UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA CIVIL

Germano Henrique Gelain

OTIMIZAÇÃO DE PERFIS SOLDADOS DE SEÇÃO I

Porto Alegre
2018

GERMANO HENRIQUE GELAIN

OTIMIZAÇÃO DE PERFIS SOLDADOS DE SEÇÃO I

Trabalho de Diplomação apresentado à Comissão de Graduação do curso de Engenharia Civil da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Engenheiro Civil

Orientador: Felipe Schaedler de Almeida

Porto Alegre

2018

GERMANO HENRIQUE GELAIN

OTIMIZAÇÃO DE PERFIS SOLDADOS DE SEÇÃO I

Este Trabalho de Diplomação foi julgado adequado como pré-requisito para a obtenção do título de ENGENHEIRO CIVIL e aprovado em sua forma final pelo Professor Orientador e pela Coordenadora da disciplina Trabalho de Diplomação Engenharia Civil II (ENG01040) da Universidade Federal do Rio Grande do Sul.

Porto Alegre, 19 de julho de 2018

Prof. Felipe Schaedler de Almeida
Doutor pela Universidade Federal do Rio Grande do Sul
Orientador

Prof. Samir Maghous
Doutor pela Ecole Nationale Des Ponts Chaussées (França)
Relator

Dedico este trabalho a meus pais, Romeo e Lúcia, que sempre me apoiaram, e, especialmente durante o período do meu Curso de Graduação, estiveram ao meu lado.

AGRADECIMENTOS

Agradeço ao professor Felipe Schaedler de Almeida por me orientar e aconselhar durante o desenvolvimento deste trabalho, possibilitando a elaboração e conclusão do mesmo.

Agradeço ao corpo docente da Universidade Federal do Rio Grande do Sul, por possibilitar o aprendizado e me guiar durante minha formação como engenheiro civil.

Agradeço aos meus valiosos amigos e colegas de faculdade, com quem pude contar nos momentos mais difíceis e que me mantiveram firme e confiante durante todo o curso de graduação. Em especial, agradeço a Bruno Valadão Cunha, Juliana Bertelli Nora e Alexandre Führ de Oliveira.

Agradeço à minha namorada, Vanessa Zaniol, pelo imenso suporte emocional e pela constante motivação, essenciais durante a elaboração deste trabalho.

Por fim, agradeço aos meus pais, Romeo José Gelain e Lúcia Smiderle Gelain, que sempre me forneceram todo apoio necessário e foram essenciais na construção do meu caráter e personalidade. Através deles, agradeço a todos membros da minha família, que igualmente colaboraram para meu desenvolvimento pessoal e acadêmico.

“Fairy tales are more than true: not because they tell us that dragons exist, but because they tell us that dragons can be beaten.”

Neil Gaiman – Coraline

RESUMO

Este trabalho consiste na confecção de uma ferramenta computacional para otimização de perfis de aço soldados de seção I, sujeitos a esforços de compressão axial simples. Foram considerados os critérios de segurança para o dimensionamento de estruturas de aço, determinados pela Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008), assim como as restrições arquitetônicas e comerciais. O trabalho versa sobre os problemas de otimização e o método *Sequential Quadratic Programming* (SQP) adotado na ferramenta, bem como as etapas de confecção do programa. Por fim, são realizados estudos de caso, com o intuito de comparar os resultados otimizados com perfis padronizados pela norma e visualizar a influência de cada dimensão da seção transversal e das restrições arquitetônicas no dimensionamento de perfis I otimizados. Conclui-se, assim, que a utilização de um algoritmo de otimização para o dimensionamento de estruturas reduz de forma significativa o consumo de materiais e, consequentemente, os custos de uma edificação. Também foi possível observar que as mesas possuem maior influência do que as almas na resistência de perfis I submetidos à compressão simples.

Palavras-chave: otimização estrutural, estruturas metálicas, SQP, NBR 8800, Python

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ASTM – *American Society for Testing and Materials*, ou Sociedade Americana de Testes e Materiais

NBR – Norma Brasileira

SQP – *Sequential Quadratic Programming*, ou Programação Quadrática Sequencial

SLSQP – *Sequential Least Squares Programming*, ou Programação Sequencial de Mínimos Quadráticos

UFRGS – Universidade Federal do Rio Grande do Sul

LISTA DE SÍMBOLOS

A – Área da seção transversal;

A_{ef} – Área efetiva da seção transversal;

A_g – Área bruta da seção transversal da barra;

b_{ef} – Largura efetiva da alma;

b_w – Largura das mesas;

E – Módulo de elasticidade do aço;

f_y – Tensão de resistência ao escoamento do aço;

h – Altura total da viga;

I_x – Momento de inércia da seção transversal em relação ao eixo x;

K_x – Coeficiente de flambagem;

L_x – Comprimento da barra;

$N_{c,Rd}$ – Força axial de compressão resistente de cálculo;

$N_{c,Sd}$ – Força axial de compressão solicitante de cálculo;

N_e – Força axial de flambagem elástica;

Q – Fator de redução total associado à flambagem local;

Q_A – Fator de redução associado à flambagem local referente à alma;

Q_S – Fator de redução associado à flambagem local referente às mesas;

t_f – Espessura das mesas;

t_w – Espessura da alma;

γ_{a1} – Coeficiente de ponderação das resistências;

λ_0 – Índice de esbeltez reduzido;

σ – Tensão que pode atuar na alma;

χ – Fator de redução associado à resistência à compressão.

SUMÁRIO

1 INTRODUÇÃO

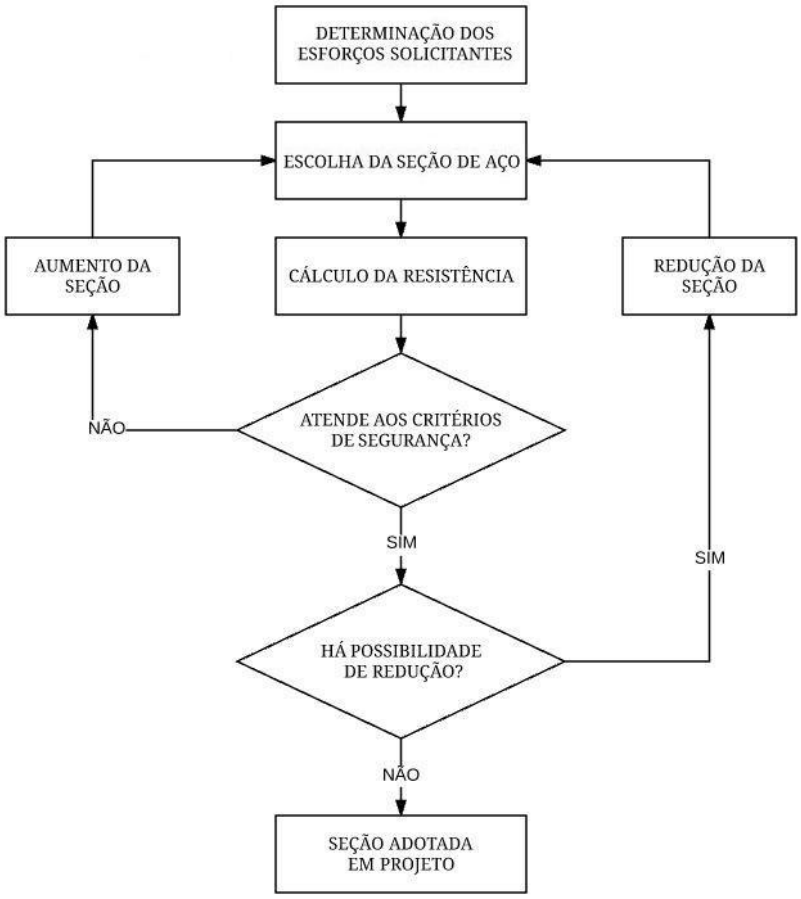
É ampla a variedade de possíveis soluções a serem aplicadas em estruturas convencionais na construção civil. Dentre as mais comuns e mais bem estabelecidas no Brasil estão o concreto armado, a alvenaria estrutural e as estruturas de aço. Segundo Pinheiro (2005, pg. 1), as vantagens da utilização de estruturas de aço podem ser listadas como: fabricação das estruturas com precisão milimétrica, possibilitando um alto controle de qualidade do produto acabado; Garantia das dimensões e propriedades dos materiais; Material resistente à vibração e a choques; Possibilidade de execução de obras mais rápidas e limpas; Em caso de necessidade, possibilita a desmontagem das estruturas e sua posterior montagem em outro local; Alta resistência estrutural, possibilitando a execução de estruturas leves para vencer grandes vãos; Possibilidade de reaproveitamento dos materiais em estoque, ou mesmo, sobras de obra. As desvantagens são: limitação de execução em fábrica, em função de transporte até o local de sua montagem final; Necessidade de tratamento superficial das peças contra oxidação, devido ao contato com o ar atmosférica; Necessidade de mão de obra e equipamentos especializados para sua fabricação e montagem; Limitação de fornecimento de perfis estruturais.

As estruturas de aço também possuem variação entre si. Fatores como os componentes da liga metálica, a robustez dos perfis e os processos de fabricação e de montagem são essenciais para a escolha da solução e a determinação dos procedimentos de dimensionamento e verificação de segurança. Os perfis soldados apresentam vantagem sobre os perfis laminados quando aplicados em estruturas mais robustas, pois, apesar de representarem custos maiores (em função dos seus processos de montagem), possuem maior flexibilidade quanto às suas dimensões, permitindo a confecção de seções maiores e mais ajustadas à demanda em cada elemento da estrutura.

O dimensionamento de estruturas de aço no Brasil é regulado pela Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) e consiste na aplicação do método dos estados-limites Últimos. O processo de dimensionamento pode ser dividido, de forma simplificada, em duas principais etapas: determinação dos esforços solicitantes e determinação da resistência ao esforço correspondente.

O objetivo de um bom projetista estrutural será sempre o de obter o menor custo possível para a execução de seu projeto, garantindo as condições de segurança e desempenho da estrutura. Isso implica na busca por seções ótimas, capazes de suportar os carregamentos existentes no projeto com a menor área de seção transversal possível. O trabalho a ser realizado para obter uma seção otimizada torna-se muito extenso, pois, conforme visto na Figura 1, o projetista precisa passar por um processo iterativo durante o dimensionamento de cada perfil.

Figura 1 – Fluxograma simplificado para dimensionamento de perfis de aço.



(Fonte: GELAIN, 2018)

Tendo esses fatores como motivação, este trabalho dedica-se a criar uma ferramenta capaz de facilitar o procedimento de dimensionamento. O programa desenvolvido deverá realizar os processos iterativos necessários e fornecer a seção ótima conforme solicitações definidas em projeto.

A otimização de estruturas é uma área já bastante estudada, com grande número de trabalhos desenvolvidos sobre o assunto. Segundo Haftka e Gürdal (1992, p. 1, tradução nossa): “A otimização preocupa-se em chegar ao melhor resultado de uma dada operação, ao mesmo tempo

que satisfaz determinadas restrições. Os seres humanos, guiados e influenciados pelos seus meios naturais, quase instintivamente exercem todas as suas funções de uma maneira que economize energia ou minimize o desconforto e a dor. A motivação é explorar os recursos disponíveis limitados de uma maneira que maximize a produção ou o lucro. As invenções antigas dos mecanismos de alavanca ou da polia são manifestações claras do desejo do homem de maximizar a eficiência mecânica".

Existem diferentes métodos para se atingir a otimização de funções matemáticas. Esses métodos variam de acordo com o tipo de função a ser otimizada, com as variáveis de projeto e com as restrições existentes. Sendo assim, é importante conhecer bem o problema, para que se possa escolher o método de otimização mais adequado. No caso deste trabalho, o objetivo é minimizar a função da área da seção transversal de um perfil I submetido à força axial de compressão. Para que isso seja possível, a área precisa ser descrita em função das variáveis de projeto. Também será considerada uma série de restrições, de forma a garantir que as geometrias obtidas sejam plausíveis e atendam aos critérios de segurança.

As variáveis de projeto consideradas neste trabalho serão as dimensões da seção transversal dos perfis. No caso dos perfis I soldados, a seção é composta pela união de três chapas independentes, com espessuras limitadas às dimensões fornecidas no mercado. Ficam como variáveis, então, as suas espessuras e as suas larguras. As restrições de projeto são divididas em duas partes: os valores mínimos e máximos de largura das chapas definidos por razões construtivas ou arquitetônicas e as verificações de segurança da Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) conforme os esforços já citados anteriormente.

Com o problema estruturado, segue-se para a confecção do programa em si. A linguagem de programação escolhida para este trabalho é a linguagem Python, que é gratuita e fornece ampla gama de bibliotecas e funções. Funciona com código aberto, de modo que o usuário pode modificar tais funções, caso julgue necessário, permitindo uma maior flexibilidade e garantindo a confecção de um programa mais ajustado ao objetivo desejado.

2 DIRETRIZES DE PESQUISA

O trabalho apresentado tem como objetivo principal desenvolver uma ferramenta computacional capaz de determinar a menor seção transversal de perfis I soldados, para suportar os esforços de compressão simples definidos em projeto, considerando os materiais e as dimensões existentes no mercado. Além disso, são realizados estudos de caso, análises paramétricas de problemas reais e comparativos com as soluções padronizadas encontradas no mercado.

O trabalho assume como critérios de segurança, para o dimensionamento dos perfis, os constantes na Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008), assim como as diretrizes de cálculo presentes na mesma Norma.

Com o intuito de simplificar o trabalho e garantir a obtenção de resultados coesos, optou-se por analisar somente os esforços de compressão simples, ignorando-se outros esforços e efeitos de segunda ordem. Assim, busca-se revisar critérios de dimensionamento de estruturas metálicas e adquirir conhecimentos sobre métodos de otimização e programação, permitindo a união entre conhecimentos prévios do curso de Engenharia Civil e habilidades novas, que serão úteis durante a vida profissional. A experiência obtida no presente trabalho é facilmente transferível para o tratamento de problemas similares, abrangendo outras configurações estruturais, materiais, solicitações e restrições.

É importante citar que, à partir da ferramenta produzida neste trabalho, a consideração de novos esforços solicitantes (tais como flexão e flexo-compressão) depende apenas da inserção das equações de verificação de segurança e suas respectivas derivadas ao código. Assim, abre-se a possibilidade de continuidade do presente trabalho, aproximando a ferramenta à casos reais.

Como limitações, o trabalho apresenta:

a) as solicitações consideradas: serão abordados apenas os esforços de compressão simples, excluindo-se os esforços de flexão, torção, cortantes e eventuais combinações dos mesmos;

b) os perfis analisados: serão considerados apenas perfis soldados, de seção I, com dupla simetria e seção transversal constante, formados por placas de espessuras comerciais;

c) as verificações realizadas: não serão abordadas neste trabalho as verificações quanto às ligações soldadas e/ou aos vínculos com outros elementos da estrutura;

d) a linguagem de programação adotada: será utilizada a linguagem Python, por se tratar de uma plataforma aberta e sem custos;

e) o método de otimização adotado: a abordagem do problema será feita através do método "Programação sequencial quadrática", que faz uso da projeção de vetores gradientes.

Assim, o trabalho delimita-se à criação de um programa apto a fornecer as dimensões para a menor seção transversal possível de perfis metálicos soldados de seção I, observando os critérios de segurança da Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) quanto aos esforços de compressão simples.

2.1 DELINEAMENTO

A confecção do trabalho deu-se através das etapas listadas a seguir:

a) Delimitação do problema

b) Pesquisa bibliográfica

c) Determinação dos parâmetros e das restrições de projeto

d) Ajuste das equações ao método de otimização

e) Confecção do programa

f) Análises paramétricas e comparativos

Durante a delimitação do problema, foi determinada a abrangência do trabalho quanto aos esforços solicitantes e ao tipo de estrutura considerado. Foram escolhidos para análise os esforços de compressão simples e as vigas de aço soldadas, de seção I.

A pesquisa bibliográfica teve como objetivo fornecer os conhecimentos necessários para confecção do trabalho, prezando por fontes como livros e trabalhos acadêmicos. Essa etapa estendeu-se ao longo de toda a elaboração do trabalho, colaborando para sua qualificação em termos de contextualização e referenciação.

Para determinar os parâmetros de dimensionamento, foi necessário consultar e revisar os critérios de segurança constantes na Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008). Quanto às restrições de projeto, não apenas a Norma foi consultada, como também outras bibliografias foram utilizadas, a fim de se identificar as dimensões disponíveis dos produtos visados usualmente oferecidos pelo mercado.

Após a determinação das equações utilizadas na verificação de segurança, foi necessário calcular suas derivadas em relação às variáveis de projeto, a fim de possibilitar a utilização do método de otimização escolhido. Também foi necessário transformar as restrições em inequações, de modo a adequá-las ao algoritmo de otimização adotado.

Em seguida, deu-se início à confecção do programa em si. Nessa etapa, foi necessária a realização de um pequeno curso de programação sobre a linguagem escolhida, permitindo a estruturação geral do programa e a definição das funções utilizadas. Além disso, uma extensa pesquisa foi realizada a fim de identificar bibliotecas que contenham funções capazes de realizar a otimização em questão, permitindo funções e restrições conforme as descritas anteriormente.

Por fim, através da realização de um estudo de caso, foi possível traçar comparativos entre os resultados obtidos através da otimização e os resultados padronizados encontrados na bibliografia. Dessa forma, foi possível visualizar a eficiência e a importância da aplicação de otimização estrutural em problemas reais. Além disso, foram abordados problemas secundários paralelos, nos quais avaliou-se a influência das restrições geométricas no dimensionamento de estruturas.

3 CRITÉRIOS DE SEGURANÇA

Nesta seção, abordamos as verificações de segurança adotadas pela Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) referentes aos estados-limites últimos (ELU) de barras comprimidas.

Os estados-limites últimos correspondem às condições a partir das quais a estrutura deixa de atender a uma das finalidades de sua construção. Pode estar associada à ruptura das seções, ao colapso da estrutura, à perda de estabilidade, à deterioração por fadiga, entre outras complicações. Assim, os esforços resistentes de projeto devem ser sempre superiores aos valores de esforços solicitantes obtidos.

Alguns parâmetros devem ser determinados previamente para que se possa calcular os esforços resistentes de projeto de barras de aço. São eles: as propriedades inerentes ao material adotado, as propriedades geométricas do perfil e o comprimento de flambagem das barras.

Este trabalho não versará sobre a obtenção dos esforços solicitantes de cálculo.

3.1 PARÂMETROS INICIAIS

Abaixo, serão abordadas as propriedades do aço utilizado no dimensionamento das estruturas e as propriedades geométricas da seção I.

3.1.1 Propriedades do material

Os principais aços utilizados em perfis soldados na construção civil são classificados de acordo com suas tensões limites de escoamento, sendo eles MR-250 (ASTM A36), AR-290, AR-350 (ASTM A5761), AR-415, entre outros. No presente trabalho, será adotado como padrão o aço AR-350.

As propriedades de interesse para a determinação da resistência de projeto dos perfis analisados são:

a) tensão limite de escoamento: a Norma NBR 7007 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) estabelece o valor de 350MPa para o aço AR-350;

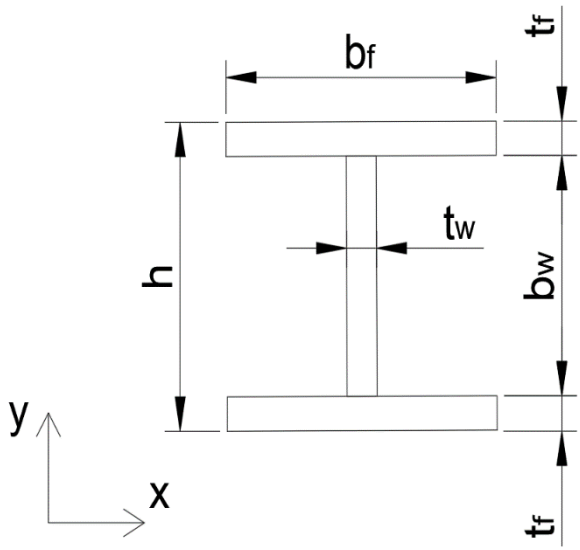
b) módulo de elasticidade longitudinal: o valor estabelecido para o aço é de 200.000 MPa;

c) módulo de elasticidade transversal: calculado através do módulo de elasticidade longitudinal e do coeficiente de Poisson (0,30 para o aço), dado como 75.800MPa.

3.1.2 Propriedades geométricas

Os perfis analisados são compostos por seções I de dupla simetria, conforme retratado na Figura 2:

Figura 2 – Seção transversal de um perfil I soldado, com suas dimensões principais e os eixos x e y.



(Fonte: GELAIN, 2018)

de modo que:

h = altura total do perfil;

t_f = espessura das mesas;

t_w = espessura da alma;

b_f = largura das mesas.

Também podemos introduzir a dimensão correspondente à largura da alma:

$$b_w = (h - 2t_f) \qquad \text{(fórmula 1)}$$

A área bruta da seção transversal da barra é dada por:

$$A_g = 2b_f t_f + b_w t_w \qquad \text{(fórmula 2)}$$

Os momentos de inércia da seção transversal, em relação aos eixos x e y, são definidos, respectivamente, por:

$$I_x = \frac{t_w {b_w}^3}{12} + 2\left[\frac{b_f {t_f}^3}{12} + b_f t_f \left(\frac{h - t_f}{2}\right)^2\right] \qquad \text{(fórmula 3)}$$

e:

$$I_y = \frac{b_w {t_w}^3}{12} + 2\frac{t_f {b_f}^3}{12} \qquad \text{(fórmula 4)}$$

O raio de giração da seção em relação ao eixo central x é:

$$r_x = \sqrt{\frac{I_x}{A_g}} \qquad \text{(fórmula 5)}$$

analogamente, o raio de giração da seção em relação ao eixo central y é:

$$r_y = \sqrt{\frac{I_y}{A_g}} \qquad \text{(fórmula 6)}$$

e o raio de giração polar da seção em relação ao centro de cisalhamento, que coincide com o centroide (caso de seções com dupla simetria) é dado por:

$$r_0 = \sqrt{\left({r_x}^2 + {r_y}^2\right)} \qquad \text{(fórmula 7)}$$

A constante de empenamento da seção transversal, para seções I é:

$$C_w = \frac{I_y\left(h - t_f\right)^2}{4} \qquad \text{(fórmula 8)}$$

A constante de torção da seção transversal, para seções I é:

$$J = \frac{2b_f {t_f}^3 + \left(h - t_f\right){t_w}^3}{3} \qquad \text{(fórmula 9)}$$

3.2 VERIFICAÇÕES DE SEGURANÇA

Para os esforços de compressão, a Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) estabelece a seguinte verificação:

$$N_{c,Sd} \leq N_{c,Rd} \qquad \text{(fórmula 10)}$$

em que:

$N_{c,Sd}$ = força axial de compressão solicitante de cálculo;

$N_{c,Rd}$ = força axial de compressão resistente de cálculo.

Para a determinação da força de compressão resistente de cálculo ($N_{c,Rd}$) a Norma estabelece:

$$N_{c,Rd} = \frac{\chi Q A_g f_y}{\gamma_{a1}} \qquad \text{(fórmula 11)}$$

em que:

χ = fator de redução associado à resistência à compressão;

Q = fator de redução total associado à flambagem local;

f_y = tensão de resistência ao escoamento do aço;

γ_{a1} = coeficiente de ponderação das resistências.

O fator de redução associado à resistência à compressão (χ) é dado por:

para $\lambda_0 \leq 1{,}5$ $\qquad \chi = 0{,}658^{{\lambda_0}^2} \qquad$ (fórmula 12)

para $\lambda_0 > 1{,}5$: $$\chi = \frac{0{,}877}{{\lambda_0}^2} \quad \text{(fórmula 13)}$$

onde λ_0 é o índice de esbeltez reduzido, dado por:

$$\lambda_0 = \sqrt{\frac{QA_g f_y}{N_e}} \quad \text{(fórmula 14)}$$

sendo N_e a força axial de flambagem elástica, que é dada pelo menor valor entre N_{ex}, N_{ey}, e N_{ez}, determinados conforme as fórmulas à seguir:

$$N_{ex} = \frac{\pi^2 E I_x}{(K_x L_x)^2} \quad \text{(fórmula 15)}$$

em que:

E = módulo de elasticidade do aço;

K_x = coeficiente de flambagem na direção x;

L_x = comprimento de flambagem por flexão em relação ao eixo x.

$$N_{ey} = \frac{\pi^2 E I_y}{(K_y L_y)^2} \quad \text{(fórmula 16)}$$

Em que:

K_y = coeficiente de flambagem na direção y;

L_y = comprimento de flambagem por flexão em relação ao eixo y.

$$N_{ez} = \frac{1}{{r_0}^2}\left[\frac{\pi^2 E C_w}{(K_z L_z)^2} + GJ\right] \quad \text{(fórmula 17)}$$

Em que:

G = módulo de elasticidade transversal do aço;

J = constante de torção da seção transversal;

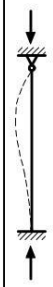
K_z = coeficiente de flambagem na direção z;

L_z = comprimento de flambagem por torção.

Os comprimentos de flambagem são determinados pelo vão livre entre os dois vínculos de extremidade de uma barra. Já os respectivos coeficientes de flambagem são determinados de

acordo com o tipo de vínculo atribuído. A Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) recomenda o uso dos coeficientes conforme os valores dados na Tabela 1:

Tabela 1 – Coeficientes de flambagem nas direções x e y, de acordo com a Norma NBR 8800.

A linha tracejada indica a linha elástica de flambagem	**(a)** 	**(b)** 	**(c)** 	**(d)** 	**(e)** 	**(f)** 
Valores teóricos de K_x ou K_y	0,5	0,7	1,0	1,0	2,0	2,0
Valores recomendados	0,65	0,80	1,2	1,0	2,1	2,0
Código para condição de apoio		Rotação e translação impedidas Rotação livre, translação impedida Rotação impedida, translação livre Rotação e translação livres				

(Fonte: Norma NBR8800, 2008)

Os valores teóricos devem ser utilizados em situações em que seja possível obter vinculação perfeita e valores recomendados em situações contrárias. Para barras contraventadas e para os contraventamentos em si, os valores de K_x e K_y devem ser iguais a 1,0.

O valor de K_z deve ser determinado por análise estrutural ou, simplificadamente, tomado igual a 1,00, quando ambas as extremidades da barra possuírem rotação em torno do eixo longitudinal impedida e empenamento livre, ou igual a 2,00, quando uma das extremidades possuir rotação em torno do eixo longitudinal e empenamento livre.

Para determinar o fator de redução total associado à flambagem local (Q) é necessário que se faça uma análise mais detalhada da seção, separando-a em seus elementos constituintes, mesas e alma. Os elementos são classificados entre: AA (duas bordas longitudinais vinculadas), correspondente à alma, e AL (apenas uma borda longitudinal vinculada), correspondente às mesas.

O valor de Q é dado por:

$$Q = Q_A Q_S \quad \text{(fórmula 18)}$$

em que:

Q_A = fator de redução referente à alma;

Q_S = fator de redução referente às mesas.

Para determinar os valores de Q_A e Q_S, deve-se verificar se a relação entre largura e espessura de cada elemento está abaixo do limite esperado para que não haja redução por flambagem local. As verificações são dadas a seguir:

para a alma: $$\frac{b_w}{t_w} \leq 1{,}49\sqrt{\frac{E}{f_y}} \quad \text{(fórmula 19)}$$

para as mesas: $$\frac{b_f/2}{t_f} \leq 0{,}64\sqrt{\frac{E}{f_y/k_c}} \quad \text{(fórmula 20)}$$

em que k_c é dado por:

$$k_c = \frac{4}{\sqrt{b_w/t_w}} \quad \text{(fórmula 21)}$$

sendo que: $$0{,}35 \leq k_c \leq 0{,}76 \quad \text{(fórmula 22)}$$

Assim, caso o limite expressado na fórmula 19 seja respeitado, o valor de Q_A será 1, significando que não há redução da força de compressão resistente de cálculo devido à flambagem local da alma. Porém, caso o limite não seja respeitado, o valor de Q_A será dado pela fórmula a seguir:

$$Q_A = \frac{A_{ef}}{A_g} \quad \text{(fórmula 23)}$$

em que A_{ef} é a área efetiva da seção transversal que, em perfis de seção I soldada, é dada por:

$$A_{ef} = A_g - [b_w - b_{ef}]t_w \quad \text{(fórmula 24)}$$

em que b_{ef} é a largura efetiva da alma, dada por:

$$b_{ef} = 1{,}92 t_w \sqrt{\frac{E}{\sigma}} \left[1 - \frac{0{,}34}{b_w/t_w} \sqrt{\frac{E}{\sigma}}\right] \leq b_w \quad \text{(fórmula 25)}$$

em que σ é a tensão que pode atuar na alma, dada por (adotando-se Q igual a 1,0 no cálculo de χ):

$$\sigma = \chi f_y \quad \text{(fórmula 26)}$$

Analogamente, caso o limite expressado na fórmula 20 seja respeitado, o valor de Q_S será 1, significando que não há redução da força de compressão resistente de cálculo devido à flambagem local das mesas. Porém, caso o limite não seja respeitado, o valor de Q_S será dado pelas fórmulas a seguir:

para: $$0{,}64 \sqrt{\frac{E}{f_y/k_c}} < \frac{b_f/2}{t_f} < 1{,}17 \sqrt{\frac{E}{f_y/k_c}} \quad \text{(fórmula 27)}$$

tem-se: $$Q_s = 1{,}415 - 0{,}65 \frac{b_f/2}{t_f} \sqrt{\frac{f_y}{k_c E}} \quad \text{(fórmula 28)}$$

e para: $$\frac{b_f/2}{t_f} > 1{,}17 \sqrt{\frac{E}{f_y/k_c}} \quad \text{(fórmula 29)}$$

tem-se: $$Q_s = \frac{0{,}90 E k_c}{f_y \left(\frac{b_f/2}{t_f}\right)^2} \quad \text{(fórmula 30)}$$

Finalmente, o coeficiente de ponderação das resistências (γ_{a1}) é dado como 1,10 para combinações normais de projeto, conforme a Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008).

A Norma também determina uma limitação para o índice de esbeltez das barras, conforme as relações dadas a seguir:

$$\frac{K_x L_x}{r_x} < 200 \qquad \text{(fórmula 31)}$$

e:

$$\frac{K_y L_y}{r_y} < 200 \qquad \text{(fórmula 32)}$$

4 FORMULAÇÃO DO PROBLEMA

Em problemas de otimização estrutural, o projetista deve definir a função objetivo, as variáveis de projeto e as restrições. Esses critérios são particulares de cada problema, e as suas características delimitam quais métodos de otimização podem ou não ser aplicados na solução. O esquema genérico de um problema de otimização é descrito por:

Minimizar/maximizar: $f(x)$

Tal que: $\begin{cases} g_i(x) \leq 0; & i = 1,2,3 \dots n_1 \\ h_j(x) = 0; & j = 1,2,3 \dots n_2 \end{cases}$

Onde $f(x)$ é a função objetivo, $g_i(x) \leq 0$ e $h_j(x) = 0$ são as restrições de desigualdade e de igualdade, respectivamente, e x são as variáveis de projeto.

Sobre problemas de otimização restringidos, Haftka e Gürdal (1992, p. 135, tradução nossa) afirmam que: "as restrições dividem o espaço de dimensionamento em dois domínios, o domínio viável, no qual as restrições são satisfeitas, e o domínio inviável, no qual pelo menos uma das restrições é violada. Em grande parte dos problemas práticos, o mínimo é encontrado na fronteira entre os domínios viável e inviável, ou seja, um ponto em que $g_k(x) = 0$ para pelo menos um k. Caso contrário, as inequações de restrição podem ser removidas sem alterar a solução. ".

4.1 VARIÁVEIS DE PROJETO

De acordo com Haftka e Gürdal (1992, p. 3, tradução nossa), "a noção de aperfeiçoar ou otimizar uma estrutura pressupõe, implicitamente, certa liberdade para modificá-la. O potencial de mudança é tipicamente expressado em termos da gama de mudanças permissíveis de um grupo de parâmetros. Tais parâmetros são usualmente chamados de *variáveis de projeto* na linguagem de otimização estrutural. Variáveis de projeto podem ser dimensões de seções transversais ou comprimento de membros, podem ser parâmetros controlando a geometria da estrutura, as propriedades de seus materiais, etc. Variáveis de projeto podem ser *contínuas* ou *discretas*. Variáveis contínuas possuem uma amplitude de variação, e podem assumir qualquer

valor neste intervalo. (...) Variáveis discretas podem assumir apenas valores isolados, tipicamente de uma lista de valores permissíveis."

Em um problema de otimização, buscando a menor área de seção transversal, as variáveis de projeto serão os valores das dimensões da geometria em questão. No caso deste trabalho, as dimensões de um perfil I, soldado. Ou seja, as larguras e espessuras de seus elementos, conforme mostra a Figura 2.

As chapas horizontais são chamadas de mesas, e a vertical é denominada alma da viga. No caso dos perfis soldados, cada peça é fornecida separadamente, fazendo com que as dimensões de projeto a serem passadas para o fabricante sejam referentes à largura e espessura, além, é claro, do comprimento das chapas.

Segundo Pfeil e Pfeil (2009, p. 8), a fabricação de chapas de aço passa por um processo denominado laminação. Esse processo acarreta restrições referentes aos valores de espessura possíveis, visto que o custo para que se possa alterar a configuração dos rolos a cada chapa produzida seria inviável, fazendo com que as espessuras assumam uma série de valores discretos. A mesma restrição não se aplica quanto às larguras das chapas produzidas, pois os procedimentos de corte existentes na indústria permitem que elas assumam valores contínuos. Sendo assim, ficam definidas como variáveis de projeto as dimensões h e b_f, contínuas. Para considerar as dimensões t_w e t_f no processo de otimização, o programa desenvolvido considerará um par de espessuras por vez, obtendo um resultado ótimo para o par analisado e comparando-o com o melhor resultado obtido até então. Assim, ao final da otimização, o par de espessuras que apresentar o resultado com a menor área da seção transversal será também o resultado ótimo do problema principal.

4.2 FUNÇÃO OBJETIVO

O objetivo do processo de otimização é encontrar o valor das variáveis de projeto que levam ao menor valor possível da função objetivo, respeitando as funções de restrição. A função objetivo define matematicamente uma quantidade que reflete o objetivo da otimização. Segundo Klarbring e Christensen (2009, pg. 3, tradução nossa), a função objetivo (f) é descrita como uma função utilizada para classificar o dimensionamento. Para cada configuração possível, f retorna um número que indica a qualidade da configuração. Usualmente se escolhe f, de modo

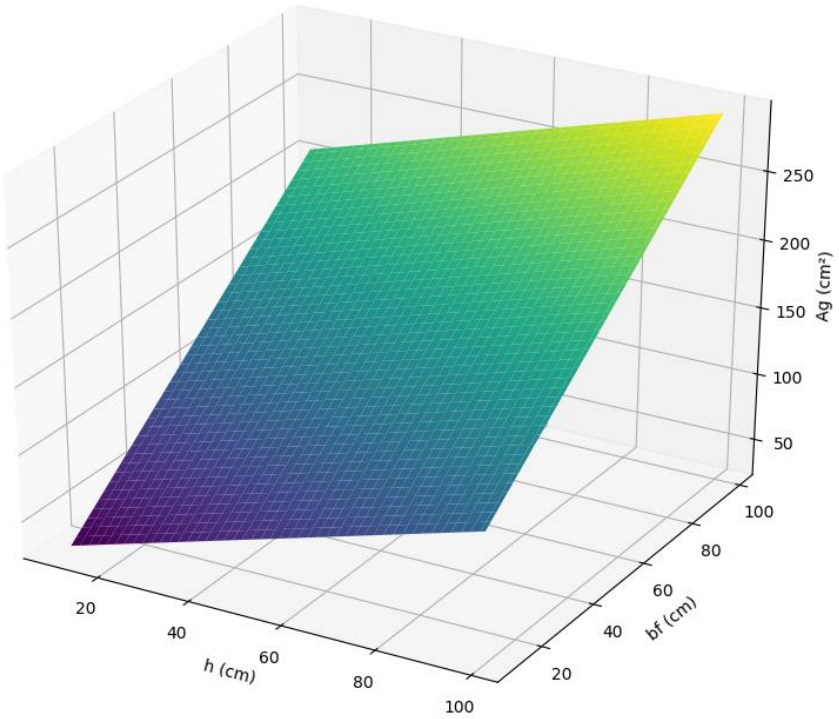
que um valor pequeno é melhor do que um grande (um problema de minimização). Frequentemente, f mede peso, deslocamento em uma dada direção, tensão efetiva ou até custo de produção.

A função objetivo deste trabalho será a função que descreve a área da seção transversal de um perfil I. Conforme mostrado na Figura 2, a área pode ser descrita em função das variáveis h, b_f, t_f e t_w. Porém, para o método de otimização adotado neste trabalho, as variáveis discretas serão consideradas como constantes para cada versão do problema, conforme será desenvolvido no item 4 do texto. Sendo assim, a função da área é dada a seguir:

$$A(h, b_f) = \ 2b_f t_f + \ (h - 2t_f)t_w \qquad \text{(fórmula 33)}$$

Assumindo valores unitários para t_w e t_f, é possível se obter uma visualização gráfica da função, conforme retratado na Figura 3; possibilitando, então, concluir que a relação entre h e b_f será sempre linear, para qualquer valor fixado de A.

Figura 3 – Função A(h, b_f), plotada em um espaço 3D.



(Fonte: GELAIN, 2018)

4.3 RESTRIÇÕES

As restrições de projeto são as condições que devem ser satisfeitas para que o resultado obtido na função objetivo seja válido, ou seja, são as equações e inequações limitantes do projeto. Esses são os critérios que fazem com que seja necessária a aplicação de um método de otimização no problema deste trabalho, pois, sem ele, as variáveis de projeto assumiriam sempre o menor valor possível.

As restrições serão divididas em três categorias: restrições comerciais, definidas pelas espessuras das chapas de aço encontradas no mercado, restrições de segurança, correspondentes às verificações existentes na Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) e restrições arquitetônicas, estabelecidas por questões estéticas, construtivas ou de utilização das edificações.

4.3.1 Restrições comerciais

Conforme abordado anteriormente, os processos de fabricação por laminação impõem limites às espessuras das chapas de aço. Por outro lado, os valores de largura, inerentes aos processos de corte das chapas, podem adotar valores contínuos.

Os valores usuais de mercado, que serão utilizados, neste trabalho, para as dimensões de chapas de aço, são fornecidos por Pfeil e Pfeil (2009, p. 336-347): 6,3mm, 8mm, 9,5mm, 12,5mm, 16mm, 19mm, 22,4mm, 25mm, 31,5mm, 37,5mm e 45mm de espessura, tanto para as mesas quanto para a alma dos perfis.

4.3.2 Restrições de segurança

As restrições definidas pelos critérios de segurança, impostos pela Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) foram descritas no capítulo 3 deste trabalho. Porém, para que tais equações funcionem de forma mais adequada para os métodos de otimização, elas deverão passar por pequenas manipulações matemáticas, resultando nas expressões a seguir:

$$N_{c,Sd} - N_{c,Rd} < 0 \qquad \text{(fórmula 34)}$$

$$200 - \frac{K_x L_x}{r_x} < 0 \qquad \text{(fórmula 35)}$$

$$200 - \frac{K_y L_y}{r_y} < 0 \qquad \text{(fórmula 36)}$$

4.3.3 Restrições arquitetônicas

As restrições arquitetônicas são determinadas de acordo com as limitações geométricas de cada projeto em particular. É comum em obras de engenharia que as peças estruturais tenham que se adaptar a elementos construtivos ou arquitetônicos predeterminados, o que impõe limites nas dimensões de cada elemento do perfil a ser dimensionado. Dessa forma, tais limitações determinam intervalos nos quais os valores de h e b_f devem estar contidos. Os limites serão representados pelos termos h_{MIN}, h_{MAX}, $b_{f\,MIN}$ e $b_{f\,MAX}$.

4.4 FORMULAÇÃO RESUMIDA

Com a função objetivo, as variáveis de projeto e as restrições devidamente especificadas, o problema de otimização abordado no presente trabalho fica resumido por:

Minimizar: $A(h, b_f) = 2b_f t_f + (h - 2t_f)t_w$

Tal que:

$$\begin{cases} g_1(h, b_f) = N_{c,Rd} - N_{c,Sd} < 0 \\ g_2(h, b_f) = 200 - \frac{K_x L_x}{r_x} < 0 \\ g_3(h, b_f) = 200 - \frac{K_y L_y}{r_y} < 0 \\ h_{MIN} \leq h \leq h_{MAX} \\ b_{f\,MIN} \leq b_f \leq b_{f\,MAX} \end{cases}$$

5 IMPLEMENTAÇÃO DO PROGRAMA DE DIMENSIONAMENTO OTIMIZADO

Com o problema de otimização devidamente posto, é possível partir para a confecção do algoritmo computacional que será responsável por resolvê-lo. Nessa etapa, dois fatores devem ser levados em consideração: a escolha de uma linguagem de programação para a qual estejam disponíveis bibliotecas adequadas para a resolução do problema e que seja familiar ao programador; e a criação de um algoritmo que represente adequadamente o problema abordado.

5.1 LINGUAGEM DE PROGRAMAÇÃO

Para aprimorar o trabalho, foi necessário escolher uma linguagem que fosse amigável a novos usuários e que permitisse seu aprendizado de forma rápida e eficiente. Portanto, selecionamos a linguagem Python, concebida com o objetivo de aprimorar a leitura e a visualização do código, permitindo maior facilidade de interpretação, o que a torna bastante apropriada a novos usuários.

Além disso, as bibliotecas de funções criadas em Python são necessariamente de código aberto, implicando que qualquer usuário no mundo pode modificá-las para melhor se adaptar aos seus problemas, ou incrementá-las/otimizá-las, aumentando sua eficiência. Dessa forma, e em conjunto com uma comunidade bastante ativa de programadores, Python conta com uma larga escala de bibliotecas e funções capazes de abordar os mais diversos problemas, mesmo sendo uma linguagem relativamente nova em relação a linguagens mais tradicionais, como C, C++ e Fortran. Especificamente, para o presente trabalho, foi necessário adotar uma linguagem com disponibilidade de rotinas e algoritmos de otimização, o que pôde ser verificado em Python.

5.2 ESTRUTURA DO PROGRAMA

Após a realização de um curso introdutório às funcionalidades da linguagem adotada, foi possível conceber o algoritmo através do qual o problema deveria ser resolvido. O problema consiste na otimização de uma função, dependente de quatro variáveis (duas contínuas e duas discretas), submetida a cinco restrições (três inequações e dois intervalos de valores para as variáveis contínuas). As etapas de elaboração do programa capaz de resolvê-lo serão descritas nas subseções a seguir. As linhas de código escritas durante a confecção do programa, por sua vez, estão disponíveis no Apêndice A, apresentando indicações e legendas para cada etapa do programa.

5.2.1 Definição das funções

Primeiramente, todas as funções inerentes ao dimensionamento e à verificação das barras escolhidas foram implementadas, conforme exemplo ilustrado na Figura 4. As funções são dependentes de um vetor "x", composto pelas duas variáveis contínuas do problema, h e b_f, dos demais dados de geometria, como l_x, e de outras funções previamente definidas no código, como $r_0(x)$.

Figura 4 – Definição das funções de determinação da força axial de flambagem elástica.

```
def Nex(x):
    return ((math.pi**2)*E*Ix(x))/((kx*lx)**2)
def Ney(x):
    return ((math.pi**2)*E*Iy(x))/((ky*ly)**2)
def Nez(x):
    return (1/(r0(x)**2))*(((math.pi**2)*E*Cw(x))/((kz*lz)**2)+(G*J(x)))
def Ne(x):
    if Nex(x) < Ney(x) and Nex(x) < Nez(x):
        return Nex(x)
    elif Ney(x) < Nex(x) and Ney(x) < Nez(x):
        return Ney(x)
    else:
        return Nez(x)
```

(Fonte: GELAIN, 2018)

Isto ocasionou a criação de um programa prévio, mais simples, capaz de realizar a verificação de segurança de perfis de aço de seção I, soldados, submetidos a esforços de compressão simples, conforme dados de geometria e solicitações informados. Esse programa permitiu que as equações traduzidas em código fossem verificadas, através de comparações com exemplos existentes na literatura e exercícios realizados manualmente, garantindo que sua formulação tenha sido corretamente inserida no código.

5.2.2 Variáveis discretas

Como o método de otimização utilizado considera variáveis contínuas, foi preciso utilizar uma estratégia diferente para incluir as espessuras das chapas na otimização. A metodologia adotada foi a de criar dois *loops*, um dentro do outro, cada um contendo uma lista com os valores de espessura disponíveis, conforme ilustrado na Figura 5.

Figura 5 – Estratégia adotada para determinação das espessuras ótimas.

```
tw_lista = [0.63, 0.8, 0.95, 1.25, 1.6, 1.9, 2.24, 2.5, 3.15, 3.75, 4.5]
tf_lista = [0.63, 0.8, 0.95, 1.25, 1.6, 1.9, 2.24, 2.5, 3.15, 3.75, 4.5]
for tw in tw_lista:
    for tf in tf_lista:
```

(Fonte: GELAIN, 2018)

Dessa forma, o código realiza um processo iterativo para cada par de espessuras existente, totalizando 121 processos, buscando os valores ótimos de h e b_f para cada t_f e t_w fixados dentro do *loop* corrente. Após a realização de cada processo iterativo, o programa compara o valor obtido para a função objetivo com o menor valor registrado até o momento. Caso o novo valor seja menor, os valores das variáveis de projeto são armazenados nas variáveis "ótimas", ocasionando que, ao final dos *loops*, o valor armazenado corresponda ao mínimo global do problema.

Figura 6 – Comparação entre os valores de Ag ótima global e Ag ótima obtida ao final de cada *loop*.

```
if res.fun < Ag_otim and (Ncrd(res.x)/Ncsd) > (0.999999) and res.message == 'Optimization terminated successfully.':
    Ag_otim = Ag(res.x)
    x_otim = [res.x[0], res.x[1]]
    tw_otim = tw
    tf_otim = tf
    Ncrd_otim = Ncrd(res.x)
```

(Fonte: GELAIN, 2018)

5.2.3 Variáveis contínuas

Nessa etapa do código, ocorrem as iterações para determinação dos valores ótimos de h e b_f e, portanto, da função objetivo. Para que a solução de um problema de otimização seja eficiente e forneça o melhor resultado possível (ótimo global), é necessário que se faça uma análise das características da formulação dele. O tipo de função objetivo, a existência ou não de restrições e a forma como elas apresentam-se são fatores determinantes na escolha da metodologia aplicada.

No problema tratado neste trabalho, a função objetivo é uma função linear de duas variáveis (posto que as espessuras foram retiradas do processo iterativo) e as restrições apresentam-se através de inequações (os intervalos de valores máximos e mínimos podem ser descritos como pares de inequações). Essas são as características que devem ser adequadas para o método de otimização adotado.

O método de programação quadrática sequencial (*Sequential Quadratic Programming* – SQP) é um poderoso método capaz de resolver problemas de otimização com as características estipuladas acima e, portanto, é o método adotado no presente trabalho.

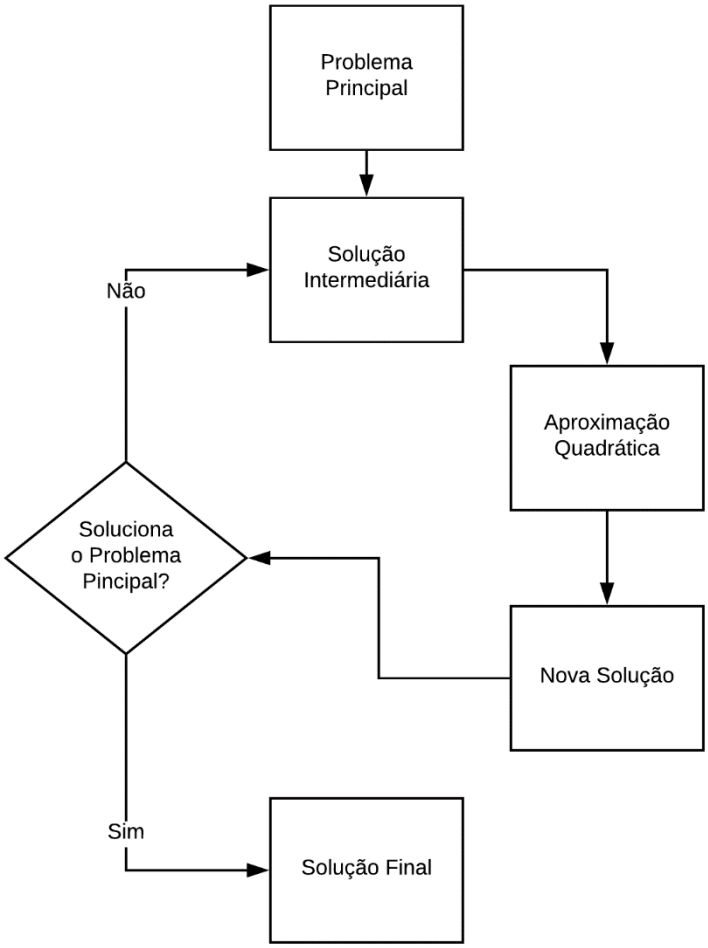
5.2.3.1 *Sequential Quadratic Programming* – SQP

De forma resumida, o método baseia-se na resolução de uma sequência de subproblemas mais simples, gerados a partir do problema inicial, cujas soluções convergem para a solução do problema principal. Nos subproblemas gerados, as funções objetivo são aproximações quadráticas da função objetivo original e as restrições são aproximações lineares das restrições originais, justificando o nome de Programação Quadrática Sequencial.

A partir de um conjunto de variáveis x^i (que pode ou não satisfazer as restrições do problema original) é gerado um subproblema quadrático, cujas função objetivo e restrições são aproximações fiéis ao problema original na região do ponto adotado. Esse subproblema, então, é resolvido através de métodos de otimização mais simples, e sua solução é adotada como um novo conjunto de variáveis x^{i+1}, dando origem a um novo subproblema quadrático. As iterações seguem até que seja encontrado um conjunto de variáveis que satisfaça o problema de otimização original. A Figura 7 apresenta um fluxograma resumido das etapas executadas durante a solução de um problema de programação quadrática sequencial.

Os métodos SQP variam, portanto, de acordo com o método de otimização utilizado para resolver os subproblemas mais simples, gerados a partir da função objetivo. Como os subproblemas são compostos por uma função objetivo quadrática e funções de restrição lineares, diversos métodos podem ser aplicados nessa etapa.

Figura 7 - Fluxograma resumido do processo iterativo realizado em problemas SQP



Para que fosse possível solucionar o problema através de um método SQP, foi necessário buscar uma biblioteca de funções em Python que contivesse tal ferramenta. A biblioteca encontrada que melhor satisfaz os critérios necessários é a SciPy, que possui uma ferramenta de minimização de funções, contando com diversos métodos de otimização embutidos.

O método SQP contido nessa biblioteca é chamado de Programação de Mínimos Quadráticos Sequenciais (*Sequential Least Squares Programming* – SLSQP), uma variante que utiliza o método dos mínimos quadrados na determinação dos valores ótimos dos subproblemas quadráticos.

5.2.3.2 *Sequential Least Squares Programming* – SLSQP

Segundo The SciPy Community (2018), o método foi baseado na rotina de programação descrita por por Kraft, D. (1988, p. 9-13) para aplicação de métodos SQP que, por sua vez, cita o método descrito por Schittkowski, K. (1981, p. 115 – 127) para substituição dos subproblemas de programação quadrática por subproblemas de mínimos quadráticos.

O método dos mínimos quadráticos consiste em encontrar o vetor de soluções que minimiza a soma dos quadrados das diferenças entre um valor estimado e a solução real. Ou seja, o método busca encontrar o conjunto de variáveis mais próximo possível do vetor de soluções reais, minimizando a soma do quadrado das distâncias (em módulo) entre eles para cada ponto. O método pode ser aplicado somente após a realização de diversas operações matemáticas, descritas por Schittkowski, K. (1981, p. 115 – 127).

Figura 8 – Ferramenta de otimização utilizada e definição dos argumentos inseridos.

```
def jac_fun(x):
    dAg_dh = tw
    dAg_dbf = 2*tf
    return np.array([dAg_dh, dAg_dbf])
def jac_con1(x):
    return np.array([dNcrd_dh(x), dNcrd_dbf(x)])
def jac_con2(x):
    dh = -(kx*lx*drx_dh(x))/(rx(x)**2)
    dbf = -(kx*lx*drx_dbf(x))/(rx(x)**2)
    return np.array([dh, dbf])
def jac_con3(x):
    dh = -(ky*ly*dry_dh(x))/(ry(x)**2)
    dbf = -(ky*ly*dry_dbf(x))/(ry(x)**2)
    return np.array([dh, dbf])
bnds = ((h_inf, h_sup), (bf_inf, bf_sup))
cons = ({'type': 'ineq',
         'fun': lambda x: Ncrd(x) - Ncsd,
         'jac': jac_con1},
        {'type': 'ineq',
         'fun': lambda x: 200 - ((kx*lx)/rx(x)),
         'jac': jac_con2},
        {'type': 'ineq',
         'fun': lambda x: 200 - ((ky*ly)/ry(x)),
         'jac': jac_con3})
res = opt.minimize(Ag,
                   (h_sup, bf_sup),
                   method='SLSQP',
                   jac=jac_fun,
                   bounds=bnds,
                   constraints=cons,
                   options={'maxiter': 200})
```

(Fonte: GELAIN, 2018)

Para implementação no código, a ferramenta "*opt.minimize*" exige que sejam informados alguns parâmetros e permite outros parâmetros opcionais, conforme Figura 7. Obrigatoriamente, devem ser fornecidas a função objetivo, um palpite inicial, o método de otimização escolhido (*method*) e as funções restritivas (*constraints*). Opcionalmente, podem ser

fornecidos os vetores jacobianos da função objetivo (*jac*), os limites superiores e inferiores para as variáveis (*bounds*) e o número máximo de iterações (*maxiter*). Caso não sejam informados, os vetores jacobianos são determinados numericamente através de diferenças finitas.

Para o palpite inicial, optou-se por adotar os limites superiores de h e b_f, permitindo que o algoritmo inicie a busca no ponto viável mais alto da função, o que facilita a obtenção dos resultados, visto que o problema é de minimização e os subproblemas são funções quadráticas.

Para o presente trabalho, as derivadas parciais de todas as equações foram determinadas, a fim de fornecer o valor exato dos vetores jacobianos associados ao algoritmo de otimização, e estão apresentadas no Apêndice B. As derivadas foram verificadas através da confecção de um programa paralelo, que calcula seu valor por diferenças finitas e compara com o valor obtido através das equações calculadas. Essa verificação foi realizada para diversos conjuntos de variáveis, visando ativar todas as fórmulas existentes no dimensionamento de estruturas de aço submetidas à esforços de compressão, além de considerar valores de escalas de grandeza diferentes para o parâmetro delta no método das diferenças finitas.

A ferramenta, assim sendo, aplica o método selecionado (SLSQP) e realiza o processo iterativo, buscando a solução ótima para o problema apresentado. Como resultado, fornece uma gama de dados que podem ser utilizados e analisados posteriormente. Além do vetor de variáveis ótimas, fornece o valor ótimo da função objetivo, o número de iterações realizadas, as mensagens informando sucesso ou falha do processo iterativo, entre outros dados. Essa ferramenta, portanto, facilita o manejo e a análise dos dados, permitindo a criação de uma estratégia para determinação do resultado real do problema.

6 ESTUDO DE CASO

Comumente, o engenheiro de estruturas conta com suas experiências prévias para determinação das dimensões dos perfis adotados em projeto. Posteriormente, são realizadas as verificações de segurança, dando origem ao processo iterativo em busca da menor seção transversal. Porém, é comum que o projetista não realize um grande número de verificações, adotando dimensões que satisfaçam às solicitações de projeto, mesmo que não caracterizem uma solução ótima.

Assim, com o objetivo de fornecer uma visão mais abrangente da aplicação e da importância do presente trabalho, foi realizado um estudo de caso, comparando perfis padronizados apresentados pela Norma NBR 8800 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2008) com soluções obtidas através do uso do programa desenvolvido. Os perfis padronizados são compostos por chapas de dimensões inteiras (sem valores fracionados), aumentando de 50 em 50 mm. Em contrapartida, os perfis otimizados podem assumir dimensões com valores fracionados, o que os tornam mais "flexíveis" ao dimensionamento.

O estudo realizado fixa valores limite para as dimensões (larguras, espessuras e comprimentos de flambagem) e varia os valores da solicitação de compressão. Uma comparação é realizada entre os valores de área bruta da seção transversal fornecidos pela bibliografia e os obtidos pelo algoritmo de otimização. Além disso, será analisada a evolução de cada uma das dimensões da seção transversal, fornecendo uma visão do comportamento e da influência de cada uma na resistência dos perfis.

Em seguida, são gerados novos problemas, modificando restrições arquitetônicas ou comprimentos de flambagem do problema original, fornecendo uma análise da influência de cada um desses fatores no dimensionamento dos perfis.

6.1 PROBLEMA PRINCIPAL

O problema analisado consiste no dimensionamento de uma coluna, submetida a esforços de compressão simples, com comprimentos de flambagem iguais a 500 cm em todas as direções.

Os valores limite para as larguras das chapas são especificados na Tabela 2 e os valores de espessura são os apresentados na Seção 4.3.1.

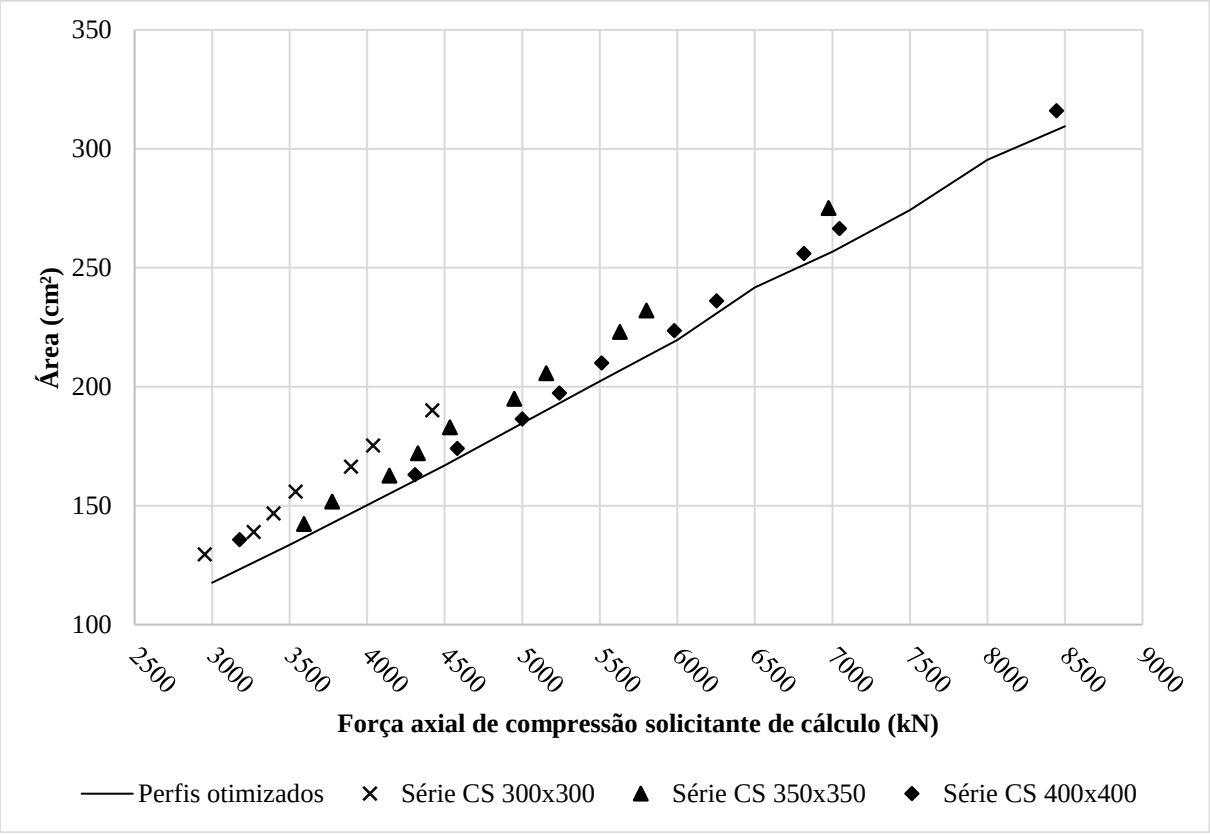
Tabela 2 – Valores limite para as larguras dos perfis.

h_{MIN}	h_{MAX}	$b_{f\,MIN}$	$b_{f\,MAX}$
10 cm	40 cm	10 cm	40 cm

(Fonte: GELAIN, 2018)

Os resultados obtidos nas otimizações, para forças solicitantes entre 3000 kN e 8500 kN (tomadas a cada 500 kN), estão ilustrados na Figura 9.

Figura 9 – Comparativo entre área de seções otimizadas e de perfis padronizados.



(Fonte: GELAIN, 2018)

Na Figura 9 também são apresentados os pontos $A_g x N_{c,Rd}$ para as colunas de seção I padronizadas das séries CS 300x300, CS 350x350 e CS 400x400. Dessa forma, fica claro que a utilização de algoritmos de otimização pode proporcionar uma grande redução no consumo de materiais e nos custos de perfis submetidos a solicitações de compressão simples. A Tabela 3 apresenta uma comparação entre os valores de área obtidos pela ferramenta de otimização e os fornecidos pelas seções padronizadas, assim como a porcentagem de redução da mesma.

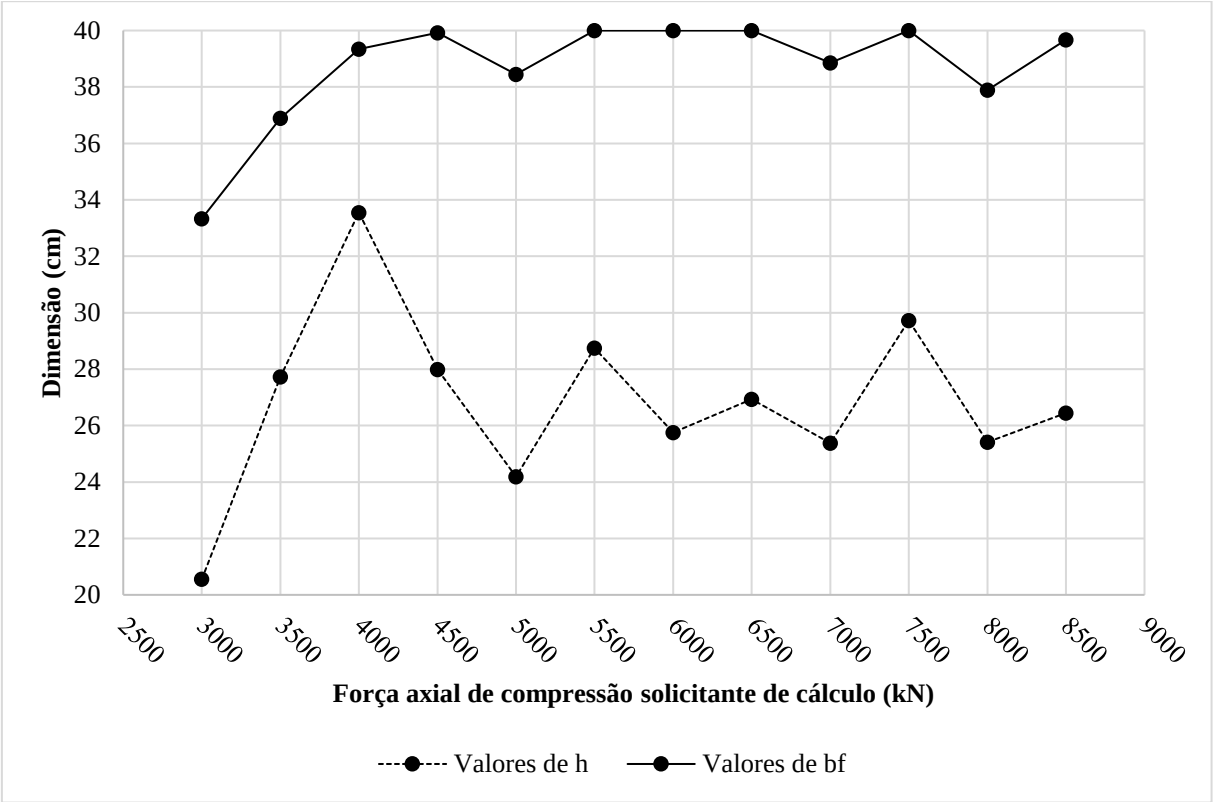
Tabela 3 - Redução da área da seção transversal através da utilização da ferramenta de otimização em relação aos perfis padronizados.

SÉRIE CS 300			**OTIMIZADOS**		**REDUÇÃO**
Perfil	Nc,Rd	Ag	Nc,Rd	Ag	%
300x102	2951.933	129.5	2951.933	116.2849	10.20
300x122	3538.251	155.92	3538.251	134.8022	13.54
300x138	4039.558	175.232	4039.558	151.7935	13.38
300x149	4420.249	190	4420.249	164.2202	13.57
SÉRIE CS 350			**OTIMIZADOS**		**REDUÇÃO**
Perfil	Nc,Rd	Ag	Nc,Rd	Ag	%
350x112	3592.546	142.21	3592.546	136.6259	3.93
350x135	4326.771	172	4326.771	161.0202	6.38
350x161	5156.074	205.632	5156.074	189.0533	8.06
350x216	6977.094	275.03	6977.094	256.1657	6.86
SÉRIE CS 400			**OTIMIZADOS**		**REDUÇÃO**
Perfil	Nc,Rd	Ag	Nc,Rd	Ag	%
400x128	4309.568	162.96	4309.568	160.4313	1.55
400x155	5239.525	197.25	5239.525	191.4005	2.97
400x185	6252.997	236.032	6252.997	230.7404	2.24
400x248	8446.16	316.03	8446.16	307.9671	2.55

A seguir estão ilustradas a evolução das larguras (Figura 10) e das espessuras (Figura 11) das chapas nos perfis otimizados. É possível notar que os perfis otimizados possuem sempre dimensões maiores para as mesas do que para as almas, ilustrando que as mesas são mais influentes na resistência dos perfis.

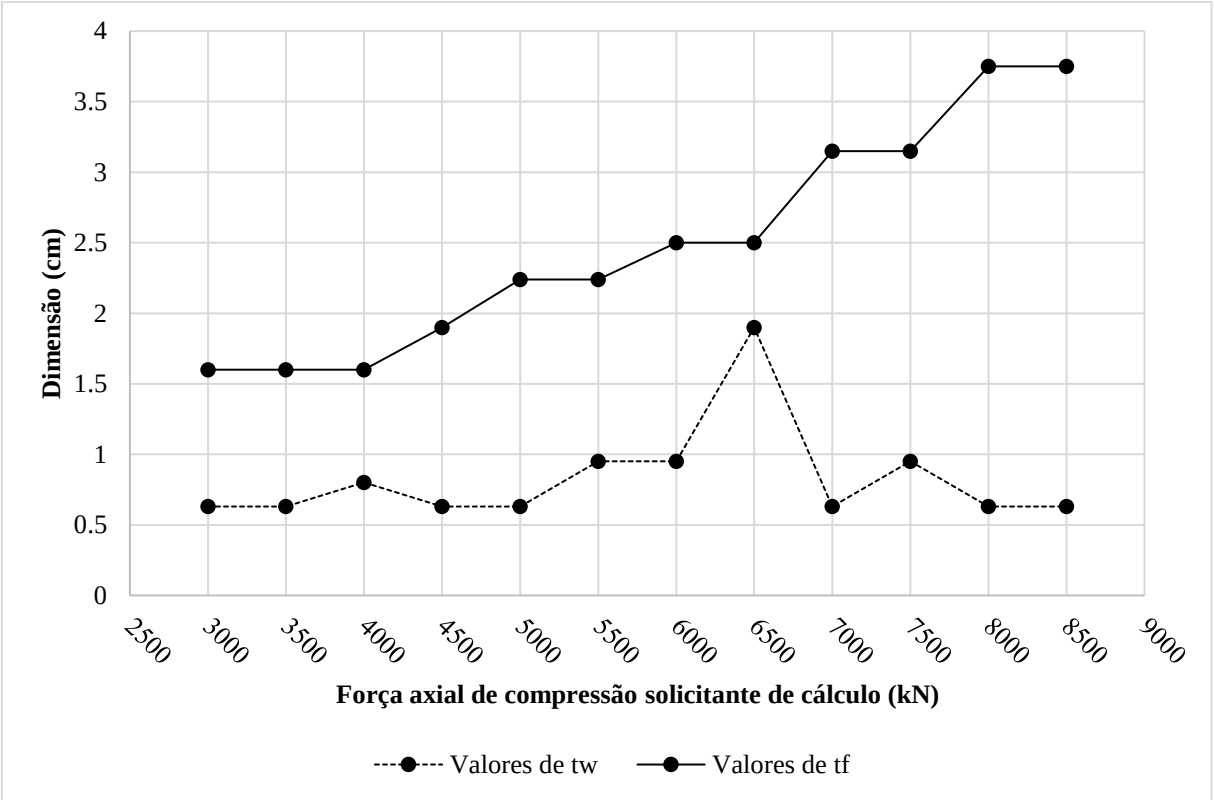
Com pequenas variações nas dimensões h e b_f, o momento de inércia em torno do eixo y cresce mais do que o momento de inércia em torno do eixo x, para valores próximos de N_{ex} e N_{ey}. Logo, o valor máximo para as dimensões das mesas será mais influente no problema do que o valor máximo para as dimensões das almas. A Figura 10 também mostra que o valor máximo de b_f age como uma restrição ativa no dimensionamento dos perfis. Já o valor máximo de h nunca configura uma restrição ativa para as solicitações estudadas. O valor de b_f atinge o máximo com uma solicitação de 4500 kN e, após isso, assume valores menores apenas em situações onde t_f aumenta (conforme visto na Figura 11), retornando para o valor máximo em seguida.

Figura 10 – Evolução dos valores de h e bf para o caso principal.



(Fonte: GELAIN, 2018)

Figura 11 – Evolução dos valores de tw e tf para o caso principal.



(Fonte: GELAIN, 2018)

Como as espessuras estão limitadas a valores discretos, na Figura 11 é possível notar que, para cada aumento de t_f, uma ou mais dimensões são reduzidas. Isso se dá pois o valor da espessura "salta" para um valor muito maior do que o anterior, permitindo que outras dimensões sejam levemente reduzidas.

A Figura 12 mostra os resultados calculados pela ferramenta para as solicitações do caso principal. Juntamente com a solicitação, são exibidas as dimensões ótimas e os valores de A_g e $N_{c,Rd}$, calculados a partir delas. Nota-se que a restrição $g_1(h, b_f) = N_{c,Sd} - N_{c,Rd} < 0$ está ativa para todos as solicitações, visto que o resultado da diferença entre $N_{c,Rd}$ e $N_{c,Sd}$ é muito próximo de zero.

Figura 12 – Resultados obtidos através da ferramenta para o caso principal

```
Ncsd =  3000
h ótima:  20.553058663774948
bf ótima:  33.33224280564204
tw ótima:  0.63
tf ótima:  1.6
Área ótima:  117.59560393623275
Nc,Rd adotado:  3000.0000608082614

Ncsd =  3500
h ótima:  27.730374928202362
bf ótima:  36.894925589047034
tw ótima:  0.63
tf ótima:  1.6
Área ótima:  133.517898089718
Nc,Rd adotado:  3500.0000066644243

Ncsd =  4000
h ótima:  33.550307956442985
bf ótima:  39.34302531954836
tw ótima:  0.8
tf ótima:  1.6
Área ótima:  150.17792738770916
Nc,Rd adotado:  4000.0000000931905

Ncsd =  4500
h ótima:  27.991140136915245
bf ótima:  39.92373935211038
tw ótima:  0.63
tf ótima:  1.9
Área ótima:  166.95062782427607
Nc,Rd adotado:  4500.00001106289

Ncsd =  5000
h ótima:  24.19027963854999
bf ótima:  38.45046760688742
tw ótima:  0.63
tf ótima:  2.24
Área ótima:  184.67557105114216
Nc,Rd adotado:  5000.000004176305

Ncsd =  5500
h ótima:  28.743029562587225
bf ótima:  40.0
tw ótima:  0.95
tf ótima:  2.24
Área ótima:  202.2498780844579
Nc,Rd adotado:  5500.000001090709

Ncsd =  6000
h ótima:  25.754067411741744
bf ótima:  40.0
tw ótima:  0.95
tf ótima:  2.5
Área ótima:  219.71636404115466
Nc,Rd adotado:  6000.000001155712

Ncsd =  6500
h ótima:  26.93324283593821
bf ótima:  40.0
tw ótima:  1.9
tf ótima:  2.5
Área ótima:  241.6731613882826
Nc,Rd adotado:  6500.000001777644

Ncsd =  7000
h ótima:  25.376112423488443
bf ótima:  38.85574221620192
tw ótima:  0.63
tf ótima:  3.15
Área ótima:  256.80912678886983
Nc,Rd adotado:  7000.0000001712115

Ncsd =  7500
h ótima:  29.724542179287873
bf ótima:  40.0
tw ótima:  0.95
tf ótima:  3.15
Área ótima:  274.2533150703235
Nc,Rd adotado:  7500.00000060039

Ncsd =  8000
h ótima:  25.418132437619434
bf ótima:  37.88569222588573
tw ótima:  0.63
tf ótima:  3.75
Área ótima:  295.4311151298432
Nc,Rd adotado:  8000.000000403532

Ncsd =  8500
h ótima:  26.447344056815243
bf ótima:  39.673127267444585
tw ótima:  0.63
tf ótima:  3.75
Área ótima:  309.485281261628
Nc,Rd adotado:  8500.000002032531
```

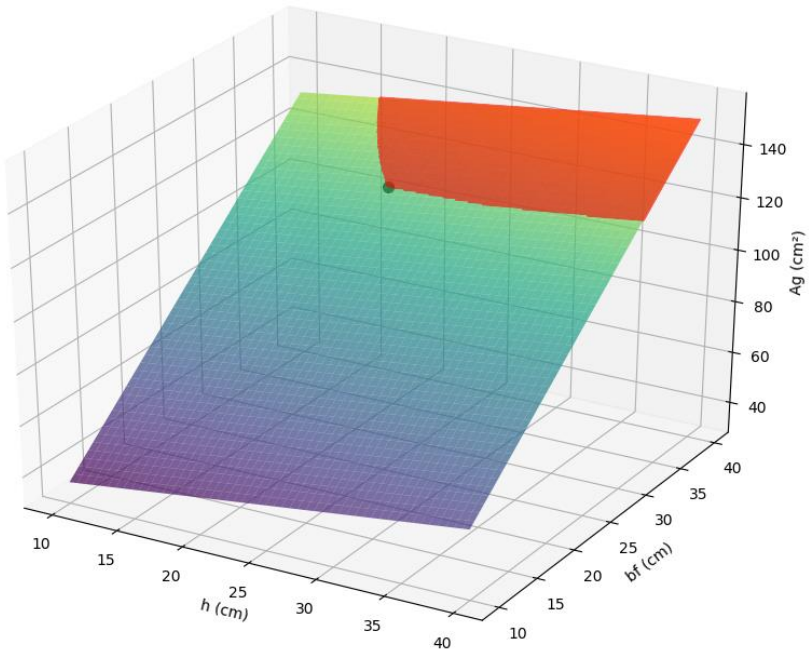
(Fonte: GELAIN, 2018)

Na Figura 13 é possível observar, em vermelho, como a restrição g_1 se comporta quando projetada sobre a função objetivo. Ou seja, os valores de $N_{c,Rd}$ calculados para qualquer ponto $(h,\ b_f)$, contido na região vermelha, serão maiores do que 3000 kN, satisfazendo a restrição. As demais restrições (g_2 e g_3) abrangem praticamente toda área da função objetivo para todos os valores de solicitação considerados, de modo que nunca representam restrições ativas para o problema analisado e, portanto, não foram representadas. O valor ótimo da função objetivo está representado pelo ponto preto no gráfico e, dada configuração da região viável, é possível afirmar que se trata de um mínimo global, e não local.

O mesmo pode ser observado na Figura 14 e na Figura 15, para os valores de solicitação de 5000 kN e 8000 kN, respectivamente. É importante ressaltar que, para a construção destes gráficos, foi necessário fixar os valores de t_w e t_f, dados conforme Figura 12. Desta forma, a Figura 15 apresenta uma região viável maior, mesmo que g_1 esteja restringindo mais a função objetivo. Isso ocorre pois t_f é maior para 8000 kN do que para 5000 kN, fazendo também com que os valores de A_g aumentem. Mesmo assim, os valores ótimos de área aumentam conforme aumentam as solicitações, como se espera que ocorra.

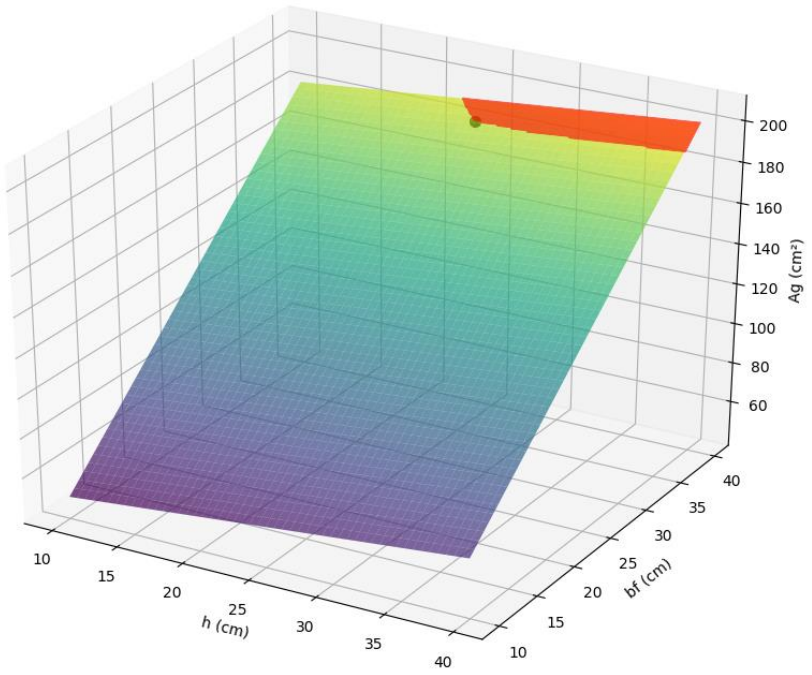
Caso fossem utilizados os mesmos valores de espessura para todos os gráficos, os valores apresentados não seriam mínimos globais.

Figura 13 - Região viável da função objetivo para solicitação de 3000 kN, e seu ponto de mínimo.



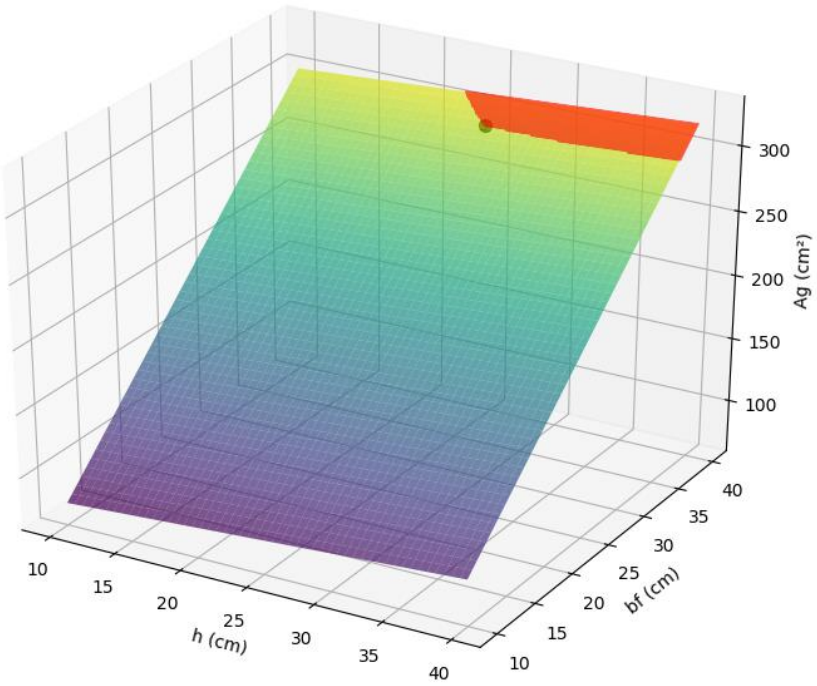
(Fonte: GELAIN, 2018)

Figura 14 - Região viável da função objetivo para solicitação de 5000 kN, e seu ponto de mínimo.



(Fonte: GELAIN, 2018)

Figura 15 - Região viável da função objetivo para solicitação de 8000 kN, e seu ponto de mínimo.



(Fonte: GELAIN, 2018)

6.2 ANÁLISE DE INFLUÊNCIA DAS RESTRIÇÕES

Para ilustrar como as restrições arquitetônicas influenciam no dimensionamento de perfis otimizados, serão analisados dois casos secundários, derivados do caso principal. Em cada caso, uma dimensão terá seus valores limite modificados, enquanto as demais dimensões serão mantidas.

O problema principal será chamado de “Caso 1” e os problemas secundários serão nomeados sequencialmente a partir desse. A fim de facilitar a visualização e a comparação dos resultados, os casos serão agrupados por afinidade e exibidos juntamente com o problema principal.

Os casos secundários estudados estão listados a seguir:

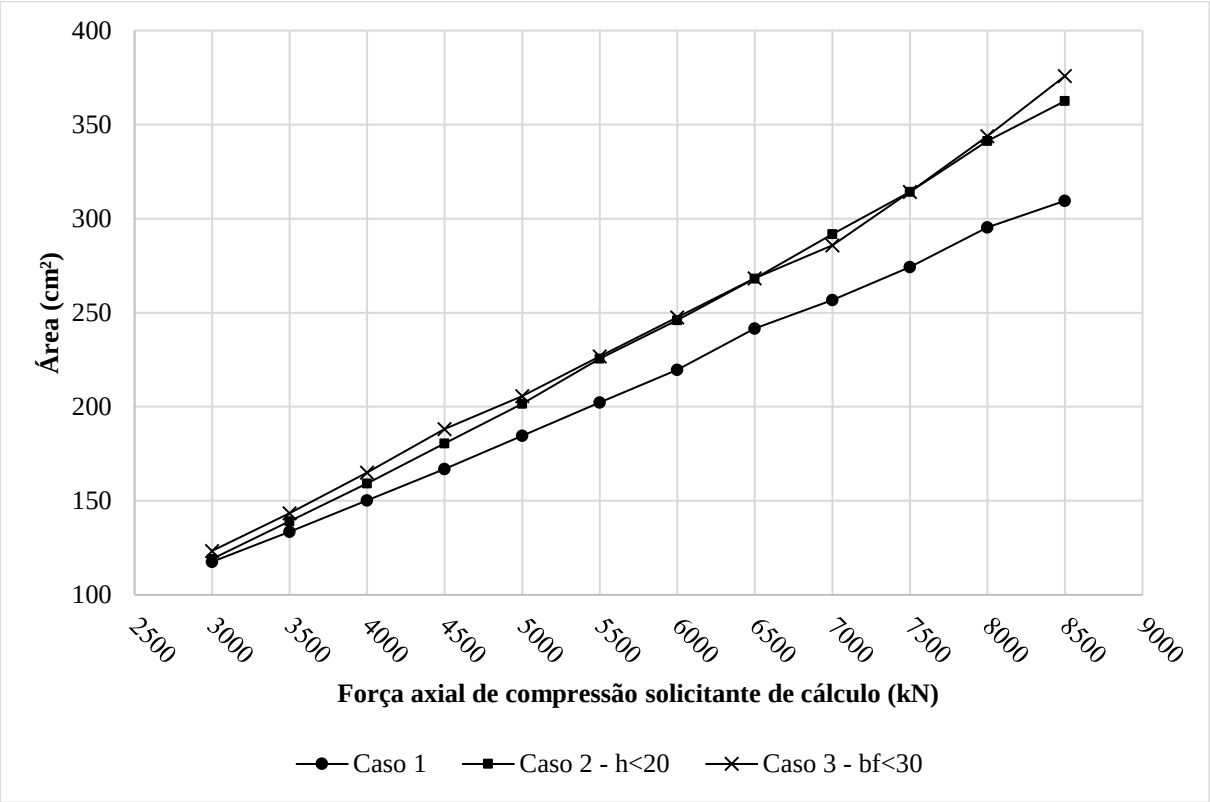
a) Caso 2: $h_{MAX} = 200mm$:

b) Caso 3: $b_{f_{MAX}} = 300mm$:

Inicialmente, para o Caso 3, considerou-se restringir o valor de $b_{f_{MAX}}$ em até 200 mm, porém o algoritmo encontrou soluções apenas até solicitações de 3500 kN. Portanto, foi adotado o valor máximo de 300 mm. Esse fato corrobora o que foi dito anteriormente sobre a relevância das dimensões das mesas no dimensionamento de perfis otimizados.

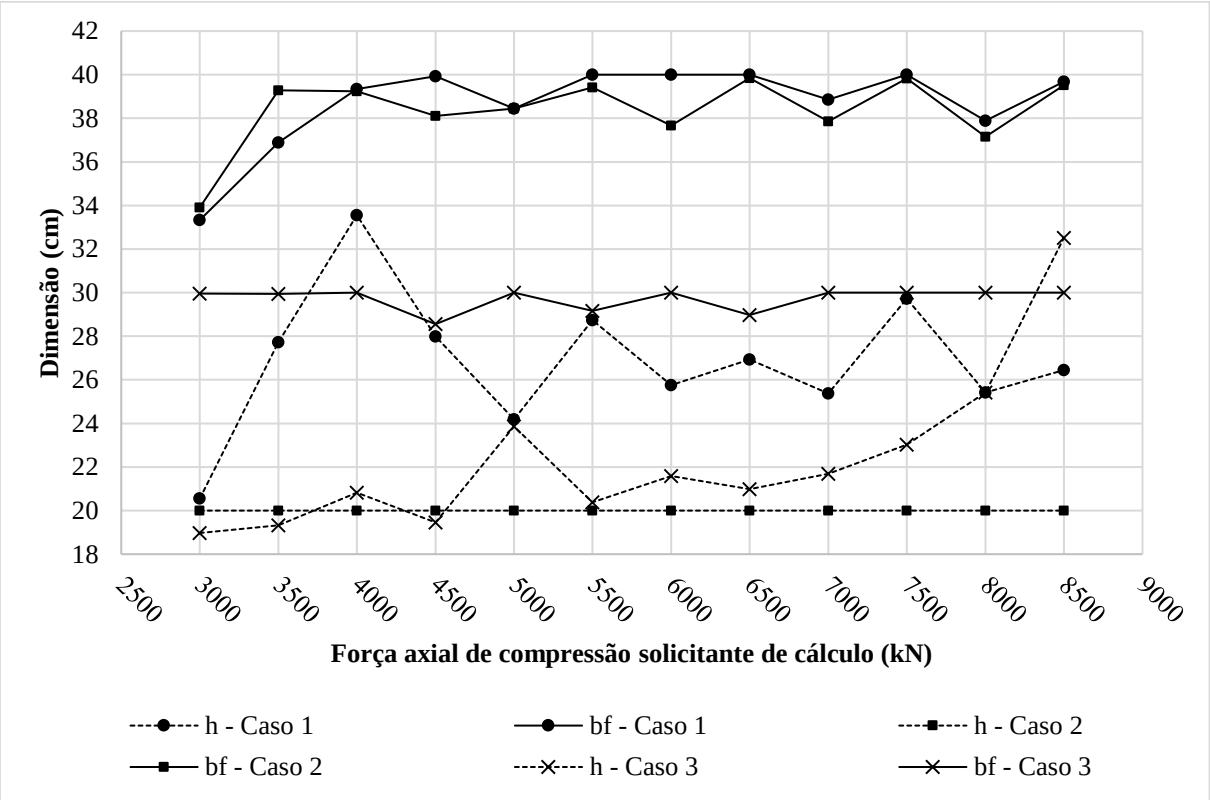
A Figura 16 mostra que os casos restringidos apresentam uma curva de crescimento da seção transversal com inclinação maior do que o Caso 1. Quanto maior for a solicitação, maior será a diferença entre os valores do Caso 1 e os valores dos Casos 2 e 3. Também é possível notar que os valores de área para os Casos 2 e 3 crescem praticamente juntos até o valor de 8000 kN. A partir desse valor, o Caso 3 passa a assumir uma inclinação maior em sua curva de crescimento. As causas disso podem ser observadas na Figura 17 e na Figura 18.

Figura 16 – Comparação entre a área da seção transversal dos casos 1, 2 e 3.



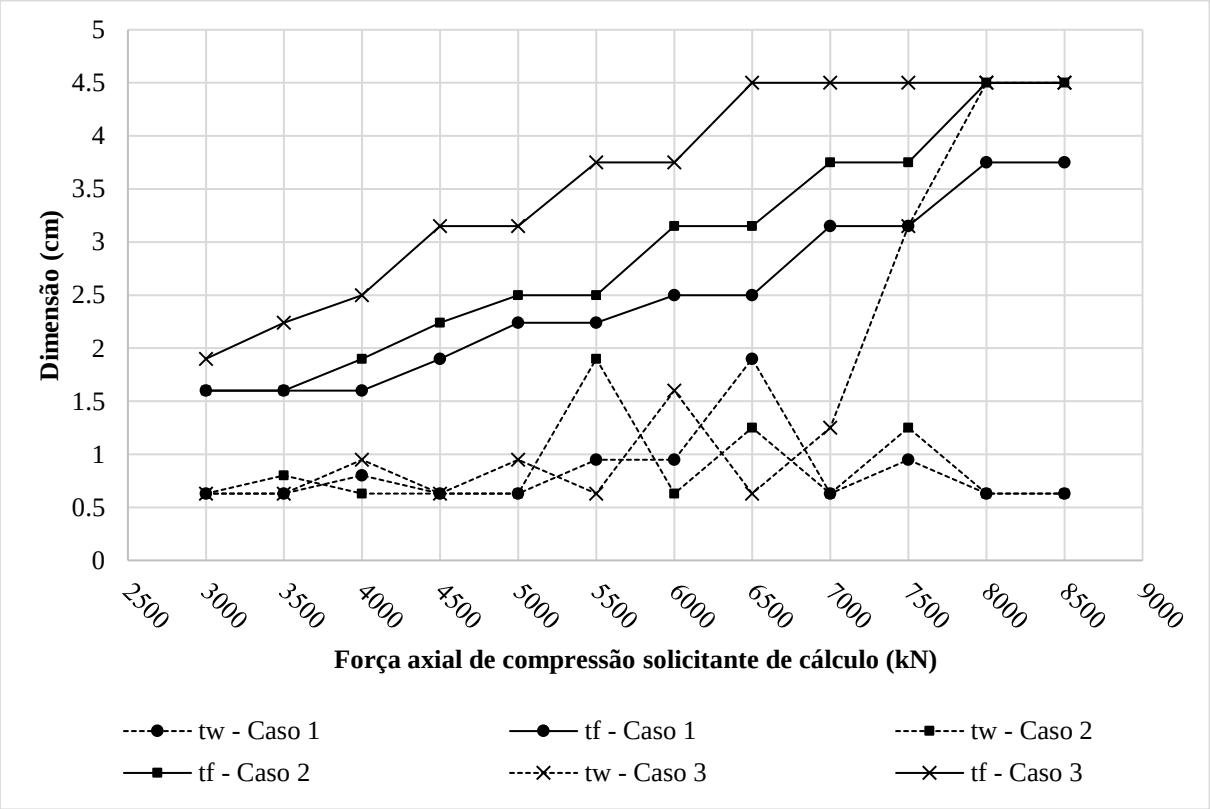
(Fonte: GELAIN, 2018)

Figura 17 – Comparação entre as larguras h e bf dos casos 1, 2 e 3.



(Fonte: GELAIN, 2018)

Figura 18 – Comparação entre as espessuras tw e tf dos casos 1, 2 e 3.



(Fonte: GELAIN, 2018)

Primeiramente, é possível notar que alguns fatores observados no Caso 1 se repetem para os casos restringidos. A largura das mesas adota valores próximos ao seu limite máximo, caracterizando uma restrição ativa, e os valores de espessura das mesas sempre aumentam, causando reduções nas outras dimensões quando isso ocorre. Para os casos restringidos, também fica claro que os limites adotados funcionam como restrições ativas desde os valores menores de solicitação, "forçando" o aumento das outras dimensões.

No Caso 3, observa-se que a espessura das mesas atinge o valor máximo disponível (4.5 cm) em solicitações de 6500 kN. A partir desse ponto, os valores de t_w aumentam drasticamente, pois sua influência na resistência do perfil é menor, até que também atinjam o valor máximo, em 8000 kN. Esse é o ponto a partir do qual a curva de crescimento da área do Caso 3 distancia-se do Caso 2, conforme observado anteriormente. Nesse ponto, a única dimensão que ainda pode ser aumentada é h, pois quase todas as restrições de projeto estão ativas. Isso significa que estamos próximos de atingir a capacidade máxima de resistência para perfis com essas restrições.

Como forma de ilustrar esse fato, a resistência à compressão de um perfil I, com h igual a 40 cm, b_f igual a 30 cm, t_w e t_f iguais a 4,5 cm e os comprimentos de flambagem iguais a 500 cm, é igual a 8992.023 kN. Ou seja, não seria possível encontrar uma solução viável para o Caso 3, caso analisássemos solicitações até 9000 kN.

Tabela 4 – Dimensões das seções otimizadas para solicitações de 8992.023 kN, nos casos 1, 2 e 3.

Caso	A_g(cm²)	h (cm)	b_f (cm)	t_w (cm)	t_f (cm)	Restrições ativas
1	329.227	30.88	40	1.25	3.75	1
2	390.95	20	39.59	3.15	4.5	2
3	409.5	40	30	4.5	4.5	4

(Fonte: GELAIN, 2018)

A Tabela 4 fornece as dimensões das seções otimizadas para o esforço de 8992.023 kN em cada caso estudado. Com isso, podemos observar que, quanto maior o número de restrições ativas no dimensionamento de uma estrutura otimizada, maior será a inclinação da curva de crescimento da área, até que todas as restrições estejam ativas e o valor máximo de resistência seja atingido, como no Caso 3.

7 CONSIDERAÇÕES FINAIS

Em relação à ferramenta, considera-se que os resultados obtidos foram satisfatórios, cumprindo o objetivo principal do trabalho e fornecendo os dados necessários para o dimensionamento de estruturas otimizadas. Porém, pela ausência de uma interface ou de um arquivo executável, a ferramenta torna-se de difícil utilização, exigindo que a máquina, na qual seria realizado o processamento, possua previamente instalados os programas e bibliotecas necessárias para sua execução. Como forma de avançar os trabalhos, sugere-se o desenvolvimento de uma interface simples, capaz de coletar os dados do problema e de fornecer resultados. Tais resultados podem ser simplificados ou mais detalhados, fornecendo informações sobre o número de iterações ou resultados parciais para determinados valores de espessura das chapas, conforme interesse do projetista.

Com o intuito de simplificar o trabalho, optou-se por restringir os esforços solicitantes à compressão simples, fazendo com que a ferramenta possua aplicabilidade limitada em problemas reais, visto que raramente uma estrutura estará sujeita apenas a esforços de compressão simples. Porém, com a ferramenta já desenvolvida, torna-se viável a inserção de novas restrições de segurança, permitindo a análise de estruturas sujeitas a esforços combinados e aumentando sua aplicabilidade. Para que isso ocorra, devem ser descritas as funções relevantes ao dimensionamento e suas derivadas devem ser determinadas em relação às variáveis de projeto. Posteriormente, ainda é possível a inserção de diferentes tipos de seção transversal, abrangendo ainda mais casos reais de projeto.

O desenvolvimento do presente trabalho de conclusão de curso envolveu a retomada de conhecimentos prévios do curso de graduação, alinhando-os a novos conhecimentos e habilidades. Nesse caso, foi realizada uma revisão dos critérios de dimensionamento de estruturas metálicas e adquiridas noções e capacidades sobre programação e problemas matemáticos de otimização. Tais ferramentas podem ser úteis durante a vida profissional de um engenheiro civil projetista estrutural.

Ficou claro que o método SQP se mostrou uma ferramenta muito mais robusta do que o necessário para resolução deste problema, visto que o mesmo é designado a resolver problemas com funções muito mais complexas. Funções lineares, como a que descreve a área de uma seção transversal em I, podem ser otimizadas a partir de métodos muito mais simples. Porém a linguagem de programação escolhida impôs uma limitação em relação às bibliotecas existentes

e capazes de solucionar o problema abordado. Isso posto, mesmo que a ferramenta seja mais robusta do que o necessário, a mesma se mostrou eficiente e apresentou resultados satisfatórios.

Por fim, o estudo de caso realizado forneceu uma visão sobre como uma rotina de otimização pode influenciar nos custos finais de um projeto, fornecendo áreas de seção transversal consideravelmente menores se comparadas a seções padronizadas. As reduções variaram entre 1% e 13,4%, sendo a média em torno dos 4%. Reduções desta escala, se levadas a todos os elementos estruturais de uma edificação, podem resultar em uma diminuição significativa dos custos da obra. Além disso ainda pode ser considerada a redução do tempo de trabalho do engenheiro projetista, reduzindo ainda mais os custos.

Além disso, foi possível visualizar a influência de cada uma das dimensões de um perfil I soldado em sua resistência à compressão, além da influência causada pela imposição de restrições arquitetônicas no dimensionamento. Proporcionando, assim, experiência e conhecimento para confecção de projetos futuros.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 8.800: projeto de estruturas de aço e de estruturas mistas de aço e concreto de edifícios. Rio de Janeiro, 2008.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 7.007: Aço-carbono e aço microligado para barras e perfis laminados a quente para uso estrutural — Requisitos. Rio de Janeiro, 2016.

CHRISTENSEN, P. W.; KLARBRING, A. An Introduction to Structural Optimization. 1 ed. Suécia: Springer, 2009.

HAFTKA, R. T.; GÜRDAL, Z. Elements of Structural Optimization. 3rd rev. exp. ed. Dordrecht: Kluwer Academic, 1992.

KRAFT, D. A Software Package for Sequential Quadratic Programming. DLR German Aerospace Center – Institute for Flight Mechanics. Oberpfaffenhofen, Alemanha. 1988.

PFEIL, W.; PFEIL, M. Estruturas de Aço: Dimensionamento Prático de Acordo com a NBR 8800:2008. 8. ed. Rio de Janeiro: LTC, 2009 (reimpr. 2014).

PINHEIRO, A. C. da F. B. Estruturas metálicas: cálculos, detalhes, exercícios e projetos. 2. ed. (3. reimpr.) São Paulo: Blucher, 2005 (reimpr. 2010).

SCHITTKOWSKI, K. The Nonlinear Programming Method of Wilson, Han and Powell with na Augmented Lagrangian Type Line Search Function. Part 2: An Efficient Implementantion with Linear Least Squares Subproblems. Institut für Angewandte Mathematik und Statistik, Universität Würzburg. Würzburg, Alemanha. 1981.

The SciPy Community. Sem título, Maio de 2018. Disponível em: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>. Acesso em 10 de junho de 2018.

APÊNDICE A – ROTINA COMPUTACIONAL PARA OTIMIZAÇÃO DE PERFIS DE AÇO SOLDADOS, DE SEÇÃO I, SUBMETIDOS A ESFORÇOS DE COMPRESSÃO SIMPLES.

```
"""
FERRAMENTA DE OTIMIZAÇÃO DE PERFIS SOLDADOS DE SEÇÃO I, SUJEITOS À ESFORÇOS DE COMPRESSÃO AXIAL SIMPLES

Autor: Germano Henrique Gelain

"""

import numpy as np
import math
import scipy.optimize as opt

"""
================================================================================
                                DEFINIÇÃO DAS VARIÁVEIS
================================================================================
"""

tw_lista = [0.63, 0.8, 0.95, 1.25, 1.6, 1.9, 2.24, 2.5, 3.15, 3.75, 4.5] #cm
tf_lista = [0.63, 0.8, 0.95, 1.25, 1.6, 1.9, 2.24, 2.5, 3.15, 3.75, 4.5] #cm
kx = float(input('Digite o valor de kx: '))
lx = float(input('Digite o valor de lx, em cm: '))
ky = float(input('Digite o valor de ky: '))
ly = float(input('Digite o valor de ly, em cm: '))
kz = float(input('Digite o valor de kz: '))
lz = float(input('Digite o valor de lz, em cm: '))
Ncsd = float(input('Digite o valor de Nc,sd, em kN: '))
h_inf = float(input('Digite o limite inferior para h, em cm: '))
h_sup = float(input('Digite o limite superior para h, em cm: '))
bf_inf = float(input('Digite o limite inferior para bf, em cm: '))
bf_sup = float(input('Digite o limite superior para bf, em cm: '))
E = 20000 #kN/cm²
fy = 35 #kN/cm²
G = E/(2*(1.3)) #MPa
Ag_otim = 2000000

"""
================================================================================
                                DEFINIÇÃO DAS FUNÇÕES
================================================================================
"""

def bw(x):
    return x[0] - 2*tf
def Ag(x):
    return 2*x[1]*tf + tw*(bw(x))
def Ix(x):
    a1 = (tw*(bw(x)**3))/12
    b1 = (x[1]*(tf**3))/12
    b2 = x[1]*tf*(((x[0]-tf)/2)**2)
    return a1 + 2*(b1 + b2)
def Iy(x):
    a1 = (bw(x)*(tw**3))/12
    b1 = (tf*(x[1]**3))/12
    return a1 + 2*b1
def Wx(x):
    return Ix(x)/(x[0]/2)
def Zx(x):
    return ((x[0]**2)*x[1])/4 - ((bw(x)**2)*(x[1]-tw))/4
def rx(x):
    return (Ix(x)/Ag(x))**(1/2)
def ry(x):
    return (Iy(x)/Ag(x))**(1/2)
def r0(x):
    return ((rx(x)**2)+(ry(x)**2))**(1/2)
```

```
def Cw(x):
    return (Iy(x)*((x[0]-tf)**2))/4
def J(x):
    return ((2*x[1]*(tf**3))+((tw**3)*(x[0]-tf)))/3
def Nex(x):
    return ((math.pi**2)*E*Ix(x))/((kx*lx)**2)
def Ney(x):
    return ((math.pi**2)*E*Iy(x))/((ky*ly)**2)
def Nez(x):
    return (1/(r0(x)**2))*((((math.pi**2)*E*Cw(x))/((kz*lz)**2))+(G*J(x)))
def Ne(x):
    if Nex(x) < Ney(x) and Nex(x) < Nez(x):
        return Nex(x)
    elif Ney(x) < Nex(x) and Ney(x) < Nez(x):
        return Ney(x)
    else:
        return Nez(x)
def kc(x):
    a1 = 4/(np.sqrt(bw(x)/tw))
    if a1 < 0.35:
        return 0.35
    elif a1 > 0.76:
        return 0.76
    else:
        return a1
def Qs(x):
    a1 = (x[1]/2)/tf
    a2 = 0.64*(np.sqrt(E/(fy/kc(x))))
    a3 = 1.17*(np.sqrt(E/(fy/kc(x))))
    b1 = 1.415 - 0.65*((x[1]/2)/tf)*(np.sqrt(fy/(E*kc(x))))
    b2 = (0.90*E*kc(x))/(fy*((((x[1]/2)/tf)**2)))
    if a1 <= a2:
        return 1
    elif a1 > a3:
        return b2
           else:
               return b1
       def lambda_sigma(x):
           return np.sqrt(Ag(x)*fy/Ne(x))
       def qui_sigma(x):
           if lambda_sigma(x) <= 1.5:
               return 0.658**(lambda_sigma(x)**2)
           else:
               return 0.877/(lambda_sigma(x)**2)
       def sigma(x):
           return qui_sigma(x)*fy
       def bef(x):
           a1 = 1.92*tw*np.sqrt(E/sigma(x))
           a2 = (0.34/(bw(x)/tw))*np.sqrt(E/sigma(x))
           if a1*(1 - a2) <= bw(x):
               return a1*(1 - a2)
           else:
               return bw(x)
       def Aef(x):
           return Ag(x) - tw*(bw(x) - bef(x))
       def Qa(x):
           a1 = bw(x)/tw
           a2 = 1.49*np.sqrt(E/fy)
           if a1 <= a2:
               return 1
           else:
               return Aef(x)/Ag(x)
       def Q(x):
           return Qs(x)*Qa(x)
       def lambda0(x):
```

```
130.            return np.sqrt(Q(x)*Ag(x)*fy/Ne(x))
131.        def qui(x):
132.            if lambda0(x) <= 1.5:
133.                return 0.658**(lambda0(x)**2)
134.            else:
135.                return 0.877/(lambda0(x)**2)
136.        def Ncrd(x):
137.            return (qui(x)*Q(x)*Ag(x)*fy)/1.1
138.
139.        """
140.        ================================================================================
141.                                    DEFINIÇÃO DAS DERIVADAS
142.        ================================================================================
143.        """
144.
145.        def dbw_dh(x):
146.            return 1
147.        def dbw_dbf(x):
148.            return 0
149.        def dIx_dh(x):
150.            a1 = (tw*(bw(x)**2))/4
151.            a2 = x[1]*tf*(x[0] - tf)
152.            return a1 + a2
153.        def dIx_dbf(x):
154.            a1 = (tf**3)/12
155.            a2 = tf*(((x[0] - tf)/2)**2)
156.            return 2*(a1 + a2)
157.        def dIy_dh(x):
158.            return (tw**3)/12
159.        def dIy_dbf(x):
160.            return (tf/2)*(x[1]**2)
161.        def dAg_dh(x):
162.            return tw
163.        def dAg_dbf(x):
164.            return 2*tf
165.        def drx_dh(x):
166.            a1 = dIx_dh(x)*Ag(x)
167.            a2 = dAg_dh(x)*Ix(x)
168.            a3 = 2*(Ag(x)**2)*rx(x)
169.            return (a1 -a2)/a3
170.        def drx_dbf(x):
171.            a1 = dIx_dbf(x)*Ag(x)
172.            a2 = dAg_dbf(x)*Ix(x)
173.            a3 = 2*(Ag(x)**2)*rx(x)
174.            return (a1 -a2)/a3
175.        def dry_dh(x):
176.            a1 = dIy_dh(x)*Ag(x)
177.            a2 = dAg_dh(x)*Iy(x)
178.            a3 = 2*(Ag(x)**2)*ry(x)
179.            return (a1 -a2)/a3
180.        def dry_dbf(x):
181.            a1 = dIy_dbf(x)*Ag(x)
182.            a2 = dAg_dbf(x)*Iy(x)
183.            a3 = 2*(Ag(x)**2)*ry(x)
184.            return (a1 -a2)/a3
185.        def dr0_dh(x):
186.            a1 = drx_dh(x)*rx(x)
187.            a2 = dry_dh(x)*ry(x)
188.            return (a1 + a2)/r0(x)
189.        def dr0_dbf(x):
190.            a1 = drx_dbf(x)*rx(x)
191.            a2 = dry_dbf(x)*ry(x)
192.            return (a1 + a2)/r0(x)
193.        def dCw_dh(x):
194.            a1 = dIy_dh(x)*((x[0] - tf)**2)
```

```
195.            a2 = 2*Iy(x)*(x[0] - tf)
196.            return (1/4)*(a1 + a2)
197.        def dCw_dbf(x):
198.            a1 = dIy_dbf(x)*((x[0] - tf)**2)
199.            return (1/4)*a1
200.        def dJ_dh(x):
201.            return (tw**3)/3
202.        def dJ_dbf(x):
203.            return (2/3)*(tf**3)
204.        def dNex_dh(x):
205.            a1 = (3.141592**2)*E
206.            a2 = (kx*lx)**2
207.            return (a1/a2)*dIx_dh(x)
208.        def dNex_dbf(x):
209.            a1 = (3.141592**2)*E
210.            a2 = (kx*lx)**2
211.            return (a1/a2)*dIx_dbf(x)
212.        def dNey_dh(x):
213.            a1 = (3.141592**2)*E
214.            a2 = (ky*ly)**2
215.            return (a1/a2)*dIy_dh(x)
216.        def dNey_dbf(x):
217.            a1 = (3.141592**2)*E
218.            a2 = (ky*ly)**2
219.            return (a1/a2)*dIy_dbf(x)
220.        def dNez_dh(x):
221.            a1 = (3.141592**2)*E
222.            a2 = (kz*lz)**2
223.            b1 = (a1/a2)*dCw_dh(x) + G*dJ_dh(x)
224.            b2 = 1/(r0(x)**2)
225.            b3 = (2*dr0_dh(x))/(r0(x)**3)
226.            b4 = (a1/a2)*Cw(x) + G*J(x)
227.            return b1*b2 - b3*b4
228.        def dNez_dbf(x):
229.            a1 = (3.141592**2)*E
230.            a2 = (kz*lz)**2
231.            b1 = (a1/a2)*dCw_dbf(x) + G*dJ_dbf(x)
232.            b2 = 1/(r0(x)**2)
233.            b3 = (2*dr0_dbf(x))/(r0(x)**3)
234.            b4 = (a1/a2)*Cw(x) + G*J(x)
235.            return b1*b2 - b3*b4
236.        def dNe_dh(x):
237.            Nex = ((math.pi**2)*E*Ix(x))/((kx*lx)**2)
238.            Ney = ((math.pi**2)*E*Iy(x))/((ky*ly)**2)
239.            Nez = (1/(r0(x)**2))*(((math.pi**2)*E*Cw(x))/((kz*lz)**2)+(G*J(x)))
240.            if Nex < Ney and Nex < Nez:
241.                return dNex_dh(x)
242.            elif Ney < Nex and Ney < Nez:
243.                return dNey_dh(x)
244.            else:
245.                return dNez_dh(x)
246.        def dNe_dbf(x):
247.            Nex = ((math.pi**2)*E*Ix(x))/((kx*lx)**2)
248.            Ney = ((math.pi**2)*E*Iy(x))/((ky*ly)**2)
249.            Nez = (1/(r0(x)**2))*(((math.pi**2)*E*Cw(x))/((kz*lz)**2)+(G*J(x)))
250.            if Nex < Ney and Nex < Nez:
251.                return dNex_dbf(x)
252.            elif Ney < Nex and Ney < Nez:
253.                return dNey_dbf(x)
254.            else:
255.                return dNez_dbf(x)
256.        def dlambda_sigma_dh(x):
257.            a1 = fy*((dAg_dh(x)*Ne(x))-(Ag(x)*dNe_dh(x)))
258.            a2 = 2*(Ne(x)**2)
259.            a3 = ((fy*Ag(x))/Ne(x))**(1/2)
```

```
260.            return a1/(a2*a3)
261.        def dlambda_sigma_dbf(x):
262.            a1 = fy*((dAg_dbf(x)*Ne(x))-(Ag(x)*dNe_dbf(x)))
263.            a2 = 2*(Ne(x)**2)
264.            a3 = ((fy*Ag(x))/Ne(x))**(1/2)
265.            return a1/(a2*a3)
266.        def dqui_sigma_dh(x):
267.            a1 =  0.658**(lambda_sigma(x)**2)
268.            a2 = 2*lambda_sigma(x)*dlambda_sigma_dh(x)
269.            a3 = -1.754/(lambda_sigma(x)**3)
270.            if lambda_sigma(x) <= 1.5:
271.                return a1*a2*(np.log(0.658))
272.            else:
273.                return a3*dlambda_sigma_dh(x)
274.        def dqui_sigma_dbf(x):
275.            a1 =  0.658**(lambda_sigma(x)**2)
276.            a2 = 2*lambda_sigma(x)*dlambda_sigma_dbf(x)
277.            a3 = -1.754/(lambda_sigma(x)**3)
278.            if lambda_sigma(x) <= 1.5:
279.                return a1*a2*(np.log(0.658))
280.            else:
281.                return a3*dlambda_sigma_dbf(x)
282.        def dsigma_dh(x):
283.            return fy*dqui_sigma_dh(x)
284.        def dsigma_dbf(x):
285.            return fy*dqui_sigma_dbf(x)
286.        def dbef_dh(x):
287.            a1 = 0.6528*(tw**2)*E
288.            a2 = (1/((bw(x)**2)*sigma(x)))
289.            a3 = ((1/(bw(x)*(sigma(x)**2)))*dsigma_dh(x))
290.            b1 = a1*(a2 + a3)
291.            a4 = 1.92*tw*(E**(1/2))
292.            a5 = 2*(sigma(x)**(3/2))
293.            b2 = (a4/a5)*dsigma_dh(x)
294.            c1 = 1.92*tw*np.sqrt(E/sigma(x))
295.            c2 = (0.34/(bw(x)/tw))*np.sqrt(E/sigma(x))
296.            if c1*(1 - c2) <= bw(x):
297.                return b1 - b2
298.            else:
299.                return 1
300.        def dbef_dbf(x):
301.            a1 = 0.6528*(tw**2)*E
302.            a2 = (bw(x)*(sigma(x)**2))
303.            b1 = a1/a2
304.            a3 = 1.92*tw*(E**(1/2))
305.            a4 = (2*(sigma(x)**(3/2)))
306.            b2 = a3/a4
307.            c1 = 1.92*tw*np.sqrt(E/sigma(x))
308.            c2 = (0.34/(bw(x)/tw))*np.sqrt(E/sigma(x))
309.            if c1*(1 - c2) <= bw(x):
310.                return dsigma_dbf(x)*(b1 - b2)
311.            else:
312.                return 0
313.        def dAef_dh(x):
314.            return dAg_dh(x) - tw*(1 - dbef_dh(x))
315.        def dAef_dbf(x):
316.            return dAg_dbf(x) + tw*dbef_dbf(x)
317.        def dQa_dh(x):
318.            a1 = dAef_dh(x)*Ag(x)
319.            a2 = Aef(x)*dAg_dh(x)
320.            a3 = Ag(x)**2
321.            a4 = bw(x)/tw
322.            a5 = 1.49*np.sqrt(E/fy)
323.            if a4 <= a5:
324.                return 0
```

```
325.             else:
326.                 return (a1 - a2)/a3
327.         def dQa_dbf(x):
328.             a1 = dAef_dbf(x)*Ag(x)
329.             a2 = Aef(x)*dAg_dbf(x)
330.             a3 = Ag(x)**2
331.             a4 = bw(x)/tw
332.             a5 = 1.49*np.sqrt(E/fy)
333.             if a4 <= a5:
334.                 return 0
335.             else:
336.                 return (a1 - a2)/a3
337.         def dkc_dh(x):
338.                 a1 = 4/(np.sqrt(bw(x)/tw))
339.                 if a1 < 0.35:
340.                     return 0
341.                 elif a1 > 0.76:
342.                     return 0
343.                 else:
344.                     return -2*((tw/(bw(x)**3))**(1/2))
345.         def dQs1_dh(x):
346.             a1 = 0.65/(4*tf)
347.             a2 = (fy/((kc(x)**3)*E))**(1/2)
348.             return a1*a2*dkc_dh(x)
349.         def dQs1_dbf(x):
350.             a1 = -0.65/(2*tf)
351.             a2 = (fy/(kc(x)*E))**(1/2)
352.             return a1*a2
353.         def dQs2_dh(x):
354.             return ((0.90*E)/(fy*((((x[1]/2)/tf)**2))))*dkc_dh(x)
355.         def dQs2_dbf(x):
356.             a1 = -(0.9*E*8*kc(x)*(tf**2))
357.             a2 = fy*(x[1]**3)
358.             return a1/a2
359.         def dQs_dh(x):
360.             a1 = (x[1]/2)/tf
361.             a2 = 0.64*(np.sqrt(E/(fy/kc(x))))
362.             a3 = 1.17*(np.sqrt(E/(fy/kc(x))))
363.             if a1 <= a2:
364.                 return 0
365.             elif a1 > a3:
366.                 return dQs2_dh(x)
367.             else:
368.                 return dQs1_dh(x)
369.         def dQs_dbf(x):
370.             a1 = (x[1]/2)/tf
371.             a2 = 0.64*(np.sqrt(E/(fy/kc(x))))
372.             a3 = 1.17*(np.sqrt(E/(fy/kc(x))))
373.             if a1 <= a2:
374.                 return 0
375.             elif a1 > a3:
376.                 return dQs2_dbf(x)
377.             else:
378.                 return dQs1_dbf(x)
379.         def dQ_dh(x):
380.             a1 = dQa_dh(x)*Qs(x)
381.             a2 = Qa(x)*dQs_dh(x)
382.             return a1 + a2
383.         def dQ_dbf(x):
384.             a1 = dQa_dbf(x)*Qs(x)
385.             a2 = Qa(x)*dQs_dbf(x)
386.             return a1 + a2
387.         def dlambda0_dh(x):
388.             a1 = dQ_dh(x)*Ag(x)*Ne(x)
389.             a2 = Q(x)*dAg_dh(x)*Ne(x)
```

```
            a3 = Q(x)*Ag(x)*dNe_dh(x)
            a4 = (fy**(1/2))/(2*(Ne(x)**2))
            a5 = ((Q(x)*Ag(x))/Ne(x))**(1/2)
            return (a4*(a1 + a2 - a3))/a5
        def dlambda0_dbf(x):
            a1 = dQ_dbf(x)*Ag(x)*Ne(x)
            a2 = Q(x)*dAg_dbf(x)*Ne(x)
            a3 = Q(x)*Ag(x)*dNe_dbf(x)
            a4 = (fy**(1/2))/(2*(Ne(x)**2))
            a5 = ((Q(x)*Ag(x))/Ne(x))**(1/2)
            return (a4*(a1 + a2 - a3))/a5
        def dqui_dh(x):
            a1 =  0.658**(lambda0(x)**2)
            a2 = 2*lambda0(x)*dlambda0_dh(x)
            a3 = -1.754/(lambda0(x)**3)
            if lambda0(x) <= 1.5:
                return a1*a2*(np.log(0.658))
            else:
                return a3*dlambda0_dh(x)
        def dqui_dbf(x):
            a1 =  0.658**(lambda0(x)**2)
            a2 = 2*lambda0(x)*dlambda0_dbf(x)
            a3 = -1.754/(lambda0(x)**3)
            if lambda0(x) <= 1.5:
                return a1*a2*(np.log(0.658))
            else:
                return a3*dlambda0_dbf(x)
        def dNcrd_dh(x):
            a1 = dqui_dh(x)*Q(x)*Ag(x)
            a2 = qui(x)*dQ_dh(x)*Ag(x)
            a3 = qui(x)*Q(x)*dAg_dh(x)
            return (fy/1.15)*(a1 + a2 + a3)
        def dNcrd_dbf(x):
            a1 = dqui_dbf(x)*Q(x)*Ag(x)
            a2 = qui(x)*dQ_dbf(x)*Ag(x)
            a3 = qui(x)*Q(x)*dAg_dbf(x)
            return (fy/1.15)*(a1 + a2 + a3)

        """
        ================================================================================
                            DEFINIÇÃO DOS ARGUMENTOS PARA A OTIMIZAÇÃO
        ================================================================================
        """

        def jac_fun(x):
            dAg_dh = tw
            dAg_dbf = 2*tf
            return np.array([dAg_dh, dAg_dbf])

        def jac_con1(x):
            return np.array([dNcrd_dh(x), dNcrd_dbf(x)])

        def jac_con2(x):
            dh = -(kx*lx*drx_dh(x))/(rx(x)**2)
            dbf = -(kx*lx*drx_dbf(x))/(rx(x)**2)
            return np.array([dh, dbf])

        def jac_con3(x):
            dh = -(ky*ly*dry_dh(x))/(ry(x)**2)
            dbf = -(ky*ly*dry_dbf(x))/(ry(x)**2)
            return np.array([dh, dbf])

        """
        ================================================================================
                                    ALGORITMO DE OTIMIZAÇÃO
```

```
455.        ================================================================================
456.        """
457.
458.        bnds = ((h_inf, h_sup), (bf_inf, bf_sup))
459.        cons = ({'type': 'ineq',
460.                 'fun': lambda x: Ncrd(x) - Ncsd,
461.                 'jac': jac_con1},
462.                {'type': 'ineq',
463.                 'fun': lambda x: 200 - ((kx*lx)/rx(x)),
464.                 'jac': jac_con2},
465.                {'type': 'ineq',
466.                 'fun': lambda x: 200 - ((ky*ly)/ry(x)),
467.                 'jac': jac_con3})
468.
469.        for tw in tw_lista:
470.            for tf in tf_lista:
471.                res = opt.minimize(Ag,
472.                                   (h_sup, bf_sup),
473.                                   method='SLSQP',
474.                                   jac=jac_fun,
475.                                   bounds=bnds,
476.                                   constraints=cons,
477.                                   options={'maxiter': 200})
478.                if res.fun < Ag_otim and (Ncrd(res.x)/Ncsd) > (0.999999) and res.mess
    age == 'Optimization terminated successfully.':
479.                    Ag_otim = Ag(res.x)
480.                    x_otim = [res.x[0], res.x[1]]
481.                    tw_otim = tw
482.                    tf_otim = tf
483.                    Ncrd_otim = Ncrd(res.x)
484.
485.        """
486.        ================================================================================
487.                                        RESULTADOS
488.        ================================================================================
489.        """
490.
491.        print ('')
492.        print ('h ótima: ', x_otim[0])
493.        print ('bf ótima: ', x_otim[1])
494.        print ('tw ótima: ', tw_otim)
495.        print ('tf ótima: ', tf_otim)
496.        print ('Área ótima: ', Ag_otim)
497.        print ('Nc,Rd adotado: ', Ncrd_otim)
```

APÊNDICE B – DERIVADAS DAS EQUAÇÕES APLICADAS NA VERIFICAÇÃO DE SEGURANÇA DOS PERFIS EM RELAÇÃO ÀS VARIÁVEIS DE PROJETO.

Quadro 1 - Equações utilizadas no dimensionamento de estruturas de aço e suas derivadas em relação a h.

$f(h, b_f)$	$\frac{\partial f}{\partial h}$
$I_x = \frac{t_w {b_w}^3}{12} + 2\left[\frac{b_f {t_f}^3}{12} + b_f t_f \left(\frac{h - t_f}{2}\right)^2\right]$	$\frac{t_w {b_w}^2}{4} + b_f t_f (h - t_f)$
$I_y = \frac{b_w {t_w}^3}{12} + 2\frac{t_f {b_f}^3}{12}$	$\frac{{t_w}^3}{12}$
$A_g = 2b_f t_f + b_w t_w$	t_w
$r_x = \sqrt{\frac{I_x}{A_g}}$	$\frac{\frac{\partial I_x}{\partial h} A_g - \frac{\partial A_g}{\partial h} I_x}{2{A_g}^2 r_x}$
$r_y = \sqrt{\frac{I_y}{A_g}}$	$\frac{\frac{\partial I_y}{\partial h} A_g - \frac{\partial A_g}{\partial h} I_y}{2{A_g}^2 r_y}$
$r_0 = \sqrt{({r_x}^2 + {r_y}^2)}$	$\frac{\left(\frac{\partial r_x}{\partial h} r_x + \frac{\partial r_y}{\partial h} r_y\right)}{r_0}$
$C_w = \frac{I_y (h - t_f)^2}{4}$	$\frac{1}{4}\left[\frac{\partial I_y}{\partial h}(h - t_f)^2 + 2I_y(h - t_f)\right]$
$J = \frac{2b_f {t_f}^3 + (h - t_f){t_w}^3}{3}$	$\frac{{t_w}^3}{3}$
$N_{ex} = \frac{\pi^2 E I_x}{(K_x L_x)^2}$	$\left[\frac{\pi^2 E}{(K_x L_x)^2}\right]\frac{\partial I_x}{\partial h}$
$N_{ey} = \frac{\pi^2 E I_y}{(K_y L_y)^2}$	$\left[\frac{\pi^2 E}{(K_y L_y)^2}\right]\frac{\partial I_y}{\partial h}$

$N_{ez} = \frac{1}{{r_0}^2}\left[\frac{\pi^2 E C_w}{(K_z L_z)^2} + GJ\right]$	Ver quadro 2 - A
$\lambda_{0_\sigma} = \sqrt{\frac{A_g f_y}{N_e}}$	$\frac{f_y\left[\frac{\partial A_g}{\partial h}N_e - \frac{\partial N_e}{\partial h}A_g\right]}{2{N_e}^2\sqrt{\frac{A_g f_y}{N_e}}}$
$\chi_{\sigma 1} = 0{,}658^{{\lambda_{0_\sigma}}^2}$	$\chi_{\sigma 1}\left(2\lambda_{0_\sigma}\frac{\partial \lambda_{0_\sigma}}{\partial h}\right) ln(0{,}658)$
$\chi_{\sigma 2} = \frac{0{,}877}{{\lambda_{0_\sigma}}^2}$	$\frac{-1{,}754}{{\lambda_{0_\sigma}}^3}\frac{\partial \lambda_{0_\sigma}}{\partial h}$
$\sigma = \chi_\sigma f_y$	$f_y \frac{\partial \chi_\sigma}{\partial h}$
$b_{ef} = 1{,}92 t_w \sqrt{\frac{E}{\sigma}}\left[1 - \frac{0{,}34}{b_w/t_w}\sqrt{\frac{E}{\sigma}}\right]$	Ver quadro 2 - B
$A_{ef} = A_g - \left[b_w - b_{ef}\right]t_w$	$\frac{\partial A_g}{\partial h} - t_w\left[1 - \frac{\partial b_{ef}}{\partial h}\right]$
$Q_A = \frac{A_{ef}}{A_g}$	$\frac{\frac{\partial A_{ef}}{\partial h}A_g - \frac{\partial A_g}{\partial h}A_{ef}}{{A_g}^2}$
$k_c = \frac{4}{\sqrt{b_w/t_w}}$	$-2\sqrt{\frac{t_w}{{b_w}^3}}$
$Q_{s1} = 1{,}415 - 0{,}65\frac{b_f/2}{t_f}\sqrt{\frac{f_y}{k_c E}}$	$\frac{0{,}65 b_f}{4 t_f}\sqrt{\frac{f_y}{{k_c}^3 E}}\frac{\partial k_c}{\partial h}$
$Q_{s2} = \frac{0{,}90 E k_c}{f_y\left(\frac{b_f/2}{t_f}\right)^2}$	$\frac{0{,}9E}{f_y}\left(\frac{2t_f}{b_f}\right)^2\frac{\partial k_c}{\partial h}$

$Q = Q_A Q_S$	$\frac{\partial Q_A}{\partial h} Q_S + \frac{\partial Q_S}{\partial h} Q_A$
$\lambda_0 = \sqrt{\frac{Q A_g f_y}{N_e}}$	Ver quadro 2 - C
$\chi_1 = 0{,}658^{\lambda_0{}^2}$	$\chi_1 \left(2\lambda_0 \frac{\partial \lambda_0}{\partial h}\right) ln(0{,}658)$
$\chi_2 = \frac{0{,}877}{\lambda_0{}^2}$	$\frac{-1{,}754}{\lambda_0{}^3} \frac{\partial \lambda_0}{\partial h}$
$N_{c,Rd} = \frac{\chi Q A_g f_y}{\gamma_{a1}}$	$\frac{f_y}{\gamma_{a1}} \left[\frac{\partial \chi}{\partial h} Q A_g + \chi \frac{\partial Q}{\partial h} A_g + \chi Q \frac{\partial A_g}{\partial h}\right]$

Quadro 2 - Derivadas estendidas do Quadro 1.

A	$\frac{\partial N_{ez}}{\partial h} = \frac{1}{r_0{}^2} \left[\left(\frac{\pi^2 E}{(K_z L_z)^2}\right) \frac{\partial C_w}{\partial h} + G \frac{\partial \mathrm{J}}{\partial h}\right] - \frac{2}{r_0{}^3} \frac{\partial r_0}{\partial h} \left[\left(\frac{\pi^2 E}{(K_z L_z)^2}\right) C_w + GJ\right]$
B	$\frac{\partial b_{ef}}{\partial h} = 0{,}6528 t_{\mathrm{w}}{}^2 E \left(\frac{1}{b_{\mathrm{w}}{}^2 \sigma} + \frac{1}{b_{\mathrm{w}} \sigma^2} \frac{\partial \sigma}{\partial h}\right) - \frac{1{,}92 t_{\mathrm{w}} \sqrt{E}}{2\sqrt{\sigma^3}} \frac{\partial \sigma}{\partial h}$
C	$\frac{\partial \lambda_0}{\partial h} = \frac{\sqrt{f_y} \left(\frac{\partial \mathrm{Q}}{\partial h} A_g N_e + \mathrm{Q} \frac{\partial A_g}{\partial h} N_e - \mathrm{Q} A_g \frac{\partial N_e}{\partial h}\right)}{2 N_e{}^2 \sqrt{\frac{\mathrm{Q} A_g}{N_e}}}$

Quadro 3 - Equações utilizadas no dimensionamento de estruturas de aço e suas derivadas em relação a bf.

$f(h, b_f)$	$\frac{\partial f}{\partial b_f}$
$I_x = \frac{t_w {b_w}^3}{12} + 2\left[\frac{b_f {t_f}^3}{12} + b_f t_f \left(\frac{h - t_f}{2}\right)^2\right]$	$2\left[\frac{{t_f}^3}{12} + t_f \left(\frac{h - t_f}{2}\right)^2\right]$
$I_y = \frac{b_w {t_w}^3}{12} + 2\frac{t_f {b_f}^3}{12}$	$\frac{t_f {b_f}^2}{2}$
$A_g = 2b_f t_f + b_w t_w$	$2t_f$
$r_x = \sqrt{\frac{I_x}{A_g}}$	$\frac{\frac{\partial I_x}{\partial b_f} A_g - \frac{\partial A_g}{\partial b_f} I_x}{2{A_g}^2 r_x}$
$r_y = \sqrt{\frac{I_y}{A_g}}$	$\frac{\frac{\partial I_y}{\partial b_f} A_g - \frac{\partial A_g}{\partial b_f} I_y}{2{A_g}^2 r_y}$
$r_0 = \sqrt{({r_x}^2 + {r_y}^2)}$	$\frac{\left(\frac{\partial r_x}{\partial b_f} r_x + \frac{\partial r_y}{\partial b_f} r_y\right)}{r_0}$
$C_w = \frac{I_y (h - t_f)^2}{4}$	$\frac{1}{4}\left[\frac{\partial I_y}{\partial b_f}(h - t_f)^2\right]$
$J = \frac{2b_f {t_f}^3 + (h - t_f){t_w}^3}{3}$	$\frac{2{t_f}^3}{3}$
$N_{ex} = \frac{\pi^2 E I_x}{(K_x L_x)^2}$	$\left[\frac{\pi^2 E}{(K_x L_x)^2}\right]\frac{\partial I_x}{\partial b_f}$
$N_{ey} = \frac{\pi^2 E I_y}{(K_y L_y)^2}$	$\left[\frac{\pi^2 E}{(K_y L_y)^2}\right]\frac{\partial I_y}{\partial b_f}$

$N_{ez} = \frac{1}{{r_0}^2}\left[\frac{\pi^2 E C_w}{(K_z L_z)^2} + GJ\right]$	Ver quadro 4 - A
$\lambda_{0_\sigma} = \sqrt{\frac{A_g f_y}{N_e}}$	$\frac{f_y\left[\frac{\partial A_g}{\partial b_f} N_e - \frac{\partial N_e}{\partial b_f} A_g\right]}{2{N_e}^2\sqrt{\frac{A_g f_y}{N_e}}}$
$\chi_{\sigma 1} = 0{,}658^{{\lambda_{0_\sigma}}^2}$	$\chi_{\sigma 1}\left(2\lambda_{0_\sigma}\frac{\partial \lambda_{0_\sigma}}{\partial b_f}\right) ln(0{,}658)$
$\chi_{\sigma 2} = \frac{0{,}877}{{\lambda_{0_\sigma}}^2}$	$\frac{-2(0{,}877)}{{\lambda_{0_\sigma}}^3}\frac{\partial \lambda_{0_\sigma}}{\partial b_f}$
$\sigma = \chi_\sigma f_y$	$f_y \frac{\partial \chi_\sigma}{\partial b_f}$
$b_{ef} = 1{,}92 t_w \sqrt{\frac{E}{\sigma}}\left[1 - \frac{0{,}34}{b_w/t_w}\sqrt{\frac{E}{\sigma}}\right]$	Ver quadro 4 - B
$A_{ef} = A_g - \left[b_w - b_{ef}\right] t_w$	$\frac{\partial A_g}{\partial b_f} + t_w \frac{\partial b_{ef}}{\partial b_f}$
$Q_A = \frac{A_{ef}}{A_g}$	$\frac{\frac{\partial A_{ef}}{\partial b_f} A_g - \frac{\partial A_g}{\partial b_f} A_{ef}}{{A_g}^2}$
$k_c = \frac{4}{\sqrt{h/t_w}}$	0
$Q_{s1} = 1{,}415 - 0{,}65\frac{b_f/2}{t_f}\sqrt{\frac{f_y}{k_c E}}$	$-\frac{0{,}65}{2t_f}\sqrt{\frac{f_y}{k_c E}}$

$Q_{s2} = \dfrac{0{,}90Ek_c}{f_y\left(\dfrac{b_f/2}{t_f}\right)^2}$	$-\dfrac{0{,}9E8{t_f}^2k_c}{f_y{b_f}^3}$
$Q = Q_A Q_S$	$\dfrac{\partial Q_A}{\partial b_f}Q_S + \dfrac{\partial Q_S}{\partial b_f}Q_A$
$\lambda_0 = \sqrt{\dfrac{QA_g f_y}{N_e}}$	Ver quadro 4 - C
$\chi_1 = 0{,}658^{{\lambda_0}^2}$	$\chi_1\left(2\lambda_0\dfrac{\partial\lambda_0}{\partial b_f}\right)ln(0{,}658)$
$\chi_2 = \dfrac{0{,}877}{{\lambda_0}^2}$	$\dfrac{-1{,}754}{{\lambda_0}^3}\dfrac{\partial\lambda_0}{\partial b_f}$
$N_{c,Rd} = \dfrac{\chi Q A_g f_y}{\gamma_{a1}}$	$\dfrac{f_y}{\gamma_{a1}}\left[\dfrac{\partial\chi}{\partial b_f}QA_g + \chi\dfrac{\partial Q}{\partial b_f}A_g + \chi Q\dfrac{\partial A_g}{\partial b_f}\right]$

Quadro 4 - Derivadas estendidas do Quadro 3.

A	$\dfrac{\partial N_{ez}}{\partial b_f} = \dfrac{1}{{r_0}^2}\left[\left(\dfrac{\pi^2E}{(K_zL_z)^2}\right)\dfrac{\partial C_w}{\partial b_f} + G\dfrac{\partial \mathrm{J}}{\partial b_f}\right] - \dfrac{2}{{r_0}^3}\dfrac{\partial r_0}{\partial b_f}\left[\left(\dfrac{\pi^2E}{(K_zL_z)^2}\right)C_w + GJ\right]$
B	$\dfrac{\partial b_{ef}}{\partial b_f} = \dfrac{\partial\sigma}{\partial b_f}\left(\dfrac{0{,}6528{t_\mathrm{w}}^2E}{b_\mathrm{w}\sigma^2} - \dfrac{1{,}92t_\mathrm{w}\sqrt{E}}{2\sqrt{\sigma^3}}\right)$
C	$\dfrac{\partial\lambda_0}{\partial b_f} = \dfrac{\sqrt{f_y}\left(\dfrac{\partial \mathrm{Q}}{\partial b_f}A_gN_e + \mathrm{Q}\dfrac{\partial A_g}{\partial b_f}N_e - \mathrm{Q}A_g\dfrac{\partial N_e}{\partial b_f}\right)}{2{N_e}^2\sqrt{\dfrac{\mathrm{Q}A_g}{N_e}}}$