Tensão analítica				Tensão numérica			
Perfil	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)	σcs (MPa)	σci (MPa)	σas (MPa)	σai (MPa)
P1	2,64	3,32	-47,3	11,9	0,6	3,0	-77,2	36,8
P2	2,64	3,32	-47,3	11,9	0,6	3,0	-76,6	31,2

Comparação tensões numéricas x analíticas								
Perfil	P1				P2			
Tensões	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)
Analítica	2,6	3,3	-47,3	11,88	2,64	3,32	-47,3	11,9
Numérica	0,6	3,0	-77,2	36,8	0,6	3,0	-76,6	31,2
Ana/Num	77%	10%	-63%	-210%	77%	10%	-62%	-163%

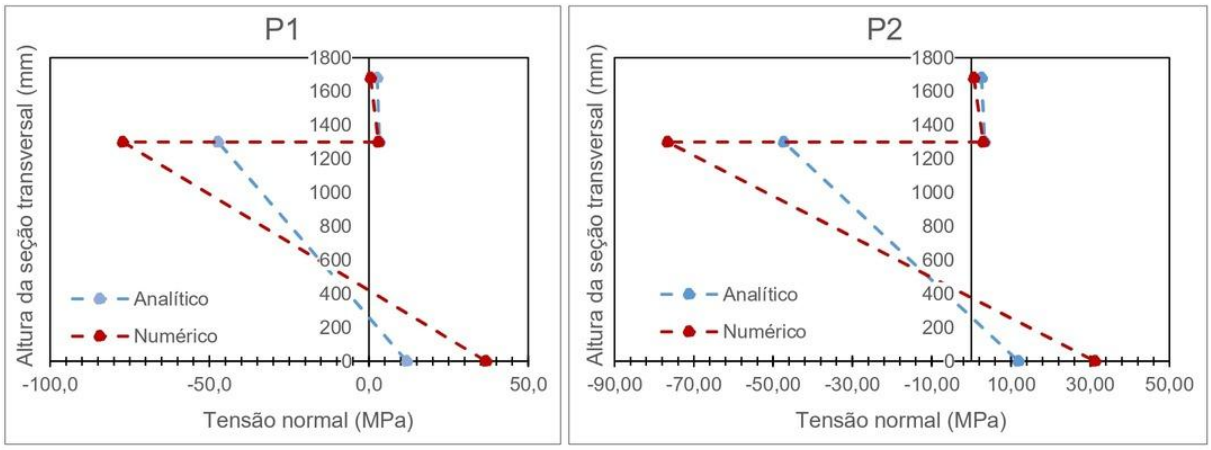


Figura A6.3. Distribuição de tensões numéricas e analíticas na seção transversal da longarina devido à retração.

→ Escolher entre as tensões numéricas ou analíticas:	Tensões numéricas

Tensões de projeto				
Perfil	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)
P1	0,6	3,0	-77,2	36,8
P2	0,6	3,0	-76,6	31,2

A 6.4. VARIAÇÃO DE TEMPERATURA

As tensões analíticas devido à variação de temperatura foram obtidas a partir do modelo de Moga et al (2016).

Encurtamento da laje de concreto	ΔTenc (°C):	-5
Dilatação da laje de concreto	ΔTdil (°C):	12
Coeficiente de dilatação EN1-5 Tab.C.1	α :	0,00001
Coeficiente de curta duração NBR 8800 O.1.2.1	η0 :	6,8

*Calculado em relação às temperatura médias das capitais brasileiras.

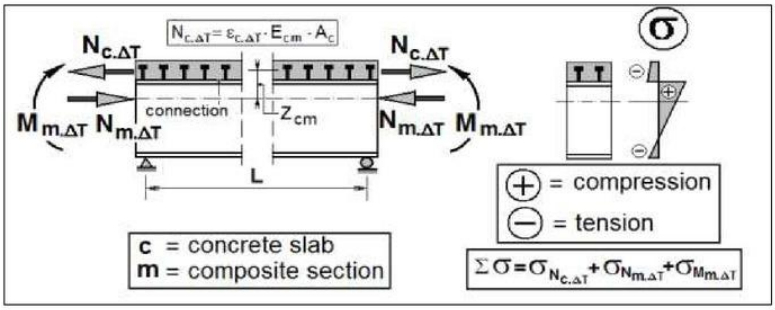


Figura A6.4. Modelo analítico para variação de temperatura proposto por Moga et al (2016).

$$\varepsilon_{c,\Delta t} = \alpha_T \cdot \Delta T_M$$

$$N_{c,\Delta T} = -N_{m,\Delta T} = -\varepsilon_{c,\Delta t} \cdot E_{cm} \cdot A_c$$

$$\sigma_c = \frac{N_{c,\Delta T}}{A_c} + \frac{1}{\eta_0} \cdot \left(\frac{N_{m,\Delta T}}{A_m} + \frac{M_{m,\Delta T} \cdot z}{I_m}\right)$$

$$\sigma_a = \frac{N_{m,\Delta T}}{A_m} + \frac{M_{m,\Delta T} \cdot z}{I_m}$$

Ecm = Ecs

ou:

$$\sigma_{cs} = \pm\alpha_T \cdot \Delta T \cdot E_a \cdot \frac{A_a}{\eta_0 \cdot A_{m(\eta_0)}} \cdot \left[1 - \frac{A_{m(\eta_0)} \cdot a_{a(\eta_0)}}{W^s_{c(\eta_0)}}\right]$$

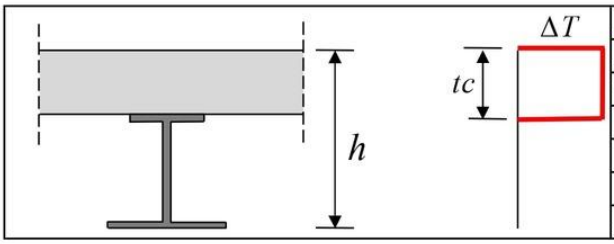
$$\sigma_{ci} = \pm\alpha_T \cdot \Delta T \cdot E_a \cdot \frac{A_a}{\eta_0 \cdot A_{m(\eta_0)}} \cdot \left[1 - \frac{A_{m(\eta_0)} \cdot a_{a(\eta_0)}}{W^i_{c(\eta_0)}}\right]$$

$$\sigma_{as} = \pm\alpha_T \cdot \Delta T \cdot E_a \cdot \frac{A_c}{\eta_0 \cdot A_{m(\eta_0)}} \cdot \left[1 + \frac{A_{m(\eta_0)} \cdot a_{c(\eta_0)}}{W^s_{a(\eta_0)}}\right]$$

$$\sigma_{ai} = \pm\alpha_T \cdot \Delta T \cdot E_a \cdot \frac{A_c}{\eta_0 \cdot A_{m(\eta_0)}} \cdot \left[1 - \frac{A_{m(\eta_0)} \cdot a_{c(\eta_0)}}{W^s_{a(\eta_0)}}\right]$$

Esforços:

Os esforços devido a variação não-uniforme de temperatura foram estimados a partir do modelo de Moga et al (2016) e através do modelo numérico. Os resultados obtidos são comparados em sequência. No modelo numérico foi utilizado o gradiente de temperatura ilustrado abaixo.



ΔTdil (°C):	12,0
ΔTenc (°C):	-5,00
tc (mm):	280
hpre (mm):	100
bf (mm):	2400
Ecs (MPa):	29403
αm:	9,6E-06

Esforços devido à variação de temperatura	Analítico				Numérico	
	P1		P2		P1/P2	
	ΔTenc	ΔTdil	ΔTenc	ΔTdil	ΔTenc	ΔTdil
εcΔt:	-5E-05	1E-04	-5E-05	1E-04	-5E-05	1E-04
Nm,Δt (kN): Beam	-635	1524	-635	1524	-11,3	12,5
Mc,Δt (kNm): Beam	140	-336	140	-336	-4,2	4,7

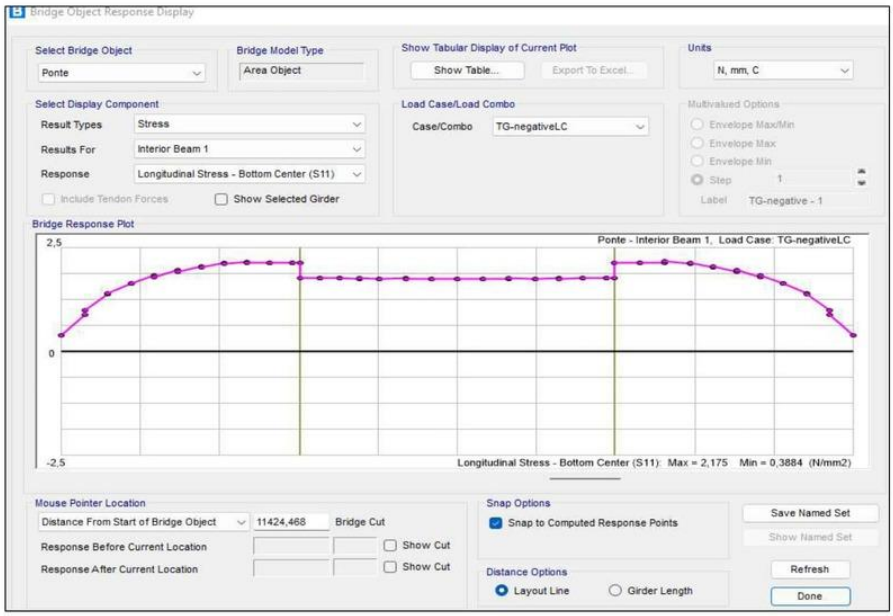


Figura A6.5. Distribuição de tensões numéricas ao longo do comprimento da longarina.

- Comparação entre as tensões numéricas e analíticas:

ΔTenc (°C): TG-negativeLC								
Tensão	Tensão analítica				Tensão numérica			
	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)
P1	-0,3	-0,1	-5,1	1,2	0,2	0,2	0,0	0,6
P2	-0,3	-0,1	-5,1	1,2	0,2	0,2	-0,1	0,4
h (mm):	1680	1300	1300	0	1680	1300	1300	0

ΔTdil (°C): TG-Positive								
Tensão	Tensão analítica				Tensão numérica			
	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)
P1	0,6	0,3	12,2	-2,9	-0,1	0,0	18,6	-7,4
P2	0,6	0,3	12,2	-2,9	-0,1	0,0	18,6	-7,4

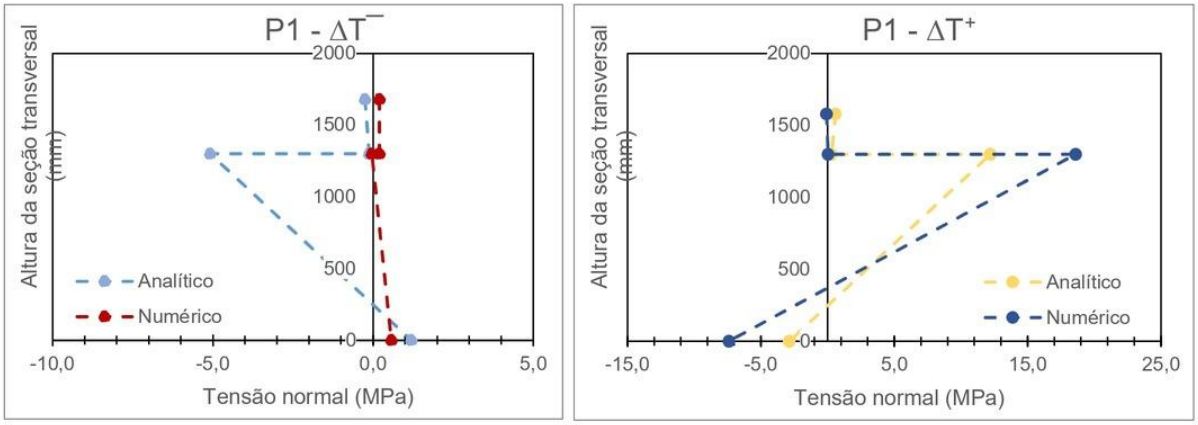


Figura A6.6. Distribuição de tensões numéricas e analíticas na seção transversal da longarina devido à variação de temperatura no P1.

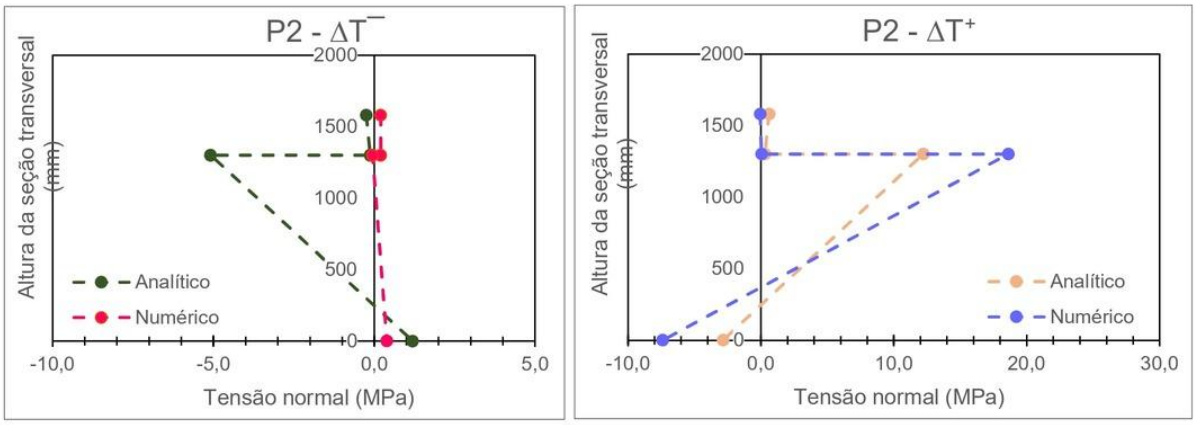


Figura A6.7. Distribuição de tensões numéricas e analíticas na seção transversal da longarina devido à variação de temperatura no P2.

→ Escolher entre as tensões numéricas ou analíticas:	Tensões numéricas

Tensões de projeto					
Perfil	(°C)	σsc (MPa)	σci (MPa)	σas (MPa)	σai (MPa)
P1	ΔTenc	0,2	0,2	0,0	0,6
	ΔTdil	-0,1	0,0	18,6	-7,4
P2	ΔTenc	0,2	0,2	-0,1	0,4
	ΔTdil	-0,1	0,0	18,6	-7,4

A 6.5. VENTO

Após a cura do concreto, a laje de concreto trava contínuamente o flange superior da longarina. Por outro lado, o flange inferior da longarina permanece destravado podendo sofrer flexão devido ao vento. Assim, as tensão de compressão e tração no flange inferior são dadas por:

Comprimento destravado	Lb (mm):	6400
Força por comprimento devido ao vento	Fw (kN/m):	3,3
Momento fletor em torno de y devido ao vento	Mw (kNm/m):	13,6

Perfil		P1	P2
Módulo elástico mesa inferior	Wbi (mm³):	9,3E+05	9,3E+05
Tensão na mesa inferior	σfiw (MPa):	14,5	14,5

$$F_w(kN/m) = F_{an} \cdot \frac{d}{2}$$

$$M_w(kN/m) = \frac{F_w \cdot L_b^2}{10}$$

$$\sigma_{fiW} = \pm \frac{M_w}{W_{bi}}$$

$$W_{bi} = \frac{t_i \cdot b_i^2}{6}$$

A 6.6. TENSÕES SOLICITANTES

$$\tau = \frac{V}{h \cdot t_w}$$

$$\sigma_{as} = \begin{cases} \dfrac{M_{G1/G2}}{W_s} \\ \dfrac{M_{G3/G4/G5}}{W_{aiL}} \\ \dfrac{M_{qM}}{W_{as0}} \end{cases}$$

$$\sigma_{cs} = \begin{cases} \dfrac{M_{G3/G4/G5}}{\eta_L \cdot W_{csL}} \\ \dfrac{M_{qM}}{\eta_L \cdot W_{csL}} \end{cases}$$

$$\sigma_{ai} = \begin{cases} \dfrac{M_{G1/G2}}{W_i} \\ \dfrac{M_{G3/G4/G5}}{W_{aiL}} \\ \dfrac{M_{qM}}{W_{ai0}} \end{cases}$$

$$\sigma_{ci} = \begin{cases} \dfrac{M_{G3/G4/G5}}{\eta_L \cdot W_{ciL}} \\ \dfrac{M_{qM}}{\eta_L \cdot W_{ciL}} \end{cases}$$

Perfil	Perfil 2				Perfil 1	
Posição	x = 0m		x = 4,05 m		x = 10 m	
Ações	Qs0 (kN)	τas0 (MPa)	Qs1 (kN)	τas1 (MPa)	Qs2 (kN)	τas2 (MPa)
G1	35	2,2	20	1,3	0	3,1E-13
G2	199	12,6	125	8,0	6	0,0
G3	41	2,6	19	1,2	1	0,0
G4	65	4,1	40	2,5	2	0,0
G5	54	3,5	33	2,1	2	0,0
qM	508	32,3	349	22,2	200	12,7
R	129	8,2	29	1,8	1	0,0
T	3	0,2	1	0,1	0	0,0

Perfil 2					
x = 4,05 m					
Ações	Ms1 (kNm)	σsc (MPa)	σic (MPa)	σas (MPa)	σai (MPa)
G1	107	-	-	-6,9	6,3
G2	654	-	-	-42,2	38,7
G3	115	-0,1	-0,1	-1,5	5,2
G4	212	-0,3	-0,1	-2,8	9,6
G5	176	-0,2	-0,1	-2,3	8,0
qM	1445	-2,6	-1,3	-4,0	60,8
R	-	0,0	0,0	-76,6	31,2
T	-	-0,1	0,0	-0,1	0,4
W	-	-	-	-	14,5

Perfil 1					
x = 10 m					
Ações	Ms2 (kNm)	σsc (MPa)	σic (MPa)	σas (MPa)	σai (MPa)
G1	167	-	-	-10,8	9,9
G2	1014	-	-	-65,4	59,9
G3	164	-0,2	-0,1	-2,1	7,4
G4	327	-0,4	-0,2	-4,2	14,8
G5	272	-0,3	-0,2	-3,5	12,3
qM	2137	-3,9	-2,0	-5,9	89,9
R	-170	0,0	0,0	-77,2	36,8
T	-	-0,1	0,0	0,0	0,6
W	-	-	-	-	14,5

A 6.7. TENSÃO SOLICITANTE DE CÁLCULO

COMB4:	1.25G1 + 1.3G2 + 1.35G3 + 1.35(G4+G5) + 1.2R + 1.5qM · CI + 0.6 · 1.2T + 0.6 · 1.2W
COMB5:	1.25G1 + 1.3G2 + 1.35G3 + 1.35(G4+G5) + 1.2R + 0.7 · 1.5qM · CI + 1.2T + 0.6 · 1.2W
COMB6:	1.25G1 + 1.3G2 + 1.35G3 + 1.35(G4+G5) + 1.2R + 0.7 · 1.5qM · CI + 0.6 · 1.2T + 1.2W

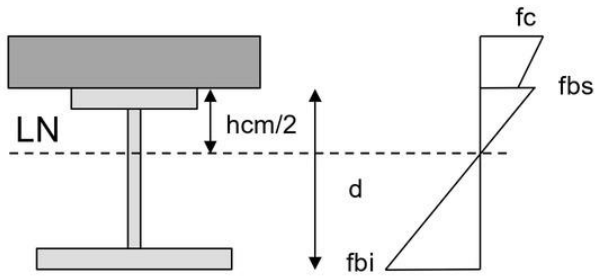


Figura A6.8. Distribuição de tensões na seção transversal mista.

Combinação	COMB4		COMB5		COMB6	
x (m):	10	4,05	10	4,05	10	4,05
Perfil:	P1	P2	P1	P2	P1	P2
σcs-sd (MPa):	-7,2	-4,9	-5,5	-3,7	-5,4	-3,7
σci-sd (MPa):	-3,6	-2,4	-2,7	-1,8	-2,7	-1,8
σas-sd (MPa):	-213,4	-170,4	-210,8	-168,6	-210,7	-168,6
σai-sd (MPa):	326,6	228,3	286,4	201,1	293,1	207,9
τsd (MPa):	19,1	91,3	13,4	76,8	13,4	76,8

* A tensão cisalhante devido à variação de temperatura foi desconsiderada nos cálculos.

Face superior da laje de concreto:

$$\sigma_{cs-sd} = 1.3 \cdot \frac{M_{G3}}{\eta_L \cdot W_{csL}} + 1.35 \cdot \frac{M_{G4} + M_{G5}}{\eta_L \cdot W_{csL}} + 1.2 \cdot \sigma_{csR} + 1.5 \cdot \frac{M_{qM}}{\eta_0 \cdot W_{cs0}} + 0,6 \cdot 1.2 \cdot \sigma_{csT}$$

Face inferior da laje de concreto:

$$\sigma_{ci-sd} = 1.3 \cdot \frac{M_{G3}}{\eta_L \cdot W_{ciL}} + 1.35 \cdot \frac{M_{G4} + M_{G5}}{\eta_L \cdot W_{ciL}} + 1.2 \cdot \sigma_{ciR} + 1.5 \cdot \frac{M_{qM}}{\eta_0 \cdot W_{ci0}} + 0,6 \cdot 1.2 \cdot \sigma_{ciT}$$

Mesa superior do perfil:

$$\sigma_{as-sd} = 1.25 \cdot \frac{M_{G1}}{W_s} + 1.35 \cdot \frac{M_{G2}}{W_s} + 1.3 \cdot \frac{M_{G3}}{W_{asL}} + 1.35 \cdot \frac{M_{G4} + M_{G5}}{W_{asL}} + 1.2 \cdot \sigma_{asR} + 1.5 \cdot \frac{M_{qM}}{W_{as0}} + 0,6 \cdot 1.2 \cdot \sigma_{asT}$$

Mesa inferior do perfil:

$$\sigma_{ai-sd} = 1.25 \cdot \frac{M_{G1}}{W_i} + 1.35 \cdot \frac{M_{G2}}{W_i} + 1.3 \cdot \frac{M_{G3}}{W_{aiL}} + 1.35 \cdot \frac{M_{G4} + M_{G5}}{W_{aiL}} + 1.2 \cdot \sigma_{aiR} + 1.5 \cdot \frac{M_{qM}}{W_{ai0}} + 0,6 \cdot 1.2 \cdot \sigma_{aiT} + 0,6 \cdot 1.2 \cdot \sigma_{aiW}$$

Seção S0: Perfil 2

$$\tau_{sd} = \frac{1}{h \cdot t_w} \cdot (1.25 \cdot V_{G1} + 1.35 \cdot V_{G2} + 1.3 \cdot V_{G3} + 1.35 \cdot V_{G4} + 1.35 \cdot V_{G5} + 1.5 \cdot V_{qM} + 1.2 \cdot V_R)$$

A 6.8. VERIFICAÇÃO

Combinação	COMB4		COMB5		COMB6	
Perfil	P1	P2	P1	P2	P1	P2
Comparação	SD/RD	SD/RD	SD/RD	SD/RD	SD/RD	SD/RD
σcs-sd (MPa):	0,34	0,23	0,26	0,17	0,26	0,17
σci-sd (MPa):	0,17	0,11	0,13	0,09	0,13	0,09
σas-sd (MPa):	0,68	0,54	0,67	0,54	0,67	0,54
σai-sd (MPa):	**1,04**	0,73	0,91	0,64	0,93	0,66
τsd (MPa):	0,10	0,49	0,07	0,41	0,07	0,41

Verificação		
σcs-sd (MPa):	Face superior da laje de concreto	**OK**
σci-sd (MPa):	Face inferior da laje de concreto	**OK**
σas-sd (MPa):	Mesa superior do perfil	**OK**
σai-sd (MPa):	Mesa inferior do perfil	
τsd (MPa):	Seção S0: Perfil 2	**OK**

APÊNDICE A

Códigos elaborados para a conexão API com o CSiBridge via Python

O apêndice A contém todos os códigos elaborados para a automatização da modelagem numérica e extração de esforços realizadas, conforme as análises propostas.

```
        # ESSE PROGRAMA IRÁ RELIZAR A ANÁLISE DA SUPERESTRUTURA DA MESOESTRUTURA DA PONTE CONFORME AS VARIÁVEIS TANTO INPUTADAS NA PLANILHA precast-variables.xlsx QUANTO NA SEÇÃO DE INPUT DE VARIÁVEIS.
        # ELE TAMBÉM ALIMENTA AS PLANILHAS DE DIMENSIONAMENTO, A SABER:
            # slab-preslab-template.xlsx
            # column-template.xlsx
            # shrinkage-creep-template.xlsx
            # girder-template.xlsm
        # CASO A ANÁLISE SEJA DE SUPERESTRUTURA E A ANÁLISE DE FLECHAS SEJA RELEVANTE, RODAR O OUTRO CÓDIGO ('precast-diplacements'.py) DEPOIS DE REALIZAR OS AJUSTES DA PROTENSÃO NA PLANILHA DE DIMENSIONAMENTO DAS LONGARINAS
        # ALÉM DISSO, ELE FAZ TODA A DEFINIÇÃO E ATRIBUIÇÃO DOS ELEMENTOS DE TODA A ESTRUTURA DA PONTE
        # INPUTAR AS TIPOLOGIAS A SEREM RODADAS NO VETOR TIPOLOGIAS
        # CASO A ANÁLISE SEJA DE MESOESTRUTURA, INPUTAR AS ALTURAS DOS PILARES CENTRAIS A SEREM VARIADAS (CASO DA ANÁLISE COM VÃOS IGUAIS. OS VÃOS DIFERENTES SÃO ANALISADOS COM OUTRO CÓDIGO)
        # CASO A ANÁLISE SEJA DE SUPERESTRUTURA, NÃO SERÁ REALIZADO O TRANSLADO DA PONTE PARA QUE SEJA POSSÍVEL A PASSAGEM DO TREM TIPO NA LAJE DE TRANSIÇÃO. ISSO SÓ SERÁ FEITO PARA A ANÁLISE DOS ENCONTROS

        import os
        import sys
        import clr
        import pandas as pd
        from enum import Enum
        import numpy as np
        import xlwings as xw

        clr.AddReference("System.Runtime.InteropServices")
        clr.AddReference(r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge1.dll') #set the following path to the installed CSiBridge program directory
        from System.Runtime.InteropServices import Marshal
        from CSiBridge1 import *

        #set the following flag to True to execute on a remote computer

        Remote = False

        #if the above flag is True, set the following variable to the hostname of the remote computer

        #remember that the remote computer must have CSiBridge installed and be running the CSiAPIService.exe

        RemoteComputer = "SpareComputer-DT"

        #set the following flag to True to attach to an existing instance of the program

        #otherwise a new instance of the program will be started

        AttachToInstance = True

        #set the following flag to True to manually specify the path to CSiBridge.exe

        #this allows for a connection to a version of CSiBridge other than the latest installation

        #otherwise the latest installed version of CSiBridge will be launched

        SpecifyPath = True

        #if the above flag is set to True, specify the path to CSiBridge below

        ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge.exe'

        #full path to the model

        #set it to the desired path of your model

        #if executing remotely, ensure that this folder already exists on the remote computer

        #the below command will only create the folder locally

        main_folder = os.path.dirname(os.path.abspath(__file__))
        APIPath = main_folder
```

```
        if not os.path.exists(APIPath):

            try:

                os.makedirs(APIPath)

            except OSError:

                pass

        def chunk_list(input_list, chunk_size):
        # Create a list of lists with fixed size from another list
            return [input_list[i:i + chunk_size] for i in range(0, len(input_list),
chunk_size)]

        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'precast-variables.xlsx')
        variables = pd.read_excel(variables_data_file, sheet_name="Main", header=2,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-PPC-A-25-P']

        # ['S-PPC-A-20-P', 'S-PPC-A-25-P', 'S-PPC-A-30-P', 'S-PPC-A-35-P', 'S-PPC-A-20-S', 'S-
PPC-A-25-S', 'S-PPC-A-30-S', 'S-PPC-A-35-S', 'S-PPC-B-20-P', 'S-PPC-B-25-P',
        # 'S-PPC-B-30-P', 'S-PPC-B-35-P', 'S-PPC-B-20-S', 'S-PPC-B-25-S', 'S-PPC-B-30-S', 'S-
PPC-B-35-S']

        Tipo_analise = 'MESOESTRUTURA' # Tipos contemplados: SUPERESTRUTURA, ENCONTROS,
MESOESTRUTURA.
        # CASO SEJA ANÁLISE DE MESOESTRUTURA, ALIMENTAR O VETOR DE ALTURA DOS PILARES NA SEÇÃO
DE INPUTS

        for Tipologia in Tipologias: # Rodar todo o código com base em um elemento da lista
Tipologias
            xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
            GeneralPath = os.path.join(main_folder, Tipologia)
            os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à tipologia. O
parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
            ModelPath = os.path.join(GeneralPath, f'{Tipologia} ({Tipo_analise}).bdb') #
Caminho do modelo .bdb

            #create API helper object

            helper = cHelper(Helper())

            if AttachToInstance:

                #attach to a running instance of CSiBridge

                try:

                    #get the active CSiBridge object

                    if Remote:

                        mySAPObject = cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                    else:

                        mySAPObject = cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                except:

                    print("No running instance of the program found or failed to attach.")

                    sys.exit(-1)

            else:

                if SpecifyPath:

                    try:
```

```
                        #'create an instance of the CSiBridge object from the specified path

                        if Remote:

                            mySAPObject     =     cOAPI(helper.CreateObjectHost(RemoteComputer,
ProgramPath))

                        else:

                            mySAPObject = cOAPI(helper.CreateObject(ProgramPath))

                    except :

                        print("Cannot start a new instance of the program from " + ProgramPath)

                        sys.exit(-1)

                else:


                    try:

                        #create an instance of the CSiBridge object from the latest installed
CSiBridge

                        if Remote:

                            mySAPObject  =  cOAPI(helper.CreateObjectProgIDHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                        else:

                            mySAPObject                                                    =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))


                    except:

                        print("Cannot start a new instance of the program.")

                        sys.exit(-1)


                #start CSiBridge application
                mySAPObject.ApplicationStart()

            #*********************************      INÍCIO      VARIÁVEIS      DE      INPUT
*******************************************#
            f_ck = 35
            Alturas_pilarescentrais = [400, 600, 800] # Análises de MESOESTRUTURA
            NVãos = 1 if Tipo_analise == 'SUPERESTRUTURA' else 3 if Tipo_analise == 'ENCONTROS'
else 4
            df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia

            # Dimensões da longarina

            b_long = df_ponte.loc[:, 'B1': 'B4'].values.flatten().tolist() #B1 a B4 conforme
caixa de diálogo de seção pré-moldada do tipo I
            d_long = df_ponte.loc[:, 'D1': 'D7'].values.flatten().tolist() #D1 a D7
            t_long = df_ponte.loc[:, 'T1': 'T2'].values.flatten().tolist() #T1 a T2
            c1_long = 0 #Bottom flange chamfer dimension
            h_long = d_long[0]
            OverhangLength  =  df_ponte['Balanço  (cm)'].iloc[0]  #balanço  laje/longarina  #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
            NLongarinasPorVão = int(df_ponte['Nlong'].iloc[0])
            NumberInteriorGirders = NLongarinasPorVão - 2 #The number of interior girders
            TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
            Espaçamento_Long             =             (TotalWidthOfDeckSection             -
2*OverhangLength)/(NLongarinasPorVão-1)
```

```
            # Diâmetro do pilar
            t3_pilar = df_ponte['Dp'].iloc[0]

            # Travessas e transversinas

            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]

            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

            h_transv = df_ponte['hD'].iloc[0]
            bw_transv = df_ponte['bwD'].iloc[0]

            # Tabuleiro

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - b_long[0]

            # Geometria geral ponte

            CVão = df_ponte['Vão (cm)'].iloc[0]
            SpanLength = [CVão] * NVãos #A zero-based array containing the length of each span.
[L]
            CTotal = sum(SpanLength)
            SpanName = []
            for i in range (NVãos):
                SpanName.append(f'Vão {i+1}')
            NDiafragmasPorVão = int(df_ponte['Ndiaf'].iloc[0])
            DiaphragmDistance = [15, 3/8*CVão, 6/8*CVão, CVão-15] #The distance from the start
of the specified span to the diaphragm. [L]
            LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)


            # Aparelho de apoio
            a_ap = df_ponte['aA'].iloc[0]
            b_ap = df_ponte['aA'].iloc[0]
            G_ap = 0.1 # Rigidez do aparelho de apoio em kN/cm² (1 MPa)
            h_ap = df_ponte['hA'].iloc[0] # Altura total do aparelho de apoio
            hd_ap = h_ap - 2*(2.5/10) - 25/10 # Altura de cálculo da rigidez do aparelho de
apoio, subtraindo-se o cobrimento vertical (2,5 mm) nas duas faces e supondo 25 mm em todas as
chapas de fretagem
            d_ap = df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta
            K_ap = 100
            # K_ap = a_ap * b_ap * G_ap / hd_ap

            # Bents

            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = 300 if Tipo_analise == 'ENCONTROS' else df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                Distance_ext = [Distextp, C_travessaext - Distextp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, C_travessaext/2, C_travessaext - Distextp]
                Height_ext = [Hextp, Hextp, Hextp]

            C_travessaint = df_ponte['Cint'].iloc[0]
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
```

```
            SupportType_int = 1
            if N_pilaresint == 2:
                Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # Laje de Transição
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

            # Muro
            AlturaM = df_ponte['AlturaM'].iloc[0]

            # Ala
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

            #**********************************       FIM       VARIÁVEIS       DE       INPUT
*****************************************#

            # 1 Definição dos objetos primitivos
            SapModel = cSapModel(mySAPObject.SapModel)
            MyBridgeModeler = cBridgeModeler_1(SapModel.BridgeModeler_1) # necessário para
acessar os métodos do CSiBridge
            ret = SapModel.File.OpenFile(os.path.join(main_folder, 'precast-template.bdb'))
            if Tipo_analise != 'MESOESTRUTURA':
                ret = SapModel.File.Save(os.path.join(main_folder, Tipologia, f'{Tipologia}
({Tipo_analise}).bdb'))

            kN_cm_C = 13
            kN_m_C = 6
            kN_mm_C = 5
            ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

            # 2 Propriedades dos materiais

            # 2.1 Definições iniciais

            PropMaterial = cPropMaterial(SapModel.PropMaterial)

            #Criação de materiais do tipo concreto e tendon
            MATERIAL_CONCRETE = 2
            MATERIAL_TENDON = 7
            ret = SapModel.PropMaterial.ChangeName('Concreto - Longarina', f'C{f_ck} -
Longarina')
            ret = SapModel.PropMaterial.ChangeName('Concreto - Mesoestrutura', f'C{f_ck} -
Mesoestrutura')
            ret = SapModel.PropMaterial.ChangeName('Concreto - Tabuleiro', f'C{f_ck} -
Tabuleiro')
            ret = SapModel.PropMaterial.SetMaterial('CP190RB', eMatType(MATERIAL_TENDON))

            # 2.2 Atribuição das propriedades mecânicas elásticas aos materiais
            E = (5600*f_ck**(1/2))/10
            E_p = 20000
            v = 0.2
            v_p = 0.3
            alfa_T = 10**(-5)
            T_ambiente = 25
            ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Longarina', E, v, 1*10**-5,
T_ambiente)
            ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Mesoestrutura', E, v,
1*10**-5, T_ambiente)
            ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Tabuleiro', E, v, 1*10**-5,
T_ambiente)
            ret = SapModel.PropMaterial.SetMPIsotropic('CP190RB', E_p, v_p, 1*10**-5,
T_ambiente)
```

```
            # 2.3 Atribuição do peso específico aos materiais
            γ_concreto = 25/(100**3)
            γ_aco = 78.5/(100**3)
            ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Longarina', 1, γ_concreto,
T_ambiente)
            ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Mesoestrutura', 1,
γ_concreto, T_ambiente)
            ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Tabuleiro', 1, γ_concreto,
T_ambiente)
            ret = SapModel.PropMaterial.SetWeightAndMass('CP190RB', 1, γ_aco, T_ambiente)

            # 2.4 Atribuição das resistência aos materiais e curvas características de tensão-
deformação
            fptk = 190
            fpyk = 0.90*fptk
            ret = SapModel.PropMaterial.SetOTendon_1('CP190RB', fpyk, fptk, 0, 0, -0.1,
T_ambiente)

            IsLightweight = False
            fcsfactor = 1 #The shear strength reduction factor for lightweight concrete
            SSType = 0
            # This is 0, 1 or 2, indicating the stress-strain curve type.
            # 0 = User defined
            # 1 = Parametric - Simple
            # 2 = Parametric - Mander
            SSHysType = 0
            # This is 0, 1 or 2, indicating the stress-strain hysteresis type.
            # 0 = Elastic
            # 1 = Kinematic
            # 2 = Takeda
            StrainAtfc = 0.002#This item applies only to parametric stress-strain curves. It
is the strain at the unconfined compressive strength.
            StrainUltimate = 0.0035 #This item applies only to parametric stress-strain curves.
It is the ultimate unconfined strain capacity. This item must be larger than the StrainAtfc
item.
            FinalSlope = 1 #This item applies only to parametric stress-strain curves. It is a
multiplier on the material modulus of elasticity, E. This value multiplied times E gives the
final slope on the compression side of the curve.
            ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Longarina', f_ck/10,
IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)
            ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Mesoestrutura', f_ck/10,
IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)
            ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Tabuleiro', f_ck/10,
IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)




            # 3 Definição das propriedades das seções

            # 3.1 Frame Sections

            # 3.1.1 Longarinas
            ret = SapModel.PropFrame.SetPrecastI_1(f'Longarina (C{f_ck})', f'C{f_ck} -
Longarina', b_long, d_long, t_long, c1_long)

            # 3.1.2 Pilares
            Name = f'Pilar (C{f_ck})'
            MatProp = f'C{f_ck} - Mesoestrutura'
            Color = -1
            Notes = ''
            GUID = 'Default'
            ret = SapModel.PropFrame.SetCircle(Name, MatProp, t3_pilar, Color, Notes, GUID) #
definição da seção

            Name = f'Pilar (C{f_ck})' # The name of an existing frame section property.
            MatPropLong = 'A615Gr60' # The name of the rebar material property for the
longitudinal rebar.
            MatPropConfine = 'A615Gr60' # The name of the rebar material property for the
confinement rebar.
            Pattern = 2
            # This is either 1 or 2, indicating the rebar configuration.
            # 1 = Rectangular
            # 2 = Circular
```

```
            # For circular frame section properties this item must be 2; otherwise an error is
returned.
            ConfineType = 1
            # This is either 1 or 2, indicating the confinement bar type.
            # 1 = Ties
            # 2 = Spiral
            # This item applies only when Pattern = 2. If Pattern = 1, the confinement bar type
is assumed to be ties.
            Cover = 4 # The clear cover for the confinement steel (ties). In the special case
of circular reinforcement in a rectangular column, this is the minimum clear cover. [L]
            NumberCBars = 14 # This item applies to a circular rebar configuration, Pattern =
2. It is the total number of longitudinal reinforcing bars in the column.
            NumberR3Bars = int() # This item applies to a rectangular rebar configuration,
Pattern = 1. It is the number of longitudinal bars (including the corner bar) on each face of
the column that is parallel to the local 3-axis of the column.
            NumberR2Bars = int() # This item applies to a rectangular rebar configuration,
Pattern = 1. It is the number of longitudinal bars (including the corner bar) on each face of
the column that is parallel to the local 2-axis of the column.
            RebarSize = '25d' # The rebar name for the longitudinal rebar in the column.
            TieSize = '8d' # The rebar name for the confinement rebar in the column.
            TieSpacingLongit = 20 # The longitudinal spacing of the confinement bars (ties).
[L]
            Number2DirTieBars = int() # This item applies to a rectangular reinforcing
configuration, Pattern = 1. It is the number of confinement bars (tie legs) running in the local
2-axis direction of the column.
            Number3DirTieBars = int() # This item applies to a rectangular reinforcing
configuration, Pattern = 1. It is the number of confinement bars (tie legs) running in the local
3-axis direction of the column.
            ToBeDesigned = False # If this item is True, the column longitudinal rebar is to
be designed; otherwise it is to be checked.
            ret = SapModel.PropFrame.SetRebarColumn(Name, MatPropLong, MatPropConfine,
Pattern, ConfineType, Cover, NumberCBars, NumberR3Bars, NumberR2Bars, RebarSize, TieSize,
            TieSpacingLongit, Number2DirTieBars, Number3DirTieBars, ToBeDesigned)

            # 3.1.3 Travessas
            ret = SapModel.PropFrame.SetRectangle(f'Travessa encontro (C{f_ck})', f'C{f_ck} -
Mesoestrutura', h_travessaext, bw_travessaext)

            ret = SapModel.PropFrame.SetRectangle(f'Travessa intermediária (C{f_ck})',
f'C{f_ck} - Mesoestrutura', h_travessaint, bw_travessaint)

            # 3.1.4 Transversinas
            ret = SapModel.PropFrame.SetRectangle(f'Transversina de apoio (C{f_ck})',
'A709Gr50', h_transv, bw_transv) # Foi colocado o aço aqui por limitação da API por não permitir
a modelagem da transversina como barra em seções de concreto
            # Porém, o código atribui ao diafragma o material de aço e, na definição da
transversina, muda o material para o correto

            # 3.2 Area Sections

            Name = 'Encontro'
            ShellType = 1
            IncludeDrillingDOF = True
            MatProp = f'C{f_ck} - Mesoestrutura'
            MatAng = 0
            Thickness = hAla
            Bending = hAla
            Color = -1
            Notes = ''
            GUID = 'Default'
            ret = SapModel.PropArea.SetShell_1(Name, ShellType, IncludeDrillingDOF, MatProp,
MatAng, Thickness, Bending, Color, Notes, GUID)

            Name = 'Laje de transição'
            ShellType = 1
            IncludeDrillingDOF = True
            MatProp = f'C{f_ck} - Tabuleiro'
            MatAng = 0
            Thickness = EspessuraLT
            Bending = EspessuraLT
            Color = -1
            Notes = ''
            GUID = 'Default'
            ret = SapModel.PropArea.SetShell_1(Name, ShellType, IncludeDrillingDOF, MatProp,
MatAng, Thickness, Bending, Color, Notes, GUID)
```

```
            # 3.3 Alimentação dos dados das seções na planilha de dimensionamento

            # 3.3.1 Longarinas
            modelo_long = os.path.join(main_folder, 'girder-template.xlsm')
            wb_long = xw.Book(modelo_long)
            ws_long = wb_long.sheets["2 Seção transversal"]

            Name = f'Longarina (C{f_ck})' # The name of an existing precast concrete I girder
frame section property.
            FileName = '' # If the section property was imported from a property file, this is
the name of that file. If the section property was not imported, this item is blank.
            MatProp = '' # The name of the material property for the section.
            b = []
            d = []
            t = []
            c = float()
            Color = int() # The display color assigned to the section.
            Notes = '' # The notes, if any, assigned to the section.
            GUID = '' # The GUID (global unique identifier), if any, assigned to the section.
            ret = SapModel.PropFrame.GetPrecastI_1(Name, FileName, MatProp, b, d, t, c, Color,
Notes, GUID)
            Dimensoes_Longarina = []
            Dimensoes_Longarina.extend(list(ret[3]))
            Dimensoes_Longarina.extend(list(ret[4]))
            Dimensoes_Longarina.extend(list(ret[5]))
            Dimensoes_Longarina.append(ret[6])
            for i in range(len(Dimensoes_Longarina)):
                ws_long[f'C{12+i}'].value = Dimensoes_Longarina[i]

            Name = f'Longarina (C{f_ck})' # The name of an existing frame section property.
            Area = float() # The cross-sectional area. [L2]
            As2 = float() # The shear area for forces in the section local 2-axis direction.
[L2]
            As3 = float() # The shear area for forces in the section local 3-axis direction.
[L2]
            Torsion = float() # The torsional constant. [L4]
            I22 = float() # The moment of inertia for bending about the local 2 axis. [L4]
            I33 = float() # The moment of inertia for bending about the local 3 axis. [L4]
            S22 = float() # The section modulus for bending about the local 2 axis. [L3]
            S33 = float() # The section modulus for bending about the local 3 axis. [L3]
            Z22 = float() # The plastic modulus for bending about the local 2 axis. [L3]
            Z33 = float() # The plastic modulus for bending about the local 3 axis. [L3]
            R22 = float() # The radius of gyration about the local 2 axis. [L]
            R33 = float() # The radius of gyration about the local 3 axis. [L]
            ret = SapModel.PropFrame.GetSectProps(Name, Area, As2, As3, Torsion, I22, I33, S22,
S33, Z22, Z33, R22, R33)

            ws_long['D5'].value = CVão/100
            ws_long['D6'].value = Espaçamento_Long/100
            ws_long['D8'].value = h_laje/100
            ws_long['D32'].value = ret[1]/(100**2)
            ws_long['D36'].value = -ret[6]/(100**4)/(ret[8]/(100**3))
            ws_long['D38'].value = ret[6]/(100**4)
            ws_long['D42'].value = ret[5]/(100**4)

            # 3.4 Atribuição das propriedades dependentes do tempo (curvas de retração e
fluência e ganho de resistência/rigidez conforme o CEB FIP 90)

            # 3.4.1 Longarina

            A_c = ws_long['D32'].value #Área da seção de concreto
            U_ar = ws_long['D33'].value #Perímetro da seção em contato com o ar

            # 3.4.1.1 Alimentação da planilha de retração e fluência
            modelo_retração_fluência     =     os.path.join(main_folder,     'shrinkage-creep-
template.xlsx')
            wb_retração_fluência = xw.Book(modelo_retração_fluência)
            ws_retração_fluência = wb_retração_fluência.sheets["Retração na longarina"]
            ws_retração_fluência['D3'].value = A_c*10**4
            ws_retração_fluência['D4'].value = U_ar*100


            # 3.4.3 CP190RB
```

```
            ret = SapModel.PropMaterial.TimeDep.SetTendonCEBFIP90('CP190RB', True, 2, 0, 12,
T_ambiente)

            # 3.4.4 Tabuleiro

            A_c = ws_long['D56'].value #Área da seção de concreto  #the notional size is equal
to two times the cross-sectional area of the member divided by the perimeter of the member in
contact with the atmosphere.
            U_ar = ws_long['D57'].value #Perímetro da seção em contato com o ar

            #  Caso  o  modelo  de  retração  e  fluência  não  for  adequado,  ver  método
PropMaterial.TimeDep.SetConcreteScaleFactors

            # 3.4.4.1 Alimentação da planilha de retração e fluência
            ws_retração_fluência = wb_retração_fluência.sheets["Retração no tabuleiro"]
            ws_retração_fluência['D3'].value = A_c*10**4
            ws_retração_fluência['D4'].value = U_ar*100

            # 3.5 Retração diferencial
            εdif = ws_retração_fluência['N107'].value
            ws_long = wb_long.sheets["6 Retração e temperatura"]
            ws_long['E52'].value = εdif

            # 3.6 Atribuição das curvas de ganho de resistência

            # Dados de ganho de resistência do concreto
            ws_retração_fluência = wb_retração_fluência.sheets["Resistência do concreto"]
            ws_retração_fluência['C4'].value = f_ck
            resistência_fator = list(ws_retração_fluência['L4:L1014'].value)
            resistência_dias = list(ws_retração_fluência['K4:K1014'].value)

            # Atribuição da curva de resistência do concreto aos materiais da longarina,
tabuleiro e mesoestrutura

            TableKey = 'Material Properties 05d - Time Dependence - User Stiffness'
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = []
            NumberRecords = int()
            TableData = []
            ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,    GroupName,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
            ColumnLabels = list(ret[2])
            Resultados = list(ret[4])

            # Criação do dataframe a partir do dicionário

            resist_dict = {
                           'Material': [],
                           'Age': [],
                           'Multiplier': []
                           }

            Material_Text  =  [f'C{f_ck}  -  Longarina',  f'C{f_ck}  -  Tabuleiro',  f'C{f_ck}  -
Mesoestrutura']
            Age = resistência_dias
            Multiplier = resistência_fator
            for i in range(3): # Foi convertido em string para alimentar as tabelas no programa
                resist_dict['Material'].extend([str(Material_Text[i])] * len(Multiplier))
                resist_dict['Age'].extend(str(element).replace(".", ",") for element in Age)
                resist_dict['Multiplier'].extend(str(element).replace(".", ",") for element in
Multiplier)

            df_concreto = pd.DataFrame(resist_dict)
            result = df_concreto.values.ravel().tolist()

            TableKey = 'Material Properties 05d - Time Dependence - User Stiffness'
            TableVersion = int()
            FieldKeysIncluded = list(resist_dict.keys())
            NumberRecords = int()
            TableData = result
            ret   =   SapModel.DatabaseTables.SetTableForEditingArray(TableKey,   TableVersion,
FieldKeysIncluded, NumberRecords, TableData)
            ret = SapModel.DatabaseTables.ApplyEditedTables(False, int(), int(), int(), int(),
'')
```

```
            # 3.7 Atribuição das curvas de fluência aos materiais da longarina e tabuleiro

            # Dados de fluência na longarina
            ws_retração_fluência = wb_retração_fluência.sheets["Fluência na longarina"]
            deformação_longarina = list(ws_retração_fluência['R4:R109'].value)
            dias_longarina = list(ws_retração_fluência['J4:J109'].value)

            # Dados de fluência no tabuleiro
            ws_retração_fluência = wb_retração_fluência.sheets["Fluência no tabuleiro"]
            deformação_tabuleiro = list(ws_retração_fluência['R4:R109'].value)
            dias_tabuleiro = list(ws_retração_fluência['J4:J109'].value)

            TableKey = 'Material Properties 05e - Time Dependence - User Creep'
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = []
            NumberRecords = int()
            TableData = []
            ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,    GroupName,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
            ColumnLabels = list(ret[2])
            Resultados = list(ret[4])

            # Criação do dataframe a partir do dicionário

            fluência_dict = {
                            'Material': [],
                            'A': [],
                            'B': [],
                            'Ho': [],
                            'AgeAtLoad': [],
                            'Days': [],
                            'CreepCoeff': []
                            }

            Material_Text = [f'C{f_ck} - Longarina', f'C{f_ck} - Tabuleiro']
            Days = [dias_longarina, dias_tabuleiro]
            CreepCoeff = [deformação_longarina, deformação_tabuleiro]
            for i in range(2): # Foi convertido em string para alimentar as tabelas no programa
                fluência_dict['Material'].extend([str(Material_Text[i])]                    *
len(deformação_longarina))
                fluência_dict['A'].extend([str(1)] * len(deformação_longarina))
                fluência_dict['B'].extend([str(1)] * len(deformação_longarina))
                fluência_dict['Ho'].extend([str(10)] * len(deformação_longarina))
                fluência_dict['AgeAtLoad'].extend([str(28)] * len(deformação_longarina))
                fluência_dict['Days'].extend(str(element).replace(".",  ",")  for  element  in
Days[i])
                fluência_dict['CreepCoeff'].extend(str(element).replace(".", ",") for element
in CreepCoeff[i])

            df_concreto = pd.DataFrame(fluência_dict)
            result = df_concreto.values.ravel().tolist()

            TableKey = 'Material Properties 05e - Time Dependence - User Creep'
            TableVersion = int()
            FieldKeysIncluded = list(fluência_dict.keys())
            NumberRecords = int()
            TableData = result
            ret   =   SapModel.DatabaseTables.SetTableForEditingArray(TableKey,   TableVersion,
FieldKeysIncluded, NumberRecords, TableData)
            ret = SapModel.DatabaseTables.ApplyEditedTables(False, int(), int(), int(), int(),
'')

            # 3.7 Atribuição das curvas de retração nos materiais aos materiais da longarina e
tabuleiro

            # Dados de retração na longarina
            ws_retração_fluência = wb_retração_fluência.sheets["Retração na longarina"]
            deformação_longarina = list(ws_retração_fluência['N4:N105'].value)
            dias_longarina = list(ws_retração_fluência['K4:K105'].value)

            # Dados de retração no tabuleiro
            ws_retração_fluência = wb_retração_fluência.sheets["Retração no tabuleiro"]
            deformação_tabuleiro = list(ws_retração_fluência['N4:N105'].value)
```

```
            dias_tabuleiro = list(ws_retração_fluência['K4:K105'].value)

            # Atribuição da retração aos materiais da longarina e tabuleiro por tabelas

            TableKey = 'Material Properties 05f - Time Dependence - User Shrinkage'
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = []
            NumberRecords = int()
            TableData = []
            ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,    GroupName,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
            ColumnLabels = list(ret[2])
            Resultados = list(ret[4])

            # Criação do dataframe a partir do dicionário

            retração_dict = {
                            'Material': [],
                            'A': [],
                            'B': [],
                            'Ho': [],
                            'Age': [],
                            'ShrinkStrn': []
                            }

            Material_Text = [f'C{f_ck} - Longarina', f'C{f_ck} - Tabuleiro']
            Age = [dias_longarina, dias_tabuleiro]
            Strain = [deformação_longarina, deformação_tabuleiro]
            for i in range(2): # Foi convertido em string para alimentar as tabelas no programa
                retração_dict['Material'].extend([str(Material_Text[i])]                    *
len(deformação_longarina))
                retração_dict['A'].extend([str(1)] * len(deformação_longarina))
                retração_dict['B'].extend([str(1)] * len(deformação_longarina))
                retração_dict['Ho'].extend([str(10)] * len(deformação_longarina))
                retração_dict['Age'].extend(str(element).replace(".",  ",")  for  element  in
Age[i])
                retração_dict['ShrinkStrn'].extend(str(element).replace(".", ",") for element
in Strain[i])

            df_concreto = pd.DataFrame(retração_dict)
            result = df_concreto.values.ravel().tolist()

            TableKey = 'Material Properties 05f - Time Dependence - User Shrinkage'
            TableVersion = int()
            FieldKeysIncluded = list(retração_dict.keys())
            NumberRecords = int()
            TableData = result
            ret   =   SapModel.DatabaseTables.SetTableForEditingArray(TableKey,   TableVersion,
FieldKeysIncluded, NumberRecords, TableData)
            ret = SapModel.DatabaseTables.ApplyEditedTables(False, int(), int(), int(), int(),
'')




            # 4 Carregamentos

            # 4.1 Load Patterns/Cases
            # Wearing surface e sidewalk são carregamentos a serem definidos especificamente
no módulo do CSiBridge (serão definidos posteriormente)
            # MyType
            LTYPE_DEAD = 1
            LTYPE_LIVE = 3
            LTYPE_WIND= 6
            LTYPE_OTHER = 8
            LTYPE_TEMPERATURE = 10
            LTYPE_BRAKING = 15
            LTYPE_HORIZONTALEARTHPRESSURE = 20
            LTYPE_TEMPERATUREGRADIENT = 26
            LTYPE_SHRINKAGE = 28
            LTYPE_CREEP = 29
            LTYPE_WATERLOADPRESSURE = 30
            #LTYPE_PEDESTRIANLL = 33
```

```
            LTYPE_PRESTRESS = 34
            LTYPE_IMPACT = 38
            #LTYPE_CONSTRUCTION = 39
            LTYPE_DEADWEARING = 40
            SelfWTMultiplier = 0 #The self weight multiplier for the new load pattern
            AddLoadCase = True #Associa o load pattern a uma load case
            ret    =    SapModel.LoadPatterns.Add('Defensas',    eLoadPatternType(LTYPE_DEAD),
SelfWTMultiplier, AddLoadCase)
            ret          =          SapModel.LoadPatterns.Add('Pav          +          recap',
eLoadPatternType(LTYPE_DEADWEARING), SelfWTMultiplier, AddLoadCase)
            #   ret   =   SapModel.LoadPatterns.Add('Passeio',   eLoadPatternType(LTYPE_DEAD),
SelfWTMultiplier, AddLoadCase)
            #   ret   =   SapModel.LoadPatterns.Add('Haunch',   eLoadPatternType(LTYPE_DEAD),
SelfWTMultiplier, AddLoadCase)
            ret     =     SapModel.LoadPatterns.Add('Vento',     eLoadPatternType(LTYPE_WIND),
SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       uniforme       pos',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       uniforme       neg',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Frenagem       e       aceleracao',
eLoadPatternType(LTYPE_BRAKING), SelfWTMultiplier, AddLoadCase)
            ret             =             SapModel.LoadPatterns.Add('Empuxo             terra',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
            ret      =      SapModel.LoadPatterns.Add('Empuxo     terra     carga     móvel',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       gradiente       pos',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       gradiente       neg',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
            ret = SapModel.LoadPatterns.Add('Água', eLoadPatternType(LTYPE_WATERLOADPRESSURE),
SelfWTMultiplier, AddLoadCase)
            ret  =  SapModel.LoadPatterns.Add('Protensão',  eLoadPatternType(LTYPE_PRESTRESS),
SelfWTMultiplier, AddLoadCase)
            ret    =    SapModel.LoadPatterns.Add('Impacto',    eLoadPatternType(LTYPE_IMPACT),
SelfWTMultiplier, AddLoadCase)

            # 4.2 Load Distributions

            # 4.2.1 Gradiente de temperatura
            ret   =   MyBridgeModeler.LoadDistribution.TemperatureGradient.AddNew('Temperatura
gradiente')

            Name = 'Temperatura gradiente' #The name of an existing bridge temperature gradient
load distribution.
            NumberDistances = 4 #The number of specified distances (between 1 and 5, inclusive).
            DistanceThatMayVary = 3 #The specified distance that may vary.
            TempDifferencesToConsider = 0
            # The temperature differences to consider.
            # 0 = Positive and Negative
            # 1 = Positive Only
            # 2 = Negative Only
            d_1 = min(0.3*(h_laje + h_long), 15)
            d_2 = min(max(0.3*(h_laje + h_long), 10), 25)
            d_3 = min(min(0.3*(h_laje + h_long), 10 + 10), (h_laje + h_long)-d_1-d_2)
            if (h_laje + h_long) < 20:
                ΔT_1 = 8.5
                ΔT_2 = 3.5
                ΔT_3 = 0.5
            elif 20 < (h_laje + h_long) <= 40:
                ΔT_1 = np.interp((h_laje + h_long), [20, 40], [8.5, 12])
                ΔT_2 = np.interp((h_laje + h_long), [20, 40], [3.5, 3])
                ΔT_3 = np.interp((h_laje + h_long), [20, 40], [0.5, 1.5])
            elif 40 < (h_laje + h_long) <= 60:
                ΔT_1 = np.interp((h_laje + h_long), [40, 60], [12, 13])
                ΔT_2 = np.interp((h_laje + h_long), [40, 60], [3, 3])
                ΔT_3 = np.interp((h_laje + h_long), [40, 60], [1.5, 2])
            elif 60 < (h_laje + h_long) < 80:
                ΔT_1 = np.interp((h_laje + h_long), [60, 80], [13, 13.5])
                ΔT_2 = np.interp((h_laje + h_long), [60, 80], [3, 3])
                ΔT_3 = np.interp((h_laje + h_long), [60, 80], [2, 2.5])
            else:
                ΔT_1 = 13.5
                ΔT_2 = 3.0
                ΔT_3 = 2.5
```

```
            d = [d_1, d_2, float(), d_3] #A zero-based one-dimensional array dimensioned to
(NumberDistances - 1) containing the specified distances. [L]
            TPos = [ΔT_1, ΔT_2, 0, 0, ΔT_3] #A zero-based one-dimensional array dimensioned to
(NumberDistances) containing the specified positive temperatures. [T]
            TNeg = [-5, -5, 0, 0, 0] #A zero-based one-dimensional array dimensioned to
(NumberDistances) containing the specified negative temperatures. [T]
            ret                                                                          =
MyBridgeModeler.LoadDistribution.TemperatureGradient.SetUserGeneralData(Name, NumberDistances,
DistanceThatMayVary, TempDifferencesToConsider, d, TPos, TNeg)

            # 4.2.2 Vento
            modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
            wb_pil_ext = xw.Book(modelo_pil_ext)
            ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

            Name = 'Vento'
            ret = MyBridgeModeler.LoadDistribution.Line.AddNew(Name)

            Name = Name # The name of an existing bridge line load distribution
            LoadType = 1 # The line load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 5
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LoadValue = (ws_pil_ext['D31'].value/(CVão/100))/100 # The load value. [F/L] for
force; [F] for moment
            ReferenceLocation = 2
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LoadDistance = 0 # The distance from the reference location to the load. [L]
            ret  =  MyBridgeModeler.LoadDistribution.Line.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LoadValue, ReferenceLocation, LoadDistance)

            # 4.2.3 Frenagem e aceleração
            Name = 'Frenagem e aceleracao'
            ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

            Name = Name # The name of an existing bridge area load distribution.
            LoadType = 1 # The area load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 4
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LeftEdgeValue = 0.32E-04 # The left edge load value. [F/L2] for force; [F/L] for
moment
            RightEdgeValue = 0.32E-04 # The right edge load value. [F/L2] for force; [F/L] for
moment
            LeftReferenceLocation = 1 #The left reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LeftLoadDistance = 0 # The distance from the left edge of load to the left reference
location. [L]
            RightReferenceLocation = 2 # The right reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            RightLoadDistance = 0 # The distance from the right edge of load to the right
reference location. [L]
```

```
            ret  =  MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
            LeftReferenceLocation,          LeftLoadDistance,          RightReferenceLocation,
RightLoadDistance)



            # 5 Criação do modelo da ponte - (cBridgeModeler_1.NewBridgeModel)

            StartStation = 0 #The starting station for the bridge. [L]
            InitialBearing = "N900000E" #The initial bearing for the bridge. This should be a
valid bearing angle such as "S624355E".
            Grade = 0 #The grade for the bridge specified in percent. A positive grade represents
an incline with increasing station.
            Skew = "Default"
            #The skew angle for the bridge. This should be a valid bearing angle such as
"S624355E", "Default" or "DefaultX"
            #where X is a positive or negative number indicating the number of degrees the
bearing is different from the default skew.
            #The default skew is perpendicular to the reference line.
            Radius = 0 #The bridge radius. A positive radius curves to the right and a negative
radius curves to the left. A radius of zero means an infinite radius (i.e., straight). [L]
            SuperElevation = 0 #The bridge superelevation specified in percent. A positive
value rotates clockwise corresponding to a curve right and a negative value rotates
counterclockwise corresponding to a curve left.
            SectionType = 11 # Diferença na numeração do pc do lab e o daqui de casa?
            #This is an integer between 2 and 16, inclusive, indicating the bridge section
type.
            # 2 = Concrete Box Girder - Exterior Girders Vertical
            # 3 = Concrete Box Girder - Exterior Girders Sloped
            # 4 = Concrete Box Girder - Exterior Girders Clipped
            # 5 = Concrete Box Girder - Exterior Girders with Radius
            # 6 = Concrete Box Girder - Exterior Girders Sloped Maximum
            # 7 = Concrete Tee Beam
            # 8 = Concrete Flat Slab
            # 9 = Steel I Girder
            # 10 = Concrete Box Girder - AASHTO-PCI-ASBI Standard
            # 11 = Precast Concrete I Girder
            # 12 = Precast Concrete U Girder
            # 13 = Concrete Box Girder - Advanced
            # 14 = Steel U Girder
            # 15 = Concrete Solid Girder
            # 16 = Precast Concreter Super-T Girder
            ret  =  MyBridgeModeler.NewBridgeModel(StartStation,  InitialBearing,  Grade,  Skew,
Radius, SuperElevation, SpanLength, SectionType)

            # 5.1 Objetos de interesse
            NumberNames = int()
            MyName = []
            ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
            myBridgeObjName = list(ret[2])[0]

            NumberNames = int()
            MyName = []
            BridgeSectionType = [SectionType]
            ret      =      MyBridgeModeler.DeckSection.GetNameList(NumberNames,      MyName,
BridgeSectionType)
            myDeckSectionName = list(ret[2])[0]

            Name = myBridgeObjName
            LayoutLine = ''
            ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
            myLayoutLineName = ret[1]

            ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
            myLaneName = list(ret[2])[0]

            # 5.2 Definição do Layout Line - (cBMLayoutLine.SetGeneralData). O layout aqui
definido será o principal (superestrutura). A passagem do veículo será definida pela criação de
outros layoutlines

            Name = myLayoutLineName
            CoordinateSystem = 'GLOBAL'
```

```
            StartX = 0
            StartY = 0
            StartZ = 0
            ret = MyBridgeModeler.LayoutLine.SetGeneralData(Name, CoordinateSystem, StartX,
StartY, StartZ)

            # 5.3 Bridge Object Data (cBMBridgeObject.SetSpanLayoutByLength Method)
            Name = myBridgeObjName
            StartStation = 0
            ret = MyBridgeModeler.BridgeObject.SetSpanLayoutByLength(Name, StartStation,
SpanName, SpanLength)

            # 5.4 Veiculos (cBMVehicle Methods)
            Region = 'User'
            Standard = 'User'
            Vehicle = 'All'
            ret = MyBridgeModeler.Vehicle.ImportVehicle(Region, Standard, Vehicle)
            ret = MyBridgeModeler.Vehicle.ConvertToUser('Multidao')

            # Correção (carregamento de Multidao que depende da largura do tabuleiro)

            Name = 'Multidao' # The name of the vehicle.
            UniformLoadSF = 1 # The uniform load scale factor.
            AxleLoadSF = 1 # The axle load scale factor.
            NumberLoadDataSets = 2 # The number of load data sets defined for the vehicle.
            LoadLengthType = [1, 2]
            # A zero-based one-dimensional array containing the load length type for each load
data set.
            # 1 = Leading Load
            # 2 = Trailing Load
            # 3 = Fixed Length
            # 4 = Variable Length
            MinimumDistance = [0, 0] # A zero-based one-dimensional array containing the minimum
distance between the current and previous axle. This item is only applicable if the
LoadLengthType is 3 or 4 (Fixed Length or Variable Length). [L]
            MaximumDistance = [float(), float()] # A zero-based one-dimensional array
containing the maximum distance between the current and previous axle. This item is only
applicable if the LoadLengthType is 4 (Variable Length). This item must either be zero (meaning
infinity) or it must be greater than or equal to MinimumDistance item. [L]
            UniformLoad = [(LarguraRolamento)/100*(5/100), (LarguraRolamento)/100*(5/100)] # A
zero-based one-dimensional array containing the uniform load. [F/L]
            UniformWidthType = [2, 2]
            # A zero-based one-dimensional array containing the width type for the uniform
load.
            # 1 = Zero Width
            # 2 = Lane Width
            # 3 = Fixed Width
            UniformWidth = [float(), float()] # A zero-based one-dimensional array containing
the uniform load width. This item is only applicable if the UnifWidthType is 3 (Fixed Width).
[L]
            AxleLoad = [0, float()] # A zero-based one-dimensional array containing the total
load for the axle. This item does not apply when the LoadLengthType item is 4 (Variable Length).
[F]
            AxleWidthType = [1, int()]
            # A zero-based one-dimensional array containing the axle width type indicating how
the axle load is to be distributed. This item does not apply when the LoadLengthType item is 4
(Variable Length).
            # 1 = One Point
            # 2 = Two Points
            # 3 = Four Points
            # 4 = Lane Width Line
            # 5 = Fixed Width Line
            AxleWidth = [float(), float()] # A zero-based one-dimensional array containing the
axle width. This item is only applicable if the AxleWidthType item is 2, 3 or 5 (Two Points,
Four Points or Fixed Width Line).
            # This item does not apply when the LoadLengthType item is 4 (Variable Length). [L]
            ret = MyBridgeModeler.Vehicle.SetVerticalLoads(Name, UniformLoadSF, AxleLoadSF,
NumberLoadDataSets, LoadLengthType, MinimumDistance, MaximumDistance, UniformLoad,
UniformWidthType, UniformWidth, AxleLoad, AxleWidthType, AxleWidth)

            ret = SapModel.LoadCases.Delete('MOVE1')
            ret = SapModel.LoadCases.Moving.SetCase('Veiculo')
            ret = SapModel.LoadCases.Moving.SetCase('Multidao')

            ret = SapModel.LoadCases.Moving.SetInitialCase('Veiculo', '')
```

```
            ret = SapModel.LoadCases.Moving.SetInitialCase('Multidao', '')

            NumberLoads = 1 #The number of loads assigned to the specified analysis case.
            MyClass = ['Veiculo'] #This is an array that includes the vehicle class for each
load assigned to the load case.
            SF = [1] #This is an array that includes the scale factor for each load assigned
to the load case.
            Min = [0] #This is an array that includes the minimum number of lanes loaded for
each load assigned to the load case.
            Max = [0] #This is an array that includes the maximum number of lanes loaded for
each load assigned to the load case. This item must be 0, or it must be greater than or equal
to Min. If this item is 0, all available lanes are loaded.
            ret = SapModel.LoadCases.Moving.SetLoads('Veiculo', NumberLoads, MyClass, SF, Min,
Max)
            MyClass = ['Multidao']
            ret = SapModel.LoadCases.Moving.SetLoads('Multidao', NumberLoads, MyClass, SF, Min,
Max)

            LoadNumber = 1 #The load assignment number.
            NumberItems = 1 #The number of lanes loaded for the specified load assignment
number.
            MyName = [f'{myLaneName}'] #This is an array that includes the name of each lane
loaded for the specified load assignment number.
            ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber, NumberItems,
MyName)
            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Multidao',     LoadNumber,
NumberItems, MyName)




            #  6  Aparelhos  de  apoio  a  partir  de  um  link  definido  pelo  usuário
(cBMBearing.SetBearingUser Method)

            # 6.1 Aparelho de apoio

            Name = 'Elastômero' #The name of an existing or new link property. If this is an
existing property, that property is modified; otherwise, a new property is added.
            DOF = [True, True, True, False, False, False]
            # This is a boolean array, dimensioned to 5, indicating if properties exist for a
specified degree of freedom.
            # DOF(0) = U1
            # DOF(1) = U2
            # DOF(2) = U3
            # DOF(3) = R1
            # DOF(4) = R2
            # DOF(5) = R3
            Fixed = [True, False, False, False, False, False]
            # This is a boolean array, dimensioned to 5, indicating if the specified degree of
freedom is fixed (restrained).
            # Fixed(0) = U1 fixity if DOF(0) = True
            # Fixed(1) = U2 fixity if DOF(1) = True
            # Fixed(2) = U3 fixity if DOF(2) = True
            # Fixed(3) = R1 fixity if DOF(3) = True
            # Fixed(4) = R2 fixity if DOF(4) = True
            # Fixed(5) = R3 fixity if DOF(5) = True
            Ke = [float(), K_ap, K_ap, float(), float(), float()]
            # This is an array of stiffness terms for the link property. There are 6 terms in
the array if the stiffness is uncoupled and 21 if it is coupled. The KeCoupled item indicates
if the stiffness is coupled.
            # If the stiffness is uncoupled:
            # Ke(0) = U1 [F/L]
            # Ke(1) = U2 [F/L]
            # Ke(2) = U3 [F/L]
            # Ke(3) = R1 [FL]
            # Ke(4) = R2 [FL]
            # Ke(5) = R3 [FL]
            # If the stiffness is coupled:
            # Ke(0) = U1U1 [F/L]
            # Ke(1) = U1U2 [F/L]
            # Ke(2) = U2U2 [F/L]
            # Ke(3) = U1U3 [F/L]
            # Ke(4) = U2U3 [F/L]
            # Ke(5) = U3U3 [F/L]
```

```
            # Ke(6) = U1R1 [F]
            # Ke(7) = U2R1 [F]
            # Ke(8) = U3R1 [F]
            # Ke(9) = R1R1 [FL]
            # Ke(10) = U1R2 [F]
            # Ke(11) = U2R2 [F]
            # Ke(12) = U3R2 [F]
            # Ke(13) = R1R2 [FL]
            # Ke(14) = R2R2 [FL]
            # Ke(15) = U1R3 [F]
            # Ke(16) = U2R3 [F]
            # Ke(17) = U3R3 [F]
            # Ke(18) = R1R3 [FL]
            # Ke(19) = R2R3 [FL]
            # Ke(20) = R3R3 [FL]
            Ce = [float(), 0, 0, float(), float(), float()]
            # This is an array of damping terms for the link property. There are 6 terms in the
array if the damping is uncoupled and 21 if it is coupled. The CeCoupled item indicates if the
damping is coupled.
            # If the damping is uncoupled:
            # Ce(0) = U1 [F/L]
            # Ce(1) = U2 [F/L]
            # Ce(2) = U3 [F/L]
            # Ce(3) = R1 [FL]
            # Ce(4) = R2 [FL]
            # Ce(5) = R3 [FL]
            # If the damping is coupled:
            # Ce(0) = U1U1 [F/L]
            # Ce(1) = U1U2 [F/L]
            # Ce(2) = U2U2 [F/L]
            # Ce(3) = U1U3 [F/L]
            # Ce(4) = U2U3 [F/L]
            # Ce(5) = U3U3 [F/L]
            # Ce(6) = U1R1 [F]
            # Ce(7) = U2R1 [F]
            # Ce(8) = U3R1 [F]
            # Ce(9) = R1R1 [FL]
            # Ce(10) = U1R2 [F]
            # Ce(11) = U2R2 [F]
            # Ce(12) = U3R2 [F]
            # Ce(13) = R1R2 [FL]
            # Ce(14) = R2R2 [FL]
            # Ce(15) = U1R3 [F]
            # Ce(16) = U2R3 [F]
            # Ce(17) = U3R3 [F]
            # Ce(18) = R1R3 [FL]
            # Ce(19) = R2R3 [FL]
            dj2 = 0 #The distance from the J-End of the link to the U2 shear spring. This item
applies only when DOF(1) = True. [L]
            dj3 = 0 #The distance from the J-End of the link to the U3 shear spring. This item
applies only when DOF(2) = True. [L]
            KeCoupled = False #This item is True if the link stiffness, Ke, is coupled. There
are 21 terms in the Ke array if Ke is coupled; otherwise there are 6 terms.
            CeCoupled = False #This item is True if the link damping, Ce, is coupled. There are
21 terms in the Ce array if Ce is coupled; otherwise there are 6 terms.
            Notes = '' #The notes, if any, assigned to the property.
            GUID = '' #The GUID (global unique identifier), if any, assigned to the property.
If this item is input as Default, the program assigns a GUID to the property.
            ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3, KeCoupled,
CeCoupled, Notes, GUID) #Criação do link Neoprene

            ret = MyBridgeModeler.Bearing.AddNew('Ap. de apoio')
            ret = MyBridgeModeler.Bearing.SetBearingLink('Ap. de apoio', 'Elastômero')

            # 6.2 Link Rígido
            Name = 'Link rigido' #The name of an existing or new link property. If this is an
existing property, that property is modified; otherwise, a new property is added.
            DOF = [True, True, True, True, True, True]
            Fixed = [True, True, True, True, True, True]
            Ke = [float(), float(), float(), float(), float(), float()]
            Ce = [float(), float(), float(), float(), float(), float()]
            dj2 = 0
            dj3 = 0
            KeCoupled = False
            CeCoupled = False
```

```
            Notes = ''
            GUID = ''
            ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3, KeCoupled,
CeCoupled, Notes, GUID)

            Name = 'Link rigido (esforços)' #The name of an existing or new link property. If
this is an existing property, that property is modified; otherwise, a new property is added.
            DOF = [True, True, True, True, True, True]
            Fixed = [False, False, False, False, False, False]
            Ke = [1E9, 1E9, 1E9, 1E13, 1E13, 1E13] # Segundo a documentação do CSiBridge, a
rigidez de 1e9 kN/cm é uma rigidez suficientemente infinita para as translações e 1e13 para as
rotações
            Ce = [float(), float(), float(), float(), float(), float()]
            dj2 = 0
            dj3 = 0
            KeCoupled = False
            CeCoupled = False
            Notes = ''
            GUID = ''
            ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3, KeCoupled,
CeCoupled, Notes, GUID)




            # 7. Definição dos apoios extremos, intermediários (travessas), abutments e
atribuições

            ret = MyBridgeModeler.Bent.AddNew('Apoios extremos')
            ret = MyBridgeModeler.Bent.AddNew('Apoios intermediários')
            ret = MyBridgeModeler.Abutment.AddNew('Apoio aparelho de apoio')

            # 7.1 Definição - Travessas extremas

            Name = 'Apoios extremos' #The name of an existing bridge bent.
            CapBeamSection = f'Travessa encontro (C{f_ck})' #This is either "None" or the name
of the frame section property assigned to the cap beam.
            Foundation = 'Fixed' #Containing either Fixed, Pinned, or a defined foundation
spring.
            BentType = 0
            # 0 = Single bearing line (continuous superstructure)
            # 1 = Double bearing line (discontinuous superstructure)
            # 2 = Double bearing line with closure pour
            ret = MyBridgeModeler.Bent.SetGeneralData(Name, CapBeamSection, C_travessaext,
SupportType_ext, N_pilaresext, Foundation, BentType)

            Section = [f'Pilar (C{f_ck})'] * N_pilaresext #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing the frame section properties assigned to each
column.
            Angle = [0] * N_pilaresext #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the orientation of the column in degrees counterclockwise between
a line parallel to the bent and the column local 2 axis. [deg]
            Foundation = ['Fixed'] * N_pilaresext #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing either Fixed, Pinned, a defined foundation spring,
or Previous. Note that an error will occur if Previous is assigned to the first column.
            ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_ext, Height_ext,
Angle, Foundation)

            SuperConnection = 1
            # The superstructure connection.
            # 0 = Connect to girder top and bottom
            # 1 = Connect to girder bottom only
            ret = MyBridgeModeler.Bent.SetSuperConnectionSingleLine(Name, SuperConnection)

            # 7.2 Definição - Travessas intermediárias
            Name = 'Apoios intermediários'
            CapBeamSection = f'Travessa intermediária (C{f_ck})'
            Foundation = 'Fixed'
            BentType = 1
            ret = MyBridgeModeler.Bent.SetGeneralData(Name, CapBeamSection, C_travessaint,
SupportType_int, N_pilaresint, Foundation, BentType)

            Section = [f'Pilar (C{f_ck})'] * N_pilaresint
            Angle = [0] * N_pilaresint
```

```
            Foundation = ['Fixed'] * N_pilaresint
            ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_int, Height_int,
Angle, Foundation)

            BeforeSuperConnection = 1
            # The superstructure connection before the bent.
            # 0 = Connect to girder top and bottom
            # 1 = Connect to girder bottom only
            BeforeDistanceToBearingLine = d_ap #The distance to the bearing line before the
bent. [L]
            BeforeDistanceToEndOfGirder = 0 #The distance to the end of the girder before the
bent. [L]
            AfterSuperConnection = 1
            # The superstructure connection after the bent.
            # 0 = Connect to girder top and bottom
            # 1 = Connect to girder bottom only
            AfterDistanceToBearingLine = d_ap #The distance to the bearing line after the bent.
[L]
            AfterDistanceToEndOfGirder = 0 #AfterDistanceToEndOfGirder
            ret           =           MyBridgeModeler.Bent.SetSuperConnectionDoubleLine(Name,
BeforeSuperConnection,        BeforeDistanceToBearingLine,        BeforeDistanceToEndOfGirder,
AfterSuperConnection, AfterDistanceToBearingLine, AfterDistanceToEndOfGirder)

            # 7.5 Definição - Apoio aparelho de apoio
            Name = 'Apoio aparelho de apoio' # The name of an existing bridge abutment.
            SuperConnection = 1
            # The superstructure connection.
            # 0 = Integral
            # 1 = Connect to Girder Bottom Only
            DistanceFromBridgeEnd = float() # The distance from the bridge end to the bearing
line. [L]
            FoundationProperty = 'Fixed' # This is either Fixed, Pinned or the name of a defined
foundation spring.
            SubstructureType = 1
            # The substructure type.
            # 0 = None
            # 1 = Foundation spring
            # 2 = Continuous beam (continuously supported)
            BeamProperty = '' # The frame section property assigned to the continuous beam.
This item only applies when the SubstructureType item is 1, Continuous beam (continuously
supported).
            BeamLength = float() # The length of the continuous beam. This item only applies
when the SubstructureType item is 1, Continuous beam (continuously supported). [L]
            ret      =      MyBridgeModeler.Abutment.SetAbutment(Name,      SuperConnection,
DistanceFromBridgeEnd, FoundationProperty, SubstructureType, BeamProperty, BeamLength)

            # 7.6 Atribuições

            Name = myBridgeObjName #The name of a bridge object.

            if NVãos != 1:
                for i in range(NVãos - 1):
                    SpanNameBeforeBent = f'Vão {i+1}' #The name of the span before the
considered bent.
                    BentProperty = 'Apoios intermediários' #The bent property assigned to the
specified bent.
                    BentDirection = 'Default' #The bearing of a line parallel to the bent. It
is either Default or a valid bearing string.
                    ret   =   MyBridgeModeler.BridgeObject.BentAssign.SetBentAssignment(Name,
SpanNameBeforeBent, BentProperty, BentDirection)

                    BentSupportName = f'Apoio {i+1}'
                    ret     =     MyBridgeModeler.BridgeObject.BentAssign.SetSupportName(Name,
SpanNameBeforeBent, BentSupportName)

                    SuperstructureIsContinousAtBent   =   False   #Indicates   whether   the
superstructure is continuous (True) or discontinuous (False) at the bent.
                    DiaphragmBeforeBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the diaphragm property at the bent.
                    # If the specified bent property is None or it has a single line of bearings
(continuous superstructure) this is the property of the diaphragm that occurs at the bent.
                    # If the specified bent property has a double line of bearings
(discontinuous superstructure) this is the property of the diaphragm that occurs just before
the bent.
```

```
                    # This item does not apply if the specified bent property is a double
bearing line with closure pour.
                    InternalDiaphragmBeforeBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the internal diaphragm property at the bent.
                    # This only applies to bents at steel U girder or super T girder deck
sections.
                    # If the specified bent property is None or it has a single line of bearings
(continuous superstructure) this is the property of the internal diaphragm that occurs at the
bent.
                    # If the specified bent property has a double line of bearings this is the
property of the internal diaphragm that occurs just before the bent.
                    DiaphragmAfterBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the diaphragm property after the bent.
                    # It applies when the specified bent property has a double line of bearings
(discontinuous superstructure).
                    # This item does not apply if the specified bent property is a double
bearing line with closure pour.
                    InternalDiaphragmAfterBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the internal diaphragm property after the bent.
                    # This only applies to bents at steel U girder or super T girder deck
sections.
                    # It applies when the specified bent property has a double line of bearings
and is not a closure pour.
                    ClosurePourDiaphragm = 'None' #This is the name of a defined closure pour
bridge diaphragm property. This item only applies when the specified bent type is Double Bearing
Line with Closure Pour.
                    Restrainer = 'None' #This is either None or the name of a defined bridge
diaphragm property indicating the restrainers used at the bent. It only applies when the
superstructure is discontinuous at the bent.
                    RestrainerElevation = 0 #The restrainer elevation (at layout line, global
Z). this item only applies when the superstructure is discontinuous at the bent. [L]
                    MeshSuperstructureToMatchBent = False
                    # This is a flag indicating if the superstructure is to be meshed to match
the bent.
                    # This item only applies when the superstructure is continuous at the bent.
                    MeshSteelUGirderToMatchBent = False
                    # This is a flag indicating if the steel U girders are to be meshed to
match the bent.
                    # This item only applies to bents at steel U girder or super T girder deck
sections when the superstructure is continuous at the bent.
                    ret                                                                    =
MyBridgeModeler.BridgeObject.BentAssign.SetSuperstructureAssignment(Name,   SpanNameBeforeBent,
SuperstructureIsContinousAtBent,       DiaphragmBeforeBent,       InternalDiaphragmBeforeBent,
DiaphragmAfterBent, InternalDiaphragmAfterBent,
                    ClosurePourDiaphragm,          Restrainer,          RestrainerElevation,
MeshSuperstructureToMatchBent, MeshSteelUGirderToMatchBent)

                    Elevation = -h_laje -h_long - h_ap #The global Z elevation of the bent. [L]
                    HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the cap beam. [L]
                    ret    =    MyBridgeModeler.BridgeObject.BentAssign.SetBentLocation(Name,
SpanNameBeforeBent, Elevation, HorizontalOffset)

                    for j in range(2):
                        BearingLocation = [1, 2]
                        # This is one of the following indicating the bent bearing location:
                        # 1 = Single line of bearings or bearings before bent for double line
of bearings.
                        # 2 = Bearings after bent for double line of bearings.
                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the bent.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the bent.
                        ElevationAtLayoutLine = -h_laje -h_long - h_ap/2 #The bearing elevation
(global Z) at the layout line. [L]
```

```
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int()
                        #The number of bearings for the bridge section or for abutments at
Steel U Girder deck sections the number of bearings at the U girder.
                        # For bearings at steel U girder sections this item must be 1 or 2.
                        # This item only applies when the bearing assignment type is General
or when the abutment is at a Steel U Girder deck section.
                        UniformSpacing = int() #The uniform spacing of bearings. This item only
applies when the bearing assignment type is General. [L]
                        BearingOffset = int() #The offset from the section reference point to
the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                              =
MyBridgeModeler.BridgeObject.BentAssign.SetBearingAssignment(Name,          SpanNameBeforeBent,
BearingLocation[j],      BearingAssignmentType,      BearingProperty,      RestrainerProperty,
ElevationAtLayoutLine, RotationAngle, NumberOfBearings,
                        UniformSpacing, BearingOffset)




            # 8 Definição e atribuição dos diafragmas

            Name = myBridgeObjName
            SpanName = '' #The name of the span from which the diaphragms are deleted. If this
is input as a blank string then all diaphragms in all spans will be deleted.
            ret = MyBridgeModeler.BridgeObject.DiaphragmAssign.DeleteAll(Name, SpanName)

            NumberNames = int() #The number of diaphragm names retrieved by the program.
            MyName = [] #This is a one-dimensional array of diaphragm names. The MyName array
is created as a dynamic, zero-based array by the API user: Dim MyName() as String.
            #The array is dimensioned to (NumberNames - 1) inside the program, filled with the
names, and returned to the API user.
            MyType = []
            # This is a one-dimensional array, dimensioned to (NumberNames - 1), of diaphragm
types.
            # 1 = Solid
            # 2 = Chord and Brace
            # 3 = Single Beam
            # 4 = Steel Plate
            # 5 = Closure Pour
            ret = MyBridgeModeler.Diaphragm.GetNameList(NumberNames, MyName, MyType)
            NDiaphragms = ret[1]
            DiaphragmNames = list(ret[2])
            ValidDiapgragmNames = ['Transversina']
            for i in range (NDiaphragms):
                if DiaphragmNames[i] not in ValidDiapgragmNames:
                    ret = MyBridgeModeler.Diaphragm.Delete(DiaphragmNames[i])

            Name = 'Transversina'
            MyType = 3
            # The diaphragm type.
            # 1 = Solid
            # 2 = Chord and Brace
            # 3 = Single Beam
            # 4 = Steel Plate
            # 5 = Closure Pour
            ret = MyBridgeModeler.Diaphragm.AddNew(Name, MyType)

            Name = 'Transversina' #The name of an existing single beam-type bridge diaphragm.
            BeamSection = f'Transversina de apoio (C{f_ck})' #The steel frame section property
used for the beam.
            ElevationChangeTop = d_long[1] #The elevation change from top of beam to top of
adjacent girder. [L]
            ModelMixed = bool(False) #Indicates if the beam is modeled as mixed frame and shell
            ConnectionPlateWidth = 15 #The width of the connection plate. [L]
            ConnectionPlateThickness = 0.8 #The thickness of the connection plate. [L]
            ConnectionPlateMaterial = 'A709Gr50' #The steel material for the connection plate.
            ConnectionPlateOnBothSidesOfWeb = bool(True) #Indicates if the connection plate is
provided on both sides of the web.
            ret       =       MyBridgeModeler.Diaphragm.SetSingleBeam(Name,       BeamSection,
ElevationChangeTop,      ModelMixed,      ConnectionPlateWidth,      ConnectionPlateThickness,
ConnectionPlateMaterial, ConnectionPlateOnBothSidesOfWeb)
```

```
            ret = SapModel.PropFrame.SetRectangle(f'Transversina de apoio (C{f_ck})', f'C{f_ck}
- Tabuleiro', h_transv, bw_transv) # Redefinição do material (limitação da API)

            for j in range(NVãos):
                for i in range(NDiafragmasPorVão):
                    Name = myBridgeObjName #The name of a bridge object.
                    SpanName = f'Vão {j+1}' #The name of the span in which the diaphragm exists.
                    DiaphragmProperty = 'Transversina' #The name of a defined diaphram property
assigned to the diaphragm.
                    Location = int(0)
                    # This is one of the following indicating the diaphragm location across the
bridge section.
                    # 0 = All Spaces
                    # 1 = Girder n to Girder n + 1
                    # 2 = All Girder Interiors
                    # 3 = Girder n Interior
                    GirderNumber = int() #The girder number identified as n when the Location
item is 1 or 3 (Girder n to Girder n + 1 or Girder n Interior). This item does not apply when
the Location item is 0 or 2.
                    Bearing = 'Default' #This is either Default or a valid bearing string
                    ReferenceLine = 0 #This is either 0 indicating the layout line or nonzero
indicating a girder number. This item is only under user control when the Location item is 1
(Girder n to Girder n + 1).
                    ErrorMessage = '' #If the function returns an error related to the diaphragm
distance this string may provide information about the error.
                    ret = MyBridgeModeler.BridgeObject.DiaphragmAssign.AddNew(Name, SpanName,
DiaphragmDistance[i], DiaphragmProperty, Location, GirderNumber, Bearing, ReferenceLine,
ErrorMessage)




            # 9 Bridge Section Data - Definition (cBMDeckSectionPrecastConcreteIGirder
Interface)
            LeftFillet = 0 #The horizontal dimension of the fillet on the left side of the deck
(identified as f1 on the Define Bridge Section Data form). [L]
            RightFillet = 0 #The horizontal dimension of the fillet on the right side of the
deck (identified as f2 on the Define Bridge Section Data form). [L]
            ret                                                                              =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetFilletHorizontalDimensions(myDeckSection
Name, LeftFillet, RightFillet)

            SlabMaterial = f'C{f_ck} - Tabuleiro' #The name of the slab material property. This
is intended to be a concrete property
            GirderLongitudinalLayout = 1
            # Indicates how the girders are laid out at a horizontal curve in a bridge object.
            # 1 = Along layout line.
            # 2 = Straight line.
            ret                                                                              =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetGeneralData(myDeckSectionName,
SlabMaterial, NumberInteriorGirders, TotalWidthOfDeckSection, GirderLongitudinalLayout)

            ConstantGirderSection = True #Indicates if all of the girder sections are the same.
            GirderSection = [f'Longarina (C{f_ck})']
            ret                                                                              =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetGirderSectionProperties(myDeckSectionNam
e, ConstantGirderSection, GirderSection)

            ConstantSpacing = True
            Spacing = []
            ret                                                                              =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetGirderSpacingData(myDeckSectionName,
ConstantSpacing, Spacing)


            DistanceToFillet = b_long[0]/2 #The overhang distance to the fillet (identified as
L3 on the Define Bridge Section Data form). [L] Qualquer valor inferior à metade da largura do
flange superior dará erro na definição da seção
            OuterThickness = h_laje #The overhang outer thickness (identified as t5 on the
Define Bridge Section Data form). [L]
            ret                                                                              =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetLeftOverhangDimensions(myDeckSectionName
, OverhangLength, DistanceToFillet, OuterThickness)
```

```
            DistanceToFillet = b_long[0]/2 #The overhang distance to the fillet (identified as
L3 on the Define Bridge Section Data form). [L]
            OuterThickness = h_laje #The overhang outer thickness (identified as t5 on the
Define Bridge Section Data form). [L]
            ret                                                                          =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetRightOverhangDimensions(myDeckSectionNam
e, OverhangLength, DistanceToFillet, OuterThickness)

            TopSlabThickness = h_laje #The thickness of the top slab (t1 on Define Bridge
Section Data form). [L]
            ConstantHaunchHeight = True #Indicates if the haunch heights are constant for all
girders.
            HaunchHeight = [0]
            # A zero-based, one-dimensional array of girder haunch heights dimensioned to
(NumberInteriorGirders + 1)
            # The haunch heights are entered in order in this array from the left exterior
girder to the right exterior girder.
            # Only the first height in this array is used if the ConstantHaunchHeight item is
True. [L]
            ret                                                                          =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetSlabThicknessData(myDeckSectionName,
TopSlabThickness, ConstantHaunchHeight, HaunchHeight)

            X = 0 #The offset in the X direction from the reference point to the insertion
point. [L]
            Y = 0 #The offset in the Y direction from the reference point to the insertion
point. [L]
            ret = MyBridgeModeler.DeckSection.InsertionPoint.SetPoint(myDeckSectionName, X, Y)




            # 10 Bridge Section Data - Loads (cBMDeckSectionLoadData Interface) - Métodos:
SetBarrierLoads, SetConcretePourFormLoads, SetConcretePourHaunchLoads,
            #          SetConcretePourPermanentLoads,          SetConcretePourTemporaryLoads,
SetLiveLoadCurbLocations, SetSidewalkLoads, SetTemperatureLoads, SetWearingSurfaceLoads

            # 10.1 Carregamentos Deck Section Data

            LoadPatternName = 'Defensas' #This is either None or the name of the load pattern
to which the barrier load is applied. The intent is that the load pattern would be of type Dead.
            NumberBarriers = 2 #The number of barriers specified
            MeasureFrom = [1,2]
            # A zero-based one-dimensional array indicating the reference point the barrier
location is measured from for each barrier.
            # 1 = Left Edge of Deck
            # 2 = Right Edge of Deck
            DistanceToOuterEdge = [0,0] #The distance from the reference point specified by the
MeasureFrom item to the outer edge of the barrier. [L]
            Width = [40,40] #The barrier width. [L]
            Height = [81,81] #The barrier height. [L]
            WeightEccentricity = [0,0] #The eccentricity of the barrier weight. [L]
            WeightPerLength = [5.075/100,5.075/100] #The barrier weight per unit length. [F/L]
            ret    =    MyBridgeModeler.DeckSection.LoadData.SetBarrierLoads(myDeckSectionName,
LoadPatternName,    NumberBarriers,    MeasureFrom,    DistanceToOuterEdge,    Width,    Height,
WeightEccentricity, WeightPerLength)

            LoadPatternName = '' #This is either None or the name of the load pattern to which
the haunch load is applied. The intent is that the load pattern would be of type Dead.
            UnitWeight = float() #The unit weight of the form load. [F/L2]
            ret                                                                          =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourFormLoads(myDeckSectionName,
LoadPatternName, UnitWeight)

            LoadPatternName = 'None'
            ProgramDetermined = True #This flag indicates whether the haunch load is program
determined or user defined.
            UserValue  =  float()#The  haunch  load  value.  This  item  only  applies  when  the
ProgramDetermined item is False. [F/L]
            ret                                                                          =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourHaunchLoads(myDeckSectionName,
LoadPatternName, ProgramDetermined, UserValue)

            LoadPatternName = ''
```

```
            WearingSurface = float() #The integral wearing surface load. [F/L2]
            Utilities = float() #The utilities load. [F/L2]
            Miscellaneous = float() #The miscellaneous load. [F/L2]
            ret                                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourPermanentLoads(myDeckSectionName,
LoadPatternName, WearingSurface, Utilities, Miscellaneous)

            AdditionalOverhang = float()#The additional overhang load. [F/L2]
            ret                                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourTemporaryLoads(myDeckSectionName,
LoadPatternName, AdditionalOverhang, Utilities, Miscellaneous)

            DistToInsideEdgeOfLeftCurb = float()
            DistToInsideEdgeOfRightCurb = float()
            DistanceToCenterlineOfMedianCurb = float()
            WidthOfMedianCurb = float()
            ret                                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetLiveLoadCurbLocations(myDeckSectionName,
ProgramDetermined,          DistToInsideEdgeOfLeftCurb,          DistToInsideEdgeOfRightCurb,
DistanceToCenterlineOfMedianCurb, WidthOfMedianCurb)

            LoadPatternName = '' #This is either None or the name of the load pattern to which
the sidewalk load is applied. The intent is that the load pattern would be of type Dead.
            NumberSidewalks = 0
            MeasureFrom = []
            DistanceToOuterEdge =[]
            Width = []
            WeightPerArea = []
            ret   =   MyBridgeModeler.DeckSection.LoadData.SetSidewalkLoads(myDeckSectionName,
LoadPatternName, NumberSidewalks, MeasureFrom, DistanceToOuterEdge, Width, WeightPerArea)

            LoadPatternPositiveName = 'Temperatura uniforme pos'#This is either None or the
name of the load pattern to which the positive temperature load is applied. The intent is that
the load pattern would be of type Temperature.
            TemperatureChangePositive = 15 #The positive temperature change. [T]
            LoadPatternNegativeName = 'Temperatura uniforme neg'#This is either None or the
name of the load pattern to which the negative temperature load is applied. The intent is that
the load pattern would be of type Temperature.
            TemperatureChangeNegative = -15 #The negative temperature change that should be
entered as zero or a negative number. [T]
            ret = MyBridgeModeler.DeckSection.LoadData.SetTemperatureLoads(myDeckSectionName,
LoadPatternPositiveName,          TemperatureChangePositive,          LoadPatternNegativeName,
TemperatureChangeNegative)

            LoadPatternName = 'Pav + recap'
            WeightPerArea = 4.4/(10**4)
            ret                                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetWearingSurfaceLoads(myDeckSectionName,
LoadPatternName, WeightPerArea)

            # 10.2 Limpeza das inutilizadas
            ret = SapModel.LoadPatterns.Delete('Temperature Pos')
            ret = SapModel.LoadPatterns.Delete('Temperature Neg')
            ret = SapModel.LoadPatterns.Delete('ConcPour Permanent')
            ret = SapModel.LoadPatterns.Delete('ConcPour Temporary')
            ret = SapModel.LoadPatterns.Delete('PreTension')
            ret = SapModel.LoadPatterns.Delete('Barrier')




            # 11 Retração e fluência - Staged Construction nos elementos da ponte
            # Apenas os load cases Veiculo, Multidao, Retração, Fluência e Cronograma são
necessários de definição (os load patterns já viram load cases automaticamente pela definição
dada)

            # 11.1 Update no modelo para ser possível selecionar os elementos da ponte
            Name = myBridgeObjName
            Action = 4 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
```

```
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ModelType = 2
            #An integer value between 1 and 3, inclusive, indicating the model type to be
created. This item applies when the Action item is 1 or 4.
            # 1 = Update as Spine Model Using Frame Objects
            # 2 = Update as Area Object Modell
            # 3 = Update as Solid Object Model
            SubmeshSize = 50 #The preferred maximum submesh size. This item applies when the
Action item is 1 or 4 and the ModelType item is 2 or 3. [L]
            ApplyAreaPropModifier = False #This item is to indicate whether or not to apply
property modifiers to area objects representing the bridge section to adjust for overlap.
            # This option only applies to bridge sections of type concrete box-girder, concrete
T-beam, flat slab, and user-defined (Section Designer).
            # When the bridge object is updated as an area-object model (ModelType is "Area"),
entering True will apply property modifiers to the area objects representing the bridge cross
section that will adjust their weight, mass, and stiffness factors f11 and m11
            # to match a corresponding solid-object model (ModelType is "Solid"), which has no
overlaps. This item applies when the Action item is 1 or 4 and the ModelType item is 2. [L]
            # True = Apply area property modifiers
            # False = Do not apply area property modifiers
            MaxDeckSegmentLength = 50 #The maximum segment length for deck spans. This item
applies when the Action item is 1 or 4. [L]
            MaxCapSegmentLength = 50 #The maximum segment length for bent cap beams. This item
applies when the Action item is 1 or 4. [L]
            MaxColumnSegmentLength = 50 #The maximum segment length for bent columns. This item
applies when the Action item is 1 or 4. [L]
            MeshSlabAtCriticalSteelIGirderLocations = False #A flag indicating that the slab
should be meshed at critical steel I-girder locations. This item applies when the Action item
is 1 or 4 and the bridge deck section is a Steel I-Girder section.
            ret = MyBridgeModeler.BridgeObject.Update(Name, Action, ModelType, SubmeshSize,
ApplyAreaPropModifier, MaxDeckSegmentLength, MaxCapSegmentLength, MaxColumnSegmentLength,
MeshSlabAtCriticalSteelIGirderLocations)

            # 11.2 Retração e fluência

            ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('Retração e fluência')

            InitialCase = ''
            ret = SapModel.LoadCases.StaticNonlinearStaged.GetInitialCase('Retração e
fluência', InitialCase)

            NLGeomType = 1
            # This is 0, 1 or 2, indicating the geometric nonlinearity option selected for the
load case.
            # 0 = None
            # 1 = P-delta
            # 2 = P-delta plus large displacements
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('Retração
e fluência', NLGeomType)

            TimeDepMatProp = True
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('Retração e
fluência', TimeDepMatProp)

            StagedSaveOption = 0
            # This is 0, 1, 2 or 3, indicating the results saved option for the load case.
            # 0 = End of final stage
            # 1 = End of each stage
            # 2 = Start and end of each stage
            # 3 = Two or more times in each stage


            ret = SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('Retração e
fluência', StagedSaveOption)

            NumberStages = 2 #The number of stages defined for the specified load case.
            Duration = [3, 3000]#This is an array that includes the duration in days for each
stage.
            Output = [True, True]#This is an array that includes True or False, indicating if
analysis output is to be saved for each stage.
            OutputName = ['Step 1', 'Step 2']#This is an array that includes a user-specified
output name for each stage.
            Comment = ['', '']#This is an array that includes a comment for each stage. The
comment may be a blank string.
```

```
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('Retração e fluência', NumberStages, Duration, Output, OutputName, Comment)

            Stage = 1 #The stage in the specified load case to which the data applies. Stages are numbered sequentially starting from 1.
            NumberOperations = 2 #The number of operations in the specified stage
            Operation = [1,1]
            # This is an array that includes 1, 2, 3, 4, 5, 6, 7, or 11, indicating an operation type.
            # 1 = Add structure
            # 2 = Remove structure
            # 3 = Load objects if new
            # 4 = Load objects
            # 5 = Change section properties
            # 6 = Change section property modifiers
            # 7 = Change releases
            # 11 = Change section properties and age
            ObjectType = ['Group', 'Group']
            # This is an array that includes the object type associated with the specified operation. The object type may be one of the following:
            # Group
            # Frame
            # Cable
            # Tendon
            # Area
            # Solid
            # Link
            # Point
            # The following list shows which object types are applicable to each operation type:
            # Operation = 1 (Add structure): All object types
            # Operation = 2 (Remove structure): All object types
            # Operation = 3 (Load objects if new): All object types
            # Operation = 4 (Load objects): All object types
            # Operation = 5 (Change section properties): All object types except Point
            # Operation = 6 (Change section property modifiers): Group, Frame, Cable, Area
            # Operation = 7 (Change releases): Group, Frame
            # Operation = 11 (Change section properties and age): All object types except Point
            ObjectName = [f'<{myBridgeObjName}>ALL_BUT_SLAB', f'<{myBridgeObjName}>SLAB']
            # This is an array that includes the name of the object associated with the specified operation.
            # This is the name of a Group, Frame object, Cable object, Tendon object, Area object, Solid object, Link object or Point object, depending on the ObjectType item.
            Age = [28, 38]#This is an array that includes the age of the added structure, at the time it is added, in days. This item applies only to operations with Operation = 1.
            MyType = ['','']
            # This is an array that includes a load type or an object type, depending on what is specified for the Operation item. This item applies only to operations with Operation = 3, 4, 5, 6, 7, or 11.
            # When Operation = 3 or 4, this is an array that includes Load or Accel, indicating the load type of an added load.
            # When Operation = 5 or 11, and the ObjectType item is Group, this is an array that includes Frame, Cable, Tendon, Area, Solid or Link, indicating the object type for which the section property is changed.
            # When Operation = 6 and the ObjectType item is Group, this is an array that includes Frame, Cable or Area, indicating the object type for which the section property modifiers are changed.
            # When Operation = 7 and the ObjectType item is Group, this is an array that includes Frame, indicating the object type for which the releases are changed.
            # When Operation = 5, 6, 7, or 11, and the ObjectType item is not Group and not Point, this item is ignored and the type is picked up from the ObjectType item.
            MyName = ['','']
            # This is an array that includes a load assignment or an object name, depending on what is specified for the Operation item. This item applies only to operations with Operation = 3, 4, 5, 6, 7 or 11.
            # When Operation = 3 or 4, this is an array that includes the name of the load assigned to the operation. If the associated LoadType item is Load, this item is the name of a defined load pattern. If the associated LoadType item is Accel, this item is UX, UY, UZ, RX, RY or RZ, indicating the direction of the load.
            # When Operation = 5 or 11, this is the name of a Frame, Cable, Tendon, Area, Solid or Link object, depending on the object type specified.
            # When Operation = 6, this is the name of a Frame, Cable or Area object, depending on the object type specified.
            # When Operation = 7, this is the name of a Frame object.
```

```
            SF = [float(), float()]
            # This is an array that includes the scale factor for the load assigned to the
operation, if any. [L/s2] for Accel UX UY and UZ; otherwise unitless
            # This item applies only to operations with Operation = 3 or 4.
            ret    =    SapModel.LoadCases.StaticNonlinearStaged.SetStageData_2('Retração    e
fluência', Stage, NumberOperations, Operation, ObjectType, ObjectName, Age, MyType, MyName, SF)

            Stage = 2
            NumberOperations = 1
            Operation = [1]
            ObjectType = ['Group']
            ObjectName = [f'<{myBridgeObjName}>ALL']
            Age = [3000]
            MyType = ['']
            MyName = ['']
            SF = [float()]
            ret    =    SapModel.LoadCases.StaticNonlinearStaged.SetStageData_2('Retração    e
fluência', Stage, NumberOperations, Operation, ObjectType, ObjectName, Age, MyType, MyName, SF)

            # Combinações de ações (definir depois de retração e fluência)

            ws_long = wb_long.sheets["5 Esforços solicitantes"]
            CI = ws_long['F190'].value # Coeficiente de impacto

            # Defensas + pav + recap
            Name = 'G3' # The name of a new load combination.
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Defensas', 'Pav + recap']
            for loadcase in CName:
                CNameType = eCNameType(0)
                SF = 1
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            # Carga permanente
            Name = 'Carga permanente' # The name of a new load combination.
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)
            ret = SapModel.RespCombo.SetCaseList(Name, eCNameType(0), 'DEAD', 1)
            ret = SapModel.RespCombo.SetCaseList(Name, eCNameType(1), 'G3', 1)

            # TB-450

            Name = 'TB-450' # The name of a new load combination.
            ComboType = 0
            # This is 0, 1, 2, 3 or 4 indicating the load combination type.
            # 0 = Linear Additive
            # 1 = Envelope
            # 2 = Absolute Additive
            # 3 = SRSS
            # 4 = Range Additive
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Veiculo', 'Multidao']
            for loadcase in CName:
                CNameType = eCNameType(0)
                # This is one of the following items in the eCNameType enumeration:
                # LoadCase = 0
                # LoadCombo = 1
                # The name of the load case or load combination to be added to or modified in
the combination specified by the Name item.
                # If the load case or combination already exists in the combination specified
by the Name item, the scale factor is modified as indicated by the SF item for that load case
or combination.
                #If the analysis case or combination does not exist in the combination specified
by the Name item, it is added.
                SF = 1 # The scale factor multiplying the case or combination indicated by the
CName item.
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            # ENVOLTÓRIA TEMPERATURA

            Name = 'ENV-TEMP' # The name of a new load combination.
            ComboType = 1
```

```
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Temperatura uniforme pos', 'Temperatura uniforme neg', 'Temperatura
gradiente pos', 'Temperatura gradiente neg']
            for loadcase in CName:
                CNameType = eCNameType(0)
                SF = 1
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)


            # CRIAÇÃO DO DICIONÁRIO PARA REALIZAR AS COMBINAÇÕES -> o primeiro valor da lista
referente ao ELU é quando a ação é a AVP

            Comb_dict = {
                        'Carga permanente': {'Type': 'Load Combo', 'ELU' : [1.35], 'ELU-fad' :
1, 'ELS-W': 1, 'EXC' : 1.1},
                        'Empuxo terra': {'Type': 'Load Case', 'ELU' : [1.35], 'ELU-fad' : 1,
'ELS-W': 1, 'EXC' : 1.1},
                        'Retração e fluência': {'Type': 'Load Case', 'ELU' : [1.2], 'ELU-fad' :
0, 'ELS-W': 0, 'EXC' : 0},
                        'Protensão': {'Type': 'Load Case', 'ELU' : [1.2], 'ELU-fad' : 0, 'ELS-
W': 0, 'EXC' : 0},
                        'TB-450': {'Type': 'Load Combo', 'ELU' : [CI * 1.5, CI * 1.05], 'ELU-
fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                        'Frenagem e aceleracao': {'Type': 'Load Case', 'ELU' : [CI * 1.5, CI *
1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                        'Empuxo terra carga móvel': {'Type': 'Load Case', 'ELU' : [CI * 1.5,
CI * 1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                        'Vento': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-fad' : 0,
'ELS-W': 0, 'EXC' : 0.6},
                        'Água': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-fad' : 0, 'ELS-
W': 0, 'EXC' : 0.6},
                        'ENV-TEMP': {'Type': 'Load Combo', 'ELU' : [1.2, 0.72], 'ELU-fad' :
0.3, 'ELS-W': 0.3, 'EXC' : 0.6},
                        'Impacto': {'Type': 'Load Case', 'ELU' : 0, 'ELU-fad' : 0, 'ELS-W': 0,
'EXC' : 1}
                        }

            AV = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento',
'Água', 'ENV-TEMP'] # Lista das ações variáveis

            Name = 'COMB1 - ELU (CM)' # Carga móvel como ação variável principal
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência', 'Protensão',
'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP']
            AVP = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase not in AV
else Comb_dict[loadcase]['ELU'][1]
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            Name = 'COMB2 - ELU (VENTO)' # Vento como ação variável principal
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência', 'Protensão',
'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP']
            AVP = ['Vento']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase not in AV
else Comb_dict[loadcase]['ELU'][1]
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            Name = 'COMB3 - ELU (ÁGUA)' # Ação da água como AVP
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência', 'Protensão',
'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP']
```

```
            AVP = ['Água']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase not in AV
else Comb_dict[loadcase]['ELU'][1]
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            Name = 'COMB4 - ELU (TEMPERATURA)' # Temperatura como AVP
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência', 'Protensão',
'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP']
            AVP = ['ENV-TEMP']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase not in AV
else Comb_dict[loadcase]['ELU'][1]
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            Name = 'COMB5 - ELU-fad' # Combinação de fadiga
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'TB-450', 'Frenagem e aceleracao',
'Empuxo terra carga móvel', 'ENV-TEMP']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['ELU-fad']
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            Name = 'COMB6 - ELS-W' # Combinação de estado limite de abertura de fissuras
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'TB-450', 'Frenagem e aceleracao',
'Empuxo terra carga móvel', 'ENV-TEMP']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['ELS-W']
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            Name = 'COMB7 - EXC' # Combinação excepcional
            ComboType = 0
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Carga permanente', 'Empuxo terra', 'TB-450', 'Frenagem e aceleracao',
'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP', 'Impacto']
            AVP = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel']
            for loadcase in CName:
                CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
                SF = Comb_dict[loadcase]['EXC']
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            # Envoltória ELU
            Name = 'ENV - ELU' # The name of a new load combination.
            ComboType = 1
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['COMB1 - ELU (CM)', 'COMB2 - ELU (VENTO)', 'COMB3 - ELU (ÁGUA)', 'COMB4 -
ELU (TEMPERATURA)']
            for loadcase in CName:
                CNameType = eCNameType(1)
                SF = 1
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)


            NumberNames = int() # The number of vehicle class names retrieved by the program.
            MyName = []
```

```
            # This is a one-dimensional array of vehicle class names. The MyName array is
created as a dynamic, zero-based array by the API user: Dim MyName() as String.
            # The array is dimensioned to (NumberNames - 1) inside the program, filled with the
names, and returned to the API user.
            ret = MyBridgeModeler.Vehicle.GetNameList(NumberNames, MyName)
            NumberVehicles = ret[1]
            VehiclesList = list(ret[2])
            ValidVehiclesList = ['Veiculo', 'Multidao']
            for i in range (NumberVehicles):
                if VehiclesList[i] not in ValidVehiclesList:
                    ret = MyBridgeModeler.Vehicle.Delete(VehiclesList[i])




            # 12 Atribuição dos Encontros (cBMAbutmentAssign Methods)

            Elevation_StartAbutment = -h_laje -h_long - h_ap #The global Z elevation of the
substructure. (start abutment) -> Distância entre o topo da laje e o CG do aparelho de apoio
            ElevationAtLayoutLine_StartAbutment = -h_laje -h_long - h_ap/2 #The bearing
elevation (global Z) at the layout line. [L] -> Distância entre o topo da laje e a base do
aparelho de apoio
            Elevation_EndAbutment = Elevation_StartAbutment #The global Z elevation of the
substructure.
            ElevationAtLayoutLine_EndAbutement = ElevationAtLayoutLine_StartAbutment #The
bearing elevation (global Z) at the layout line. [L]

            # 12.1 Start Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 1
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'Start Abutment' #The support name. The support name cannot be blank
and it can not be the same as another abutment or bent support name
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
            ret                                                                          =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation, SupportName, AbutmentDirection, DiaphragmProperty, InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            if Tipo_analise == 'SUPERESTRUTURA' or Tipo_analise == 'MESOESTRUTURA':
                SubstructureType = 1
                AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the abutment
or bent property used for the substructure. [L]
            else:
                SubstructureType = 2
                AbutmentOrBentProperty = 'Apoios extremos'
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation, SubstructureType, AbutmentOrBentProperty, Elevation_StartAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
```

```
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
            NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
            ret    =    MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_StartAbutment,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            # 12.2 End Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 2
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'End Abutment'
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
            ret                                                                          =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            SubstructureType = 1
            AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the abutment or
bent property used for the substructure. [L]
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation,     SubstructureType,     AbutmentOrBentProperty,     Elevation_EndAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
            NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
```

```
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
            ret    =    MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_EndAbutement,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            ret = MyBridgeModeler.Abutment.Delete('BABT1')




            # 13 Schedule
            ret = SapModel.Scheduler.Add('Cronograma')

            Name = 'Cronograma'
            NumberTasks = 11
            Expanded = [False, True, True, True, True, True, True, True, False, True, True]
            IndentLevel = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1]
            TaskName  =  ['Superestrutura',  'Guide',  'Longarina',  'Protensão',  'Intervalo',
'Transversina  +  Laje',  'Defensa',  'Pavimentação',  'Pós-construção',  'Início  da  operação',
'Final']
            Duration = ['69', '1', '28', '1', '1', '10', '27', '1', '3028', '28', '3000']
            CompletionDate = [69, 1, 29, 30, 31, 41, 68, 69, 3097, 97, 3097]
            Dependencies = ['', '', '2', '3', '4', '5', '6', '7', '8', '', '10']
            Operation = ['', 'Add Guide Structure', 'Add Structure', 'Load Objects', '', 'Add
Structure', 'Load Objects', 'Load Objects', '', '', '']
            IncludeSW = [bool(), bool(), True, bool(), bool(), True, bool(), bool(), bool(),
bool(), bool()]
            ObjectType = ['', 'Group', 'Group', 'Group', '', 'Group', 'Group', 'Group', '', '',
'']
            ObjectName = ['', f'<{myBridgeObjName}>SLAB', f'<{myBridgeObjName}>ALL_BUT_SLAB',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '', f'<{myBridgeObjName}>SLAB', 'ALL', 'ALL', '', '', '']
            Details = ['', '', '', '', '', '', '', '', '', '', '']
            Age  =  [float(),  float(),  28,  float(),  float(),  3,  float(),  float(),  float(),
float(), float()]
            LoadType  =  ['',  '',  'Load  pattern',  'Load  pattern',  '',  'Load  pattern',  'Load
pattern', 'Load pattern', '', '', '']
            LoadName = ['', '', 'DEAD', 'Protensão', '', 'DEAD', 'Defensas', 'Pav + recap', '',
'', '']
            LoadSF = [float(), float(), 1, 1, float(), 1, 1, 1, float(), float(), float()]
            Solve  =  [False,  False,  False,  False,  False,  False,  False,  False,  False,  False,
False]
            SaveStiffness = [False, False, False, False, False, False, False, False, False,
False, False]
            OutputID = ['', 'GuideSlab', '', 'Protensão', '', 'Transversina + laje', 'Defensa',
'Pavimentação', '', 'Início da operação', 'Final']
            ItemType = ['', '', '', '', '', '', '', '', '', '', '']
            ItemName = ['', '', '', '', '', '', '', '', '', '', '']
            Notes = ['', '', '', '', '', '', '', '', '', '', '']
            ret    =    SapModel.Scheduler.SetScheduleData_1(Name,    NumberTasks,    Expanded,
IndentLevel, TaskName, Duration, CompletionDate, Dependencies, Operation, IncludeSW, ObjectType,
ObjectName,
            Details, Age, LoadType, LoadName, LoadSF, Solve, SaveStiffness, OutputID, ItemType,
ItemName, Notes)

            Name = 'Cronograma'
            Unloading = 'Unload Entire'
            GeoNonLin = 'P-Delta'
            MaxTotal = 100
            MaxNull = 200
            MaxIterCS = 200
            MaxIterNR = 200
            ItConvTol = 0.001
            UseEvStep = 'Yes'
            EvLumpTol = 0.01
            LSPerIter = 20
            LSTol = 0.1
            LSStepFact = 1.618
```

```
            StageSave = 'End of Final Stage'
            StageMinIns = int()
            StageMinTD = int()
            TimeDepMat = 'Yes'
            TFMaxIter = 100
            TFTol = 0.01
            TFAccelFact = 1
            TFNoStop = 'No'
            ret   =   SapModel.Scheduler.SetScheduleParameters(Name,   Unloading,   GeoNonLin,
MaxTotal, MaxNull, MaxIterCS, MaxIterNR, ItConvTol, UseEvStep, EvLumpTol, LSPerIter, LSTol,
LSStepFact,
            StageSave, StageMinIns, StageMinTD, TimeDepMat, TFMaxIter, TFTol, TFAccelFact,
TFNoStop)




            # 14 Carregamentos atuantes
            # 14.1 Bridge Object - Slab Wet Concrete Load Assigns
            Name = myBridgeObjName
            ConcretePourName = '' #The concrete pour name to be deleted. Enter a blank string
to delete all slab wet concrete load assignments.
            ret                                                                          =
MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.Delete(myBridgeObjName, '')

            # 14.2 Bridge Object - Temperature Load Assigns
            LoadPat = ['Temperatura gradiente pos', 'Temperatura gradiente neg'] # A zero-based
one-dimensional array of the load pattern used for each load assignment.
            TempGradientName = ['Temperatura gradiente', 'Temperatura gradiente'] # A zero-
based one-dimensional array of the name of the defined temperature gradient for each load
assignment. This item only applies when the TemperatureOption item is 1 (Temperature gradient).
            GradientType = [0, 1]
            # A zero-based one-dimensional array of the gradient type for each load assignment.
This item only applies when the TemperatureOption item is 1 (Temperature gradient).
            # 0 = Positive temperature change
            # 1 = Negative temperature change
            for i in range(NVãos):
                for j in range(2):
                    Name = myBridgeObjName
                    SpanName = f'Vão {i+1}' # A zero-based one-dimensional array of the bridge
span for each load assignment.
                    TemperatureOption = 1
                    # A zero-based one-dimensional array of the temperature option for each
load assignment.
                    # 0 = Uniform temperature change
                    # 1 = Temperature gradient
                    UniformTempChange = float() # A zero-based one-dimensional array of the
uniform temperature change for each load assignment. This item only applies when the
TemperatureOption item is 0 (Uniform temperature change). [T]
                    ret = MyBridgeModeler.BridgeObject.LoadAssignTemperature.SetOneLoad(Name,
SpanName,     LoadPat[j],     TemperatureOption,     UniformTempChange,     TempGradientName[j],
GradientType[j])

            # 14.3 Bridge Object - Line Load Assigns

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Vento' # The load pattern for the load assignment.
            LoadDistribution = 'Vento' #The named load distribution for the load assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
            TransverseVariation = 'None' # The variation used for the load assignment. This is
either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignLine.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, TransverseVariation)

            # 14.4 Bridge Object - Area Load Assigns

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Frenagem e aceleracao' # The load pattern for the load assignment.
            LoadDistribution = 'Frenagem e aceleracao' #The named load distribution for the
load assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
```

```
            LeftEdgeVariation = 'None' # The left edge variation used for the load assignment.
This is either None or the name of a defined variation.
            RightEdgeVariation = 'None' # The right edge variation used for the load assignment.
This is either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)




            # 15 Update no modelo
            Name = myBridgeObjName
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)




            # 16 Definição dos grupos comuns para pegar resultados da análise e modelagem dos
encontros

            # Grupos comuns
            Elementos = ['Longarinas', 'Transversinas']
            PropSecoes = [f'Longarina (C{f_ck})', f'Transversina de apoio (C{f_ck})']
            for i in range(len(Elementos)):
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.GroupDef.SetGroup(Elementos[i])
                ret = SapModel.SelectObj.PropertyFrame(PropSecoes[i])
                ret = SapModel.FrameObj.SetGroupAssign('', Elementos[i], False, eItemType(2))

            ret = SapModel.SelectObj.ClearSelection()

            # 16 Modelagem do encontro

            if Tipo_analise == 'ENCONTROS':

                # 16.1 Alas
                OffsetEncontro = 75

                # Ala da direita
                NumberPoints = 4 # The number of points in the area abject.
                x   =   [-OffsetEncontro,   -OffsetEncontro-CompA,   -OffsetEncontro-CompA,   -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                # Escolheu-se o sentido anti-horário, sendo o primeiro ponto o mais próximo da
laje (em x e z)
                # The coordinates should be ordered to run clockwise or counter clockwise around
the area object.
                Name = ''
                # This is the name that the program ultimately assigns to the area object. If
no UserName is specified, the program assigns a default name to the area object.
                # If a UserName is specified and that name is not used for another area object,
the UserName is assigned to the area object; otherwise a default name is assigned to the area
object.
                PropName = 'Encontro'
                # This is Default, None or the name of a defined area property.
                # If it is Default, the program assigns a default area property to the area
object.
                # If it is None, no area property is assigned to the area object. If it is the
name of a defined area property, that property is assigned to the area object.
                UserName = 'Ala direita 1/2' # This is an optional user specified name for the
area object. If a UserName is specified and that name is already used for another area object,
the program ignores the UserName.
```

```
                CSys = 'GLOBAL' # The name of the coordinate system in which the area object
point coordinates are defined.
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro-CompA, -OffsetEncontro-ChanfroH, -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala direita 2/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala direita 1/2' # The name of an existing area object.
                MeshType = 1
                # This item is 1, 2, 3, 4, 5 or 6, indicating the mesh type for the area object.
                # 1 = Mesh area into a specified number of objects
                # 2 = Mesh area into objects of a specified maximum size
                # 3 = Mesh area based on points on area edges
                # 4 = Cookie cut mesh area based on lines intersecting edges
                # 5 = Cookie cut mesh area based on points
                # 6 = Mesh area using General Divide Tool
                AreaName = [] # This is an array of the name of each area object created when
the specified area object is divided.
                n1_aladireita1 = 10 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 2.
                n2_aladireita1 = 5 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 3.
                NumberAreas = n1_aladireita1*n2_aladireita1 #The number of area objects created
when the specified area object is divided.
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_aladireita1, n2_aladireita1)
                Ala_direita = list(ret[2])

                Name = 'Ala direita 2/2'
                MeshType = 1
                AreaName = []
                n1_aladireita2 = 10
                n2_aladireita2 = 10
                NumberAreas = n1_aladireita2*n2_aladireita2
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_aladireita2, n2_aladireita2)
                Ala_direita.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Ala direita') # This is the name of a group.
If this is the name of an existing group, that group is modified, otherwise a new group is
added.
                for Area in Ala_direita:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala direita' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # Ala da esquerda
```

```
                NumberPoints = 4
                x   =   [-OffsetEncontro,   -OffsetEncontro-CompA,   -OffsetEncontro-CompA,   -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 1/2'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-ChanfroH,  -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 2/2'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala esquerda 1/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda1 = 10
                n2_alaesquerda1 = 5
                NumberAreas = n1_alaesquerda1*n2_alaesquerda1
                ret   =   SapModel.EditArea.Divide(Name,   MeshType,   NumberAreas,   AreaName,
n1_alaesquerda1, n2_alaesquerda1)
                Ala_esquerda = list(ret[2])

                Name = 'Ala esquerda 2/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda2 = 10
                n2_alaesquerda2 = 10
                NumberAreas = n1_alaesquerda2*n2_alaesquerda2
                ret   =   SapModel.EditArea.Divide(Name,   MeshType,   NumberAreas,   AreaName,
n1_alaesquerda2, n2_alaesquerda2)
                Ala_esquerda.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Ala esquerda') # This is the name of a group.
If this is the name of an existing group,  that group is modified, otherwise a new group is
added.
                for Area in Ala_esquerda:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala esquerda' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 16.2 Parede cortina
                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
```

```
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, -ChanfroV, -ChanfroV, 0]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 1/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -AlturaM, -AlturaM, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 2/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Discretização
                Name = 'Cortina 1/2'
                MeshType = 1
                AreaName = []
                n1_cortina1 = 5
                n2_cortina1 = 20
                NumberAreas = n1_cortina1*n2_cortina1
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_cortina1, n2_cortina1)
                Cortina = list(ret[2])

                Name = 'Cortina 2/2'
                MeshType = 1
                AreaName = []
                n1_cortina2 = 10
                n2_cortina2 = 20
                NumberAreas = n1_cortina2*n2_cortina2
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_cortina2, n2_cortina2)
                Cortina.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Cortina')
                for Area in Cortina:
                    GroupName = 'Cortina'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 16.3 Console
                NumberPoints = 4
                Largura_console = 30
                x = [-OffsetEncontro, -OffsetEncontro-Largura_console, -OffsetEncontro-
Largura_console, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla, -TotalWidthOfDeckSection/2 + hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1), -
2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1)] # Convencionado dessa forma para que
a discretização fique robusta
                Name = ''
                PropName = 'Encontro'
                UserName = 'Console'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Determinação dos pontos para realizar a discretização
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Cortina', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_cortina = list(ret[3])
                Points_cortina = []
                for Area in Areas_cortina:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
```

```
                    Points_cortina.extend(list(ret[2]))

                Pointscortina_coord = []
                for point in Points_cortina:
                    ret   =   SapModel.PointObj.GetCoordCartesian(point,   float(),   float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointscortina_coord.append(ret)

                for i in range(len(Pointscortina_coord)):
                    Pointscortina_coord[i].insert(0,Points_cortina[i])

                df_pontos_cortina = pd.DataFrame(Pointscortina_coord, columns = ['Joint', 'X',
'Y', 'Z'])
                df_pontos_cortina = df_pontos_cortina.round(4)

                Pointscortina_mesh    =    df_pontos_cortina[(df_pontos_cortina["Z"]    ==    -
2*(ChanfroV/n2_aladireita1))]['Joint'].unique().tolist()

                ret = SapModel.SelectObj.ClearSelection()
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização
                Name = 'Console'
                MeshType = 3
                AreaName = []
                n1_console = 2
                n2_console = 20
                NumberAreas = n1_console*n2_console
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)

                Console = list(ret[2])
                ret = SapModel.GroupDef.SetGroup('Console') # This is the name of a group.  If
this is the name of an existing group,  that group is modified, otherwise a new group is added.
                for Area in Console:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName  =  'Console'  #  The  name  of  an  existing  group  to  which  the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 16.4 Laje de transição
                OffsetLT = OffsetEncontro + Largura_console
                NumberPoints = 4
                x = [-OffsetLT, -OffsetLT - LarguraLT, -OffsetLT - LarguraLT, -OffsetLT]
                y  =  [-TotalWidthOfDeckSection/2  +  hAla,  -TotalWidthOfDeckSection/2  +  hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-h_laje/2, -h_laje/2, -h_laje/2, -h_laje/2]
                Name = ''
                PropName = 'Laje de transição'
                UserName = 'LT'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Obtenção dos pontos de discretização
                ret = SapModel.SelectObj.ClearSelection()
                Pointscortina_mesh = Pointscortina_mesh[1:-1]
```

```
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                dx = - float(Largura_console)
                dz = 2*(ChanfroV/n2_aladireita1) - h_laje/2
                ret = SapModel.EditGeneral.ReplicateLinear(dx, float(0), dz, int(1), int(), [],
[], False)

                PointsLT_mesh = list(ret[2])
                ret = SapModel.SelectObj.ClearSelection()
                for point in PointsLT_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização nos pontos do console
                Name = 'LT'
                MeshType = 3
                AreaName = []
                n1_LT = 10
                n2_LT = 20
                NumberAreas = n1_LT*n2_LT
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)
                LT = list(ret[2])

                # Discretização LT
                LT_adic = []
                for Area in LT:
                    ret = SapModel.EditArea.Divide(Area, 1, int(), AreaName, 10, 1)
                    LT_adic.extend(list(ret[2]))

                LT.extend(LT_adic)
                ret = SapModel.GroupDef.SetGroup('LT') # This is the name of a group.  If this
is the name of an existing group,  that group is modified, otherwise a new group is added.

                for Area in LT:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'LT' # The name of an existing group to which the assignment
is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 16.5 Links laje de transição
                XMin = -OffsetLT - LarguraLT
                XMax = -OffsetLT - LarguraLT/2
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = False
                Line = False
                Area = True
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                Name = '' # The name of an existing area object or group, depending on the
value of the ItemType item.
```

```
                MyType = 1
                # This is either 1 or 2, indicating the spring property type.
                # 1 = Simple spring
                # 2 = Link property
                s = 0.05 # The simple spring stiffness per unit area of the specified area
object face. This item applies only when MyType = 1. [F/L3]
                SimpleSpringType = 2
                # This is 1, 2 or 3, indicating the simple spring type. This item applies only
when MyType = 1.
                # 1 = Spring resists tension and compression
                # 2 = Spring resists compression only
                # 3 = Spring resists tension only
                LinkProp = '' # The name of the link property assigned to the spring. This item
applies only when MyType = 2.
                Face = -2
                # This is -1, -2 or a nonzero, positive integer indicating the area object face
to which the specified spring assignment applies.
                # -1 = Bottom face
                # -2 = Top face
                # >0 = Edge face
                SpringLocalOneType = 2
                # This is 1, 2 or 3, indicating the method used to specify the spring positive
local 1-axis orientation.
                # 1 = Parallel to area object local axis
                # 2 = Normal to specified area object face
                # 3 = User specified direction vector
                Dir = int() # This is 1, 2, 3, -1, -2 or -3, indicating the area object local
axis that corresponds to the positive local 1-axis of the spring. This item applies only when
SpringLocalOneType = 1.
                Outward = False # This item is True if the spring positive local 1 axis is
outward from the specified area object face. This item applies only when SpringLocalOneType =
2.
                Vec = [] # This is an array of three values that define the direction vector
of the spring positive local 1-axis.
                # The direction vector is in the coordinate system specified by the CSys item.
This item applies only when SpringLocalOneType = 3.
                Ang = float() # This is the angle that the link local 2-axis is rotated from
its default orientation. This item applies only when MyType = 2. [deg]
                Replace = True #If this item is True, all existing spring assignments to the
area object are removed before assigning the specified spring.
                # If it is False, the specified spring is added to any existing springs already
assigned to the area object.
                CSys = 'Local' #This is Local (meaning the area object local coordinate system)
or the name of a defined coordinate system.
                #This item is the coordinate system in which the user specified direction
vector, Vec, is specified. This item applies only when SpringLocalOneType = 3.
                ItemType = eItemType(2)
                ret = SapModel.AreaObj.SetSpring(Name, MyType, s, SimpleSpringType, LinkProp,
Face, SpringLocalOneType, Dir, Outward, Vec, Ang, Replace, CSys, ItemType)

                ret = SapModel.SelectObj.ClearSelection()

                # 16.6 Links console
                XMin = -OffsetLT
                XMax = -OffsetLT
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = True
                Line = False
                Area = False
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])

                Points_linkconsole = list(ret[3])
                x = []
                y = []
                z = []
```

```
                for point in Points_linkconsole:
                    ret   =   SapModel.PointObj.GetCoordCartesian(point,   float(),   float(),
float(), 'GLOBAL')
                    x.append(ret[1])
                    y.append(ret[2])
                    z.append(ret[3])

                ret = SapModel.SelectObj.ClearSelection()
                Links_console = []
                for i in range(len(Points_linkconsole)):
                    xi = x[i] # The coordinates of the I-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item.
                    yi = y[i]
                    zi = z[i]
                    xj = x[i] # The coordinates of the J-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item. These coordinates are ignored
if the IsSingleJoint item is True.
                    yj = y[i]
                    zj = -2*(ChanfroV/n2_aladireita1)
                    Name = ''
                    # This is the name that the program ultimately assigns for the link object.
                    # If no UserName is specified, the program assigns a default name to the
link object.
                    # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                    IsSingleJoint = False # This item is True if a one-joint link is added and
False if a two-joint link is added.
                    PropName = 'Link rigido (esforços)'
                    UserName = ''
                    CSys = 'GLOBAL'
                    ret   =   SapModel.LinkObj.AddByCoord(xi,   yi,   zi,   xj,   yj,   zj,   Name,
IsSingleJoint, PropName, UserName, CSys)
                    Links_console.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links console')
                for link in Links_console:
                    GroupName = 'Links console'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 16.7 Links travessa # Nao gostei muito dessa solução, investigar outra forma
de fazer
                Name = f'Travessa encontro (C{f_ck})'
                DeSelect = False
                ret = SapModel.SelectObj.PropertyFrame(Name, DeSelect)

                NumberItems = int() # The number of selected objects.
                ObjectType = []
                # This is an array that includes the object type of each selected object.
                # 1 = Point object
                # 2 = Frame object
                # 3 = Cable object
                # 4 = Tendon object
                # 5 = Area object
                # 6 = Solid object
                # 7 = Link object
                ObjectName = [] # This is an array that includes the name of each selected
object.
                ret = SapModel.SelectObj.GetSelected(NumberItems, ObjectType, ObjectName)

                Name = f'{list(ret[3])[0]}'
                Point1 = ''
                Point2 = ''
                ret = SapModel.FrameObj.GetPoints(Name, Point1, Point2)

                Name = ret[1]
                x = float()
                y = float()
                z = float()
                CSys = 'GLOBAL'
                ret = SapModel.PointObj.GetCoordCartesian(Name, x, y, z, CSys)
                xiTravessa = ret[1]
                yiTravessa = ret[2]
```

```
                ziTravessa = ret[3]

                zi = [-AlturaM, -AlturaM + ((AlturaM - ChanfroV)/(n1_cortina2))* 6]
                Links_travessa = []
                for j in range (2):
                    for i in range (n2_cortina2+1):
                        xi = -OffsetEncontro # The coordinates of the I-End of the added link
object. The coordinates are in the coordinate system defined by the CSys item.
                        yi     =     -TotalWidthOfDeckSection/2     +     hAla/2     -     i*(-
TotalWidthOfDeckSection/2 + hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                        xj = xiTravessa # The coordinates of the J-End of the added link object.
The coordinates are in the coordinate system defined by the CSys item. These coordinates are
ignored if the IsSingleJoint item is True.
                        yj     =     -TotalWidthOfDeckSection/2     +     hAla/2     -     i*(-
TotalWidthOfDeckSection/2 + hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                        zj = ziTravessa
                        Name = ''
                        # This is the name that the program ultimately assigns for the link
object.
                        # If no UserName is specified, the program assigns a default name to
the link object.
                        # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                        IsSingleJoint = False # This item is True if a one-joint link is added
and False if a two-joint link is added.
                        PropName = 'Link rigido'
                        UserName = ''
                        CSys = 'GLOBAL'
                        ret = SapModel.LinkObj.AddByCoord(xi, yi, zi[j], xj, yj, zj, Name,
IsSingleJoint, PropName, UserName, CSys)
                        Links_travessa.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links travessa')
                for link in Links_travessa:
                    GroupName = 'Links travessa'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 16.8 Definição dos grupos

                ret = SapModel.GroupDef.SetGroup('Encontro')

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Links console')
                ret = SapModel.SelectObj.Group('Links travessa')
                ret = SapModel.LinkObj.SetGroupAssign('', 'Encontro', False, eItemType(2))

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Ala direita')
                ret = SapModel.SelectObj.Group('Ala esquerda')
                ret = SapModel.SelectObj.Group('Console')
                ret = SapModel.SelectObj.Group('Cortina')
                ret = SapModel.SelectObj.Group('LT')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Encontro', False, eItemType(2))

                ret = SapModel.GroupDef.SetGroup('Todas lajes')
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Laje de transição')
                ret = SapModel.SelectObj.Group(f'<{myBridgeObjName}>SLAB')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Todas lajes', False, eItemType(2))

                # 16.9 Atribuição dos carregamentos nos encontros

                ret = SapModel.SelectObj.ClearSelection()
                EmpuxoTerra = ws_pil_ext['D82'].value/(100)**2
                EmpuxoTerraCargaMóvel = ws_pil_ext['D93'].value/(100)**2
                Cases = [['Ala direita', 'Empuxo terra', -EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
                ['Ala direita', 'Empuxo terra carga móvel', -EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Ala esquerda', 'Empuxo terra', EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
```

```
                ['Ala esquerda', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra', EmpuxoTerra, 4, False, 'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 4, False,
'GLOBAL', eItemType(1)]]

                for case in Cases:
                    Name, LoadPat, Value, Dir, Replace, CSys, ItemType = case
                    ret = SapModel.AreaObj.SetLoadUniform(Name, LoadPat, Value, Dir, Replace,
CSys, ItemType)

                # 16.10 Impacto na defensa

                # 16.10.1 Obtenção dos pontos do encontro para a atribuição do carregamento

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Encontro', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_encontro = list(ret[3])
                Points_encontro = [] # Nomes dos pontos que definem as áreas
                for Area in Areas_encontro:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_encontro.extend(list(ret[2]))

                Pointsencontro_coord = []
                for point in Points_encontro:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointsencontro_coord.append(ret)

                for i in range(len(Pointsencontro_coord)):
                    Pointsencontro_coord[i].insert(0,Points_encontro[i]) # Inserção dos nomes
dos pontos em cada lista

                # 16.10.2 Criação do dataframe com todos os elementos de área e os quatro pontos
que os definem (start e end station será feito depois de discretizar a laje)
                df_pontos_encontro = pd.DataFrame(Pointsencontro_coord, columns = ['Joint',
'X', 'Y', 'Z']) # DataFrame com todas as áreas, sendo cada linha um ponto que a define
                df_pontos_encontro = df_pontos_encontro.round(4) # Arredondar todas as colunas
do DataFrame para 4 casas decimais

                # 16.10.3 Obtenção dos pontos

                Points_impacto_esq = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) &
(df_pontos_encontro["Y"] == -TotalWidthOfDeckSection/2 + hAla/2) & (df_pontos_encontro["X"] <=
-OffsetEncontro)]['Joint'].unique().tolist()
                Points_impacto_dir = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) &
(df_pontos_encontro["Y"] == TotalWidthOfDeckSection/2 - hAla/2) & (df_pontos_encontro["X"] <= -
OffsetEncontro)]['Joint'].unique().tolist()

                # 16.10.4 Atribuição do carregamento
                fdist_impacto = 100/(50 + 2*87) # em kN/cm
                fno_impacto = (fdist_impacto * CompA)/len(Points_impacto_esq)

                mdist_impacto = 100*87/(50 + 2*87) # em kN.cm/cm
                mno_impacto = mdist_impacto*CompA/len(Points_impacto_esq)

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_esq:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , fno_impacto, 0, -
mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_dir:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , -fno_impacto, 0,
mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))
```

```
            # 17 Definição das Lanes - (cBMLane.SetLaneFromLayoutLine) [myLaneName sempre será a faixa de tráfego]

            Name = myLaneName #The name for the existing lane defined from a layout line.
            NumberLaneStations = 2 #The number of lane points, that is, the number of items included in the that lane will be overwritten. LayoutLine, Station, Offset, Width and Radius arrays.
            LayoutLine = [myLayoutLineName, myLayoutLineName]#A zero-based array containing the name of the bridge layout line associated with each lane station.
            Station = [0, CTotal]#A zero-based array containing the station at each lane station. [L]
            Offset = [0]*2 #A zero-based array containing the offset from the layout line to the center of the lane at each lane station. [L]
            Width = [LarguraRolamento]*2 #A zero-based array containing the the width of the lane at each lane station. [L]
            Radius = [0]*2 #A zero-based array containing the the radius of the lane at each lane station. A positive radius curves to the right and negative to the left. [L]
            LaneType = 0
            #The lane type (fixed or floating).
            # 0 = Fixed lane
            # 1 = Floating lane set
            if Tipo_analise == 'ENCONTROS':
                LoadGroup = 'Todas lajes'
            else:
                LoadGroup = f'<{myBridgeObjName}>SLAB'
            LeftEdgeIsExterior = False #Indicates whether the left edge of the lane is exterior or interior.
            RightEdgeIsExterior = False #Indicates whether the right edge of the lane is exterior or interior.
            MaxDiscAlongLane = 300 #The maximum lane load discretization length along the lane. [L]
            MaxDiscAcrossLane = 300 #The maximum lane load discretization length across the lane. [L]
            UseDiscFactorSpan = True #Indicates whether the the DiscFactorSpan item is to apply to the lane.
            UseDiscFactorLane = True #Indicates whether the the DiscFactorLane item is to apply to the lane.
            DiscFactorSpan = 4 #The lane load discretization length along the lane will not be more than 1/x of the span length where x is the DiscFactorSpan value.
            #This item is only applicable when the UseDiscFactorSpan item is True.
            DiscFactorLane = 10 #The lane load discretization length along the lane will not be more than 1/x of the lane length where x is the DiscFactorLane value.
            #This item is only applicable when the UseDiscFactorLane item is True.
            Color = int()#The color assigned to the lane. (Optional)
            Notes = ''#The notes, if any, assigned to the lane. (Optional)
            ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine(Name, NumberLaneStations, LayoutLine, Station, Offset, Width, Radius, LaneType, LoadGroup, LeftEdgeIsExterior, RightEdgeIsExterior,
            MaxDiscAlongLane, MaxDiscAcrossLane, UseDiscFactorSpan, UseDiscFactorLane, DiscFactorSpan, DiscFactorLane, Color, Notes)

            if Tipo_analise == 'ENCONTROS':
                ret = MyBridgeModeler.LayoutLine.AddNew('BLL0')

                Name = 'BLL0'
                CoordinateSystem = 'GLOBAL'
                StartX = -1000
                StartY = 0
                StartZ = 0
                ret = MyBridgeModeler.LayoutLine.SetGeneralData(Name, CoordinateSystem, StartX, StartY, StartZ)

                Name = 'BLL0'
                NumberSegments = 2
                SegmentType = [1, 2]
                Station = [0, 1000]
                Radius = [float(), float()]
                Bearing = ['N900000E', 'N900000E']
                ret = MyBridgeModeler.LayoutLine.SetHorizontalLayoutData(Name, NumberSegments, SegmentType, Station, Radius, Bearing)

                ret = MyBridgeModeler.Lane.AddNewFromLayoutLine('LANE0', 'BLL0')
                ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine('LANE0', 2, ['BLL0', 'BLL0'], [0, 1000], [0]*2, [LarguraRolamento]*2, [0]*2, LaneType, LoadGroup, LeftEdgeIsExterior, RightEdgeIsExterior,
```

```
                MaxDiscAlongLane,  MaxDiscAcrossLane,  UseDiscFactorSpan,  UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

                LoadNumber = 1 #The load assignment number.
                NumberItems = 2 #The number of lanes loaded for the specified load assignment
number.
                MyName = [f'{myLaneName}', 'LANE0'] #This is an array that includes the name
of each lane loaded for the specified load assignment number.
                ret    =    SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo',    LoadNumber,
NumberItems, MyName)
                ret    =    SapModel.LoadCases.Moving.SetLanesLoaded('Multidao',    LoadNumber,
NumberItems, MyName)

            if Tipo_analise != 'MESOESTRUTURA':
                ret = SapModel.File.Save()



            # 18 Alimentação dos dados iniciais das planilhas de dimensionamento das longarinas
            ws_long = wb_long.sheets["5 Esforços solicitantes"]

            G_3 = 2*0.203*25/NLongarinasPorVão # Defensas
            G_4       =       (0.1*(TotalWidthOfDeckSection/100       -       2*0.40)*24       +
2*(TotalWidthOfDeckSection/100 - 2*0.40))/NLongarinasPorVão # Pavimentação + recap
            g3 = G_3 + G_4
            ws_long.range('C28').value = g3
            ws_long.range('C29').value = (TotalWidthOfDeckSection)/100*2/NLongarinasPorVão




        # 19 Atribuição de carregamentos nos elementos
            if Tipo_analise != 'SUPERESTRUTURA':
                # 19.1 Pressão da água nos pilares
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyFrame(f'Pilar (C{f_ck})')
                modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
                wb_pil_ext = xw.Book(modelo_pil_ext)
                ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

                Name = '' # The name of an existing frame object or group, depending on the
value of the ItemType item.
                LoadPat = 'Água'
                MyType = 1
                # This is 1 or 2, indicating the type of distributed load.
                # 1 = Force per unit length
                # 2 = Moment per unit length
                Dir = 5
                # This is an integer between 1 and 11, indicating the direction of the load.
                # 1 = Local 1 axis (only applies when CSys is Local)
                # 2 = Local 2 axis (only applies when CSys is Local)
                # 3 = Local 3 axis (only applies when CSys is Local)
                # 4 = X direction (does not apply when CSys is Local)
                # 5 = Y direction (does not apply when CSys is Local)
                # 6 = Z direction (does not apply when CSys is Local)
                # 7 = Projected X direction (does not apply when CSys is Local)
                # 8 = Projected Y direction (does not apply when CSys is Local)
                # 9 = Projected Z direction (does not apply when CSys is Local)
                # 10 = Gravity direction (only applies when CSys is Global)
                # 11 = Projected Gravity direction (only applies when CSys is Global)
                Dist1 = 0
                # This is the distance from the I-End of the frame object to the start of the
distributed load.
                # This may be a relative distance (0 <= Dist1 <= 1) or an actual distance,
depending on the value of the RelDist item. [L] when RelDist is False
                Dist2 = 1
                # This is the distance from the I-End of the frame object to the end of the
distributed load.
                # This may be a relative distance (0 <= Dist2 <= 1) or an actual distance,
depending on the value of the RelDist item. [L] when RelDist is False
                Val1 = ws_pil_ext['D57'].value/100 # This is the load value at the start of the
distributed load. [F/L] when MyType is 1 and [FL/L] when MyType is 2
                Val2 = Val1 # This is the load value at the end of the distributed load. [F/L]
when MyType is 1 and [FL/L] when MyType is 2
```

```
                CSys = 'GLOBAL' # This is Local or the name of a defined coordinate system. It
is the coordinate system in which the loads are specified.
                RelDist = True # If this item is True, the specified Dist item is a relative
distance, otherwise it is an actual distance.
                Replace = False # If this item is True, all previous distributed loads, if any,
assigned to the specified frame object(s), in the specified load pattern, are deleted before
making the new assignment.
                ItemType = eItemType(2)
                # This is one of the following items in the eItemType enumeration:
                # Object = 0
                # Group = 1
                # SelectedObjects = 2
                # If this item is Object, the assignment is made to the frame object specified
by the Name item.
                # If this item is Group, the assignment is made to all frame objects in the
group specified by the Name item.
                # If this item is SelectedObjects, assignment is made to all selected frame
objects, and the Name item is ignored.
                ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType, Dir, Dist1,
Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)

                ret = SapModel.SelectObj.ClearSelection()

                # 19.2 Empuxo de terra nos pilares

                ret = SapModel.SelectObj.ClearSelection()

                XMin = -1 # The maximum and minimum coordinates of the selection box in the
specified coordinate system.
                XMax = 1
                YMin = -TotalWidthOfDeckSection
                YMax = TotalWidthOfDeckSection
                ZMin = -1000
                ZMax = -400
                DeSelect = False # The item is False if objects are to be selected and True if
they are to be deselected.
                CSys = 'GLOBAL' # The name of the coordinate system in which XMin, XMax, YMin,
YMax, ZMin and ZMax are specified.
                IncludeIntersections = True
                # When this item is True, objects that are inside the box or intersecting the
sides and edges of the box are selected or deselected.
                # When this item is False, only objects that are fully inside the box are
selected or deselected.
                Point = False # Point objects that fall inside the box are only selected or
deselected when this item is True.
                Line = True
                Area = False
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                Name = ''
                LoadPat = 'Empuxo terra'
                MyType = 1
                Dir = 4
                Dist1 = 0
                Dist2 = 1
                Val1 = ws_pil_ext['D115'].value/100
                Val2 = Val1
                CSys = 'GLOBAL'
                RelDist = True
                Replace = False
                ItemType = eItemType(2)
                # This is one of the following items in the eItemType enumeration:
                # Object = 0
                # Group = 1
                # SelectedObjects = 2
                # If this item is Object, the assignment is made to the frame object specified
by the Name item.
                # If this item is Group, the assignment is made to all frame objects in the
group specified by the Name item.
                # If this item is SelectedObjects, assignment is made to all selected frame
objects, and the Name item is ignored.
```

```
                ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType, Dir, Dist1,
Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)

                ret = SapModel.SelectObj.ClearSelection()




            # 20 Pré-análise: organização e filtragem dos elementos para retirada de esforços
            # 20.1 Obtenção de todos os elementos de área e as coordenadas dos 4 pontos que os
definem

            if Tipo_analise == "SUPERESTRUTURA":
                ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar os dados em kN, kNm e
Celsius
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyMaterial(f'C{f_ck} - Tabuleiro', False)
                ret = SapModel.FrameObj.SetSelected('', False, eItemType(2))
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_list = list(ret[3])
                Areas_points = [] # Nomes dos pontos que definem as áreas
                for Area in Areas_list:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Areas_points.extend(list(ret[2]))

                Areas_pointscoord = []
                for point in Areas_points:
                    ret   =   SapModel.PointObj.GetCoordCartesian(point,   float(),   float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Areas_pointscoord.append(ret)

                for i in range(len(Areas_list)):
                    for j in range(4):
                        Areas_pointscoord[i*4+j].insert(0, Areas_list[i]) # Inserção dos nomes
das áreas em cada lista

                # 20.2 Criação do dataframe com todos os elementos de área e os quatro pontos
que os definem (start e end station será feito depois de discretizar a laje)
                df_pontos_lajes_temp = pd.DataFrame(Areas_pointscoord, columns = ['Area', 'X',
'Y', 'Z']) # DataFrame com todas as áreas, sendo cada linha um ponto que a define
                df_pontos_lajes_temp  =  df_pontos_lajes_temp.round(4)  #  Arredondar  todas  as
colunas do DataFrame para 4 casas decimais

                # 20.3 Definição dos elementos de extremidade para posterior discretização
                df_pontos_lajes_temp_min       =       df_pontos_lajes_temp.groupby(['Area'],
as_index=False).min()
                df_pontos_lajes_temp_max       =       df_pontos_lajes_temp.groupby(['Area'],
as_index=False).max()
                df_pontos_lajes_temp_join                                                    =
df_pontos_lajes_temp_min.merge(df_pontos_lajes_temp_max, on = 'Area')
                df_pontos_lajes_temp_join    =    df_pontos_lajes_temp_join.rename(columns    =
{"X_x" : "X_min", "Y_x" : "Y_min", "Z_x" : "Z_min", "X_y" : "X_max", "Y_y" : "Y_max", "Z_y" :
"Z_max" })

                Areas_lajes_extremidades_temp                                                =
df_pontos_lajes_temp_join[(df_pontos_lajes_temp_join["X_min"]    >=    StartX/100    +    1)    &
(df_pontos_lajes_temp_join["X_min"] <= StartX/100 + 2)]['Area'].unique().tolist()

                # 20.4 Discretização mais refinada da região das áreas de extremidade
                ret = SapModel.SelectObj.ClearSelection()
                for Area in Areas_lajes_extremidades_temp:
                    ret = SapModel.EditArea.Divide(Area, 2, int(), [], int(), int(), 10, 25,
bool(), bool(), bool() , bool() , float() , float(), bool() , bool() , bool(), bool())

                # 20.5 Coordenadas das longarinas (Pegar todos os pontos que definem as barras
das longarinas)

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyMaterial(f'C{f_ck} - Longarina', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Longarinas_list = list(ret[3])
                Longarinas_points = [] # Nomes dos pontos que definem as longarinas
```

```
                for Longarina in Longarinas_list:
                    ret = SapModel.FrameObj.GetPoints(Longarina, '', '')
                    Longarinas_points.append(ret[1])
                    Longarinas_points.append(ret[2])

                Longarinas_pointscoord = []
                for point in Longarinas_points:
                    ret   =   SapModel.PointObj.GetCoordCartesian(point,   float(),   float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Longarinas_pointscoord.append(ret)

                for i in range(len(Longarinas_list)):
                    for j in range(2):
                        Longarinas_pointscoord[i*2+j].insert(0, Longarinas_list[i]) # Inserção
dos nomes dos frames em cada lista

                # 20.6 Obtenção dos Start e End Stations de cada longarina

                ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar esforços em kNm/m e kN/m

                ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False,
1, 1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

                TableKey = 'Frame Bridge Object Flags'
                FieldKeyList = ''
                GroupName = ''
                TableVersion = int()
                FieldKeysIncluded = ['']
                NumberRecords = int()
                TableData = []
                ret = SapModel.DatabaseTables.GetTableForDisplayArray(TableKey, FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

                ColumnLabels = list(ret[3])
                Resultados = list(ret[5])

                # 20.7 Criação do dataframe e conversão dos tipos de dados (Associar todas as
longarinas com os pontos que as definem, bem como o Start Station e EndStation)

                ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
                df_longarinas_stations       =       pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
                df_longarinas_stations = df_longarinas_stations.replace({',' : '.'}, regex =
True) # Trocar vírgula por ponto em todas as colunas
                dados_numéricos = ['StartSta', 'EndSta'] # Colunas com dados numéricos
                for dado in dados_numéricos:
                    df_longarinas_stations[dado] = pd.to_numeric(df_longarinas_stations[dado])
# Mudar string para valor numérico

                df_longarinas_stations                                                    =
df_longarinas_stations[df_longarinas_stations['BOItemType'] == 'Girder'] # Pegando apenas as
longarinas
                LongarinasStartStations = df_longarinas_stations['StartSta'].tolist()
                LongarinasEndStations = df_longarinas_stations['EndSta'].tolist()

                for i in range(len(Longarinas_list)):
                    for j in range(2):
                        Longarinas_pointscoord[i*2+j].insert(0, LongarinasEndStations[i])
                        Longarinas_pointscoord[i*2+j].insert(0, LongarinasStartStations[i])

                df_pontos_longarinas = pd.DataFrame(Longarinas_pointscoord, columns = ['Start
Station', 'End Station', 'Frame', 'X', 'Y', 'Z']) # DataFrame com todos os frames, sendo cada
linha um ponto que a define
                df_pontos_longarinas = df_pontos_longarinas.round(4)   # Arredondar todas as
colunas do DataFrame para duas casas decimais
                CoordenadasYLongarinas                                                    =
df_pontos_longarinas['Y'].sort_values().unique().tolist()  # Coordenadas Y das longarinas em
ordem crescente

                # 20.8 Redefinição do vetor Areas_pointscoord para incluir as novas áreas
inseridas pela nova discretização no DataFrame

                ret = SapModel.SelectObj.ClearSelection()
```

```
                ret = SapModel.SelectObj.PropertyMaterial(f'C{f_ck} - Tabuleiro', False)
                ret = SapModel.FrameObj.SetSelected('', False, eItemType(2))
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_list = list(ret[3])
                Areas_points = [] # Nomes dos pontos que definem as áreas
                for Area in Areas_list:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Areas_points.extend(list(ret[2]))

                Areas_pointscoord = []
                for point in Areas_points:
                    ret   =   SapModel.PointObj.GetCoordCartesian(point,   float(),   float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Areas_pointscoord.append(ret)

                for i in range(len(Areas_list)):
                    for j in range(4):
                        Areas_pointscoord[i*4+j].insert(0, Areas_list[i]) # Inserção dos nomes
das áreas em cada lista

                # 20.9 Obtenção dos Start e End Stations de cada elemento de área

                ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False,
1, 1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

                TableKey = 'Area Bridge Object Flags'
                FieldKeyList = ''
                GroupName = ''
                TableVersion = int()
                FieldKeysIncluded = ['']
                NumberRecords = int()
                TableData = []
                ret = SapModel.DatabaseTables.GetTableForDisplayArray(TableKey, FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

                ColumnLabels = list(ret[3])
                Resultados = list(ret[5])

                # 20.10 Criação do DataFrame e conversão dos tipos de dados (Associar todos os
elementos de área do modelo a um StartStation e EndStation, bem como às coordendas de todos os
pontos que definem o elemento de área)
                ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
                df_laje_stations          =          pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
                df_laje_stations = df_laje_stations.replace({',' : '.'}, regex = True) # Trocar
vírgula por ponto em todas as colunas
                dados_numéricos = ['StartSta', 'EndSta'] # Colunas com dados numéricos
                for dado in dados_numéricos:
                    df_laje_stations[dado]  =  pd.to_numeric(df_laje_stations[dado])  #  Mudar
string para valor numérico

                AreasStartStations = df_laje_stations['StartSta'].tolist()
                AreasEndStations = df_laje_stations['EndSta'].tolist()

                for i in range(len(Areas_list)):
                    for j in range(4):
                        Areas_pointscoord[i*4+j].insert(0, AreasEndStations[i])
                        Areas_pointscoord[i*4+j].insert(0, AreasStartStations[i])

                df_pontos_lajes = pd.DataFrame(Areas_pointscoord, columns = ['Start Station',
'End Station', 'Area', 'X', 'Y', 'Z']) # DataFrame com todas as áreas, sendo cada linha um ponto
que a define
                df_pontos_lajes = df_pontos_lajes.round(4) # Arredondar todas as colunas do
DataFrame para 4 casas decimais

                # 20.11 Definição dos elementos de área de extremidade para o dimensionamento

                Areas_lajes_centrais = []
                for i in range (NLongarinasPorVão-1):
                    Areas_lajes_centrais.extend(df_pontos_lajes[(df_pontos_lajes["Y"]       <
CoordenadasYLongarinas[i+1])
```

```
                & (df_pontos_lajes["Y"] > CoordenadasYLongarinas[i]) & (df_pontos_lajes["Start
Station"] <= StartX + CVão/100/2 + 1) & (df_pontos_lajes["End Station"] >= StartX + CVão/100/2
- 1)]['Area'].unique().tolist())

                Areas_lajes_balanços      =      df_pontos_lajes[((df_pontos_lajes["Y"]      >
TotalWidthOfDeckSection/100/2 - OverhangLength/100)
                |      (df_pontos_lajes["Y"]      <      -TotalWidthOfDeckSection/100/2      +
OverhangLength/100))  &  (df_pontos_lajes["Start  Station"]  <=  StartX  +  CVão/100/2  +  1)  &
(df_pontos_lajes["End Station"] >= StartX + CVão/100/2 - 1)]['Area'].unique().tolist()
                Areas_lajes_extremidades       =       df_pontos_lajes[(df_pontos_lajes["Start
Station"] >= 1) & (df_pontos_lajes["Start Station"] <= 2)]['Area'].unique().tolist()

                for Area in Areas_lajes_extremidades:
                    ret = SapModel.AreaObj.SetAutoMesh('', 0, int(), int(), float(), float(),
bool(), bool(), bool(), float(), float() , bool() , bool() , bool() , bool() , 'ALL', bool() ,
float() , eItemType(2)) # Retirar discretização automática

                # 20.12 Elementos de área realmente utilizados para o dimensionamento das seções
S1 e S2

                df_pontos_lajes_min = df_pontos_lajes.groupby(['Area'], as_index=False).min()
# DataFrame composto pelo agrupamento do DataFrame original da coluna 'Area', pegando as
coordenas mínimas X, Y e Z dos nós que definem o elemento de área
                df_pontos_lajes_max = df_pontos_lajes.groupby(['Area'], as_index=False).max()
# Mesma coisa, só que máximos. O parâmetro as_index mantém os dados referentes à coluna Area
                df_pontos_lajes_join  =  df_pontos_lajes_min.merge(df_pontos_lajes_max,  on  =
['Area', 'Start Station', 'End Station']) # unindo os dataframes com base nas colunas em comum
                df_pontos_lajes_join = df_pontos_lajes_join.rename(columns = {"X_x" : "X_min",
"Y_x" : "Y_min", "Z_x" : "Z_min", "X_y" : "X_max", "Y_y" : "Y_max", "Z_y" : "Z_max" }) # renomear
colunas
                Areas_lajes_extremidades_S1 = []
                Areas_lajes_extremidades_S2 = []
                for i in range (NLongarinasPorVão):

Areas_lajes_extremidades_S1.extend(df_pontos_lajes_join[(df_pontos_lajes_join["Start
Station"] >= StartX/100 + 1) & (df_pontos_lajes_join["Start Station"] <= StartX/100 + 2) &
(df_pontos_lajes_join["Y_max"] <= CoordenadasYLongarinas[i] - t_long[0]/100/2)
                &     (df_pontos_lajes_join["Y_min"]     >=     CoordenadasYLongarinas[i]     -
t_long[0]/100/2 - h_laje/100)]['Area'].unique().tolist())

Areas_lajes_extremidades_S2.extend(df_pontos_lajes_join[(df_pontos_lajes_join["Start
Station"] >= StartX/100 + 1) & (df_pontos_lajes_join["Start Station"] <= StartX/100 + 2) &
(df_pontos_lajes_join["Y_max"] < CoordenadasYLongarinas[i] - b_long[0]/100/2)
                &     (df_pontos_lajes_join["Y_min"]     >=     CoordenadasYLongarinas[i]     -
b_long[0]/100/2 - h_laje/100)]['Area'].unique().tolist())




                # 21 Análise do modelo (pré-dimensionamento da protensão)

                # 21.1 Load Cases to be Run

            Name = ''
            Run = True # If this item is True, the specified load case is to be run.
            All = True # If this item is True, the run flag is set as specified by the Run item
for all load cases, and the Name item is ignored.
            ret = SapModel.Analyze.SetRunCaseFlag(Name, Run, All)

            if Tipo_analise == "ENCONTROS":
                ret  =  MyBridgeModeler.BridgeObject.UpdateWithSameParameters(myBridgeObjName,
1)

            if Tipo_analise != 'MESOESTRUTURA':
                ret = SapModel.Analyze.RunAnalysis()

            if Tipo_analise == 'SUPERESTRUTURA':
                ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar os dados em kN, kNm e
Celsius

                OutputCase = [] # Lista para filtrar o dataframe posteriormente

                ret = SapModel.LoadPatterns.GetNameList(NumberNames = int(), MyName = [])
                LoadPatternList = list(ret[2])
```

```
                ret = SapModel.LoadCases.GetNameList_1(NumberNames = int(), MyName = [])
                LoadCasesList = list(ret[2])
                OutputCase = LoadCasesList # Lista para filtrar o dataframe posteriormente

                ret = SapModel.RespCombo.GetNameList(NumberNames = int(), MyName = [])
                LoadCombinationsList = list(ret[2])
                OutputCase.extend(LoadCombinationsList)

                # 21.2 Extração dos esforços das longarinas

                ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False,
1, 1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

                ret                                                
```

```
                stations_iniciais = df["Station"].unique() # Obtenção dos stations iniciais
gerados pelo programa após filtragem

                Cases_analise = ['DEAD', 'G3', 'TB-450max', 'TB-450min'] # Criação de
dicionários para a inclusão dos resultados
                M3 = {'DEAD': [0],
                    'G3': [0],
                    'TB-450max' : [0],
                    'TB-450min': [0]}

                V2 = {'DEAD': [],
                    'G3': [],
                    'TB-450max' : [],
                    'TB-450min': []}

                T = {'DEAD': [],
                    'G3': [],
                    'TB-450max' : [],
                    'TB-450min': []}

                for j in range (len(Girders_stations)):
                    for k in range(len(Cases_analise)):
                        if  str(Girders_stations[j])  in  str(stations_iniciais[j])  and
Cases_analise[k] != 'TB-450max' and Cases_analise[k] != 'TB-450min': # Para o TB-450, deve-se
filtrar a tabela entre as envoltórias de máximos e mínimos
                            filtered_df  =  df[(df["Station"]  ==  Girders_stations[j])  &
(df["OutputCase"]  ==  Cases_analise[k])] # Filtrando dataframe para a station e loadcase
específicos
                            V2[f'{Cases_analise[k]}'].append(filtered_df['V2'].abs().max())
                            T[f'{Cases_analise[k]}'].append(filtered_df['T'].abs().max())

                        if  str(Girders_stations[j])  in  str(stations_iniciais[j])  and
Cases_analise[k] == 'TB-450max':
                            filtered_df  =  df[(df["Station"]  ==  Girders_stations[j])  &
(df["OutputCase"] == 'TB-450') & (df["StepType"] == 'Max')]
                            V2['TB-450max'].append(filtered_df['V2'].abs().max())
                            T['TB-450max'].append(filtered_df['T'].abs().max())

                        if  str(Girders_stations[j])  in  str(stations_iniciais[j])  and
Cases_analise[k] == 'TB-450min':
                            filtered_df  =  df[(df["Station"]  ==  Girders_stations[j])  &
(df["OutputCase"] == 'TB-450') & (df["StepType"] == 'Min')]
                            V2['TB-450min'].append(filtered_df['V2'].abs().max())
                            T['TB-450min'].append(filtered_df['T'].abs().max())

                        # Casos de interpolação
                        if  str(Girders_stations[j])  not  in  str(stations_iniciais[j])  and
Cases_analise[k] == 'TB-450max': # Envoltória de máximos
                            filtered_df  =  df[(df["Station"]  <=  Girders_stations[j])  &
(df["OutputCase"] == 'TB-450')  & (df["StepType"] == 'Max')].iloc[-len(Girders_list):]
                            M3max_x1 = filtered_df['M3'].abs().max()
                            Vmax_x1 = filtered_df['V2'].abs().max()
                            Tmax_x1 = filtered_df['T'].abs().max()
                            x1 = filtered_df['Station'].abs().max()
                            filtered_df  =  df[(df["Station"]  >=  Girders_stations[j])  &
(df["OutputCase"] == 'TB-450')  & (df["StepType"] == 'Max')].iloc[:len(Girders_list)]
                            M3max_x2 = filtered_df['M3'].abs().max()
                            Vmax_x2 = filtered_df['V2'].abs().max()
                            Tmax_x2 = filtered_df['T'].abs().max()
                            x2 = filtered_df['Station'].abs().max()

                            M3_interp = np.array([M3max_x1, M3max_x2])
                            V2_interp = np.array([Vmax_x1, Vmax_x2])
                            T_interp = np.array([Tmax_x1, Tmax_x2])
                            x = np.array([x1, x2])
                            M3['TB-450max'].append(np.interp(Girders_stations[j],          x,
M3_interp))
                            V2['TB-450max'].append(np.interp(Girders_stations[j],          x,
V2_interp))
                            T['TB-450max'].append(np.interp(Girders_stations[j], x, T_interp))

                        if  str(Girders_stations[j])  not  in  str(stations_iniciais[j])  and
Cases_analise[k] == 'TB-450min': # Envoltória de mínimos
                            filtered_df  =  df[(df["Station"]  <=  Girders_stations[j])  &
(df["OutputCase"] == 'TB-450')  & (df["StepType"] == 'Min')].iloc[-len(Girders_list):]
```

```
                            M3min_x1 = filtered_df['M3'].abs().max()
                            Vmin_x1 = filtered_df['V2'].abs().max()
                            Tmin_x1 = filtered_df['T'].abs().max()
                            x1 = filtered_df['Station'].abs().max()
                            filtered_df = df[(df["Station"] >= Girders_stations[j]) &
(df["OutputCase"] == 'TB-450') & (df["StepType"] == 'Min')].iloc[:len(Girders_list)]
                            M3min_x2 = filtered_df['M3'].abs().max()
                            Vmin_x2 = filtered_df['V2'].abs().max()
                            Tmin_x2 = filtered_df['T'].abs().max()
                            x2 = filtered_df['Station'].abs().max()

                            M3_interp = np.array([M3min_x1, M3min_x2])
                            V2_interp = np.array([Vmin_x1, Vmin_x2])
                            T_interp = np.array([Tmin_x1, Tmin_x2])
                            x = np.array([x1, x2])
                            M3['TB-450min'].append(np.interp(Girders_stations[j], x,
M3_interp))
                            V2['TB-450min'].append(np.interp(Girders_stations[j], x,
V2_interp))
                            T['TB-450min'].append(np.interp(Girders_stations[j], x, T_interp))

                        if str(Girders_stations[j]) not in str(stations_iniciais[j]) and
Cases_analise[k] != 'TB-450max' and Cases_analise[k] != 'TB-450min':
                            filtered_df = df[(df["Station"] <= Girders_stations[j]) &
(df["OutputCase"] == Cases_analise[k])].iloc[-len(Girders_list):]
                            M3_x1 = filtered_df['M3'].abs().max()
                            V_x1 = filtered_df['V2'].abs().max()
                            T_x1 = filtered_df['T'].abs().max()
                            x1 = filtered_df['Station'].abs().max()
                            filtered_df = df[(df["Station"] >= Girders_stations[j]) &
(df["OutputCase"] == Cases_analise[k])].iloc[:len(Girders_list)]
                            M3_x2 = filtered_df['M3'].abs().max()
                            V_x2 = filtered_df['V2'].abs().max()
                            T_x2 = filtered_df['T'].abs().max()
                            x2 = filtered_df['Station'].abs().max()

                            M3_interp = np.array([M3_x1, M3_x2])
                            V2_interp = np.array([V_x1, V_x2])
                            T_interp = np.array([T_x1, T_x2])
                            x = np.array([x1, x2])
                            M3[f'{Cases_analise[k]}'].append(np.interp(Girders_stations[j],
x, M3_interp))
                            V2[f'{Cases_analise[k]}'].append(np.interp(Girders_stations[j],
x, V2_interp))
                            T[f'{Cases_analise[k]}'].append(np.interp(Girders_stations[j], x,
T_interp))

                # 21.3 Alimentação da planilha de dimensionamento das longarinas
                ws_long = wb_long.sheets["5 Esforços solicitantes"]


                # Momentos fletores

                ws_long['D48:H48'].value = M3['DEAD']
                ws_long['D50:H50'].value = M3['G3']
                ws_long['D52:H52'].value = M3['TB-450max']
                ws_long['D53:H53'].value = [x *-1 for x in M3['TB-450min']]

                # Forças cortantes

                V2['DEAD'].pop() # retirar último item da lista
                V2['G3'].pop() # retirar último item da lista
                ws_long['E59:H59'].value = V2['DEAD']
                ws_long['E61:H61'].value = V2['G3']
                ws_long['E62:H62'].value = [max(V2['TB-450max'][i], V2['TB-450max'][i + 1]) for
i in range(len(V2['TB-450max']) - 1)] # Cria a nova lista com os valores máximos entre os itens
adjacentes
                ws_long['E63:H63'].value = [x*-1 for x in [max(V2['TB-450min'][i], V2['TB-
450min'][i + 1]) for i in range(len(V2['TB-450min']) - 1)]]

                # Momentos torçores

                ws_long['E68:H68'].value = [max(T['DEAD'][i], T['DEAD'][i + 1]) for i in
range(len(T['DEAD']) - 1)]
```

```
                ws_long['E69:H69'].value = [max(T['TB-450max'][i], T['TB-450max'][i + 1]) for
i in range(len(T['TB-450max']) - 1)]
                ws_long['E70:H70'].value = [x*-1 for x in [max(T['TB-450min'][i], T['TB-
450min'][i + 1]) for i in range(len(T['TB-450min']) - 1)]]


                # Esforços de retração e fluência e temperatura
                N_r = -1*df[df["OutputCase"] == 'Retração e fluência']['P'].abs().max()
                M_r = df[df["OutputCase"] == 'Retração e fluência']['M3'].abs().max()

                N_Δt = df[df["OutputCase"] == 'Temperatura gradiente neg']['P'].abs().max()
                M_Δt = df[df["OutputCase"] == 'Temperatura gradiente neg']['M3'].abs().max()

                N_Δt2 = df[df["OutputCase"] == 'Temperatura gradiente pos']['P'].abs().max()
                M_Δt2     =     -1*df[df["OutputCase"]     ==     'Temperatura     gradiente
pos']['M3'].abs().max()

                ws_long = wb_long.sheets["6 Retração e temperatura"]
                ws_long['E56'].value = N_r
                ws_long['E57'].value = M_r

                ws_long['E114'].value = N_Δt
                ws_long['E115'].value = M_Δt

                ws_long['E151'].value = N_Δt2
                ws_long['E152'].value = M_Δt2

                # Flechas
                ws_long = wb_long.sheets["5 Esforços solicitantes"]

                # 21.4 Extração dos esforços das lajes (Filtragem dos dados para não
sobrecarregar de informações)
                ret  =  SapModel.DatabaseTables.SetLoadPatternsSelectedForDisplay(['Veiculo',
'Multidao', 'Pav + recap', 'Defensas', 'DEAD'])
                ret    =    SapModel.DatabaseTables.SetLoadCasesSelectedForDisplay(['Veiculo',
'Multidao', 'Pav + recap', 'Defensas', 'DEAD'])
                ret = SapModel.DatabaseTables.SetLoadCombinationsSelectedForDisplay(['TB-450',
'Carga permanente'])

                ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar esforços em kNm/m e kN/m

                ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False,
1, 1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

                TableKey = 'Element Forces - Area Shells'
                FieldKeyList = ''
                GroupName = ''
                TableVersion = int()
                FieldKeysIncluded = ['']
                NumberRecords = int()
                TableData = []
                ret = SapModel.DatabaseTables.GetTableForDisplayArray(TableKey, FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

                ColumnLabels = list(ret[3])
                Resultados = list(ret[5])

                # Criação do dataframe e conversão dos tipos de dados
                ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
                df_laje_esforços          =          pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
                df_laje_esforços = df_laje_esforços.replace({',' : '.'}, regex = True) # Trocar
vírgula por ponto em todas as colunas
                dados_numéricos = ['F11', 'F22', 'F12', 'FMax', 'FMin', 'FAngle', 'FVM', 'M11',
'M22', 'M12', 'MMax', 'MMin', 'MAngle', 'V13', 'V23', 'VMax', 'VAngle'] # Colunas com dados
numéricos
                for dado in dados_numéricos:
                    df_laje_esforços[dado]  =  pd.to_numeric(df_laje_esforços[dado])  #  Mudar
string para valor numérico


                df_lajes_centrais                                                          =
df_laje_esforços[df_laje_esforços['Area'].isin(Areas_lajes_centrais)]
                df_lajes_balanços                                                          =
df_laje_esforços[df_laje_esforços['Area'].isin(Areas_lajes_balanços)]
```

```
                df_lajes_extremidades                                                                    =
df_laje_esforços[df_laje_esforços['Area'].isin(Areas_lajes_extremidades)]
                df_lajes_extremidades_S1                                                                 =
df_laje_esforços[df_laje_esforços['Area'].isin(Areas_lajes_extremidades_S1)]
                df_lajes_extremidades_S2                                                                 =
df_laje_esforços[df_laje_esforços['Area'].isin(Areas_lajes_extremidades_S2)]

                Mgy_central  =  df_lajes_centrais[(df_lajes_centrais["OutputCase"]  ==  'Carga
permanente')]['M22'].abs().max()
                Mgx_central  =  df_lajes_centrais[(df_lajes_centrais["OutputCase"]  ==  'Carga
permanente')]['M11'].abs().max()

                M_qypos_central  = df_lajes_centrais[(df_lajes_centrais["OutputCase"] == 'TB-
450') & (df_lajes_centrais["StepType"] == 'Max')]['M22'].abs().max()
                M_qyneg_central = -df_lajes_centrais[(df_lajes_centrais["OutputCase"] == 'TB-
450') & (df_lajes_centrais["StepType"] == 'Min')]['M22'].abs().max()

                M_qxpos_central  = df_lajes_centrais[(df_lajes_centrais["OutputCase"] == 'TB-
450') & (df_lajes_centrais["StepType"] == 'Max')]['M11'].abs().max()
                M_qxneg_central = -df_lajes_centrais[(df_lajes_centrais["OutputCase"] == 'TB-
450') & (df_lajes_centrais["StepType"] == 'Min')]['M11'].abs().max()

                Mgy_balanços  = -df_lajes_balanços[(df_lajes_balanços["OutputCase"] == 'Carga
permanente')]['M22'].abs().max()
                M_qypos_balanços  = df_lajes_balanços[(df_lajes_balanços["OutputCase"] == 'TB-
450') & (df_lajes_balanços["StepType"] == 'Max')]['M22'].abs().max()
                M_qyneg_balanços = -df_lajes_balanços[(df_lajes_balanços["OutputCase"] == 'TB-
450') & (df_lajes_balanços["StepType"] == 'Min')]['M22'].abs().max()

                Vg_S1  =  df_lajes_extremidades_S1[(df_lajes_extremidades_S1["OutputCase"]  ==
'Carga permanente')]['V13'].abs().mean()
                Vg_S2  =  df_lajes_extremidades_S2[(df_lajes_extremidades_S2["OutputCase"]  ==
'Carga permanente')]['V13'].abs().mean()

                Vq_S1  =  df_lajes_extremidades_S1[(df_lajes_extremidades_S1["OutputCase"]  ==
'TB-450') & (df_lajes_extremidades_S1["StepType"] == 'Max')]['V13'].abs().mean()
                Vq_S2  =  df_lajes_extremidades_S2[(df_lajes_extremidades_S2["OutputCase"]  ==
'TB-450') & (df_lajes_extremidades_S2["StepType"] == 'Max')]['V13'].abs().mean()

                # 21.5 Alimentação da planilha de dimensionamento das lajes com os esforços

                modelo_laje_prelaj = os.path.join(main_folder, 'slab-preslab-template.xlsx')
                wb_laje_prelaj = xw.Book(modelo_laje_prelaj)
                ws_laje_prelaj = wb_laje_prelaj.sheets["3 Esforços solicitantes"]
                ws_laje_prelaj['E18'].value = CVão/100
                ws_laje_prelaj['E49'].value = Mgy_central
                ws_laje_prelaj['E50'].value = Mgx_central
                ws_laje_prelaj['E55'].value = M_qypos_central
                ws_laje_prelaj['E56'].value = M_qyneg_central
                ws_laje_prelaj['E57'].value = M_qxpos_central
                ws_laje_prelaj['E58'].value = M_qxneg_central
                ws_laje_prelaj['E242'].value = Mgy_balanços
                ws_laje_prelaj['E246'].value = M_qypos_balanços
                ws_laje_prelaj['E247'].value = M_qyneg_balanços
                ws_laje_prelaj['E277'].value = Vg_S1
                ws_laje_prelaj['E278'].value = Vg_S2
                ws_laje_prelaj['E283'].value = Vq_S1
                ws_laje_prelaj['E284'].value = Vq_S2

                # 21.6 Alimentação da planilha de dimensionamento das lajes com os dados
iniciais

                ws_laje_prelaj = wb_laje_prelaj.sheets["2 Dimensões"]
                ws_laje_prelaj['G10'].value = h_laje/100
                ws_laje_prelaj['G11'].value = h_laje/100 + d_long[1]/100 + d_long[2]/100
                ws_laje_prelaj['G12'].value = Espaçamento_Long/100 - b_long[0]/100
                ws_laje_prelaj['F25'].value = OverhangLength/100 - b_long[0]/100/2
                ws_laje_prelaj['F35'].value = h_prelaj/100
                ws_laje_prelaj['F36'].value = bw_prelaj/100
                ws_laje_prelaj['F37'].value = l_prelaj/100
```

```
            # 22 Salvar modelo e planilhas
            if Tipo_analise != 'MESOESTRUTURA':
                ret = SapModel.SetPresentUnits(eUnits(kN_m_C))
                ret = SapModel.File.Save()
            if Tipo_analise == 'SUPERESTRUTURA':
                wb_long.save(os.path.join(main_folder, Tipologia, f'{Tipologia}_LONG.xlsm'))
                wb_pil_ext.save(os.path.join(main_folder,                              Tipologia,
f'{Tipologia}_PIL_EXTREMO.xlsx'))
                wb_retração_fluência.save(os.path.join(main_folder,                    Tipologia,
f'{Tipologia}_RETRAÇÃO_FLUÊNCIA.xlsx'))
                wb_laje_prelaj.save(os.path.join(main_folder,                          Tipologia,
f'{Tipologia}_LAJE_PRELAJ.xlsx'))




            # 23 ANÁLISE DE MESOESTRUTURA COM VÃOS IGUAIS

            # 23.1 Definição - Travessas centrais (utilizadas apenas para análise dos pilares
e travessas). Essa seção basicamente define um novo bent na travessa central para que seja
possível variar a altura do pilar

            if Tipo_analise == 'MESOESTRUTURA':
                ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

                ret = MyBridgeModeler.Bent.AddNew('Apoios centrais')
                Name = 'Apoios centrais'
                CapBeamSection = f'Travessa intermediária (C{f_ck})'
                Foundation = 'Fixed'
                BentType = 1
                ret = MyBridgeModeler.Bent.SetGeneralData(Name, CapBeamSection, C_travessaint,
SupportType_int, N_pilaresint, Foundation, BentType)

                for altura in Alturas_pilarescentrais:
                    if N_pilaresint == 2:
                        Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                        Height_int = [altura, altura] #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing the column height. [L]
                    else:
                        Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                        Height_int = [altura, altura, altura]

                    Section = [f'Pilar (C{f_ck})'] * N_pilaresint
                    Angle = [0] * N_pilaresint
                    Foundation = ['Fixed'] * N_pilaresint
                    ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_int,
Height_int, Angle, Foundation)

                    BeforeSuperConnection = 1
                    # The superstructure connection before the bent.
                    # 0 = Connect to girder top and bottom
                    # 1 = Connect to girder bottom only
                    BeforeDistanceToBearingLine = d_ap #The distance to the bearing line before
the bent. [L]
                    BeforeDistanceToEndOfGirder = 0 #The distance to the end of the girder
before the bent. [L]
                    AfterSuperConnection = 1
                    # The superstructure connection after the bent.
                    # 0 = Connect to girder top and bottom
                    # 1 = Connect to girder bottom only
                    AfterDistanceToBearingLine = d_ap #The distance to the bearing line after
the bent. [L]
                    AfterDistanceToEndOfGirder = 0 #AfterDistanceToEndOfGirder
                    ret       =       MyBridgeModeler.Bent.SetSuperConnectionDoubleLine(Name,
BeforeSuperConnection,        BeforeDistanceToBearingLine,        BeforeDistanceToEndOfGirder,
AfterSuperConnection, AfterDistanceToBearingLine, AfterDistanceToEndOfGirder)

                    SpanNameBeforeBent = 'Vão 2' #The name of the span before the considered
bent.
                    BentProperty = 'Apoios centrais' #The bent property assigned to the
specified bent.
```

```
                    BentDirection = 'Default' #The bearing of a line parallel to the bent. It
is either Default or a valid bearing string.
                    ret    =    MyBridgeModeler.BridgeObject.BentAssign.SetBentAssignment(Name,
SpanNameBeforeBent, BentProperty, BentDirection)

                    BentSupportName = 'Apoio 2'
                    ret     =     MyBridgeModeler.BridgeObject.BentAssign.SetSupportName(Name,
SpanNameBeforeBent, BentSupportName)

                    SuperstructureIsContinousAtBent    =    False    #Indicates    whether    the
superstructure is continuous (True) or discontinuous (False) at the bent.
                    DiaphragmBeforeBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the diaphragm property at the bent.
                    # If the specified bent property is None or it has a single line of bearings
(continuous superstructure) this is the property of the diaphragm that occurs at the bent.
                    #  If  the  specified  bent  property  has  a  double  line  of  bearings
(discontinuous superstructure) this is the property of the diaphragm that occurs just before
the bent.
                    # This item does not apply if the specified bent property is a double
bearing line with closure pour.
                    InternalDiaphragmBeforeBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the internal diaphragm property at the bent.
                    # This only applies to bents at steel U girder or super T girder deck
sections.
                    # If the specified bent property is None or it has a single line of bearings
(continuous superstructure) this is the property of the internal diaphragm that occurs at the
bent.
                    # If the specified bent property has a double line of bearings this is the
property of the internal diaphragm that occurs just before the bent.
                    DiaphragmAfterBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the diaphragm property after the bent.
                    # It applies when the specified bent property has a double line of bearings
(discontinuous superstructure).
                    # This item does not apply if the specified bent property is a double
bearing line with closure pour.
                    InternalDiaphragmAfterBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the internal diaphragm property after the bent.
                    # This only applies to bents at steel U girder or super T girder deck
sections.
                    # It applies when the specified bent property has a double line of bearings
and is not a closure pour.
                    ClosurePourDiaphragm = 'None' #This is the name of a defined closure pour
bridge diaphragm property. This item only applies when the specified bent type is Double Bearing
Line with Closure Pour.
                    Restrainer = 'None' #This is either None or the name of a defined bridge
diaphragm  property  indicating  the  restrainers  used  at  the  bent.  It  only  applies  when  the
superstructure is discontinuous at the bent.
                    RestrainerElevation = 0 #The restrainer elevation (at layout line, global
Z). this item only applies when the superstructure is discontinuous at the bent. [L]
                    MeshSuperstructureToMatchBent = False
                    # This is a flag indicating if the superstructure is to be meshed to match
the bent.
                    # This item only applies when the superstructure is continuous at the bent.
                    MeshSteelUGirderToMatchBent = False
                    # This is a flag indicating if the steel U girders are to be meshed to
match the bent.
                    # This item only applies to bents at steel U girder or super T girder deck
sections when the superstructure is continuous at the bent.
                    ret                                                                    =
MyBridgeModeler.BridgeObject.BentAssign.SetSuperstructureAssignment(Name,   SpanNameBeforeBent,
SuperstructureIsContinousAtBent,       DiaphragmBeforeBent,       InternalDiaphragmBeforeBent,
DiaphragmAfterBent, InternalDiaphragmAfterBent,
                    ClosurePourDiaphragm,          Restrainer,          RestrainerElevation,
MeshSuperstructureToMatchBent, MeshSteelUGirderToMatchBent)

                    Elevation = -h_laje -h_long - h_ap #The global Z elevation of the bent. [L]
                    HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the cap beam. [L]
                    ret    =    MyBridgeModeler.BridgeObject.BentAssign.SetBentLocation(Name,
SpanNameBeforeBent, Elevation, HorizontalOffset)
```

```
                    for j in range(2):
                        BearingLocation = [1, 2]
                        # This is one of the following indicating the bent bearing location:
                        # 1 = Single line of bearings or bearings before bent for double line
of bearings.
                        # 2 = Bearings after bent for double line of bearings.
                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the bent.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the bent.
                        ElevationAtLayoutLine = -h_laje -h_long - h_ap/2 #The bearing elevation
(global Z) at the layout line. [L]
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int()
                        #The number of bearings for the bridge section or for abutments at
Steel U Girder deck sections the number of bearings at the U girder.
                        # For bearings at steel U girder sections this item must be 1 or 2.
                        # This item only applies when the bearing assignment type is General
or when the abutment is at a Steel U Girder deck section.
                        UniformSpacing = int() #The uniform spacing of bearings. This item only
applies when the bearing assignment type is General. [L]
                        BearingOffset = int() #The offset from the section reference point to
the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                                 =
MyBridgeModeler.BridgeObject.BentAssign.SetBearingAssignment(Name,          SpanNameBeforeBent,
BearingLocation[j],      BearingAssignmentType,      BearingProperty,      RestrainerProperty,
ElevationAtLayoutLine, RotationAngle, NumberOfBearings,
                        UniformSpacing, BearingOffset)

                    ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)

                    ret = SapModel.SetPresentUnits(eUnits(kN_m_C))

                    ret     =     SapModel.File.Save(os.path.join(main_folder,     Tipologia,
f'{Tipologia} ({Tipo_analise}) PILAR {altura/100} m VAOS IGUAIS.bdb'))

                    ret = SapModel.Analyze.SetRunCaseFlag('', True, True)

                    ret = SapModel.Analyze.RunAnalysis()

                    ret = SapModel.File.Save()

            wb_long.app.quit()
            # ret = mySAPObject.ApplicationExit(False)
            SapModel = None
            mySAPObject = None
            MyBridgeModeler = None
```

```
        # ESSE PROGRAMA FOI FEITO PARA SER USADO EM CONJUNTO COM O 'precast-main'.py APÓS REALIZADA A ANÁLISE DA SUPERESTRUTURA. DEPOIS DE REALIZAR OS AJUSTES NAS PLANILHAS DE DIMENSIONAMENTO DAS LONGARINAS DAS TIPOLOGIAS RODADAS, ELE IRÁ EFETUAR O READ E WRITE DOS DADOS RELATIVOS À PROTENSÃO E INSERÍ-LOS NO MODELO
        # APÓS ISSO, ELE IRÁ REALIZAR A ANÁLISE DA ESTRUTURA PARA SE OBTEREM AS FLECHAS PARA O DIMENSIONAMENTO DO ELS-DEF.
        # INPUTAR AS TIPOLOGIAS A SEREM RODADAS NO VETOR TIPOLOGIAS

        import os
        import sys
        import clr
        import pandas as pd

        from enum import Enum

        clr.AddReference("System.Runtime.InteropServices")

        from System.Runtime.InteropServices import Marshal

        #set the following path to the installed CSiBridge program directory

        clr.AddReference(r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge1.dll')

        from CSiBridge1 import *

        import numpy as np

        from PIL import Image

        import openpyxl # Comunicação com Excel

        import xlwings as xw

        #set the following flag to True to execute on a remote computer

        Remote = False


        #if the above flag is True, set the following variable to the hostname of the remote computer

        #remember that the remote computer must have CSiBridge installed and be running the CSiAPIService.exe

        RemoteComputer = "SpareComputer-DT"


        #set the following flag to True to attach to an existing instance of the program

        #otherwise a new instance of the program will be started

        AttachToInstance = True


        #set the following flag to True to manually specify the path to CSiBridge.exe

        #this allows for a connection to a version of CSiBridge other than the latest installation

        #otherwise the latest installed version of CSiBridge will be launched

        SpecifyPath = True


        #if the above flag is set to True, specify the path to CSiBridge below

        ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge.exe'

        #full path to the model
```

```
        #set it to the desired path of your model

        #if executing remotely, ensure that this folder already exists on the remote computer

        #the below command will only create the folder locally

        main_folder = os.path.dirname(os.path.abspath(__file__))
        APIPath = main_folder

        if not os.path.exists(APIPath):

            try:

                os.makedirs(APIPath)

            except OSError:

                pass

        def chunk_list(input_list, chunk_size):
        # Create a list of lists with fixed size from another list
            return [input_list[i:i + chunk_size] for i in range(0, len(input_list), chunk_size)]

        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'VARIÁVEIS-PPC.xlsx')
        variables    =    pd.read_excel(variables_data_file,    sheet_name="Main",    header=2,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-PPC-A-20-P', 'S-PPC-A-25-P', 'S-PPC-A-30-P', 'S-PPC-A-35-P', 'S-PPC-
A-20-S', 'S-PPC-A-25-S', 'S-PPC-A-30-S', 'S-PPC-A-35-S', 'S-PPC-B-20-P', 'S-PPC-B-25-P',
        'S-PPC-B-30-P', 'S-PPC-B-35-P', 'S-PPC-B-20-S', 'S-PPC-B-25-S', 'S-PPC-B-30-S', 'S-
PPC-B-35-S']

        for Tipologia in Tipologias: # Rodar todo o código com base em um elemento da lista
Tipologias
            xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
            GeneralPath = APIPath + os.sep + Tipologia
            os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à tipologia. O
parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
            ModelPath = GeneralPath + os.sep + f'{Tipologia}.bdb' # Caminho do modelo .bdb

            #create API helper object

            helper = cHelper(Helper())


            if AttachToInstance:

                #attach to a running instance of CSiBridge

                try:

                    #get the active CSiBridge object

                    if Remote:

                                        mySAPObject   =   cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                    else:

                        mySAPObject = cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                except:

                    print("No running instance of the program found or failed to attach.")

                    sys.exit(-1)


            else:
```

```
                if SpecifyPath:
                    try:
                        #'create an instance of the CSiBridge object from the specified path
                        if Remote:
                                mySAPObject = cOAPI(helper.CreateObjectHost(RemoteComputer,
ProgramPath))
                        else:
                            mySAPObject = cOAPI(helper.CreateObject(ProgramPath))
                    except :
                        print("Cannot start a new instance of the program from " + ProgramPath)
                        sys.exit(-1)
                else:

                    try:
                        #create an instance of the CSiBridge object from the latest installed
CSiBridge
                        if Remote:
                            mySAPObject = cOAPI(helper.CreateObjectProgIDHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))
                        else:
                                                                                mySAPObject      =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))

                    except:
                        print("Cannot start a new instance of the program.")
                        sys.exit(-1)

                #start CSiBridge application
                mySAPObject.ApplicationStart()

            StartX = 0 # Começo do layoutline no modelo

                #*********************************    INÍCIO    VARIÁVEIS    DE    INPUT
*******************************************#
            f_ck = 35
            df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia

            # Dimensões da longarina

            b_long = df_ponte.loc[:, 'B1': 'B4'].values.flatten().tolist() #B1 a B4 conforme
caixa de diálogo de seção pré-moldada do tipo I
            d_long = df_ponte.loc[:, 'D1': 'D7'].values.flatten().tolist() #D1 a D7
            t_long = df_ponte.loc[:, 'T1': 'T2'].values.flatten().tolist() #T1 a T2
            c1_long = 0 #Bottom flange chamfer dimension
            h_long = d_long[0]
             OverhangLength = df_ponte['Balanço (cm)'].iloc[0] #balanço laje/longarina #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
            NLongarinasPorVao = int(df_ponte['Nlong'].iloc[0])
            NumberInteriorGirders = NLongarinasPorVao - 2 #The number of interior girders
            TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
```

```
            Espaçamento_Long = (TotalWidthOfDeckSection - 2*OverhangLength)/(NLongarinasPorVao-
1)

            # Diâmetro do pilar
            t3_pilar = df_ponte['Dp'].iloc[0]

            # Travessas e transversinas

            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]

            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

            h_transv = df_ponte['hD'].iloc[0]
            bw_transv = df_ponte['bwD'].iloc[0]

            # Tabuleiro

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - b_long[0]

            # Geometria geral ponte
            NVãos = 1
            CVão = df_ponte['Vão (cm)'].iloc[0]
            CTotal = NVãos * CVão
            SpanLength = [CVão] #A zero-based array containing the length of each span. [L]
            SpanName = ['Vao 1']
            NDiafragmasPorVao = int(df_ponte['Ndiaf'].iloc[0])
            DiaphragmDistance = [15, 3/8*CVão, 6/8*CVão, CVão-15] #The distance from the start
of the specified span to the diaphragm. [L]
            LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)

            # Aparelho de apoio

            K_ap = 47.17 # Rigidez do aparelho de apoio na direção longitudinal e transversal
à longarina
            h_ap = df_ponte['hA'].iloc[0] # Altura total do aparelho de apoio
             d_ap = df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta

            # Bents

            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                 Distance_ext = [Distextp, TotalWidthOfDeckSection - Distextp] #A zero-based
one-dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, TotalWidthOfDeckSection/2, TotalWidthOfDeckSection -
Distextp]
                Height_ext = [Hextp, Hextp, Hextp]


            C_travessaint = C_travessaext - 150
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
            SupportType_int = 1
            if N_pilaresint == 2:
                 Distance_int = [Distintp, TotalWidthOfDeckSection - Distintp] #A zero-based
one-dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
```

```
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, TotalWidthOfDeckSection/2, TotalWidthOfDeckSection -
Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # Laje de Transição
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

            # Muro
            AlturaM = df_ponte['AlturaM'].iloc[0]

            # Ala
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

                    #*********************************     FIM     VARIÁVEIS     DE     INPUT
*******************************************#



            # 1 Definição dos objetos primitivos
            SapModel = cSapModel(mySAPObject.SapModel)
             MyBridgeModeler  =  cBridgeModeler_1(SapModel.BridgeModeler_1)  #  necessário  para
acessar os métodos do CSiBridge

             ret  =  SapModel.File.OpenFile(os.path.join(main_folder,  Tipologia,  f'{Tipologia}
(SUPERESTRUTURA).bdb'))
              ret   =   SapModel.File.Save(os.path.join(main_folder,   Tipologia,   f'{Tipologia}
(FLECHAS).bdb'))
            kN_cm_C = 13
            kN_m_C = 6
            kN_mm_C = 5
            ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

            # 2 Objetos de interesse
            NumberNames = int()
            MyName = []
            ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
            myBridgeObjName = list(ret[2])[0]

            SectionType = 11
            NumberNames = int()
            MyName = []
            BridgeSectionType = [SectionType]
                  ret     =     MyBridgeModeler.DeckSection.GetNameList(NumberNames,     MyName,
BridgeSectionType)
            myDeckSectionName = list(ret[2])[0]

            Name = myBridgeObjName
            LayoutLine = ''
            ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
            myLayoutLineName = ret[1]

            ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
            myLaneName = list(ret[2])[0]



             # 3 Cabos de protensão (cBMPrestressTendonAssign) - Realizado após o pré-
dimensionamento (após os ajustes manuais da planilha de dimensionamento das longarinas)
            # Redefinição da planilha de longarina
            long_path = os.path.join(main_folder, Tipologia, f'{Tipologia}_LONG.xlsm')
            long_tipologia = xw.Book(long_path)

            # 3.1 Leitura dos dados referentes ao traçado dos cabos
            ret = SapModel.SetModelIsLocked(False)
            ws_long = long_tipologia.sheets["3 Traçado dos cabos"]
            e_inícioVão = ws_long['D22'].value*100
            e_meioVão = ws_long['F22'].value*100
```

```
            # 3.2 Definição dos cabos
            ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))
            for i in range(NVãos):
                xlwingsapp = xw.App(visible=False)

               ProgramDefinedNewTendonName = f'TEN{1 + i*(NLongarinasPorVao)}' # The name that
the program assigns to the tendon.
                                                                                ret                =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.AddNew(myBridgeObjName,
ProgramDefinedNewTendonName)

                Name = myBridgeObjName
                TendonName = f'TEN{1 + i*(NLongarinasPorVao)}' # The name of the tendon.
                 ReferenceLine = 1 # This is either 0, indicating the bridge centerline, or a
girder number where the girders are numbered from left to right.
                ModelingOption = 2
                # This is one of the following modeling options.
                # 1 = Model as loads
                # 2 = Model as elements
                                                                              ret                =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetGeneralParameters(Name,      TendonName,
ReferenceLine, ModelingOption)

                Name = myBridgeObjName
                TendonName = TendonName
                StartReferenceSpan = f'Vao {i+1}' # The span used as a reference span for the
tendon start location.
                StartReferenceLocation = 1
                 # The location within the start reference span used as a reference point for
the tendon start location.
                # 1 = Start
                # 2 = Center
                # 3 = End
                StartDistanceType = 2
                # The distance type for locating the tendon start location.
                # 1 = Absolute
                # 2 = Variable
                 StartOffsetFromReferenceLocation = 2 # The distance from the reference point
to the start of the tendon. [L]
                EndReferenceSpan = f'Vao {i+1}'
                EndReferenceLocation = 3
                EndDistanceType = 2
                EndOffsetFromReferenceLocation = -2
                                                                              ret                =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetTendonEndPoints(Name,        TendonName,
StartReferenceSpan,                StartReferenceLocation,                StartDistanceType,
StartOffsetFromReferenceLocation,
                                EndReferenceSpan,     EndReferenceLocation,     EndDistanceType,
EndOffsetFromReferenceLocation)

                ws_long = long_tipologia.sheets["7 Estim. Força de Protensão"]
                Name = myBridgeObjName
                TendonName = TendonName
                JackFrom = 3
                # This is one of the following indicating where the tendon jacks are located.
                # 1 = Tendon start
                # 2 = Tendon end
                # 3 = Both
                 MaterialName = 'CP190RB' # The name of the tendon material. This must be a
material of type tendon.
                TendonArea = ws_long['E163'].value # The tendon cross-sectional area. [L2]
               MaxDiscretizationLength = 200 #The maximum discretization length for the tendon.
[L]
                                                                              ret                =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetTendonParameters(Name,       TendonName,
JackFrom, MaterialName, TendonArea, MaxDiscretizationLength)

                ws_long = long_tipologia.sheets["8 Per. imed."]
                Name = myBridgeObjName
                TendonName = TendonName
                CurvatureCoefficient = 0.24 # The curvature coefficient.
                WobbleCoefficient = 1E-03 #The wobble coefficient.
                AnchorageSetSlip = 0.6 #The anchorage set slip. [L]
```

```
                ElasticShorteningStress = abs(min(ws_long.range("D78:H78").value)/10000) #The
elastic shortening stress. [F/L2]

                ws_long = long_tipologia.sheets["9 Per. prog. Met. Geral"]
                  CreepStress  =  abs(min(ws_long.range("E286:I286").value)/10000)  #  The  creep
stress. [F/L2]
                ShrinkageStress = CreepStress # The shrinkage stress. [F/L2]
                SteelRelaxationStress = abs(min(ws_long.range("E266:I266").value)/10000) # The
steel relaxation stress. [F/L2]
                                                                                ret               =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetTendonLossParameters(Name,    TendonName,
CurvatureCoefficient,     WobbleCoefficient,     AnchorageSetSlip,     ElasticShorteningStress,
CreepStress, ShrinkageStress, SteelRelaxationStress)

                Name = myBridgeObjName
                TendonName = TendonName
                BondingType = 1
                # This is one of the following indicating where the bonding type.
                # 1 = Grouted
                # 2 = Ungrouted
                DuctDiameter = 0 #The diameter of the tendon duct. Zero means no duct exists.
[L]
                                                                                ret               =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetTendonDesignParameters(Name,  TendonName,
BondingType, DuctDiameter)

                ws_long = long_tipologia.sheets["7 Estim. Força de Protensão"]
                Name = myBridgeObjName
                TendonName = TendonName
                 TendonLoadPattern = 'Protensao' # The load pattern in which the tendon loads
are applied.
                LoadType = 0
                # This is one of the following indicating the load type.
                # 0 = Force
                # 1 = Stress
                LoadValue = abs(ws_long['E169'].value) #The load value. [F] for load type Force
and [F/L2] for load type Stress.
                 ret  =  MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetTendonLoad(Name,
TendonName, TendonLoadPattern, LoadType, LoadValue)

                Name = myBridgeObjName
                TendonName = TendonName
                NumberPoints = 2 # The number of points defining the layout.
                SegmentType = [1, 2]
                # A zero-based one-dimensional array defining the segment type at each point.
                # 1 = Start of Tendon
                # 2 = Linear to End of Tendon
                # 3 = Linear
                # 4 = Parabola End Point and Initial Slope
                # 5 = Parabola End Point and Final Slope
                # 6 = Parabola End Point
                # 7 = Parabola Intermediate Point
                # 8 = Circle End Point
                # 9 = Circle Intermediate Point
                 # The first point type must be 1 (Start of Tendon) and the last point must be
2 (Linear to End of Tendon).
                SegmentLengthType = [int(), 2]
                # A zero-based one-dimensional array defining the segment length type for each
segment.
                # 1 = Absolute
                # 2 = Variable
                TendonDistance = [0, CVão- 4] #A zero-based one-dimensional array of the tendon
distance for each point. [L]
                HorizontalOffset = [0, 0] #A zero-based one-dimensional array of the horizontal
offset for each point. [L]
                 Slope = [float(), float()]# A zero-based one-dimensional array of the tendon
slope for each point. This item only applies when the SegmentType is 4 (Parabola End Point and
Initial Slope) or 5 (Parabola End Point and Final Slope).
                                                                                ret               =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetHorizontalLayout(Name,       TendonName,
NumberPoints, SegmentType, SegmentLengthType, TendonDistance, HorizontalOffset, Slope)

                Name = myBridgeObjName
                TendonName = TendonName
                ShapeOption = 7
```

```
                SpanDefinitionOption = 1
                                                                                ret                =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.QuickStartVerticalLayout(Name,   TendonName,
ShapeOption, SpanDefinitionOption)

                Name = myBridgeObjName
                TendonName = TendonName
                NumberPoints = 4
                SegmentType = [1, 7, 6, 2]
                SegmentLengthType = [int(), 2, 2, 2]
                TendonDistance = [0, CVão/2 - 2, CVão -4, CVão -4]
                VerticalOffset = [e_iníciovão, e_meiovão, e_iníciovão, e_iníciovão]
                Slope = [0, float(), float(), 0]
                                                                                ret                =
MyBridgeModeler.BridgeObject.PrestressTendonAssign.SetVerticalLayout(Name,          TendonName,
NumberPoints, SegmentType, SegmentLengthType, TendonDistance, VerticalOffset, Slope)

                Name = myBridgeObjName #The name of a bridge object.
                 TendonNameToCopy = f'TEN{1 + i*(NLongarinasPorVao)}' # The name of the tendon
to be copied. The tendon category of this tendon must be 0 (General).
                DeleteCopiedTendon = False # Indicates if the copied tendon is to be deleted.
                 ProgramDefinedNewTendonName = [] #A zero-based one-dimensional array of the
names that the program assigns to the new tendons.
               ret = MyBridgeModeler.BridgeObject.PrestressTendonAssign.CopyToAllGirders(Name,
TendonNameToCopy, DeleteCopiedTendon, ProgramDefinedNewTendonName)



            # 4 Update no modelo
            Name = myBridgeObjName
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)



             # 5 Coordenadas das longarinas (Pegar todos os pontos que definem as barras das
longarinas)

            ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar os dados em kN, m e Celsius
            ret = SapModel.SelectObj.ClearSelection()
            ret = SapModel.SelectObj.PropertyMaterial(f'C{f_ck} - Longarina', False)
            ret = SapModel.SelectObj.GetSelected(int(), [], [])
            Longarinas_list = list(ret[3])
            Longarinas_points = [] # Nomes dos pontos que definem as longarinas
            for Longarina in Longarinas_list:
                ret = SapModel.FrameObj.GetPoints(Longarina, '', '')
                Longarinas_points.append(ret[1])
                Longarinas_points.append(ret[2])

            Longarinas_pointscoord = []
            for point in Longarinas_points:
                 ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(), float(),
'GLOBAL')
                ret = list(ret)
                ret.pop(0)
                Longarinas_pointscoord.append(ret)

            for i in range(len(Longarinas_list)):
                for j in range(2):
                    Longarinas_pointscoord[i*2+j].insert(0, Longarinas_list[i]) # Inserção dos
nomes dos frames em cada lista

            # 5.1 Obtenção dos Start e End Stations de cada longarina

            ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar esforços em kNm/m e kN/m

             ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)
```

```
            TableKey = 'Frame Bridge Object Flags'
            FieldKeyList = ''
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
              ret  =  SapModel.DatabaseTables.GetTableForDisplayArray(TableKey,  FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

             # 5.2 Criação do dataframe e conversão dos tipos de dados (Associar todas as
longarinas com os pontos que as definem, bem como o Start Station e EndStation)

            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
                    df_longarinas_stations     =     pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
            df_longarinas_stations = df_longarinas_stations.replace({',' : '.'}, regex = True)
# Trocar vírgula por ponto em todas as colunas
            dados_numéricos = ['StartSta', 'EndSta'] # Colunas com dados numéricos
            for dado in dados_numéricos:
                 df_longarinas_stations[dado] = pd.to_numeric(df_longarinas_stations[dado]) #
Mudar string para valor numérico

                                                    df_longarinas_stations                    =
df_longarinas_stations[df_longarinas_stations['BOItemType'] == 'Girder'] # Pegando apenas as
longarinas
            LongarinasStartStations = df_longarinas_stations['StartSta'].tolist()
            LongarinasEndStations = df_longarinas_stations['EndSta'].tolist()

            for i in range(len(Longarinas_list)):
                for j in range(2):
                    Longarinas_pointscoord[i*2+j].insert(0, LongarinasEndStations[i])
                    Longarinas_pointscoord[i*2+j].insert(0, LongarinasStartStations[i])

             df_pontos_longarinas  =  pd.DataFrame(Longarinas_pointscoord,  columns  =  ['Start
Station', 'End Station', 'Frame', 'X', 'Y', 'Z']) # DataFrame com todos os frames, sendo cada
linha um ponto que a define
            df_pontos_longarinas = df_pontos_longarinas.round(4)  # Arredondar todas as colunas
do DataFrame para duas casas decimais
            CoordenadasYLongarinas = df_pontos_longarinas['Y'].sort_values().unique().tolist()
# Coordenadas Y das longarinas em ordem crescente



            # 6 Rodar modelo novamente para realizar a avaliação da protensao (dimensionamento
da protensao)

            # 6.1 Load Cases to be Run

            Name = ''
            Run = True # If this item is True, the specified load case is to be run.
            All = True # If this item is True, the run flag is set as specified by the Run item
for all load cases, and the Name item is ignored.
            ret = SapModel.Analyze.SetRunCaseFlag(Name, Run, All)

            ret = SapModel.Analyze.RunAnalysis()
            ret = SapModel.SetPresentUnits(eUnits(kN_m_C)) # Pegar os dados em kN, m e Celsius

            OutputCase = [] # Lista para filtrar o dataframe posteriormente

            ret = SapModel.LoadPatterns.GetNameList(NumberNames = int(), MyName = [])
            LoadPatternList = list(ret[2])

            ret = SapModel.LoadCases.GetNameList_1(NumberNames = int(), MyName = [])
            LoadCasesList = list(ret[2])
            OutputCase = LoadCasesList # Lista para filtrar o dataframe posteriormente

            ret = SapModel.RespCombo.GetNameList(NumberNames = int(), MyName = [])
            LoadCombinationsList = list(ret[2])
            OutputCase.extend(LoadCombinationsList)
```

```
            # 6.2 Extração dos joints criados no modelo

             ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            ret = SapModel.DatabaseTables.SetLoadPatternsSelectedForDisplay(LoadPatternList)
            ret = SapModel.DatabaseTables.SetLoadCasesSelectedForDisplay(LoadCasesList)
                                                                    ret                          =
SapModel.DatabaseTables.SetLoadCombinationsSelectedForDisplay(LoadCombinationsList)

            TableKey = 'Joint Coordinates'
            FieldKeyList = ''
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
              ret  =  SapModel.DatabaseTables.GetTableForDisplayArray(TableKey,  FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

            # Criação do dataframe e conversão dos tipos de dados (associar todos os resultados
dos esforços aos nós que definem as barras)
            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
            df_joints = pd.DataFrame(ListadeListasDosResultados, columns=ColumnLabels)
            df_joints = df_joints.replace({',' : '.'}, regex = True) # Trocar vírgula por ponto
em todas as colunas
            dados_numéricos = ['XorR', 'Y', 'Z', 'GlobalX', 'GlobalY', 'GlobalZ'] # Colunas com
dados numéricos
            for dado in dados_numéricos:
                 df_joints[dado] = pd.to_numeric(df_joints[dado]) # Mudar string para valor
numérico

            df_joints = df_joints.round(4)
            # Definição dos elementos de joints para a verificação do ELS-DEF (na coordenada Y
das longarinas)

             joints_centrais = df_joints[(df_joints["GlobalX"] <= StartX + CVão/100/2 + 1) &
(df_joints["GlobalX"] >= StartX + CVão/100/2 - 1) &
            (df_joints["GlobalY"].isin(CoordenadasYLongarinas))]['Joint'].unique().tolist()

            # 6.3 Retirada das flechas nos joints
            ret = SapModel.SetPresentUnits(eUnits(kN_mm_C)) # Pegar as flechas em mm

             ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            ret = SapModel.DatabaseTables.SetLoadPatternsSelectedForDisplay(LoadPatternList)
            ret = SapModel.DatabaseTables.SetLoadCasesSelectedForDisplay(LoadCasesList)
                                                                    ret                          =
SapModel.DatabaseTables.SetLoadCombinationsSelectedForDisplay(LoadCombinationsList)

            TableKey = 'Joint Displacements'
            FieldKeyList = ''
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
              ret  =  SapModel.DatabaseTables.GetTableForDisplayArray(TableKey,  FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

            # Criação do dataframe e conversão dos tipos de dados (associar todos os resultados
dos esforços aos nós que definem as barras)
            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
            df_flechas = pd.DataFrame(ListadeListasDosResultados, columns=ColumnLabels)
             df_flechas = df_flechas.replace({',' : '.'}, regex = True) # Trocar vírgula por
ponto em todas as colunas
            dados_numéricos = ['U1', 'U2', 'U3', 'R1', 'R2', 'R3'] # Colunas com dados numéricos
            for dado in dados_numéricos:
```

```
                df_flechas[dado] = pd.to_numeric(df_flechas[dado]) # Mudar string para valor
numérico

            # 6.4 Alimentação da planilha de dimensionamento das longarinas com as flechas
            df_flechas_centrais = df_flechas[df_flechas['Joint'].isin(joints_centrais)]
            ws_long = long_tipologia.sheets["17 ELS-DEF"]
            ws_long['F139'].value = -df_flechas_centrais[(df_flechas_centrais['OutputCase'] ==
'Cronograma-1')]['U3'].abs().max()
            ws_long['F140'].value = -df_flechas_centrais[(df_flechas_centrais["OutputCase"] ==
'TB-450')]['U3'].abs().max()



            # 7 Salvar modelo e planilhas
            ret = SapModel.File.Save()
            long_tipologia.save(long_path)
            long_tipologia.app.quit()
            #ret = mySAPObject.ApplicationExit(False)
            SapModel = None
            mySAPObject = None
            MyBridgeModeler = None
```

```
        # ESSE PROGRAMA FOI FEITO PARA SER USADO EM CONJUNTO COM O 'precast-main'.py APÓS REALIZADA A ANÁLISE DA SUPERESTRUTURA. ELE IRA CRIAR OS MODELOS DOS ENCONTROS COM APENAS UM VÃO, TOMANDO-SE O MODELO DE SUPERESTRUTURA GERADO.
        # INPUTAR AS TIPOLOGIAS A SEREM RODADAS NO VETOR TIPOLOGIAS

        import os
        import sys
        import clr
        import pandas as pd

        from enum import Enum

        import numpy as np

        import xlwings as xw

        clr.AddReference("System.Runtime.InteropServices")
        clr.AddReference(r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge1.dll') #set the following path to the installed CSiBridge program directory
        from System.Runtime.InteropServices import Marshal
        from CSiBridge1 import *

        #set the following flag to True to execute on a remote computer

        Remote = False


        #if the above flag is True, set the following variable to the hostname of the remote computer

        #remember that the remote computer must have CSiBridge installed and be running the CSiAPIService.exe

        RemoteComputer = "SpareComputer-DT"


        #set the following flag to True to attach to an existing instance of the program

        #otherwise a new instance of the program will be started

        AttachToInstance = True


        #set the following flag to True to manually specify the path to CSiBridge.exe

        #this allows for a connection to a version of CSiBridge other than the latest installation

        #otherwise the latest installed version of CSiBridge will be launched

        SpecifyPath = True


        #if the above flag is set to True, specify the path to CSiBridge below

        ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge.exe'

        #full path to the model

        #set it to the desired path of your model

        #if executing remotely, ensure that this folder already exists on the remote computer

        #the below command will only create the folder locally

        main_folder = os.path.dirname(os.path.abspath(__file__))
        APIPath = main_folder

        if not os.path.exists(APIPath):

            try:
```

```
                os.makedirs(APIPath)

            except OSError:

                pass

        def chunk_list(input_list, chunk_size):
        # Create a list of lists with fixed size from another list
            return [input_list[i:i + chunk_size] for i in range(0, len(input_list), chunk_size)]

        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'precast-variables.xlsx')
        variables    =    pd.read_excel(variables_data_file,    sheet_name="Main",    header=2,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-PPC-A-20-P', 'S-PPC-A-25-P', 'S-PPC-A-30-P', 'S-PPC-A-35-P', 'S-PPC-
A-20-S', 'S-PPC-A-25-S', 'S-PPC-A-30-S', 'S-PPC-A-35-S', 'S-PPC-B-20-P', 'S-PPC-B-25-P',
        'S-PPC-B-30-P', 'S-PPC-B-35-P', 'S-PPC-B-20-S', 'S-PPC-B-25-S', 'S-PPC-B-30-S', 'S-
PPC-B-35-S']

        # ['S-PPC-A-20-P', 'S-PPC-A-25-P', 'S-PPC-A-30-P', 'S-PPC-A-35-P', 'S-PPC-A-20-S', 'S-
PPC-A-25-S', 'S-PPC-A-30-S', 'S-PPC-A-35-S', 'S-PPC-B-20-P', 'S-PPC-B-25-P',
        # 'S-PPC-B-30-P', 'S-PPC-B-35-P', 'S-PPC-B-20-S', 'S-PPC-B-25-S', 'S-PPC-B-30-S', 'S-
PPC-B-35-S']

        Tipo_analise = 'ENCONTROS' # Tipos contemplados: SUPERESTRUTURA, ENCONTROS,
MESOESTRUTURA.
        # CASO SEJA ANÁLISE DE MESOESTRUTURA, ALIMENTAR O VETOR DE ALTURA DOS PILARES NA SEÇÃO
DE INPUTS

        for Tipologia in Tipologias: # Rodar todo o código com base em um elemento da lista
Tipologias
            xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
            GeneralPath = os.path.join(main_folder, Tipologia)
            os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à tipologia. O
parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
            ModelPath = os.path.join(GeneralPath, f'{Tipologia} (SUPERESTRUTURA).bdb') #Caminho
do modelo .bdb

            #create API helper object

            helper = cHelper(Helper())


            if AttachToInstance:

                #attach to a running instance of CSiBridge

                try:

                    #get the active CSiBridge object

                    if Remote:

                                        mySAPObject    =    cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                    else:

                        mySAPObject = cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                except:

                    print("No running instance of the program found or failed to attach.")

                    sys.exit(-1)


            else:

                if SpecifyPath:
```

```
                    try:
                        #'create an instance of the CSiBridge object from the specified path
                        if Remote:
                                mySAPObject  =  cOAPI(helper.CreateObjectHost(RemoteComputer,
ProgramPath))
                        else:
                            mySAPObject = cOAPI(helper.CreateObject(ProgramPath))
                    except :
                       print("Cannot start a new instance of the program from " + ProgramPath)
                        sys.exit(-1)
                else:

                    try:
                        #create an instance of the CSiBridge object from the latest installed
CSiBridge
                        if Remote:
                            mySAPObject = cOAPI(helper.CreateObjectProgIDHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))
                        else:
                                                                                mySAPObject      =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))

                    except:
                        print("Cannot start a new instance of the program.")
                        sys.exit(-1)

                #start CSiBridge application
                mySAPObject.ApplicationStart()

            StartX = 0 # Começo do layoutline no modelo

                  #*********************************     INÍCIO     VARIÁVEIS     DE     INPUT
*******************************************#
            f_ck = 35
            Alturas_pilarescentrais = [400, 600, 800] # Análises de MESOESTRUTURA
            NVãos = 1
             df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia

            # Dimensões da longarina

             b_long = df_ponte.loc[:, 'B1': 'B4'].values.flatten().tolist() #B1 a B4 conforme
caixa de diálogo de seção pré-moldada do tipo I
            d_long = df_ponte.loc[:, 'D1': 'D7'].values.flatten().tolist() #D1 a D7
            t_long = df_ponte.loc[:, 'T1': 'T2'].values.flatten().tolist() #T1 a T2
            c1_long = 0 #Bottom flange chamfer dimension
            h_long = d_long[0]
             OverhangLength  =  df_ponte['Balanço  (cm)'].iloc[0]  #balanço  laje/longarina  #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
            NLongarinasPorVão = int(df_ponte['Nlong'].iloc[0])
            NumberInteriorGirders = NLongarinasPorVão - 2 #The number of interior girders
```

```
            TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
            Espaçamento_Long = (TotalWidthOfDeckSection - 2*OverhangLength)/(NLongarinasPorVão-
1)

            # Diâmetro do pilar
            t3_pilar = df_ponte['Dp'].iloc[0]

            # Travessas e transversinas

            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]

            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

            h_transv = df_ponte['hD'].iloc[0]
            bw_transv = df_ponte['bwD'].iloc[0]

            # Tabuleiro

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - b_long[0]

            # Geometria geral ponte

            CVão = df_ponte['Vão (cm)'].iloc[0]
            SpanLength = [CVão] * NVãos #A zero-based array containing the length of each span.
[L]
            CTotal = sum(SpanLength)
            SpanName = []
            for i in range (NVãos):
                SpanName.append(f'Vão {i+1}')
            NDiafragmasPorVão = int(df_ponte['Ndiaf'].iloc[0])
            DiaphragmDistance = [15, 3/8*CVão, 6/8*CVão, CVão-15] #The distance from the start
of the specified span to the diaphragm. [L]
            LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)

            # Aparelho de apoio
            a_ap = df_ponte['aA'].iloc[0]
            b_ap = df_ponte['aA'].iloc[0]
            G_ap = 0.1 # Rigidez do aparelho de apoio em kN/cm² (1 MPa)
            h_ap = df_ponte['hA'].iloc[0] # Altura total do aparelho de apoio
             hd_ap = h_ap - 2*(2.5/10) - 25/10 # Altura de cálculo da rigidez do aparelho de
apoio, subtraindo-se o cobrimento vertical (2,5 mm) nas duas faces e supondo 25 mm em todas as
chapas de fretagem
             d_ap = df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta
            K_ap = 100
            # K_ap = a_ap * b_ap * G_ap / hd_ap

            # Bents

            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = 300 if Tipo_analise == 'ENCONTROS' else df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                  Distance_ext = [Distextp, C_travessaext - Distextp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, C_travessaext/2, C_travessaext - Distextp]
                Height_ext = [Hextp, Hextp, Hextp]

            C_travessaint = df_ponte['Cint'].iloc[0]
```

```
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
            SupportType_int = 1
            if N_pilaresint == 2:
                  Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column centerline measured from the left end of the cap beam. [L]
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to (NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # Laje de Transição
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

            # Muro
            AlturaM = df_ponte['AlturaM'].iloc[0]

            # Ala
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

                  #********************************     FIM     VARIÁVEIS     DE     INPUT ******************************************#

            # 1 Definição dos objetos primitivos
            SapModel = cSapModel(mySAPObject.SapModel)
             MyBridgeModeler = cBridgeModeler_1(SapModel.BridgeModeler_1) # necessário para acessar os métodos do CSiBridge

             ret = SapModel.File.OpenFile(os.path.join(main_folder, Tipologia, f'{Tipologia} (SUPERESTRUTURA).bdb'))
              ret = SapModel.File.Save(os.path.join(main_folder, Tipologia, f'{Tipologia} (ENCONTROS_VÃO ÚNICO).bdb'))
            kN_cm_C = 13
            kN_m_C = 6
            kN_mm_C = 5
            ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

            # 2 Objetos de interesse
            NumberNames = int()
            MyName = []
            ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
            myBridgeObjName = list(ret[2])[0]

            SectionType = 11
            NumberNames = int()
            MyName = []
            BridgeSectionType = [SectionType]
                  ret    =    MyBridgeModeler.DeckSection.GetNameList(NumberNames,    MyName, BridgeSectionType)
            myDeckSectionName = list(ret[2])[0]

            Name = myBridgeObjName
            LayoutLine = ''
            ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
            myLayoutLineName = ret[1]

            ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
            myLaneName = list(ret[2])[0]

            # 3 Atribuição dos Encontros (cBMAbutmentAssign Methods)

            # Redefinição dos bents

            Name = 'Apoios extremos'
            Section = [f'Pilar (C{f_ck})'] * N_pilaresext #A zero-based one-dimensional array dimensioned to (NumberColumns - 1) containing the frame section properties assigned to each column.
```

```
            Angle = [0] * N_pilaresext #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the orientation of the column in degrees counterclockwise between
a line parallel to the bent and the column local 2 axis. [deg]
             Foundation = ['Fixed'] * N_pilaresext #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing either Fixed, Pinned, a defined foundation spring,
or Previous. Note that an error will occur if Previous is assigned to the first column.
            ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_ext, Height_ext,
Angle, Foundation)

            Elevation_StartAbutment = -h_laje -h_long - h_ap #The global Z elevation of the
substructure. (start abutment) -> Distância entre o topo da laje e o CG do aparelho de apoio
             ElevationAtLayoutLine_StartAbutment = -h_laje -h_long - h_ap/2 #The bearing
elevation (global Z) at the layout line. [L] -> Distância entre o topo da laje e a base do
aparelho de apoio
            Elevation_EndAbutment = Elevation_StartAbutment #The global Z elevation of the
substructure.
             ElevationAtLayoutLine_EndAbutement = ElevationAtLayoutLine_StartAbutment #The
bearing elevation (global Z) at the layout line. [L]

            # 3.1 Start Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 1
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'Start Abutment' #The support name. The support name cannot be blank
and it can not be the same as another abutment or bent support name
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
                                                                ret                          =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            SubstructureType = 2
            AbutmentOrBentProperty = 'Apoios extremos'
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
             ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation,    SubstructureType,    AbutmentOrBentProperty,    Elevation_StartAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
             # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
             NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
```

```
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
               ret   =   MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_StartAbutment,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            # 3.2 End Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 2
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'End Abutment'
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
                                                                    ret                          =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            SubstructureType = 2
             AbutmentOrBentProperty = 'Apoios extremos' #The name of the abutment or bent
property used for the substructure. [L]
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation,     SubstructureType,     AbutmentOrBentProperty,     Elevation_EndAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
             NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
               ret   =   MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_EndAbutement,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            ret = MyBridgeModeler.Abutment.Delete('BABT1')
```

```
            # Update no modelo
            Name = myBridgeObjName
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)



            # 4 Modelagem do encontro

            if Tipo_analise == 'ENCONTROS':

                # 16.1 Alas
                OffsetEncontro = 75

                # Ala da direita
                NumberPoints = 4 # The number of points in the area abject.
                    x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-CompA,  -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                # Escolheu-se o sentido anti-horário, sendo o primeiro ponto o mais próximo da
laje (em x e z)
               # The coordinates should be ordered to run clockwise or counter clockwise around
the area object.
                Name = ''
                 # This is the name that the program ultimately assigns to the area object. If
no UserName is specified, the program assigns a default name to the area object.
                # If a UserName is specified and that name is not used for another area object,
the UserName is assigned to the area object; otherwise a default name is assigned to the area
object.
                PropName = 'Encontro'
                # This is Default, None or the name of a defined area property.
                 # If it is Default, the program assigns a default area property to the area
object.
                # If it is None, no area property is assigned to the area object. If it is the
name of a defined area property, that property is assigned to the area object.
                UserName = 'Ala direita 1/2' # This is an optional user specified name for the
area object. If a UserName is specified and that name is already used for another area object,
the program ignores the UserName.
                 CSys = 'GLOBAL' # The name of the coordinate system in which the area object
point coordinates are defined.
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                   x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-ChanfroH,  -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala direita 2/2'
                CSys = 'GLOBAL'
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala direita 1/2' # The name of an existing area object.
                MeshType = 1
                # This item is 1, 2, 3, 4, 5 or 6, indicating the mesh type for the area object.
                # 1 = Mesh area into a specified number of objects
                # 2 = Mesh area into objects of a specified maximum size
                # 3 = Mesh area based on points on area edges
                # 4 = Cookie cut mesh area based on lines intersecting edges
                # 5 = Cookie cut mesh area based on points
                # 6 = Mesh area using General Divide Tool
```

```
                AreaName = [] # This is an array of the name of each area object created when
the specified area object is divided.
                n1_aladireita1 = 10 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 2.
                n2_aladireita1 = 5 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 3.
                NumberAreas = n1_aladireita1*n2_aladireita1 #The number of area objects created
when the specified area object is divided.
                    ret  =  SapModel.EditArea.Divide(Name,  MeshType,  NumberAreas,  AreaName,
n1_aladireita1, n2_aladireita1)
                Ala_direita = list(ret[2])

                Name = 'Ala direita 2/2'
                MeshType = 1
                AreaName = []
                n1_aladireita2 = 10
                n2_aladireita2 = 10
                NumberAreas = n1_aladireita2*n2_aladireita2
                    ret  =  SapModel.EditArea.Divide(Name,  MeshType,  NumberAreas,  AreaName,
n1_aladireita2, n2_aladireita2)
                Ala_direita.extend(list(ret[2]))

                  ret  =  SapModel.GroupDef.SetGroup('Ala  direita')  #  This  is  the  name  of  a
group.  If this is the name of an existing group,  that group is modified, otherwise a new group
is added.
                for Area in Ala_direita:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala direita' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # Ala da esquerda

                NumberPoints = 4
                    x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-CompA,  -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 1/2'
                CSys = 'GLOBAL'
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                  x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-ChanfroH,  -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 2/2'
                CSys = 'GLOBAL'
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Discretização
```

```
                Name = 'Ala esquerda 1/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda1 = 10
                n2_alaesquerda1 = 5
                NumberAreas = n1_alaesquerda1*n2_alaesquerda1
                    ret  =  SapModel.EditArea.Divide(Name,  MeshType,  NumberAreas,  AreaName,
n1_alaesquerda1, n2_alaesquerda1)
                Ala_esquerda = list(ret[2])

                Name = 'Ala esquerda 2/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda2 = 10
                n2_alaesquerda2 = 10
                NumberAreas = n1_alaesquerda2*n2_alaesquerda2
                    ret  =  SapModel.EditArea.Divide(Name,  MeshType,  NumberAreas,  AreaName,
n1_alaesquerda2, n2_alaesquerda2)
                Ala_esquerda.extend(list(ret[2]))

                  ret  =  SapModel.GroupDef.SetGroup('Ala  esquerda')  #  This  is  the  name  of  a
group.  If this is the name of an existing group,  that group is modified, otherwise a new group
is added.
                for Area in Ala_esquerda:
                     # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                     GroupName = 'Ala esquerda' # The name of an existing group to which the
assignment is made.
                      Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                     # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                     # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                     # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 3.2 Parede cortina
                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, -ChanfroV, -ChanfroV, 0]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 1/2'
                CSys = 'GLOBAL'
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -AlturaM, -AlturaM, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 2/2'
                CSys = 'GLOBAL'
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Discretização
                Name = 'Cortina 1/2'
                MeshType = 1
                AreaName = []
                n1_cortina1 = 5
                n2_cortina1 = 20
```

```
                NumberAreas = n1_cortina1*n2_cortina1
                    ret  =  SapModel.EditArea.Divide(Name,  MeshType,  NumberAreas,  AreaName,
n1_cortina1, n2_cortina1)
                Cortina = list(ret[2])

                Name = 'Cortina 2/2'
                MeshType = 1
                AreaName = []
                n1_cortina2 = 10
                n2_cortina2 = 20
                NumberAreas = n1_cortina2*n2_cortina2
                    ret  =  SapModel.EditArea.Divide(Name,  MeshType,  NumberAreas,  AreaName,
n1_cortina2, n2_cortina2)
                Cortina.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Cortina')
                for Area in Cortina:
                    GroupName = 'Cortina'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 3.3 Console
                NumberPoints = 4
                Largura_console = 30
                   x  =  [-OffsetEncontro,  -OffsetEncontro-Largura_console,  -OffsetEncontro-
Largura_console, -OffsetEncontro]
                 y = [-TotalWidthOfDeckSection/2 + hAla, -TotalWidthOfDeckSection/2 + hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                    z  =  [-2*(ChanfroV/n2_aladireita1),  -2*(ChanfroV/n2_aladireita1),  -
2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1)] # Convencionado dessa forma para que
a discretização fique robusta
                Name = ''
                PropName = 'Encontro'
                UserName = 'Console'
                CSys = 'GLOBAL'
                  ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Determinação dos pontos para realizar a discretização
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Cortina', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_cortina = list(ret[3])
                Points_cortina = []
                for Area in Areas_cortina:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_cortina.extend(list(ret[2]))

                Pointscortina_coord = []
                for point in Points_cortina:
                   ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(), float(),
'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointscortina_coord.append(ret)

                for i in range(len(Pointscortina_coord)):
                    Pointscortina_coord[i].insert(0,Points_cortina[i])

                df_pontos_cortina = pd.DataFrame(Pointscortina_coord, columns = ['Joint', 'X',
'Y', 'Z'])
                df_pontos_cortina = df_pontos_cortina.round(4)

                     Pointscortina_mesh  =  df_pontos_cortina[(df_pontos_cortina["Z"]  ==  -
2*(ChanfroV/n2_aladireita1))]['Joint'].unique().tolist()

                ret = SapModel.SelectObj.ClearSelection()
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização
                Name = 'Console'
                MeshType = 3
                AreaName = []
```

```
                n1_console = 2
                n2_console = 20
                NumberAreas = n1_console*n2_console
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)

                Console = list(ret[2])
                ret = SapModel.GroupDef.SetGroup('Console') # This is the name of a group.  If
this is the name of an existing group,  that group is modified, otherwise a new group is added.
                for Area in Console:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                  GroupName = 'Console' # The name of an existing group to which the assignment
is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 3.4 Laje de transição
                OffsetLT = OffsetEncontro + Largura_console
                NumberPoints = 4
                x = [-OffsetLT, -OffsetLT - LarguraLT, -OffsetLT - LarguraLT, -OffsetLT]
                 y = [-TotalWidthOfDeckSection/2 + hAla, -TotalWidthOfDeckSection/2 + hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-h_laje/2, -h_laje/2, -h_laje/2, -h_laje/2]
                Name = ''
                PropName = 'Laje de transição'
                UserName = 'LT'
                CSys = 'GLOBAL'
                  ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Obtenção dos pontos de discretização
                ret = SapModel.SelectObj.ClearSelection()
                Pointscortina_mesh = Pointscortina_mesh[1:-1]
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                dx = - float(Largura_console)
                dz = 2*(ChanfroV/n2_aladireita1) - h_laje/2
                ret = SapModel.EditGeneral.ReplicateLinear(dx, float(0), dz, int(1), int(), [],
[], False)

                PointsLT_mesh = list(ret[2])
                ret = SapModel.SelectObj.ClearSelection()
                for point in PointsLT_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização nos pontos do console
                Name = 'LT'
                MeshType = 3
                AreaName = []
                n1_LT = 10
                n2_LT = 20
                NumberAreas = n1_LT*n2_LT
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)
                LT = list(ret[2])

                # Discretização LT
                LT_adic = []
                for Area in LT:
                    ret = SapModel.EditArea.Divide(Area, 1, int(), AreaName, 10, 1)
```

```
                    LT_adic.extend(list(ret[2]))

                LT.extend(LT_adic)
                ret = SapModel.GroupDef.SetGroup('LT') # This is the name of a group.  If this
is the name of an existing group,  that group is modified, otherwise a new group is added.

                for Area in LT:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'LT' # The name of an existing group to which the assignment
is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 3.5 Links laje de transição
                XMin = -OffsetLT - LarguraLT
                XMax = -OffsetLT - LarguraLT/2
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = False
                Line = False
                Area = True
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                Name = '' # The name of an existing area object or group, depending on the
value of the ItemType item.
                MyType = 1
                # This is either 1 or 2, indicating the spring property type.
                # 1 = Simple spring
                # 2 = Link property
                s = 0.05 # The simple spring stiffness per unit area of the specified area
object face. This item applies only when MyType = 1. [F/L3]
                SimpleSpringType = 2
                # This is 1, 2 or 3, indicating the simple spring type. This item applies only
when MyType = 1.
                # 1 = Spring resists tension and compression
                # 2 = Spring resists compression only
                # 3 = Spring resists tension only
                LinkProp = '' # The name of the link property assigned to the spring. This item
applies only when MyType = 2.
                Face = -2
                # This is -1, -2 or a nonzero, positive integer indicating the area object face
to which the specified spring assignment applies.
                # -1 = Bottom face
                # -2 = Top face
                # >0 = Edge face
                SpringLocalOneType = 2
                # This is 1, 2 or 3, indicating the method used to specify the spring positive
local 1-axis orientation.
                # 1 = Parallel to area object local axis
                # 2 = Normal to specified area object face
                # 3 = User specified direction vector
```

```
                Dir = int() # This is 1, 2, 3, -1, -2 or -3, indicating the area object local
axis that corresponds to the positive local 1-axis of the spring. This item applies only when
SpringLocalOneType = 1.
                 Outward = False # This item is True if the spring positive local 1 axis is
outward from the specified area object face. This item applies only when SpringLocalOneType =
2.
                Vec = [] # This is an array of three values that define the direction vector
of the spring positive local 1-axis.
                # The direction vector is in the coordinate system specified by the CSys item.
This item applies only when SpringLocalOneType = 3.
                Ang = float() # This is the angle that the link local 2-axis is rotated from
its default orientation. This item applies only when MyType = 2. [deg]
                 Replace = True #If this item is True, all existing spring assignments to the
area object are removed before assigning the specified spring.
                # If it is False, the specified spring is added to any existing springs already
assigned to the area object.
                CSys = 'Local' #This is Local (meaning the area object local coordinate system)
or the name of a defined coordinate system.
                 #This item is the coordinate system in which the user specified direction
vector, Vec, is specified. This item applies only when SpringLocalOneType = 3.
                ItemType = eItemType(2)
                ret = SapModel.AreaObj.SetSpring(Name, MyType, s, SimpleSpringType, LinkProp,
Face, SpringLocalOneType, Dir, Outward, Vec, Ang, Replace, CSys, ItemType)

                ret = SapModel.SelectObj.ClearSelection()

                # 3.6 Links console
                XMin = -OffsetLT
                XMax = -OffsetLT
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = True
                Line = False
                Area = False
                Solid = False
                Link = False
                 ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])

                Points_linkconsole = list(ret[3])
                x = []
                y = []
                z = []
                for point in Points_linkconsole:
                   ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(), float(),
'GLOBAL')
                    x.append(ret[1])
                    y.append(ret[2])
                    z.append(ret[3])

                ret = SapModel.SelectObj.ClearSelection()
                Links_console = []
                for i in range(len(Points_linkconsole)):
                     xi = x[i] # The coordinates of the I-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item.
                    yi = y[i]
                    zi = z[i]
                     xj = x[i] # The coordinates of the J-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item. These coordinates are ignored
if the IsSingleJoint item is True.
                    yj = y[i]
                    zj = -2*(ChanfroV/n2_aladireita1)
                    Name = ''
                    # This is the name that the program ultimately assigns for the link object.
                     # If no UserName is specified, the program assigns a default name to the
link object.
                     # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
```

```
                    IsSingleJoint = False # This item is True if a one-joint link is added and
False if a two-joint link is added.
                    PropName = 'Link rigido'
                    UserName = ''
                    CSys = 'GLOBAL'
                        ret = SapModel.LinkObj.AddByCoord(xi, yi, zi, xj, yj, zj, Name,
IsSingleJoint, PropName, UserName, CSys)
                    Links_console.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links console')
                for link in Links_console:
                    GroupName = 'Links console'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 3.7 Links travessa # Nao gostei muito dessa solução, investigar outra forma
de fazer
                Name = f'Travessa encontro (C{f_ck})'
                DeSelect = False
                ret = SapModel.SelectObj.PropertyFrame(Name, DeSelect)

                NumberItems = int() # The number of selected objects.
                ObjectType = []
                # This is an array that includes the object type of each selected object.
                # 1 = Point object
                # 2 = Frame object
                # 3 = Cable object
                # 4 = Tendon object
                # 5 = Area object
                # 6 = Solid object
                # 7 = Link object
                 ObjectName = [] # This is an array that includes the name of each selected
object.
                ret = SapModel.SelectObj.GetSelected(NumberItems, ObjectType, ObjectName)

                Name = f'{list(ret[3])[0]}'
                Point1 = ''
                Point2 = ''
                ret = SapModel.FrameObj.GetPoints(Name, Point1, Point2)

                Name = ret[1]
                x = float()
                y = float()
                z = float()
                CSys = 'GLOBAL'
                ret = SapModel.PointObj.GetCoordCartesian(Name, x, y, z, CSys)
                xiTravessa = ret[1]
                yiTravessa = ret[2]
                ziTravessa = ret[3]

                zi = [-AlturaM, -AlturaM + ((AlturaM - ChanfroV)/(n1_cortina2))* 6]
                Links_travessa = []
                for j in range (2):
                    for i in range (n2_cortina2+1):
                        xi = -OffsetEncontro # The coordinates of the I-End of the added link
object. The coordinates are in the coordinate system defined by the CSys item.
                      yi = -TotalWidthOfDeckSection/2 + hAla/2 - i*(-TotalWidthOfDeckSection/2
+ hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                      xj = xiTravessa # The coordinates of the J-End of the added link object.
The coordinates are in the coordinate system defined by the CSys item. These coordinates are
ignored if the IsSingleJoint item is True.
                     yj = -TotalWidthOfDeckSection/2 + hAla/2 - i*(-TotalWidthOfDeckSection/2
+ hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                      zj = ziTravessa
                      Name = ''
                       # This is the name that the program ultimately assigns for the link
object.
                       # If no UserName is specified, the program assigns a default name to
the link object.
                      # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                      IsSingleJoint = False # This item is True if a one-joint link is added
and False if a two-joint link is added.
```

```
                        PropName = 'Link rigido'
                        UserName = ''
                        CSys = 'GLOBAL'
                          ret = SapModel.LinkObj.AddByCoord(xi, yi, zi[j], xj, yj, zj, Name,
IsSingleJoint, PropName, UserName, CSys)
                        Links_travessa.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links travessa')
                for link in Links_travessa:
                    GroupName = 'Links travessa'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 3.8 Definição dos grupos e mirror

                ret = SapModel.GroupDef.SetGroup('Encontro')

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Links console')
                ret = SapModel.SelectObj.Group('Links travessa')
                ret = SapModel.LinkObj.SetGroupAssign('', 'Encontro', False, eItemType(2))

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Ala direita')
                ret = SapModel.SelectObj.Group('Ala esquerda')
                ret = SapModel.SelectObj.Group('Console')
                ret = SapModel.SelectObj.Group('Cortina')
                ret = SapModel.SelectObj.Group('LT')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Encontro', False, eItemType(2))

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Encontro')

                Plane = 1
                # This is 1, 2, 3 or 4, indicating the mirror plane type.
                # 1 = Parallel to Z
                # 2 = Parallel to X
                # 3 = Parallel to Y
                # 4 = 3D plane
                x1 = CTotal/2
                y1 = TotalWidthOfDeckSection/2
                z1 = float()
                x2 = CTotal/2
                y2 = -TotalWidthOfDeckSection/2
                z2 = float()
                x3 = float()
                y3 = float()
                z3 = float()
                 # These are the coordinates of three points used to define the mirror plane.
[L]
                # When Plane = 1, x1, y1, x2 and y2 define the intersection of the mirror plane
with the XY plane.
                # When Plane = 2, y1, z1, y2 and z2 define the intersection of the mirror plane
with the YZ plane.
                # When Plane = 3, x1, z1, x2 and z2 define the intersection of the mirror plane
with the XZ plane.
                 # When Plane = 4, x1, y1, z1, x2, y2, z2, x3, y3 and z3 define three points
that define the mirror plane.
                 NumberObjects = int() # The number of new objects created by the replication
process.
                 ObjectName = [] # This is an array of the name of each object created by the
replication process.
                ObjectType = []
                  # This is an array of the type of each object created by the replication
process.
                # 1 = Point object
                # 2 = Frame object
                # 3 = Cable object
                # 4 = Tendon object
                # 5 = Area object
                # 6 = Solid object
                # 7 = Link object
                 Remove = False # If this item is True, the originally selected objects are
deleted after the replication is complete.
```

```
                ret = SapModel.EditGeneral.ReplicateMirror(Plane, x1, y1, z1, x2, y2, z2, x3,
y3, z3, NumberObjects, ObjectName, ObjectType)

                ret = SapModel.GroupDef.SetGroup('Todas lajes')
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Laje de transição')
                ret = SapModel.SelectObj.Group(f'<{myBridgeObjName}>SLAB')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Todas lajes', False, eItemType(2))
                ret = SapModel.SelectObj.ClearSelection()

                # 3.9 Atribuição dos carregamentos nos encontros

                ret = SapModel.SelectObj.ClearSelection()
                modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
                wb_pil_ext = xw.Book(modelo_pil_ext)
                ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

                EmpuxoTerra = ws_pil_ext['D82'].value/(100)**2
                EmpuxoTerraCargaMóvel = ws_pil_ext['D93'].value/(100)**2
                 Cases = [['Ala direita', 'Empuxo terra', -EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
                ['Ala direita', 'Empuxo terra carga móvel', -EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
               ['Ala esquerda', 'Empuxo terra', EmpuxoTerra, 5, False, 'GLOBAL', eItemType(1)],
                ['Ala esquerda', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra', EmpuxoTerra, 4, False, 'GLOBAL', eItemType(1)],
                  ['Cortina', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 4, False,
'GLOBAL', eItemType(1)]]

                for case in Cases:
                    Name, LoadPat, Value, Dir, Replace, CSys, ItemType = case
                     ret = SapModel.AreaObj.SetLoadUniform(Name, LoadPat, Value, Dir, Replace,
CSys, ItemType)

                # 3.10 Impacto na defensa

                # 3.10.1 Obtenção dos pontos do encontro para a atribuição do carregamento

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Encontro', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_encontro = list(ret[3])
                Points_encontro = [] # Nomes dos pontos que definem as áreas
                for Area in Areas_encontro:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_encontro.extend(list(ret[2]))

                Pointsencontro_coord = []
                for point in Points_encontro:
                   ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(), float(),
'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointsencontro_coord.append(ret)

                for i in range(len(Pointsencontro_coord)):
                     Pointsencontro_coord[i].insert(0,Points_encontro[i]) # Inserção dos nomes
dos pontos em cada lista

               # 3.10.2 Criação do dataframe com todos os elementos de área e os quatro pontos
que os definem (start e end station será feito depois de discretizar a laje)
                 df_pontos_encontro = pd.DataFrame(Pointsencontro_coord, columns = ['Joint',
'X', 'Y', 'Z']) # DataFrame com todas as áreas, sendo cada linha um ponto que a define
                df_pontos_encontro = df_pontos_encontro.round(4) # Arredondar todas as colunas
do DataFrame para 4 casas decimais

                # 3.10.3 Obtenção dos pontos

                  Points_impacto_esq = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) &
(df_pontos_encontro["Y"] == -TotalWidthOfDeckSection/2 + hAla/2) & (df_pontos_encontro["X"] <=
-OffsetEncontro)]['Joint'].unique().tolist()
                  Points_impacto_dir = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) &
(df_pontos_encontro["Y"] == TotalWidthOfDeckSection/2 - hAla/2) & (df_pontos_encontro["X"] <= -
OffsetEncontro)]['Joint'].unique().tolist()
```

```
                # 3.10.4 Atribuição do carregamento
                fdist_impacto = 100/(50 + 2*87) # em kN/cm
                fno_impacto = (fdist_impacto * CompA)/len(Points_impacto_esq)

                mdist_impacto = 100*87/(50 + 2*87) # em kN.cm/cm
                mno_impacto = mdist_impacto*CompA/len(Points_impacto_esq)

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_esq:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , fno_impacto, 0, -
mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_dir:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , -fno_impacto, 0,
mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))



            # 4 Definição das Lanes - (cBMLane.SetLaneFromLayoutLine) [myLaneName sempre será
a faixa de tráfego]

            Name = myLaneName #The name for the existing lane defined from a layout line.
            NumberLaneStations = 2 #The number of lane points, that is, the number of items
included in the that lane will be overwritten. LayoutLine, Station, Offset, Width and Radius
arrays.
            LayoutLine = [myLayoutLineName, myLayoutLineName]#A zero-based array containing the
name of the bridge layout line associated with each lane station.
            Station = [0, CTotal]#A zero-based array containing the station at each lane
station. [L]
            Offset = [0, 0]#A zero-based array containing the offset from the layout line to
the center of the lane at each lane station. [L]
            Width = [LarguraRolamento, LarguraRolamento]#A zero-based array containing the the
width of the lane at each lane station. [L]
            Radius = [0,0]#A zero-based array containing the the radius of the lane at each
lane station. A positive radius curves to the right and negative to the left. [L]
            LaneType = 0
            #The lane type (fixed or floating).
            # 0 = Fixed lane
            # 1 = Floating lane set
            if Tipo_analise == 'ENCONTROS':
                LoadGroup = 'Todas lajes'
            else:
                LoadGroup = f'<{myBridgeObjName}>SLAB'
            LeftEdgeIsExterior = False #Indicates whether the left edge of the lane is exterior
or interior.
            RightEdgeIsExterior = False #Indicates whether the right edge of the lane is
exterior or interior.
            MaxDiscAlongLane = 300 #The maximum lane load discretization length along the lane.
[L]
            MaxDiscAcrossLane = 300 #The maximum lane load discretization length across the
lane. [L]
            UseDiscFactorSpan = True #Indicates whether the the DiscFactorSpan item is to apply
to the lane.
            UseDiscFactorLane = True #Indicates whether the the DiscFactorLane item is to apply
to the lane.
            DiscFactorSpan = 4 #The lane load discretization length along the lane will not be
more than 1/x of the span length where x is the DiscFactorSpan value.
            #This item is only applicable when the UseDiscFactorSpan item is True.
            DiscFactorLane = 10 #The lane load discretization length along the lane will not
be more than 1/x of the lane length where x is the DiscFactorLane value.
            #This item is only applicable when the UseDiscFactorLane item is True.
            Color = int()#The color assigned to the lane. (Optional)
            Notes = ''#The notes, if any, assigned to the lane. (Optional)
               ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine(Name, NumberLaneStations,
LayoutLine, Station, Offset, Width, Radius, LaneType, LoadGroup, LeftEdgeIsExterior,
RightEdgeIsExterior,
                  MaxDiscAlongLane, MaxDiscAcrossLane, UseDiscFactorSpan, UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

            # Laje de transição inicial
            ret = MyBridgeModeler.LayoutLine.AddNew('BLL0')
```

```
            Name = 'BLL0'
            CoordinateSystem = 'GLOBAL'
            StartX = -1000
            StartY = 0
            StartZ = 0
             ret = MyBridgeModeler.LayoutLine.SetGeneralData(Name, CoordinateSystem, StartX,
StartY, StartZ)

            Name = 'BLL0'
            NumberSegments = 2
            SegmentType = [1, 2]
            Station = [0, 1000]
            Radius = [float(), float()]
            Bearing = ['N900000E', 'N900000E']
              ret = MyBridgeModeler.LayoutLine.SetHorizontalLayoutData(Name, NumberSegments,
SegmentType, Station, Radius, Bearing)

            ret = MyBridgeModeler.Lane.AddNewFromLayoutLine('LANE0', 'BLL0')
            ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine('LANE0', 2, ['BLL0', 'BLL0'], [0,
1000], [0]*2, [LarguraRolamento]*2, [0]*2, LaneType, LoadGroup, LeftEdgeIsExterior,
RightEdgeIsExterior,
              MaxDiscAlongLane, MaxDiscAcrossLane, UseDiscFactorSpan, UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

            LoadNumber = 1 #The load assignment number.
             NumberItems = 2 #The number of lanes loaded for the specified load assignment
number.
             MyName = [f'{myLaneName}', 'LANE0'] #This is an array that includes the name of
each lane loaded for the specified load assignment number.
            ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber, NumberItems,
MyName)
            ret = SapModel.LoadCases.Moving.SetLanesLoaded('Multidao', LoadNumber, NumberItems,
MyName)

            # Laje de transição final
            # ret = MyBridgeModeler.LayoutLine.AddNew('BLL2')

            # Name = 'BLL2'
            # CoordinateSystem = 'GLOBAL'
            # StartX = CTotal
            # StartY = 0
            # StartZ = 0
            # ret = MyBridgeModeler.LayoutLine.SetGeneralData(Name, CoordinateSystem, StartX,
StartY, StartZ)

            # Name = 'BLL2'
            # NumberSegments = 2
            # SegmentType = [1, 2]
            # Station = [0, 1000]
            # Radius = [float(), float()]
            # Bearing = ['N900000E', 'N900000E']
             # ret = MyBridgeModeler.LayoutLine.SetHorizontalLayoutData(Name, NumberSegments,
SegmentType, Station, Radius, Bearing)

            # ret = MyBridgeModeler.Lane.AddNewFromLayoutLine('LANE2', 'BLL2')
             # ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine('LANE2', 2, ['BLL2', 'BLL2'],
[0, 1000], [0]*2, [LarguraRolamento]*2, [0]*2, LaneType, LoadGroup, LeftEdgeIsExterior,
RightEdgeIsExterior,
              # MaxDiscAlongLane, MaxDiscAcrossLane, UseDiscFactorSpan, UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

            # LoadNumber = 1 #The load assignment number.
             # NumberItems = 3 #The number of lanes loaded for the specified load assignment
number.
            # MyName = [f'{myLaneName}', 'LANE0', 'LANE2'] #This is an array that includes the
name of each lane loaded for the specified load assignment number.
               # ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber,
NumberItems, MyName)
               # ret = SapModel.LoadCases.Moving.SetLanesLoaded('Multidao', LoadNumber,
NumberItems, MyName)

            # 5 Update no modelo e rodar análise
            Name = myBridgeObjName
```

```
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)

            Name = ''
            Run = True # If this item is True, the specified load case is to be run.
            All = True # If this item is True, the run flag is set as specified by the Run item
for all load cases, and the Name item is ignored.
            ret = SapModel.Analyze.SetRunCaseFlag(Name, Run, All)

            ret = SapModel.Analyze.RunAnalysis()

            # 6 Salvar modelo e planilhas
            ret = SapModel.File.Save()
            #ret = mySAPObject.ApplicationExit(False)
            SapModel = None
            mySAPObject = None
            MyBridgeModeler = None
```

```
        # ESSE PROGRAMA APENAS SERÁ USADO PARA A CRIAÇÃO DOS MODELOS DE ANÁLISE DE MESOESTRUTURA
COM VÃOS DIFERENTES, NÃO SENDO UTILIZADO PARA NENHUM OUTRO FIM. ELE APENAS CRIARÁ O MODELO E O
DEIXARÁ RODADO.

        import os
        import sys
        import clr
        import pandas as pd
        from enum import Enum
        import numpy as np
        import xlwings as xw

        clr.AddReference("System.Runtime.InteropServices")
        clr.AddReference(r'C:\Program      Files\Computers      and      Structures\CSiBridge
25\CSiBridge1.dll') #set the following path to the installed CSiBridge program directory
        from System.Runtime.InteropServices import Marshal
        from CSiBridge1 import *

        #set the following flag to True to execute on a remote computer

        Remote = False

        #if the above flag is True, set the following variable to the hostname of the remote
computer

        #remember that the remote computer must have CSiBridge installed and be running the
CSiAPIService.exe

        RemoteComputer = "SpareComputer-DT"

        #set the following flag to True to attach to an existing instance of the program

        #otherwise a new instance of the program will be started

        AttachToInstance = True

        #set the following flag to True to manually specify the path to CSiBridge.exe

        #this  allows  for  a  connection  to  a  version  of  CSiBridge  other  than  the  latest
installation

        #otherwise the latest installed version of CSiBridge will be launched

        SpecifyPath = True

        #if the above flag is set to True, specify the path to CSiBridge below

        ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 25\CSiBridge.exe'

        #full path to the model

        #set it to the desired path of your model

        #if executing remotely, ensure that this folder already exists on the remote computer

        #the below command will only create the folder locally

        main_folder = os.path.dirname(os.path.abspath(__file__))
        APIPath = main_folder

        if not os.path.exists(APIPath):

            try:

                os.makedirs(APIPath)

            except OSError:

                pass

        def chunk_list(input_list, chunk_size):
        # Create a list of lists with fixed size from another list
            return  [input_list[i:i  +  chunk_size]  for  i  in  range(0,  len(input_list),
chunk_size)]
```

```
        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'precast-variables.xlsx')
        variables    =    pd.read_excel(variables_data_file,    sheet_name="Main",    header=2,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-PPC-A-20-P', 'S-PPC-A-25-P', 'S-PPC-A-30-P', 'S-PPC-A-35-P', 'S-PPC-
A-20-S', 'S-PPC-A-25-S', 'S-PPC-A-30-S', 'S-PPC-A-35-S', 'S-PPC-B-20-P', 'S-PPC-B-25-P',
        'S-PPC-B-30-P', 'S-PPC-B-35-P', 'S-PPC-B-20-S', 'S-PPC-B-25-S', 'S-PPC-B-30-S', 'S-
PPC-B-35-S']

        # ['S-PPC-A-20-P', 'S-PPC-A-25-P', 'S-PPC-A-30-P', 'S-PPC-A-35-P', 'S-PPC-A-20-S', 'S-
PPC-A-25-S', 'S-PPC-A-30-S', 'S-PPC-A-35-S', 'S-PPC-B-20-P', 'S-PPC-B-25-P',
        # 'S-PPC-B-30-P', 'S-PPC-B-35-P', 'S-PPC-B-20-S', 'S-PPC-B-25-S', 'S-PPC-B-30-S', 'S-
PPC-B-35-S']

        Tipo_analise = 'MESOESTRUTURA' # Tipos contemplados: SUPERESTRUTURA, ENCONTROS,
MESOESTRUTURA.
        # CASO SEJA ANÁLISE DE MESOESTRUTURA, ALIMENTAR O VETOR DE ALTURA DOS PILARES NA SEÇÃO
DE INPUTS

        for Tipologia in Tipologias: # Para cada tipologia, será variado a altura dos pilares
e também o comprimento dos vãos

            #**********************************      INÍCIO      VARIÁVEIS      DE      INPUT
*****************************************#
            f_ck = 35
            Alturas_pilarescentrais = [400, 600, 800] # Análises de MESOESTRUTURA
            NVãos = 4 # Para análises de mesoestrutura, serão considerados 4 vãos
            df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia

            # Dimensões da longarina

            b_long = df_ponte.loc[:, 'B1': 'B4'].values.flatten().tolist() #B1 a B4 conforme
caixa de diálogo de seção pré-moldada do tipo I
            d_long = df_ponte.loc[:, 'D1': 'D7'].values.flatten().tolist() #D1 a D7
            t_long = df_ponte.loc[:, 'T1': 'T2'].values.flatten().tolist() #T1 a T2
            c1_long = 0 #Bottom flange chamfer dimension
            h_long = d_long[0]
            OverhangLength = df_ponte['Balanço (cm)'].iloc[0] #balanço laje/longarina #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
            NLongarinasPorVão = int(df_ponte['Nlong'].iloc[0])
            NumberInteriorGirders = NLongarinasPorVão - 2 #The number of interior girders
            TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
            Espaçamento_Long             =             (TotalWidthOfDeckSection             -
2*OverhangLength)/(NLongarinasPorVão-1)

            # Diâmetro do pilar
            t3_pilar = df_ponte['Dp'].iloc[0]

            # Travessas e transversinas

            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]

            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

            h_transv = df_ponte['hD'].iloc[0]
            bw_transv = df_ponte['bwD'].iloc[0]

            # Tabuleiro

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - b_long[0]

            # Geometria geral ponte

            CVão = df_ponte['Vão (cm)'].iloc[0]
            SpanLength = [CVão] * NVãos #A zero-based array containing the length of each span.
[L]
```

```
            CTotal = sum(SpanLength)
            SpanName = []
            for i in range (NVãos):
                SpanName.append(f'Vão {i+1}')
            NDiafragmasPorVão = int(df_ponte['Ndiaf'].iloc[0])
            DiaphragmDistance = [15, 3/8*CVão, 6/8*CVão, CVão-15] #The distance from the start
of the specified span to the diaphragm. [L]
            LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)


            # Aparelho de apoio
            a_ap = df_ponte['aA'].iloc[0]
            b_ap = df_ponte['aA'].iloc[0]
            G_ap = 0.1 # Rigidez do aparelho de apoio em kN/cm² (1 MPa)
            h_ap = df_ponte['hA'].iloc[0] # Altura total do aparelho de apoio
            hd_ap = h_ap - 2*(2.5/10) - 25/10 # Altura de cálculo da rigidez do aparelho de
apoio, subtraindo-se o cobrimento vertical (2,5 mm) nas duas faces e supondo 25 mm em todas as
chapas de fretagem
            d_ap = df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta
            K_ap = 100
            # K_ap = a_ap * b_ap * G_ap / hd_ap

            # Bents

            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = 300 if Tipo_analise == 'ENCONTROS' else df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                Distance_ext = [Distextp, C_travessaext - Distextp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, C_travessaext/2, C_travessaext - Distextp]
                Height_ext = [Hextp, Hextp, Hextp]

            C_travessaint = df_ponte['Cint'].iloc[0]
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
            SupportType_int = 1
            if N_pilaresint == 2:
                Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # Laje de Transição
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

            # Muro
            AlturaM = df_ponte['AlturaM'].iloc[0]

            # Ala
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

            #********************************      FIM      VARIÁVEIS      DE      INPUT
*****************************************# E fim das definições dos elementos
```

```
            Alturas_pilarescentrais = [400, 600, 800] # Para cada altura de pilar

            for altura in Alturas_pilarescentrais: # Para cada variação de altura
                if N_pilaresint == 2:
                    Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                    Height_int = [altura, altura] #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing the column height. [L]
                else:
                    Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                    Height_int = [altura, altura, altura]

                Vãos = [2000, 2500, 3000, 3500]
                for Vão in Vãos:
                    if CVão > Vão:

                        # Comunicação com a API
                        xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
                        GeneralPath = os.path.join(main_folder, Tipologia)
                        os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à
tipologia. O parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
                        ModelPath       =       os.path.join(GeneralPath,       f'{Tipologia}
({Tipo_analise}).bdb') # Caminho do modelo .bdb

                        #create API helper object

                        helper = cHelper(Helper())

                        if AttachToInstance:

                            #attach to a running instance of CSiBridge

                            try:

                                #get the active CSiBridge object

                                if Remote:

                                    mySAPObject  =  cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                                else:

                                    mySAPObject                                          =
cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                            except:

                                print("No running instance of the program found or failed to
attach.")

                                sys.exit(-1)

                        else:

                            if SpecifyPath:

                                try:

                                    #'create an instance of the CSiBridge object from the
specified path

                                    if Remote:

                                        mySAPObject                                      =
cOAPI(helper.CreateObjectHost(RemoteComputer, ProgramPath))

                                    else:

                                        mySAPObject = cOAPI(helper.CreateObject(ProgramPath))

                                except :
```

```
                                    print("Cannot start a new instance of the program from " +
ProgramPath)

                                    sys.exit(-1)

                                else:


                                    try:

                                        #create an instance of the CSiBridge object from the latest
installed CSiBridge

                                        if Remote:

                                            mySAPObject                                                 =
cOAPI(helper.CreateObjectProgIDHost(RemoteComputer, "CSI.CSiBridge.API.SAPObject"))

                                        else:

                                            mySAPObject                                                 =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))


                                    except:

                                        print("Cannot start a new instance of the program.")

                                        sys.exit(-1)


                                #start CSiBridge application
                                mySAPObject.ApplicationStart()


                            # Redefinição do vão para vãos diferentes

                            SpanLength[1] = np.float64(Vão)
                            SpanLength[3] = np.float64(Vão)
                            CTotal = sum(SpanLength)



                            # 1 Definição dos objetos primitivos
                            SapModel = cSapModel(mySAPObject.SapModel)
                            MyBridgeModeler    =    cBridgeModeler_1(SapModel.BridgeModeler_1)    #
necessário para acessar os métodos do CSiBridge
                            ret    =    SapModel.File.OpenFile(os.path.join(main_folder,    'precast-
template.bdb'))
                            if Tipo_analise != 'MESOESTRUTURA':
                                ret    =    SapModel.File.Save(os.path.join(main_folder,    Tipologia,
f'{Tipologia} ({Tipo_analise}).bdb'))

                            kN_cm_C = 13
                            kN_m_C = 6
                            kN_mm_C = 5
                            ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

                            # 2 Propriedades dos materiais

                            # 2.1 Definições iniciais

                            PropMaterial = cPropMaterial(SapModel.PropMaterial)

                            #Criação de materiais do tipo concreto e tendon
                            MATERIAL_CONCRETE = 2
                            MATERIAL_TENDON = 7
                            ret    =    SapModel.PropMaterial.ChangeName('Concreto    -    Longarina',
f'C{f_ck} - Longarina')
                            ret    =    SapModel.PropMaterial.ChangeName('Concreto    -    Mesoestrutura',
f'C{f_ck} - Mesoestrutura')
```

```
                        ret = SapModel.PropMaterial.ChangeName('Concreto - Tabuleiro',
f'C{f_ck} - Tabuleiro')
                        ret = SapModel.PropMaterial.SetMaterial('CP190RB',
eMatType(MATERIAL_TENDON))

                        # 2.2 Atribuição das propriedades mecânicas elásticas aos materiais
                        E = (5600*f_ck**(1/2))/10
                        E_p = 20000
                        v = 0.2
                        v_p = 0.3
                        alfa_T = 10**(-5)
                        T_ambiente = 25
                        ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Longarina', E,
v, 1*10**-5, T_ambiente)
                        ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Mesoestrutura',
E, v, 1*10**-5, T_ambiente)
                        ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Tabuleiro', E,
v, 1*10**-5, T_ambiente)
                        ret = SapModel.PropMaterial.SetMPIsotropic('CP190RB', E_p, v_p,
1*10**-5, T_ambiente)

                        # 2.3 Atribuição do peso específico aos materiais
                        γ_concreto = 25/(100**3)
                        γ_aco = 78.5/(100**3)
                        ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Longarina',
1, γ_concreto, T_ambiente)
                        ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} -
Mesoestrutura', 1, γ_concreto, T_ambiente)
                        ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Tabuleiro',
1, γ_concreto, T_ambiente)
                        ret = SapModel.PropMaterial.SetWeightAndMass('CP190RB', 1, γ_aco,
T_ambiente)

                        # 2.4 Atribuição das resistência aos materiais e curvas características
de tensão-deformação
                        fptk = 190
                        fpyk = 0.90*fptk
                        ret = SapModel.PropMaterial.SetOTendon_1('CP190RB', fpyk, fptk, 0, 0,
-0.1, T_ambiente)

                        IsLightweight = False
                        fcsfactor = 1 #The shear strength reduction factor for lightweight
concrete
                        SSType = 0
                        # This is 0, 1 or 2, indicating the stress-strain curve type.
                        # 0 = User defined
                        # 1 = Parametric - Simple
                        # 2 = Parametric - Mander
                        SSHysType = 0
                        # This is 0, 1 or 2, indicating the stress-strain hysteresis type.
                        # 0 = Elastic
                        # 1 = Kinematic
                        # 2 = Takeda
                        StrainAtfc = 0.002#This item applies only to parametric stress-strain
curves. It is the strain at the unconfined compressive strength.
                        StrainUltimate = 0.0035 #This item applies only to parametric stress-
strain curves. It is the ultimate unconfined strain capacity. This item must be larger than the
StrainAtfc item.
                        FinalSlope = 1 #This item applies only to parametric stress-strain
curves. It is a multiplier on the material modulus of elasticity, E. This value multiplied times
E gives the final slope on the compression side of the curve.
                        ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Longarina',
f_ck/10, IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)
                        ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Mesoestrutura',
f_ck/10, IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)
                        ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Tabuleiro',
f_ck/10, IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)




                        # 3 Definição das propriedades das seções

                        # 3.1 Frame Sections
```

```
                        # 3.1.1 Longarinas
                        ret    =    SapModel.PropFrame.SetPrecastI_1(f'Longarina    (C{f_ck})',
f'C{f_ck} - Longarina', b_long, d_long, t_long, c1_long)

                        # 3.1.2 Pilares
                        Name = f'Pilar (C{f_ck})'
                        MatProp = f'C{f_ck} - Mesoestrutura'
                        Color = -1
                        Notes = ''
                        GUID = 'Default'
                        ret  =  SapModel.PropFrame.SetCircle(Name,  MatProp,  t3_pilar,  Color,
Notes, GUID) # definição da seção

                        Name  =  f'Pilar  (C{f_ck})'  # The name of an existing frame section
property.
                        MatPropLong = 'A615Gr60' # The name of the rebar material property for
the longitudinal rebar.
                        MatPropConfine = 'A615Gr60' # The name of the rebar material property
for the confinement rebar.
                        Pattern = 2
                        # This is either 1 or 2, indicating the rebar configuration.
                        # 1 = Rectangular
                        # 2 = Circular
                        # For circular frame section properties this item must be 2; otherwise
an error is returned.
                        ConfineType = 1
                        # This is either 1 or 2, indicating the confinement bar type.
                        # 1 = Ties
                        # 2 = Spiral
                        #  This  item  applies  only  when  Pattern  =  2.  If  Pattern  =  1,  the
confinement bar type is assumed to be ties.
                        Cover = 4 # The clear cover for the confinement steel (ties). In the
special case of circular reinforcement in a rectangular column, this is the minimum clear cover.
[L]
                        NumberCBars = 14 # This item applies to a circular rebar configuration,
Pattern = 2. It is the total number of longitudinal reinforcing bars in the column.
                        NumberR3Bars  =  int()  #  This  item  applies  to  a  rectangular  rebar
configuration, Pattern = 1. It is the number of longitudinal bars (including the corner bar) on
each face of the column that is parallel to the local 3-axis of the column.
                        NumberR2Bars  =  int()  #  This  item  applies  to  a  rectangular  rebar
configuration, Pattern = 1. It is the number of longitudinal bars (including the corner bar) on
each face of the column that is parallel to the local 2-axis of the column.
                        RebarSize = '25d' # The rebar name for the longitudinal rebar in the
column.
                        TieSize = '8d' # The rebar name for the confinement rebar in the column.
                        TieSpacingLongit  =  20  #  The  longitudinal  spacing  of  the  confinement
bars (ties). [L]
                        Number2DirTieBars  =  int()  #  This  item  applies  to  a  rectangular
reinforcing configuration, Pattern = 1. It is the number of confinement bars (tie legs) running
in the local 2-axis direction of the column.
                        Number3DirTieBars  =  int()  #  This  item  applies  to  a  rectangular
reinforcing configuration, Pattern = 1. It is the number of confinement bars (tie legs) running
in the local 3-axis direction of the column.
                        ToBeDesigned = False # If this item is True, the column longitudinal
rebar is to be designed; otherwise it is to be checked.
                        ret     =     SapModel.PropFrame.SetRebarColumn(Name,     MatPropLong,
MatPropConfine, Pattern, ConfineType, Cover, NumberCBars, NumberR3Bars, NumberR2Bars, RebarSize,
TieSize,
                        TieSpacingLongit, Number2DirTieBars, Number3DirTieBars, ToBeDesigned)

                        # 3.1.3 Travessas
                        ret = SapModel.PropFrame.SetRectangle(f'Travessa encontro (C{f_ck})',
f'C{f_ck} - Mesoestrutura', h_travessaext, bw_travessaext)

                        ret    =    SapModel.PropFrame.SetRectangle(f'Travessa    intermediária
(C{f_ck})', f'C{f_ck} - Mesoestrutura', h_travessaint, bw_travessaint)

                        # 3.1.4 Transversinas
                        ret    =    SapModel.PropFrame.SetRectangle(f'Transversina    de    apoio
(C{f_ck})', 'A709Gr50', h_transv, bw_transv) # Foi colocado o aço aqui por limitação da API por
não permitir a modelagem da transversina como barra em seções de concreto
                        # Porém, o código atribui ao diafragma o material de aço e, na definição
da transversina, muda o material para o correto
```

```
                        # 3.2 Area Sections

                        Name = 'Encontro'
                        ShellType = 1
                        IncludeDrillingDOF = True
                        MatProp = f'C{f_ck} - Mesoestrutura'
                        MatAng = 0
                        Thickness = hAla
                        Bending = hAla
                        Color = -1
                        Notes = ''
                        GUID = 'Default'
                        ret       =       SapModel.PropArea.SetShell_1(Name,       ShellType,
IncludeDrillingDOF, MatProp, MatAng, Thickness, Bending, Color, Notes, GUID)

                        Name = 'Laje de transição'
                        ShellType = 1
                        IncludeDrillingDOF = True
                        MatProp = f'C{f_ck} - Tabuleiro'
                        MatAng = 0
                        Thickness = EspessuraLT
                        Bending = EspessuraLT
                        Color = -1
                        Notes = ''
                        GUID = 'Default'
                        ret       =       SapModel.PropArea.SetShell_1(Name,       ShellType,
IncludeDrillingDOF, MatProp, MatAng, Thickness, Bending, Color, Notes, GUID)

                        # 3.3 Alimentação dos dados das seções na planilha de dimensionamento

                        # 3.3.1 Longarinas
                        modelo_long = os.path.join(main_folder, 'girder-template.xlsm')
                        wb_long = xw.Book(modelo_long)
                        ws_long = wb_long.sheets["2 Seção transversal"]

                        Name = f'Longarina (C{f_ck})' # The name of an existing precast concrete
I girder frame section property.
                        FileName = '' # If the section property was imported from a property
file, this is the name of that file. If the section property was not imported, this item is
blank.
                        MatProp = '' # The name of the material property for the section.
                        b = []
                        d = []
                        t = []
                        c = float()
                        Color = int() # The display color assigned to the section.
                        Notes = '' # The notes, if any, assigned to the section.
                        GUID = '' # The GUID (global unique identifier), if any, assigned to
the section.
                        ret = SapModel.PropFrame.GetPrecastI_1(Name, FileName, MatProp, b, d,
t, c, Color, Notes, GUID)
                        Dimensoes_Longarina = []
                        Dimensoes_Longarina.extend(list(ret[3]))
                        Dimensoes_Longarina.extend(list(ret[4]))
                        Dimensoes_Longarina.extend(list(ret[5]))
                        Dimensoes_Longarina.append(ret[6])
                        for i in range(len(Dimensoes_Longarina)):
                            ws_long[f'C{12+i}'].value = Dimensoes_Longarina[i]

                        Name = f'Longarina (C{f_ck})' # The name of an existing frame section
property.
                        Area = float() # The cross-sectional area. [L2]
                        As2 = float() # The shear area for forces in the section local 2-axis
direction. [L2]
                        As3 = float() # The shear area for forces in the section local 3-axis
direction. [L2]
                        Torsion = float() # The torsional constant. [L4]
                        I22 = float() # The moment of inertia for bending about the local 2
axis. [L4]
                        I33 = float() # The moment of inertia for bending about the local 3
axis. [L4]
                        S22 = float() # The section modulus for bending about the local 2 axis.
[L3]
                        S33 = float() # The section modulus for bending about the local 3 axis.
[L3]
```

```
                        Z22 = float() # The plastic modulus for bending about the local 2 axis.
[L3]
                        Z33 = float() # The plastic modulus for bending about the local 3 axis.
[L3]
                        R22 = float() # The radius of gyration about the local 2 axis. [L]
                        R33 = float() # The radius of gyration about the local 3 axis. [L]
                        ret = SapModel.PropFrame.GetSectProps(Name, Area, As2, As3, Torsion,
I22, I33, S22, S33, Z22, Z33, R22, R33)

                        ws_long['D5'].value = CVão/100
                        ws_long['D6'].value = Espaçamento_Long/100
                        ws_long['D8'].value = h_laje/100
                        ws_long['D32'].value = ret[1]/(100**2)
                        ws_long['D36'].value = -ret[6]/(100**4)/(ret[8]/(100**3))
                        ws_long['D38'].value = ret[6]/(100**4)
                        ws_long['D42'].value = ret[5]/(100**4)

                        # 3.4 Atribuição das propriedades dependentes do tempo (curvas de
retração e fluência e ganho de resistência/rigidez conforme o CEB FIP 90)

                        # 3.4.1 Longarina

                        A_c = ws_long['D32'].value #Área da seção de concreto
                        U_ar = ws_long['D33'].value #Perímetro da seção em contato com o ar

                        # 3.4.1.1 Alimentação da planilha de retração e fluência
                        modelo_retração_fluência = os.path.join(main_folder, 'shrinkage-creep-
template.xlsx')
                        wb_retração_fluência = xw.Book(modelo_retração_fluência)
                        ws_retração_fluência = wb_retração_fluência.sheets["Retração na
longarina"]
                        ws_retração_fluência['D3'].value = A_c*10**4
                        ws_retração_fluência['D4'].value = U_ar*100


                        # 3.4.3 CP190RB

                        ret = SapModel.PropMaterial.TimeDep.SetTendonCEBFIP90('CP190RB', True,
2, 0, 12, T_ambiente)

                        # 3.4.4 Tabuleiro

                        A_c = ws_long['D56'].value #Área da seção de concreto  #the notional
size is equal to two times the cross-sectional area of the member divided by the perimeter of
the member in contact with the atmosphere.
                        U_ar = ws_long['D57'].value #Perímetro da seção em contato com o ar

                        # Caso o modelo de retração e fluência não for adequado, ver método
PropMaterial.TimeDep.SetConcreteScaleFactors

                        # 3.4.4.1 Alimentação da planilha de retração e fluência
                        ws_retração_fluência = wb_retração_fluência.sheets["Retração no
tabuleiro"]
                        ws_retração_fluência['D3'].value = A_c*10**4
                        ws_retração_fluência['D4'].value = U_ar*100

                        # 3.5 Retração diferencial
                        εdif = ws_retração_fluência['N107'].value
                        ws_long = wb_long.sheets["6 Retração e temperatura"]
                        ws_long['E52'].value = εdif

                        # 3.6 Atribuição das curvas de ganho de resistência

                        # Dados de ganho de resistência do concreto
                        ws_retração_fluência = wb_retração_fluência.sheets["Resistência do
concreto"]
                        ws_retração_fluência['C4'].value = f_ck
                        resistência_fator = list(ws_retração_fluência['L4:L1014'].value)
                        resistência_dias = list(ws_retração_fluência['K4:K1014'].value)

                        # Atribuição da curva de resistência do concreto aos materiais da
longarina, tabuleiro e mesoestrutura

                        TableKey = 'Material Properties 05d - Time Dependence - User Stiffness'
                        GroupName = ''
```

```
                        TableVersion = int()
                        FieldKeysIncluded = []
                        NumberRecords = int()
                        TableData = []
                        ret     =     SapModel.DatabaseTables.GetTableForEditingArray(TableKey,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ColumnLabels = list(ret[2])
                        Resultados = list(ret[4])

                        # Criação do dataframe a partir do dicionário

                        resist_dict = {
                                        'Material': [],
                                        'Age': [],
                                        'Multiplier': []
                                        }

                        Material_Text  =  [f'C{f_ck}  -  Longarina',  f'C{f_ck}  -  Tabuleiro',
f'C{f_ck} - Mesoestrutura']
                        Age = resistência_dias
                        Multiplier = resistência_fator
                        for i in range(3): # Foi convertido em string para alimentar as tabelas
no programa
                            resist_dict['Material'].extend([str(Material_Text[i])]           *
len(Multiplier))
                            resist_dict['Age'].extend(str(element).replace(".",    ",")    for
element in Age)
                            resist_dict['Multiplier'].extend(str(element).replace(".",    ",")
for element in Multiplier)

                        df_concreto = pd.DataFrame(resist_dict)
                        result = df_concreto.values.ravel().tolist()

                        TableKey = 'Material Properties 05d - Time Dependence - User Stiffness'
                        TableVersion = int()
                        FieldKeysIncluded = list(resist_dict.keys())
                        NumberRecords = int()
                        TableData = result
                        ret     =     SapModel.DatabaseTables.SetTableForEditingArray(TableKey,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ret  =  SapModel.DatabaseTables.ApplyEditedTables(False,  int(),  int(),
int(), int(), '')

                        # 3.7 Atribuição das curvas de fluência aos materiais da longarina e
tabuleiro

                        # Dados de fluência na longarina
                        ws_retração_fluência    =    wb_retração_fluência.sheets["Fluência    na
longarina"]
                        deformação_longarina = list(ws_retração_fluência['R4:R109'].value)
                        dias_longarina = list(ws_retração_fluência['J4:J109'].value)

                        # Dados de fluência no tabuleiro
                        ws_retração_fluência    =    wb_retração_fluência.sheets["Fluência    no
tabuleiro"]
                        deformação_tabuleiro = list(ws_retração_fluência['R4:R109'].value)
                        dias_tabuleiro = list(ws_retração_fluência['J4:J109'].value)

                        TableKey = 'Material Properties 05e - Time Dependence - User Creep'
                        GroupName = ''
                        TableVersion = int()
                        FieldKeysIncluded = []
                        NumberRecords = int()
                        TableData = []
                        ret     =     SapModel.DatabaseTables.GetTableForEditingArray(TableKey,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ColumnLabels = list(ret[2])
                        Resultados = list(ret[4])

                        # Criação do dataframe a partir do dicionário

                        fluência_dict = {
                                        'Material': [],
                                        'A': [],
                                        'B': [],
```

```
                                        'Ho': [],
                                        'AgeAtLoad': [],
                                        'Days': [],
                                        'CreepCoeff': []
                                        }

                        Material_Text = [f'C{f_ck} - Longarina', f'C{f_ck} - Tabuleiro']
                        Days = [dias_longarina, dias_tabuleiro]
                        CreepCoeff = [deformação_longarina, deformação_tabuleiro]
                        for i in range(2): # Foi convertido em string para alimentar as tabelas
no programa
                            fluência_dict['Material'].extend([str(Material_Text[i])]         *
len(deformação_longarina))
                            fluência_dict['A'].extend([str(1)] * len(deformação_longarina))
                            fluência_dict['B'].extend([str(1)] * len(deformação_longarina))
                            fluência_dict['Ho'].extend([str(10)] * len(deformação_longarina))
                            fluência_dict['AgeAtLoad'].extend([str(28)]                      *
len(deformação_longarina))
                            fluência_dict['Days'].extend(str(element).replace(".",  ",")  for
element in Days[i])
                            fluência_dict['CreepCoeff'].extend(str(element).replace(".",  ",")
for element in CreepCoeff[i])

                        df_concreto = pd.DataFrame(fluência_dict)
                        result = df_concreto.values.ravel().tolist()

                        TableKey = 'Material Properties 05e - Time Dependence - User Creep'
                        TableVersion = int()
                        FieldKeysIncluded = list(fluência_dict.keys())
                        NumberRecords = int()
                        TableData = result
                        ret    =    SapModel.DatabaseTables.SetTableForEditingArray(TableKey,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ret  =  SapModel.DatabaseTables.ApplyEditedTables(False,  int(),  int(),
int(), int(), '')

                        # 3.7 Atribuição das curvas de retração nos materiais aos materiais da
longarina e tabuleiro

                        # Dados de retração na longarina
                        ws_retração_fluência   =   wb_retração_fluência.sheets["Retração   na
longarina"]
                        deformação_longarina = list(ws_retração_fluência['N4:N105'].value)
                        dias_longarina = list(ws_retração_fluência['K4:K105'].value)

                        # Dados de retração no tabuleiro
                        ws_retração_fluência   =   wb_retração_fluência.sheets["Retração   no
tabuleiro"]
                        deformação_tabuleiro = list(ws_retração_fluência['N4:N105'].value)
                        dias_tabuleiro = list(ws_retração_fluência['K4:K105'].value)

                        # Atribuição da retração aos materiais da longarina e tabuleiro por
tabelas

                        TableKey = 'Material Properties 05f - Time Dependence - User Shrinkage'
                        GroupName = ''
                        TableVersion = int()
                        FieldKeysIncluded = []
                        NumberRecords = int()
                        TableData = []
                        ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ColumnLabels = list(ret[2])
                        Resultados = list(ret[4])

                        # Criação do dataframe a partir do dicionário

                        retração_dict = {
                                        'Material': [],
                                        'A': [],
                                        'B': [],
                                        'Ho': [],
                                        'Age': [],
                                        'ShrinkStrn': []
                                        }
```

```
                        Material_Text = [f'C{f_ck} - Longarina', f'C{f_ck} - Tabuleiro']
                        Age = [dias_longarina, dias_tabuleiro]
                        Strain = [deformação_longarina, deformação_tabuleiro]
                        for i in range(2): # Foi convertido em string para alimentar as tabelas
no programa
                            retração_dict['Material'].extend([str(Material_Text[i])]        *
len(deformação_longarina))
                            retração_dict['A'].extend([str(1)] * len(deformação_longarina))
                            retração_dict['B'].extend([str(1)] * len(deformação_longarina))
                            retração_dict['Ho'].extend([str(10)] * len(deformação_longarina))
                            retração_dict['Age'].extend(str(element).replace(".",   ",")   for
element in Age[i])
                            retração_dict['ShrinkStrn'].extend(str(element).replace(".",  ",")
for element in Strain[i])

                        df_concreto = pd.DataFrame(retração_dict)
                        result = df_concreto.values.ravel().tolist()

                        TableKey = 'Material Properties 05f - Time Dependence - User Shrinkage'
                        TableVersion = int()
                        FieldKeysIncluded = list(retração_dict.keys())
                        NumberRecords = int()
                        TableData = result
                        ret     =     SapModel.DatabaseTables.SetTableForEditingArray(TableKey,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ret  =  SapModel.DatabaseTables.ApplyEditedTables(False,  int(),  int(),
int(), int(), '')




                        # 4 Carregamentos

                        # 4.1 Load Patterns/Cases
                        #  Wearing  surface  e  sidewalk  são  carregamentos  a  serem  definidos
especificamente no módulo do CSiBridge (serão definidos posteriormente)
                        # MyType
                        LTYPE_DEAD = 1
                        LTYPE_LIVE = 3
                        LTYPE_WIND= 6
                        LTYPE_OTHER = 8
                        LTYPE_TEMPERATURE = 10
                        LTYPE_BRAKING = 15
                        LTYPE_HORIZONTALEARTHPRESSURE = 20
                        LTYPE_TEMPERATUREGRADIENT = 26
                        LTYPE_SHRINKAGE = 28
                        LTYPE_CREEP = 29
                        LTYPE_WATERLOADPRESSURE = 30
                        #LTYPE_PEDESTRIANLL = 33
                        LTYPE_PRESTRESS = 34
                        LTYPE_IMPACT = 38
                        #LTYPE_CONSTRUCTION = 39
                        LTYPE_DEADWEARING = 40
                        SelfWTMultiplier  =  0  #The  self  weight  multiplier  for  the  new  load
pattern
                        AddLoadCase = True #Associa o load pattern a uma load case
                        ret              =              SapModel.LoadPatterns.Add('Defensas',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        ret       =       SapModel.LoadPatterns.Add('Pav       +       recap',
eLoadPatternType(LTYPE_DEADWEARING), SelfWTMultiplier, AddLoadCase)
                        #          ret          =          SapModel.LoadPatterns.Add('Passeio',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        #          ret          =          SapModel.LoadPatterns.Add('Haunch',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        ret = SapModel.LoadPatterns.Add('Vento', eLoadPatternType(LTYPE_WIND),
SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    uniforme    pos',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    uniforme    neg',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
                        ret     =     SapModel.LoadPatterns.Add('Frenagem     e     aceleracao',
eLoadPatternType(LTYPE_BRAKING), SelfWTMultiplier, AddLoadCase)
```

```
                        ret         =         SapModel.LoadPatterns.Add('Empuxo         terra',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Empuxo    terra    carga    móvel',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    gradiente    pos',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    gradiente    neg',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
                        ret                 =                 SapModel.LoadPatterns.Add('Água',
eLoadPatternType(LTYPE_WATERLOADPRESSURE), SelfWTMultiplier, AddLoadCase)
                        ret              =              SapModel.LoadPatterns.Add('Protensão',
eLoadPatternType(LTYPE_PRESTRESS), SelfWTMultiplier, AddLoadCase)
                        ret               =               SapModel.LoadPatterns.Add('Impacto',
eLoadPatternType(LTYPE_IMPACT), SelfWTMultiplier, AddLoadCase)

                        # 4.2 Load Distributions

                        # 4.2.1 Gradiente de temperatura
                        ret                                                                    =
MyBridgeModeler.LoadDistribution.TemperatureGradient.AddNew('Temperatura gradiente')

                        Name  =  'Temperatura  gradiente'  #The  name  of  an  existing  bridge
temperature gradient load distribution.
                        NumberDistances = 4 #The number of specified distances (between 1 and
5, inclusive).
                        DistanceThatMayVary = 3 #The specified distance that may vary.
                        TempDifferencesToConsider = 0
                        # The temperature differences to consider.
                        # 0 = Positive and Negative
                        # 1 = Positive Only
                        # 2 = Negative Only
                        d_1 = min(0.3*(h_laje + h_long), 15)
                        d_2 = min(max(0.3*(h_laje + h_long), 10), 25)
                        d_3 = min(min(0.3*(h_laje + h_long), 10 + 10), (h_laje + h_long)-d_1-
d_2)
                        if (h_laje + h_long) < 20:
                            ΔT_1 = 8.5
                            ΔT_2 = 3.5
                            ΔT_3 = 0.5
                        elif 20 < (h_laje + h_long) <= 40:
                            ΔT_1 = np.interp((h_laje + h_long), [20, 40], [8.5, 12])
                            ΔT_2 = np.interp((h_laje + h_long), [20, 40], [3.5, 3])
                            ΔT_3 = np.interp((h_laje + h_long), [20, 40], [0.5, 1.5])
                        elif 40 < (h_laje + h_long) <= 60:
                            ΔT_1 = np.interp((h_laje + h_long), [40, 60], [12, 13])
                            ΔT_2 = np.interp((h_laje + h_long), [40, 60], [3, 3])
                            ΔT_3 = np.interp((h_laje + h_long), [40, 60], [1.5, 2])
                        elif 60 < (h_laje + h_long) < 80:
                            ΔT_1 = np.interp((h_laje + h_long), [60, 80], [13, 13.5])
                            ΔT_2 = np.interp((h_laje + h_long), [60, 80], [3, 3])
                            ΔT_3 = np.interp((h_laje + h_long), [60, 80], [2, 2.5])
                        else:
                            ΔT_1 = 13.5
                            ΔT_2 = 3.0
                            ΔT_3 = 2.5
                        d  =  [d_1,  d_2,  float(),  d_3]  #A  zero-based  one-dimensional  array
dimensioned to (NumberDistances - 1) containing the specified distances. [L]
                        TPos  =  [ΔT_1,  ΔT_2,  0,  0,  ΔT_3]  #A  zero-based  one-dimensional  array
dimensioned to (NumberDistances) containing the specified positive temperatures. [T]
                        TNeg = [-5, -5, 0, 0, 0] #A zero-based one-dimensional array dimensioned
to (NumberDistances) containing the specified negative temperatures. [T]
                        ret                                                                    =
MyBridgeModeler.LoadDistribution.TemperatureGradient.SetUserGeneralData(Name,  NumberDistances,
DistanceThatMayVary, TempDifferencesToConsider, d, TPos, TNeg)

                        # 4.2.2 Vento
                        modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
                        wb_pil_ext = xw.Book(modelo_pil_ext)
                        ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

                        Name = 'Vento'
                        ret = MyBridgeModeler.LoadDistribution.Line.AddNew(Name)

                        Name = Name # The name of an existing bridge line load distribution
                        LoadType = 1 # The line load type
```

```
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 5
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LoadValue = (ws_pil_ext['D31'].value/(CVão/100))/100 # The load value.
[F/L] for force; [F] for moment
                        ReferenceLocation = 2
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LoadDistance = 0 # The distance from the reference location to the
load. [L]
                        ret = MyBridgeModeler.LoadDistribution.Line.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LoadValue, ReferenceLocation, LoadDistance)

                        # 4.2.3 Frenagem e aceleração
                        Name = 'Frenagem e aceleracao'
                        ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

                        Name = Name # The name of an existing bridge area load distribution.
                        LoadType = 1 # The area load type
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 4
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LeftEdgeValue = 0.32E-04 # The left edge load value. [F/L2] for force;
[F/L] for moment
                        RightEdgeValue = 0.32E-04 # The right edge load value. [F/L2] for force;
[F/L] for moment
                        LeftReferenceLocation = 1 #The left reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LeftLoadDistance = 0 # The distance from the left edge of load to the
left reference location. [L]
                        RightReferenceLocation = 2 # The right reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        RightLoadDistance = 0 # The distance from the right edge of load to the
right reference location. [L]
                        ret = MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
                        LeftReferenceLocation,    LeftLoadDistance,    RightReferenceLocation,
RightLoadDistance)




                        # 5 Criação do modelo da ponte - (cBridgeModeler_1.NewBridgeModel)

                        StartStation = 0 #The starting station for the bridge. [L]
                        InitialBearing = "N900000E" #The initial bearing for the bridge. This
should be a valid bearing angle such as "S624355E".
                        Grade = 0 #The grade for the bridge specified in percent. A positive
grade represents an incline with increasing station.
                        Skew = "Default"
```

```
                        #The skew angle for the bridge. This should be a valid bearing angle
such as “S624355E”, “Default” or “DefaultX”
                        #where X is a positive or negative number indicating the number of
degrees the bearing is different from the default skew.
                        #The default skew is perpendicular to the reference line.
                        Radius = 0 #The bridge radius. A positive radius curves to the right
and a negative radius curves to the left. A radius of zero means an infinite radius (i.e.,
straight). [L]
                        SuperElevation = 0 #The bridge superelevation specified in percent. A
positive value rotates clockwise corresponding to a curve right and a negative value rotates
counterclockwise corresponding to a curve left.
                        SectionType = 11 # Diferença na numeração do pc do lab e o daqui de
casa?
                        #This is an integer between 2 and 16, inclusive, indicating the bridge
section type.
                        # 2 = Concrete Box Girder - Exterior Girders Vertical
                        # 3 = Concrete Box Girder - Exterior Girders Sloped
                        # 4 = Concrete Box Girder - Exterior Girders Clipped
                        # 5 = Concrete Box Girder - Exterior Girders with Radius
                        # 6 = Concrete Box Girder - Exterior Girders Sloped Maximum
                        # 7 = Concrete Tee Beam
                        # 8 = Concrete Flat Slab
                        # 9 = Steel I Girder
                        # 10 = Concrete Box Girder - AASHTO-PCI-ASBI Standard
                        # 11 = Precast Concrete I Girder
                        # 12 = Precast Concrete U Girder
                        # 13 = Concrete Box Girder - Advanced
                        # 14 = Steel U Girder
                        # 15 = Concrete Solid Girder
                        # 16 = Precast Concreter Super-T Girder
                        ret  =  MyBridgeModeler.NewBridgeModel(StartStation,  InitialBearing,
Grade, Skew, Radius, SuperElevation, SpanLength, SectionType)

                        # 5.1 Objetos de interesse
                        NumberNames = int()
                        MyName = []
                        ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
                        myBridgeObjName = list(ret[2])[0]

                        NumberNames = int()
                        MyName = []
                        BridgeSectionType = [SectionType]
                        ret  =  MyBridgeModeler.DeckSection.GetNameList(NumberNames,  MyName,
BridgeSectionType)
                        myDeckSectionName = list(ret[2])[0]

                        Name = myBridgeObjName
                        LayoutLine = ''
                        ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
                        myLayoutLineName = ret[1]

                        ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
                        myLaneName = list(ret[2])[0]

                        # 5.2 Definição do Layout Line - (cBMLayoutLine.SetGeneralData). O
layout aqui definido será o principal (superestrutura). A passagem do veículo será definida pela
criação de outros layoutlines

                        Name = myLayoutLineName
                        CoordinateSystem = 'GLOBAL'
                        StartX = 0
                        StartY = 0
                        StartZ = 0
                        ret         =         MyBridgeModeler.LayoutLine.SetGeneralData(Name,
CoordinateSystem, StartX, StartY, StartZ)

                        # 5.3 Bridge Object Data (cBMBridgeObject.SetSpanLayoutByLength Method)
                        Name = myBridgeObjName
                        StartStation = 0
                        ret     =     MyBridgeModeler.BridgeObject.SetSpanLayoutByLength(Name,
StartStation, SpanName, SpanLength)

                        # 5.4 Veiculos (cBMVehicle Methods)
                        Region = 'User'
                        Standard = 'User'
```

```
                        Vehicle = 'All'
                        ret = MyBridgeModeler.Vehicle.ImportVehicle(Region, Standard, Vehicle)
                        ret = MyBridgeModeler.Vehicle.ConvertToUser('Multidao')

                        # Correção (carregamento de Multidao que depende da largura do
tabuleiro)

                        Name = 'Multidao' # The name of the vehicle.
                        UniformLoadSF = 1 # The uniform load scale factor.
                        AxleLoadSF = 1 # The axle load scale factor.
                        NumberLoadDataSets = 2 # The number of load data sets defined for the
vehicle.
                        LoadLengthType = [1, 2]
                        # A zero-based one-dimensional array containing the load length type
for each load data set.
                        # 1 = Leading Load
                        # 2 = Trailing Load
                        # 3 = Fixed Length
                        # 4 = Variable Length
                        MinimumDistance = [0, 0] # A zero-based one-dimensional array
containing the minimum distance between the current and previous axle. This item is only
applicable if the LoadLengthType is 3 or 4 (Fixed Length or Variable Length). [L]
                        MaximumDistance = [float(), float()] # A zero-based one-dimensional
array containing the maximum distance between the current and previous axle. This item is only
applicable if the LoadLengthType is 4 (Variable Length). This item must either be zero (meaning
infinity) or it must be greater than or equal to MinimumDistance item. [L]
                        UniformLoad = [(LarguraRolamento)/100*(5/100),
(LarguraRolamento)/100*(5/100)] # A zero-based one-dimensional array containing the uniform
load. [F/L]
                        UniformWidthType = [2, 2]
                        # A zero-based one-dimensional array containing the width type for the
uniform load.
                        # 1 = Zero Width
                        # 2 = Lane Width
                        # 3 = Fixed Width
                        UniformWidth = [float(), float()] # A zero-based one-dimensional array
containing the uniform load width. This item is only applicable if the UnifWidthType is 3 (Fixed
Width). [L]
                        AxleLoad = [0, float()] # A zero-based one-dimensional array containing
the total load for the axle. This item does not apply when the LoadLengthType item is 4 (Variable
Length). [F]
                        AxleWidthType = [1, int()]
                        # A zero-based one-dimensional array containing the axle width type
indicating how the axle load is to be distributed. This item does not apply when the
LoadLengthType item is 4 (Variable Length).
                        # 1 = One Point
                        # 2 = Two Points
                        # 3 = Four Points
                        # 4 = Lane Width Line
                        # 5 = Fixed Width Line
                        AxleWidth = [float(), float()] # A zero-based one-dimensional array
containing the axle width. This item is only applicable if the AxleWidthType item is 2, 3 or 5
(Two Points, Four Points or Fixed Width Line).
                        # This item does not apply when the LoadLengthType item is 4 (Variable
Length). [L]
                        ret = MyBridgeModeler.Vehicle.SetVerticalLoads(Name, UniformLoadSF,
AxleLoadSF, NumberLoadDataSets, LoadLengthType, MinimumDistance, MaximumDistance, UniformLoad,
UniformWidthType, UniformWidth, AxleLoad, AxleWidthType, AxleWidth)

                        ret = SapModel.LoadCases.Delete('MOVE1')
                        ret = SapModel.LoadCases.Moving.SetCase('Veiculo')
                        ret = SapModel.LoadCases.Moving.SetCase('Multidao')

                        ret = SapModel.LoadCases.Moving.SetInitialCase('Veiculo', '')
                        ret = SapModel.LoadCases.Moving.SetInitialCase('Multidao', '')

                        NumberLoads = 1 #The number of loads assigned to the specified analysis
case.
                        MyClass = ['Veiculo'] #This is an array that includes the vehicle class
for each load assigned to the load case.
                        SF = [1] #This is an array that includes the scale factor for each load
assigned to the load case.
                        Min = [0] #This is an array that includes the minimum number of lanes
loaded for each load assigned to the load case.
```

```
                        Max = [0] #This is an array that includes the maximum number of lanes
loaded for each load assigned to the load case. This item must be 0, or it must be greater than
or equal to Min. If this item is 0, all available lanes are loaded.
                        ret   =   SapModel.LoadCases.Moving.SetLoads('Veiculo',   NumberLoads,
MyClass, SF, Min, Max)
                        MyClass = ['Multidao']
                        ret   =   SapModel.LoadCases.Moving.SetLoads('Multidao',   NumberLoads,
MyClass, SF, Min, Max)

                        LoadNumber = 1 #The load assignment number.
                        NumberItems = 1 #The number of lanes loaded for the specified load
assignment number.
                        MyName = [f'{myLaneName}'] #This is an array that includes the name of
each lane loaded for the specified load assignment number.
                        ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber,
NumberItems, MyName)
                        ret = SapModel.LoadCases.Moving.SetLanesLoaded('Multidao', LoadNumber,
NumberItems, MyName)




                        # 6 Aparelhos de apoio a partir de um link definido pelo usuário
(cBMBearing.SetBearingUser Method)

                        # 6.1 Aparelho de apoio

                        Name = 'Elastômero' #The name of an existing or new link property. If
this is an existing property, that property is modified; otherwise, a new property is added.
                        DOF = [True, True, True, False, False, False]
                        # This is a boolean array, dimensioned to 5, indicating if properties
exist for a specified degree of freedom.
                        # DOF(0) = U1
                        # DOF(1) = U2
                        # DOF(2) = U3
                        # DOF(3) = R1
                        # DOF(4) = R2
                        # DOF(5) = R3
                        Fixed = [True, False, False, False, False, False]
                        # This is a boolean array, dimensioned to 5, indicating if the specified
degree of freedom is fixed (restrained).
                        # Fixed(0) = U1 fixity if DOF(0) = True
                        # Fixed(1) = U2 fixity if DOF(1) = True
                        # Fixed(2) = U3 fixity if DOF(2) = True
                        # Fixed(3) = R1 fixity if DOF(3) = True
                        # Fixed(4) = R2 fixity if DOF(4) = True
                        # Fixed(5) = R3 fixity if DOF(5) = True
                        Ke = [float(), K_ap, K_ap, float(), float(), float()]
                        # This is an array of stiffness terms for the link property. There are
6 terms in the array if the stiffness is uncoupled and 21 if it is coupled. The KeCoupled item
indicates if the stiffness is coupled.
                        # If the stiffness is uncoupled:
                        # Ke(0) = U1 [F/L]
                        # Ke(1) = U2 [F/L]
                        # Ke(2) = U3 [F/L]
                        # Ke(3) = R1 [FL]
                        # Ke(4) = R2 [FL]
                        # Ke(5) = R3 [FL]
                        # If the stiffness is coupled:
                        # Ke(0) = U1U1 [F/L]
                        # Ke(1) = U1U2 [F/L]
                        # Ke(2) = U2U2 [F/L]
                        # Ke(3) = U1U3 [F/L]
                        # Ke(4) = U2U3 [F/L]
                        # Ke(5) = U3U3 [F/L]
                        # Ke(6) = U1R1 [F]
                        # Ke(7) = U2R1 [F]
                        # Ke(8) = U3R1 [F]
                        # Ke(9) = R1R1 [FL]
                        # Ke(10) = U1R2 [F]
                        # Ke(11) = U2R2 [F]
                        # Ke(12) = U3R2 [F]
                        # Ke(13) = R1R2 [FL]
                        # Ke(14) = R2R2 [FL]
```

```
                        # Ke(15) = U1R3 [F]
                        # Ke(16) = U2R3 [F]
                        # Ke(17) = U3R3 [F]
                        # Ke(18) = R1R3 [FL]
                        # Ke(19) = R2R3 [FL]
                        # Ke(20) = R3R3 [FL]
                        Ce = [float(), 0, 0, float(), float(), float()]
                        # This is an array of damping terms for the link property. There are 6
terms in the array if the damping is uncoupled and 21 if it is coupled. The CeCoupled item
indicates if the damping is coupled.
                        # If the damping is uncoupled:
                        # Ce(0) = U1 [F/L]
                        # Ce(1) = U2 [F/L]
                        # Ce(2) = U3 [F/L]
                        # Ce(3) = R1 [FL]
                        # Ce(4) = R2 [FL]
                        # Ce(5) = R3 [FL]
                        # If the damping is coupled:
                        # Ce(0) = U1U1 [F/L]
                        # Ce(1) = U1U2 [F/L]
                        # Ce(2) = U2U2 [F/L]
                        # Ce(3) = U1U3 [F/L]
                        # Ce(4) = U2U3 [F/L]
                        # Ce(5) = U3U3 [F/L]
                        # Ce(6) = U1R1 [F]
                        # Ce(7) = U2R1 [F]
                        # Ce(8) = U3R1 [F]
                        # Ce(9) = R1R1 [FL]
                        # Ce(10) = U1R2 [F]
                        # Ce(11) = U2R2 [F]
                        # Ce(12) = U3R2 [F]
                        # Ce(13) = R1R2 [FL]
                        # Ce(14) = R2R2 [FL]
                        # Ce(15) = U1R3 [F]
                        # Ce(16) = U2R3 [F]
                        # Ce(17) = U3R3 [F]
                        # Ce(18) = R1R3 [FL]
                        # Ce(19) = R2R3 [FL]
                        dj2 = 0 #The distance from the J-End of the link to the U2 shear spring.
This item applies only when DOF(1) = True. [L]
                        dj3 = 0 #The distance from the J-End of the link to the U3 shear spring.
This item applies only when DOF(2) = True. [L]
                        KeCoupled = False #This item is True if the link stiffness, Ke, is
coupled. There are 21 terms in the Ke array if Ke is coupled; otherwise there are 6 terms.
                        CeCoupled = False #This item is True if the link damping, Ce, is
coupled. There are 21 terms in the Ce array if Ce is coupled; otherwise there are 6 terms.
                        Notes = '' #The notes, if any, assigned to the property.
                        GUID = '' #The GUID (global unique identifier), if any, assigned to the
property. If this item is input as Default, the program assigns a GUID to the property.
                        ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3,
KeCoupled, CeCoupled, Notes, GUID) #Criação do link Neoprene

                        ret = MyBridgeModeler.Bearing.AddNew('Ap. de apoio')
                        ret    =    MyBridgeModeler.Bearing.SetBearingLink('Ap.    de    apoio',
'Elastômero')




                        #  7.  Definição  dos  apoios  extremos,  intermediários  (travessas),
abutments e atribuições

                        ret = MyBridgeModeler.Bent.AddNew('Apoios extremos')
                        ret = MyBridgeModeler.Bent.AddNew('Apoios intermediários')
                        ret = MyBridgeModeler.Abutment.AddNew('Apoio aparelho de apoio')

                        # 7.1 Definição - Travessas extremas

                        Name = 'Apoios extremos' #The name of an existing bridge bent.
                        CapBeamSection = f'Travessa encontro (C{f_ck})' #This is either "None"
or the name of the frame section property assigned to the cap beam.
                        Foundation = 'Fixed' #Containing either Fixed, Pinned, or a defined
foundation spring.
                        BentType = 0
```

```
                        # 0 = Single bearing line (continuous superstructure)
                        # 1 = Double bearing line (discontinuous superstructure)
                        # 2 = Double bearing line with closure pour
                        ret    =    MyBridgeModeler.Bent.SetGeneralData(Name,    CapBeamSection,
C_travessaext, SupportType_ext, N_pilaresext, Foundation, BentType)

                        Section  =  [f'Pilar  (C{f_ck})']  *  N_pilaresext  #A  zero-based  one-
dimensional array dimensioned to (NumberColumns - 1) containing the frame section properties
assigned to each column.
                        Angle  =  [0]  *  N_pilaresext  #A  zero-based  one-dimensional  array
dimensioned to (NumberColumns - 1) containing the orientation of the column in degrees
counterclockwise between a line parallel to the bent and the column local 2 axis. [deg]
                        Foundation = ['Fixed'] * N_pilaresext #A zero-based one-dimensional
array dimensioned to (NumberColumns - 1) containing either Fixed, Pinned, a defined foundation
spring, or Previous. Note that an error will occur if Previous is assigned to the first column.
                        ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_ext,
Height_ext, Angle, Foundation)

                        SuperConnection = 1
                        # The superstructure connection.
                        # 0 = Connect to girder top and bottom
                        # 1 = Connect to girder bottom only
                        ret      =      MyBridgeModeler.Bent.SetSuperConnectionSingleLine(Name,
SuperConnection)

                        # 7.2 Definição - Travessas intermediárias
                        Name = 'Apoios intermediários'
                        CapBeamSection = f'Travessa intermediária (C{f_ck})'
                        Foundation = 'Fixed'
                        BentType = 1
                        ret    =    MyBridgeModeler.Bent.SetGeneralData(Name,    CapBeamSection,
C_travessaint, SupportType_int, N_pilaresint, Foundation, BentType)

                        Section = [f'Pilar (C{f_ck})'] * N_pilaresint
                        Angle = [0] * N_pilaresint
                        Foundation = ['Fixed'] * N_pilaresint
                        ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_int,
Height_int, Angle, Foundation)

                        BeforeSuperConnection = 1
                        # The superstructure connection before the bent.
                        # 0 = Connect to girder top and bottom
                        # 1 = Connect to girder bottom only
                        BeforeDistanceToBearingLine = d_ap #The distance to the bearing line
before the bent. [L]
                        BeforeDistanceToEndOfGirder = 0 #The distance to the end of the girder
before the bent. [L]
                        AfterSuperConnection = 1
                        # The superstructure connection after the bent.
                        # 0 = Connect to girder top and bottom
                        # 1 = Connect to girder bottom only
                        AfterDistanceToBearingLine = d_ap #The distance to the bearing line
after the bent. [L]
                        AfterDistanceToEndOfGirder = 0 #AfterDistanceToEndOfGirder
                        ret      =      MyBridgeModeler.Bent.SetSuperConnectionDoubleLine(Name,
BeforeSuperConnection,         BeforeDistanceToBearingLine,         BeforeDistanceToEndOfGirder,
AfterSuperConnection, AfterDistanceToBearingLine, AfterDistanceToEndOfGirder)

                        # 7.5 Definição - Apoio aparelho de apoio
                        Name = 'Apoio aparelho de apoio' # The name of an existing bridge
abutment.
                        SuperConnection = 1
                        # The superstructure connection.
                        # 0 = Integral
                        # 1 = Connect to Girder Bottom Only
                        DistanceFromBridgeEnd = float() # The distance from the bridge end to
the bearing line. [L]
                        FoundationProperty = 'Fixed' # This is either Fixed, Pinned or the name
of a defined foundation spring.
                        SubstructureType = 1
                        # The substructure type.
                        # 0 = None
                        # 1 = Foundation spring
                        # 2 = Continuous beam (continuously supported)
```

```
                        BeamProperty = '' # The frame section property assigned to the
continuous beam. This item only applies when the SubstructureType item is 1, Continuous beam
(continuously supported).
                        BeamLength = float() # The length of the continuous beam. This item
only applies when the SubstructureType item is 1, Continuous beam (continuously supported). [L]
                        ret   =   MyBridgeModeler.Abutment.SetAbutment(Name,   SuperConnection,
DistanceFromBridgeEnd, FoundationProperty, SubstructureType, BeamProperty, BeamLength)

                        # 7.6 Atribuições

                        Name = myBridgeObjName #The name of a bridge object.

                        if NVãos != 1:
                            for i in range(NVãos - 1):
                                SpanNameBeforeBent = f'Vão {i+1}' #The name of the span before
the considered bent.
                                BentProperty = 'Apoios intermediários' #The bent property
assigned to the specified bent.
                                BentDirection = 'Default' #The bearing of a line parallel to
the bent. It is either Default or a valid bearing string.
                                ret                                                         =
MyBridgeModeler.BridgeObject.BentAssign.SetBentAssignment(Name,             SpanNameBeforeBent,
BentProperty, BentDirection)

                                BentSupportName = f'Apoio {i+1}'
                                ret                                                         =
MyBridgeModeler.BridgeObject.BentAssign.SetSupportName(Name,                SpanNameBeforeBent,
BentSupportName)

                                SuperstructureIsContinousAtBent = False #Indicates whether the
superstructure is continuous (True) or discontinuous (False) at the bent.
                                DiaphragmBeforeBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the diaphragm property at the bent.
                                # If the specified bent property is None or it has a single
line of bearings (continuous superstructure) this is the property of the diaphragm that occurs
at the bent.
                                # If the specified bent property has a double line of bearings
(discontinuous superstructure) this is the property of the diaphragm that occurs just before
the bent.
                                # This item does not apply if the specified bent property is a
double bearing line with closure pour.
                                InternalDiaphragmBeforeBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the internal diaphragm property at the bent.
                                # This only applies to bents at steel U girder or super T girder
deck sections.
                                # If the specified bent property is None or it has a single
line of bearings (continuous superstructure) this is the property of the internal diaphragm that
occurs at the bent.
                                # If the specified bent property has a double line of bearings
this is the property of the internal diaphragm that occurs just before the bent.
                                DiaphragmAfterBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the diaphragm property after the bent.
                                # It applies when the specified bent property has a double line
of bearings (discontinuous superstructure).
                                # This item does not apply if the specified bent property is a
double bearing line with closure pour.
                                InternalDiaphragmAfterBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the internal diaphragm property after the bent.
                                # This only applies to bents at steel U girder or super T girder
deck sections.
                                # It applies when the specified bent property has a double line
of bearings and is not a closure pour.
                                ClosurePourDiaphragm = 'None' #This is the name of a defined
closure pour bridge diaphragm property. This item only applies when the specified bent type is
Double Bearing Line with Closure Pour.
                                Restrainer = 'None' #This is either None or the name of a
defined bridge diaphragm property indicating the restrainers used at the bent. It only applies
when the superstructure is discontinuous at the bent.
                                RestrainerElevation = 0 #The restrainer elevation (at layout
line, global Z). this item only applies when the superstructure is discontinuous at the bent.
[L]
```

```
                                MeshSuperstructureToMatchBent = False
                                # This is a flag indicating if the superstructure is to be
meshed to match the bent.
                                # This item only applies when the superstructure is continuous
at the bent.
                                MeshSteelUGirderToMatchBent = False
                                # This is a flag indicating if the steel U girders are to be
meshed to match the bent.
                                # This item only applies to bents at steel U girder or super T
girder deck sections when the superstructure is continuous at the bent.
                                ret                                                          =
MyBridgeModeler.BridgeObject.BentAssign.SetSuperstructureAssignment(Name,   SpanNameBeforeBent,
SuperstructureIsContinousAtBent,       DiaphragmBeforeBent,       InternalDiaphragmBeforeBent,
DiaphragmAfterBent, InternalDiaphragmAfterBent,
                                ClosurePourDiaphragm,     Restrainer,     RestrainerElevation,
MeshSuperstructureToMatchBent, MeshSteelUGirderToMatchBent)

                                Elevation = -h_laje -h_long - h_ap #The global Z elevation of
the bent. [L]
                                HorizontalOffset = 0 #The horizontal offset from the layout
line to the midlength of the cap beam. [L]
                                ret                                                          =
MyBridgeModeler.BridgeObject.BentAssign.SetBentLocation(Name,  SpanNameBeforeBent,  Elevation,
HorizontalOffset)

                                for j in range(2):
                                    BearingLocation = [1, 2]
                                    # This is one of the following indicating the bent bearing
location:
                                    # 1 = Single line of bearings or bearings before bent for
double line of bearings.
                                    # 2 = Bearings after bent for double line of bearings.
                                    BearingAssignmentType = 1
                                    # This is one of the following indicating the bearing
assignment type:
                                    # 1 = Girder-by-girder
                                    # 2 = General
                                    # The General assignment type only applies to abutments at
concrete box girders deck sections or concrete flat slab deck sections. General assignments to
abutments at other deck section types will return an error.
                                    BearingProperty = 'Ap. de apoio' #The name of a defined
bearing property assigned to the bent.
                                    RestrainerProperty = 'None' #This is either None or the
name of a defined restrainer property assigned to the bent.
                                    ElevationAtLayoutLine = -h_laje -h_long - h_ap/2 #The
bearing elevation (global Z) at the layout line. [L]
                                    RotationAngle = 0 #The bearing rotation angle from the
bridge default.
                                    NumberOfBearings = int()
                                    #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
                                    # For bearings at steel U girder sections this item must
be 1 or 2.
                                    # This item only applies when the bearing assignment type
is General or when the abutment is at a Steel U Girder deck section.
                                    UniformSpacing = int() #The uniform spacing of bearings.
This item only applies when the bearing assignment type is General. [L]
                                    BearingOffset = int() #The offset from the section
reference point to the bearing center. This item only applies when the bearing assignment type
is General. [L]
                                    ret                                                      =
MyBridgeModeler.BridgeObject.BentAssign.SetBearingAssignment(Name,          SpanNameBeforeBent,
BearingLocation[j],      BearingAssignmentType,      BearingProperty,      RestrainerProperty,
ElevationAtLayoutLine, RotationAngle, NumberOfBearings,
                                    UniformSpacing, BearingOffset)



                        # 8 Definição e atribuição dos diafragmas

                        Name = myBridgeObjName
                        SpanNames = '' #The name of the span from which the diaphragms are
deleted. If this is input as a blank string then all diaphragms in all spans will be deleted.
```

```
                        ret   =   MyBridgeModeler.BridgeObject.DiaphragmAssign.DeleteAll(Name,
SpanNames)

                        NumberNames = int() #The number of diaphragm names retrieved by the
program.
                        MyName = [] #This is a one-dimensional array of diaphragm names. The
MyName array is created as a dynamic, zero-based array by the API user: Dim MyName() as String.
                        #The array is dimensioned to (NumberNames - 1) inside the program,
filled with the names, and returned to the API user.
                        MyType = []
                        # This is a one-dimensional array, dimensioned to (NumberNames - 1),
of diaphragm types.
                        # 1 = Solid
                        # 2 = Chord and Brace
                        # 3 = Single Beam
                        # 4 = Steel Plate
                        # 5 = Closure Pour
                        ret   =   MyBridgeModeler.Diaphragm.GetNameList(NumberNames,   MyName,
MyType)
                        NDiaphragms = ret[1]
                        DiaphragmNames = list(ret[2])
                        ValidDiapgragmNames = ['Transversina']
                        for i in range (NDiaphragms):
                            if DiaphragmNames[i] not in ValidDiapgragmNames:
                                ret = MyBridgeModeler.Diaphragm.Delete(DiaphragmNames[i])

                        Name = 'Transversina'
                        MyType = 3
                        # The diaphragm type.
                        # 1 = Solid
                        # 2 = Chord and Brace
                        # 3 = Single Beam
                        # 4 = Steel Plate
                        # 5 = Closure Pour
                        ret = MyBridgeModeler.Diaphragm.AddNew(Name, MyType)

                        Name = 'Transversina' #The name of an existing single beam-type bridge
diaphragm.
                        BeamSection = f'Transversina de apoio (C{f_ck})' #The steel frame
section property used for the beam.
                        ElevationChangeTop = d_long[1] #The elevation change from top of beam
to top of adjacent girder. [L]
                        ModelMixed = bool(False) #Indicates if the beam is modeled as mixed
frame and shell
                        ConnectionPlateWidth = 15 #The width of the connection plate. [L]
                        ConnectionPlateThickness = 0.8 #The thickness of the connection plate.
[L]
                        ConnectionPlateMaterial = 'A709Gr50' #The steel material for the
connection plate.
                        ConnectionPlateOnBothSidesOfWeb = bool(True) #Indicates if the
connection plate is provided on both sides of the web.
                        ret   =   MyBridgeModeler.Diaphragm.SetSingleBeam(Name,   BeamSection,
ElevationChangeTop,      ModelMixed,      ConnectionPlateWidth,      ConnectionPlateThickness,
ConnectionPlateMaterial, ConnectionPlateOnBothSidesOfWeb)

                        ret   =   SapModel.PropFrame.SetRectangle(f'Transversina   de   apoio
(C{f_ck})', f'C{f_ck} - Tabuleiro', h_transv, bw_transv) # Redefinição do material (limitação
da API)

                        DiaphragmDistance = []
                        for span in SpanLength:
                            DiaphragmDistance.append([15, 3/8*span, 6/8*span, span-15])

                        for j in range(NVãos):
                            for i in range(NDiafragmasPorVão):
                                Name = myBridgeObjName #The name of a bridge object.
                                SpanNames = f'Vão {j+1}' #The name of the span in which the
diaphragm exists.
                                DiaphragmProperty = 'Transversina' #The name of a defined
diaphram property assigned to the diaphragm.
                                Location = int(0)
                                # This is one of the following indicating the diaphragm location
across the bridge section.
                                # 0 = All Spaces
                                # 1 = Girder n to Girder n + 1
```

```
                                # 2 = All Girder Interiors
                                # 3 = Girder n Interior
                                GirderNumber = int() #The girder number identified as n when
the Location item is 1 or 3 (Girder n to Girder n + 1 or Girder n Interior). This item does not
apply when the Location item is 0 or 2.
                                Bearing = 'Default' #This is either Default or a valid bearing
string
                                ReferenceLine = 0 #This is either 0 indicating the layout line
or nonzero indicating a girder number. This item is only under user control when the Location
item is 1 (Girder n to Girder n + 1).
                                ErrorMessage = '' #If the function returns an error related to
the diaphragm distance this string may provide information about the error.
                                ret                                                    =
MyBridgeModeler.BridgeObject.DiaphragmAssign.AddNew(Name, SpanNames, DiaphragmDistance[j][i],
DiaphragmProperty, Location, GirderNumber, Bearing, ReferenceLine, ErrorMessage)




                        #       9       Bridge       Section       Data       -       Definition
(cBMDeckSectionPrecastConcreteIGirder Interface)
                        LeftFillet = 0 #The horizontal dimension of the fillet on the left side
of the deck (identified as f1 on the Define Bridge Section Data form). [L]
                        RightFillet = 0 #The horizontal dimension of the fillet on the right
side of the deck (identified as f2 on the Define Bridge Section Data form). [L]
                        ret                                                            =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetFilletHorizontalDimensions(myDeckSection
Name, LeftFillet, RightFillet)

                        SlabMaterial = f'C{f_ck} - Tabuleiro' #The name of the slab material
property. This is intended to be a concrete property
                        GirderLongitudinalLayout = 1
                        # Indicates how the girders are laid out at a horizontal curve in a
bridge object.
                        # 1 = Along layout line.
                        # 2 = Straight line.
                        ret                                                            =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetGeneralData(myDeckSectionName,
SlabMaterial, NumberInteriorGirders, TotalWidthOfDeckSection, GirderLongitudinalLayout)

                        ConstantGirderSection = True #Indicates if all of the girder sections
are the same.
                        GirderSection = [f'Longarina (C{f_ck})']
                        ret                                                            =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetGirderSectionProperties(myDeckSectionNam
e, ConstantGirderSection, GirderSection)

                        ConstantSpacing = True
                        Spacing = []
                        ret                                                            =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetGirderSpacingData(myDeckSectionName,
ConstantSpacing, Spacing)


                        DistanceToFillet = b_long[0]/2 #The overhang distance to the fillet
(identified as L3 on the Define Bridge Section Data form). [L] Qualquer valor inferior à metade
da largura do flange superior dará erro na definição da seção
                        OuterThickness = h_laje #The overhang outer thickness (identified as
t5 on the Define Bridge Section Data form). [L]
                        ret                                                            =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetLeftOverhangDimensions(myDeckSectionName
, OverhangLength, DistanceToFillet, OuterThickness)

                        DistanceToFillet = b_long[0]/2 #The overhang distance to the fillet
(identified as L3 on the Define Bridge Section Data form). [L]
                        OuterThickness = h_laje #The overhang outer thickness (identified as
t5 on the Define Bridge Section Data form). [L]
                        ret                                                            =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetRightOverhangDimensions(myDeckSectionNam
e, OverhangLength, DistanceToFillet, OuterThickness)

                        TopSlabThickness = h_laje #The thickness of the top slab (t1 on Define
Bridge Section Data form). [L]
```

```
                        ConstantHaunchHeight = True #Indicates if the haunch heights are
constant for all girders.
                        HaunchHeight = [0]
                        # A zero-based, one-dimensional array of girder haunch heights
dimensioned to (NumberInteriorGirders + 1)
                        # The haunch heights are entered in order in this array from the left
exterior girder to the right exterior girder.
                        # Only the first height in this array is used if the
ConstantHaunchHeight item is True. [L]
                        ret                                                              =
MyBridgeModeler.DeckSection.PrecastConcreteIGirder.SetSlabThicknessData(myDeckSectionName,
TopSlabThickness, ConstantHaunchHeight, HaunchHeight)

                        X = 0 #The offset in the X direction from the reference point to the
insertion point. [L]
                        Y = 0 #The offset in the Y direction from the reference point to the
insertion point. [L]
                        ret                                                              =
MyBridgeModeler.DeckSection.InsertionPoint.SetPoint(myDeckSectionName, X, Y)




                        # 10 Bridge Section Data - Loads (cBMDeckSectionLoadData Interface) -
Métodos: SetBarrierLoads, SetConcretePourFormLoads, SetConcretePourHaunchLoads,
                        #    SetConcretePourPermanentLoads,    SetConcretePourTemporaryLoads,
SetLiveLoadCurbLocations, SetSidewalkLoads, SetTemperatureLoads, SetWearingSurfaceLoads

                        # 10.1 Carregamentos Deck Section Data

                        LoadPatternName = 'Defensas' #This is either None or the name of the
load pattern to which the barrier load is applied. The intent is that the load pattern would be
of type Dead.
                        NumberBarriers = 2 #The number of barriers specified
                        MeasureFrom = [1,2]
                        # A zero-based one-dimensional array indicating the reference point the
barrier location is measured from for each barrier.
                        # 1 = Left Edge of Deck
                        # 2 = Right Edge of Deck
                        DistanceToOuterEdge = [0,0] #The distance from the reference point
specified by the MeasureFrom item to the outer edge of the barrier. [L]
                        Width = [40,40] #The barrier width. [L]
                        Height = [81,81] #The barrier height. [L]
                        WeightEccentricity = [0,0] #The eccentricity of the barrier weight. [L]
                        WeightPerLength = [5.075/100,5.075/100] #The barrier weight per unit
length. [F/L]
                        ret                                                              =
MyBridgeModeler.DeckSection.LoadData.SetBarrierLoads(myDeckSectionName,       LoadPatternName,
NumberBarriers,   MeasureFrom,   DistanceToOuterEdge,   Width,   Height,   WeightEccentricity,
WeightPerLength)

                        LoadPatternName = '' #This is either None or the name of the load
pattern to which the haunch load is applied. The intent is that the load pattern would be of
type Dead.
                        UnitWeight = float() #The unit weight of the form load. [F/L2]
                        ret                                                              =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourFormLoads(myDeckSectionName,
LoadPatternName, UnitWeight)

                        LoadPatternName = 'None'
                        ProgramDetermined = True #This flag indicates whether the haunch load
is program determined or user defined.
                        UserValue = float()#The haunch load value. This item only applies when
the ProgramDetermined item is False. [F/L]
                        ret                                                              =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourHaunchLoads(myDeckSectionName,
LoadPatternName, ProgramDetermined, UserValue)

                        LoadPatternName = ''
                        WearingSurface = float() #The integral wearing surface load. [F/L2]
                        Utilities = float() #The utilities load. [F/L2]
                        Miscellaneous = float() #The miscellaneous load. [F/L2]
```

```
                        ret = MyBridgeModeler.DeckSection.LoadData.SetConcretePourPermanentLoads(myDeckSectionName, LoadPatternName, WearingSurface, Utilities, Miscellaneous)

                        AdditionalOverhang = float()#The additional overhang load. [F/L2]
                        ret = MyBridgeModeler.DeckSection.LoadData.SetConcretePourTemporaryLoads(myDeckSectionName, LoadPatternName, AdditionalOverhang, Utilities, Miscellaneous)

                        DistToInsideEdgeOfLeftCurb = float()
                        DistToInsideEdgeOfRightCurb = float()
                        DistanceToCenterlineOfMedianCurb = float()
                        WidthOfMedianCurb = float()
                        ret = MyBridgeModeler.DeckSection.LoadData.SetLiveLoadCurbLocations(myDeckSectionName, ProgramDetermined,           DistToInsideEdgeOfLeftCurb,           DistToInsideEdgeOfRightCurb, DistanceToCenterlineOfMedianCurb, WidthOfMedianCurb)

                        LoadPatternName = '' #This is either None or the name of the load pattern to which the sidewalk load is applied. The intent is that the load pattern would be of type Dead.
                        NumberSidewalks = 0
                        MeasureFrom = []
                        DistanceToOuterEdge =[]
                        Width = []
                        WeightPerArea = []
                        ret = MyBridgeModeler.DeckSection.LoadData.SetSidewalkLoads(myDeckSectionName,      LoadPatternName, NumberSidewalks, MeasureFrom, DistanceToOuterEdge, Width, WeightPerArea)

                        LoadPatternPositiveName = 'Temperatura uniforme pos'#This is either None or the name of the load pattern to which the positive temperature load is applied. The intent is that the load pattern would be of type Temperature.
                        TemperatureChangePositive = 15 #The positive temperature change. [T]
                        LoadPatternNegativeName = 'Temperatura uniforme neg'#This is either None or the name of the load pattern to which the negative temperature load is applied. The intent is that the load pattern would be of type Temperature.
                        TemperatureChangeNegative = -15 #The negative temperature change that should be entered as zero or a negative number. [T]
                        ret = MyBridgeModeler.DeckSection.LoadData.SetTemperatureLoads(myDeckSectionName, LoadPatternPositiveName,          TemperatureChangePositive,          LoadPatternNegativeName, TemperatureChangeNegative)

                        LoadPatternName = 'Pav + recap'
                        WeightPerArea = 4.4/(10**4)
                        ret = MyBridgeModeler.DeckSection.LoadData.SetWearingSurfaceLoads(myDeckSectionName, LoadPatternName, WeightPerArea)

                        # 10.2 Limpeza das inutilizadas
                        ret = SapModel.LoadPatterns.Delete('Temperature Pos')
                        ret = SapModel.LoadPatterns.Delete('Temperature Neg')
                        ret = SapModel.LoadPatterns.Delete('ConcPour Permanent')
                        ret = SapModel.LoadPatterns.Delete('ConcPour Temporary')
                        ret = SapModel.LoadPatterns.Delete('PreTension')
                        ret = SapModel.LoadPatterns.Delete('Barrier')




                        # 11 Retração e fluência - Staged Construction nos elementos da ponte
                        #  Apenas os load cases Veiculo, Multidao, Retração, Fluência e Cronograma são necessários de definição (os load patterns já viram load cases automaticamente pela definição dada)

                        # 11.1 Update no modelo para ser possível selecionar os elementos da ponte
                        Name = myBridgeObjName
                        Action = 4 #Atualização do modelo para ser possível de selecionar os elementos do objeto ponte
                        # An integer value between 1 and 4, inclusive, indicating the update action to be taken.
                        # 1 = Update Linked Model
```

```
                        # 2 = Clear All From Linked Model
                        # 3 = Convert To Unlinked Model
                        # 4 = Clear and Create Linked Model
                        ModelType = 2
                        #An integer value between 1 and 3, inclusive, indicating the model type
to be created. This item applies when the Action item is 1 or 4.
                        # 1 = Update as Spine Model Using Frame Objects
                        # 2 = Update as Area Object Modell
                        # 3 = Update as Solid Object Model
                        SubmeshSize = 50 #The preferred maximum submesh size. This item applies
when the Action item is 1 or 4 and the ModelType item is 2 or 3. [L]
                        ApplyAreaPropModifier = False #This item is to indicate whether or not
to apply property modifiers to area objects representing the bridge section to adjust for
overlap.
                        # This option only applies to bridge sections of type concrete box-
girder, concrete T-beam, flat slab, and user-defined (Section Designer).
                        # When the bridge object is updated as an area-object model (ModelType
is "Area"), entering True will apply property modifiers to the area objects representing the
bridge cross section that will adjust their weight, mass, and stiffness factors f11 and m11
                        # to match a corresponding solid-object model (ModelType is "Solid"),
which has no overlaps. This item applies when the Action item is 1 or 4 and the ModelType item
is 2. [L]
                        # True = Apply area property modifiers
                        # False = Do not apply area property modifiers
                        MaxDeckSegmentLength = 50 #The maximum segment length for deck spans.
This item applies when the Action item is 1 or 4. [L]
                        MaxCapSegmentLength = 50 #The maximum segment length for bent cap beams.
This item applies when the Action item is 1 or 4. [L]
                        MaxColumnSegmentLength = 50 #The maximum segment length for bent
columns. This item applies when the Action item is 1 or 4. [L]
                        MeshSlabAtCriticalSteelIGirderLocations = False #A flag indicating that
the slab should be meshed at critical steel I-girder locations. This item applies when the
Action item is 1 or 4 and the bridge deck section is a Steel I-Girder section.
                        ret = MyBridgeModeler.BridgeObject.Update(Name, Action, ModelType,
SubmeshSize, ApplyAreaPropModifier, MaxDeckSegmentLength, MaxCapSegmentLength,
MaxColumnSegmentLength, MeshSlabAtCriticalSteelIGirderLocations)

                        # 11.2 Retração e fluência

                        ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('Retração e
fluência')

                        InitialCase = ''
                        ret =
SapModel.LoadCases.StaticNonlinearStaged.GetInitialCase('Retração e fluência', InitialCase)

                        NLGeomType = 1
                        # This is 0, 1 or 2, indicating the geometric nonlinearity option
selected for the load case.
                        # 0 = None
                        # 1 = P-delta
                        # 2 = P-delta plus large displacements
                        ret =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('Retração e fluência',
NLGeomType)

                        TimeDepMatProp = True
                        ret =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('Retração e fluência',
TimeDepMatProp)

                        StagedSaveOption = 0
                        # This is 0, 1, 2 or 3, indicating the results saved option for the
load case.
                        # 0 = End of final stage
                        # 1 = End of each stage
                        # 2 = Start and end of each stage
                        # 3 = Two or more times in each stage


                        ret =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('Retração e fluência',
StagedSaveOption)
```

```
                        NumberStages = 2 #The number of stages defined for the specified load
case.
                        Duration = [3, 3000]#This is an array that includes the duration in
days for each stage.
                        Output = [True, True]#This is an array that includes True or False,
indicating if analysis output is to be saved for each stage.
                        OutputName = ['Step 1', 'Step 2']#This is an array that includes a
user-specified output name for each stage.
                        Comment = ['', '']#This is an array that includes a comment for each
stage. The comment may be a blank string.

                        ret                                                                =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('Retração      e      fluência',
NumberStages, Duration, Output, OutputName, Comment)

                        Stage = 1 #The stage in the specified load case to which the data
applies. Stages are numbered sequentially starting from 1.
                        NumberOperations = 2 #The number of operations in the specified stage
                        Operation = [1,1]
                        # This is an array that includes 1, 2, 3, 4, 5, 6, 7, or 11, indicating
an operation type.
                        # 1 = Add structure
                        # 2 = Remove structure
                        # 3 = Load objects if new
                        # 4 = Load objects
                        # 5 = Change section properties
                        # 6 = Change section property modifiers
                        # 7 = Change releases
                        # 11 = Change section properties and age
                        ObjectType = ['Group', 'Group']
                        # This is an array that includes the object type associated with the
specified operation. The object type may be one of the following:
                        # Group
                        # Frame
                        # Cable
                        # Tendon
                        # Area
                        # Solid
                        # Link
                        # Point
                        # The following list shows which object types are applicable to each
operation type:
                        # Operation = 1 (Add structure):  All object types
                        # Operation = 2 (Remove structure):  All object types
                        # Operation = 3 (Load objects if new):  All object types
                        # Operation = 4 (Load objects):  All object types
                        # Operation = 5 (Change section properties):  All object types except
Point
                        # Operation = 6 (Change section property modifiers):  Group, Frame,
Cable, Area
                        # Operation = 7 (Change releases):  Group, Frame
                        # Operation = 11 (Change section properties and age): All object types
except Point
                        ObjectName           =           [f'<{myBridgeObjName}>ALL_BUT_SLAB',
f'<{myBridgeObjName}>SLAB']
                        # This is an array that includes the name of the object associated with
the specified operation.
                        # This is the name of a Group, Frame object, Cable object, Tendon
object, Area object, Solid object, Link object or Point object, depending on the ObjectType
item.
                        Age = [28, 38]#This is an array that includes the age of the added
structure, at the time it is added, in days. This item applies only to operations with Operation
= 1.
                        MyType = ['','']
                        # This is an array that includes a load type or an object type, depending
on what is specified for the Operation item. This item applies only to operations with Operation
= 3, 4, 5, 6, 7, or 11.
                        # When Operation = 3 or 4, this is an array that includes Load or Accel,
indicating the load type of an added load.
                        # When Operation = 5 or 11, and the ObjectType item is Group, this is
an array that includes Frame, Cable, Tendon, Area, Solid or Link, indicating the object type
for which the section property is changed.
                        # When Operation = 6 and the ObjectType item is Group, this is an array
that includes Frame, Cable or Area, indicating the object type for which the section property
modifiers are changed.
```

```
                        # When Operation = 7 and the ObjectType item is Group, this is an array
that includes Frame, indicating the object type for which the releases are changed.
                        # When Operation = 5, 6, 7, or 11,  and the ObjectType item is not
Group and not Point, this item is ignored and the type is picked up from the ObjectType item.
                        MyName = ['','']
                        # This is an array that includes a load assignment or an object name,
depending on what is specified for the Operation item. This item  applies only to operations
with Operation = 3, 4, 5, 6, 7 or 11.
                        # When Operation = 3 or 4, this is an array that includes the name of
the load assigned to the operation. If the associated LoadType item is Load, this item is the
name of a defined load pattern. If the associated LoadType item is Accel, this item is UX, UY,
UZ, RX, RY or RZ, indicating the direction of the load.
                        # When Operation = 5 or 11, this is the name of a Frame, Cable, Tendon,
Area, Solid or Link object, depending on the object type specified.
                        # When Operation = 6, this is the name of a Frame, Cable or Area object,
depending on the object type specified.
                        # When Operation = 7, this is the name of a Frame object.
                        SF = [float(), float()]
                        # This is an array that includes the scale factor for the load assigned
to the operation, if any. [L/s2] for Accel UX UY and UZ; otherwise unitless
                        # This item applies only to operations with Operation = 3 or 4.
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetStageData_2('Retração    e    fluência',    Stage,
NumberOperations, Operation, ObjectType, ObjectName, Age, MyType, MyName, SF)

                        Stage = 2
                        NumberOperations = 1
                        Operation = [1]
                        ObjectType = ['Group']
                        ObjectName = [f'<{myBridgeObjName}>ALL']
                        Age = [3000]
                        MyType = ['']
                        MyName = ['']
                        SF = [float()]
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetStageData_2('Retração    e    fluência',    Stage,
NumberOperations, Operation, ObjectType, ObjectName, Age, MyType, MyName, SF)

                        # Combinações de ações (definir depois de retração e fluência)

                        ws_long = wb_long.sheets["5 Esforços solicitantes"]
                        CI = ws_long['F190'].value # Coeficiente de impacto

                        # Defensas + pav + recap
                        Name = 'G3' # The name of a new load combination.
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName = ['Defensas', 'Pav + recap']
                        for loadcase in CName:
                            CNameType = eCNameType(0)
                            SF = 1
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        # Carga permanente
                        Name = 'Carga permanente' # The name of a new load combination.
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)
                        ret = SapModel.RespCombo.SetCaseList(Name, eCNameType(0), 'DEAD', 1)
                        ret = SapModel.RespCombo.SetCaseList(Name, eCNameType(1), 'G3', 1)

                        # TB-450

                        Name = 'TB-450' # The name of a new load combination.
                        ComboType = 0
                        # This is 0, 1, 2, 3 or 4 indicating the load combination type.
                        # 0 = Linear Additive
                        # 1 = Envelope
                        # 2 = Absolute Additive
                        # 3 = SRSS
                        # 4 = Range Additive
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName = ['Veiculo', 'Multidao']
```

```
                        for loadcase in CName:
                            CNameType = eCNameType(0)
                            # This is one of the following items in the eCNameType enumeration:
                            # LoadCase = 0
                            # LoadCombo = 1
                            # The name of the load case or load combination to be added to or
modified in the combination specified by the Name item.
                            # If the load case or combination already exists in the combination
specified by the Name item, the scale factor is modified as indicated by the SF item for that
load case or combination.
                            #If the analysis case or combination does not exist in the
combination specified by the Name item, it is added.
                            SF = 1 # The scale factor multiplying the case or combination
indicated by the CName item.
                            ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase,
SF)

                        # ENVOLTÓRIA TEMPERATURA

                        Name = 'ENV-TEMP' # The name of a new load combination.
                        ComboType = 1
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName = ['Temperatura uniforme pos', 'Temperatura uniforme neg',
'Temperatura gradiente pos', 'Temperatura gradiente neg']
                        for loadcase in CName:
                            CNameType = eCNameType(0)
                            SF = 1
                            ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase,
SF)


                        # CRIAÇÃO DO DICIONÁRIO PARA REALIZAR AS COMBINAÇÕES -> o primeiro
valor da lista referente ao ELU é quando a ação é a AVP

                        Comb_dict = {
                                    'Carga permanente': {'Type': 'Load Combo', 'ELU' : [1.35],
'ELU-fad' : 1, 'ELS-W': 1, 'EXC' : 1.1},
                                    'Empuxo terra': {'Type': 'Load Case', 'ELU' : [1.35], 'ELU-
fad' : 1, 'ELS-W': 1, 'EXC' : 1.1},
                                    'Retração e fluência': {'Type': 'Load Case', 'ELU' : [1.2],
'ELU-fad' : 0, 'ELS-W': 0, 'EXC' : 0},
                                    'Protensão': {'Type': 'Load Case', 'ELU' : [1.2], 'ELU-
fad' : 0, 'ELS-W': 0, 'EXC' : 0},
                                    'TB-450': {'Type': 'Load Combo', 'ELU' : [CI * 1.5, CI *
1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                                    'Frenagem e aceleracao': {'Type': 'Load Case', 'ELU' : [CI
* 1.5, CI * 1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                                    'Empuxo terra carga móvel': {'Type': 'Load Case', 'ELU' :
[CI * 1.5, CI * 1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                                    'Vento': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-
fad' : 0, 'ELS-W': 0, 'EXC' : 0.6},
                                    'Água': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-
fad' : 0, 'ELS-W': 0, 'EXC' : 0.6},
                                    'ENV-TEMP': {'Type': 'Load Combo', 'ELU' : [1.2, 0.72],
'ELU-fad' : 0.3, 'ELS-W': 0.3, 'EXC' : 0.6},
                                    'Impacto': {'Type': 'Load Case', 'ELU' : 0, 'ELU-fad' : 0,
'ELS-W': 0, 'EXC' : 1}
                                    }

                        AV = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel',
'Vento', 'Água', 'ENV-TEMP'] # Lista das ações variáveis

                        Name = 'COMB1 - ELU (CM)' # Carga móvel como ação variável principal
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência',
'Protensão', 'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água',
'ENV-TEMP']
                        AVP = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel']
                        for loadcase in CName:
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
```

```
                            SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase
not in AV else Comb_dict[loadcase]['ELU'][1]
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        Name = 'COMB2 - ELU (VENTO)' # Vento como ação variável principal
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['Carga permanente', 'Empuxo terra', 'Retração e fluência',
'Protensão', 'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água',
'ENV-TEMP']
                        AVP = ['Vento']
                        for loadcase in CName:
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                            SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase
not in AV else Comb_dict[loadcase]['ELU'][1]
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        Name = 'COMB3 - ELU (ÁGUA)' # Ação da água como AVP
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['Carga permanente', 'Empuxo terra', 'Retração e fluência',
'Protensão', 'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água',
'ENV-TEMP']
                        AVP = ['Água']
                        for loadcase in CName:
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                            SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase
not in AV else Comb_dict[loadcase]['ELU'][1]
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        Name = 'COMB4 - ELU (TEMPERATURA)' # Temperatura como AVP
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['Carga permanente', 'Empuxo terra', 'Retração e fluência',
'Protensão', 'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água',
'ENV-TEMP']
                        AVP = ['ENV-TEMP']
                        for loadcase in CName:
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                            SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase
not in AV else Comb_dict[loadcase]['ELU'][1]
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        Name = 'COMB5 - ELU-fad' # Combinação de fadiga
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['Carga permanente', 'Empuxo terra', 'TB-450', 'Frenagem e
aceleracao', 'Empuxo terra carga móvel', 'ENV-TEMP']
                        for loadcase in CName:
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                            SF = Comb_dict[loadcase]['ELU-fad']
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        Name = 'COMB6 - ELS-W' # Combinação de estado limite de abertura de
fissuras
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['Carga permanente', 'Empuxo terra', 'TB-450', 'Frenagem e
aceleracao', 'Empuxo terra carga móvel', 'ENV-TEMP']
                        for loadcase in CName:
```

```
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                            SF = Comb_dict[loadcase]['ELS-W']
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        Name = 'COMB7 - EXC' # Combinação excepcional
                        ComboType = 0
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['Carga  permanente',  'Empuxo  terra',  'TB-450',  'Frenagem  e
aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP', 'Impacto']
                        AVP = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel']
                        for loadcase in CName:
                            CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                            SF = Comb_dict[loadcase]['EXC']
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)

                        # Envoltória ELU
                        Name = 'ENV - ELU' # The name of a new load combination.
                        ComboType = 1
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName  =  ['COMB1  -  ELU  (CM)',  'COMB2  -  ELU  (VENTO)',  'COMB3  -  ELU
(ÁGUA)', 'COMB4 - ELU (TEMPERATURA)']
                        for loadcase in CName:
                            CNameType = eCNameType(1)
                            SF = 1
                            ret  =  SapModel.RespCombo.SetCaseList(Name,  CNameType,  loadcase,
SF)


                        NumberNames = int() # The number of vehicle class names retrieved by
the program.
                        MyName = []
                        # This is a one-dimensional array of vehicle class names. The MyName
array is created as a dynamic, zero-based array by the API user: Dim MyName() as String.
                        # The array is dimensioned to (NumberNames - 1) inside the program,
filled with the names, and returned to the API user.
                        ret = MyBridgeModeler.Vehicle.GetNameList(NumberNames, MyName)
                        NumberVehicles = ret[1]
                        VehiclesList = list(ret[2])
                        ValidVehiclesList = ['Veiculo', 'Multidao']
                        for i in range (NumberVehicles):
                            if VehiclesList[i] not in ValidVehiclesList:
                                ret = MyBridgeModeler.Vehicle.Delete(VehiclesList[i])




                        # 12 Atribuição dos Encontros (cBMAbutmentAssign Methods)

                        Elevation_StartAbutment  =  -h_laje  -h_long  -  h_ap  #The  global  Z
elevation of the substructure. (start abutment) -> Distância entre o topo da laje e o CG do
aparelho de apoio
                        ElevationAtLayoutLine_StartAbutment = -h_laje -h_long - h_ap/2 #The
bearing elevation (global Z) at the layout line. [L] -> Distância entre o topo da laje e a base
do aparelho de apoio
                        Elevation_EndAbutment = Elevation_StartAbutment #The global Z elevation
of the substructure.
                        ElevationAtLayoutLine_EndAbutement                                    =
ElevationAtLayoutLine_StartAbutment #The bearing elevation (global Z) at the layout line. [L]

                        # 12.1 Start Abutment
                        Name = myBridgeObjName #The name of a bridge object.
                        AbutmentLocation = 1
                        # This is one of the following indicating the abutment location:
                        # 1 = Start abutment.
                        # 2 = End abutment.
                        SupportName  =  'Start  Abutment'  #The  support  name.  The  support  name
cannot be blank and it can not be the same as another abutment or bent support name
```

```
                        AbutmentDirection = 'Default' #The bearing of a line parallel to the
abutment. It is either Default or a valid bearing string.
                        DiaphragmProperty = 'None' #This is either None (meaning no diaphragm
exists at the abutment) or the name of a defined bridge diaphragm property.
                        InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-
girder or Super-T girder deck sections.
                        #It does not apply to other deck sections. It is either None (meaning
no diaphragm interior diaphragm exists) or the name of a defined bridge diaphragm property.
                        ClosurePourDiaphragm = 'None' #This is either None or the name of a
defined bridge diaphragm property of type Closure Pour.
                        #This only applies if the substructure type is Bent Property and the
specified bent property is of type Double Bearing Line with Closure Pour
                        DiaphragmOffsetLocation = 1
                        #This is one of the following indicating the diaphragm offset location.
This item only applies when the type of diaphragm defined by the DiaphragmProperty item is
Solid.
                        # 0 = Flush
                        # 1 = Centerline
                        ret                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

                        if Tipo_analise == 'SUPERESTRUTURA' or Tipo_analise == 'MESOESTRUTURA':
                            SubstructureType = 1
                            AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the
abutment or bent property used for the substructure. [L]
                        else:
                            SubstructureType = 2
                            AbutmentOrBentProperty = 'Apoios extremos'
                        HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the abutment. [L]
                        ret                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,   AbutmentLocation,
SubstructureType, AbutmentOrBentProperty, Elevation_StartAbutment, HorizontalOffset)

                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the substructure.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the substructure.
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int() #The number of bearings for the bridge section
or for abutments at Steel U Girder deck sections the number of bearings at the U girder.
                        #For bearings at steel U girder sections this item must be 1 or 2. This
item only applies when the bearing assignment type is General or when the abutment is at a Steel
U Girder deck section.
                        UniformSpacing = float() #The uniform spacing of bearings. This item
only applies when the bearing assignment type is General. [L]
                        BearingOffset = float() #The offset from the section reference point
to the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,       AbutmentLocation,
BearingAssignmentType,                  BearingProperty,                RestrainerProperty,
ElevationAtLayoutLine_StartAbutment,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

                        # 12.2 End Abutment
                        Name = myBridgeObjName #The name of a bridge object.
                        AbutmentLocation = 2
                        # This is one of the following indicating the abutment location:
                        # 1 = Start abutment.
                        # 2 = End abutment.
                        SupportName = 'End Abutment'
                        AbutmentDirection = 'Default' #The bearing of a line parallel to the
abutment. It is either Default or a valid bearing string.
                        DiaphragmProperty = 'None' #This is either None (meaning no diaphragm
exists at the abutment) or the name of a defined bridge diaphragm property.
```

```
                        InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-
girder or Super-T girder deck sections.
                        #It does not apply to other deck sections. It is either None (meaning
no diaphragm interior diaphragm exists) or the name of a defined bridge diaphragm property.
                        ClosurePourDiaphragm = 'None' #This is either None or the name of a
defined bridge diaphragm property of type Closure Pour.
                        #This only applies if the substructure type is Bent Property and the
specified bent property is of type Double Bearing Line with Closure Pour
                        DiaphragmOffsetLocation = 1
                        #This is one of the following indicating the diaphragm offset location.
This item only applies when the type of diaphragm defined by the DiaphragmProperty item is
Solid.
                        # 0 = Flush
                        # 1 = Centerline
                        ret                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

                        SubstructureType = 1
                        AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the
abutment or bent property used for the substructure. [L]
                        HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the abutment. [L]
                        ret                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,   AbutmentLocation,
SubstructureType, AbutmentOrBentProperty, Elevation_EndAbutment, HorizontalOffset)

                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the substructure.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the substructure.
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int() #The number of bearings for the bridge section
or for abutments at Steel U Girder deck sections the number of bearings at the U girder.
                        #For bearings at steel U girder sections this item must be 1 or 2. This
item only applies when the bearing assignment type is General or when the abutment is at a Steel
U Girder deck section.
                        UniformSpacing = float() #The uniform spacing of bearings. This item
only applies when the bearing assignment type is General. [L]
                        BearingOffset = float() #The offset from the section reference point
to the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,        AbutmentLocation,
BearingAssignmentType, BearingProperty, RestrainerProperty, ElevationAtLayoutLine_EndAbutement,
RotationAngle, NumberOfBearings, UniformSpacing, BearingOffset)

                        ret = MyBridgeModeler.Abutment.Delete('BABT1')




                        # 13 Schedule
                        ret = SapModel.Scheduler.Add('Cronograma')

                        Name = 'Cronograma'
                        NumberTasks = 11
                        Expanded = [False, True, True, True, True, True, True, True, False,
True, True]
                        IndentLevel = [0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1]
                        TaskName  =  ['Superestrutura',  'Guide',  'Longarina',  'Protensão',
'Intervalo', 'Transversina + Laje', 'Defensa', 'Pavimentação', 'Pós-construção', 'Início da
operação', 'Final']
                        Duration = ['69', '1', '28', '1', '1', '10', '27', '1', '3028', '28',
'3000']
                        CompletionDate = [69, 1, 29, 30, 31, 41, 68, 69, 3097, 97, 3097]
                        Dependencies = ['', '', '2', '3', '4', '5', '6', '7', '8', '', '10']
```

```
                        Operation = ['', 'Add Guide Structure', 'Add Structure', 'Load
Objects', '', 'Add Structure', 'Load Objects', 'Load Objects', '', '', '']
                        IncludeSW = [bool(), bool(), True, bool(), bool(), True, bool(),
bool(), bool(), bool(), bool()]
                        ObjectType = ['', 'Group', 'Group', 'Group', '', 'Group', 'Group',
'Group', '', '', '']
                        ObjectName          =          ['',          f'<{myBridgeObjName}>SLAB',
f'<{myBridgeObjName}>ALL_BUT_SLAB',           f'<{myBridgeObjName}>ALL_BUT_SLAB',           '',
f'<{myBridgeObjName}>SLAB', 'ALL', 'ALL', '', '', '']
                        Details = ['', '', '', '', '', '', '', '', '', '', '']
                        Age = [float(), float(), 28, float(), float(), 3, float(), float(),
float(), float(), float()]
                        LoadType = ['', '', 'Load pattern', 'Load pattern', '', 'Load pattern',
'Load pattern', 'Load pattern', '', '', '']
                        LoadName = ['', '', 'DEAD', 'Protensão', '', 'DEAD', 'Defensas', 'Pav
+ recap', '', '', '']
                        LoadSF = [float(), float(), 1, 1, float(), 1, 1, 1, float(), float(),
float()]
                        Solve = [False, False, False, False, False, False, False, False, False,
False, False]
                        SaveStiffness = [False, False, False, False, False, False, False,
False, False, False, False]
                        OutputID = ['', 'GuideSlab', '', 'Protensão', '', 'Transversina +
laje', 'Defensa', 'Pavimentação', '', 'Início da operação', 'Final']
                        ItemType = ['', '', '', '', '', '', '', '', '', '', '']
                        ItemName = ['', '', '', '', '', '', '', '', '', '', '']
                        Notes = ['', '', '', '', '', '', '', '', '', '', '']
                        ret    =    SapModel.Scheduler.SetScheduleData_1(Name,    NumberTasks,
Expanded, IndentLevel, TaskName, Duration, CompletionDate, Dependencies, Operation, IncludeSW,
ObjectType, ObjectName,
                        Details, Age, LoadType, LoadName, LoadSF, Solve, SaveStiffness,
OutputID, ItemType, ItemName, Notes)

                        Name = 'Cronograma'
                        Unloading = 'Unload Entire'
                        GeoNonLin = 'P-Delta'
                        MaxTotal = 100
                        MaxNull = 200
                        MaxIterCS = 200
                        MaxIterNR = 200
                        ItConvTol = 0.001
                        UseEvStep = 'Yes'
                        EvLumpTol = 0.01
                        LSPerIter = 20
                        LSTol = 0.1
                        LSStepFact = 1.618
                        StageSave = 'End of Final Stage'
                        StageMinIns = int()
                        StageMinTD = int()
                        TimeDepMat = 'Yes'
                        TFMaxIter = 100
                        TFTol = 0.01
                        TFAccelFact = 1
                        TFNoStop = 'No'
                        ret    =    SapModel.Scheduler.SetScheduleParameters(Name,    Unloading,
GeoNonLin, MaxTotal, MaxNull, MaxIterCS, MaxIterNR, ItConvTol, UseEvStep, EvLumpTol, LSPerIter,
LSTol, LSStepFact,
                        StageSave, StageMinIns, StageMinTD, TimeDepMat, TFMaxIter, TFTol,
TFAccelFact, TFNoStop)




                        # 14 Carregamentos atuantes
                        # 14.1 Bridge Object - Slab Wet Concrete Load Assigns
                        Name = myBridgeObjName
                        ConcretePourName = '' #The concrete pour name to be deleted. Enter a
blank string to delete all slab wet concrete load assignments.
                        ret                                                                =
MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.Delete(myBridgeObjName, '')

                        # 14.2 Bridge Object - Temperature Load Assigns
                        LoadPat = ['Temperatura gradiente pos', 'Temperatura gradiente neg'] #
A zero-based one-dimensional array of the load pattern used for each load assignment.
```

```
                        TempGradientName = ['Temperatura gradiente', 'Temperatura gradiente']
# A zero-based one-dimensional array of the name of the defined temperature gradient for each
load assignment. This item only applies when the TemperatureOption item is 1 (Temperature
gradient).
                        GradientType = [0, 1]
                        # A zero-based one-dimensional array of the gradient type for each load
assignment. This item only applies when the TemperatureOption item is 1 (Temperature gradient).
                        # 0 = Positive temperature change
                        # 1 = Negative temperature change
                        for i in range(NVãos):
                            for j in range(2):
                                Name = myBridgeObjName
                                SpanNames = f'Vão {i+1}' # A zero-based one-dimensional array
of the bridge span for each load assignment.
                                TemperatureOption = 1
                                # A zero-based one-dimensional array of the temperature option
for each load assignment.
                                # 0 = Uniform temperature change
                                # 1 = Temperature gradient
                                UniformTempChange = float() # A zero-based one-dimensional
array of the uniform temperature change for each load assignment. This item only applies when
the TemperatureOption item is 0 (Uniform temperature change). [T]
                                ret                                                    =
MyBridgeModeler.BridgeObject.LoadAssignTemperature.SetOneLoad(Name,   SpanNames,   LoadPat[j],
TemperatureOption, UniformTempChange, TempGradientName[j], GradientType[j])

                        # 14.3 Bridge Object - Line Load Assigns

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Vento' # The load pattern for the load assignment.
                        LoadDistribution = 'Vento' #The named load distribution for the load
assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        TransverseVariation = 'None' # The variation used for the load
assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignLine.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, TransverseVariation)

                        # 14.4 Bridge Object - Area Load Assigns

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Frenagem e aceleracao' # The load pattern for the load
assignment.
                        LoadDistribution = 'Frenagem e aceleracao' #The named load distribution
for the load assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        LeftEdgeVariation = 'None' # The left edge variation used for the load
assignment. This is either None or the name of a defined variation.
                        RightEdgeVariation = 'None' # The right edge variation used for the
load assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)




                        # 15 Update no modelo
                        Name = myBridgeObjName
                        Action = 1 #Atualização do modelo para ser possível de selecionar os
elementos do objeto ponte
                        # An integer value between 1 and 4, inclusive, indicating the update
action to be taken.
                        # 1 = Update Linked Model
                        # 2 = Clear All From Linked Model
                        # 3 = Convert To Unlinked Model
                        # 4 = Clear and Create Linked Model
                        ret    =    MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name,
Action)
```

```
                        # 16 Definição dos grupos comuns para pegar resultados da análise e
modelagem dos encontros

                        # Grupos comuns
                        Elementos = ['Longarinas', 'Transversinas']
                        PropSecoes  =  [f'Longarina  (C{f_ck})',  f'Transversina  de  apoio
(C{f_ck})']
                        for i in range(len(Elementos)):
                            ret = SapModel.SelectObj.ClearSelection()
                            ret = SapModel.GroupDef.SetGroup(Elementos[i])
                            ret = SapModel.SelectObj.PropertyFrame(PropSecoes[i])
                            ret  =  SapModel.FrameObj.SetGroupAssign('',  Elementos[i],  False,
eItemType(2))

                        ret = SapModel.SelectObj.ClearSelection()




                        # 17 Definição das Lanes - (cBMLane.SetLaneFromLayoutLine) [myLaneName
sempre será a faixa de tráfego]

                        Name = myLaneName #The name for the existing lane defined from a layout
line.
                        NumberLaneStations = 2 #The number of lane points, that is, the number
of items included in the that lane will be overwritten. LayoutLine, Station, Offset, Width and
Radius arrays.
                        LayoutLine  =  [myLayoutLineName, myLayoutLineName]#A zero-based array
containing the name of the bridge layout line associated with each lane station.
                        Station = [0, CTotal]#A zero-based array containing the station at each
lane station. [L]
                        Offset = [0]*2 #A zero-based array containing the offset from the layout
line to the center of the lane at each lane station. [L]
                        Width  =  [LarguraRolamento]*2  #A  zero-based  array  containing  the  the
width of the lane at each lane station. [L]
                        Radius  =  [0]*2  #A  zero-based  array  containing  the  the  radius  of  the
lane at each lane station. A positive radius curves to the right and negative to the left. [L]
                        LaneType = 0
                        #The lane type (fixed or floating).
                        # 0 = Fixed lane
                        # 1 = Floating lane set
                        if Tipo_analise == 'ENCONTROS':
                            LoadGroup = 'Todas lajes'
                        else:
                            LoadGroup = f'<{myBridgeObjName}>SLAB'
                        LeftEdgeIsExterior = False #Indicates whether the left edge of the lane
is exterior or interior.
                        RightEdgeIsExterior = False #Indicates whether the right edge of the
lane is exterior or interior.
                        MaxDiscAlongLane  =  300  #The  maximum  lane  load  discretization  length
along the lane. [L]
                        MaxDiscAcrossLane = 300 #The maximum lane load discretization length
across the lane. [L]
                        UseDiscFactorSpan = True #Indicates whether the the DiscFactorSpan item
is to apply to the lane.
                        UseDiscFactorLane = True #Indicates whether the the DiscFactorLane item
is to apply to the lane.
                        DiscFactorSpan = 4 #The lane load discretization length along the lane
will not be more than 1/x of the span length where x is the DiscFactorSpan value.
                        #This item is only applicable when the UseDiscFactorSpan item is True.
                        DiscFactorLane = 10 #The lane load discretization length along the lane
will not be more than 1/x of the lane length where x is the DiscFactorLane value.
                        #This item is only applicable when the UseDiscFactorLane item is True.
                        Color = int()#The color assigned to the lane. (Optional)
                        Notes = ''#The notes, if any, assigned to the lane. (Optional)
                        ret         =         MyBridgeModeler.Lane.SetLaneFromLayoutLine(Name,
NumberLaneStations,  LayoutLine,  Station,  Offset,  Width,  Radius,  LaneType,  LoadGroup,
LeftEdgeIsExterior, RightEdgeIsExterior,
                        MaxDiscAlongLane,        MaxDiscAcrossLane,        UseDiscFactorSpan,
UseDiscFactorLane, DiscFactorSpan, DiscFactorLane, Color, Notes)

                        if Tipo_analise == 'ENCONTROS':
                            ret = MyBridgeModeler.LayoutLine.AddNew('BLL0')
```

```
                            Name = 'BLL0'
                            CoordinateSystem = 'GLOBAL'
                            StartX = -1000
                            StartY = 0
                            StartZ = 0
                            ret       =       MyBridgeModeler.LayoutLine.SetGeneralData(Name,
CoordinateSystem, StartX, StartY, StartZ)

                            Name = 'BLL0'
                            NumberSegments = 2
                            SegmentType = [1, 2]
                            Station = [0, 1000]
                            Radius = [float(), float()]
                            Bearing = ['N900000E', 'N900000E']
                            ret   =   MyBridgeModeler.LayoutLine.SetHorizontalLayoutData(Name,
NumberSegments, SegmentType, Station, Radius, Bearing)

                            ret = MyBridgeModeler.Lane.AddNewFromLayoutLine('LANE0', 'BLL0')
                            ret   =   MyBridgeModeler.Lane.SetLaneFromLayoutLine('LANE0',   2,
['BLL0',  'BLL0'],  [0,  1000],  [0]*2,  [LarguraRolamento]*2,  [0]*2,  LaneType,  LoadGroup,
LeftEdgeIsExterior, RightEdgeIsExterior,
                            MaxDiscAlongLane,      MaxDiscAcrossLane,      UseDiscFactorSpan,
UseDiscFactorLane, DiscFactorSpan, DiscFactorLane, Color, Notes)

                            LoadNumber = 1 #The load assignment number.
                            NumberItems = 2 #The number of lanes loaded for the specified load
assignment number.
                            MyName = [f'{myLaneName}', 'LANE0'] #This is an array that includes
the name of each lane loaded for the specified load assignment number.
                            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo',
LoadNumber, NumberItems, MyName)
                            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Multidao',
LoadNumber, NumberItems, MyName)

                        if Tipo_analise != 'MESOESTRUTURA':
                            ret = SapModel.File.Save()



                        # 18 Alimentação dos dados iniciais das planilhas de dimensionamento
das longarinas
                        ws_long = wb_long.sheets["5 Esforços solicitantes"]

                        G_3 = 2*0.203*25/NLongarinasPorVão # Defensas
                        G_4    =    (0.1*(TotalWidthOfDeckSection/100    -    2*0.40)*24    +
2*(TotalWidthOfDeckSection/100 - 2*0.40))/NLongarinasPorVão # Pavimentação + recap
                        g3 = G_3 + G_4
                        ws_long.range('C28').value = g3
                        ws_long.range('C29').value                                          =
(TotalWidthOfDeckSection)/100*2/NLongarinasPorVão




                    # 19 Atribuição de carregamentos nos elementos
                        if Tipo_analise != 'SUPERESTRUTURA':
                            # 19.1 Pressão da água nos pilares
                            ret = SapModel.SelectObj.ClearSelection()
                            ret = SapModel.SelectObj.PropertyFrame(f'Pilar (C{f_ck})')
                            modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
                            wb_pil_ext = xw.Book(modelo_pil_ext)
                            ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

                            Name = '' # The name of an existing frame object or group, depending
on the value of the ItemType item.
                            LoadPat = 'Água'
                            MyType = 1
                            # This is 1 or 2, indicating the type of distributed load.
                            # 1 = Force per unit length
                            # 2 = Moment per unit length
                            Dir = 5
                            # This is an integer between 1 and 11, indicating the direction of
the load.
```

```
                            # 1 = Local 1 axis (only applies when CSys is Local)
                            # 2 = Local 2 axis (only applies when CSys is Local)
                            # 3 = Local 3 axis (only applies when CSys is Local)
                            # 4 = X direction (does not apply when CSys is Local)
                            # 5 = Y direction (does not apply when CSys is Local)
                            # 6 = Z direction (does not apply when CSys is Local)
                            # 7 = Projected X direction (does not apply when CSys is Local)
                            # 8 = Projected Y direction (does not apply when CSys is Local)
                            # 9 = Projected Z direction (does not apply when CSys is Local)
                            # 10 = Gravity direction (only applies when CSys is Global)
                            # 11 = Projected Gravity direction (only applies when CSys is
Global)
                            Dist1 = 0
                            # This is the distance from the I-End of the frame object to the
start of the distributed load.
                            # This may be a relative distance (0 <= Dist1 <= 1) or an actual
distance, depending on the value of the RelDist item. [L] when RelDist is False
                            Dist2 = 1
                            # This is the distance from the I-End of the frame object to the
end of the distributed load.
                            # This may be a relative distance (0 <= Dist2 <= 1) or an actual
distance, depending on the value of the RelDist item. [L] when RelDist is False
                            Val1 = ws_pil_ext['D57'].value/100 # This is the load value at the
start of the distributed load. [F/L] when MyType is 1 and [FL/L] when MyType is 2
                            Val2 = Val1 # This is the load value at the end of the distributed
load. [F/L] when MyType is 1 and [FL/L] when MyType is 2
                            CSys = 'GLOBAL' # This is Local or the name of a defined coordinate
system. It is the coordinate system in which the loads are specified.
                            RelDist = True # If this item is True, the specified Dist item is
a relative distance, otherwise it is an actual distance.
                            Replace = False # If this item is True, all previous distributed
loads, if any, assigned to the specified frame object(s), in the specified load pattern, are
deleted before making the new assignment.
                            ItemType = eItemType(2)
                            # This is one of the following items in the eItemType enumeration:
                            # Object = 0
                            # Group = 1
                            # SelectedObjects = 2
                            # If this item is Object, the assignment is made to the frame object
specified by the Name item.
                            # If this item is Group, the assignment is made to all frame objects
in the group specified by the Name item.
                            # If this item is SelectedObjects, assignment is made to all
selected frame objects, and the Name item is ignored.
                            ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType,
Dir, Dist1, Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)

                            ret = SapModel.SelectObj.ClearSelection()

                            # 19.2 Empuxo de terra nos pilares

                            ret = SapModel.SelectObj.ClearSelection()

                            XMin = -1 # The maximum and minimum coordinates of the selection
box in the specified coordinate system.
                            XMax = 1
                            YMin = -TotalWidthOfDeckSection
                            YMax = TotalWidthOfDeckSection
                            ZMin = -1000
                            ZMax = -400
                            DeSelect = False # The item is False if objects are to be selected
and True if they are to be deselected.
                            CSys = 'GLOBAL' # The name of the coordinate system in which XMin,
XMax, YMin, YMax, ZMin and ZMax are specified.
                            IncludeIntersections = True
                            # When this item is True, objects that are inside the box or
intersecting the sides and edges of the box are selected or deselected.
                            # When this item is False, only objects that are fully inside the
box are selected or deselected.
                            Point = False # Point objects that fall inside the box are only
selected or deselected when this item is True.
                            Line = True
                            Area = False
                            Solid = False
                            Link = False
```

```
                            ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax,
ZMin, ZMax, DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                            Name = ''
                            LoadPat = 'Empuxo terra'
                            MyType = 1
                            Dir = 4
                            Dist1 = 0
                            Dist2 = 1
                            Val1 = ws_pil_ext['D115'].value/100
                            Val2 = Val1
                            CSys = 'GLOBAL'
                            RelDist = True
                            Replace = False
                            ItemType = eItemType(2)
                            # This is one of the following items in the eItemType enumeration:
                            # Object = 0
                            # Group = 1
                            # SelectedObjects = 2
                            # If this item is Object, the assignment is made to the frame object
specified by the Name item.
                            # If this item is Group, the assignment is made to all frame objects
in the group specified by the Name item.
                            # If this item is SelectedObjects, assignment is made to all
selected frame objects, and the Name item is ignored.
                            ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType,
Dir, Dist1, Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)

                            ret = SapModel.SelectObj.ClearSelection()




                            # # 21 Análise do modelo (pré-dimensionamento da protensão)

                            # # 21.1 Load Cases to be Run

                            # Name = ''
                            # Run = True # If this item is True, the specified load case is to
be run.
                            # All = True # If this item is True, the run flag is set as specified
by the Run item for all load cases, and the Name item is ignored.
                            # ret = SapModel.Analyze.SetRunCaseFlag(Name, Run, All)

                        if Tipo_analise == "ENCONTROS":
                            ret                                                         =
MyBridgeModeler.BridgeObject.UpdateWithSameParameters(myBridgeObjName, 1)

                        ret    =    SapModel.File.Save(os.path.join(main_folder,    Tipologia,
f'{Tipologia} ({Tipo_analise}) PILAR {altura/100} m VAOS DIFERENTES {CVão/100}_{Vão/100}.bdb'))

                        ret = SapModel.Analyze.SetRunCaseFlag('', True, True)

                        ret = SapModel.Analyze.RunAnalysis()

                        ret = SapModel.File.Save()

                        wb_long.app.quit()
                        # ret = mySAPObject.ApplicationExit(False)
                        SapModel = None
                        mySAPObject = None
                        MyBridgeModeler = None
```

```
import os
import sys
import clr
import pandas as pd
from enum import Enum
import numpy as np
import xlwings as xw
import math

clr.AddReference("System.Runtime.InteropServices")
clr.AddReference(r'C:\Program Files\Computers and Structures\CSiBridge 26\CSiBridge1.dll')
from System.Runtime.InteropServices import Marshal
from CSiBridge1 import *

# Funções definidas

from utils import *

#set the following flag to True to execute on a remote computer

Remote = False


#if the above flag is True, set the following variable to the hostname of the remote computer

#remember that the remote computer must have CSiBridge installed and be running the CSiAPIService.exe

RemoteComputer = "SpareComputer-DT"



#set the following flag to True to attach to an existing instance of the program

#otherwise a new instance of the program will be started

AttachToInstance = True



#set the following flag to True to manually specify the path to CSiBridge.exe

#this allows for a connection to a version of CSiBridge other than the latest installation

#otherwise the latest installed version of CSiBridge will be launched

SpecifyPath = True

#if the above flag is set to True, specify the path to CSiBridge below

ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 26\CSiBridge.exe'

#full path to the model

#set it to the desired path of your model

#if executing remotely, ensure that this folder already exists on the remote computer

#the below command will only create the folder locally

main_folder = os.path.dirname(os.path.abspath(__file__))
APIPath = main_folder

# Planilhas

composite_template = os.path.join(main_folder, 'composite-template.xlsx')
wb = xw.Book(composite_template)
ws_dados = wb.sheets["1- Dados"]
ws_acoes = wb.sheets["2-Ações"]
ws_prop = wb.sheets["3-Prop Geo"]
ws_esforcos = wb.sheets["4-Esforços"]
ws_perfil = wb.sheets["5- VP-SCF"]
```

```
        ws_mista = wb.sheets["6- VSM-C"]
        ws_conectores = wb.sheets["7-Conectores"]
        ws_flecha = wb.sheets["8-ELS-Flecha"]
        ws_enrijecedores = wb.sheets["9-Enrijecedores"]
        ws_soldac = wb.sheets["10-Solda de composição"]
        ws_soldaf = wb.sheets["11-Soldas-fadiga"]
        ws_emendas = wb.sheets["12-Emendas"]
        ws_diafragmasapoio = wb.sheets["13-Diafragma apoio"]
        ws_diafragmasintermediarios = wb.sheets["14-Diafragma int"]
        ws_lajes = wb.sheets["17-Lajes"]

        if not os.path.exists(APIPath):

            try:

                os.makedirs(APIPath)

            except OSError:

                pass

        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'composite-variables.xlsx')
        variables    =    pd.read_excel(variables_data_file,    sheet_name="Main",    header=3,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-MAC-A-20-P', 'S-MAC-A-25-P', 'S-MAC-A-30-P', 'S-MAC-A-35-P', 'S-MAC-
A-20-S', 'S-MAC-A-25-S', 'S-MAC-A-30-S', 'S-MAC-A-35-S', 'S-MAC-B-20-P', 'S-MAC-B-25-P',
         'S-MAC-B-30-P', 'S-MAC-B-35-P', 'S-MAC-B-20-S', 'S-MAC-B-25-S', 'S-MAC-B-30-S', 'S-
MAC-B-35-S']

        # ['S-MAC-A-20-P', 'S-MAC-A-25-P', 'S-MAC-A-30-P', 'S-MAC-A-35-P', 'S-MAC-A-20-S', 'S-
MAC-A-25-S', 'S-MAC-A-30-S', 'S-MAC-A-35-S', 'S-MAC-B-20-P', 'S-MAC-B-25-P',
        # 'S-MAC-B-30-P', 'S-MAC-B-35-P', 'S-MAC-B-20-S', 'S-MAC-B-25-S', 'S-MAC-B-30-S', 'S-
MAC-B-35-S']

        Tipo_analise = 'SUPERESTRUTURA' # Tipos contemplados: SUPERESTRUTURA, ENCONTROS,
MESOESTRUTURA

        for Tipologia in Tipologias: # Rodar todo o código com base em um elemento da lista
Tipologias
            xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
            GeneralPath = os.path.join(main_folder, Tipologia)
            os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à tipologia. O
parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
            ModelPath = os.path.join(GeneralPath, f'{Tipologia} ({Tipo_analise}).bdb') #
Caminho do modelo .bdb

            #create API helper object

            helper = cHelper(Helper())

            if AttachToInstance:

                #attach to a running instance of CSiBridge

                try:

                    #get the active CSiBridge object

                    if Remote:

                        mySAPObject = cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                    else:

                        mySAPObject = cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                except:

                    print("No running instance of the program found or failed to attach.")

                    sys.exit(-1)
```

```
            else:
                if SpecifyPath:
                    try:
                        #'create an instance of the CSiBridge object from the specified path
                        if Remote:
                            mySAPObject     =     cOAPI(helper.CreateObjectHost(RemoteComputer,
ProgramPath))
                        else:
                            mySAPObject = cOAPI(helper.CreateObject(ProgramPath))
                    except :
                        print("Cannot start a new instance of the program from " + ProgramPath)
                        sys.exit(-1)
                else:

                    try:
                        #create an instance of the CSiBridge object from the latest installed
CSiBridge
                        if Remote:
                            mySAPObject = cOAPI(helper.CreateObjectProgIDHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))
                        else:
                            mySAPObject                                                    =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))

                    except:
                        print("Cannot start a new instance of the program.")
                        sys.exit(-1)

                #start CSiBridge application
                mySAPObject.ApplicationStart()

            #**********************************      INÍCIO      VARIÁVEIS      DE      INPUT
*****************************************#
            f_ck = 35
            Alturas_pilarescentrais = [400, 600, 800] # Análises de MESOESTRUTURA
            NVãos = 1 if Tipo_analise == 'SUPERESTRUTURA' else 3 if Tipo_analise == 'ENCONTROS'
else 4
            df_ponte = variables[(variables['Tipologia'] == Tipologia)]

            # SEÇÃO TRANSVERSAL
            CVão = df_ponte['Vão (cm)'].iloc[0]
            NVãos = 1
            CTotal = NVãos * CVão # Caso de pontes com múltiplos vãos
            TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
            NLongarinasPorVao = int(df_ponte['Nlong'].iloc[0])
            NumberInteriorGirders = NLongarinasPorVao - 2 #The number of interior girders
```

```
            OverhangLength = df_ponte['Balanço (cm)'].iloc[0] #balanço laje/longarina #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
            Espaçamento_Long = (TotalWidthOfDeckSection -
2*OverhangLength)/(NLongarinasPorVao-1)
            NDiafragmas = int(df_ponte['Ndiaf'].iloc[0])
            LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)

            # AÇO ESTRUTURAL A588Gr50
            fy_A588Gr50 = 345
            fu_A588Gr50 = 480
            fye_A588Gr50 = fy_A588Gr50/1.15
            fue_A588Gr50 = fye_A588Gr50/1.15
            df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia

            # DIMENSÕES DOS ELEMENTOS ESTRUTURAIS

                # PERFIL 1
            t3_1 = df_ponte['t3_1'].iloc[0] # The section depth
            t2_1 = df_ponte['t2_1'].iloc[0] # The top flange width
            tf_1 = df_ponte['tf_1'].iloc[0] # The top flange thickness
            tw_1 = df_ponte['tw_1'].iloc[0] # The web thickness
            t2b_1 = df_ponte['t2b_1'].iloc[0] # The bottom flange width
            tfb_1 = df_ponte['tfb_1'].iloc[0] # The bottom flange thickness
            FilletRadius_1 = df_ponte['rad_1'].iloc[0] # The fillet radius

                # PERFIL 2
            t3_2 = df_ponte['t3_2'].iloc[0] # The section depth
            t2_2 = df_ponte['t2_2'].iloc[0] # The top flange width
            tf_2 = df_ponte['tf_2'].iloc[0] # The top flange thickness
            tw_2 = df_ponte['tw_2'].iloc[0] # The web thickness
            t2b_2 = df_ponte['t2b_2'].iloc[0] # The bottom flange width
            tfb_2 = df_ponte['tfb_2'].iloc[0] # The bottom flange thickness
            FilletRadius_2 = df_ponte['rad_2'].iloc[0] # The fillet radius

                # DIAFRAGMAS EXTREMOS
            t3diaf_1 = df_ponte['t3diaf_1'].iloc[0] # The section depth
            t2diaf_1 = df_ponte['t2diaf_1'].iloc[0] # The top flange width
            tfdiaf_1 = df_ponte['tfdiaf_1'].iloc[0] # The top flange thickness
            twdiaf_1 = df_ponte['twdiaf_1'].iloc[0] # The web thickness
            t2bdiaf_1 = df_ponte['t2bdiaf_1'].iloc[0] # The bottom flange width
            tfbdiaf_1 = df_ponte['tfbdiaf_1'].iloc[0] # The bottom flange thickness
            FilletRadiusdiaf_1 = df_ponte['raddiaf_1'].iloc[0] # The fillet radius

                # DIAFRAGMAS INTERMEDIÁRIOS
            t3diaf_2 = df_ponte['t3diaf_2'].iloc[0] # The section depth
            t2diaf_2 = df_ponte['t2diaf_2'].iloc[0] # The top flange width
            tfdiaf_2 = df_ponte['tfdiaf_2'].iloc[0] # The top flange thickness
            twdiaf_2 = df_ponte['twdiaf_2'].iloc[0] # The web thickness
            t2bdiaf_2 = df_ponte['t2bdiaf_2'].iloc[0] # The bottom flange width
            tfbdiaf_2 = df_ponte['tfbdiaf_2'].iloc[0] # The bottom flange thickness
            FilletRadiusdiaf_2 = df_ponte['raddiaf_2'].iloc[0] # The fillet radius

                # PILARES
            t3_pilar = df_ponte['Dp'].iloc[0]

                # TRAVESSAS
            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]
            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

                # TABULEIRO

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - t2_1

                # APARELHO DE APOIO

            a_ap = df_ponte['aA'].iloc[0]
            b_ap = df_ponte['bA'].iloc[0]
            ShoreA_ap = 60 # Dureza Shore A do aparelho de apoio
```

```
            G_ap = 1000 # Módulo de elasticidade transversal do aparelho de apoio em kN/m²
            sigmaadm_ap = 15000 # Tensão admissível no aparelho de apoio
            hs_ap = 4 # Espessura das chapas de fretagem em mm
            hi_ap = 12 # Espessura das camadas intermediárias do Neoprene
            cv_ap = 2.5 # Cobrimento vertical
            ch_ap = 4 # Cobrimento horizontal
            n_ap = df_ponte['nA'].iloc[0] # Número de camadas do aparelho de apoio
            h_ap = (n_ap * hi_ap + 2* cv_ap + (n_ap+1)*hs_ap)/10 # Pré-dimensionamento da altura
do aparelho de apoio
            K_ap = a_ap/100 * b_ap/100 * G_ap/(cv_ap*2 + hi_ap*n_ap)*10 # Rigidez do aparelho
de apoio na direção longitudinal e transversal à longarina
            d_ap =  df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta

                # LAJE DE TRANSIÇÃO
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

                # MURO
            AlturaM = df_ponte['AlturaM'].iloc[0]

                # ALA
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

            # DEFINIÇÃO DE BENTS

            # EXTREMOS
            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = 300 if Tipo_analise == 'ENCONTROS' else df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                Distance_ext  =  [Distextp,  C_travessaext  -  Distextp]  #A  zero-based  one-
dimensional  array  dimensioned  to  (NumberColumns  -  1)  containing  the  distance  to  the  column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, C_travessaext/2, C_travessaext - Distextp]
                Height_ext = [Hextp, Hextp, Hextp]

            # INTERMEDIÁRIOS
            C_travessaint = df_ponte['Cint'].iloc[0]
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
            SupportType_int = 1
            if N_pilaresint == 2:
                Distance_int  =  [Distintp,  C_travessaint  -  Distintp]  #A  zero-based  one-
dimensional  array  dimensioned  to  (NumberColumns  -  1)  containing  the  distance  to  the  column
centerline measured from the left end of the cap beam. [L]
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # ASPECTOS DO PROJETO
            e_a = 40 # Distância entre o início do perfil da longarina e o centro do apoio
            CEfetivo = CVão - 2*e_a # Vão efetivo da ponte
            NTrechos = 3 # Número de perfis para vencer um vão

                # PERFIS
            NPerfil1 = NTrechos if NTrechos <= 2 else 1 if NTrechos == 3 else 2
            NPerfil2 = 0 if NTrechos <= 2 else 2
            CPerfil1  =  CVão  if  NTrechos  ==  1  else  CVão/NTrechos  if  NTrechos  ==  2  and
CVão/NTrechos <= 1190 else 1190
```

```
            NEmendas = math.floor(CVão/1190)
            ejunta = 5
            CPerfil2 = (CVão - CPerfil1)/NPerfil2
            x_S1 = 0 if NTrechos == 1 else CPerfil1 if NTrechos == 2 else CPerfil2 # Posição
da Emenda 1
            x_S2 = 0 if NTrechos <= 2 else CVão - CPerfil2 if NTrechos == 3 else CVão/2 #
Posição da Emenda 2

                # DIAFRAGMAS

            eet = 1.25 # Espessura dos enrijecedores transversais
            let = 15 # Largura dos enrijecedores transversais
            eea = 1.6 # Espessura dos enrijecedores de apoio
            lea = 20 # Largura dos enrijecedores de apoio

                # DIAFRAGMAS
            DistDiafragmas = CEfetivo/(NDiafragmas-1)
            x_Diafragmas = [e_a]
            x_Acumulado = e_a
            while x_Acumulado <= CEfetivo:
                x_Acumulado += DistDiafragmas
                x_Diafragmas.append(x_Acumulado)

            SpanLength = [] # A zero-based array containing the length of each span. [L]
            SpanName = []
            for i in range(NVãos):
                if NTrechos <= 2:
                    SpanLength.extend([CPerfil1, CPerfil1]) #A zero-based array containing the
length of each span. [L]
                    SpanName.extend([f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'])
                elif NTrechos == 3:
                    SpanLength.extend([CPerfil2, CPerfil1, CPerfil2]) #A zero-based array
containing the length of each span. [L]
                    SpanName.extend([f'Perfil 2.{i+1}.1', f'Perfil 1.{i+1}', f'Perfil
2.{i+1}.2'])
                else:
                    SpanLength.extend([CPerfil2, CPerfil1, CPerfil1, CPerfil2]) #A zero-based
array containing the length of each span. [L]
                    SpanName.extend([f'Perfil 2.{i+1}.1', f'Perfil 1.{i+1}.1', f'Perfil
1.{i+1}.2', f'Perfil 2.{i+1}.2'])

            SpanStartStations = [0]
            SpanAcumulado = 0
            for i in range(len(SpanName)-1):
                SpanAcumulado += SpanLength[i]
                SpanStartStations.append(SpanAcumulado)
            SpanStartStations.append(CVão) # Adicionar o fim

            DiaphragmDistanceSpan = []
            for i in range (len(SpanStartStations)-1):
                DiaphragmDistance = []
                for j in range(len(x_Diafragmas)):
                    if x_Diafragmas[j] < SpanStartStations[i+1] and x_Diafragmas[j] -
SpanStartStations[i] > 0:
                        DiaphragmDistance.append(x_Diafragmas[j] - SpanStartStations[i])
                DiaphragmDistanceSpan.append(DiaphragmDistance) # A intenção é criar uma lista
de listas, com cada lista referente aos valores de distância para cada vão

            # Alimentação das planilhas 2-Ações e 3-Prop Geo
            ws_dados['H108'].value = df_ponte['Classe'].iloc[0]
            ws_dados['J109'].value = CVão/100
            ws_dados['J110'].value = df_ponte['nf'].iloc[0]
            ws_dados['J111'].value = df_ponte['fr'].iloc[0]/100
            ws_dados['J112'].value = df_ponte['ace'].iloc[0]/100
            ws_dados['J113'].value = df_ponte['aci'].iloc[0]/100
            ws_dados['J114'].value = df_ponte['qp'].iloc[0]
            ws_dados['J115'].value = df_ponte['Lp'].iloc[0]/100
            ws_dados['J116'].value = NLongarinasPorVao
            ws_dados['J117'].value = df_ponte['GC'].iloc[0]/100
            ws_dados['J118'].value = Espaçamento_Long/100
            ws_dados['J119'].value = TotalWidthOfDeckSection/100
            ws_dados['J120'].value = OverhangLength/100
            ws_dados['J122'].value = h_laje/100
            ws_dados['J128'].value = h_prelaj/100
            ws_dados['I148'].value = e_a/100
```

```
            ws_dados['H163'].value = NTrechos
            ws_dados['H164'].value = NPerfil1
            ws_dados['H165'].value = NPerfil2
            ws_dados['H166'].value = CPerfil1/100
            ws_dados['H167'].value = CPerfil2/100
            ws_dados['I169'].value = NEmendas
            ws_dados['I170'].value = x_S1/100
            ws_dados['I172'].value = x_S2/100
            ws_dados['H186'].value = NDiafragmas
            ws_dados['H187'].value = DistDiafragmas*10

            # Aparelho de apoio
            ws_dados['I347'].value = a_ap*10
            ws_dados['I348'].value = b_ap*10
            ws_dados['I349'].value = h_ap*10
            ws_dados['I350'].value = hi_ap
            ws_dados['I351'].value = n_ap
            ws_dados['I352'].value = hs_ap
            ws_dados['I353'].value = n_ap + 1
            ws_dados['I354'].value = n_ap * hi_ap

            # Propriedades geométricas
            ws_prop['H26'].value = tw_1*10
            ws_prop['H27'].value = t3_1*10
            ws_prop['H28'].value = tf_1*10
            ws_prop['H29'].value = tfb_1*10
            ws_prop['H30'].value = t2_1*10
            ws_prop['H31'].value = t2b_1*10

            ws_prop['I26'].value = tw_2*10
            ws_prop['I27'].value = t3_2*10
            ws_prop['I28'].value = tf_2*10
            ws_prop['I29'].value = tfb_2*10
            ws_prop['I30'].value = t2_2*10
            ws_prop['I31'].value = t2b_2*10

            #**********************************       FIM       VARIÁVEIS       DE       INPUT
*****************************************#

            # 1 Definição dos objetos primitivos
            SapModel = cSapModel(mySAPObject.SapModel)
            MyBridgeModeler  =  cBridgeModeler_1(SapModel.BridgeModeler_1)  #  necessário  para
acessar os métodos do CSiBridge
            ret = SapModel.File.OpenFile(os.path.join(main_folder, 'composite-template.bdb'))
            ret   =   SapModel.File.Save(os.path.join(main_folder,   Tipologia,   f'{Tipologia}
({Tipo_analise}).bdb'))

            kN_cm_C = 13
            kN_m_C = 6
            kN_mm_C = 5
            ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

            # 2 Propriedades dos materiais

            # 2.1 Definições iniciais

            PropMaterial = cPropMaterial(SapModel.PropMaterial)

            #Criação de materiais do tipo concreto e aço
            MATERIAL_CONCRETE = 2
            MATERIAL_STEEL = 1
            ret  =  SapModel.PropMaterial.ChangeName('Concreto  -  Mesoestrutura',  f'C{f_ck}  -
Mesoestrutura')
            ret   =   SapModel.PropMaterial.ChangeName('Concreto   -   Tabuleiro',   f'C{f_ck}   -
Tabuleiro')
            ret = SapModel.PropMaterial.SetMaterial('A588Gr50', eMatType(MATERIAL_STEEL))

            # 2.2 Atribuição das propriedades mecânicas elásticas aos materiais
            alfa_i = min(0.8 + 0.2*f_ck/80, 1)
            E  =  alfa_i*(5600*f_ck**(1/2))/10  if  f_ck  <=  50  else  (21.5*10^3*(f_ck/10  +
1,25)^(1/3))/10  # Considerando o uso de granito e gnaisse
            E_aco = 21000
            v = 0.2
            v_aco = 0.3
            alfa_T = 10**(-5)
```

```
            T_ambiente = 25
            ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Mesoestrutura', E, v,
alfa_T, T_ambiente)
            ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Tabuleiro', E, v, alfa_T,
T_ambiente)
            ret = SapModel.PropMaterial.SetMPIsotropic('A588Gr50', E_aco, v_aco, alfa_T,
T_ambiente)

            # 2.3 Atribuição do peso específico aos materiais
            γ_concreto = 25/(100**3)
            γ_aco = 78.5/(100**3)
            ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Mesoestrutura', 1,
γ_concreto, T_ambiente)
            ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Tabuleiro', 1, γ_concreto,
T_ambiente)
            ret = SapModel.PropMaterial.SetWeightAndMass('A588Gr50', 1, γ_aco, T_ambiente)

            # 2.4 Atribuição das resistência aos materiais e curvas características de tensão-
deformação
            IsLightweight = False
            fcsfactor = 1 #The shear strength reduction factor for lightweight concrete
            SSType = 0
            # This is 0, 1 or 2, indicating the stress-strain curve type.
            # 0 = User defined
            # 1 = Parametric - Simple
            # 2 = Parametric - Mander
            SSHysType = 0
            # This is 0, 1 or 2, indicating the stress-strain hysteresis type.
            # 0 = Elastic
            # 1 = Kinematic
            # 2 = Takeda
            StrainAtfc = 0.002#This item applies only to parametric stress-strain curves. It
is the strain at the unconfined compressive strength.
            StrainUltimate = 0.0035 #This item applies only to parametric stress-strain curves.
It is the ultimate unconfined strain capacity. This item must be larger than the StrainAtfc
item.
            FinalSlope = 1 #This item applies only to parametric stress-strain curves. It is a
multiplier on the material modulus of elasticity, E. This value multiplied times E gives the
final slope on the compression side of the curve.
            ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Mesoestrutura',
f_ck/10/1.4, IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate,
FinalSlope)
            ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Tabuleiro', f_ck/10/1.4,
IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate, FinalSlope)

            SSType = 1
            # This is 0 or 1, indicating the stress-strain curve type.
            # 0 = User defined
            # 1 = Parametric - Simple
            SSHysType = 1
            # This is 0, 1 or 2, indicating the stress-strain hysteresis type.
            # 0 = Elastic
            # 1 = Kinematic
            # 2 = Takeda
            StrainAtHardening = 0.015 # This item applies only to parametric stress-strain
curves. It is the strain at the onset of strain hardening.
            StrainAtMaxStress = 0.11 # This item applies only to parametric stress-strain
curves. It is the strain at maximum stress. This item must be larger than the StrainAtHardening
item.
            StrainAtRupture = 0.17 # This item applies only to parametric stress-strain curves.
It is the strain at rupture. This item must be larger than the StrainAtMaxStress item.
            FinalSlope = -0.1 # This item applies only to parametric stress-strain curves. It
is a multiplier on the material modulus of elasticity, E. This value multiplied times E gives
the final slope of the curve.
            # This item is the temperature at which the specified data applies. The temperature
must have been defined previously for the material.
            ret = SapModel.PropMaterial.SetOSteel_1('A588Gr50', fy_A588Gr50/10,
fu_A588Gr50/10, fye_A588Gr50/10, fue_A588Gr50/10, SSType, SSHysType, StrainAtHardening,
StrainAtMaxStress, StrainAtRupture, FinalSlope, T_ambiente)



            # 3 Definição das propriedades das seções
```

```
            # 3.1 Frame Sections

            # 3.1.1 Longarinas
            ret = SapModel.PropFrame.SetISection_1('Perfil 1', 'A588Gr50', t3_1, t2_1, tf_1,
tw_1, t2b_1, tfb_1, FilletRadius_1, -1, '', 'Default')
            ret = SapModel.PropFrame.SetISection_1('Perfil 2', 'A588Gr50', t3_2, t2_2, tf_2,
tw_2, t2b_2, tfb_2, FilletRadius_2, -1, '', 'Default')
            ret = SapModel.PropFrame.SetISection_1('Diafragma extremo', 'A588Gr50', t3diaf_1,
t2diaf_1, tfdiaf_1, twdiaf_1, t2bdiaf_1, tfbdiaf_1, FilletRadiusdiaf_1, -1, '', 'Default')
            ret = SapModel.PropFrame.SetISection_1('Diafragma intermediário', 'A588Gr50',
t3diaf_2, t2diaf_2, tfdiaf_2, twdiaf_2, t2bdiaf_2, tfbdiaf_2, FilletRadiusdiaf_2, -1, '',
'Default')
            h_long = t3_1

            # 3.1.2 Pilares
            Name = f'Pilar (C{f_ck})'
            MatProp = f'C{f_ck} - Mesoestrutura'
            Color = -1
            Notes = ''
            GUID = 'Default'
            ret = SapModel.PropFrame.SetCircle(Name, MatProp, t3_pilar, Color, Notes, GUID) #
definição da seção

            Name = f'Pilar (C{f_ck})' # The name of an existing frame section property.
            MatPropLong = 'A615Gr60' # The name of the rebar material property for the
longitudinal rebar.
            MatPropConfine = 'A615Gr60' # The name of the rebar material property for the
confinement rebar.
            Pattern = 2
            # This is either 1 or 2, indicating the rebar configuration.
            # 1 = Rectangular
            # 2 = Circular
            # For circular frame section properties this item must be 2; otherwise an error is
returned.
            ConfineType = 1
            # This is either 1 or 2, indicating the confinement bar type.
            # 1 = Ties
            # 2 = Spiral
            # This item applies only when Pattern = 2. If Pattern = 1, the confinement bar type
is assumed to be ties.
            Cover = 4 # The clear cover for the confinement steel (ties). In the special case
of circular reinforcement in a rectangular column, this is the minimum clear cover. [L]
            NumberCBars = 14 # This item applies to a circular rebar configuration, Pattern =
2. It is the total number of longitudinal reinforcing bars in the column.
            NumberR3Bars = int() # This item applies to a rectangular rebar configuration,
Pattern = 1. It is the number of longitudinal bars (including the corner bar) on each face of
the column that is parallel to the local 3-axis of the column.
            NumberR2Bars = int() # This item applies to a rectangular rebar configuration,
Pattern = 1. It is the number of longitudinal bars (including the corner bar) on each face of
the column that is parallel to the local 2-axis of the column.
            RebarSize = '25d' # The rebar name for the longitudinal rebar in the column.
            TieSize = '8d' # The rebar name for the confinement rebar in the column.
            TieSpacingLongit = 20 # The longitudinal spacing of the confinement bars (ties).
[L]
            Number2DirTieBars = int() # This item applies to a rectangular reinforcing
configuration, Pattern = 1. It is the number of confinement bars (tie legs) running in the local
2-axis direction of the column.
            Number3DirTieBars = int() # This item applies to a rectangular reinforcing
configuration, Pattern = 1. It is the number of confinement bars (tie legs) running in the local
3-axis direction of the column.
            ToBeDesigned = False # If this item is True, the column longitudinal rebar is to
be designed; otherwise it is to be checked.
            ret = SapModel.PropFrame.SetRebarColumn(Name, MatPropLong, MatPropConfine,
Pattern, ConfineType, Cover, NumberCBars, NumberR3Bars, NumberR2Bars, RebarSize, TieSize,
            TieSpacingLongit, Number2DirTieBars, Number3DirTieBars, ToBeDesigned)

            # 3.1.3 Travessas
            ret = SapModel.PropFrame.SetRectangle(f'Travessa encontro (C{f_ck})', f'C{f_ck} -
Mesoestrutura', h_travessaext, bw_travessaext)
            ret = SapModel.PropFrame.SetRectangle(f'Travessa intermediária (C{f_ck})',
f'C{f_ck} - Mesoestrutura', h_travessaint, bw_travessaint)

            # 3.2 Area Sections
```

```
            Name = 'Encontro'
            ShellType = 1
            IncludeDrillingDOF = True
            MatProp = f'C{f_ck} - Mesoestrutura'
            MatAng = 0
            Thickness = hAla
            Bending = hAla
            Color = -1
            Notes = ''
            GUID = 'Default'
            ret = SapModel.PropArea.SetShell_1(Name, ShellType, IncludeDrillingDOF, MatProp,
MatAng, Thickness, Bending, Color, Notes, GUID)

            Name = 'Laje de transição'
            ShellType = 1
            IncludeDrillingDOF = True
            MatProp = f'C{f_ck} - Tabuleiro'
            MatAng = 0
            Thickness = EspessuraLT
            Bending = EspessuraLT
            Color = -1
            Notes = ''
            GUID = 'Default'
            ret = SapModel.PropArea.SetShell_1(Name, ShellType, IncludeDrillingDOF, MatProp,
MatAng, Thickness, Bending, Color, Notes, GUID)




            # 4 Carregamentos

            # 4.1 Load Patterns/Cases
            # Wearing surface e sidewalk são carregamentos a serem definidos especificamente
no módulo do CSiBridge (serão definidos posteriormente)
            # MyType
            LTYPE_DEAD = 1
            LTYPE_LIVE = 3
            LTYPE_WIND= 6
            LTYPE_OTHER = 8
            LTYPE_TEMPERATURE = 10
            LTYPE_BRAKING = 15
            LTYPE_HORIZONTALEARTHPRESSURE = 20
            LTYPE_TEMPERATUREGRADIENT = 26
            LTYPE_WATERLOADPRESSURE = 30
            LTYPE_IMPACT = 38
            LTYPE_CONSTRUCTION = 39
            LTYPE_DEADWEARING = 40
            SelfWTMultiplier = 0 #The self weight multiplier for the new load pattern
            AddLoadCase = True #Associa o load pattern a uma load case
            ret    =    SapModel.LoadPatterns.Add('Defensas',    eLoadPatternType(LTYPE_DEAD),
SelfWTMultiplier, AddLoadCase)
            ret                  =                 SapModel.LoadPatterns.Add('Pavimentação',
eLoadPatternType(LTYPE_DEADWEARING), SelfWTMultiplier, AddLoadCase)
            ret                  =                 SapModel.LoadPatterns.Add('Recapeamento',
eLoadPatternType(LTYPE_DEADWEARING), SelfWTMultiplier, AddLoadCase)
            ret     =     SapModel.LoadPatterns.Add('Vento',     eLoadPatternType(LTYPE_WIND),
SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       uniforme       pos',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       uniforme       neg',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
            ret        =       SapModel.LoadPatterns.Add('Frenagem        e        aceleracao',
eLoadPatternType(LTYPE_BRAKING), SelfWTMultiplier, AddLoadCase)
            ret             =             SapModel.LoadPatterns.Add('Empuxo             terra',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
            ret      =      SapModel.LoadPatterns.Add('Empuxo      terra      carga      móvel',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       gradiente       pos',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
            ret       =       SapModel.LoadPatterns.Add('Temperatura       gradiente       neg',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
            ret = SapModel.LoadPatterns.Add('Água', eLoadPatternType(LTYPE_WATERLOADPRESSURE),
SelfWTMultiplier, AddLoadCase)
```

```
            ret    =    SapModel.LoadPatterns.Add('Impacto',    eLoadPatternType(LTYPE_IMPACT),
SelfWTMultiplier, AddLoadCase)
            ret    =    SapModel.LoadPatterns.Add('Haunch',    eLoadPatternType(LTYPE_DEAD),
SelfWTMultiplier, AddLoadCase)
            ret    =    SapModel.LoadPatterns.Add('SIPForm',    eLoadPatternType(LTYPE_DEAD),
SelfWTMultiplier, AddLoadCase)
            #       ret       =       SapModel.LoadPatterns.Add('ConcPour       Permanent',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
            #       ret       =       SapModel.LoadPatterns.Add('ConcPour       Temporary',
eLoadPatternType(LTYPE_CONSTRUCTION), SelfWTMultiplier, AddLoadCase)

            # 4.2 Load Distributions

            # 4.2.1 Gradiente de temperatura
            ret   =   MyBridgeModeler.LoadDistribution.TemperatureGradient.AddNew('Temperatura
gradiente')

            Name = 'Temperatura gradiente' #The name of an existing bridge temperature gradient
load distribution.
            NumberDistances = 4 #The number of specified distances (between 1 and 5, inclusive).
            DistanceThatMayVary = 3 #The specified distance that may vary.
            TempDifferencesToConsider = 0
            # The temperature differences to consider.
            # 0 = Positive and Negative
            # 1 = Positive Only
            # 2 = Negative Only
            d_1 = min(0.3*(h_laje + h_long), 15)
            d_2 = min(max(0.3*(h_laje + h_long), 10), 25)
            d_3 = min(min(0.3*(h_laje + h_long), 10 + 10), (h_laje + h_long)-d_1-d_2)
            if (h_laje + h_long) < 20:
                ΔT_1 = 8.5
                ΔT_2 = 3.5
                ΔT_3 = 0.5
            elif 20 < (h_laje + h_long) <= 40:
                ΔT_1 = np.interp((h_laje + h_long), [20, 40], [8.5, 12])
                ΔT_2 = np.interp((h_laje + h_long), [20, 40], [3.5, 3])
                ΔT_3 = np.interp((h_laje + h_long), [20, 40], [0.5, 1.5])
            elif 40 < (h_laje + h_long) <= 60:
                ΔT_1 = np.interp((h_laje + h_long), [40, 60], [12, 13])
                ΔT_2 = np.interp((h_laje + h_long), [40, 60], [3, 3])
                ΔT_3 = np.interp((h_laje + h_long), [40, 60], [1.5, 2])
            elif 60 < (h_laje + h_long) < 80:
                ΔT_1 = np.interp((h_laje + h_long), [60, 80], [13, 13.5])
                ΔT_2 = np.interp((h_laje + h_long), [60, 80], [3, 3])
                ΔT_3 = np.interp((h_laje + h_long), [60, 80], [2, 2.5])
            else:
                ΔT_1 = 13.5
                ΔT_2 = 3.0
                ΔT_3 = 2.5
            d = [d_1, d_2, float(), d_3] #A zero-based one-dimensional array dimensioned to
(NumberDistances - 1) containing the specified distances. [L]
            TPos = [ΔT_1, ΔT_2, 0, 0, ΔT_3] #A zero-based one-dimensional array dimensioned to
(NumberDistances) containing the specified positive temperatures. [T]
            TNeg = [-5, -5, 0, 0, 0] #A zero-based one-dimensional array dimensioned to
(NumberDistances) containing the specified negative temperatures. [T]
            ret                                                                          =
MyBridgeModeler.LoadDistribution.TemperatureGradient.SetUserGeneralData(Name, NumberDistances,
DistanceThatMayVary, TempDifferencesToConsider, d, TPos, TNeg)

            # 4.2.2 Vento

            # Força distribuída linearmente
            Name = 'Vento força'
            ret = MyBridgeModeler.LoadDistribution.Line.AddNew(Name)

            Name = Name # The name of an existing bridge line load distribution
            LoadType = 1 # The line load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 5
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
```

```
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LoadValue = ws_acoes['J309'].value/100 # The load value. [F/L] for force; [F] for
moment
            ReferenceLocation = 2
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LoadDistance = 0 # The distance from the reference location to the load. [L]
            ret  =  MyBridgeModeler.LoadDistribution.Line.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LoadValue, ReferenceLocation, LoadDistance)

            # Momento distribuído linearmente
            Name = 'Vento momento'
            ret = MyBridgeModeler.LoadDistribution.Line.AddNew(Name)

            Name = Name
            LoadType = 2 # The line load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 4
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LoadValue = -ws_acoes['J310'].value # The load value. [F/L] for force; [F] for
moment
            ReferenceLocation = 2
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LoadDistance = 0 # The distance from the reference location to the load. [L]
            ret  =  MyBridgeModeler.LoadDistribution.Line.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LoadValue, ReferenceLocation, LoadDistance)

            # 4.2.3 Frenagem e aceleração
            Name = 'Frenagem e aceleracao'
            ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

            Name = Name # The name of an existing bridge area load distribution.
            LoadType = 1 # The area load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 4
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LeftEdgeValue = 0.32E-04 # The left edge load value. [F/L2] for force; [F/L] for
moment
            RightEdgeValue = 0.32E-04 # The right edge load value. [F/L2] for force; [F/L] for
moment
            LeftReferenceLocation = 1 #The left reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LeftLoadDistance = 0 # The distance from the left edge of load to the left reference
location. [L]
            RightReferenceLocation = 2 # The right reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            RightLoadDistance = 0 # The distance from the right edge of load to the right
reference location. [L]
```

```
            ret  =  MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
            LeftReferenceLocation,          LeftLoadDistance,          RightReferenceLocation,
RightLoadDistance)

            # 4.2.4 Pavimentação
            Name = 'Pavimentação'
            ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

            Name = Name # The name of an existing bridge area load distribution.
            LoadType = 1 # The area load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 10
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LeftEdgeValue = 2.4E-04 # The left edge load value. [F/L2] for force; [F/L] for
moment
            RightEdgeValue = 2.4E-04 # The right edge load value. [F/L2] for force; [F/L] for
moment
            LeftReferenceLocation = 1 #The left reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LeftLoadDistance = 40 # The distance from the left edge of load to the left reference
location. [L]
            RightReferenceLocation = 2 # The right reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            RightLoadDistance = 40 # The distance from the right edge of load to the right
reference location. [L]
            ret  =  MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
            LeftReferenceLocation,          LeftLoadDistance,          RightReferenceLocation,
RightLoadDistance)

            # 4.2.5 Recapeamento
            Name = 'Recapeamento'
            ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

            Name = Name # The name of an existing bridge area load distribution.
            LoadType = 1 # The area load type
            # 1 = Force
            # 2 = Moment
            CoordSystem = 'GLOBAL' # This is either Local or the name of a defined coordinate
system.
            Direction = 10
            # The load direction.
            # 1 = 1; 2 = 2; 3 = 3
            # 4 = X; 5 = Y; 6 = Z
            # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
            # 10 = Gravity; 11 = Gravity Projected
            # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
            # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem is Local.
            # Directions 10 and 11 only apply when the CoordSystem is Global.
            LeftEdgeValue = 2E-04 # The left edge load value. [F/L2] for force; [F/L] for moment
            RightEdgeValue = 2E-04 # The right edge load value. [F/L2] for force; [F/L] for
moment
            LeftReferenceLocation = 1 #The left reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            LeftLoadDistance = 40 # The distance from the left edge of load to the left reference
location. [L]
            RightReferenceLocation = 2 # The right reference location.
            # 1 = Left edge of deck
            # 2 = Right edge of deck
            RightLoadDistance = 40 # The distance from the right edge of load to the right
reference location. [L]
```

```
            ret  =  MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,  LoadType,
CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
            LeftReferenceLocation,          LeftLoadDistance,          RightReferenceLocation,
RightLoadDistance)



            # 5 Criação do modelo da ponte - (cBridgeModeler_1.NewBridgeModel)

            StartStation = 0 #The starting station for the bridge. [L]
            InitialBearing = "N900000E" #The initial bearing for the bridge. This should be a
valid bearing angle such as "S624355E".
            Grade = 0 #The grade for the bridge specified in percent. A positive grade represents
an incline with increasing station.
            Skew = "Default"
            #The skew angle for the bridge. This should be a valid bearing angle such as
"S624355E", "Default" or "DefaultX"
            #where X is a positive or negative number indicating the number of degrees the
bearing is different from the default skew.
            #The default skew is perpendicular to the reference line.
            Radius = 0 #The bridge radius. A positive radius curves to the right and a negative
radius curves to the left. A radius of zero means an infinite radius (i.e., straight). [L]
            SuperElevation = 0 #The bridge superelevation specified in percent. A positive
value rotates clockwise corresponding to a curve right and a negative value rotates
counterclockwise corresponding to a curve left.
            SectionType = 9 # Diferença na numeração do pc do lab e o daqui de casa?
            #This is an integer between 2 and 16, inclusive, indicating the bridge section
type.
            # 2 = Concrete Box Girder - Exterior Girders Vertical
            # 3 = Concrete Box Girder - Exterior Girders Sloped
            # 4 = Concrete Box Girder - Exterior Girders Clipped
            # 5 = Concrete Box Girder - Exterior Girders with Radius
            # 6 = Concrete Box Girder - Exterior Girders Sloped Maximum
            # 7 = Concrete Tee Beam
            # 8 = Concrete Flat Slab
            # 9 = Steel I Girder
            # 10 = Concrete Box Girder - AASHTO-PCI-ASBI Standard
            # 11 = Precast Concrete I Girder
            # 12 = Precast Concrete U Girder
            # 13 = Concrete Box Girder - Advanced
            # 14 = Steel U Girder
            # 15 = Concrete Solid Girder
            # 16 = Precast Concreter Super-T Girder
            ret  =  MyBridgeModeler.NewBridgeModel(StartStation,  InitialBearing,  Grade,  Skew,
Radius, SuperElevation, SpanLength, SectionType)

            # 5.1 Objetos de interesse
            NumberNames = int()
            MyName = []
            ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
            myBridgeObjName = list(ret[2])[0]

            NumberNames = int()
            MyName = []
            BridgeSectionType = [SectionType]
            ret      =      MyBridgeModeler.DeckSection.GetNameList(NumberNames,      MyName,
BridgeSectionType)
            myDeckSectionName_1 = list(ret[2])[0]
            ret  =  MyBridgeModeler.DeckSection.ChangeName(myDeckSectionName_1,  'SeçãoMista-
Perfil1')
            myDeckSectionName_1 = 'SeçãoMista-Perfil1'

            Name = myBridgeObjName
            LayoutLine = ''
            ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
            myLayoutLineName = ret[1]

            ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
            myLaneName = list(ret[2])[0]

            # 5.2 Definição do Layout Line - (cBMLayoutLine.SetGeneralData)
            Name = myLayoutLineName
            CoordinateSystem = 'GLOBAL'
```

```
            StartX = 0
            StartY = 0
            StartZ = 0
            ret = MyBridgeModeler.LayoutLine.SetGeneralData(Name, CoordinateSystem, StartX,
StartY, StartZ)

            # 5.3 Bridge Object Data (cBMBridgeObject.SetSpanLayoutByLength Method)
            Name = myBridgeObjName
            StartStation = 0
            ret = MyBridgeModeler.BridgeObject.SetSpanLayoutByLength(Name, StartStation,
SpanName, SpanLength)

            # 5.4 Veiculos (cBMVehicle Methods)
            Region = 'User'
            Standard = 'User'
            Vehicle = 'All'
            ret = MyBridgeModeler.Vehicle.ImportVehicle(Region, Standard, Vehicle)
            ret = MyBridgeModeler.Vehicle.ConvertToUser('Multidao')

            # Correção (carregamento de Multidao que depende da largura do tabuleiro)

            Name = 'Multidao' # The name of the vehicle.
            UniformLoadSF = 1 # The uniform load scale factor.
            AxleLoadSF = 1 # The axle load scale factor.
            NumberLoadDataSets = 2 # The number of load data sets defined for the vehicle.
            LoadLengthType = [1, 2]
            # A zero-based one-dimensional array containing the load length type for each load
data set.
            # 1 = Leading Load
            # 2 = Trailing Load
            # 3 = Fixed Length
            # 4 = Variable Length
            MinimumDistance = [0, 0] # A zero-based one-dimensional array containing the minimum
distance between the current and previous axle. This item is only applicable if the
LoadLengthType is 3 or 4 (Fixed Length or Variable Length). [L]
            MaximumDistance = [float(), float()] # A zero-based one-dimensional array
containing the maximum distance between the current and previous axle. This item is only
applicable if the LoadLengthType is 4 (Variable Length). This item must either be zero (meaning
infinity) or it must be greater than or equal to MinimumDistance item. [L]
            UniformLoad = [(LarguraRolamento)/100*(5/100), (LarguraRolamento)/100*(5/100)] # A
zero-based one-dimensional array containing the uniform load. [F/L]
            UniformWidthType = [2, 2]
            # A zero-based one-dimensional array containing the width type for the uniform
load.
            # 1 = Zero Width
            # 2 = Lane Width
            # 3 = Fixed Width
            UniformWidth = [float(), float()] # A zero-based one-dimensional array containing
the uniform load width. This item is only applicable if the UnifWidthType is 3 (Fixed Width).
[L]
            AxleLoad = [0, float()] # A zero-based one-dimensional array containing the total
load for the axle. This item does not apply when the LoadLengthType item is 4 (Variable Length).
[F]
            AxleWidthType = [1, int()]
            # A zero-based one-dimensional array containing the axle width type indicating how
the axle load is to be distributed. This item does not apply when the LoadLengthType item is 4
(Variable Length).
            # 1 = One Point
            # 2 = Two Points
            # 3 = Four Points
            # 4 = Lane Width Line
            # 5 = Fixed Width Line
            AxleWidth = [float(), float()] # A zero-based one-dimensional array containing the
axle width. This item is only applicable if the AxleWidthType item is 2, 3 or 5 (Two Points,
Four Points or Fixed Width Line).
            # This item does not apply when the LoadLengthType item is 4 (Variable Length). [L]
            ret = MyBridgeModeler.Vehicle.SetVerticalLoads(Name, UniformLoadSF, AxleLoadSF,
NumberLoadDataSets, LoadLengthType, MinimumDistance, MaximumDistance, UniformLoad,
UniformWidthType, UniformWidth, AxleLoad, AxleWidthType, AxleWidth)

            ret = SapModel.LoadCases.Delete('MOVE1')
            ret = SapModel.LoadCases.Moving.SetCase('Veiculo')
            ret = SapModel.LoadCases.Moving.SetCase('Multidao')

            ret = SapModel.LoadCases.Moving.SetInitialCase('Veiculo', '')
```

```
            ret = SapModel.LoadCases.Moving.SetInitialCase('Multidao', '')

            NumberLoads = 1 #The number of loads assigned to the specified analysis case.
            MyClass = ['Veiculo'] #This is an array that includes the vehicle class for each
load assigned to the load case.
            SF = [1] #This is an array that includes the scale factor for each load assigned
to the load case.
            Min = [0] #This is an array that includes the minimum number of lanes loaded for
each load assigned to the load case.
            Max = [0] #This is an array that includes the maximum number of lanes loaded for
each load assigned to the load case. This item must be 0, or it must be greater than or equal
to Min. If this item is 0, all available lanes are loaded.
            ret = SapModel.LoadCases.Moving.SetLoads('Veiculo', NumberLoads, MyClass, SF, Min,
Max)
            MyClass = ['Multidao']
            ret = SapModel.LoadCases.Moving.SetLoads('Multidao', NumberLoads, MyClass, SF, Min,
Max)

            LoadNumber = 1 #The load assignment number.
            NumberItems = 1 #The number of lanes loaded for the specified load assignment
number.
            MyName = [f'{myLaneName}'] #This is an array that includes the name of each lane
loaded for the specified load assignment number.
            ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber, NumberItems,
MyName)
            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Multidao',     LoadNumber,
NumberItems, MyName)

            # TB- 450

            Name = 'TB-450' # The name of a new load combination.
            ComboType = 0
            # This is 0, 1, 2, 3 or 4 indicating the load combination type.
            # 0 = Linear Additive
            # 1 = Envelope
            # 2 = Absolute Additive
            # 3 = SRSS
            # 4 = Range Additive
            ret = SapModel.RespCombo.Add(Name, ComboType)

            CName = ['Veiculo', 'Multidao']
            for i in range(2):
                CNameType = eCNameType(0)
                # This is one of the following items in the eCNameType enumeration:
                # LoadCase = 0
                # LoadCombo = 1
                # The name of the load case or load combination to be added to or modified in
the combination specified by the Name item.
                # If the load case or combination already exists in the combination specified
by the Name item, the scale factor is modified as indicated by the SF item for that load case
or combination.
                #If the analysis case or combination does not exist in the combination specified
by the Name item, it is added.
                SF = 1 # The scale factor multiplying the case or combination indicated by the
CName item.
                ret = SapModel.RespCombo.SetCaseList(Name, CNameType, CName[i], SF)




            #  6  Aparelhos  de  apoio  a  partir  de  um  link  definido  pelo  usuário
(cBMBearing.SetBearingUser Method)

            # 6.1 Aparelho de apoio

            Name = 'Elastômero' #The name of an existing or new link property. If this is an
existing property, that property is modified; otherwise, a new property is added.
            DOF = [True, True, True, False, False, False]
            # This is a boolean array, dimensioned to 5, indicating if properties exist for a
specified degree of freedom.
            # DOF(0) = U1
            # DOF(1) = U2
            # DOF(2) = U3
            # DOF(3) = R1
```

```
            # DOF(4) = R2
            # DOF(5) = R3
            Fixed = [True, False, False, False, False, False]
            # This is a boolean array, dimensioned to 5, indicating if the specified degree of
freedom is fixed (restrained).
            # Fixed(0) = U1 fixity if DOF(0) = True
            # Fixed(1) = U2 fixity if DOF(1) = True
            # Fixed(2) = U3 fixity if DOF(2) = True
            # Fixed(3) = R1 fixity if DOF(3) = True
            # Fixed(4) = R2 fixity if DOF(4) = True
            # Fixed(5) = R3 fixity if DOF(5) = True
            Ke = [float(), K_ap, K_ap, float(), float(), float()]
            # This is an array of stiffness terms for the link property. There are 6 terms in
the array if the stiffness is uncoupled and 21 if it is coupled. The KeCoupled item indicates
if the stiffness is coupled.
            # If the stiffness is uncoupled:
            # Ke(0) = U1 [F/L]
            # Ke(1) = U2 [F/L]
            # Ke(2) = U3 [F/L]
            # Ke(3) = R1 [FL]
            # Ke(4) = R2 [FL]
            # Ke(5) = R3 [FL]
            # If the stiffness is coupled:
            # Ke(0) = U1U1 [F/L]
            # Ke(1) = U1U2 [F/L]
            # Ke(2) = U2U2 [F/L]
            # Ke(3) = U1U3 [F/L]
            # Ke(4) = U2U3 [F/L]
            # Ke(5) = U3U3 [F/L]
            # Ke(6) = U1R1 [F]
            # Ke(7) = U2R1 [F]
            # Ke(8) = U3R1 [F]
            # Ke(9) = R1R1 [FL]
            # Ke(10) = U1R2 [F]
            # Ke(11) = U2R2 [F]
            # Ke(12) = U3R2 [F]
            # Ke(13) = R1R2 [FL]
            # Ke(14) = R2R2 [FL]
            # Ke(15) = U1R3 [F]
            # Ke(16) = U2R3 [F]
            # Ke(17) = U3R3 [F]
            # Ke(18) = R1R3 [FL]
            # Ke(19) = R2R3 [FL]
            # Ke(20) = R3R3 [FL]
            Ce = [float(), 0, 0, float(), float(), float()]
            # This is an array of damping terms for the link property. There are 6 terms in the
array if the damping is uncoupled and 21 if it is coupled. The CeCoupled item indicates if the
damping is coupled.
            # If the damping is uncoupled:
            # Ce(0) = U1 [F/L]
            # Ce(1) = U2 [F/L]
            # Ce(2) = U3 [F/L]
            # Ce(3) = R1 [FL]
            # Ce(4) = R2 [FL]
            # Ce(5) = R3 [FL]
            # If the damping is coupled:
            # Ce(0) = U1U1 [F/L]
            # Ce(1) = U1U2 [F/L]
            # Ce(2) = U2U2 [F/L]
            # Ce(3) = U1U3 [F/L]
            # Ce(4) = U2U3 [F/L]
            # Ce(5) = U3U3 [F/L]
            # Ce(6) = U1R1 [F]
            # Ce(7) = U2R1 [F]
            # Ce(8) = U3R1 [F]
            # Ce(9) = R1R1 [FL]
            # Ce(10) = U1R2 [F]
            # Ce(11) = U2R2 [F]
            # Ce(12) = U3R2 [F]
            # Ce(13) = R1R2 [FL]
            # Ce(14) = R2R2 [FL]
            # Ce(15) = U1R3 [F]
            # Ce(16) = U2R3 [F]
            # Ce(17) = U3R3 [F]
            # Ce(18) = R1R3 [FL]
```

```
            # Ce(19) = R2R3 [FL]
            dj2 = 0 #The distance from the J-End of the link to the U2 shear spring. This item
applies only when DOF(1) = True. [L]
            dj3 = 0 #The distance from the J-End of the link to the U3 shear spring. This item
applies only when DOF(2) = True. [L]
            KeCoupled = False #This item is True if the link stiffness, Ke, is coupled. There
are 21 terms in the Ke array if Ke is coupled; otherwise there are 6 terms.
            CeCoupled = False #This item is True if the link damping, Ce, is coupled. There are
21 terms in the Ce array if Ce is coupled; otherwise there are 6 terms.
            Notes = '' #The notes, if any, assigned to the property.
            GUID = '' #The GUID (global unique identifier), if any, assigned to the property.
If this item is input as Default, the program assigns a GUID to the property.
            ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3, KeCoupled,
CeCoupled, Notes, GUID) #Criação do link Neoprene

            ret = MyBridgeModeler.Bearing.AddNew('Ap. de apoio')
            ret = MyBridgeModeler.Bearing.SetBearingLink('Ap. de apoio', 'Elastômero')

            # 6.2 Link Rígido
            Name = 'Link rigido' #The name of an existing or new link property. If this is an
existing property, that property is modified; otherwise, a new property is added.
            DOF = [True, True, True, True, True, True]
            Fixed = [True, True, True, True, True, True]
            Ke = [float(), float(), float(), float(), float(), float()]
            Ce = [float(), float(), float(), float(), float(), float()]
            dj2 = 0
            dj3 = 0
            KeCoupled = False
            CeCoupled = False
            Notes = ''
            GUID = ''
            ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3, KeCoupled,
CeCoupled, Notes, GUID)

            Name = 'Link rigido (esforços)' #The name of an existing or new link property. If
this is an existing property, that property is modified; otherwise, a new property is added.
            DOF = [True, True, True, True, True, True]
            Fixed = [False, False, False, False, False, False]
            Ke = [1E9, 1E9, 1E9, 1E13, 1E13, 1E13] # Segundo a documentação do CSiBridge, a
rigidez de 1e9 kN/cm é uma rigidez suficientemente infinita para as translações e 1e13 para as
rotações
            Ce = [float(), float(), float(), float(), float(), float()]
            dj2 = 0
            dj3 = 0
            KeCoupled = False
            CeCoupled = False
            Notes = ''
            GUID = ''
            ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3, KeCoupled,
CeCoupled, Notes, GUID)




            # 7. Definição dos apoios extremos, intermediários (travessas), abutments e
atribuições

            ret = MyBridgeModeler.Bent.AddNew('Apoios extremos')
            ret = MyBridgeModeler.Bent.AddNew('Apoios intermediários')
            ret = MyBridgeModeler.Abutment.AddNew('Apoio aparelho de apoio')

            # 7.1 Definição - Travessas extremas

            Name = 'Apoios extremos' #The name of an existing bridge bent.
            CapBeamSection = f'Travessa encontro (C{f_ck})' #This is either "None" or the name
of the frame section property assigned to the cap beam.
            Foundation = 'Fixed' #Containing either Fixed, Pinned, or a defined foundation
spring.
            BentType = 0
            # 0 = Single bearing line (continuous superstructure)
            # 1 = Double bearing line (discontinuous superstructure)
            # 2 = Double bearing line with closure pour
            ret = MyBridgeModeler.Bent.SetGeneralData(Name, CapBeamSection, C_travessaext,
SupportType_ext, N_pilaresext, Foundation, BentType)
```

```
            Section = [f'Pilar (C{f_ck})', f'Pilar (C{f_ck})'] #A zero-based one-dimensional
array dimensioned to (NumberColumns - 1) containing the frame section properties assigned to
each column.
            Angle = [0, 0] #A zero-based one-dimensional array dimensioned to (NumberColumns -
1) containing the orientation of the column in degrees counterclockwise between a line parallel
to the bent and the column local 2 axis. [deg]
            Foundation = ['Fixed', 'Fixed'] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing either Fixed, Pinned, a defined foundation spring, or Previous.
Note that an error will occur if Previous is assigned to the first column.
            ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_ext, Height_ext,
Angle, Foundation)

            SuperConnection = 1
            # The superstructure connection.
            # 0 = Connect to girder top and bottom
            # 1 = Connect to girder bottom only
            ret = MyBridgeModeler.Bent.SetSuperConnectionSingleLine(Name, SuperConnection)

            # 7.2 Definição - Travessas intermediárias
            Name = 'Apoios intermediários'
            CapBeamSection = f'Travessa intermediária (C{f_ck})'
            Foundation = 'Fixed'
            BentType = 1
            ret  =  MyBridgeModeler.Bent.SetGeneralData(Name,  CapBeamSection,  C_travessaint,
SupportType_int, N_pilaresint, Foundation, BentType)

            Section = [f'Pilar (C{f_ck})', f'Pilar (C{f_ck})']
            Angle = [0, 0]
            Foundation = ['Fixed', 'Fixed']
            ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_int, Height_int,
Angle, Foundation)

            BeforeSuperConnection = 1
            # The superstructure connection before the bent.
            # 0 = Connect to girder top and bottom
            # 1 = Connect to girder bottom only
            BeforeDistanceToBearingLine = d_ap #The distance to the bearing line before the
bent. [L]
            BeforeDistanceToEndOfGirder = 0 #The distance to the end of the girder before the
bent. [L]
            AfterSuperConnection = 1
            # The superstructure connection after the bent.
            # 0 = Connect to girder top and bottom
            # 1 = Connect to girder bottom only
            AfterDistanceToBearingLine = d_ap #The distance to the bearing line after the bent.
[L]
            AfterDistanceToEndOfGirder = 0 #AfterDistanceToEndOfGirder
            ret           =           MyBridgeModeler.Bent.SetSuperConnectionDoubleLine(Name,
BeforeSuperConnection,        BeforeDistanceToBearingLine,        BeforeDistanceToEndOfGirder,
AfterSuperConnection, AfterDistanceToBearingLine, AfterDistanceToEndOfGirder)

            # 7.3 Definição - Apoio aparelho de apoio (apenas para análises de superestruturas)
            Name = 'Apoio aparelho de apoio' # The name of an existing bridge abutment.
            SuperConnection = 1
            # The superstructure connection.
            # 0 = Integral
            # 1 = Connect to Girder Bottom Only
            DistanceFromBridgeEnd = float() # The distance from the bridge end to the bearing
line. [L]
            FoundationProperty = 'Fixed' # This is either Fixed, Pinned or the name of a defined
foundation spring.
            SubstructureType = 1
            # The substructure type.
            # 0 = None
            # 1 = Foundation spring
            # 2 = Continuous beam (continuously supported)
            BeamProperty = '' # The frame section property assigned to the continuous beam.
This item only applies when the SubstructureType item is 1, Continuous beam (continuously
supported).
            BeamLength = float() # The length of the continuous beam. This item only applies
when the SubstructureType item is 1, Continuous beam (continuously supported). [L]
            ret      =      MyBridgeModeler.Abutment.SetAbutment(Name,      SuperConnection,
DistanceFromBridgeEnd, FoundationProperty, SubstructureType, BeamProperty, BeamLength)
```

```
            # 7.4 Atribuições

            Name = myBridgeObjName #The name of a bridge object.

            if NVãos != 1:
                for i in range(NVãos-1):
                    SpanNameBeforeBent = SpanName[2 + i*2] #The name of the span before the
considered bent.
                    BentProperty = 'Apoios intermediários' #The bent property assigned to the
specified bent.
                    BentDirection = 'Default' #The bearing of a line parallel to the bent. It
is either Default or a valid bearing string.
                    ret    =    MyBridgeModeler.BridgeObject.BentAssign.SetBentAssignment(Name,
SpanNameBeforeBent, BentProperty, BentDirection)

                    BentSupportName = SpanName[2 + i*2]
                    ret     =     MyBridgeModeler.BridgeObject.BentAssign.SetSupportName(Name,
SpanNameBeforeBent, BentSupportName)

                    SuperstructureIsContinousAtBent    =    False    #Indicates    whether    the
superstructure is continuous (True) or discontinuous (False) at the bent.
                    DiaphragmBeforeBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the diaphragm property at the bent.
                    # If the specified bent property is None or it has a single line of bearings
(continuous superstructure) this is the property of the diaphragm that occurs at the bent.
                    #  If  the  specified  bent  property  has  a  double  line  of  bearings
(discontinuous superstructure) this is the property of the diaphragm that occurs just before
the bent.
                    # This item does not apply if the specified bent property is a double
bearing line with closure pour.
                    InternalDiaphragmBeforeBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the internal diaphragm property at the bent.
                    # This only applies to bents at steel U girder or super T girder deck
sections.
                    # If the specified bent property is None or it has a single line of bearings
(continuous superstructure) this is the property of the internal diaphragm that occurs at the
bent.
                    # If the specified bent property has a double line of bearings this is the
property of the internal diaphragm that occurs just before the bent.
                    DiaphragmAfterBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the diaphragm property after the bent.
                    # It applies when the specified bent property has a double line of bearings
(discontinuous superstructure).
                    # This item does not apply if the specified bent property is a double
bearing line with closure pour.
                    InternalDiaphragmAfterBent = 'None'
                    # This is either None or the name of a defined bridge diaphragm property
indicating the internal diaphragm property after the bent.
                    # This only applies to bents at steel U girder or super T girder deck
sections.
                    # It applies when the specified bent property has a double line of bearings
and is not a closure pour.
                    ClosurePourDiaphragm = 'None' #This is the name of a defined closure pour
bridge diaphragm property. This item only applies when the specified bent type is Double Bearing
Line with Closure Pour.
                    Restrainer = 'None' #This is either None or the name of a defined bridge
diaphragm  property  indicating  the  restrainers  used  at  the  bent.  It  only  applies  when  the
superstructure is discontinuous at the bent.
                    RestrainerElevation = 0 #The restrainer elevation (at layout line, global
Z). this item only applies when the superstructure is discontinuous at the bent. [L]
                    MeshSuperstructureToMatchBent = False
                    # This is a flag indicating if the superstructure is to be meshed to match
the bent.
                    # This item only applies when the superstructure is continuous at the bent.
                    MeshSteelUGirderToMatchBent = False
                    # This is a flag indicating if the steel U girders are to be meshed to
match the bent.
                    # This item only applies to bents at steel U girder or super T girder deck
sections when the superstructure is continuous at the bent.
                    ret                                                                    =
MyBridgeModeler.BridgeObject.BentAssign.SetSuperstructureAssignment(Name,   SpanNameBeforeBent,
```

```
SuperstructureIsContinousAtBent,       DiaphragmBeforeBent,       InternalDiaphragmBeforeBent,
DiaphragmAfterBent, InternalDiaphragmAfterBent,
                    ClosurePourDiaphragm,          Restrainer,          RestrainerElevation,
MeshSuperstructureToMatchBent, MeshSteelUGirderToMatchBent)

                    Elevation = -h_laje -t3_1 - h_ap #The global Z elevation of the bent. [L]
                    HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the cap beam. [L]
                    ret    =    MyBridgeModeler.BridgeObject.BentAssign.SetBentLocation(Name,
SpanNameBeforeBent, Elevation, HorizontalOffset)

                    for j in range(NVãos - 1):
                        BearingLocation = [1, 2]
                        # This is one of the following indicating the bent bearing location:
                        # 1 = Single line of bearings or bearings before bent for double line
of bearings.
                        # 2 = Bearings after bent for double line of bearings.
                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the bent.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the bent.
                        ElevationAtLayoutLine = -h_laje -t3_1 - h_ap/2 #The bearing elevation
(global Z) at the layout line. [L]
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int()
                        #The number of bearings for the bridge section or for abutments at
Steel U Girder deck sections the number of bearings at the U girder.
                        # For bearings at steel U girder sections this item must be 1 or 2.
                        # This item only applies when the bearing assignment type is General
or when the abutment is at a Steel U Girder deck section.
                        UniformSpacing = int() #The uniform spacing of bearings. This item only
applies when the bearing assignment type is General. [L]
                        BearingOffset = int() #The offset from the section reference point to
the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                                =
MyBridgeModeler.BridgeObject.BentAssign.SetBearingAssignment(Name,          SpanNameBeforeBent,
BearingLocation[j],      BearingAssignmentType,      BearingProperty,      RestrainerProperty,
ElevationAtLayoutLine, RotationAngle, NumberOfBearings,
                        UniformSpacing, BearingOffset)

            ValidBentsList = ['Apoios intermediários', 'Apoios extremos']
            NumberNames = int()
            MyNames = []
            ret = MyBridgeModeler.Bent.GetNameList(NumberNames, MyNames)
            BentsList = list(ret[2])
            for i in range (ret[1]):
                if BentsList[i] not in ValidBentsList:
                    ret = MyBridgeModeler.Bent.Delete(BentsList[i])

            # Atribuição da continuidade entre perfis - GetTableForEditingArray
            ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            TableKey = 'Bridge Object Definitions 07 - Bents'
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = []
            NumberRecords = int()
            TableData = []
            ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,    GroupName,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
            ColumnLabels = list(ret[2])
            Resultados = list(ret[4])

            # Criação do dataframe - trocar BENT1 pela continuidade

            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
            df_bents = pd.DataFrame(ListadeListasDosResultados, columns=ColumnLabels)
```

```
            df_bents = df_bents.replace({',' : '.'}, regex = True) # Trocar vírgula por ponto
em todas as colunas
            df_bents = df_bents.replace(to_replace='BENT1', value = 'Free')
            result = df_bents.values.ravel().tolist()

            TableKey = 'Bridge Object Definitions 07 - Bents'
            TableVersion = int()
            FieldKeysIncluded = ColumnLabels
            NumberRecords = int()
            TableData = result
            ret   =   SapModel.DatabaseTables.SetTableForEditingArray(TableKey,   TableVersion,
FieldKeysIncluded, NumberRecords, TableData)
            ret = SapModel.DatabaseTables.ApplyEditedTables(False, int(), int(), int(), int(),
'')




            # 8 Definição e atribuição dos diafragmas

            Name = myBridgeObjName
            ret = MyBridgeModeler.BridgeObject.DiaphragmAssign.DeleteAll(Name, '')

            NumberNames = int() #The number of diaphragm names retrieved by the program.
            MyName = [] #This is a one-dimensional array of diaphragm names. The MyName array
is created as a dynamic, zero-based array by the API user: Dim MyName() as String.
            #The array is dimensioned to (NumberNames - 1) inside the program, filled with the
names, and returned to the API user.
            MyType = []
            # This is a one-dimensional array, dimensioned to (NumberNames - 1), of diaphragm
types.
            # 1 = Solid
            # 2 = Chord and Brace
            # 3 = Single Beam
            # 4 = Steel Plate
            # 5 = Closure Pour
            ret = MyBridgeModeler.Diaphragm.GetNameList(NumberNames, MyName, MyType)
            NDiaphragms = ret[1]
            DiaphragmNames = list(ret[2])
            ValidDiaphgragmNames = ['Diafragmas extremos', 'Diafragmas intermediários']
            for i in range (NDiaphragms):
                if DiaphragmNames[i] not in ValidDiaphgragmNames:
                    ret = MyBridgeModeler.Diaphragm.Delete(DiaphragmNames[i])

            Name = 'Diafragmas extremos'
            MyType = 3
            # The diaphragm type.
            # 1 = Solid
            # 2 = Chord and Brace
            # 3 = Single Beam
            # 4 = Steel Plate
            # 5 = Closure Pour
            ret = MyBridgeModeler.Diaphragm.AddNew(Name, MyType)

            Name = 'Diafragmas intermediários'
            MyType = 3
            # The diaphragm type.
            # 1 = Solid
            # 2 = Chord and Brace
            # 3 = Single Beam
            # 4 = Steel Plate
            # 5 = Closure Pour
            ret = MyBridgeModeler.Diaphragm.AddNew(Name, MyType)

            Name = 'Diafragmas extremos' #The name of an existing single beam-type bridge
diaphragm.
            BeamSection = 'Diafragma extremo' #The steel frame section property used for the
beam.
            ElevationChangeTop = 15 #The elevation change from top of beam to top of adjacent
girder. [L]
            ModelMixed = bool(False) #Indicates if the beam is modeled as mixed frame and shell
            ConnectionPlateWidth = lea #The width of the connection plate. [L]
            ConnectionPlateThickness = eea #The thickness of the connection plate. [L]
            ConnectionPlateMaterial = 'A588Gr50' #The steel material for the connection plate.
```

```
            ConnectionPlateOnBothSidesOfWeb = bool(True) #Indicates if the connection plate is
provided on both sides of the web.
            ret       =       MyBridgeModeler.Diaphragm.SetSingleBeam(Name,       BeamSection,
ElevationChangeTop,      ModelMixed,      ConnectionPlateWidth,      ConnectionPlateThickness,
ConnectionPlateMaterial, ConnectionPlateOnBothSidesOfWeb)

            Name = 'Diafragmas intermediários' #The name of an existing single beam-type bridge
diaphragm.
            BeamSection = 'Diafragma intermediário' #The steel frame section property used for
the beam.
            ElevationChangeTop = 15 #The elevation change from top of beam to top of adjacent
girder. [L]
            ModelMixed = bool(False) #Indicates if the beam is modeled as mixed frame and shell
            ConnectionPlateWidth = let #The width of the connection plate. [L]
            ConnectionPlateThickness = eet #The thickness of the connection plate. [L]
            ConnectionPlateMaterial = 'A588Gr50' #The steel material for the connection plate.
            ConnectionPlateOnBothSidesOfWeb = bool(True) #Indicates if the connection plate is
provided on both sides of the web.
            ret       =       MyBridgeModeler.Diaphragm.SetSingleBeam(Name,       BeamSection,
ElevationChangeTop,      ModelMixed,      ConnectionPlateWidth,      ConnectionPlateThickness,
ConnectionPlateMaterial, ConnectionPlateOnBothSidesOfWeb)

            for i in range(len(SpanName)):
                for j in range(len(DiaphragmDistanceSpan[i])):
                    Name = myBridgeObjName #The name of a bridge object.
                    Location = int(0)
                    # This is one of the following indicating the diaphragm location across the
bridge section.
                    # 0 = All Spaces
                    # 1 = Girder n to Girder n + 1
                    # 2 = All Girder Interiors
                    # 3 = Girder n Interior
                    GirderNumber = int() #The girder number identified as n when the Location
item is 1 or 3 (Girder n to Girder n + 1 or Girder n Interior). This item does not apply when
the Location item is 0 or 2.
                    Bearing = 'Default' #This is either Default or a valid bearing string
                    ReferenceLine = 0 #This is either 0 indicating the layout line or nonzero
indicating a girder number. This item is only under user control when the Location item is 1
(Girder n to Girder n + 1).
                    ErrorMessage = '' #If the function returns an error related to the diaphragm
distance this string may provide information about the error.
                    if    i    ==    0    and    j    ==    0    or    i    ==    len(SpanName)-1    and    j    ==
len(DiaphragmDistanceSpan[i]) - 1:
                        DiaphragmProperty = 'Diafragmas extremos' #The name of a defined
diaphram property assigned to the diaphragm.
                    else:
                        DiaphragmProperty = 'Diafragmas intermediários'
                    ret       =       MyBridgeModeler.BridgeObject.DiaphragmAssign.AddNew(Name,
SpanName[i], DiaphragmDistanceSpan[i][j], DiaphragmProperty, Location, GirderNumber, Bearing,
ReferenceLine, ErrorMessage)




            # 9 Bridge Section Data - Definition (cBMDeckSectionSteelIGirder Interface)
            LeftFillet = 0 #The horizontal dimension of the fillet on the left side of the deck
(identified as f1 on the Define Bridge Section Data form). [L]
            RightFillet = 0 #The horizontal dimension of the fillet on the right side of the
deck (identified as f2 on the Define Bridge Section Data form). [L]
            ret                                                                                =
MyBridgeModeler.DeckSection.SteelIGirder.SetFilletHorizontalDimensions(myDeckSectionName_1,
LeftFillet, RightFillet)

            SlabMaterial = f'C{f_ck} - Tabuleiro' #The name of the slab material property. This
is intended to be a concrete property
            GirderLongitudinalLayout = 1
            # Indicates how the girders are laid out at a horizontal curve in a bridge object.
            # 1 = Along layout line.
            # 2 = Straight line.
            ret = MyBridgeModeler.DeckSection.SteelIGirder.SetGeneralData(myDeckSectionName_1,
SlabMaterial, NumberInteriorGirders, TotalWidthOfDeckSection, GirderLongitudinalLayout)

            ConstantGirderSection = True #Indicates if all of the girder sections are the same.
            GirderSection = ['Perfil 1']
```

```
            ret                                                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderSectionProperties(myDeckSectionName_1,
ConstantGirderSection, GirderSection)

            ConstantSpacing = True
            Spacing = []
            ret                                                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderSpacingData(myDeckSectionName_1,
ConstantSpacing, Spacing)


            DistanceToFillet = t2_1/2 #The overhang distance to the fillet (identified as L3
on the Define Bridge Section Data form). [L] Qualquer valor inferior à metade da largura do
flange superior dará erro na definição da seção
            OuterThickness = h_laje #The overhang outer thickness (identified as t5 on the
Define Bridge Section Data form). [L]
            ret                                                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetLeftOverhangDimensions(myDeckSectionName_1,
OverhangLength, DistanceToFillet, OuterThickness)

            DistanceToFillet = t2_1/2 #The overhang distance to the fillet (identified as L3
on the Define Bridge Section Data form). [L]
            OuterThickness = h_laje #The overhang outer thickness (identified as t5 on the
Define Bridge Section Data form). [L]
            ret                                                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetRightOverhangDimensions(myDeckSectionName_1,
OverhangLength, DistanceToFillet, OuterThickness)

            TopSlabThickness = h_laje #The thickness of the top slab (t1 on Define Bridge
Section Data form). [L]
            ConstantHaunchHeight = True #Indicates if the haunch heights are constant for all
girders.
            HaunchHeight = [2.5]
            # A zero-based, one-dimensional array of girder haunch heights dimensioned to
(NumberInteriorGirders + 1)
            # The haunch heights are entered in order in this array from the left exterior
girder to the right exterior girder.
            # Only the first height in this array is used if the ConstantHaunchHeight item is
True. [L]
            ret                                                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetSlabThicknessData(myDeckSectionName_1,
TopSlabThickness, ConstantHaunchHeight, HaunchHeight)

            ModelingType = 0
            # The girder modeling object type.
            # 0 = Frame
            # 1 = Shell
            # 2 = Mixed
            MaxElementWidthFlange =  40 # The maximum meshed element width for the girder
flange. This item only applies when the ModelingType item is 1 (Shell). [L]
            MaxElementHeightWeb = 40 # The maximum meshed element height for the girder web.
This item only applies when the ModelingType item is either 1 or 2 (Shell or Mixed). [L]
            ret                                                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderModelingForAreaObjectModels(myDeckSectionNam
e_1, ModelingType, MaxElementWidthFlange, MaxElementHeightWeb)

            X = 0 #The offset in the X direction from the reference point to the insertion
point. [L]
            Y = 0 #The offset in the Y direction from the reference point to the insertion
point. [L]
            ret = MyBridgeModeler.DeckSection.InsertionPoint.SetPoint(myDeckSectionName_1, X,
Y)




            # 10 Bridge Section Data - Loads (cBMDeckSectionLoadData Interface) - Métodos:
SetBarrierLoads, SetConcretePourFormLoads, SetConcretePourHaunchLoads,
            #          SetConcretePourPermanentLoads,          SetConcretePourTemporaryLoads,
SetLiveLoadCurbLocations, SetSidewalkLoads, SetTemperatureLoads, SetWearingSurfaceLoads


            # 10.1 Carregamentos Deck Section Data
```

```
            LoadPatternName = 'Defensas' #This is either None or the name of the load pattern
to which the barrier load is applied. The intent is that the load pattern would be of type Dead.
            NumberBarriers = 2 #The number of barriers specified
            MeasureFrom = [1,2]
            # A zero-based one-dimensional array indicating the reference point the barrier
location is measured from for each barrier.
            # 1 = Left Edge of Deck
            # 2 = Right Edge of Deck
            DistanceToOuterEdge = [0,0] #The distance from the reference point specified by the
MeasureFrom item to the outer edge of the barrier. [L]
            Width = [40,40] #The barrier width. [L]
            Height = [81,81] #The barrier height. [L]
            WeightEccentricity = [0,0] #The eccentricity of the barrier weight. [L]
            WeightPerLength = [5.625/100,5.625/100] #The barrier weight per unit length. [F/L]
            ret  =  MyBridgeModeler.DeckSection.LoadData.SetBarrierLoads(myDeckSectionName_1,
LoadPatternName,   NumberBarriers,   MeasureFrom,   DistanceToOuterEdge,   Width,   Height,
WeightEccentricity, WeightPerLength)

            LoadPatternName = '' #This is either None or the name of the load pattern to which
the haunch load is applied. The intent is that the load pattern would be of type Dead.
            UnitWeight = float() #The unit weight of the form load. [F/L2]
            ret                                                                              =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourFormLoads(myDeckSectionName_1,
LoadPatternName, UnitWeight)

            LoadPatternName = ''
            ProgramDetermined = True #This flag indicates whether the haunch load is program
determined or user defined.
            UserValue  =  float()#The  haunch  load  value.  This  item  only  applies  when  the
ProgramDetermined item is False. [F/L]
            ret                                                                              =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourHaunchLoads(myDeckSectionName_1,
LoadPatternName, ProgramDetermined, UserValue)

            LoadPatternName = ''
            WearingSurface = float() #The integral wearing surface load. [F/L2]
            Utilities = float() #The utilities load. [F/L2]
            Miscellaneous = float() #The miscellaneous load. [F/L2]
            ret                                                                              =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourPermanentLoads(myDeckSectionName_1,
LoadPatternName, WearingSurface, Utilities, Miscellaneous)

            LoadPatternName = 'ConcPour Temporary'
            Miscellaneous = 2E-04
            AdditionalOverhang = float() #The additional overhang load. [F/L2]
            ret                                                                              =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourTemporaryLoads(myDeckSectionName_1,
LoadPatternName, AdditionalOverhang, Utilities, Miscellaneous)

            ProgramDetermined = True
            DistToInsideEdgeOfLeftCurb = 40
            DistToInsideEdgeOfRightCurb = 40
            DistanceToCenterlineOfMedianCurb = float()
            WidthOfMedianCurb = float()
            ret                                                                              =
MyBridgeModeler.DeckSection.LoadData.SetLiveLoadCurbLocations(myDeckSectionName_1,
ProgramDetermined,          DistToInsideEdgeOfLeftCurb,          DistToInsideEdgeOfRightCurb,
DistanceToCenterlineOfMedianCurb, WidthOfMedianCurb)

            LoadPatternName = '' #This is either None or the name of the load pattern to which
the sidewalk load is applied. The intent is that the load pattern would be of type Dead.
            NumberSidewalks = 0
            MeasureFrom = []
            DistanceToOuterEdge =[]
            Width = []
            WeightPerArea = []
            ret  =  MyBridgeModeler.DeckSection.LoadData.SetSidewalkLoads(myDeckSectionName_1,
LoadPatternName, NumberSidewalks, MeasureFrom, DistanceToOuterEdge, Width, WeightPerArea)

            LoadPatternName = ''
            WeightPerArea = float()
            ret                                                                              =
MyBridgeModeler.DeckSection.LoadData.SetWearingSurfaceLoads(myDeckSectionName_1,
LoadPatternName, WeightPerArea)
```

```
            LoadPatternPositiveName = 'Temperatura uniforme pos'#This is either None or the
name of the load pattern to which the positive temperature load is applied. The intent is that
the load pattern would be of type Temperature.
            TemperatureChangePositive = 15 #The positive temperature change. [T]
            LoadPatternNegativeName = 'Temperatura uniforme neg'#This is either None or the
name of the load pattern to which the negative temperature load is applied. The intent is that
the load pattern would be of type Temperature.
            TemperatureChangeNegative = -15 #The negative temperature change that should be
entered as zero or a negative number. [T]
            ret                                                                          =
MyBridgeModeler.DeckSection.LoadData.SetTemperatureLoads(myDeckSectionName_1,
LoadPatternPositiveName,          TemperatureChangePositive,          LoadPatternNegativeName,
TemperatureChangeNegative)

            ret          =          MyBridgeModeler.DeckSection.AddCopy('SeçãoMista-Perfil2',
myDeckSectionName_1) # Outro Deck Section -> referente ao perfil 2
            ConstantGirderSection = True #Indicates if all of the girder sections are the same.
            GirderSection = ['Perfil 2']
            ret                                                                          =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderSectionProperties('SeçãoMista-Perfil2',
ConstantGirderSection, GirderSection)
            myDeckSectionName_2 = 'SeçãoMista-Perfil2'

            for i in range(NVãos):
                if NTrechos <= 2:
                    ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName,
[f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'], [myDeckSectionName_1])
                if NTrechos == 3:
                    ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName,
[f'Perfil 2.{i+1}.1', f'Perfil 2.{i+1}.2'], [myDeckSectionName_2, myDeckSectionName_2])
                    ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName,
[f'Perfil 1.{i+1}'], [myDeckSectionName_1])
                else:
                    ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName,
[f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'], [myDeckSectionName_1])
                    ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName,
[f'Perfil 2.{i+1}.1', f'Perfil 2.{i+1}.2'], [myDeckSectionName_2])

            # 10.2 Limpeza das inutilizadas
            ret = SapModel.LoadPatterns.Delete('Temperature Pos')
            ret = SapModel.LoadPatterns.Delete('Temperature Neg')
            ret = SapModel.LoadPatterns.Delete('PreTension')
            ret = SapModel.LoadPatterns.Delete('Barrier')




            # 11 Atribuição dos Encontros (cBMAbutmentAssign Methods)

            Elevation_StartAbutment = -h_laje -t3_1 - h_ap #The global Z elevation of the
substructure. (start abutment) -> Distância entre o topo da laje e o CG do aparelho de apoio
            ElevationAtLayoutLine_StartAbutment = -h_laje -t3_1 - h_ap/2 #The bearing elevation
(global Z) at the layout line. [L] -> Distância entre o topo da laje e a base do aparelho de
apoio
            Elevation_EndAbutment = Elevation_StartAbutment #The global Z elevation of the
substructure.
            ElevationAtLayoutLine_EndAbutement  =  ElevationAtLayoutLine_StartAbutment  #The
bearing elevation (global Z) at the layout line. [L]

            # 11.1 Start Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 1
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'Start Abutment' #The support name. The support name cannot be blank
and it can not be the same as another abutment or bent support name
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
```

```
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
            ret                                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            if Tipo_analise == 'SUPERESTRUTURA' or Tipo_analise == 'MESOESTRUTURA':
                SubstructureType = 1
                AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the abutment
or bent property used for the substructure. [L]
            else:
                SubstructureType = 2
                AbutmentOrBentProperty = 'Apoios extremos'
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation,    SubstructureType,    AbutmentOrBentProperty,    Elevation_StartAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
            NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
            ret    =    MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_StartAbutment,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            # 11.2 End Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 2
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'End Abutment'
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
```

```
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
            ret                                                                                =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            if Tipo_analise == 'SUPERESTRUTURA' or Tipo_analise == 'MESOESTRUTURA':
                SubstructureType = 1
                AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the abutment
or bent property used for the substructure. [L]
            else:
                SubstructureType = 2
                AbutmentOrBentProperty = 'Apoios extremos'
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation,    SubstructureType,    AbutmentOrBentProperty,    Elevation_StartAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
            NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
            ret    =    MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_EndAbutement,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            ret = MyBridgeModeler.Abutment.Delete('BABT1')




            # 12 Carregamentos atuantes
            # 12.1 Bridge Object - Slab Wet Concrete Load Assigns

            Name = myBridgeObjName # The name of a bridge object.
            ConcretePourName = '' # The concrete pour name to be deleted. Enter a blank string
to delete all slab wet concrete load assignments.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.Delete(Name,
ConcretePourName)

            Name = myBridgeObjName
            ConcretePourName = 'CPSlab Permanent Construction' # The unique concrete pour name
for the load assignment.
            TopSlabGroup = 'SLAB' # The top slab type bridge group specified for the load
assignment.
            SlabWeightFactor = float(1) # The slab weight load factor for the load assignment.
            HaunchLoadFactor = float(0)
            # The haunch load factor for the load assignment. This item only applies if the
bridge section type is one of the following:
            # Steel I Girder
            # Steel U Girder
            # Precast Concrete I Girder
```

```
            # Precast Concrete U Girder
            SIPFormLoadFactor = float(0) # The SIP Form load factor for the load assignment.
This item only applies if the bridge section type is Steel I Girder.
            BridgeSectPermanentLoadFactor = float(1) # The bridge section permanent load factor
for the load assignment.
            BridgeSectTemporaryLoadFactor = float(0) # The bridge section temporary load factor
for the load assignment.
            AdditionalLoadPermanent = float(0) # The permanent additional load for the load
assignment. [F/L2]
            AdditionalLoadTemporary = float(0) # The temporary additional load for the load
assignment. [F/L2]
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.AddNew(Name,
ConcretePourName, TopSlabGroup, SlabWeightFactor, HaunchLoadFactor,
            SIPFormLoadFactor,  BridgeSectPermanentLoadFactor,  BridgeSectTemporaryLoadFactor,
AdditionalLoadPermanent, AdditionalLoadTemporary)

            Name = myBridgeObjName
            ConcretePourName = 'CPSlab All Construction' # The unique concrete pour name for
the load assignment.
            TopSlabGroup = 'SLAB' # The top slab type bridge group specified for the load
assignment.
            SlabWeightFactor = float(0) # The slab weight load factor for the load assignment.
            HaunchLoadFactor = float(0)
            # The haunch load factor for the load assignment. This item only applies if the
bridge section type is one of the following:
            # Steel I Girder
            # Steel U Girder
            # Precast Concrete I Girder
            # Precast Concrete U Girder
            SIPFormLoadFactor = float(0) # The SIP Form load factor for the load assignment.
This item only applies if the bridge section type is Steel I Girder.
            BridgeSectPermanentLoadFactor = float(0) # The bridge section permanent load factor
for the load assignment.
            BridgeSectTemporaryLoadFactor = float(1) # The bridge section temporary load factor
for the load assignment.
            AdditionalLoadPermanent = float(0) # The permanent additional load for the load
assignment. [F/L2]
            AdditionalLoadTemporary = float(0) # The temporary additional load for the load
assignment. [F/L2]
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.AddNew(Name,
ConcretePourName, TopSlabGroup, SlabWeightFactor, HaunchLoadFactor,
            SIPFormLoadFactor,  BridgeSectPermanentLoadFactor,  BridgeSectTemporaryLoadFactor,
AdditionalLoadPermanent, AdditionalLoadTemporary)

            # 12.2 Bridge Object - Temperature Load Assigns
            LoadPat = ['Temperatura gradiente pos', 'Temperatura gradiente neg'] # A zero-based
one-dimensional array of the load pattern used for each load assignment.
            TempGradientName = ['Temperatura gradiente', 'Temperatura gradiente'] # A zero-
based one-dimensional array of the name of the defined temperature gradient for each load
assignment. This item only applies when the TemperatureOption item is 1 (Temperature gradient).
            GradientType = [0, 1]
            # A zero-based one-dimensional array of the gradient type for each load assignment.
This item only applies when the TemperatureOption item is 1 (Temperature gradient).
            # 0 = Positive temperature change
            # 1 = Negative temperature change
            for i in range(NTrechos):
                for j in range(2):
                    Name = myBridgeObjName
                    TemperatureOption = 1
                    # A zero-based one-dimensional array of the temperature option for each
load assignment.
                    # 0 = Uniform temperature change
                    # 1 = Temperature gradient
                    UniformTempChange = float() # A zero-based one-dimensional array of the
uniform temperature change for each load assignment. This item only applies when the
TemperatureOption item is 0 (Uniform temperature change). [T]
                    ret = MyBridgeModeler.BridgeObject.LoadAssignTemperature.SetOneLoad(Name,
SpanName[i],    LoadPat[j],    TemperatureOption,    UniformTempChange,    TempGradientName[j],
GradientType[j])

            # 12.3 Bridge Object - Line Load Assigns

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Vento' # The load pattern for the load assignment.
```

```
            LoadDistribution = 'Vento força' #The named load distribution for the load
assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
            TransverseVariation = 'None' # The variation used for the load assignment. This is
either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignLine.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, TransverseVariation)

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Vento' # The load pattern for the load assignment.
            LoadDistribution = 'Vento momento' #The named load distribution for the load
assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
            TransverseVariation = 'None' # The variation used for the load assignment. This is
either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignLine.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, TransverseVariation)

            # 12.4 Bridge Object - Area Load Assigns

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Frenagem e aceleracao' # The load pattern for the load assignment.
            LoadDistribution = 'Frenagem e aceleracao' #The named load distribution for the
load assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
            LeftEdgeVariation = 'None' # The left edge variation used for the load assignment.
This is either None or the name of a defined variation.
            RightEdgeVariation = 'None' # The right edge variation used for the load assignment.
This is either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Pavimentação' # The load pattern for the load assignment.
            LoadDistribution = 'Pavimentação' #The named load distribution for the load
assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
            LeftEdgeVariation = 'None' # The left edge variation used for the load assignment.
This is either None or the name of a defined variation.
            RightEdgeVariation = 'None' # The right edge variation used for the load assignment.
This is either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)

            Name = myBridgeObjName # The name of a bridge object.
            LoadPat = 'Recapeamento' # The load pattern for the load assignment.
            LoadDistribution = 'Recapeamento' #The named load distribution for the load
assignment.
            StartStation = 0 # The start station for the load assignment. [L]
            EndStation = CTotal # The end station for the load assignment. [L]
            LeftEdgeVariation = 'None' # The left edge variation used for the load assignment.
This is either None or the name of a defined variation.
            RightEdgeVariation = 'None' # The right edge variation used for the load assignment.
This is either None or the name of a defined variation.
            ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,     LoadPat,
LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)




            # 13 Staged Construction Load Cases (Retração e fluência será atribuído após a
criação do encontro)
            # Apenas os load cases Veiculo, Multidao, Retração, Fluência e Cronograma são
necessários de definição (os load patterns já viram load cases automaticamente pela definição
dada)

            # 13.1 Update no modelo para ser possível selecionar os elementos da ponte
            Name = myBridgeObjName
            Action = 4 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
```

```
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ModelType = 2
            #An integer value between 1 and 3, inclusive, indicating the model type to be
created. This item applies when the Action item is 1 or 4.
            # 1 = Update as Spine Model Using Frame Objects
            # 2 = Update as Area Object Modell
            # 3 = Update as Solid Object Model
            SubmeshSize = 50 #The preferred maximum submesh size. This item applies when the
Action item is 1 or 4 and the ModelType item is 2 or 3. [L]
            ApplyAreaPropModifier = False #This item is to indicate whether or not to apply
property modifiers to area objects representing the bridge section to adjust for overlap.
            # This option only applies to bridge sections of type concrete box-girder, concrete
T-beam, flat slab, and user-defined (Section Designer).
            # When the bridge object is updated as an area-object model (ModelType is "Area"),
entering True will apply property modifiers to the area objects representing the bridge cross
section that will adjust their weight, mass, and stiffness factors f11 and m11
            # to match a corresponding solid-object model (ModelType is "Solid"), which has no
overlaps. This item applies when the Action item is 1 or 4 and the ModelType item is 2. [L]
            # True = Apply area property modifiers
            # False = Do not apply area property modifiers
            MaxDeckSegmentLength = 50 #The maximum segment length for deck spans. This item
applies when the Action item is 1 or 4. [L]
            MaxCapSegmentLength = 50 #The maximum segment length for bent cap beams. This item
applies when the Action item is 1 or 4. [L]
            MaxColumnSegmentLength = 50 #The maximum segment length for bent columns. This item
applies when the Action item is 1 or 4. [L]
            MeshSlabAtCriticalSteelIGirderLocations = False #A flag indicating that the slab
should be meshed at critical steel I-girder locations. This item applies when the Action item
is 1 or 4 and the bridge deck section is a Steel I-Girder section.
            ret = MyBridgeModeler.BridgeObject.Update(Name, Action, ModelType, SubmeshSize,
ApplyAreaPropModifier, MaxDeckSegmentLength, MaxCapSegmentLength, MaxColumnSegmentLength,
MeshSlabAtCriticalSteelIGirderLocations)

            # 13.2 Retração e Fluência

            ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('Retração e Fluência')

            InitialCase = ''
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('Retração e
Fluência', InitialCase)

            NLGeomType = 1
            # This is 0, 1 or 2, indicating the geometric nonlinearity option selected for the
load case.
            # 0 = None
            # 1 = P-delta
            # 2 = P-delta plus large displacements
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('Retração
e Fluência', NLGeomType)

            TimeDepMatProp = True
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('Retração e
Fluência', TimeDepMatProp)

            StagedSaveOption = 2
            # This is 0, 1, 2 or 3, indicating the results saved option for the load case.
            # 0 = End of final stage
            # 1 = End of each stage
            # 2 = Start and end of each stage
            # 3 = Two or more times in each stage
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('Retração e
Fluência', StagedSaveOption)

            NumberStages = 5 #The number of stages defined for the specified load case.
            Duration = [0, 28, 3000, 3000, 4000]#This is an array that includes the duration
in days for each stage.
            Output = [True, True, True, True, True]#This is an array that includes True or
False, indicating if analysis output is to be saved for each stage.
            OutputName = ['Retração-0', 'Retração-28', 'Retração-3000', 'Retração-6000',
'Retração-10000']#This is an array that includes a user-specified output name for each stage.
```

```
            Comment = ['GL2000', 'GL2000', 'GL2000', 'GL2000', 'GL2000']#This is an array that
includes a comment for each stage. The comment may be a blank string.

            ret  =  SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('Retração  e
Fluência', NumberStages, Duration, Output, OutputName, Comment)

            # 13.3 GirdersAndDiaphragms (peso da estrutura metálica)
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('GirdersAndDiaphragms')

            InitialCase = ''
            ret                                                                              =
SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('GirdersAndDiaphragms', InitialCase)

            NLGeomType = 1
            ret                                                                              =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('GirdersAndDiaphragms',
NLGeomType)

            TimeDepMatProp = True
            ret                                                                              =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('GirdersAndDiaphragms',
TimeDepMatProp)

            StagedSaveOption = 2
            ret                                                                              =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('GirdersAndDiaphragms',
StagedSaveOption)

            NumberStages = 1
            Duration = [0]
            Output = [True]
            OutputName = ['Diafragmas']
            Comment = ['PP']
            ret                                                                              =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('GirdersAndDiaphragms',
NumberStages, Duration, Output, OutputName, Comment)

            # 13.4 DeckPlus
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('DeckPlus')

            InitialCase = ''
            ret     =     SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('DeckPlus',
InitialCase)

            NLGeomType = 1
            ret                                                                              =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('DeckPlus', NLGeomType)

            TimeDepMatProp = True
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('DeckPlus',
TimeDepMatProp)

            StagedSaveOption = 2
            ret     =     SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('DeckPlus',
StagedSaveOption)

            NumberStages = 1
            Duration = [0]
            Output = [True]
            OutputName = ['ConcretePour']
            Comment = ['Deck+']
            ret  =  SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('DeckPlus',
NumberStages, Duration, Output, OutputName, Comment)

            # 13.5 DeckCured - No Composed
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('DeckCured - No Composed')

            InitialCase = 'DeckPlus'
            ret  =  SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('DeckCured  -  No
Composed', InitialCase)

            NLGeomType = 1
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('DeckCured
- No Composed', NLGeomType)
```

```
            TimeDepMatProp = True
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('DeckCured
- No Composed', TimeDepMatProp)

            StagedSaveOption = 2
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('DeckCured - No
Composed', StagedSaveOption)

            NumberStages = 1
            Duration = [0]
            Output = [True]
            OutputName = ['ConcretePour Cured']
            Comment = ['Remove Forms']
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('DeckCured -
No Composed', NumberStages, Duration, Output, OutputName, Comment)

            # 13.6 ConcPour Temporary
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('ConcPour Temporary')

            InitialCase = ''
            ret      =      SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('ConcPour
Temporary', InitialCase)

            NLGeomType = 1
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('ConcPour
Temporary', NLGeomType)

            TimeDepMatProp = True
            ret = SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('ConcPour
Temporary', TimeDepMatProp)

            StagedSaveOption = 2
            ret      =      SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('ConcPour
Temporary', StagedSaveOption)

            NumberStages = 1
            Duration = [0]
            Output = [True]
            OutputName = ['Construção']
            Comment = ['Sobrecarga']
            ret   =   SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('ConcPour
Temporary', NumberStages, Duration, Output, OutputName, Comment)

            # 13.7 Set Stage Data (métodos não estavam funcionando)

            ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            TableKey = 'Case - Static 6 - Nonlinear Stage Data'
            GroupName = ''
            TableVersion = int()
            FieldKeysIncluded = []
            NumberRecords = int()
            TableData = []
            ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,    GroupName,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
            ColumnLabels = list(ret[2])
            Resultados = list(ret[4])

            TableKey = 'Case - Static 6 - Nonlinear Stage Data'
            TableVersion = int()
            FieldKeysIncluded = ColumnLabels
            NumberRecords = int()
            TableData = [
            'ConcPour       Temporary',       '1',       'Add       Structure',       'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '3', '', '', '', '', '', '',
            'ConcPour     Temporary',     '1',     'Add     Guide     Structure',     'Group',
f'<{myBridgeObjName}>SLAB', '', '', '', '', '', '', '',
            'ConcPour     Temporary',     '1',     'Pour     Concrete',     'Concrete     Pour',
f'<{myBridgeObjName}>CPSlab All Construction', '', '', '', '', '', '', '',
            'GirdersAndDiaphragms',        '1',        'Add       Structure',        'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '0', '', '', '', '', '', '',
            'GirdersAndDiaphragms',    '1',    'Load    Objects    If    Added',    'Group',
f'<{myBridgeObjName}>GIRDER_DIAPH', '', 'Load pattern', 'DEAD', '1', '', '', '',
```

```
            'Retração e Fluência', '1', 'Add Structure', 'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '28', '', '', '', '', '', '',
            'Retração e Fluência', '1', 'Add Structure', 'Group', f'<{myBridgeObjName}>SLAB',
'36', '', '', '', '', '', '',
            'DeckCured - No Composed', '1', 'Remove Forms', 'Concrete Pour',
f'<{myBridgeObjName}>CPSlab Permanent Construction', '0', '', '', '', '', '', '',
            'DeckPlus', '1', 'Add Structure', 'Group', f'<{myBridgeObjName}>ALL_BUT_SLAB', '3',
'', '', '', '', '', '', '',
            'DeckPlus', '1', 'Add Guide Structure', 'Group', f'<{myBridgeObjName}>SLAB', '',
'', '', '', '', '', '', '',
            'DeckPlus', '1', 'Pour Concrete', 'Concrete Pour', f'<{myBridgeObjName}>CPSlab
Permanent Construction', '', '', '', '', '', '', ''
            ]
            ret = SapModel.DatabaseTables.SetTableForEditingArray(TableKey, TableVersion,
FieldKeysIncluded, NumberRecords, TableData)
            ret = SapModel.DatabaseTables.ApplyEditedTables(False, int(), int(), int(), int(),
'')

            # Update no modelo
            Name = myBridgeObjName
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)




            # 14 Definição dos grupos comuns para pegar resultados da análise e encontros

            # Grupos comuns
            ret = SapModel.SelectObj.ClearSelection()
            Elementos = ['Diafragmas', 'Diafragmas', 'Pilares', 'Travessas', 'Travessas']
            PropSecoes = ['Diafragma extremo', 'Diafragma intermediário', f'Pilar (C{f_ck})',
f'Travessa encontro (C{f_ck})', f'Travessa intermediária (C{f_ck})']
            for i in range(len(Elementos)):
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.GroupDef.SetGroup(Elementos[i])
                ret = SapModel.SelectObj.PropertyFrame(PropSecoes[i])
                ret = SapModel.FrameObj.SetGroupAssign('', Elementos[i], False, eItemType(2))




            # 15 Modelagem do encontro

            if Tipo_analise == 'ENCONTROS':

                # 15.1 Alas
                OffsetEncontro = 75

                # Ala da direita
                NumberPoints = 4 # The number of points in the area abject.
                x = [-OffsetEncontro, -OffsetEncontro-CompA, -OffsetEncontro-CompA, -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                # Escolheu-se o sentido anti-horário, sendo o primeiro ponto o mais próximo da
laje (em x e z)
                # The coordinates should be ordered to run clockwise or counter clockwise around
the area object.
                Name = ''
                # This is the name that the program ultimately assigns to the area object. If
no UserName is specified, the program assigns a default name to the area object.
                # If a UserName is specified and that name is not used for another area object,
the UserName is assigned to the area object; otherwise a default name is assigned to the area
object.
```

```
                PropName = 'Encontro'
                # This is Default, None or the name of a defined area property.
                # If it is Default, the program assigns a default area property to the area
object.
                # If it is None, no area property is assigned to the area object. If it is the
name of a defined area property, that property is assigned to the area object.
                UserName = 'Ala direita 1/2' # This is an optional user specified name for the
area object. If a UserName is specified and that name is already used for another area object,
the program ignores the UserName.
                CSys = 'GLOBAL' # The name of the coordinate system in which the area object
point coordinates are defined.
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-ChanfroH,  -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala direita 2/2'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala direita 1/2' # The name of an existing area object.
                MeshType = 1
                # This item is 1, 2, 3, 4, 5 or 6, indicating the mesh type for the area object.
                # 1 = Mesh area into a specified number of objects
                # 2 = Mesh area into objects of a specified maximum size
                # 3 = Mesh area based on points on area edges
                # 4 = Cookie cut mesh area based on lines intersecting edges
                # 5 = Cookie cut mesh area based on points
                # 6 = Mesh area using General Divide Tool
                AreaName = [] # This is an array of the name of each area object created when
the specified area object is divided.
                n1_aladireita1 = 10 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 2.
                n2_aladireita1 = 5 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 3.
                NumberAreas = n1_aladireita1*n2_aladireita1 #The number of area objects created
when the specified area object is divided.
                ret   =   SapModel.EditArea.Divide(Name,   MeshType,   NumberAreas,   AreaName,
n1_aladireita1, n2_aladireita1)
                Ala_direita = list(ret[2])

                Name = 'Ala direita 2/2'
                MeshType = 1
                AreaName = []
                n1_aladireita2 = 10
                n2_aladireita2 = 10
                NumberAreas = n1_aladireita2*n2_aladireita2
                ret   =   SapModel.EditArea.Divide(Name,   MeshType,   NumberAreas,   AreaName,
n1_aladireita2, n2_aladireita2)
                Ala_direita.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Ala direita') # This is the name of a group.
If this is the name of an existing group,  that group is modified, otherwise a new group is
added.
                for Area in Ala_direita:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala direita' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
```

```
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # Ala da esquerda

                NumberPoints = 4
                x   =   [-OffsetEncontro,   -OffsetEncontro-CompA,   -OffsetEncontro-CompA,   -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 1/2'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                x  =  [-OffsetEncontro,  -OffsetEncontro-CompA,  -OffsetEncontro-ChanfroH,  -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 2/2'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala esquerda 1/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda1 = 10
                n2_alaesquerda1 = 5
                NumberAreas = n1_alaesquerda1*n2_alaesquerda1
                ret   =   SapModel.EditArea.Divide(Name,   MeshType,   NumberAreas,   AreaName,
n1_alaesquerda1, n2_alaesquerda1)
                Ala_esquerda = list(ret[2])

                Name = 'Ala esquerda 2/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda2 = 10
                n2_alaesquerda2 = 10
                NumberAreas = n1_alaesquerda2*n2_alaesquerda2
                ret   =   SapModel.EditArea.Divide(Name,   MeshType,   NumberAreas,   AreaName,
n1_alaesquerda2, n2_alaesquerda2)
                Ala_esquerda.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Ala esquerda') # This is the name of a group.
If this is the name of an existing group,  that group is modified, otherwise a new group is
added.
                for Area in Ala_esquerda:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala esquerda' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
```

```
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 15.2 Parede cortina
                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, -ChanfroV, -ChanfroV, 0]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 1/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -AlturaM, -AlturaM, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 2/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Discretização
                Name = 'Cortina 1/2'
                MeshType = 1
                AreaName = []
                n1_cortina1 = 5
                n2_cortina1 = 20
                NumberAreas = n1_cortina1*n2_cortina1
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_cortina1, n2_cortina1)
                Cortina = list(ret[2])

                Name = 'Cortina 2/2'
                MeshType = 1
                AreaName = []
                n1_cortina2 = 10
                n2_cortina2 = 20
                NumberAreas = n1_cortina2*n2_cortina2
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_cortina2, n2_cortina2)
                Cortina.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Cortina')
                for Area in Cortina:
                    GroupName = 'Cortina'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 15.3 Console
                NumberPoints = 4
                Largura_console = 30
                x = [-OffsetEncontro, -OffsetEncontro-Largura_console, -OffsetEncontro-
Largura_console, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla, -TotalWidthOfDeckSection/2 + hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1), -
2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1)] # Convencionado dessa forma para que
a discretização fique robusta
                Name = ''
                PropName = 'Encontro'
                UserName = 'Console'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)
```

```
                # Determinação dos pontos para realizar a discretização
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Cortina', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_cortina = list(ret[3])
                Points_cortina = []
                for Area in Areas_cortina:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_cortina.extend(list(ret[2]))

                Pointscortina_coord = []
                for point in Points_cortina:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(), float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointscortina_coord.append(ret)

                for i in range(len(Pointscortina_coord)):
                    Pointscortina_coord[i].insert(0,Points_cortina[i])

                df_pontos_cortina = pd.DataFrame(Pointscortina_coord, columns = ['Joint', 'X', 'Y', 'Z'])
                df_pontos_cortina = df_pontos_cortina.round(4)

                Pointscortina_mesh = df_pontos_cortina[(df_pontos_cortina["Z"] == -2*(ChanfroV/n2_aladireita1))]['Joint'].unique().tolist()

                ret = SapModel.SelectObj.ClearSelection()
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização
                Name = 'Console'
                MeshType = 3
                AreaName = []
                n1_console = 2
                n2_console = 20
                NumberAreas = n1_console*n2_console
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(), float(), float(), False, False, True)

                Console = list(ret[2])
                ret = SapModel.GroupDef.SetGroup('Console') # This is the name of a group.  If this is the name of an existing group,  that group is modified, otherwise a new group is added.
                for Area in Console:
                    # Name = The name of an existing area object or group, depending on the value of the ItemType item.
                    GroupName = 'Console' # The name of an existing group to which the assignment is made.
                    Remove = False # If this item is False, the specified area objects are added to the group specified by the GroupName item. If it is True, the area objects are removed from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 15.4 Laje de transição
                OffsetLT = OffsetEncontro + Largura_console
                NumberPoints = 4
                x = [-OffsetLT, -OffsetLT - LarguraLT, -OffsetLT - LarguraLT, -OffsetLT]
                y = [-TotalWidthOfDeckSection/2 + hAla, -TotalWidthOfDeckSection/2 + hAla, TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-h_laje/2, -h_laje/2, -h_laje/2, -h_laje/2]
                Name = ''
```

```
                PropName = 'Laje de transição'
                UserName = 'LT'
                CSys = 'GLOBAL'
                ret =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Obtenção dos pontos de discretização
                ret = SapModel.SelectObj.ClearSelection()
                Pointscortina_mesh = Pointscortina_mesh[1:-1]
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                dx = - float(Largura_console)
                dz = 2*(ChanfroV/n2_aladireita1) - h_laje/2
                ret = SapModel.EditGeneral.ReplicateLinear(dx, float(0), dz, int(1), int(), [],
[], False)

                PointsLT_mesh = list(ret[2])
                ret = SapModel.SelectObj.ClearSelection()
                for point in PointsLT_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização nos pontos do console
                Name = 'LT'
                MeshType = 3
                AreaName = []
                n1_LT = 10
                n2_LT = 20
                NumberAreas = n1_LT*n2_LT
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)
                LT = list(ret[2])

                # Discretização LT
                LT_adic = []
                for Area in LT:
                    ret = SapModel.EditArea.Divide(Area, 1, int(), AreaName, 10, 1)
                    LT_adic.extend(list(ret[2]))

                LT.extend(LT_adic)
                ret = SapModel.GroupDef.SetGroup('Lajes de transição') # This is the name of a
group.  If this is the name of an existing group,  that group is modified, otherwise a new group
is added.

                for Area in LT:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Lajes de transição' # The name of an existing group to which
the assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 15.5 Links laje de transição
                XMin = -OffsetLT - LarguraLT
                XMax = -OffsetLT - LarguraLT/2
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
```

```
                Point = False
                Line = False
                Area = True
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                Name = '' # The name of an existing area object or group, depending on the
value of the ItemType item.
                MyType = 1
                # This is either 1 or 2, indicating the spring property type.
                # 1 = Simple spring
                # 2 = Link property
                s = 0.05 # The simple spring stiffness per unit area of the specified area
object face. This item applies only when MyType = 1. [F/L3]
                SimpleSpringType = 2
                # This is 1, 2 or 3, indicating the simple spring type. This item applies only
when MyType = 1.
                # 1 = Spring resists tension and compression
                # 2 = Spring resists compression only
                # 3 = Spring resists tension only
                LinkProp = '' # The name of the link property assigned to the spring. This item
applies only when MyType = 2.
                Face = -2
                # This is -1, -2 or a nonzero, positive integer indicating the area object face
to which the specified spring assignment applies.
                # -1 = Bottom face
                # -2 = Top face
                # >0 = Edge face
                SpringLocalOneType = 2
                # This is 1, 2 or 3, indicating the method used to specify the spring positive
local 1-axis orientation.
                # 1 = Parallel to area object local axis
                # 2 = Normal to specified area object face
                # 3 = User specified direction vector
                Dir = int() # This is 1, 2, 3, -1, -2 or -3, indicating the area object local
axis that corresponds to the positive local 1-axis of the spring. This item applies only when
SpringLocalOneType = 1.
                Outward = False # This item is True if the spring positive local 1 axis is
outward from the specified area object face. This item applies only when SpringLocalOneType =
2.
                Vec = [] # This is an array of three values that define the direction vector
of the spring positive local 1-axis.
                # The direction vector is in the coordinate system specified by the CSys item.
This item applies only when SpringLocalOneType = 3.
                Ang = float() # This is the angle that the link local 2-axis is rotated from
its default orientation. This item applies only when MyType = 2. [deg]
                Replace = True #If this item is True, all existing spring assignments to the
area object are removed before assigning the specified spring.
                # If it is False, the specified spring is added to any existing springs already
assigned to the area object.
                CSys = 'Local' #This is Local (meaning the area object local coordinate system)
or the name of a defined coordinate system.
                #This item is the coordinate system in which the user specified direction
vector, Vec, is specified. This item applies only when SpringLocalOneType = 3.
                ItemType = eItemType(2)
                ret = SapModel.AreaObj.SetSpring(Name, MyType, s, SimpleSpringType, LinkProp,
Face, SpringLocalOneType, Dir, Outward, Vec, Ang, Replace, CSys, ItemType)

                ret = SapModel.SelectObj.ClearSelection()

                # 15.6 Links console
                XMin = -OffsetLT
                XMax = -OffsetLT
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = True
                Line = False
                Area = False
```

```
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])

                Points_linkconsole = list(ret[3])
                x = []
                y = []
                z = []
                for point in Points_linkconsole:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(),
float(), 'GLOBAL')
                    x.append(ret[1])
                    y.append(ret[2])
                    z.append(ret[3])

                ret = SapModel.SelectObj.ClearSelection()
                Links_console = []
                for i in range(len(Points_linkconsole)):
                    xi = x[i] # The coordinates of the I-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item.
                    yi = y[i]
                    zi = z[i]
                    xj = x[i] # The coordinates of the J-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item. These coordinates are ignored
if the IsSingleJoint item is True.
                    yj = y[i]
                    zj = -2*(ChanfroV/n2_aladireita1)
                    Name = ''
                    # This is the name that the program ultimately assigns for the link object.
                    # If no UserName is specified, the program assigns a default name to the
link object.
                    # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                    IsSingleJoint = False # This item is True if a one-joint link is added and
False if a two-joint link is added.
                    PropName = 'Link rigido (esforços)'
                    UserName = ''
                    CSys = 'GLOBAL'
                    ret = SapModel.LinkObj.AddByCoord(xi, yi, zi, xj, yj, zj, Name,
IsSingleJoint, PropName, UserName, CSys)
                    Links_console.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links console')
                for link in Links_console:
                    GroupName = 'Links console'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 15.7 Links travessa # Nao gostei muito dessa solução, investigar outra forma
de fazer
                Name = f'Travessa encontro (C{f_ck})'
                DeSelect = False
                ret = SapModel.SelectObj.PropertyFrame(Name, DeSelect)

                NumberItems = int() # The number of selected objects.
                ObjectType = []
                # This is an array that includes the object type of each selected object.
                # 1 = Point object
                # 2 = Frame object
                # 3 = Cable object
                # 4 = Tendon object
                # 5 = Area object
                # 6 = Solid object
                # 7 = Link object
                ObjectName = [] # This is an array that includes the name of each selected
object.
                ret = SapModel.SelectObj.GetSelected(NumberItems, ObjectType, ObjectName)

                Name = f'{list(ret[3])[0]}'
                Point1 = ''
                Point2 = ''
```

```
                ret = SapModel.FrameObj.GetPoints(Name, Point1, Point2)

                Name = ret[1]
                x = float()
                y = float()
                z = float()
                CSys = 'GLOBAL'
                ret = SapModel.PointObj.GetCoordCartesian(Name, x, y, z, CSys)
                xiTravessa = ret[1]
                yiTravessa = ret[2]
                ziTravessa = ret[3]

                zi = [-AlturaM, -AlturaM + ((AlturaM - ChanfroV)/(n1_cortina2))* 6]
                Links_travessa = []
                for j in range (2):
                    for i in range (n2_cortina2+1):
                        xi = -OffsetEncontro # The coordinates of the I-End of the added link
object. The coordinates are in the coordinate system defined by the CSys item.
                        yi     =     -TotalWidthOfDeckSection/2     +     hAla/2     -     i*(-
TotalWidthOfDeckSection/2 + hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                        xj = xiTravessa # The coordinates of the J-End of the added link object.
The coordinates are in the coordinate system defined by the CSys item. These coordinates are
ignored if the IsSingleJoint item is True.
                        yj     =     -TotalWidthOfDeckSection/2     +     hAla/2     -     i*(-
TotalWidthOfDeckSection/2 + hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                        zj = ziTravessa
                        Name = ''
                        # This is the name that the program ultimately assigns for the link
object.
                        # If no UserName is specified, the program assigns a default name to
the link object.
                        # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                        IsSingleJoint = False # This item is True if a one-joint link is added
and False if a two-joint link is added.
                        PropName = 'Link rigido'
                        UserName = ''
                        CSys = 'GLOBAL'
                        ret = SapModel.LinkObj.AddByCoord(xi, yi, zi[j], xj, yj, zj, Name,
IsSingleJoint, PropName, UserName, CSys)
                        Links_travessa.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links travessa')
                for link in Links_travessa:
                    GroupName = 'Links travessa'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 15.8 Definição dos grupos

                ret = SapModel.GroupDef.SetGroup('Encontro')

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Links console')
                ret = SapModel.SelectObj.Group('Links travessa')
                ret = SapModel.LinkObj.SetGroupAssign('', 'Encontro', False, eItemType(2))


                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Ala direita')
                ret = SapModel.SelectObj.Group('Ala esquerda')
                ret = SapModel.SelectObj.Group('Console')
                ret = SapModel.SelectObj.Group('Cortina')
                ret = SapModel.SelectObj.Group('Lajes de transição')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Encontro', False, eItemType(2))

                ret = SapModel.GroupDef.SetGroup('Todas lajes')
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Laje de transição')
                ret = SapModel.SelectObj.Group(f'<{myBridgeObjName}>SLAB')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Todas lajes', False, eItemType(2))

                # 15.9 Atribuição dos carregamentos nos encontros
```

```
                ret = SapModel.SelectObj.ClearSelection()
                EmpuxoTerra = ws_pil_ext['D82'].value/(100)**2
                EmpuxoTerraCargaMóvel = ws_pil_ext['D93'].value/(100)**2
                Cases = [['Ala direita', 'Empuxo terra', -EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
                ['Ala direita', 'Empuxo terra carga móvel', -EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Ala esquerda', 'Empuxo terra', EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
                ['Ala esquerda', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra', EmpuxoTerra, 4, False, 'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 4, False,
'GLOBAL', eItemType(1)]]

                for case in Cases:
                    Name, LoadPat, Value, Dir, Replace, CSys, ItemType = case
                    ret = SapModel.AreaObj.SetLoadUniform(Name, LoadPat, Value, Dir, Replace,
CSys, ItemType)

                # 16.10 Impacto na defensa

                # 16.10.1 Obtenção dos pontos do encontro para a atribuição do carregamento

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Encontro', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_encontro = list(ret[3])
                Points_encontro = [] # Nomes dos pontos que definem as áreas
                for Area in Areas_encontro:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_encontro.extend(list(ret[2]))

                Pointsencontro_coord = []
                for point in Points_encontro:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointsencontro_coord.append(ret)

                for i in range(len(Pointsencontro_coord)):
                    Pointsencontro_coord[i].insert(0,Points_encontro[i]) # Inserção dos nomes
dos pontos em cada lista

                # 16.10.2 Criação do dataframe com todos os elementos de área e os quatro pontos
que os definem (start e end station será feito depois de discretizar a laje)
                df_pontos_encontro = pd.DataFrame(Pointsencontro_coord, columns = ['Joint',
'X', 'Y', 'Z']) # DataFrame com todas as áreas, sendo cada linha um ponto que a define
                df_pontos_encontro = df_pontos_encontro.round(4) # Arredondar todas as colunas
do DataFrame para 4 casas decimais

                # 16.10.3 Obtenção dos pontos

                Points_impacto_esq = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) &
(df_pontos_encontro["Y"] == -TotalWidthOfDeckSection/2 + hAla/2) & (df_pontos_encontro["X"] <=
-OffsetEncontro)]['Joint'].unique().tolist()
                Points_impacto_dir = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) &
(df_pontos_encontro["Y"] == TotalWidthOfDeckSection/2 - hAla/2) & (df_pontos_encontro["X"] <= -
OffsetEncontro)]['Joint'].unique().tolist()

                # 16.10.4 Atribuição do carregamento
                fdist_impacto = 100/(50 + 2*87) # em kN/cm
                fno_impacto = (fdist_impacto * CompA)/len(Points_impacto_esq)

                mdist_impacto = 100*87/(50 + 2*87) # em kN.cm/cm
                mno_impacto = mdist_impacto*CompA/len(Points_impacto_esq)

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_esq:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , fno_impacto, 0, -
mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))
```

```
                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_dir:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , -fno_impacto, 0,
mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))




            # 15 Schedule
            if Tipo_analise == 'ENCONTROS':
                ret = SapModel.Scheduler.Add('Cronograma')

                Name = 'Cronograma'
                NumberTasks = 17
                Expanded = [True, True, True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True]
                IndentLevel = [0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1]
                TaskName = ['Infraestrutura', 'Pilares', 'Travessas', 'Encontros', 'Aparelho
de apoio', 'Superestrutura', 'Estrutura metálica', 'Pré-lajes', 'Concretagem', 'Concretagem',
'Cura do concreto', 'Lajes de transição', 'Barreiras', 'Pavimentação', 'Recapeamento', 'Pós
contrução', 'Carga móvel']
                Duration = ['390', '180', '120', '90', '30', '210', '30', '15', '1', '30',
'30', '30', '14', '30', '30', '30', '3600', '3600']
                CompletionDate = [390, 180, 300, 390, 420, 600, 420, 435, 436, 466, 496, 526,
540, 570, 600, 4200, 4200]
                Dependencies = ['', '', '2', '3', '4', '', '1', '7', '8', '9', '10', '11',
'12', '13', '14', '', '15']
                Operation = [None, 'Add Structure', 'Add Structure', 'Add Structure', 'Add
Structure', None, 'Add Structure', 'Load Objects', 'Add Guide Structure', 'Pour Concrete',
'Remove Forms', 'Add Structure', 'Load Objects If Added', 'Load Objects If Added', 'Load
Objects', None, 'Load Objects']
                IncludeSW = [False, True, True, True, False, False, True, False, False, False,
False, True, False, False, False, False, False]
                ObjectType = [None, 'Group', 'Group', 'Group', 'Group', None, 'Group', 'Group',
'Group', 'Concrete Pour', 'Concrete Pour', 'Group', 'Group', 'Group', 'Group', None, 'Group']
                ObjectName = [None, 'Pilares', 'Travessas', 'Encontro', 'Links', None,
f'<{myBridgeObjName}>ALL_BUT_SLAB',                          f'<{myBridgeObjName}>GIRDER_DIAPH',
f'<{myBridgeObjName}>SLAB', f'Concrete Pour <{myBridgeObjName}>CPSlab Permanent Construction',
f'Concrete Pour <{myBridgeObjName}>CPSlab Permanent Construction', 'Lajes de transição', 'ALL',
'ALL', f'<{myBridgeObjName}>ALL', None, f'<{myBridgeObjName}>ALL']
                Details = [None, 'Age=3; SW=DEAD; SF=1', 'Age=3; SW=DEAD; SF=1', 'Age=3;
SW=DEAD; SF=1', 'Age=3; Without SW', None, 'Age=1; SW=DEAD; SF=1', "Load pattern 'SIPForm'
SF=1", 'Guide', 'Wet Slabs', 'Forms', 'Age=3; SW=DEAD; SF=1', "Load pattern 'Defensas' SF=1",
"Load pattern 'Pavimentação' SF=1", "Load pattern 'Recapeamento' SF=1", None, "Load pattern
'Veiculo' SF=1"]
                Age = [0.0, 3.0, 3.0, 3.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0,
0.0, 0.0, 0.0]
                LoadType = [None, 'Load pattern', 'Load pattern', 'Load pattern', '', None,
'Load pattern', 'Load pattern', '', '', '', 'Load pattern', 'Load pattern', 'Load pattern',
'Load pattern', None, 'Load pattern']
                LoadName = [None, 'DEAD', 'DEAD', 'DEAD', '', None, 'DEAD', 'SIPForm', '', '',
'', 'DEAD', 'Defensas', 'Pavimentação', 'Recapeamento', None, 'Veiculo']
                LoadSF = [0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0,
1.0, 0.0, 1.0]
                Solve = [False, True, True, True, True, False, True, True, False, True, True,
True, True, True, True, True, False, True]
                SaveStiffness = [False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False]
                OutputID = [None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None]
                ItemType = [None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None]
                ItemName = [None, 'DEAD', 'DEAD', 'DEAD', '', None, 'DEAD', 'SIPForm', '', '',
'', 'DEAD', 'Defensas', 'Pavimentação', 'Recapeamento', None, 'Veiculo']
                Notes = ['', 'Add Group Pilares; age at add=3; Include SW Load Pattern=DEAD;
SF=1', 'Add Group Travessas; age at add=3; Include SW Load Pattern=DEAD; SF=1', 'Add Group
Encontro; age at add=3; Include SW Load Pattern=DEAD; SF=1', 'Add Group Links; age at add=3;
Without SW', '', 'Add Group <Ponte>ALL_BUT_SLAB; age at add=1; Include SW Load Pattern=DEAD;
SF=1', "Load all objects in group ' <Ponte>GIRDER_DIAPH' that are already present or are being
added at this completion day and stage.\nLoad Type=Load pattern; Load Name=SIPForm; Scale
Factor=1", 'Set the initial deflection of Group <Ponte>SLAB', 'Add the weight of a top slab
bridge group as equivalent line loads as defined in Concrete Pour <Ponte>CPSlab Permanent
Construction', 'Add the actual slab objects for a top slab bridge group with their full stiffness
```

```
and weight, age=0 day(s), while removing the equivalent line loads from the girders as defined
in Concrete Pour <Ponte>CPSlab Permanent Construction', 'Add Group Lajes de transição; age at
add=3; Include SW Load Pattern=DEAD; SF=1', "Load only objects in group' ALL' that are being
added at this completion day and stage. Objects already present will not be loaded.\nLoad
Type=Load pattern; Load Name=Defensas; Scale Factor=1", "Load only objects in group' ALL' that
are being added at this completion day and stage. Objects already present will not be
loaded.\nLoad Type=Load pattern; Load Name=Pavimentação; Scale Factor=1", "Load all objects in
group ' <Ponte>ALL' that are already present or are being added at this completion day and
stage.\nLoad Type=Load pattern; Load Name=Recapeamento; Scale Factor=1", '', "Load all objects
in group ' <Ponte>ALL' that are already present or are being added at this completion day and
stage.\nLoad Type=Load pattern; Load Name=Veiculo; Scale Factor=1"]
                ret = SapModel.Scheduler.SetScheduleData_1(Name, NumberTasks, Expanded,
IndentLevel, TaskName, Duration, CompletionDate, Dependencies, Operation, IncludeSW, ObjectType,
ObjectName,
                Details, Age, LoadType, LoadName, LoadSF, Solve, SaveStiffness, OutputID,
ItemType, ItemName, Notes)

                Name = 'Cronograma'
                Unloading = 'Unload Entire'
                GeoNonLin = 'P-Delta'
                MaxTotal = 1000
                MaxNull = 200
                MaxIterCS = 200
                MaxIterNR = 200
                ItConvTol = 0.001
                UseEvStep = 'Yes'
                EvLumpTol = 0.01
                LSPerIter = 20
                LSTol = 0.1
                LSStepFact = 1.618
                StageSave = 'Start and End of Each Stage'
                StageMinIns = int()
                StageMinTD = int()
                TimeDepMat = 'Yes'
                TFMaxIter = 100
                TFTol = 0.01
                TFAccelFact = 1
                TFNoStop = 'No'
                ret = SapModel.Scheduler.SetScheduleParameters(Name, Unloading, GeoNonLin,
MaxTotal, MaxNull, MaxIterCS, MaxIterNR, ItConvTol, UseEvStep, EvLumpTol, LSPerIter, LSTol,
LSStepFact,
                StageSave, StageMinIns, StageMinTD, TimeDepMat, TFMaxIter, TFTol, TFAccelFact,
TFNoStop)




        # 17 Atribuição de carregamentos nos elementos
            if Tipo_analise != 'SUPERESTRUTURA':
                # 17.1 Pressão da água nos pilares
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyFrame(f'Pilar (C{f_ck})')
                modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
                wb_pil_ext = xw.Book(modelo_pil_ext)
                ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

                Name = '' # The name of an existing frame object or group, depending on the
value of the ItemType item.
                LoadPat = 'Água'
                MyType = 1
                # This is 1 or 2, indicating the type of distributed load.
                # 1 = Force per unit length
                # 2 = Moment per unit length
                Dir = 5
                # This is an integer between 1 and 11, indicating the direction of the load.
                # 1 = Local 1 axis (only applies when CSys is Local)
                # 2 = Local 2 axis (only applies when CSys is Local)
                # 3 = Local 3 axis (only applies when CSys is Local)
                # 4 = X direction (does not apply when CSys is Local)
                # 5 = Y direction (does not apply when CSys is Local)
                # 6 = Z direction (does not apply when CSys is Local)
                # 7 = Projected X direction (does not apply when CSys is Local)
                # 8 = Projected Y direction (does not apply when CSys is Local)
                # 9 = Projected Z direction (does not apply when CSys is Local)
```

```
                # 10 = Gravity direction (only applies when CSys is Global)
                # 11 = Projected Gravity direction (only applies when CSys is Global)
                Dist1 = 0
                # This is the distance from the I-End of the frame object to the start of the
distributed load.
                # This may be a relative distance (0 <= Dist1 <= 1) or an actual distance,
depending on the value of the RelDist item. [L] when RelDist is False
                Dist2 = 1
                # This is the distance from the I-End of the frame object to the end of the
distributed load.
                # This may be a relative distance (0 <= Dist2 <= 1) or an actual distance,
depending on the value of the RelDist item. [L] when RelDist is False
                Val1 = ws_pil_ext['D57'].value/100 # This is the load value at the start of the
distributed load. [F/L] when MyType is 1 and [FL/L] when MyType is 2
                Val2 = Val1 # This is the load value at the end of the distributed load. [F/L]
when MyType is 1 and [FL/L] when MyType is 2
                CSys = 'GLOBAL' # This is Local or the name of a defined coordinate system. It
is the coordinate system in which the loads are specified.
                RelDist = True # If this item is True, the specified Dist item is a relative
distance, otherwise it is an actual distance.
                Replace = False # If this item is True, all previous distributed loads, if any,
assigned to the specified frame object(s), in the specified load pattern, are deleted before
making the new assignment.
                ItemType = eItemType(2)
                # This is one of the following items in the eItemType enumeration:
                # Object = 0
                # Group = 1
                # SelectedObjects = 2
                # If this item is Object, the assignment is made to the frame object specified
by the Name item.
                # If this item is Group, the assignment is made to all frame objects in the
group specified by the Name item.
                # If this item is SelectedObjects, assignment is made to all selected frame
objects, and the Name item is ignored.
                ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType, Dir, Dist1,
Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)

                ret = SapModel.SelectObj.ClearSelection()

                # 17.2 Empuxo de terra nos pilares

                ret = SapModel.SelectObj.ClearSelection()

                XMin = -1 # The maximum and minimum coordinates of the selection box in the
specified coordinate system.
                XMax = 1
                YMin = -TotalWidthOfDeckSection
                YMax = TotalWidthOfDeckSection
                ZMin = -1000
                ZMax = -400
                DeSelect = False # The item is False if objects are to be selected and True if
they are to be deselected.
                CSys = 'GLOBAL' # The name of the coordinate system in which XMin, XMax, YMin,
YMax, ZMin and ZMax are specified.
                IncludeIntersections = True
                # When this item is True, objects that are inside the box or intersecting the
sides and edges of the box are selected or deselected.
                # When this item is False, only objects that are fully inside the box are
selected or deselected.
                Point = False # Point objects that fall inside the box are only selected or
deselected when this item is True.
                Line = True
                Area = False
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                Name = ''
                LoadPat = 'Empuxo terra'
                MyType = 1
                Dir = 4
                Dist1 = 0
                Dist2 = 1
                Val1 = ws_pil_ext['D115'].value/100
```

```
                Val2 = Val1
                CSys = 'GLOBAL'
                RelDist = True
                Replace = False
                ItemType = eItemType(2)
                # This is one of the following items in the eItemType enumeration:
                # Object = 0
                # Group = 1
                # SelectedObjects = 2
                # If this item is Object, the assignment is made to the frame object specified
by the Name item.
                # If this item is Group, the assignment is made to all frame objects in the
group specified by the Name item.
                # If this item is SelectedObjects, assignment is made to all selected frame
objects, and the Name item is ignored.
                ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType, Dir, Dist1,
Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)
                ret = SapModel.SelectObj.ClearSelection()




            # 18 Definição das Lanes - (cBMLane.SetLaneFromLayoutLine) [myLaneName sempre será
a faixa de tráfego]

            Name = myLaneName #The name for the existing lane defined from a layout line.
            NumberLaneStations = 2 #The number of lane points, that is, the number of items
included in the that lane will be overwritten. LayoutLine, Station, Offset, Width and Radius
arrays.
            LayoutLine = [myLayoutLineName, myLayoutLineName]#A zero-based array containing the
name of the bridge layout line associated with each lane station.
            Station = [0, CTotal]#A zero-based array containing the station at each lane
station. [L]
            Offset = [0, 0]#A zero-based array containing the offset from the layout line to
the center of the lane at each lane station. [L]
            Width = [LarguraRolamento, LarguraRolamento]#A zero-based array containing the the
width of the lane at each lane station. [L]
            Radius = [0,0]#A zero-based array containing the the radius of the lane at each
lane station. A positive radius curves to the right and negative to the left. [L]
            LaneType = 0
            #The lane type (fixed or floating).
            # 0 = Fixed lane
            # 1 = Floating lane set
            if Tipo_analise == 'ENCONTROS':
                LoadGroup = 'Todas lajes'
            else:
                LoadGroup = f'<{myBridgeObjName}>SLAB'
            LeftEdgeIsExterior = False #Indicates whether the left edge of the lane is exterior
or interior.
            RightEdgeIsExterior = False #Indicates whether the right edge of the lane is
exterior or interior.
            MaxDiscAlongLane = 300 #The maximum lane load discretization length along the lane.
[L]
            MaxDiscAcrossLane = 300 #The maximum lane load discretization length across the
lane. [L]
            UseDiscFactorSpan = True #Indicates whether the the DiscFactorSpan item is to apply
to the lane.
            UseDiscFactorLane = True #Indicates whether the the DiscFactorLane item is to apply
to the lane.
            DiscFactorSpan = 4 #The lane load discretization length along the lane will not be
more than 1/x of the span length where x is the DiscFactorSpan value.
            #This item is only applicable when the UseDiscFactorSpan item is True.
            DiscFactorLane = 10 #The lane load discretization length along the lane will not
be more than 1/x of the lane length where x is the DiscFactorLane value.
            #This item is only applicable when the UseDiscFactorLane item is True.
            Color = int()#The color assigned to the lane. (Optional)
            Notes = ''#The notes, if any, assigned to the lane. (Optional)
            ret    =    MyBridgeModeler.Lane.SetLaneFromLayoutLine(Name,    NumberLaneStations,
LayoutLine,   Station,   Offset,   Width,   Radius,   LaneType,   LoadGroup,   LeftEdgeIsExterior,
RightEdgeIsExterior,
            MaxDiscAlongLane,    MaxDiscAcrossLane,    UseDiscFactorSpan,    UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

            # 19 Ajustes finais do modelo
```

```
            # 19.1 Rotulação diafragmas

            ret = SapModel.SelectObj.Group('Diafragmas', False)

            Name = '' # The name of an existing frame object or group, depending on the value
of the ItemType item.
            ii = [False, False, False, False, True, True]
            jj = [False, False, False, False, True, True]
            # These are arrays of six booleans indicating the I-End and J-End releases for the
frame object.
            # ii(0) and jj(0) = U1 release
            # ii(1) and jj(1) = U2 release
            # ii(2) and jj(2) = U3 release
            # ii(3) and jj(3) = R1 release
            # ii(4) and jj(4) = R2 release
            # ii(5) and jj(5) = R3 release
            StartValue = [float()] * 6
            EndValue = [float()] * 6
            # These are arrays of six values indicating the I-End and J-End partial fixity
springs for the frame object.
            # StartValue(0) and EndValue(0) = U1 partial fixity [F/L]
            # StartValue(1) and EndValue(1) = U2 partial fixity [F/L]
            # StartValue(2) and EndValue(2) = U3 partial fixity [F/L]
            # StartValue(3) and EndValue(3) = R1 partial fixity [FL/rad]
            # StartValue(4) and EndValue(4) = R2 partial fixity [FL/rad]
            # StartValue(5) and EndValue(5) = R3 partial fixity [FL/rad]
            ItemType = eItemType(2)
            # This is one of the following items in the eItemType enumeration:
            # Object = 0
            # Group = 1
            # SelectedObjects = 2
            # If this item is Object, the assignment is made to the frame object specified by
the Name item.
            # If this item is Group, the assignment is made to all frame objects in the group
specified by the Name item.
            # If this item is SelectedObjects, assignment is made to all selected frame objects,
and the Name item is ignored.
            ret = SapModel.FrameObj.SetReleases(Name, ii, jj, StartValue, EndValue, ItemType)

        #     # Combinações de ações

        #     Comb_dict = {
        #             'Carga permanente': {'Type': 'Load Combo', 'ELU' : [1.35], 'ELU-fad' :
1, 'ELS-W': 1, 'EXC' : 1.1},
        #              'Empuxo terra': {'Type': 'Load Case', 'ELU' : [1.35], 'ELU-fad' : 1,
'ELS-W': 1, 'EXC' : 1.1},
        #             'Retração e fluência': {'Type': 'Load Case', 'ELU' : [1.2], 'ELU-fad' :
0, 'ELS-W': 0, 'EXC' : 0},
        #              'Protensão': {'Type': 'Load Case', 'ELU' : [1.2], 'ELU-fad' : 0, 'ELS-
W': 0, 'EXC' : 0},
        #              'TB-450': {'Type': 'Load Combo', 'ELU' : [CI * 1.5, CI * 1.05], 'ELU-
fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
        #             'Frenagem e aceleracao': {'Type': 'Load Case', 'ELU' : [CI * 1.5, CI *
1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
        #             'Empuxo terra carga móvel': {'Type': 'Load Case', 'ELU' : [CI * 1.5, CI
* 1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
        #            'Vento': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-fad' : 0, 'ELS-
W': 0, 'EXC' : 0.6},
        #             'Água': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-fad' : 0, 'ELS-
W': 0, 'EXC' : 0.6},
        #            'ENV-TEMP': {'Type': 'Load Combo', 'ELU' : [1.2, 0.72], 'ELU-fad' : 0.3,
'ELS-W': 0.3, 'EXC' : 0.6},
        #              'Impacto': {'Type': 'Load Case', 'ELU' : 0, 'ELU-fad' : 0, 'ELS-W': 0,
'EXC' : 1}
        #             }

        # AV = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água',
'ENV-TEMP'] # Lista das ações variáveis

        # Name = 'COMB1 - ELU (CM)' # Carga móvel como ação variável principal
        # ComboType = 0
        # ret = SapModel.RespCombo.Add(Name, ComboType)
```

```
        # CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência', 'Protensão', 'TB-
450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água', 'ENV-TEMP']
        # AVP = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel']
        # for loadcase in CName:
        #       CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load Case' else
eCNameType(1)
        #     SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase not in AV else
Comb_dict[loadcase]['ELU'][1]
        #     ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

            # 19.2 User Discretization Points
            Name = myBridgeObjName # The name of a bridge object.
            SpanNames = [SpanName[0], SpanName[0], SpanName[1]] # A zero-based, one-dimensional
array of the span names associated with each discretization point. If a span name is not
recognized the discretization point will be skipped with no error indicated.
            Distance = [e_a, x_S1, CVão/2 - CPerfil2] # A zero-based, one-dimensional array of
the distances of each discretization point from the start of the specified span. The distance
cannot specify a point closer than 100 times the specified auto merge tolerance to either end
of the span. if it is the discretization point will be skipped with no error indicated. [L]
            Bearing = ['Default']*3 # A zero-based, one-dimensional array of the bearing of the
mesh line across the bridge at the discretization point. The bearing may be specified as Default
or a valid bearing string.
            ret  =  MyBridgeModeler.BridgeObject.SetUserDiscretizationPoints(Name,  SpanNames,
Distance, Bearing)
            x_girder_forces = [e_a, x_S1, (CVão/2)]
            x_girder_forces = [elemento / 100 for elemento in x_girder_forces]
            x_diaphragm_forces = [elemento / 100 for elemento in x_Diafragmas]
            x_forces = list(np.unique(x_girder_forces + x_diaphragm_forces))

            # 19.3 Update para discretização com os user discretization points
            Name = myBridgeObjName
            Action = 1
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)

            # 19.4 Pegar os esforços e tabelas em kN.m
            ret = SapModel.SetPresentUnits(eUnits(kN_m_C))

            # 19.5 Salvar e rodar modelo
            ret = SapModel.File.Save()
            Name = ''
            Run = True # If this item is True, the specified load case is to be run.
            All = True # If this item is True, the run flag is set as specified by the Run item
for all load cases, and the Name item is ignored.
            ret = SapModel.Analyze.SetRunCaseFlag(Name, Run, All)
            ret = SapModel.Analyze.RunAnalysis()




            # 20 Definição do grupo de longarinas (por algum motivo, a análise não estava
convergindo caso o grupo fosse criado antes)

            ret = SapModel.SelectObj.ClearSelection()
            ret = SapModel.GroupDef.SetGroup('Longarinas')
            ret = SapModel.SelectObj.PropertyFrame('Perfil 1', False)
            ret = SapModel.SelectObj.PropertyFrame('Perfil 2', False)
            ret = SapModel.FrameObj.SetGroupAssign('', 'Longarinas', False, eItemType(2))
            ret = SapModel.SelectObj.ClearSelection()

            # 20.1 Pegar os textos dos frames e os stations relativos às longarinas (filtrar
os frames para gerar menos dados)
            ret  = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0,  0,  0,  False,  1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            TableKey = 'Frame Bridge Object Flags'
            FieldKeyList = ''
            GroupName = 'Longarinas'
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
            ret   =   SapModel.DatabaseTables.GetTableForDisplayArray(TableKey,   FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)
```

```
            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))

            df = pd.DataFrame(ListadeListasDosResultados, columns=ColumnLabels)
            df = df.replace({',' : '.'}, regex = True) # Trocar vírgula por ponto em todas as
colunas
            dados_numéricos = ['StartSta', 'EndSta'] # Colunas com dados numéricos
            for dado in dados_numéricos:
                df[dado] = pd.to_numeric(df[dado]) # Mudar string para valor numérico

            girders_stations = df.drop(columns = ['AutoBridge', 'BridgeObj', 'BOSpan',
'BOItemType', 'DiaphType', 'BOBent', 'BOBentCol', 'BORailTrack', 'BOEndI', 'BOEndJ',
'BOSpanEndI', 'BOSpanEndJ'])

            # 20.2 Configuração inicial de output das tabelas

            OutputCase = [] # Lista para filtrar o dataframe posteriormente

            ret = SapModel.LoadPatterns.GetNameList(NumberNames = int(), MyName = [])
            LoadPatternList = list(ret[2])

            ret = SapModel.LoadCases.GetNameList_1(NumberNames = int(), MyName = [])
            LoadCasesList = list(ret[2])
            OutputCase = LoadCasesList # Lista para filtrar o dataframe posteriormente

            ret = SapModel.RespCombo.GetNameList(NumberNames = int(), MyName = [])
            LoadCombinationsList = list(ret[2])
            OutputCase.extend(LoadCombinationsList)

            # 20.3 Extração dos esforços das longarinas

            # 20.3.1 Extração dos esforços solicitantes

            ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            ret = SapModel.DatabaseTables.SetLoadPatternsSelectedForDisplay(LoadPatternList)
            ret = SapModel.DatabaseTables.SetLoadCasesSelectedForDisplay(LoadCasesList)
            ret                                                                    =
SapModel.DatabaseTables.SetLoadCombinationsSelectedForDisplay(LoadCombinationsList)

            TableKey = 'Bridge Object Girder Forces'
            FieldKeyList = ''
            GroupName = 'Longarinas' # Gerar a tabela apenas para o grupo de longarinas
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
            ret = SapModel.DatabaseTables.GetTableForDisplayArray(TableKey, FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

            # Criação do dataframe e conversão dos tipos de dados (associar todos os resultados
dos esforços aos nós que definem as barras)
            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
            girder_forces_table = pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
            girder_forces_table = girder_forces_table.replace({',' : '.'}, regex = True) #
Trocar vírgula por ponto em todas as colunas
            dados_numéricos = ['Station', 'P', 'V2', 'V3', 'T', 'M2', 'M3'] # Colunas com dados
numéricos
            for dado in dados_numéricos:
                girder_forces_table[dado] = pd.to_numeric(girder_forces_table[dado]) # Mudar
string para valor numérico

            # 20.4 Extração dos esforços dos perfis isolados

            # 20.4.1 Extração dos esforços solicitantes

            ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)
```

```
            ret = SapModel.DatabaseTables.SetLoadPatternsSelectedForDisplay(LoadPatternList)
            ret = SapModel.DatabaseTables.SetLoadCasesSelectedForDisplay(LoadCasesList)
            ret                                                                    =
SapModel.DatabaseTables.SetLoadCombinationsSelectedForDisplay(LoadCombinationsList)

            TableKey = 'Element Forces - Frames'
            FieldKeyList = ''
            GroupName = 'Longarinas' # Gerar a tabela apenas para o grupo de longarinas
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
            ret   =   SapModel.DatabaseTables.GetTableForDisplayArray(TableKey,   FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

            # Criação do dataframe e conversão dos tipos de dados (associar todos os resultados
dos esforços aos nós que definem as barras)
            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
            beam_forces_table = pd.DataFrame(ListadeListasDosResultados, columns=ColumnLabels)
            beam_forces_table = beam_forces_table.replace({',' : '.'}, regex = True) # Trocar
vírgula por ponto em todas as colunas

            df_beam_forces = pd.merge(beam_forces_table, girders_stations, on='Frame')
            dados_numéricos = ['StepNum', 'P', 'V2', 'V3', 'T', 'M2', 'M3', 'ElemStation',
'StartSta', 'EndSta'] # Colunas com dados numéricos
            for dado in dados_numéricos:
                df_beam_forces[dado] = pd.to_numeric(df_beam_forces[dado]) # Mudar string para
valor numérico

            # 20.4.2 Extração das tensões solicitantes

            ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0, False, 1,
1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

            ret = SapModel.DatabaseTables.SetLoadPatternsSelectedForDisplay(LoadPatternList)
            ret = SapModel.DatabaseTables.SetLoadCasesSelectedForDisplay(LoadCasesList)
            ret                                                                    =
SapModel.DatabaseTables.SetLoadCombinationsSelectedForDisplay(LoadCombinationsList)

            TableKey = 'Element Stresses - Frames'
            FieldKeyList = ''
            GroupName = 'Longarinas' # Gerar a tabela apenas para o grupo de longarinas
            TableVersion = int()
            FieldKeysIncluded = ['']
            NumberRecords = int()
            TableData = []
            ret   =   SapModel.DatabaseTables.GetTableForDisplayArray(TableKey,   FieldKeyList,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)

            ColumnLabels = list(ret[3])
            Resultados = list(ret[5])

            # Criação do dataframe e conversão dos tipos de dados (associar todos os resultados
dos esforços aos nós que definem as barras)
            ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
            beam_stresses_table          =          pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
            beam_stresses_table = beam_stresses_table.replace({',' : '.'}, regex = True) #
Trocar vírgula por ponto em todas as colunas

            df_beam_stresses = pd.merge(beam_stresses_table, girders_stations, on='Frame')
            dados_numéricos = ['StepNum', 'Point', 'X2', 'X3', 'S11', 'S12', 'S13', 'SMax',
'SMin', 'SVM', 'EndSta'] # Colunas com dados numéricos
            for dado in dados_numéricos:
                df_beam_stresses[dado] = pd.to_numeric(df_beam_stresses[dado]) # Mudar string
para valor numérico

            # 20.5 Planilha 4-Esforços

            # 20.5.1 Filtragem dos resultados das tabelas importadas do CSiBridge
```

```
            # Longarinas (seção composta)
            df_girder_forces                                                    =
girder_forces_table[girder_forces_table["Station"].isin(x_forces)]
            df_girder_forces  =  df_girder_forces.drop(columns  =  ['BridgeObj',  'BridgeCut',
'Span', 'GirderDist', 'CaseType', 'GlobalX', 'GlobalY', 'GlobalZ'])

            # Seção isolada

            df_beam_forces  =  df_beam_forces[df_beam_forces["EndSta"].isin(x_forces)]  #  Por
convenção, serão utilizados os elementos de barra que se encerram no ponto especificado
            df_beam_forces = df_beam_forces.drop(columns = ['StartSta', 'Station', 'CaseType',
'FrameElem'])

            # 20.5.2 Alimentação da planilha

            load_cases = ['GirdersAndDiaphragms', 'DeckPlus', 'ConcPour Temporary', 'Retração
e Fluência', 'Temperatura gradiente pos', 'Temperatura gradiente neg']
            girder_bendingmoment_beam_cells = cells_list('E', len(x_girder_forces), 13, 14, 20,
23)
            girder_shear_beam_cells = cells_list('H', len(x_girder_forces), 13, 14, 20, 23)
            diaphragm_shear_beam_cells = cells_list('D', len(x_diaphragm_forces), 30, 31, 37,
40)

            for index, element in enumerate(load_cases):
                # ESFORÇOS LONGARINAS ISOLADAS
                for i in range(len(x_girder_forces)):
                    if element == 'Temperatura gradiente pos':
                        ws_esforcos[girder_bendingmoment_beam_cells[i][index]].value        =
df_beam_forces[(df_beam_forces["EndSta"] == x_girder_forces[i]) &
                        (df_beam_forces["ElemStation"] > 0) & (df_beam_forces["OutputCase"] ==
element)]['M3'].min()
                        ws_esforcos[girder_shear_beam_cells[i][index]].value                =
df_beam_forces[(df_beam_forces["EndSta"] == x_girder_forces[i]) &
                        (df_beam_forces["ElemStation"] > 0) & (df_beam_forces["OutputCase"] ==
element)]['V2'].abs().max()

                    else:
                        ws_esforcos[girder_bendingmoment_beam_cells[i][index]].value        =
df_beam_forces[(df_beam_forces["EndSta"] == x_girder_forces[i]) &
                        (df_beam_forces["ElemStation"] > 0) & (df_beam_forces["OutputCase"] ==
element)]['M3'].max()
                        ws_esforcos[girder_shear_beam_cells[i][index]].value                =
df_beam_forces[(df_beam_forces["EndSta"] == x_girder_forces[i]) &
                        (df_beam_forces["ElemStation"] > 0) & (df_beam_forces["OutputCase"] ==
element)]['V2'].abs().max()

                # ESFORÇOS DIAFRAGMAS
                for j in range(len(x_diaphragm_forces)):
                    ws_esforcos[diaphragm_shear_beam_cells[j][index]].value                 =
df_beam_forces[(df_beam_forces["EndSta"] == x_diaphragm_forces[j]) &
                    (df_beam_forces["ElemStation"]  >  0)  &  (df_beam_forces["OutputCase"]  ==
element)]['V2'].abs().max()

            # Girders
            load_cases = ['Defensas', 'Pavimentação', 'Recapeamento', 'TB-450', 'TB-450']
            girder_bendingmoment_beam_cells = cells_list('E', len(x_girder_forces), 15, 16, 17,
19)
            girder_shear_beam_cells = cells_list('H', len(x_girder_forces), 15, 16, 17, 19)
            diaphragm_shear_beam_cells = cells_list('D', len(x_diaphragm_forces), 32, 33, 34,
36)

            for index, element in enumerate(load_cases):
                # ESFORÇOS LONGARINAS COM SEÇÃO COMPOSTA
                for i in range(len(x_girder_forces)):
                    if element == 'TB-450' and index == 3:
                        steptype = 'Max'
                        ws_esforcos[girder_bendingmoment_beam_cells[i][index]].value        =
df_girder_forces[(df_girder_forces["Station"] == x_girder_forces[i]) &
                        (df_girder_forces["OutputCase"]          ==          element)          &
(df_girder_forces["StepType"] == steptype)]['M3'].max()
                        ws_esforcos[girder_shear_beam_cells[i][index]].value                =
df_girder_forces[(df_girder_forces["Station"] == x_girder_forces[i]) &
                        (df_girder_forces["OutputCase"]          ==          element)          &
(df_girder_forces["StepType"] == steptype)]['V2'].abs().max()
```

```
                    elif element == 'TB-450' and index == 4:
                        steptype = 'Min'
                        ws_esforcos[girder_bendingmoment_beam_cells[i][index]].value         =
df_girder_forces[(df_girder_forces["Station"] == x_girder_forces[i]) &
                        (df_girder_forces["OutputCase"]          ==          element)          &
(df_girder_forces["StepType"] == steptype)]['M3'].min()
                        ws_esforcos[girder_shear_beam_cells[i][index]].value                 =
df_girder_forces[(df_girder_forces["Station"] == x_girder_forces[i]) &
                        (df_girder_forces["OutputCase"]          ==          element)          &
(df_girder_forces["StepType"] == steptype)]['V2'].abs().max()
                    else:
                        ws_esforcos[girder_bendingmoment_beam_cells[i][index]].value         =
df_girder_forces[(df_girder_forces["Station"] == x_girder_forces[i]) &
                        (df_girder_forces["OutputCase"] == element)]['M3'].max()
                        ws_esforcos[girder_shear_beam_cells[i][index]].value                 =
df_girder_forces[(df_girder_forces["Station"] == x_girder_forces[i]) &
                        (df_girder_forces["OutputCase"] == element)]['V2'].abs().max()

                # ESFORÇOS DIAFRAGMAS
                for j in range(len(x_diaphragm_forces)):
                    if element == 'TB-450' and index == 3:
                        steptype = 'Max'
                        ws_esforcos[diaphragm_shear_beam_cells[j][index]].value              =
df_girder_forces[(df_girder_forces["Station"] == x_diaphragm_forces[j]) &
                        (df_girder_forces["OutputCase"]          ==          element)          &
(df_girder_forces["StepType"] == steptype)]['V2'].abs().max()

                    elif element == 'TB-450' and index == 4:
                        steptype = 'Min'
                        ws_esforcos[diaphragm_shear_beam_cells[j][index]].value              =
df_girder_forces[(df_girder_forces["Station"] == x_diaphragm_forces[j]) &
                        (df_girder_forces["OutputCase"]          ==          element)          &
(df_girder_forces["StepType"] == steptype)]['V2'].abs().max()

                    else:
                        ws_esforcos[diaphragm_shear_beam_cells[j][index]].value              =
df_girder_forces[(df_girder_forces["Station"] == x_diaphragm_forces[j]) &
                        (df_girder_forces["OutputCase"] == element)]['V2'].abs().max()

            # 20.6 Planilha 6-VSM-C

            # Correspondência dos pontos de tensão do perfil

            bottom_left = 1
            bottom_center = 2
            bottom_right = 3
            top_left = 6
            top_center = 7
            top_right = 8

            # Momento devido à retração no perfil isolado
            ws_mista['J182'].value = df_beam_forces[(df_beam_forces["OutputCase"] == 'Retração
e Fluência') & (df_beam_forces["StepNum"] == 9)]['M3'].abs().max()

            # Tensões no perfil 1 devido à retração e fluência
            ws_mista['I205'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Retração e Fluência') & (df_beam_stresses["StepNum"] == 9) &
            (df_beam_stresses["Point"]  ==  top_center)  &  (df_beam_stresses["StartSta"]  >=
x_S1/100) & (df_beam_stresses["EndSta"] <= x_S2/100)]['S11'].min()/1000
            ws_mista['J205'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Retração e Fluência') & (df_beam_stresses["StepNum"] == 9) &
            (df_beam_stresses["Point"]  ==  bottom_center)  &  (df_beam_stresses["StartSta"]  >=
x_S1/100) & (df_beam_stresses["EndSta"] <= x_S2/100)]['S11'].max()/1000

            # Tensões no perfil 2 devido à retração e fluência
            ws_mista['I206'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Retração e Fluência') & (df_beam_stresses["StepNum"] == 9) &
            (df_beam_stresses["Point"]  ==  top_center)  &  ((df_beam_stresses["StartSta"]  <=
x_S1/100) | (df_beam_stresses["EndSta"] >= x_S2/100))]['S11'].min()/1000
            ws_mista['J206'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Retração e Fluência') & (df_beam_stresses["StepNum"] == 9) &
            (df_beam_stresses["Point"]  ==  bottom_center)  &  ((df_beam_stresses["StartSta"]  <
x_S1/100) | (df_beam_stresses["EndSta"] > x_S2/100))]['S11'].max()/1000

            # Esforços nos perfis isolados devido à variação de temperatura
```

```
            ws_mista['I324'].value    =    df_beam_forces[(df_beam_forces["OutputCase"]    ==
'Temperatura gradiente neg')]['P'].min()
            ws_mista['J324'].value    =    df_beam_forces[(df_beam_forces["OutputCase"]    ==
'Temperatura gradiente pos')]['P'].max()
            ws_mista['I325'].value    =    df_beam_forces[(df_beam_forces["OutputCase"]    ==
'Temperatura gradiente neg')]['M3'].min()
            ws_mista['J325'].value    =    df_beam_forces[(df_beam_forces["OutputCase"]    ==
'Temperatura gradiente pos')]['M3'].max()

            # Tensões no perfil 1 devido à variação negativa de temperatura
            ws_mista['I356'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Temperatura gradiente neg') &
            (df_beam_stresses["Point"]  ==  top_center)  &  (df_beam_stresses["StartSta"]  >=
x_S1/100) & (df_beam_stresses["EndSta"] <= x_S2/100)]['S11'].min()/1000
            ws_mista['J356'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Temperatura gradiente neg') &
            (df_beam_stresses["Point"]  == bottom_center)  &  (df_beam_stresses["StartSta"]  >=
x_S1/100) & (df_beam_stresses["EndSta"] <= x_S2/100)]['S11'].max()/1000

            # Tensões no perfil 2 devido à variação negativa de temperatura
            ws_mista['I357'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Temperatura gradiente neg') &
            (df_beam_stresses["Point"]  ==  top_center)  &  ((df_beam_stresses["StartSta"]  <=
x_S1/100) | (df_beam_stresses["EndSta"] >= x_S2/100))]['S11'].min()/1000
            ws_mista['J357'].value   =   df_beam_stresses[(df_beam_stresses["OutputCase"]   ==
'Temperatura gradiente neg') &
            (df_beam_stresses["Point"]  == bottom_center)  &  ((df_beam_stresses["StartSta"]  <
x_S1/100) | (df_beam_stresses["EndSta"] > x_S2/100))]['S11'].max()/1000


            ret = SapModel.File.Save()
            wb.save(os.path.join(main_folder, Tipologia, f'{Tipologia}.xlsx'))
            wb.app.quit()
            SapModel = None
            mySAPObject = None
            MyBridgeModeler = None
```

```
        # ESSE PROGRAMA FOI FEITO PARA SER USADO EM CONJUNTO COM O 'composite-main'.py APÓS REALIZADA A ANÁLISE DA SUPERESTRUTURA. ELE IRA CRIAR OS MODELOS DOS ENCONTROS COM APENAS UM VÃO, TOMANDO-SE O MODELO DE SUPERESTRUTURA GERADO.
        # INPUTAR AS TIPOLOGIAS A SEREM RODADAS NO VETOR TIPOLOGIAS

        import os
        import sys
        import clr
        import pandas as pd
        from enum import Enum
        import numpy as np
        import xlwings as xw
        import math

        clr.AddReference("System.Runtime.InteropServices")
        clr.AddReference(r'C:\Program Files\Computers and Structures\CSiBridge 26\CSiBridge1.dll')
        from System.Runtime.InteropServices import Marshal
        from CSiBridge1 import *

        # Funções definidas

        from utils import *

        #set the following flag to True to execute on a remote computer

        Remote = False


        #if the above flag is True, set the following variable to the hostname of the remote computer

        #remember that the remote computer must have CSiBridge installed and be running the CSiAPIService.exe

        RemoteComputer = "SpareComputer-DT"



        #set the following flag to True to attach to an existing instance of the program

        #otherwise a new instance of the program will be started

        AttachToInstance = True



        #set the following flag to True to manually specify the path to CSiBridge.exe

        #this allows for a connection to a version of CSiBridge other than the latest installation

        #otherwise the latest installed version of CSiBridge will be launched

        SpecifyPath = True

        #if the above flag is set to True, specify the path to CSiBridge below

        ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 26\CSiBridge.exe'

        #full path to the model

        #set it to the desired path of your model

        #if executing remotely, ensure that this folder already exists on the remote computer

        #the below command will only create the folder locally

        main_folder = os.path.dirname(os.path.abspath(__file__))
        APIPath = main_folder

        if not os.path.exists(APIPath):

            try:
```

```
                os.makedirs(APIPath)

            except OSError:

                pass

        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'composite-variables.xlsx')
        variables    =    pd.read_excel(variables_data_file,    sheet_name="Main",    header=3,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-MAC-A-20-P', 'S-MAC-A-25-P', 'S-MAC-A-30-P', 'S-MAC-A-35-P', 'S-MAC-
A-20-S', 'S-MAC-A-25-S', 'S-MAC-A-30-S', 'S-MAC-A-35-S', 'S-MAC-B-20-P', 'S-MAC-B-25-P',
        'S-MAC-B-30-P', 'S-MAC-B-35-P', 'S-MAC-B-20-S', 'S-MAC-B-25-S', 'S-MAC-B-30-S', 'S-
MAC-B-35-S']

        # ['S-MAC-A-20-P', 'S-MAC-A-25-P', 'S-MAC-A-30-P', 'S-MAC-A-35-P', 'S-MAC-A-20-S', 'S-
MAC-A-25-S', 'S-MAC-A-30-S', 'S-MAC-A-35-S', 'S-MAC-B-20-P', 'S-MAC-B-25-P',
        # 'S-MAC-B-30-P', 'S-MAC-B-35-P', 'S-MAC-B-20-S', 'S-MAC-B-25-S', 'S-MAC-B-30-S', 'S-
MAC-B-35-S']

        Tipo_analise = 'ENCONTROS' # Tipos contemplados: AQUI SERÁ APENAS 'ENCONTROS'

        for Tipologia in Tipologias: # Rodar todo o código com base em um elemento da lista
Tipologias
            xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
            GeneralPath = os.path.join(main_folder, Tipologia)
            os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à tipologia. O
parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
            ModelPath  =  os.path.join(GeneralPath,  f'{Tipologia}  (SUPERESTRUTURA).bdb')  #
Caminho do modelo .bdb

            #create API helper object

            helper = cHelper(Helper())

            if AttachToInstance:

                #attach to a running instance of CSiBridge

                try:

                    #get the active CSiBridge object

                    if Remote:

                        mySAPObject        =        cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                    else:

                        mySAPObject = cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                except:

                    print("No running instance of the program found or failed to attach.")

                    sys.exit(-1)


            else:

                if SpecifyPath:

                    try:

                        #'create an instance of the CSiBridge object from the specified path

                        if Remote:

                            mySAPObject    =    cOAPI(helper.CreateObjectHost(RemoteComputer,
ProgramPath))
```

```
                    else:

                        mySAPObject = cOAPI(helper.CreateObject(ProgramPath))

                except :

                    print("Cannot start a new instance of the program from " + ProgramPath)

                    sys.exit(-1)

            else:


                try:

                    #create an instance of the CSiBridge object from the latest installed
CSiBridge

                    if Remote:

                        mySAPObject = cOAPI(helper.CreateObjectProgIDHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                    else:

                        mySAPObject                                                    =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))


                except:

                    print("Cannot start a new instance of the program.")

                    sys.exit(-1)


            #start CSiBridge application
            mySAPObject.ApplicationStart()

        StartX = 0

        #**********************************      INÍCIO      VARIÁVEIS      DE      INPUT
*****************************************#
        f_ck = 35
        Alturas_pilarescentrais = [400, 600, 800] # Análises de MESOESTRUTURA
        NVãos = 1
        df_ponte = variables[(variables['Tipologia'] == Tipologia)]

        # SEÇÃO TRANSVERSAL
        CVão = df_ponte['Vão (cm)'].iloc[0]
        NVãos = 1
        CTotal = NVãos * CVão # Caso de pontes com múltiplos vãos
        TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
        NLongarinasPorVao = int(df_ponte['Nlong'].iloc[0])
        NumberInteriorGirders = NLongarinasPorVao - 2 #The number of interior girders
        OverhangLength = df_ponte['Balanço (cm)'].iloc[0] #balanço laje/longarina #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
        Espaçamento_Long             =             (TotalWidthOfDeckSection             -
2*OverhangLength)/(NLongarinasPorVao-1)
        NDiafragmas = int(df_ponte['Ndiaf'].iloc[0])
        LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)

        # AÇO ESTRUTURAL A588Gr50
        fy_A588Gr50 = 345
        fu_A588Gr50 = 480
        fye_A588Gr50 = fy_A588Gr50/1.15
        fue_A588Gr50 = fye_A588Gr50/1.15
        df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia
```

```
            # DIMENSÕES DOS ELEMENTOS ESTRUTURAIS

                # PERFIL 1
            t3_1 = df_ponte['t3_1'].iloc[0] # The section depth
            t2_1 = df_ponte['t2_1'].iloc[0] # The top flange width
            tf_1 = df_ponte['tf_1'].iloc[0] # The top flange thickness
            tw_1 = df_ponte['tw_1'].iloc[0] # The web thickness
            t2b_1 = df_ponte['t2b_1'].iloc[0] # The bottom flange width
            tfb_1 = df_ponte['tfb_1'].iloc[0] # The bottom flange thickness
            FilletRadius_1 = df_ponte['rad_1'].iloc[0] # The fillet radius
            h_long = t3_1

                # PERFIL 2
            t3_2 = df_ponte['t3_2'].iloc[0] # The section depth
            t2_2 = df_ponte['t2_2'].iloc[0] # The top flange width
            tf_2 = df_ponte['tf_2'].iloc[0] # The top flange thickness
            tw_2 = df_ponte['tw_2'].iloc[0] # The web thickness
            t2b_2 = df_ponte['t2b_2'].iloc[0] # The bottom flange width
            tfb_2 = df_ponte['tfb_2'].iloc[0] # The bottom flange thickness
            FilletRadius_2 = df_ponte['rad_2'].iloc[0] # The fillet radius

                # DIAFRAGMAS EXTREMOS
            t3diaf_1 = df_ponte['t3diaf_1'].iloc[0] # The section depth
            t2diaf_1 = df_ponte['t2diaf_1'].iloc[0] # The top flange width
            tfdiaf_1 = df_ponte['tfdiaf_1'].iloc[0] # The top flange thickness
            twdiaf_1 = df_ponte['twdiaf_1'].iloc[0] # The web thickness
            t2bdiaf_1 = df_ponte['t2bdiaf_1'].iloc[0] # The bottom flange width
            tfbdiaf_1 = df_ponte['tfbdiaf_1'].iloc[0] # The bottom flange thickness
            FilletRadiusdiaf_1 = df_ponte['raddiaf_1'].iloc[0] # The fillet radius

                # DIAFRAGMAS INTERMEDIÁRIOS
            t3diaf_2 = df_ponte['t3diaf_2'].iloc[0] # The section depth
            t2diaf_2 = df_ponte['t2diaf_2'].iloc[0] # The top flange width
            tfdiaf_2 = df_ponte['tfdiaf_2'].iloc[0] # The top flange thickness
            twdiaf_2 = df_ponte['twdiaf_2'].iloc[0] # The web thickness
            t2bdiaf_2 = df_ponte['t2bdiaf_2'].iloc[0] # The bottom flange width
            tfbdiaf_2 = df_ponte['tfbdiaf_2'].iloc[0] # The bottom flange thickness
            FilletRadiusdiaf_2 = df_ponte['raddiaf_2'].iloc[0] # The fillet radius

                # PILARES
            t3_pilar = df_ponte['Dp'].iloc[0]

                # TRAVESSAS
            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]
            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

                # TABULEIRO

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - t2_1

                # APARELHO DE APOIO

            a_ap = df_ponte['aA'].iloc[0]
            b_ap = df_ponte['bA'].iloc[0]
            ShoreA_ap = 60 # Dureza Shore A do aparelho de apoio
            G_ap = 1000 # Módulo de elasticidade transversal do aparelho de apoio em kN/m²
            sigmaadm_ap = 15000 # Tensão admissível no aparelho de apoio
            hs_ap = 4 # Espessura das chapas de fretagem em mm
            hi_ap = 12 # Espessura das camadas intermediárias do Neoprene
            cv_ap = 2.5 # Cobrimento vertical
            ch_ap = 4 # Cobrimento horizontal
            n_ap = df_ponte['nA'].iloc[0] # Número de camadas do aparelho de apoio
            h_ap = (n_ap * hi_ap + 2* cv_ap + (n_ap+1)*hs_ap)/10 # Pré-dimensionamento da altura
do aparelho de apoio
            K_ap = a_ap/100 * b_ap/100 * G_ap/(cv_ap*2 + hi_ap*n_ap)*10 # Rigidez do aparelho
de apoio na direção longitudinal e transversal à longarina
            d_ap =  df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta

                # LAJE DE TRANSIÇÃO
```

```
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

                # MURO
            AlturaM = df_ponte['AlturaM'].iloc[0]

                # ALA
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

            # DEFINIÇÃO DE BENTS

            # EXTREMOS
            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = 300 if Tipo_analise == 'ENCONTROS' else df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                Distance_ext = [Distextp, C_travessaext - Distextp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, C_travessaext/2, C_travessaext - Distextp]
                Height_ext = [Hextp, Hextp, Hextp]

            # INTERMEDIÁRIOS
            C_travessaint = df_ponte['Cint'].iloc[0]
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
            SupportType_int = 1
            if N_pilaresint == 2:
                Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # ASPECTOS DO PROJETO
            e_a = 40 # Distância entre o início do perfil da longarina e o centro do apoio
            CEfetivo = CVão - 2*e_a # Vão efetivo da ponte
            NTrechos = 3 # Número de perfis para vencer um vão

                # PERFIS
            NPerfil1 = NTrechos if NTrechos <= 2 else 1 if NTrechos == 3 else 2
            NPerfil2 = 0 if NTrechos <= 2 else 2
            CPerfil1 = CVão if NTrechos == 1 else CVão/NTrechos if NTrechos == 2 and
CVão/NTrechos <= 1190 else 1190
            NEmendas = math.floor(CVão/1190)
            ejunta = 5
            CPerfil2 = (CVão - CPerfil1)/NPerfil2
            x_S1 = 0 if NTrechos == 1 else CPerfil1 if NTrechos == 2 else CPerfil2 # Posição
da Emenda 1
            x_S2 = 0 if NTrechos <= 2 else CVão - CPerfil2 if NTrechos == 3 else CVão/2 #
Posição da Emenda 2

                # DIAFRAGMAS

            eet = 1.25 # Espessura dos enrijecedores transversais
            let = 15 # Largura dos enrijecedores transversais
            eea = 1.6 # Espessura dos enrijecedores de apoio
            lea = 20 # Largura dos enrijecedores de apoio
```

```
            # DIAFRAGMAS
        DistDiafragmas = CEfetivo/(NDiafragmas-1)
        x_Diafragmas = [e_a]
        x_Acumulado = e_a
        while x_Acumulado <= CEfetivo:
            x_Acumulado += DistDiafragmas
            x_Diafragmas.append(x_Acumulado)

        SpanLength = [] # A zero-based array containing the length of each span. [L]
        SpanName = []
        for i in range(NVãos):
            if NTrechos <= 2:
                SpanLength.extend([CPerfil1, CPerfil1]) #A zero-based array containing the
length of each span. [L]
                SpanName.extend([f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'])
            elif NTrechos == 3:
                SpanLength.extend([CPerfil2, CPerfil1, CPerfil2]) #A zero-based array
containing the length of each span. [L]
                SpanName.extend([f'Perfil 2.{i+1}.1', f'Perfil 1.{i+1}', f'Perfil
2.{i+1}.2'])
            else:
                SpanLength.extend([CPerfil2, CPerfil1, CPerfil1, CPerfil2]) #A zero-based
array containing the length of each span. [L]
                SpanName.extend([f'Perfil 2.{i+1}.1', f'Perfil 1.{i+1}.1', f'Perfil
1.{i+1}.2', f'Perfil 2.{i+1}.2'])

        SpanStartStations = [0]
        SpanAcumulado = 0
        for i in range(len(SpanName)-1):
            SpanAcumulado += SpanLength[i]
            SpanStartStations.append(SpanAcumulado)
        SpanStartStations.append(CVão) # Adicionar o fim

        DiaphragmDistanceSpan = []
        for i in range (len(SpanStartStations)-1):
            DiaphragmDistance = []
            for j in range(len(x_Diafragmas)):
                if x_Diafragmas[j] < SpanStartStations[i+1] and x_Diafragmas[j] -
SpanStartStations[i] > 0:
                    DiaphragmDistance.append(x_Diafragmas[j] - SpanStartStations[i])
            DiaphragmDistanceSpan.append(DiaphragmDistance) # A intenção é criar uma lista
de listas, com cada lista referente aos valores de distância para cada vão

        #********************************* FIM VARIÁVEIS DE INPUT
*****************************************#

        # 1 Definição dos objetos primitivos
        SapModel = cSapModel(mySAPObject.SapModel)
        MyBridgeModeler = cBridgeModeler_1(SapModel.BridgeModeler_1) # necessário para
acessar os métodos do CSiBridge

        ret = SapModel.File.OpenFile(os.path.join(main_folder, Tipologia, f'{Tipologia}
(SUPERESTRUTURA).bdb'))
        ret = SapModel.File.Save(os.path.join(main_folder, Tipologia, f'{Tipologia}
(ENCONTROS_VÃO ÚNICO).bdb'))
        kN_cm_C = 13
        kN_m_C = 6
        kN_mm_C = 5
        ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))




        # 2 Objetos de interesse
        NumberNames = int()
        MyName = []
        ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
        myBridgeObjName = list(ret[2])[0]

        Name = myBridgeObjName
        LayoutLine = ''
        ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
        myLayoutLineName = ret[1]
```

```
            ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
            myLaneName = list(ret[2])[0]



            # 3 Atribuição dos Encontros

            # Redefinição dos bents

            Name = 'Apoios extremos'
            Section = [f'Pilar (C{f_ck})'] * N_pilaresext #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing the frame section properties assigned to each
column.
            Angle = [0] * N_pilaresext #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the orientation of the column in degrees counterclockwise between
a line parallel to the bent and the column local 2 axis. [deg]
            Foundation = ['Fixed'] * N_pilaresext #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing either Fixed, Pinned, a defined foundation spring,
or Previous. Note that an error will occur if Previous is assigned to the first column.
            ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_ext, Height_ext,
Angle, Foundation)

            Elevation_StartAbutment = -h_laje -h_long - h_ap #The global Z elevation of the
substructure. (start abutment) -> Distância entre o topo da laje e o CG do aparelho de apoio
            ElevationAtLayoutLine_StartAbutment = -h_laje -h_long - h_ap/2 #The bearing
elevation (global Z) at the layout line. [L] -> Distância entre o topo da laje e a base do
aparelho de apoio
            Elevation_EndAbutment = Elevation_StartAbutment #The global Z elevation of the
substructure.
            ElevationAtLayoutLine_EndAbutement = ElevationAtLayoutLine_StartAbutment #The
bearing elevation (global Z) at the layout line. [L]

            # 3.1 Start Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 1
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'Start Abutment' #The support name. The support name cannot be blank
and it can not be the same as another abutment or bent support name
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
            ret                                                                          =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation, SupportName, AbutmentDirection, DiaphragmProperty, InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            SubstructureType = 2
            AbutmentOrBentProperty = 'Apoios extremos'
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation, SubstructureType, AbutmentOrBentProperty, Elevation_StartAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
```

```
            # 2 = General
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
            NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
            ret    =    MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_StartAbutment,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

            # 3.2 End Abutment
            Name = myBridgeObjName #The name of a bridge object.
            AbutmentLocation = 2
            # This is one of the following indicating the abutment location:
            # 1 = Start abutment.
            # 2 = End abutment.
            SupportName = 'End Abutment'
            AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It
is either Default or a valid bearing string.
            DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the
abutment) or the name of a defined bridge diaphragm property.
            InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-
T girder deck sections.
            #It does not apply to other deck sections. It is either None (meaning no diaphragm
interior diaphragm exists) or the name of a defined bridge diaphragm property.
            ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge
diaphragm property of type Closure Pour.
            #This only applies if the substructure type is Bent Property and the specified bent
property is of type Double Bearing Line with Closure Pour
            DiaphragmOffsetLocation = 1
            #This is one of the following indicating the diaphragm offset location. This item
only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
            # 0 = Flush
            # 1 = Centerline
            ret                                                                              =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

            SubstructureType = 2
            AbutmentOrBentProperty = 'Apoios extremos' #The name of the abutment or bent
property used for the substructure. [L]
            HorizontalOffset = 0 #The horizontal offset from the layout line to the midlength
of the abutment. [L]
            ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,
AbutmentLocation,     SubstructureType,     AbutmentOrBentProperty,     Elevation_EndAbutment,
HorizontalOffset)

            BearingAssignmentType = 1
            # This is one of the following indicating the bearing assignment type:
            # 1 = Girder-by-girder
            # 2 = General
            # The General assignment type only applies to abutments at concrete box girders
deck sections or concrete flat slab deck sections. General assignments to abutments at other
deck section types will return an error.
            BearingProperty = 'Ap. de apoio' #The name of a defined bearing property assigned
to the substructure.
            RestrainerProperty = 'None' #This is either None or the name of a defined restrainer
property assigned to the substructure.
            RotationAngle = 0 #The bearing rotation angle from the bridge default.
            NumberOfBearings = int() #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
```

```
            #For bearings at steel U girder sections this item must be 1 or 2. This item only
applies when the bearing assignment type is General or when the abutment is at a Steel U Girder
deck section.
            UniformSpacing = float() #The uniform spacing of bearings. This item only applies
when the bearing assignment type is General. [L]
            BearingOffset = float() #The offset from the section reference point to the bearing
center. This item only applies when the bearing assignment type is General. [L]
            ret    =    MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,
AbutmentLocation,       BearingAssignmentType,       BearingProperty,       RestrainerProperty,
ElevationAtLayoutLine_EndAbutement,     RotationAngle,     NumberOfBearings,     UniformSpacing,
BearingOffset)

            ret = MyBridgeModeler.Abutment.Delete('BABT1')

            # Update no modelo
            Name = myBridgeObjName
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)
            Name = myBridgeObjName #The name of a bridge object.




            # 4 Modelagem do encontro

            if Tipo_analise == 'ENCONTROS':

                # 4.1 Alas
                OffsetEncontro = 75

                # Ala da direita
                NumberPoints = 4 # The number of points in the area abject.
                x   =   [-OffsetEncontro,   -OffsetEncontro-CompA,   -OffsetEncontro-CompA,   -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                # Escolheu-se o sentido anti-horário, sendo o primeiro ponto o mais próximo da
laje (em x e z)
                # The coordinates should be ordered to run clockwise or counter clockwise around
the area object.
                Name = ''
                # This is the name that the program ultimately assigns to the area object. If
no UserName is specified, the program assigns a default name to the area object.
                # If a UserName is specified and that name is not used for another area object,
the UserName is assigned to the area object; otherwise a default name is assigned to the area
object.
                PropName = 'Encontro'
                # This is Default, None or the name of a defined area property.
                # If it is Default, the program assigns a default area property to the area
object.
                # If it is None, no area property is assigned to the area object. If it is the
name of a defined area property, that property is assigned to the area object.
                UserName = 'Ala direita 1/2' # This is an optional user specified name for the
area object. If a UserName is specified and that name is already used for another area object,
the program ignores the UserName.
                CSys = 'GLOBAL' # The name of the coordinate system in which the area object
point coordinates are defined.
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                NumberPoints = 4
                x   =   [-OffsetEncontro,   -OffsetEncontro-CompA,   -OffsetEncontro-ChanfroH,   -
OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
```

```
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala direita 2/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala direita 1/2' # The name of an existing area object.
                MeshType = 1
                # This item is 1, 2, 3, 4, 5 or 6, indicating the mesh type for the area object.
                # 1 = Mesh area into a specified number of objects
                # 2 = Mesh area into objects of a specified maximum size
                # 3 = Mesh area based on points on area edges
                # 4 = Cookie cut mesh area based on lines intersecting edges
                # 5 = Cookie cut mesh area based on points
                # 6 = Mesh area using General Divide Tool
                AreaName = [] # This is an array of the name of each area object created when
the specified area object is divided.
                n1_aladireita1 = 10 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 2.
                n2_aladireita1 = 5 # This item applies when MeshType = 1. It is the number of
objects created along the edge of the meshed area object that runs from point 1 to point 3.
                NumberAreas = n1_aladireita1*n2_aladireita1 #The number of area objects created
when the specified area object is divided.
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_aladireita1, n2_aladireita1)
                Ala_direita = list(ret[2])

                Name = 'Ala direita 2/2'
                MeshType = 1
                AreaName = []
                n1_aladireita2 = 10
                n2_aladireita2 = 10
                NumberAreas = n1_aladireita2*n2_aladireita2
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_aladireita2, n2_aladireita2)
                Ala_direita.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Ala direita') # This is the name of a group.
If this is the name of an existing group, that group is modified, otherwise a new group is
added.
                for Area in Ala_direita:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala direita' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # Ala da esquerda

                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro-CompA, -OffsetEncontro-CompA, -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, 0, -ChanfroV, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 1/2'
                CSys = 'GLOBAL'
```

```
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro-CompA, -OffsetEncontro-ChanfroH, -
OffsetEncontro]
                y = [TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -ChanfroV, -AlturaM, -AlturaM]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Ala esquerda 2/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Discretização
                Name = 'Ala esquerda 1/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda1 = 10
                n2_alaesquerda1 = 5
                NumberAreas = n1_alaesquerda1*n2_alaesquerda1
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_alaesquerda1, n2_alaesquerda1)
                Ala_esquerda = list(ret[2])

                Name = 'Ala esquerda 2/2'
                MeshType = 1
                AreaName = []
                n1_alaesquerda2 = 10
                n2_alaesquerda2 = 10
                NumberAreas = n1_alaesquerda2*n2_alaesquerda2
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_alaesquerda2, n2_alaesquerda2)
                Ala_esquerda.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Ala esquerda') # This is the name of a group.
If this is the name of an existing group, that group is modified, otherwise a new group is
added.
                for Area in Ala_esquerda:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Ala esquerda' # The name of an existing group to which the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 4.2 Parede cortina
                NumberPoints = 4
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [0, -ChanfroV, -ChanfroV, 0]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 1/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                NumberPoints = 4
```

```
                x = [-OffsetEncontro, -OffsetEncontro, -OffsetEncontro, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla/2, -TotalWidthOfDeckSection/2 + hAla/2,
TotalWidthOfDeckSection/2 - hAla/2, TotalWidthOfDeckSection/2 - hAla/2]
                z = [-ChanfroV, -AlturaM, -AlturaM, -ChanfroV]
                Name = ''
                PropName = 'Encontro'
                UserName = 'Cortina 2/2'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Discretização
                Name = 'Cortina 1/2'
                MeshType = 1
                AreaName = []
                n1_cortina1 = 5
                n2_cortina1 = 20
                NumberAreas = n1_cortina1*n2_cortina1
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_cortina1, n2_cortina1)
                Cortina = list(ret[2])

                Name = 'Cortina 2/2'
                MeshType = 1
                AreaName = []
                n1_cortina2 = 10
                n2_cortina2 = 20
                NumberAreas = n1_cortina2*n2_cortina2
                ret = SapModel.EditArea.Divide(Name, MeshType, NumberAreas, AreaName,
n1_cortina2, n2_cortina2)
                Cortina.extend(list(ret[2]))

                ret = SapModel.GroupDef.SetGroup('Cortina')
                for Area in Cortina:
                    GroupName = 'Cortina'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 4.3 Console
                NumberPoints = 4
                Largura_console = 30
                x = [-OffsetEncontro, -OffsetEncontro-Largura_console, -OffsetEncontro-
Largura_console, -OffsetEncontro]
                y = [-TotalWidthOfDeckSection/2 + hAla, -TotalWidthOfDeckSection/2 + hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1), -
2*(ChanfroV/n2_aladireita1), -2*(ChanfroV/n2_aladireita1)] # Convencionado dessa forma para que
a discretização fique robusta
                Name = ''
                PropName = 'Encontro'
                UserName = 'Console'
                CSys = 'GLOBAL'
                ret = SapModel.AreaObj.AddByCoord(NumberPoints, x, y, z, Name, PropName,
UserName, CSys)

                # Determinação dos pontos para realizar a discretização
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Cortina', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_cortina = list(ret[3])
                Points_cortina = []
                for Area in Areas_cortina:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_cortina.extend(list(ret[2]))

                Pointscortina_coord = []
                for point in Points_cortina:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointscortina_coord.append(ret)

                for i in range(len(Pointscortina_coord)):
```

```
                    Pointscortina_coord[i].insert(0,Points_cortina[i])

                df_pontos_cortina = pd.DataFrame(Pointscortina_coord, columns = ['Joint', 'X',
'Y', 'Z'])
                df_pontos_cortina = df_pontos_cortina.round(4)

                Pointscortina_mesh    =    df_pontos_cortina[(df_pontos_cortina["Z"]    ==    -
2*(ChanfroV/n2_aladireita1))]['Joint'].unique().tolist()

                ret = SapModel.SelectObj.ClearSelection()
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização
                Name = 'Console'
                MeshType = 3
                AreaName = []
                n1_console = 2
                n2_console = 20
                NumberAreas = n1_console*n2_console
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)

                Console = list(ret[2])
                ret = SapModel.GroupDef.SetGroup('Console') # This is the name of a group.  If
this is the name of an existing group,  that group is modified, otherwise a new group is added.
                for Area in Console:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName  =  'Console'  #  The  name  of  an  existing  group  to  which  the
assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 4.4 Laje de transição
                OffsetLT = OffsetEncontro + Largura_console
                NumberPoints = 4
                x = [-OffsetLT, -OffsetLT - LarguraLT, -OffsetLT - LarguraLT, -OffsetLT]
                y  =  [-TotalWidthOfDeckSection/2  +  hAla,  -TotalWidthOfDeckSection/2  +  hAla,
TotalWidthOfDeckSection/2 - hAla, TotalWidthOfDeckSection/2 - hAla]
                z = [-h_laje/2, -h_laje/2, -h_laje/2, -h_laje/2]
                Name = ''
                PropName = 'Laje de transição'
                UserName = 'LT'
                CSys = 'GLOBAL'
                ret  =  SapModel.AreaObj.AddByCoord(NumberPoints,  x,  y,  z,  Name,  PropName,
UserName, CSys)

                # Obtenção dos pontos de discretização
                ret = SapModel.SelectObj.ClearSelection()
                Pointscortina_mesh = Pointscortina_mesh[1:-1]
                for point in Pointscortina_mesh:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                dx = - float(Largura_console)
                dz = 2*(ChanfroV/n2_aladireita1) - h_laje/2
                ret = SapModel.EditGeneral.ReplicateLinear(dx, float(0), dz, int(1), int(), [],
[], False)

                PointsLT_mesh = list(ret[2])
                ret = SapModel.SelectObj.ClearSelection()
                for point in PointsLT_mesh:
```

```
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))

                # Discretização nos pontos do console
                Name = 'LT'
                MeshType = 3
                AreaName = []
                n1_LT = 10
                n2_LT = 20
                NumberAreas = n1_LT*n2_LT
                ret = SapModel.EditArea.Divide(Name, MeshType, int(), AreaName, int(), int(),
float(), float(), False, False, True)
                LT = list(ret[2])

                # Discretização LT
                LT_adic = []
                for Area in LT:
                    ret = SapModel.EditArea.Divide(Area, 1, int(), AreaName, 10, 1)
                    LT_adic.extend(list(ret[2]))

                LT.extend(LT_adic)
                ret = SapModel.GroupDef.SetGroup('Lajes de transição') # This is the name of a
group.  If this is the name of an existing group,  that group is modified, otherwise a new group
is added.

                for Area in LT:
                    # Name = The name of an existing area object or group, depending on the
value of the ItemType item.
                    GroupName = 'Lajes de transição' # The name of an existing group to which
the assignment is made.
                    Remove = False # If this item is False, the specified area objects are
added to the group specified by the GroupName item. If it is True, the area objects are removed
from the group.
                    ItemType = eItemType(0)
                    # This is one of the following items in the eItemType enumeration:
                    # Object = 0
                    # Group = 1
                    # SelectedObjects = 2
                    # If this item is Object, the area object specified by the Name item is
added or removed from the group specified by the GroupName item.
                    # If this item is Group, all area objects in the group specified by the
Name item are added or removed from the group specified by the GroupName item.
                    # If this item is SelectedObjects, all selected area objects are added or
removed from the group specified by the GroupName item, and the Name item is ignored.
                    ret = SapModel.AreaObj.SetGroupAssign(Area, GroupName, Remove, ItemType)

                # 4.5 Links laje de transição
                XMin = -OffsetLT - LarguraLT
                XMax = -OffsetLT - LarguraLT/2
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = False
                Line = False
                Area = True
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                Name = '' # The name of an existing area object or group, depending on the
value of the ItemType item.
                MyType = 1
                # This is either 1 or 2, indicating the spring property type.
                # 1 = Simple spring
                # 2 = Link property
                s = 0.05 # The simple spring stiffness per unit area of the specified area
object face. This item applies only when MyType = 1. [F/L3]
                SimpleSpringType = 2
                # This is 1, 2 or 3, indicating the simple spring type. This item applies only
when MyType = 1.
                # 1 = Spring resists tension and compression
```

```
                # 2 = Spring resists compression only
                # 3 = Spring resists tension only
                LinkProp = '' # The name of the link property assigned to the spring. This item
applies only when MyType = 2.
                Face = -2
                # This is -1, -2 or a nonzero, positive integer indicating the area object face
to which the specified spring assignment applies.
                # -1 = Bottom face
                # -2 = Top face
                # >0 = Edge face
                SpringLocalOneType = 2
                # This is 1, 2 or 3, indicating the method used to specify the spring positive
local 1-axis orientation.
                # 1 = Parallel to area object local axis
                # 2 = Normal to specified area object face
                # 3 = User specified direction vector
                Dir = int() # This is 1, 2, 3, -1, -2 or -3, indicating the area object local
axis that corresponds to the positive local 1-axis of the spring. This item applies only when
SpringLocalOneType = 1.
                Outward = False # This item is True if the spring positive local 1 axis is
outward from the specified area object face. This item applies only when SpringLocalOneType =
2.
                Vec = [] # This is an array of three values that define the direction vector
of the spring positive local 1-axis.
                # The direction vector is in the coordinate system specified by the CSys item.
This item applies only when SpringLocalOneType = 3.
                Ang = float() # This is the angle that the link local 2-axis is rotated from
its default orientation. This item applies only when MyType = 2. [deg]
                Replace = True #If this item is True, all existing spring assignments to the
area object are removed before assigning the specified spring.
                # If it is False, the specified spring is added to any existing springs already
assigned to the area object.
                CSys = 'Local' #This is Local (meaning the area object local coordinate system)
or the name of a defined coordinate system.
                #This item is the coordinate system in which the user specified direction
vector, Vec, is specified. This item applies only when SpringLocalOneType = 3.
                ItemType = eItemType(2)
                ret = SapModel.AreaObj.SetSpring(Name, MyType, s, SimpleSpringType, LinkProp,
Face, SpringLocalOneType, Dir, Outward, Vec, Ang, Replace, CSys, ItemType)

                ret = SapModel.SelectObj.ClearSelection()

                # 4.6 Links console
                XMin = -OffsetLT
                XMax = -OffsetLT
                YMin = -TotalWidthOfDeckSection/2 + hAla
                YMax = TotalWidthOfDeckSection/2 - hAla
                ZMin = -h_laje/2
                ZMax = -h_laje/2
                DeSelect = False
                CSys = 'GLOBAL'
                IncludeIntersections = False
                Point = True
                Line = False
                Area = False
                Solid = False
                Link = False
                ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax, ZMin, ZMax,
DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])

                Points_linkconsole = list(ret[3])
                x = []
                y = []
                z = []
                for point in Points_linkconsole:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(),
float(), 'GLOBAL')
                    x.append(ret[1])
                    y.append(ret[2])
                    z.append(ret[3])

                ret = SapModel.SelectObj.ClearSelection()
                Links_console = []
                for i in range(len(Points_linkconsole)):
```

```
                    xi = x[i] # The coordinates of the I-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item.
                    yi = y[i]
                    zi = z[i]
                    xj = x[i] # The coordinates of the J-End of the added link object. The
coordinates are in the coordinate system defined by the CSys item. These coordinates are ignored
if the IsSingleJoint item is True.
                    yj = y[i]
                    zj = -2*(ChanfroV/n2_aladireita1)
                    Name = ''
                    # This is the name that the program ultimately assigns for the link object.
                    # If no UserName is specified, the program assigns a default name to the
link object.
                    # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                    IsSingleJoint = False # This item is True if a one-joint link is added and
False if a two-joint link is added.
                    PropName = 'Link rigido (esforços)'
                    UserName = ''
                    CSys = 'GLOBAL'
                    ret = SapModel.LinkObj.AddByCoord(xi, yi, zi, xj, yj, zj, Name,
IsSingleJoint, PropName, UserName, CSys)
                    Links_console.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links console')
                for link in Links_console:
                    GroupName = 'Links console'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 4.7 Links travessa # Nao gostei muito dessa solução, investigar outra forma
de fazer
                Name = f'Travessa encontro (C{f_ck})'
                DeSelect = False
                ret = SapModel.SelectObj.PropertyFrame(Name, DeSelect)

                NumberItems = int() # The number of selected objects.
                ObjectType = []
                # This is an array that includes the object type of each selected object.
                # 1 = Point object
                # 2 = Frame object
                # 3 = Cable object
                # 4 = Tendon object
                # 5 = Area object
                # 6 = Solid object
                # 7 = Link object
                ObjectName = [] # This is an array that includes the name of each selected
object.
                ret = SapModel.SelectObj.GetSelected(NumberItems, ObjectType, ObjectName)

                Name = f'{list(ret[3])[0]}'
                Point1 = ''
                Point2 = ''
                ret = SapModel.FrameObj.GetPoints(Name, Point1, Point2)

                Name = ret[1]
                x = float()
                y = float()
                z = float()
                CSys = 'GLOBAL'
                ret = SapModel.PointObj.GetCoordCartesian(Name, x, y, z, CSys)
                xiTravessa = ret[1]
                yiTravessa = ret[2]
                ziTravessa = ret[3]

                zi = [-AlturaM, -AlturaM + ((AlturaM - ChanfroV)/(n1_cortina2))* 6]
                Links_travessa = []
                for j in range (2):
                    for i in range (n2_cortina2+1):
                        xi = -OffsetEncontro # The coordinates of the I-End of the added link
object. The coordinates are in the coordinate system defined by the CSys item.
                        yi = -TotalWidthOfDeckSection/2 + hAla/2 - i*(-
TotalWidthOfDeckSection/2 + hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
```

```
                        xj = xiTravessa # The coordinates of the J-End of the added link object.
The coordinates are in the coordinate system defined by the CSys item. These coordinates are
ignored if the IsSingleJoint item is True.
                        yj     =     -TotalWidthOfDeckSection/2     +     hAla/2     -     i*(-
TotalWidthOfDeckSection/2 + hAla/2+(-TotalWidthOfDeckSection/2 + hAla/2))/n2_cortina2
                        zj = ziTravessa
                        Name = ''
                        # This is the name that the program ultimately assigns for the link
object.
                        # If no UserName is specified, the program assigns a default name to
the link object.
                        # If a UserName is specified and that name is not used for another link
object, the UserName is assigned to the link object; otherwise a default name is assigned to
the link object.
                        IsSingleJoint = False # This item is True if a one-joint link is added
and False if a two-joint link is added.
                        PropName = 'Link rigido'
                        UserName = ''
                        CSys = 'GLOBAL'
                        ret  =  SapModel.LinkObj.AddByCoord(xi,  yi,  zi[j],  xj,  yj,  zj,  Name,
IsSingleJoint, PropName, UserName, CSys)
                        Links_travessa.append(ret[1])

                ret = SapModel.GroupDef.SetGroup('Links travessa')
                for link in Links_travessa:
                    GroupName = 'Links travessa'
                    Remove = False
                    ItemType = eItemType(0)
                    ret = SapModel.LinkObj.SetGroupAssign(link, GroupName, Remove, ItemType)

                # 4.8 Definição dos grupos

                ret = SapModel.GroupDef.SetGroup('Encontro')

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Links console')
                ret = SapModel.SelectObj.Group('Links travessa')
                ret = SapModel.LinkObj.SetGroupAssign('', 'Encontro', False, eItemType(2))


                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.Group('Ala direita')
                ret = SapModel.SelectObj.Group('Ala esquerda')
                ret = SapModel.SelectObj.Group('Console')
                ret = SapModel.SelectObj.Group('Cortina')
                ret = SapModel.SelectObj.Group('Lajes de transição')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Encontro', False, eItemType(2))
                ret = SapModel.SelectObj.ClearSelection()

                ret = SapModel.SelectObj.Group('Encontro')

                Plane = 1
                # This is 1, 2, 3 or 4, indicating the mirror plane type.
                # 1 = Parallel to Z
                # 2 = Parallel to X
                # 3 = Parallel to Y
                # 4 = 3D plane
                x1 = CTotal/2
                y1 = TotalWidthOfDeckSection/2
                z1 = float()
                x2 = CTotal/2
                y2 = -TotalWidthOfDeckSection/2
                z2 = float()
                x3 = float()
                y3 = float()
                z3 = float()
                # These are the coordinates of three points used to define the mirror plane.
[L]
                # When Plane = 1, x1, y1, x2 and y2 define the intersection of the mirror plane
with the XY plane.
                # When Plane = 2, y1, z1, y2 and z2 define the intersection of the mirror plane
with the YZ plane.
                # When Plane = 3, x1, z1, x2 and z2 define the intersection of the mirror plane
with the XZ plane.
```

```
                # When Plane = 4, x1, y1, z1, x2, y2, z2, x3, y3 and z3 define three points
that define the mirror plane.
                NumberObjects = int() # The number of new objects created by the replication
process.
                ObjectName = [] # This is an array of the name of each object created by the
replication process.
                ObjectType = []
                # This is an array of the type of each object created by the replication
process.
                # 1 = Point object
                # 2 = Frame object
                # 3 = Cable object
                # 4 = Tendon object
                # 5 = Area object
                # 6 = Solid object
                # 7 = Link object
                Remove = False # If this item is True, the originally selected objects are
deleted after the replication is complete.
                ret = SapModel.EditGeneral.ReplicateMirror(Plane, x1, y1, z1, x2, y2, z2, x3,
y3, z3, NumberObjects, ObjectName, ObjectType)

                ret = SapModel.GroupDef.SetGroup('Todas lajes')
                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Laje de transição')
                ret = SapModel.SelectObj.Group(f'<{myBridgeObjName}>SLAB')
                ret = SapModel.AreaObj.SetGroupAssign('', 'Todas lajes', False, eItemType(2))
                ret = SapModel.SelectObj.ClearSelection()

                # 4.9 Atribuição dos carregamentos nos encontros
                modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
                wb_pil_ext = xw.Book(modelo_pil_ext)
                ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]
                ret = SapModel.SelectObj.ClearSelection()
                EmpuxoTerra = ws_pil_ext['D82'].value/(100)**2
                EmpuxoTerraCargaMóvel = ws_pil_ext['D93'].value/(100)**2
                Cases = [['Ala direita', 'Empuxo terra', -EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
                ['Ala direita', 'Empuxo terra carga móvel', -EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Ala esquerda', 'Empuxo terra', EmpuxoTerra, 5, False, 'GLOBAL',
eItemType(1)],
                ['Ala esquerda', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 5, False,
'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra', EmpuxoTerra, 4, False, 'GLOBAL', eItemType(1)],
                ['Cortina', 'Empuxo terra carga móvel', EmpuxoTerraCargaMóvel, 4, False,
'GLOBAL', eItemType(1)]]

                for case in Cases:
                    Name, LoadPat, Value, Dir, Replace, CSys, ItemType = case
                    ret = SapModel.AreaObj.SetLoadUniform(Name, LoadPat, Value, Dir, Replace,
CSys, ItemType)

                # 4.10 Impacto na defensa

                # 4.10.1 Obtenção dos pontos do encontro para a atribuição do carregamento

                ret = SapModel.SelectObj.ClearSelection()
                ret = SapModel.SelectObj.PropertyArea('Encontro', False)
                ret = SapModel.SelectObj.GetSelected(int(), [], [])
                Areas_encontro = list(ret[3])
                Points_encontro = [] # Nomes dos pontos que definem as áreas
                for Area in Areas_encontro:
                    ret = SapModel.AreaObj.GetPoints(Area, int(), [])
                    Points_encontro.extend(list(ret[2]))

                Pointsencontro_coord = []
                for point in Points_encontro:
                    ret = SapModel.PointObj.GetCoordCartesian(point, float(), float(),
float(), 'GLOBAL')
                    ret = list(ret)
                    ret.pop(0)
                    Pointsencontro_coord.append(ret)

                for i in range(len(Pointsencontro_coord)):
```

```
                    Pointsencontro_coord[i].insert(0,Points_encontro[i]) # Inserção dos nomes dos pontos em cada lista

                # 4.10.2 Criação do dataframe com todos os elementos de área e os quatro pontos que os definem (start e end station será feito depois de discretizar a laje)
                df_pontos_encontro = pd.DataFrame(Pointsencontro_coord, columns = ['Joint', 'X', 'Y', 'Z']) # DataFrame com todas as áreas, sendo cada linha um ponto que a define
                df_pontos_encontro = df_pontos_encontro.round(4) # Arredondar todas as colunas do DataFrame para 4 casas decimais

                # 4.10.3 Obtenção dos pontos

                Points_impacto_esq = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) & (df_pontos_encontro["Y"] == -TotalWidthOfDeckSection/2 + hAla/2) & (df_pontos_encontro["X"] <= -OffsetEncontro)]['Joint'].unique().tolist()
                Points_impacto_dir = df_pontos_encontro[(df_pontos_encontro["Z"] == 0) & (df_pontos_encontro["Y"] == TotalWidthOfDeckSection/2 - hAla/2) & (df_pontos_encontro["X"] <= -OffsetEncontro)]['Joint'].unique().tolist()

                # 4.10.4 Atribuição do carregamento
                fdist_impacto = 100/(50 + 2*87) # em kN/cm
                fno_impacto = (fdist_impacto * CompA)/len(Points_impacto_esq)

                mdist_impacto = 100*87/(50 + 2*87) # em kN.cm/cm
                mno_impacto = mdist_impacto*CompA/len(Points_impacto_esq)

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_esq:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , fno_impacto, 0, -mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))

                ret = SapModel.SelectObj.ClearSelection()
                for point in Points_impacto_dir:
                    ret = SapModel.PointObj.SetSelected(point, True, eItemType(0))
                    ret = SapModel.PointObj.SetLoadForce('', 'Impacto', [0 , -fno_impacto, 0, mno_impacto, 0, 0], True, 'GLOBAL', eItemType(2))


            # 5 Definição das Lanes - (cBMLane.SetLaneFromLayoutLine) [myLaneName sempre será a faixa de tráfego]

            Name = myLaneName #The name for the existing lane defined from a layout line.
            NumberLaneStations = 2 #The number of lane points, that is, the number of items included in the that lane will be overwritten. LayoutLine, Station, Offset, Width and Radius arrays.
            LayoutLine = [myLayoutLineName, myLayoutLineName]#A zero-based array containing the name of the bridge layout line associated with each lane station.
            Station = [0, CTotal]#A zero-based array containing the station at each lane station. [L]
            Offset = [0, 0]#A zero-based array containing the offset from the layout line to the center of the lane at each lane station. [L]
            Width = [LarguraRolamento, LarguraRolamento]#A zero-based array containing the the width of the lane at each lane station. [L]
            Radius = [0,0]#A zero-based array containing the the radius of the lane at each lane station. A positive radius curves to the right and negative to the left. [L]
            LaneType = 0
            #The lane type (fixed or floating).
            # 0 = Fixed lane
            # 1 = Floating lane set
            if Tipo_analise == 'ENCONTROS':
                LoadGroup = 'Todas lajes'
            else:
                LoadGroup = f'<{myBridgeObjName}>SLAB'
            LeftEdgeIsExterior = False #Indicates whether the left edge of the lane is exterior or interior.
            RightEdgeIsExterior = False #Indicates whether the right edge of the lane is exterior or interior.
            MaxDiscAlongLane = 300 #The maximum lane load discretization length along the lane. [L]
            MaxDiscAcrossLane = 300 #The maximum lane load discretization length across the lane. [L]
            UseDiscFactorSpan = True #Indicates whether the the DiscFactorSpan item is to apply to the lane.
```

```
            UseDiscFactorLane = True #Indicates whether the the DiscFactorLane item is to apply
to the lane.
            DiscFactorSpan = 4 #The lane load discretization length along the lane will not be
more than 1/x of the span length where x is the DiscFactorSpan value.
            #This item is only applicable when the UseDiscFactorSpan item is True.
            DiscFactorLane = 10 #The lane load discretization length along the lane will not
be more than 1/x of the lane length where x is the DiscFactorLane value.
            #This item is only applicable when the UseDiscFactorLane item is True.
            Color = int()#The color assigned to the lane. (Optional)
            Notes = ''#The notes, if any, assigned to the lane. (Optional)
            ret    =    MyBridgeModeler.Lane.SetLaneFromLayoutLine(Name,    NumberLaneStations,
LayoutLine,   Station,   Offset,   Width,   Radius,   LaneType,   LoadGroup,   LeftEdgeIsExterior,
RightEdgeIsExterior,
            MaxDiscAlongLane,    MaxDiscAcrossLane,    UseDiscFactorSpan,    UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

            # Laje de transição inicial

            ret = MyBridgeModeler.LayoutLine.AddNew('BLL0')

            Name = 'BLL0'
            CoordinateSystem = 'GLOBAL'
            StartX = -1000
            StartY = 0
            StartZ = 0
            ret  =  MyBridgeModeler.LayoutLine.SetGeneralData(Name,  CoordinateSystem,  StartX,
StartY, StartZ)

            Name = 'BLL0'
            NumberSegments = 2
            SegmentType = [1, 2]
            Station = [0, 1000]
            Radius = [float(), float()]
            Bearing = ['N900000E', 'N900000E']
            ret   =  MyBridgeModeler.LayoutLine.SetHorizontalLayoutData(Name,  NumberSegments,
SegmentType, Station, Radius, Bearing)

            ret = MyBridgeModeler.Lane.AddNewFromLayoutLine('LANE0', 'BLL0')
            ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine('LANE0', 2, ['BLL0', 'BLL0'], [0,
1000],   [0]*2,   [LarguraRolamento]*2,   [0]*2,   LaneType,   LoadGroup,   LeftEdgeIsExterior,
RightEdgeIsExterior,
            MaxDiscAlongLane,    MaxDiscAcrossLane,    UseDiscFactorSpan,    UseDiscFactorLane,
DiscFactorSpan, DiscFactorLane, Color, Notes)

            LoadNumber = 1 #The load assignment number.
            NumberItems = 2 #The number of lanes loaded for the specified load assignment
number.
            MyName = [f'{myLaneName}', 'LANE0'] #This is an array that includes the name of
each lane loaded for the specified load assignment number.
            ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber, NumberItems,
MyName)
            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Multidao',     LoadNumber,
NumberItems, MyName)




            # 6 Update no modelo e rodar análise
            Name = myBridgeObjName
            Action = 1 #Atualização do modelo para ser possível de selecionar os elementos do
objeto ponte
            # An integer value between 1 and 4, inclusive, indicating the update action to be
taken.
            # 1 = Update Linked Model
            # 2 = Clear All From Linked Model
            # 3 = Convert To Unlinked Model
            # 4 = Clear and Create Linked Model
            ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name, Action)

            Name = ''
            Run = True # If this item is True, the specified load case is to be run.
            All = True # If this item is True, the run flag is set as specified by the Run item
for all load cases, and the Name item is ignored.
            ret = SapModel.Analyze.SetRunCaseFlag(Name, Run, All)
```

```
ret = SapModel.Analyze.RunAnalysis()



# 7 Salvar modelo e planilhas
ret = SapModel.File.Save()
#ret = mySAPObject.ApplicationExit(False)
SapModel = None
mySAPObject = None
MyBridgeModeler = None
```

```
        # ESSE PROGRAMA APENAS SERÁ USADO PARA A CRIAÇÃO DOS MODELOS DE ANÁLISE DE MESOESTRUTURA
COM VÃOS DIFERENTES, NÃO SENDO UTILIZADO PARA NENHUM OUTRO FIM. ELE APENAS CRIARÁ O MODELO E O
DEIXARÁ RODADO.


        import os
        import sys
        import clr
        import pandas as pd
        from enum import Enum
        import numpy as np
        import xlwings as xw
        import math

        clr.AddReference("System.Runtime.InteropServices")
        clr.AddReference(r'C:\Program      Files\Computers      and      Structures\CSiBridge
26\CSiBridge1.dll')
        from System.Runtime.InteropServices import Marshal
        from CSiBridge1 import *

        # Funções definidas

        from utils import *

        #set the following flag to True to execute on a remote computer

        Remote = False


        #if the above flag is True, set the following variable to the hostname of the remote
computer

        #remember that the remote computer must have CSiBridge installed and be running the
CSiAPIService.exe

        RemoteComputer = "SpareComputer-DT"



        #set the following flag to True to attach to an existing instance of the program

        #otherwise a new instance of the program will be started

        AttachToInstance = True



        #set the following flag to True to manually specify the path to CSiBridge.exe

        #this allows for a connection to a version of CSiBridge other than the latest
installation

        #otherwise the latest installed version of CSiBridge will be launched

        SpecifyPath = True

        #if the above flag is set to True, specify the path to CSiBridge below

        ProgramPath = r'C:\Program Files\Computers and Structures\CSiBridge 26\CSiBridge.exe'

        #full path to the model

        #set it to the desired path of your model

        #if executing remotely, ensure that this folder already exists on the remote computer

        #the below command will only create the folder locally

        main_folder = os.path.dirname(os.path.abspath(__file__))
        APIPath = main_folder

        # Planilhas

        composite_template = os.path.join(main_folder, 'composite-template.xlsx')
        wb = xw.Book(composite_template)
```

```
        ws_dados = wb.sheets["1- Dados"]
        ws_acoes = wb.sheets["2-Ações"]
        ws_prop = wb.sheets["3-Prop Geo"]
        ws_esforcos = wb.sheets["4-Esforços"]
        ws_perfil = wb.sheets["5- VP-SCF"]
        ws_mista = wb.sheets["6- VSM-C"]
        ws_conectores = wb.sheets["7-Conectores"]
        ws_flecha = wb.sheets["8-ELS-Flecha"]
        ws_enrijecedores = wb.sheets["9-Enrijecedores"]
        ws_soldac = wb.sheets["10-Solda de composição"]
        ws_soldaf = wb.sheets["11-Soldas-fadiga"]
        ws_emendas = wb.sheets["12-Emendas"]
        ws_diafragmasapoio = wb.sheets["13-Diafragma apoio"]
        ws_diafragmasintermediarios = wb.sheets["14-Diafragma int"]
        ws_lajes = wb.sheets["17-Lajes"]

        modelo_pil_ext = os.path.join(main_folder, 'column-template.xlsx')
        wb_pil_ext = xw.Book(modelo_pil_ext)
        ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

        if not os.path.exists(APIPath):

            try:

                os.makedirs(APIPath)

            except OSError:

                pass

        # INÍCIO DO LOOP
        variables_data_file = os.path.join(main_folder, 'composite-variables.xlsx')
        variables    =    pd.read_excel(variables_data_file,    sheet_name="Main",    header=3,
index_col=False, keep_default_na=True)
        variables = variables.astype(float,errors='ignore') # Converter todos os dados para
float
        Tipologias = ['S-MAC-A-20-P', 'S-MAC-A-25-P', 'S-MAC-A-30-P', 'S-MAC-A-35-P', 'S-MAC-
A-20-S', 'S-MAC-A-25-S', 'S-MAC-A-30-S', 'S-MAC-A-35-S', 'S-MAC-B-20-P', 'S-MAC-B-25-P',
        'S-MAC-B-30-P', 'S-MAC-B-35-P', 'S-MAC-B-20-S', 'S-MAC-B-25-S', 'S-MAC-B-30-S', 'S-
MAC-B-35-S']

        # ['S-MAC-A-20-P', 'S-MAC-A-25-P', 'S-MAC-A-30-P', 'S-MAC-A-35-P', 'S-MAC-A-20-S', 'S-
MAC-A-25-S', 'S-MAC-A-30-S', 'S-MAC-A-35-S', 'S-MAC-B-20-P', 'S-MAC-B-25-P',
        # 'S-MAC-B-30-P', 'S-MAC-B-35-P', 'S-MAC-B-20-S', 'S-MAC-B-25-S', 'S-MAC-B-30-S', 'S-
MAC-B-35-S']

        Tipo_analise = 'MESOESTRUTURA' # Tipos contemplados: AQUI SERÁ APENAS 'MESOESTRUTURA'
        # CASO SEJA ANÁLISE DE MESOESTRUTURA, ALIMENTAR O VETOR DE ALTURA DOS PILARES NA SEÇÃO
DE INPUTS

        for Tipologia in Tipologias: # Para cada tipologia, será variada a altura dos pilares
e também o comprimento dos vãos

            #**********************************      INÍCIO      VARIÁVEIS      DE      INPUT
*****************************************#
            f_ck = 35
            NVãos = 1 if Tipo_analise == 'SUPERESTRUTURA' else 3 if Tipo_analise == 'ENCONTROS'
else 4
            df_ponte = variables[(variables['Tipologia'] == Tipologia)]

            # SEÇÃO TRANSVERSAL
            CVão = df_ponte['Vão (cm)'].iloc[0]
            CTotal = NVãos * CVão # Caso de pontes com múltiplos vãos
            TotalWidthOfDeckSection = df_ponte['Largura (cm)'].iloc[0] #The total width of the
deck section. [L]
            NLongarinasPorVao = int(df_ponte['Nlong'].iloc[0])
            NumberInteriorGirders = NLongarinasPorVao - 2 #The number of interior girders
            OverhangLength  =  df_ponte['Balanço  (cm)'].iloc[0]  #balanço  laje/longarina  #The
overhang length (identified as L1 on the Define Bridge Section Data form). [L]
            Espaçamento_Long            =            (TotalWidthOfDeckSection            -
2*OverhangLength)/(NLongarinasPorVao-1)
            NDiafragmas = int(df_ponte['Ndiaf'].iloc[0])
            LarguraRolamento = TotalWidthOfDeckSection - 2*40 # Será desconsiderado, para fins
de dimensionamento, os passeios (prevendo um possível alargamento da ponte)
```

```
            # AÇO ESTRUTURAL A588Gr50
            fy_A588Gr50 = 345
            fu_A588Gr50 = 480
            fye_A588Gr50 = fy_A588Gr50/1.15
            fue_A588Gr50 = fye_A588Gr50/1.15
            df_ponte = variables[(variables['Tipologia'] == Tipologia)] # DataFrame filtrado
para a tipologia

            # DIMENSÕES DOS ELEMENTOS ESTRUTURAIS

                # PERFIL 1
            t3_1 = df_ponte['t3_1'].iloc[0] # The section depth
            t2_1 = df_ponte['t2_1'].iloc[0] # The top flange width
            tf_1 = df_ponte['tf_1'].iloc[0] # The top flange thickness
            tw_1 = df_ponte['tw_1'].iloc[0] # The web thickness
            t2b_1 = df_ponte['t2b_1'].iloc[0] # The bottom flange width
            tfb_1 = df_ponte['tfb_1'].iloc[0] # The bottom flange thickness
            FilletRadius_1 = df_ponte['rad_1'].iloc[0] # The fillet radius

                # PERFIL 2
            t3_2 = df_ponte['t3_2'].iloc[0] # The section depth
            t2_2 = df_ponte['t2_2'].iloc[0] # The top flange width
            tf_2 = df_ponte['tf_2'].iloc[0] # The top flange thickness
            tw_2 = df_ponte['tw_2'].iloc[0] # The web thickness
            t2b_2 = df_ponte['t2b_2'].iloc[0] # The bottom flange width
            tfb_2 = df_ponte['tfb_2'].iloc[0] # The bottom flange thickness
            FilletRadius_2 = df_ponte['rad_2'].iloc[0] # The fillet radius

                # DIAFRAGMAS EXTREMOS
            t3diaf_1 = df_ponte['t3diaf_1'].iloc[0] # The section depth
            t2diaf_1 = df_ponte['t2diaf_1'].iloc[0] # The top flange width
            tfdiaf_1 = df_ponte['tfdiaf_1'].iloc[0] # The top flange thickness
            twdiaf_1 = df_ponte['twdiaf_1'].iloc[0] # The web thickness
            t2bdiaf_1 = df_ponte['t2bdiaf_1'].iloc[0] # The bottom flange width
            tfbdiaf_1 = df_ponte['tfbdiaf_1'].iloc[0] # The bottom flange thickness
            FilletRadiusdiaf_1 = df_ponte['raddiaf_1'].iloc[0] # The fillet radius

                # DIAFRAGMAS INTERMEDIÁRIOS
            t3diaf_2 = df_ponte['t3diaf_2'].iloc[0] # The section depth
            t2diaf_2 = df_ponte['t2diaf_2'].iloc[0] # The top flange width
            tfdiaf_2 = df_ponte['tfdiaf_2'].iloc[0] # The top flange thickness
            twdiaf_2 = df_ponte['twdiaf_2'].iloc[0] # The web thickness
            t2bdiaf_2 = df_ponte['t2bdiaf_2'].iloc[0] # The bottom flange width
            tfbdiaf_2 = df_ponte['tfbdiaf_2'].iloc[0] # The bottom flange thickness
            FilletRadiusdiaf_2 = df_ponte['raddiaf_2'].iloc[0] # The fillet radius

                # PILARES
            t3_pilar = df_ponte['Dp'].iloc[0]

                # TRAVESSAS
            h_travessaext = df_ponte['hT'].iloc[0]
            bw_travessaext = df_ponte['bwT'].iloc[0]
            h_travessaint = df_ponte['hT'].iloc[0]
            bw_travessaint = df_ponte['bwT'].iloc[0]

                # TABULEIRO

            h_laje = df_ponte['h_laje (cm)'].iloc[0]
            h_prelaj = df_ponte['hPL'].iloc[0]
            bw_prelaj = df_ponte['bwPL'].iloc[0]
            l_prelaj = Espaçamento_Long - t2_1

                # APARELHO DE APOIO

            a_ap = df_ponte['aA'].iloc[0]
            b_ap = df_ponte['bA'].iloc[0]
            ShoreA_ap = 60 # Dureza Shore A do aparelho de apoio
            G_ap = 1000 # Módulo de elasticidade transversal do aparelho de apoio em kN/m²
            sigmaadm_ap = 15000 # Tensão admissível no aparelho de apoio
            hs_ap = 4 # Espessura das chapas de fretagem em mm
            hi_ap = 12 # Espessura das camadas intermediárias do Neoprene
            cv_ap = 2.5 # Cobrimento vertical
            ch_ap = 4 # Cobrimento horizontal
            n_ap = df_ponte['nA'].iloc[0] # Número de camadas do aparelho de apoio
```

```
            h_ap = (n_ap * hi_ap + 2* cv_ap + (n_ap+1)*hs_ap)/10 # Pré-dimensionamento da altura
do aparelho de apoio
            K_ap = a_ap/100 * b_ap/100 * G_ap/(cv_ap*2 + hi_ap*n_ap)*10 # Rigidez do aparelho
de apoio na direção longitudinal e transversal à longarina
            d_ap =  df_ponte['Dapoio'].iloc[0] # Distância do centro do aparelho de apoio à
travessa em planta

                # LAJE DE TRANSIÇÃO
            CompLT = df_ponte['CompLT'].iloc[0]
            LarguraLT = df_ponte['LarguraLT'].iloc[0]
            EspessuraLT = df_ponte['LarguraLT'].iloc[0]

                # MURO
            AlturaM = df_ponte['AlturaM'].iloc[0]

                # ALA
            CompA = df_ponte['CompA'].iloc[0]
            hAla = df_ponte['hAla'].iloc[0]
            ChanfroV = df_ponte['ChanfroV'].iloc[0]
            ChanfroH = df_ponte['ChanfroH'].iloc[0]

            # DEFINIÇÃO DE BENTS

            # EXTREMOS
            C_travessaext = TotalWidthOfDeckSection #The length of the cap beam. [L]
            N_pilaresext = int(df_ponte['Nextp'].iloc[0]) #The number of columns in the bent.
            Hextp = 300 if Tipo_analise == 'ENCONTROS' else df_ponte['Hextp'].iloc[0]
            Distextp = df_ponte['Distextp'].iloc[0]
            SupportType_ext = 1
            # 0 = None (No column or Foundation Spring)
            # 1 = Column
            # 2 = Foundation Spring
            if N_pilaresext == 2:
                Distance_ext  =  [Distextp,  C_travessaext  -  Distextp]  #A  zero-based  one-
dimensional  array  dimensioned  to  (NumberColumns  -  1)  containing  the  distance  to  the  column
centerline measured from the left end of the cap beam. [L]
                Height_ext = [Hextp, Hextp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_ext = [Distextp, C_travessaext/2, C_travessaext - Distextp]
                Height_ext = [Hextp, Hextp, Hextp]

            # INTERMEDIÁRIOS
            C_travessaint = df_ponte['Cint'].iloc[0]
            N_pilaresint = int(df_ponte['Nintp'].iloc[0])
            Hintp = df_ponte['Hintp'].iloc[0]
            Distintp = df_ponte['Distintp'].iloc[0]
            SupportType_int = 1
            if N_pilaresint == 2:
                Distance_int  =  [Distintp,  C_travessaint  -  Distintp]  #A  zero-based  one-
dimensional  array  dimensioned  to  (NumberColumns  -  1)  containing  the  distance  to  the  column
centerline measured from the left end of the cap beam. [L]
                Height_int = [Hintp, Hintp] #A zero-based one-dimensional array dimensioned to
(NumberColumns - 1) containing the column height. [L]
            else:
                Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                Height_int = [Hintp, Hintp, Hintp]

            # ASPECTOS DO PROJETO
            e_a = 40 # Distância entre o início do perfil da longarina e o centro do apoio
            CEfetivo = CVão - 2*e_a # Vão efetivo da ponte
            NTrechos = 3 # Número de perfis para vencer um vão

                # PERFIS
            NPerfil1 = NTrechos if NTrechos <= 2 else 1 if NTrechos == 3 else 2
            NPerfil2 = 0 if NTrechos <= 2 else 2
            CPerfil1  =  CVão  if  NTrechos  ==  1  else  CVão/NTrechos  if  NTrechos  ==  2  and
CVão/NTrechos <= 1190 else 1190
            NEmendas = math.floor(CVão/1190)
            ejunta = 5
            CPerfil2 = (CVão - CPerfil1)/NPerfil2
            x_S1 = 0 if NTrechos == 1 else CPerfil1 if NTrechos == 2 else CPerfil2 # Posição
da Emenda 1
            x_S2  =  0  if  NTrechos  <=  2  else  CVão  -  CPerfil2  if  NTrechos  ==  3  else  CVão/2  #
Posição da Emenda 2
```

```
            # DIAFRAGMAS

        eet = 1.25 # Espessura dos enrijecedores transversais
        let = 15 # Largura dos enrijecedores transversais
        eea = 1.6 # Espessura dos enrijecedores de apoio
        lea = 20 # Largura dos enrijecedores de apoio

            # DIAFRAGMAS
        DistDiafragmas = CEfetivo/(NDiafragmas-1)
        x_Diafragmas = [e_a]
        x_Acumulado = e_a
        while x_Acumulado <= CEfetivo:
            x_Acumulado += DistDiafragmas
            x_Diafragmas.append(x_Acumulado)

        SpanLength = [] # A zero-based array containing the length of each span. [L]
        SpanName = []
        for i in range(NVãos):
            if NTrechos <= 2:
                SpanLength.extend([CPerfil1, CPerfil1]) #A zero-based array containing the
length of each span. [L]
                SpanName.extend([f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'])
            elif NTrechos == 3:
                SpanLength.extend([CPerfil2, CPerfil1, CPerfil2]) #A zero-based array
containing the length of each span. [L]
                SpanName.extend([f'Perfil 2.{i+1}.1', f'Perfil 1.{i+1}', f'Perfil
2.{i+1}.2'])
            else:
                SpanLength.extend([CPerfil2, CPerfil1, CPerfil1, CPerfil2]) #A zero-based
array containing the length of each span. [L]
                SpanName.extend([f'Perfil 2.{i+1}.1', f'Perfil 1.{i+1}.1', f'Perfil
1.{i+1}.2', f'Perfil 2.{i+1}.2'])

        # Alimentação das planilhas 2-Ações e 3-Prop Geo
        ws_dados['H108'].value = df_ponte['Classe'].iloc[0]
        ws_dados['J109'].value = CVão/100
        ws_dados['J110'].value = df_ponte['nf'].iloc[0]
        ws_dados['J111'].value = df_ponte['fr'].iloc[0]/100
        ws_dados['J112'].value = df_ponte['ace'].iloc[0]/100
        ws_dados['J113'].value = df_ponte['aci'].iloc[0]/100
        ws_dados['J114'].value = df_ponte['qp'].iloc[0]
        ws_dados['J115'].value = df_ponte['Lp'].iloc[0]/100
        ws_dados['J116'].value = NLongarinasPorVao
        ws_dados['J117'].value = df_ponte['GC'].iloc[0]/100
        ws_dados['J118'].value = Espaçamento_Long/100
        ws_dados['J119'].value = TotalWidthOfDeckSection/100
        ws_dados['J120'].value = OverhangLength/100
        ws_dados['J122'].value = h_laje/100
        ws_dados['J128'].value = h_prelaj/100
        ws_dados['I148'].value = e_a/100
        ws_dados['H163'].value = NTrechos
        ws_dados['H164'].value = NPerfil1
        ws_dados['H165'].value = NPerfil2
        ws_dados['H166'].value = CPerfil1/100
        ws_dados['H167'].value = CPerfil2/100
        ws_dados['I169'].value = NEmendas
        ws_dados['I170'].value = x_S1/100
        ws_dados['I172'].value = x_S2/100
        ws_dados['H186'].value = NDiafragmas
        ws_dados['H187'].value = DistDiafragmas*10

        # Aparelho de apoio
        ws_dados['I347'].value = a_ap*10
        ws_dados['I348'].value = b_ap*10
        ws_dados['I349'].value = h_ap*10
        ws_dados['I350'].value = hi_ap
        ws_dados['I351'].value = n_ap
        ws_dados['I352'].value = hs_ap
        ws_dados['I353'].value = n_ap + 1
        ws_dados['I354'].value = n_ap * hi_ap

        # Propriedades geométricas
        ws_prop['H26'].value = tw_1*10
        ws_prop['H27'].value = t3_1*10
```

```
            ws_prop['H28'].value = tf_1*10
            ws_prop['H29'].value = tfb_1*10
            ws_prop['H30'].value = t2_1*10
            ws_prop['H31'].value = t2b_1*10

            ws_prop['I26'].value = tw_2*10
            ws_prop['I27'].value = t3_2*10
            ws_prop['I28'].value = tf_2*10
            ws_prop['I29'].value = tfb_2*10
            ws_prop['I30'].value = t2_2*10
            ws_prop['I31'].value = t2b_2*10

            #********************************       FIM       VARIÁVEIS       DE       INPUT
*****************************************# E fim das definições dos elementos

            Alturas_pilarescentrais = [400, 600, 800] # Para cada altura de pilar

            for altura in Alturas_pilarescentrais:
                if N_pilaresint == 2:
                    Distance_int = [Distintp, C_travessaint - Distintp] #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the distance to the column
centerline measured from the left end of the cap beam. [L]
                    Height_int = [altura, altura] #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing the column height. [L]
                else:
                    Distance_int = [Distintp, C_travessaint/2, C_travessaint - Distintp]
                    Height_int = [altura, altura, altura]

                Vãos = [2000, 2500, 3000, 3500]
                for Vão in Vãos:
                    if CVão > Vão:

                        # Comunicação com a API
                        xlwingsapp = xw.App(visible=False) # Faz o Excel ficar invisível
                        GeneralPath = os.path.join(main_folder, Tipologia)
                        os.makedirs(GeneralPath, exist_ok=True) # Cria o diretório relativo à
tipologia. O parâmetro exist_ok ignora caso a pasta já esteja criada, não emitindo erros
                        ModelPath       =       os.path.join(GeneralPath,       f'{Tipologia}
({Tipo_analise}).bdb') # Caminho do modelo .bdb

                        #create API helper object

                        helper = cHelper(Helper())

                        if AttachToInstance:

                            #attach to a running instance of CSiBridge

                            try:

                                #get the active CSiBridge object

                                if Remote:

                                    mySAPObject  =  cOAPI(helper.GetObjectHost(RemoteComputer,
"CSI.CSiBridge.API.SAPObject"))

                                else:

                                    mySAPObject                                          =
cOAPI(helper.GetObject("CSI.CSiBridge.API.SAPObject"))

                            except:

                                print("No running instance of the program found or failed to
attach.")

                                sys.exit(-1)


                        else:

                            if SpecifyPath:
```

```
                                try:
                                    #'create an instance of the CSiBridge object from the
specified path
                                    if Remote:
                                        mySAPObject                                         =
cOAPI(helper.CreateObjectHost(RemoteComputer, ProgramPath))
                                    else:
                                        mySAPObject = cOAPI(helper.CreateObject(ProgramPath))
                                except :
                                    print("Cannot start a new instance of the program from " +
ProgramPath)
                                    sys.exit(-1)
                            else:


                                try:
                                    #create an instance of the CSiBridge object from the latest
installed CSiBridge
                                    if Remote:
                                        mySAPObject                                         =
cOAPI(helper.CreateObjectProgIDHost(RemoteComputer, "CSI.CSiBridge.API.SAPObject"))
                                    else:
                                        mySAPObject                                         =
cOAPI(helper.CreateObjectProgID("CSI.CSiBridge.API.SAPObject"))


                                except:
                                    print("Cannot start a new instance of the program.")
                                    sys.exit(-1)


                            #start CSiBridge application
                            mySAPObject.ApplicationStart()


                        # Redefinição do vão para vãos diferentes

                        NPerfil1 = NTrechos if NTrechos <= 2 else 1 if NTrechos == 3 else 2
                        NPerfil2 = 0 if NTrechos <= 2 else 2
                        CPerfil1 = Vão if NTrechos == 1 else Vão/NTrechos if NTrechos == 2 and
Vão/NTrechos <= 1190 else 1190
                        CPerfil2 = (Vão - CPerfil1)/NPerfil2

                        # Alterando comprimentos do vetor SpanLength nos vãos 2 e 4 para o novo
comprimento
                        SpanLength [4] = CPerfil1
                        SpanLength [10] = CPerfil1
                        SpanLength [3] = CPerfil2
                        SpanLength[5] = CPerfil2
                        SpanLength[9] = CPerfil2
                        SpanLength[11] = CPerfil2
                        CTotal = sum(SpanLength)

                        # Novas posições dos diafragmas. A letra B será para indicar as novas
posições
```

```
                        e_a = 40 # Distância entre o início do perfil da longarina e o centro
do apoio
                        CEfetivo = Vão - 2*e_a # Vão efetivo da ponte
                        NDiafragmasB = 4 if Vão == 2000 else 5 if Vão == 2500 else 6 if Vão ==
3000 else 7

                        DistDiafragmasB = CEfetivo/(NDiafragmasB-1)
                        x_DiafragmasB = [e_a]
                        x_AcumuladoB = e_a
                        while x_AcumuladoB <= CEfetivo:
                            x_AcumuladoB += DistDiafragmasB
                            x_DiafragmasB.append(x_AcumuladoB) # Coordenadas dos diafragmas em
relação ao início da longarina

                        SpanStartStations = [0]
                        SpanAcumulado = 0
                        for i in range(len(SpanName)-1):
                            SpanAcumulado += SpanLength[i]
                            SpanStartStations.append(SpanAcumulado)
                        SpanStartStations.append(CTotal) # Adicionar o fim

                        DiaphragmDistanceSpan = [[] for _ in range(NTrechos * NVãos)]
                        # Diafragmas dos vãos 1 e 3
                        SpanStartStationsA = SpanStartStations[0:4]
                        for i in range (NTrechos):
                            DiaphragmDistance = []
                            for j in range(len(x_Diafragmas)):
                                if    x_Diafragmas[j]    <    SpanStartStationsA[i+1]    and
x_Diafragmas[j] - SpanStartStationsA[i] > 0:
                                    DiaphragmDistance.append(x_Diafragmas[j]                -
SpanStartStationsA[i])
                            DiaphragmDistanceSpan[i],       DiaphragmDistanceSpan[i+6]       =
DiaphragmDistance, DiaphragmDistance # A intenção é criar uma lista de listas, com cada lista
referente aos valores de distância para cada vão

                        # Diafragmas dos vãos 2 e 4
                        SpanStartStationsB = [SpanStartStations[i] - CVão for i in range(3,7)]
                        for i in range (NTrechos):
                            DiaphragmDistance = []
                            for j in range(len(x_DiafragmasB)):
                                if    x_DiafragmasB[j]    <    SpanStartStationsB[i+1]    and
x_DiafragmasB[j] - SpanStartStationsB[i] > 0:
                                    DiaphragmDistance.append(x_DiafragmasB[j]                -
SpanStartStationsB[i])
                            DiaphragmDistanceSpan[i+3],      DiaphragmDistanceSpan[i+9]      =
DiaphragmDistance, DiaphragmDistance # A intenção é criar uma lista de listas, com cada lista
referente aos valores de distância para cada vão

                        # 1 Definição dos objetos primitivos
                        SapModel = cSapModel(mySAPObject.SapModel)
                        MyBridgeModeler    =    cBridgeModeler_1(SapModel.BridgeModeler_1)    #
necessário para acessar os métodos do CSiBridge
                        ret   =   SapModel.File.OpenFile(os.path.join(main_folder,   'composite-
template.bdb'))
                        if Tipo_analise != 'MESOESTRUTURA':
                            ret   =   SapModel.File.Save(os.path.join(main_folder,   Tipologia,
f'{Tipologia} ({Tipo_analise}).bdb'))

                        kN_cm_C = 13
                        kN_m_C = 6
                        kN_mm_C = 5
                        ret = SapModel.SetPresentUnits(eUnits(kN_cm_C))

                        # 2 Propriedades dos materiais

                        # 2.1 Definições iniciais

                        PropMaterial = cPropMaterial(SapModel.PropMaterial)

                        #Criação de materiais do tipo concreto e aço
                        MATERIAL_CONCRETE = 2
                        MATERIAL_STEEL = 1
                        ret   =   SapModel.PropMaterial.ChangeName('Concreto   -   Mesoestrutura',
f'C{f_ck} - Mesoestrutura')
```

```
                        ret = SapModel.PropMaterial.ChangeName('Concreto - Tabuleiro',
f'C{f_ck} - Tabuleiro')
                        ret = SapModel.PropMaterial.SetMaterial('A588Gr50',
eMatType(MATERIAL_STEEL))

                        # 2.2 Atribuição das propriedades mecânicas elásticas aos materiais
                        alfa_i = min(0.8 + 0.2*f_ck/80, 1)
                        E = alfa_i*(5600*f_ck**(1/2))/10 if f_ck <= 50 else (21.5*10^3*(f_ck/10
+ 1,25)^(1/3))/10  # Considerando o uso de granito e gnaisse
                        E_aco = 21000
                        v = 0.2
                        v_aco = 0.3
                        alfa_T = 10**(-5)
                        T_ambiente = 25
                        ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Mesoestrutura',
E, v, alfa_T, T_ambiente)
                        ret = SapModel.PropMaterial.SetMPIsotropic(f'C{f_ck} - Tabuleiro', E,
v, alfa_T, T_ambiente)
                        ret = SapModel.PropMaterial.SetMPIsotropic('A588Gr50', E_aco, v_aco,
alfa_T, T_ambiente)

                        # 2.3 Atribuição do peso específico aos materiais
                        γ_concreto = 25/(100**3)
                        γ_aco = 78.5/(100**3)
                        ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} -
Mesoestrutura', 1, γ_concreto, T_ambiente)
                        ret = SapModel.PropMaterial.SetWeightAndMass(f'C{f_ck} - Tabuleiro',
1, γ_concreto, T_ambiente)
                        ret = SapModel.PropMaterial.SetWeightAndMass('A588Gr50', 1, γ_aco,
T_ambiente)

                        # 2.4 Atribuição das resistência aos materiais e curvas características
de tensão-deformação
                        IsLightweight = False
                        fcsfactor = 1 #The shear strength reduction factor for lightweight
concrete
                        SSType = 0
                        # This is 0, 1 or 2, indicating the stress-strain curve type.
                        # 0 = User defined
                        # 1 = Parametric - Simple
                        # 2 = Parametric - Mander
                        SSHysType = 0
                        # This is 0, 1 or 2, indicating the stress-strain hysteresis type.
                        # 0 = Elastic
                        # 1 = Kinematic
                        # 2 = Takeda
                        StrainAtfc = 0.002#This item applies only to parametric stress-strain
curves. It is the strain at the unconfined compressive strength.
                        StrainUltimate = 0.0035 #This item applies only to parametric stress-
strain curves. It is the ultimate unconfined strain capacity. This item must be larger than the
StrainAtfc item.
                        FinalSlope = 1 #This item applies only to parametric stress-strain
curves. It is a multiplier on the material modulus of elasticity, E. This value multiplied times
E gives the final slope on the compression side of the curve.
                        ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Mesoestrutura',
f_ck/10/1.4, IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate,
FinalSlope)
                        ret = SapModel.PropMaterial.SetOConcrete_1(f'C{f_ck} - Tabuleiro',
f_ck/10/1.4, IsLightweight, fcsfactor, SSType, SSHysType, StrainAtfc, StrainUltimate,
FinalSlope)

                        SSType = 1
                        # This is 0 or 1, indicating the stress-strain curve type.
                        # 0 = User defined
                        # 1 = Parametric - Simple
                        SSHysType = 1
                        # This is 0, 1 or 2, indicating the stress-strain hysteresis type.
                        # 0 = Elastic
                        # 1 = Kinematic
                        # 2 = Takeda
                        StrainAtHardening = 0.015 # This item applies only to parametric stress-
strain curves. It is the strain at the onset of strain hardening.
                        StrainAtMaxStress = 0.11 # This item applies only to parametric stress-
strain curves. It is the strain at maximum stress. This item must be larger than the
StrainAtHardening item.
```

```
                        StrainAtRupture = 0.17 # This item applies only to parametric stress-
strain curves. It is the strain at rupture. This item must be larger than the StrainAtMaxStress
item.
                        FinalSlope = -0.1 # This item applies only to parametric stress-strain
curves. It is a multiplier on the material modulus of elasticity, E. This value multiplied times
E gives the final slope of the curve.
                        # This item is the temperature at which the specified data applies. The
temperature must have been defined previously for the material.
                        ret  =  SapModel.PropMaterial.SetOSteel_1('A588Gr50',  fy_A588Gr50/10,
fu_A588Gr50/10,  fye_A588Gr50/10,  fue_A588Gr50/10,  SSType,  SSHysType,  StrainAtHardening,
StrainAtMaxStress, StrainAtRupture, FinalSlope, T_ambiente)




                        # 3 Definição das propriedades das seções

                        # 3.1 Frame Sections

                        # 3.1.1 Longarinas
                        ret = SapModel.PropFrame.SetISection_1('Perfil 1', 'A588Gr50', t3_1,
t2_1, tf_1, tw_1, t2b_1, tfb_1, FilletRadius_1, -1, '', 'Default')
                        ret = SapModel.PropFrame.SetISection_1('Perfil 2', 'A588Gr50', t3_2,
t2_2, tf_2, tw_2, t2b_2, tfb_2, FilletRadius_2, -1, '', 'Default')
                        ret     =     SapModel.PropFrame.SetISection_1('Diafragma     extremo',
'A588Gr50', t3diaf_1, t2diaf_1, tfdiaf_1, twdiaf_1, t2bdiaf_1, tfbdiaf_1, FilletRadiusdiaf_1, -
1, '', 'Default')
                        ret   =   SapModel.PropFrame.SetISection_1('Diafragma  intermediário',
'A588Gr50', t3diaf_2, t2diaf_2, tfdiaf_2, twdiaf_2, t2bdiaf_2, tfbdiaf_2, FilletRadiusdiaf_2, -
1, '', 'Default')
                        h_long = t3_1

                        # 3.1.2 Pilares
                        Name = f'Pilar (C{f_ck})'
                        MatProp = f'C{f_ck} - Mesoestrutura'
                        Color = -1
                        Notes = ''
                        GUID = 'Default'
                        ret  =  SapModel.PropFrame.SetCircle(Name,  MatProp,  t3_pilar,  Color,
Notes, GUID) # definição da seção

                        Name  =  f'Pilar  (C{f_ck})'  # The name of an existing frame section
property.
                        MatPropLong = 'A615Gr60' # The name of the rebar material property for
the longitudinal rebar.
                        MatPropConfine = 'A615Gr60' # The name of the rebar material property
for the confinement rebar.
                        Pattern = 2
                        # This is either 1 or 2, indicating the rebar configuration.
                        # 1 = Rectangular
                        # 2 = Circular
                        # For circular frame section properties this item must be 2; otherwise
an error is returned.
                        ConfineType = 1
                        # This is either 1 or 2, indicating the confinement bar type.
                        # 1 = Ties
                        # 2 = Spiral
                        #  This  item  applies  only  when  Pattern  =  2.  If  Pattern  =  1,  the
confinement bar type is assumed to be ties.
                        Cover = 4 # The clear cover for the confinement steel (ties). In the
special case of circular reinforcement in a rectangular column, this is the minimum clear cover.
[L]
                        NumberCBars = 14 # This item applies to a circular rebar configuration,
Pattern = 2. It is the total number of longitudinal reinforcing bars in the column.
                        NumberR3Bars  =  int()  #  This  item  applies  to  a  rectangular  rebar
configuration, Pattern = 1. It is the number of longitudinal bars (including the corner bar) on
each face of the column that is parallel to the local 3-axis of the column.
                        NumberR2Bars  =  int()  #  This  item  applies  to  a  rectangular  rebar
configuration, Pattern = 1. It is the number of longitudinal bars (including the corner bar) on
each face of the column that is parallel to the local 2-axis of the column.
                        RebarSize = '25d' # The rebar name for the longitudinal rebar in the
column.
                        TieSize = '8d' # The rebar name for the confinement rebar in the column.
```

```
                        TieSpacingLongit = 20 # The longitudinal spacing of the confinement
bars (ties). [L]
                        Number2DirTieBars = int() # This item applies to a rectangular
reinforcing configuration, Pattern = 1. It is the number of confinement bars (tie legs) running
in the local 2-axis direction of the column.
                        Number3DirTieBars = int() # This item applies to a rectangular
reinforcing configuration, Pattern = 1. It is the number of confinement bars (tie legs) running
in the local 3-axis direction of the column.
                        ToBeDesigned = False # If this item is True, the column longitudinal
rebar is to be designed; otherwise it is to be checked.
                        ret     =     SapModel.PropFrame.SetRebarColumn(Name,     MatPropLong,
MatPropConfine, Pattern, ConfineType, Cover, NumberCBars, NumberR3Bars, NumberR2Bars, RebarSize,
TieSize,
                        TieSpacingLongit, Number2DirTieBars, Number3DirTieBars, ToBeDesigned)

                        # 3.1.3 Travessas
                        ret = SapModel.PropFrame.SetRectangle(f'Travessa encontro (C{f_ck})',
f'C{f_ck} - Mesoestrutura', h_travessaext, bw_travessaext)
                        ret    =    SapModel.PropFrame.SetRectangle(f'Travessa    intermediária
(C{f_ck})', f'C{f_ck} - Mesoestrutura', h_travessaint, bw_travessaint)

                        # 3.2 Area Sections

                        Name = 'Encontro'
                        ShellType = 1
                        IncludeDrillingDOF = True
                        MatProp = f'C{f_ck} - Mesoestrutura'
                        MatAng = 0
                        Thickness = hAla
                        Bending = hAla
                        Color = -1
                        Notes = ''
                        GUID = 'Default'
                        ret       =       SapModel.PropArea.SetShell_1(Name,       ShellType,
IncludeDrillingDOF, MatProp, MatAng, Thickness, Bending, Color, Notes, GUID)

                        Name = 'Laje de transição'
                        ShellType = 1
                        IncludeDrillingDOF = True
                        MatProp = f'C{f_ck} - Tabuleiro'
                        MatAng = 0
                        Thickness = EspessuraLT
                        Bending = EspessuraLT
                        Color = -1
                        Notes = ''
                        GUID = 'Default'
                        ret       =       SapModel.PropArea.SetShell_1(Name,       ShellType,
IncludeDrillingDOF, MatProp, MatAng, Thickness, Bending, Color, Notes, GUID)




                        # 4 Carregamentos

                        # 4.1 Load Patterns/Cases
                        # Wearing surface e sidewalk são carregamentos a serem definidos
especificamente no módulo do CSiBridge (serão definidos posteriormente)
                        # MyType
                        LTYPE_DEAD = 1
                        LTYPE_LIVE = 3
                        LTYPE_WIND= 6
                        LTYPE_OTHER = 8
                        LTYPE_TEMPERATURE = 10
                        LTYPE_BRAKING = 15
                        LTYPE_HORIZONTALEARTHPRESSURE = 20
                        LTYPE_TEMPERATUREGRADIENT = 26
                        LTYPE_WATERLOADPRESSURE = 30
                        LTYPE_IMPACT = 38
                        LTYPE_CONSTRUCTION = 39
                        LTYPE_DEADWEARING = 40
                        SelfWTMultiplier = 0 #The self weight multiplier for the new load
pattern
                        AddLoadCase = True #Associa o load pattern a uma load case
```

```
                        ret              =              SapModel.LoadPatterns.Add('Defensas',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        ret            =            SapModel.LoadPatterns.Add('Pavimentação',
eLoadPatternType(LTYPE_DEADWEARING), SelfWTMultiplier, AddLoadCase)
                        ret            =            SapModel.LoadPatterns.Add('Recapeamento',
eLoadPatternType(LTYPE_DEADWEARING), SelfWTMultiplier, AddLoadCase)
                        ret = SapModel.LoadPatterns.Add('Vento', eLoadPatternType(LTYPE_WIND),
SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    uniforme    pos',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    uniforme    neg',
eLoadPatternType(LTYPE_TEMPERATURE), SelfWTMultiplier, AddLoadCase)
                        ret     =     SapModel.LoadPatterns.Add('Frenagem     e     aceleracao',
eLoadPatternType(LTYPE_BRAKING), SelfWTMultiplier, AddLoadCase)
                        ret         =         SapModel.LoadPatterns.Add('Empuxo         terra',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Empuxo    terra    carga    móvel',
eLoadPatternType(LTYPE_HORIZONTALEARTHPRESSURE), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    gradiente    pos',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
                        ret    =    SapModel.LoadPatterns.Add('Temperatura    gradiente    neg',
eLoadPatternType(LTYPE_TEMPERATUREGRADIENT), SelfWTMultiplier, AddLoadCase)
                        ret                =                SapModel.LoadPatterns.Add('Água',
eLoadPatternType(LTYPE_WATERLOADPRESSURE), SelfWTMultiplier, AddLoadCase)
                        ret              =              SapModel.LoadPatterns.Add('Impacto',
eLoadPatternType(LTYPE_IMPACT), SelfWTMultiplier, AddLoadCase)
                        ret               =               SapModel.LoadPatterns.Add('Haunch',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        ret              =              SapModel.LoadPatterns.Add('SIPForm',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        #     ret     =     SapModel.LoadPatterns.Add('ConcPour     Permanent',
eLoadPatternType(LTYPE_DEAD), SelfWTMultiplier, AddLoadCase)
                        #     ret     =     SapModel.LoadPatterns.Add('ConcPour     Temporary',
eLoadPatternType(LTYPE_CONSTRUCTION), SelfWTMultiplier, AddLoadCase)

                        # 4.2 Load Distributions

                        # 4.2.1 Gradiente de temperatura
                        ret                                                                =
MyBridgeModeler.LoadDistribution.TemperatureGradient.AddNew('Temperatura gradiente')

                        Name  =  'Temperatura  gradiente'  #The  name  of  an  existing  bridge
temperature gradient load distribution.
                        NumberDistances = 4 #The number of specified distances (between 1 and
5, inclusive).
                        DistanceThatMayVary = 3 #The specified distance that may vary.
                        TempDifferencesToConsider = 0
                        # The temperature differences to consider.
                        # 0 = Positive and Negative
                        # 1 = Positive Only
                        # 2 = Negative Only
                        d_1 = min(0.3*(h_laje + h_long), 15)
                        d_2 = min(max(0.3*(h_laje + h_long), 10), 25)
                        d_3 = min(min(0.3*(h_laje + h_long), 10 + 10), (h_laje + h_long)-d_1-
d_2)
                        if (h_laje + h_long) < 20:
                            ΔT_1 = 8.5
                            ΔT_2 = 3.5
                            ΔT_3 = 0.5
                        elif 20 < (h_laje + h_long) <= 40:
                            ΔT_1 = np.interp((h_laje + h_long), [20, 40], [8.5, 12])
                            ΔT_2 = np.interp((h_laje + h_long), [20, 40], [3.5, 3])
                            ΔT_3 = np.interp((h_laje + h_long), [20, 40], [0.5, 1.5])
                        elif 40 < (h_laje + h_long) <= 60:
                            ΔT_1 = np.interp((h_laje + h_long), [40, 60], [12, 13])
                            ΔT_2 = np.interp((h_laje + h_long), [40, 60], [3, 3])
                            ΔT_3 = np.interp((h_laje + h_long), [40, 60], [1.5, 2])
                        elif 60 < (h_laje + h_long) < 80:
                            ΔT_1 = np.interp((h_laje + h_long), [60, 80], [13, 13.5])
                            ΔT_2 = np.interp((h_laje + h_long), [60, 80], [3, 3])
                            ΔT_3 = np.interp((h_laje + h_long), [60, 80], [2, 2.5])
                        else:
                            ΔT_1 = 13.5
                            ΔT_2 = 3.0
                            ΔT_3 = 2.5
```

```
                        d = [d_1, d_2, float(), d_3] #A zero-based one-dimensional array
dimensioned to (NumberDistances - 1) containing the specified distances. [L]
                        TPos = [ΔT_1, ΔT_2, 0, 0, ΔT_3] #A zero-based one-dimensional array
dimensioned to (NumberDistances) containing the specified positive temperatures. [T]
                        TNeg = [-5, -5, 0, 0, 0] #A zero-based one-dimensional array dimensioned
to (NumberDistances) containing the specified negative temperatures. [T]
                        ret                                                              =
MyBridgeModeler.LoadDistribution.TemperatureGradient.SetUserGeneralData(Name, NumberDistances,
DistanceThatMayVary, TempDifferencesToConsider, d, TPos, TNeg)

                        # 4.2.2 Vento

                        # Força distribuída linearmente
                        Name = 'Vento força'
                        ret = MyBridgeModeler.LoadDistribution.Line.AddNew(Name)

                        Name = Name # The name of an existing bridge line load distribution
                        LoadType = 1 # The line load type
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 5
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LoadValue = ws_acoes['J309'].value/100 # The load value. [F/L] for
force; [F] for moment
                        ReferenceLocation = 2
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LoadDistance = 0 # The distance from the reference location to the
load. [L]
                        ret = MyBridgeModeler.LoadDistribution.Line.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LoadValue, ReferenceLocation, LoadDistance)

                        # Momento distribuído linearmente
                        Name = 'Vento momento'
                        ret = MyBridgeModeler.LoadDistribution.Line.AddNew(Name)

                        Name = Name
                        LoadType = 2 # The line load type
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 4
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LoadValue = -ws_acoes['J310'].value # The load value. [F/L] for force;
[F] for moment
                        ReferenceLocation = 2
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LoadDistance = 0 # The distance from the reference location to the
load. [L]
                        ret = MyBridgeModeler.LoadDistribution.Line.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LoadValue, ReferenceLocation, LoadDistance)

                        # 4.2.3 Frenagem e aceleração
                        Name = 'Frenagem e aceleracao'
                        ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)
```

```
                        Name = Name # The name of an existing bridge area load distribution.
                        LoadType = 1 # The area load type
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 4
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LeftEdgeValue = 0.32E-04 # The left edge load value. [F/L2] for force;
[F/L] for moment
                        RightEdgeValue = 0.32E-04 # The right edge load value. [F/L2] for force;
[F/L] for moment
                        LeftReferenceLocation = 1 #The left reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LeftLoadDistance = 0 # The distance from the left edge of load to the
left reference location. [L]
                        RightReferenceLocation = 2 # The right reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        RightLoadDistance = 0 # The distance from the right edge of load to the
right reference location. [L]
                        ret = MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
                        LeftReferenceLocation,    LeftLoadDistance,    RightReferenceLocation,
RightLoadDistance)

                        # 4.2.4 Pavimentação
                        Name = 'Pavimentação'
                        ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

                        Name = Name # The name of an existing bridge area load distribution.
                        LoadType = 1 # The area load type
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 10
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LeftEdgeValue = 2.4E-04 # The left edge load value. [F/L2] for force;
[F/L] for moment
                        RightEdgeValue = 2.4E-04 # The right edge load value. [F/L2] for force;
[F/L] for moment
                        LeftReferenceLocation = 1 #The left reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LeftLoadDistance = 40 # The distance from the left edge of load to the
left reference location. [L]
                        RightReferenceLocation = 2 # The right reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        RightLoadDistance = 40 # The distance from the right edge of load to
the right reference location. [L]
                        ret = MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
                        LeftReferenceLocation,    LeftLoadDistance,    RightReferenceLocation,
RightLoadDistance)
```

```
                        # 4.2.5 Recapeamento
                        Name = 'Recapeamento'
                        ret = MyBridgeModeler.LoadDistribution.Area.AddNew(Name)

                        Name = Name # The name of an existing bridge area load distribution.
                        LoadType = 1 # The area load type
                        # 1 = Force
                        # 2 = Moment
                        CoordSystem = 'GLOBAL' # This is either Local or the name of a defined
coordinate system.
                        Direction = 10
                        # The load direction.
                        # 1 = 1; 2 = 2; 3 = 3
                        # 4 = X; 5 = Y; 6 = Z
                        # 7 = X Projected; 8 = Y Projected; 9 = Z Projected
                        # 10 = Gravity; 11 = Gravity Projected
                        # Directions 1, 2 and 3 only apply when the CoordSystem is Local.
                        # Direction 4, 5, 6, 7, 8, 9, 10 and 11 do NOT apply when the CoordSystem
is Local.
                        # Directions 10 and 11 only apply when the CoordSystem is Global.
                        LeftEdgeValue = 2E-04 # The left edge load value. [F/L2] for force;
[F/L] for moment
                        RightEdgeValue = 2E-04 # The right edge load value. [F/L2] for force;
[F/L] for moment
                        LeftReferenceLocation = 1 #The left reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        LeftLoadDistance = 40 # The distance from the left edge of load to the
left reference location. [L]
                        RightReferenceLocation = 2 # The right reference location.
                        # 1 = Left edge of deck
                        # 2 = Right edge of deck
                        RightLoadDistance = 40 # The distance from the right edge of load to
the right reference location. [L]
                        ret = MyBridgeModeler.LoadDistribution.Area.SetLoadDistribution(Name,
LoadType, CoordSystem, Direction, LeftEdgeValue, RightEdgeValue,
                        LeftReferenceLocation,    LeftLoadDistance,    RightReferenceLocation,
RightLoadDistance)




                        # 5 Criação do modelo da ponte - (cBridgeModeler_1.NewBridgeModel)

                        StartStation = 0 #The starting station for the bridge. [L]
                        InitialBearing = "N900000E" #The initial bearing for the bridge. This
should be a valid bearing angle such as “S624355E”.
                        Grade = 0 #The grade for the bridge specified in percent. A positive
grade represents an incline with increasing station.
                        Skew = "Default"
                        #The skew angle for the bridge. This should be a valid bearing angle
such as “S624355E”, “Default” or “DefaultX”
                        #where X is a positive or negative number indicating the number of
degrees the bearing is different from the default skew.
                        #The default skew is perpendicular to the reference line.
                        Radius = 0 #The bridge radius. A positive radius curves to the right
and a negative radius curves to the left. A radius of zero means an infinite radius (i.e.,
straight). [L]
                        SuperElevation = 0 #The bridge superelevation specified in percent. A
positive value rotates clockwise corresponding to a curve right and a negative value rotates
counterclockwise corresponding to a curve left.
                        SectionType = 9 # Diferença na numeração do pc do lab e o daqui de
casa?
                        #This is an integer between 2 and 16, inclusive, indicating the bridge
section type.
                        # 2 = Concrete Box Girder - Exterior Girders Vertical
                        # 3 = Concrete Box Girder - Exterior Girders Sloped
                        # 4 = Concrete Box Girder - Exterior Girders Clipped
                        # 5 = Concrete Box Girder - Exterior Girders with Radius
                        # 6 = Concrete Box Girder - Exterior Girders Sloped Maximum
                        # 7 = Concrete Tee Beam
                        # 8 = Concrete Flat Slab
                        # 9 = Steel I Girder
                        # 10 = Concrete Box Girder - AASHTO-PCI-ASBI Standard
```

```
                        # 11 = Precast Concrete I Girder
                        # 12 = Precast Concrete U Girder
                        # 13 = Concrete Box Girder - Advanced
                        # 14 = Steel U Girder
                        # 15 = Concrete Solid Girder
                        # 16 = Precast Concreter Super-T Girder
                        ret  =  MyBridgeModeler.NewBridgeModel(StartStation,  InitialBearing,
Grade, Skew, Radius, SuperElevation, SpanLength, SectionType)

                        # 5.1 Objetos de interesse
                        NumberNames = int()
                        MyName = []
                        ret = MyBridgeModeler.BridgeObject.GetNameList(NumberNames, MyName)
                        myBridgeObjName = list(ret[2])[0]

                        NumberNames = int()
                        MyName = []
                        BridgeSectionType = [SectionType]
                        ret  =  MyBridgeModeler.DeckSection.GetNameList(NumberNames,  MyName,
BridgeSectionType)
                        myDeckSectionName_1 = list(ret[2])[0]
                        ret   =   MyBridgeModeler.DeckSection.ChangeName(myDeckSectionName_1,
'SeçãoMista-Perfil1')
                        myDeckSectionName_1 = 'SeçãoMista-Perfil1'

                        Name = myBridgeObjName
                        LayoutLine = ''
                        ret = MyBridgeModeler.BridgeObject.GetLayoutLine(Name, LayoutLine)
                        myLayoutLineName = ret[1]

                        ret = MyBridgeModeler.Lane.GetNameList(int(), [], [])
                        myLaneName = list(ret[2])[0]

                        # 5.2 Definição do Layout Line - (cBMLayoutLine.SetGeneralData)
                        Name = myLayoutLineName
                        CoordinateSystem = 'GLOBAL'
                        StartX = 0
                        StartY = 0
                        StartZ = 0
                        ret          =         MyBridgeModeler.LayoutLine.SetGeneralData(Name,
CoordinateSystem, StartX, StartY, StartZ)

                        # 5.3 Bridge Object Data (cBMBridgeObject.SetSpanLayoutByLength Method)
                        Name = myBridgeObjName
                        StartStation = 0
                        ret     =     MyBridgeModeler.BridgeObject.SetSpanLayoutByLength(Name,
StartStation, SpanName, SpanLength)

                        # 5.4 Veiculos (cBMVehicle Methods)
                        Region = 'User'
                        Standard = 'User'
                        Vehicle = 'All'
                        ret = MyBridgeModeler.Vehicle.ImportVehicle(Region, Standard, Vehicle)
                        ret = MyBridgeModeler.Vehicle.ConvertToUser('Multidao')

                        #  Correção  (carregamento  de  Multidao  que  depende  da  largura  do
tabuleiro)

                        Name = 'Multidao' # The name of the vehicle.
                        UniformLoadSF = 1 # The uniform load scale factor.
                        AxleLoadSF = 1 # The axle load scale factor.
                        NumberLoadDataSets = 2 # The number of load data sets defined for the
vehicle.
                        LoadLengthType = [1, 2]
                        # A zero-based one-dimensional array containing the load length type
for each load data set.
                        # 1 = Leading Load
                        # 2 = Trailing Load
                        # 3 = Fixed Length
                        # 4 = Variable Length
                        MinimumDistance  =  [0,  0]  #  A  zero-based  one-dimensional  array
containing the minimum distance between the current and previous axle. This item is only
applicable if the LoadLengthType is 3 or 4 (Fixed Length or Variable Length). [L]
                        MaximumDistance = [float(), float()] # A zero-based one-dimensional
array containing the maximum distance between the current and previous axle. This item is only
```

```
applicable if the LoadLengthType is 4 (Variable Length). This item must either be zero (meaning
infinity) or it must be greater than or equal to MinimumDistance item. [L]
                        UniformLoad             =             [(LarguraRolamento)/100*(5/100),
(LarguraRolamento)/100*(5/100)] # A zero-based one-dimensional array containing the uniform
load. [F/L]
                        UniformWidthType = [2, 2]
                        # A zero-based one-dimensional array containing the width type for the
uniform load.
                        # 1 = Zero Width
                        # 2 = Lane Width
                        # 3 = Fixed Width
                        UniformWidth = [float(), float()] # A zero-based one-dimensional array
containing the uniform load width. This item is only applicable if the UnifWidthType is 3 (Fixed
Width). [L]
                        AxleLoad = [0, float()] # A zero-based one-dimensional array containing
the total load for the axle. This item does not apply when the LoadLengthType item is 4 (Variable
Length). [F]
                        AxleWidthType = [1, int()]
                        # A zero-based one-dimensional array containing the axle width type
indicating how the axle load is to be distributed. This item does not apply when the
LoadLengthType item is 4 (Variable Length).
                        # 1 = One Point
                        # 2 = Two Points
                        # 3 = Four Points
                        # 4 = Lane Width Line
                        # 5 = Fixed Width Line
                        AxleWidth = [float(), float()] # A zero-based one-dimensional array
containing the axle width. This item is only applicable if the AxleWidthType item is 2, 3 or 5
(Two Points, Four Points or Fixed Width Line).
                        # This item does not apply when the LoadLengthType item is 4 (Variable
Length). [L]
                        ret  =  MyBridgeModeler.Vehicle.SetVerticalLoads(Name,  UniformLoadSF,
AxleLoadSF, NumberLoadDataSets, LoadLengthType, MinimumDistance, MaximumDistance, UniformLoad,
UniformWidthType, UniformWidth, AxleLoad, AxleWidthType, AxleWidth)

                        ret = SapModel.LoadCases.Delete('MOVE1')
                        ret = SapModel.LoadCases.Moving.SetCase('Veiculo')
                        ret = SapModel.LoadCases.Moving.SetCase('Multidao')

                        ret = SapModel.LoadCases.Moving.SetInitialCase('Veiculo', '')
                        ret = SapModel.LoadCases.Moving.SetInitialCase('Multidao', '')

                        NumberLoads = 1 #The number of loads assigned to the specified analysis
case.
                        MyClass = ['Veiculo'] #This is an array that includes the vehicle class
for each load assigned to the load case.
                        SF = [1] #This is an array that includes the scale factor for each load
assigned to the load case.
                        Min = [0] #This is an array that includes the minimum number of lanes
loaded for each load assigned to the load case.
                        Max = [0] #This is an array that includes the maximum number of lanes
loaded for each load assigned to the load case. This item must be 0, or it must be greater than
or equal to Min. If this item is 0, all available lanes are loaded.
                        ret   =   SapModel.LoadCases.Moving.SetLoads('Veiculo',   NumberLoads,
MyClass, SF, Min, Max)
                        MyClass = ['Multidao']
                        ret   =   SapModel.LoadCases.Moving.SetLoads('Multidao',   NumberLoads,
MyClass, SF, Min, Max)

                        LoadNumber = 1 #The load assignment number.
                        NumberItems = 1 #The number of lanes loaded for the specified load
assignment number.
                        MyName = [f'{myLaneName}'] #This is an array that includes the name of
each lane loaded for the specified load assignment number.
                        ret = SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo', LoadNumber,
NumberItems, MyName)
                        ret = SapModel.LoadCases.Moving.SetLanesLoaded('Multidao', LoadNumber,
NumberItems, MyName)

                        # TB- 450

                        Name = 'TB-450' # The name of a new load combination.
                        ComboType = 0
                        # This is 0, 1, 2, 3 or 4 indicating the load combination type.
                        # 0 = Linear Additive
```

```
                        # 1 = Envelope
                        # 2 = Absolute Additive
                        # 3 = SRSS
                        # 4 = Range Additive
                        ret = SapModel.RespCombo.Add(Name, ComboType)

                        CName = ['Veiculo', 'Multidao']
                        for i in range(2):
                            CNameType = eCNameType(0)
                            # This is one of the following items in the eCNameType enumeration:
                            # LoadCase = 0
                            # LoadCombo = 1
                            # The name of the load case or load combination to be added to or
modified in the combination specified by the Name item.
                            # If the load case or combination already exists in the combination
specified by the Name item, the scale factor is modified as indicated by the SF item for that
load case or combination.
                            #If the analysis case or combination does not exist in the
combination specified by the Name item, it is added.
                            SF = 1 # The scale factor multiplying the case or combination
indicated by the CName item.
                            ret = SapModel.RespCombo.SetCaseList(Name, CNameType, CName[i],
SF)




                        # 6 Aparelhos de apoio a partir de um link definido pelo usuário
(cBMBearing.SetBearingUser Method)

                        # 6.1 Aparelho de apoio

                        Name = 'Elastômero' #The name of an existing or new link property. If
this is an existing property, that property is modified; otherwise, a new property is added.
                        DOF = [True, True, True, False, False, False]
                        # This is a boolean array, dimensioned to 5, indicating if properties
exist for a specified degree of freedom.
                        # DOF(0) = U1
                        # DOF(1) = U2
                        # DOF(2) = U3
                        # DOF(3) = R1
                        # DOF(4) = R2
                        # DOF(5) = R3
                        Fixed = [True, False, False, False, False, False]
                        # This is a boolean array, dimensioned to 5, indicating if the specified
degree of freedom is fixed (restrained).
                        # Fixed(0) = U1 fixity if DOF(0) = True
                        # Fixed(1) = U2 fixity if DOF(1) = True
                        # Fixed(2) = U3 fixity if DOF(2) = True
                        # Fixed(3) = R1 fixity if DOF(3) = True
                        # Fixed(4) = R2 fixity if DOF(4) = True
                        # Fixed(5) = R3 fixity if DOF(5) = True
                        Ke = [float(), K_ap, K_ap, float(), float(), float()]
                        # This is an array of stiffness terms for the link property. There are
6 terms in the array if the stiffness is uncoupled and 21 if it is coupled. The KeCoupled item
indicates if the stiffness is coupled.
                        # If the stiffness is uncoupled:
                        # Ke(0) = U1 [F/L]
                        # Ke(1) = U2 [F/L]
                        # Ke(2) = U3 [F/L]
                        # Ke(3) = R1 [FL]
                        # Ke(4) = R2 [FL]
                        # Ke(5) = R3 [FL]
                        # If the stiffness is coupled:
                        # Ke(0) = U1U1 [F/L]
                        # Ke(1) = U1U2 [F/L]
                        # Ke(2) = U2U2 [F/L]
                        # Ke(3) = U1U3 [F/L]
                        # Ke(4) = U2U3 [F/L]
                        # Ke(5) = U3U3 [F/L]
                        # Ke(6) = U1R1 [F]
                        # Ke(7) = U2R1 [F]
                        # Ke(8) = U3R1 [F]
                        # Ke(9) = R1R1 [FL]
```

```
                        # Ke(10) = U1R2 [F]
                        # Ke(11) = U2R2 [F]
                        # Ke(12) = U3R2 [F]
                        # Ke(13) = R1R2 [FL]
                        # Ke(14) = R2R2 [FL]
                        # Ke(15) = U1R3 [F]
                        # Ke(16) = U2R3 [F]
                        # Ke(17) = U3R3 [F]
                        # Ke(18) = R1R3 [FL]
                        # Ke(19) = R2R3 [FL]
                        # Ke(20) = R3R3 [FL]
                        Ce = [float(), 0, 0, float(), float(), float()]
                        # This is an array of damping terms for the link property. There are 6
terms in the array if the damping is uncoupled and 21 if it is coupled. The CeCoupled item
indicates if the damping is coupled.
                        # If the damping is uncoupled:
                        # Ce(0) = U1 [F/L]
                        # Ce(1) = U2 [F/L]
                        # Ce(2) = U3 [F/L]
                        # Ce(3) = R1 [FL]
                        # Ce(4) = R2 [FL]
                        # Ce(5) = R3 [FL]
                        # If the damping is coupled:
                        # Ce(0) = U1U1 [F/L]
                        # Ce(1) = U1U2 [F/L]
                        # Ce(2) = U2U2 [F/L]
                        # Ce(3) = U1U3 [F/L]
                        # Ce(4) = U2U3 [F/L]
                        # Ce(5) = U3U3 [F/L]
                        # Ce(6) = U1R1 [F]
                        # Ce(7) = U2R1 [F]
                        # Ce(8) = U3R1 [F]
                        # Ce(9) = R1R1 [FL]
                        # Ce(10) = U1R2 [F]
                        # Ce(11) = U2R2 [F]
                        # Ce(12) = U3R2 [F]
                        # Ce(13) = R1R2 [FL]
                        # Ce(14) = R2R2 [FL]
                        # Ce(15) = U1R3 [F]
                        # Ce(16) = U2R3 [F]
                        # Ce(17) = U3R3 [F]
                        # Ce(18) = R1R3 [FL]
                        # Ce(19) = R2R3 [FL]
                        dj2 = 0 #The distance from the J-End of the link to the U2 shear spring.
This item applies only when DOF(1) = True. [L]
                        dj3 = 0 #The distance from the J-End of the link to the U3 shear spring.
This item applies only when DOF(2) = True. [L]
                        KeCoupled = False #This item is True if the link stiffness, Ke, is
coupled. There are 21 terms in the Ke array if Ke is coupled; otherwise there are 6 terms.
                        CeCoupled = False #This item is True if the link damping, Ce, is
coupled. There are 21 terms in the Ce array if Ce is coupled; otherwise there are 6 terms.
                        Notes = '' #The notes, if any, assigned to the property.
                        GUID = '' #The GUID (global unique identifier), if any, assigned to the
property. If this item is input as Default, the program assigns a GUID to the property.
                        ret = SapModel.PropLink.SetLinear(Name, DOF, Fixed, Ke, Ce, dj2, dj3,
KeCoupled, CeCoupled, Notes, GUID) #Criação do link Neoprene

                        ret = MyBridgeModeler.Bearing.AddNew('Ap. de apoio')
                        ret    =    MyBridgeModeler.Bearing.SetBearingLink('Ap.    de    apoio',
'Elastômero')




                        #  7.  Definição  dos  apoios  extremos,  intermediários  (travessas),
abutments e atribuições

                        ret = MyBridgeModeler.Bent.AddNew('Apoios extremos')
                        ret = MyBridgeModeler.Bent.AddNew('Apoios intermediários')
                        ret = MyBridgeModeler.Abutment.AddNew('Apoio aparelho de apoio')

                        # 7.1 Definição - Travessas extremas

                        Name = 'Apoios extremos' #The name of an existing bridge bent.
```

```
                        CapBeamSection = f'Travessa encontro (C{f_ck})' #This is either "None"
or the name of the frame section property assigned to the cap beam.
                        Foundation = 'Fixed' #Containing either Fixed, Pinned, or a defined
foundation spring.
                        BentType = 0
                        # 0 = Single bearing line (continuous superstructure)
                        # 1 = Double bearing line (discontinuous superstructure)
                        # 2 = Double bearing line with closure pour
                        ret    =    MyBridgeModeler.Bent.SetGeneralData(Name,    CapBeamSection,
C_travessaext, SupportType_ext, N_pilaresext, Foundation, BentType)

                        Section = [f'Pilar (C{f_ck})'] * N_pilaresext #A zero-based one-
dimensional array dimensioned to (NumberColumns - 1) containing the frame section properties
assigned to each column.
                        Angle = [0] * N_pilaresext #A zero-based one-dimensional array
dimensioned to (NumberColumns - 1) containing the orientation of the column in degrees
counterclockwise between a line parallel to the bent and the column local 2 axis. [deg]
                        Foundation = ['Fixed'] * N_pilaresext #A zero-based one-dimensional
array dimensioned to (NumberColumns - 1) containing either Fixed, Pinned, a defined foundation
spring, or Previous. Note that an error will occur if Previous is assigned to the first column.
                        ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_ext,
Height_ext, Angle, Foundation)

                        SuperConnection = 1
                        # The superstructure connection.
                        # 0 = Connect to girder top and bottom
                        # 1 = Connect to girder bottom only
                        ret     =     MyBridgeModeler.Bent.SetSuperConnectionSingleLine(Name,
SuperConnection)

                        # 7.2 Definição - Travessas intermediárias
                        Name = 'Apoios intermediários'
                        CapBeamSection = f'Travessa intermediária (C{f_ck})'
                        Foundation = 'Fixed'
                        BentType = 1
                        ret    =    MyBridgeModeler.Bent.SetGeneralData(Name,    CapBeamSection,
C_travessaint, SupportType_int, N_pilaresint, Foundation, BentType)

                        Section = [f'Pilar (C{f_ck})'] * N_pilaresint
                        Angle = [0] * N_pilaresint
                        Foundation = ['Fixed'] * N_pilaresint
                        ret = MyBridgeModeler.Bent.SetColumnData(Name, Section, Distance_int,
Height_int, Angle, Foundation)

                        BeforeSuperConnection = 1
                        # The superstructure connection before the bent.
                        # 0 = Connect to girder top and bottom
                        # 1 = Connect to girder bottom only
                        BeforeDistanceToBearingLine = d_ap #The distance to the bearing line
before the bent. [L]
                        BeforeDistanceToEndOfGirder = 0 #The distance to the end of the girder
before the bent. [L]
                        AfterSuperConnection = 1
                        # The superstructure connection after the bent.
                        # 0 = Connect to girder top and bottom
                        # 1 = Connect to girder bottom only
                        AfterDistanceToBearingLine = d_ap #The distance to the bearing line
after the bent. [L]
                        AfterDistanceToEndOfGirder = 0 #AfterDistanceToEndOfGirder
                        ret     =     MyBridgeModeler.Bent.SetSuperConnectionDoubleLine(Name,
BeforeSuperConnection,       BeforeDistanceToBearingLine,       BeforeDistanceToEndOfGirder,
AfterSuperConnection, AfterDistanceToBearingLine, AfterDistanceToEndOfGirder)

                        # 7.3 Definição - Apoio aparelho de apoio (apenas para análises de
superestruturas)
                        Name = 'Apoio aparelho de apoio' # The name of an existing bridge
abutment.
                        SuperConnection = 1
                        # The superstructure connection.
                        # 0 = Integral
                        # 1 = Connect to Girder Bottom Only
                        DistanceFromBridgeEnd = float() # The distance from the bridge end to
the bearing line. [L]
                        FoundationProperty = 'Fixed' # This is either Fixed, Pinned or the name
of a defined foundation spring.
```

```
                        SubstructureType = 1
                        # The substructure type.
                        # 0 = None
                        # 1 = Foundation spring
                        # 2 = Continuous beam (continuously supported)
                        BeamProperty = '' # The frame section property assigned to the
continuous beam. This item only applies when the SubstructureType item is 1, Continuous beam
(continuously supported).
                        BeamLength = float() # The length of the continuous beam. This item
only applies when the SubstructureType item is 1, Continuous beam (continuously supported). [L]
                        ret = MyBridgeModeler.Abutment.SetAbutment(Name, SuperConnection,
DistanceFromBridgeEnd, FoundationProperty, SubstructureType, BeamProperty, BeamLength)

                        # 7.4 Atribuições

                        Name = myBridgeObjName #The name of a bridge object.

                        if NVãos != 1:
                            for i in range(NVãos-1):
                                SpanNameBeforeBent = SpanName[2 + i*3] #The name of the span
before the considered bent.
                                BentProperty = 'Apoios intermediários' #The bent property
assigned to the specified bent.
                                BentDirection = 'Default' #The bearing of a line parallel to
the bent. It is either Default or a valid bearing string.
                                ret =
MyBridgeModeler.BridgeObject.BentAssign.SetBentAssignment(Name, SpanNameBeforeBent,
BentProperty, BentDirection)

                                BentSupportName = SpanName[2 + i*2]
                                ret =
MyBridgeModeler.BridgeObject.BentAssign.SetSupportName(Name, SpanNameBeforeBent,
BentSupportName)

                                SuperstructureIsContinousAtBent = False #Indicates whether the
superstructure is continuous (True) or discontinuous (False) at the bent.
                                DiaphragmBeforeBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the diaphragm property at the bent.
                                # If the specified bent property is None or it has a single
line of bearings (continuous superstructure) this is the property of the diaphragm that occurs
at the bent.
                                # If the specified bent property has a double line of bearings
(discontinuous superstructure) this is the property of the diaphragm that occurs just before
the bent.
                                # This item does not apply if the specified bent property is a
double bearing line with closure pour.
                                InternalDiaphragmBeforeBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the internal diaphragm property at the bent.
                                # This only applies to bents at steel U girder or super T girder
deck sections.
                                # If the specified bent property is None or it has a single
line of bearings (continuous superstructure) this is the property of the internal diaphragm that
occurs at the bent.
                                # If the specified bent property has a double line of bearings
this is the property of the internal diaphragm that occurs just before the bent.
                                DiaphragmAfterBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the diaphragm property after the bent.
                                # It applies when the specified bent property has a double line
of bearings (discontinuous superstructure).
                                # This item does not apply if the specified bent property is a
double bearing line with closure pour.
                                InternalDiaphragmAfterBent = 'None'
                                # This is either None or the name of a defined bridge diaphragm
property indicating the internal diaphragm property after the bent.
                                # This only applies to bents at steel U girder or super T girder
deck sections.
                                # It applies when the specified bent property has a double line
of bearings and is not a closure pour.
                                ClosurePourDiaphragm = 'None' #This is the name of a defined
closure pour bridge diaphragm property. This item only applies when the specified bent type is
Double Bearing Line with Closure Pour.
```

```
                                Restrainer = 'None' #This is either None or the name of a
defined bridge diaphragm property indicating the restrainers used at the bent. It only applies
when the superstructure is discontinuous at the bent.
                                RestrainerElevation = 0 #The restrainer elevation (at layout
line, global Z). this item only applies when the superstructure is discontinuous at the bent.
[L]
                                MeshSuperstructureToMatchBent = False
                                # This is a flag indicating if the superstructure is to be
meshed to match the bent.
                                # This item only applies when the superstructure is continuous
at the bent.
                                MeshSteelUGirderToMatchBent = False
                                # This is a flag indicating if the steel U girders are to be
meshed to match the bent.
                                # This item only applies to bents at steel U girder or super T
girder deck sections when the superstructure is continuous at the bent.
                                ret                                                         =
MyBridgeModeler.BridgeObject.BentAssign.SetSuperstructureAssignment(Name,   SpanNameBeforeBent,
SuperstructureIsContinousAtBent,       DiaphragmBeforeBent,       InternalDiaphragmBeforeBent,
DiaphragmAfterBent, InternalDiaphragmAfterBent,
                                ClosurePourDiaphragm,     Restrainer,     RestrainerElevation,
MeshSuperstructureToMatchBent, MeshSteelUGirderToMatchBent)

                                Elevation = -h_laje -t3_1 - h_ap #The global Z elevation of the
bent. [L]
                                HorizontalOffset = 0 #The horizontal offset from the layout
line to the midlength of the cap beam. [L]
                                ret                                                         =
MyBridgeModeler.BridgeObject.BentAssign.SetBentLocation(Name,  SpanNameBeforeBent,  Elevation,
HorizontalOffset)

                                for j in range(2):
                                    BearingLocation = [1, 2]
                                    # This is one of the following indicating the bent bearing
location:
                                    # 1 = Single line of bearings or bearings before bent for
double line of bearings.
                                    # 2 = Bearings after bent for double line of bearings.
                                    BearingAssignmentType = 1
                                    # This is one of the following indicating the bearing
assignment type:
                                    # 1 = Girder-by-girder
                                    # 2 = General
                                    # The General assignment type only applies to abutments at
concrete box girders deck sections or concrete flat slab deck sections. General assignments to
abutments at other deck section types will return an error.
                                    BearingProperty = 'Ap. de apoio' #The name of a defined
bearing property assigned to the bent.
                                    RestrainerProperty = 'None' #This is either None or the
name of a defined restrainer property assigned to the bent.
                                    ElevationAtLayoutLine = -h_laje -t3_1 - h_ap/2 #The bearing
elevation (global Z) at the layout line. [L]
                                    RotationAngle = 0 #The bearing rotation angle from the
bridge default.
                                    NumberOfBearings = int()
                                    #The number of bearings for the bridge section or for
abutments at Steel U Girder deck sections the number of bearings at the U girder.
                                    # For bearings at steel U girder sections this item must
be 1 or 2.
                                    # This item only applies when the bearing assignment type
is General or when the abutment is at a Steel U Girder deck section.
                                    UniformSpacing = int() #The uniform spacing of bearings.
This item only applies when the bearing assignment type is General. [L]
                                    BearingOffset  =  int()  #The  offset  from  the  section
reference point to the bearing center. This item only applies when the bearing assignment type
is General. [L]
                                    ret                                                     =
MyBridgeModeler.BridgeObject.BentAssign.SetBearingAssignment(Name,          SpanNameBeforeBent,
BearingLocation[j],      BearingAssignmentType,      BearingProperty,      RestrainerProperty,
ElevationAtLayoutLine, RotationAngle, NumberOfBearings,
                                    UniformSpacing, BearingOffset)

                        ValidBentsList = ['Apoios intermediários', 'Apoios extremos']
                        NumberNames = int()
                        MyNames = []
```

```
                        ret = MyBridgeModeler.Bent.GetNameList(NumberNames, MyNames)
                        BentsList = list(ret[2])
                        for i in range (ret[1]):
                            if BentsList[i] not in ValidBentsList:
                                ret = MyBridgeModeler.Bent.Delete(BentsList[i])

                        # Atribuição da continuidade entre perfis - GetTableForEditingArray
                        ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0,
False, 1, 1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

                        TableKey = 'Bridge Object Definitions 07 - Bents'
                        GroupName = ''
                        TableVersion = int()
                        FieldKeysIncluded = []
                        NumberRecords = int()
                        TableData = []
                        ret     =     SapModel.DatabaseTables.GetTableForEditingArray(TableKey,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ColumnLabels = list(ret[2])
                        Resultados = list(ret[4])

                        # Criação do dataframe - trocar BENT1 pela continuidade

                        ListadeListasDosResultados = chunk_list(Resultados, len(ColumnLabels))
                        df_bents           =           pd.DataFrame(ListadeListasDosResultados,
columns=ColumnLabels)
                        df_bents = df_bents.replace({',' : '.'}, regex = True) # Trocar vírgula
por ponto em todas as colunas
                        df_bents = df_bents.replace(to_replace='BENT1', value = 'Free')
                        result = df_bents.values.ravel().tolist()

                        TableKey = 'Bridge Object Definitions 07 - Bents'
                        TableVersion = int()
                        FieldKeysIncluded = ColumnLabels
                        NumberRecords = int()
                        TableData = result
                        ret     =     SapModel.DatabaseTables.SetTableForEditingArray(TableKey,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ret  =  SapModel.DatabaseTables.ApplyEditedTables(False,  int(),  int(),
int(), int(), '')




                        # 8 Definição e atribuição dos diafragmas

                        Name = myBridgeObjName
                        ret = MyBridgeModeler.BridgeObject.DiaphragmAssign.DeleteAll(Name, '')

                        NumberNames = int() #The number of diaphragm names retrieved by the
program.
                        MyName = [] #This is a one-dimensional array of diaphragm names. The
MyName array is created as a dynamic, zero-based array by the API user: Dim MyName() as String.
                        #The array is dimensioned to (NumberNames - 1) inside the program,
filled with the names, and returned to the API user.
                        MyType = []
                        # This is a one-dimensional array, dimensioned to (NumberNames - 1),
of diaphragm types.
                        # 1 = Solid
                        # 2 = Chord and Brace
                        # 3 = Single Beam
                        # 4 = Steel Plate
                        # 5 = Closure Pour
                        ret   =   MyBridgeModeler.Diaphragm.GetNameList(NumberNames,   MyName,
MyType)
                        NDiaphragms = ret[1]
                        DiaphragmNames = list(ret[2])
                        ValidDiaphgragmNames    =    ['Diafragmas    extremos',    'Diafragmas
intermediários']
                        for i in range (NDiaphragms):
                            if DiaphragmNames[i] not in ValidDiaphgragmNames:
                                ret = MyBridgeModeler.Diaphragm.Delete(DiaphragmNames[i])

                        Name = 'Diafragmas extremos'
```

```
                        MyType = 3
                        # The diaphragm type.
                        # 1 = Solid
                        # 2 = Chord and Brace
                        # 3 = Single Beam
                        # 4 = Steel Plate
                        # 5 = Closure Pour
                        ret = MyBridgeModeler.Diaphragm.AddNew(Name, MyType)

                        Name = 'Diafragmas intermediários'
                        MyType = 3
                        # The diaphragm type.
                        # 1 = Solid
                        # 2 = Chord and Brace
                        # 3 = Single Beam
                        # 4 = Steel Plate
                        # 5 = Closure Pour
                        ret = MyBridgeModeler.Diaphragm.AddNew(Name, MyType)

                        Name = 'Diafragmas extremos' #The name of an existing single beam-type
bridge diaphragm.
                        BeamSection = 'Diafragma extremo' #The steel frame section property
used for the beam.
                        ElevationChangeTop = 15 #The elevation change from top of beam to top
of adjacent girder. [L]
                        ModelMixed = bool(False) #Indicates if the beam is modeled as mixed
frame and shell
                        ConnectionPlateWidth = lea #The width of the connection plate. [L]
                        ConnectionPlateThickness = eea #The thickness of the connection plate.
[L]
                        ConnectionPlateMaterial = 'A588Gr50' #The steel material for the
connection plate.
                        ConnectionPlateOnBothSidesOfWeb = bool(True) #Indicates if the
connection plate is provided on both sides of the web.
                        ret = MyBridgeModeler.Diaphragm.SetSingleBeam(Name, BeamSection,
ElevationChangeTop, ModelMixed, ConnectionPlateWidth, ConnectionPlateThickness,
ConnectionPlateMaterial, ConnectionPlateOnBothSidesOfWeb)

                        Name = 'Diafragmas intermediários' #The name of an existing single
beam-type bridge diaphragm.
                        BeamSection = 'Diafragma intermediário' #The steel frame section
property used for the beam.
                        ElevationChangeTop = 15 #The elevation change from top of beam to top
of adjacent girder. [L]
                        ModelMixed = bool(False) #Indicates if the beam is modeled as mixed
frame and shell
                        ConnectionPlateWidth = let #The width of the connection plate. [L]
                        ConnectionPlateThickness = eet #The thickness of the connection plate.
[L]
                        ConnectionPlateMaterial = 'A588Gr50' #The steel material for the
connection plate.
                        ConnectionPlateOnBothSidesOfWeb = bool(True) #Indicates if the
connection plate is provided on both sides of the web.
                        ret = MyBridgeModeler.Diaphragm.SetSingleBeam(Name, BeamSection,
ElevationChangeTop, ModelMixed, ConnectionPlateWidth, ConnectionPlateThickness,
ConnectionPlateMaterial, ConnectionPlateOnBothSidesOfWeb)

                        for i in range(len(SpanName)):
                            for j in range(len(DiaphragmDistanceSpan[i])):
                                Name = myBridgeObjName #The name of a bridge object.
                                Location = int(0)
                                # This is one of the following indicating the diaphragm location
across the bridge section.
                                # 0 = All Spaces
                                # 1 = Girder n to Girder n + 1
                                # 2 = All Girder Interiors
                                # 3 = Girder n Interior
                                GirderNumber = int() #The girder number identified as n when
the Location item is 1 or 3 (Girder n to Girder n + 1 or Girder n Interior). This item does not
apply when the Location item is 0 or 2.
                                Bearing = 'Default' #This is either Default or a valid bearing
string
                                ReferenceLine = 0 #This is either 0 indicating the layout line
or nonzero indicating a girder number. This item is only under user control when the Location
item is 1 (Girder n to Girder n + 1).
```

```
                                ErrorMessage = '' #If the function returns an error related to
the diaphragm distance this string may provide information about the error.
                                if   i == 0 and j == 0 or i == len(SpanName)-1 and j ==
len(DiaphragmDistanceSpan[i]) - 1:
                                    DiaphragmProperty = 'Diafragmas extremos' #The name of a
defined diaphram property assigned to the diaphragm.
                                else:
                                    DiaphragmProperty = 'Diafragmas intermediários'
                                ret                                                         =
MyBridgeModeler.BridgeObject.DiaphragmAssign.AddNew(Name,                          SpanName[i],
DiaphragmDistanceSpan[i][j], DiaphragmProperty, Location, GirderNumber, Bearing, ReferenceLine,
ErrorMessage)




                            # 9 Bridge Section Data - Definition (cBMDeckSectionSteelIGirder
Interface)
                            LeftFillet = 0 #The horizontal dimension of the fillet on the left side
of the deck (identified as f1 on the Define Bridge Section Data form). [L]
                            RightFillet = 0 #The horizontal dimension of the fillet on the right
side of the deck (identified as f2 on the Define Bridge Section Data form). [L]
                            ret                                                             =
MyBridgeModeler.DeckSection.SteelIGirder.SetFilletHorizontalDimensions(myDeckSectionName_1,
LeftFillet, RightFillet)

                            SlabMaterial = f'C{f_ck} - Tabuleiro' #The name of the slab material
property. This is intended to be a concrete property
                            GirderLongitudinalLayout = 1
                            # Indicates how the girders are laid out at a horizontal curve in a
bridge object.
                            # 1 = Along layout line.
                            # 2 = Straight line.
                            ret                                                             =
MyBridgeModeler.DeckSection.SteelIGirder.SetGeneralData(myDeckSectionName_1,     SlabMaterial,
NumberInteriorGirders, TotalWidthOfDeckSection, GirderLongitudinalLayout)

                            ConstantGirderSection = True #Indicates if all of the girder sections
are the same.
                            GirderSection = ['Perfil 1']
                            ret                                                             =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderSectionProperties(myDeckSectionName_1,
ConstantGirderSection, GirderSection)

                            ConstantSpacing = True
                            Spacing = []
                            ret                                                             =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderSpacingData(myDeckSectionName_1,
ConstantSpacing, Spacing)


                            DistanceToFillet  =  t2_1/2  #The  overhang  distance  to  the  fillet
(identified as L3 on the Define Bridge Section Data form). [L] Qualquer valor inferior à metade
da largura do flange superior dará erro na definição da seção
                            OuterThickness = h_laje #The overhang outer thickness (identified as
t5 on the Define Bridge Section Data form). [L]
                            ret                                                             =
MyBridgeModeler.DeckSection.SteelIGirder.SetLeftOverhangDimensions(myDeckSectionName_1,
OverhangLength, DistanceToFillet, OuterThickness)

                            DistanceToFillet  =  t2_1/2  #The  overhang  distance  to  the  fillet
(identified as L3 on the Define Bridge Section Data form). [L]
                            OuterThickness = h_laje #The overhang outer thickness (identified as
t5 on the Define Bridge Section Data form). [L]
                            ret                                                             =
MyBridgeModeler.DeckSection.SteelIGirder.SetRightOverhangDimensions(myDeckSectionName_1,
OverhangLength, DistanceToFillet, OuterThickness)

                            TopSlabThickness = h_laje #The thickness of the top slab (t1 on Define
Bridge Section Data form). [L]
                            ConstantHaunchHeight  =  True  #Indicates  if  the  haunch  heights  are
constant for all girders.
                            HaunchHeight = [2.5]
```

```
                        # A zero-based, one-dimensional array of girder haunch heights
dimensioned to (NumberInteriorGirders + 1)
                        # The haunch heights are entered in order in this array from the left
exterior girder to the right exterior girder.
                        # Only the first height in this array is used if the
ConstantHaunchHeight item is True. [L]
                        ret                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetSlabThicknessData(myDeckSectionName_1,
TopSlabThickness, ConstantHaunchHeight, HaunchHeight)

                        ModelingType = 0
                        # The girder modeling object type.
                        # 0 = Frame
                        # 1 = Shell
                        # 2 = Mixed
                        MaxElementWidthFlange =  40 # The maximum meshed element width for the
girder flange. This item only applies when the ModelingType item is 1 (Shell). [L]
                        MaxElementHeightWeb = 40 # The maximum meshed element height for the
girder web. This item only applies when the ModelingType item is either 1 or 2 (Shell or Mixed).
[L]
                        ret                                                                 =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderModelingForAreaObjectModels(myDeckSectionNam
e_1, ModelingType, MaxElementWidthFlange, MaxElementHeightWeb)

                        X = 0 #The offset in the X direction from the reference point to the
insertion point. [L]
                        Y = 0 #The offset in the Y direction from the reference point to the
insertion point. [L]
                        ret                                                                 =
MyBridgeModeler.DeckSection.InsertionPoint.SetPoint(myDeckSectionName_1, X, Y)




                        # 10 Bridge Section Data - Loads (cBMDeckSectionLoadData Interface) -
Métodos: SetBarrierLoads, SetConcretePourFormLoads, SetConcretePourHaunchLoads,
                        #     SetConcretePourPermanentLoads,    SetConcretePourTemporaryLoads,
SetLiveLoadCurbLocations, SetSidewalkLoads, SetTemperatureLoads, SetWearingSurfaceLoads


                        # 10.1 Carregamentos Deck Section Data

                        LoadPatternName = 'Defensas' #This is either None or the name of the
load pattern to which the barrier load is applied. The intent is that the load pattern would be
of type Dead.
                        NumberBarriers = 2 #The number of barriers specified
                        MeasureFrom = [1,2]
                        # A zero-based one-dimensional array indicating the reference point the
barrier location is measured from for each barrier.
                        # 1 = Left Edge of Deck
                        # 2 = Right Edge of Deck
                        DistanceToOuterEdge = [0,0] #The distance from the reference point
specified by the MeasureFrom item to the outer edge of the barrier. [L]
                        Width = [40,40] #The barrier width. [L]
                        Height = [81,81] #The barrier height. [L]
                        WeightEccentricity = [0,0] #The eccentricity of the barrier weight. [L]
                        WeightPerLength = [5.625/100,5.625/100] #The barrier weight per unit
length. [F/L]
                        ret                                                                 =
MyBridgeModeler.DeckSection.LoadData.SetBarrierLoads(myDeckSectionName_1,     LoadPatternName,
NumberBarriers,   MeasureFrom,   DistanceToOuterEdge,   Width,   Height,   WeightEccentricity,
WeightPerLength)

                        LoadPatternName = '' #This is either None or the name of the load
pattern to which the haunch load is applied. The intent is that the load pattern would be of
type Dead.
                        UnitWeight = float() #The unit weight of the form load. [F/L2]
                        ret                                                                 =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourFormLoads(myDeckSectionName_1,
LoadPatternName, UnitWeight)

                        LoadPatternName = ''
                        ProgramDetermined = True #This flag indicates whether the haunch load
is program determined or user defined.
```

```
                        UserValue = float()#The haunch load value. This item only applies when
the ProgramDetermined item is False. [F/L]
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourHaunchLoads(myDeckSectionName_1,
LoadPatternName, ProgramDetermined, UserValue)

                        LoadPatternName = ''
                        WearingSurface = float() #The integral wearing surface load. [F/L2]
                        Utilities = float() #The utilities load. [F/L2]
                        Miscellaneous = float() #The miscellaneous load. [F/L2]
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourPermanentLoads(myDeckSectionName_1,
LoadPatternName, WearingSurface, Utilities, Miscellaneous)

                        LoadPatternName = 'ConcPour Temporary'
                        Miscellaneous = 2E-04
                        AdditionalOverhang = float() #The additional overhang load. [F/L2]
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetConcretePourTemporaryLoads(myDeckSectionName_1,
LoadPatternName, AdditionalOverhang, Utilities, Miscellaneous)

                        ProgramDetermined = True
                        DistToInsideEdgeOfLeftCurb = 40
                        DistToInsideEdgeOfRightCurb = 40
                        DistanceToCenterlineOfMedianCurb = float()
                        WidthOfMedianCurb = float()
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetLiveLoadCurbLocations(myDeckSectionName_1,
ProgramDetermined,          DistToInsideEdgeOfLeftCurb,          DistToInsideEdgeOfRightCurb,
DistanceToCenterlineOfMedianCurb, WidthOfMedianCurb)

                        LoadPatternName = '' #This is either None or the name of the load
pattern to which the sidewalk load is applied. The intent is that the load pattern would be of
type Dead.
                        NumberSidewalks = 0
                        MeasureFrom = []
                        DistanceToOuterEdge =[]
                        Width = []
                        WeightPerArea = []
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetSidewalkLoads(myDeckSectionName_1,    LoadPatternName,
NumberSidewalks, MeasureFrom, DistanceToOuterEdge, Width, WeightPerArea)

                        LoadPatternName = ''
                        WeightPerArea = float()
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetWearingSurfaceLoads(myDeckSectionName_1,
LoadPatternName, WeightPerArea)

                        LoadPatternPositiveName = 'Temperatura uniforme pos'#This is either
None or the name of the load pattern to which the positive temperature load is applied. The
intent is that the load pattern would be of type Temperature.
                        TemperatureChangePositive = 15 #The positive temperature change. [T]
                        LoadPatternNegativeName = 'Temperatura uniforme neg'#This is either
None or the name of the load pattern to which the negative temperature load is applied. The
intent is that the load pattern would be of type Temperature.
                        TemperatureChangeNegative = -15 #The negative temperature change that
should be entered as zero or a negative number. [T]
                        ret                                                                    =
MyBridgeModeler.DeckSection.LoadData.SetTemperatureLoads(myDeckSectionName_1,
LoadPatternPositiveName,          TemperatureChangePositive,          LoadPatternNegativeName,
TemperatureChangeNegative)

                        ret     =     MyBridgeModeler.DeckSection.AddCopy('SeçãoMista-Perfil2',
myDeckSectionName_1) # Outro Deck Section -> referente ao perfil 2
                        ConstantGirderSection = True #Indicates if all of the girder sections
are the same.
                        GirderSection = ['Perfil 2']
                        ret                                                                    =
MyBridgeModeler.DeckSection.SteelIGirder.SetGirderSectionProperties('SeçãoMista-Perfil2',
ConstantGirderSection, GirderSection)
                        myDeckSectionName_2 = 'SeçãoMista-Perfil2'

                        for i in range(NVãos):
                            if NTrechos <= 2:
```

```
                                ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName, [f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'], [myDeckSectionName_1])
                            if NTrechos == 3:
                                ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName, [f'Perfil 2.{i+1}.1', f'Perfil 2.{i+1}.2'], [myDeckSectionName_2, myDeckSectionName_2])
                                ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName, [f'Perfil 1.{i+1}'], [myDeckSectionName_1])
                            else:
                                ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName, [f'Perfil 1.{i+1}.1', f'Perfil 1.{i+1}.2'], [myDeckSectionName_1])
                                ret = MyBridgeModeler.BridgeObject.SetDeckSectionAssigns(myBridgeObjName, [f'Perfil 2.{i+1}.1', f'Perfil 2.{i+1}.2'], [myDeckSectionName_2])

                        # 10.2 Limpeza das inutilizadas
                        ret = SapModel.LoadPatterns.Delete('Temperature Pos')
                        ret = SapModel.LoadPatterns.Delete('Temperature Neg')
                        ret = SapModel.LoadPatterns.Delete('PreTension')
                        ret = SapModel.LoadPatterns.Delete('Barrier')




                        # 11 Atribuição dos Encontros (cBMAbutmentAssign Methods)

                        Elevation_StartAbutment = -h_laje -t3_1 - h_ap #The global Z elevation of the substructure. (start abutment) -> Distância entre o topo da laje e o CG do aparelho de apoio
                        ElevationAtLayoutLine_StartAbutment = -h_laje -t3_1 - h_ap/2 #The bearing elevation (global Z) at the layout line. [L] -> Distância entre o topo da laje e a base do aparelho de apoio
                        Elevation_EndAbutment = Elevation_StartAbutment #The global Z elevation of the substructure.
                        ElevationAtLayoutLine_EndAbutement = ElevationAtLayoutLine_StartAbutment #The bearing elevation (global Z) at the layout line. [L]

                        # 11.1 Start Abutment
                        Name = myBridgeObjName #The name of a bridge object.
                        AbutmentLocation = 1
                        # This is one of the following indicating the abutment location:
                        # 1 = Start abutment.
                        # 2 = End abutment.
                        SupportName = 'Start Abutment' #The support name. The support name cannot be blank and it can not be the same as another abutment or bent support name
                        AbutmentDirection = 'Default' #The bearing of a line parallel to the abutment. It is either Default or a valid bearing string.
                        DiaphragmProperty = 'None' #This is either None (meaning no diaphragm exists at the abutment) or the name of a defined bridge diaphragm property.
                        InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-girder or Super-T girder deck sections.
                        #It does not apply to other deck sections. It is either None (meaning no diaphragm interior diaphragm exists) or the name of a defined bridge diaphragm property.
                        ClosurePourDiaphragm = 'None' #This is either None or the name of a defined bridge diaphragm property of type Closure Pour.
                        #This only applies if the substructure type is Bent Property and the specified bent property is of type Double Bearing Line with Closure Pour
                        DiaphragmOffsetLocation = 1
                        #This is one of the following indicating the diaphragm offset location. This item only applies when the type of diaphragm defined by the DiaphragmProperty item is Solid.
                        # 0 = Flush
                        # 1 = Centerline
                        ret = MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name, AbutmentLocation, SupportName, AbutmentDirection, DiaphragmProperty, InteriorDiaphragm, ClosurePourDiaphragm, DiaphragmOffsetLocation)

                        if Tipo_analise == 'SUPERESTRUTURA':
                            SubstructureType = 1
```

```
                            AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the
abutment or bent property used for the substructure. [L]
                        else:
                            SubstructureType = 2
                            AbutmentOrBentProperty = 'Apoios extremos'
                        HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the abutment. [L]
                        ret                                                                    =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,   AbutmentLocation,
SubstructureType, AbutmentOrBentProperty, Elevation_StartAbutment, HorizontalOffset)

                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the substructure.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the substructure.
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int() #The number of bearings for the bridge section
or for abutments at Steel U Girder deck sections the number of bearings at the U girder.
                        #For bearings at steel U girder sections this item must be 1 or 2. This
item only applies when the bearing assignment type is General or when the abutment is at a Steel
U Girder deck section.
                        UniformSpacing = float() #The uniform spacing of bearings. This item
only applies when the bearing assignment type is General. [L]
                        BearingOffset = float() #The offset from the section reference point
to the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                                    =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,       AbutmentLocation,
BearingAssignmentType,                  BearingProperty,                 RestrainerProperty,
ElevationAtLayoutLine_StartAbutment,    RotationAngle,    NumberOfBearings,    UniformSpacing,
BearingOffset)

                        # 11.2 End Abutment
                        Name = myBridgeObjName #The name of a bridge object.
                        AbutmentLocation = 2
                        # This is one of the following indicating the abutment location:
                        # 1 = Start abutment.
                        # 2 = End abutment.
                        SupportName = 'End Abutment'
                        AbutmentDirection = 'Default' #The bearing of a line parallel to the
abutment. It is either Default or a valid bearing string.
                        DiaphragmProperty = 'None' #This is either None (meaning no diaphragm
exists at the abutment) or the name of a defined bridge diaphragm property.
                        InteriorDiaphragm = '' #This is the interior diaphragm for Steel U-
girder or Super-T girder deck sections.
                        #It does not apply to other deck sections. It is either None (meaning
no diaphragm interior diaphragm exists) or the name of a defined bridge diaphragm property.
                        ClosurePourDiaphragm = 'None' #This is either None or the name of a
defined bridge diaphragm property of type Closure Pour.
                        #This only applies if the substructure type is Bent Property and the
specified bent property is of type Double Bearing Line with Closure Pour
                        DiaphragmOffsetLocation = 1
                        #This is one of the following indicating the diaphragm offset location.
This item only applies when the type of diaphragm defined by the DiaphragmProperty item is
Solid.
                        # 0 = Flush
                        # 1 = Centerline
                        ret                                                                    =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSuperstructureAssignment(Name,
AbutmentLocation,   SupportName,   AbutmentDirection,   DiaphragmProperty,   InteriorDiaphragm,
ClosurePourDiaphragm, DiaphragmOffsetLocation)

                        if Tipo_analise == 'SUPERESTRUTURA':
                            SubstructureType = 1
                            AbutmentOrBentProperty = 'Apoio aparelho de apoio' #The name of the
abutment or bent property used for the substructure. [L]
                        else:
                            SubstructureType = 2
                            AbutmentOrBentProperty = 'Apoios extremos'
```

```
                        HorizontalOffset = 0 #The horizontal offset from the layout line to the
midlength of the abutment. [L]
                        ret                                                                    =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetSubstructureAssignment(Name,   AbutmentLocation,
SubstructureType, AbutmentOrBentProperty, Elevation_StartAbutment, HorizontalOffset)

                        BearingAssignmentType = 1
                        # This is one of the following indicating the bearing assignment type:
                        # 1 = Girder-by-girder
                        # 2 = General
                        # The General assignment type only applies to abutments at concrete box
girders deck sections or concrete flat slab deck sections. General assignments to abutments at
other deck section types will return an error.
                        BearingProperty = 'Ap. de apoio' #The name of a defined bearing property
assigned to the substructure.
                        RestrainerProperty = 'None' #This is either None or the name of a
defined restrainer property assigned to the substructure.
                        RotationAngle = 0 #The bearing rotation angle from the bridge default.
                        NumberOfBearings = int() #The number of bearings for the bridge section
or for abutments at Steel U Girder deck sections the number of bearings at the U girder.
                        #For bearings at steel U girder sections this item must be 1 or 2. This
item only applies when the bearing assignment type is General or when the abutment is at a Steel
U Girder deck section.
                        UniformSpacing = float() #The uniform spacing of bearings. This item
only applies when the bearing assignment type is General. [L]
                        BearingOffset = float() #The offset from the section reference point
to the bearing center. This item only applies when the bearing assignment type is General. [L]
                        ret                                                                    =
MyBridgeModeler.BridgeObject.AbutmentAssign.SetBearingAssignment(Name,        AbutmentLocation,
BearingAssignmentType, BearingProperty, RestrainerProperty, ElevationAtLayoutLine_EndAbutement,
RotationAngle, NumberOfBearings, UniformSpacing, BearingOffset)

                        ret = MyBridgeModeler.Abutment.Delete('BABT1')




                        # 12 Carregamentos atuantes
                        # 12.1 Bridge Object - Slab Wet Concrete Load Assigns

                        Name = myBridgeObjName # The name of a bridge object.
                        ConcretePourName = '' # The concrete pour name to be deleted. Enter a
blank string to delete all slab wet concrete load assignments.
                        ret                                                                    =
MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.Delete(Name, ConcretePourName)

                        Name = myBridgeObjName
                        ConcretePourName  =  'CPSlab  Permanent  Construction'  #  The  unique
concrete pour name for the load assignment.
                        TopSlabGroup = 'SLAB' # The top slab type bridge group specified for
the load assignment.
                        SlabWeightFactor = float(1) # The slab weight load factor for the load
assignment.
                        HaunchLoadFactor = float(0)
                        # The haunch load factor for the load assignment. This item only applies
if the bridge section type is one of the following:
                        # Steel I Girder
                        # Steel U Girder
                        # Precast Concrete I Girder
                        # Precast Concrete U Girder
                        SIPFormLoadFactor = float(0) # The SIP Form load factor for the load
assignment. This item only applies if the bridge section type is Steel I Girder.
                        BridgeSectPermanentLoadFactor = float(1) # The bridge section permanent
load factor for the load assignment.
                        BridgeSectTemporaryLoadFactor = float(0) # The bridge section temporary
load factor for the load assignment.
                        AdditionalLoadPermanent = float(0) # The permanent additional load for
the load assignment. [F/L2]
                        AdditionalLoadTemporary = float(0) # The temporary additional load for
the load assignment. [F/L2]
                        ret                                                                    =
MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.AddNew(Name,          ConcretePourName,
TopSlabGroup, SlabWeightFactor, HaunchLoadFactor,
```

```
                        SIPFormLoadFactor,                    BridgeSectPermanentLoadFactor,
BridgeSectTemporaryLoadFactor, AdditionalLoadPermanent, AdditionalLoadTemporary)

                        Name = myBridgeObjName
                        ConcretePourName = 'CPSlab All Construction' # The unique concrete pour
name for the load assignment.
                        TopSlabGroup = 'SLAB' # The top slab type bridge group specified for
the load assignment.
                        SlabWeightFactor = float(0) # The slab weight load factor for the load
assignment.
                        HaunchLoadFactor = float(0)
                        # The haunch load factor for the load assignment. This item only applies
if the bridge section type is one of the following:
                        # Steel I Girder
                        # Steel U Girder
                        # Precast Concrete I Girder
                        # Precast Concrete U Girder
                        SIPFormLoadFactor = float(0) # The SIP Form load factor for the load
assignment. This item only applies if the bridge section type is Steel I Girder.
                        BridgeSectPermanentLoadFactor = float(0) # The bridge section permanent
load factor for the load assignment.
                        BridgeSectTemporaryLoadFactor = float(1) # The bridge section temporary
load factor for the load assignment.
                        AdditionalLoadPermanent = float(0) # The permanent additional load for
the load assignment. [F/L2]
                        AdditionalLoadTemporary = float(0) # The temporary additional load for
the load assignment. [F/L2]
                        ret                                                                 =
MyBridgeModeler.BridgeObject.LoadAssignSlabWetConcrete.AddNew(Name,          ConcretePourName,
TopSlabGroup, SlabWeightFactor, HaunchLoadFactor,
                        SIPFormLoadFactor,                    BridgeSectPermanentLoadFactor,
BridgeSectTemporaryLoadFactor, AdditionalLoadPermanent, AdditionalLoadTemporary)

                        # 12.2 Bridge Object - Temperature Load Assigns
                        LoadPat = ['Temperatura gradiente pos', 'Temperatura gradiente neg'] #
A zero-based one-dimensional array of the load pattern used for each load assignment.
                        TempGradientName = ['Temperatura gradiente', 'Temperatura gradiente']
# A zero-based one-dimensional array of the name of the defined temperature gradient for each
load assignment. This item only applies when the TemperatureOption item is 1 (Temperature
gradient).
                        GradientType = [0, 1]
                        # A zero-based one-dimensional array of the gradient type for each load
assignment. This item only applies when the TemperatureOption item is 1 (Temperature gradient).
                        # 0 = Positive temperature change
                        # 1 = Negative temperature change
                        for i in range(NTrechos):
                            for j in range(2):
                                Name = myBridgeObjName
                                TemperatureOption = 1
                                # A zero-based one-dimensional array of the temperature option
for each load assignment.
                                # 0 = Uniform temperature change
                                # 1 = Temperature gradient
                                UniformTempChange = float() # A zero-based one-dimensional
array of the uniform temperature change for each load assignment. This item only applies when
the TemperatureOption item is 0 (Uniform temperature change). [T]
                                ret                                                         =
MyBridgeModeler.BridgeObject.LoadAssignTemperature.SetOneLoad(Name,  SpanName[i],  LoadPat[j],
TemperatureOption, UniformTempChange, TempGradientName[j], GradientType[j])

                        # 12.3 Bridge Object - Line Load Assigns

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Vento' # The load pattern for the load assignment.
                        LoadDistribution = 'Vento força' #The named load distribution for the
load assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        TransverseVariation = 'None' # The variation used for the load
assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignLine.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, TransverseVariation)

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Vento' # The load pattern for the load assignment.
```

```
                        LoadDistribution = 'Vento momento' #The named load distribution for the
load assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        TransverseVariation = 'None' # The variation used for the load
assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignLine.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, TransverseVariation)

                        # 12.4 Bridge Object - Area Load Assigns

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Frenagem e aceleracao' # The load pattern for the load
assignment.
                        LoadDistribution = 'Frenagem e aceleracao' #The named load distribution
for the load assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        LeftEdgeVariation = 'None' # The left edge variation used for the load
assignment. This is either None or the name of a defined variation.
                        RightEdgeVariation = 'None' # The right edge variation used for the
load assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Pavimentação' # The load pattern for the load assignment.
                        LoadDistribution = 'Pavimentação' #The named load distribution for the
load assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        LeftEdgeVariation = 'None' # The left edge variation used for the load
assignment. This is either None or the name of a defined variation.
                        RightEdgeVariation = 'None' # The right edge variation used for the
load assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)

                        Name = myBridgeObjName # The name of a bridge object.
                        LoadPat = 'Recapeamento' # The load pattern for the load assignment.
                        LoadDistribution = 'Recapeamento' #The named load distribution for the
load assignment.
                        StartStation = 0 # The start station for the load assignment. [L]
                        EndStation = CTotal # The end station for the load assignment. [L]
                        LeftEdgeVariation = 'None' # The left edge variation used for the load
assignment. This is either None or the name of a defined variation.
                        RightEdgeVariation = 'None' # The right edge variation used for the
load assignment. This is either None or the name of a defined variation.
                        ret     =     MyBridgeModeler.BridgeObject.LoadAssignArea.AddNew(Name,
LoadPat, LoadDistribution, StartStation, EndStation, LeftEdgeVariation, RightEdgeVariation)




                        # 13 Staged Construction Load Cases (Retração e fluência será atribuído
após a criação do encontro)
                        # Apenas os load cases Veiculo, Multidao, Retração, Fluência e
Cronograma são necessários de definição (os load patterns já viram load cases automaticamente
pela definição dada)

                        # 13.1 Update no modelo para ser possível selecionar os elementos da
ponte
                        Name = myBridgeObjName
                        Action = 4 #Atualização do modelo para ser possível de selecionar os
elementos do objeto ponte
                        # An integer value between 1 and 4, inclusive, indicating the update
action to be taken.
                        # 1 = Update Linked Model
                        # 2 = Clear All From Linked Model
                        # 3 = Convert To Unlinked Model
                        # 4 = Clear and Create Linked Model
                        ModelType = 2
                        #An integer value between 1 and 3, inclusive, indicating the model type
to be created. This item applies when the Action item is 1 or 4.
```

```
                        # 1 = Update as Spine Model Using Frame Objects
                        # 2 = Update as Area Object Modell
                        # 3 = Update as Solid Object Model
                        SubmeshSize = 50 #The preferred maximum submesh size. This item applies
when the Action item is 1 or 4 and the ModelType item is 2 or 3. [L]
                        ApplyAreaPropModifier = False #This item is to indicate whether or not
to apply property modifiers to area objects representing the bridge section to adjust for
overlap.
                        # This option only applies to bridge sections of type concrete box-
girder, concrete T-beam, flat slab, and user-defined (Section Designer).
                        # When the bridge object is updated as an area-object model (ModelType
is "Area"), entering True will apply property modifiers to the area objects representing the
bridge cross section that will adjust their weight, mass, and stiffness factors f11 and m11
                        # to match a corresponding solid-object model (ModelType is "Solid"),
which has no overlaps. This item applies when the Action item is 1 or 4 and the ModelType item
is 2. [L]
                        # True = Apply area property modifiers
                        # False = Do not apply area property modifiers
                        MaxDeckSegmentLength = 50 #The maximum segment length for deck spans.
This item applies when the Action item is 1 or 4. [L]
                        MaxCapSegmentLength = 50 #The maximum segment length for bent cap beams.
This item applies when the Action item is 1 or 4. [L]
                        MaxColumnSegmentLength = 50 #The maximum segment length for bent
columns. This item applies when the Action item is 1 or 4. [L]
                        MeshSlabAtCriticalSteelIGirderLocations = False #A flag indicating that
the slab should be meshed at critical steel I-girder locations. This item applies when the
Action item is 1 or 4 and the bridge deck section is a Steel I-Girder section.
                        ret = MyBridgeModeler.BridgeObject.Update(Name, Action, ModelType,
SubmeshSize, ApplyAreaPropModifier, MaxDeckSegmentLength, MaxCapSegmentLength,
MaxColumnSegmentLength, MeshSlabAtCriticalSteelIGirderLocations)

                        # 13.2 Retração e Fluência

                        ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('Retração e
Fluência')

                        InitialCase = ''
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('Retração e Fluência', InitialCase)

                        NLGeomType = 1
                        # This is 0, 1 or 2, indicating the geometric nonlinearity option
selected for the load case.
                        # 0 = None
                        # 1 = P-delta
                        # 2 = P-delta plus large displacements
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('Retração e Fluência',
NLGeomType)

                        TimeDepMatProp = True
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('Retração e Fluência',
TimeDepMatProp)

                        StagedSaveOption = 2
                        # This is 0, 1, 2 or 3, indicating the results saved option for the
load case.
                        # 0 = End of final stage
                        # 1 = End of each stage
                        # 2 = Start and end of each stage
                        # 3 = Two or more times in each stage
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('Retração e Fluência',
StagedSaveOption)

                        NumberStages = 5 #The number of stages defined for the specified load
case.
                        Duration = [0, 28, 3000, 3000, 4000]#This is an array that includes the
duration in days for each stage.
                        Output = [True, True, True, True, True]#This is an array that includes
True or False, indicating if analysis output is to be saved for each stage.
                        OutputName = ['Retração-0', 'Retração-28', 'Retração-3000', 'Retração-
6000', 'Retração-10000']#This is an array that includes a user-specified output name for each
stage.
```

```
                        Comment = ['GL2000', 'GL2000', 'GL2000', 'GL2000', 'GL2000']#This is
an array that includes a comment for each stage. The comment may be a blank string.

                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('Retração      e      Fluência',
NumberStages, Duration, Output, OutputName, Comment)

                        # 13.3 GirdersAndDiaphragms (peso da estrutura metálica)
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetCase('GirdersAndDiaphragms')

                        InitialCase = ''
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('GirdersAndDiaphragms', InitialCase)

                        NLGeomType = 1
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('GirdersAndDiaphragms',
NLGeomType)

                        TimeDepMatProp = True
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('GirdersAndDiaphragms',
TimeDepMatProp)

                        StagedSaveOption = 2
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('GirdersAndDiaphragms',
StagedSaveOption)

                        NumberStages = 1
                        Duration = [0]
                        Output = [True]
                        OutputName = ['Diafragmas']
                        Comment = ['PP']
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('GirdersAndDiaphragms',
NumberStages, Duration, Output, OutputName, Comment)

                        # 13.4 DeckPlus
                        ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('DeckPlus')

                        InitialCase = ''
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('DeckPlus', InitialCase)

                        NLGeomType = 1
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('DeckPlus', NLGeomType)

                        TimeDepMatProp = True
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('DeckPlus', TimeDepMatProp)

                        StagedSaveOption = 2
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('DeckPlus', StagedSaveOption)

                        NumberStages = 1
                        Duration = [0]
                        Output = [True]
                        OutputName = ['ConcretePour']
                        Comment = ['Deck+']
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('DeckPlus',       NumberStages,
Duration, Output, OutputName, Comment)

                        # 13.5 DeckCured - No Composed
                        ret = SapModel.LoadCases.StaticNonlinearStaged.SetCase('DeckCured - No
Composed')

                        InitialCase = 'DeckPlus'
                        ret                                                                 =
SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('DeckCured - No Composed', InitialCase)
```

```
                        NLGeomType = 1
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('DeckCured  -  No  Composed',
NLGeomType)

                        TimeDepMatProp = True
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('DeckCured  -  No  Composed',
TimeDepMatProp)

                        StagedSaveOption = 2
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('DeckCured     -     No     Composed',
StagedSaveOption)

                        NumberStages = 1
                        Duration = [0]
                        Output = [True]
                        OutputName = ['ConcretePour Cured']
                        Comment = ['Remove Forms']
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('DeckCured   -   No   Composed',
NumberStages, Duration, Output, OutputName, Comment)

                        # 13.6 ConcPour Temporary
                        ret    =    SapModel.LoadCases.StaticNonlinearStaged.SetCase('ConcPour
Temporary')

                        InitialCase = ''
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetInitialCase('ConcPour Temporary', InitialCase)

                        NLGeomType = 1
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetGeometricNonlinearity('ConcPour        Temporary',
NLGeomType)

                        TimeDepMatProp = True
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetMaterialNonlinearity('ConcPour         Temporary',
TimeDepMatProp)

                        StagedSaveOption = 2
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetResultsSaved('ConcPour                 Temporary',
StagedSaveOption)

                        NumberStages = 1
                        Duration = [0]
                        Output = [True]
                        OutputName = ['Construção']
                        Comment = ['Sobrecarga']
                        ret                                                                    =
SapModel.LoadCases.StaticNonlinearStaged.SetStageDefinitions_2('ConcPour           Temporary',
NumberStages, Duration, Output, OutputName, Comment)

                        # 13.7 Set Stage Data (métodos não estavam funcionando)

                        ret = SapModel.DatabaseTables.SetTableOutputOptionsForDisplay(0, 0, 0,
False, 1, 1, False, 1, 6, 1, 1, 2, 3, 1, 1, 2, 2, 2)

                        TableKey = 'Case - Static 6 - Nonlinear Stage Data'
                        GroupName = ''
                        TableVersion = int()
                        FieldKeysIncluded = []
                        NumberRecords = int()
                        TableData = []
                        ret    =    SapModel.DatabaseTables.GetTableForEditingArray(TableKey,
GroupName, TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ColumnLabels = list(ret[2])
                        Resultados = list(ret[4])

                        TableKey = 'Case - Static 6 - Nonlinear Stage Data'
                        TableVersion = int()
                        FieldKeysIncluded = ColumnLabels
```

```
                        NumberRecords = int()
                        TableData = [
                        'ConcPour    Temporary',    '1',    'Add    Structure',    'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '3', '', '', '', '', '', '',
                        'ConcPour   Temporary',   '1',   'Add   Guide   Structure',   'Group',
f'<{myBridgeObjName}>SLAB', '', '', '', '', '', '', '',
                        'ConcPour   Temporary',   '1',   'Pour   Concrete',   'Concrete   Pour',
f'<{myBridgeObjName}>CPSlab All Construction', '', '', '', '', '', '', '',
                        'GirdersAndDiaphragms',     '1',     'Add     Structure',     'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '0', '', '', '', '', '', '',
                        'GirdersAndDiaphragms',   '1',   'Load   Objects   If   Added',   'Group',
f'<{myBridgeObjName}>GIRDER_DIAPH', '', 'Load pattern', 'DEAD', '1', '', '', '',
                        'Retração    e    Fluência',    '1',    'Add    Structure',    'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '28', '', '', '', '', '', '',
                        'Retração    e    Fluência',    '1',    'Add    Structure',    'Group',
f'<{myBridgeObjName}>SLAB', '36', '', '', '', '', '', '',
                        'DeckCured  -  No  Composed',  '1',  'Remove  Forms',  'Concrete  Pour',
f'<{myBridgeObjName}>CPSlab Permanent Construction', '0', '', '', '', '', '', '',
                        'DeckPlus',         '1',         'Add         Structure',         'Group',
f'<{myBridgeObjName}>ALL_BUT_SLAB', '3', '', '', '', '', '', '',
                        'DeckPlus',      '1',      'Add      Guide      Structure',      'Group',
f'<{myBridgeObjName}>SLAB', '', '', '', '', '', '', '',
                        'DeckPlus',      '1',      'Pour      Concrete',      'Concrete      Pour',
f'<{myBridgeObjName}>CPSlab Permanent Construction', '', '', '', '', '', '', ''
                        ]
                        ret     =     SapModel.DatabaseTables.SetTableForEditingArray(TableKey,
TableVersion, FieldKeysIncluded, NumberRecords, TableData)
                        ret  =  SapModel.DatabaseTables.ApplyEditedTables(False,  int(),  int(),
int(), int(), '')

                        # Update no modelo
                        Name = myBridgeObjName
                        Action = 1 #Atualização do modelo para ser possível de selecionar os
elementos do objeto ponte
                        # An integer value between 1 and 4, inclusive, indicating the update
action to be taken.
                        # 1 = Update Linked Model
                        # 2 = Clear All From Linked Model
                        # 3 = Convert To Unlinked Model
                        # 4 = Clear and Create Linked Model
                        ret    =    MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name,
Action)




                        # 14 Definição dos grupos comuns para pegar resultados da análise e
encontros

                        # Grupos comuns
                        ret = SapModel.SelectObj.ClearSelection()
                        Elementos   =   ['Diafragmas',   'Diafragmas',   'Pilares',   'Travessas',
'Travessas']
                        PropSecoes = ['Diafragma extremo', 'Diafragma intermediário', f'Pilar
(C{f_ck})', f'Travessa encontro (C{f_ck})', f'Travessa intermediária (C{f_ck})']
                        for i in range(len(Elementos)):
                            ret = SapModel.SelectObj.ClearSelection()
                            ret = SapModel.GroupDef.SetGroup(Elementos[i])
                            ret = SapModel.SelectObj.PropertyFrame(PropSecoes[i])
                            ret  =  SapModel.FrameObj.SetGroupAssign('',  Elementos[i],  False,
eItemType(2))




                        # # 15 Schedule
                        # if Tipo_analise == 'ENCONTROS':
                        #     ret = SapModel.Scheduler.Add('Cronograma')

                        #     Name = 'Cronograma'
                        #     NumberTasks = 17
                        #    Expanded = [True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True]
```

```
                        #     IndentLevel = [0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1]
                        #    TaskName = ['Infraestrutura', 'Pilares', 'Travessas', 'Encontros',
'Aparelho de apoio', 'Superestrutura', 'Estrutura metálica', 'Pré-lajes', 'Concretagem',
'Concretagem', 'Cura do concreto', 'Lajes de transição', 'Barreiras', 'Pavimentação',
'Recapeamento', 'Pós contrução', 'Carga móvel']
                        #     Duration = ['390', '180', '120', '90', '30', '210', '30', '15',
'1', '30', '30', '30', '14', '30', '30', '3600', '3600']
                        #     CompletionDate = [390, 180, 300, 390, 420, 600, 420, 435, 436,
466, 496, 526, 540, 570, 600, 4200, 4200]
                        #     Dependencies = ['', '', '2', '3', '4', '', '1', '7', '8', '9',
'10', '11', '12', '13', '14', '', '15']
                        #       Operation = [None, 'Add Structure', 'Add Structure', 'Add
Structure', 'Add Structure', None, 'Add Structure', 'Load Objects', 'Add Guide Structure', 'Pour
Concrete', 'Remove Forms', 'Add Structure', 'Load Objects If Added', 'Load Objects If Added',
'Load Objects', None, 'Load Objects']
                        #     IncludeSW = [False, True, True, True, False, False, True, False,
False, False, False, True, False, False, False, False, False]
                        #      ObjectType = [None, 'Group', 'Group', 'Group', 'Group', None,
'Group', 'Group', 'Group', 'Concrete Pour', 'Concrete Pour', 'Group', 'Group', 'Group', 'Group',
None, 'Group']
                        #     ObjectName = [None, 'Pilares', 'Travessas', 'Encontro', 'Links',
None,          f'<{myBridgeObjName}>ALL_BUT_SLAB',          f'<{myBridgeObjName}>GIRDER_DIAPH',
f'<{myBridgeObjName}>SLAB', f'Concrete Pour <{myBridgeObjName}>CPSlab Permanent Construction',
f'Concrete Pour <{myBridgeObjName}>CPSlab Permanent Construction', 'Lajes de transição', 'ALL',
'ALL', f'<{myBridgeObjName}>ALL', None, f'<{myBridgeObjName}>ALL']
                        #     Details = [None, 'Age=3; SW=DEAD; SF=1', 'Age=3; SW=DEAD; SF=1',
'Age=3; SW=DEAD; SF=1', 'Age=3; Without SW', None, 'Age=1; SW=DEAD; SF=1', "Load pattern
'SIPForm' SF=1", 'Guide', 'Wet Slabs', 'Forms', 'Age=3; SW=DEAD; SF=1', "Load pattern 'Defensas'
SF=1", "Load pattern 'Pavimentação' SF=1", "Load pattern 'Recapeamento' SF=1", None, "Load
pattern 'Veiculo' SF=1"]
                        #     Age = [0.0, 3.0, 3.0, 3.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
3.0, 0.0, 0.0, 0.0, 0.0, 0.0]
                        #     LoadType = [None, 'Load pattern', 'Load pattern', 'Load pattern',
'', None, 'Load pattern', 'Load pattern', '', '', '', 'Load pattern', 'Load pattern', 'Load
pattern', 'Load pattern', None, 'Load pattern']
                        #      LoadName = [None, 'DEAD', 'DEAD', 'DEAD', '', None, 'DEAD',
'SIPForm', '', '', '', 'DEAD', 'Defensas', 'Pavimentação', 'Recapeamento', None, 'Veiculo']
                        #     LoadSF = [0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0, 0.0, 1.0]
                        #     Solve = [False, True, True, True, True, False, True, True, False,
True, True, True, True, True, True, False, True]
                        #     SaveStiffness = [False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False]
                        #     OutputID = [None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
                        #     ItemType = [None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None]
                        #      ItemName = [None, 'DEAD', 'DEAD', 'DEAD', '', None, 'DEAD',
'SIPForm', '', '', '', 'DEAD', 'Defensas', 'Pavimentação', 'Recapeamento', None, 'Veiculo']
                        #     Notes = ['', 'Add Group Pilares; age at add=3; Include SW Load
Pattern=DEAD; SF=1', 'Add Group Travessas; age at add=3; Include SW Load Pattern=DEAD; SF=1',
'Add Group Encontro; age at add=3; Include SW Load Pattern=DEAD; SF=1', 'Add Group Links; age
at add=3; Without SW', '', 'Add Group <Ponte>ALL_BUT_SLAB; age at add=1; Include SW Load
Pattern=DEAD; SF=1', "Load all objects in group ' <Ponte>GIRDER_DIAPH' that are already present
or are being added at this completion day and stage.\nLoad Type=Load pattern; Load Name=SIPForm;
Scale Factor=1", 'Set the initial deflection of Group <Ponte>SLAB', 'Add the weight of a top
slab bridge group as equivalent line loads as defined in Concrete Pour <Ponte>CPSlab Permanent
Construction', 'Add the actual slab objects for a top slab bridge group with their full stiffness
and weight, age=0 day(s), while removing the equivalent line loads from the girders as defined
in Concrete Pour <Ponte>CPSlab Permanent Construction', 'Add Group Lajes de transição; age at
add=3; Include SW Load Pattern=DEAD; SF=1', "Load only objects in group' ALL' that are being
added at this completion day and stage. Objects already present will not be loaded.\nLoad
Type=Load pattern; Load Name=Defensas; Scale Factor=1", "Load only objects in group' ALL' that
are being added at this completion day and stage. Objects already present will not be
loaded.\nLoad Type=Load pattern; Load Name=Pavimentação; Scale Factor=1", "Load all objects in
group ' <Ponte>ALL' that are already present or are being added at this completion day and
stage.\nLoad Type=Load pattern; Load Name=Recapeamento; Scale Factor=1", '', "Load all objects
in group ' <Ponte>ALL' that are already present or are being added at this completion day and
stage.\nLoad Type=Load pattern; Load Name=Veiculo; Scale Factor=1"]
                        #      ret = SapModel.Scheduler.SetScheduleData_1(Name, NumberTasks,
Expanded, IndentLevel, TaskName, Duration, CompletionDate, Dependencies, Operation, IncludeSW,
ObjectType, ObjectName,
                        #     Details, Age, LoadType, LoadName, LoadSF, Solve, SaveStiffness,
OutputID, ItemType, ItemName, Notes)
```

```
                        #     Name = 'Cronograma'
                        #     Unloading = 'Unload Entire'
                        #     GeoNonLin = 'P-Delta'
                        #     MaxTotal = 1000
                        #     MaxNull = 200
                        #     MaxIterCS = 200
                        #     MaxIterNR = 200
                        #     ItConvTol = 0.001
                        #     UseEvStep = 'Yes'
                        #     EvLumpTol = 0.01
                        #     LSPerIter = 20
                        #     LSTol = 0.1
                        #     LSStepFact = 1.618
                        #     StageSave = 'Start and End of Each Stage'
                        #     StageMinIns = int()
                        #     StageMinTD = int()
                        #     TimeDepMat = 'Yes'
                        #     TFMaxIter = 100
                        #     TFTol = 0.01
                        #     TFAccelFact = 1
                        #     TFNoStop = 'No'
                        #      ret = SapModel.Scheduler.SetScheduleParameters(Name, Unloading,
GeoNonLin, MaxTotal, MaxNull, MaxIterCS, MaxIterNR, ItConvTol, UseEvStep, EvLumpTol, LSPerIter,
LSTol, LSStepFact,
                        #     StageSave, StageMinIns, StageMinTD, TimeDepMat, TFMaxIter, TFTol,
TFAccelFact, TFNoStop)




                    # 17 Atribuição de carregamentos nos elementos
                        if Tipo_analise != 'SUPERESTRUTURA':
                            # 17.1 Pressão da água nos pilares
                            ret = SapModel.SelectObj.ClearSelection()
                            ws_pil_ext = wb_pil_ext.sheets["3 Cálculo de esforços"]

                            Name = '' # The name of an existing frame object or group, depending
on the value of the ItemType item.
                            LoadPat = 'Água'
                            MyType = 1
                            # This is 1 or 2, indicating the type of distributed load.
                            # 1 = Force per unit length
                            # 2 = Moment per unit length
                            Dir = 5
                            # This is an integer between 1 and 11, indicating the direction of
the load.
                            # 1 = Local 1 axis (only applies when CSys is Local)
                            # 2 = Local 2 axis (only applies when CSys is Local)
                            # 3 = Local 3 axis (only applies when CSys is Local)
                            # 4 = X direction (does not apply when CSys is Local)
                            # 5 = Y direction (does not apply when CSys is Local)
                            # 6 = Z direction (does not apply when CSys is Local)
                            # 7 = Projected X direction (does not apply when CSys is Local)
                            # 8 = Projected Y direction (does not apply when CSys is Local)
                            # 9 = Projected Z direction (does not apply when CSys is Local)
                            # 10 = Gravity direction (only applies when CSys is Global)
                            #  11  =  Projected  Gravity  direction  (only  applies  when  CSys  is
Global)
                            Dist1 = 0
                            # This is the distance from the I-End of the frame object to the
start of the distributed load.
                            # This may be a relative distance (0 <= Dist1 <= 1) or an actual
distance, depending on the value of the RelDist item. [L] when RelDist is False
                            Dist2 = 1
                            # This is the distance from the I-End of the frame object to the
end of the distributed load.
                            # This may be a relative distance (0 <= Dist2 <= 1) or an actual
distance, depending on the value of the RelDist item. [L] when RelDist is False
                            Val1 = ws_pil_ext['D57'].value/100 # This is the load value at the
start of the distributed load. [F/L] when MyType is 1 and [FL/L] when MyType is 2
                            Val2 = Val1 # This is the load value at the end of the distributed
load. [F/L] when MyType is 1 and [FL/L] when MyType is 2
```

```
                            CSys = 'GLOBAL' # This is Local or the name of a defined coordinate
system. It is the coordinate system in which the loads are specified.
                            RelDist = True # If this item is True, the specified Dist item is
a relative distance, otherwise it is an actual distance.
                            Replace = False # If this item is True, all previous distributed
loads, if any, assigned to the specified frame object(s), in the specified load pattern, are
deleted before making the new assignment.
                            ItemType = eItemType(2)
                            # This is one of the following items in the eItemType enumeration:
                            # Object = 0
                            # Group = 1
                            # SelectedObjects = 2
                            # If this item is Object, the assignment is made to the frame object
specified by the Name item.
                            # If this item is Group, the assignment is made to all frame objects
in the group specified by the Name item.
                            # If this item is SelectedObjects, assignment is made to all
selected frame objects, and the Name item is ignored.
                            ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType,
Dir, Dist1, Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)

                            ret = SapModel.SelectObj.ClearSelection()

                            # 17.2 Empuxo de terra nos pilares

                            ret = SapModel.SelectObj.ClearSelection()

                            XMin = -1 # The maximum and minimum coordinates of the selection
box in the specified coordinate system.
                            XMax = 1
                            YMin = -TotalWidthOfDeckSection
                            YMax = TotalWidthOfDeckSection
                            ZMin = -1000
                            ZMax = -400
                            DeSelect = False # The item is False if objects are to be selected
and True if they are to be deselected.
                            CSys = 'GLOBAL' # The name of the coordinate system in which XMin,
XMax, YMin, YMax, ZMin and ZMax are specified.
                            IncludeIntersections = True
                            # When this item is True, objects that are inside the box or
intersecting the sides and edges of the box are selected or deselected.
                            # When this item is False, only objects that are fully inside the
box are selected or deselected.
                            Point = False # Point objects that fall inside the box are only
selected or deselected when this item is True.
                            Line = True
                            Area = False
                            Solid = False
                            Link = False
                            ret = SapModel.SelectObj.CoordinateRange(XMin, XMax, YMin, YMax,
ZMin, ZMax, DeSelect, CSys, IncludeIntersections, Point, Line, Area, Solid, Link)

                            Name = ''
                            LoadPat = 'Empuxo terra'
                            MyType = 1
                            Dir = 4
                            Dist1 = 0
                            Dist2 = 1
                            Val1 = ws_pil_ext['D115'].value/100
                            Val2 = Val1
                            CSys = 'GLOBAL'
                            RelDist = True
                            Replace = False
                            ItemType = eItemType(2)
                            # This is one of the following items in the eItemType enumeration:
                            # Object = 0
                            # Group = 1
                            # SelectedObjects = 2
                            # If this item is Object, the assignment is made to the frame object
specified by the Name item.
                            # If this item is Group, the assignment is made to all frame objects
in the group specified by the Name item.
                            # If this item is SelectedObjects, assignment is made to all
selected frame objects, and the Name item is ignored.
```

```
                            ret = SapModel.FrameObj.SetLoadDistributed(Name, LoadPat, MyType,
Dir, Dist1, Dist2, Val1, Val2, CSys, RelDist, Replace, ItemType)
                            ret = SapModel.SelectObj.ClearSelection()



                        # 18 Definição das Lanes - (cBMLane.SetLaneFromLayoutLine) [myLaneName
sempre será a faixa de tráfego]

                        Name = myLaneName #The name for the existing lane defined from a layout
line.
                        NumberLaneStations = 2 #The number of lane points, that is, the number
of items included in the that lane will be overwritten. LayoutLine, Station, Offset, Width and
Radius arrays.
                        LayoutLine = [myLayoutLineName, myLayoutLineName]#A zero-based array
containing the name of the bridge layout line associated with each lane station.
                        Station = [0, CTotal]#A zero-based array containing the station at each
lane station. [L]
                        Offset = [0, 0]#A zero-based array containing the offset from the layout
line to the center of the lane at each lane station. [L]
                        Width = [LarguraRolamento, LarguraRolamento]#A zero-based array
containing the the width of the lane at each lane station. [L]
                        Radius = [0,0]#A zero-based array containing the the radius of the lane
at each lane station. A positive radius curves to the right and negative to the left. [L]
                        LaneType = 0
                        #The lane type (fixed or floating).
                        # 0 = Fixed lane
                        # 1 = Floating lane set
                        if Tipo_analise == 'ENCONTROS':
                            LoadGroup = 'Todas lajes'
                        else:
                            LoadGroup = f'<{myBridgeObjName}>SLAB'
                        LeftEdgeIsExterior = False #Indicates whether the left edge of the lane
is exterior or interior.
                        RightEdgeIsExterior = False #Indicates whether the right edge of the
lane is exterior or interior.
                        MaxDiscAlongLane = 300 #The maximum lane load discretization length
along the lane. [L]
                        MaxDiscAcrossLane = 300 #The maximum lane load discretization length
across the lane. [L]
                        UseDiscFactorSpan = True #Indicates whether the the DiscFactorSpan item
is to apply to the lane.
                        UseDiscFactorLane = True #Indicates whether the the DiscFactorLane item
is to apply to the lane.
                        DiscFactorSpan = 4 #The lane load discretization length along the lane
will not be more than 1/x of the span length where x is the DiscFactorSpan value.
                        #This item is only applicable when the UseDiscFactorSpan item is True.
                        DiscFactorLane = 10 #The lane load discretization length along the lane
will not be more than 1/x of the lane length where x is the DiscFactorLane value.
                        #This item is only applicable when the UseDiscFactorLane item is True.
                        Color = int()#The color assigned to the lane. (Optional)
                        Notes = ''#The notes, if any, assigned to the lane. (Optional)
                        ret = MyBridgeModeler.Lane.SetLaneFromLayoutLine(Name,
NumberLaneStations, LayoutLine, Station, Offset, Width, Radius, LaneType, LoadGroup,
LeftEdgeIsExterior, RightEdgeIsExterior,
                        MaxDiscAlongLane, MaxDiscAcrossLane, UseDiscFactorSpan,
UseDiscFactorLane, DiscFactorSpan, DiscFactorLane, Color, Notes)

                        if Tipo_analise == 'ENCONTROS':
                            ret = MyBridgeModeler.LayoutLine.AddNew('BLL0')

                            Name = 'BLL0'
                            CoordinateSystem = 'GLOBAL'
                            StartX = -1000
                            StartY = 0
                            StartZ = 0
                            ret = MyBridgeModeler.LayoutLine.SetGeneralData(Name,
CoordinateSystem, StartX, StartY, StartZ)

                            Name = 'BLL0'
                            NumberSegments = 2
                            SegmentType = [1, 2]
                            Station = [0, 1000]
```

```
                            Radius = [float(), float()]
                            Bearing = ['N900000E', 'N900000E']
                            ret   =   MyBridgeModeler.LayoutLine.SetHorizontalLayoutData(Name,
NumberSegments, SegmentType, Station, Radius, Bearing)

                            ret = MyBridgeModeler.Lane.AddNewFromLayoutLine('LANE0', 'BLL0')
                            ret   =   MyBridgeModeler.Lane.SetLaneFromLayoutLine('LANE0',   2,
['BLL0',  'BLL0'],  [0,  1000],  [0]*2,  [LarguraRolamento]*2,  [0]*2,  LaneType,  LoadGroup,
LeftEdgeIsExterior, RightEdgeIsExterior,
                            MaxDiscAlongLane,      MaxDiscAcrossLane,      UseDiscFactorSpan,
UseDiscFactorLane, DiscFactorSpan, DiscFactorLane, Color, Notes)

                            LoadNumber = 1 #The load assignment number.
                            NumberItems = 2 #The number of lanes loaded for the specified load
assignment number.
                            MyName = [f'{myLaneName}', 'LANE0'] #This is an array that includes
the name of each lane loaded for the specified load assignment number.
                            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Veiculo',
LoadNumber, NumberItems, MyName)
                            ret     =     SapModel.LoadCases.Moving.SetLanesLoaded('Multidao',
LoadNumber, NumberItems, MyName)




                        # 19 Ajustes finais do modelo

                        # 19.1 Rotulação diafragmas

                        ret = SapModel.SelectObj.Group('Diafragmas', False)

                        Name = '' # The name of an existing frame object or group, depending
on the value of the ItemType item.
                        ii = [False, False, False, False, True, True]
                        jj = [False, False, False, False, True, True]
                        # These are arrays of six booleans indicating the I-End and J-End
releases for the frame object.
                        # ii(0) and jj(0) = U1 release
                        # ii(1) and jj(1) = U2 release
                        # ii(2) and jj(2) = U3 release
                        # ii(3) and jj(3) = R1 release
                        # ii(4) and jj(4) = R2 release
                        # ii(5) and jj(5) = R3 release
                        StartValue = [float()] * 6
                        EndValue = [float()] * 6
                        # These are arrays of six values indicating the I-End and J-End partial
fixity springs for the frame object.
                        # StartValue(0) and EndValue(0) = U1 partial fixity [F/L]
                        # StartValue(1) and EndValue(1) = U2 partial fixity [F/L]
                        # StartValue(2) and EndValue(2) = U3 partial fixity [F/L]
                        # StartValue(3) and EndValue(3) = R1 partial fixity [FL/rad]
                        # StartValue(4) and EndValue(4) = R2 partial fixity [FL/rad]
                        # StartValue(5) and EndValue(5) = R3 partial fixity [FL/rad]
                        ItemType = eItemType(2)
                        # This is one of the following items in the eItemType enumeration:
                        # Object = 0
                        # Group = 1
                        # SelectedObjects = 2
                        # If this item is Object, the assignment is made to the frame object
specified by the Name item.
                        # If this item is Group, the assignment is made to all frame objects
in the group specified by the Name item.
                        # If this item is SelectedObjects, assignment is made to all selected
frame objects, and the Name item is ignored.
                        ret = SapModel.FrameObj.SetReleases(Name, ii, jj, StartValue, EndValue,
ItemType)

                    #     # Combinações de ações

                    #     Comb_dict = {
                    #             'Carga permanente': {'Type': 'Load Combo', 'ELU' : [1.35],
'ELU-fad' : 1, 'ELS-W': 1, 'EXC' : 1.1},
                    #             'Empuxo terra': {'Type': 'Load Case', 'ELU' : [1.35], 'ELU-
fad' : 1, 'ELS-W': 1, 'EXC' : 1.1},
```

```
                    #             'Retração e fluência': {'Type': 'Load Case', 'ELU' : [1.2],
'ELU-fad' : 0, 'ELS-W': 0, 'EXC' : 0},
                    #            'Protensão': {'Type': 'Load Case', 'ELU' : [1.2], 'ELU-fad' :
0, 'ELS-W': 0, 'EXC' : 0},
                    #              'TB-450': {'Type': 'Load Combo', 'ELU' : [CI * 1.5, CI *
1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                    #            'Frenagem e aceleracao': {'Type': 'Load Case', 'ELU' : [CI *
1.5, CI * 1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                    #             'Empuxo terra carga móvel': {'Type': 'Load Case', 'ELU' :
[CI * 1.5, CI * 1.05], 'ELU-fad' : CI * 0.7, 'ELS-W': CI * 0.5, 'EXC' : CI * 0.7},
                    #              'Vento': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-
fad' : 0, 'ELS-W': 0, 'EXC' : 0.6},
                    #              'Água': {'Type': 'Load Case', 'ELU' : [1.4, 0.84], 'ELU-
fad' : 0, 'ELS-W': 0, 'EXC' : 0.6},
                    #            'ENV-TEMP': {'Type': 'Load Combo', 'ELU' : [1.2, 0.72], 'ELU-
fad' : 0.3, 'ELS-W': 0.3, 'EXC' : 0.6},
                    #             'Impacto': {'Type': 'Load Case', 'ELU' : 0, 'ELU-fad' : 0,
'ELS-W': 0, 'EXC' : 1}
                    #            }

                    # AV = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel',
'Vento', 'Água', 'ENV-TEMP'] # Lista das ações variáveis

                    # Name = 'COMB1 - ELU (CM)' # Carga móvel como ação variável principal
                    # ComboType = 0
                    # ret = SapModel.RespCombo.Add(Name, ComboType)

                    # CName = ['Carga permanente', 'Empuxo terra', 'Retração e fluência',
'Protensão', 'TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel', 'Vento', 'Água',
'ENV-TEMP']
                    # AVP = ['TB-450', 'Frenagem e aceleracao', 'Empuxo terra carga móvel']
                    # for loadcase in CName:
                    #     CNameType = eCNameType(0) if Comb_dict[loadcase]['Type'] == 'Load
Case' else eCNameType(1)
                    #     SF = Comb_dict[loadcase]['ELU'][0] if loadcase in AVP or loadcase not
in AV else Comb_dict[loadcase]['ELU'][1]
                    #     ret = SapModel.RespCombo.SetCaseList(Name, CNameType, loadcase, SF)

                        # 19.2 User Discretization Points
                        Name = myBridgeObjName # The name of a bridge object.
                        SpanNames = [SpanName[0], SpanName[0], SpanName[1]] # A zero-based,
one-dimensional array of the span names associated with each discretization point. If a span
name is not recognized the discretization point will be skipped with no error indicated.
                        Distance = [e_a, x_S1, CVão/2 - CPerfil2] # A zero-based, one-
dimensional array of the distances of each discretization point from the start of the specified
span. The distance cannot specify a point closer than 100 times the specified auto merge
tolerance to either end of the span. if it is the discretization point will be skipped with no
error indicated. [L]
                        Bearing = ['Default']*3 # A zero-based, one-dimensional array of the
bearing of the mesh line across the bridge at the discretization point. The bearing may be
specified as Default or a valid bearing string.
                        ret = MyBridgeModeler.BridgeObject.SetUserDiscretizationPoints(Name,
SpanNames, Distance, Bearing)
                        x_girder_forces = [e_a, x_S1, (CVão/2)]
                        x_girder_forces = [elemento / 100 for elemento in x_girder_forces]
                        x_diaphragm_forces = [elemento / 100 for elemento in x_Diafragmas]
                        x_forces = list(np.unique(x_girder_forces + x_diaphragm_forces))

                        # 19.3 Update para discretização com os user discretization points
                        Name = myBridgeObjName
                        Action = 1
                        ret = MyBridgeModeler.BridgeObject.UpdateWithSameParameters(Name,
Action)

                        # 19.4 Pegar os esforços e tabelas em kN.m
                        ret = SapModel.SetPresentUnits(eUnits(kN_m_C))

                        # 19.5 Salvar e rodar modelo

                        ret = SapModel.File.Save(os.path.join(main_folder, Tipologia,
f'{Tipologia} ({Tipo_analise}) PILAR {altura/100} m VAOS DIFERENTES {CVão/100}_{Vão/100}.bdb'))

                        ret = SapModel.Analyze.SetRunCaseFlag('', True, True)

                        ret = SapModel.Analyze.RunAnalysis()
```

```
ret = SapModel.File.Save()

#wb.app.quit()
SapModel = None
mySAPObject = None
MyBridgeModeler = None
```