UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-graduação em Engenharia de Estruturas

Caroline Martins Calisto

ESTUDO NUMÉRICO DE VIGAS HÍBRIDAS DE AÇO SUBMETIDAS À FLEXÃO

Belo Horizonte

2022

Caroline Martins Calisto

ESTUDO NUMÉRICO DE VIGAS HÍBRIDAS DE AÇO SUBMETIDAS À FLEXÃO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Estruturas da Escola de Engenharia da Universidade Federal de Minas Gerais como parte dos requisitos necessários à obtenção do título de "Mestre em Engenharia de Estruturas".

Orientador: Prof. Dr. Hermes Carvalho

Coorientadora: Prof. Dra. Ana Lydia Reis de Castro e Silva

Belo Horizonte

2022

Calisto, Caroline Martins.
C154e Estudo numérico de vigas híbridas de aço submetidas à flexão [recurso eletrônico] / Caroline Martins Calisto. - 2022.
1 recurso online (216 f. : il., color.) : pdf.

Orientador: Hermes Carvalho.
Coorientadora: Ana Lydia Reis de Castro e Silva.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices e anexos: f. 110-216.

Bibliografia: f. 106-109.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia de estruturas - Teses. 2. Aço - Estruturas - Teses. 3. Vigas - Teses. 4. Analise funcional não-linear - Teses. I. Carvalho, Hermes. II. Castro e Silva, Ana Lydia Reis de. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 624(043)

Ficha catalográfica elaborada pela Bibliotecária Roseli Alves de Oliveira CRB/6 2121
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ESTRUTURAS

PROPEEs

UFMG

ATA DA DEFESA DE DISSERTAÇÃO DE MESTRADO EM ENGENHARIA DE ESTRUTURAS Nº: 394 DA ALUNA CAROLINE MARTINS CALISTO

Às **09:00** horas do dia **22** do mês de **dezembro** de **2022**, reuniu-se, na Escola de Engenharia da Universidade Federal de Minas Gerais - UFMG, a Comissão Examinadora indicada pelo Colegiado do Programa em **16 de dezembro de 2022**, para julgar a defesa da Dissertação de Mestrado intitulada **"Estudo Numérico de Vigas Híbridas de Aço compactas e esbeltas com seção transversal do Tipo I"**, cuja aprovação é um dos requisitos para a obtenção do Grau de MESTRE EM ENGENHARIA DE ESTRUTURAS na área de ESTRUTURAS.

Abrindo a sessão, o Presidente da Comissão, **Prof. Dr. Hermes Carvalho**, após dar a conhecer aos presentes o teor das Normas Regulamentares passou a palavra à candidata para apresentação de seu trabalho. Seguiu-se a arguição pelos examinadores, com a respectiva defesa da candidata. Logo após, a Comissão se reuniu, sem a presença da candidata e do público, para julgamento e expedição do resultado final. Foram atribuídas as seguintes indicações

Prof. Dr. Hermes Carvalho - DEES - UFMG (Orientador)

Profa. Dra. Ana Lydia Reis de Castro e Silva - DEES-UFMG (Coorientadora)

Prof. Dr. Rodrigo Barreto Caldas - DEES - UFMG

Prof. Dr. Renata Gomes Lanna da Silva - CEFET-MG

Após reunião, a Comissão considerou a candidata APROVADA, conforme pareceres em anexo.

O resultado final foi comunicado publicamente a candidata pelo Presidente da Comissão. Nada mais havendo a tratar, o Presidente encerrou a reunião e lavrou a presente ATA, que será assinada por todos os membros participantes da Comissão Examinadora.

Belo Horizonte, 22 de dezembro de 2022.

Observações:

1. A aprovação da candidata na defesa da Tese de Doutorado não significa que a mesma tenha cumprido todos os requisitos necessários para obtenção do Grau de Doutor em Engenharia de

Estruturas;

2. Este documento não terá validade sem a assinatura do Coordenador do Programa de Pós-Graduação.

Documento assinado eletronicamente por **Hermes Carvalho**, **Professor do Magistério Superior**, em 22/12/2022, às 12:21, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Renata Gomes de Lanna da Silva**, **Usuário Externo**, em 22/12/2022, às 14:14, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Ana Lydia Reis de Castro e Silva**, **Professora do Magistério Superior**, em 22/12/2022, às 14:24, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rodrigo Barreto Caldas**, **Professor do Magistério Superior**, em 28/02/2023, às 16:26, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Felicio Bruzzi Barros**, **Coordenador(a) de curso de pós-graduação**, em 02/02/2024, às 10:30, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1970742** e o código CRC **1644338B**.

AGRADECIMENTOS

A Deus pelo dom da vida, pela saúde, por me dar forças e por guiar o meu caminho.

Aos meus pais, Jaqueline e Paulo, pelo amor incondicional, pela preocupação e por sempre serem presentes.

A minha irmã, Graziele, pelo amor e por me incentivar nesse trabalho.

Ao Quarteto: Bruna, Fernanda e Ana Luiza pela amizade de anos, pela irmandade, pelo apoio, por sempre estarem presentes mesmo quando a distância física é grande e por sempre acreditarem em mim.

A minha amiga, Jovana, por entender minha ausência durante a elaboração desse trabalho, por estar sempre presente e por sempre confiar em mim.

A minha amiga, Joana, por nossa reaproximação depois de alguns anos, pela preocupação e pelo carinho de sempre.

A minha amiga, Ariany, por ser o presente que o mestrado e Natal me proporcionaram, por me ajudar sempre, pelo carinho e pela preocupação.

Aos meus amigos da pós-graduação, em especial ao Ruan (agradeço pela parceria, pelos perrengues e por toda a ajuda ao longo do caminho), ao Elvys (obrigada pelo apoio, pela parceria e pelo exemplo de dedicação), a Bárbara (pela amizade, pela preocupação e pelo carinho), a Isadora (por transmitir seus conhecimentos da linguagem de programação Phyton e pela alegria), a Isabela (pela amizade construída em Natal e pelas gargalhadas compartilhadas). Agradeço também a todos os colegas do mestrado que fizeram parte dessa trajetória.

Agradeço ao meu orientador Dr. Hermes Carvalho, por me aceitar como sua orientanda, pela sugestão do tema, pela dedicação, por todo conhecimento transmitido e pela confiança no meu trabalho.

A minha coorientadora, Dr. Ana Lydia Reis de Castro e Silva, falar do conhecimento repassado seria pouco perto do que você representou para mim ao longo desse trabalho. Agradeço imensamente pela paciência, pela ajuda, pela disponibilidade, por sempre acreditar que seria possível, pelas palavras e áudios de apoio mesmo quando eu achava que não estava no caminho certo, enfim, por ser luz na minha caminhada para construção dessa dissertação.

Agradeço também a todos os professores do Departamento de Engenharia das Estruturas que não foram citados diretamente.

Ao pessoal da secretaria do departamento, Juliana, Lucíola, Patrícia e Luciell, pela disponibilidade e atenção de sempre.

Agradeço às instituições de fomento, CAPES, CNPq e FAPEMIG que colaboraram com esse trabalho e contribuem com o desenvolvimento de tantos trabalhos ao redor do país.

“Tudo o que um sonho precisa para ser realizado é alguém que acredite que ele possa ser realizado.”

Roberto Shinyashiki

RESUMO

Este trabalho apresenta um estudo do comportamento de vigas híbridas de aço com seção transversal do tipo I submetidas à flexão. Foram desenvolvidos e validados modelos numéricos em elementos finitos no software ABAQUS. Nos modelos, desconsidera-se a flambagem lateral com torção, uma vez que a viga será contida lateralmente, de maneira que somente as instabilidades locais estejam presentes e assim avaliadas. As análises foram divididas em duas etapas, a saber: primeiramente, realizou-se a análise de flambagem elástica para obter as cargas críticas de flambagem e os modos de flambagem das vigas compactas e esbeltas; posteriormente, efetuou-se a análise da capacidade resistente última das vigas considerando tensões residuais e imperfeições iniciais. Após definido o modelo numérico, desenvolveu-se a partir de uma linguagem de programação (*Python*) uma rotina para execução dos modelos e realizou-se um estudo de sensibilidade da resistência ao escoamento do aço da mesa. Além disso, com a finalidade de alinhar os resultados obtidos com a expectativa comercial das soluções, foram avaliados os custos de cada modelo numérico. Os resultados foram satisfatórios e, como esperado, mostraram que as vigas híbridas resistem a um esforço maior de momento fletor quando comparadas com suas vigas homogêneas correspondentes. Concluiu-se que as vigas híbridas, no geral, possuem uma resistência 33% maior do que as vigas homogêneas correspondentes, enquanto que o aumento no custo foi de apenas 3%.

Palavras-chave: vigas híbridas de aço; instabilidade local; seção transversal do tipo I; análise não linear.

ABSTRACT

This work presents a study of I-shaped hybrid steel girders behavior subjected to bending. For this purpose, numerical models with finite elements were developed and validated in the ABAQUS software. In these models is not considered the lateral torsional buckling since the beam will be restrained against out-of-plane translation, so only the local instabilities were evaluated. The analyses were divided into two stages, namely: first, it was performed a linearized eigenvalue buckling analysis to obtain the critical buckling loads and the buckling modes; then, it was performed an analysis of the ultimate resistant capacity considering residual stresses and initial imperfections. After defining the numerical model, the routine for running the models was developed from a programming language (*Python*) and a sensitivity study of the flange yield strength was carried out. Furthermore, the costs of the models were evaluated in order to make in parallel the results obtained with the expectations of commercial solutions. The outcomes obtained from the numerical models developed were satisfactory and, as expected, showed that the hybrid steel girders resist a greater bending when compared with their equivalent homogeneous beams. It was concluded that hybrid girders, in general, have a resistance 33% higher than their equivalent homogeneous beams, while the increase in cost was only 3%.

Keywords: hybrid steel girders; local instabilities; I-section girders; nonlinear analysis.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Letras romanas minúsculas

a	Distância da carga até a extremidade da viga
b	Largura do elemento
b_{eff}	Largura efetiva
b_f	Largura da mesa
b_{fb}	Largura da mesa inferior
b_{fu}	Largura da mesa superior
$b_{i,edge,eff}$	Largura efetiva de extremidade
$b_{loc,i}$	Largura da parte comprimida de cada sub-painel i individualmente
b_{sl}	Largura do enrijecedor longitudinal retangular
f_y	Resistência ao escoamento do aço
f_{yf}	Resistência ao escoamento do aço da mesa
f_{yw}	Resistência ao escoamento do aço da alma
f_u	Resistência à ruptura do aço à tração
h	Altura da alma
h'	Altura da alma considerada no plano médio da seção transversal
h_{wi}	Altura livre de sub-painéis *i*
i	Raio de giração do enrijecedor
u_x	Translações nos eixos x
u_y	Translações nos eixos y
u_z	Translações nos eixos z
t	Espessura do elemento
t_f	Espessura da mesa
t_{fb}	Espessura da mesa inferior
t_{fu}	Espessura da mesa superior
t_p	Tempo de processamento do modelo numérico
t_r	Tempo de processamento do modelo de referência
t_{sl}	Espessura do enrijecedor longitudinal retangular
t_w	Espessura da alma

Letras romanas maiúsculas

A_c	Área bruta da parte comprimida da placa enrijecida
$A_{c,eff}$	Área efetiva final da zona comprimida
$A_{c,eff,loc}$	Soma das áreas efetivas dos sub-painéis e enrijecedores total ou parcialmente comprimidos
A_{eff}	Área efetiva da seção transversal
$A_{sl,1}$	Área bruta do enrijecedor
$A_{sl,1,eff}$	Área efetiva da seção transversal do enrijecedor e das as partes adjacentes da placa
$A_{sl,eff}$	Soma das áreas efetivas dos enrijecedores longitudinais
E	Módulo de Young
$F_{EN\ 1993-1-5}$	Resistência última de acordo com a EN 1993-1-5
I	Momento de inércia
I_{eff}	Momento de inércia de área
$I_{sl,1}$	Momento de inércia de área do enrijecedor
L	Comprimento destravado da viga
M	Momento fletor
$M_{f,Rd}$	Momento fletor resistente de plastificação de cálculo
M_{max}	Momento máximo
M_p	Momento totalmente plástico
M_{pl}	Momento fletor de plastificação
$M_{pl,Rd}$	Momento fletor resistente de plastificação de cálculo
$M_{Rd,x}$	Momento fletor resistente
P	Carga
$V_{bf,Rd}$	Capacidade resistente das mesas
$V_{bw,Rd}$	Capacidade resistente da alma
W_{eff}	Módulo elástico efetivo
$W_{el,eff,x}$	Módulo de flexão efetivo

Letras gregas

α_e	Parâmetro de imperfeição

ε	Fator de escoamento
ε_{st}	Deformação normal específica correspondente à tensão limite da região de plastificação
ε_u	Deformação de ruptura
ε_y	Deformação normal específica correspondente à tensão limite da região elástica
Δ_{max}	Deslocamento máximo
θ	Rotação
θ_{max}	Rotação máxima
θ_p	Rotação correspondente a primeira rótula plástica
θ_{ru}	Rotação plástica última
θ_u	Rotação última
$\mu_{\theta r}$	Capacidade de rotação
$\sigma_{cr,c}$	Tensão crítica flambagem elástica *Tipo Pilar*
$\sigma_{cr,p}$	Tensão crítica flambagem elástica *Tipo Placa*
$\sigma_{cr,sl}$	Tensão de flambagem
σ_w	Tensão na alma
ρ_{loc}	Fator de redução para flambagem de placa
$\bar{\lambda}_p$	Esbeltez da placa
$\bar{\lambda}_w$	Esbeltez normalizada da alma
ψ	Razão de tensões nas duas bordas da placa
k_σ	Coeficiente de flambagem
k_τ	Coeficiente de flambagem por cisalhamento
$k_{\tau i}$	Coeficiente de flambagem por cisalhamento de sub-painéis *i*
υ	Coeficiente de Poisson
ρ_c	Fator de redução da flambagem *Tipo Placa*
$\rho_{loc,i}$	Fator de redução de cada sub-painel i
χ_c	Fator de redução da flambagem *Tipo Pilar*
χ_w	Fator de redução para a capacidade resistente à flambagem por cisalhamento da alma
γ_{M1}	Fator parcial de segurança para capacidade resistente relacionada à instabilidade
φ_y	Rotações em torno de y
φ_z	Rotações em torno de z

LISTA DE ABREVIATURAS E SIGLAS

AAR	Aços de alta resistência
AASHTO	*American Association of State Highway and Transportation Officials*
ABNT	Associação Brasileira de Normas Técnicas
AISC	*American Institute of Steel Construction*
EN	*European Committee for Standardization*
FLA	Flambagem local da alma
FLM	Flambagem local da mesa
FLT	Flambagem lateral com torção
LVDT	*Linear Variable Differential Transformer*
MEF	Método dos Elementos Finitos
MLE	Método da Largura Efetiva
MTR	Método da Tensão Reduzida
TFA	*Tension Field Action*
UFMG	Universidade Federal de Minas Gerais

SUMÁRIO

1 INTRODUÇÃO

1.1 Aspectos gerais

Segundo Chacón (2009), os aços de alta resistência (AAR) possibilitam novas opções para projetos de estruturas, mas há um descompasso entre os avanços científicos e tecnológicos e a aplicação prática, devido à falta de normas para esses tipos de aço. Prova disso é que, na Europa, somente a pouco tempo foi criada a EN 1993-1-12:2007 a qual oferece especificações para o uso de aços S460 e S700.

Em contrapartida, as vantagens da utilização dos aços de alta resistência são cada vez mais notáveis e uma das suas aplicabilidades refere-se às vigas híbridas. Nesse tipo de viga, o perfil I é soldado e possui diferentes tipos de aço nas mesas e na alma, sendo que, na maioria dos casos, as mesas possuem um aço de maior resistência. Segundo Beg *et al*. (2010), as vigas híbridas são consideradas como uma solução econômica, pois o material mais resistente e, consequentemente, mais caro (aço de alta resistência) somente é utilizado nos locais mais solicitados do perfil.

Segundo Gogou (2012), ao comparar projetos híbridos e seus projetos homogêneos equivalentes para a ponte 'Schellingwouderbrug', localizada nos Países Baixos, verificou-se que os projetos híbridos resultaram em uma economia de peso de até 65% em alguns casos. Ainda de acordo com Gogou (2012), o preço elevado dos aços de alta resistência tem como consequência um maior custo para projetos híbridos, em torno de 4%. Porém, a redução de peso nesses projetos é bastante significativa, o que acarreta uma redução de até 6% quando se inclui a fabricação, transporte, montagem e manutenção. Dessa forma, esse design híbrido tem sido empregado em pontes nos Estados Unidos, na Europa e no Japão.

1.2 Justificativa

Alguns exemplos de uso das vigas híbridas estão associados à economia e ao fato de serem mais leves do que as vigas convencionais. Pesquisas realizadas em outros países têm demonstrado que esse assunto é relevante e fornece vantagens comprovadas, conforme Figura 1.1. No Brasil, o assunto ainda é pouco conhecido e carece de pesquisas e normativas próprias, que possam estimular e alavancar o uso dessa solução para aplicação em pontes, edifícios e outros. Além de não abordar o dimensionamento de elementos híbridos, a ABNT NBR

8800:2008 contempla estruturas com classe de resistência ao escoamento até 450 MPa, sendo os AAR não abrangidos por seus cálculos e verificações.

Para que seja possível realizar um estudo mais consistente sobre o tema é necessário realizar estudos paramétricos, análises estatísticas, definição de equações de dimensionamento dos perfis, entre outras investigações. Para isso, é essencial obter um modelo numérico confiável e calibrado por modelo experimental.

(a) (b)

Figura 1.1 - Utilização de perfis híbridos em pontes: (a) Carolina do Sul, E.U.A. (Fenkel et al., 2007) e (b) Rångedala, Suécia (COMBRI, 2008).

1.3 Objetivos

1.3.1 Objetivo geral

O presente trabalho tem como objetivo a realização de um estudo numérico da capacidade resistente de vigas compostas por diferentes aços (híbridas), com seção transversal do tipo I, compactas e esbeltas submetidas à flexão.

1.3.2 Objetivos específicos

Os objetivos específicos são:

- Realizar um estudo numérico no programa comercial ABAQUS de maneira a validar os estudos numéricos e experimentais obtidos por Shokouhian (2014) e Sinur (2011);
- Elaborar, a partir de uma linguagem de programação (*Python*), rotinas para execução dos modelos, de vigas híbridas, descritas em detalhes no Apêndice B e no Apêndice C;
- Efetuar um estudo de sensibilidade da resistência ao escoamento do aço empregado nos flanges dos perfis I;

- Avaliar os custos de produção dos elementos estruturais híbridos, com o seu posterior impacto econômico no mercado.

1.4 Organização do texto

Neste Capítulo 1 apresenta-se uma introdução ao tema, a justificativa e os objetivos desse estudo.

O Capítulo 2 apresenta uma revisão bibliográfica dos principais trabalhos sobre vigas híbridas e alguns conceitos relevantes.

No Capítulo 3 encontra-se de maneira resumida o procedimento, de acordo com a EN 1993-1-5:2006, de verificação de seções esbeltas enrijecidas longitudinalmente para comportamento último plástico.

O Capítulo 4 detalha os trabalhos de Shokouhian (2014) e Sinur (2011), os quais são os estudos que foram utilizados como referência para elaboração da metodologia utilizada para o desenvolvimento do modelo numérico no programa comercial ABAQUS, a qual também está presente neste capítulo.

O Capítulo 5 apresenta os resultados obtidos a partir da análise da sensibilidade da variação dos parâmetros no modelo numérico verificado no capítulo anterior.

Já o Capítulo 6 e o Capítulo 7 apresentam, respectivamente, as conclusões desse projeto de pesquisa e propostas de trabalhos futuros e as referências bibliográficas citadas no decorrer deste trabalho.

E, por fim, na seção pós-texto é possível observar os anexos e apêndices que complementam o estudo elaborado ao longo dos capítulos.

2 REVISÃO DA LITERATURA E CONCEITOS RELEVANTES

2.1 Generalidades

Apresenta-se nesse capítulo os principais trabalhos desenvolvidos relacionados ao comportamento das vigas híbridas. Também são descritos, de maneira sucinta, os principais conceitos utilizados no decorrer dos capítulos. Ao final apresenta-se uma síntese dos procedimentos e formulações matemáticas de verificação de seções esbeltas enrijecidas longitudinalmente para comportamento último plástico de acordo com a norma europeia EN 1993-1-5:2006.

2.2 Breve histórico de estudos pregressos de vigas híbridas

Nesse tópico apresenta-se uma revisão sobre os principais trabalhos desenvolvidos relacionados ao comportamento das vigas híbridas a partir do estudo da arte elaborado por Chacón (2009) e aprimorado 5 anos depois em Chacón (2014) e do estudo de Kulkarni & Gupta (2017).

Levando em consideração esses estudos, a primeira proposta de projeto híbrido foi apresentada na América do Norte por Wilson em 1944. Sua investigação foi baseada em uma solução relacionada ao peso e ao custo do material de uma viga com aço-carbono na alma e do aço-silício nas mesas. Mas, somente em 1961, esse conceito foi reintroduzido nas pesquisas por Haaijer em seu trabalho sobre os perfis de alta resistência.

Posteriormente, em 1964, Frost & Schilling apresentaram um trabalho mais completo que englobou o campo teórico e experimental sobre o comportamento das vigas híbridas de aço sob momento fletor puro e sob a combinação de momento e cortante, desenvolvendo gráficos da relação momento versus rotação. Considera-se que as primeiras regras para dimensionamento de vigas híbridas da AASHTO em 1968 foram baseadas nesse trabalho.

Sarsam (1966) investigou o comportamento estrutural e estudou a distribuição de tensões das vigas híbridas formadas com o aço A441 nas mesas e aço A36 na alma e sujeitas à flexão pura e combinação de flexão e cortante. Uma das conclusões do autor é que o limite mais prático para o dimensionamento de vigas híbridas é a resistência ao escoamento da mesa.

Em 1967, Schilling apresentou o primeiro artigo sobre vigas híbridas submetidas ao carregamento concentrado, com o objetivo de avaliar o potencial de escoamento da alma

causado pelo momento associado a esse carregamento. Para isso, o autor realizou dois testes na mesma viga híbrida, sendo que no primeiro uma força transversal compressiva foi aplicada na mesa comprimida e no segundo, a força transversal foi aplicada na mesa tracionada. Concluiu que as forças podem ser aplicadas em qualquer mesa do perfil mesmo quando a tensão longitudinal na mesa está próxima à resistência ao escoamento. Com relação à carga última, esta foi definida como a carga na qual a linearidade do gráfico carga *versus* deslocamento não é mais observada. Foi inferido por Chacón (2009) que essas vigas são capazes de se submeterem a cargas maiores do que as definidas pelos autores.

No mesmo ano, Lew & Toprac (1967) realizaram testes estáticos em seis vigas híbridas compostas do aço A514 nas mesas e aço A36 na alma. Cinco das seis vigas foram submetidas à flexão pura e a outra à cortante. O objetivo dos ensaios de flexão era fornecer informações sobre o comportamento de flambagem da mesa comprimida e como era afetado pelo escoamento da alma. Além disso, determinar as cargas últimas e comparar com os valores teóricos. O autor concluiu que a teoria do campo de tração pode ser utilizada para prever a carga última para vigas híbridas sujeitas a valores altos de cisalhamento.

Schilling e Carskaddan apresentaram outros trabalhos em 1968. Carskaddan (1968) introduziu uma aproximação teórica em uma possível aplicação de vigas híbridas adicionando uma laje de concreto, chamada de *composite hybrid girder*, e Schilling (1968) relacionou a instabilidade de vigas híbridas sem enrijecimento. Para isso, Schilling propôs uma fórmula aproximada para o dimensionamento e realizou comparações entre vigas híbridas e homogêneas com uma laje de concreto. Já a investigação de Carskaddan foi conduzida de maneira a determinar a máxima relação de esbeltez da alma que pode ser usada para uma viga híbrida sem enrijecedor sem causar problemas de instabilidade. Primeiramente, Carskaddan desenvolveu uma aproximação teórica a partir da análise de energia e então, um programa experimental foi executado. Embora seu estudo teórico tenha mostrado que para altas tensões normais devidas à flexão, o escoamento da alma reduz de forma significativa a rigidez da mesma e, consequentemente, a resistência ao cisalhamento, o programa desenvolvido mostrou uma carga última consideravelmente maior do que as previstas teoricamente.

Foi apresentado por Maeda (1971) um estudo sobre a resistência estática última de vigas híbridas em flexão. Para isso, quatro vigas híbridas soldadas com enrijecedores longitudinais submetidas a duas cargas concentradas foram testadas até o colapso, sendo que a variável

adotada no estudo foi a esbeltez da alma (150, 200, 250 e 300). A carga crítica, o momento último, o modo de falha e a localização da falha foram alguns dos resultados obtidos pelo autor. Além disso, a contribuição da resistência pós-flambagem da alma na resistência última a flexão, o efeito dos enrijecedores longitudinais na resistência pós-flambagem da alma e o modo de falha final sendo controlados pela rigidez da mesa comprimida e dos enrijecedores longitudinais foram observados nas vigas híbridas e homogêneas. O autor concluiu que o uso do aço de alta resistência na mesa sob tração é benéfico para a parte da seção que está comprimida. Isso ocorre porque a linha neutra da seção se move para cima devido ao fato da área da mesa tracionada ser aproximadamente 40% menor do que em vigas convencionais.

Após Maeda (1971), Nethercot abordou a estabilidade das vigas híbridas a partir das flambagens elásticas e inelásticas no ano de 1976. Para isso, o autor usou uma técnica numérica geral e os resultados obtidos mostraram que o escoamento precoce da alma levou a um pequeno efeito na estabilidade lateral da viga, enquanto que, as tensões residuais apresentaram um efeito significativo nesse caso.

Os únicos registros encontrados sobre programação de vigas híbridas datam de 1976, 1986 e 1987. Em 1976, Chong (1976) *apud* Abuyounest & Adeli (1987) estudou a otimização das vigas híbridas sem enrijecimento considerando o menor custo de otimização para determinar o momento fletor e a força cortante. Já em 1986, Adeli & Phan (1986) desenvolveram um programa para o dimensionamento de vigas híbridas e homogêneas soldadas de acordo com a norma AISC de 1980. Posteriormente, Abuyounest & Adeli (1987) apresentaram um projeto de peso mínimo para vigas híbridas com ou sem enrijecimento submetidas a uma carga arbitrária a partir de uma implementação no programa FORTRAN.

Na Suécia em 1994, Åhlenius também contribuiu no âmbito das pesquisas sobre vigas híbridas com um manual de regras gerais bastante extenso. O conteúdo do manual está voltado para a fadiga, o cálculo do momento fletor e da força cortante resistente e a interação entre esses esforços.

Axhag (1998) abordou em seus experimentos quatorze vigas, nas quais duas eram híbridas com seção transversal do tipo I sendo uma delas esbelta submetidas à flexão, a fim de determinar a relação momento *versus* rotação para cada viga. As almas das vigas híbridas possuíam um aço

com resistência ao escoamento (f_y) igual a 220 N/mm² enquanto as mesas eram compostas por um aço com resistência ao escoamento (f_y) igual a 700 N/mm².

Barker & Schrage (2000) apresentaram os benefícios do aço de alta resistência para vigas de uma ponte ao analisar três vigas compostas por um AAR com $f_y = 485$ N/mm², duas vigas constituídas por aço convencional com $f_y = 345$ N/mm² e uma viga híbrida com os dois tipos de aço mencionados acima. Como conclusão, o projeto híbrido mostrou benefícios, como sendo a estrutura mais leve e com menor custo. Em contrapartida, embora as vigas homogêneas compostas pelo AAR apresentaram uma significativa redução no peso, o custo desse material superou os benefícios de se projetar uma estrutura mais leve. A Tabela 2.1 apresenta a comparação do peso e do custo entre as três alternativas de projeto para as vigas com vão igual a 3250 mm, analisadas por Barker & Schrage (2000).

Tabela 2.1 – Comparação entre as vigas de aço (Barker & Schrage, 2000).

Alternativas de projeto	**Peso do aço (t)**	**Redução no peso (%)**	**Custo total (US$)**	**Redução no custo (%)**
f_y = 345 N/mm²	310,5	Referência	505132,00	Referência
f_y = 485 N/mm²	270,6	-12,8%	550036,00	8,9%
Projeto híbrido	276,7	-10,9%	449591,00	-11,0%

De acordo com Chacón (2009), o avanço com relação a resistência a cortante das vigas híbridas deu-se em 2002, quando Barker *et al*. apresentaram uma abordagem teórica com o objetivo de avaliar a influência do escoamento da alma na resistência pós-flambagem, chamada de *tension field action (TFA)* - ação do campo de tração. No ano seguinte, Greco & Earls (2003) apresentaram um estudo experimental de vigas híbridas de alta resistência com seção transversal do tipo I. A análise foi realizada com relação a seções transversais compactas e os critérios de suporte adotados. Os autores concluíram que AASHTO de 1998 e AISC de 1999 eram insuficientes para fornecer a capacidade de rotação necessária para a redistribuição da força inelástica e perfis híbridos de alta performance. E após alguns anos, Barker (2005) desenvolveu um programa para validar seu estudo inicial.

A partir de então, outros autores também contribuíram para essa investigação, sendo eles: Rush em 2001, Zentz em 2002 e Azizinamini *et al*. (2007). Os autores obtiveram a força cortante de vigas híbridas ao considerar a *TFA*. Outros resultados desses estudos proporcionaram a retirada das restrições relacionadas às vigas híbridas e as regras quanto a interação entre cortante e momento na versão da AASHTO de 2004. Sendo assim, as regras para a força cortante aplicada a partir de então são as mesmas para vigas homogêneas e híbridas.

Ito *et al*. (2005) estudaram a capacidade de rotação das vigas híbridas e concluíram que para as condições adotadas as vigas híbridas possuíam uma capacidade de deformação maior do que as vigas homogêneas e que a curva momento *versus* rotação é mais conservativa quando se considera uma maior esbeltez da alma.

Um projeto foi desenvolvido para verificações de determinação da classe da seção transversal, da resistência ao momento fletor, da resistência ao cisalhamento, forças transversais e fadiga cujos resultados foram apresentados por Bitar *et al*. em 2003, Veljkovic & Johansson (2004) e Johansson & Collin (2005). Com isso, Chacón (2009) concluiu que o dimensionamento das vigas híbridas pode ser realizado utilizando as regras para dimensionamento de viga homogêneas de acordo com a EN 1993-1-5:2006 com algumas modificações.

Dois trabalhos de Barth *et al*. (2007) também contribuíram para o estado da arte de vigas híbridas. Na primeira pesquisa, os autores avaliaram a aplicabilidade da segunda e terceira edições da AASHTO para vigas híbridas com o aço de maior resistência nas mesas do perfil. No segundo trabalho, foi avaliado se os procedimentos da AASHTO de 2004 poderiam ser aplicados em vigas híbridas contendo o aço HPS 485W. Também em 2007, a utilização dos aços de alta resistência e o dimensionamento das vigas híbridas foram discutidos em COMBRI (2008), o qual é um programa de pesquisa da *Research Fund for Coal and Steel*.

No mesmo ano, Fenkel *et al*. (2007) conduziram um estudo com medições *in loco* que apresentou resultados analíticos e experimentais, sendo que o trabalho teve como foco a investigação da performance estrutural e a avalição de alternativas de projeto para uma ponte em Gaffney, na Carolina do Sul. As vigas possuíam o aço HPS 70W nas regiões de momentos negativos. Uma das conclusões obtidas, para esse caso, foi que a viga híbrida pode se comportar de maneira similar a viga homogênea do aço HPS 70W devido ao fato de que as mesas, que contém o aço de maior resistência, correspondem à parte do perfil que resistem à maior parcela do momento.

Em 2008, Real *et al*. realizaram testes em vigas híbridas sujeitas à cortante e os resultados foram utilizados para prever a capacidade resistente última dessas vigas. Em paralelo, Petel *et al*. mostraram as vantagens relacionadas à redução do peso que as vigas híbridas proporcionam utilizando-se a EN 1993-1-5:2006.

Chacón (2009) abordou vigas híbridas submetidas a cargas concentradas. Nesse trabalho, o autor propôs uma modificação no procedimento implementado por EN 1993-1-5:2006. Na sequência, Chacón *et al*. (2010) apresentaram resultados experimentais encontrados na literatura assim como simulações numéricas desenvolvidas pelos autores, além disso, avaliaram a formulação de EN 1993-1-5:2006 e algumas peculiaridades sobre as vigas híbridas submetidas à carga concentrada. No ano seguinte, Chacón *et al*. (2011) realizaram simulações em vigas híbridas longitudinalmente enrijecidas submetidas a cargas concentradas. Em 2012, Chacón *et al*. (2012) também abordaram as vigas híbridas sujeitas a cargas concentradas e fizeram sugestões para EN 1993-1-5:2006, visto que o estudo levou a resultados contrários à segurança. Em 2013, Chacón *et al*. (2013) estudaram duas séries de três vigas híbridas e obtiveram resultados de capacidade resistente última assim como respostas estruturais. E por último, Chacón (2014) realizou um trabalho sobre o estado da arte de vigas híbridas.

Ajeesh (2011) teve seu estudo embasado na resistência ao cisalhamento de vigas híbridas com relação a alguns parâmetros como, por exemplo, a esbeltez. No ano de 2012, Gogou (2012) comparou o custo base para alternativas preliminares de projeto para a ponte 'Schellingwouderbrug' e concluiu que, estimando os custos totais, as vigas híbridas (combinação dos aços S355 e S690) levaram a uma solução mais econômica para a ponte se comparada com viga equivalente homogênea com S355 em toda a sua seção.

Shokouhian & Shi (2014), Shokouhian & Shi (2015) e Shokouhian *et al.* (2016) apresentaram resultados teóricos, analíticos e experimentais sobre capacidade de rotação, ductibilidade e modos de falha de vigas híbridas e finalmente, foram sugeridas equações empíricas. Também em 2015, Chacón & Rojas-Blonval (2015) apresentaram uma análise de vigas híbridas sujeitas a forças de cisalhamento. O estudo foi baseado em uma compilação bibliográfica de ensaios experimentais encontrados na literatura com um estudo numérico conduzido pelos autores. Em 2013, Rojas-Blonval (2013) já havia investigado a instabilidade por cortante nas vigas híbridas.

Wang *et al.* (2016) estudaram treze vigas híbridas com resistência ao escoamento do aço (f_y) igual a 485 N/mm² nas mesas e com resistência ao escoamento do aço (f_y) igual a 235 e 345 N/mm² na alma. Adicionalmente os autores também abordaram cinco vigas com diferentes resistências ao escoamento do aço (f_y) nas mesas. Eles concluíram que a esbeltez da mesa ou da alma é o principal fator que afeta a capacidade de flexão e a ductibilidade da viga. Após, Kulkarni & Gupta (2017) realizaram testes de flexão em cinco vigas com seção transversal do

tipo I, sendo duas homogêneas e três híbridas. As vigas híbridas foram testadas somente para cargas estáticas e algumas sugestões foram realizadas para a norma EN 1993-1-1:2005.

Subramanian & White (2016) avaliaram a contribuição de um enrijecedor longitudinal no estado de escoamento limite de resistência à flexão em vigas híbridas e homogêneas esbeltas de aço, sendo que nas vigas híbridas a resistência ao escoamento do aço (f_y) igual a 485 N/mm² nas mesas e com resistência ao escoamento do aço (f_y) igual a 345 N/mm² na alma. Com base nos resultados os autores recomendaram um modelo de seção transversal que captura a resposta de pós-flambagem da alma para ambos tipos de vigas sujeitas a momento uniforme.

Ghadami & Broujerdian (2018) investigaram, teórica e experimentalmente, a interação entre momento e cortante no comportamento das vigas híbridas em temperatura ambiente e em temperaturas elevadas. Como resultado, os autores propuseram equações teóricas para alcançar a curva da interação momento *versus* cortante sem considerar a flambagem por cisalhamento. Já Shokouhian *et al.* (2018) investigaram a interação dos modos de flambagem das vigas híbridas com seção transversal do tipo I submetidas à flexão, na qual um método baseado na esbeltez é apresentado para tratar da ductilidade e da capacidade do momento afetados pelas instabilidades locais e globais.

Recentemente, Biscaya et al. (2019) abordaram a interação entre momento e cortante de vigas híbridas comprimidas e enrijecidas longitudinalmente porém, nos casos analisados, as mesas possuíam um aço de menor resistência (f_y=335 N/mm²) do que a alma (f_y=344 N/mm²). E Lalthazuala & Singh (2019) apresentaram estudos numéricos em três tipos de vigas, sendo um deles considerando uma viga híbrida sujeita a flexão com resistência ao escoamento do aço (f_y) igual a 460 N/mm² nas mesas e com resistência ao escoamento do aço (f_y) igual a 345 N/mm² na alma.

Além dos estudos de vigas híbridas, também é importante ressaltar alguns estudos recentes e importantes sobre pilares e vigas-pilares que abordam aços de alta resistência. Alguns deles são Su *et al.* (2021) e Filho *et al.* (2022).

Tabela 2.2 - Pesquisas relacionadas às vigas híbridas com seção transversal do tipo I sem furos/aberturas apresentadas de forma cronológica (adaptado de Chacón, 2014).

Pesquisadores	Ano	Tipo de pesquisa	Tópicos
Wilson	1944	T	Definição de viga híbrida
Haaijer	1961	T	Aços de alta resistência
Frost & Schilling	1964	T e E	Flexão e cortante
Sarsam	1966	T e E	Flexão e cortante
Schilling	1967	T e E	Cargas concentradas
Lew & Toprac	1967	T e E	Flexão e cortante
Schilling	1968	T	Vigas híbridas mistas
Carskaddan	1968	T e E	Instabilidade por cortante
Maeda	1971	T e E	Flexão
Nethercot	1976	T e E	Instabilidade por cortante
Chong	1976	-	Programação
Adeli & Phan	1986	-	Programação
Abuyounest & Adeli	1987	-	Programação
Åhlenius	1994	Geral	Manual de projeto
Axhag	1998	E	Flexão
Barker & Schrage	2000	Economia	Vantagens econômicas
Rush	2001	E	Instabilidade por cortante
Barker *et al*.	2002	T	Instabilidade por cortante
Zentz	2002	E	Instabilidade por cortante
Greco & Earls	2003	T e N	Flexão
Bitar *et al*.	2003	T, E e N	Flexão
Veljkovic & Johansson	2004	Economia, N e E	Dimensionamento
Ito *et al*.	2005	T, E e N	Flexão
Johansson & Collin	2005	Economia	Vantagens econômicas
Barker *et al*.	2005	E	Instabilidade por cortante
Fenkel *et al*.	2007	T, E e N	Flambagem lateral
Barth *et al*.	2007	N	Flexão
Azizinamini	2007	E	Instabilidade por cortante
Projeto COMBRI	2008	T, E e N	Projeto
Petel *et al*.	2008	Economia	Eficiência estrutural
Real *et al*.	2008	E	Instabilidade por cortante
Chacón *et al*.	2009	T, E e N	Cargas concentradas
Chacón *et al*.	2010	T e N	Cargas concentradas
Ajeesh	2011	T	Cisalhamento
Chacón *et al*.	2011	T e N	Cargas concentradas
Gogou	2012	Economia	Alternativas de projeto
Chacón *et al*.	2012	N	Cargas concentradas
Chacón *et al*.	2013	N e E	Cargas concentradas
Rojas Blonval	2013	T e N	Instabilidade por cortante
Chacón	2014	T	Estado da arte
Shokouhian & Shi	2014	T e N	Flexão
Shokouhian & Shi	2015	T e E	Flexão
Chacón & Rojas-Blonval	2015	N	Instabilidade por cortante
Shokouhian *et al.*	2016	T e E	Modos de falha
Wang *et al.*	2016	T, E e N	Flexão
Subramanian & White	2016	T e N	Flexão
Kulkarni & Gupta	2017	T, E e N	Flexão
Ghadami & Broujerdian	2018	T e N	Flexão e cortante
Shokouhian *et al.*	2018	T e E	Flexão e cortante
Lalthazuala & Singh	2019	N	Flexão
Biscaya *et al*.	2019	E, N	Momento, cortante e normal

Filho *et al*. (2022) realizaram estudo experimental e numérico para investigar o comportamento de colunas com seção I soldada com aço S690 falhando por flambagem por flexão em torno do eixo maior e menor inércia e vigas-pilares falhando por flambagem lateral com torção. E concluíram que EN 1993-1-1:2005, EN 1993-1-12:2007, ANSI/AISC 360-16 e AS 4100-2016 exibiram resultados conservadores contra as resistências obtidas na análise experimental.

A Tabela 2.2, adaptada de Chacón (2014), apresenta as pesquisas relacionadas ao uso de aços de alta resistência em vigas, de maneira cronológica, considerando o tipo de pesquisa definido como teórico (T), experimental (E) e numérico (N).

2.3 Vigas de aço de alma cheia submetidas à flexão

2.3.1 Modos de falha

O estudo se baseia em vigas biapoiadas com perfil I fletido em relação ao eixo de maior inércia (eixo x). Como não haverá furos, atuação de forças localizadas ou existência de aberturas para passagem de dutos nas vigas estudadas, elas serão verificadas aos estados-limites últimos relacionados ao momento fletor e à força cortante.

Segundo Fakury *et al*. (2016), quando há a atuação do momento fletor, o colapso pode ocorrer por plastificação total da seção transversal, por flambagem da viga (flambagem lateral com torção) ou por flambagem local dos elementos, parcial ou totalmente comprimidos, da seção transversal.

Ao se estudar as barras, percebe-se que elas apresentam uma curvatura inicial. Essa imperfeição é definida por um deslocamento transversal pequeno em sua seção transversal. À medida que o momento fletor se torna maior em relação ao eixo de maior inércia, esse deslocamento e a torção aumentam, até o momento no qual a barra não resiste mais à solicitação. Nesse ponto é definido o estado-limite último, denominado instabilidade da barra.

Ainda conforme Fakury *et al*. (2016), ao atingir o momento fletor de colapso, a viga pode apresentar translação lateral e torção. Esse colapso caracteriza o estado-limite último chamado flambagem lateral com torção (FLT).

Para impedir que a FLT aconteça, é necessário conter a seção lateralmente. Isso é explicado pelo fato da flambagem ser constituída de dois movimentos que se manifestam em conjunto, a

translação e a rotação, sendo que o impedimento de um dos dois é suficiente para que não ocorra esse tipo de flambagem (Fakury *et al*., 2016).

Outro estado-limite último é a instabilidade local dos elementos. Fakury *et al*. (2016) afirmam que, no caso do perfil I fletido em relação ao eixo de maior inércia (eixo x), avalia-se a possibilidade de ocorrer a flambagem local da mesa comprimida (FLM) e a flambagem local da alma que fica parcialmente comprimida (FLA), como pode ser observado na Figura 2.1:

(a) (b)

Figura 2.1 - Flambagens locais: (a) flambagem local da mesa comprimida e (b) flambagem local da alma (Fakury et al., 2016).

2.3.2 Momento plástico da seção

Na sua forma geral, o momento fletor de plastificação da seção transversal (M_{pl}) também pode ser utilizado quando há diferentes tipos de aços nas mesas e na alma perfil. Dessa maneira, o momento fletor de plastificação para uma seção geometricamente duplamente simétrica é definido a partir da Equação (2.1).

$$M_{pl} = 2t_f b_f \left(\frac{h' + t_f}{2}\right) f_{yf} + ht_w \left(\frac{h'}{4}\right) f_{yw} \tag{2.1}$$

Sendo que: t_w é a espessura da alma, t_f é a espessura da mesa, b_f é a largura da mesa, h' é a altura da alma considerada no plano médio da seção transversal, f_{yf} é a resistência ao escoamento do aço da mesa e f_{yw} é a resistência ao escoamento do aço da alma.

Além disso, foram calculados o deslocamento máximo no meio do vão e a rotação máxima na extremidade das vigas biapoiadas com duas cargas concentradas de mesmo valor, igualmente afastadas dos apoios. Assim, tem-se o deslocamento máximo no meio do vão (Δ_{max}) definido na Equação (2.2) e a rotação máxima na extremidade (θ_{max}) na Equação (2.3).

$$\Delta_{max} = \frac{Pa}{24EI}(3L^2 - 4a^2) \tag{2.2}$$

$$\theta_{max} = \frac{Pa}{2EI}(L - a) \tag{2.3}$$

Nas quais, P é o valor da força em cada ponto de aplicação, a é a distância da carga até a extremidade da viga, L é o comprimento destravado da viga, E é o módulo de Young e I é o momento de inércia em relação ao eixo x.

2.3.3 Capacidade de rotação

A capacidade de rotação pode ser determinada de três maneiras, as quais estão representadas na Figura 2.2.

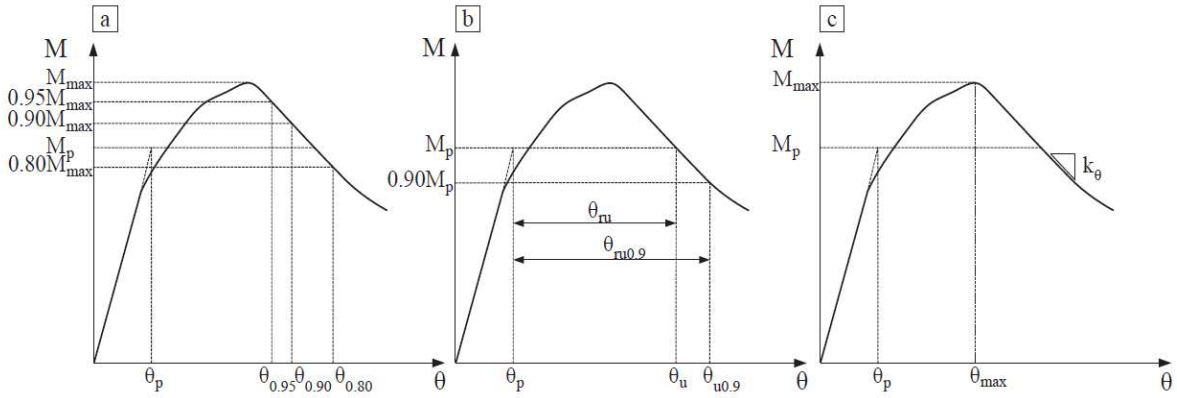


Figura 2.2 – Definição da capacidade de rotação última: (a) relacionada com o momento máximo, (b) relacionada com o momento totalmente plástico e (c) relacionada com a inclinação pós-flambagem (Shokouhian, 2014).

A primeira definição possibilita um resultado mais racional, mas não há um critério para escolher o nível de momento correspondente à situação última. O segundo é bastante útil nas propostas práticas uma vez que é utilizado em programas para análise estrutural. O último é promissor porque é importante conhecer o comportamento do elemento após a flambagem porém, é difícil utilizar esse procedimento visto que geralmente somente uma curva bilinear de momento *versus* rotação pode ser considerada nos programas mais usuais. Dessa maneira, assim como Kulkarni & Gupta (2017), Wang *et al*. (2016) e Shokouhian (2014) adotou-se a definição relacionada ao momento totalmente plástico das vigas a partir das Equações (2.4):

$$\mu_{\theta r} = \frac{\theta_{ru}}{\theta_p} = \frac{\theta_u}{\theta_p} - 1 \tag{2.4a}$$

$$\mu_{\theta r0,9} = \frac{\theta_{ru0,9}}{\theta_p} = \frac{\theta_{u0,9}}{\theta_p} - 1 \tag{2.4b}$$

Sendo que M_p é o momento totalmente plástico, o valor $0{,}9$ corresponde para o caso em que a capacidade de rotação é determinada por $0{,}9M_p$, θ_u é a rotação última, θ_p é a rotação correspondente a primeira rótula plástica, θ_{ru} é a rotação plástica última e $\mu_{\theta r}$ é a capacidade de rotação.

2.3.4 Classificação da seção transversal de acordo com a Eurocode 3

De acordo com Barros *et al*. (2013), em termos simples, a classificação da seção transversal reflete a sua capacidade de resistir e se deformar quando submetida à flexão antes da ocorrência da flambagem local. Sendo que um dos principais parâmetros que afetam essa classificação é a esbeltez das placas que compõem o perfil. Além disso, outros fatores como: condições de apoio, resistência ao escoamento do material e o tipo de forças internas que atuam na seção.

A partir da Seção 5.5 da EN 1993-1-1:2005 é possível definir quatro diferentes classes para as seções transversais, como pode ser observado na Figura 2.3. É importante ressaltar que, para os perfis I, a classe da seção transversal é tomada como a classe menos favorável entre a classe das mesas e da alma.

As seções transversais pertencentes à Classe 1 são capazes de alcançar o momento plástico (M_{pl}) e de desenvolver deformações além do limite elástico, ou seja, podem formar uma rótula plástica com a capacidade de rotação requerida de uma análise plástica sem sofrer redução na resistência. Já a Classe 2 engloba as seções transversais que podem desenvolver o momento plástico, porém possuem uma capacidade de rotação limitada devido à flambagem local, ou seja, exibem uma capacidade de deformação inelástica muito menor quando comparadas às seções da classe anterior. As seções transversais da Classe 3 são esbeltas a tal ponto que apenas a resistência ao escoamento é alcançada, ou seja, são seções nas quais a tensão na fibra de compressão extrema pode atingir a resistência ao escoamento, mas a flambagem local é passível de impedir o desenvolvimento da resistência do momento plástico. Por fim, a Classe 4 refere-se as seções transversais compostas por placas extremamente esbeltas que desenvolvem a

flambagem local antes de alcançar a resistência ao escoamento em uma ou mais partes da seção transversal (EN 1993-1-1:2005).

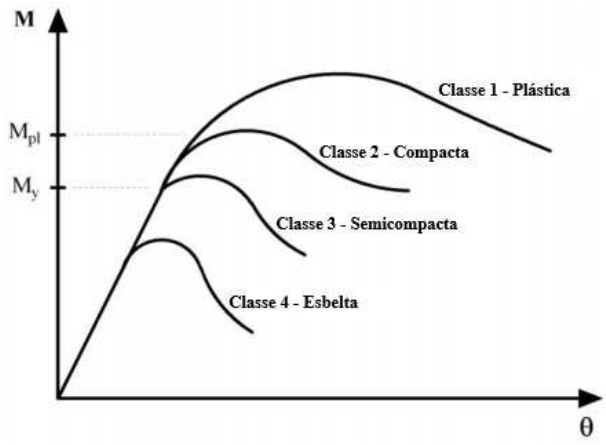


Figura 2.3 - Classificação da seção transversal de acordo com a Eurocode 3 (adaptado de Barros et al., 2013).

2.3.5 Imperfeições geométricas

Ao se estudar as barras, percebe-se que elas apresentam uma curvatura inicial. Essa imperfeição é definida por um deslocamento transversal pequeno em sua seção transversal. Ela afeta o comportamento das estruturas pois, a medida que o momento fletor se torna maior em relação ao eixo de maior inércia, esse deslocamento e a torção aumentam, até o momento no qual a barra não resiste mais à solicitação. Nesse ponto é definido o estado-limite último, denominado instabilidade da barra.

2.3.6 Não linearidade do material

O material presente nas estruturas é muito importante para cálculo dos esforços, uma vez que o entendimento do seu comportamento mecânico nos permite avaliar de modo correto como ele irá se deformar à medida que ocorre o aumento das tensões.

Segundo Pinto (1997), a não linearidade física do material acarreta numa diminuição de seu módulo de elasticidade, por consequência da perda da integridade da seção ou do material. Essa característica não linear está presente no concreto, aço e outros materiais e também influencia na curva momento *versus* curvatura da seção transversal.

Essa não linearidade está relacionada ao fato do material não apresentar uma relação tensão-deformação linear na região plástica, ou seja, o comportamento do material segue a lei de Hooke somente na região elástica. Dessa forma, é importante representar de maneira adequada essa

relação tensão *versus* deformação principalmente na região plástica, pois os efeitos não lineares são descritos por formas mais complexas de equações constitutivas.

2.3.7 Tensões residuais

Trahair *et al.* (2008) definem que as tensões residuais são tensões causadas por deformação plástica não uniforme ou por resfriamento desigual após laminação, corte com chama ou soldagem em um elemento sem carga, sendo a análise dessas tensões importante visto que, a presença delas pode causar uma redução significativa na resistência à flambagem.

2.3.8 Algumas especificidades das vigas híbridas

2.3.8.1 Distribuição de tensões

A distribuição de tensões de vigas híbridas submetidas à flexão está representada na Figura 2.4. De acordo com Frost & Schilling (1964), a tensão é linearmente distribuída ao longo da altura da seção na fase inicial do carregamento, assim como nas vigas homogêneas. Entretanto, na seção híbrida, o início do escoamento ocorre na fibra mais afastada da alma, longe da linha neutra, enquanto que nas seções homogêneas acontece na fibra mais externa de toda a seção transversal. Com o aumento da carga, como na viga híbrida a alma possui um aço de menor resistência, o campo de escoamento se expande da alma para a mesa. Já na viga homogênea o campo de escoamento parte da fibra mais externa da seção transversal para a linha neutra da seção gradativamente (Kulkarni & Gupta, 2017) (Wang *et al*., 2016)

Ainda, de acordo com Beg *et al*. (2010) para que seja possível empregar um aço mais resistente nas mesas do perfil I, algumas considerações são necessárias:

- Para utilizar toda a resistência das mesas, uma redistribuição parcial plástica das tensões é permitida na alma, como pode ser visto na Figura 2.5.

- Para garantir alguns requisitos mínimos de ductibilidade da alma, a área efetiva desta deve ser determinada de acordo com a resistência ao escoamento da mesa e não da alma.

- Para limitar a distribuição parcial elástica de tensões na alma é recomendado que a razão das resistências ao escoamento entre a mesa do perfil e a alma não supere o limite, o qual é igual a dois, como pode ser observado na Equação (2.5)

$$\phi_h = \frac{f_{yf}}{f_{yw}} \leq 2{,}0 \tag{2.5}$$

Na qual: f_{yf} é a resistência ao escoamento do aço da mesa e f_{yw} é a resistência ao escoamento do aço da alma.

Além disso, ao calcular a localização do centro de gravidade da área efetiva (G'), a limitação da tensão na alma $\sigma_w \leq f_{yw}$ é desconsiderada e assume-se uma distribuição linear de tensão na alma do perfil.

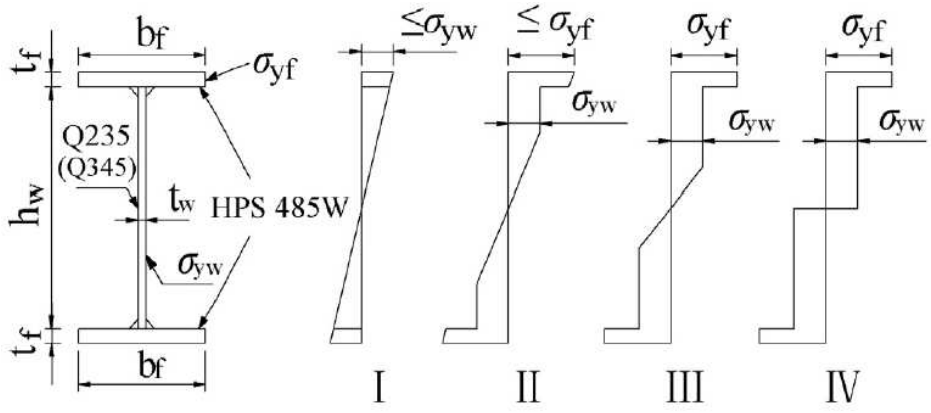


Figura 2.4 - Distribuição de tensões nas vigas híbridas (Wang et al., 2016)

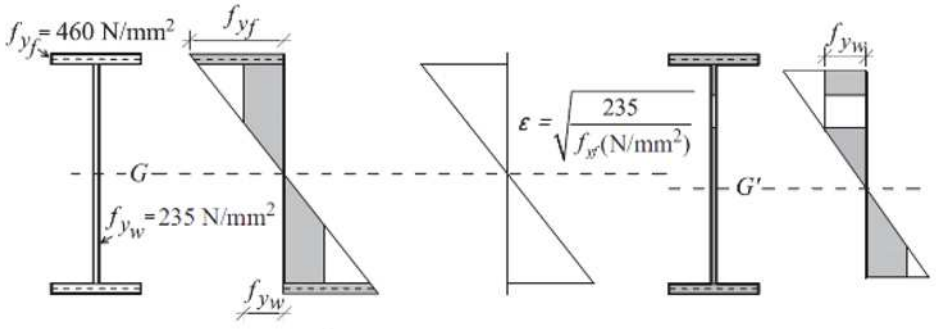


Figura 2.5 - Distribuição de tensões assumidas em uma viga híbrida com $\phi_h = 1{,}96$ (Beg *et al*., 2010)

2.3.8.2 Fabricação

Segundo Veljkovic & Johansson (2004), o conceito de vigas híbridas pressupõe o uso do aço de alta resistência. Esse tipo de aço possui um procedimento de fabricação um pouco diferente do que o aço convencional. Na prática, a soldagem de diferentes tipos de aço não representa um problema, mas é necessário definir os eletrodos correspondentes. As regras presentes em EN

1993-1-1:2005 requerem eletrodos correspondentes e essa exigência pode ser utilizada até S690. Para a solda entre a alma e a mesa sugere-se que sejam utilizados eletrodos correspondentes a resistência da alma. Por fim, a solda deve ser do mesmo material da alma do perfil.

3 VERIFICAÇÃO DE SEÇÕES ESBELTAS ENRIJECIDAS LONGITUDINALMENTE QUANTO A TENSÕES NORMAIS E CISALHANTES

Neste capítulo são descritos de maneira sucinta os procedimentos de verificação de seções transversais esbeltas a tensões normais e cisalhantes de acordo com a norma europeia EN 1993-1-5:2006. É importante ressaltar que, essa verificação, de acordo com a norma, pode ser realizada pelo Método da Largura Efetiva (MLE) ou pelo Método da Tensão Reduzida (MTR). Porém, este trabalho aborda somente o primeiro método.

O MLE realiza uma verificação para cada tipo de esforço e em seguida uma verificação para a combinação desses tipos de esforços a partir de uma equação de interação.

3.1 Resistência a tensões normais

Para determinar a resistência de seções transversais da Classe 4 sujeitas a tensões normais utilizando o MLE, as larguras efetivas de cada elemento comprimido são determinadas de maneira independente. Tendo como base essas larguras efetivas, é possível determinar a área efetiva da seção transversal (A_{eff}), o momento de inércia de área (I_{eff}) e o módulo elástico efetivo (W_{eff}).

A resistência última de placas pequenas depende de dois tipos de flambagem, flambagem global *Tipo Placa* e flambagem global *Tipo Pilar*, e por último, uma interpolação entre os dois tipos de comportamento.

Portanto, em termos genéricos, para a determinação do esforço resistente a tensões normais para seções esbeltas deve-se seguir as seguintes etapas: cálculo da seção efetiva para a flambagem local dos sub-painéis, cálculo da seção efetiva para a flambagem global da alma reforçada, cálculo dos fatores de redução para cada sub-painel, cálculo do fator de redução para a flambagem global do *Tipo Placa*, cálculo do fator de redução para a flambagem global do *Tipo Pilar*, cálculo do fator de redução global e, finalmente, cálculo do momento fletor resistente.

3.1.1 Flambagem global *Tipo Placa*

- Placas não enrijecidas longitudinalmente

A largura efetiva (b_{eff}) é determinada pela Equação (3.1):

$$b_{eff} = \rho_{loc} b \tag{3.1}$$

Sendo que: b é a largura do elemento e ρ_{loc} é um fator de redução para flambagem de placa e de acordo com a EN 1993-1-5:2006 deve ser calculado de maneiras distintas se o elemento comprimido é interno (AA) ou externo (AL).

- Elementos AA - Equações (3.2)

$$\rho_{loc} = 1{,}0 \quad \text{para} \quad \bar{\lambda}_p \leq 0{,}5 + \sqrt{0{,}085 - 0{,}055\psi} \tag{3.2a}$$

$$\rho_{loc} = \frac{\bar{\lambda}_p - 0{,}0055(3+\psi)}{\bar{\lambda}_p{}^2} \leq 1{,}0 \quad \text{para} \quad \bar{\lambda}_p > 0{,}5 + \sqrt{0{,}085 - 0{,}055\psi} \tag{3.2b}$$

- Elementos AL - Equações (3.3):

$$\rho_{loc} = 1{,}0 \quad \text{para} \quad \bar{\lambda}_p \leq 0{,}748 \tag{3.3a}$$

$$\rho_{loc} = \frac{\bar{\lambda}_p - 0{,}188}{\bar{\lambda}_p{}^2} \leq 1{,}0 \quad \text{para} \quad \bar{\lambda}_p > 0{,}748 \tag{3.3b}$$

Nas quais: ψ é a razão de tensões nas duas bordas da placa, com a máxima tensão de compressão no denominador e $\bar{\lambda}_p$ é a esbeltez da placa determinada pela Equação (3.4):

$$\bar{\lambda}_p = \sqrt{\frac{f_y}{\sigma_{cr,p}}} \tag{3.4}$$

Sendo: $\sigma_{cr,p}$ é a tensão crítica de flambagem elástica da placa e encontrada através da Equação (3.5):

$$\sigma_{cr,p} = k_\sigma \frac{\pi^2 E}{12(1-\upsilon^2)} \left(\frac{t}{b}\right)^2 \tag{3.5}$$

Na qual, t é a espessura do elemento, E é o módulo de elasticidade do aço, υ é o coeficiente de Poisson e k_σ é o coeficiente de flambagem.

Para placas com razão de aspecto, $\alpha = a/b \geq 1{,}0$, o coeficiente de flambagem, k_σ, é dado pela Tabela 4.1 e Tabela 4.2 da EN 1993-1-5:2006. Já para placas com razão de aspecto, $\alpha \leq 1{,}0$ e submetida à compressão uniforme, o coeficiente de flambagem, k_σ, é dado pela Equação (3.6):

$$k_\sigma = \left(\alpha + \frac{1}{\alpha}\right)^2 \tag{3.6}$$

Em todos os casos, os elementos de placa são considerados como simplesmente apoiados ao longo das bordas.

E finalmente, a área efetiva (A_{eff}) pode ser expressa pela Equação (3.7):

$$A_{c,eff} = \rho_{loc} A_c \tag{3.7}$$

- Placas enrijecidas longitudinalmente

Quando os sub-painéis de largura b_i são esbeltos e submetidos à flambagem local, a interação de flambagem local e global deve ser considerada. Esta interação é obtida por meio da modificação da esbeltez relativa da placa equivalente apresentada na Equação (3.8):

$$\bar{\lambda}_p = \sqrt{\frac{N_y}{N_{cr}}} = \sqrt{\frac{A_{c,eff,loc}\, f_y}{A_c \sigma_{cr,p}}} = \sqrt{\frac{\beta_{A,c}\, f_y}{\sigma_{cr,p}}} \tag{3.8}$$

Na qual: $\sigma_{cr,p}$ é a tensão crítica de flambagem elástica da placa enrijecida e de $\beta_{A,c}$ é dado pela Equação (3.9):

$$\beta_{A,c} = \frac{A_{c,eff,loc}}{A_c} \tag{3.9}$$

Sendo que: $A_{c,eff,loc}$ é a soma das áreas efetivas dos sub-painéis e enrijecedores total ou parcialmente comprimidos, exceto as partes efetivas de bordas com larguras $b_{edge,eff}$, em que são suportadas por um outro elemento de placa adjacente e A_c é a área bruta da parte comprimida da placa enrijecida, exceto as partes efetivas de bordas com larguras b_{edge}, em que são suportadas por um outro elemento de placa adjacente.

A área efetiva, $A_{c,eff,loc}$, é dada pela Equação (3.10):

$$A_{c,eff,loc} = A_{sl,eff} + \sum_i \rho_{loc,i}\, b_{loc,i} t \tag{3.10}$$

Na qual: $A_{sl,eff}$ é a soma das áreas efetivas dos enrijecedores longitudinais, $\rho_{loc,i}$ é o fator de redução de cada sub-painel i e $b_{loc,i}$ é a largura da parte comprimida de cada sub-painel i individualmente.

Beg *et al.* (2010) apresentam um método analítico para determinar essa tensão crítica de flambagem elástica um ou dois enrijecedores na zona comprimida de acordo com a Equações (3.11).

$$\sigma_{cr,p} = \frac{b_c}{b_{c2}}\sigma_{cr,sl} \tag{3.11a}$$

Sendo:

$$\sigma_{cr,sl} = \frac{1{,}05E}{A_{sl,1}}\frac{\sqrt{I_{sl,1}t^3b}}{b_1b_2} \quad \text{se} \quad \alpha \geq \alpha_c \tag{3.11b}$$

$$\sigma_{cr,sl} = \frac{\pi^2 E I_{sl,1}}{A_{sl,1}a^2} + \frac{Et^3ba^2}{4\pi^2(1-v^2)A_{sl,1}b_1^2b_2^2} \quad \text{se} \quad \alpha \leq \alpha_c \tag{3.11c}$$

$$\alpha_c = 4{,}33\sqrt[4]{\frac{I_{sl,1}b_1^2b_2^2}{t^3b}} \tag{3.11d}$$

Nas quais: $A_{sl,1}$ é a área bruta do enrijecedor e $I_{sl,1}$ é o momento de inércia de área do enrijecedor.

No caso de dois enrijecedores – Figura A.3 da EN 1993-1-5:2006 – o procedimento com as Equações (3.11) deve ser realizado três vezes: para cada enrijecedor considerando que o outro enrijecedor é um suporte rígido a alma e para um fictício enrijecedor que leva em consideração a junção dos dois enrijecedores iniciais e é calculado pela Equação (3.12).

$$b_1^{I+II} = \frac{A_{sl,2}b_{c,2}}{A_{sl,1}b_{c,1} + A_{sl,2}b_{c,2}}b_2^I + b_1^I \tag{3.12}$$

3.1.2 Flambagem global *Tipo Pilar*

- Placas não enrijecidas longitudinalmente

A esbeltez ($\bar{\lambda}_c$) é determinada pela Equação (3.13):

$$\bar{\lambda}_c = \sqrt{\frac{f_y}{\sigma_{cr,c}}} \tag{3.13}$$

Sendo: $\sigma_{cr,c}$ é a tensão crítica de flambagem elástica para comportamento *Tipo Pilar* e pode ser encontrada através da Equação (3.14):

$$\sigma_{cr,c} = \frac{\pi^2 E t^2}{12(1-\upsilon^2)a^2} \tag{3.14}$$

Na qual: a é o comprimento de flambagem normalmente igual a distância entre rígidos enrijecedores transversais.

Pela seleção da curva a ($\alpha = 0{,}21$) do item 6.3.1.2 da EN 1993-1-1:2005, o fator de redução para flambagem (χ_c) é calculado pelas Equações (3.15):

$$\chi_c = \frac{1}{\phi + \sqrt{\phi^2 - \bar{\lambda}_c{}^2}} \tag{3.15a}$$

$$\phi = 0{,}5\left[1 + \alpha\left(\bar{\lambda}_c - 0{,}2\right) + \bar{\lambda}_c{}^2\right] \tag{3.15b}$$

- Placas enrijecidas longitudinalmente

A tensão crítica ($\sigma_{cr,c}$) é determinada como a tensão de flambagem ($\sigma_{cr,sl}$) de um único enrijecedor próximo a borda do painel possuindo a maior tensão de compressão e pode ser obtida pela Equação (3.16):

$$\sigma_{cr,sl} = \frac{\pi^2 E I_{sl,1}}{A_{sl,1} a^2} \tag{3.16}$$

Na qual: $I_{sl,1}$ e $A_{sl,1}$ são o segundo momento de área e a área bruta, respectivamente. Esses dois parâmetros consideram a área igual a área bruta do enrijecedor e das partes adjacentes da placa de acordo com Figura 4.4, Figura A.1 e Figura A.2 da EN 1993-1-5:2006.

Para compressão uniforme $\sigma_{cr,sl} = \sigma_{cr,c}$. Para outras razões de tensões (ψ), $\sigma_{cr,c}$ é calculado pela Equação (3.11a).

A esbeltez é dada pela Equação (3.17):

$$\bar{\lambda}_c = \sqrt{\frac{\beta_{A,c}\, f_y}{\sigma_{cr,c}}} \tag{3.17}$$

Sendo que $\beta_{A,c}$ é dado pela Equação (3.18):

$$\beta_{A,c} = \frac{A_{sl,1,eff}}{A_{sl,1}} \tag{3.18}$$

Na qual: $A_{sl,1,eff}$ é a área efetiva da seção transversal do enrijecedor e das as partes adjacentes da placa levando em conta a flambagem dos sub-painéis de acordo com Figura 4.4 e Figura A.1 da EN 1993-1-5:2006.

A partir do item 6.3.1.2 da EN 1993-1-1:2005 pode-se obter o fator de redução para flambagem (χ_c). E, para placas enrijecidas o parâmetro de imperfeição α deve ser determinado a partir da Equação (3.19).

$$\alpha_e = \alpha + \frac{0{,}09}{i/e} \tag{3.19}$$

Sendo que: $\alpha = 0{,}34$ para enrijecedores com seção fechada e $\alpha = 0{,}49$ para enrijecedores com seção aberta, $e = \max(e_1, e_2)$ de maneira que e_1 é a distância entre os centros de gravidade do enrijecedor isolado e do enrijecedor com a contribuição da placa e e_2 é a distância entre os centros de gravidade da placa isolada e do enrijecedor com a contribuição da placa e i é o raio de giração do enrijecedor calculado pela Equação (3.20):

$$i = \sqrt{\frac{I_{sl,1}}{A_{sl,1}}} \tag{3.20}$$

Por fim, a partir das Equações (3.15) é possível determinar o fator de redução (χ_c).

3.1.3 Interpolação entre os comportamentos *Tipo Placa* e *Tipo Pilar*

Para placas curtas, o comportamento *Tipo Pilar* se torna importante e a interação entre os dois tipos de comportamento deve ser considerada. A fórmula de interpolação para obter o fator de redução final (ρ_c) é dada pelas Equações (3.21):

$$\rho_c = (\rho - \chi_c)\, \xi\, (2 - \xi) + \chi_c \tag{3.21a}$$

$$\xi = \frac{\sigma_{cr,p}}{\sigma_{cr,c}} - 1 \quad \text{sendo } 0 \leq \xi \leq 1 \tag{3.21b}$$

Nas quais: ρ é o fator de redução da flambagem *Tipo Placa*, χ_c é o fator de redução da flambagem *Tipo Pilar*, $\sigma_{cr,p}$ é a tensão crítica flambagem elástica *Tipo Placa* e $\sigma_{cr,c}$ é a tensão crítica de flambagem elástica *Tipo Pilar*.

Assim, a Equação (3.22) e a Equação (3.23) representam a área efetiva final da zona comprimida ($A_{c,eff}$) de placas não enrijecidas e placas enrijecidas, respectivamente.

$$A_{c,eff} = \rho_c \, b_{eff} \tag{3.22}$$

$$A_{c,eff} = \rho_c \, A_{c,eff,loc} + \sum_i b_{i,edge,eff} \, t \tag{3.23}$$

Onde: $A_{c,eff,loc}$ é a área efetiva, dada pela Equação (3.10) e $b_{i,edge,eff}$ é a largura efetiva de extremidade dada pela Figura 4.4 da EN 1993-1-5:2006.

3.1.4 Cálculo do momento fletor resistente

A área efetiva da seção é dada pela Equação (3.24):

$$A_{eff} = A_{c,eff} + b_t \, t_w + 2 b_f \, t_f \tag{3.24}$$

Também se determina o módulo de flexão efetivo para a seção $W_{el,eff,x}$ e, posteriormente, o momento fletor a partir da Equação (3.25).

$$M_{Rd,x} = W_{el,eff,x} \, \frac{f_y}{\gamma_{M0}} \tag{3.25}$$

3.2 Resistência ao cisalhamento

Deve ser verificada quando a razão h_w/t_w for maior que os valores obtidos a partir da Equação (3.26) e Equação (3.27).

- Almas não enrijecidas longitudinalmente

$$\frac{h_w}{t_w} > \frac{72}{\eta} \varepsilon \tag{3.26}$$

- Almas enrijecidas longitudinalmente

$$\frac{h_w}{t_w} > \frac{31}{\eta}\varepsilon\sqrt{k_\tau} \tag{3.27}$$

Sendo que: ε é o fator de escoamento, k_τ é o coeficiente de flambagem por cisalhamento calculado pelo item A.3 do Anexo A da EN 1993-1-5:2006 η é o coeficiente que considera o acréscimo da capacidade resistente com a menor esbeltez da alma de forma que $\eta = 1{,}2$ para $f_y \leq 460\ MPa$ e $\eta = 1{,}0$ para $f_y > 460\ MPa$.

A capacidade resistente total de cálculo à força cortante para almas enrijecidas ou não enrijecidas é dada pela Equação (3.28).

$$V_{b,Rd} = V_{bw,Rd} + V_{bf,Rd} \leq h_w\, t_w \frac{\eta\, f_{yw}}{\sqrt{3}\,\gamma_{M1}} \tag{3.28}$$

Em que: $V_{bw,Rd}$ é a capacidade resistente da alma, $V_{bf,Rd}$ é a capacidade resistente das mesas e γ_{M1} é o fator parcial de segurança para capacidade resistente relacionada à instabilidade.

3.2.1 Contribuição da alma

A capacidade resistente da alma é dada pela Equação (3.29):

$$V_{bw,Rd} = \chi_w\, h_w\, t_w \frac{f_{yw}}{\sqrt{3}\gamma_{M1}} \tag{3.29}$$

Na qual: χ_w é o fator de redução para a capacidade resistente à flambagem por cisalhamento da alma e é obtido pelas Equações (3.30) e Equações (3.31).

- Comprimento adicional não rígido

$$\chi_w = \eta \quad para \quad \bar{\lambda}_w < \frac{0{,}83}{\eta} \tag{3.30a}$$

$$\chi_w = \frac{0{,}83}{\bar{\lambda}_w} \quad para \quad \bar{\lambda}_w \geq \frac{0{,}83}{\eta} \tag{3.30b}$$

- Comprimento adicional rígido

$$\chi_w = \eta \quad para \quad \bar{\lambda}_w < \frac{0{,}83}{\eta} \tag{3.31a}$$

$$\chi_w = \frac{0{,}83}{\bar{\lambda}_w} \quad para \quad \frac{0{,}83}{\eta} \leq \bar{\lambda}_w < 1{,}08 \tag{3.31b}$$

$$\chi_w = \frac{1{,}37}{0{,}7 + \bar{\lambda}_w} \quad para \quad \bar{\lambda}_w \geq 1{,}08 \tag{3.31b}$$

Sendo que: $\bar{\lambda}_w$ é a esbeltez normalizada da alma e é determinada através da Equação (3.32) ou Equação (3.33).

- Para enrijecedores transversais somente nos apoios

$$\bar{\lambda}_w = \frac{h_w}{86{,}4\ t_w\ \varepsilon} \tag{3.32}$$

- Para enrijecedores transversais nos apoios, enrijecedores transversais intermediários e/ou enrijecedores longitudinais

$$\bar{\lambda}_w = \max\left(\frac{h_w}{37{,}4\ t_w\ \varepsilon\ \sqrt{k_\tau}}; \frac{h_{wi}}{37{,}4\ t_w\ \varepsilon\ \sqrt{k_{\tau i}}}\right) \tag{3.33}$$

Em que: k_τ é o coeficiente de flambagem por cisalhamento da alma entre as mesas, $k_{\tau i}$ é o coeficiente de flambagem por cisalhamento de sub-painéis *i* e h_{wi} é a altura livre de sub-painéis *i*.

3.3 Interação entre momento fletor e força cortante

As Equações (3.34) representam a interação na alma de viga com seção do tipo I e essa interação deve ser considerada se o momento fletor não puder ser resistido pelas mesas sozinhas ($M_{Ed} > M_{f,Rd}$) ou se a força cortante não puder ser resistida pela alma sozinha ($\bar{\eta}_3 = V_{Ed}/V_{bw,Rd} > 0{,}5$).

$$\bar{\eta}_1 + \left[1 - \frac{M_{f,Rd}}{M_{pl,Rd}}\right][2\bar{\eta}_3 - 1]^2 \leq 1{,}0 \tag{3.34a}$$

$$\bar{\eta}_1 = \frac{M_{Ed}}{M_{pl,Rd}} \tag{3.34a}$$

Nas quais: $M_{f,Rd}$ é o momento fletor resistente de plastificação de cálculo da seção composta pela área efetiva apenas das mesas e $M_{pl,Rd}$ é o momento fletor resistente de plastificação de

cálculo da seção composta pela área efetiva das mesas e pela área total da alma, independentemente da classe da seção.

4 DESENVOLVIMENTO E VALIDAÇÃO DOS MODELOS NUMÉRICOS

4.1 Considerações iniciais

Este capítulo apresenta o comportamento de vigas de aço com seção transversal do tipo I submetidas à flexão. Para isso, utilizou-se o programa ABAQUS (SIMULIA, 2014) para a elaboração de três modelos numéricos fundamentados no Método dos Elementos Finitos (MEF). O primeiro modelo se refere às vigas compactas. Já o segundo e o terceiro relacionam-se às vigas esbeltas com um e dois enrijecedores longitudinais, respectivamente. Importante ressaltar que todos os modelos são contidos lateralmente.

No *programa,* adotou-se o mesmo procedimento de análise para os modelos propostos. Primeiramente, realizou-se a análise de flambagem elástica para obter as cargas críticas de flambagem e uma aproximação dos modos de flambagem das vigas. Em seguida, foi realizada uma análise da capacidade resistente última de acordo com a aproximação do primeiro modo de flambagem. Sendo que, nesta última análise, considerou-se a introdução das tensões residuais e das imperfeições geométricas iniciais no modelo compacto, enquanto nos modelos esbeltos foram consideradas somente as imperfeições geométricas iniciais, fato que será explicado posteriormente.

Dessa maneira, mais especificamente, este capítulo busca detalhar o desenvolvimento dos modelos numéricos, validar os modelos numéricos e experimentais obtidos por Shokouhian (2014) e Sinur (2011) e, finalmente, propor modelos numéricos para a realização dos estudos de variação de parâmetros. Importante ressaltar que são três modelos numéricos com as mesmas características de aplicação de carga e condições de contorno, para que seja possível alcançar um modelo numérico geral.

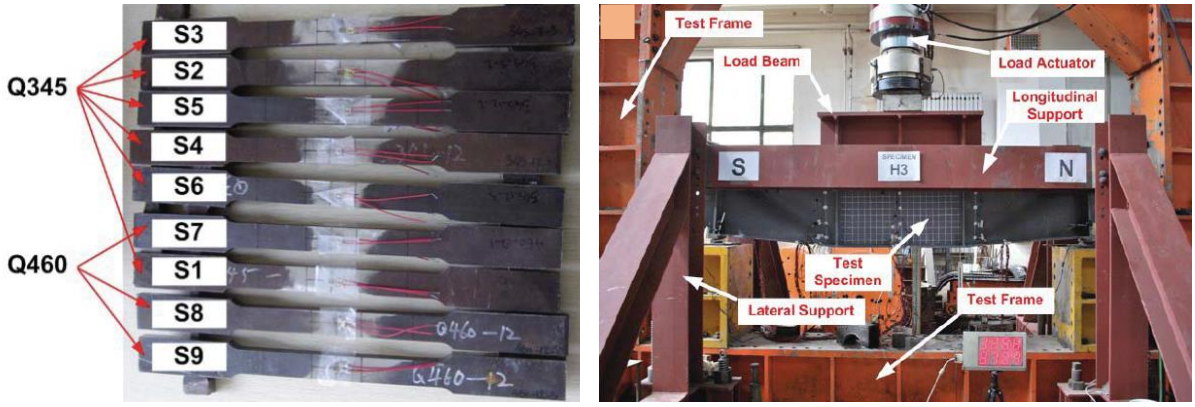
4.2 Modelo compacto

4.2.1 Shokouhian (2014)

Shokouhian (2014) estudou a ductilidade e a capacidade resistente última de vigas fletidas. Além da análise teórica, o autor abordou ensaios experimentais e numéricos dessas vigas com a utilização do programa de elementos finitos ANSYS 14.5.

4.2.1.1 Descrição da análise experimental e resultados obtidos por Shokouhian (2014)

Nesta análise, as seis vigas foram fabricadas, instrumentadas e submetidas ao carregamento para os ensaios de flexão de quatro pontos. Essas vigas foram fabricadas com chapas com resistência ao escoamento nominal igual a 355 N/mm² e 460 N/mm², sendo três convencionais (C1, C2 e C3) e três híbridas (H1, H2 e H3). Com relação ao carregamento, uma viga de 1300 mm foi utilizada entre o aplicador de carga e a viga biapoiada analisada, de maneira que a viga de carregamento forneça duas forças concentradas iguais na mesa superior da viga ensaiada. Também se restringiu lateralmente a viga ao longo do seu comprimento para evitar a flambagem lateral por torção (FLT) e suportes laterais foram adicionados ao final de cada viga para restringir os deslocamentos laterais. As amostras para o ensaio de tração e a configuração do ensaio viga estão apresentadas na Figura 4.1.



(a) (b)

Figura 4.1 – Análise experimental: (a) amostras preparadas para o ensaio de tração e (b) vista frontal da configuração dos ensaios (Shokouhian, 2014).

Ao final, percebeu-se que a falha nas vigas C1, C2 e H1 ocorreu devido à flambagem local da mesa (FLM), em contrapartida na viga H3 aconteceu a falha pela flambagem local da alma (FLA). Por outro lado, nas vigas C3 e H2 há uma interação entre FLA e FLM, sendo que na viga C3 o modo de falha teve início na alma enquanto que na viga H2 o modo de falha ocorreu, primeiramente, na mesa. Além disso, a viga C1 mostrou a maior capacidade de rotação e um bom nível de ductilidade, quando comparada com as outras vigas, sendo que o menor nível de ductilidade foi observado na viga C2. Todavia, para as outras vigas não foi possível determinar as respectivas ductilidades.

É importante ressaltar que, nesse estudo experimental foram gerados dados referentes à geometria, às especificações dos materiais e às imperfeições que serão utilizadas nos estudos numérico e teórico do autor, de tal modo que, as propriedades mecânicas dos materiais foram

definidas como a média entre os valores obtidos a partir do ensaio de tração de três chapas de cada amostra. Além disso, as larguras e espessuras das mesas e as alturas das almas foram determinadas como o valor médio entre valores obtidos em cinco seções transversais de cada chapa. Já as espessuras das almas foram definidas como a média entre duas seções transversais, uma em cada extremidade da viga. As imperfeições locais das mesas e da alma de cada viga foram medidas com o auxílio de transdutores de deslocamento linear baseados na tecnologia LVDT e consideradas como o maior valor entre os valores obtidos também em cinco seções transversais de cada viga em análise, sendo duas seções nas extremidades, duas nos terços do vão e uma na seção central da viga. A imperfeição global foi determinada como o maior desvio, o qual foi possível obter determinando-se a amplitude da curvatura inicial da viga em cinco seções transversais, entre uma linha reta e as duas extremidades das vigas.

4.2.1.2 Descrição da análise numérica desenvolvida por Shokouhian (2014)

Conforme mencionado anteriormente, para essa análise, os modelos de elementos finitos das vigas foram elaborados a partir dos dados da geometria e especificações dos materiais adquiridos no estudo experimental. A partir desses dados, adotou-se a relação constitutiva dos materiais, conforme apresentado na Figura 4.2, de maneira que: f_y é a resistência ao escoamento do aço, f_u é a resistência à ruptura do aço à tração, ε_y é a deformação normal específica correspondente à tensão limite da região elástica, ε_{st} é a deformação normal específica correspondente à tensão limite da região de plastificação e ε_u é a deformação de ruptura.

Para simular as condições de contorno, as três translações e as duas rotações fora do plano foram restringidas na linha determinada pela junção entre mesa inferior e alma do perfil de uma das extremidades. Na outra extremidade, apenas duas translações foram impedidas de maneira que a translação na direção do comprimento da viga não foi restringida. Em termos de rotação, apenas a rotação no plano foi liberada. Além disso, os modelos numéricos também foram restringidos nas mesas superior e inferior quanto à translação fora do plano da alma. O carregamento foi representado por forças distribuídas nas mesas superiores, conforme Figura 4.3.

Com relação às imperfeições, o autor associou a imperfeição geométrica inicial referente ao primeiro modo de flambagem da viga para realizar a análise da capacidade resistente última da mesma. Ao se tratar da malha de elementos finitos, o autor limita-se a mencionar que adotou

uma malha densa, considerando o custo computacional, de forma a representar de maneira mais fiel possível o modelo.

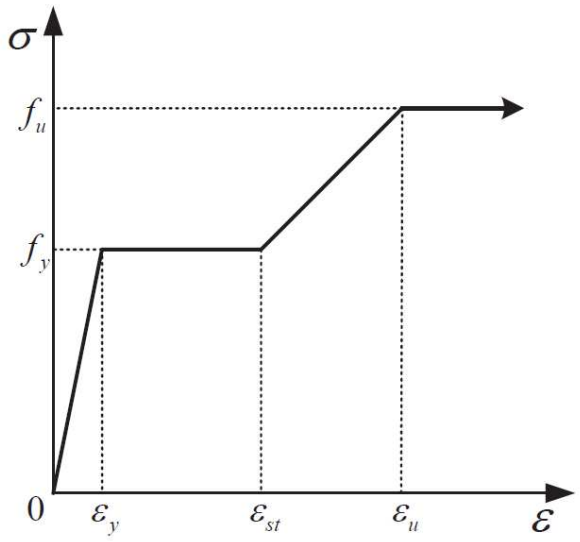


Figura 4.2 – Diagrama tensão *x* deformação (Shokouhian, 2014).

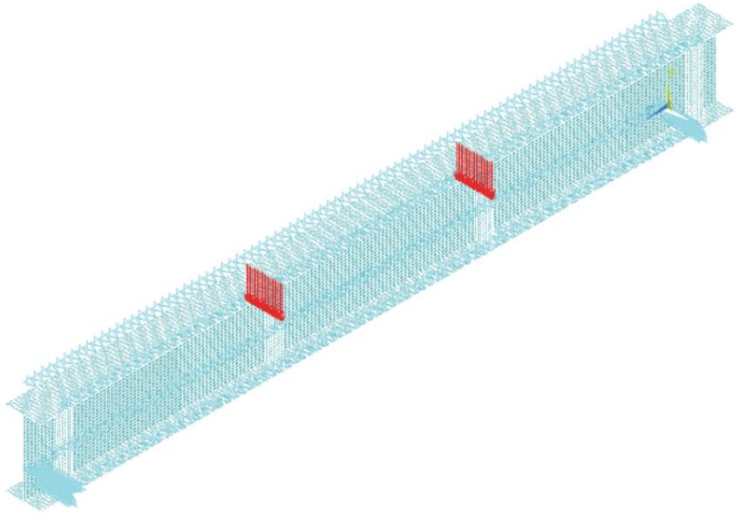


Figura 4.3 – Condições de contorno e aplicação de forças (Shokouhian, 2014).

O autor adotou a distribuição de tensões residuais proposta por Ban *et al*. (2013). Os estudos experimentais desse autor foram realizados em vigas convencionais e consideraram uma distribuição de tensões residuais para as mesas e alma. Embora o ideal fosse adotar uma distribuição para vigas convencionais e outra para vigas híbridas, devido à falta de dados sobre a distribuição de tensões residuais neste segundo caso, adotou-se a mesma distribuição das vigas convencionais. Entretanto, baseado nos estudos que o autor conduziu, concluiu-se que a influência das tensões residuais no comportamento à flexão das vigas não é muito significativo.

Além disso, todos os modelos numéricos mostraram uma boa concordância com os resultados experimentais em relação à capacidade resistente última. Com relação à capacidade de rotação, os modelos das vigas C1 e C2 obtiveram bons resultados. As vigas C3 e H1 alcançaram o momento plástico na análise numérica, o que não aconteceu no experimento de C3, visto que nas análises numéricas há a consideração de $0{,}9M_P$ em vez de M_P para determinar a capacidade de rotação. Os modelos numérico e experimental de H2 não alcançaram o momento plástico mesmo com a consideração de $0{,}9M_P$. Finalmente, o modelo numérico de H3 alcançou uma capacidade de rotação igual a 0,4, o que mostra um baixo nível de ductilidade, entretanto o seu modelo experimental não atingiu o momento plástico.

4.2.1.3 Análise teórica desenvolvida por Shokouhian (2014)

Nessa etapa, o autor determinou o momento fletor de plastificação da seção transversal (M_{pl}) das seis vigas - três convencionais (C1, C2 e C3) e suas híbridas correspondentes (H1, H2 e H3) - a partir das dimensões reais e propriedades mecânicas dos materiais obtidas através do ensaio de tração. Além disso, foram calculados os deslocamentos máximos e a rotação máxima na extremidade das vigas para o caso de carregamento analisado – viga biapoiada com duas cargas concentradas.

4.2.2 Descrição dos procedimentos de modelagem do presente estudo

4.2.2.1 Geometria

No presente estudo, seis vigas biapoiadas de seção transversal do tipo I submetidas a duas forças localizadas foram analisadas, sendo três convencionais (C1, C2 e C3) e três híbridas (H1, H2 e H3). As forças se localizam nos terços do comprimento da viga, o que produz um momento constante na região central da mesma.

A distância entre os apoios (L) que garantem a condição de viga biapoiada é de 3 m, sendo que há um comprimento adicional correspondente a $L/15$ em ambos os lados, modelado com as mesmas dimensões da viga. São considerados quatro enrijecedores verticais de cada lado da alma, sendo que dois deles estão localizados nas reações de apoio e os outros dois nos pontos de aplicação das forças concentradas, com objetivo de impedir a ocorrência da instabilidade devido à compressão localizada na alma, conforme pode ser observado na Figura 4.4.

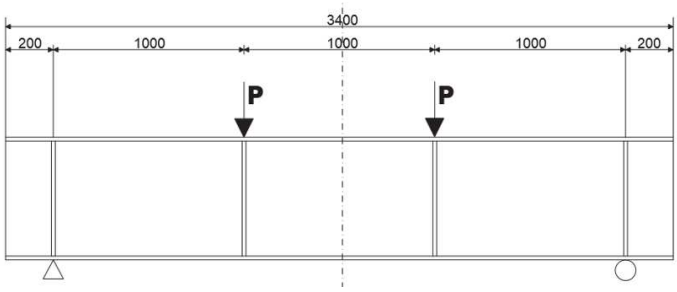


Figura 4.4 - Especificação geométrica do modelo numérico (Shokouhian, 2014).

As dimensões foram consideradas no plano médio da seção transversal das seis vigas e as resistências ao escoamento das mesas e da alma foram adotadas de acordo com a análise experimental do autor. Elas estão apresentadas nas Figura 4.5 e Tabela 4.1.

Sendo b_{fb} a largura da mesa inferior; b_{fu} a largura da mesa superior; h a altura da alma; h' a altura da alma considerando o plano médio das mesas; t_{fb} a espessura da mesa inferior; t_{fu} a espessura da mesa superior; t_w a espessura da alma; f_{yf} a resistência ao escoamento do aço das mesas e f_{yw} a resistência ao escoamento do aço da alma.

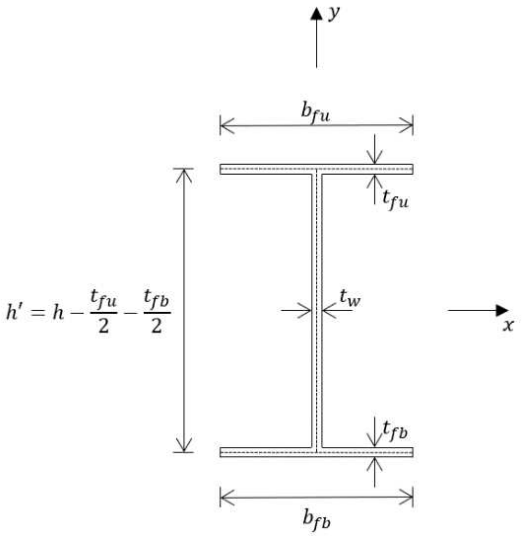


Figura 4.5 - Seção transversal considerada para as análises do modelo compacto.

Tabela 4.1 - Dimensões e resistências ao escoamento das mesas e alma das seções transversais.

Modelo	h' **(mm)**	t_w **(mm)**	b_{fu} **(mm)**	t_{fu} **(mm)**	b_{fb} **(mm)**	t_{fb} **(mm)**	f_{yf} **(N/mm²)**	f_{yw} **(N/mm²)**
C1	369,30	7,87	169,00	11,91	170,20	11,89	408,20	442,80
C2	370,34	7,82	263,60	11,85	263,80	11,87	408,20	442,80
C3	610,34	7,94	168,20	11,75	168,60	11,77	408,20	442,80
H1	371,08	7,83	168,20	12,56	168,00	12,60	545,10	442,80
H2	371,59	7,79	264,80	12,68	263,60	12,57	545,10	442,80
H3	610,77	7,93	167,00	12,63	168,00	12,63	545,10	442,80

Os enrijecedores são contínuos ao longo de toda altura da alma e possuem espessura igual a 14 mm em todas as análises, de forma que a sua largura e altura variam de acordo com a largura da mesa e a altura da alma de cada viga. A resistência ao escoamento do aço do enrijecedor para cada modelo é igual ao da alma.

4.2.2.2 Material

Adotou-se o comportamento elasto-plástico dos aços estruturais de acordo com o estudo experimental do autor e conforme apresentado na Figura 4.6. Para o aço com f_y = 408,20 N/mm², tem-se: módulo de elasticidade longitudinal (E) = 178,50 GPa, f_u = 529,50 N/mm², $\varepsilon_{st}/\varepsilon_y$ = 4,77 e $\varepsilon_u/\varepsilon_y$ = 33,99. Para o aço com f_y = 442,80 N/mm², tem-se: E = 197,60 GPa, f_u = 555,30 N/mm², $\varepsilon_{st}/\varepsilon_y$ = 8,06 e $\varepsilon_u/\varepsilon_y$ = 38,45. E, finalmente, para o aço com f_y= 545,10 N/mm², tem-se: E = 206,80 GPa, f_u = 627,20 N/mm², $\varepsilon_{st}/\varepsilon_y$ = 11,45 e $\varepsilon_u/\varepsilon_y$ = 32,80. Tomou-se coeficiente de Poisson (υ) igual a 0,3. Importante ressaltar que, após a ruptura considerou-se que a tensão retorna a um valor bem próximo do valor nulo para que fosse possível observar de maneira mais clara a capacidade resistente última da viga.

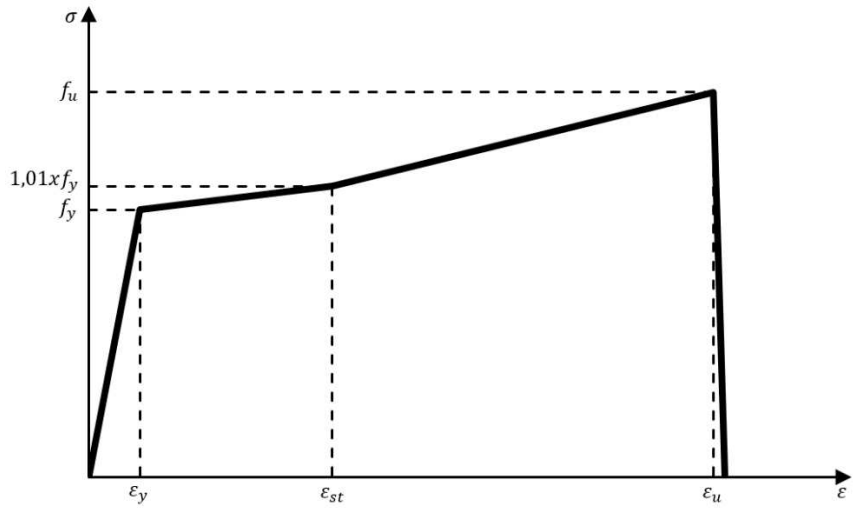


Figura 4.6 - Diagrama tensão x deformação elasto-plástico adotado nas análises dos modelos compactos.

4.2.2.3 Condições de contorno

O sistema de eixos é definido de maneira que o eixo z corresponde à direção longitudinal da viga, enquanto os eixos x e y representam o maior e menor eixo de inércia, respectivamente. Para garantir o comportamento esperado foram restringidas as translações nos eixos x (u_x) e y (u_y) e a rotação em torno de e z (φ_z) ao longo da junção entre mesa inferior e enrijecedor transversal nas duas extremidades da viga. Além disso, a translação no eixo z (u_z) em uma

dessas extremidade também foi restringida. A fim de concentrar as influências das instabilidades locais no comportamento da viga, os modelos numéricos também foram restringidos quanto à translação fora do plano da alma ao longo da junção entre alma e mesa superior do perfil, conforme Figura 4.7.

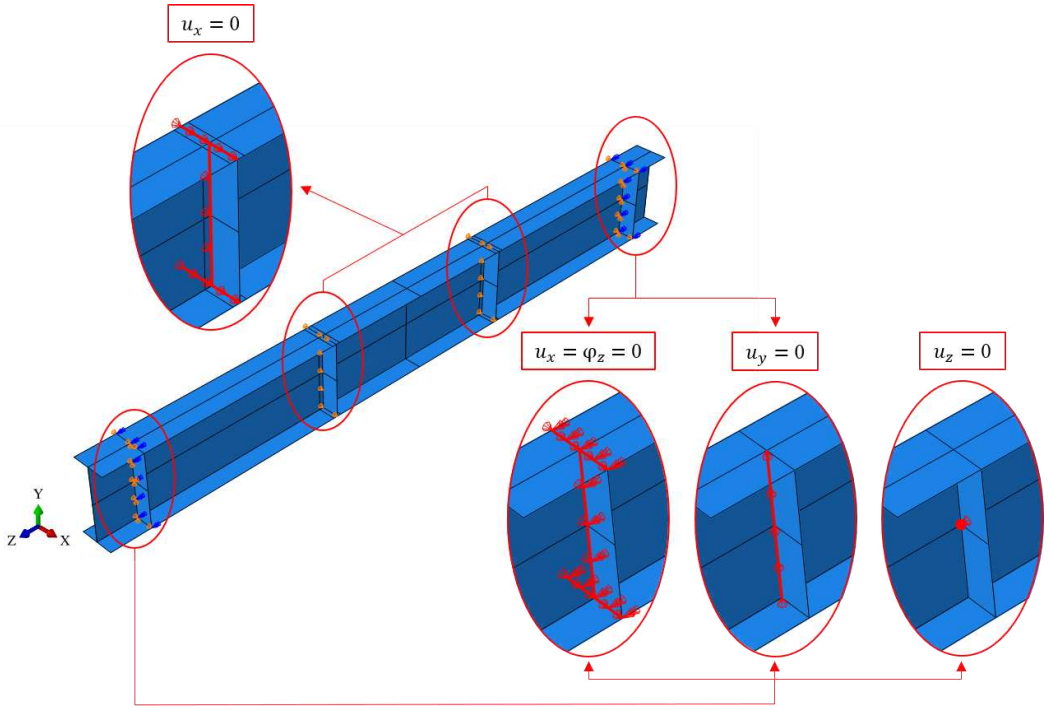


Figura 4.7 – Condições de contorno adotadas no modelo compacto.

4.2.2.4 Aplicação do carregamento

Adotou-se um carregamento distribuído do tipo *pressure* aplicado em dois sólidos, os quais foram denominados de chapas de carregamento, de acordo com a Figura 4.8. Cada chapa de carregamento possui comprimento igual à largura da mesa superior do perfil, espessura de 20 mm e largura de 50 mm.

É importante ressaltar que a ligação entre o perfil I e as chapas de carregamento foi realizada através da restrição do tipo *tie*, como mostra a Figura 4.9. Com esse tipo de restrição foi possível conectar as duas superfícies e impedir o movimento relativo entre elas.

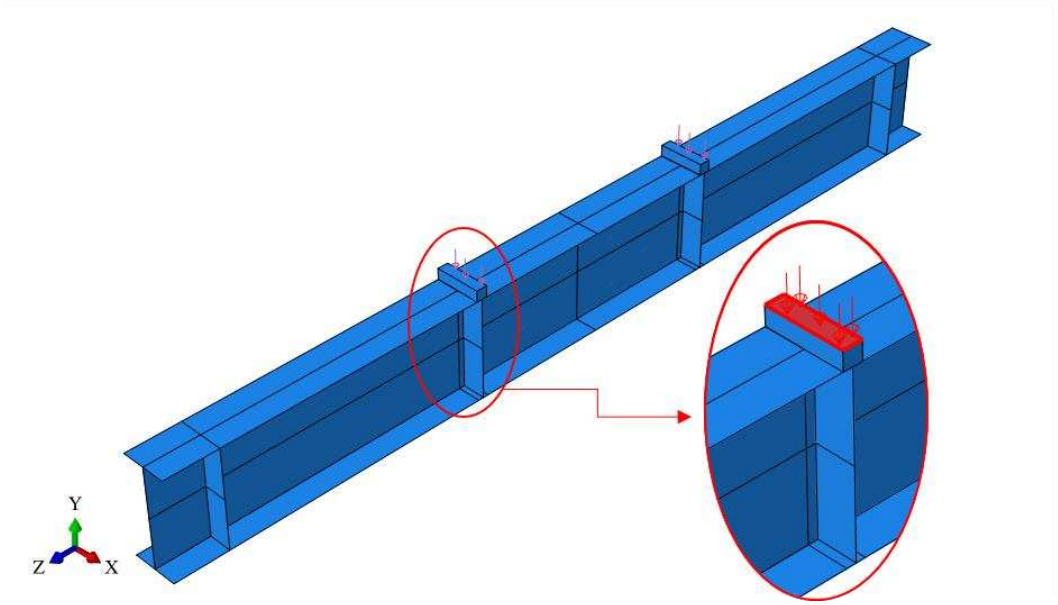


Figura 4.8 – Detalhe do carregamento distribuído aplicado no perfil I no modelo compacto.

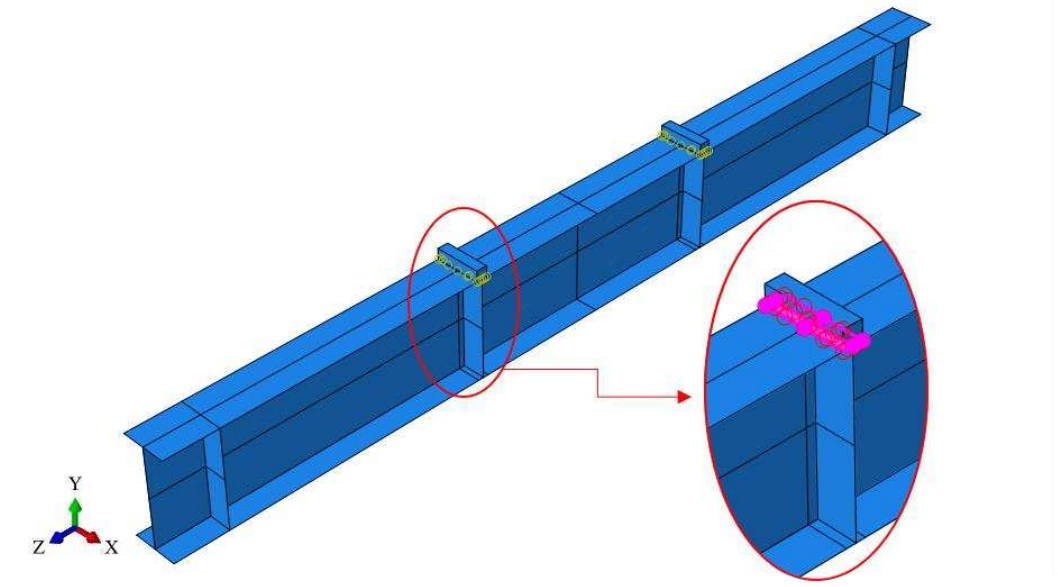


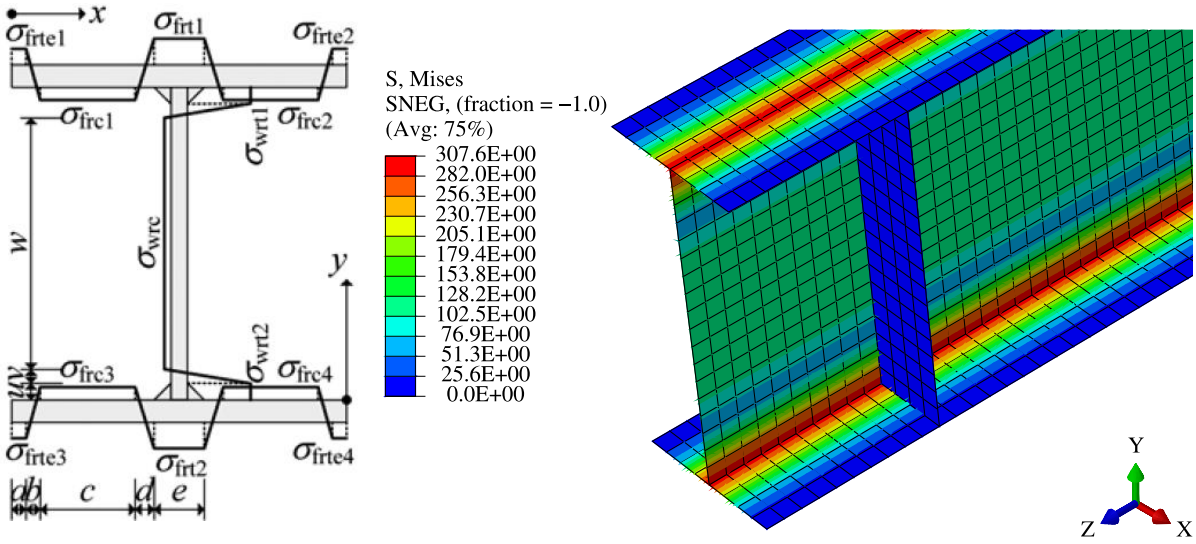
Figura 4.9 – Detalhe da restrição do tipo *tie* aplicada no perfil I e na chapa de carregamento no modelo compacto.

4.2.2.5 Imperfeições geométricas iniciais e tensões residuais

É de conhecimento que as imperfeições iniciais estão presentes nas chapas que compõem o perfil. Porém, primeiramente, realizou-se uma análise para comparar o comportamento da viga quando há a presença das imperfeições iniciais e quando as mesmas estão ausentes.

Logo, assim como Shokouhian (2014), associou-se o valor da imperfeição geométrica inicial medida pelo autor referente ao primeiro modo de flambagem da viga e, embora o ideal fosse adotar uma distribuição das tensões residuais para vigas convencionais e outra para vigas híbridas, devido à falta de dados sobre a distribuição de tensões residuais neste segundo caso,

o autor adotou a distribuição das tensões residuais proposta por Ban *et al*. (2013). Ressalta-se, entretanto, que os estudos experimentais de Ban *et al.* (2013) foram realizados somente em vigas convencionais compostas pelo mesmo aço, conforme pode ser observado na Figura 4.10.



(a) (b)

Figura 4.10 – Tensões residuais: (a) distribuição das tensões residuais (Ban *et al*., 2013) e (b) tensão residual aplicada ao modelo numérico C1.

4.2.2.6 Discretização do modelo numérico

Nesse modelo, optou-se pelo elemento S4R, um tipo de elemento de casca com integração reduzida que possui quatro nós, conforme Figura 4.11.

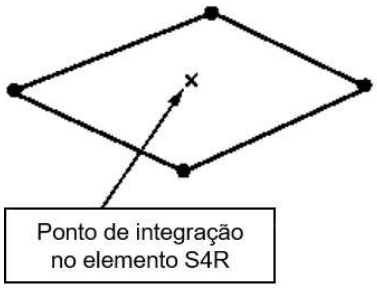
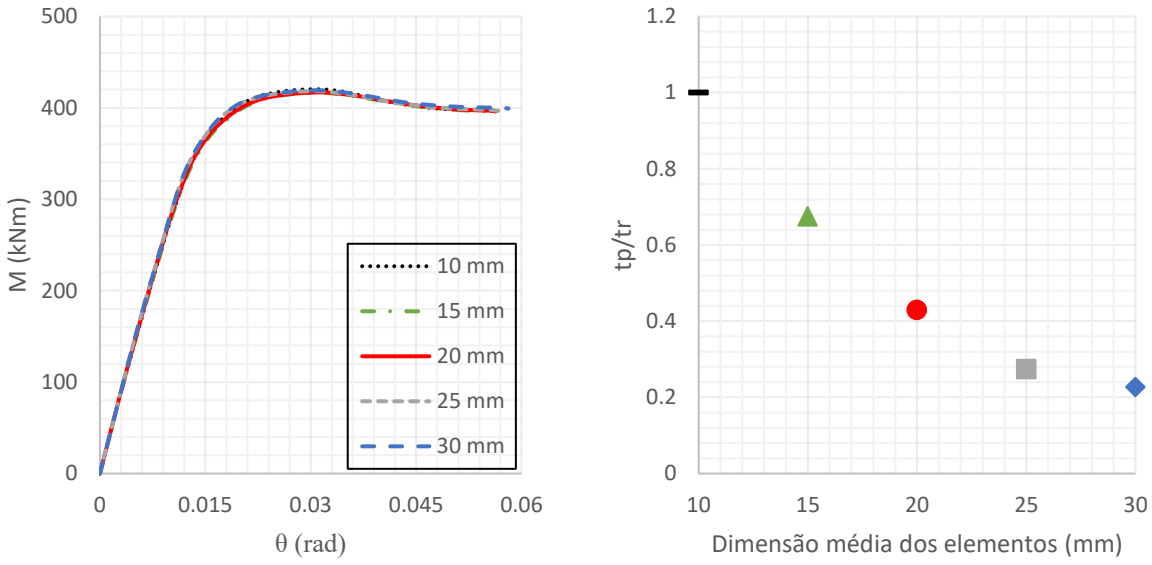


Figura 4.11 – Elemento de malha S4R.

Sabe-se que a utilização de uma malha mais refinada de elementos finitos geralmente proporciona uma melhor aproximação do modelo numérico quando comparada a uma malha mais grosseira. Uma vez que o autor não mencionou a dimensão dos elementos finitos adotada nos modelos, realizou-se um estudo de sensibilidade da malha com elementos de

aproximadamente 10 mm, 15 mm, 20 mm, 25mm e 30 mm. Considerando a curva momento fletor (M) *versus* rotação (θ) e a relação entre o tempo de processamento de cada modelo numérico (t_p) e o tempo de processamento do modelo de referência (t_r), tomado como o tempo de processamento do modelo com malha de elementos de aproximadamente 10 mm, definiu-se a dimensão média dos elementos finitos igual a 20 mm no perfil I e 5 mm nas chapas de carregamento, de acordo com a Figura 4.12 e Figura 4.13.



(a) (b)

Figura 4.12 – Dimensão média dos elementos: (a) impacto nas curvas momento fletor *versus* rotação e (b) relação t_p/t_r para cada modelo.

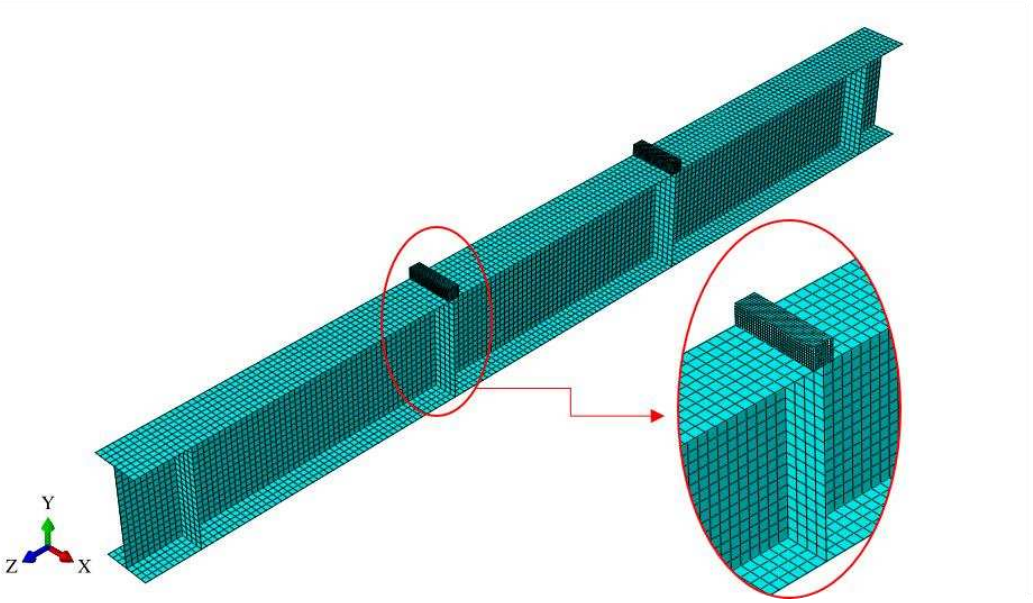


Figura 4.13 - Detalhe da malha de elementos finitos adotada no modelo compacto.

4.2.3 Aferição do modelo numérico

Após a análise da capacidade resistente última, para expressar o comportamento à flexão de vigas, foi realizada uma análise entre o momento normalizado (M/M_{pl}) e a rotação normalizada do nó na extremidade da viga (θ/θ_p). A Figura 4.14, Figura 4.15 e Figura 4.16 apresentam a comparação dos resultados obtidos neste estudo com os resultados experimentais e numéricos obtidos por Shokouhian (2014).

Levando em consideração as análises dos momentos resistentes últimos obtidos percebe-se que, para a maioria das vigas, o modelo numérico adotado atingiu resultados suficientemente próximos, tanto do modelo numérico como do modelo experimental proposto por Shokouhian (2014). Entretanto, para a viga H2, o erro relativo foi maior do que 10%. Fato este que pode ser explicado uma vez que H2 é a única viga que possui 800 mm de distância entre as forças concentradas. Mesmo assim, o erro ficou em torno de 16%, o que foi considerado um erro aceitável tendo em vista as incertezas do modelo, devido à falta de maiores informações.

As Figuras 4.17, 4.19, 4.21, 4.23, 4.25 e 4.27 apresentam as tensões de von Mises para as vigas no último passo de carga, ou seja, na iminência do escoamento da seção, e as Figuras 4.18, 4.20, 4.22, 4.24, 4.26 e 4.28 mostram as deformadas finais das vigas.

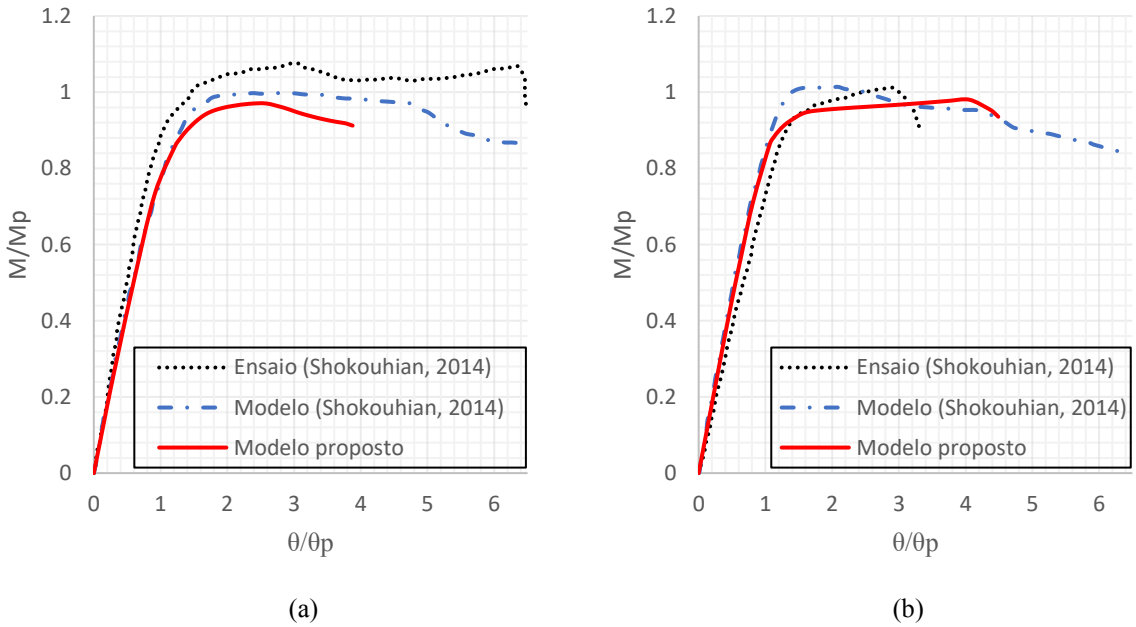
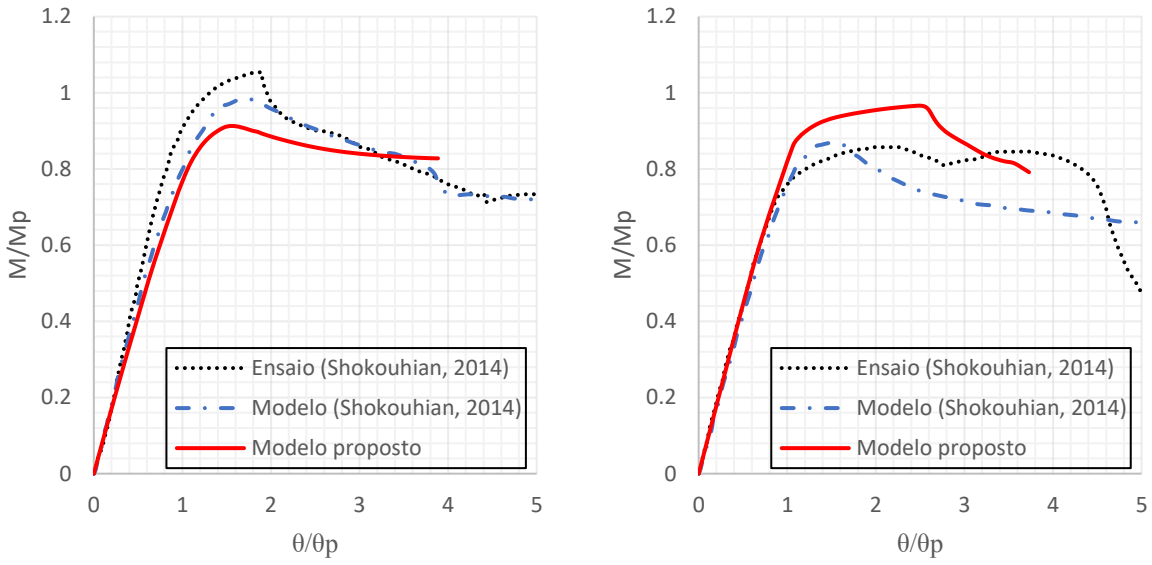
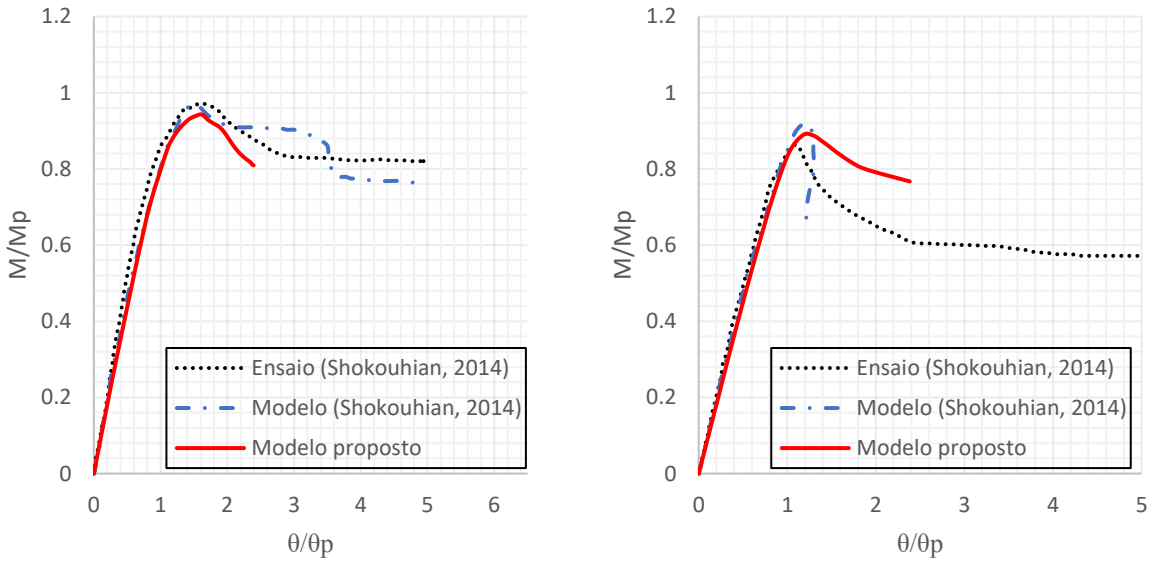


Figura 4.14 - Comparação entre momento e rotação: (a) Viga C1 e (b) Viga H1 (adaptado de Shokouhian, 2014).



(a) (b)

Figura 4.15 - Comparação entre momento e rotação: (a) Viga C2 e (b) Viga H2 (adaptado de Shokouhian, 2014).



(a) (b)

Figura 4.16 - Comparação entre momento e rotação: (a) Viga C3 e (b) Viga H3 (adaptado de Shokouhian, 2014).

As Figuras 4.17 e 4.18 mostram a verificação do resultado para a viga C1. Como pode ser visualizado, a mesa comprimida falhou por flambagem local e nos últimos passos induziu a flambagem da alma. Assim como no modelo de Shokouhian (2014), houve a formação de uma semionda positiva na parte central da mesa comprimida e duas semiondas negativas ao lado da primeira semionda. Também foi possível observar semiondas opostas na direção contrária a x.

As Figuras 4.19 e 4.20 mostram a verificação do resultado para a viga H1. Nesse caso, assim como para a viga C1, a mesa comprimida falhou por flambagem local e nos últimos passos induziu a flambagem da alma. Novamente, como no modelo de Shokouhian (2014), houve a formação de uma semionda na parte central da mesa comprimida. Também foi possível observar essa semionda opostas na direção contrária a x.

As Figuras 4.21 e 4.22 mostram a verificação do resultado para a viga C2. Nesse caso, assim como para a viga C1 e H1, a mesa comprimida falhou por flambagem local e nos últimos passos induziu a flambagem da alma. No modelo de Shokouhian (2014), houve a formação duas semiondas de direções opostas de cada lado da mesa comprimida.

As Figuras 4.23 e 4.24 mostram a verificação do resultado para a viga H2. Nesse caso, assim como para a viga C1, H1 e C2, a mesa comprimida falhou por flambagem local e nos últimos passos induziu a flambagem da alma.

As Figuras 4.25 e 4.26 mostram a verificação do resultado para a viga C3. Devido ao fato de a altura da alma ser maior do que a altura da alma da viga C1, ocorreu a flambagem local da alma, que induziu a flambagem local da mesa. Houve duas semiondas nas almas nos dois painéis finais e finalmente, a flambagem local da mesa ocorreu na parte central da viga.

As Figuras 4.27 e 4.28 mostram a verificação do resultado para a viga H3. Assim como a viga C3, ocorreu a flambagem diagonal local da alma nos dois painéis finais. Nesse modelo não foi observada a flambagem local da mesa.

Além disso, como demonstrado na Tabela 4.2, é possível concluir que, para os casos estudados, a capacidade resistente das vigas híbridas é em torno de 14% a 28% maior que as determinadas para as vigas convencionas. O resultado obtido é esperado, tendo em vista o emprego de aços com maior resistência nas vigas híbridas, que levam ao aumento da capacidade resistente.

Tabela 4.2 – Comparação da resistência última entre os estudos.

Modelo	$P_{ul\,(Modelo\ proposto)}$ **(N)**	$P_{ul\,(Ensaio-\text{Shokouhian})}$ **(N)**	$P_{ul\,(Modelo-\text{Shokouhian})}$ **(N)**
C1	416.875,38	447.940,17	414.282,33
C2	549.032,44	616.742,33	575.275,75
C3	804.235,49	784.882,46	788.115,76
H1	530.868,10	546.161,28	546.700,43
H2	711.872,17	615.085,58	621.522,52
H3	921.827,77	879.891,82	936.921,85

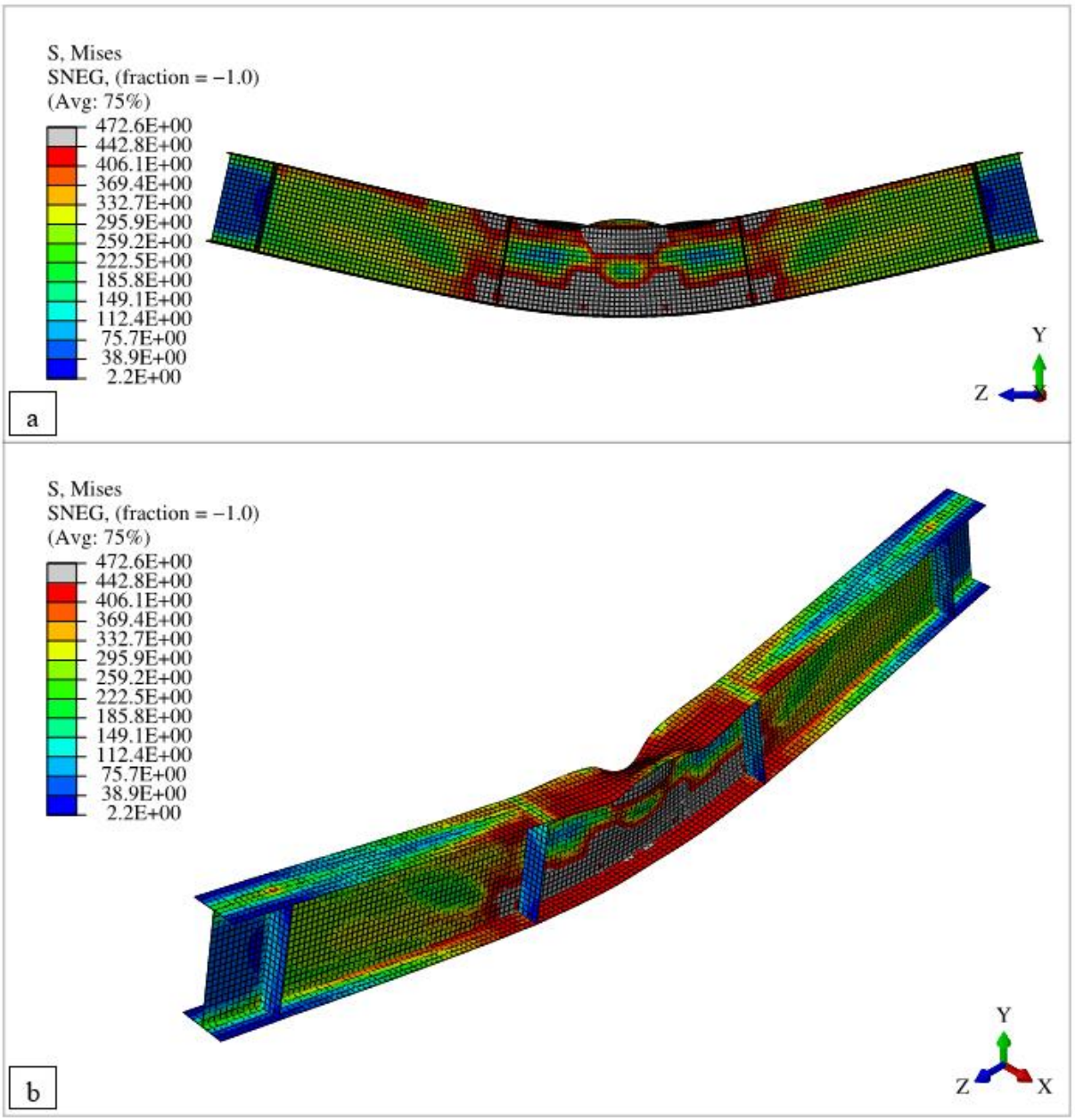


Figura 4.17 - Verificação do modelo de elementos finitos para viga C1: (a) Vista frontal das tensões de von Mises na iminência do escoamento da seção e (b) Vista isométrica das tensões de von Mises na iminência do escoamento da seção.

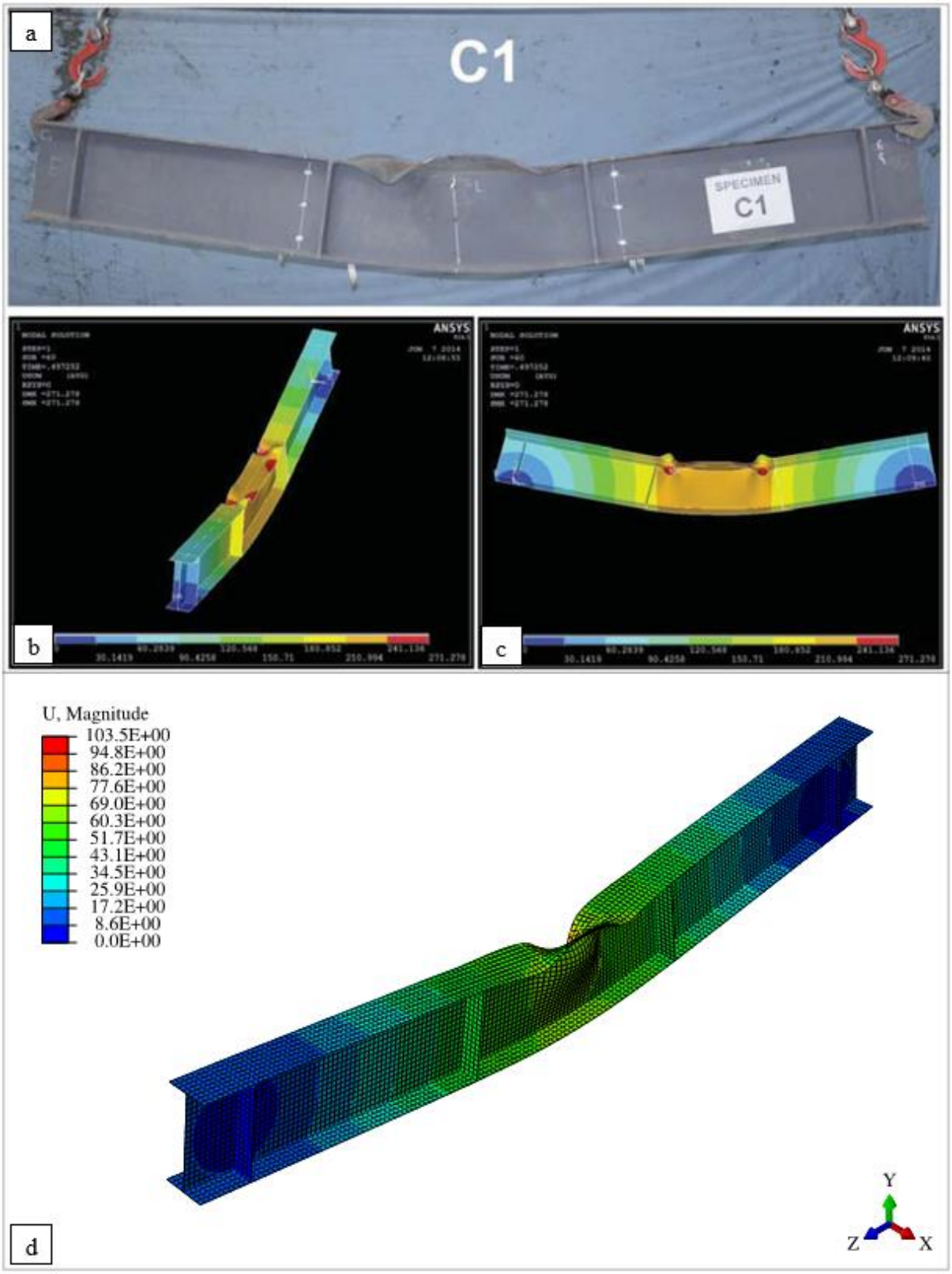


Figura 4.18 - Verificação do modelo de elementos finitos para viga C1: (a) Ensaio experimental de Shokouhian (2014), (b) Vista isométrica do modelo numérico de Shokouhian (2014), (c) Vista frontal do modelo numérico de Shokouhian (2014) e (d) Vista isométrica do modelo numérico proposto (adaptado de Shokouhian, 2014).

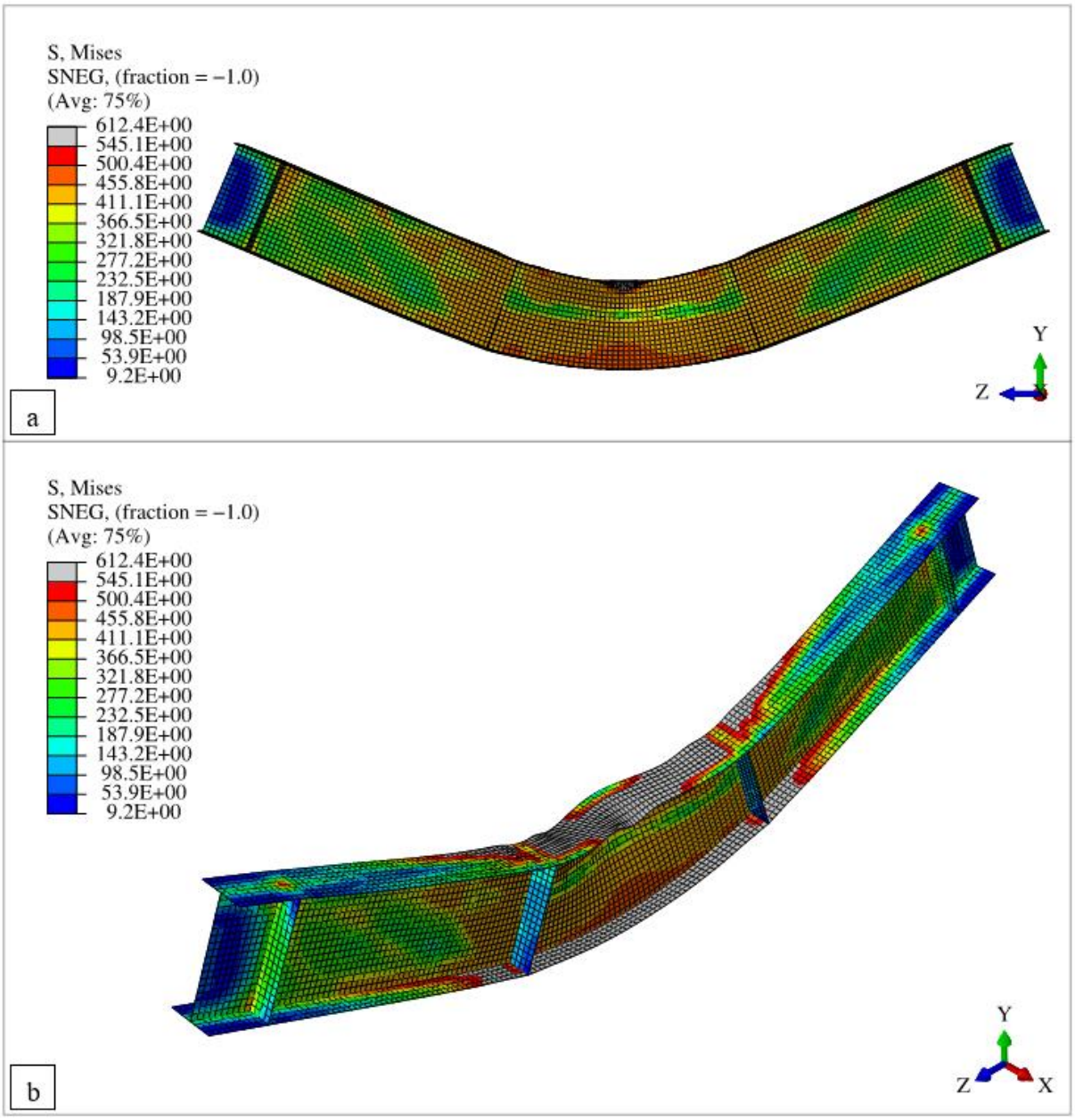


Figura 4.19 - Verificação do modelo de elementos finitos para viga H1: (a) Vista frontal das tensões de von Mises na iminência do escoamento da seção e (b) Vista isométrica das tensões de von Mises na iminência do escoamento da seção.

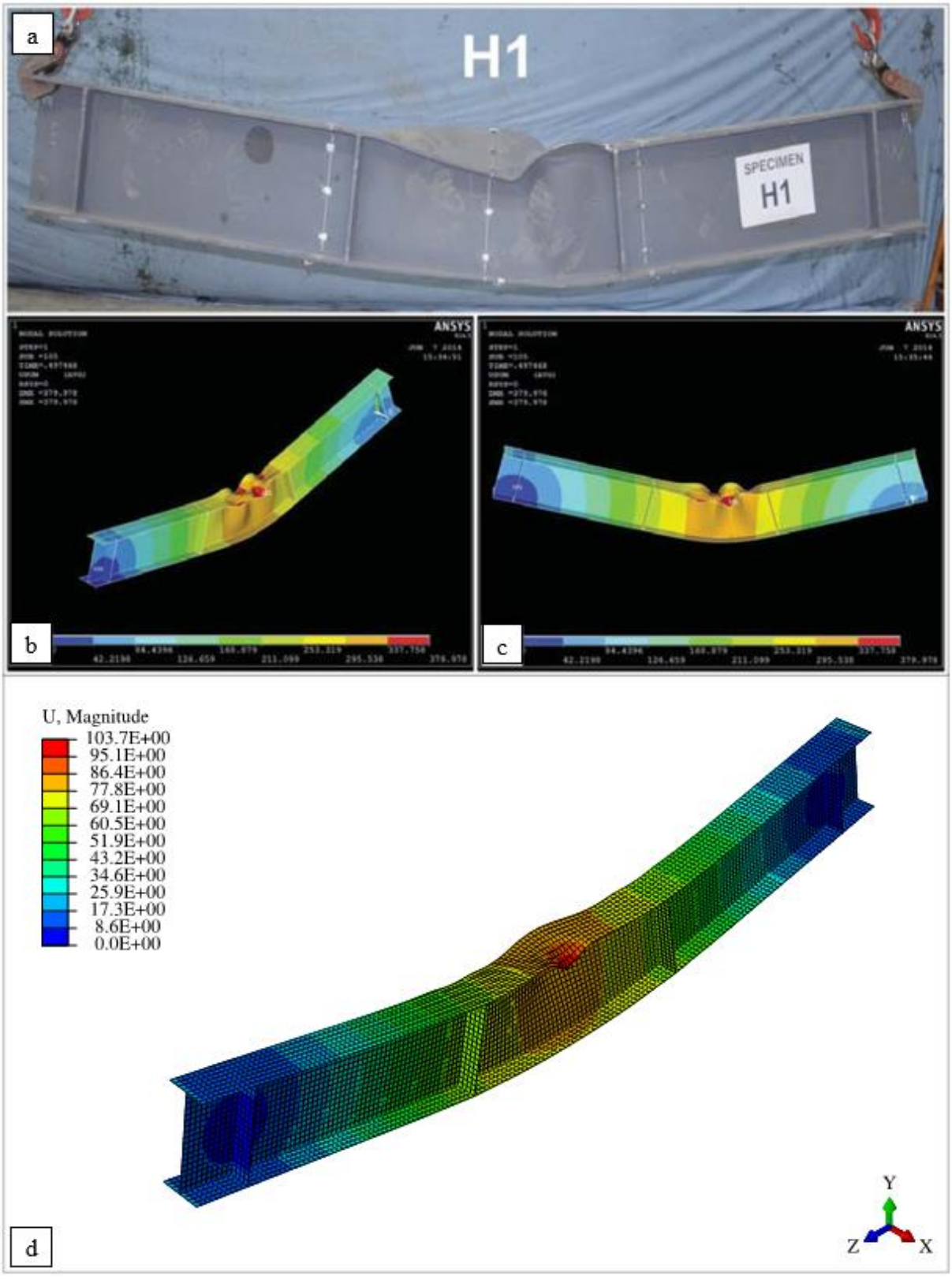


Figura 4.20 - Verificação do modelo de elementos finitos para viga H1: (a) Ensaio experimental de Shokouhian (2014), (b) Vista isométrica do modelo numérico de Shokouhian (2014), (c) Vista frontal do modelo numérico de Shokouhian (2014) e (d) Vista isométrica do modelo numérico proposto (adaptado de Shokouhian, 2014).

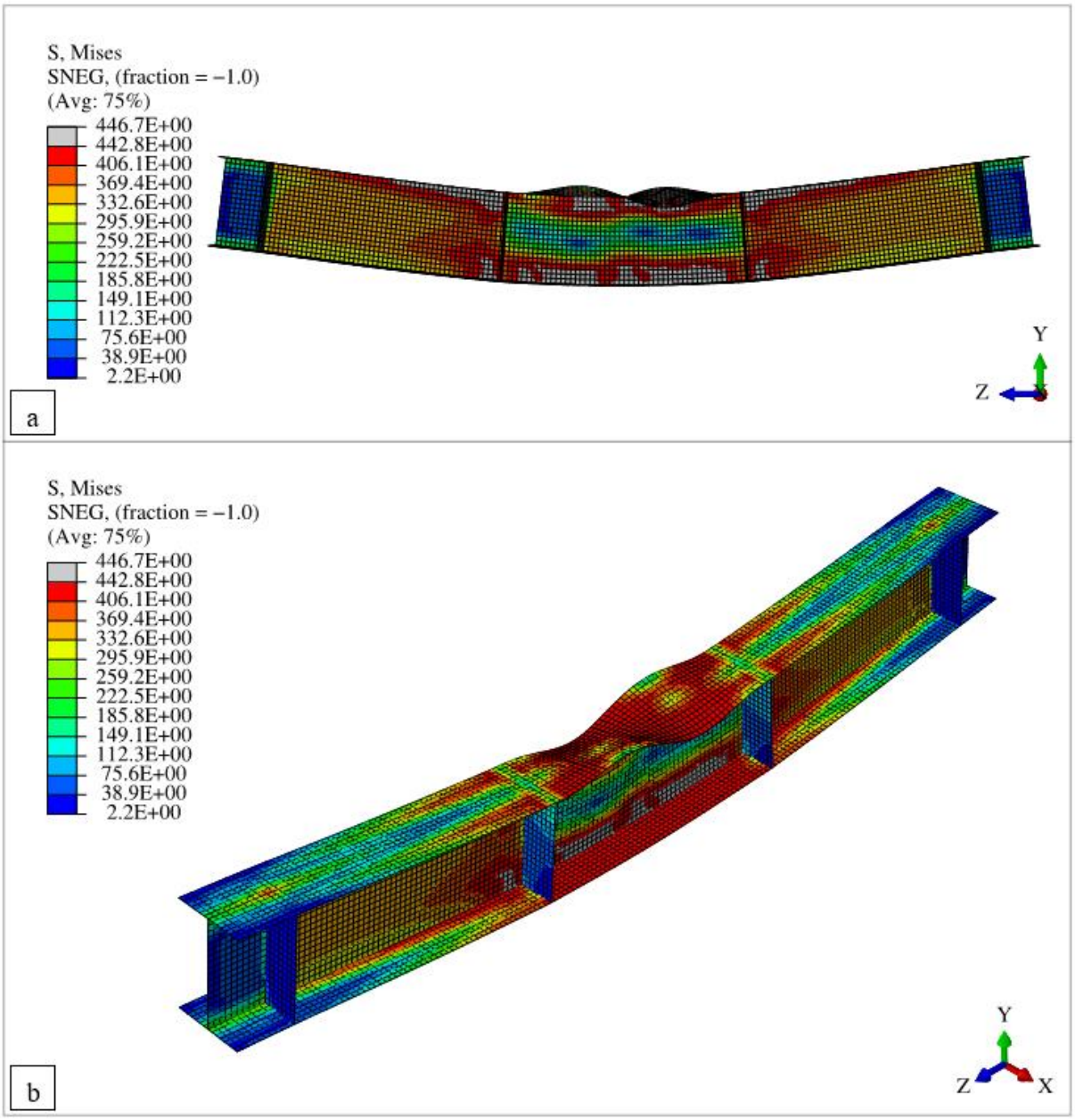


Figura 4.21- Verificação do modelo de elementos finitos para viga C2: (a) Vista frontal das tensões de von Mises na iminência do escoamento da seção e (b) Vista isométrica das tensões de von Mises na iminência do escoamento da seção.

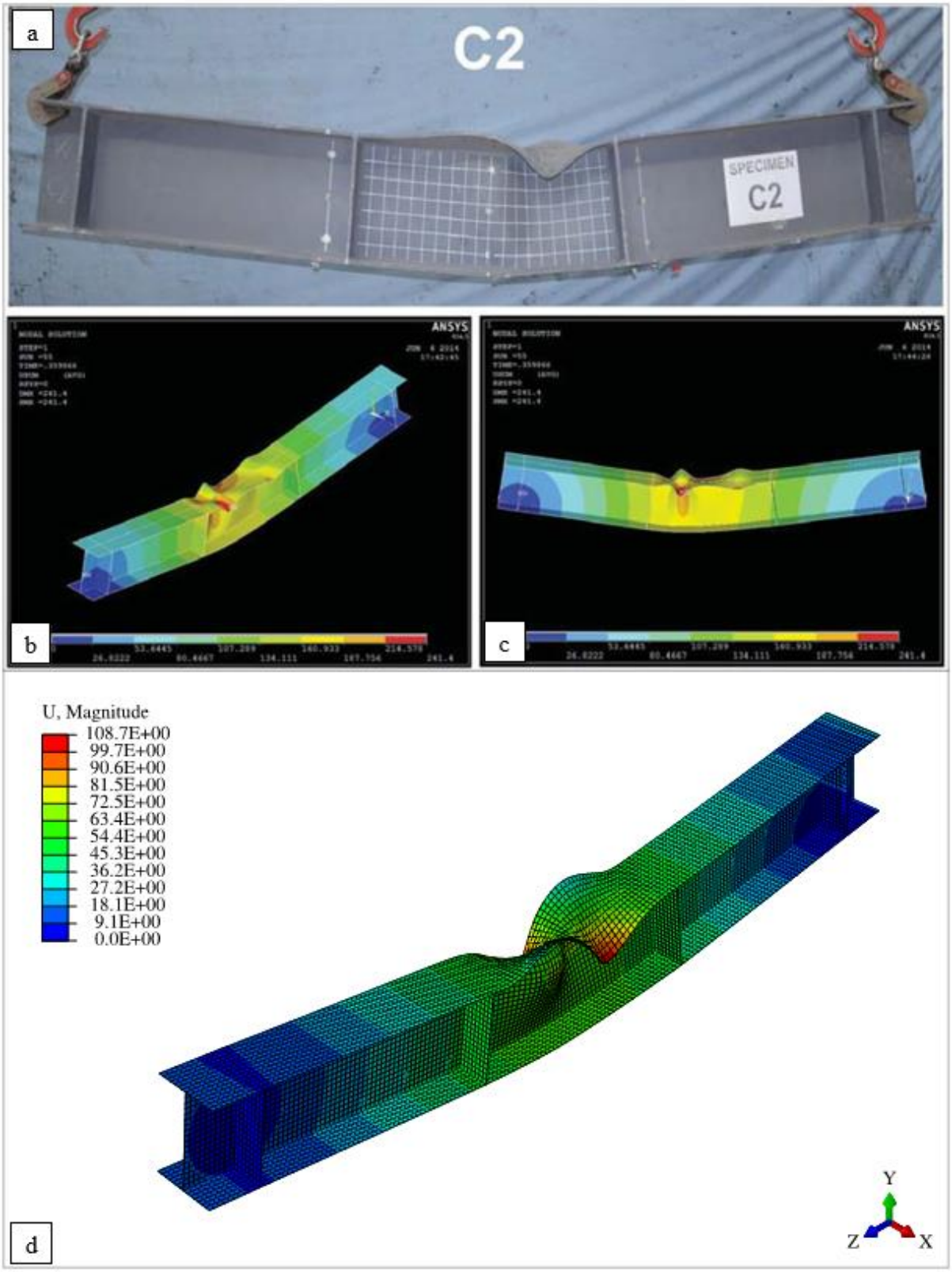


Figura 4.22 - Verificação do modelo de elementos finitos para viga C2: (a) Ensaio experimental de Shokouhian (2014), (b) Vista isométrica do modelo numérico de Shokouhian (2014), (c) Vista frontal do modelo numérico de Shokouhian (2014) e (d) Vista isométrica do modelo numérico proposto (adaptado de Shokouhian, 2014).

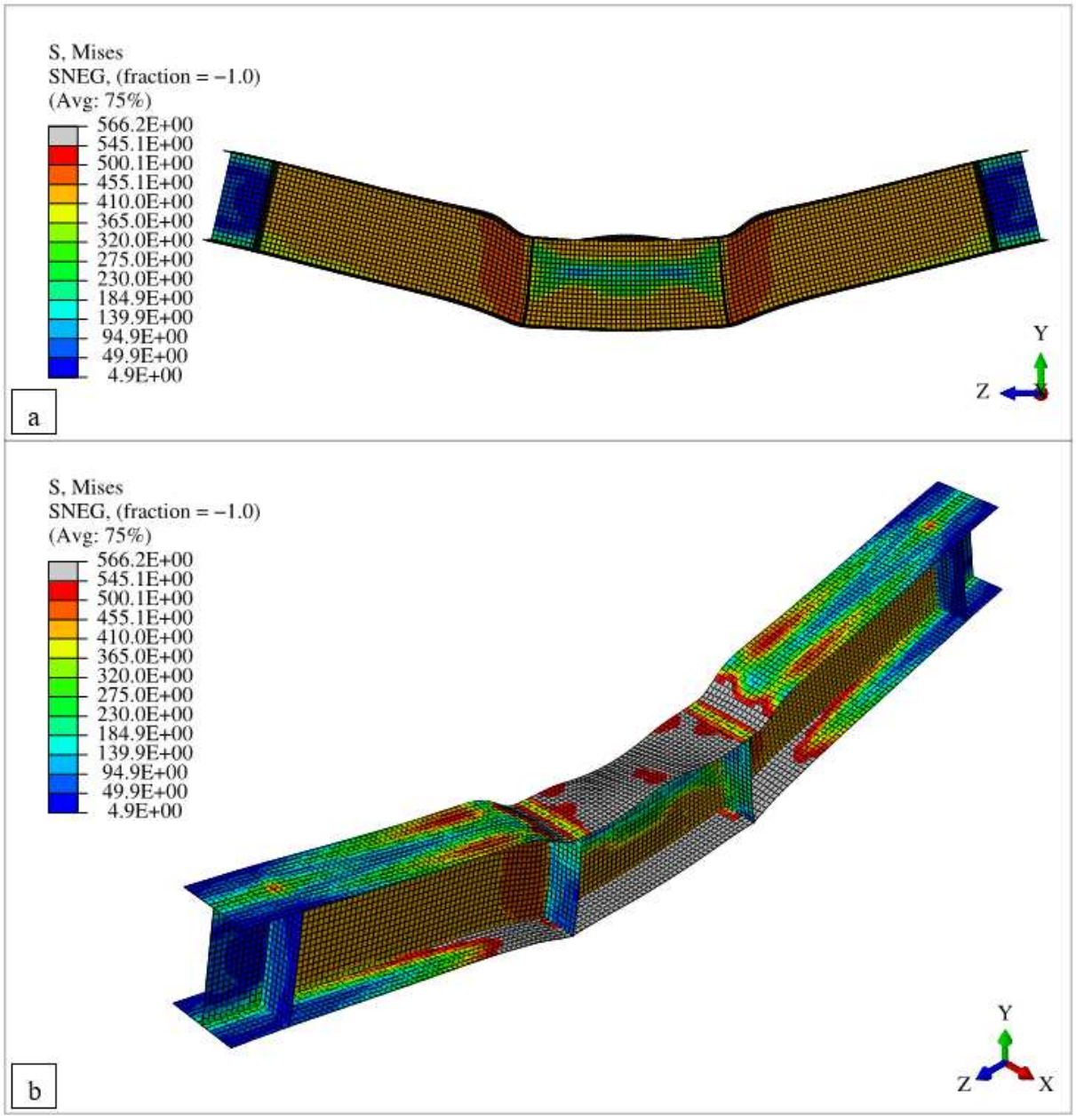


Figura 4.23- Verificação do modelo de elementos finitos para viga H2: (a) Vista frontal das tensões de von Mises na iminência do escoamento da seção e (b) Vista isométrica das tensões de von Mises na iminência do escoamento da seção.

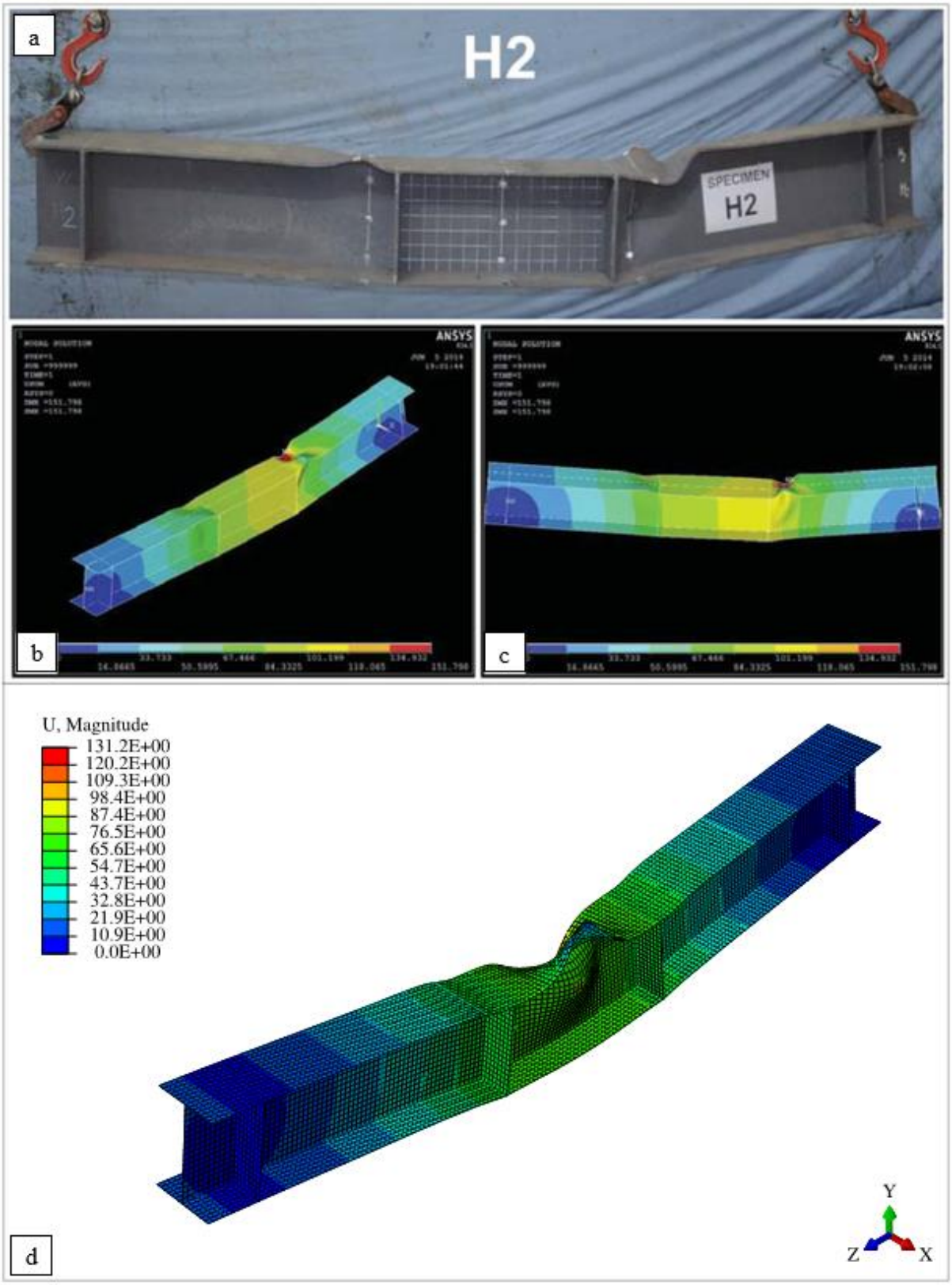


Figura 4.24 - Verificação do modelo de elementos finitos para viga H2: (a) Ensaio experimental de Shokouhian (2014), (b) Vista isométrica do modelo numérico de Shokouhian (2014), (c) Vista frontal do modelo numérico de Shokouhian (2014) e (d) Vista isométrica do modelo numérico proposto (adaptado de Shokouhian, 2014).

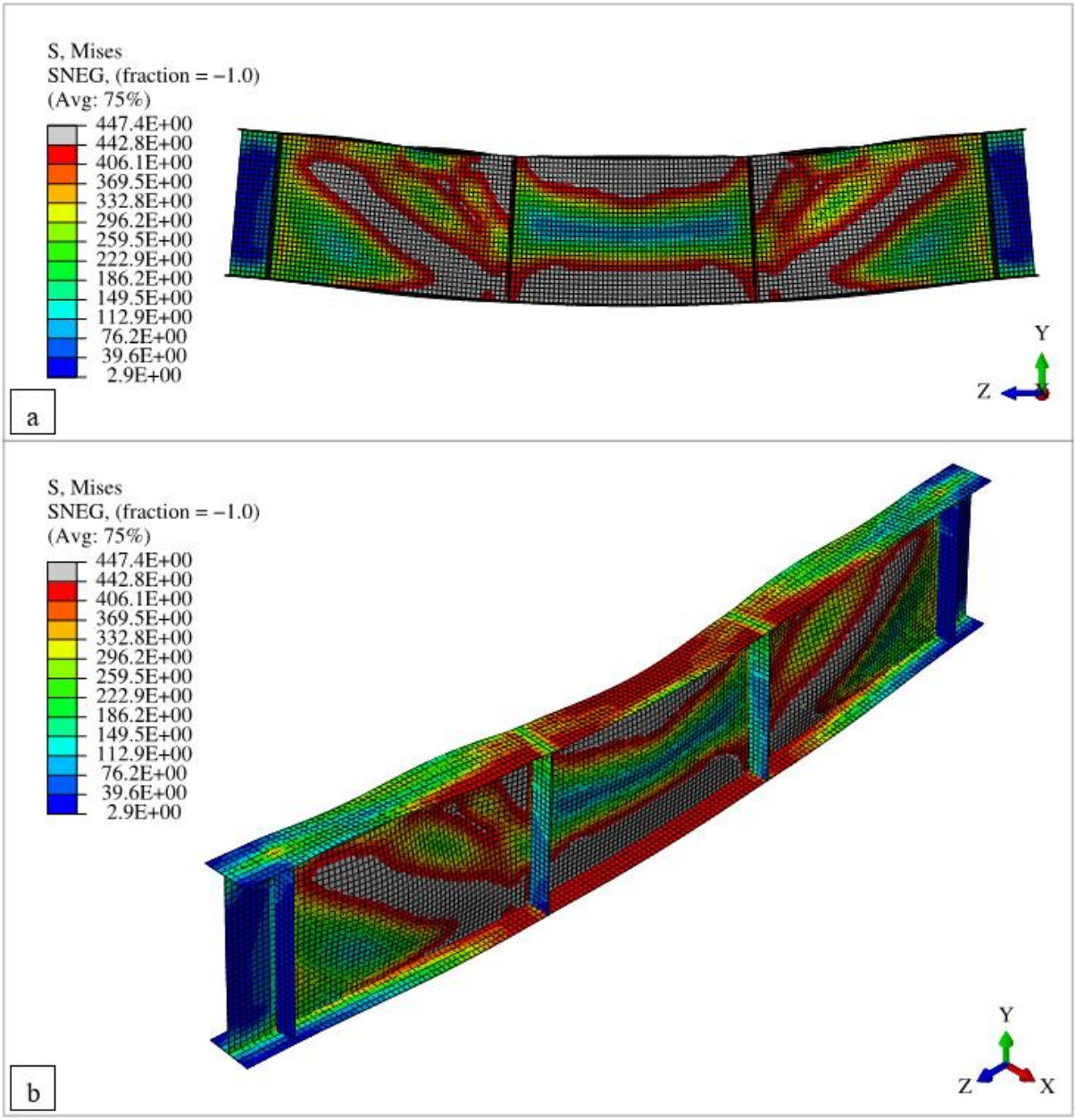


Figura 4.25 - Verificação do modelo de elementos finitos para viga C3: (a) Vista frontal das tensões de von Mises na iminência do escoamento da seção e (b) Vista isométrica das tensões de von Mises na iminência do escoamento da seção.

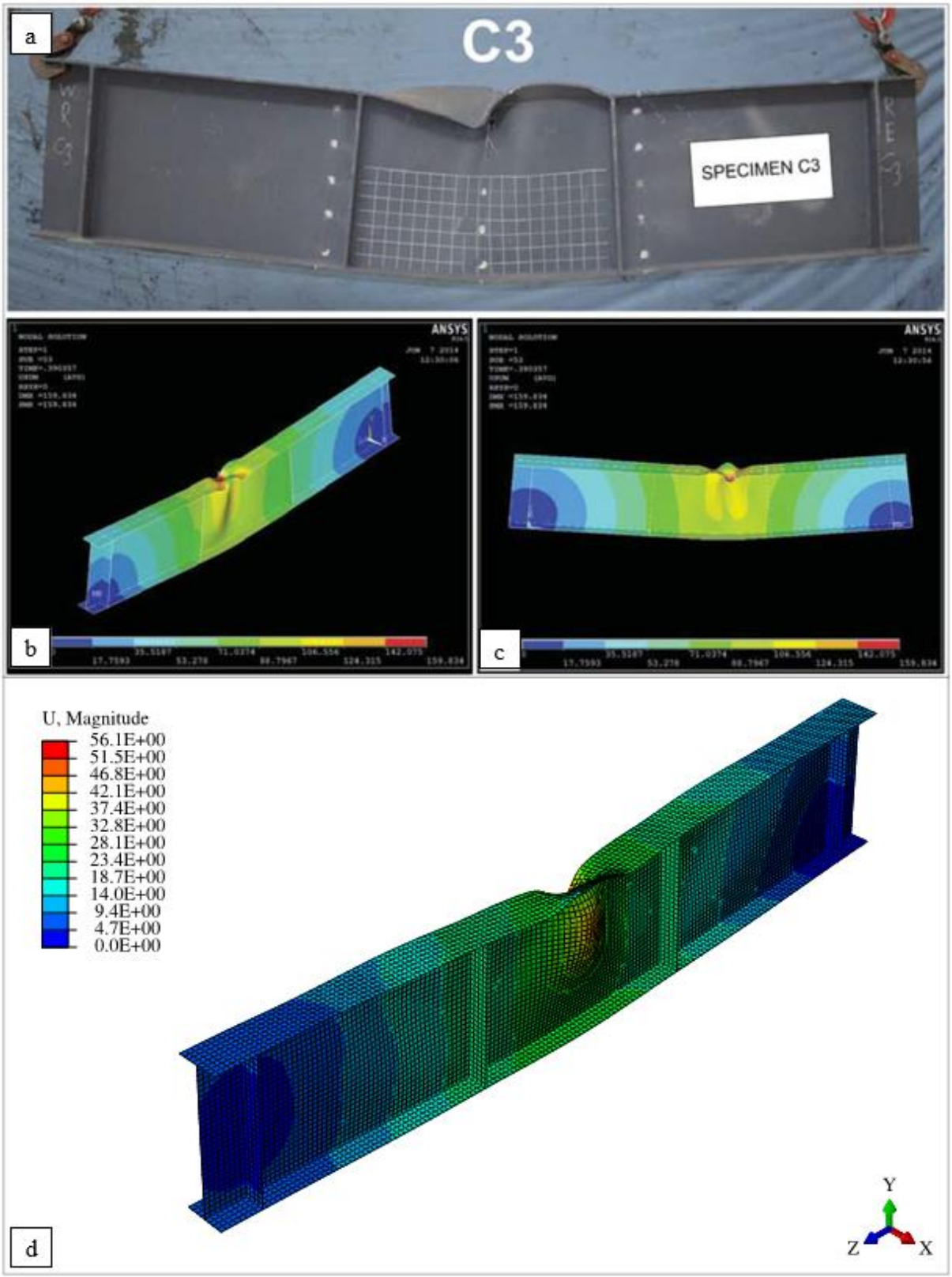


Figura 4.26 - Verificação do modelo de elementos finitos para viga C3: (a) Ensaio experimental de Shokouhian (2014), (b) Vista isométrica do modelo numérico de Shokouhian (2014), (c) Vista frontal do modelo numérico de Shokouhian (2014) e (d) Vista isométrica do modelo numérico proposto (adaptado de Shokouhian, 2014).

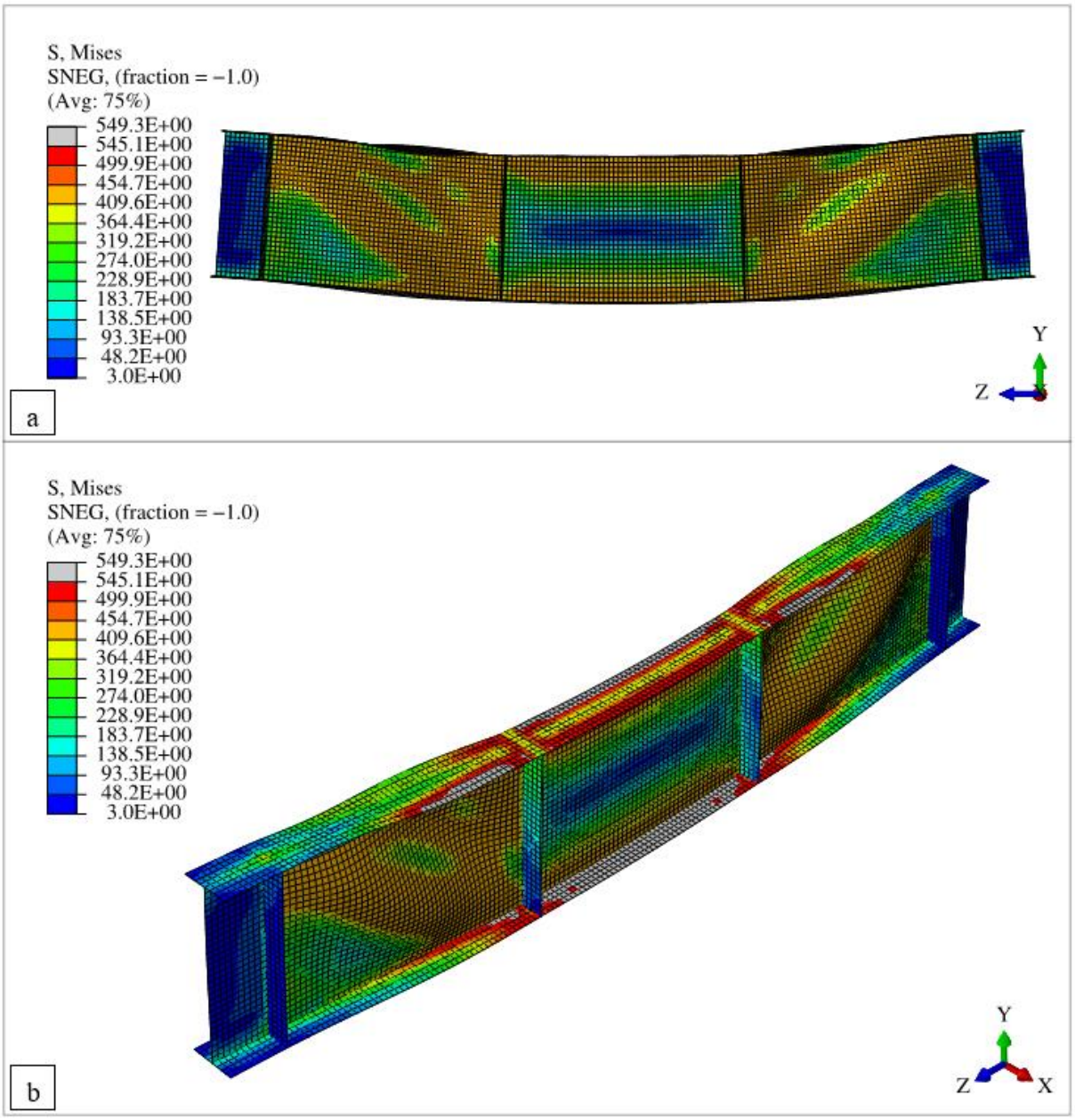


Figura 4.27- Verificação do modelo de elementos finitos para viga H3: (a) Vista frontal das tensões de von Mises na iminência do escoamento da seção e (b) Vista isométrica das tensões de von Mises na iminência do escoamento da seção.

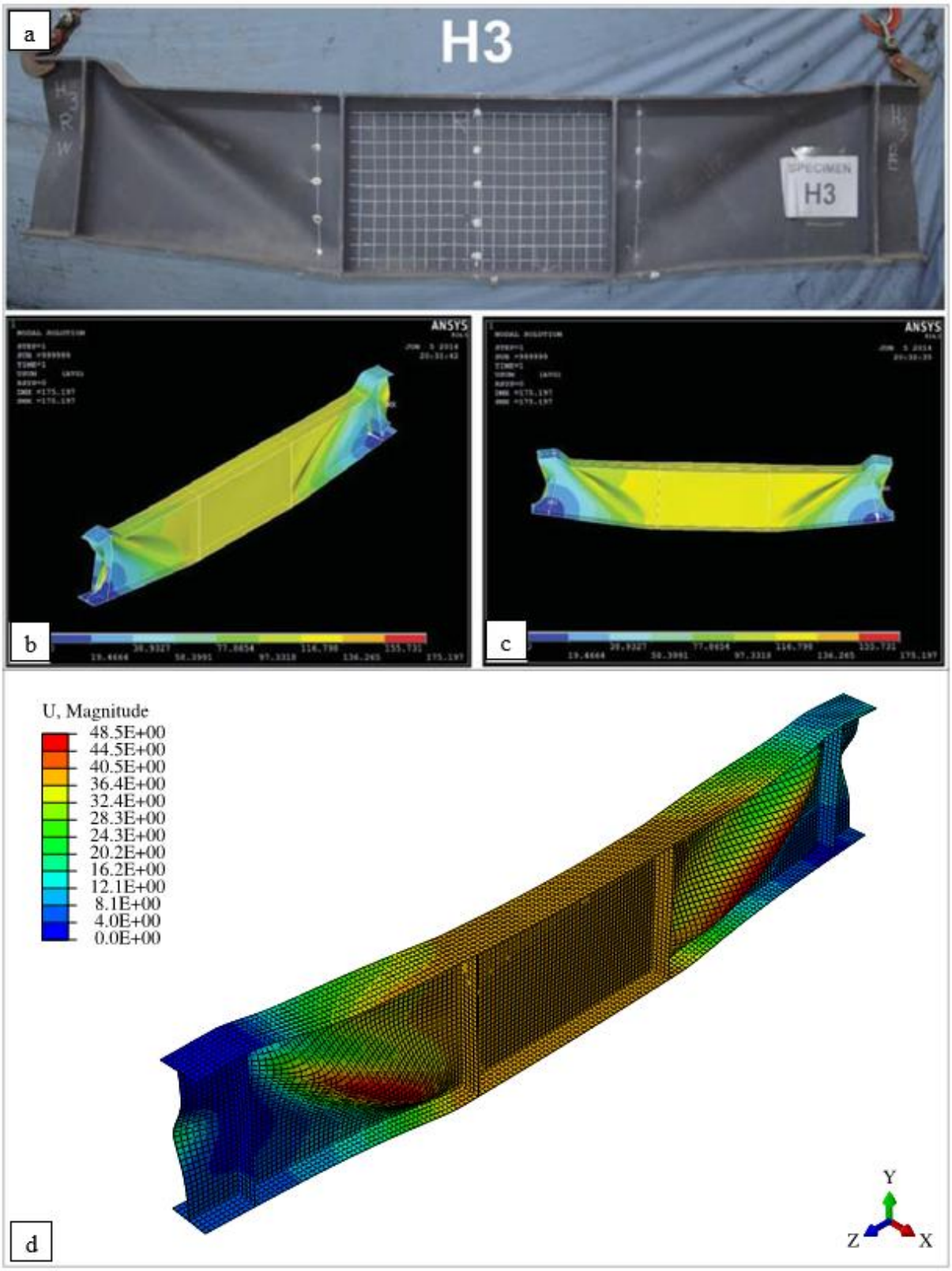


Figura 4.28 - Verificação do modelo de elementos finitos para viga H3: (a) Ensaio experimental de Shokouhian (2014), (b) Vista isométrica do modelo numérico de Shokouhian (2014), (c) Vista frontal do modelo numérico de Shokouhian (2014) e (d) Vista isométrica do modelo numérico proposto (adaptado de Shokouhian, 2014).

4.3 Modelo esbelto

4.3.1 Sinur (2011)

Sinur (2011) investigou o comportamento de quatro painéis de duas vigas transversal e longitudinalmente enrijecidas submetidas à flexão e à cortante através de análise experimental, numérica e teórica. Uma viga possui seção transversal simétrica com um enrijecedor retangular e outro trapezoidal e a outra viga tem seção transversal assimétrica com dois enrijecedores retangulares e um trapezoidal. É importante ressaltar que para este estudo somente os painéis com enrijecedores retangulares foram considerados, portanto, somente um painel de cada viga foi analisado.

É importante observar que se optou por validar o modelo numérico esbelto a partir do trabalho de Sinur (2011) uma que vez este possui um material de estudo mais amplo e detalhado de flexão de vigas esbeltas com enrijecedores, mesmo não sendo um trabalho específico de vigas híbridas.

4.3.1.1 Análise experimental realizada por Sinur (2011)

Na análise experimental, duas vigas foram fabricadas, instrumentadas e submetidas ao carregamento para os ensaios de flexão de três pontos. As vigas foram fabricadas com chapas nas quais a resistência ao escoamento nominal é de 355 N/mm². O carregamento foi aplicado através de uma chapa circular rígida com diâmetro de 200 mm e espessura de 60 mm articulado em um atuador de carga, conforme Figura 4.29.

Figura 4.29 - Configuração do ensaio no laboratório para a viga simétrica (Sinur, 2011).

As extremidades do painel analisado das vigas biapoiadas também foram restringidas quanto à translação fora do plano. Ao longo dos testes, as deformações nas mesas, enrijecedores transversais e longitudinais foram medidos por extensômetros uniaxiais enquanto os deslocamentos verticais e fora do plano foram determinados utilizando LVDT e indicadores digitais de discagem. Além disso, com a finalidade de obter o campo completo de deslocamento dos painéis para diferentes níveis de carregamento, para os deslocamentos fora do plano também foi utilizada a fotogrametria.

Com relação às imperfeições iniciais, as imperfeições geométricas dos painéis foram determinadas empregando a fotogrametria e as tensões residuais foram obtidas pela técnica destrutiva do seccionamento. Entretanto, a partir do estudo, o autor observou que a resistência da viga com imperfeições geométrica e tensões residuais foi reduzida adicionalmente de 0,8% comparada à viga na qual somente imperfeições geométricas foram consideradas. E, o autor concluiu que para o caso de almas esbeltas, algumas das tensões residuais são transformadas na geometria inicial deformada da placa e, portanto, as tensões residuais são muito mais baixas do que as obtidas para uma placa compacta. Assim, Sinur (2011) estudou a influência das tensões residuais e imperfeições geométricas iniciais em conexão com a análise de sensibilidade de imperfeições iniciais na resistência da viga e, como a influência das tensões residuais foi bastante pequena, apenas as imperfeições geométricas iniciais foram consideradas na validação do modelo numérico.

4.3.1.2 Análise numérica realizada por Sinur (2011)

Assim como no modelo compacto, o estudo numérico se ampara nos dados referentes à geometria, às especificações dos materiais e às imperfeições obtidos através da análise experimental.

Nessa análise, utilizou-se o programa de elementos finitos ABAQUS. A curva tensão *versus* deformação foi elaborada a partir da média entre os três ensaios de tração realizados na análise experimental para cada chapa e o aço estrutural foi modelado como um material elasto-plástico isotrópico. Além disso, através do modelo numérico, foi realizada uma análise da convergência da malha de elementos finitos na qual foi possível observar que resultados satisfatórios podem ser obtidos se a viga for discretizada em elementos finitos com uma distância média de 50 mm, no máximo. Sendo assim, o autor conduziu o seu estudo de verificação do seu modelo numérico com uma malha de elementos finitos com um tamanho menor ou igual a 50 mm.

Ao final, o autor concluiu que houve uma boa concordância entre os resultados experimentais e a simulação numérica.

4.3.1.3 Análise teórica realizada por Sinur (2011)

De acordo com a EN 1993-1-5, o autor determinou a resistência última a partir do MLE.

4.3.2 Descrição dos procedimentos de modelagem do presente estudo

4.3.2.1 Geometria

No presente estudo, duas vigas de seção transversal do tipo I foram analisadas. A primeira viga possui seção transversal simétrica e comprimento de 11,16 m. A segunda possui seção transversal assimétrica e comprimento de 11,35 m. Ambas possuem enrijecedores longitudinais retangular e trapezoidal, porém, este trabalho aborda somente um painel em cada uma das vigas e, os painéis analisados em ambas são os painéis com a presença dos enrijecedores longitudinais retangulares.

A viga simétrica possui um enrijecedor longitudinal retangular e um enrijecedor trapezoidal e, o centro de gravidade desses está posicionado na zona comprimida da alma, a 350 mm da mesa superior. Há sete enrijecedores transversais intermediários com espessura de 15 mm e largura de 120 mm e seis enrijecedores transversais duplos com espessura de 20 mm e largura de 156 mm. Os enrijecedores transversais duplos foram utilizados na parte central da viga para aplicar a força externa e também foram utilizados após o comprimento adicional dos apoios para garantir a extremidade rígida na viga. Além disso, fora da área de investigação, 120 mm dos enrijecedores transversais intermediários, a espessura da alma é de 8 mm e somente na parte central da viga (no comprimento de 800 mm) o autor considerou a espessura da alma igual a 15 mm para garantir o comportamento elástico nessa área de transição. Para esse trabalho, nesta viga analisou-se o painel SO que possui um enrijecedor longitudinal retangular.

A viga assimétrica possui dois enrijecedores longitudinais retangulares e um enrijecedor trapezoidal e, assim como na viga simétrica, o centro de gravidade do primeiro enrijecedor longitudinal retangular e do enrijecedor trapezoidal estão localizados a 350 mm da mesa superior. Já o segundo enrijecedor longitudinal retangular está localizado a 350 mm do centro de gravidade do primeiro enrijecedor longitudinal retangular. Há seis enrijecedores transversais intermediários com espessura de 20 mm e largura de 122 mm e seis enrijecedores transversais

duplos com espessura de 20 mm e largura de 122 mm localizados da mesma maneira da viga anterior. Fora da área de investigação, 120 mm dos enrijecedores transversais intermediários, a espessura da alma é de 7 mm e assim como na viga anterior a espessura da alma igual a 15 mm foi considerada na parte central da viga. Nesta viga analisou-se o painel UO que possui dois enrijecedores longitudinais retangulares.

Para a viga simétrica, o enrijecedor longitudinal trapezoidal possui largura igual a 90 mm, espessura igual a 10 mm e comprimento igual a 5675 mm. Já para a viga assimétrica, o enrijecedor longitudinal trapezoidal possui largura igual a 100 mm, espessura igual a 10 mm igual e comprimento igual a 5815 mm.

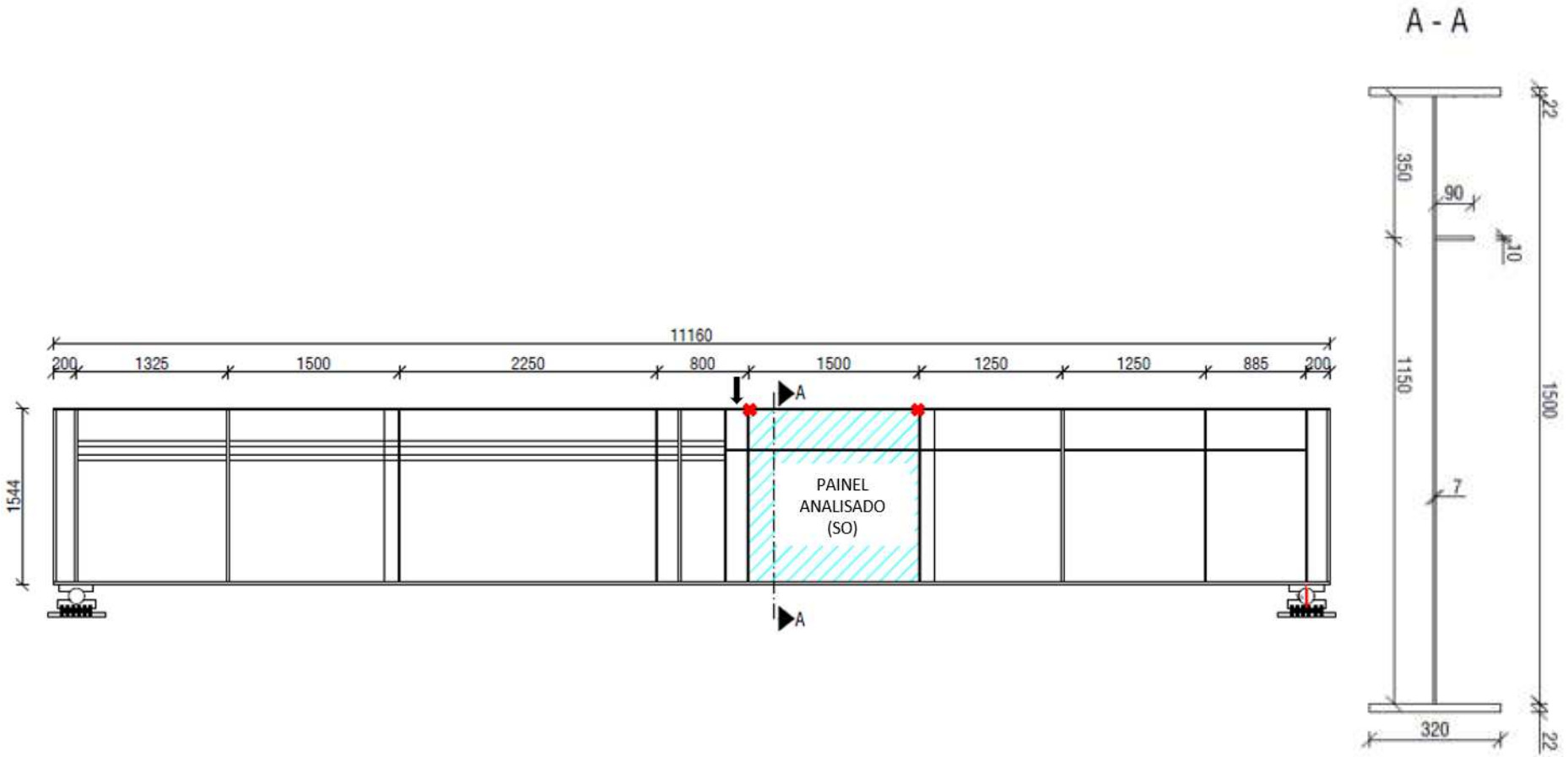
A Tabela 4.3, apresenta as dimensões das mesas, almas e enrijecedores longitudinais retangulares consideradas no plano médio da seção transversal dos dois painéis analisados.

Tabela 4.3 - Dimensões dos painéis considerados para as análises.

Painel	h' **(mm)**	t_w **(mm)**	a **(mm)**	b_{fu} **(mm)**	t_{fu} **(mm)**	b_{fb} **(mm)**	t_{fb} **(mm)**	b_{sl} **(mm)**	t_{sl} **(mm)**
SO	1520,285	7,180	1498,200	320,900	22,290	318,700	22,280	90,000	9,800
UO	1818,010	5,900	1797,500	249,500	20,010	451,200	20,010	100,100	10,230

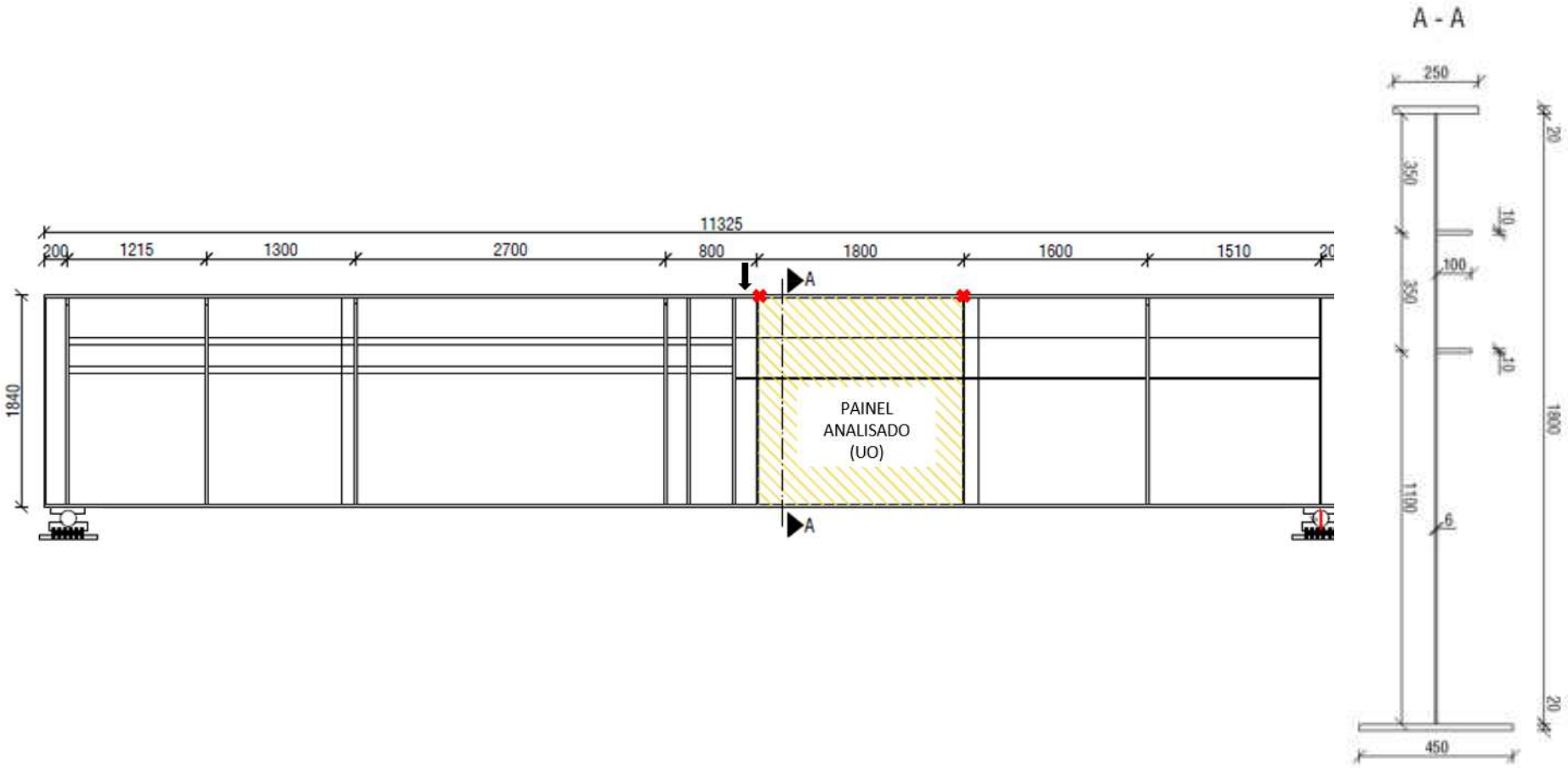
Sendo b_{fb} a largura da mesa inferior; b_{fu} a largura da mesa superior; h' a altura da alma considerando o plano médio da seção transversal; t_{fb} a espessura da mesa inferior; t_{fu} a espessura da mesa superior; t_w a espessura da alma; b_{sl} a largura do enrijecedor longitudinal retangular e t_{sl} a espessura do enrijecedor longitudinal retangular.

Nas Figuras 4.30 e 4.31 é possível visualizar as dimensões das duas vigas, a aplicação do carregamento e a restrição lateral nos painéis analisados



(a) (b)

Figura 4.30 – Viga simétrica: (a) Especificação geométrica do modelo numérico para análise do painel SO e (b) Detalhamento da seção A-A do painel analisado (Sinur, 2011).



(a) (b)

Figura 4.31 – Viga assimétrica: (a) Especificação geométrica do modelo numérico para análise do painel UO e (b) Detalhamento da seção A-A do painel analisado (Sinur, 2011).

4.3.2.2 Material

A partir dos resultados experimentais, as curvas de tensão *versus* deformação reais foram determinadas. O aço possui f_y = 355,00 N/mm², módulo de elasticidade longitudinal (E) = 210,00 GPa e coeficiente de Poisson (υ) igual a 0,3. Na Figura 4.32 é possível observar o comportamento e os valores de cada uma das chapas utilizadas tanto no modelo com um enrijecedor longitudinal retangular quanto no modelo com dois enrijecedores longitudinais retangulares, sendo que esses comportamentos foram adotados no modelo numérico.

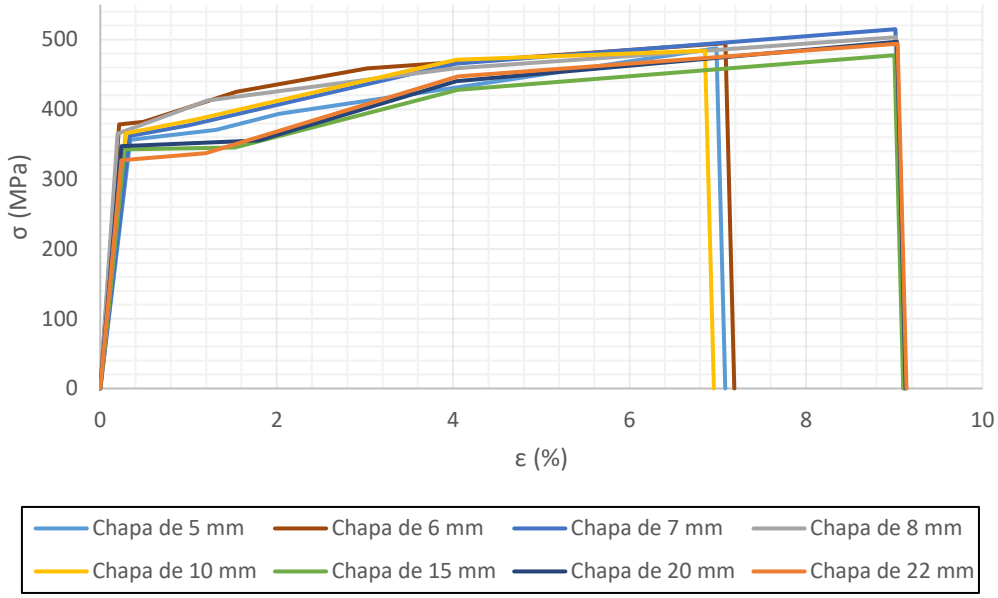


Figura 4.32 - Curvas tensão x deformação adotadas nas análises dos modelos esbeltos (adaptado de Sinur, 2011)

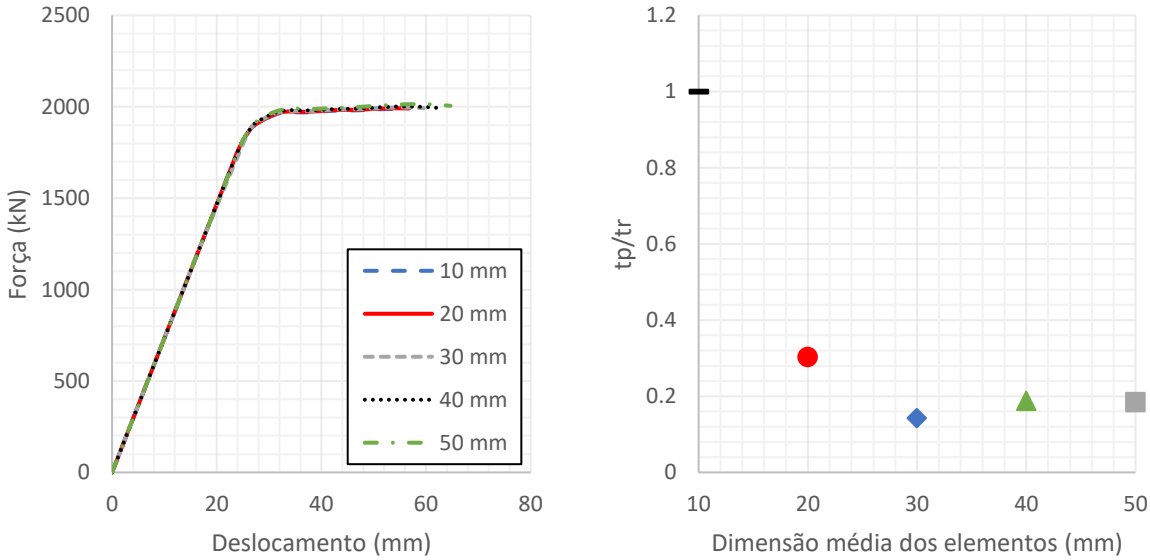
4.3.2.3 Condições de contorno, aplicação do carregamento e imperfeições iniciais

Para o modelo esbelto foram adotadas as mesmas condições de contorno do modelo compacto. Uma vez que o objetivo do estudo é encontrar um único modelo numérico que seja capaz de representar vigas compactas e esbeltas restringidas lateralmente submetidas a flexão. Assim, a aplicação do carregamento distribuído do tipo *pressure* e a restrição do tipo *tie* também foram utilizadas. Porém, o carregamento foi aplicado em um sólido rígido circular o qual possui diâmetro de 200 mm e espessura de 60 mm. Com relação as imperfeições iniciais, de acordo com o que foi explicado no item 3.3.1.1., somente as imperfeições geométricas foram consideradas. E, uma vez que essas imperfeições geométricas foram proporcionadas através da fotogrametria, para a análise da capacidade resistente última adotou-se a imperfeição igual ao

maior deslocamento fora do plano do painel para o primeiro modo de falha. O detalhamento das condições de contorno e aplicação do carregamento constam no Anexo A.

4.3.2.4 Discretização do modelo numérico

Conforme mencionado anteriormente, a partir de um estudo de convergência da malha, Sinur (2011) concluiu que discretizar a viga com elementos finitos com um tamanho médio de, no mínimo, 50 mm proporciona resultados satisfatórios. Dessa maneira, realizou-se um estudo de malha avaliando elementos de aproximadamente 50 mm, 40 mm, 30 mm, 20 mm e 10 mm na viga com o painel SO e o resultado também foi considerado para a viga com o painel UO. Após analisar a curva força versus deslocamento vertical e o tempo de processamento de cada modelo numérico, definiu-se, assim como no modelo compacto a dimensão média dos elementos finitos igual a 20 mm, como pode ser observado na Figura 4.33 e na Figura 4.34.



(a) (b)

Figura 4.33 – Dimensão média dos elementos: (a) impacto nas curvas momento fletor *versus* deslocamento e (b) relação t_p/t_r para cada modelo.

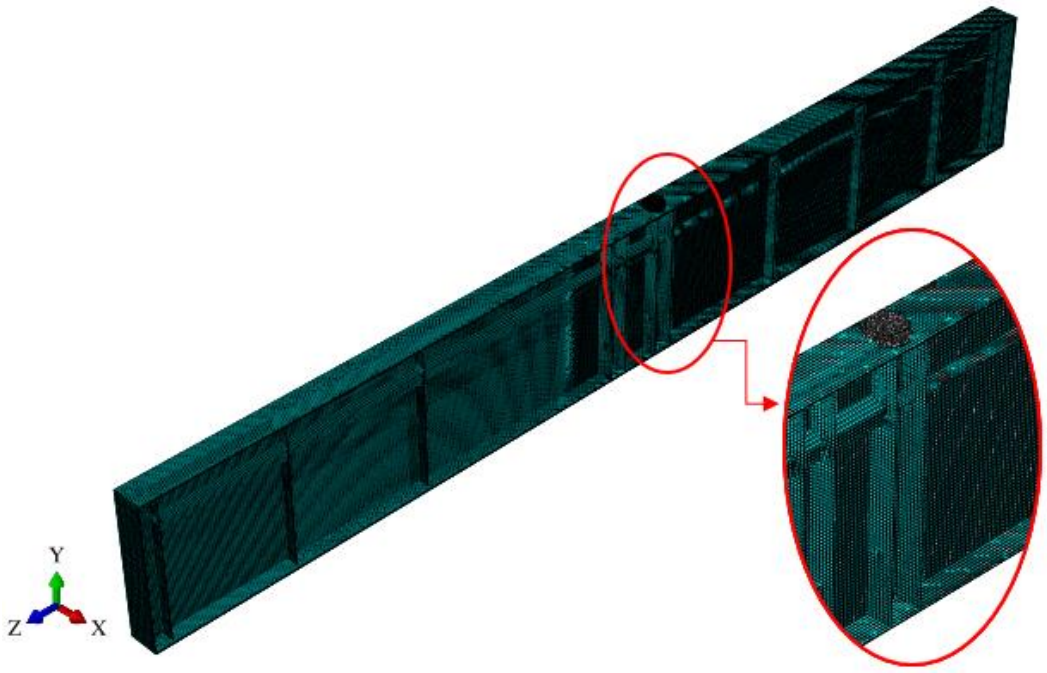


Figura 4.34- Detalhe da malha de elementos finitos adotada no modelo esbelto.

4.3.3 Aferição do modelo numérico

As Figuras 4.35 e 4.36 mostram a comparação das curvas forças versus deslocamento vertical para os painéis analisados e as Figuras 4.37 e 4.38 mostram as deformadas desses painéis.

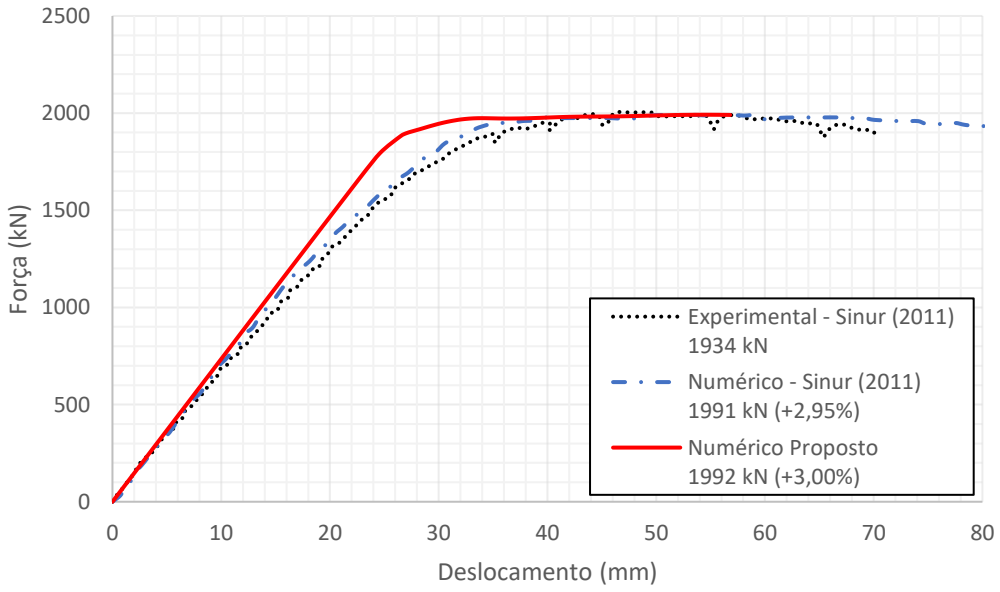


Figura 4.35 – Comparação das curvas força *versus* deslocamento vertical para o painel SO.

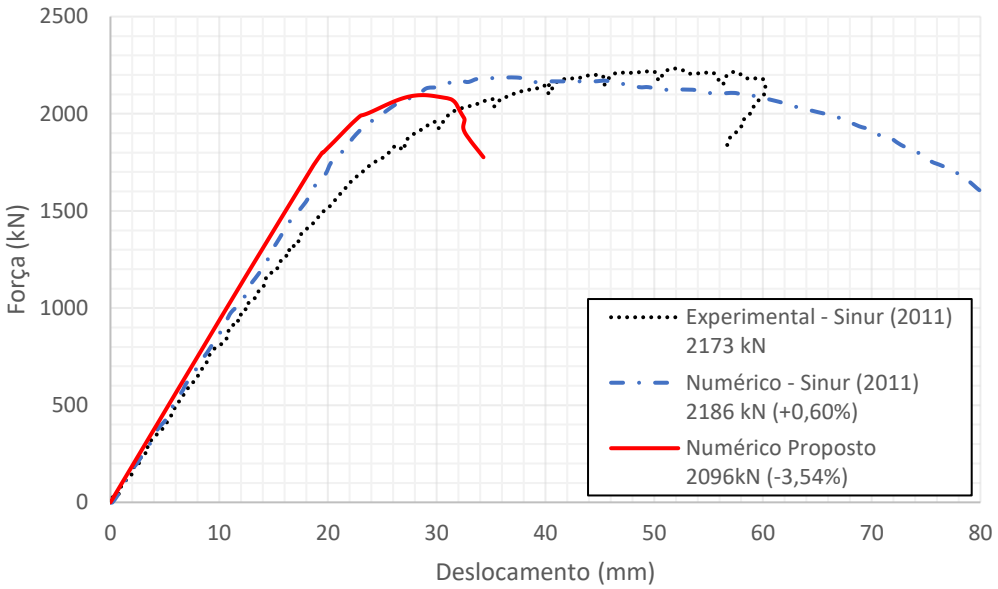
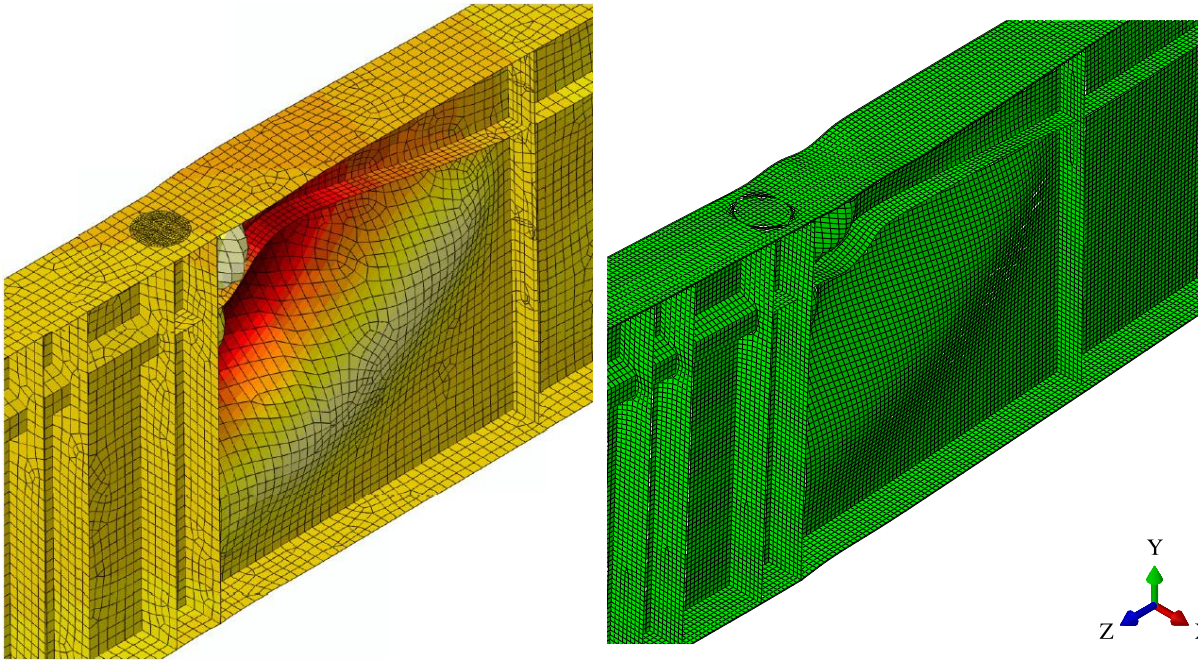
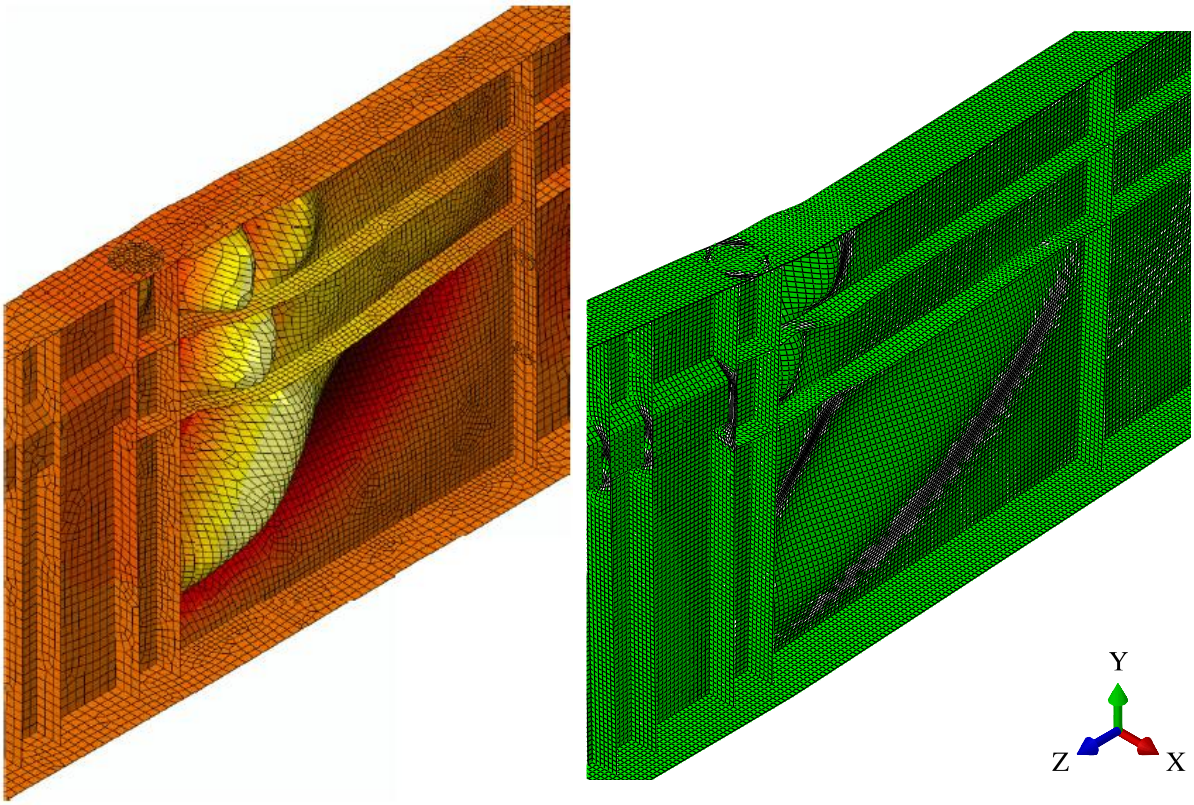


Figura 4.36 - Comparação das curvas força *versus* deslocamento vertical para o painel UO.



(a) (b)

Figura 4.37 – Comparação das deformadas para o painel SO: (a) Deformada obtida por Sinur e (b) Deformada obtida pelo modelo proposto.

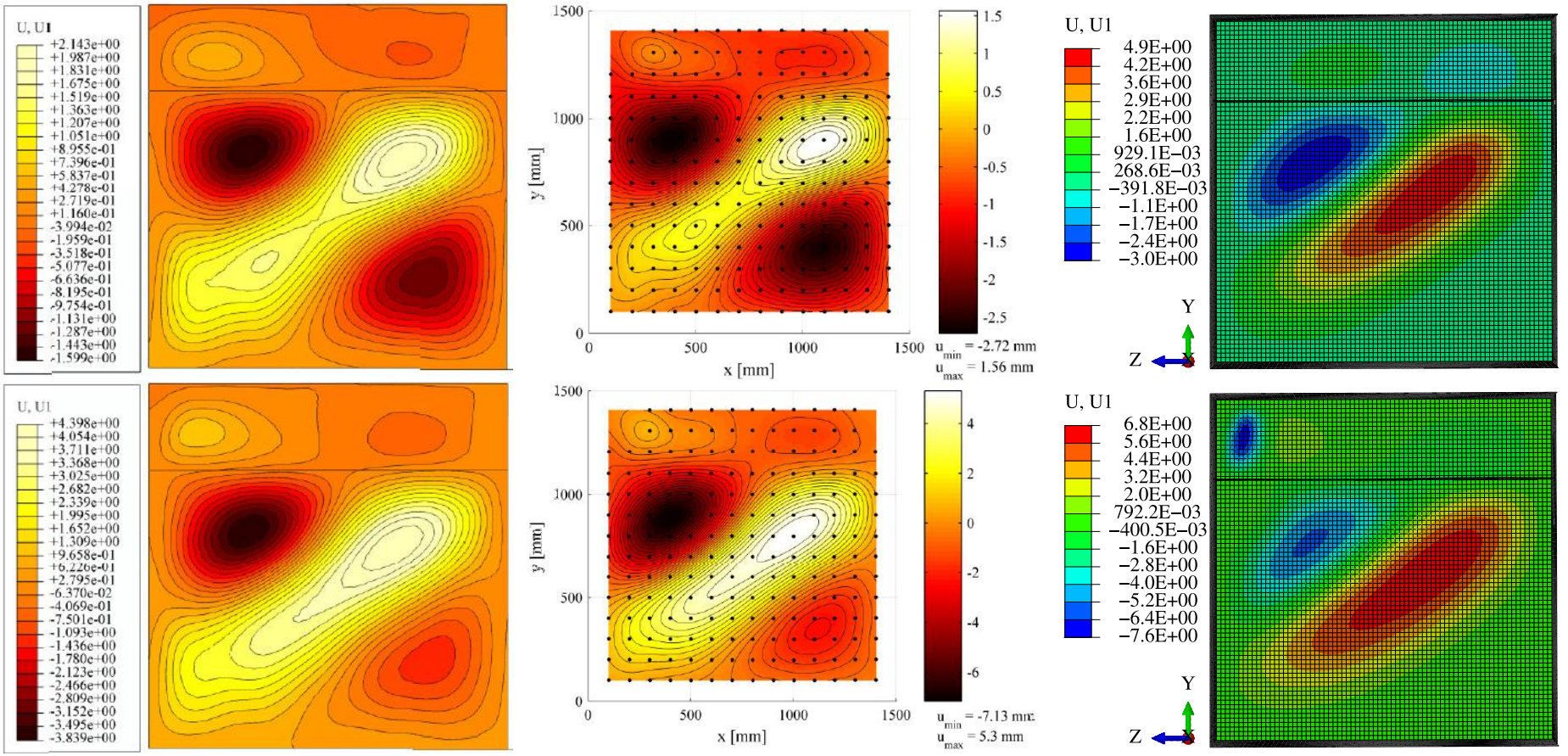


(a) (b)

Figura 4.38 – Comparação das deformadas para o painel UO: (a) Deformada obtida por Sinur e (b) Deformada obtida pelo modelo proposto.

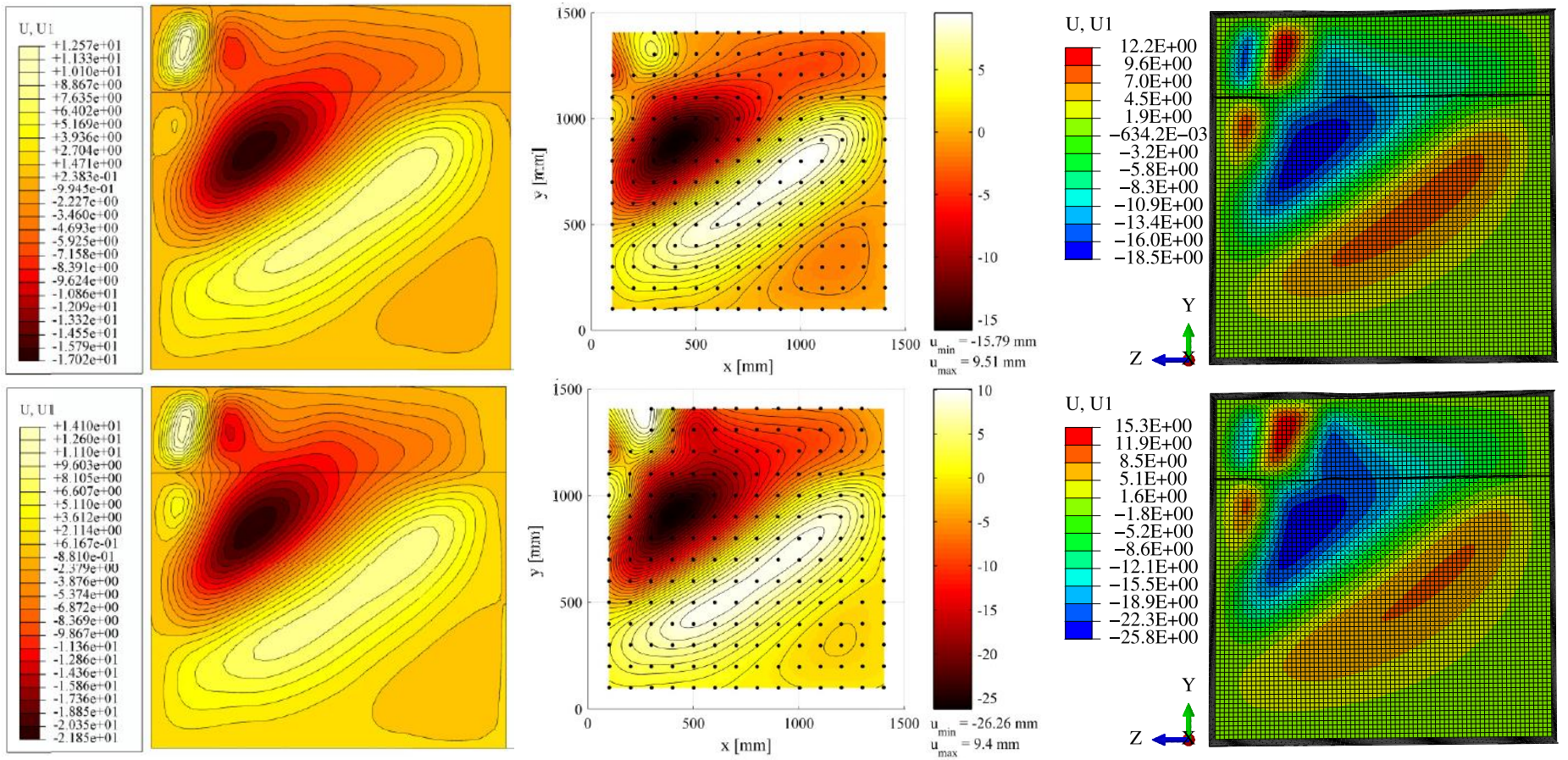
Nas Figuras 4.39 a 4.42 é possível observar a evolução do deslocamento fora do plano e da tensão de von Misses no painel SO: (a) resultados do modelo experimental para deslocamentos verticais de 20,18 mm, 30,18 mm, 45,18 mm e 55,18 mm, respectivamente (Sinur,2011) e (b) resultados do modelo numérico proposto para deslocamentos verticais de 22,70 mm, 30,03 mm, 44,98 mm e 54,07 mm, respectivamente.

Nas Figuras 4.43 a 4.46 têm-se a evolução do deslocamento fora do plano e da tensão de von Misses no painel UO: (a) resultados do modelo experimental para deslocamentos verticais de 20,18 mm, 30,21 mm, 50,24 mm e 60,24 mm, respectivamente (Sinur,2011) e (b) resultados do modelo numérico proposto para deslocamentos verticais de 20,19 mm, 28,93 mm e 34,30 mm.



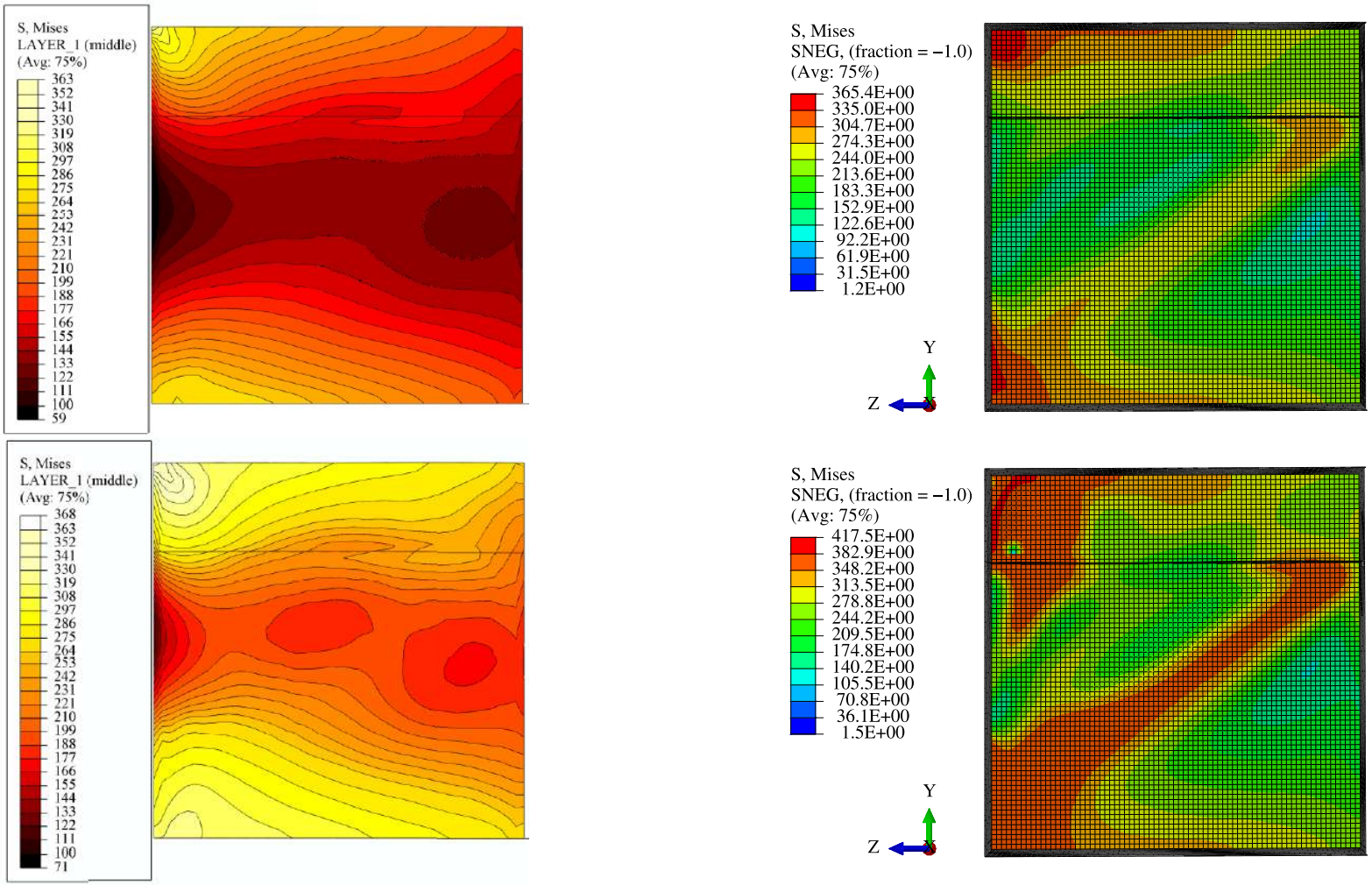
(a) (b) (c)

Figura 4.39 – Evolução do deslocamento fora do plano no painel SO: (a) resultados do modelo numérico (Sinur,2011), (b) resultados do modelo experimental (Sinur,2011) e (c) resultados do modelo numérico proposto (continua)



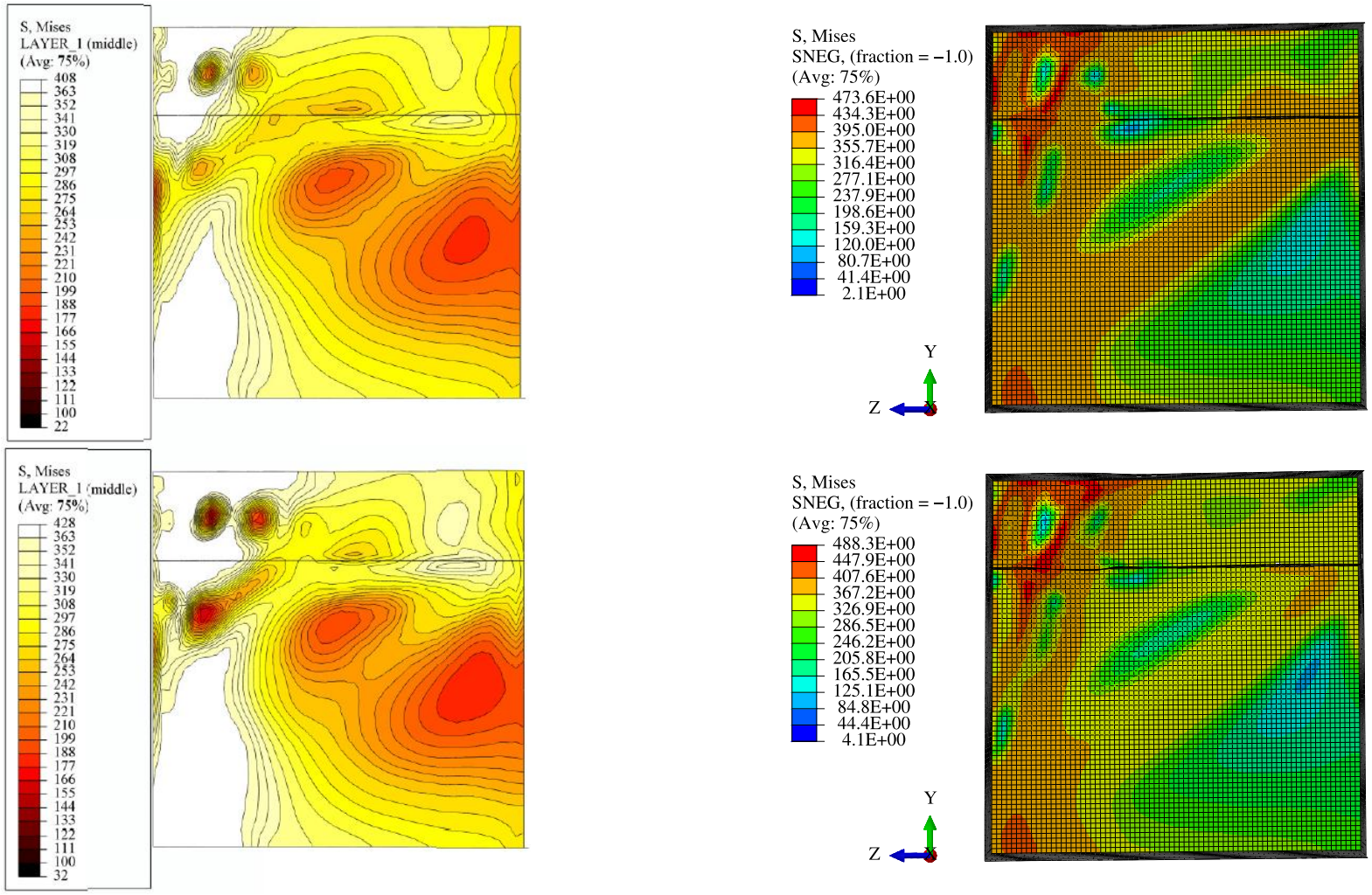
(a) (b) (c)

Figura 4.40 – Evolução do deslocamento fora do plano no painel SO: (a) resultados do modelo numérico (Sinur,2011), (b) resultados do modelo experimental (Sinur,2011) e (c) resultados do modelo numérico proposto (conclusão).



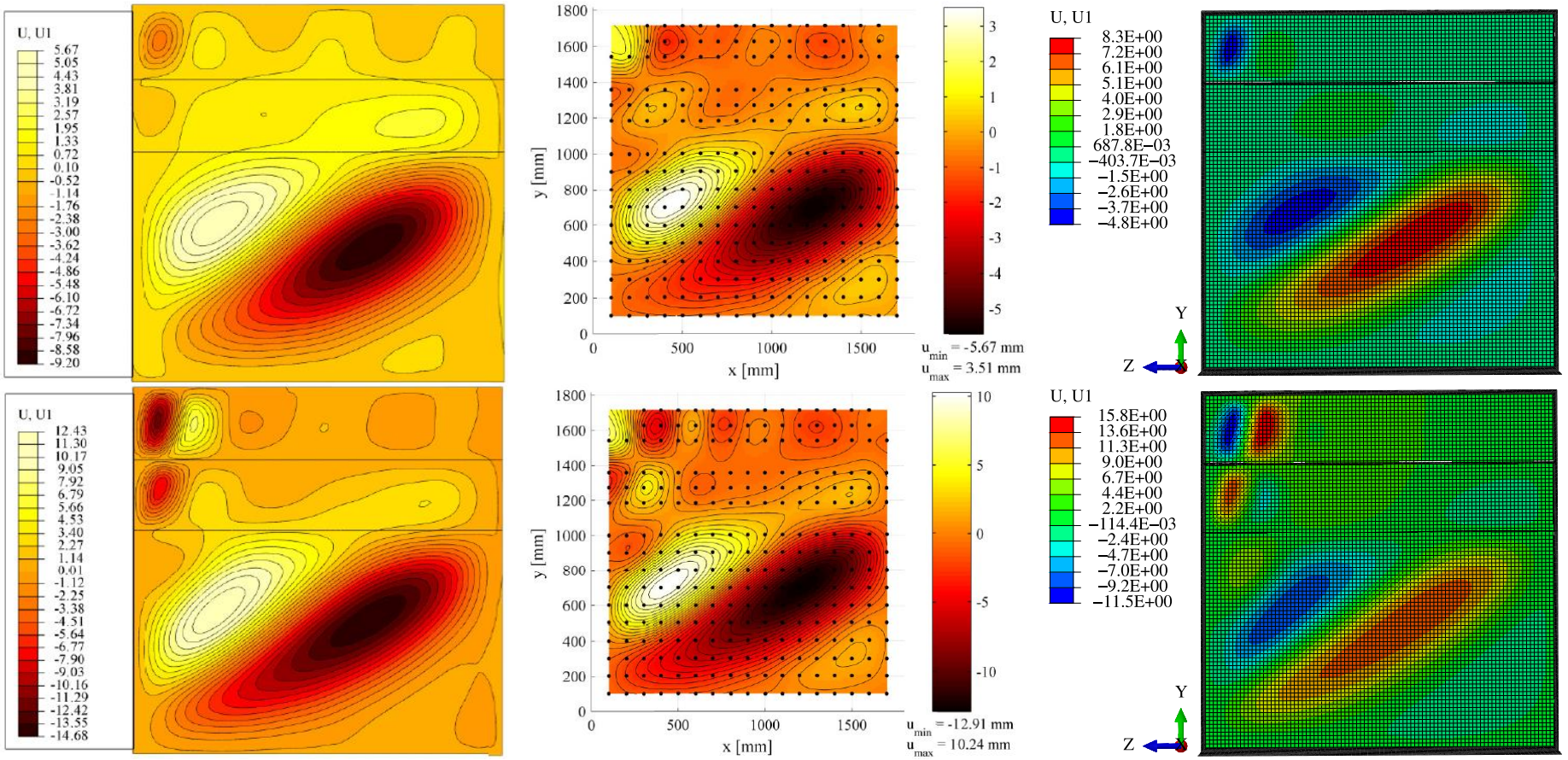
(a) (b)

Figura 4.41 – Evolução da tensão de von Misses no painel SO: (a) resultados do modelo numérico (Sinur,2011) e (b) resultados do modelo numérico proposto (continua)



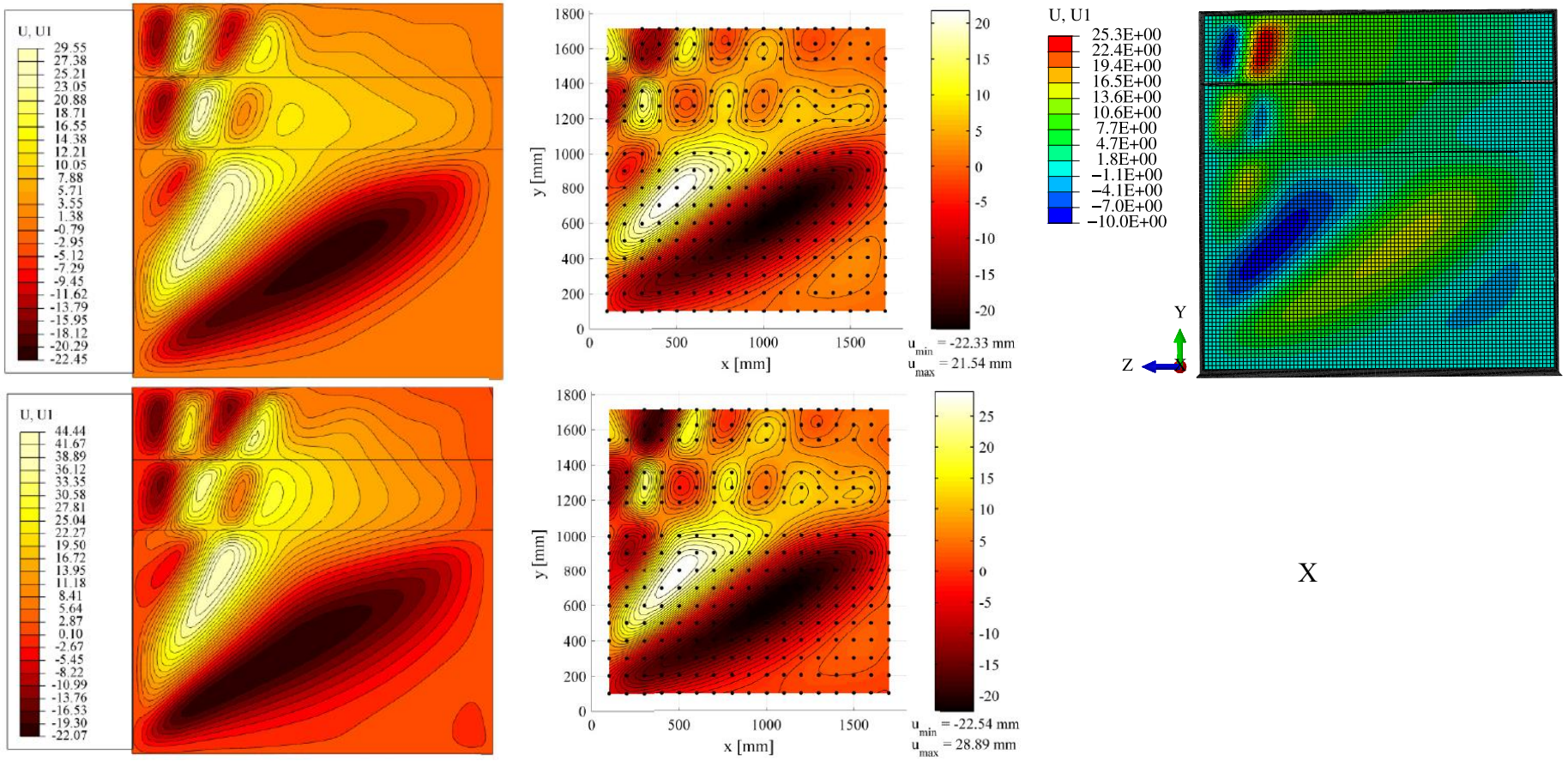
(a) (b)

Figura 4.42 – Evolução da tensão de von Misses no painel SO: (a) resultados do modelo numérico (Sinur,2011) e (b) resultados do modelo numérico proposto (conclusão).



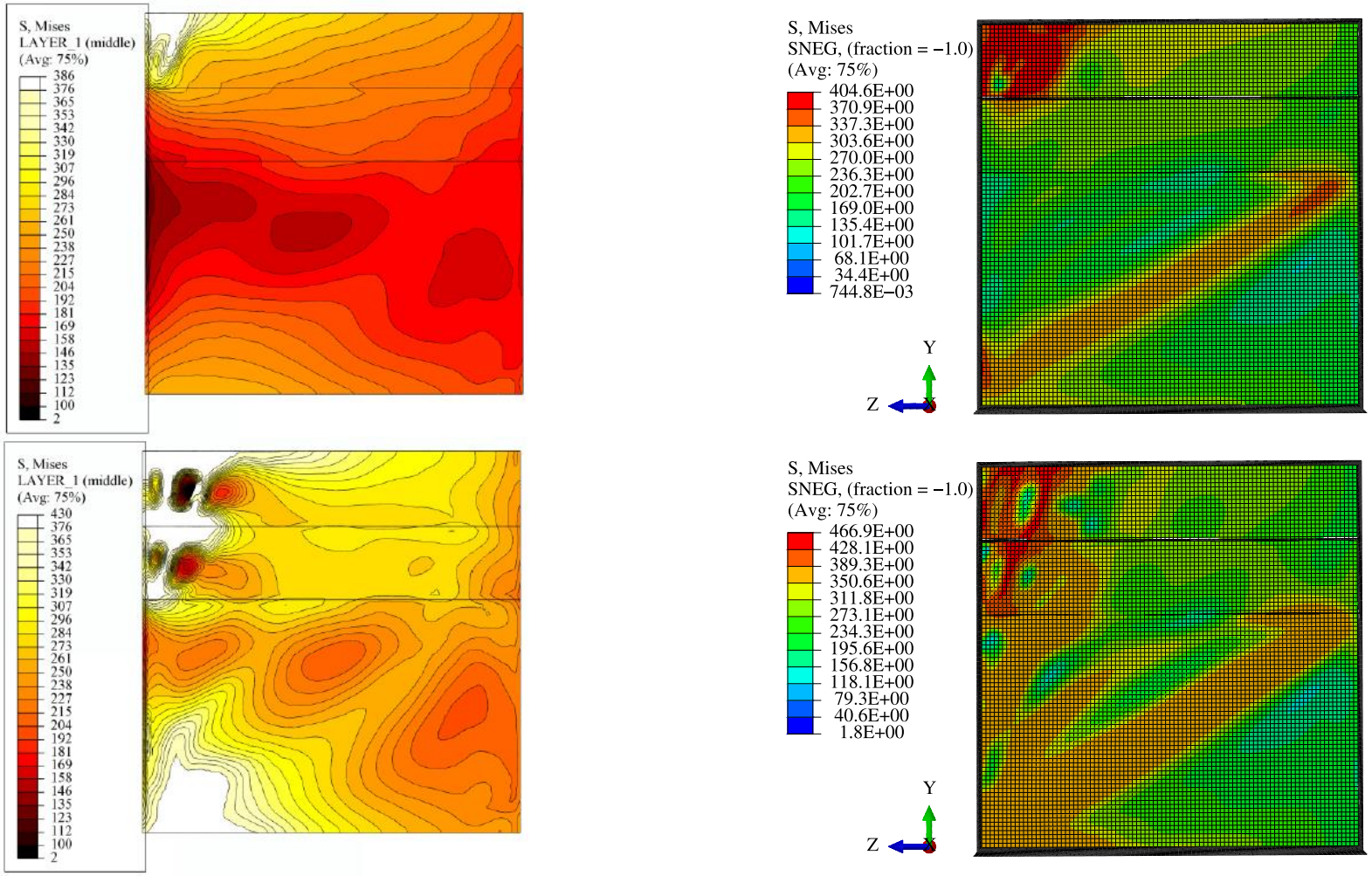
(a) (b) (c)

Figura 4.43 – Evolução do deslocamento fora do plano no painel UO: (a) resultados do modelo numérico (Sinur,2011), (b) resultados do modelo experimental (Sinur,2011) e (c) resultados do modelo numérico proposto (continua)



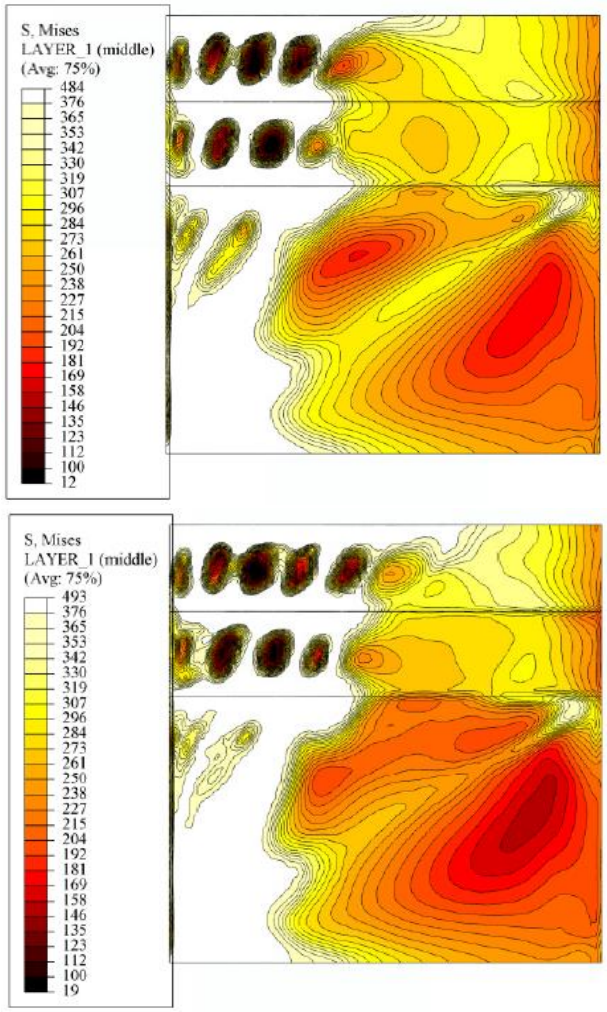
(a) (b) (c)

Figura 4.44 – Evolução do deslocamento fora do plano no painel UO: (a) resultados do modelo numérico (Sinur,2011), (b) resultados do modelo experimental (Sinur,2011) e (c) resultados do modelo numérico proposto (conclusão).

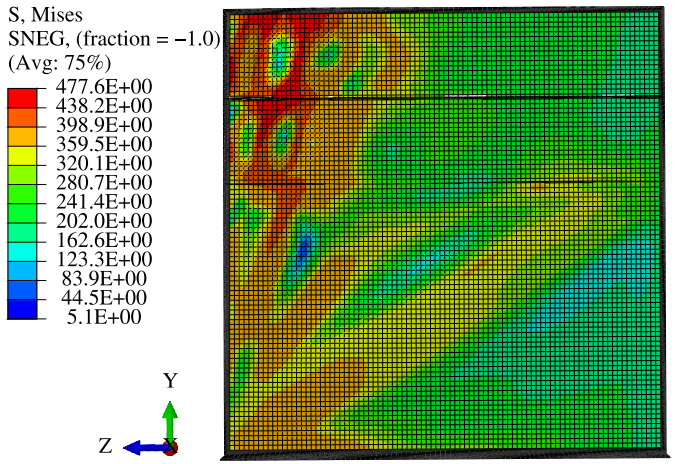


(a) (b)

Figura 4.45 – Evolução da tensão de von Misses no painel UO: (a) resultados do modelo numérico (Sinur,2011) e (b) resultados do modelo numérico proposto (continua)



(a)



(b)

Figura 4.46 – Evolução da tensão de von Misses no painel UO: (a) resultados do modelo numérico (Sinur,2011) e (b) resultados do modelo numérico proposto (conclusão).

Obteve-se uma boa concordância entre os estudos experimentais e numéricos do autor com os resultados numéricos do modelo proposto. Além disso, ao comparar as resistências últimas obtidas no modelo proposto com a resistência última de acordo com a EN 1993-1-5 (veja Capítulo 3) tem-se $F_{EN\ 1993-1-5} = 1792\ kN$ para o painel SO e $F_{EN\ 1993-1-5} = 1746\ kN$ para o painel UO. Isso significa que, com relação às cargas últimas, o modelo proposto apresentou um erro relativo menor que 4% em ambos os casos quando comparado com os resultados experimentais do autor. Os resultados numéricos do autor apresentaram uma diferença de 11,10% e 25,20% para os painéis SO e UO, respectivamente, que são bastante similares aos resultados encontrados pelo modelo proposto, que foram de 11,16% e 20,05%, respectivamente.

Os modos de falha são similares e a evolução do deslocamento fora do plano e da tensão de von Misses nas análises também apresentaram uma convergência com os resultados do autor. A maior divergência está nos deslocamentos fora do plano encontrados no painel UO de acordo com os deslocamentos verticais, o que explica o fato do modelo numérico ter abortado antes do previsto. A explicação para esse acontecimento é também uma das pautas do trabalho de Sinur (2011), pois no seu estudo de convergência de malha foi detectado que os deslocamentos fora do plano são o parâmetro mais sensível da análise. Portanto, o modelo proposto comportou-se de maneira satisfatória.

5 ESTUDO DA SENSIBILIDADE DE PARÂMETRO

Este capítulo tem como objetivo analisar a influência da variação das resistências ao escoamento das mesas e da alma no comportamento de vigas compactas, painéis esbeltos simétricos com um enrijecedor retangular longitudinal e painéis esbeltos assimétricos com dois enrijecedores retangulares longitudinais. Para os painéis esbeltos, a razão f_{yf}/f_{yw} não deve ultrapassar o valor de 2,0, limite este recomendado para estudo de vigas híbridas de acordo com a Equação (2.5).

Outro objetivo desse capítulo é avaliar os custos de produção dos elementos estruturais híbridos, com o seu posterior impacto econômico no mercado. Assim, com a auxílio dos Engs. Leandro Antunes e Júlia Torres da Arcelor Mittal obtiveram-se as relações preço/kg *versus* espessura para as seguintes classes de aços estruturais: S235, S355 e S460. Admitiu-se que o número das classes é a resistência ao escoamento mínima.

A Tabela 5.1 e a Tabela 5.2 mostram essa relação com o valor do preço/kg faturado de Contagem-MG para entregar em São Paulo-SP com 12% de ICMS incluído e 5% de IPI a incluir. É perceptível o aumento do preço do aço de acordo com as tabelas o que, segundo resultados levantados pela S&P Global Platts, se deve a uma escalada que teve início em junho de 2020, quando da reativação da economia e do comércio pós-pico da pandemia da COVID-19 no Brasil. Nesse momento, esperava-se uma retração forte na demanda de aço, o que fez com que as siderúrgicas reduzissem a produção em um momento em que já havia pouco estoque. Quando houve um aumento da demanda havia pouca oferta, o que acarretou em aumentos de 60% a 70% nos preços dependendo do produto. Essa tendência não se reverteu em 2021. Importante ressaltar que essas tabelas serviram como base para a estimativa da relação preço/kg para as espessuras dos modelos analisados e a densidade adotado do aço é 7850 kg/m³.

Dessa maneira, neste capítulo a viga C3 do modelo compacto e os painéis SO e UO são analisados. Além disso, são utilizadas as classes de aços estruturais mencionadas no parágrafo anterior. Admitiu-se que o número das classes é a resistência ao escoamento mínima. Adotou-se o comportamento elasto-plástico dos aços estruturais apresentado na Figura 5.1. Tomaram-se o módulo de elasticidade longitudinal (E) = 200 GPa e o coeficiente de Poisson (υ) igual a 0,3. A deformação dos trechos foi definida de acordo com o trabalho de Almeida (2012) de maneira que a deformação $10\varepsilon_y$ define o fim do patamar de escoamento, o qual foi assumida

igual a 1,01f_y e a tensão última (f_u) corresponde a 100ε_y. Com relação às imperfeições, adotou-se a mesma distribuição de tensões residuais que Castro e Silva (2006), ver Anexo B, e a imperfeição geométrica igual a $L/1500$. A Tabela 5.3 e a Figura 5.1 representam os resultados das resistências últimas da viga C3.

As composições de cada um dos modelos foram:

1) Alma e enrijecedores: S235 / Mesas: S235
2) Alma e enrijecedores: S355 / Mesas: S355
3) Alma e enrijecedores: S460 / Mesas: S460
4) Alma e enrijecedores: S235 / Mesas: S355
5) Alma e enrijecedores: S235 / Mesas: S460
6) Alma e enrijecedores: S355 / Mesas: S460

Tabela 5.1 - Relação entre espessura x preço x classes dos aços estruturais em 31/07/2020 (Arcelor Mittal,2020)

Preço/kg			
Espessura (mm)	**Classe do aço estrutural**		
	S235	**S355**	**S460**
1,50	R$ 4,87	R$ 4,87	R$ 5,11
1,80	R$ 4,66	R$ 4,66	R$ 4,89
2,00 - 2,25	R$ 4,62	R$ 4,62	R$ 4,85
2,65 - 3,00 - 3,35	R$ 4,52	R$ 4,52	R$ 4,74
3,75 - 4,25 - 4,75 - 6,30 - 8,00	R$ 4,45	R$ 4,45	R$ 4,66

Tabela 5.2 - Relação entre espessura x preço x classes dos aços estruturais em 20/05/2021 (Arcelor Mittal,2021)

Preço/kg			
Espessura (mm)	**Classe do aço estrutural**		
	S235	**S355**	**S460**
1,50	R$ 10,29	R$ 10,29	R$ 10,77
1,80	R$ 9,72	R$ 9,72	R$ 10,18
2,00 - 2,25	R$ 9,72	R$ 9,72	R$ 10,18
2,65 - 3,00 - 3,35	R$ 9,65	R$ 9,65	R$ 10,10
3,75 - 4,25 - 4,75 - 6,30 - 8,00	R$ 9,21	R$ 9,21	R$ 9,64

Tabela 5.3 - Comparação da resistência última entre as análises da viga C3.

Modelo	$M_{ul\ (Modelo\ proposto)}$ **(kNm)**	$M_{p\ (teórico\ )}$ **(kNm)**
1	454,79	467,47
2	677,37	706,18
3	864,96	915,05
4	583,87	617,46
5	605,92	748,71
6	801,79	837,42

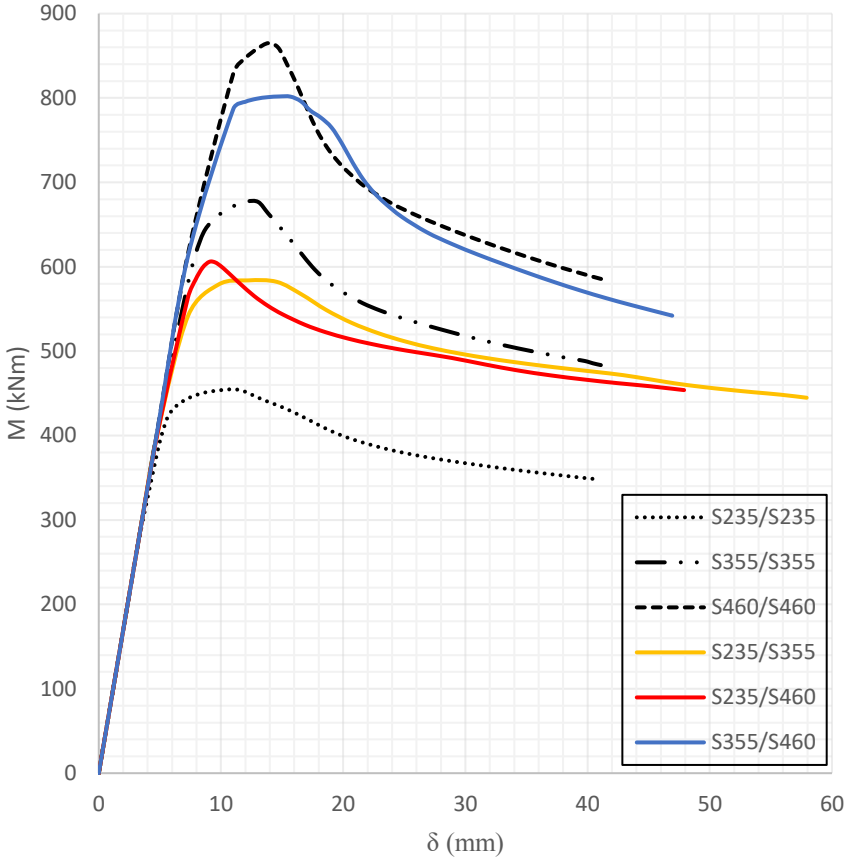


Figura 5.1 - Momento *versus* deflexão para a viga C3 de acordo com classes de aços utilizadas.

A maioria dos modelos apresentou uma diferença de, no máximo, 6,00% quando comparados com os resultados teóricos de momento plástico. Somente o modelo 5 apresentou uma diferença de 19,07% o qual pode ser explicado pelo fato de que a razão f_{yf}/f_{yw} está bastante próxima do limite recomendado para estudo de vigas híbridas. Já a Figura 5.2 representa os resultados das resistências últimas dos painéis SO e UO.

Com relação ao preço, a partir das tabelas de referência nota-se que o preço/kg das classes S235 e S355 são iguais, enquanto que o preço/kg da classe S460 é superior em 5% em relação ao preço/kg das classes S235 e S355. Considerando somente o preço/kg dos aços fornecidos pela Arcelor Mittal, ou seja, desconsiderando os valores de fabricação do perfil I, têm-se as Tabelas 5.4, 5.5 e 5.6 relacionando os seguintes custos aproximados para os modelos.

Considerando os casos nos quais há variação de preço de acordo com a tabela fornecida pela Arcelor Mittal, tem-se um aumento de resistência de até 33% enquanto que o aumento no custo representa 3% quando são comparadas as vigas híbridas e vigas homogêneas.

Ainda, tomou-se como exemplo os modelos 1 e 5 do painel UO para demonstrar o que foi explicado no item 2.3.8.1 deste trabalho. Conforme esperado, na Figura 5.3 é mostrado que na seção híbrida o escoamento tem início na fibra mais afastada da alma. Para essa análise tem-se $f_{yw} = 235{,}58\ MPa$.

Tabela 5.4 – Custo aproximado e peso equivalente S235 para análises da viga C3.

Modelo	Custo aproximado	Peso (kg)	Peso Equivalente S235 (kg)
1, 2 e 4	R$ 17.331,78	1146,18	1146,18
3	R$ 18.138,03	1146,18	1203,48
5 e 6	R$ 17.400,65	1146,18	1151,56

Tabela 5.5 – Custo aproximado e peso equivalente S235 para análises do painel SO.

Modelo	Custo aproximado	Peso (kg)	Peso Equivalente S235 (kg)
1, 2 e 4	R$ 13.959,91	359,77	359,77
3	R$ 14.379,36	359,77	377,76
5 e 6	R$ 14.162,50	359,77	368,06

Tabela 5.6 – Custo aproximado e peso equivalente S235 para análises do painel UO.

Modelo	Custo aproximado	Peso (kg)	Peso Equivalente S235 (kg)
1, 2 e 4	R$ 15.093,77	449,90	449,90
3	R$ 16.522,97	449,90	472,39
5 e 6	R$ 15.546,78	449,90	459,79

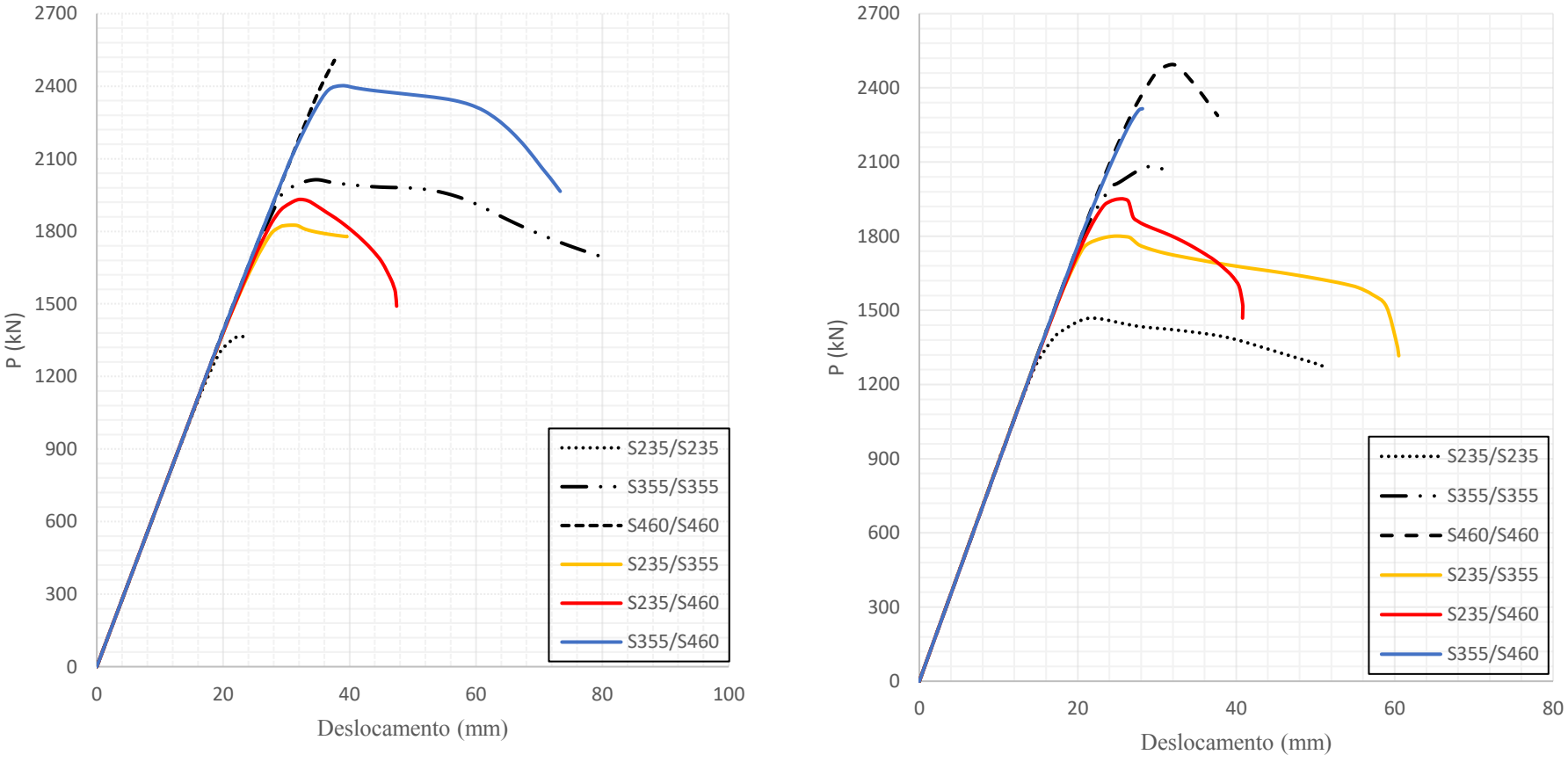


Figura 5.2 - Carga *versus* deslocamento de acordo com classes de aços utilizadas: (a) painel SO e (b) painel UO.

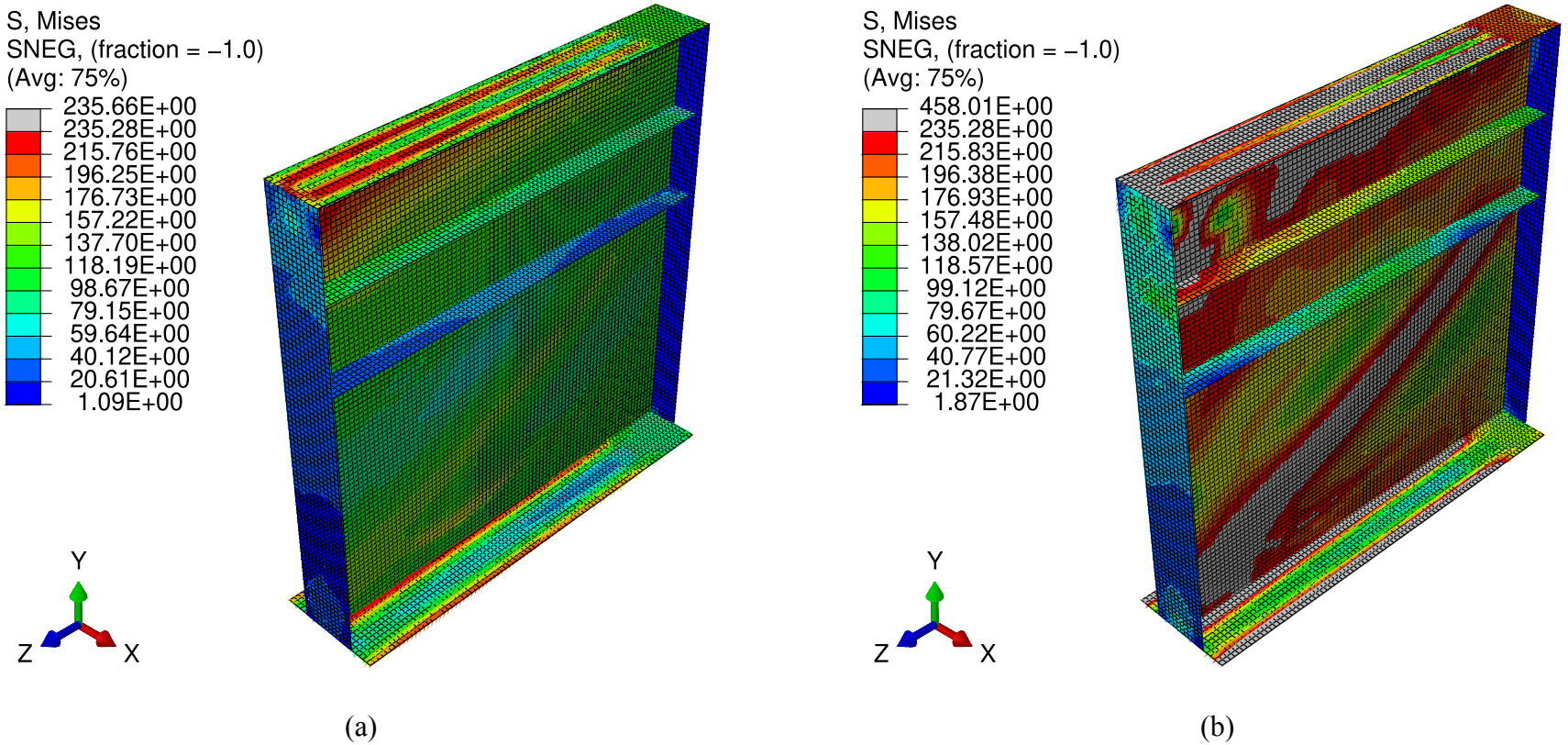


Figura 5.3 - Tensões de von Mises: (a) painel UO homogêneo e (b) painel UO híbrido.

6 CONCLUSÃO

6.1 Considerações gerais

No presente trabalho foi realizado um estudo numérico baseado em vigas biapoiadas contidas lateralmente com perfil I fletido em relação ao eixo de maior inércia. A validação dos três modelos numéricos fundamentados no Método dos Elementos Finitos foi elaborada no programa comercial ABAQUS. O primeiro modelo se refere às vigas compactas do estudo de Shokouhian (2014) e os outros dois relacionam-se à painéis esbeltos com um e dois enrijecedores longitudinais a partir do estudo de Sinur (2011). Os três modelos numéricos possuem às mesmas características de aplicação de carga e condições de contorno, para que pudesse ser possível atingir um modelo geral único para as vigas e painéis.

O software ABAQUS foi utilizado para a validação do modelo numérico e para a análise de sensibilidade de parâmetro. Realizou-se análise de flambagem elástica e, na sequência foi realizada análise da capacidade resistente última, com a aproximação pelo primeiro modo de flambagem. Nessa última análise, para o estudo de variação de parâmetro considerou-se a introdução das tensões residuais adotada por Castro e Silva (2006) e a imperfeição geométrica igual a L/1500.

Em comparação com o trabalho de vigas compactas de Shokouhian (2014), o modelo numérico proposto apresentou concordância com os modos de falhas obtidos pelo autor. Com relação aos momentos resistentes últimos, os resultados obtidos se mostraram bastante próximos aos de Shokouhian (2014) para a maioria das vigas analisadas. Exclusivamente na análise da viga H2 encontrou-se um erro relativo de 16%, o que pode ser explicado pela diferença de geometria dessa viga. Por fim, com esse estudo, concluiu-se que a capacidade resistente das vigas híbridas é em torno de 14% a 28% maior que as determinadas para as vigas convencionas.

O modelo numérico para os painéis esbeltos proposto foi confrontado com os resultados de Sinur (2011). O painel SO possui seção transversal simétrica com um enrijecedor retangular e o painel UO seção transversal assimétrica com dois enrijecedores retangulares. Como resultado, obteve-se um erro relativo menor que 4% em ambos os casos com relação às cargas últimas. Além disso, as deformadas, a evolução do deslocamento fora do plano e da tensão de von Mises convergiram com os resultados do autor. Nota-se que houve uma maior divergência relacionada

aos deslocamentos fora do plano encontrados na análise do painel UO, o que é explicado pelo fato de que os deslocamentos fora do plano são o parâmetro mais sensível da análise.

Com a finalidade de mostrar que as vigas híbridas resistem a um esforço maior com baixo aumento no custo, na análise de sensibilidade de parâmetro foi avaliada a influência da variação da resistência ao escoamento da mesa. Concluiu-se que as vigas híbridas, no geral, possuem uma resistência 33% maior do que as vigas homogêneas correspondentes, enquanto que o aumento no custo foi de apenas 3%. Além disso, foi possível observar o início do escoamento nas duas análises e verificar que para as vigas homogêneas o escoamento tem início na fibra mais externa de toda a seção transversal e na viga híbrida o escoamento tem início na fibra mais afastada da alma.

Por fim, pode-se dizer que este estudo cumpriu com os objetivos propostos e resultou na elaboração de um modelo numérico capaz de representar vigas compactas e esbeltas submetidas à flexão, disponibilizando rotinas de programação elaborada na linguagem Python para a criação desses modelos, apresentadas nos Apêndices B e C.

6.2 Sugestões para trabalhos futuros

Algumas sugestões para trabalhos futuros são:

- ✓ Análise da sensibilidade da variação de outros parâmetros, como por exemplo, da geometria.
- ✓ Investigar o comportamento da viga híbrida caso o aço de maior resistência esteja localizado na alma pois a probabilidade de ocorrência de flambagem na alma ou na mesa podem estar combinadas com o local do emprego do aço de alta resistência.
- ✓ Avaliar o comportamento do modelo numérico proposto para os outros dois painéis das vigas esbeltas, os quais possuem enrijecedor trapezoidal longitudinal.
- ✓ Comparar o custo de uma viga híbrida e uma viga homogênea que possuem a mesma resistência última.
- ✓ Analisar a FLT de vigas híbridas.

REFERÊNCIAS

AASHTO LRFD Bridge Design Specifications. (2014). Washington: American Association of State Highway and Transportation Officials.

Abuyounest, S., & Adeli, H. (1987). Optimization of hybrid steel plate girders. Computers & Structures, 27, 575-582.

Adeli, H., & Phan, K. (1986). Interactive computer-aided design of non-hybrid and hybrid plate girders. Computers & Structures, 22, 267-289.

Ajeesh, S. S. (2011). Shear Resistance of Hybrid Plate Girder. INSDAG'S STEEL IN CONSTRUCTION, 12, 33-44.

Axhag, F. (1998). Plastic design of slender steel bridge girders. Doctoral Thesis. Luleå University of Technology.

Azizinamini, A., Hash, B., Yakel, A. J., & Farimani, R. (2007). Shear Capacity of Hybrid Plate Girders. Journal of Bridge Engineering - ASCE, 535-543.

Ban, H., Shi, G., Bai, Y., Shi, Y., & Wang, Y. (2013). Residual Stress of 460 MPa High Strength Steel Welded I Section: Experimental Investigation and Modeling. International Journal of Steel Structures, 691-705.

Barker, M. G. (2005). Shear Tests of High Performance Steel Hybrid Girders.

Barker, M. G., & Schrage, S. D. (2000). High-Performance Steel Bridge Design and Cost Comparisons. Transportation Research Record, 33-39.

Barker M., Hurst A., White D., Tension Field Action in Hybrid Steel Girders Engineering Journal, AISC, Vol. 39, Issue 1.pp. 52-62. 2002

Barros, R., Granado, J., Rio, J., & Castro, J. M. (2013). Development of structural applications using web based technologies. IX Congresso de Construção Metálica e Mista & I Congresso Luso-Brasileiro de Construção Metálica Sustentável.

Barth, K. E., Righman, J. E., & Freeman, L. B. (2007). Assessment of AASHTO LFRD Specifications for Hybrid HPS 690W Steel I-Girders. Journal of Bridge Engineering - ASCE.

Barth, K. E., Yang, L., & Righman, J. (2007). Simplified Moment Redistribution of Hybrid HPS 485W Bridge Girders in Negative Bending. Journal of Bridge Engineering - ASCE, 456-466.

Beg, D., Kuhlmann, U., Davaine, L., & Braun, B. (2010). Design of Plated Structures. ECCS - European Convention for Constructional Steelwork.

Biscaya, A., Pedro, J. O., & Kuhlmann, U. (2019). Experimental and numerical studies on the M-V-N interaction of longitudinally stiffened steel I-girders. Annual Stability Conference Structural Stability Research Council. St. Louis, Missouri.

Bitar D., Résistance a la flexion des poutres hybrides à section en I. Construction Métallique, vol. 40, n o 2, pp. 77-92 2003 (in french)

Carskaddan P. Shear buckling of unstiffened hybrid beams. Journal of the structural division. ASCE; Vol. 94. Issue ST8, pp. 1965–1990. 1968

Castro e Silva, A. L. (2006). Análise numérica não linear da flambagem local de perfis de aço estrutural submetidos à compressão uniaxial. Belo Horizonte: Tese de Doutorado.

Chacón, R. F. (2009). Resistance of Transversally Stiffened Hybrid Steel Plate Girders to Concentrated Loads. 235. Barcelona: Doctoral Thesis.

Chacón, R. F. (2014). Vigas armadas híbridas de acero. Estado del conocimiento. Revista Ciencia e Ingeniería, 95-102.

Chacón, R., & Bock, M. R. (2011). Longitudinally stiffened hybrid steel plate girders subjected to patch loading. Journal of Constructional Steel Research, 1310-1324.

Chacón, R., & Mirambell, E. R. (2010). Hybrid steel plate girders subjected to patch loading, Part 1: Numerical study. Journal of Constructional Steel Research, 695-708.

Chacón, R., & Mirambell, E. R. (2010). Hybrid steel plate girders subjected to patch loading, Part 2: Design proposal. Journal of Constructional Steel Research, 709-715.

Chacón, R., & Mirambell, E. R. (2013). Transversally stiffened plate girders subjected to patch loading. Part 1. Preliminary study. Journal of Constructional Steel Research, 483-491.

Chacón, R., & Rojas-Blonval, J. E. (2015). Evaluación de la resistencia a abolladura por cortante de vigas armadas híbridas de acero según la norma venezolana COVENIN 1618:1998. Informes de la Construcción, 67.

Chacón, R., Bock, M., Mirambell, E., & Real, E. (2012). Hybrid steel plate girders subjected to path loading. Steel Construction, 3-9.

COMBRI. (2008). Competitive steel and composite bridges by innovative steel-plated structures. European Commission.

Eurocode 3 - Design of steel structures - Part 1-1: General rules and rules for buildings. (2005).

Eurocode 3 - Design of steel structures - Part 1-12: Additional rules for the extension of EN 1993 up to steel grades S700. (2007).

Eurocode 3 - Design of steel structures - Part 1-5: Plated structural elements. (2006).

Fakury, R. H., Castro e Silva, A. L., & Caldas, R. B. (2016). Dimensionamento de Elementos Estruturais de Aço e Mistos de Aço e Concreto. São Paulo: Pearson.

Fenkel, J. P., Rizos, D. C., & Ziehl, P. H. (2007). Structural performance and design evaluation of HPS 70W bridge girders. Journal of Constructional Steel Research, 909-921.

Filho, J. O. P., Tankova, T., Carvalho, H., Martins, C. & Simões da Silva, L. (2022). Experimental and numerical flexural buckling resistance of high strength steel columns and beam-columns. Engineering Structures 265 (1-3): 114414

Frost R., Schilling C., Behaviour of hybrid beams subjected to static loads. Journal of the Structural Division ASCE, Vol.90, ST3, 55–86. 1964

Ghadami, A., & Broujerdian, V. (Advances in Structural Engineering). Flexure-shear interaction in hybrid steel I-girders at ambient and elevated temperatures. 2018, 1-16.

Gogou, E. (2012). Use of High Strength Steel Grades for Economical Bridge Design. Amsterdam: Master Thesis Study.

Greco, N., & Earls, C. J. (2003). Structural Ductility in Hybrid High Performance Steel Beams. Journal of Structural Engineering - ASCE, 1584-1595.

Ito, M., Nozaka, K., Shirosaki, T., & Yamasaki, K. (2005). Experimental Study on Moment-Plastic Rotation Capacity of Hybrid Beams. Journal of Bridge Engineering - ASCE.

Johansson, B., & Collin, P. (2005). Eurocode for High Strength Steel and Applications in Construction.

Kulkarni, A. S., & Gupta, L. M. (2017). Experimental Investigation on Flexural Response of Hybrid Steel Plate Girder. KSCE Journal of Civil Engineering.

Lalthazuala, R., & Singh, K. D. (2019). Investigations on structural performance of hybrid stainless steel I-beams based on slenderness. Thin-Walled Structures, 197-212.

Lew, H. S., & Toprac, A. A. (1967). Static Tests on hybrid plate girders. Center for highway research, The University of Texas, Austin, Texas.

Maeda, Y. (1971). Additional study on static strength of hybrid plate girders in bending. ETH Library, 409-413.

Nethercot D. Buckling of welded hybrid steel I-beams. Journal of the structural division. ASCE. Vol.102 (ST3), pp.461–74. 1976

Petel A., Picard L., Imberty F., Raoul J. Design of a composite road bridge with high strength steels and ultra-high performance fiber reinforced concrete. Proceedings of 5th European Conference on Steel and Composite Structures, Graz, Austria, pp. 159-164. September 3-5, 2008. ISBN 92-0147-00-90

PINTO, R. S. Não-Linearidade Física e Geométrica no Projeto de Edifícios Usuais de Concreto Armado. 108p. São Carlos, 1997. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo

Real E., Chacon R., Mirambell E., Shear response of hybrid steel plate girders. Proceedings of 5th European Conference on Steel and Composite Structures, Graz, Austria, pp. 159-164. September 3-5, 2008. ISBN 92-0147-00-90

Rojas-Blonval, J. E. (2013). Vigas híbridas sometidas a solicitaciones de cortante. Tesis de Máster. Universitat Politècnica de Catalunya.

Rush C., Experimental tension field action behaviour in HPS plate girders. MS thesis, Univ. of Missouri-Columbia, U.S.A 2001

Sarsam, J. B. (1966). Behavior of thin web hybrid beams subjected to static loads. Master Thesis. South Dakota State University.

Schilling C., Bending behaviour of composite hybrid beams. Journal of the Structural Division, ASCE, Vol.94 (ST8),1968

Schilling C., Web Crippling Test on Hybrid Beams. Journal of the Structural Division, ASCE, Vol.93, 1967

Shokouhian, M. (2014). Investigation of Ductility and Section Resistance in Hybrid Flexural Members with Q460 High Strength Steel. Doctoral Thesis.

Shokouhian, M., & Shi, Y. (2014). Investigation of Ductility in Hybrid and High Strength Steel Beams. International Journal of Steel Structures, 265-279.

Shokouhian, M., & Shi, Y. (2015). Flexural strength of hybrid steel I-beams based on slenderness. Engineering Structures, 114-128.

Shokouhian, M., Head, M., & Shi, Y. (2018). A Direct Design Method for Hybrid High Strength Steel Beams Based on Slenderness. Los Angeles, California: Eleventh U.S. National Conference on Earthquake Engineering.

Shokouhian, M., Shi, Y., & Head, M. (2016). Interactive buckling failure modes of hybrid steel flexural members. Engineering Structures, 153-166.

Simões da Silva, L., & Gervásio, H. (2007). Manual de Dimensionamento de Estruturas Metálicas: Métodos Avançados. CMM - Associação Portuguesa de Construção Metálica e Mista.

Sinur, F. (2011). Behaviour of Longitudinally Stiffened Plate Girders Subjected to Bending-Shear Interaction. Ljubljana: Doctoral thesis.

Su A, Liang Y, Zhao O. Experimental and numerical studies of S960 ultra-high strength steel welded I-section columns. Thin-Walled Struct 2020;107166.

Subramanian, L., & W., W. D. (2016). Flexural Resistance of Longitudinally Stiffened I-Girders. I: Yield Limit State. Journal of Bridge Engineering - ASCE.

Trahair, N. S., Bradford, M. A., Nethercot, D. A., & Gardner, L. (2008). The Behaviour and Design of Steel Structures to EC3. New York: Taylor & Francis.

Veljkovic, M., & Johansson, B. (2004). Design of hybrid steel girders. Journal of Constructional Steel Research, 535-547.

Zentz A., Experimental moment-shear interaction and TFA behaviour in hybrid plate girders. MS thesis, Univ. of Missouri-Columbia, U.S.A. 2002

Wang, C. S., Duan, L., Chen, Y. F., & Wang, S. C. (2016). Flexural behavior and ductility of hybrid high performance steel I-girders. Journal of Constructional Steel Research, 1-14.

Åhlenius E., Hybridbalkar I stål. Swedish Institute of Steel Construction, Pub.147, 1994. (in Swedish)

APÊNDICE A – DETALHAMENTO DAS CONDIÇÕES DE CONTORNO E APLICAÇÃO DE CARREGAMENTO NO MODELO ESBELTO

Como já mencionado anteriormente, as condições de contorno e a aplicação do carregamento são semelhantes as adotadas no modelo compacto. Como forma de exemplificar, a Figura Ap.A.1, Figura Ap.A.2 e Figura Ap.A.3 mostram os detalhes. Importante ressaltar que, essas três imagens abaixo são do modelo esbelto com um enrijecedor longitudinal retangular, porém as mesmas condições foram adotadas para o modelo esbelto com dois enrijecedores longitudinais retangulares.

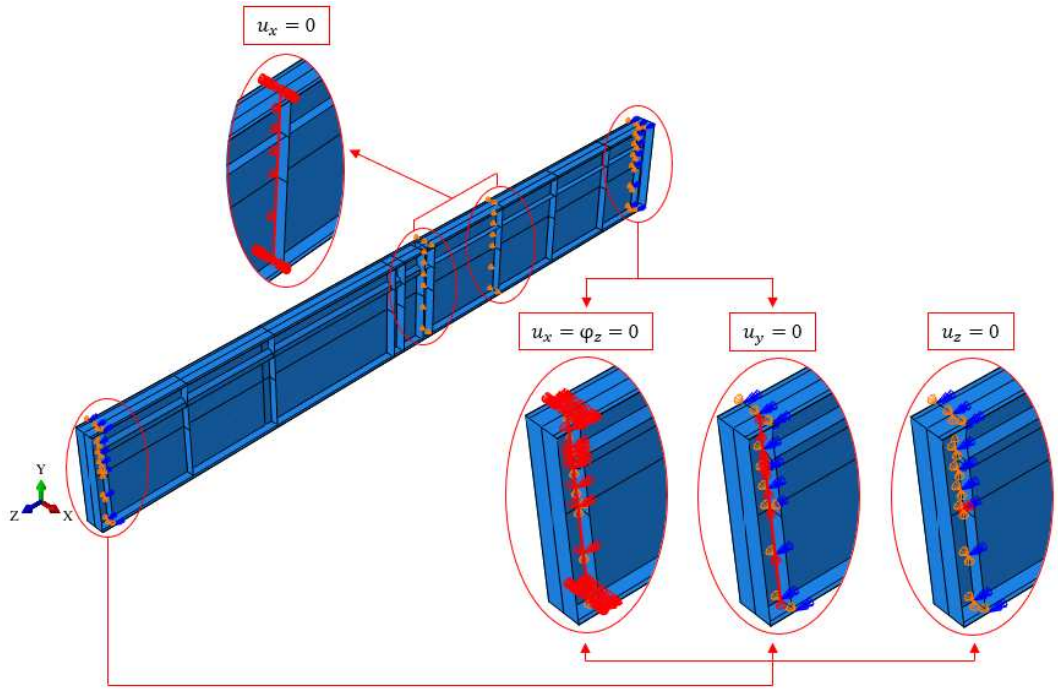


Figura Ap.A.1 - Condições de contorno adotadas no modelo esbelto.

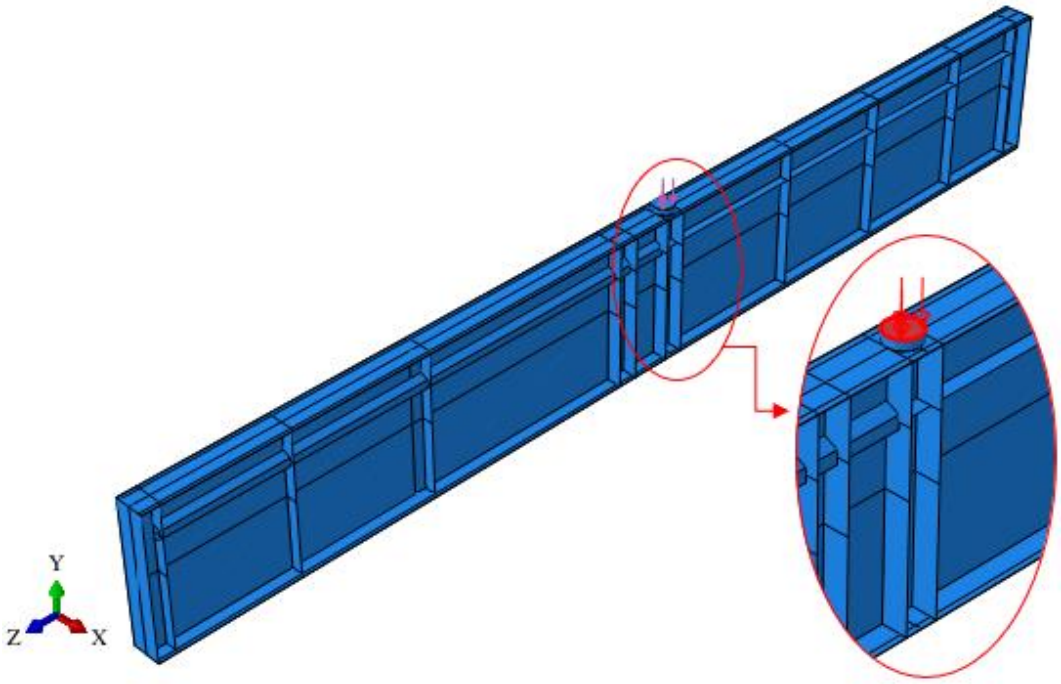


Figura Ap.A.2 - Detalhe do carregamento distribuído aplicado no perfil I no modelo esbelto.

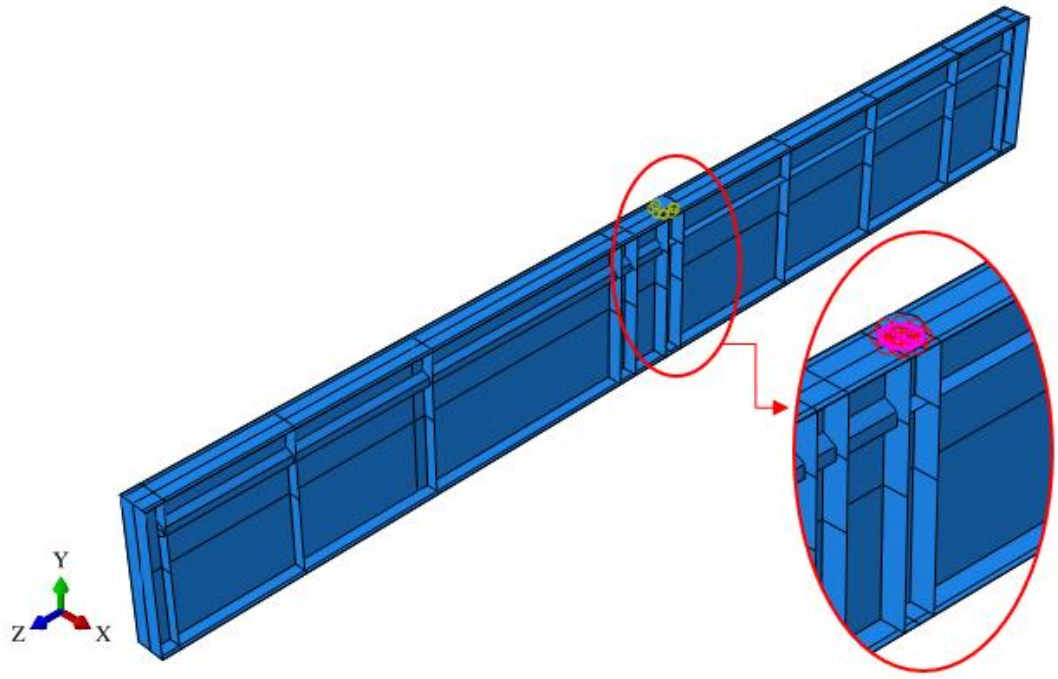


Figura Ap.A.3 - Detalhe da restrição do tipo *tie* aplicada no perfil I e na chapa de carregamento no modelo esbelto

APÊNDICE B – ROTINA DE PROGRAMAÇÃO PARA EXECUÇÃO DOS MODELOS DE VIGAS COMPACTAS

Neste apêndice é apresentado o modelo numérico utilizado para o estudo de sensibilidade de variação dos parâmetros das vigas compactas submetidas a flexão, de forma que a seção seja contida lateralmente para que a FLT não aconteça. Essa rotina foi elaborada a partir do *Python*.

```
# -*- coding: mbcs -*-
##############################################################################
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from abaqusConstants import *
from caeModules import *
from driverUtils import executeOnCaeStartup
import os
```

```
session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=100.0, height=142.0)
session.viewports['Viewport: 1'].makeCurrent()
session.viewports['Viewport: 1'].maximize()
executeOnCaeStartup()
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(referenceRepresentation=ON)

############################## CREATE A BEAM MODEL ################################

class FLABeam:
    def __init__(self, baseNameBuckling, baseNameNL, da, bft, tft, bfb, tfb, tw, tbs, fyf, fyw, fybs, dbearingstiffeners, doverhang, L, hybridBeam, analysisType, Lcr):
        self.baseNameBuckling = baseNameBuckling
        self.baseNameNL = baseNameNL

        self.da = da
        self.bft = bft
        self.tft = tft
        self.bfb = bfb
        self.tfb = tfb
        self.tw = tw
        self.tbs = tbs
        self.fyf = fyf
        self.fyw = fyw
        self.fybs = fybs
        self.dbearingstiffeners = dbearingstiffeners
        self.doverhang = doverhang
```

```
self.L = L

self.hybridBeam = hybridBeam
self.analysisType = analysisType

if self.hybridBeam:
    #definir material 1  - mesas
    self.elasticM1 = ((200000, 0.3), )
    self.plasticM1 = ((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
    #definir material 2 - alma e enrijecedores
    self.elasticM2 = ((200000, 0.3), )
    self.plasticM2 = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.11))

else:
    #definir material 1  - mesas
    self.elasticM1 = ((200000, 0.3), )
    self.plasticM1 = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.11))
    #definir material 2 - alma e enrijecedores
    self.elasticM2 = ((200000, 0.3), )
    self.plasticM2 = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.11))

#definir material da chapa de carregamento
self.elasticFP = ((200000, 0.3), )

self.arco0 = 0.01
```

```
    self.arcMax = 1E036
    self.arcMin = 1E-020
    self.numberOfInc = 1000
    self.sizeOfElements = 20
    self.numberOfEigen = 5

    self.imperfectionFilename=self.baseNameBuckling + '-B'

    if self.analysisType == 'Buckling':
        self.loadMagnitude = -1.0
        self.modelName = self.baseNameBuckling + '-B'
    else:
        self.loadMagnitude = -1.0*Lcr
        self.modelName = self.baseNameNL  + '-NL'

def create_new_model(self):
    mdb.close()
    myBeam = mdb.Model(name=self.modelName, description='')
    del mdb.models['Model-1']
    return myBeam

def create_flange_and_web(self, myBeam):
    flange1Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    flange1Sketch.Line(point1=(-self.bft/2, 0.0), point2=(self.bft/2, 0.0))
    flange1Sketch.HorizontalConstraint(addUndoState=False, entity=flange1Sketch.geometry.findAt((0.0, 0.0), ))
```

```
        myBeam.Part(dimensionality=THREE_D, name='Flange1', type=DEFORMABLE_BODY)
        myBeam.parts['Flange1'].BaseShellExtrude(depth=self.L, sketch=flange1Sketch)
        del myBeam.sketches['__profile__']

        flange2Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        flange2Sketch.Line(point1=(-self.bfb/2, 0.0), point2=(self.bfb/2, 0.0))
        flange2Sketch.HorizontalConstraint(addUndoState=False, entity=flange2Sketch.geometry.findAt((0.0, 0.0), ))
        myBeam.Part(dimensionality=THREE_D, name='Flange2', type=DEFORMABLE_BODY)
        myBeam.parts['Flange2'].BaseShellExtrude(depth=self.L, sketch=flange2Sketch)
        del myBeam.sketches['__profile__']

        webSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        webSketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
        webSketch.VerticalConstraint(addUndoState=False, entity=webSketch.geometry.findAt((0.0, 0.0), ))
        myBeam.Part(dimensionality=THREE_D, name='Web', type=DEFORMABLE_BODY)
        myBeam.parts['Web'].BaseShellExtrude(depth=self.L, sketch=webSketch)
        del myBeam.sketches['__profile__']

    def create_filler_plate_and_bearing_stiffeners(self, myBeam):
        twosidedbearingstiffenerSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        g, v, d, c = twosidedbearingstiffenerSketch.geometry, twosidedbearingstiffenerSketch.vertices, twosidedbearingstiffenerSketch.dimensions,
twosidedbearingstiffenerSketch.constraints
        twosidedbearingstiffenerSketch.setPrimaryObject(option=STANDALONE)
        twosidedbearingstiffenerSketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
        twosidedbearingstiffenerSketch.VerticalConstraint(entity=g[2], addUndoState=False)
```

```
    twosidedbearingstiffenerSketch.Line(point1=(0.0, self.da/2), point2=(self.bft/2, self.da/2))
    twosidedbearingstiffenerSketch.HorizontalConstraint(entity=g[3], addUndoState=False)
    twosidedbearingstiffenerSketch.PerpendicularConstraint(entity1=g[2], entity2=g[3], addUndoState=False)
    twosidedbearingstiffenerSketch.Line(point1=(self.bft/2, self.da/2), point2=(self.bfb/2, -self.da/2))
    twosidedbearingstiffenerSketch.Line(point1=(self.bfb/2, -self.da/2), point2=(0.0, -self.da/2))
    twosidedbearingstiffenerSketch.HorizontalConstraint(entity=g[5], addUndoState=False)
    p = myBeam.Part(name='TwoSidedBearingStiffener', dimensionality=THREE_D,
        type=DEFORMABLE_BODY)
    p = myBeam.parts['TwoSidedBearingStiffener']
    p.BaseShell(sketch=twosidedbearingstiffenerSketch)
    twosidedbearingstiffenerSketch.unsetPrimaryObject()
    p = myBeam.parts['TwoSidedBearingStiffener']
    del myBeam.sketches['__profile__']

    fillerplateSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    fillerplateSketch.rectangle(point1=(-self.bft/2, 20.0), point2=(self.bft/2, -20.0))
    myBeam.Part(dimensionality=THREE_D, name='FillerPlate', type=DEFORMABLE_BODY)
    myBeam.parts['FillerPlate'].BaseSolidExtrude(sketch=fillerplateSketch, depth=40.0)
    del myBeam.sketches['__profile__']

def create_material(self, myBeam):
    if self.hybridBeam:
        myBeam.Material(name='Steel-1')
        myBeam.materials['Steel-1'].Elastic(table=self.elasticM1)
        if self.analysisType != 'Buckling':
```

```
            myBeam.materials['Steel-1'].Plastic(table=self.plasticM1)

        myBeam.Material(name='Steel-2')
        myBeam.materials['Steel-2'].Elastic(table=self.elasticM2)
        if self.analysisType != 'Buckling':
            myBeam.materials['Steel-2'].Plastic(table=self.plasticM2)

    else:
        myBeam.Material(name='Steel-1')
        myBeam.materials['Steel-1'].Elastic(table=self.elasticM1)
        if self.analysisType != 'Buckling':
            myBeam.materials['Steel-1'].Plastic(table=self.plasticM1)

        myBeam.Material(name='Steel-2')
        myBeam.materials['Steel-2'].Elastic(table=self.elasticM2)
        if self.analysisType != 'Buckling':
            myBeam.materials['Steel-2'].Plastic(table=self.plasticM2)

    myBeam.Material(name='Steel-FillerPlate')
    myBeam.materials['Steel-FillerPlate'].Elastic(table=self.elasticFP)

def create_sections(self, myBeam):
    if self.hybridBeam:
        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-1', name='tft', numIntPts=5,
```

```
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tft, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-1', name='tfb', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tfb, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-2', name='tw', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-2', name='tbs', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tbs, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

    else:
        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-1', name='tft', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tft, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-1', name='tfb', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tfb, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-2', name='tw', numIntPts=5,
```

```
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

          myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-2', name='tbs', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tbs, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

       myBeam.HomogeneousSolidSection(name='FillerPlate', material='Steel-FillerPlate', thickness=None)

    def assign_sections(self, myBeam):
       myBeam.parts['Flange1'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
          faces=myBeam.parts['Flange1'].faces.findAt(((0.48*self.bft, 0.0, 0.0),(-0.48*self.bft, 0.0, self.L)), )), sectionName='tft', thicknessAssignment=FROM_SECTION)

       myBeam.parts['Flange2'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
          faces=myBeam.parts['Flange2'].faces.findAt(((0.48*self.bfb, 0.0, 0.0),(-0.48*self.bfb, 0.0, self.L)), )), sectionName='tfb', thicknessAssignment=FROM_SECTION)

       myBeam.parts['Web'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
          faces=myBeam.parts['Web'].faces.findAt(((0.0, -0.48*self.da, 0.0), (0.0, 0.48*self.da, self.L)), )), sectionName='tw', thicknessAssignment=FROM_SECTION)

       delta = 1
       myBeam.parts['TwoSidedBearingStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
          faces=myBeam.parts['TwoSidedBearingStiffener'].faces.getByBoundingBox(0.0-delta, -self.da/2-delta, 0.0-delta,
          self.bfb/2+delta, self.da/2+delta, 0.0+delta)), sectionName='tbs', thicknessAssignment=FROM_SECTION)
```

```
        myBeam.parts['FillerPlate'].SectionAssignment(region=regionToolset.Region(cells=myBeam.parts['FillerPlate'].cells.getByBoundingBox(-self.bft/2-delta,      -20.0-delta,
0.0-delta, self.bft/2+delta, 20.0+delta, 40.0+delta)),
            sectionName='FillerPlate', offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTION)

    def create_beam_instance(self, myBeam):
        myBeam.rootAssembly.DatumCsysByDefault(CARTESIAN)
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web', part=myBeam.parts['Web'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Flange1', part=myBeam.parts['Flange1'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Flange2', part=myBeam.parts['Flange2'])
        myBeam.rootAssembly.translate(instanceList=('Flange1', ), vector=(0.0, self.da/2, 0.0))
        myBeam.rootAssembly.translate(instanceList=('Flange2', ), vector=(0.0, -self.da/2, 0.0))
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener1', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener2', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener3', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener4', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener5', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener6', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener7', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedBearingStiffener8', part=myBeam.parts['TwoSidedBearingStiffener'])
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener1', ), vector=(0.0, 0.0, self.doverhang))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener2', ), vector=(0.0, 0.0, self.doverhang+self.dbearingstiffeners))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener3', ), vector=(0.0, 0.0, self.doverhang+2*self.dbearingstiffeners))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener4', ), vector=(0.0, 0.0, self.doverhang+3*self.dbearingstiffeners))
        myBeam.rootAssembly.rotate(instanceList=('TwoSidedBearingStiffener5', ), axisPoint=(0.0, 184.65, 0.0), axisDirection=(0.0, -369.3, 0.0), angle=180.0)
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener5', ), vector=(0.0, 0.0, self.doverhang))
```

```
        myBeam.rootAssembly.rotate(instanceList=('TwoSidedBearingStiffener6', ), axisPoint=(0.0, 184.65, 0.0), axisDirection=(0.0, -369.3, 0.0), angle=180.0)
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener6', ), vector=(0.0, 0.0, self.doverhang+self.dbearingstiffeners))
        myBeam.rootAssembly.rotate(instanceList=('TwoSidedBearingStiffener7', ), axisPoint=(0.0, 184.65, 0.0), axisDirection=(0.0, -369.3, 0.0), angle=180.0)
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener7', ), vector=(0.0, 0.0, self.doverhang+2*self.dbearingstiffeners))
        myBeam.rootAssembly.rotate(instanceList=('TwoSidedBearingStiffener8', ), axisPoint=(0.0, 184.65, 0.0), axisDirection=(0.0, -369.3, 0.0), angle=180.0)
        myBeam.rootAssembly.translate(instanceList=('TwoSidedBearingStiffener8', ), vector=(0.0, 0.0, self.doverhang+3*self.dbearingstiffeners))
        myBeam.rootAssembly.Instance(dependent=OFF, name='FillerPlate1', part=myBeam.parts['FillerPlate'])
        myBeam.rootAssembly.translate(instanceList=('FillerPlate1', ), vector=(0.0, 20.0+self.da/2, self.doverhang+self.dbearingstiffeners-20.0))
        myBeam.rootAssembly.Instance(dependent=OFF, name='FillerPlate2', part=myBeam.parts['FillerPlate'])
        myBeam.rootAssembly.translate(instanceList=('FillerPlate2', ), vector=(0.0, 20.0+self.da/2, self.doverhang+2*self.dbearingstiffeners-20.0))

        myBeam.rootAssembly.InstanceFromBooleanMerge(domain=GEOMETRY, instances=(myBeam.rootAssembly.instances['Web'],
myBeam.rootAssembly.instances['Flange1'], myBeam.rootAssembly.instances['Flange2'],
            myBeam.rootAssembly.instances['TwoSidedBearingStiffener1'], myBeam.rootAssembly.instances['TwoSidedBearingStiffener2'],
myBeam.rootAssembly.instances['TwoSidedBearingStiffener3'],
            myBeam.rootAssembly.instances['TwoSidedBearingStiffener4'], myBeam.rootAssembly.instances['TwoSidedBearingStiffener5'],
myBeam.rootAssembly.instances['TwoSidedBearingStiffener6'],
            myBeam.rootAssembly.instances['TwoSidedBearingStiffener7'], myBeam.rootAssembly.instances['TwoSidedBearingStiffener8']), keepIntersections=ON,
name='Beam', originalInstances=DELETE)

        myInstance1 = myBeam.rootAssembly.instances['Beam-1']
        myBeam.rootAssembly.makeIndependent(instances=(myInstance1, ))

    def partition_face(self,myBeam):
        myInstance1 = myBeam.rootAssembly.instances['Beam-1']
```

```
        xzPlane1= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=0.0, principalPlane=XZPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane1.id], faces=myInstance1.faces)

        xyPlane1= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.doverhang+self.dbearingstiffeners-20.0, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane1.id], faces=myInstance1.faces)

        xyPlane2= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.doverhang+self.dbearingstiffeners+20.0, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane2.id], faces=myInstance1.faces)

        xyPlane3= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.doverhang+3*self.dbearingstiffeners/2, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane3.id], faces=myInstance1.faces)

        xyPlane4= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.doverhang+2*self.dbearingstiffeners-20.0, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane4.id], faces=myInstance1.faces)

        xyPlane5= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.doverhang+2*self.dbearingstiffeners+20.0, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane5.id], faces=myInstance1.faces)


    def surfaces(self, myBeam):
        delta=1

        myBeam.rootAssembly.Surface(side2Faces=myBeam.rootAssembly.instances['Beam-1'].faces.getByBoundingBox(-self.bft/2-delta,
self.da/2-delta,
self.doverhang+self.dbearingstiffeners-20.0-delta,
```

```
        self.bft/2+delta, self.da/2+delta, self.doverhang+self.dbearingstiffeners+20.0+delta), name='SurfaceBeam-Tie1')

    myBeam.rootAssembly.Surface(side2Faces=myBeam.rootAssembly.instances['Beam-1'].faces.getByBoundingBox(-self.bft/2-delta, self.da/2-delta,
self.doverhang+2*self.dbearingstiffeners-20.0-delta,
        self.bft/2+delta, self.da/2+delta, self.doverhang+2*self.dbearingstiffeners+20.0+delta), name='SurfaceBeam-Tie2')

    myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['FillerPlate1'].faces.getByBoundingBox(-self.bft/2-delta, self.da/2-delta,
self.doverhang+self.dbearingstiffeners-20.0-delta,
        self.bft/2+delta, self.da/2+delta, self.doverhang+self.dbearingstiffeners+20.0+delta), name='SurfacePlate1-Tie1')

    myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['FillerPlate2'].faces.getByBoundingBox(-self.bft/2-delta, self.da/2-delta,
self.doverhang+2*self.dbearingstiffeners-20.0-delta,
        self.bft/2+delta, self.da/2+delta, self.doverhang+2*self.dbearingstiffeners+20.0+delta), name='SurfacePlate2-Tie2')

    myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['FillerPlate1'].faces.getByBoundingBox(-self.bft/2-delta, self.da/2+40.0-delta,
self.doverhang+self.dbearingstiffeners-20.0-delta,
        self.bft/2+delta, self.da/2+40.0+delta, self.doverhang+self.dbearingstiffeners+20.0+delta), name='SurfacePlate1-Load1')

    myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['FillerPlate2'].faces.getByBoundingBox(-self.bft/2-delta, self.da/2+40.0-delta,
self.doverhang+2*self.dbearingstiffeners-20.0-delta,
        self.bft/2+delta, self.da/2+40.0+delta, self.doverhang+2*self.dbearingstiffeners+20.0+delta), name='SurfacePlate2-Load2')


  def mesh_instance(self, myBeam):
    myInstance1 = myBeam.rootAssembly.instances['Beam-1']
```

```
    myBeam.rootAssembly.setMeshControls(regions=myInstance1.faces, technique=STRUCTURED)
    myBeam.rootAssembly.setElementType(elemTypes=(ElemType(elemCode=S4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF), ElemType(elemCode=S3,
elemLibrary=STANDARD)), regions=Region(myInstance1.faces))
    myBeam.rootAssembly.seedPartInstance(deviationFactor=0.1, minSizeFactor=0.1, regions=(myInstance1, ), size=self.sizeOfElements)
    myBeam.rootAssembly.generateMesh(regions=(myInstance1, ))

    a = myBeam.rootAssembly
    session.viewports['Viewport: 1'].setValues(displayedObject=a)
    session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON,
        optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
    session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
        meshTechnique=ON)
    elemType1 = mesh.ElemType(elemCode=C3D8R, elemLibrary=STANDARD,
        kinematicSplit=AVERAGE_STRAIN, secondOrderAccuracy=OFF,
        hourglassControl=DEFAULT, distortionControl=DEFAULT)
    elemType2 = mesh.ElemType(elemCode=C3D6, elemLibrary=STANDARD)
    elemType3 = mesh.ElemType(elemCode=C3D4, elemLibrary=STANDARD)
    a = myBeam.rootAssembly
    c1 = a.instances['FillerPlate1'].cells
    cells1 = c1.getSequenceFromMask(mask=('[#1 ]', ), )
    c2 = a.instances['FillerPlate2'].cells
    cells2 = c2.getSequenceFromMask(mask=('[#1 ]', ), )
    pickedRegions =((cells1+cells2), )
    a.setElementType(regions=pickedRegions, elemTypes=(elemType1, elemType2,
        elemType3))
```

```
        a = myBeam.rootAssembly
        partInstances =(a.instances['FillerPlate1'], a.instances['FillerPlate2'], )
        a.seedPartInstance(regions=partInstances, size=5.0, deviationFactor=0.1,
            minSizeFactor=0.1)
        a = myBeam.rootAssembly
        partInstances =(a.instances['FillerPlate1'], a.instances['FillerPlate2'], )
        a.generateMesh(regions=partInstances)

    def create_sets(self, myBeam):
        delta = 1
        myInstance1 = myBeam.rootAssembly.instances['Beam-1']

        myBeam.rootAssembly.Set(nodes=myInstance1.nodes.getByBoundingBox(0.0-delta, -self.da/2-delta, 0.0-delta,
            0.0+delta, self.da/2+delta, 2*self.doverhang+3*self.dbearingstiffeners+delta), name='Set-AllWebNodes')

        myBeam.rootAssembly.Set(nodes=myInstance1.nodes.getByBoundingBox(-self.bft/2-delta, self.da/2-delta, 0.0-delta,
            self.bft/2+delta, self.da/2+delta, 2*self.doverhang+3*self.dbearingstiffeners+delta), name='Set-AllFlangeNodes1')

        myBeam.rootAssembly.Set(nodes=myInstance1.nodes.getByBoundingBox(-self.bft/2-delta, -self.da/2-delta, 0.0-delta,
            self.bft/2+delta, -self.da/2+delta, 2*self.doverhang+3*self.dbearingstiffeners+delta), name='Set-AllFlangeNodes2')

        myBeam.rootAssembly.SetByBoolean(name='Set-AllFlangeNodes', sets=(myBeam.rootAssembly.sets['Set-AllFlangeNodes1'], myBeam.rootAssembly.sets['Set-
AllFlangeNodes2'], ))

        myBeam.rootAssembly.Set(edges=myInstance1.edges.findAt(((-0.99*self.bfb/2, -self.da/2, self.doverhang), ), ((0.99*self.bfb/2, -self.da/2, self.doverhang), ), ((
```

```
        0.0, -0.01, self.doverhang), ), ((0.0, 0.01, self.doverhang), ), ((-0.99*self.bfb/2, self.da/2, self.doverhang), ), ((0.99*self.bfb/2, self.da/2, self.doverhang), ), ((
        -0.99*self.bfb/2, -self.da/2, self.doverhang+3*self.dbearingstiffeners), ), ((0.99*self.bfb/2, -self.da/2, self.doverhang+3*self.dbearingstiffeners), ),
        ((0.0, -0.01, self.doverhang+3*self.dbearingstiffeners), ), ((0.0, 0.01, self.doverhang+3*self.dbearingstiffeners), ), ((-0.99*self.bfb/2, self.da/2,
self.doverhang+3*self.dbearingstiffeners), ), ((
        0.99*self.bfb/2, self.da/2, self.doverhang+3*self.dbearingstiffeners), ), ), name='Set-BoundaryCondition1')

    myBeam.rootAssembly.Set(edges=myInstance1.edges.findAt(((0.0, -0.01, self.doverhang), ), ((0.0, 0.01, self.doverhang), ), ((0.0, -0.01,
self.doverhang+3*self.dbearingstiffeners), ), ((
        0.0, 0.01, self.doverhang+3*self.dbearingstiffeners), ), ), name='Set-BoundaryCondition2')

    myBeam.rootAssembly.Set(edges=myInstance1.edges.findAt(((-0.99*self.bfb/2, -self.da/2, self.doverhang+self.dbearingstiffeners), ), ((0.99*self.bfb/2, -self.da/2,
self.doverhang+self.dbearingstiffeners), ), ((
        0.0, -0.01, self.doverhang+self.dbearingstiffeners), ), ((0.0, 0.01, self.doverhang+self.dbearingstiffeners), ), ((-0.99*self.bfb/2, self.da/2,
self.doverhang+self.dbearingstiffeners), ), ((
        0.99*self.bfb/2, self.da/2, self.doverhang+self.dbearingstiffeners), ), ((-0.99*self.bfb/2, -self.da/2, self.doverhang+2*self.dbearingstiffeners), ), ((0.99*self.bfb/2, -
self.da/2, self.doverhang+2*self.dbearingstiffeners), ),
        ((0.0, -0.01, self.doverhang+2*self.dbearingstiffeners), ), ((0.0, 0.01, self.doverhang+2*self.dbearingstiffeners), ), ((-0.99*self.bfb/2, self.da/2,
self.doverhang+2*self.dbearingstiffeners), ), ((
        0.99*self.bfb/2, self.da/2, self.doverhang+2*self.dbearingstiffeners), ), ), name='Set-BoundaryCondition3')

    myBeam.rootAssembly.Set(name='Set-NC1', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+self.dbearingstiffeners), )))

    myBeam.rootAssembly.Set(name='Set-NC2', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+3*self.dbearingstiffeners/2), )))

    myBeam.rootAssembly.Set(name='Set-NC3', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+2*self.dbearingstiffeners), )))
```

```
myBeam.rootAssembly.Set(name='Set-NC4', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.doverhang+self.dbearingstiffeners), )))

myBeam.rootAssembly.Set(name='Set-NC5', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.doverhang+3*self.dbearingstiffeners/2), )))

myBeam.rootAssembly.Set(name='Set-NC6', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.doverhang+2*self.dbearingstiffeners), )))

myBeam.rootAssembly.Set(name='Set-NC7', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+self.dbearingstiffeners), )))

myBeam.rootAssembly.Set(name='Set-NC8', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+3*self.dbearingstiffeners/2), )))

myBeam.rootAssembly.Set(name='Set-NC9', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+2*self.dbearingstiffeners), )))

myBeam.rootAssembly.Set(name='Set-NC10', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 0.0), )))

myBeam.rootAssembly.Set(name='Set-NC11', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang), )))

myBeam.rootAssembly.Set(name='Set-NC12', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+3*self.dbearingstiffeners), )))

myBeam.rootAssembly.Set(name='Set-NC13', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 2*self.doverhang+3*self.dbearingstiffeners), )))

myBeam.rootAssembly.Set(name='Set-NC14', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang), )))

myBeam.rootAssembly.Set(name='Set-NC15', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+3*self.dbearingstiffeners), )))
```

```
        myBeam.rootAssembly.Set(name='Set-NC16', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.doverhang), )))

        myBeam.rootAssembly.Set(name='Set-NC17', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.doverhang+3*self.dbearingstiffeners), )))

        myBeam.rootAssembly.Set(name='Set-NC18', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+self.dbearingstiffeners-20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC19', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+self.dbearingstiffeners+20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC20', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+2*self.dbearingstiffeners-20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC21', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.doverhang+2*self.dbearingstiffeners+20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC22', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+self.dbearingstiffeners-20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC23', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+self.dbearingstiffeners+20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC24', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+2*self.dbearingstiffeners-20.0), )))

        myBeam.rootAssembly.Set(name='Set-NC25', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.doverhang+2*self.dbearingstiffeners+20.0), )))

        myBeam.rootAssembly.Set(edges=
            myInstance1.edges.findAt(((-0.48*self.bft, self.da/2, self.doverhang+self.dbearingstiffeners), ), ((0.48*self.bft, self.da/2, self.doverhang+self.dbearingstiffeners), ), ),
name='Set-FlangeNodes1')

        myBeam.rootAssembly.Set(edges=
```

```
        myInstance1.edges.findAt(((-0.48*self.bft, self.da/2, self.doverhang+2*self.dbearingstiffeners), ), ((0.48*self.bft, self.da/2, self.doverhang+2*self.dbearingstiffeners), ),
), name='Set-FlangeNodes2')

    myBeam.rootAssembly.Set(nodes=myBeam.rootAssembly.sets['Set-FlangeNodes1'].nodes, name='Set-Nodes1')

    myBeam.rootAssembly.Set(nodes=myBeam.rootAssembly.sets['Set-FlangeNodes2'].nodes, name='Set-Nodes2')

  def create_step(self, myBeam):
    if self.analysisType == 'Buckling':
      myBeam.BuckleStep(eigensolver=SUBSPACE, name='Step-1',
        numEigen=self.numberOfEigen, maxIterations=1000, previous='Initial')
    else:
      myBeam.StaticRiksStep(initialArcInc=self.arco0, minArcInc=self.arcMin, maxArcInc=self.arcMax, name='Step-1',
        nlgeom=ON, previous='Initial', maxNumInc = self.numberOfInc)

  def constraint(self, myBeam):
    myBeam.Tie(name='Tie-Load1', master=myBeam.rootAssembly.surfaces['SurfacePlate1-Tie1'], slave=myBeam.rootAssembly.surfaces['SurfaceBeam-Tie1'],
      positionToleranceMethod=COMPUTED, adjust=ON, tieRotations=ON, thickness=ON)
    myBeam.Tie(name='Tie-Load2', master=myBeam.rootAssembly.surfaces['SurfacePlate2-Tie2'], slave=myBeam.rootAssembly.surfaces['SurfaceBeam-Tie2'],
      positionToleranceMethod=COMPUTED, adjust=ON, tieRotations=ON, thickness=ON)

  def create_load(self, myBeam):
    area=self.bft*40.0
    loadapplied=self.loadMagnitude/area
```

```
        myBeam.Pressure(name='Load-1',
            createStepName='Step-1', region=myBeam.rootAssembly.surfaces['SurfacePlate1-Load1'], distributionType=UNIFORM, field='', magnitude=-loadapplied)
        myBeam.Pressure(name='Load-2',
            createStepName='Step-1', region=myBeam.rootAssembly.surfaces['SurfacePlate2-Load2'], distributionType=UNIFORM, field='', magnitude=-loadapplied)

    def set_boundary_conditions(self, myBeam):
        myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BoundaryCondition-1',
            region=myBeam.rootAssembly.sets['Set-BoundaryCondition1'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=SET)
        myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BoundaryCondition-2',
            region=myBeam.rootAssembly.sets['Set-BoundaryCondition2'], u1=UNSET, u2=SET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
        myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BoundaryCondition-
NC17',
            region=myBeam.rootAssembly.sets['Set-NC17'], u1=UNSET, u2=UNSET, u3=SET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
        myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BoundaryCondition-3',
            region=myBeam.rootAssembly.sets['Set-BoundaryCondition3'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

    def createResidualStress(self, myBeam):
        myInstance1 = myBeam.rootAssembly.instances['Beam-1']
        delta = 1 #mm

        # Mesa Superior
        myBeam.rootAssembly.Set(edges=
        myInstance1.edges.findAt(((-0.48*self.bft, self.da/2, self.doverhang+self.dbearingstiffeners), ), ((0.48*self.bft, self.da/2, self.doverhang+self.dbearingstiffeners), ), ),
name='Set-FlangeNodes')
        numberFlangeElements = int(len(myBeam.rootAssembly.sets['Set-FlangeNodes'].nodes)-1)
```

```
    sizeOfFlangeElements = (self.bft)/(numberFlangeElements)

    # Definir tensao residual para as mesas
    for x in range(int(numberFlangeElements/2)):
        setName = 'TRFU' + str(x+1)
        x1 = x*sizeOfFlangeElements
        x2 = (x+1)*sizeOfFlangeElements
        y = self.da/2
        f1 = ((3.397*self.fyf)/((self.bft/2.0)*(self.bft/2.0)))*(x1*x1)-((3.497*self.fyf)/(self.bft/2.0))*x1+(0.6*self.fyf)
        f2 = ((3.397*self.fyf)/((self.bft/2.0)*(self.bft/2.0)))*(x2*x2)-((3.497*self.fyf)/(self.bft/2.0))*x2+(0.6*self.fyf)
        elementS = (f2+f1)/2.0

        e1 = myInstance1.elements.getByBoundingBox(x1-delta, y-delta, -delta, x2+delta, y+delta, self.L+delta)
        e2 = myInstance1.elements.getByBoundingBox(-x2-delta, y-delta, -delta, -x1+delta, y+delta, self.L+delta)

        elements1 = e1+ e2
        myBeam.rootAssembly.Set(elements = elements1, name=setName)

        myBeam.Stress(name='TRFU'+str(x+1), region=myBeam.rootAssembly.sets['TRFU'+str(x+1)], distributionType=UNIFORM, sigma11=0.0,
            sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

    # Mesa Inferior
    myBeam.rootAssembly.Set(edges=
    myInstance1.edges.findAt(((-0.48*self.bfb, self.da/2, self.doverhang+self.dbearingstiffeners), ), ((0.48*self.bfb, self.da/2, self.doverhang+self.dbearingstiffeners), ), ),
name='Set-FlangeNodes')
```

```
numberFlangeElements = int(len(myBeam.rootAssembly.sets['Set-FlangeNodes'].nodes)-1)
sizeOfFlangeElements = (self.bfb)/(numberFlangeElements)

# Definir tensao residual para as mesas
for x in range(int(numberFlangeElements/2)):
    setName = 'TRFB' + str(x+1)
    x1 = x*sizeOfFlangeElements
    x2 = (x+1)*sizeOfFlangeElements
    y = self.da/2
    f1 = ((3.397*self.fyf)/((self.bft/2.0)*(self.bft/2.0)))*(x1*x1)-((3.497*self.fyf)/(self.bft/2.0))*x1+(0.6*self.fyf)
    f2 = ((3.397*self.fyf)/((self.bft/2.0)*(self.bft/2.0)))*(x2*x2)-((3.497*self.fyf)/(self.bft/2.0))*x2+(0.6*self.fyf)
    elementS = (f2+f1)/2.0

    if abs(elementS) < (5e-7):
        elementS = 0

    e1 = myInstance1.elements.getByBoundingBox(x1-delta, -y-delta, -delta, x2+delta, -y+delta, self.L+delta)
    e2 = myInstance1.elements.getByBoundingBox(-x2-delta, -y-delta, -delta, -x1+delta, -y+delta, self.L+delta)

    elements1 = e1+e2
    myBeam.rootAssembly.Set(elements = elements1, name=setName)

    myBeam.Stress(name='TRFB'+str(x+1), region=myBeam.rootAssembly.sets['TRFB'+str(x+1)], distributionType=UNIFORM, sigma11=0.0,
        sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)
```

```
    # Alma
    myBeam.rootAssembly.Set(edges=
    myInstance1.edges.findAt(((0.0, -0.48*self.da, self.doverhang+self.dbearingstiffeners), ), ((0.0, 0.48*self.da, self.doverhang+self.dbearingstiffeners), ), ), name='Set-
WebNodes')
    numberWebElements = int(len(myBeam.rootAssembly.sets['Set-WebNodes'].nodes)-1)
    sizeOfWebElements = self.da/(numberWebElements)

    # Definir tensao residual para a alma
    for y in range(int(numberWebElements/2)):
      setName = 'TRW' + str(y+1)
      y1 = y*sizeOfWebElements
      y2 = (y+1)*sizeOfWebElements
      x = 0.0
      g = ((-0.1*self.fyw)*(0.1225*self.da+0.0393*self.bft)-(0.6*self.fyw*0.1225*self.da)-(0.6*self.fyw*0.0393*self.bft))/(-0.6*self.fyw)

      if (y1 > self.da/2.0-g) & (y2 > self.da/2.0-g):
        f1 = ((-0.6*self.fyw*(self.da/2.0-y1)+0.0735*self.fyw*self.da+0.02362*self.fyw*self.bft))/(0.1225*self.da+0.0393*self.bft)
        f2 = ((-0.6*self.fyw*(self.da/2.0-y2)+0.0735*self.fyw*self.da+0.02362*self.fyw*self.bft))/(0.1225*self.da+0.0393*self.bft)
        elementS = (f2+f1)/2

      if (y1 < self.da/2.0-g) & (y2 > self.da/2.0-g):
        f1 = -0.1*self.fyw
        f2 = ((-0.6*self.fyw*(self.da/2.0-y2)+0.0735*self.fyw*self.da+0.02362*self.fyw*self.bft))/(0.1225*self.da+0.0393*self.bft)
        elementS = (f2+f1)/2
```

```
        if (y1 < self.da/2.0-g) & (y2 < self.da/2.0-g):
            f1 = -0.1*self.fyw
            f2 = -0.1*self.fyw
            elementS = (f2+f1)/2

        if abs(elementS) < (5e-7):
            elementS = 0

        e1 = myInstance1.elements.getByBoundingBox(x-delta, y1-delta, -delta, x+delta, y2+delta, self.L+delta)
        e2 = myInstance1.elements.getByBoundingBox(x-delta, -y2-delta, -delta, x+delta, -y1+delta, self.L+delta)

        elements1 = e1+e2
        myBeam.rootAssembly.Set(elements = elements1, name=setName)

        myBeam.Stress(name='TRW'+str(y+1), region=myBeam.rootAssembly.sets['TRW'+str(y+1)], distributionType=UNIFORM, sigma11=0.0,
            sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

def edit_text(self, myBeam, textStop, texttoReplace):
    myBeam.keywordBlock.synchVersions(storeNodesAndElements=False)
    linePosition = 0
    for keyWords in myBeam.keywordBlock.sieBlocks:
        if keyWords.startswith(textStop):
            myBeam.keywordBlock.replace(linePosition, texttoReplace)
            break
        linePosition = linePosition + 1
```

```
def create_jobs(self, jobName):
    mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF, explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
        memory=90, memoryUnits=PERCENTAGE, model=self.modelName, modelPrint=OFF, multiprocessingMode=DEFAULT, name=jobName,
        nodalOutputPrecision=SINGLE, numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)

def create_beam(self, nameofModel, jobName):
    self.create_flange_and_web(nameofModel)
    self.create_filler_plate_and_bearing_stiffeners(nameofModel)
    self.create_material(nameofModel)
    self.create_sections(nameofModel)
    self.assign_sections(nameofModel)
    self.create_beam_instance(nameofModel)
    self.partition_face(nameofModel)
    self.surfaces(nameofModel)
    self.mesh_instance(nameofModel)
    self.create_sets(nameofModel)
    self.create_step(nameofModel)
    self.constraint(nameofModel)
    self.create_load(nameofModel)
    self.set_boundary_conditions(nameofModel)


    if self.analysisType != 'Buckling':
        self.createResidualStress(nameofModel)
```

```
        if self.analysisType == 'Buckling':
            textStop = '*Output'
            texttoReplace = '*Output, field, variable=PRESELECT\n*NODE FILE\nU,'
        else:
            textStop = '*End Assembly'
            texttoReplace = '*End Assembly\n*IMPERFECTION, FILE=' + self.imperfectionFilename + ', STEP=1\n1, ' + str(2.0) + '\n'

        self.edit_text(nameofModel, textStop, texttoReplace)
        self.create_jobs(jobName)

hybridBeam = 1
if hybridBeam:
    baseNameBuckling = 'C3'
    baseNameNL = 'C3-235-355'
    fyf = 355.00 #tensão de escoamento da mesa
    fyw = 235.00 #tensão de escoamento da alma
    fybs = 235.00 #tensão de escoamento do enrijecedor transversal
else:
    baseNameBuckling = 'C3'
    baseNameNL = 'C3-235-235'
    fyf = 235.00 #tensão de escoamento da mesa
    fyw = 235.00 #tensão de escoamento da alma
    fybs = 235.00 #tensão de escoamento do enrijecedor transversal
```

```
#Medidas em milimetros e Newton
da = 608.0 #altura da alma (linha do esqueleto)
bft = 168.0 #largura da mesa superior
tft = 12.0 #espessura da mesa superior
bfb = 168.0 #largura da mesa superior
tfb = 12.0 #espessura da mesa superior
tw = 8.0 #espessura da alma
tbs = 14.0 #espessura do enrijecedor transversal
dbearingstiffeners = 1000.0 #distancia entre os bearing stiffeners
doverhang = 200.0 #distancia depois de cada apoio
L = 3400.0 #comprimento total da viga

analysisType = 'NL'
Lcr = 1234980.0

beamExample = FLABeam(baseNameBuckling, baseNameNL, da, bft, tft, bfb, tfb, tw, tbs, fyf, fyw, fybs, dbearingstiffeners, doverhang, L, hybridBeam, analysisType, Lcr)
beamExampleModel=beamExample.create_new_model()
if analysisType == 'Buckling':
    beamExample.create_beam(beamExampleModel, baseNameBuckling + '-B')
else:
    beamExample.create_beam(beamExampleModel, baseNameNL + '-NL')
```

APÊNDICE C – ROTINA DE PROGRAMAÇÃO PARA EXECUÇÃO DOS MODELOS DE VIGAS ESBELTAS

Neste apêndice é apresentado o modelo numérico utilizado para o estudo de sensibilidade de variação dos parâmetros dos painéis das vigas esbeltas submetidas a flexão, de forma que a seção seja contida lateralmente para que a FLT não aconteça.

C.1 Painel da viga esbelta com um enrijecedor retangular longitudinal

```
# -*- coding: mbcs -*-

##############################################################################################################################
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from abaqusConstants import *
from caeModules import *
```

```
from driverUtils import executeOnCaeStartup
import os

session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=100.0, height=142.0)
session.viewports['Viewport: 1'].makeCurrent()
session.viewports['Viewport: 1'].maximize()
executeOnCaeStartup()
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(referenceRepresentation=ON)

########################################################## CREATE A BEAM MODEL ##########################################################

class FLABeam:
    def __init__(self, baseNameBuckling, baseNameNL, fyf, fyw, da, bf, tf, tw7, tw8, tw15, tots, wots, tdts, wdts, tlos, wlos, tlcs, wlcs, b, los, lcs, loverhang, daSO, twSO, lSO,
bfuSO, tfuSO, bfbSO, tfbSO, tlosSO, wlosSO, losSO, dP1, dP2, dP3, L, lP1, lP2, lP3, lP4, analysisType, Lcr):
        self.baseNameBuckling = baseNameBuckling
        self.baseNameNL = baseNameNL

        self.fyf=fyf
        self.fyw=fyw
        self.da=da
        self.bf=bf
        self.tf=tf
        self.tw7=tw7
        self.tw8=tw8
        self.tw15=tw15
```

```
self.tots=tots
self.wots=wots
self.tdts=tdts
self.wdts=wdts
self.tlos=tlos
self.wlos=wlos
self.tlcs=tlcs
self.wlcs=wlcs
self.b=b
self.los=los
self.lcs=lcs
self.loverhang=loverhang

self.daSO=daSO
self.twSO=twSO
self.lSO=lSO
self.bfuSO=bfuSO
self.tfuSO=tfuSO
self.bfbSO=bfbSO
self.tfbSO=tfbSO
self.tlosSO=tlosSO
self.wlosSO=wlosSO
self.losSO=losSO

self.dP1=dP1
```

```
self.dP2=dP2
self.dP3=dP3
self.L=L
self.lP1=lP1
self.lP2=lP2
self.lP3=lP3
self.lP4=lP4

self.analysisType=analysisType

#definir material das mesas
self.elasticF=((200000, 0.3), )
self.plasticF=((461.06, 0.0), (475.29, 0.0203631), (664.20, 0.2036932), (0.000001, 0.21))
#definir material da alma (parte em analise)
self.elasticW7=((200000, 0.3), )
self.plasticW7=((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
#definir material da alma (120mm depois do enrijecedor intermediario)
self.elasticW8=((200000, 0.3), )
self.plasticW8=((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
#definir material do enrijecedor retangular longitudinal
self.elasticLOS=((200000, 0.3), )
self.plasticLOS=((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
#definir material do enrijecedor trapeizodal longitudinal
self.elasticLCS=((200000, 0.3), )
self.plasticLCS=((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
```

```
#definir material do enrijecedor transversal de um lado
self.elasticOTS=((200000, 0.3), )
self.plasticOTS=((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
#definir material do enrijecedor transversal dos dois lados
self.elasticDTS=((200000, 0.3), )
self.plasticDTS=((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
#definir material do solido
self.elasticS=((200000, 0.3), )

self.arco0=0.1
self.arcMax=1E036
self.arcMin=1E-020
self.numberOfInc=10000
self.sizeOfElements=20
self.numberOfEigen=13

self.imperfectionFilename=self.baseNameBuckling+'-B'

if self.analysisType=='Buckling':
    self.loadMagnitude=-1.0
    self.modelName=self.baseNameBuckling+'-B'
else:
    self.loadMagnitude=-1.0*Lcr
    self.modelName=self.baseNameNL+'-NL'
```

```
def create_new_model(self):
    mdb.close()
    myBeam=mdb.Model(name=self.modelName, description='')
    del mdb.models['Model-1']
    return myBeam

def create_flange_web_and_stiffeners(self, myBeam):
    flange1Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    flange1Sketch.Line(point1=(-self.bfuSO/2, 0.0), point2=(self.bfuSO/2, 0.0))
    flange1Sketch.HorizontalConstraint(addUndoState=False, entity=flange1Sketch.geometry.findAt((0.0, 0.0), ))
    myBeam.Part(dimensionality=THREE_D, name='Flange1', type=DEFORMABLE_BODY)
    myBeam.parts['Flange1'].BaseShellExtrude(depth=self.L, sketch=flange1Sketch)
    del myBeam.sketches['__profile__']

    flange2Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    flange2Sketch.Line(point1=(-self.bfbSO/2, 0.0), point2=(self.bfbSO/2, 0.0))
    flange2Sketch.HorizontalConstraint(addUndoState=False, entity=flange2Sketch.geometry.findAt((0.0, 0.0), ))
    myBeam.Part(dimensionality=THREE_D, name='Flange2', type=DEFORMABLE_BODY)
    myBeam.parts['Flange2'].BaseShellExtrude(depth=self.L, sketch=flange2Sketch)
    del myBeam.sketches['__profile__']

    web1Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    web1Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
    web1Sketch.VerticalConstraint(addUndoState=False, entity=web1Sketch.geometry.findAt((0.0, 0.0), ))
    myBeam.Part(dimensionality=THREE_D, name='Web1', type=DEFORMABLE_BODY)
```

```
myBeam.parts['Web1'].BaseShellExtrude(depth=self.dP1, sketch=web1Sketch)
del myBeam.sketches['__profile__']

web2Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web2Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web2Sketch.VerticalConstraint(addUndoState=False, entity=web2Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web2', type=DEFORMABLE_BODY)
myBeam.parts['Web2'].BaseShellExtrude(depth=120.0, sketch=web2Sketch)
del myBeam.sketches['__profile__']

web3Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web3Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web3Sketch.VerticalConstraint(addUndoState=False, entity=web3Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web3', type=DEFORMABLE_BODY)
myBeam.parts['Web3'].BaseShellExtrude(depth=self.lSO, sketch=web3Sketch)
del myBeam.sketches['__profile__']

web4Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web4Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web4Sketch.VerticalConstraint(addUndoState=False, entity=web4Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web4', type=DEFORMABLE_BODY)
myBeam.parts['Web4'].BaseShellExtrude(depth=self.loverhang, sketch=web4Sketch)
del myBeam.sketches['__profile__']

web5Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
```

```
web5Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web5Sketch.VerticalConstraint(addUndoState=False, entity=web5Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web5', type=DEFORMABLE_BODY)
myBeam.parts['Web5'].BaseShellExtrude(depth=2*self.loverhang, sketch=web5Sketch)
del myBeam.sketches['__profile__']

web6Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web6Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web6Sketch.VerticalConstraint(addUndoState=False, entity=web6Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web6', type=DEFORMABLE_BODY)
myBeam.parts['Web6'].BaseShellExtrude(depth=self.loverhang, sketch=web6Sketch)
del myBeam.sketches['__profile__']

web7Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web7Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web7Sketch.VerticalConstraint(addUndoState=False, entity=web7Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web7', type=DEFORMABLE_BODY)
myBeam.parts['Web7'].BaseShellExtrude(depth=2370.0, sketch=web7Sketch)
del myBeam.sketches['__profile__']

web8Sketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web8Sketch.Line(point1=(0.0, -self.daSO/2), point2=(0.0, self.daSO/2))
web8Sketch.VerticalConstraint(addUndoState=False, entity=web8Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web8', type=DEFORMABLE_BODY)
myBeam.parts['Web8'].BaseShellExtrude(depth=self.dP3, sketch=web8Sketch)
```

```
del myBeam.sketches['__profile__']

twosidedtransversestiffenerSketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
twosidedtransversestiffenerSketch.rectangle(point1=(-self.wdts, -self.daSO/2), point2=(self.wdts, self.daSO/2))
myBeam.Part(dimensionality=THREE_D, name='TwoSidedTransverseStiffener', type=DEFORMABLE_BODY)
myBeam.parts['TwoSidedTransverseStiffener'].BaseShell(sketch=twosidedtransversestiffenerSketch)
del myBeam.sketches['__profile__']

onesidedtransversestiffenerSketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
onesidedtransversestiffenerSketch.rectangle(point1=(0.0, -self.daSO/2), point2=(self.wots, self.daSO/2))
myBeam.Part(dimensionality=THREE_D, name='OneSidedTransverseStiffener', type=DEFORMABLE_BODY)
myBeam.parts['OneSidedTransverseStiffener'].BaseShell(sketch=onesidedtransversestiffenerSketch)
del myBeam.sketches['__profile__']

longitudinalopenstiffenerSketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
longitudinalopenstiffenerSketch.Line(point1=(0.0, 0.0), point2=(self.wlosSO, 0.0))
longitudinalopenstiffenerSketch.HorizontalConstraint(addUndoState=False, entity=longitudinalopenstiffenerSketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='LongitudinalOpenStiffener', type=DEFORMABLE_BODY)
myBeam.parts['LongitudinalOpenStiffener'].BaseShellExtrude(depth=self.losSO, sketch=longitudinalopenstiffenerSketch)
del myBeam.sketches['__profile__']

longitudinalclosedstiffenerSketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
g, v, d, c = longitudinalclosedstiffenerSketch.geometry, longitudinalclosedstiffenerSketch.vertices, longitudinalclosedstiffenerSketch.dimensions,
longitudinalclosedstiffenerSketch.constraints
longitudinalclosedstiffenerSketch.Line(point1=(0.0, -self.wlcs), point2=(self.wlcs, -self.wlcs/2))
```

```
    longitudinalclosedstiffenerSketch.Line(point1=(self.wlcs, -self.wlcs/2), point2=(self.wlcs, self.wlcs/2))
    longitudinalclosedstiffenerSketch.Line(point1=(self.wlcs, self.wlcs/2), point2=(0.0, self.wlcs))
    longitudinalclosedstiffenerSketch.VerticalConstraint(addUndoState=False, entity=g[3])
    myBeam.Part(dimensionality=THREE_D, name='LongitudinalClosedStiffener', type=DEFORMABLE_BODY)
    myBeam.parts['LongitudinalClosedStiffener'].BaseShellExtrude(depth=self.lcs, sketch=longitudinalclosedstiffenerSketch)
    del myBeam.sketches['__profile__']

def create_solid(self, myBeam):
    solidSketch=myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    solidSketch.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(100.0, 0.0))
    myBeam.Part(dimensionality=THREE_D, name='Solid', type=DEFORMABLE_BODY)
    myBeam.parts['Solid'].BaseSolidExtrude(depth=60.0, sketch=solidSketch)
    del myBeam.sketches['__profile__']

def create_material(self, myBeam):
    myBeam.Material(name='Steel-Flange1')
    myBeam.materials['Steel-Flange1'].Elastic(table=self.elasticF)
    if self.analysisType!='Buckling':
        myBeam.materials['Steel-Flange1'].Plastic(table=self.plasticF)

    myBeam.Material(name='Steel-Flange2')
    myBeam.materials['Steel-Flange2'].Elastic(table=self.elasticF)
    if self.analysisType!='Buckling':
        myBeam.materials['Steel-Flange2'].Plastic(table=self.plasticF)
```

```
myBeam.Material(name='Steel-Web1')
myBeam.materials['Steel-Web1'].Elastic(table=self.elasticW8)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web1'].Plastic(table=self.plasticW8)

myBeam.Material(name='Steel-Web2')
myBeam.materials['Steel-Web2'].Elastic(table=self.elasticW7)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web2'].Plastic(table=self.plasticW7)

myBeam.Material(name='Steel-Web3')
myBeam.materials['Steel-Web3'].Elastic(table=self.elasticW7)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web3'].Plastic(table=self.plasticW7)

myBeam.Material(name='Steel-Web4')
myBeam.materials['Steel-Web4'].Elastic(table=self.elasticOTS)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web4'].Plastic(table=self.plasticOTS)

myBeam.Material(name='Steel-Web5')
myBeam.materials['Steel-Web5'].Elastic(table=self.elasticOTS)
if self.analysisType != 'Buckling':
    myBeam.materials['Steel-Web5'].Plastic(table=self.plasticOTS)
```

```
myBeam.Material(name='Steel-Web6')
myBeam.materials['Steel-Web6'].Elastic(table=self.elasticOTS)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web6'].Plastic(table=self.plasticOTS)

myBeam.Material(name='Steel-Web7')
myBeam.materials['Steel-Web7'].Elastic(table=self.elasticW8)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web7'].Plastic(table=self.plasticW8)

myBeam.Material(name='Steel-Web8')
myBeam.materials['Steel-Web8'].Elastic(table=self.elasticW8)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-Web8'].Plastic(table=self.plasticW8)

myBeam.Material(name='Steel-LongitudinalOpenStiffener')
myBeam.materials['Steel-LongitudinalOpenStiffener'].Elastic(table=self.elasticLOS)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-LongitudinalOpenStiffener'].Plastic(table=self.plasticLOS)

myBeam.Material(name='Steel-LongitudinalClosedStiffener')
myBeam.materials['Steel-LongitudinalClosedStiffener'].Elastic(table=self.elasticLCS)
if self.analysisType!='Buckling':
    myBeam.materials['Steel-LongitudinalClosedStiffener'].Plastic(table=self.plasticLCS)
```

```
        myBeam.Material(name='Steel-TwoSidedTransverseStiffener')
        myBeam.materials['Steel-TwoSidedTransverseStiffener'].Elastic(table=self.elasticDTS)
        if self.analysisType!='Buckling':
            myBeam.materials['Steel-TwoSidedTransverseStiffener'].Plastic(table=self.plasticDTS)

        myBeam.Material(name='Steel-OneSidedTransverseStiffener')
        myBeam.materials['Steel-OneSidedTransverseStiffener'].Elastic(table=self.elasticOTS)
        if self.analysisType!='Buckling':
            myBeam.materials['Steel-OneSidedTransverseStiffener'].Plastic(table=self.plasticOTS)

        myBeam.Material(name='Steel-Solid')
        myBeam.materials['Steel-Solid'].Elastic(table=self.elasticS)

    def create_sections(self, myBeam):
        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Flange1', name='Flange1', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tfuSO, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Flange2', name='Flange2', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tfbSO, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web1', name='Web1', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw8, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)
```

```
myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web2', name='Web2', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.twSO, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web3', name='Web3', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.twSO, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web4', name='Web4', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tots, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web5', name='Web5', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tots, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web6', name='Web6', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tots, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web7', name='Web7', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw8, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)
```

```
myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web8', name='Web8', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw8, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-LongitudinalOpenStiffener',
name='LongitudinalOpenStiffener', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tlosSO, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-LongitudinalClosedStiffener',
name='LongitudinalClosedStiffener', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tlcs, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-TwoSidedTransverseStiffener',
name='TwoSidedTransverseStiffener', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tdts, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-OneSidedTransverseStiffener',
name='OneSidedTransverseStiffener', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tots, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousSolidSection(name='Solid', material='Steel-Solid', thickness=None)
```

```
    def assign_sections(self, myBeam):
        myBeam.parts['Flange1'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Flange1'].faces.findAt(((0.49*self.bfuSO, 0.0, 0.0),(-0.49*self.bfuSO, 0.0, self.L)), )), sectionName='Flange1',
thicknessAssignment=FROM_SECTION)

        myBeam.parts['Flange2'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Flange2'].faces.findAt(((0.49*self.bfbSO, 0.0, 0.0),(-0.49*self.bfbSO, 0.0, self.L)), )), sectionName='Flange2',
thicknessAssignment=FROM_SECTION)

        myBeam.parts['Web1'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Web1'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, self.dP1)), )), sectionName='Web1',
thicknessAssignment=FROM_SECTION)

        myBeam.parts['Web2'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Web2'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, 120.0)), )), sectionName='Web2',
thicknessAssignment=FROM_SECTION)

        myBeam.parts['Web3'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Web3'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, self.lSO)), )), sectionName='Web3',
thicknessAssignment=FROM_SECTION)

        myBeam.parts['Web4'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Web4'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, self.loverhang)), )), sectionName='Web4',
thicknessAssignment=FROM_SECTION)
```

```
myBeam.parts['Web5'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web5'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, 2*self.loverhang)), )), sectionName='Web5',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web6'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web6'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, self.loverhang)), )), sectionName='Web6',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web7'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web7'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, 2370.0)), )), sectionName='Web7',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web8'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web8'].faces.findAt(((0.0, -0.49*self.daSO, 0.0), (0.0, 0.49*self.daSO, self.dP3)), )), sectionName='Web8',
thicknessAssignment=FROM_SECTION)

myBeam.parts['LongitudinalOpenStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['LongitudinalOpenStiffener'].faces.findAt(((0.0, 0.0, 0.0), (self.wlosSO, 0.0, self.losSO)), )), sectionName='LongitudinalOpenStiffener',
thicknessAssignment=FROM_SECTION)

myBeam.parts['LongitudinalClosedStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['LongitudinalClosedStiffener'].faces.getByBoundingBox(0.0, -self.wlcs, 0.0, self.wlcs, self.wlcs, self.lcs)),
sectionName='LongitudinalClosedStiffener', thicknessAssignment=FROM_SECTION)
```

```
        myBeam.parts['TwoSidedTransverseStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['TwoSidedTransverseStiffener'].faces.findAt(((0.0, -0.48*self.daSO, 0.0), (0.0, 0.48*self.daSO, 0.0)), )),
sectionName='TwoSidedTransverseStiffener', thicknessAssignment=FROM_SECTION)

        myBeam.parts['OneSidedTransverseStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['OneSidedTransverseStiffener'].faces.findAt(((0.0, -0.48*self.daSO, 0.0), (0.0, 0.48*self.daSO, 0.0)), )),
sectionName='OneSidedTransverseStiffener', thicknessAssignment=FROM_SECTION)

        delta=1
        myBeam.parts['Solid'].SectionAssignment(region=regionToolset.Region(cells=myBeam.parts['Solid'].cells.getByBoundingBox(-100.0-delta, -100.0-delta, 0.0-delta,
100.0+delta, 100.0+delta, 60.0+delta)),
            sectionName='Solid', offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTION)

    def create_beam_instance(self, myBeam):
        myBeam.rootAssembly.DatumCsysByDefault(CARTESIAN)
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web1', part=myBeam.parts['Web1'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web2', part=myBeam.parts['Web2'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web3', part=myBeam.parts['Web3'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web4', part=myBeam.parts['Web4'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web5', part=myBeam.parts['Web5'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web6', part=myBeam.parts['Web6'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web7', part=myBeam.parts['Web7'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='Web8', part=myBeam.parts['Web8'])
        myBeam.rootAssembly.translate(instanceList=('Web2', ), vector=(0.0, 0.0, self.dP1))
        myBeam.rootAssembly.translate(instanceList=('Web3', ), vector=(0.0, 0.0, self.dP1+120.0))
```

```
myBeam.rootAssembly.translate(instanceList=('Web4', ), vector=(0.0, 0.0, self.dP1+120.0+self.lSO))
myBeam.rootAssembly.translate(instanceList=('Web5', ), vector=(0.0, 0.0, self.dP1+120.0+self.lSO+self.loverhang))
myBeam.rootAssembly.translate(instanceList=('Web6', ), vector=(0.0, 0.0, self.dP1+120.0+self.lSO+3*self.loverhang))
myBeam.rootAssembly.translate(instanceList=('Web7', ), vector=(0.0, 0.0, self.dP1+120.0+self.lSO+4*self.loverhang))
myBeam.rootAssembly.translate(instanceList=('Web8', ), vector=(0.0, 0.0, self.L-self.dP3))
myBeam.rootAssembly.Instance(dependent=OFF, name='Flange1', part=myBeam.parts['Flange1'])
myBeam.rootAssembly.Instance(dependent=OFF, name='Flange2', part=myBeam.parts['Flange2'])
myBeam.rootAssembly.translate(instanceList=('Flange1', ), vector=(0.0, self.daSO/2, 0.0))
myBeam.rootAssembly.translate(instanceList=('Flange2', ), vector=(0.0, -self.daSO/2, 0.0))
myBeam.rootAssembly.Instance(dependent=OFF, name='LongitudinalOpenStiffener', part=myBeam.parts['LongitudinalOpenStiffener'])
myBeam.rootAssembly.translate(instanceList=('LongitudinalOpenStiffener', ), vector=(0.0, self.daSO/2-tfuSO/2-self.b, self.loverhang))
myBeam.rootAssembly.Instance(dependent=OFF, name='LongitudinalClosedStiffener', part=myBeam.parts['LongitudinalClosedStiffener'])
myBeam.rootAssembly.translate(instanceList=('LongitudinalClosedStiffener', ), vector=(0.0, self.daSO/2-tfuSO/2-self.b, self.loverhang+self.losSO))
myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener1', part=myBeam.parts['TwoSidedTransverseStiffener'])
myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener2', part=myBeam.parts['TwoSidedTransverseStiffener'])
myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener3', part=myBeam.parts['TwoSidedTransverseStiffener'])
myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener4', part=myBeam.parts['TwoSidedTransverseStiffener'])
myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener5', part=myBeam.parts['TwoSidedTransverseStiffener'])
myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener6', part=myBeam.parts['TwoSidedTransverseStiffener'])
myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener2', ), vector=(0.0, 0.0, self.loverhang+self.losSO-self.loverhang))
myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener3', ), vector=(0.0, 0.0, self.loverhang+self.losSO-self.loverhang+self.loverhang))
myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener4', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lcs+2*self.loverhang))
myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener5', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lcs+3*self.loverhang))
myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener6', ), vector=(0.0, 0.0, self.L))
myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener1', part=myBeam.parts['OneSidedTransverseStiffener'])
```

```
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener2', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener3', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener4', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener5', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener6', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener7', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener1', ), vector=(0.0, 0.0, self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener2', ), vector=(0.0, 0.0, self.loverhang+self.lP1))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener3', ), vector=(0.0, 0.0, self.loverhang+self.lP1+self.lP2))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener4', ), vector=(0.0, 0.0, self.loverhang+self.lP1+2*self.lP2))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener5', ), vector=(0.0, 0.0, self.L-self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener6', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lP3))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener7', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lP3-self.lP4))
        myBeam.rootAssembly.Instance(dependent=OFF, name='Solid', part=myBeam.parts['Solid'])
        myBeam.rootAssembly.rotate(instanceList=('Solid', ), axisPoint=(-100.0, 0.0, 60.0), axisDirection=(100.0, 0.0, 0.0), angle=270.0)
        myBeam.rootAssembly.translate(instanceList=('Solid', ), vector=(0.0, 60.0+self.daSO/2, self.losSO+40.0))

    def partition_face(self, myBeam):

        xyPlane1=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane1.id],  faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane2=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1, principalPlane=XYPLANE)
```

```
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane2.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane3=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane3.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane4=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+2*self.lP2, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane4.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane5=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+2*self.lP2+self.lSO, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane5.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane6=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+2*self.lP2+self.lSO+self.loverhang/2, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane6.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane7=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+2*self.lP2+self.lSO+self.loverhang, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane7.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane8=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+2*self.lP2+self.lSO+3*self.loverhang, principalPlane=XYPLANE)
```

```
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane8.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane9=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+2*self.lP2+self.lSO+4*self.loverhang, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane9.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane10=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.L-self.loverhang, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane10.id],
faces=myBeam.rootAssembly.instances['Flange1'].faces + myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane11=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.L-self.loverhang-self.lP3, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane11.id],
faces=myBeam.rootAssembly.instances['Flange1'].faces + myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane12=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.L-self.loverhang-self.lP3-self.lP4, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane12.id],
faces=myBeam.rootAssembly.instances['Flange1'].faces + myBeam.rootAssembly.instances['Flange2'].faces)

        xzPlane1=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.daSO/2-tfuSO/2-self.b, principalPlane=XZPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane1.id],
faces=myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'].faces +\
            myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener4'].faces +\
            myBeam.rootAssembly.instances['TwoSidedTransverseStiffener5'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'].faces +\
            myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'].faces +\
```

```
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener4'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener5'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener6'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener7'].faces)

    xzPlane2=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=0.0, principalPlane=XZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane2.id],
faces=myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener4'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener5'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener4'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener5'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener6'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener7'].faces +\
        myBeam.rootAssembly.instances['Web1'].faces + myBeam.rootAssembly.instances['Web2'].faces + myBeam.rootAssembly.instances['Web3'].faces +\
        myBeam.rootAssembly.instances['Web4'].faces + myBeam.rootAssembly.instances['Web5'].faces + myBeam.rootAssembly.instances['Web6'].faces +\
        myBeam.rootAssembly.instances['Web7'].faces + myBeam.rootAssembly.instances['Web8'].faces)

    delta=1
    yzPlane1=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=-self.wdts, principalPlane=YZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[yzPlane1.id],  faces=myBeam.rootAssembly.instances['Flange1'].faces
+\
        myBeam.rootAssembly.instances['Flange2'].faces)

    yzPlane2=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.wdts, principalPlane=YZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[yzPlane2.id],  faces=myBeam.rootAssembly.instances['Flange1'].faces
+\
        myBeam.rootAssembly.instances['Flange2'].faces)
```

```
        yzPlane3=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.wots, principalPlane=YZPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[yzPlane3.id],  faces=myBeam.rootAssembly.instances['Flange1'].faces
+\
            myBeam.rootAssembly.instances['Flange2'].faces)

        yzPlane4=myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=-self.wots, principalPlane=YZPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[yzPlane4.id],  faces=myBeam.rootAssembly.instances['Flange1'].faces
+\
            myBeam.rootAssembly.instances['Flange2'].faces)

        delta=3
        myBeam.rootAssembly.PartitionFaceByIntersectFace(faces=myBeam.rootAssembly.instances['Flange1'].faces.getByBoundingBox(-self.bfuSO/2-delta,        self.daSO/2-
delta, self.loverhang+self.lP1+2*self.lP2+self.lSO-delta,
            self.bfuSO/2+delta,                                    self.daSO/2+delta,                                    2*self.loverhang+self.lP1+2*self.lP2+self.lSO+delta),
cuttingFaces=myBeam.rootAssembly.instances['Solid'].faces.getByBoundingBox(
            -100.0-delta,          self.daSO/2-delta,          self.loverhang+self.lP1+2*self.lP2+self.lSO-delta,          100.0+delta,          self.daSO/2+delta,
2*self.loverhang+self.lP1+2*self.lP2+self.lSO+delta))

        myBeam.rootAssembly.InstanceFromBooleanMerge(domain=GEOMETRY,                                    instances=(myBeam.rootAssembly.instances['Web1'],
myBeam.rootAssembly.instances['Web2'], myBeam.rootAssembly.instances['Web3'],
            myBeam.rootAssembly.instances['Web4'],          myBeam.rootAssembly.instances['Web5'],          myBeam.rootAssembly.instances['Web6'],
myBeam.rootAssembly.instances['Web7'], myBeam.rootAssembly.instances['Web8'],
            myBeam.rootAssembly.instances['Flange1'],          myBeam.rootAssembly.instances['Flange2'],          myBeam.rootAssembly.instances['LongitudinalOpenStiffener'],
myBeam.rootAssembly.instances['LongitudinalClosedStiffener'],
```

```
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener1'], myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'],
myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'],
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener4'], myBeam.rootAssembly.instances['TwoSidedTransverseStiffener5'],
myBeam.rootAssembly.instances['TwoSidedTransverseStiffener6'],
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'], myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'],
myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'],
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener4'], myBeam.rootAssembly.instances['OneSidedTransverseStiffener5'],
myBeam.rootAssembly.instances['OneSidedTransverseStiffener6'],
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener7']), keepIntersections=ON, name='Beam', originalInstances=DELETE)

    myInstance1=myBeam.rootAssembly.instances['Beam-1']
    myBeam.rootAssembly.makeIndependent(instances=(myInstance1, ))

  def surfaces(self, myBeam):
    delta=3
    myBeam.rootAssembly.Surface(side2Faces=myBeam.rootAssembly.instances['Beam-1'].faces.getByBoundingBox(-100.0-delta, self.daSO/2-delta,
self.loverhang+self.lP1+2*self.lP2+self.lSO-delta,
        100.0+delta, self.daSO/2+delta, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO+delta), name='SurfaceBeam-Tie')
    myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['Solid'].faces.getByBoundingBox(-100.0-delta, self.daSO/2-delta,
self.loverhang+self.lP1+2*self.lP2+self.lSO-delta,
        100.0+delta, self.daSO/2+delta, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO+delta), name='SurfaceSolid-Tie')
    myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['Solid'].faces.getByBoundingBox(-100.0-delta, self.daSO/2+60.0-delta,
self.loverhang+self.lP1+2*self.lP2+self.lSO-delta,
        100.0+delta, self.daSO/2+60.0+delta, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO+delta), name='SurfaceSolid-Load')
```

```
    def mesh_instance(self, myBeam):
        myInstance1=myBeam.rootAssembly.instances['Beam-1']
        myBeam.rootAssembly.setMeshControls(regions=myInstance1.faces, technique=STRUCTURED)
        myBeam.rootAssembly.setElementType(elemTypes=(ElemType(elemCode=S4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF), ElemType(elemCode=S3,
elemLibrary=STANDARD)), regions=Region(myInstance1.faces))
        myBeam.rootAssembly.seedPartInstance(deviationFactor=0.1, minSizeFactor=0.1, regions=(myInstance1, ), size=self.sizeOfElements)
        myBeam.rootAssembly.generateMesh(regions=(myInstance1, ))

        elemType1 = mesh.ElemType(elemCode=C3D8R, elemLibrary=STANDARD, kinematicSplit=AVERAGE_STRAIN, secondOrderAccuracy=OFF,
hourglassControl=DEFAULT, distortionControl=DEFAULT)
        elemType2 = mesh.ElemType(elemCode=C3D6, elemLibrary=STANDARD)
        elemType3 = mesh.ElemType(elemCode=C3D4, elemLibrary=STANDARD)
        delta=3
        myBeam.rootAssembly.setElementType(regions=(myBeam.rootAssembly.instances['Solid'].cells.getByBoundingBox(-100.0-delta, self.daSO/2-delta,
self.loverhang+self.lP1+2*self.lP2+self.lSO-delta,
            100.0+delta, self.daSO/2+60.0+delta, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO+delta), ), elemTypes=(elemType1, elemType2, elemType3))
        myBeam.rootAssembly.seedPartInstance(regions=(myBeam.rootAssembly.instances['Solid'], ), size=5.0, deviationFactor=0.1, minSizeFactor=0.1)
        myBeam.rootAssembly.generateMesh(regions=(myBeam.rootAssembly.instances['Solid'], ))

    def create_sets(self, myBeam):
        myInstance1=myBeam.rootAssembly.instances['Beam-1']

        delta=1
        myBeam.rootAssembly.Set(edges=
            myInstance1.edges.findAt(((0.1, -self.daSO/2, self.loverhang), ), ((0.9*self.wdts, -self.daSO/2, self.loverhang), ), ((1.01*self.wdts, -self.daSO/2, self.loverhang), ), ((
```

```
        -0.1, -self.daSO/2, self.loverhang), ), ((-0.9*self.wdts, -self.daSO/2, self.loverhang), ), ((-1.01*self.wdts, -self.daSO/2, self.loverhang), ), ((
        0.1, self.daSO/2, self.loverhang), ), ((0.9*self.wdts, self.daSO/2, self.loverhang), ), ((1.01*self.wdts, self.daSO/2, self.loverhang), ), ((
        -0.1, self.daSO/2, self.loverhang), ), ((-0.9*self.wdts, self.daSO/2, self.loverhang), ), ((-1.01*self.wdts, self.daSO/2, self.loverhang), ), ((
        0.0, -0.1, self.loverhang), ), ((0.0, 0.1, self.loverhang), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang), ), ((
        0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang), ), ((0.1, -self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((0.9*self.wdts, -self.daSO/2,
self.loverhang+self.losSO+self.lcs), ), ((
        1.01*self.wdts, -self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((
        -0.1, -self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((-0.9*self.wdts, -self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((-1.01*self.wdts, -self.daSO/2,
self.loverhang+self.losSO+self.lcs), ), ((
        0.1, self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((0.9*self.wdts, self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((1.01*self.wdts, self.daSO/2,
self.loverhang+self.losSO+self.lcs), ), ((
        -0.1, self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((-0.9*self.wdts, self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((-1.01*self.wdts, self.daSO/2,
self.loverhang+self.losSO+self.lcs), ), ((
        0.0, -0.1, self.loverhang+self.losSO+self.lcs), ), ((0.0, 0.1, self.loverhang+self.losSO+self.lcs), ), ((
        0.0, 0.7*self.daSO/2, self.loverhang+self.losSO+self.lcs), ), ((0.0, 1.1*(self.daSO/2-self.tf/2-self.b), self.loverhang+self.losSO+self.lcs), ), ((
        0.0, 0.9*(self.daSO/2-self.tf/2-self.b), self.loverhang+self.losSO+self.lcs), ), ), name='Set-BC1')

    myBeam.rootAssembly.Set(edges=
      myInstance1.edges.findAt(((0.0, -0.1, self.loverhang), ), ((0.0, 0.1, self.loverhang), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang), ), ((
        0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang), ), ((0.0, -0.1, self.loverhang+self.losSO+self.lcs), ), ((0.0, 0.1, self.loverhang+self.losSO+self.lcs), ), ((
        0.0, 0.7*self.da/2, self.loverhang+self.losSO+self.lcs), ), ((0.0, 1.1*(self.da/2-self.tf/2-self.b), self.loverhang+self.losSO+self.lcs), ), ((
        0.0, 0.9*(self.da/2-self.tf/2-self.b), self.loverhang+self.losSO+self.lcs), ), ), name='Set-BC2')

    myBeam.rootAssembly.Set(name='Set-BC3', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.loverhang+self.losSO+self.lcs), )))
```

```
myBeam.rootAssembly.Set(name='Set-LR1', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, self.loverhang), )))

myBeam.rootAssembly.Set(name='Set-LR2', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, self.loverhang+self.lP1), )))

myBeam.rootAssembly.Set(name='Set-LR3', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, self.loverhang+self.lP1+self.lP2), )))

myBeam.rootAssembly.Set(edges=
  myInstance1.edges.findAt(((0.1, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((0.9*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((1.01*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((
    -0.1, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((-0.9*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((-1.01*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((
    0.1, self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((0.9*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((1.01*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((
    -0.1, self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((-0.9*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((-1.01*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2), ), ((
    0.0, -0.1, self.loverhang+self.lP1+2*self.lP2), ), ((0.0, 0.1, self.loverhang+self.lP1+2*self.lP2), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang+self.lP1+2*self.lP2), ), ((
    0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang+self.lP1+2*self.lP2), ), ), name='Set-LR4')

myBeam.rootAssembly.Set(edges=
  myInstance1.edges.findAt(((0.1, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), ), ((0.9*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), ), ((1.01*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), ), ((
    -0.1, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), ), ((-0.9*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), ), ((-1.01*self.wdts, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), ), ((
```

0.1, self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((0.9*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((1.01*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((

-0.1, self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((-0.9*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((-1.01*self.wdts, self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((

0.0, -0.1, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((0.0, 0.1, self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang+self.lP1+2*self.lP2+self.lSO),), ((

0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang+self.lP1+2*self.lP2+self.lSO),),), name='Set-LR5')

myBeam.rootAssembly.Set(name='Set-LR6', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO),)))

myBeam.rootAssembly.Set(name='Set-LR7', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, 4*self.loverhang+self.lP1+2*self.lP2+self.lSO),)))

myBeam.rootAssembly.Set(name='Set-LR8', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, 5*self.loverhang+self.lP1+2*self.lP2+self.lSO),)))

myBeam.rootAssembly.Set(name='Set-LR9', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, 5*self.loverhang+self.lP1+2*self.lP2+self.lSO+2250.0),)))

myBeam.rootAssembly.Set(name='Set-LR10', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, 5*self.loverhang+self.lP1+2*self.lP2+self.lSO+self.lP4+2250.0),)))

myBeam.rootAssembly.Set(name='Set-LR11', vertices=myInstance1.vertices.findAt(((0.0, self.daSO/2, 5*self.loverhang+self.lP1+2*self.lP2+self.lSO+self.lP4+2250.0+self.lP3),)))

myBeam.rootAssembly.Set(name='Set-LR12', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO),)))

```
    myBeam.rootAssembly.Set(name='Set-LR13', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-LR14', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, 4*self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-LR15', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, 5*self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-V1', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2), )))
    myBeam.rootAssembly.Set(name='Set-V2', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-V3', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, 2*self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-V4', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, 4*self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-V5', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, 5*self.loverhang+self.lP1+2*self.lP2+self.lSO), )))
    myBeam.rootAssembly.Set(name='Set-V6', vertices=myInstance1.vertices.findAt(((0.0, -self.daSO/2, self.L-self.loverhang-self.lP3-self.lP4), )))

def set_boundary_conditions(self, myBeam):
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BC1',
        region=myBeam.rootAssembly.sets['Set-BC1'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=SET)
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BC2',
        region=myBeam.rootAssembly.sets['Set-BC2'], u1=UNSET, u2=SET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BC3',
        region=myBeam.rootAssembly.sets['Set-BC3'], u1=UNSET, u2=UNSET, u3=SET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
```

```
        myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='LR4',
            region=myBeam.rootAssembly.sets['Set-LR4'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
        myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='LR5',
            region=myBeam.rootAssembly.sets['Set-LR5'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

    def create_constraint(self, myBeam):
        myBeam.Tie(name='Tie', master=myBeam.rootAssembly.surfaces['SurfaceSolid-Tie'], slave=myBeam.rootAssembly.surfaces['SurfaceBeam-Tie'],
            positionToleranceMethod=COMPUTED, adjust=ON, tieRotations=ON, thickness=ON)

    def create_step(self, myBeam):
        if self.analysisType=='Buckling':
            myBeam.BuckleStep(eigensolver=SUBSPACE, name='Step-1', numEigen=self.numberOfEigen, maxIterations=1000, previous='Initial')
        else:
            myBeam.StaticRiksStep(initialArcInc=self.arco0, minArcInc=self.arcMin, maxArcInc=self.arcMax, name='Step-1', nlgeom=ON, previous='Initial', maxNumInc =
self.numberOfInc)

    def create_load(self, myBeam):
        import math
        area=math.pi*100*100
        loadapplied=self.loadMagnitude/area

        myBeam.Pressure(name='Load-1',
            createStepName='Step-1', region=myBeam.rootAssembly.surfaces['SurfaceSolid-Load'], distributionType=UNIFORM, field='', magnitude=-loadapplied)

    def createResidualStress(self, myBeam):
```

```
myInstance1 = myBeam.rootAssembly.instances['Beam-1']
delta = 12.0#mm

# Mesa Superior
delta = 10.0
myBeam.rootAssembly.Set(edges=
myInstance1.edges.findAt(((0.1, self.daSO/2, self.loverhang), ), ((0.9*self.wdts, self.daSO/2, self.loverhang), ), ((1.01*self.wdts, self.daSO/2, self.loverhang), ), ((
    -0.1, self.daSO/2, self.loverhang), ), ((-0.9*self.wdts, self.daSO/2, self.loverhang), ), ((-1.01*self.wdts, self.daSO/2, self.loverhang), ), ), name='Set-FlangeUNodes')
numberFlangeElements = int(len(myBeam.rootAssembly.sets['Set-FlangeUNodes'].nodes)-3)
sizeOfFlangeElements = (self.bfuSO)/(numberFlangeElements)

# Definir tensao residual para as mesas
for x in range(int(numberFlangeElements/2)):
    setName = 'TRFU' + str(x+1)
    x1 = x*sizeOfFlangeElements
    x2 = (x+1)*sizeOfFlangeElements
    y = self.daSO/2
    f1 = ((3.397*self.fyf)/((self.bfuSO/2.0)*(self.bfuSO/2.0)))*(x1*x1)-((3.497*self.fyf)/(self.bfuSO/2.0))*x1+(0.6*self.fyf)
    f2 = ((3.397*self.fyf)/((self.bfuSO/2.0)*(self.bfuSO/2.0)))*(x2*x2)-((3.497*self.fyf)/(self.bfuSO/2.0))*x2+(0.6*self.fyf)
    elementS = (f2+f1)/2.0

    e1 = myInstance1.elements.getByBoundingBox(x1-delta, y-delta, 3585.0-delta, x2+delta, y+delta, 5085.0+delta)
    e2 = myInstance1.elements.getByBoundingBox(-x2-delta, y-delta, 3585.0-delta, -x1+delta, y+delta, 5085.0+delta)

    elements1 = e1+ e2
```

```
    myBeam.rootAssembly.Set(elements = elements1, name=setName)

    myBeam.Stress(name='TRFU'+str(x+1), region=myBeam.rootAssembly.sets['TRFU'+str(x+1)], distributionType=UNIFORM, sigma11=0.0,
        sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

# Mesa Inferior
delta = 10.0
myBeam.rootAssembly.Set(edges=
myInstance1.edges.findAt(((0.1, -self.daSO/2, self.loverhang), ), ((0.9*self.wdts, -self.daSO/2, self.loverhang), ), ((1.01*self.wdts, -self.daSO/2, self.loverhang), ), ((
        -0.1, -self.daSO/2, self.loverhang), ), ((-0.9*self.wdts, -self.daSO/2, self.loverhang), ), ((-1.01*self.wdts, -self.daSO/2, self.loverhang), ), ), name='Set-FlangeBNodes')
numberFlangeElements = int(len(myBeam.rootAssembly.sets['Set-FlangeBNodes'].nodes)-3)
sizeOfFlangeElements = (self.bfbSO)/(numberFlangeElements)

# Definir tensao residual para as mesas
for x in range(int(numberFlangeElements/2)):
    setName = 'TRFB' + str(x+1)
    x1 = x*sizeOfFlangeElements
    x2 = (x+1)*sizeOfFlangeElements
    y = self.daSO/2
    f1 = ((3.397*self.fyf)/((self.bfbSO/2.0)*(self.bfbSO/2.0)))*(x1*x1)-((3.497*self.fyf)/(self.bfbSO/2.0))*x1+(0.6*self.fyf)
    f2 = ((3.397*self.fyf)/((self.bfbSO/2.0)*(self.bfbSO/2.0)))*(x2*x2)-((3.497*self.fyf)/(self.bfbSO/2.0))*x2+(0.6*self.fyf)
    elementS = (f2+f1)/2.0

    if abs(elementS) < (5e-7):
        elementS = 0
```

```
    e1 = myInstance1.elements.getByBoundingBox(x1-delta, -y-delta, 3585.0-delta, x2+delta, -y+delta, 5085.0+delta)
    e2 = myInstance1.elements.getByBoundingBox(-x2-delta, -y-delta, 3585.0-delta, -x1+delta, -y+delta, 5085.0+delta)

    elements1 = e1+e2
    myBeam.rootAssembly.Set(elements = elements1, name=setName)

    myBeam.Stress(name='TRFB'+str(x+1), region=myBeam.rootAssembly.sets['TRFB'+str(x+1)], distributionType=UNIFORM, sigma11=0.0,
        sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

# Alma
delta = 12.0 #mm
myBeam.rootAssembly.Set(edges=
myInstance1.edges.findAt(((0.0, -0.1, self.loverhang), ), ((0.0, 0.1, self.loverhang), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang), ), ((
      0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang), ), ), name='Set-WebNodes')
numberWebElements = int(len(myBeam.rootAssembly.sets['Set-WebNodes'].nodes)-1)
sizeOfWebElements = self.daSO/(numberWebElements)

# Definir tensao residual para a alma
for y in range(int(numberWebElements/2)):
    setName = 'TRW' + str(y+1)
    y1 = y*sizeOfWebElements
    y2 = (y+1)*sizeOfWebElements
    x = 0.0
    g = ((-0.1*self.fyw)*(0.1225*self.daSO+0.0393*self.bfuSO)-(0.6*self.fyw*0.1225*self.daSO)-(0.6*self.fyw*0.0393*self.bfuSO))/(-0.6*self.fyw)
```

```
if (y1 > self.daSO/2.0-g) & (y2 > self.daSO/2.0-g):
    f1 = ((-0.6*self.fyw*(self.daSO/2.0-y1)+0.0735*self.fyw*self.daSO+0.02362*self.fyw*self.bfuSO))/(0.1225*self.daSO+0.0393*self.bfuSO)
    f2 = ((-0.6*self.fyw*(self.daSO/2.0-y2)+0.0735*self.fyw*self.daSO+0.02362*self.fyw*self.bfuSO))/(0.1225*self.daSO+0.0393*self.bfuSO)
    elementS = (f2+f1)/2

if (y1 < self.daSO/2.0-g) & (y2 > self.daSO/2.0-g):
    f1 = -0.1*self.fyw
    f2 = ((-0.6*self.fyw*(self.daSO/2.0-y2)+0.0735*self.fyw*self.daSO+0.02362*self.fyw*self.bfuSO))/(0.1225*self.daSO+0.0393*self.bfuSO)
    elementS = (f2+f1)/2

if (y1 < self.daSO/2.0-g) & (y2 < self.daSO/2.0-g):
    f1 = -0.1*self.fyw
    f2 = -0.1*self.fyw
    elementS = (f2+f1)/2

if abs(elementS) < (5e-7):
    elementS = 0

e1 = myInstance1.elements.getByBoundingBox(x-delta, y1-delta, 3585.0-delta, x+delta, y2+delta, 5085.0+delta)
e2 = myInstance1.elements.getByBoundingBox(x-delta, -y2-delta, 3585.0-delta, x+delta, -y1+delta, 5085.0+delta)

elements1 = e1+e2
myBeam.rootAssembly.Set(elements = elements1, name=setName)
```

```
        myBeam.Stress(name='TRW'+str(y+1), region=myBeam.rootAssembly.sets['TRW'+str(y+1)], distributionType=UNIFORM, sigma11=0.0,
            sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

def edit_text(self, myBeam, textStop, texttoReplace):
    myBeam.keywordBlock.synchVersions(storeNodesAndElements=False)
    linePosition=0
    for keyWords in myBeam.keywordBlock.sieBlocks:
        if keyWords.startswith(textStop):
            myBeam.keywordBlock.replace(linePosition, texttoReplace)
            break
        linePosition=linePosition+1

def create_jobs(self, jobName):
    mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF, explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
        memory=90, memoryUnits=PERCENTAGE, model=self.modelName, modelPrint=OFF, multiprocessingMode=DEFAULT, name=jobName,
        nodalOutputPrecision=SINGLE, numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)

def create_beam(self, nameofModel, jobName):
    self.create_flange_web_and_stiffeners(nameofModel)
    self.create_solid(nameofModel)
    self.create_material(nameofModel)
    self.create_sections(nameofModel)
    self.assign_sections(nameofModel)
    self.create_beam_instance(nameofModel)
    self.partition_face(nameofModel)
```

```
        self.surfaces(nameofModel)
        self.mesh_instance(nameofModel)
        self.create_sets(nameofModel)
        self.set_boundary_conditions(nameofModel)
        self.create_constraint(nameofModel)
        self.create_step(nameofModel)
        self.create_load(nameofModel)


        if self.analysisType != 'Buckling':
            self.createResidualStress(nameofModel)

        if self.analysisType == 'Buckling':
            textStop = '*Output'
            texttoReplace = '*Output, field, variable=PRESELECT\n*NODE FILE\nU,'
        else:
            textStop = '*End Assembly'
            texttoReplace = '*End Assembly\n*IMPERFECTION, FILE=' + self.imperfectionFilename + ', STEP=1\n11, ' + str(1.0) + '\n'

        self.edit_text(nameofModel, textStop, texttoReplace)
        self.create_jobs(jobName)

baseNameBuckling='PanelSO-20mm-Solidwith5mm'
baseNameNL='PanelSO-20mm-Solidwith5mm-355-460'
fyf = 460.00 #tensão de escoamento da mesa
```

```
fyw = 355.00 #tensão de escoamento da alma
#Medidas em milimetros e Newton
da=1522.0 #altura da alma (linha do esqueleto)
bf=320.0 #largura da mesa
tf=22.0 #espessura da mesa
tw7=7.0 #espessura da alma (parte em analise)
tw8=8.0 #espessura da alma (depois do enrijecedor)
tw15=15.0 #espessura da alma entre as cargas
tots=15.0 #espessura do enrijecedor transversal de um lado da alma
wots=120.0 #largura do enrijecedor transversal de um lado da alma
tdts=20.0 #espessura do enrijecedor transversal dos dois lados da alma
wdts=156.0 #largura do enrijecedor transversal dos dois lados da alma
tlos=10.0 #espessura do enrijecedor longitudinal retangular
wlos=90.0 #largura do enrijecedor longitudinal retangular
tlcs=5.0 #espessura do enrijecedor longitudinal trapeizodal
wlcs=80.0 #largura do enrijecedor longitudinal trapeizodal
b=350.0 #distancia da parte debaixo da mesa até o centro de gravidade do enrijecedor longitudinal
los=5085.0
lcs=5675.0
loverhang=200.0

#Medidas em milimetros do painel analisado (SO)
daSO=1522.0
twSO=7.0
lSO=1500.0
```

```
bfuSO=320.0
tfuSO=22.0
bfbSO=320.0
tfbSO=22.0
tlosSO=10.0
wlosSO=90.0
losSO=5085.0

dP1=3465.0
dP2=4790.0
dP3=2905.0
L=11160.0 #comprimento total
lP1=885.0
lP2=1250.0
lP3=1325.0
lP4=1500.0

analysisType='NL'
Lcr=1686520.0

beamExample=FLABeam(baseNameBuckling, baseNameNL, fyf, fyw, da, bf, tf, tw7, tw8, tw15, tots, wots, tdts, wdts, tlos, wlos, tlcs, wlcs, b, los, lcs, loverhang, daSO, twSO,
lSO, bfuSO, tfuSO, bfbSO, tfbSO, tlosSO, wlosSO, losSO, dP1, dP2, dP3, L, lP1, lP2, lP3, lP4, analysisType, Lcr)
beamExampleModel=beamExample.create_new_model()
if analysisType=='Buckling':
    beamExample.create_beam(beamExampleModel, baseNameBuckling+'-B')
```

```
else:
    beamExample.create_beam(beamExampleModel, baseNameNL+'-NL')
```

C.2 Painel da viga esbelta com dois enrijecedores retangulares longitudinais

```
# -*- coding: mbcs -*-

##########################################################################################################################################
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from abaqusConstants import *
from caeModules import *
from driverUtils import executeOnCaeStartup
```

```
import os

session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=100.0, height=142.0)
session.viewports['Viewport: 1'].makeCurrent()
session.viewports['Viewport: 1'].maximize()
executeOnCaeStartup()
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(referenceRepresentation=ON)

########################################################## CREATE A BEAM MODEL ##########################################################

class FLABeam:
    def __init__(self, baseNameBuckling, baseNameNL, fyf, fyw, da, bft, bfb, tf, tw6, tw7, tw15, tots, wots, tdts, wdts, tlos, wlos, tlcs, wlcs, b, b1, los, lcs, loverhang, dP1, dP2,
dP3, L, lP1, lP2, lP3, lP4, lP5, lP6, analysisType, Lcr):
        self.baseNameBuckling = baseNameBuckling
        self.baseNameNL = baseNameNL

        self.fyf = fyf
        self.fyw = fyw
        self.da = da
        self.bft = bft
        self.bfb = bfb
        self.tf = tf
        self.tw6 = tw6
        self.tw7 = tw7
        self.tw15 = tw15
```

```
self.tots = tots
self.wots = wots
self.tdts = tdts
self.wdts = wdts
self.tlos = tlos
self.wlos = wlos
self.tlcs = tlcs
self.wlcs = wlcs
self.b = b
self.b1 = b1
self.los = los
self.lcs = lcs
self.loverhang = loverhang

self.dP1 = dP1
self.dP2 = dP2
self.dP3 = dP3
self.L = L
self.lP1 = lP1
self.lP2 = lP2
self.lP3 = lP3
self.lP4 = lP4
self.lP5 = lP5
self.lP6 = lP6
```

```
self.analysisType = analysisType

#definir material das mesas
self.elasticF = ((200000, 0.3), )
self.plasticF = ((355.63, 0.0), (364.91, 0.0157697), (600.53, 0.1603909), (0.000001, 0.17))
#definir material da alma (parte em análise)
self.elasticW6 = ((200000, 0.3), )
self.plasticW6 = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
#definir material da alma (120mm depois do enrijecedor intermediario)
self.elasticW7 = ((200000, 0.3), )
self.plasticW7 = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
#definir material do enrijecedor retangular longitudinal
self.elasticLOS = ((200000, 0.3), )
self.plasticLOS = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
#definir material do enrijecedor trapeizodal longitudinal
self.elasticLCS = ((200000, 0.3), )
self.plasticLCS = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
#definir material entre as cargas
self.elasticW15 = ((200000, 0.3), )
self.plasticW15 = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
#definir material do enrijecedor transversal de um lado
self.elasticOTS = ((200000, 0.3), )
self.plasticOTS = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
#definir material do enrijecedor transversal dos dois lados
self.elasticDTS = ((200000, 0.3), )
```

```
    self.plasticDTS = ((235.28, 0.0), (240.14, 0.0104808), (402.30, 0.1090825), (0.000001, 0.15))
    #definir material do solido
    self.elasticS = ((200000, 0.3), )

    self.arco0 = 0.001
    self.arcMax = 1E036
    self.arcMin = 1E-020
    self.numberOfInc = 1000
    self.sizeOfElements = 20
    self.numberOfEigen = 12

    self.imperfectionFilename=self.baseNameBuckling + '-B'

    if self.analysisType == 'Buckling':
        self.loadMagnitude = -1.0
        self.modelName = self.baseNameBuckling + '-B'
    else:
        self.loadMagnitude = -1.0*Lcr
        self.modelName = self.baseNameNL  + '-NL'

def create_new_model(self):
    mdb.close()
    myBeam = mdb.Model(name=self.modelName, description='')
    del mdb.models['Model-1']
    return myBeam
```

```
def create_flange_web_and_stiffeners(self, myBeam):
    flange1Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    flange1Sketch.Line(point1=(-self.bft/2, 0.0), point2=(self.bft/2, 0.0))
    flange1Sketch.HorizontalConstraint(addUndoState=False, entity=flange1Sketch.geometry.findAt((0.0, 0.0), ))
    myBeam.Part(dimensionality=THREE_D, name='Flange1', type=DEFORMABLE_BODY)
    myBeam.parts['Flange1'].BaseShellExtrude(depth=self.L, sketch=flange1Sketch)
    del myBeam.sketches['__profile__']

    flange2Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    flange2Sketch.Line(point1=(-self.bfb/2, 0.0), point2=(self.bfb/2, 0.0))
    flange2Sketch.HorizontalConstraint(addUndoState=False, entity=flange2Sketch.geometry.findAt((0.0, 0.0), ))
    myBeam.Part(dimensionality=THREE_D, name='Flange2', type=DEFORMABLE_BODY)
    myBeam.parts['Flange2'].BaseShellExtrude(depth=self.L, sketch=flange2Sketch)
    del myBeam.sketches['__profile__']

    web1Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    web1Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
    web1Sketch.VerticalConstraint(addUndoState=False, entity=web1Sketch.geometry.findAt((0.0, 0.0), ))
    myBeam.Part(dimensionality=THREE_D, name='Web1', type=DEFORMABLE_BODY)
    myBeam.parts['Web1'].BaseShellExtrude(depth=self.dP1, sketch=web1Sketch)
    del myBeam.sketches['__profile__']

    web2Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
    web2Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
```

```
web2Sketch.VerticalConstraint(addUndoState=False, entity=web2Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web2', type=DEFORMABLE_BODY)
myBeam.parts['Web2'].BaseShellExtrude(depth=self.lP5+120.0, sketch=web2Sketch)
del myBeam.sketches['__profile__']

web3Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web3Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
web3Sketch.VerticalConstraint(addUndoState=False, entity=web3Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web3', type=DEFORMABLE_BODY)
myBeam.parts['Web3'].BaseShellExtrude(depth=self.loverhang, sketch=web3Sketch)
del myBeam.sketches['__profile__']

web4Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web4Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
web4Sketch.VerticalConstraint(addUndoState=False, entity=web4Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web4', type=DEFORMABLE_BODY)
myBeam.parts['Web4'].BaseShellExtrude(depth=2*self.loverhang, sketch=web4Sketch)
del myBeam.sketches['__profile__']

web5Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web5Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
web5Sketch.VerticalConstraint(addUndoState=False, entity=web5Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web5', type=DEFORMABLE_BODY)
myBeam.parts['Web5'].BaseShellExtrude(depth=self.loverhang, sketch=web5Sketch)
del myBeam.sketches['__profile__']
```

```
web6Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web6Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
web6Sketch.VerticalConstraint(addUndoState=False, entity=web6Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web6', type=DEFORMABLE_BODY)
myBeam.parts['Web6'].BaseShellExtrude(depth=self.lP6+120.0, sketch=web6Sketch)
del myBeam.sketches['__profile__']

web7Sketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
web7Sketch.Line(point1=(0.0, -self.da/2), point2=(0.0, self.da/2))
web7Sketch.VerticalConstraint(addUndoState=False, entity=web7Sketch.geometry.findAt((0.0, 0.0), ))
myBeam.Part(dimensionality=THREE_D, name='Web7', type=DEFORMABLE_BODY)
myBeam.parts['Web7'].BaseShellExtrude(depth=self.dP3, sketch=web7Sketch)
del myBeam.sketches['__profile__']

twosidedtransversestiffenerSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
twosidedtransversestiffenerSketch.rectangle(point1=(-self.wdts, -self.da/2), point2=(self.wdts, self.da/2))
myBeam.Part(dimensionality=THREE_D, name='TwoSidedTransverseStiffener', type=DEFORMABLE_BODY)
myBeam.parts['TwoSidedTransverseStiffener'].BaseShell(sketch=twosidedtransversestiffenerSketch)
del myBeam.sketches['__profile__']

onesidedtransversestiffenerSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
onesidedtransversestiffenerSketch.rectangle(point1=(0.0, -self.da/2), point2=(self.wots, self.da/2))
myBeam.Part(dimensionality=THREE_D, name='OneSidedTransverseStiffener', type=DEFORMABLE_BODY)
myBeam.parts['OneSidedTransverseStiffener'].BaseShell(sketch=onesidedtransversestiffenerSketch)
```

```
        del myBeam.sketches['__profile__']

        longitudinalopenstiffenerSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        longitudinalopenstiffenerSketch.Line(point1=(0.0, 0.0), point2=(self.wlos, 0.0))
        longitudinalopenstiffenerSketch.HorizontalConstraint(addUndoState=False, entity=longitudinalopenstiffenerSketch.geometry.findAt((0.0, 0.0), ))
        myBeam.Part(dimensionality=THREE_D, name='LongitudinalOpenStiffener', type=DEFORMABLE_BODY)
        myBeam.parts['LongitudinalOpenStiffener'].BaseShellExtrude(depth=self.los, sketch=longitudinalopenstiffenerSketch)
        del myBeam.sketches['__profile__']

        longitudinalclosedstiffenerSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        g, v, d, c = longitudinalclosedstiffenerSketch.geometry, longitudinalclosedstiffenerSketch.vertices, longitudinalclosedstiffenerSketch.dimensions,
longitudinalclosedstiffenerSketch.constraints
        longitudinalclosedstiffenerSketch.Line(point1=(0.0, -148.2), point2=(self.wlcs, -88.5))
        longitudinalclosedstiffenerSketch.Line(point1=(self.wlcs, -88.5), point2=(self.wlcs, 88.5))
        longitudinalclosedstiffenerSketch.Line(point1=(self.wlcs, 88.5), point2=(0.0, 148.2))
        longitudinalclosedstiffenerSketch.VerticalConstraint(addUndoState=False, entity=g[3])
        myBeam.Part(dimensionality=THREE_D, name='LongitudinalClosedStiffener', type=DEFORMABLE_BODY)
        myBeam.parts['LongitudinalClosedStiffener'].BaseShellExtrude(depth=self.lcs, sketch=longitudinalclosedstiffenerSketch)
        del myBeam.sketches['__profile__']

    def create_solid(self, myBeam):
        solidSketch = myBeam.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        solidSketch.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(100.0, 0.0))
        myBeam.Part(dimensionality=THREE_D, name='Solid', type=DEFORMABLE_BODY)
        myBeam.parts['Solid'].BaseSolidExtrude(depth=60.0, sketch=solidSketch)
```

```
    del myBeam.sketches['__profile__']

def create_material(self, myBeam):
    myBeam.Material(name='Steel-Flange')
    myBeam.materials['Steel-Flange'].Elastic(table=self.elasticF)
    if self.analysisType != 'Buckling':
        myBeam.materials['Steel-Flange'].Plastic(table=self.plasticF)

    myBeam.Material(name='Steel-Web1')
    myBeam.materials['Steel-Web1'].Elastic(table=self.elasticW7)
    if self.analysisType != 'Buckling':
        myBeam.materials['Steel-Web1'].Plastic(table=self.plasticW7)

    myBeam.Material(name='Steel-Web2')
    myBeam.materials['Steel-Web2'].Elastic(table=self.elasticW6)
    if self.analysisType != 'Buckling':
        myBeam.materials['Steel-Web2'].Plastic(table=self.plasticW6)

    myBeam.Material(name='Steel-Web3')
    myBeam.materials['Steel-Web3'].Elastic(table=self.elasticW15)
    if self.analysisType != 'Buckling':
        myBeam.materials['Steel-Web3'].Plastic(table=self.plasticW15)

    myBeam.Material(name='Steel-Web4')
    myBeam.materials['Steel-Web4'].Elastic(table=self.elasticW15)
```

```
if self.analysisType != 'Buckling':
    myBeam.materials['Steel-Web4'].Plastic(table=self.plasticW15)

myBeam.Material(name='Steel-Web5')
myBeam.materials['Steel-Web5'].Elastic(table=self.elasticW15)
if self.analysisType != 'Buckling':
    myBeam.materials['Steel-Web5'].Plastic(table=self.plasticW15)

myBeam.Material(name='Steel-Web6')
myBeam.materials['Steel-Web6'].Elastic(table=self.elasticW7)
if self.analysisType != 'Buckling':
    myBeam.materials['Steel-Web6'].Plastic(table=self.plasticW7)

myBeam.Material(name='Steel-Web7')
myBeam.materials['Steel-Web7'].Elastic(table=self.elasticW7)
if self.analysisType != 'Buckling':
    myBeam.materials['Steel-Web7'].Plastic(table=self.plasticW7)

myBeam.Material(name='Steel-LongitudinalOpenStiffener')
myBeam.materials['Steel-LongitudinalOpenStiffener'].Elastic(table=self.elasticLOS)
if self.analysisType != 'Buckling':
    myBeam.materials['Steel-LongitudinalOpenStiffener'].Plastic(table=self.plasticLOS)

myBeam.Material(name='Steel-LongitudinalClosedStiffener')
myBeam.materials['Steel-LongitudinalClosedStiffener'].Elastic(table=self.elasticLCS)
```

```
        if self.analysisType != 'Buckling':
            myBeam.materials['Steel-LongitudinalClosedStiffener'].Plastic(table=self.plasticLCS)

        myBeam.Material(name='Steel-TwoSidedTransverseStiffener')
        myBeam.materials['Steel-TwoSidedTransverseStiffener'].Elastic(table=self.elasticDTS)
        if self.analysisType != 'Buckling':
            myBeam.materials['Steel-TwoSidedTransverseStiffener'].Plastic(table=self.plasticDTS)

        myBeam.Material(name='Steel-OneSidedTransverseStiffener')
        myBeam.materials['Steel-OneSidedTransverseStiffener'].Elastic(table=self.elasticOTS)
        if self.analysisType != 'Buckling':
            myBeam.materials['Steel-OneSidedTransverseStiffener'].Plastic(table=self.plasticOTS)

        myBeam.Material(name='Steel-Solid')
        myBeam.materials['Steel-Solid'].Elastic(table=self.elasticS)

    def create_sections(self, myBeam):
        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Flange', name='Flange', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tf, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web1', name='Web1', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw7, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)
```

```
myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web2', name='Web2', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw6, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web3', name='Web3', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw15, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web4', name='Web4', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw15, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web5', name='Web5', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw15, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web6', name='Web6', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw7, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-Web7', name='Web7', numIntPts=5,
    poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tw7, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)
```

```
        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-LongitudinalOpenStiffener',
name='LongitudinalOpenStiffener', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tlos, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-LongitudinalClosedStiffener',
name='LongitudinalClosedStiffener', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tlcs, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-TwoSidedTransverseStiffener',
name='TwoSidedTransverseStiffener', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tdts, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousShellSection(idealization=NO_IDEALIZATION, integrationRule=SIMPSON, material='Steel-OneSidedTransverseStiffener',
name='OneSidedTransverseStiffener', numIntPts=5,
            poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT, thickness=self.tots, thicknessField='', thicknessModulus=None,
thicknessType=UNIFORM, useDensity=OFF)

        myBeam.HomogeneousSolidSection(name='Solid', material='Steel-Solid', thickness=None)

    def assign_sections(self, myBeam):
        myBeam.parts['Flange1'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
            faces=myBeam.parts['Flange1'].faces.findAt(((0.49*self.bft, 0.0, 0.0),(-0.49*self.bft, 0.0, self.L)), )), sectionName='Flange', thicknessAssignment=FROM_SECTION)
```

```
myBeam.parts['Flange2'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Flange2'].faces.findAt(((0.49*self.bfb, 0.0, 0.0),(-0.49*self.bfb, 0.0, self.L)), )), sectionName='Flange', thicknessAssignment=FROM_SECTION)

myBeam.parts['Web1'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web1'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, self.dP1)), )), sectionName='Web1', thicknessAssignment=FROM_SECTION)

myBeam.parts['Web2'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web2'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, self.lP5+120.0)), )), sectionName='Web2',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web3'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web3'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, self.loverhang)), )), sectionName='Web3',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web4'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web4'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, 2*self.loverhang)), )), sectionName='Web4',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web5'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    faces=myBeam.parts['Web5'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, self.loverhang)), )), sectionName='Web5',
thicknessAssignment=FROM_SECTION)

myBeam.parts['Web6'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
```

```
        faces=myBeam.parts['Web6'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, self.lP6+120.0)), )), sectionName='Web6',
thicknessAssignment=FROM_SECTION)

    myBeam.parts['Web7'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
        faces=myBeam.parts['Web7'].faces.findAt(((0.0, -0.49*self.da, 0.0), (0.0, 0.49*self.da, self.dP3)), )), sectionName='Web7', thicknessAssignment=FROM_SECTION)

    myBeam.parts['LongitudinalOpenStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
        faces=myBeam.parts['LongitudinalOpenStiffener'].faces.findAt(((0.0, 0.0, 0.0), (self.wlos, 0.0, self.los)), )), sectionName='LongitudinalOpenStiffener',
thicknessAssignment=FROM_SECTION)

    myBeam.parts['LongitudinalClosedStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
        faces=myBeam.parts['LongitudinalClosedStiffener'].faces.getByBoundingBox(0.0, -150.0, 0.0, self.wlcs, 150.0, self.lcs)), sectionName='LongitudinalClosedStiffener',
thicknessAssignment=FROM_SECTION)

    myBeam.parts['TwoSidedTransverseStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
        faces=myBeam.parts['TwoSidedTransverseStiffener'].faces.findAt(((0.0, -0.48*self.da, 0.0), (0.0, 0.48*self.da, 0.0)), )), sectionName='TwoSidedTransverseStiffener',
thicknessAssignment=FROM_SECTION)

    myBeam.parts['OneSidedTransverseStiffener'].SectionAssignment(offset=0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
        faces=myBeam.parts['OneSidedTransverseStiffener'].faces.findAt(((0.0, -0.48*self.da, 0.0), (0.0, 0.48*self.da, 0.0)), )), sectionName='OneSidedTransverseStiffener',
thicknessAssignment=FROM_SECTION)

    delta = 1
    myBeam.parts['Solid'].SectionAssignment(region=regionToolset.Region(cells=myBeam.parts['Solid'].cells.getByBoundingBox(-100.0-delta, -100.0-delta, 0.0-delta,
100.0+delta, 100.0+delta, 60.0+delta)),
```

```
        sectionName='Solid', offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTION)

def create_beam_instance(self, myBeam):
    myBeam.rootAssembly.DatumCsysByDefault(CARTESIAN)
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web1', part=myBeam.parts['Web1'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web2', part=myBeam.parts['Web2'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web3', part=myBeam.parts['Web3'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web4', part=myBeam.parts['Web4'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web5', part=myBeam.parts['Web5'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web6', part=myBeam.parts['Web6'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Web7', part=myBeam.parts['Web7'])
    myBeam.rootAssembly.translate(instanceList=('Web2', ), vector=(0.0, 0.0, self.dP1))
    myBeam.rootAssembly.translate(instanceList=('Web3', ), vector=(0.0, 0.0, self.dP1+self.lP5+120.0))
    myBeam.rootAssembly.translate(instanceList=('Web4', ), vector=(0.0, 0.0, self.dP1+self.lP5+120.0+self.loverhang))
    myBeam.rootAssembly.translate(instanceList=('Web5', ), vector=(0.0, 0.0, self.dP1+self.lP5+120.0+3*self.loverhang))
    myBeam.rootAssembly.translate(instanceList=('Web6', ), vector=(0.0, 0.0, self.dP1+self.lP5+120.0+4*self.loverhang))
    myBeam.rootAssembly.translate(instanceList=('Web7', ), vector=(0.0, 0.0, self.L-self.dP3))
    myBeam.rootAssembly.Instance(dependent=OFF, name='Flange1', part=myBeam.parts['Flange1'])
    myBeam.rootAssembly.Instance(dependent=OFF, name='Flange2', part=myBeam.parts['Flange2'])
    myBeam.rootAssembly.translate(instanceList=('Flange1', ), vector=(0.0, self.da/2, 0.0))
    myBeam.rootAssembly.translate(instanceList=('Flange2', ), vector=(0.0, -self.da/2, 0.0))
    myBeam.rootAssembly.Instance(dependent=OFF, name='LongitudinalOpenStiffener1', part=myBeam.parts['LongitudinalOpenStiffener'])
    myBeam.rootAssembly.translate(instanceList=('LongitudinalOpenStiffener1', ), vector=(0.0, self.da/2-self.tf/2-self.b1, self.loverhang))
    myBeam.rootAssembly.Instance(dependent=OFF, name='LongitudinalOpenStiffener2', part=myBeam.parts['LongitudinalOpenStiffener'])
    myBeam.rootAssembly.translate(instanceList=('LongitudinalOpenStiffener2', ), vector=(0.0, self.da/2-self.tf/2-2*self.b1, self.loverhang))
```

```
        myBeam.rootAssembly.Instance(dependent=OFF, name='LongitudinalClosedStiffener', part=myBeam.parts['LongitudinalClosedStiffener'])
        myBeam.rootAssembly.translate(instanceList=('LongitudinalClosedStiffener', ), vector=(0.0, self.da/2-self.tf/2-self.b,
self.loverhang+self.lP1+self.lP2+self.lP5+self.loverhang))
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener1', part=myBeam.parts['TwoSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener2', part=myBeam.parts['TwoSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener3', part=myBeam.parts['TwoSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener4', part=myBeam.parts['TwoSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener5', part=myBeam.parts['TwoSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='TwoSidedTransverseStiffener6', part=myBeam.parts['TwoSidedTransverseStiffener'])
        myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener2', ), vector=(0.0, 0.0, self.loverhang+self.lP1+self.lP2+self.lP5))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener3', ), vector=(0.0, 0.0, self.loverhang+self.lP1+self.lP2+self.lP5+self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener4', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lcs+2*self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener5', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lcs+3*self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('TwoSidedTransverseStiffener6', ), vector=(0.0, 0.0, self.L))
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener1', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener2', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener3', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener4', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener5', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.Instance(dependent=OFF, name='OneSidedTransverseStiffener6', part=myBeam.parts['OneSidedTransverseStiffener'])
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener1', ), vector=(0.0, 0.0, self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener2', ), vector=(0.0, 0.0, self.loverhang+self.lP1))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener3', ), vector=(0.0, 0.0, self.loverhang+self.lP1+self.lP2))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener4', ), vector=(0.0, 0.0, self.L-self.loverhang))
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener5', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lP3))
```

```
        myBeam.rootAssembly.translate(instanceList=('OneSidedTransverseStiffener6', ), vector=(0.0, 0.0, self.L-self.loverhang-self.lP3-self.lP4))
        myBeam.rootAssembly.Instance(dependent=OFF, name='Solid', part=myBeam.parts['Solid'])
        myBeam.rootAssembly.rotate(instanceList=('Solid', ), axisPoint=(-100.0, 0.0, 60.0), axisDirection=(100.0, 0.0, 0.0), angle=270.0)
        myBeam.rootAssembly.translate(instanceList=('Solid', ), vector=(0.0, 60.0+self.da/2, self.loverhang+self.los-self.loverhang+40.0))

    def partition_face(self, myBeam):

        xyPlane1= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane1.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane2= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane2.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane3= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane3.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane4= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5, principalPlane=XYPLANE)
        myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane4.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

        xyPlane5= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+self.loverhang/2, principalPlane=XYPLANE)
```

```
myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane5.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

xyPlane6= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+self.loverhang, principalPlane=XYPLANE)
myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane6.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

xyPlane7= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+3*self.loverhang, principalPlane=XYPLANE)
myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane7.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

xyPlane8= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+4*self.loverhang, principalPlane=XYPLANE)
myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane8.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

xyPlane9= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+4*self.loverhang+self.lP6,
principalPlane=XYPLANE)
myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane9.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

xyPlane10= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+4*self.loverhang+self.lP6+self.lP4,
principalPlane=XYPLANE)
myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane10.id],
faces=myBeam.rootAssembly.instances['Flange1'].faces + myBeam.rootAssembly.instances['Flange2'].faces)
```

```
    xyPlane11= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.loverhang+self.lP1+self.lP2+self.lP5+4*self.loverhang+self.lP6+self.lP4+self.lP3,
principalPlane=XYPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xyPlane11.id],
faces=myBeam.rootAssembly.instances['Flange1'].faces + myBeam.rootAssembly.instances['Flange2'].faces)

    yzPlane1= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=-self.wdts, principalPlane=YZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[yzPlane1.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

    yzPlane2= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.wdts, principalPlane=YZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[yzPlane2.id], faces=myBeam.rootAssembly.instances['Flange1'].faces
+ myBeam.rootAssembly.instances['Flange2'].faces)

    xzPlane1= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.da/2-self.tf/2-self.b1, principalPlane=XZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane1.id],
faces=myBeam.rootAssembly.instances['TwoSidedTransverseStiffener1'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'].faces +
myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'].faces)

    xzPlane2= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.da/2-self.tf/2-2*self.b1, principalPlane=XZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane2.id],
faces=myBeam.rootAssembly.instances['TwoSidedTransverseStiffener1'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'].faces +
myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'].faces +\
```

```
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'].faces)

    xzPlane3= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=self.da/2-self.tf/2-self.b, principalPlane=XZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane3.id],
faces=myBeam.rootAssembly.instances['TwoSidedTransverseStiffener4'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener5'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener6'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener4'].faces +
myBeam.rootAssembly.instances['OneSidedTransverseStiffener5'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener6'].faces)

    xzPlane4= myBeam.rootAssembly.DatumPlaneByPrincipalPlane(offset=0.0, principalPlane=XZPLANE)
    myBeam.rootAssembly.PartitionFaceByDatumPlane(datumPlane=myBeam.rootAssembly.datums[xzPlane4.id],
faces=myBeam.rootAssembly.instances['TwoSidedTransverseStiffener1'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'].faces + myBeam.rootAssembly.instances['TwoSidedTransverseStiffener4'].faces +
myBeam.rootAssembly.instances['TwoSidedTransverseStiffener5'].faces +\
        myBeam.rootAssembly.instances['TwoSidedTransverseStiffener6'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'].faces +
myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'].faces + myBeam.rootAssembly.instances['OneSidedTransverseStiffener4'].faces +
myBeam.rootAssembly.instances['OneSidedTransverseStiffener5'].faces +\
        myBeam.rootAssembly.instances['OneSidedTransverseStiffener6'].faces + myBeam.rootAssembly.instances['Web1'].faces +
myBeam.rootAssembly.instances['Web2'].faces + myBeam.rootAssembly.instances['Web3'].faces +\
        myBeam.rootAssembly.instances['Web4'].faces + myBeam.rootAssembly.instances['Web5'].faces + myBeam.rootAssembly.instances['Web6'].faces +
myBeam.rootAssembly.instances['Web7'].faces)

    delta=3
```

```
        myBeam.rootAssembly.PartitionFaceByIntersectFace(faces=myBeam.rootAssembly.instances['Flange1'].faces.getByBoundingBox(-self.bft/2-delta, self.da/2-delta,
self.loverhang+self.lP1+self.lP2+self.lP5-delta,
            self.bft/2+delta, self.da/2+delta, 2*self.loverhang+self.lP1+self.lP2+self.lP5+delta), cuttingFaces=myBeam.rootAssembly.instances['Solid'].faces.getByBoundingBox(
            -100.0-delta, self.da/2-delta, self.loverhang+self.lP1+self.lP2+self.lP5-delta, 100.0+delta, self.da/2+delta, 2*self.loverhang+self.lP1+self.lP2+self.lP5+delta))

        myBeam.rootAssembly.InstanceFromBooleanMerge(domain=GEOMETRY, instances=(myBeam.rootAssembly.instances['Web1'],
myBeam.rootAssembly.instances['Web2'], myBeam.rootAssembly.instances['Web3'],
            myBeam.rootAssembly.instances['Web4'], myBeam.rootAssembly.instances['Web5'], myBeam.rootAssembly.instances['Web6'],
myBeam.rootAssembly.instances['Web7'],
            myBeam.rootAssembly.instances['Flange1'], myBeam.rootAssembly.instances['Flange2'], myBeam.rootAssembly.instances['LongitudinalOpenStiffener1'],
myBeam.rootAssembly.instances['LongitudinalOpenStiffener2'],
            myBeam.rootAssembly.instances['LongitudinalClosedStiffener'],myBeam.rootAssembly.instances['TwoSidedTransverseStiffener1'],
myBeam.rootAssembly.instances['TwoSidedTransverseStiffener2'],
            myBeam.rootAssembly.instances['TwoSidedTransverseStiffener3'],myBeam.rootAssembly.instances['TwoSidedTransverseStiffener4'],
myBeam.rootAssembly.instances['TwoSidedTransverseStiffener5'],
            myBeam.rootAssembly.instances['TwoSidedTransverseStiffener6'], myBeam.rootAssembly.instances['OneSidedTransverseStiffener1'],
myBeam.rootAssembly.instances['OneSidedTransverseStiffener2'],
            myBeam.rootAssembly.instances['OneSidedTransverseStiffener3'], myBeam.rootAssembly.instances['OneSidedTransverseStiffener4'],
myBeam.rootAssembly.instances['OneSidedTransverseStiffener5'],
            myBeam.rootAssembly.instances['OneSidedTransverseStiffener6']), keepIntersections=ON, name='Beam', originalInstances=DELETE)

        myInstance1=myBeam.rootAssembly.instances['Beam-1']
        myBeam.rootAssembly.makeIndependent(instances=(myInstance1, ))

    def surfaces(self, myBeam):
```

```
        delta=1
        myBeam.rootAssembly.Surface(side2Faces=myBeam.rootAssembly.instances['Beam-1'].faces.getByBoundingBox(-100.0-delta, self.da/2-delta,
self.loverhang+self.lP1+self.lP2+self.lP5-delta,
            100.0+delta, self.da/2+delta, 2*self.loverhang+self.lP1+self.lP2+self.lP5+delta), name='SurfaceBeam-Tie')
        myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['Solid'].faces.getByBoundingBox(-100.0-delta, self.da/2-delta,
self.loverhang+self.lP1+self.lP2+self.lP5-delta,
            100.0+delta, self.da/2+delta, 2*self.loverhang+self.lP1+self.lP2+self.lP5+delta), name='SurfaceSolid-Tie')
        myBeam.rootAssembly.Surface(side1Faces=myBeam.rootAssembly.instances['Solid'].faces.getByBoundingBox(-100.0-delta, self.da/2+60.0-delta,
self.loverhang+self.lP1+self.lP2+self.lP5-delta,
            100.0+delta, self.da/2+60.0+delta, 2*self.loverhang+self.lP1+self.lP2+self.lP5+delta), name='SurfaceSolid-Load')
    def mesh_instance(self, myBeam):
        myInstance1=myBeam.rootAssembly.instances['Beam-1']
        myBeam.rootAssembly.setMeshControls(regions=myInstance1.faces, technique=STRUCTURED)
        myBeam.rootAssembly.setElementType(elemTypes=(ElemType(elemCode=S4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF), ElemType(elemCode=S3,
elemLibrary=STANDARD)), regions=Region(myInstance1.faces))
        myBeam.rootAssembly.seedPartInstance(deviationFactor=0.1, minSizeFactor=0.1, regions=(myInstance1, ), size=self.sizeOfElements)
        myBeam.rootAssembly.generateMesh(regions=(myInstance1, ))

        elemType1 = mesh.ElemType(elemCode=C3D8R, elemLibrary=STANDARD, kinematicSplit=AVERAGE_STRAIN, secondOrderAccuracy=OFF,
hourglassControl=DEFAULT, distortionControl=DEFAULT)
        elemType2 = mesh.ElemType(elemCode=C3D6, elemLibrary=STANDARD)
        elemType3 = mesh.ElemType(elemCode=C3D4, elemLibrary=STANDARD)
        delta=1
        myBeam.rootAssembly.setElementType(regions=(myBeam.rootAssembly.instances['Solid'].cells.getByBoundingBox(-100.0-delta, self.da/2-delta,
self.loverhang+self.lP1+self.lP2+self.lP5-delta,
```

```
        100.0+delta, self.da/2+60.0+delta, 2*self.loverhang+self.lP1+self.lP2+self.lP5+delta), ), elemTypes=(elemType1, elemType2, elemType3))
    myBeam.rootAssembly.seedPartInstance(regions=(myBeam.rootAssembly.instances['Solid'], ), size=5.0, deviationFactor=0.1, minSizeFactor=0.1)
    myBeam.rootAssembly.generateMesh(regions=(myBeam.rootAssembly.instances['Solid'], ))

  def create_sets(self, myBeam):
    myInstance1 = myBeam.rootAssembly.instances['Beam-1']

    myBeam.rootAssembly.Set(edges=
      myInstance1.edges.findAt(((0.1, -self.da/2, self.loverhang), ), ((1.01*self.wdts, -self.da/2, self.loverhang), ), ((-0.1, -self.da/2, self.loverhang), ), ((
        -1.01*self.wdts, -self.da/2, self.loverhang), ), ((0.1, self.da/2, self.loverhang), ), ((1.01*self.wdts, self.da/2, self.loverhang), ), ((-0.1, self.da/2, self.loverhang), ), ((
        -1.01*self.wdts, self.da/2, self.loverhang), ), ((0.0, -0.1, self.loverhang), ), ((0.0, 0.1, self.loverhang), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b1), self.loverhang), ), ((
        0.0, 1.01*(self.da/2-self.tf/2-self.b1), self.loverhang), ), ((0.1, -self.da/2, self.loverhang+self.los+self.lcs), ), ((1.01*self.wdts, -self.da/2, self.loverhang+self.los+self.lcs), ), ((
        -0.1, -self.da/2, self.loverhang+self.los+self.lcs), ), ((-1.01*self.wdts, -self.da/2, self.loverhang+self.los+self.lcs), ), ((0.1, self.da/2, self.loverhang+self.los+self.lcs), ), ((
        1.01*self.wdts, self.da/2, self.loverhang+self.los+self.lcs), ), ((-0.1, self.da/2, self.loverhang+self.los+self.lcs), ), ((-1.01*self.wdts, self.da/2, self.loverhang+self.los+self.lcs), ), ((
        0.0, -0.1, self.loverhang+self.los+self.lcs), ), ((0.0, 0.1, self.loverhang+self.los+self.lcs), ), ((0.0, 0.999*self.da/2, self.loverhang+self.los+self.lcs), ), ((
        0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang+self.los+self.lcs), ), ((0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang+self.los+self.lcs), ), ), name='Set-BC1')

    myBeam.rootAssembly.Set(edges=
      myInstance1.edges.findAt(((0.0, -0.1, self.loverhang), ), ((0.0, 0.1, self.loverhang), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b1), self.loverhang), ), ((
        0.0, 1.01*(self.da/2-self.tf/2-self.b1), self.loverhang), ), ((0.0, -0.1, self.loverhang+self.los+self.lcs), ), ((0.0, 0.1, self.loverhang+self.los+self.lcs), ), ((
        0.0, 0.999*self.da/2, self.loverhang+self.los+self.lcs), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang+self.los+self.lcs), ), ((
        0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang+self.los+self.lcs), ), ), name='Set-BC2')
```

```
myBeam.rootAssembly.Set(name='Set-BC3', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.loverhang+self.los+self.lcs), )))

myBeam.rootAssembly.Set(name='Set-BC4', vertices=myInstance1.vertices.findAt(((0.0, 0.0, self.loverhang), )))

myBeam.rootAssembly.Set(name='Set-LR1', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.loverhang), )))

myBeam.rootAssembly.Set(name='Set-LR2', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, self.loverhang+self.lP1), )))

myBeam.rootAssembly.Set(edges=
    myInstance1.edges.findAt(((0.1, -self.da/2, self.loverhang+self.lP1+self.lP2), ), ((1.01*self.wdts, -self.da/2, self.loverhang+self.lP1+self.lP2), ), ((-0.1, -self.da/2, self.loverhang+self.lP1+self.lP2), ), ((
        -1.01*self.wdts, -self.da/2, self.loverhang+self.lP1+self.lP2), ), ((0.1, self.da/2, self.loverhang+self.lP1+self.lP2), ), ((1.01*self.wdts, self.da/2, self.loverhang+self.lP1+self.lP2), ), ((-0.1, self.da/2, self.loverhang+self.lP1+self.lP2), ), ((
        -1.01*self.wdts, self.da/2, self.loverhang+self.lP1+self.lP2), ), ((0.0, -0.1, self.loverhang+self.lP1+self.lP2), ), ((0.0, 0.1, self.loverhang+self.lP1+self.lP2), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b1), self.loverhang+self.lP1+self.lP2), ), ((
        0.0, 1.01*(self.da/2-self.tf/2-self.b1), self.loverhang+self.lP1+self.lP2), ), ), name='Set-LR3')

myBeam.rootAssembly.Set(edges=
    myInstance1.edges.findAt(((0.1, -self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((1.01*self.wdts, -self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((-0.1, -self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((
        -1.01*self.wdts, -self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((0.1, self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((1.01*self.wdts, self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((-0.1, self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((
        -1.01*self.wdts, self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((0.0, -0.1, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((0.0, 0.1, self.loverhang+self.lP1+self.lP2+self.lP5), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b1), self.loverhang+self.lP1+self.lP2+self.lP5), ), ((
```

```
        0.0, 1.01*(self.da/2-self.tf/2-self.b1), self.loverhang+self.lP1+self.lP2+self.lP5), ), ), name='Set-LR4')

    myBeam.rootAssembly.Set(name='Set-LR5', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 2*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-LR6', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 4*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-LR7', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 5*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-LR8', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 5*self.loverhang+self.lP1+self.lP2+self.lP5+self.lP6), )))

    myBeam.rootAssembly.Set(name='Set-LR9', vertices=myInstance1.vertices.findAt(((0.0, self.da/2, 5*self.loverhang+self.lP1+self.lP2+self.lP5+self.lP6+self.lP4), )))

    myBeam.rootAssembly.Set(name='Set-LR10', vertices=myInstance1.vertices.findAt(((0.0, self.da/2,
5*self.loverhang+self.lP1+self.lP2+self.lP5+self.lP6+self.lP4+self.lP3), )))

    myBeam.rootAssembly.Set(name='Set-LR11', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-LR12', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 2*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-LR13', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 4*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-LR14', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 5*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-V1', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.loverhang+self.lP1+self.lP2), )))
```

```
    myBeam.rootAssembly.Set(name='Set-V2', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-V3', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 2*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-V4', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 4*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-V5', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 5*self.loverhang+self.lP1+self.lP2+self.lP5), )))

    myBeam.rootAssembly.Set(name='Set-V6', vertices=myInstance1.vertices.findAt(((0.0, -self.da/2, 5*self.loverhang+self.lP1+self.lP2+self.lP5+self.lP6), )))

def set_boundary_conditions(self, myBeam):
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BC1',
        region=myBeam.rootAssembly.sets['Set-BC1'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=SET)
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BC2',
        region=myBeam.rootAssembly.sets['Set-BC2'], u1=UNSET, u2=SET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='BC3',
        region=myBeam.rootAssembly.sets['Set-BC3'], u1=UNSET, u2=UNSET, u3=SET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='LR3',
        region=myBeam.rootAssembly.sets['Set-LR3'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
    myBeam.DisplacementBC(amplitude=UNSET, createStepName='Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name='LR4',
        region=myBeam.rootAssembly.sets['Set-LR4'], u1=SET, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

def create_constraint(self, myBeam):
    myBeam.Tie(name='Tie', master=myBeam.rootAssembly.surfaces['SurfaceSolid-Tie'], slave=myBeam.rootAssembly.surfaces['SurfaceBeam-Tie'],
        positionToleranceMethod=COMPUTED, adjust=ON, tieRotations=ON, thickness=ON)
```

```
    def create_step(self, myBeam):
        if self.analysisType == 'Buckling':
            myBeam.BuckleStep(eigensolver=SUBSPACE, name='Step-1', numEigen=self.numberOfEigen, maxIterations=1000, previous='Initial')
        else:
            myBeam.StaticRiksStep(initialArcInc=self.arco0, minArcInc=self.arcMin, maxArcInc=self.arcMax, name='Step-1', nlgeom=ON, previous='Initial', maxNumInc =
self.numberOfInc)

    def createResidualStress(self, myBeam):
        myInstance1 = myBeam.rootAssembly.instances['Beam-1']
        delta = 12.0#mm

        # Mesa Superior
        delta = 10.0
        myBeam.rootAssembly.Set(edges=
        myInstance1.edges.findAt(((0.1, self.da/2, self.loverhang), ), ((0.9*self.wdts, self.da/2, self.loverhang), ), ((1.01*self.wdts, self.da/2, self.loverhang), ), ((
            -0.1, self.da/2, self.loverhang), ), ((-0.9*self.wdts, self.da/2, self.loverhang), ), ((-1.01*self.wdts, self.da/2, self.loverhang), ), ), name='Set-FlangeUNodes')
        numberFlangeElements = int(len(myBeam.rootAssembly.sets['Set-FlangeUNodes'].nodes)-3)
        sizeOfFlangeElements = (self.bft)/(numberFlangeElements)

        # Definir tensao residual para as mesas
        for x in range(int(numberFlangeElements/2)):
            setName = 'TRFU' + str(x+1)
            x1 = x*sizeOfFlangeElements
            x2 = (x+1)*sizeOfFlangeElements
```

```
    y = self.da/2
    f1 = ((3.397*self.fyf)/((self.bft/2.0)*(self.bft/2.0)))*(x1*x1)-((3.497*self.fyf)/(self.bft/2.0))*x1+(0.6*self.fyf)
    f2 = ((3.397*self.fyf)/((self.bft/2.0)*(self.bft/2.0)))*(x2*x2)-((3.497*self.fyf)/(self.bft/2.0))*x2+(0.6*self.fyf)
    elementS = (f2+f1)/2.0

    e1 = myInstance1.elements.getByBoundingBox(x1-delta, y-delta, 3585.0-delta, x2+delta, y+delta, 5085.0+delta)
    e2 = myInstance1.elements.getByBoundingBox(-x2-delta, y-delta, 3585.0-delta, -x1+delta, y+delta, 5085.0+delta)

    elements1 = e1+ e2
    myBeam.rootAssembly.Set(elements = elements1, name=setName)

    myBeam.Stress(name='TRFU'+str(x+1), region=myBeam.rootAssembly.sets['TRFU'+str(x+1)], distributionType=UNIFORM, sigma11=0.0,
        sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

# Mesa Inferior
delta = 10.0
myBeam.rootAssembly.Set(edges=
myInstance1.edges.findAt(((0.1, -self.da/2, self.loverhang), ), ((0.9*self.wdts, -self.da/2, self.loverhang), ), ((1.01*self.wdts, -self.da/2, self.loverhang), ), ((
        -0.1, -self.da/2, self.loverhang), ), ((-0.9*self.wdts, -self.da/2, self.loverhang), ), ((-1.01*self.wdts, -self.da/2, self.loverhang), ), ), name='Set-FlangeBNodes')
numberFlangeElements = int(len(myBeam.rootAssembly.sets['Set-FlangeBNodes'].nodes)-3)
sizeOfFlangeElements = (self.bfb)/(numberFlangeElements)

# Definir tensao residual para as mesas
for x in range(int(numberFlangeElements/2)):
    setName = 'TRFB' + str(x+1)
```

```
    x1 = x*sizeOfFlangeElements
    x2 = (x+1)*sizeOfFlangeElements
    y = self.da/2
    f1 = ((3.397*self.fyf)/((self.bfb/2.0)*(self.bfb/2.0)))*(x1*x1)-((3.497*self.fyf)/(self.bfb/2.0))*x1+(0.6*self.fyf)
    f2 = ((3.397*self.fyf)/((self.bfb/2.0)*(self.bfb/2.0)))*(x2*x2)-((3.497*self.fyf)/(self.bfb/2.0))*x2+(0.6*self.fyf)
    elementS = (f2+f1)/2.0

    if abs(elementS) < (5e-7):
        elementS = 0

    e1 = myInstance1.elements.getByBoundingBox(x1-delta, -y-delta, 3585.0-delta, x2+delta, -y+delta, 5085.0+delta)
    e2 = myInstance1.elements.getByBoundingBox(-x2-delta, -y-delta, 3585.0-delta, -x1+delta, -y+delta, 5085.0+delta)

    elements1 = e1+e2
    myBeam.rootAssembly.Set(elements = elements1, name=setName)

    myBeam.Stress(name='TRFB'+str(x+1), region=myBeam.rootAssembly.sets['TRFB'+str(x+1)], distributionType=UNIFORM, sigma11=0.0,
        sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

# Alma
delta = 12.0 #mm
myBeam.rootAssembly.Set(edges=
myInstance1.edges.findAt(((0.0, -0.1, self.loverhang), ), ((0.0, 0.1, self.loverhang), ), ((0.0, 0.999*(self.da/2-self.tf/2-self.b), self.loverhang), ), ((
    0.0, 1.01*(self.da/2-self.tf/2-self.b), self.loverhang), ), ), name='Set-WebNodes')
numberWebElements = int(len(myBeam.rootAssembly.sets['Set-WebNodes'].nodes)-1)
```

```
sizeOfWebElements = self.da/(numberWebElements)

# Definir tensao residual para a alma
for y in range(int(numberWebElements/2)):
    setName = 'TRW' + str(y+1)
    y1 = y*sizeOfWebElements
    y2 = (y+1)*sizeOfWebElements
    x = 0.0
    g = ((-0.1*self.fyw)*(0.1225*self.da+0.0393*self.bft)-(0.6*self.fyw*0.1225*self.da)-(0.6*self.fyw*0.0393*self.bft))/(-0.6*self.fyw)

    if (y1 > self.da/2.0-g) & (y2 > self.da/2.0-g):
        f1 = ((-0.6*self.fyw*(self.da/2.0-y1)+0.0735*self.fyw*self.da+0.02362*self.fyw*self.bft))/(0.1225*self.da+0.0393*self.bft)
        f2 = ((-0.6*self.fyw*(self.da/2.0-y2)+0.0735*self.fyw*self.da+0.02362*self.fyw*self.bft))/(0.1225*self.da+0.0393*self.bft)
        elementS = (f2+f1)/2

    if (y1 < self.da/2.0-g) & (y2 > self.da/2.0-g):
        f1 = -0.1*self.fyw
        f2 = ((-0.6*self.fyw*(self.da/2.0-y2)+0.0735*self.fyw*self.da+0.02362*self.fyw*self.bft))/(0.1225*self.da+0.0393*self.bft)
        elementS = (f2+f1)/2

    if (y1 < self.da/2.0-g) & (y2 < self.da/2.0-g):
        f1 = -0.1*self.fyw
        f2 = -0.1*self.fyw
        elementS = (f2+f1)/2
```

```
        if abs(elementS) < (5e-7):
            elementS = 0

        e1 = myInstance1.elements.getByBoundingBox(x-delta, y1-delta, 3585.0-delta, x+delta, y2+delta, 5085.0+delta)
        e2 = myInstance1.elements.getByBoundingBox(x-delta, -y2-delta, 3585.0-delta, x+delta, -y1+delta, 5085.0+delta)

        elements1 = e1+e2
        myBeam.rootAssembly.Set(elements = elements1, name=setName)

        myBeam.Stress(name='TRW'+str(y+1), region=myBeam.rootAssembly.sets['TRW'+str(y+1)], distributionType=UNIFORM, sigma11=0.0,
            sigma22=elementS, sigma12=0.0, sigma33=None, sigma13=None, sigma23=None)

def create_load(self, myBeam):
    import math
    area=math.pi*100*100
    loadapplied=self.loadMagnitude/area

    myBeam.Pressure(name='Load-1',
        createStepName='Step-1', region=myBeam.rootAssembly.surfaces['SurfaceSolid-Load'], distributionType=UNIFORM, field='', magnitude=-loadapplied)

def edit_text(self, myBeam, textStop, texttoReplace):
    myBeam.keywordBlock.synchVersions(storeNodesAndElements=False)
    linePosition = 0
    for keyWords in myBeam.keywordBlock.sieBlocks:
        if keyWords.startswith(textStop):
```

```
            myBeam.keywordBlock.replace(linePosition, texttoReplace)
            break
        linePosition = linePosition + 1

def create_jobs(self, jobName):
    mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF, explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
        memory=90, memoryUnits=PERCENTAGE, model=self.modelName, modelPrint=OFF, multiprocessingMode=DEFAULT, name=jobName,
        nodalOutputPrecision=SINGLE, numCpus=1, numGPUs=0, queue=None, scratch='', type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)

def create_beam(self, nameofModel, jobName):
    self.create_flange_web_and_stiffeners(nameofModel)
    self.create_solid(nameofModel)
    self.create_material(nameofModel)
    self.create_sections(nameofModel)
    self.assign_sections(nameofModel)
    self.create_beam_instance(nameofModel)
    self.partition_face(nameofModel)
    self.surfaces(nameofModel)
    self.mesh_instance(nameofModel)
    self.create_sets(nameofModel)
    self.set_boundary_conditions(nameofModel)
    self.create_constraint(nameofModel)
    self.create_step(nameofModel)
    self.create_load(nameofModel)
```

```
        if self.analysisType != 'Buckling':
            self.createResidualStress(nameofModel)

        if self.analysisType == 'Buckling':
            textStop = '*Output'
            texttoReplace = '*Output, field, variable=PRESELECT\n*NODE FILE\nU,'
        else:
            textStop = '*End Assembly'
            texttoReplace = '*End Assembly\n*IMPERFECTION, FILE=' + self.imperfectionFilename + ', STEP=1\n11, ' + str(1.20) + '\n'

        self.edit_text(nameofModel, textStop, texttoReplace)
        self.create_jobs(jobName)

baseNameBuckling = 'PanelUO-20mm-Solidwith5mm'
baseNameNL = 'PanelUO-20mm-Solidwith5mm-235-355'
fyf = 355.00 #tensão de escoamento da mesa
fyw = 235.00 #tensão de escoamento da alma
#Medidas em milimetros e Newton
da = 1820.0 #altura da alma (linha do esqueleto)
bft = 250.0 #largura da mesa superior
bfb = 450.0 #largura da mesa inferior
tf = 20.0 #espessura da mesa
tw6 = 6.0 #espessura da alma (parte em analise)
tw7 = 7.0 #espessura da alma (depois do enrijecedor)
```

```
tw15 = 15.0 #espessura da alma entre as cargas
tots = 20.0 #espessura do enrijecedor transversal de um lado da alma
wots = 122.0 #largura do enrijecedor transversal de um lado da alma
tdts = 20.0 #espessura do enrijecedor transversal dos dois lados da alma
wdts = 122.0 #largura do enrijecedor transversal dos dois lados da alma
tlos = 10.0 #espessura do enrijecedor longitudinal retangular
wlos = 100.0 #largura do enrijecedor longitudinal retangular
tlcs = 5.0 #espessura do enrijecedor longitudinal trapeizodal
wlcs = 80.0 #largura do enrijecedor longitudinal trapeizodal
b = 500.0 #distancia da parte debaixo da mesa até o centro de gravidade do enrijecedor longitudinal trapeizodal
b1 = 350.0 #distancia da parte debaixo da mesa até o centro de gravidade do enrijecedor longitudinal retangular
los = 5110.0
lcs = 5815.0
loverhang = 200.0

dP1 = 3190.0
dP2 = 5540.0
dP3 = 2595.0
L = 11325.0 #comprimento total
lP1 = 1510.0
lP2 = 1600.0
lP3 = 1215.0
lP4 = 1300.0
lP5 = 1800.0
lP6 = 2700.0
```

```
analysisType = 'NL'
Lcr = 1330000.0

beamExample = FLABeam(baseNameBuckling, baseNameNL, fyf, fyw, da, bft, bfb, tf, tw6, tw7, tw15, tots, wots, tdts, wdts, tlos, wlos, tlcs, wlcs, b, b1, los, lcs, loverhang,
dP1, dP2, dP3, L, lP1, lP2, lP3, lP4, lP5, lP6, analysisType, Lcr)
beamExampleModel=beamExample.create_new_model()
if analysisType == 'Buckling':
    beamExample.create_beam(beamExampleModel, baseNameBuckling + '-B')
else:
    beamExample.create_beam(beamExampleModel, baseNameNL + '-NL')
```

ANEXO A – DESLOCAMENTO VERTICAL DOS PAINÉIS ESBELTOS

A Figura An.A.1 e Figura An.A.2 mostram os pontos nos quais foi realizada a média aritmética para obtenção do deslocamento vertical dos painéis SO e UO, respectivamente.

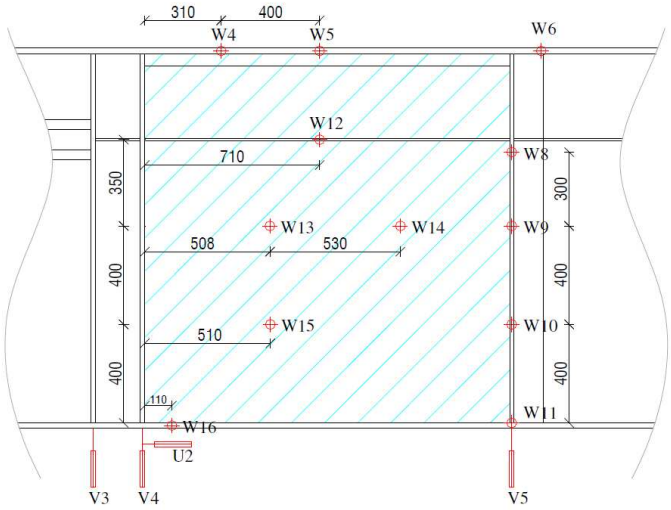


Figura An.A.3 – Pontos V3 e V4 no painel SO (Sinur,2011).

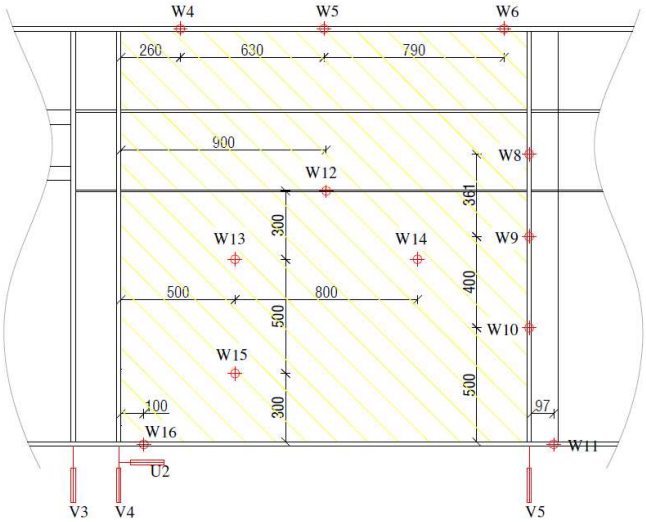


Figura An.A.4 - Pontos V3 e V4 no painel UO (Sinur,2011).

ANEXO B – DISTRIBUIÇÃO DAS TENSÕES RESIDUAIS EM PERFIS I SOLDADOS

A Figura An.B.1 apresenta a distribuição de tensões residuais adotada na análise de sensibilidade dos parâmetros.

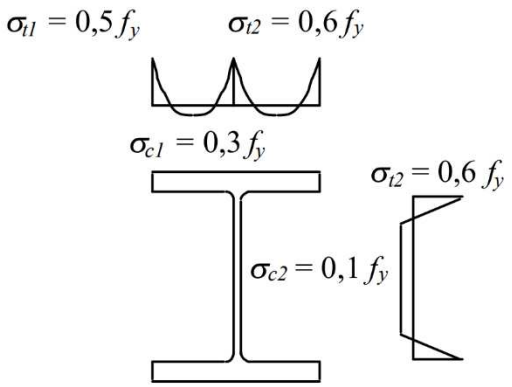


Figura An.B.5 – Distribuição de tensões residuais adotada nas análises de variações de parâmetros (Castro e Silva,2006).