Guilherme de Frias Martins
Licenciado em Ciências de Engenharia Civil

Optimização de esforços em estruturas: influência da localização e rigidez dos apoios

Dissertação para obtenção do Grau de Mestre
em Engenharia Civil - Perfil Estruturas

Orientador: Professor Doutor João Carlos Rocha de Almeida
Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente:	Professor Doutor António Manuel Pinho Ramos
Arguente:	Professor Doutor João Burguete Cardoso
Vogal:	Professor Doutor João Carlos Rocha de Almeida

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro de 2012

Agradecimentos

Concluída esta dissertação, gostaria de expressar a minha gratidão a todas as pessoas que me apoiaram, não só durante a realização deste trabalho, mas também durante todo o meu percurso académico.

Em primeiro lugar, gostaria de agradecer ao professor João Rocha de Almeida, orientador desta dissertação, pelo apoio e disponibilidade demonstradas ao longo do desenvolvimento deste trabalho.

Em segundo lugar, um agradecimento a todos os meus amigos que estiveram presentes nos maus e bons momentos, e que foram sempre um grande suporte na minha vida. Um agradecimento especial para quatro amigos: Soraia Machado, Mário Ferreira, Inês Oliveira e Raquel Frutuoso.

Um grande agradecimento ao meu colega e amigo Carlos Mata pelo apoio prestado no decorrer deste trabalho, principalmente na parte de programação em MATLAB.

O maior agradecimento vai para a minha família, em particular os meus pais, pela ajuda, não só financeira, mas também emocional prestada ao longo da minha vida.

Resumo

No presente trabalho é apresentado um procedimento para optimização estrutural dos esforços em vigas e grelhas, em função da posição e rigidez dos seus apoios.

A optimização é efectuada com recurso a métodos de optimização não-linear, nomeadamente os algoritmos de procura directa de Nelder-Mead e os algoritmos genéticos. Ambos os algoritmos estão disponíveis na ferramenta de optimização do MATLAB, *optimtool*.

Para a realização da análise estrutural foram criados dois algoritmos baseados no método dos elementos finitos, sendo um para análise de vigas e o outro para análise de grelhas. A codificação do modelo estrutural foi feita também no programa MATLAB, de modo a facilitar todo o processo de optimização.

Foram realizados vários exemplos de optimização estrutural com o objectivo de comprovar a utilidade do processo proposto. Com base nos resultados obtidos, verificou-se que a optimização da posição e rigidez dos apoios permite uma diminuição significativa dos esforços que actuam nas estruturas.

Palavras chave:

Optimização estrutural; Método de Nelder-Mead; Algoritmos genéticos; Método dos elementos finitos; Vigas; Grelhas.

Abstract

This dissertation presents a structural optimization procedure of stresses in beams and grid structures. The design variables are the position and stiffness of the supports.

Non-linear optimization methods are used to perform the optimization procedure, in particular the Nelder-Mead direct search algorithm and the genetic algorithm. Both of these algorithms are available in the MATLAB's optimization tool, *optimtool*.

Two algorithms have been made for the structural analysis, based in the finite element method, one for beam analysis and the other for grid analysis. The algorithm code used for the structural model is also in MATLAB's language, in order to ease the optimization process.

Many structural optimization examples were carried out, in order to assess and validate this procedure. From these examples, it can be seen that the optimization of the position and stiffness of the suports allows a significant reduction of the stresses in the structure.

Keywords:

Structural optimization; Nelder-Mead's method; Genetic algorithm; Finite element method; Beams; Grids.

Índice de matérias

Índice de figuras

Índice de Tabelas

Lista de abreviaturas, siglas e símbolos

Abreviaturas

MEF	Método dos elementos finitos
PTV	Princípio dos trabalhos virtuais

Siglas

$DimPop$	Número de indivíduos da população inicial
PIR	Limites do domínio da população inicial
SolConst	Solução com restrições
SolDiv	Solução com espaçamentos iguais
SolMat	Solução sem restrições

Símbolos

$\mathbf{B}$	Vector das segundas derivadas das funções de forma
E	Módulo de elasticidade
$\mathbf{F}^{(e)}$	Vector de solicitação do elemento finito
$\mathbf{F'}^{(e)}$	Vector de solicitação do elemento finito em eixos locais
$\mathbf{F}_C$	Forças concentradas
F_i	Valor da aptidão do indivíduo
G	Módulo de distorção
$\mathbf{H}$	Vector das funções de forma de Hermite
I	Momento de inércia da secção segundo o eixo x_3
J	Factor de rigidez de torção
$\mathbf{K}^{(e)}$	Matriz de rigidez do elemento finito
$\mathbf{K'}^{(e)}$	Matriz de rigidez do elemento finito em eixos locais

K_j	Rigidez das molas da viga
$\mathbf{K_j}^{(e)}$	Matriz de rigidez à torção do elemento finito
K_m	Rigidez da mola
L	Comprimento total da viga
L_i	Comprimento dos tramos da viga
L_{tot}	Comprimento total da viga x
Lx_i	Comprimento dos espaçamentos dos apoios na direcção x
Lx_{total}	Comprimento total da grelha na direcção x
Ly_i	Comprimento dos espaçamentos dos apoios na direcção y
Ly_{total}	Comprimento total da grelha na direcção y
M^*	Momento flector máximo óptimo
M_{Ed}	Momento flector actuante
M_{Rd}	Momento flector resistente
N	Vizinhança pequena do espaço admissível
P_i	Carga distribuída nas vigas da grelha; Probabilidade de selecção de um indivíduo
$P(\mathbf{x})$	Termo de penalidade
$\mathbf{R}$	Matriz de rotação
$\mathbf{R}_{apoios}$	Reacções nos apoios
S	Região admissível
Q	Carga de superfície nas grelhas
V	Volume do corpo
X	Vector das variáveis de projecto
$\mathbf{f}^B$	Forças de volume
$\mathbf{f}^{S_f}$	Forças de superficie
f_y	Tensão de cedência do aço
$f(\mathbf{x})$	Função objectivo
$\mathbf{fe}^{(e)}$	Função objectivo
$g_j(\mathbf{x})$	Restrições de desigualdade
$h_k(\mathbf{x})$	Restrições de igualdade

l	Comprimento do elemento finito
l_b	Limite inferior das varáveis
p	Carga uniformemente distribuída
$\mathbf{r}$	Vector dos parâmetros de penalidade
$\mathbf{u}^{(e)}$	Vector dos deslocamentos generalizados do elemento finito
$u(x_1)$	Deslocamentos axiais do elemento finito
v_1	Deslocamento do nó 1 segundo o eixo x_2
v_1	Deslocamento do nó 2 segundo o eixo x_2
$v(x_1)$	Função do deslocamento vertical do elemento finito ao longo de x_1
w_{el}	Módulo de flexão elástico
$\mathbf{x}$	Vector das varáveis de projecto
x_c	Ponto de contracção para fora
x_{cc}	Ponto de contracção para dentro
x_e	Ponto de expansão
x_r	Ponto de reflexão
Γ	Fronteira delimitadora da estrutura
Ω	Domínio da estrutura
χ	Coeficiente de expansão
$\delta\varepsilon$	Deformações virtuais
$\delta\mathbf{U}$	Deslocamentos virtuais
ε	Deformações; Parãmetro de tolerância do critério de paragem
ε_1	Deformações axiais do elemento finito
γ	Coeficiente de contracção
$\phi(\mathbf{x}, \mathbf{r})$	Função de transformação
ρ	Tensões do corpo; Coeficiente de reflexão
σ	Coeficiente de encolhimento
θ	Ângulo
θ_1	Rotação do nó 1 segundo o eixo x_3
θ_2	Rotação do nó 2 segundo o eixo x_3
$\theta(x_1)$	Função da rotação do elemento finito ao longo de x_1

Capítulo 1

Introdução

1.1 Enquadramento

Ao longo das últimas décadas, tem-se assistido a um grande desenvolvimento na área da optimização, tendo esta cada vez maior importância na resolução de múltiplos problemas de engenharia. Vários autores, como Schmit [35], Arora [2] e Haftka *et al* [16], desenvolveram trabalhos em que apresentam procedimentos que permitem optimizar parâmetros das estruturas. De entre os diversos procedimentos, destacam-se diversos métodos de optimização não-linear.

Os métodos de optimização não-linear são métodos numéricos que usam processos iterativos com o objectivo de encontrar um mínimo de uma função, de uma forma rápida e eficiente. Com o desenvolvimento destes métodos, têm surgido *softwares* comerciais, nos quais a programação dos métodos já se encontra definida, como é o caso do MATLAB.

Parte do processo de optimização estrutural está relacionada com a análise das estruturas. Com a generalização do uso dos computadores, a análise estrutural tem sido cada vez mais efectuada a partir de métodos numéricos, de onde se destaca o método dos elementos finitos (MEF).

Apesar do progresso que se tem verificado no campo da optimização estrutural, existe uma discrepância em termos do desenvolvimento teórico e da sua aplicação em termos práticos. Isto deve-se ao facto de os métodos de optimização se basearem em métodos matemáticos que levam a soluções "perfeitas"; no entanto, essas soluções muitas vezes não são possíveis de serem postas em prática. Deste modo, o processo de optimização estrutural faz parte de um processo mais complexo de engenharia [8].

Por norma os problemas de optimização estrutural estão relacionados com a minimização do custo ou peso da estrutura; no entanto, existem outros parâmetros possíveis de optimizar, como os esforços actuantes, dos quais podem resultar soluções estruturais mais eficientes.

Nesta dissertação pretende-se minimizar os esforços máximos actuantes nas estruturas em função da localização e rigidez dos seus apoios, supondo que existe total liberdade nos valores que estes parâmetros podem tomar. Pretende-se ainda comparar as soluções óptimas com soluções que construtivamente são mais viáveis.

Apesar da obtenção de esforços que resultam em estruturas mais rentáveis estar associada, por norma, a análises plásticas das mesmas, nesta dissertação apenas se considera uma análise estrutural elástica. Deste modo, os esforços máximos a optimizar são esforços elásticos.

1.2 Objectivos

O objectivo da presente dissertação prende-se com a optimização dos esfoços elásticos que actuam em estruturas, em função da posição e da rigidez dos apoios. As estruturas a optimizar são vigas contínuas e sistemas de grelhas. Pretende-se desta forma, criar um procedimento capaz de realizar a optimização destas estruturas o mais eficazmente possível.

Para tal, é necessário dividir o processo de optimização em duas parte: uma em que é realizada a análise estrutural dos sistemas, e outra em que é efectuada a optimização dos pârametros desses sistemas. A análise estrutural de cada tipo de estrutura será executada a partir de um programa, criado com esse propósito, no MATLAB. Este programa tem como base o método dos elementos finitos. Para a optimização estrutural, pretendem-se aplicar métodos de optmização não-linear, nomeadamente o algoritmo de Nelder-Mead e os algoritmos genéticos. Estes algoritmos estão presentes na ferramenta de optimização, *optimtool*, do MATLAB. Prentende-se também estudar a influência da optimização nos resultados obtidos e estabelecer comparações com soluções construtivas mais correntes.

1.3 Organização da dissertação

A presente dissertação encontra-se dividida em cinco capítulos, que se passam a descrever.

No primeiro capítulo é feito um enquadramento ao tema e apresentados os principais objectivos desta dissertação.

No capítulo 2 é apresentado o método dos elementos finitos, dado ser o método numérico adoptado para a análise estrutural. Inicialmente é feito um enquadramento do método, apresentando os seus princípios e fundamentos. Neste capítulo são apresentadas formulações e procedimentos para a análise estrutural de vigas e grelhas, que são necessárias para a realização dos algoritmos criados no âmbito desta dissertação. De referir que, no âmbito desta dissertação, apenas se realiza uma análise elástica das estruturas.

No capítulo 3 é feita uma abordagem geral ao tema da optimização estrutural, sendo efectuada uma pequena introdução histórica e mostrados os principais conceitos de optimização. Neste capítulo, são ainda descritos métodos de optimização não-linear, em particular o método de Nelder-Mead e os algoritmos genéticos.

Com base nos fundamentos teóricos expostos nos capítulos anteriores, são apresentados no capítulo 4 os casos de estudos efectuados no âmbito da dissertação, expondo as análises realizadas aos resultados e a respectiva discussão.

O último capítulo pretende expor, resumidamente, as principais conclusões obtidas a partir da realização desta dissertação e apresentar algumas ideias para desenvolvimento futuro, de forma a dar continuidade ao tema.

Capítulo 2

Método dos elementos finitos em análise estrutural

2.1 Princípios e fundamentos

No âmbito da engenharia, nomeadamente civil, a criação de modelos de fenómenos físicos que caracterizam os diversos fenómenos da natureza, sempre foi uma actividade de extrema importância. Estes fenómenos podem ser descritos por leis físicas e modelados através de equações algébricas, diferenciais ou integrais.

A caracterização dos fenómenos a partir de descrições analíticas é efectuada por modelos matemáticos, que podem ser definidos por um conjunto de equações que exprime características essenciais do comportamento de um corpo. A resolução de um modelo matemático depende essencialmente da sua complexidade, sendo que para um modelo simples a resolução analítica é fácil de obter, enquanto que para modelos mais complexos a resolução analítica pode ser difícil de obter ou até mesmo impossível.

No entanto, a maior utilização de computadores nos últimos anos veio possibilitar a resolução de muitos problemas de engenharia. A utilização de computadores para resolução de probelmas matemáticos implica o recurso a métodos numéricos. Dentro dos diversos métodos numéricos adoptados, destaca-se o método dos elementos finitos (MEF).

Este método foi formalmente introduzido pelos trabalhos de H. Argyris e S. Kelsey [1], e por M. J. Turner, R. W. Clough, H. C. Martin, e L. J. Topp [40], apesar de trabalhos anteriores de Hrenikoff [18] e de Courant [12], serem considerados os percursores do método. A designação de elemento finito foi utilizada, pela primeira vez na publicação de R. W. Clough [7], em 1960, e desde então, o método tem vindo a ser aperfeiçoado por vários autores.

O MEF é um método numérico em que o domínio do problema é decomposto em vários subdomínios, designados por elementos finitos, e para cada um dos elementos são definidas funções simples, como polinómios, que aproximam as variáveis do problema. Para solucionar um problema pelo MEF é necessário obter um sistema de equações que seja formulado como um integral definido em todo o domínio do problema, de modo a que todos os subdomínios respectivos possam ser definidos por esse mesmo integral. Assim, a solução numa formulação pelo MEF consiste em fazer um somatório do integral em todos os subdomínios, obtendo-se desta forma uma aproximação da solução em todo o domínio do problema. A equação diferencial baseia-se em princípios variacionais [6].

No âmbito desta dissertação, o princípio variacional usado é o princípio dos trabalhos virtuais (PTV) [3][14], pois para a análise estrutural trata-se da formulação mais intuitiva, resultando assim numa formulação de elementos finitos baseada em deslocamentos.

O princípio dos trabalhos virtuais enuncia que o equilíbrio de um corpo é satisfeito se o trabalho virtual de todas as forças que nele actuam for nulo, para todo o deslocamento virtual do corpo, ou seja, o trabalho virtual interno total do corpo tem de ser igual ao seu trabalho virtual externo total. Este princípio pode ser traduzido pela seguinte expressão [14]:

$$\int_V \delta\varepsilon^T \sigma dV = \int_V \delta\mathbf{U}^T \mathbf{f}^B dV + \int_{S_f} \delta\mathbf{U}^{S_f T} \mathbf{f}^{S_f} dS_f + \sum_i \delta\mathbf{U}_i^T \mathbf{F}_C^i \tag{2.1}$$

Onde $\delta\mathbf{U}$ e $\delta\varepsilon$ são os deslocamentos e as deformações virtuais, respectivamente, e σ são as tensões do corpo. As forças $\mathbf{f}^B$ (forças de volume), $\mathbf{f}^{S_f}$ (forças distribuídas na superfície S_f) e $\mathbf{F}_C^i$ (forças concentradas, onde i é o ponto de aplicação da carga) são forças externas aplicadas ao corpo.

Numa análise de elementos finitos, o corpo é aproximado a um conjunto de elementos finitos discretos, que são interligados em nós nas fronteiras destes. Considerando o campo de deslocamentos de cada elemento, pode-se escrever a equação (2.1), como uma soma de integrações sobre o domínio de todos os elementos finitos:[14]

$$\begin{aligned}\sum_e \int_{V^{(e)}} \delta\varepsilon^{(e)T} \sigma^{(e)} dV^{(e)} = \sum_e \int_{V^{(e)}} \delta\mathbf{u}^{(e)T} \mathbf{f}^{B(e)} dV^{(e)} + \\ + \sum_e \int_{S^{(e)}} \delta\mathbf{u}^{S(e)T} \mathbf{f}^{S(e)} dS^{(e)} + \sum_i \delta\mathbf{u}_i^T \mathbf{F}_C^i\end{aligned} \tag{2.2}$$

Em que $e = 1, 2, \ldots, k$, sendo k o número de elementos finitos definidos no corpo. De referir que na equação (2.2) os pontos onde as cargas concentradas estão aplicadas são pontos nodais discretizados na malha de elementos finitos.

Este princípio é válido para qualquer tipo de elementos que se pretendam utilizar; no entanto, as expressões vão sofrendo diferentes modificações para cada diferente tipo de elemento finito.

2.2 Elementos finitos de viga de Euler-Bernoulli

Num elemento finito de viga de Euler-Bernoulli [39], considera-se que, após a deformação, as secções mantêm-se planas e perpendiculares ao eixo da viga, não sendo, deste modo, considerada a deformação devida ao corte. Na Figura 2.1 é ilustrada uma deformação de um elemento de viga de Euler-Bernoulli com dois nós.

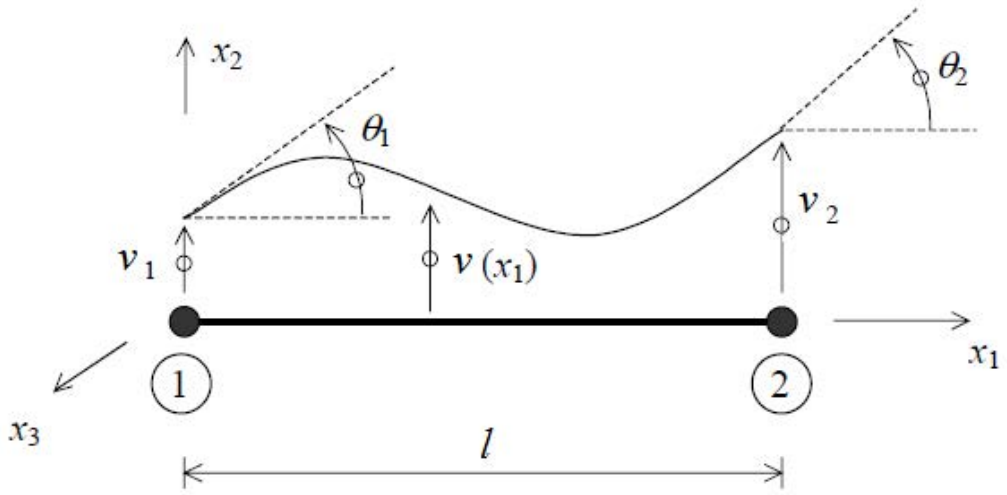


Figura 2.1: Elemento de viga de Euler-Bernoulli (adaptado de [38])

Os deslocamentos v_1 e v_2, representados na Figura 2.1, correspondem aos deslocamentos segundo o eixo x_2, e θ_1 e θ_2 correspondem às rotações segundo o eixo perpendicular ao plano da viga. Devido ao facto de o elemento ter quatro deslocamentos nodais, assume-se que o deslocamento segundo x_2 pode ser expresso pelo seguinte polinómio:

$$v(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3 \tag{2.3}$$

A partir da teoria de vigas de Euler-Bernoulli, a rotação $\theta(x_1) = \partial v(x_1)/\partial x_1$, logo:

$$\theta(x_1) = \partial v(x_1)/\partial x_1 = a_1 + 2a_2 x_1 + 3a_3 x_1^2 \tag{2.4}$$

O deslocamento $v(x_1)$, pode também ser expresso utilizando funções de forma, $\mathbf{H}$, que relacionam o deslocamento $v(x_1)$ com os deslocamentos nodais do elemento:

$$v(x_1) = H_1 v_1 + H_2 \theta_1 + H_3 v_2 + H_4 \theta_2 = \mathbf{H}\mathbf{u}^{(e)} \tag{2.5}$$

Onde $\mathbf{u}^{(e)}$ é o vector dos deslocamentos generalizados nos nós do elemento de viga, definido da seguinte forma:

$$\mathbf{u}^{(e)} = \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} \tag{2.6}$$

Considerando as condições de fronteira do elemento da Figura 2.1, é possível obter o seguinte sistema de equações:

$$\begin{array}{lcl} x_1 = 0 & \Rightarrow & v(0) = a_0 = v_1 \\ x_1 = 0 & \Rightarrow & \theta(0) = a_1 = \theta_1 \\ x_1 = l & \Rightarrow & v(l) = a_0 + a_1 l + a_2 l^2 + a_3 l^3 = v_2 \\ x_1 = l & \Rightarrow & \theta(0) = a_1 + 2a_2 l + 3a_3 l^2 = \theta_2 \end{array}$$

Resolvendo o sistema anterior e substituindo na equação (2.5), agrupam-se os termos referentes a v_1, v_2, θ_1 e θ_2, obtêm-se as funções de forma, que correspondem aos polinómios de Hermite, dados por:

$$\begin{aligned} H_1 &= 1 - 3\tfrac{x_1^2}{l^2} + 2\tfrac{x_1^3}{l^3} \\ H_2 &= x_1 - 2\tfrac{x_1^2}{l} + \tfrac{x_1^3}{l^2} \\ H_3 &= 3\tfrac{x_1^2}{l^2} - 2\tfrac{x_1^3}{l^3} \\ H_4 &= x_- \tfrac{x_1^2}{l} + \tfrac{x_1^3}{l^2} \end{aligned} \tag{2.7}$$

No gráfico da Figura 2.2 são representadas as variações dos valores das funções de forma hermitianas ao longo do comprimento do elemento finito. Estas funções são de classe C^1, o que significa que tanto $v(x_1)$ como $\partial v(x_1)/\partial x_1$ são contínuas dentro do domínio do elemento.

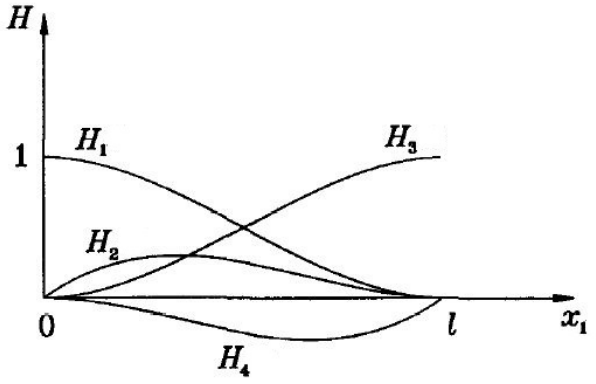


Figura 2.2: Funções de forma de Hermite (adaptado de [16])

Da teoria de vigas de Euler-Bernoulli sabe-se que o deslocamento axial $u(x_1)$ da viga é dado por:

$$u(x_1) = -x_2 \cdot \frac{\partial v(x_1)}{\partial x_1} \tag{2.8}$$

Sabendo que as deformações nas vigas são dadas pelas seguintes relações:

$$\varepsilon_1 = \frac{\partial u(x_1)}{\partial x_1} = -x_2 \cdot \frac{\partial^2 v(x_1)}{\partial x_1^2} \tag{2.9}$$

As deformações podem então ser relacionadas com as funções de forma da seguinte maneira:

$$\varepsilon_1 = x_2 \cdot \mathbf{B} \cdot \mathbf{u} \tag{2.10}$$

Onde,

$$\mathbf{B} = \left[\begin{array}{cccc} -\frac{\partial^2 H_1}{\partial x_1^2} & -\frac{\partial^2 H_2}{\partial x_1^2} & -\frac{\partial^2 H_3}{\partial x_1^2} & -\frac{\partial^2 H_1}{\partial x_1^2} \end{array} \right] \tag{2.11}$$

As tensões na viga de Euler-Bernoulli podem relacionar-se com as deformações a partir da lei de Hooke:

$$\sigma_1 = \varepsilon_1 \cdot E = E \cdot x_2 \cdot \mathbf{B} \cdot \mathbf{u}^{(e)} \tag{2.12}$$

A equação (2.1) do PTV, apresentada anteriormente, pode ser escrita para o caso das vigas de Euler-Bernoulli do seguinte modo:

$$\int_0^l \int_S \delta\varepsilon_1^T \sigma_1 dS dx_1 = \int_0^l \delta u_2 p dx_1 \tag{2.13}$$

Em que p é o carregamento que actua ao longo do eixo da viga e S é a superfície da secção transversal da viga, dada por:

$$dS = dx_2 dx_3 \tag{2.14}$$

Considerando as expressões anteriores, a equação (2.13) pode ser então expressa por:

$$EI \int_0^l \mathbf{B}^T \mathbf{B} dx_1 \mathbf{u}^{(e)} = \int_0^l \mathbf{H}^T p dx_1 \tag{2.15}$$

Em que I corresponde ao momento de inércia da secção segundo o eixo x_3, e é dado por:

$$I = \int_S x_2^2 dS \tag{2.16}$$

A equação (2.15) pode ainda ser expressa do seguinte modo:

$$\mathbf{K}^{(e)}\mathbf{u}^{(e)} = \mathbf{F}^{(e)} \tag{2.17}$$

Onde,

$$\mathbf{K}^{(e)} = EI \int_0^l \mathbf{B}^T \mathbf{B} dx_1 \tag{2.18}$$

é a matriz de rigidez e

$$\mathbf{F}^{(e)} = \int_0^l \mathbf{H}^T p dx_1 \tag{2.19}$$

é o vector de solicitação.

Resolvendo os produtos de integração que surgem na equação (2.18), obtém-se a seguinte matriz de rigidez para um elemento finito, que coincide com a que se obtém a partir dos métodos clássicos da teoria das estruturas [15]:

$$\mathbf{K}^{(e)} = EI \begin{bmatrix} \frac{12}{l^3} & \frac{6}{l^2} & -\frac{12}{l^3} & \frac{6}{l^2} \\ \frac{6}{l^2} & \frac{4}{l} & -\frac{6}{l^2} & \frac{2}{l} \\ -\frac{12}{l^3} & -\frac{6}{l^2} & \frac{12}{l^3} & -\frac{6}{l^2} \\ \frac{6}{l^2} & \frac{2}{l} & -\frac{6}{l^2} & \frac{4}{l} \end{bmatrix} \tag{2.20}$$

Assuma-se agora que a viga está sujeita a uma carga uniformente distribuída, como ilustrado na Figura 2.3.

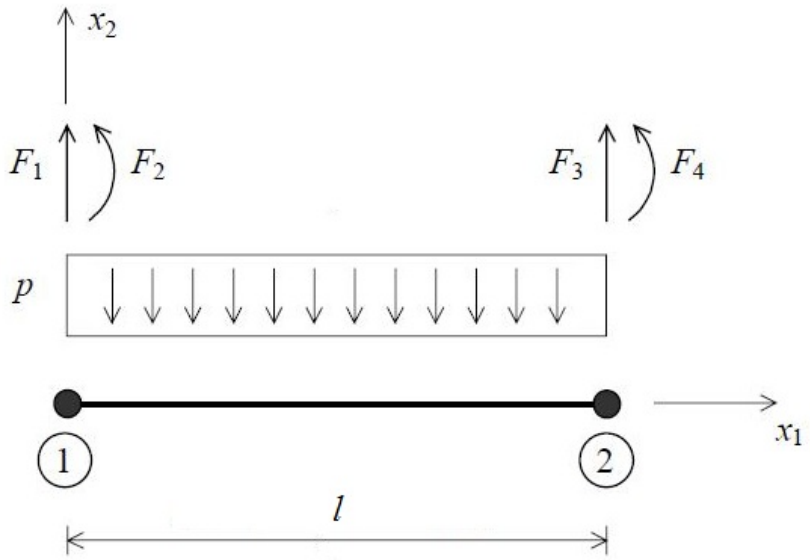


Figura 2.3: Carga uniformemente distribuída no elemento viga e respectivas forças nodais equivalentes (adaptado de [38])

Os sentidos positivos das forças nodais equivalentes considerados são os mesmos dos deslocamentos generalizados. Efectuando os cálculos da equação (2.19), o vector de solicitação passa a tomar a seguinte forma:

$$\mathbf{F}^{(e)} = -p \begin{bmatrix} l/2 \\ l^2/12 \\ l/2 \\ -l^2/12 \end{bmatrix} \tag{2.21}$$

2.3 Elementos finitos de grelha

Uma estrutura em grelha é uma estrutura plana constituída por várias vigas, em que as cargas são aplicadas perpendicularmente ao plano da estrutura. Quando as vigas estão rigidamente unidas entre si, vão estar sujeitas a momentos de flexão e de torção. A Figura 2.4 ilustra um exemplo de uma estrutura em grelha.

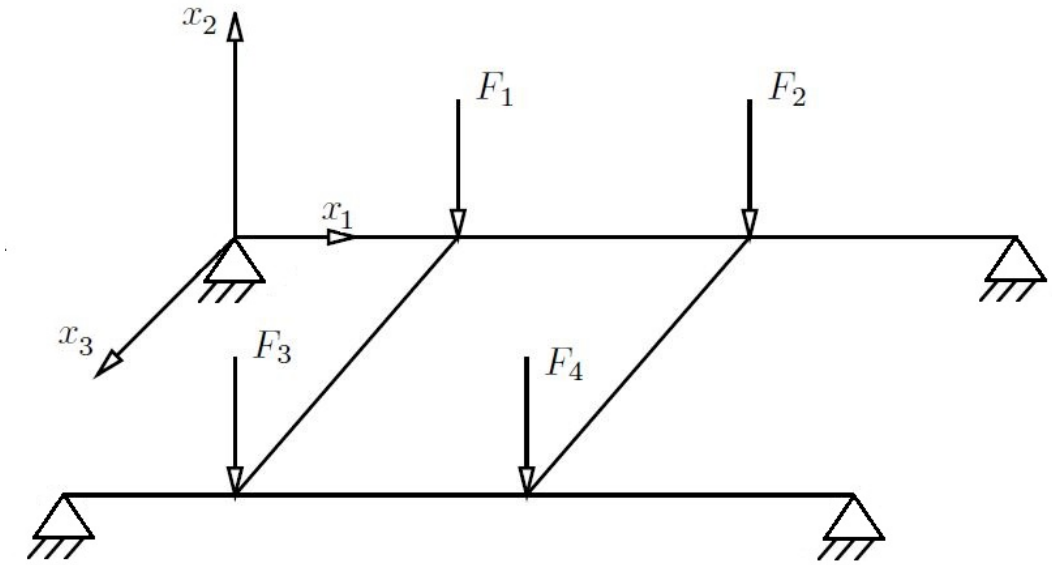


Figura 2.4: Estrutura em grelha (adaptado de [14])

Um elemento finito de grelha, como ilustrado na Figura 2.5, apresenta em cada nó um deslocamento transversal ao plano e duas rotações segundo os eixos na direcção do plano.

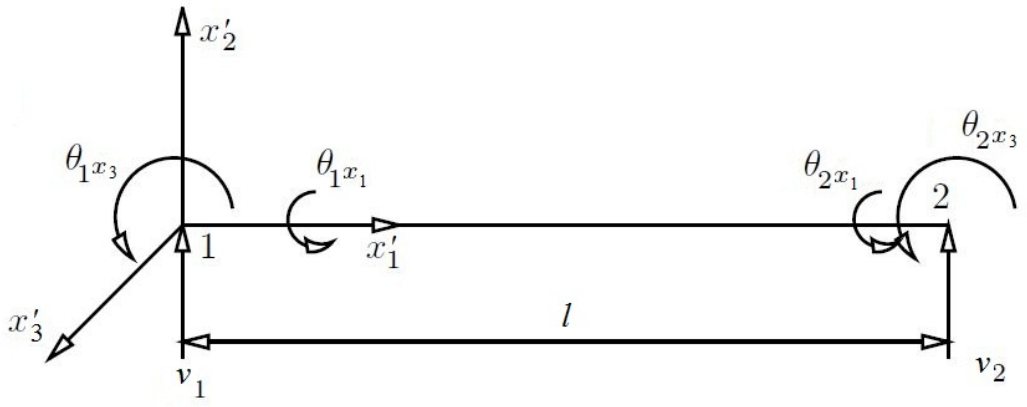


Figura 2.5: Elemento finito do tipo grelha (adaptado de [14])

Desta forma, o vector dos deslocamentos generalizados do elemento finito de grelha é expresso da seguinte forma:

$$\mathbf{u}^{(e)} = \begin{bmatrix} v_1 \\ \theta_{1_{x1}} \\ \theta_{1_{x3}} \\ v_2 \\ \theta_{2_{x1}} \\ \theta_{2_{x3}} \end{bmatrix} \tag{2.22}$$

Os deslocamentos v_1, $\theta_{1_{x3}}$, v_2 e $\theta_{2_{x3}}$ estão relacionados com a deformação da viga devida à flexão, como foi indicado no ponto anterior. Os deslocamentos $\theta_{1_{x1}}$ e $\theta_{2_{x1}}$ são rotações que estão relacionadas com a deformação da viga devida à torção. Segundo os métodos clássicos da teoria das estruturas [15] a rigidez de torção de um elemento de viga é igual a:

$$\mathbf{K_J}^{(e)} = \begin{bmatrix} \frac{GJ}{l} & -\frac{GJ}{l} \\ -\frac{GJ}{l} & \frac{GJ}{l} \end{bmatrix} \tag{2.23}$$

Onde J corresponde ao factor de rigidez de torção e G ao módulo de distorção [36]. Associando as equações (2.23) e (2.18), e ajustando-as aos respectivos deslocamentos, obtém-se a matriz de rigidez de um elemento finito para o seu sistema de eixos locais.

$$\mathbf{K'}^{(e)} = \begin{bmatrix} \frac{12EI}{l^3} & 0 & \frac{6EI}{l^2} & -\frac{12EI}{l^3} & 0 & \frac{6EI}{l^2} \\ 0 & \frac{GJ}{l} & 0 & 0 & -\frac{GJ}{l} & 0 \\ \frac{6EI}{l^2} & 0 & \frac{4EI}{l} & -\frac{6EI}{l^2} & 0 & \frac{2EI}{l} \\ -\frac{12EI}{l^3} & 0 & -\frac{6EI}{l^2} & \frac{12EI}{l^3} & 0 & -\frac{6EI}{l^2} \\ 0 & -\frac{GJ}{l} & 0 & 0 & \frac{GJ}{l} & 0 \\ \frac{6EI}{l^2} & 0 & \frac{2EI}{l} & -\frac{6EI}{l^2} & 0 & \frac{4EI}{l} \end{bmatrix} \tag{2.24}$$

Esta matriz de rigidez está associada ao sistema de eixos locais do elemento finito, que pode não corresponder ao sistema de eixos globais da estrutura. Deste modo, é necessário fazer uma transformação da matriz que permita passar dos eixos locais para os eixos globais. Para tal, é criada a matriz de rotação $\mathbf{R}$, dada por:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & C & S & 0 & 0 & 0 \\ 0 & -S & C & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & C & S \\ 0 & 0 & 0 & 0 & -S & C \end{bmatrix} \tag{2.25}$$

Com $C = \cos\theta$ e $S = \sin\theta$, sendo θ o ângulo entre o eixo x_1 e o eixo x'_1, no sentido anti-horário. A matriz de rigidez do elemento em eixos globais é então calculada a partir da seguinte expressão:

$$\mathbf{K}^{(e)} = \mathbf{R}^T \mathbf{K}'^{(e)} \mathbf{R} \tag{2.26}$$

Considerando que o elemento de grelha é também solicitado por uma carga uniformemente distribuída, como a da Figura 2.3, o vector de solicitações nos eixos locais é dado por:

$$\mathbf{F}'^{(e)} = -p \begin{bmatrix} l/2 \\ 0 \\ l^2/12 \\ l/2 \\ 0 \\ -l^2/12 \end{bmatrix} \tag{2.27}$$

Como é também necessária a sua rotação de eixos locais para eixos globais, obtem-se:

$$\mathbf{F}^{(e)} = \mathbf{R}^T \mathbf{F}'^{(e)} \tag{2.28}$$

2.4 Procedimento geral do método

Qualquer que seja o problema de elementos finitos, este deve ter em conta as seguintes etapas:

1. **Definição do sistema** - Neste passo são introduzidos todos os principais parâmetros que definem o sistema estrutural necessários para a análise estrutural recorrendo ao MEF, nomeadamente:

   - Número total de elementos;
   - Número total de nós;
   - Coordenadas de todos os nós no sistema de eixos global;
   - Tipo de elementos finitos;

   São ainda definidos parâmetros relacionados com o tipo de material, como o módulo de elasticidade e o módulo de distorção, e com o tipo de secção, como o momento de inércia e o factor de rigidez de torção.

2. **Assemblagem das matrizes e vectores** - Após calcular a matriz de rigidez e o vector de solicitação de cada elemento finito, é necessário fazer a assemblagem das contribuições de cada elemento no sistema global [14], representado pela seguinte equação:

$$\mathbf{Ku} = \mathbf{F} \tag{2.29}$$

A dimensão da matriz do sistema é igual ao número total de graus de liberdade do sistema.

3. **Imposição das condições de apoio** - As restrições das condições de apoio são impostas através de um vector que contém os graus de liberdade restringidos. Como os deslocamentos dos graus de liberdade restringidos deixam de ser incógnitas, são retiradas do sistema as respectivas parcelas, impedindo assim que a matriz se torne singular [23].

   No caso de os apoios serem molas, o valor da rigidez de cada mola é somado à rigidez do respectivo grau de liberdade na matriz de rigidez global. Considerando, por exemplo que, num sistema de elementos finitos, o i-ésimo grau de liberdade está apoiado numa mola de rigidez K_m, esta rigidez é somada ao valor de rigidez da matriz de rigidez global, na posição (i, i).

4. **Resolução do sistema** - Com os passos anteriores já efectuados, resolve-se então a equação (2.29), obtendo os valores dos deslocamentos nodais do sistema.

5. **Pós-processamento** - Obtidos os deslocamentos, é possível obter outros parâmetros como as deformações, tensões, esforços nos elementos e reacções dos apoios.

   O valor das reacções nos apoios pode ser obtido pela seguinte expressão:

$$\mathbf{R}_{apoios} = \mathbf{Ku} - \mathbf{F} \tag{2.30}$$

   Onde a parcela $\mathbf{Ku}$ calcula as forças externas em todos os nós. Deste modo, é subtraído à parcela anterior o vector de solicitação $\mathbf{F}$, obtendo-se o vector $\mathbf{R}_{apoios}$, com os valores das reacções nos graus de liberdade respectivos e nos restantes o valor zero.

   Os esforços internos em cada elemento finito podem ser calculados pelo mesmo princípio das reacções nos apoios. Desta forma, é calculado um vector para cada elemento, pela seguinte expressão:

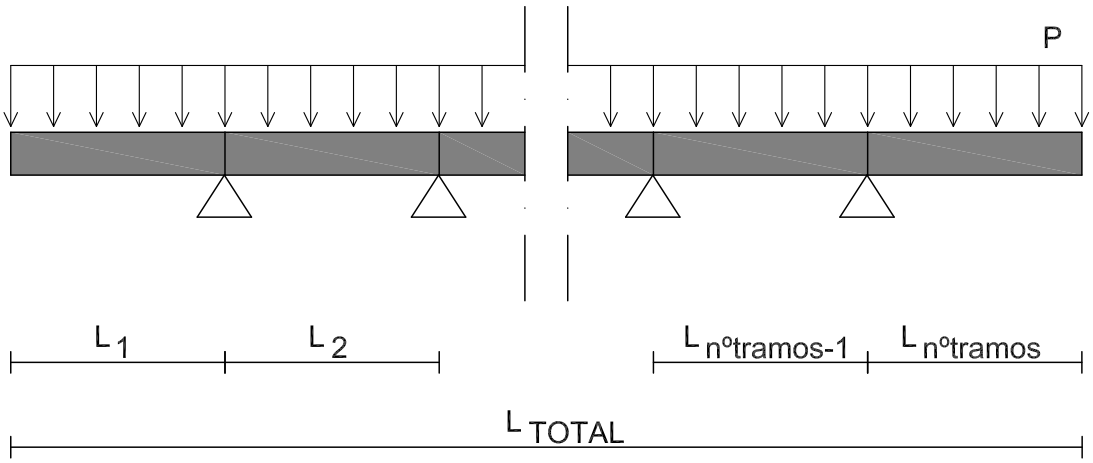
$$\mathbf{fe}^{(e)} = \mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{F}^{(e)} \tag{2.31}$$

   Em que cada valor do vector $\mathbf{fe}$ representa as forças internas no elemento finito, nos graus de liberdade respectivos. Obtidos os valores das forças é necessário ajustar os sinais para a convenção de sinais dos esforços internos pretendida.

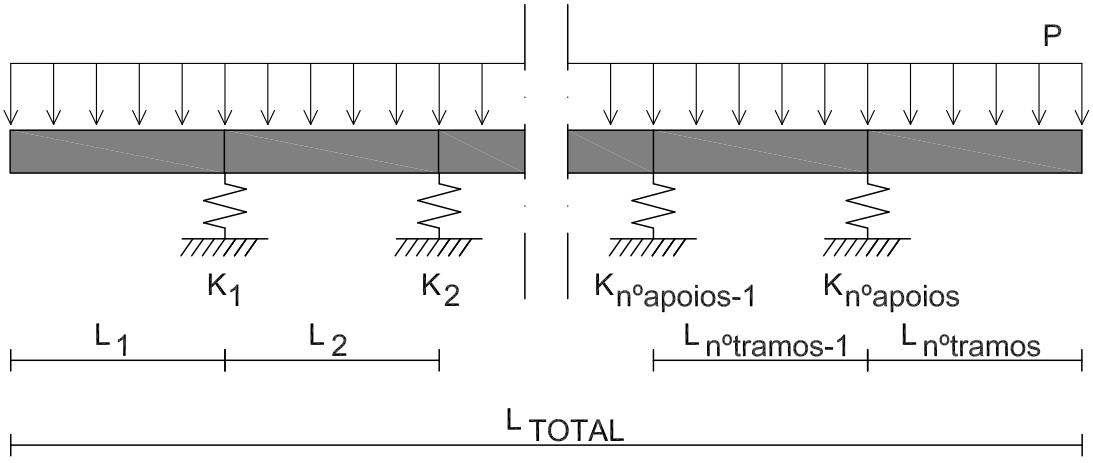
2.5 Implementação do método em MATLAB

2.5.1 Algoritmo para elementos de viga

De modo a realizar a análise estrutural de uma viga contínua, é criado um algoritmo de cálculo em MATLAB, que tem como base uma formulação em elementos finitos baseada em deslocamentos. O algoritmo permite modelar vigas contínuas, como ilustrado na Figura 2.6, com um comprimento total L e sujeitas a uma carga uniformemente distribuída P. O algoritmo permite a análise de vigas com diferentes vãos e diferentes tipos de apoio (rígidos ou flexíveis), como se indica na Figura 2.6.



(a) Apoios rígidos



(b) Apoios flexíveis

Figura 2.6: Viga contínua genérica

Como variáveis do problema consideram-se os comprimentos de cada tramo, à excepção do último, já que o seu valor é assegurado pela diferença entre o comprimento total da viga e o somatório dos restantes comprimentos dos tramos. No caso de vigas com apoios flexíveis a rigidez de cada apoio também é considerada como uma variável do problema. Deste modo, o número de variáveis do problema é obtido da seguinte forma:

$$X = [L_i, K_j] \qquad \text{com } i = 1, \ldots, n^{\text{o}}{}_{tramos} - 1 \text{ e } j = 1, \ldots, n^{\text{o}}{}_{apoios}$$

Adicionalmente é também necessário introduzir no algoritmo o módulo de elasticidade do material, E, bem como o momento de inércia da secção, I.

O código do algoritmo desenvolvido é apresentado no anexo A

2.5.2 Algoritmo para elementos de grelha

No âmbito desta dissertação, vão ser estudados dois tipos de grelha (Figura 2.7), e como tal, são criados dois algoritmos de cálculo em MATLAB com o objectivo de executar a análise estrutural de cada um dos tipos de grelha. A formulação dos algoritmos assenta numa formulação em elementos finitos baseada em deslocamentos.

As variáveis de projecto nos dois tipos de grelhas são os comprimentos de cada tramo na direcção x e na direcção y, traduzidos pela seguinte expressão:

$$X = [Lx_i, Ly_j] \quad \text{com} \quad i = 1, \ldots, n^{\text{o}}{}_{tramosx} \quad \text{e} \quad j = 1, \ldots, n^{\text{o}}{}_{tramosy}$$

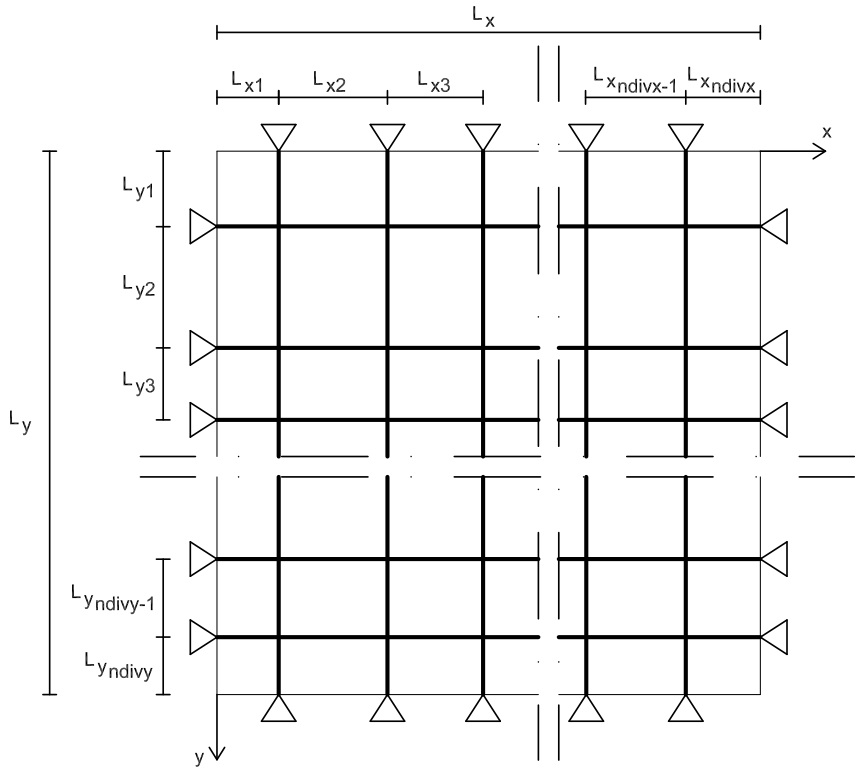
Para as grelhas do tipo I, o número de vigas em cada direcção é dado pelo número de tramos nessa direcção menos 1. Devido à existência de vigas de bordadura nas grelhas do tipo II, o número de vigas em cada direcção é dado pelo número de tramos nessa direcção mais 1.

Ambos os tipos de grelha admitem que num certo número de vigas descarrega uma carga de superfície, Q (kN/m^2), uniformemente distribuída no painel de laje com as dimensões $Lx_{total} \times Ly_{total}$. Para determinar a quantidade de carga que descarrega em cada viga, é necessário definir a largura de influência de cada viga. Definiu-se então, que ao longo de cada viga, a largura de influência mantém-se constante em todo o seu comprimento. Na Figura 2.8 são ilustradas as diferentes larguras de influência que variam consoante a localização da viga.

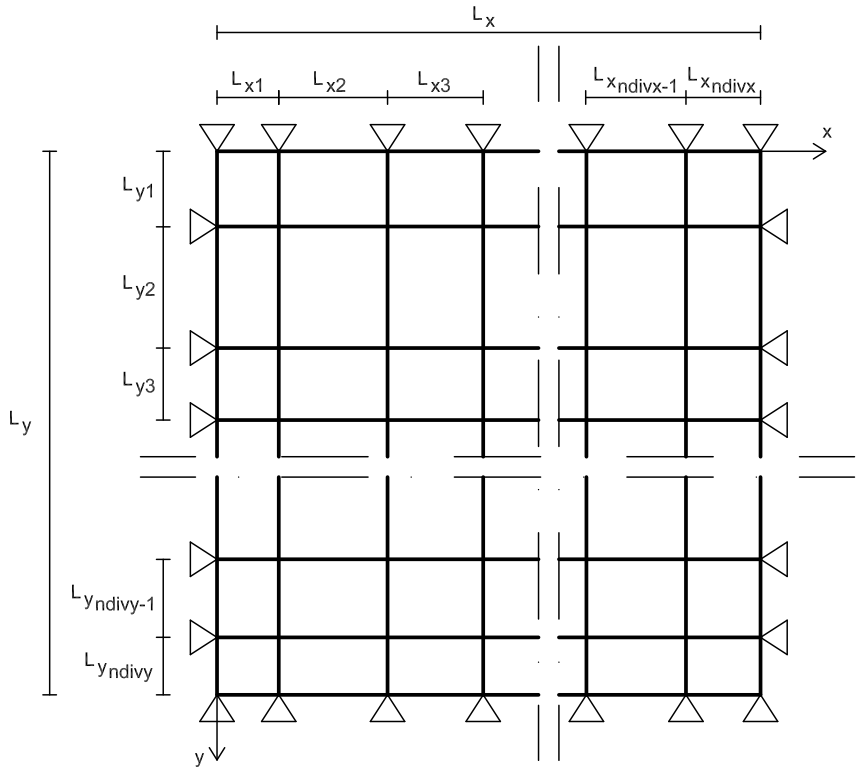
A carga uniformemente distribuída em cada viga, P_i, é calculada multiplicando a largura de influência da mesma pela carga Q. De notar que este valor deve ser dividido por 2 devido ao facto de a carga ser distribuída nas duas direcções.

Quanto às condições de apoio das grelhas são consideradas dois tipos de apoios para as grelhas do tipo I: simplesmente apoiados (grelha tipo I.a) e encastrados (grelha tipo I.b). Nas grelhas de tipo II são apenas considerados apoios simplesmente apoiados.

O código do algoritmo desenvolvido é apresentado no anexo B.

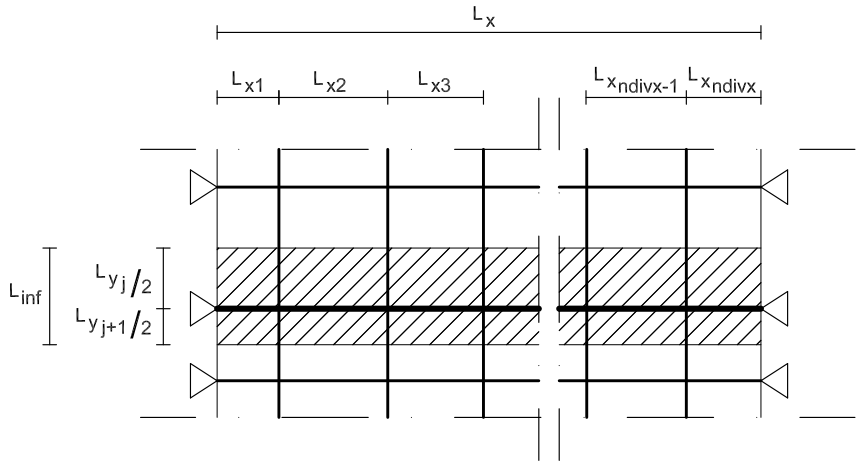


(a) Grelha tipo I

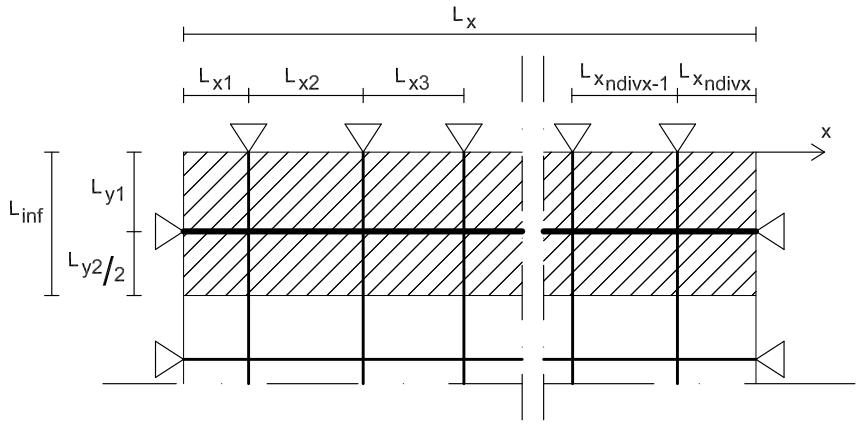


(b) Grelha tipo II

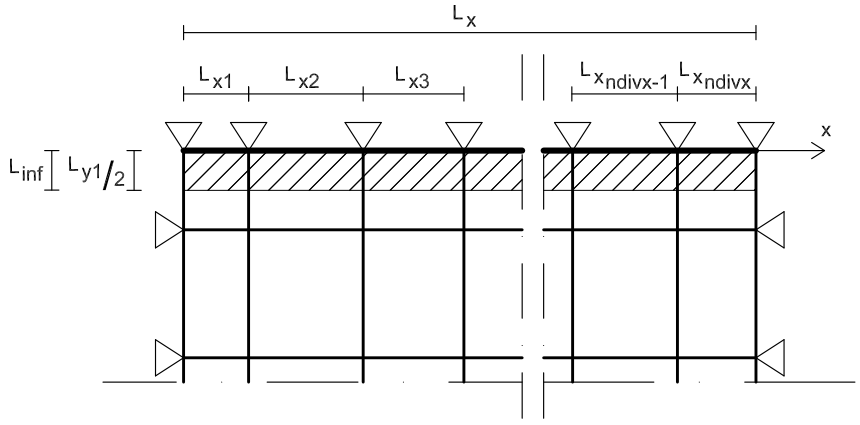
Figura 2.7: Sistema de grelhas



(a) Largura de influência das vigas interiores



(b) Largura de influência das vigas de extremidade



(c) Largura de influência das vigas de bordadura

Figura 2.8: Larguras de influência

Capítulo 3

Optimização estrutural

3.1 Introdução

A optimização estrutural é um conceito introduzido, em 1890, por Maxwell [28], que tem como objectivo achar o máximo ou mínimo de uma função que está relacionada com parâmetros característicos de uma estrutura. No seu trabalho, Maxwell decidiu optimizar uma ponte, estabelecendo que a solução óptima seria aquela que precisasse de menos material. Para tal, recorreu a métodos de cálculos analíticos. Michell [30], em 1904, retomou a ideia de optimização estrutural introduzida por Maxwell e resolveu vários projectos de optimização [26].

Contudo, a optimização estrutural só começou a ser utilizada com mais frequência a partir da década de 50, quando ocorreram desenvolvimentos significativos nos computadores digitais e surgiram métodos numéricos de programação matemática e de análise estrutural, como o método dos elementos finitos [6].

A optimização estrutural consiste então, em determinar um conjunto de parâmetros de uma estrutura, designados por variáveis de projecto, de modo a minimizar ou maximizar uma função, designada por função objectivo, respeitando um determinado número de restrições.

Normalmente a função objectivo de um problema de optimização estrutural está associada ao peso ou ao custo de uma estrutura. Contudo, existem outros parâmetros como os esforços ou a rigidez de uma estrutura que podem ter interesse num problema de optimização estrutural.

A optimização estrutural recorre a algoritmos de optimização que, através de um processo iterativo, vão alterando as variáveis de projecto, que por sua vez vão sendo analisadas, em cada iteração, por um modelo numérico de análise estrutural, do qual são obtidos valores da função objectivo. As restrições do problema são verificadas também pelo modelo numérico, que frequentemente recorre ao método dos elementos finitos. O processo iterativo termina quando se chega a uma solução de variáveis de projecto em que o valor da função objectivo verifica os critérios de convergência do algoritmo, atingindo-se assim a solução óptima do problema. Na Figura 3.1 é apresentado um fluxograma que ilustra o processo iterativo da optimização estrutural [2].

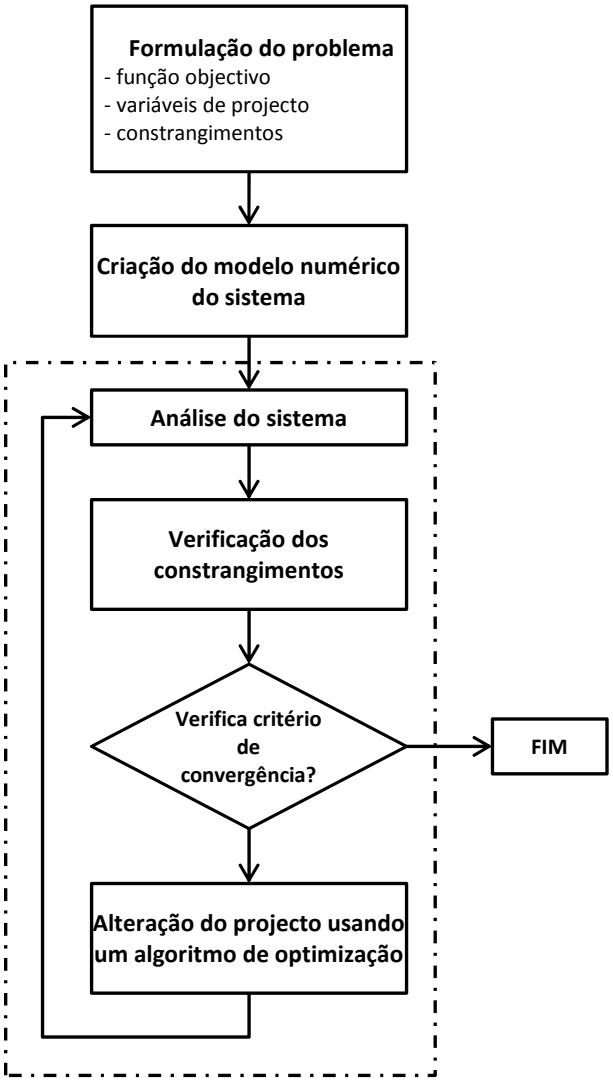


Figura 3.1: Fluxograma do processo iterativo de optimização estrutural

A formulação matemática de um problema de optimização estrutural pode ser escrita da seguinte forma:

$$
\begin{aligned}
\textbf{minimizar} \quad & f(\mathbf{x}) \\
\\
\textbf{sujeito a} \quad & g_j(\mathbf{x}) \leq 0 \qquad \text{com } j = 1, \ldots, m \\
& h_k(\mathbf{x}) = 0 \qquad \text{com } k = 1, \ldots, p \\
& x_i^l \leq x_i \leq x_i^u \quad \text{com } i = 1, \ldots, n
\end{aligned}
$$

Onde $\mathbf{x}$ é o vector das n variáveis de projecto, $f(\mathbf{x})$ é a função objectivo, $g_j(\mathbf{x})$ são as m restrições de desigualdade, $h_k(\mathbf{x})$ são as p restrições de igualdade, x_i^l e x_i^u são os limites inferior e superior das variáveis de projecto.

Num problema de optimização as funções objectivo e das restrições dependem apenas das variáveis de projecto, que por sua vez devem ser o mais independentes possíveis umas das outras. O número de

restrições de desigualdade não tem qualquer limite; contudo, é preciso ter cuidado com os seguintes casos relativos ao número de restrições de igualdade:

- Se $p > n$ - o sistema de equações é indeterminado.
- Se $p < n$ - é possível encontrar a solução óptima do problema.
- $p = n$ - o valor óptimo do problema é dada pela solução das restrições de igualdade.

Deste modo, o número de restrições de igualdade deve ser menor ou igual ao número de variáveis de projecto ($p \leq n$).

3.2 Tipos de optimização estrutural

De acordo com os parâmetros que se pretendem optimizar, um problema de optimização estrutural pode ser dividido em três tipos de optimização [2][6][26]:

- **Optimização de dimensões** (*sizing optimization*) - Este tipo de optimização consiste em optimizar parâmetros geométricos que caracterizam a estrutura, como por exemplo a altura e a espessura. Deste modo, as variáveis de projecto de um problema de optimização dimensional são as dimensões das secções transversais dos elementos da estrutura, mantendo a forma e a topologia da estrutura inalteradas durante o processo de optimização. As variáveis de projecto na optimização de dimensões são frequentemente discretas, pois podem estar associadas a secções normalizadas disponíveis no mercado. Considere-se o exemplo, ilustrado na Figura 3.2, de uma viga encastrada com uma carga concentrada aplicada na sua extremidade, em que se pretende optimizar a massa da estrutura.

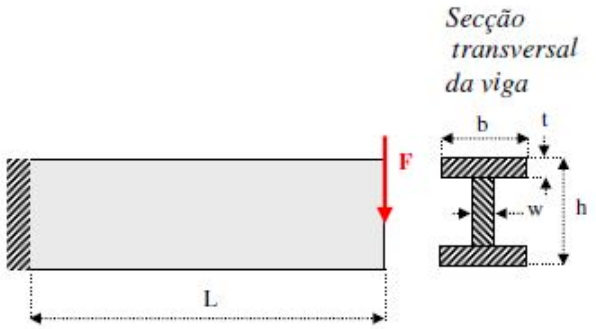


Figura 3.2: Optimização de dimensões de uma consola (adaptado de [6])

Ao se considerar uma optimização de dimensões, as variáveis de projecto são os parâmetros h, b, t e w, enquanto o comprimento da viga, L, se mantém inalterado.

- **Optimização de forma** (*shape optimization*) - Este tipo de optimização consiste em variar a fronteira Γ delimitadora do domínio Ω que a estrutura ocupa. Esta fronteira pode ser definida por um conjunto de pontos, linhas ou superfícies. Numa optimização de forma, as variáveis de

projecto são então as coordenadas desses pontos, mantendo inalteradas as dimensões das secções e a topologia da estrutura. Na optimização de forma as variáveis de projecto são contínuas e podem também ser designadas por variáveis geométricas. Tomando como exemplo o caso anterior da consola, a optimização de forma modifica as fronteiras do domínio ocupado pela viga, que inicialmente é rectangular, ou modifica as fronteiras da secção transversal, que inicialmente é em I, como é ilustrado na Figura 3.3.

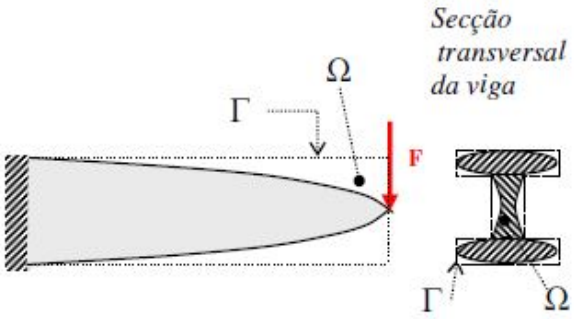


Figura 3.3: Optimização de forma de uma consola (adaptado de [6])

- **Optimização topológica** (*topology optimization*) - Este tipo de optimização consiste em encontrar a topologia da estrutura que minimiza a função objectivo, determinando em cada ponto do domínio da estrutura se deve ou não haver material. Usualmente este tipo de optimização recorre a uma malha de elementos finitos para discretizar o domínio da estrutura; cada elemento vai recebendo ao longo do processo de optimização valores de 1, para um estado de existência, e de 0, para um estado de ausência. No caso de estruturas discretas procura-se obter a ordem espacial dos elementos, enquanto, no caso de estruturas contínuas procura-se determinar a localização e geometria óptimas das cavidades do domínio da estrutura. Na Figura 3.4 mostra-se a solução óptima do exemplo anterior da consola, para uma optimização topológica.

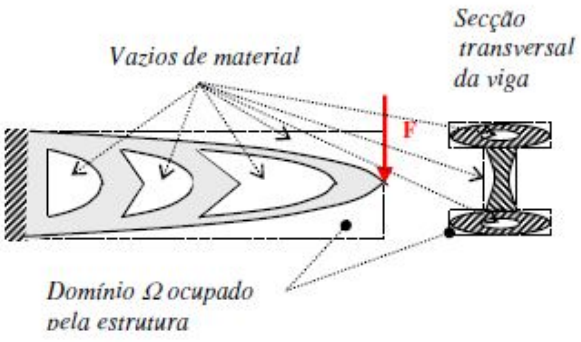


Figura 3.4: Optimização topológica de uma consola (adaptado de [6])

De entre os três tipos de optimização estrutural considerados, a optimização topológica é aquela que consegue obter melhores soluções quando se pretende optimizar o peso da estrutura.

Pode-se ainda considerar uma optimização estrutural que utilize pelo menos dois dos três tipos apresentados. Este tipo de optimização designa-se por optimização híbrida.

3.3 Formulação de um problema de optimização

Uma definição e formulação apropriadas de um problema de optimização são essenciais para que a sua resolução forneça resultados aceitáveis. As soluções óptimas de um problema de optimização serão tão boas quanto a sua formulação. Deste modo, torna-se indispensável a definição de um procedimento que formule um problema de optimização. Arora [2] define um procedimento constituído por cinco passos:

1. **Enunciado do problema/projecto**

   O procedimento começa com a criação do enunciado do problema/projecto, onde são descritos os objectivos gerais e os requerimentos a serem cumpridos.

2. **Recolha de informação e dados**

   Para o desenvolvimento de um problema de optimização, à semelhança de um problema matemático tradicional, é necessário a recolha de dados e informações para que seja possível a sua compreensão e resolução.

3. **Identificação/definição das variáveis de projecto**

   Neste passo é necessário identificar as variáveis de projecto que descrevem o sistema. As variáveis de projecto podem ser de diferentes tipos, estando relacionadas com o tipo de problema de optimização a realizar. Na Tabela 3.1 é feita uma relação entre os diferentes tipos de variáveis com os diversos tipos de optimização estrutural.

Tabela 3.1: Relação do tipo de variáveis com a formulação do problema [6]

Tipo de optmização	**Tipo da variável de projecto**	**Significado físico possível**	**Observação**
Dimensional	Inteira	Selecção de secções transversais	Procura através de uma tabela de varlores normalizados
	Real	Dimensões válidas num intervalo de variação contínuo	-
Forma ou configuração	Geométrica (real)	Coordenadas nodais	Posição de nós varia entre um limite inferior e superior
Dimensional	Booleana	Existência ou ausência de material	-
	Real	Densidade do material variável	

Como já foi referido, as variáveis de projecto devem ser o mais independentes possível umas das outras. No entanto, caso haja dependência, é necessário definir relações entre elas. O número de variáveis de projecto deve ser o mais reduzido possível de forma a facilitar o processo de cálculo.

4. **Identificação da função objectivo**

   Neste passo é identificado o parâmetro que se pretende optimizar, ou seja, identificar a função objectivo do problema. Uma função objectivo válida deve estar relacionada directa ou indirectamente com as variáveis de projecto do problema. A função objectivo deve ser uma função escalar da qual é obtido um valor numérico quando são especificados valores para as variáveis de projecto. Considera-se que um projecto óptimo é aquele que apresenta o melhor valor de função objectivo.

   Os parâmetros mais usuais que se pretendem optimizar são: o custo (minimizar), o lucro (maximizar), o peso (minimizar), os esforços (minimizar), entre outros. Em algumas situações, os problemas de optimização podem conter duas ou mais funções objectivos a optimizar, designados por problemas de optimização multiobjectivo.

5. **Identificação das restrições**

   O último passo do processo de formulação é identificar as restrições a que o problema está sujeito, desenvolvendo expressões para elas. Todas as restrições têm de depender pelo menos de uma variável de projecto.

   Como já foi referido, as restrições podem ser de igualdade ou de desigualdade. As restrições do problema podem ser lineares, caso sejam funções cujos termos das variáveis sejam de primeira ordem, ou não lineares.

   Algumas das restrições são bastante fáceis de identificar, como por exemplo os valores máximos e mínimos das variáveis. No entanto, existem outras restrições mais complexas que estão relacionadas com as variáveis de forma indirecta. A este tipo de restrições dá-se o nome de restrições implícitas, pois estas não podem ser expressas em função das variáveis de forma explícita, sendo necessário a utilização de variáveis intermédias para a formulação do problema.

3.4 Mínimo global e local

No âmbito da optimização é relevante apresentar os conceitos de mínimos globais e locais de uma função.

Uma função $f(\mathbf{x})$ de n varáveis tem um mínimo global (absoluto) em $\mathbf{x}^*$ se

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad , \forall \mathbf{x} \in S \tag{3.1}$$

Em que S é a região admissível do espaço de projecto. Caso $f(\mathbf{x}^*) < f(\mathbf{x})$, então diz-se que $\mathbf{x}^*$ é um mínimo global forte (estrito) da função; caso contrário trata-se de um mínimo global fraco.

Uma função $f(\mathbf{x})$ de n varáveis tem um mínimo local (relativo) em $\mathbf{x}^*$ se

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad , \forall \mathbf{x} \in N \tag{3.2}$$

Em que N é uma vizinhança pequena de $\mathbf{x}^*$ no espaço admissível S. A vizinhança N é definida como o seguinte conjunto de pontos:

$$N = \{\mathbf{x} | \mathbf{x} \in S, ||\mathbf{x} - \mathbf{x}^*|| < \delta\} \tag{3.3}$$

Para um pequeno $\delta > 0$. Caso $f(\mathbf{x}^*) < f(\mathbf{x})$, então diz-se que $\mathbf{x}^*$ é um mínimo local forte (estrito) da função; caso contrário trata-se de um mínimo local fraco.

Uma função pode ter um mínimo global estrito em apenas um ponto; no entanto, poderá ter um mínimo global em diversos pontos se tiver o mesmo valor nesses pontos. De maneira semelhante, uma função $f(\mathbf{x})$ apenas pode ter um mínimo local estrito num ponto, contudo, poderá ter um mínimo local em diversos pontos em N se o valor da função for o mesmo nesses pontos. Na Figura 3.5 é apresentado um exemplo de uma função de uma variável com um mínimo local (ponto A) para a região N e um mínimo global (ponto B) para a região S.

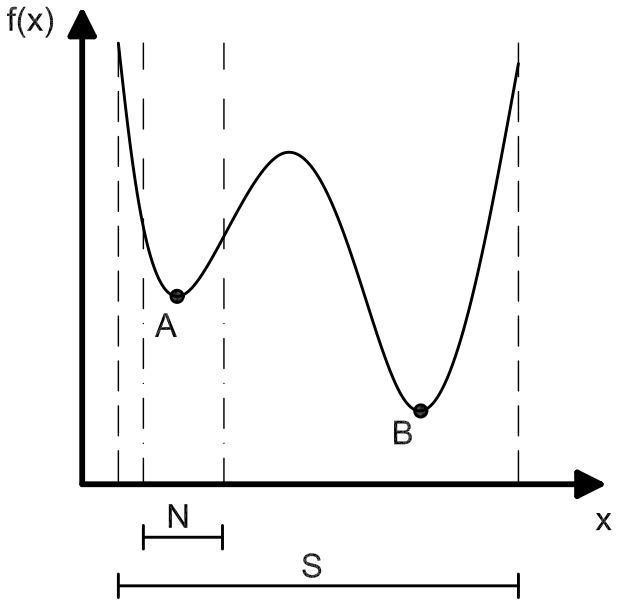


Figura 3.5: Exemplo de mínimos global e local de uma função

3.5 Métodos de optimização não-linear

Neste capítulo abordam-se diferentes métodos de optimização não-linear. Uma optimização diz-se não-linear quando a função objectivo e/ou as restrições da mesma contêm funções não lineares nas variáveis.

Este tipo de optimização utiliza métodos iterativos para encontrar soluções para o problema. Os métodos iterativos geram uma sucessão de soluções que se espera que convirja para a solução óptima do problema, ao invés de resolver o problema analiticamente como na optimização linear.

Existem diversos métodos de optimização não-linear, não sendo a priori evidente identificar aquele que melhor permite verificar as condições de optimalidade, pelo que a escolha do método a utilizar nem sempre é fácil de tomar. A especificidade do problema de optimização é também um factor a ter em conta quando se opta por um método em detrimento de outro. Na Figura 3.6 são apresentados diversos métodos de optimização não-linear.

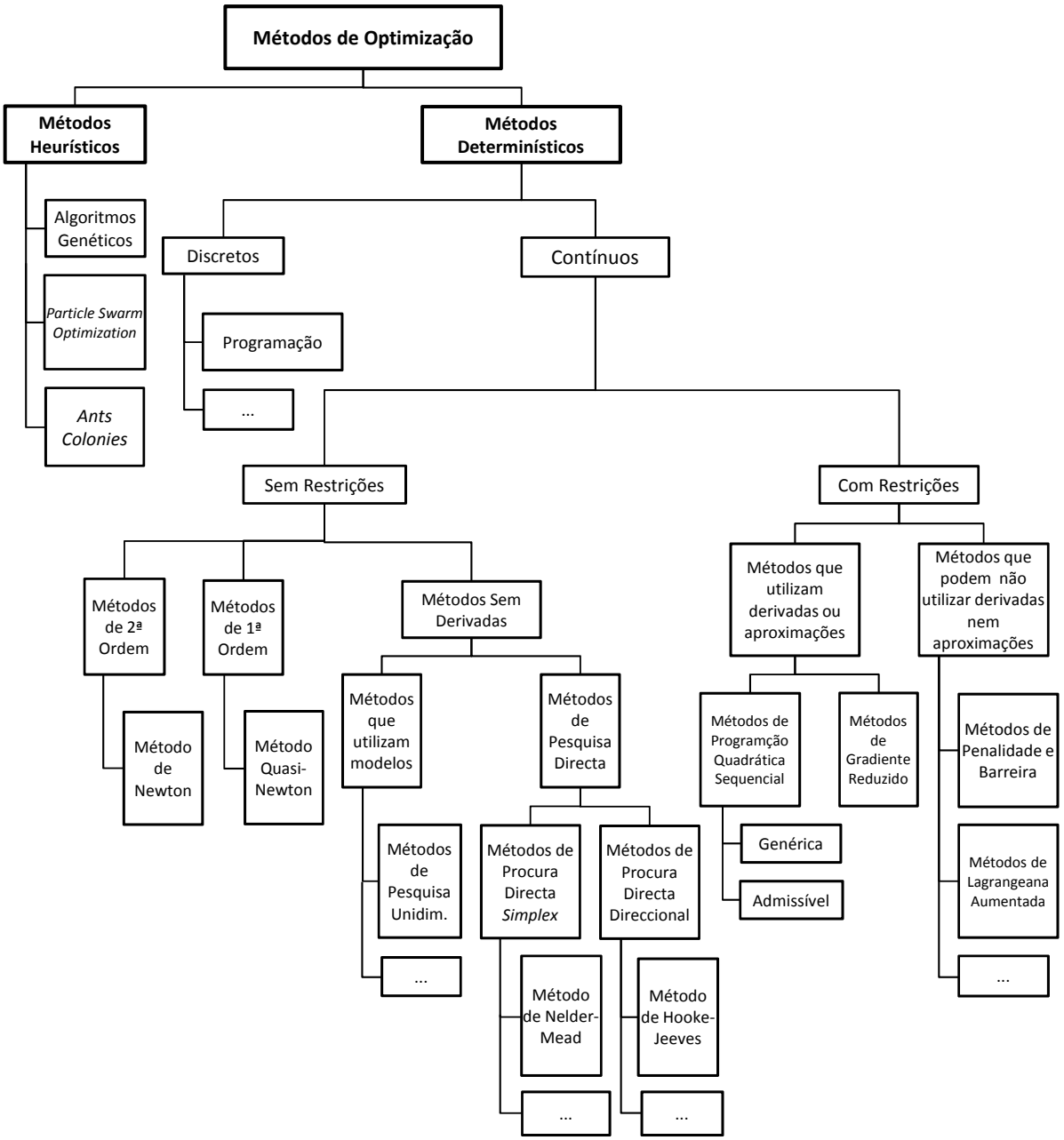


Figura 3.6: Esquema com diversos métodos de optimização não-linear (adaptado de [11])

De uma forma geral, os métodos de optimização não-linear classificam-se em:

- Métodos de Optimização Heurísticos ou Estocásticos - Estes métodos recorrem a gerações de números aleatórios, que por sua vez tentam convergir para a solução óptima do problema inspirando-se em comportamentos existentes na natureza. Segundo Conn *et. al* [9] estes métodos devem ser utilizados como último recurso e em problemas onde o universo de soluções é necessariamente grande, complexo ou não se conhece bem. Contudo, outros autores têm utilizado estes métodos com sucesso [11].

- Métodos de Optimização Determinísticos - Estes métodos baseiam-se na sua maioria em algoritmos clássicos, gerando uma sequência determinística de possíveis soluções. Nestes métodos é necessário fornecer um ponto de partida inicial, com o qual o método tentará convergir para um mínimo, que pode não ser um mínimo global da função objectivo [11].

Os métodos de optimização podem também dividir-se em métodos de procura local, onde estão inseridos os métodos de procura directa, nomeadamente o Nelder-Mead [31], e métodos de procura global, como por exemplo os algoritmos genéticos. Os métodos locais utilizam algoritmos que convergem para um mínimo de uma determinada região da função, enquanto os métodos globais usam algoritmos que convergem para o mínimo global da função objectivo. No entanto, é possível achar os mínimos globais da função objectivo utilizando apenas métodos de procura local.

3.5.1 Métodos de optimização de procura directa

Os métodos de procura directa são actualmente muito utilizados, pois não necessitam de informação sobre o gradiente da função objectivo. Estes métodos surgiram na década de 50, mas mais tarde foram criticados por parte da comunidade científica pois nem sempre garantem a convergência da solução e por esta ser geralmente mais lenta quando comparada com os métodos de gradiente.

Existem diversos métodos de optimização de procura directa, mas o procedimento comum que os rege é o seguinte:

1. Introduzir um ponto inicial que pertença ao domínio da função.

2. Calcular uma quantidade de pontos da função em diferentes direcções e a uma determinada distância do ponto inicial. É neste passo que os diversos métodos diferem uns dos outros.

3. Comparar os valores obtidos com os anteriores; caso o novo valor seja menor que o anterior, o novo valor passa a ser um ponto inicial; caso contrário altera-se a distância e repete-se a avaliação dos valores da função objectivo.

4. A paragem do método é garantida pelo critério de convergência fornecida pelo utilizador. Este critério relaciona-se com o nível de redução da distância entre o ponto inicial e os pontos que estão a ser avaliados.

O principal problema atribuído a estes métodos está essencialmente relacionado com o facto de estes frequentemente convergirem para mínimos locais. Considere-se por exemplo a função de Rastrigin dada pela seguinte expressão:

$$f(x,y) = 20 + x^2 + y^2 - 10\left(\cos\left(2\pi x\right) + \cos\left(2\pi y\right)\right) \tag{3.4}$$

Na Figura 3.7 está representada graficamente a equação de Rastrigin, onde se pode verificar que a função tem diversos mínimos locais e apenas um global no ponto de coordenadas $(0,0)$.

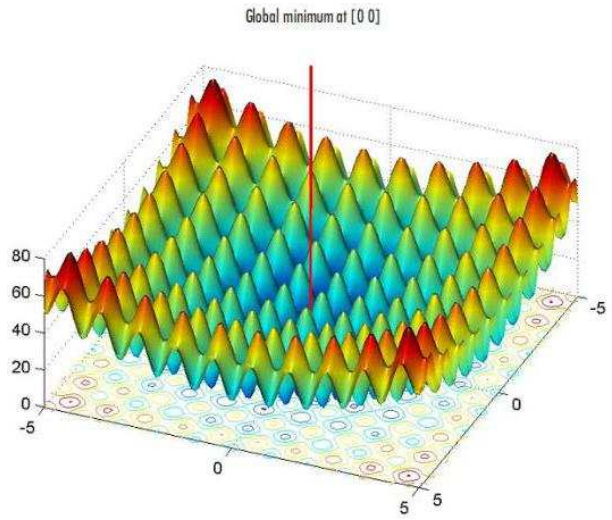


Figura 3.7: Gráfico da função de Rastrigin (adaptado de MATLAB)

Ao resolver este problema através de um método de procura directa utilizando um ponto inicial fora do cone onde está localizado o mínimo global, verificar-se-ia que a solução iria descer e permanecer "presa"num mínimo local. Desta forma, apenas é possível atingir a convergência global se o ponto inicial estiver na região onde está localizado o mínimo global da função objectivo. Para solucionar este tipo de problema pode-se tentar dispersar pontos pela função para garantir uma boa cobertura no domínio da função, ou então optar por utilizar um outro algoritmo de optimização.

Outro problema inerente a estes métodos prende-se com a lentidão da convergência, principalmente se são se conhecer o valor adequado do passo a utilizar. Se o valor do passo utilizado for muito pequeno, o algoritmo precisa de muitas iterações para convergir; no entanto, se for muito grande, os pontos a serem analisados ficam fora do domínio da função objectivo e o algoritmo não inicia.

Normalmente estes métodos são utilizados para problemas de optimização sem restrições e, segundo Lewis *et al* [25], podem ser divididos em três tipos:

- Métodos de pesquisa em padrão;
- Métodos *simplex*;
- Métodos com conjuntos de direcções de pesquisa adaptativas.

No âmbito desta dissertação apenas será abordado o método *simplex* de Nelder-Mead [31].

Método de Nelder-Mead

O método de Nelder-Mead [31], também conhecido como *downhill simplex* ou *amoeba*, consiste num método de procura directa sem restrições, introduzido em 1965 por John Nelder e Roger Mead. Este método consiste em criar um poliedro, designado de *simplex*, com $n+1$ vértices, em que n é o número de variáveis, e contraí-lo ou expandi-lo através do valor que a função obtém nesses vértices.

Para que seja possível aplicar o método é necessário especificar quatro parâmetros escalares, que são os coeficientes de reflexão (ρ), expansão (χ), contracção (γ) e encolhimento (σ). Segundo os autores tais parâmetros devem satisfazer as seguintes condições:

$$\rho > 0, \quad \chi > 1, \quad \chi > \rho, \quad 0 < \gamma < 1 \quad \text{e} \quad 0 < \sigma < 1$$

Segundo Lagarias *et al* (1998) os valores *standard* para esses coeficientes são:

$$\rho = 1, \quad \chi = 2, \quad \gamma = 1/2 \quad \text{e} \quad \sigma = 1/2$$

O algoritmo de Nelder-Mead, para uma dada iteração k pode ser descrito pelos seguintes passos:

1. **Ordenação** - Ordenar os $n+1$ vértices do *simplex* satisfazendo a condição:

$$f(x_1) < f(x_2) < \ldots < f(x_{n+1}) \tag{3.5}$$

2. **Reflexão** - Calcular o ponto de reflexão, x_r, pela seguinte expressão:

$$x_r = \bar{x} + \rho(\bar{x} - x_{n+1}) = (1+\rho)\bar{x} - \rho x_{n+1} \tag{3.6}$$

   Em que $\bar{x}$ é a média dos n melhores pontos do *simplex* (à excepção do ponto $n+1$). Avalia-se $f_r = f(x_r)$ e caso $f_1 < f_r < f_n$, o ponto é aceite e a iteração termina.

3. **Expansão** - Se $f_r < f_1$, é calculado o ponto de expansão x_e,

$$x_e = \bar{x} + \chi(x_r - \bar{x}) = \bar{x} + \rho\chi(\bar{x} - x_{n+1}) = (1+\rho\chi)\bar{x} - \rho\chi x_{n+1} \tag{3.7}$$

   E calcula-se o valor da função nesse ponto, $f_e = f(x_e)$. Caso $f_e < f_r$, aceita-se x_e e dá-se por terminada a iteração; caso contrário, aceita-se x_r.

4. **Contracção** - Se $f_r \geq f_n$, efectua-se uma contracção entre $\bar{x}$ e o melhor valor de x_{n+1} e x_r. A contracção pode ser:

   (a) **Para fora** - Esta contracção ocorre caso $f_n \leq f_r < f_{n+1}$. É então calculado:

$$x_c = \bar{x} + \gamma(x_r - \bar{x}) = \bar{x} + \gamma\rho(\bar{x} - x_{n+1}) = (1+\rho\gamma)\bar{x} - \rho\gamma x_{n+1} \tag{3.8}$$

   E avalia-se o valor $f_c = f(x_c)$. Caso $f_c \leq f_r$, aceita-se x_c e dá-se por terminada a iteração, caso contrário, passa-se para o passo 5.

(b) **Para dentro** - Se $f_r \geq f_{n+1}$, ocorre uma contracção para dentro. É então calculado:

$$x_{cc} = \bar{x} - \gamma(\bar{x} - x_{n+1}) = (1-\gamma)\bar{x} + \gamma x_{n+1} \tag{3.9}$$

E avalia-se o valor $f_{cc} = f(x_{cc})$. Caso $f_{cc} \leq f_{n+1}$, aceita-se x_{cc} e dá-se por terminada a iteração, caso contrário, passa-se para o passo 5.

5. **Encolhimento** - Este passo consiste em diminuir as dimensões do *simplex*, alterando as coordenadas de n pontos. As novas coordenadas são calculadas da seguinte forma:

$$v_i = x_1 + \sigma(x_i - x_1), \quad \text{com} \quad i = 2, ..., n+1 \tag{3.10}$$

São então calculados os valores da função objectivo nesses novos pontos e procede-se a nova iteração.

Nas Figuras 3.8 e 3.9 são representados os movimentos descritos anteriormente para um *simplex* triangular, ou seja, com duas variáveis.

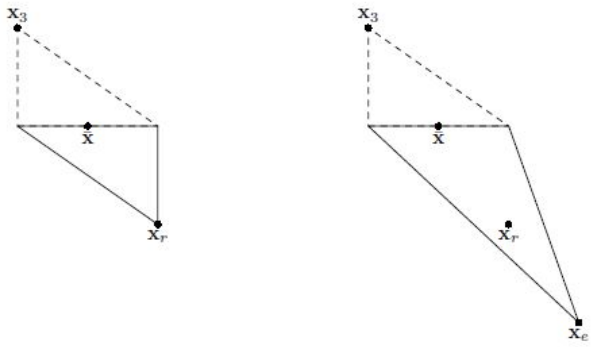


Figura 3.8: Forma do *simplex* após a reflexão e expansão (adaptado de [24])

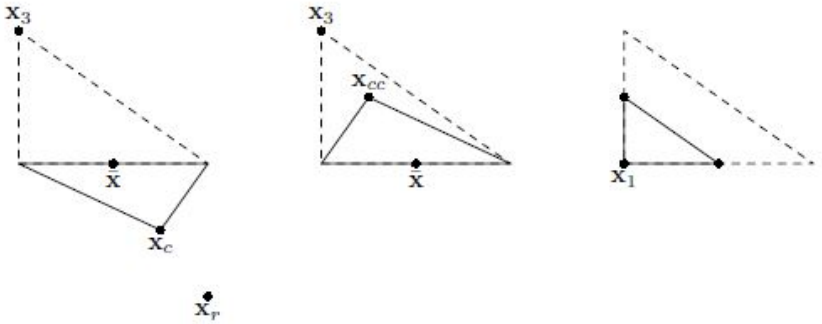


Figura 3.9: Forma do *simplex* após a contracção, para fora e para dentro, e encolhimento (adaptado de [24])

Para uma melhor compreensão do método, é apresentado na Figura 3.10, um fluxograma que resume o procedimento do método de Nelder-Mead.

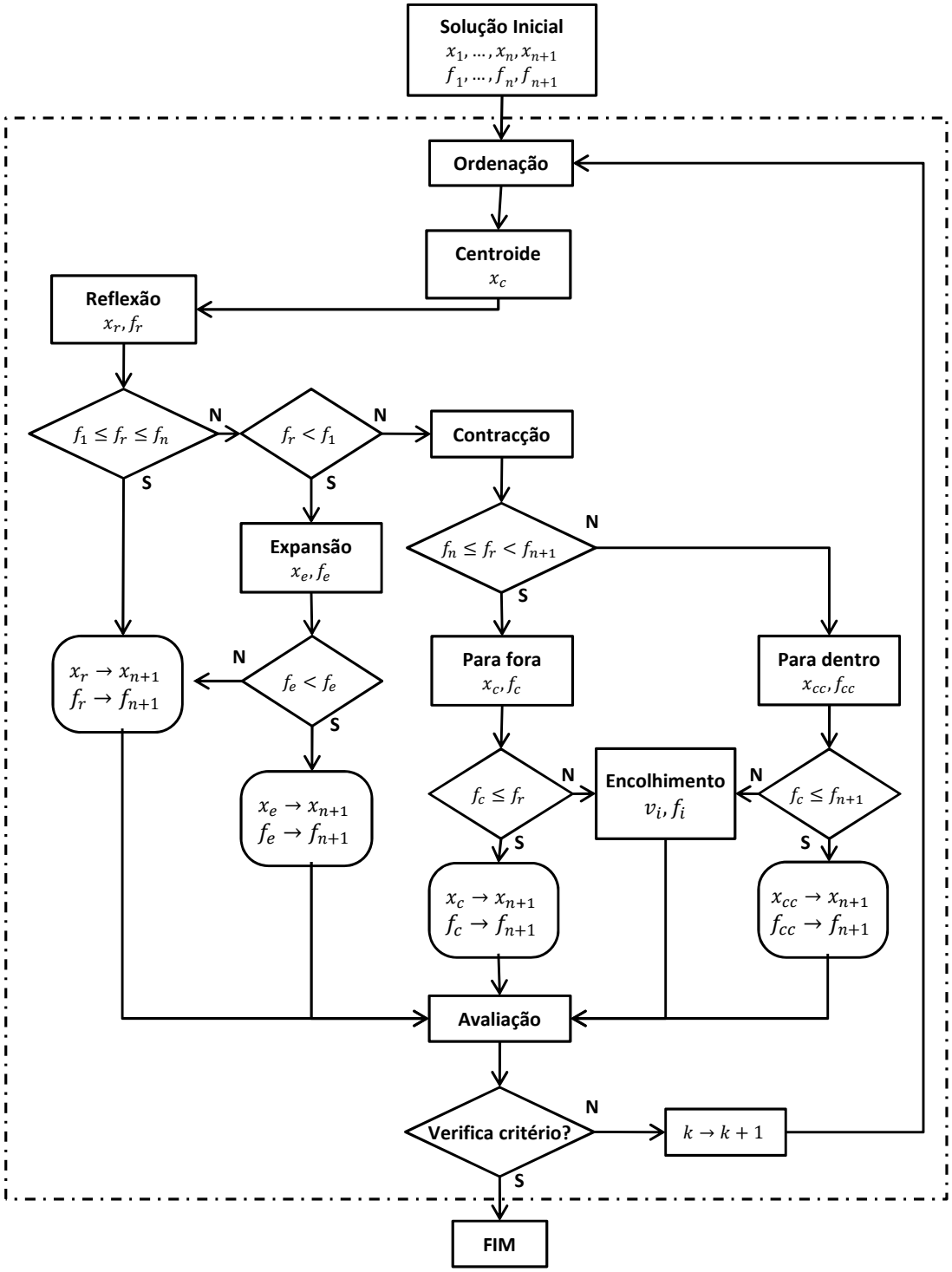


Figura 3.10: Fluxograma do método de Nelder-Mead

A convergência desse método é um assunto algo controverso, pois apesar de Lagarias *et al* [24] afirmar em que o método converge para um determinado número de funções em uma e duas dimensões, McKinnon [29] mostra que o método pode convergir para um não mínimo em diversos problemas. Contudo, este método é bastante utilizado por cientistas e engenheiros para resolução de problemas de optimização.

3.5.2 Métodos de optimização heurísticos e meta-heurísticos

Os métodos heurísticos, usualmente também designados como optimização combinatória, surgiram durante o inicio da década de 80, quando começou a haver um desenvolvimento de métodos alternativos que fossem capazes de solucionar problemas de optimização discretos, uma vez que a continuidade da solução objectivo não é garantida. Estes métodos operam a partir da criação de variáveis pseudo-aleatórias que, através de processos iterativos, tentam atingir a solução óptima da função objectivo utilizando apenas as informações contidas nesta, não precisando de informações sobre a sua continuidade ou diferenciabilidade.

Uma heurística, é então, um método que vai calculando o valor de uma função objectivo em pontos promissores, sendo estes identificados e avaliados ao longo de um processo iterativo. Quando este processo acaba, é encontrada uma boa solução (quase óptima), que eventualmente será o mínimo da função, mas sem garantias disso [6].

Meta-heurísticos são uma classe de métodos aproximados, que são utilizados em problemas de optimização complicados onde os métodos heurísticos clássicos deixam de ser eficazes e eficientes. Nestes métodos aproximados estão inseridos os algoritmos genéticos, a pesquisa *tabu*, as colónias de formiga, entre outros. Estes métodos baseiam-se em diversos conceitos como evolução das espécies, inteligência artificial, sistema nervoso ou estatística [32].

Define-se meta-heurística como um processo iterativo de geração de soluções, que utiliza heurísticas subordinadas, combinando diferentes conceitos de pesquisa e exploração do espaço de soluções com o objectivo de encontrar uma solução quase óptima [32].

Alguns dos aspectos que diferenciam as meta-heurísticas das heurísticas são os seguintes:

- Uma heurística é dependente da especificidade do problema que resolve. Uma meta-heurística tem um espectro de aplicação mais alargado.
- Uma heurística assenta em procedimentos iterativos que terminam quando não é encontrada uma solução que melhore a anterior. Uma meta-heurística incorpora estratégias de modo a explorar o espaço de soluções para além da optimalidade local.

Algoritmos genéticos

Os algoritmos genéticos foram apresentados por Holland [17] e têm como base de funcionamento a teoria de Darwin sobre a evolução das espécies, a qual afirma que as espécies evoluem de acordo com a capacidade de adaptação dos seus indivíduos ao meio ambiente que os rodeia, e que aqueles que estiverem melhor adaptados têm mais hipóteses de se reproduzir e transmitir os seus genes às gerações seguintes.

É comum utilizar-se uma terminologia diferente entre os algoritmos de optimização clássicos e os algoritmos genéticos. Na Tabela 3.2 é apresentada a correspondência entre as duas terminologias:

Tabela 3.2: Correspondência entre terminologias (adaptado de [6])

Optimização clássica	**Algoritmos genéticos**
Solução no domínio	Indivíduo
Conjunto de soluções	População
Função objectivo	Aptidão
Iteração	Geração

Os algoritmos genéticos operam, em primeira instância, com uma população inicial de indivíduos gerada aleatoriamente dentro do domínio da função. Cada individuo contém um cromossoma cuja informação genética é única. Nos algoritmos genéticos básicos essa informação é codificada através de uma codificação binária, na qual os genes que constituem o cromossoma podem tomar o valor de 0 ou 1, como ilustrado na Figura 3.11. Pretende-se que a população inicial esteja bem distribuída no domínio da função de forma a permitir uma pesquisa inicial alargada do mesmo, e deste modo acelerar a convergência do método, uma vez que aumenta as probabilidades de um dos indivíduos iniciais estar perto da solução óptima.

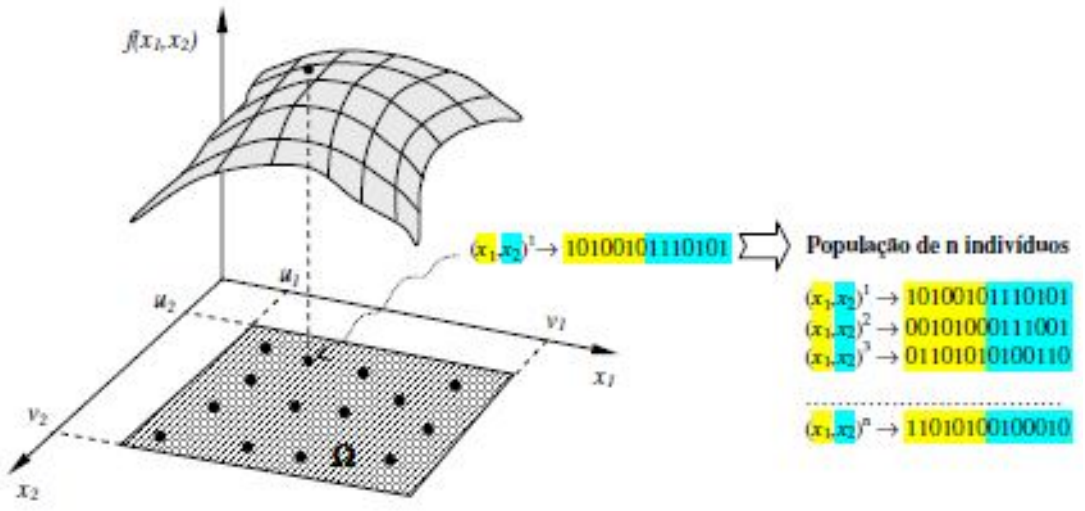


Figura 3.11: Exemplo de codificação binária de uma população de indivíduos (adaptado de [6])

Com a codificação feita, é então efectuada a selecção dos indivíduos da população com melhor aptidão e com maior probabilidade de se reproduzirem e transmitir os seus genes aos seus descendentes. Diz-se que um indivíduo com melhor aptidão é aquele que apresenta menor valor da função objectivo. A selecção é feita através do operador selecção, que funciona de forma a que os indivíduos com melhor aptidão

tenham uma maior probabilidade de serem seleccionados para reprodução. O operador selecção agrupa os indivíduos aos pares para se dar a reprodução.

No operador selecção, correspondente ao método da roleta, a probabilidade P_i do i-ésimo indivíduo da população ser seleccionado para reprodução é igual a:

$$P_i = \frac{F_i}{Q}; \quad Q = \sum_{j=1}^{N_P} F_j \tag{3.11}$$

Em que F_i é o valor da aptidão do indivíduo i e N_P é o número total de indivíduos da população. Para explicar o processo de selecção, considere-se o método da roleta ilustrado na Figura 3.12. A roleta tem N_P segmentos que cobrem toda a população e o tamanho de cada segmento é proporcional à probabilidade P_i do respectivo indivíduo. É gerado um número aleatório, w, com o valor entre 0 e 1 e a roleta gira no sentido horário, com uma rotação proporcional a w. Quando a roleta acaba de girar, o segmento para o qual a seta aponta é seleccionado para reprodução. No caso ilustrado na Figura 3.12, é o segmento 2 que é seleccionado para reprodução. Como o tamanho dos segmentos da roleta é definido de acordo com a probabilidade P_i, o processo de selecção tende para que os indivíduos com melhor aptidão sejam seleccionados. Os indivíduos seleccionados continuam a ser passíveis de serem novamente seleccionados, o que faz com que a nova população possa conter membros semelhantes e não conter alguns indivíduos da população anterior. Deste modo, a aptidão média da população aumenta a cada geração [2].

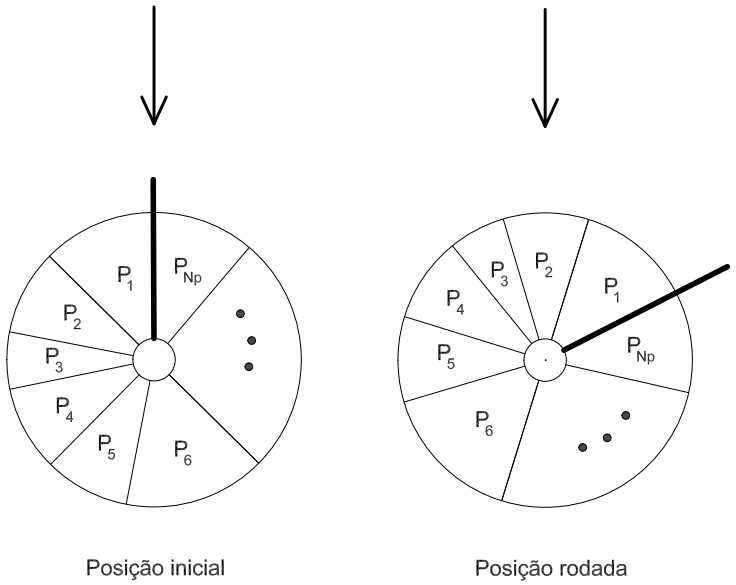
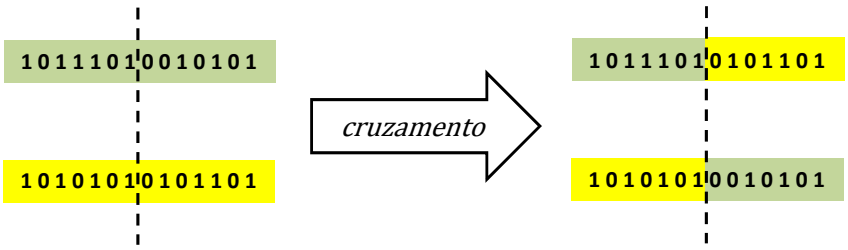
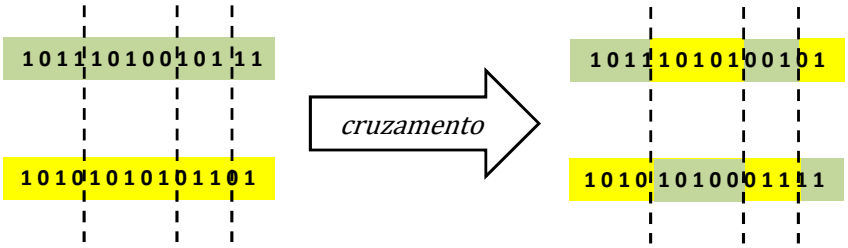


Figura 3.12: Método da roleta para a selecção dos indivíduos (adaptado de [2])

A reprodução é feita com recurso aos operadores cruzamento e mutação. A função do operador cruzamento é cruzar a informação genética de vários indivíduos, permitindo assim que a informação genética dos indivíduos descendentes contenha parte da informação de cada um dos indivíduos progenitores. Na Figura 3.13 são ilustradas as formas de funcionamento do operador cruzamento.



(a) Cruzamento de ponto único



(b) Cruzamento multiponto

Figura 3.13: Funcionamento do operador cruzamento (adaptado de [6])

O cruzamento de ponto único é usado nos algoritmos genéticos simples. Neste tipo de cruzamento é escolhido aleatoriamente um ponto no cromossoma dos progenitores, a partir do qual são trocados os genes, como ilustrado na Figura 3.13(a). Em algoritmos genéticos mais evoluídos. é utilizado um cruzamento multiponto, em que os cromossomas são cortados em vários pontos e não em apenas um, conforme ilustrado na Figura 3.13(b).

Tal como na teoria de Darwin, o número da população de indivíduos nos algoritmos genéticos mantém-se inalterado ao longo das várias gerações, ou seja, n indivíduos progenitores reproduzem n descendentes.

A mutação dá-se após o cruzamento e consiste em modificar aleatoriamente os genes dos cromossomas dos descendentes, utilizando-se para tal o operador mutação. O modo de funcionamento do operador mutação dos genes é ilustrado na Figura 3.14.

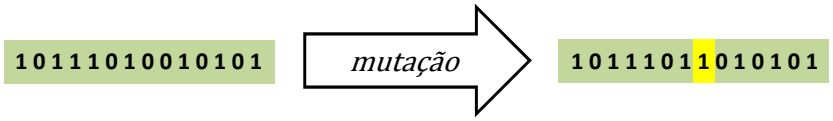


Figura 3.14: Funcionamento do operador mutação (adaptado de [6])

O operador mutação introduz alguma diversidade entre os indivíduos da população, permitindo uma melhor exploração do domínio, algo que com o operador cruzamento, só por si, não é possível. No entanto, a mutação dos genes dos indivíduos tem de ser controlada de modo que a criação de novos indivíduos não impeça a convergência do método.

Em cada iteração, o indivíduo com a melhor aptidão da população designa-se de líder da população. Se vários indivíduos têm o mesmo valor de aptidão, apenas um deles é escolhido como o líder. Deste modo, este está salvaguardado da extinção (como resultado da reprodução, cruzamento e mutação) e tem uma maior probabilidade de ser seleccionado para a reprodução. Uma vantagem de utilizar um líder é de que o melhor valor de aptidão da população nunca pode piorar de uma iteração para a outra e garante a sobrevivência dos melhores genes [2].

O algoritmo termina quando um dos seguintes critérios de paragem é verificado:

- Quando a variação da aptidão é menor que ε durante as últimas I iterações consecutivas.
- Quando o número de iterações ultrapassa um valor especifico.

Na Figura 3.15 é apresentado um fluxograma com a estrutura de um algoritmo genético simples, que resume o funcionamento do método.

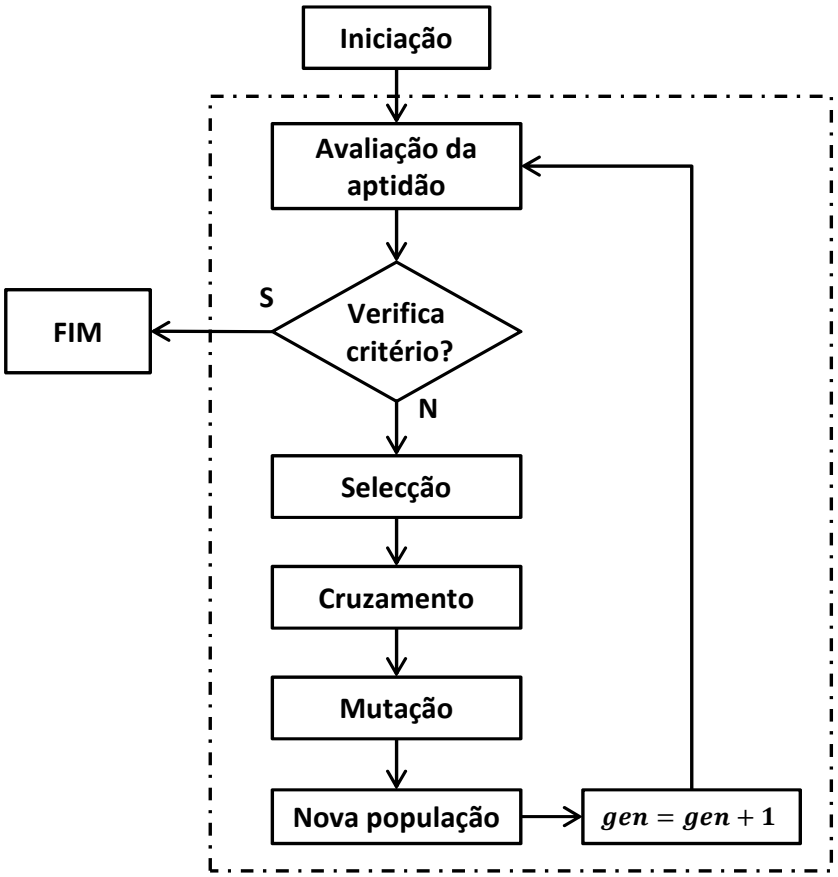


Figura 3.15: Fluxograma de um algortimo genético simples

Como já foi referido, a lentidão da convergência é geralmente o principal ponto negativo apontado a este método, pelo que este método só deve ser utilizado em casos especiais. No entanto, existem formas de acelerar a convergência, apresentadas anteriormente.

Algoritmos genéticos em problemas restringidos

Para resolver um problema de optimização com restrições usando os algoritmos genéticos é necessário transformá-lo num problema sem restrições, já que os algoritmos genéticos apenas resolvem problemas não restringidos. Para tal são utilizados métodos de penalidade [41][2].

Os métodos de penalidade transformam num problema restringido num não restringido numa de duas formas [41]. A primeira adiciona um termo de penalidade do seguinte modo:

$$\min \begin{cases} f(\mathbf{x}) & \text{, caso } \mathbf{x} \in S \\ f(\mathbf{x}) + P(\mathbf{x}) & \text{, caso contrário} \end{cases}$$

Em que S é a região admissível do problema e $P(\mathbf{x})$ é o termo de penalidade. A segunda forma multiplica um termo de penalidade do seguinte modo:

$$\min \begin{cases} f(\mathbf{x}) & \text{, caso } \mathbf{x} \in S \\ f(\mathbf{x})P(\mathbf{x}) & \text{, caso contrário} \end{cases}$$

O método de adição é bastante mais utilizado que o da multiplicação para a resolução de problemas de optimização restringidos através de algoritmos genéticos [41].

A finalidade destes métodos é penalizar a função objectivo sempre que são violadas as restrições do problema. Quanto maior for a violação maior é a penalização. Para tal são usados parâmetros de penalidade [2].

Deste modo, os métodos de penalidade transformam um problema restringido, originalmente formulado do seguinte modo,

$$\begin{aligned} \textbf{minimizar} \quad & f(\mathbf{x}) \\ \textbf{sujeito a} \quad & g_j(\mathbf{x}) \leq 0 \quad \text{com } j = 1, \ldots, m \\ & h_k(\mathbf{x}) = 0 \quad \text{com } k = 1, \ldots, p \\ & x_i^l \leq x_i \leq x_i^u \quad \text{com } i = 1, \ldots, n \end{aligned}$$

num problema não restringido, utilizando uma função de transformação $\phi(\mathbf{x}, \mathbf{r})$ [2],

$$\begin{aligned} \textbf{minimizar} \quad & \phi(\mathbf{x}, \mathbf{r}) \\ \textbf{sujeito a} \quad & x_i^l \leq x_i \leq x_i^u \quad \text{com } i = 1, \ldots, n \end{aligned}$$

Em que

$$\phi(\mathbf{x}, \mathbf{r}) = f(\mathbf{x}) + P(h(\mathbf{x}), g(\mathbf{x}), \mathbf{r}) \tag{3.12}$$

onde $\mathbf{r}$ é o vector dos parâmetros de penalidade e P é uma função real cujo propósito de impor a penalidade na função é controlado por $\mathbf{r}$. Deste modo, P toma o valor de zero caso não sejam violadas as restrições, ou toma um valor positivo caso a desigualdade é violada ($g > 0$), ou caso seja violada a igualdade ($h \neq 0$).

O processo do método resume-se inicialmente a estimar uma solução inicial $\mathbf{x}^{(0)}$ e definir a função ϕ. Define-se também os parâmetros de penalidade $\mathbf{r}$. A função ϕ é então minimizada para os valores de $\mathbf{x}$, mantendo $\mathbf{r}$ fixo. Quando o processo acaba, reajusta-se $\mathbf{r}$ e o processo repete-se até deixar de ser possível melhorar as soluções [2].

Capítulo 4

Casos de estudo

4.1 Vigas contínuas

Nesta secção são estudados problemas de optimização de vigas contínuas, onde se pretende minimizar o momento flector elástico máximo em função da localização e rigidez dos apoios. Para todos os casos considera-se que a secção da viga é constituída por um perfil de aço IPE 220, tendo-se pois $E = 210$ GPa e $I = 2772$ cm^4. Admite-se também que a viga, em todos os casos, tem um comprimento total de $L_{tot} = 10$ m e está sujeita a uma carga uniformemente distribuída $P = 20$ kN/m.

Para verificar se o perfil adoptado é suficiente para resistir à carga actuante para qualquer disposição dos apoios, considera-se a presença de apenas dois apoios situados nas extremidades. Este trata-se do caso mais condicionante para qualquer número de apoios. Desta forma, o momento flector actuante toma o seguinte valor:

$$M_{Ed} = \frac{pL^2}{8} = 250 \text{ kN.m}$$

Considerando a acção do momento flector como condicionante para a verificação da segurança, esta é assegurada caso seja satisfeita a seguinte condição:

$$\frac{M_{Ed}}{M_{Rd}} < 1 \tag{4.1}$$

Sendo M_{Rd} o momento resistente do perfil, é dado por:

$$M_{Rd} = w_{el} \cdot f_y \tag{4.2}$$

Onde f_y é a tensão de cedência do aço e w_{el} é o módulo de flexão elástico. Para o perfil em causa o módulo de flexão é $w_{el} = 252$ cm^3 e considera-se uma tensão de cedência de $f_y = 275$ MPa. Assim o valor do momento resistente do perfil é:

$$M_{Rd} = 69,3 \text{ kN.m}$$

Note-se que, como o momento flector máximo actuante é maior que o resistente a segurança não é verificada para a posição dos apoios mais desfavorável. Contudo, como o objectivo desta dissertação é optimizar o momento flector máximo sem ter em conta restrições quanto à verificação da segurança, a escolha de um perfil prende-se apenas com o facto de ser necessário atribuir valores de rigidez para que os métodos numéricos consigam proceder ao respectivo cálculo de esforços. Desta forma, adopta-se o perfil IPE 220 para todos os casos de vigas contínuas analisados em seguida.

4.1.1 Viga com 3 tramos - Caso 3R

O primeiro caso a ser analisado consiste numa viga contínua bi-apoiada, ilustrada na Figura 4.1.

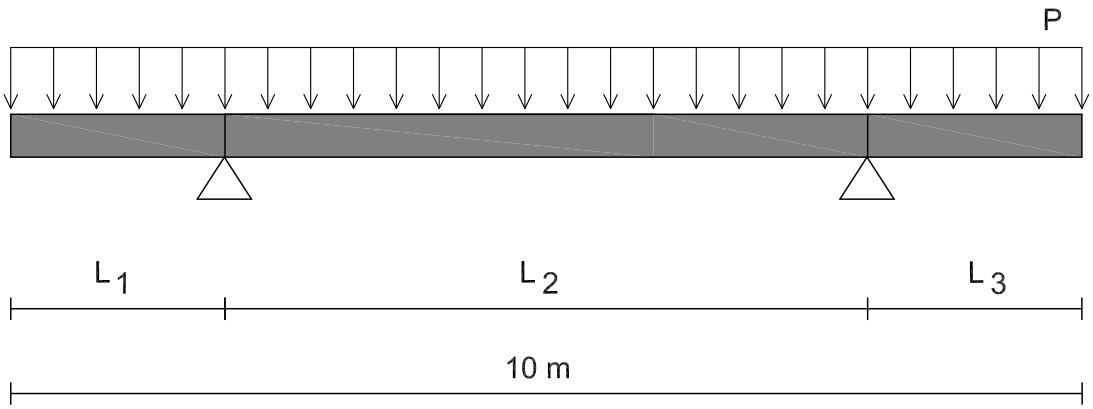


Figura 4.1: Viga contínua com 3 tramos - Caso 3R

A solução óptima para este caso é de fácil resolução, já que o menor valor do momento máximo é obtido quando o momento negativo nos apoios é igual ao momento positivo a meio vão. Este problema já foi estudado em [22] e a sua solução é dada na Tabela 4.1. O diagrama de momento flector para a solução óptima é apresentado na Figura 4.2.

Tabela 4.1: Dimensionamento óptimo - Caso 3R

L_1 (m)	L_2 (m)
2,071	5,858

Tratando-se de um problema de optimização de apenas duas variáveis é possível também resolver este problema graficamente. Para tal, é criado um gráfico de isolinhas dos valores da função objectivo para as diferentes variáveis de projecto, neste caso L_1 e L_2. Na Figura 4.3 é representado um gráfico de isolinhas, em que é possível observar um mínimo global da função, para os valores esperados das variáveis.

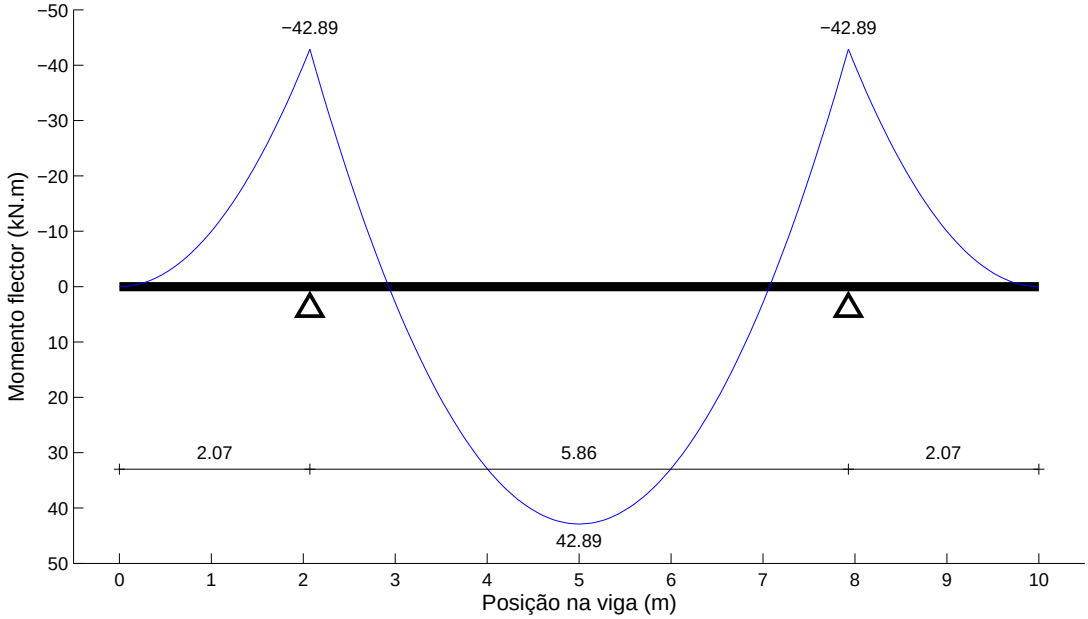


Figura 4.2: Diagrama de momento flector para o dimensionamento óptimo - Caso 3R

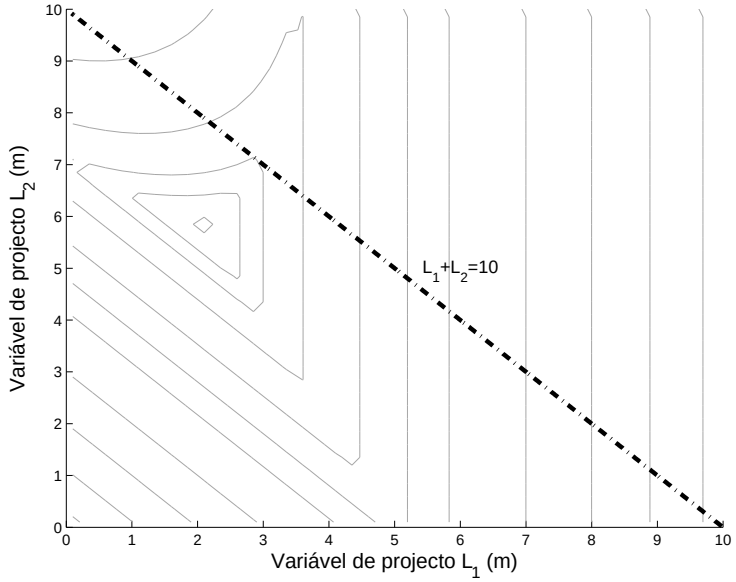


Figura 4.3: Gráfico de isolinhas da função objectivo - Caso 3R

Devido ao facto de se conhecer a solução óptima para este problema, o seu estudo será útil para testar diversos métodos de optimização pré-programados na ferramenta de optimização *optimtool* do MATLAB, que serão necessários para realizar a optimização dos restantes casos de estudo.

Neste problema de uma viga com três tramos não se considera a flexibilidade dos apoios pois a estrutura

é isostática, e como tal, por equilíbrio os esforços têm de ser obrigratoriamente os mesmos para qualquer rigidez considerada.

fminsearch

O primeiro programa a ser testado é o *fminsearch*, que utiliza um algoritmo baseado no método de optimização de procura directa de Nelder-Mead.

Tal como foi referido no ponto 3.5.1, este método necessita de pontos de partida para iniciar o processo iterativo, dos quais depende a capacidade do método convergir para um mínimo global. Como se trata de um método de optimização não-linear sem constrangimentos, os pontos de partida adoptados terão de cuidadosamente escolhidos, pois uma escolha errada desses pontos pode levar a soluções sem significado. Assim os pontos de partida para este problema devem ser positivos e a sua soma tem de ser menor que o comprimento total da viga contínua.

Devido ao facto de este método ser um método de procura local, vão ser considerados quatro diferentes conjuntos de pontos de partida, para verificar se as soluções encontradas correspondem a um mínimo global da função objectivo. Na Tabela 4.2 são apresentados os pontos de partida considerados.

Tabela 4.2: Pontos de partida considerados - Caso 3R

Pontos de partida	L_1 (m)	L_2 (m)
Caso 1	1,0	7,0
Caso 2	3,0	4,0
Caso 3	0,5	8,0
Caso 4	1,5	6,5

No gráfico da Figura 4.4, pode-se observar que todos os conjuntos de pontos convergiram para o mesmo valor e mínimo global da função. Pode-se também verificar que, quanto mais afastados forem os pontos de partida dos pontos óptimos, maior vai ser o número de iterações necessárias para que o método convirja para a solução óptima do problema. Na Figura 4.5 é ilustrado o andamento dos valores da função objectivo para os quatro casos, durante o processo de optimização.

A solução obtida a partir do programa *fminsearch* é apresentada na Tabela 4.3.

Tabela 4.3: Dimensionamento óptimo de viga com 3 tramos através do *fminsearch*

L_1 (m)	L_2 (m)	M^* (kN.m)
2,071	5,858	42,893

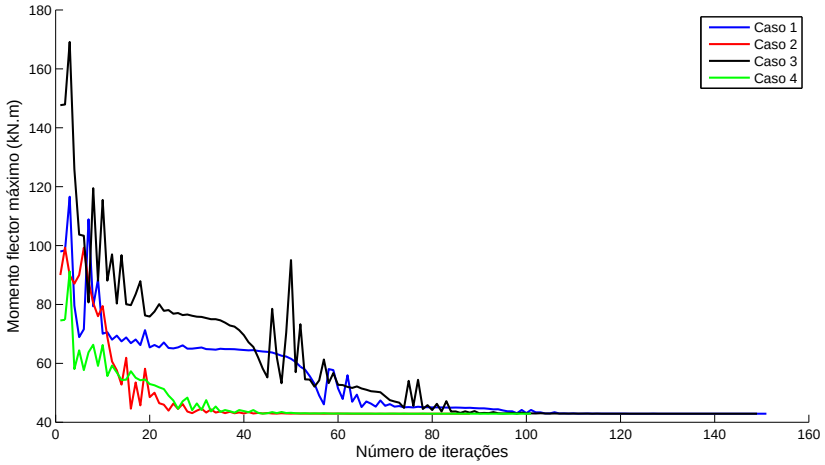
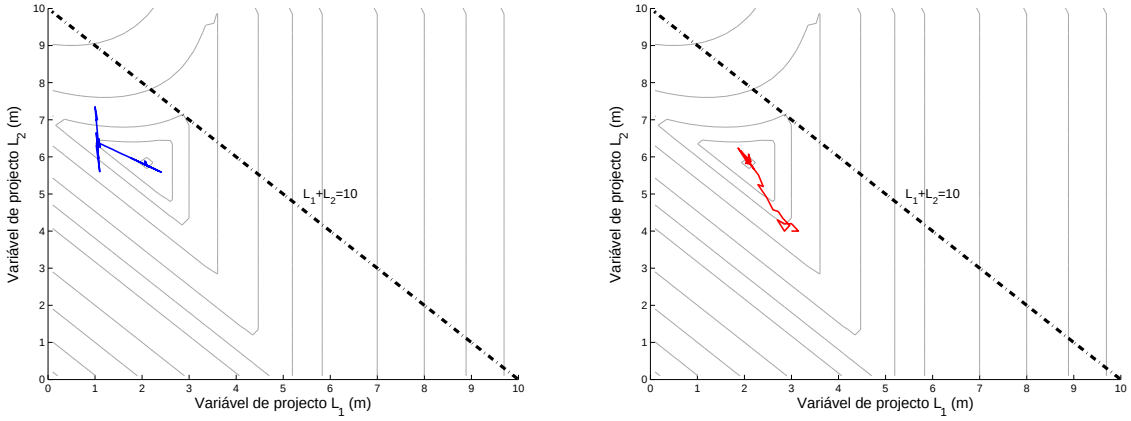
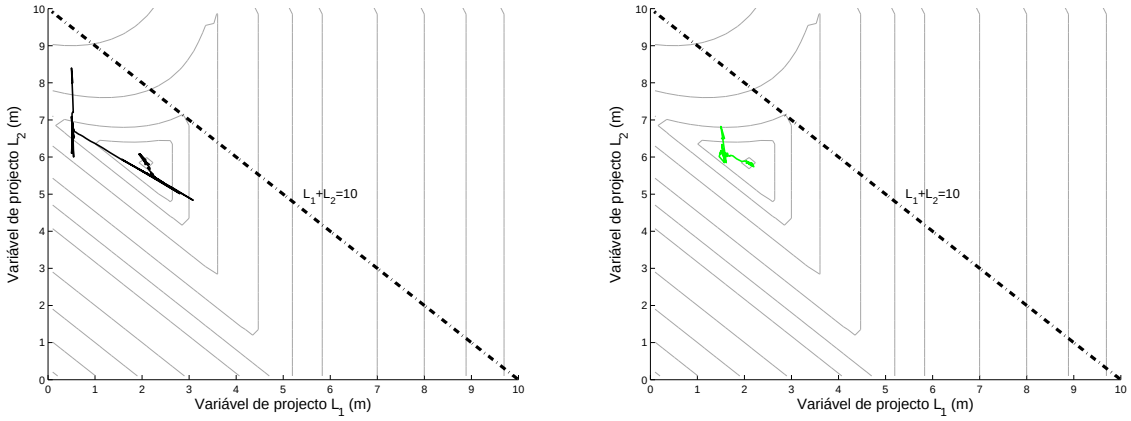


Figura 4.4: Gráfico com a variação dos valores da função objectivo ao longo do processo iterativo



(a) Caso 1

(b) Caso 2



(c) Caso 3

(d) Caso 4

Figura 4.5: Gráficos com a variação do valor da função objectivo para cada um dos casos

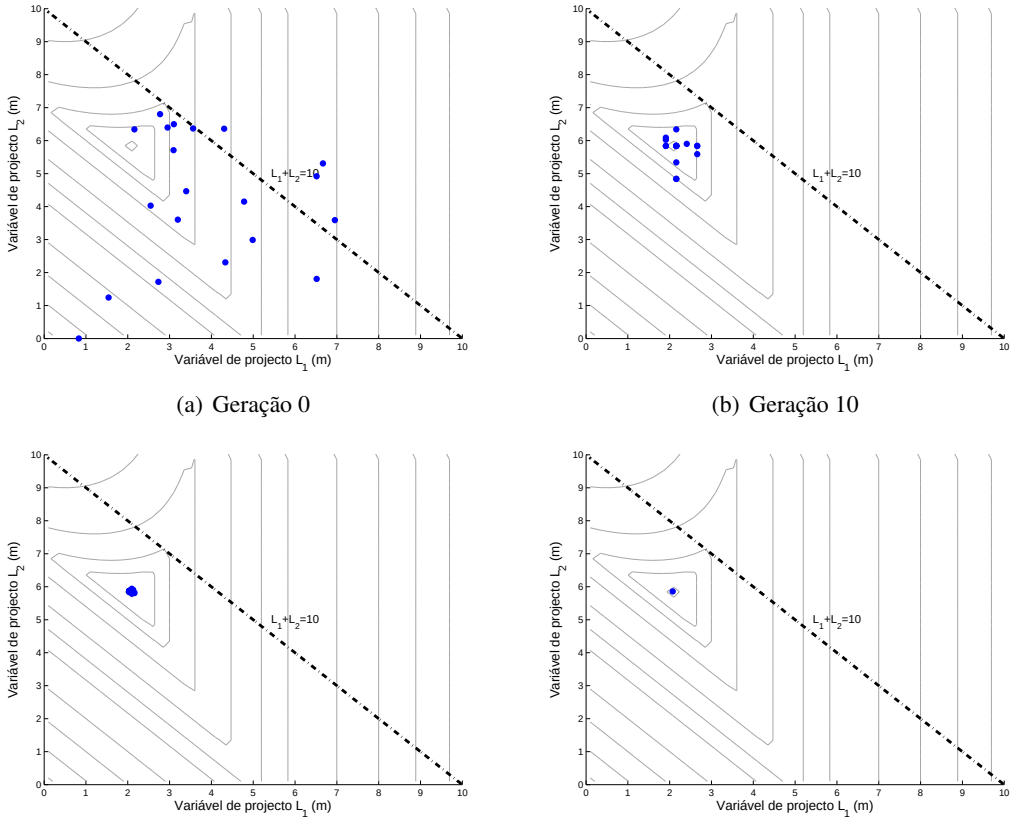
Algoritmos genéticos

Outro método de optimização que se pretende testar são os algoritmos genéticos, que podem ser utilizado no MATLAB a partir do programa *ga*. Como já foi referido no ponto 3.5.2, este é um método de procura global que, a partir de uma população inicial aleatória, desenvolve um processo iterativo em busca do mínimo global da função objectivo.

Para se dar inicio ao algoritmo é necessário introduzir o número de variáveis de projecto do problema, que neste caso é 2, de forma a ser gerada aleatoriamente uma população inicial. Esta população inicial tem um número de indivíduos que se mantém constante ao longo do processo iterativo e é introduzido pelo utilizador, bem como o intervalo de valores em que estes se inserem.

Para este caso de estudo, considera-se uma população inicial com 20 indivíduos cujos valores devem encontrar-se entre 1 e 7 metros, de maneira a não sobrecarregar a memória do computador e tornar a convergência mais rápida. O critério de convergência adoptado é aquele que se encontra definido por defeito do programa MATLAB, ou seja, a variação do valor da função objectivo seja menor que 1×10^{-6} durante 50 gerações.

Na Figura 4.6 pode-se observar a variação das populações ao longo de algumas gerações, até atingir a convergência na geração 51, chegando-se assim à solução óptima do problema, que pode ser observada na Tabela 4.4.



(a) Geração 0

(b) Geração 10

(c) Geração 20

(d) Geração 51

Figura 4.6: Gráficos de variação da população ao longo do processo iterativo

Tabela 4.4: Dimensionamento óptimo de viga com 3 tramos através de algoritmo genético

L_1 (m)	L_2 (m)	M^* (kN.m)
2,071	5,858	42,893

Comparação dos métodos

Analisando os resultados obtidos através dos dois métodos de optimização verifica-se que ambos convergem para o valor de momento flector máximo esperado. Contudo, existem alguns pontos de comparação que devem ser referidos.

O método de Nelder-Mead , como já foi referido, fica frequentemente preso em mínimos locais. No entanto, devido à simplicidade do problema, onde a função objectivo apresenta apenas um mínimo no domínio considerado, este método consegue convergir para a solução óptima de uma forma bastante rápida (Caso 1: 59 segundos; Caso 2: 32 segundos; Caso 3: 1 minuto e 9 segundos; Caso 4: 29 segundos).

Quanto aos algoritmos genéticos verifica-se que apresentam um processo de cálculo mais lento (7 minutos e 52 segundos). Contudo, este método consegue por norma convergir para soluções óptimas que eventualmente serão um mínimos da função, mas sem que se possa garantir tal facto.

Desta forma, para os casos de estudo mais simples (vigas) será utilizado o método de Nelder-Mead, enquanto que para os casos mais complexos (grelhas), onde não se conhece bem o espaço da função objectivo, serão utilizados algoritmos genéticos.

Note-se que os tempos registados foram todos obtidos utilizando o mesmo computador, com um processador Intel Core i5-2450M (2,5GHz) e com uma memória RAM 4GB.

4.1.2 Viga com 4 tramos

Neste caso de estudo pretende-se optimizar os momentos flectores de uma viga contínua composta por quatro tramos. Numa primeira análise considera-se que a viga está apoiada em três apoios rígidos e as variáveis de projecto são os comprimentos dos tramos. Numa segunda análise, os apoios deixam de ser rígidos e passam a ser flexíveis, pelo que para além dos comprimentos dos tramos, também serão consideradas como variáveis de projecto as rigidezes dos apoios.

Apoios rígidos - Caso 4R

Na Figura 4.7 é ilustrada a viga em estudo, com as respectivas variáveis de projecto.

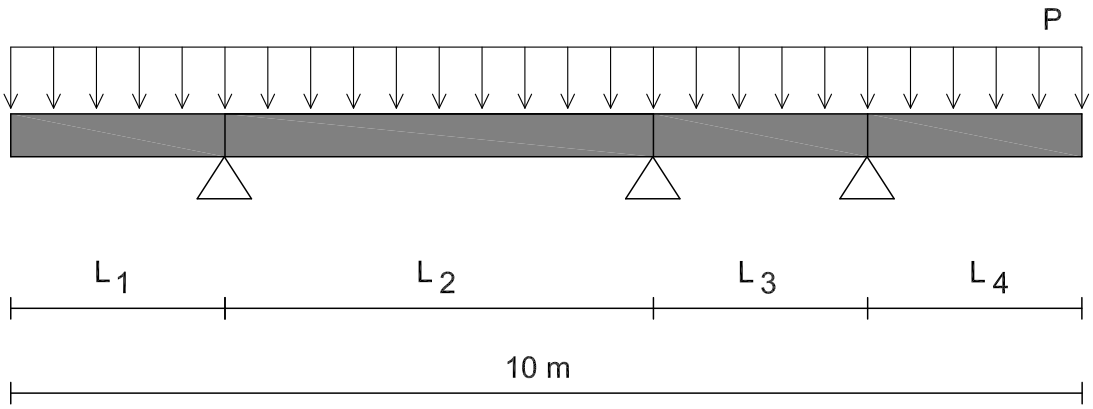


Figura 4.7: Viga contínua com 4 tramos - Caso 4R

Pelos motivos apresentados anteriormente, foram efectuados quatro testes com quatro pontos de partida diferentes, apresentados na Tabela 4.5, para verificar se o mínimo encontrado é de facto um mínimo global.

Tabela 4.5: Pontos de partida considerados - Caso 4R

Pontos de partida	L_1 (m)	L_2 (m)	L_3 (m)
Caso 1	1,0	4,0	4,0
Caso 2	2,0	3,0	3,0
Caso 3	1,5	3,5	3,5
Caso 4	1,2	3,8	4,0

Como se pode verificar pelo gráfico da Figura 4.8, conseguiu-se obter a convergência da solução. Os resultados da solução óptima podem ser visualizados na Tabela 4.6.

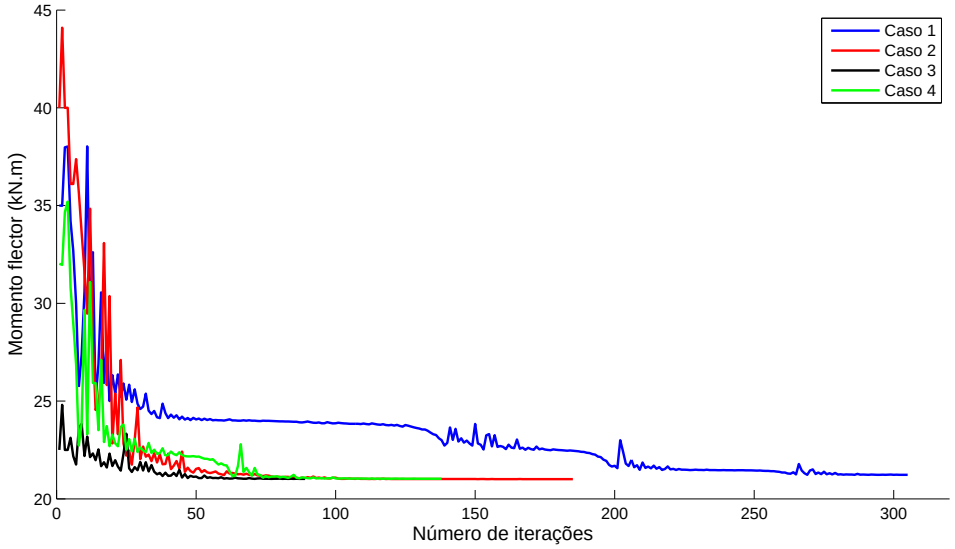


Figura 4.8: Gráfico da convergência das soluções - Caso 4R

Tabela 4.6: Solução óptima - Caso 4R

L_1 (m)	L_2 (m)	L_3 (m)	L_4 (m)	M^* (kN.m)
1,45	3,55	3,55	1,45	21,01

O dimensionamento óptimo da viga para a optimização do momento flector máximo é obtido quando os momentos negativos nos pilares se igualam entre si. Para uma melhor percepção do dimensionamento óptimo da viga, é representado na Figura 4.9 o diagrama de momento flector óptimo e as soluções do problema.

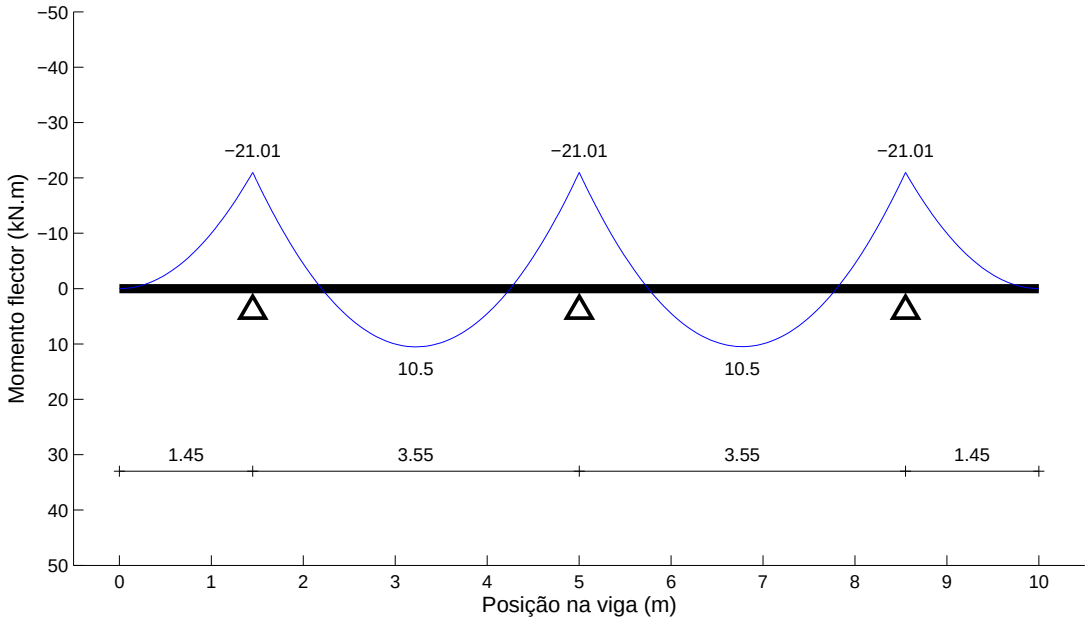


Figura 4.9: Diagrama de momento flector para o dimensionamento óptimo da viga - Caso 4R

Como é sabido, na prática dificilmente se consegue posicionar os apoios nas suas localizações óptimas, o que provoca variações nos esforços. Para se ter uma noção de como varia o momento flector máximo na viga em relação ao seu valor óptimo, foram introduzidas variações nos comprimentos dos tramos em relação às soluções óptimas. Na Tabela 4.7 são apresentadas as diferentes posições consideradas.

Tabela 4.7: Influência da variação dos comprimentos dos tramos em relação aos óptimos - Caso 4R

	L_1 (m)	L_2 (m)	L_3 (m)	L_4 (m)	M/M^*
Caso 1	1,4	3,6	3,6	1,4	1,076
Caso 2	1,5	3,5	3,5	1,5	1,071
Caso 3	1,0	4,0	4,0	1,0	1,666
Caso 4	2,0	3,0	3,0	2,0	1,904

Pode-se verificar que quando se varia os comprimentos dos tramos em relação à solução óptima, arredondando-os aos decímetros (caso 1 e caso 2), existe imediatamente um aumento do momento flector máximo de aproximadamente 8%, em ambos os casos. Nos casos 3 e 4 os comprimentos são arredondados aos metros, e verifica-se que o valor do momento flector máximo tem um aumento de 67%, para o caso 3, e de 90%, para o caso 4, em relação ao seu valor óptimo.

Na Figura 4.10 é apresentado um gráfico da variação do momento flector máximo quando se altera o comprimento do tramo L_1 com incrementos de 5 cm, entre os valores de 1 e 2 m, e mantida a simetria da viga.

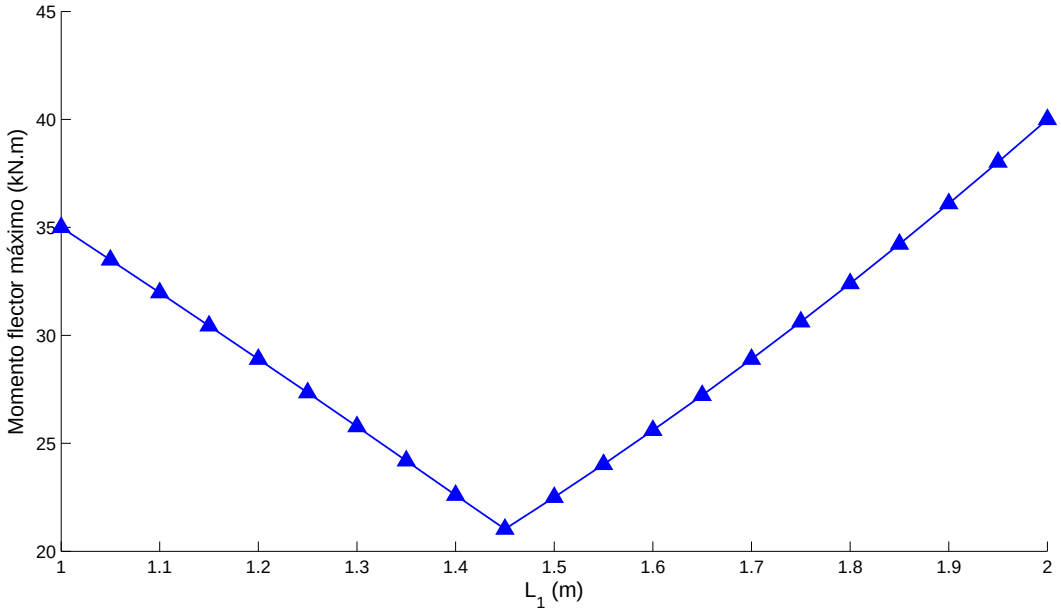


Figura 4.10: Gráfico da variação do momento flector máximo com o vão L_1 - Caso 4R

Apoios flexíveis - Caso 4F

Neste caso pretende-se não só optimizar os comprimentos dos tramos, mas também as rigidezes dos apoios em que a viga contínua está apoiada, aumentando deste modo as variáveis de projecto. Na Figura 4.11 é então ilustrada a viga contínua apoiada em apoios flexíveis que se pretende optimizar.

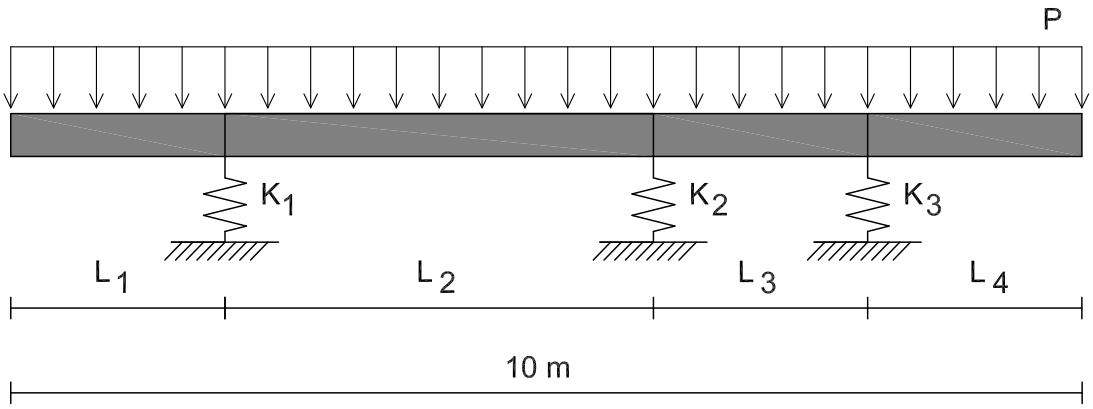


Figura 4.11: Viga contínua com 4 tramos apoiada em apoios flexíveis - Caso 4F

Neste caso o programa *fminsearch* tem alguma dificuldade em achar o óptimo global da função. Para melhorar a *performance* do programa considera-se a simetria da estrutura, diminuindo deste modo as variáveis de projecto e aumentando a capacidade do método conseguir achar o mínimo global da função. Foram então considerados os seguintes pontos iniciais:

Tabela 4.8: Pontos de partida considerados - Caso 4F

Pontos de partida	L_1 (m)	$L_2 = L_3$ (m)	$K_1 = K_3$ ($\times 10^4$ kN/m)	K_2 ($\times 10^4$ kN/m)
Caso 1	1,3	3,6	2,0	0,5
Caso 2	0,5	3,0	5,0	1,0
Caso 3	2,5	2,5	2,0	1,0
Caso 4	1,5	3,0	2,0	1,0

Como se pode verificar pela Figura 4.12, mesmo considerando as simplificações por simetria, o método teve dificuldades em convergir para um mínimo global da função objectivo; contudo, esse mínimo foi encontrado e a solução óptima é apresentada na Tabela 4.9.

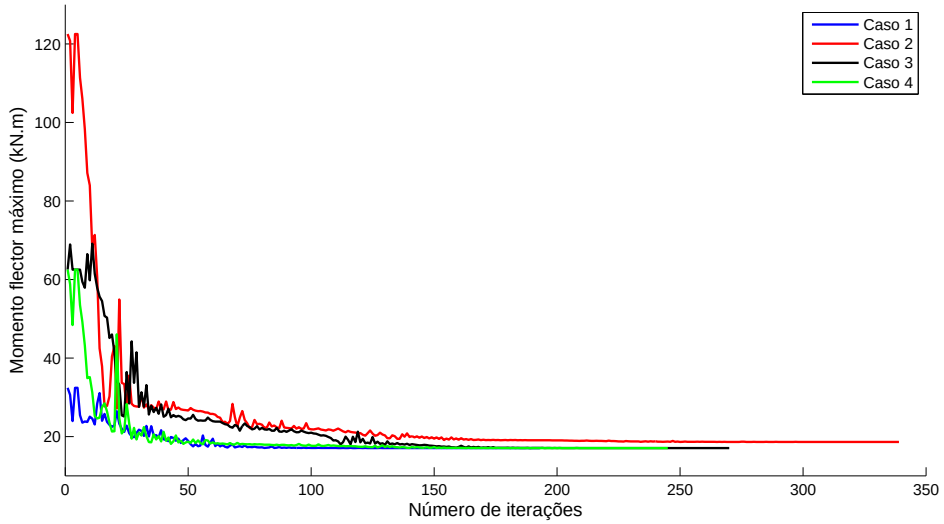


Figura 4.12: Gráfico da convergência das soluções - Caso 4F

Tabela 4.9: Solução óptima - Caso 4F

L_1 (m)	$L_2 = L_3$ (m)	L_4 (m)	$K_1 = K_3$ (kN/m)	K_2 (kN/m)	M^* (kN.m)
1,31	3,69	1,31	22116,7	7764,0	17,06

Como era esperado, os apoios flexíveis permitem aumentar os valores de momentos flectores positivos nos vãos interiores e diminuir os momentos negativos sobre os apoios, diminuindo desta forma os comprimentos dos vãos das consolas e aumentando os vãos interiores. A solução óptima é obtida quando os valores das variáveis de projecto permitem que os momentos flectores negativos e positivos se igualem entre si, como ilustrado na Figura 4.13.

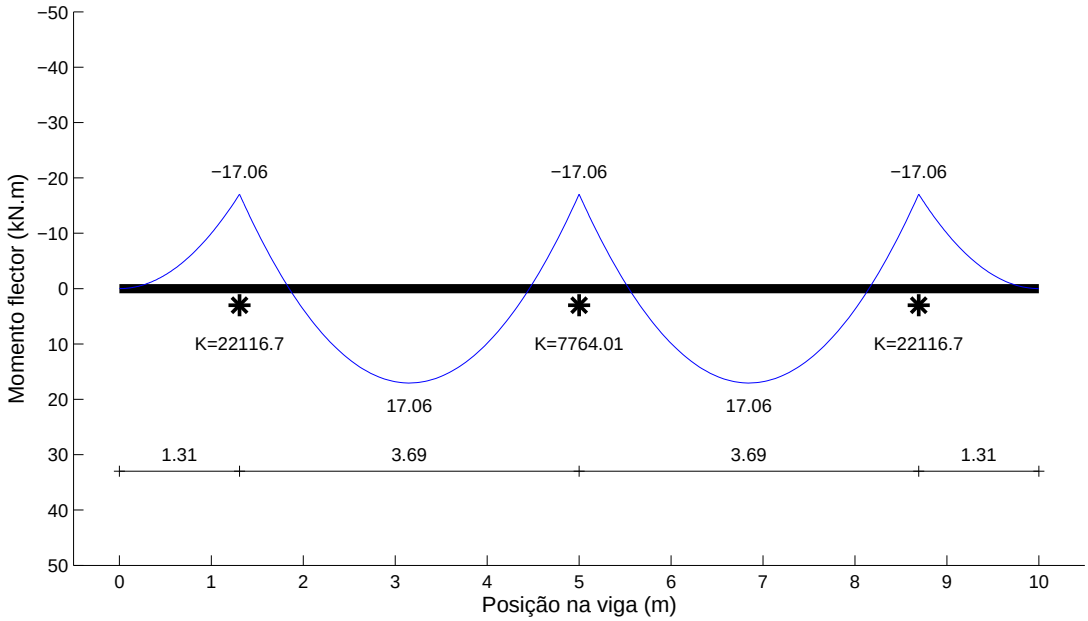


Figura 4.13: Diagrama de momento flector para o dimensionamento óptimo da viga com apoios flexíveis - Caso 4F

Comparando esta solução com a solução de uma viga contínua apoiada em apoios rígidos, verifica-se que existe uma diminuição do momento flector máximo de aproximadamente 19%, quando se consideram as rigidezes dos apoios como variáveis de projecto. No entanto, é difícil arranjar soluções construtivas que permitam simular uma determinada rigidez, pois esta rigidez está normalmente associada à rigidez axial dos elementos nos quais a viga se apoia e esta por norma é muito elevada.

4.1.3 Viga com 5 tramos

Este caso de optimização é semelhante ao anterior, diferindo apenas no número de tramos da viga contínua, que neste caso são cinco. A viga contínua está apoiada em quatro apoios que, numa primeira abordagem, são rígidos e que, numa segunda abordagem, se consideram flexíveis. O comprimento total da viga, as características da secção e o material, bem como a carga aplicada, são idênticos ao caso anterior.

Apoios rígidos - Caso 5R

Neste caso a viga que se pretende optimizar está apoiada em apoios rígidos, como ilustrado na Figura 4.14, sendo as variáveis de projecto os comprimentos dos tramos e a função objectivo o momento flector máximo a actuar na viga.

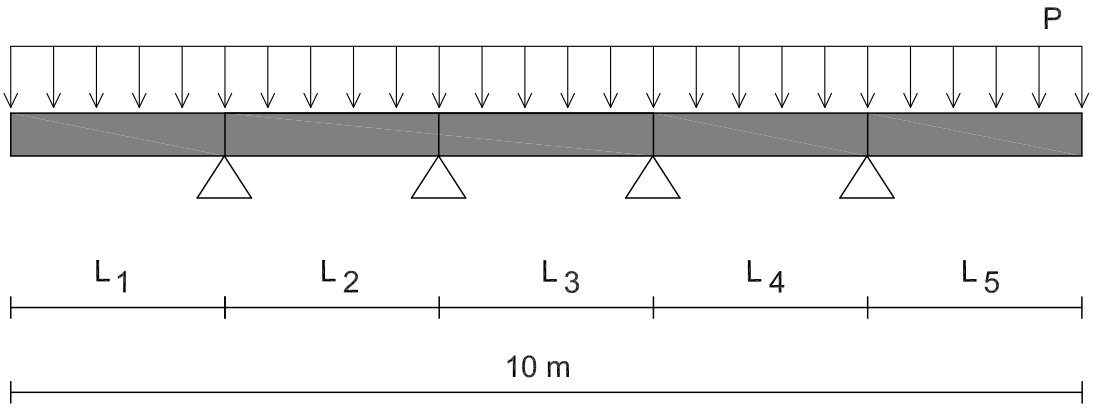


Figura 4.14: Viga contínua com 5 tramos - Caso 5R

Para verificar se o método converge para um mínimo global da função, são considerados novamente quatro conjuntos de pontos de partida diferentes, apresentados na Tabela 4.10.

Tabela 4.10: Pontos de partida considerados - Caso 5R

Pontos de partida	L_1 (m)	L_2 (m)	L_3 (m)	L_4 (m)
Caso 1	1,0	3,0	2,0	3,0
Caso 2	1,0	2,5	3,0	2,5
Caso 3	1,2	3,0	1,6	3,0
Caso 4	1,0	2,5	2,5	3,0

Como se pode verificar pelo gráfico da Figura 4.15, conseguiu-se obter a convergência para a solução óptima global, apesar de o método ter dificuldades em encontrar o mínimo global da função, devido à complexidade desta.

Os valores óptimos do problema são apresentados na Tabela 4.11. Na Figura 4.16 é apresentado o diagrama de momento flector para o dimensionamento óptimo da viga.

Tabela 4.11: Solução óptima - Caso 5R

L_1 (m)	L_2 (m)	L_2 (m)	L_4 (m)	L_5 (m)	M^* (kN.m)
1,05	2,82	2,26	2,82	1,05	11,12

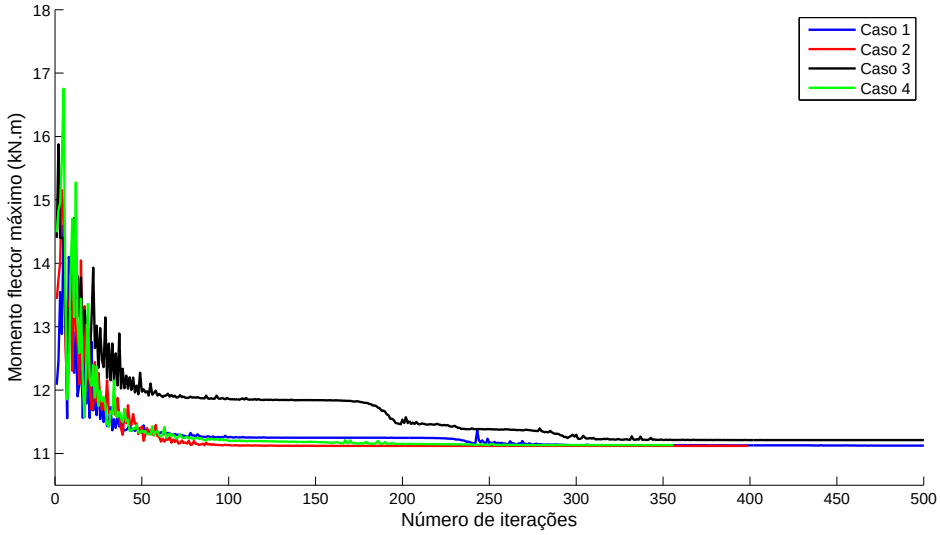


Figura 4.15: Gráfico da convergência das soluções - Caso 5R

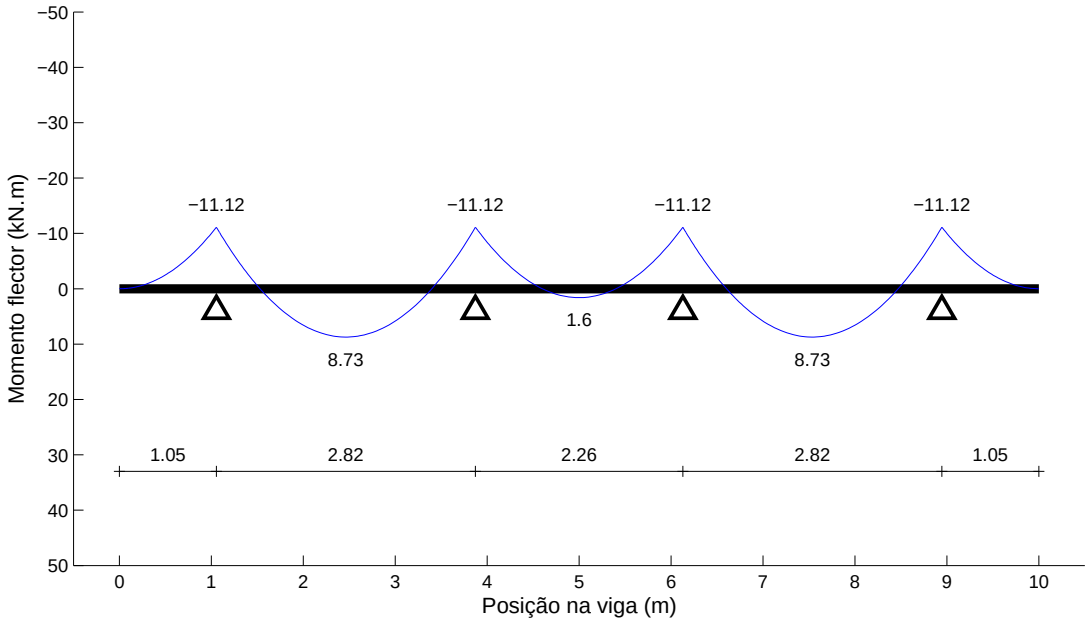


Figura 4.16: Diagrama de momento flector para o dimensionamento óptimo da viga contínua - Caso 5R

Como se pode verificar pela Figura 4.16, o dimensionamento óptimo é atingido novamente quando os momentos flectores negativos sobre os apoios se igualam entre si.

Apoios flexíveis - Caso 5F

Considera-se agora que a viga contínua anterior está apoiada em apoios flexíveis, como ilustrado na Figura 4.17. Pretende-se neste problema de optimização minimizar o momento flector máximo, alterando os comprimentos dos tramos e as rigidezes dos apoios.

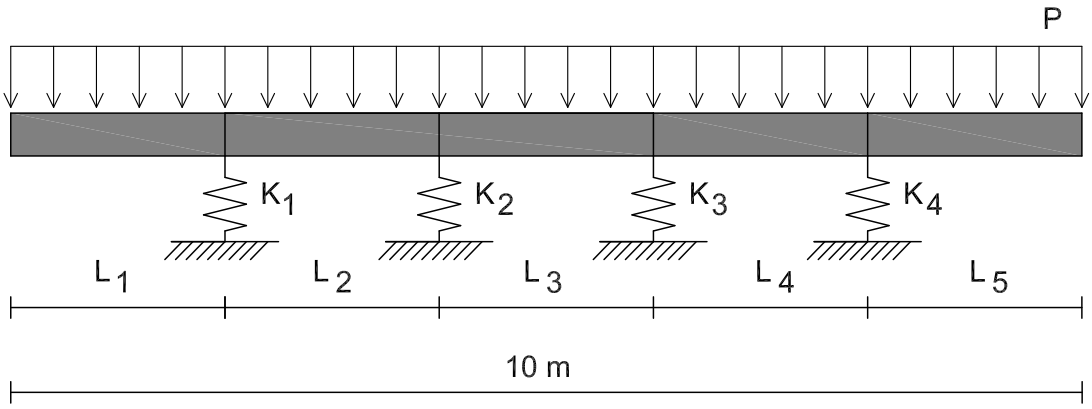


Figura 4.17: Viga contínua com 5 tramos apoiada em apoios flexíveis - Caso 5F

Devido ao facto de a função objectivo ser muito complexa, admite-se que o dimensionamento óptimo da viga surge quando os comprimentos dos tramos e as rigidezes dos apoios tomam valores simétricos. Deste modo, reduz-se o número de variáveis de projecto aumentando a *performance* do método. É usado novamente o programa *fminsearch* e considerados 4 conjuntos de pontos de partida, pelas razões já apresentadas. Na Tabela 4.12 são apresentados os pontos de partida considerados.

Tabela 4.12: Pontos de partida considerados - Caso 5F

Pontos de partida	L_1 (m)	$L_2 = L_4$ (m)	L_3 (m)	$K_1 = K_4$ ($\times 10^4$ kN.m)	$K_2 = K_3$ ($\times 10^4$ kN.m)
Caso 1	2,0	2,0	2,0	2,0	1,0
Caso 2	1,0	2,0	4,0	2,0	1,0
Caso 3	1,0	1,0	5,0	2,0	1,0
Caso 4	0,8	2,5	2,5	7,0	1,0

Como se pode observar na Figura 4.18, a convergência foi conseguida para um mínimo global da função objectivo.

De maneira semelhante ao sucedido no problema de 4 tramos, ao se considerar também as rigidezes dos apoios como variávies de projecto, o dimensionamento óptimo da viga é atingido quando os momentos flectores positivos a meio vão igualam os momentos negativos nos apoios. Na Tabela 4.13 é então apresentado a solução óptima do problema, e na Figura 4.19 é ilustrado o diagrama de momento flector para essa solução.

Tabela 4.13: Solução óptima - Caso 5F

$L_1 = L_5$ (m)	$L_2 = L_4$ (m)	L_3 (m)	$K_1 = K_4$ ($\times 10^4$ kN.m)	$K_2 = K_3$ ($\times 10^4$ kN.m)	M^* (kN.m)
0,95	2,70	2,70	6,91	1,21	9,10

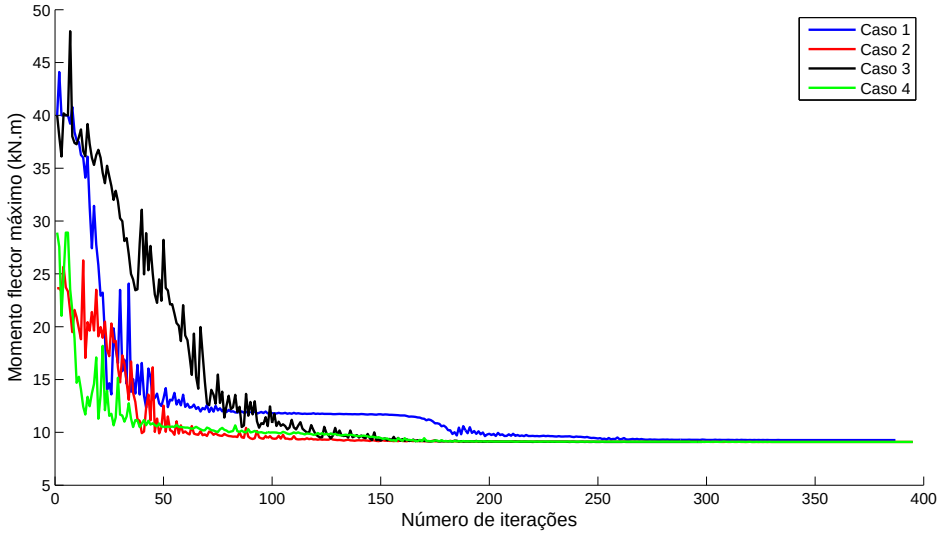


Figura 4.18: Gráfico da convergência das soluções - Caso 5F

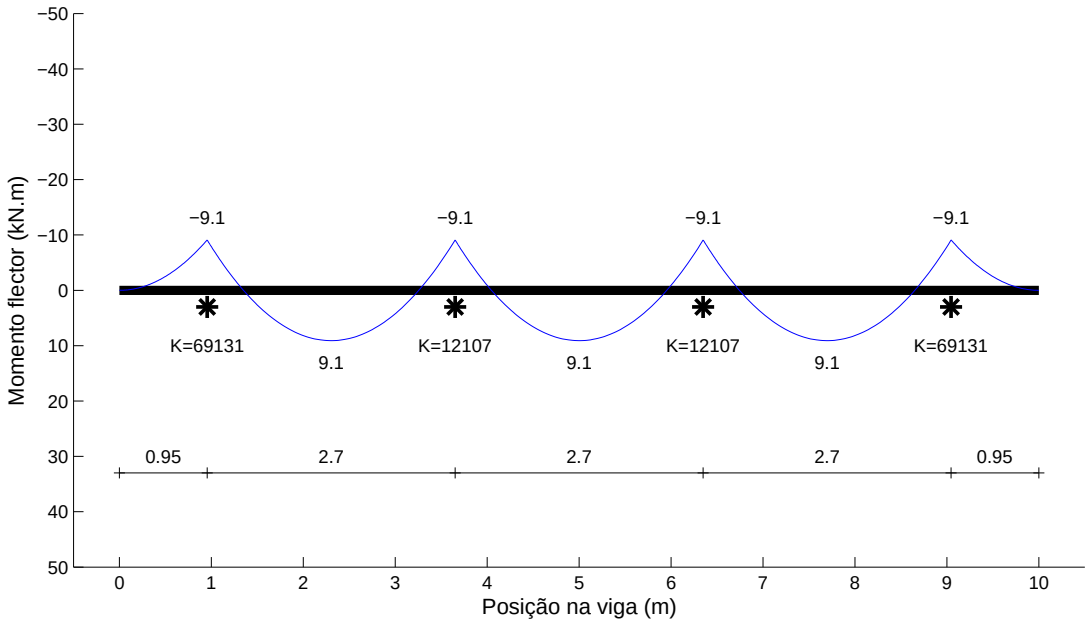


Figura 4.19: Diagrama de momento flector para o dimensionamento óptimo da viga contínua - Caso 5F

Comparando este caso com o caso em que se consideram os apoios rígidos, verifica-se que a flexibilidade dos apoios permite uma redução de cerca de 18% do valor do momento flector máximo a actuar na viga.

4.1.4 Discussão dos resultados

Analisando os resultados obtidos, verifica-se que o processo de optimização proposto consegue encontrar a posição e rigidez dos apoios que permite minimizar o valor do momento flector máximo actuante em vigas contínuas.

Nas vigas contínuas os momentos flectores máximos são minimizados quando os momentos negativos dos pilares se igualam entre si, caso os apoios sejam rígidos, ou quando os momentos negativos nos apoios e os momentos positivos nos vãos são iguais, caso os apoios tenham uma flexibilidade que o permita.

A consideração das rigidezes dos apoios como variáveis de projecto permitiu uma redução de aproximadamente 20% dos momentos flectores máximos em relação aos casos onde se consideram os apoios como rígidos.

4.2 Sistemas de grelhas

Nesta secção são estudados problemas de optimização de sistemas de grelhas. Nestes problemas o momento flector elástico volta a ser a função objectivo, apesar de poder existir torção nos elementos, e as variáveis de projecto são os espaçamentos de cada tramo no respectivo eixo. A não consideração da torção justifica-se pelo facto dos momentos torsores em geral serem bastantes inferiores aos momentos flectores, pelo que estes últimos são os que normalmente condicionam o dimensionamento. Os sistemas de grelhas considerados são aqueles apresentados em 2.5.2.

A optimização é realizada através dos algoritmos genéticos, recorrendo ao programa *ga*, presente na ferramenta *optimtool* do MATLAB. Devido à complexidade do problema, considera-se que dimensionamento óptimo é obtido para uma estrutura simétrica em cada eixo. Deste modo, diminui-se o número de variáveis de projecto simplificando o processo de cálculo.

Para garantir que as dimensões totais da grelha não se alterem durante o processo de optimização é necessário impor restrições de igualdade ao problema. Assim, considera-se que a soma dos comprimentos de cada tramo em cada direcção tem de ser igual ao comprimento total definido nessa mesma direcção:

$$\sum Lx_i = Lx_{total} \quad \text{com } i \text{ =1,...,nº de tramos em } x \tag{4.3}$$

$$\sum Ly_j = Ly_{total} \quad \text{com } j \text{ =1,...,nº de tramos em } y \tag{4.4}$$

Para evitar que o valor das variáveis convirja para valores negativos, foi definido um limite inferior das variáveis, $l_b = 1 \times 10^{-7}$ m.

O número de indivíduos da população inicial ($DimPop$), o parâmetro de tolerância do critério de paragem (ε) e os limites do domínio da população inicial (PIR) são definidos separadamente para cada caso, de forma a melhorar os respectivos resultados. Para os restantes parâmetros são utilizados os valores pré-definidos na *optimtool*.

Em todos os casos, de seguida apresentados, considera-se que as grelhas são constituídas por um sistema de perfis HEB200 ($E = 200$ GPa; $G = 84$ GPa; $I = 5696$ cm^4; $J = 59,28$ cm^4) e sujeitas a uma carga de superfície uniformemente distribuída $Q = 10$ kN/m^2.

Devido ao facto de apenas se considerarem como variáveis de projecto os espaçamentos entre vigas, ou seja, a posição dos apoios, este tipo de optimização trata-se de uma optimização estrutural de forma.

4.2.1 Grelha de 10×10 m com 5×5 tramos - Caso G5×5-10×10

O primeiro caso de estudo trata-se de um sistema de grelhas quadrado, com $Lx_{total} = Ly_{total} = 10$ m, e constituído por 5 tramos em cada direcção, como ilustrado na Figura 4.20.

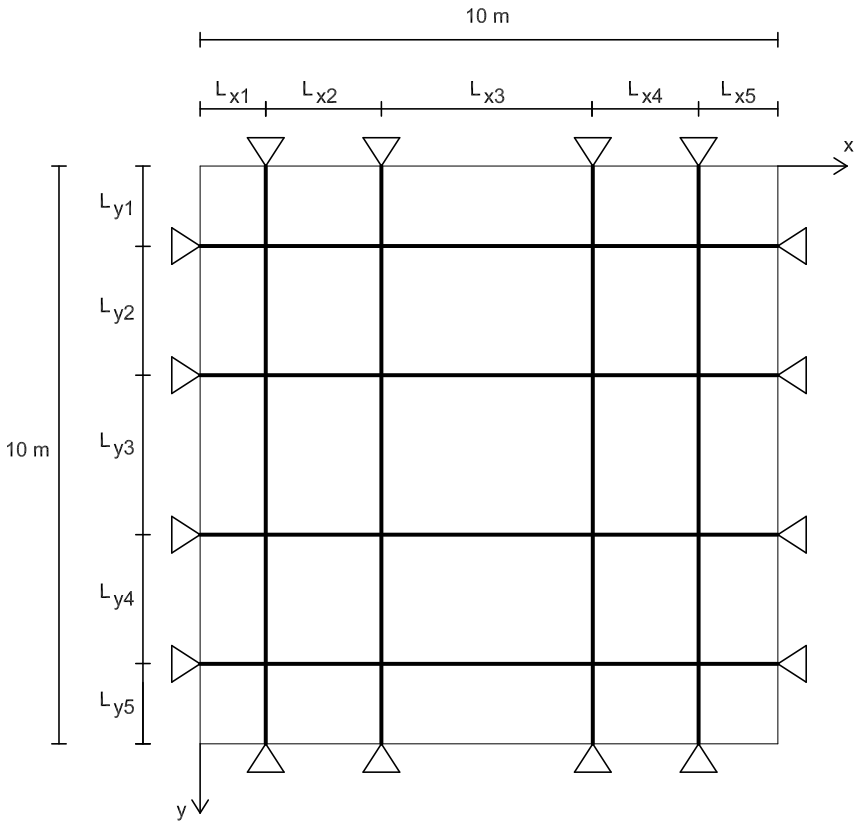


Figura 4.20: Grelha caso G5×5-10×10

Tendo em atenção a simetria da estrutura, considera-se então que o problema tem 6 variáveis de projecto, que são:

$$X = [Lx_1 = Lx_5, Lx_2 = Lx_4, Lx_3, Ly_1 = Ly_5, Ly_2 = Ly_4, Ly_3] \tag{4.5}$$

E estão condicionadas pelas seguintes igualdades:

$$2Lx_1 + 2Lx_2 + Lx_3 = 10 \tag{4.6}$$

$$2Ly_1 + 2Ly_2 + Ly_3 = 10 \tag{4.7}$$

Devido à dupla simetria da estrutura, a função objectivo apresenta dois mínimos globais.

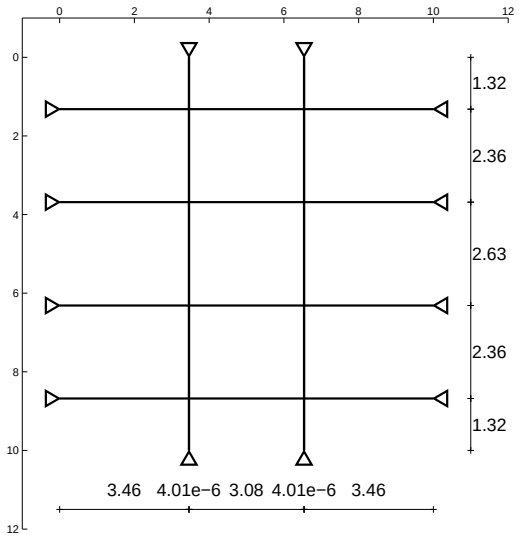
Na Tabela 4.14 são apresentados os valores óptimos para os três tipos de grelhas em estudo e na Figura 4.21 é ilustrada a disposição dos elementos da grelha para esses valores.

Tabela 4.14: Solução óptima - Caso G5×5-10×10

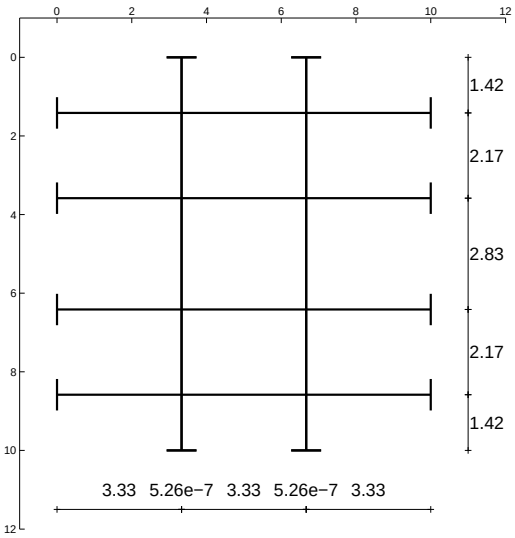
Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.a	200	[1;3]	1×10^{-6}	12,9934	58
Tipo I.b	200	[1;3]	1×10^{-6}	11,5710	51
Tipo II	250	[0.5;4]	1×10^{-6}	9,9988	55

Tipo	Valores óptimos das variávies (ga)					
	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Ly_5$ (m)	$Ly_2 = Ly_4$ (m)	Ly_3 (m)
Tipo I.a	3,4611	$4{,}005\times10^{-6}$	3,0769	1,3200	2,3648	2,6295
Tipo I.b	3,3328	$5{,}257\times10^{-6}$	3,3334	1,4159	2,1678	2,8317
Tipo II	3,2056	$6{,}217\times10^{-6}$	3,5883	2,6349	1,1097	2,5114

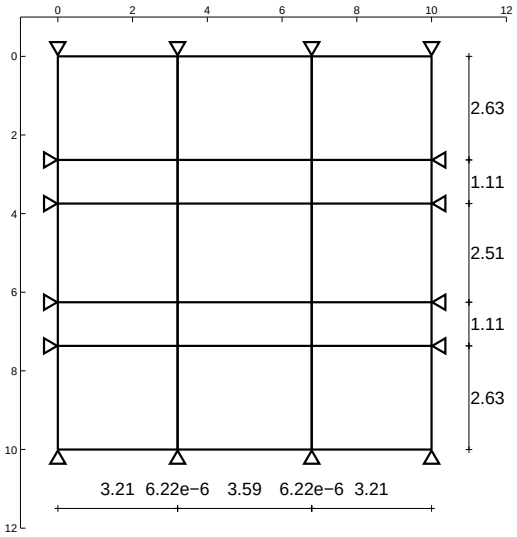
Como se pode verificar pelos resultados apresentados, o dimensionamento óptimo ocorre quando os perfis numa das direcções tendem a juntar-se, como é mostrado na Figura 4.22. Ao juntar os perfis aumenta-se nessa direcção a respectiva inércia, fazendo com que a grelha tenha uma maior rigidez de flexão numa direcção do que na outra.



(a) Grelha tipo I.a



(b) Grelha tipo I.b



(c) Grelha tipo II

Figura 4.21: Dimensionamento óptimo para o caso G5×5-10×10

Esta solução aproxima-se mais da solução ideal que seria a colocação de apoios nos nós de cruzamento das vigas, de modo que estes contrabalançassem os momentos positivos dos vãos com os momentos negativos, como acontece no caso das vigas.

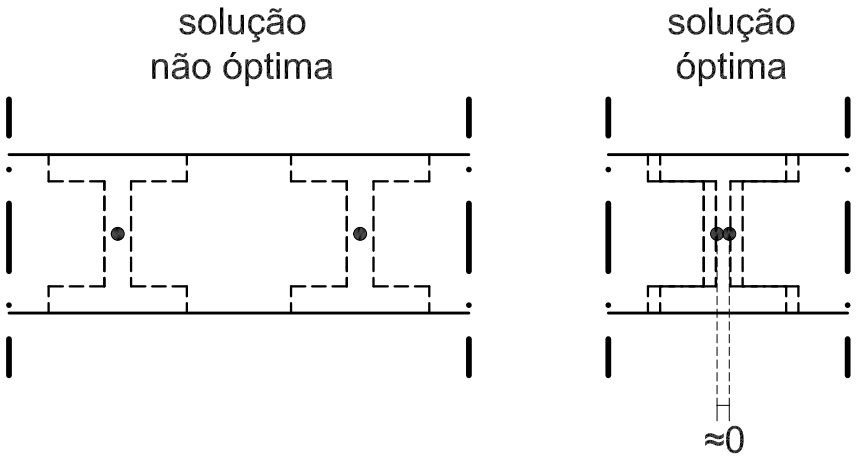


Figura 4.22: Esquema da junção dos perfis

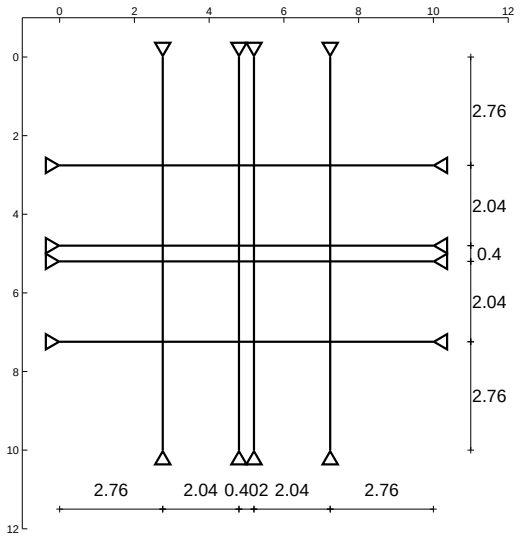
A solução óptima apresentada é a solução matemática para o problema; no entanto, é impossível de ser concebida. Com o propósito de tentar encontrar uma solução óptima para o problema que seja construtivamente possível, são alterados os limites inferiores das variáveis para o valor da largura dos perfis. No caso dos perfis HEB200 a largura é de 200 mm, pelo que se adopta um limite inferior para as variáveis de $l_b = 0,2$ m.

Na Tabela 4.15 são apresentados os valores óptimos considerando os novos limites inferiores das variáveis, e na Figura 4.23 é ilustrada a disposição dos elementos da grelha para esses valores.

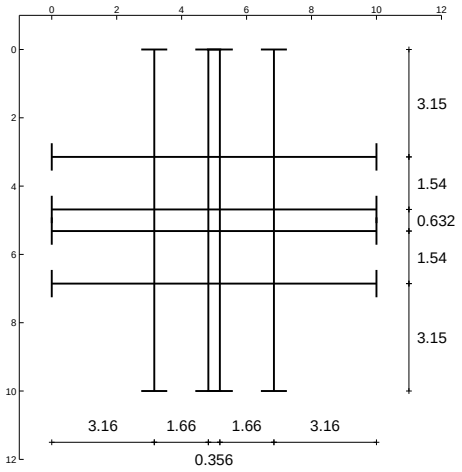
Tabela 4.15: Solução óptima - Caso G5×5-10×10-construtivo

Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.a	300	[1;4]	1×10^{-10}	153,8940	100
Tipo I.b	300	[1;4]	1×10^{-8}	100,0400	80
Tipo II	200	[0.5;4]	1×10^{-10}	96,0060	51

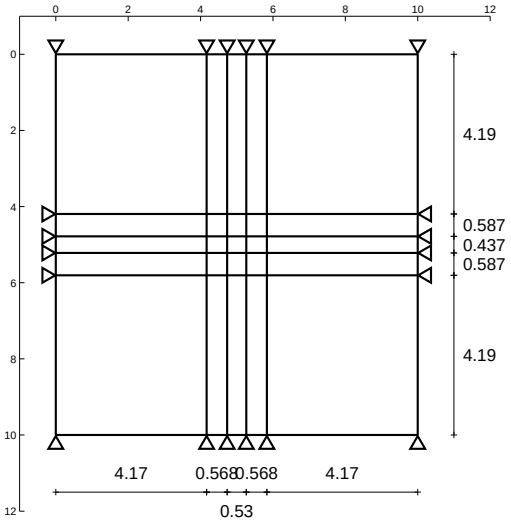
Tipo	Valores óptimos das variávies (ga)					
	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Ly_5$ (m)	$Ly_2 = Ly_4$ (m)	Ly_3 (m)
Tipo I.a	2,7593	2,0393	0,4019	2,7561	2,0434	0,4001
Tipo I.b	3,1577	1,6641	0,3555	3,1451	1,5383	0,63231
Tipo II	4,1667	0,5677	0,5303	4,1939	0,5873	0,4367



(a) Grelha tipo I.a



(b) Grelha tipo I.b



(c) Grelha tipo II

Figura 4.23: Dimensionamento óptimo para o caso G5×5-10×10-construtivo

Comparando ambos os resultados obtidos anteriormente com a solução mais usual, que é aquela em que os espaçamentos dos tramos são todos iguais, verifica-se que a solução inicial, em que não se considera a restição da largura dos perfis, apresenta valores de momento flector máximo bastante mais baixos que as outras duas (aproximadamente 90% para todos os tipos de grelha), no entanto, como já foi referido, esta trata-se de uma solução impossível de ser materializada. Comparando apenas as duas soluções construtivamente possíveis, verifica-se que, para todos os tipos de grelhas, a solução optimizada consegue reduzir efectivamente o momento flector máximo (cerca de 7,6% para o tipo I.a, 18,2% para o tipo I.b e 32,5% para o tipo II). Verifica-se também que, dos tipos de grelha considerados, o tipo II é aquele onde a optimização reduz mais o momento flector máximo. Na Tabela 4.16 é apresentado os momentos flectores máximos para as três soluções estudadas: solução sem restrições (SolMat); solução com restrições da largura dos perfis (SolConst); e a solução considerando os tramos igualmente espaçados (SolDiv).

Tabela 4.16: Momentos flectores máximos (kN.m) das diferentes soluções - G5×5-10×10

	SolMat	SolConst	SolDiv
Tipo I.a	12,99	153,89	166,56
Tipo I.b	11,57	100,04	122,26
Tipo II	10,00	96,00	142,15

4.2.2 Grelha de 10×10 m com 5×8 tramos - Caso G5×8-10×10

É agora estudado um caso em que é mantida a geometria quadrada da grelha (10×10 m), mas são acrescentados três tramos na direcção y, ou seja, são acrescentadas três vigas à grelha, como ilustrado na Figura 4.24. É esperado que este acréscimo de material diminua os esforços aplicados.

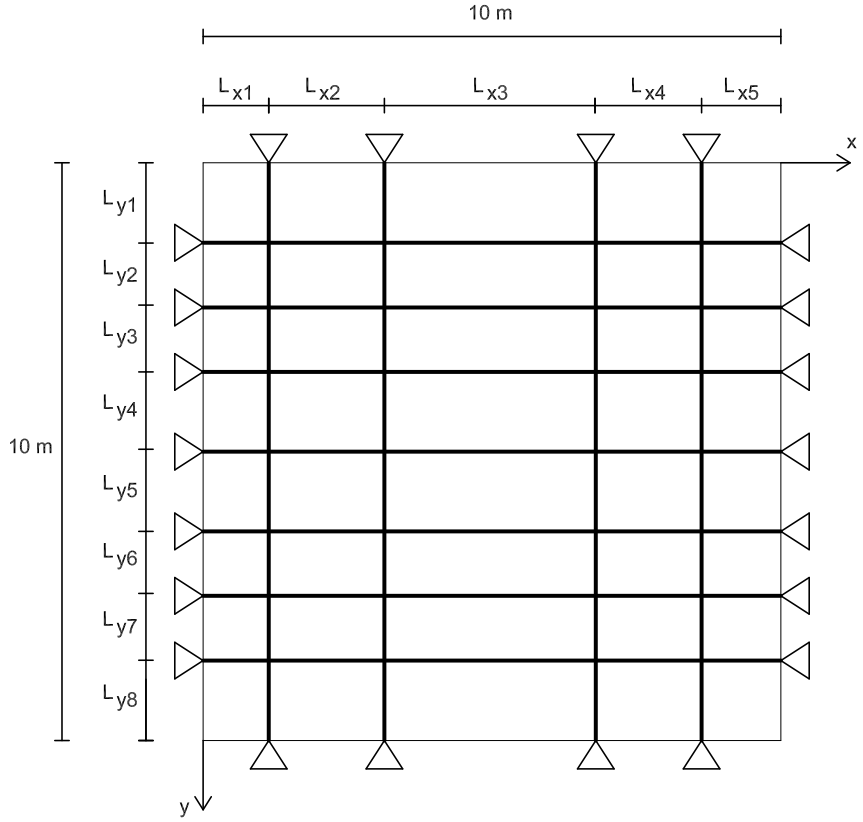


Figura 4.24: Grelha caso G5×8-10×10

Desta forma, é acrescentada uma variável de projecto em relação ao problema anterior, sendo o vector das variáveis definido da seguinte maneira:

$$X = [Lx_1 = Lx_5, Lx_2 = Lx_4, Lx_3, Ly_1 = Ly_8, Ly_2 = Ly_7, Ly_3 = Ly_6, Ly_4 = Ly_5] \tag{4.8}$$

O problema fica também sujeito a novas restrições de igualdade:

$$2Lx_1 + 2Lx_2 + Lx_3 = 10 \tag{4.9}$$

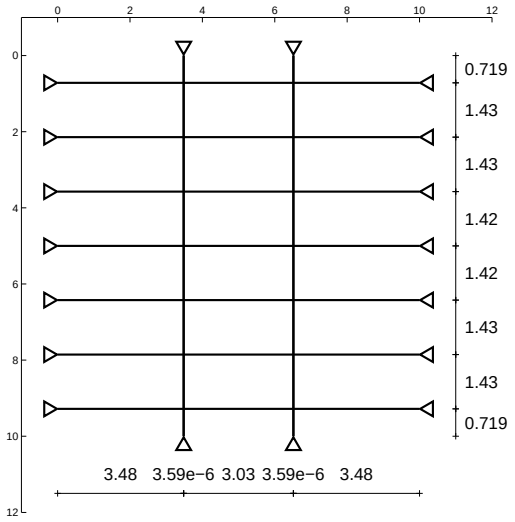
$$2Ly_1 + 2Ly_2 + 2Ly_3 + 2Ly_4 = 10 \tag{4.10}$$

Na Tabela 4.17 são apresentados os valores óptimos, obtidos a partir dos algoritmos genéticos, novamente para os três tipos de grelhas considerado. Na Figura 4.25 é ilustrada a disposição das vigas para esses valores.

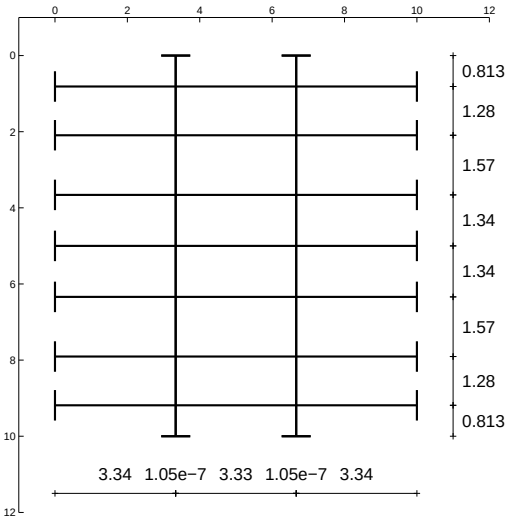
Tabela 4.17: Solução óptima - caso G5×8-10×10

Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.a	200	[0,1;3]	1×10^{-6}	7,5576	61
Tipo I.b	200	[0,1;4]	1×10^{-6}	6,7506	57
Tipo II	200	[0,5;4]	1×10^{-8}	6,9072	90

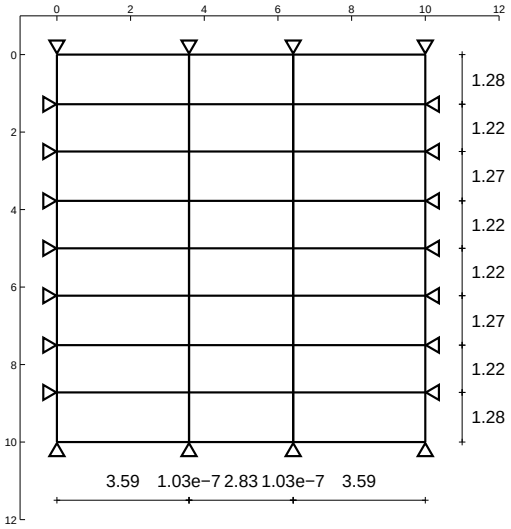
	Valores óptimos das variávies (ga)						
Tipo	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Lx_8$ (m)	$Ly_2 = Lx_7$ (m)	$Ly_3 = Lx_6$ (m)	$Ly_4 = Lx_5$ (m)
Tipo I.a	3,4843	$3{,}590\times10^{-6}$	3,0312	0,7185	1,4257	1,4306	1,4248
Tipo I.b	3,3361	$1{,}052\times10^{-7}$	3,3275	0,8132	1,2801	1,5671	1,3393
Tipo II	3,5869	$1{,}027\times10^{-7}$	2,8259	1,2813	1,2209	1,2740	1,2236



(a) Grelha tipo I.a



(b) Grelha tipo I.b



(c) Grelha tipo II

Figura 4.25: Dimensionamento óptimo para o caso G5×8-10×10

Como se pode verificar pela Tabela 4.17, o dimensionamento óptimo ocorre novamente quando as vigas numa das direcções tendem a juntar-se. Neste caso, as vigas que se juntam são as vigas com maior comprimento. Isto deve-se ao facto de os maiores momentos flectores aparecerem nas vigas de menor vão, o que leva o processo de optimização a dar maior rigidez à direcção perpendicular, de modo a minimizar esses momentos.

Procedendo do mesmo modo que o problema anterior, são alterados os limites inferiores das variáveis para o valor da largura dos perfis, de forma a encontrar uma solução óptima construtivamente possível. Na Tabela 4.18 são apresentados os valores óptimos para essa solução, e na Figura 4.26 é ilustrada a disposição dos elementos da grelha para esses valores.

Tabela 4.18: Solução óptima - Caso G5×8-10×10-construtivo

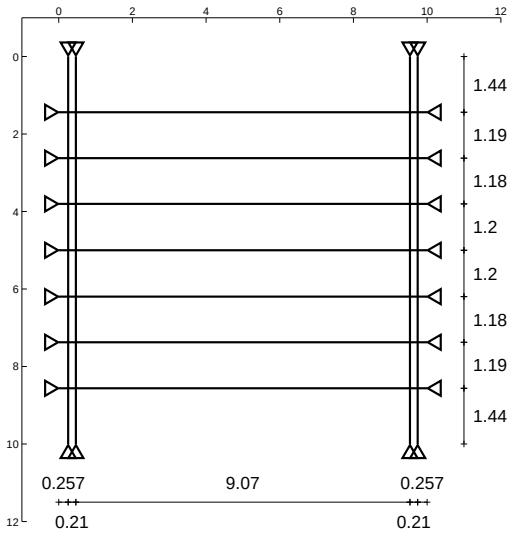
Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.a	300	[0,1;4]	1×10^{-6}	95,9807	58
Tipo I.b	200	[1;4]	1×10^{-6}	72,8250	74
Tipo II	250	[0,5;4]	1×10^{-8}	69,1440	51

Tipo	Valores óptimos das variávies (ga)						
	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Lx_8$ (m)	$Ly_2 = Lx_7$ (m)	$Ly_3 = Lx_6$ (m)	$Ly_4 = Lx_5$ (m)
Tipo I.a	0,2572	0,20962	9,0661	1,4387	1,1858	1,1784	1,1968
Tipo I.b	2,6113	0,3424	4,0916	3,3757	1,0925	0,26641	0,2649
Tipo II	3,5910	1,0879	0,6411	4,3978	0,2016	0,2001	0,2001

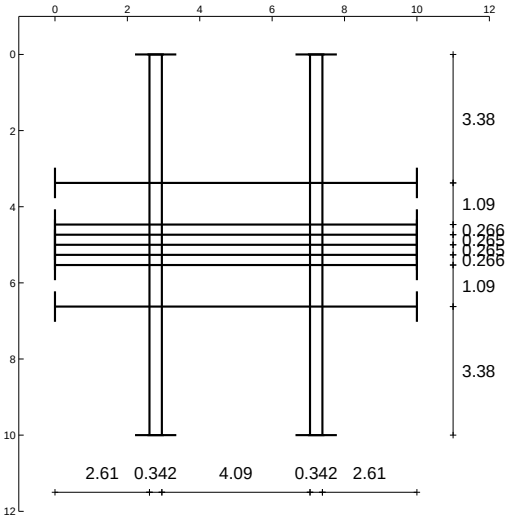
Observando novamente as soluções óptimas obtidas com a solução considerando os espaçamentos iguais entre vigas nas respectivas direcções (Tabela 4.19), verifica-se que os resultados são análogos aos obtidos no caso anterior, ou seja, a solução óptima sem restrições apresenta um valor de momento flector máximo bastante mais baixo que as restantes soluções (aproximadamente 90% em todos os tipos de grelha), e existe uma redução significativa de momentos entre as duas soluções construtivamente possíveis (cerca de 26,5% para o tipo I.a, 24,6% para o tipo I.b e 40,72% para o tipo II). Comparando também este caso com o anterior, verifica-se que o acréscimo de material (30% para as grelhas tipo I e 21% para a grelha tipo II) reduz os esforços aproximadamente entre 20 a 40% para todas as soluções.

Tabela 4.19: Momentos flectores máximos (kN.m) das diferentes soluções - G5×8-10×10

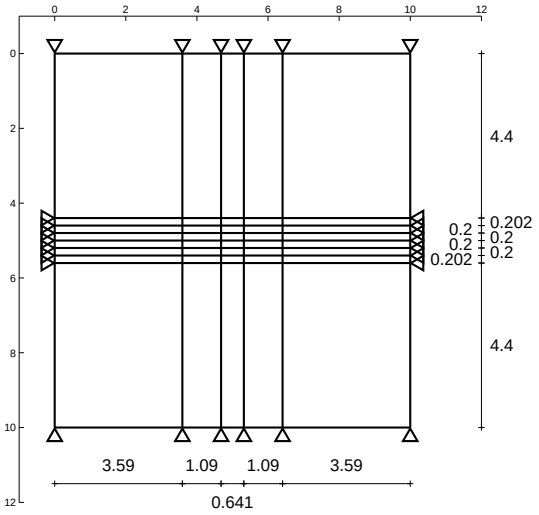
	SolMat	SolConst	SolDiv
TipoI.a	7,55	95,98	130,79
TipoI.b	6,75	72,83	96,64
TipoII	6,91	69,14	116,65



(a) Grelha tipo I.a



(b) Grelha tipo I.b



(c) Grelha tipo II

Figura 4.26: Dimensionamento óptimo para o caso G5×8-10×10-construtivo

4.2.3 Grelha de 10×10 m com 5×5 tramos - Caso G5×5-10×15

É agora estudado um caso em que as dimensões da grelha deixam de ser iguais em ambas as direcções, passando a grelha a ser rectangular, mas o número de tramos da grelha é igual em ambas as direcções. Neste caso considera-se que o vão na direcção y tem 15 m, permanecendo o vão na direcção x com 10 m, como representado na Figura 4.27.

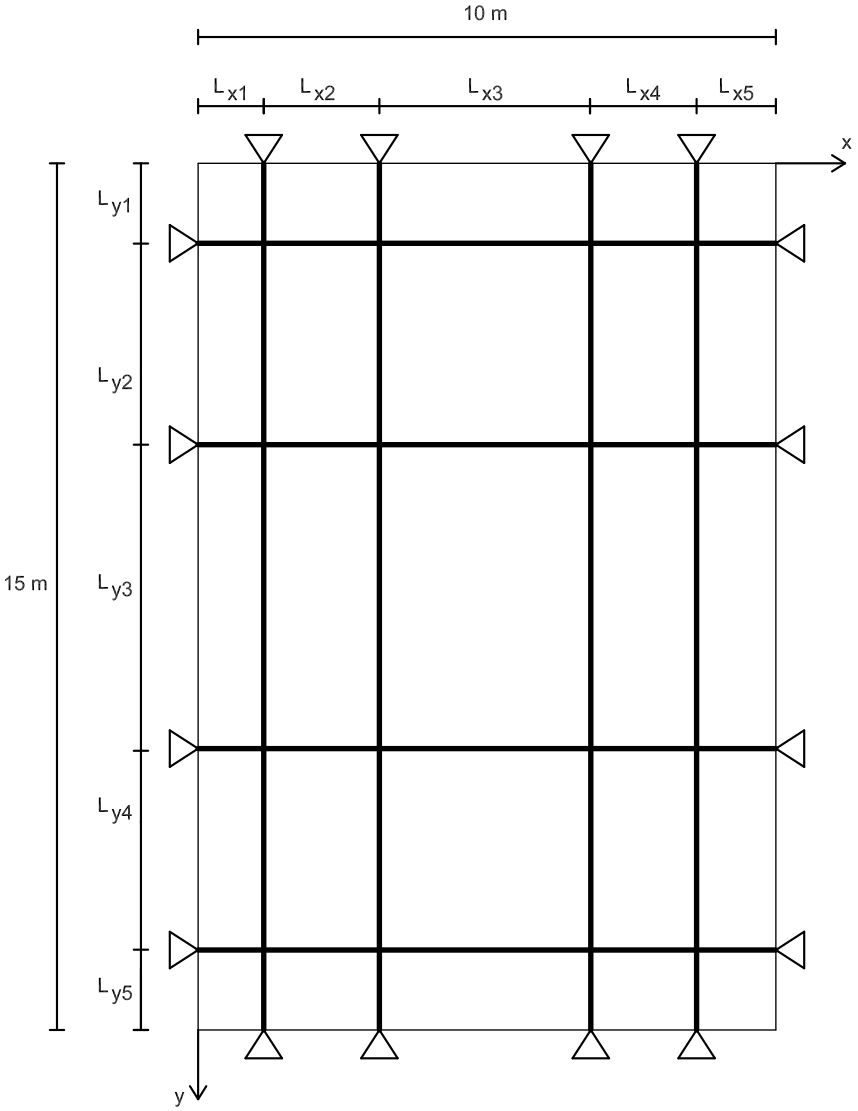


Figura 4.27: Grelha caso G5×5-10×15

Como o número de tramos em cada direcção é o mesmo que no primeiro caso de estudo, o vector das variáveis de projecto, X, é o mesmo. Contudo, as restrições do problema passam a ser as seguintes:

$$2Lx_1 + 2Lx_2 + Lx_3 = 10 \tag{4.11}$$

$$2Ly_1 + 2Ly_2 + Ly_3 = 15 \tag{4.12}$$

Devido ao facto das diferenças dos valores obtidos para cada tipo de grelha serem semelhantes aos obtidos nos dois casos anteriores, apenas são apresentados os resultados para a grelha do tipo I.b. Na Tabela 4.20 são então apresentados os valores óptimos apenas para esse tipo de grelha, e na Figura 4.28 é ilustrada a disposição dos elementos para esses valores.

Tabela 4.20: Solução óptima - Caso G5×5-10×15

Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.b	200	[1;3]	1×10^{-6}	18,2624	63

Tipo	Valores óptimos das variávies (ga)					
	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Ly_5$ (m)	$Ly_2 = Ly_4$ (m)	Ly_3 (m)
Tipo I.b	3,3329	$1{,}000\times10^{-7}$	3,3333	2,1863	3,5167	3,5931

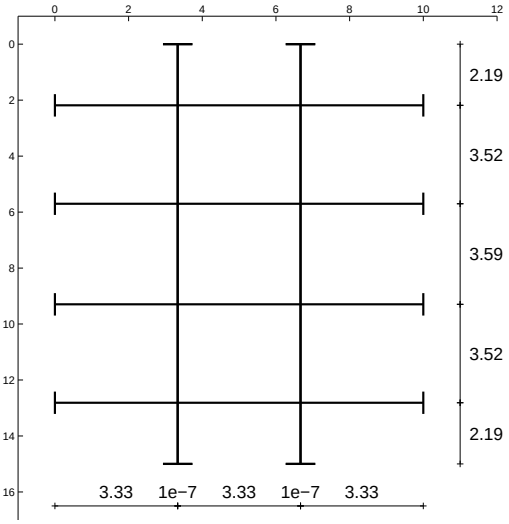


Figura 4.28: Dimensionamento óptimo para a grelha tipo I.b no caso G5×5-10×15

Novamente, a solução óptima ocorre quando há junção das vigas com maior comprimento, ou seja, as vigas perpendiculares ao eixo x. Isto ocorre pelos motivos já explicados no caso anterior.

Assim sendo, é reformulado novamente o problema considerando como limites inferiores das variáveis o valor da largura dos perfis. Na Tabela 4.21 são apresentados os valores óptimos dessa solução para a grelha do tipo I.b, e na Figura 4.29 é ilustrada a disposição dos elementos da grelha para esses valores.

Tabela 4.21: Solução óptima - Caso G5×5-10×15-construtivo

Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.b	300	[1;5]	1×10^{-6}	188,890	51

Tipo	Valores óptimos das variáveis (ga)					
	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Ly_5$ (m)	$Ly_2 = Ly_4$ (m)	Ly_3 (m)
Tipo I.b	0,2156	4,2334	1,1012	3,8769	3,4524	0,3405

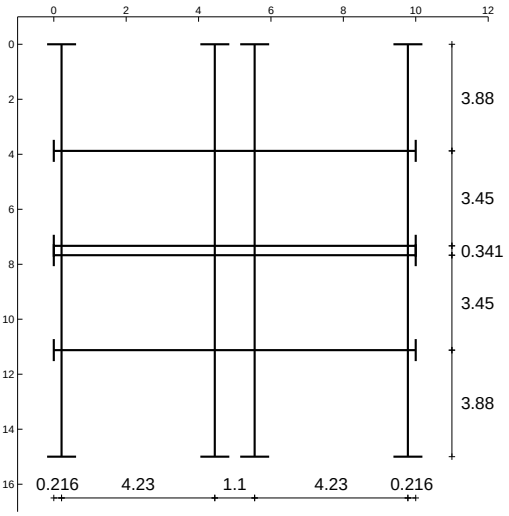


Figura 4.29: Dimensionamento óptimo para a grelha tipo I.b para o caso G5×5-10×15-construtivo

São novamente comparados os valores de momento flector máximo obtidos, em ambas as soluções óptimas com os da solução construtiva mais usual (Tabela 4.22), verificando-se que o momento flector máximo óptimo da solução sem restrições é bastante inferior (cerca de 90% a 100%) relativamente às restantes soluções, e que comparando ambas as soluções construtivamente possíveis verifica-se que a optimização reduz significativamente o momento flector máximo (cerca de 27%). Como era esperado (atendendo ao maior vão), verifica-se também que todas as soluções deste caso apresentam esforços maiores do que as soluções respectivas do primeiro caso de estudo.

Tabela 4.22: Resumo dos momentos flectores máximos das soluções - G5×5-10×15

	SolMat	SolConst	SolDiv
TipoI.b	18,26	188,89	260,88

4.2.4 Grelha de 10×10 m com 5×8 tramos - Caso G5×8-10×16

O último caso a ser estudado para as estruturas em grelha é um caso em que tanto as dimensões como o número de tramos diferem de uma direcção para a outra. Considera-se, desta forma, na direcção x cinco tramos e um $Lx_{total} = 10$ m, e na direcção y oito tramos e um $Ly_{total} = 16$ m, como ilustrado na Figura 4.30.

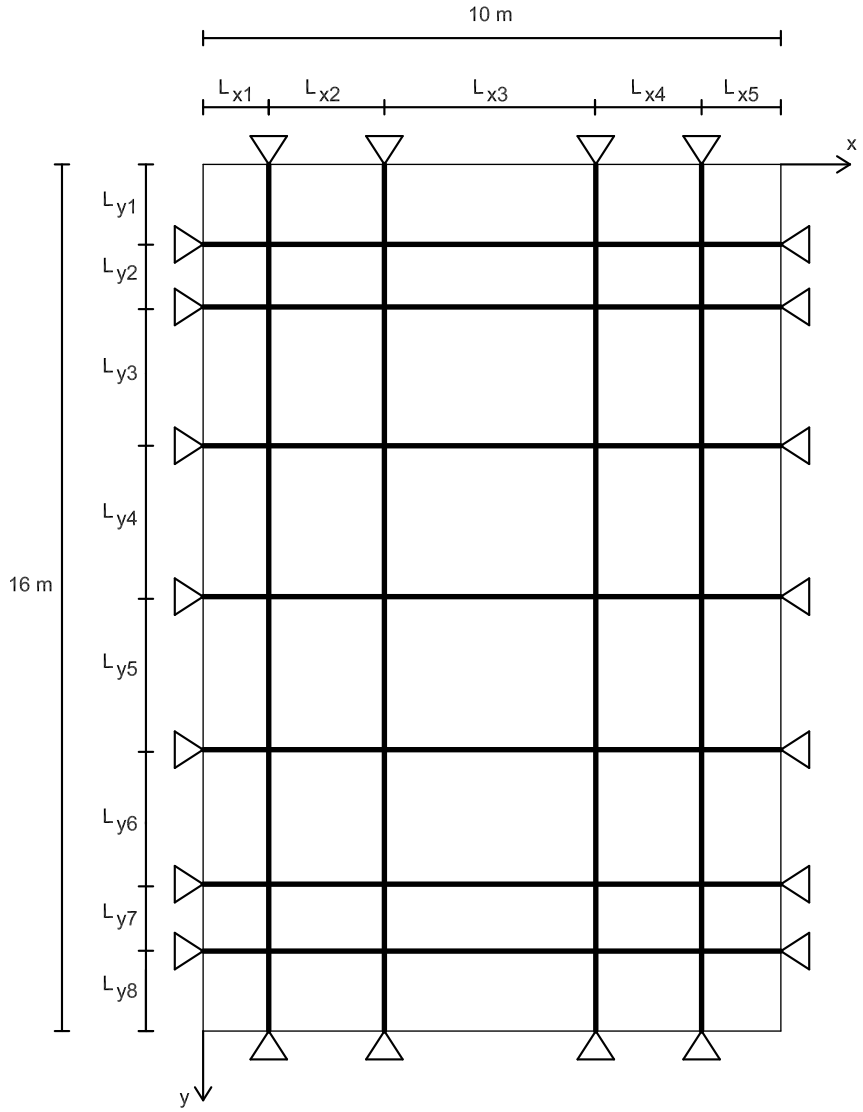


Figura 4.30: Grelha caso G5×8-10×16

O problema de optimização tem então o seguinte vector de variáveis de projecto:

$$X = [Lx_1 = Lx_5, Lx_2 = Lx_4, Lx_3, Ly_1 = Ly_8, Ly_2 = Ly_7, Ly_3 = Ly_6, Ly_4 = Ly_5] \tag{4.13}$$

E está sujeito às seguintes restrições de igualdade:

$$2Lx_1 + 2Lx_2 + Lx_3 = 10 \tag{4.14}$$

$$2Ly_1 + 2Ly_2 + 2Ly_3 + 2Ly_4 = 16 \tag{4.15}$$

Tal como no caso anterior, apenas serão apresentados os resultados da optimização para grelhas do tipo I.b, pelos motivos anteriormente apresentados. Na Tabela 4.23 são apresentados os valores óptimos para o tipo de grelha considerado e na Figura 4.31 é ilustrada a disposição dos elementos da grelha para esses valores.

Tabela 4.23: Solução óptima - Caso G5×8-10×16

Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.b	200	[1;3]	1×10^{-6}	10,911	71

	Valores óptimos das variáveis (ga)						
Tipo	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Lx_8$ (m)	$Ly_2 = Lx_7$ (m)	$Ly_3 = Lx_6$ (m)	$Ly_4 = Lx_5$ (m)
Tipo I.b	3,330	$9{,}875\times10^{-7}$	3,341	1,146	1,875	2,710	1,999

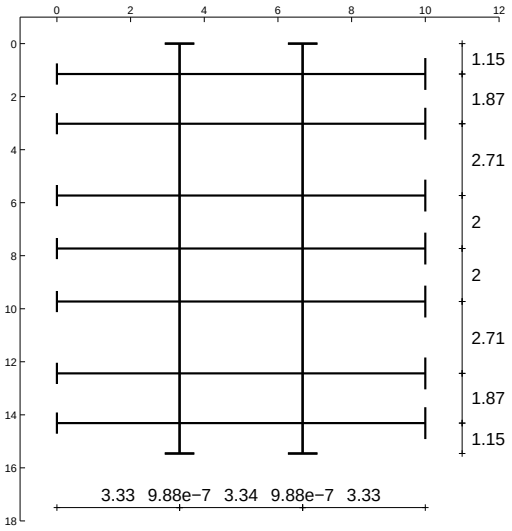


Figura 4.31: Dimensionamento óptimo para a grelha tipo I.b no caso G5×8-10×16

Tal como em todos os casos anteriores, o dimensionamento óptimo ocorre quando as vigas numa das direcções tendem a juntar-se. Neste caso são novamente as vigas com maior comprimento que se juntam pelas razões já apresentadas.

Repetindo o procedimento anterior, é reformulado o problema de optimização alterando os limites inferiores das variáveis para o valor da largura dos perfis, de forma a obter-se uma solução óptima construtivamente possível. Na Tabela 4.24 são apresentados os valores óptimos dessa reformulação para a grelha do tipo I.b, e na Figura 4.32 é ilustrada a disposição dos elementos da grelha para esses valores.

Tabela 4.24: Solução óptima - Caso G5×8-10×16-construtivo

Tipo	$DimPop$	PIR	ε	M^* (kN.m)	Nº de gerações
Tipo I.b	300	[1;5]	1×10^{-6}	119,5258	51

	Valores óptimos das variávies (ga)						
Tipo	$Lx_1 = Lx_5$ (m)	$Lx_2 = Lx_4$ (m)	Lx_3 (m)	$Ly_1 = Lx_8$ (m)	$Ly_2 = Lx_7$ (m)	$Ly_3 = Lx_6$ (m)	$Ly_4 = Lx_5$ (m)
TipoI.b	0,4304	0,2013	8,7357	2,2331	1,3953	2,3497	2,0214

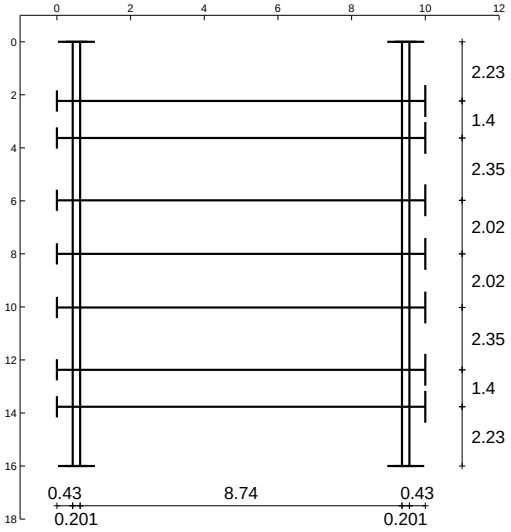


Figura 4.32: Dimensionamento óptimo para a grelha tipo I.b para o caso G5×8-10×16-construtivo

Na Tabela 4.25 são apresentados os valores dos momentos flectores máximos óptimos para ambas as soluções anteriores e para a solução mais usual construtivamente, que apresenta iguais espaçamentos entre vigas em cada direcção. Comparando todos os valores verifica-se novamente que a solução óptima sem restrições apresenta um valor de momento flector máximo bastante mais baixo que as outras duas (cerca de 90% a 100%), e que a solução óptima construtivamente possível apresenta também um valor de momento flector máximo mais reduzido do que o da solução com os espaçamentos todos iguais (cerca de 37,2%). Comparando este caso com o anterior, verifica-se que, apesar do pequeno aumento da dimensão do painel na direcção y (7%), este caso apresenta esforços mais baixos entre 26% a 40% para todas as

soluções, no entanto, apresenta uma quantidade de material maior que o do caso anterior (28%), para a grelha do tipo I.b.

Tabela 4.25: Resumo dos momentos flectores máximos (kN.m) das soluções - G5×8-10×10

	SolMat	SolConst	SolDiv
Tipo I.b	10,91	119,53	190,47

4.2.5 Discussão dos resultados

Analisando os resultados obtidos para os casos de grelha considerados, verifica-se que o dimensionamento óptimo ocorre quando as vigas numa das direcções tendem a juntar-se, fazendo com que a estrutura tenha maior rigidez numa direcção do que na outra.

Estes resultados são coerentes com os apresentados em [5], em que apesar de o problema de optimização ter como objectivo minimizar o peso de uma estrutura em cruz, variando as secções transversais, a optimalidade ocorre quando uma das vigas é mais rígida que a outra.

Estas soluções apresentam valores de momento flector máximo bastante mais baixos (na ordem dos 90% a 100%) do que aqueles que se obtêm nas soluções óptimas construtivamente possíveis e nas soluções com as vigas igualmente espaçadas.

Comparando as soluções construtivamente possíveis, verifica-se que o processo de optimização consegue encontrar soluções construtivamente possíveis em que os valores de momento flector máximo são mais baixos cerca de 7% a 44% em comparação com os valores obtidos para a solução com as vigas igualmente espaçadas.

Capítulo 5

Considerações finais

5.1 Conclusões

O objectivo desta dissertação consistia no desenvolvimento de um procedimento capaz de realizar a optimização dos momentos flectores em vigas e grelhas, em função da posição e rigidez dos apoios. A partir dos casos analisados, verificou-se que o objectivo foi cumprido, já que o processo de optimização efectuado permite uma redução significativa dos esforços que actuam nas estruturas.

Ambos os métodos de optimização utilizados nesta dissertação e apresentados no capítulo 3, conseguiram apresentar soluções óptimas para os problemas de optimização apresentados no capítulo 4. O método de Nelder-Mead, utilizado na optimização de vigas contínuas, conseguiu obter mínimos globais para as respectivas funções objectivo sem grande dificuldade para os casos mais simples, ou seja, com poucas variáveis e apoios rígidos. Nos casos em que se consideram apoios flexíveis, ou seja, casos mais complexos, o método ficou "preso"em mínimos locais mais frequentemente. Contudo, quando foram efectuadas as simplificações de simetria nesses casos, o método de Nelder-Mead conseguiu encontrar os mínimos globais das funções.

Os algoritmos genéticos foram utilizados nos problemas de grelhas, que são problemas mais complexos, onde o número de variáveis de projecto é por norma maior. Os algoritmos genéticos conseguiram efectivamente encontrar mínimos para os problemas apresentados; no entanto, devido à complexidade destes não se tem a garantia de que os mínimos sejam mínimos globais. Constatou-se que a variação de certos parâmetros, como o número de indivíduos da população inicial e os limites do domínio da mesma, têm um papel importante na capacidade do método de encontrar a solução óptima.

Quanto ao dimensionamento óptimo das vigas, verificou-se que, para o caso das vigas apoiadas em apoios rígidos, este ocorre quando o comprimento dos tramos toma valores tais que os momentos negativos nos apios se igualam entre si. Para o caso das vigas com apoios flexíveis, o dimensionamento óptimo ocorre quando o comprimento dos tramos e as rigidezes dos apoios tomam valores que permitam os momentos positivos nos vãos igualarem os momentos negativos nos apoios. Desta forma, os dimensionamentos com apoios flexíveis apresentam valores de momento flector máximo mais baixos que os dimensionamentos com os apoios rígidos.

No que toca aos problemas de optimização de grelhas, verifica-se que o dimensionamento óptimo ocorre quando as vigas numa das direcções tendem a juntar-se. Esta ocorrência faz com que a grelha tenha uma maior rigidez numa direcção do que na outra, aproximando as soluções à solução ideal que seria

a existência de apoios nos nós de cruzamento das vigas, de modo a que estes contrabalançassem os momentos positivos dos vãos com momentos negativos. No entanto, estas soluções são impossíveis de serem realizadas, e como tal, reformularam-se os problemas introduzindo limites inferiores às variáveis, de forma a que o espaçamento entre vigas não seja menor que a largura dos perfis. Esta reformulação permitiu a obtenção de dimensionamentos óptimos construtivamente possíveis, que apresentam valores de momento flector máximo mais baixos do que aqueles que surgem quando se consideram soluções com vigas igualmente espaçadas.

5.2 Desenvolvimentos futuros

Desta dissertação poderão surgir diversos desenvolvimentos, sendo que alguns poderão ser os seguintes:

- Considerar outros parâmetros como variáveis de projecto para além da flexibilidade e posição dos apoios, tais como as secções e os materiais a utilizar nas estruturas;
- Aplicar o processo de optimização a outro tipo de estruturas, como por exemplo pórticos;
- No caso das grelhas, reformular a função objectivo de forma a que não sejam apenas minimizados os momentos flectores máximos, mas também se tenha em consideração a influência do momentos torsores;
- Estender a análise estrutural a uma análise plástica, de modo a obter esforços mais reduzidos;
- Utilizar outros métodos de optimização como as colónias de formigas [32];
- Formular um problema de optimização que relacione os esforços com o preço e/ou a quantidade de material.

Bibliografia

[1] Argyris, J.H. and Kelsey, S. Energy theorems and structural analysis. *Aircraft Engineering*, 27, May 1955.

[2] Arora, J.S. *Introduction to Optimum Design*. McGraw-Hill, New York, 1989.

[3] Bathe, K.J. *Finite Element Procedures*. Prentice-Hall, 1996.

[4] Brás, L. *Optimização de Estruturas: Desenvolvimento de Software Orientado para Objectos*. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Setembro 2003.

[5] Cardoso, J., C.P.e.A.J. Aplicação de algoritmos genéticos em optimização estrutural. *Métodos numéricos en engenieria*, 5, 2000.

[6] Cardoso, J. and Coelho, P. *Métodos Computacionais em Engenharia Mecânica - Apontamentos*. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2011.

[7] Clough, R.W. The finite element in plane stress analysis. *Proc. 2nd ASCE Conf. on Electronic Computation*, September 1960.

[8] Cohn, M.Z. Theory and practice of structural optimization. *Structural Optimization*, (7):20–31, 1994.

[9] Conn, A., S.K. and Vicente, L. *Introduction to Derivative-Free Optimization*. SIAM, Philadelphia, 2009.

[10] Cook, R.D. *Finite Element Modelling for Stress Analysis*. John Wiley & Sons, Inc, 1995.

[11] Correia, A. *Métodos de Pesquisa Directa: Optimização Não Linear*. Ph.D. thesis, Universidade de Trás-Os-Montes e Alto Douro, Outubro 2010.

[12] Courant, R. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49:1–23, 1943.

[13] Cruz, P. *Optimização de Forma de Estruturas Metálicas em Regime Elástico*. Master's thesis, Universidade de Aveiro, 2007.

[14] Ferreira, A. *Elementos Finitos em MATLAB*. 2007.

[15] Ghali, A., N.A. and Brown, T. *Structural Analysis - A unified classical and matrix approach*. Spon Press, 6th edition, 2009.

[16] Haftka, R. and Gurdal, Z. *Elements of Structural Optimization*. Kluwer Academic Publishers, 1990.

[17] Holland, J.H. *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press, 1975.

[18] Hrenikoff, A. Solution of problems in elasticity by the framework method. *Journal of Applied Mechanics*, 8(4):A169–A175, 1941.

[19] Jesus, R. *Optimização da Forma Estrutural de uma Barragem*. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Julho 2011.

[20] Kattan, P. *MATLAB Guide to Finite Elements - An Interactive Approach*. Springer, second edition, 2008.

[21] Kikuchi, N., C.K.T.T. and Taylor, J. Adaptive finite element methods for shape optimization of linear elastic structures. *Computer methods in applied mechanics and engineering*, (57):67–89, 1985.

[22] Kripka, M. and Antunes, H. Otimização dos esforços em vigas pela alteração na posição dos apoios. *Engenharia Estudo e Pesquisa*, 2(2):129–142, 1999.

[23] Kwon, Y.W. and Bang, H. *The Finite Element Method using MATLAB*. CRC Press, 1996.

[24] Lagarias, J., R.J.W.M. and Wright, P. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM J. Optim*, 9(1):112–147, 1998.

[25] Lewis, R., T.V. and Trosset, M. Direct search methods: Then and now. *J. Comput. Appl. Math.*, 124:191–207, 2000.

[26] Macedo, R. *Optimização Estrutural na Análise Não Linear dos Modelos de Escoras e Tirantes*. Master's thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Dezembro 2011.

[27] Maia, F. *Análise Paramétrica e Optimização Estrutural Utilizando o Software Comercial ANSYS*. Master's thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Novembro 2011.

[28] Maxwell, C. On reciprocal figures, frames and diagrams of forces. *Scientific papers*, 2:175–177, 1890.

[29] McKinnon, K. Convergence of the nelder-mead simplex method to a nonstationary point. *SIAM J. Optim*, 9:148–158, 1998.

[30] Michell, A. The limit of economy of material in frame structures. *Philosophical Magazine*, 8:589–597, 1904.

[31] Nelder, J. and Mead, R. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[32] Osman, I.H. and Kelly, J.P. meta-heuristics: an overview. *Kluwer academic publishers*, 1997.

[33] Ruas, J. *Optimização Não-Linear Aplicada ao Dimensionamento Inicial de Navios*. Master's thesis, Instituto Superior Técnico, Outubro 2010.

[34] Saka, M., D.A. and Malhas, F. Optimum spacing design of grillage systems using a genetic algorithm. *Advances in Engineering Software*, 31:863–873, 2000.

[35] Schmit, L.A. Structural design by systematic synthesis. *Proc. 2d Conf. Electron. Computation ASCE*, pages 105–132, September 1960.

[36] Silva, V. *Mecânica e resistência dos materiais*. Edição de livros técnicos, Lda, 3ª edition, 2004.

[37] Spillers, W. and MacBain, K. *Structural Optimization*. Springer, 2009.

[38] Tavares, J.M. *Introdução ao Método dos Elementos Finitos*. Faculdade de Engenharia da Universidade do Porto, 1998.

[39] Timoshenko, S. *History of strength of materials*. McGraw-Hill New York, 1953.

[40] Turner, M. J., C.R.W.M.H.C. and Topp, L.J. Stiffness and deflection analysis of complex structures. *Journal of the Aeronautical Sciences*, 23:805–823, 1956.

[41] Yeniay, O. Penalty function method for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10:45–56, 2005.

Apêndice A

Algortimo para a análise de vigas

A.1 Programa: fobjectivo_vigas.m

```
function Mmax = fobjectivo(xi)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%                    Programa para a análise estrutural                         %
%                 de vigas com apoios rígidos ou flexíveis                      %
%                            baseado no MEF                                     %
%                                                                               %
%                            Guilherme Martins                                  %
%                                 27945                                         %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    l1=xi(1); %introduzir os comprimentos dos tramos no vector das
    l2=xi(2); %variáveis de projecto xi, na respectiva posição
    l3=xi(3);


    Ltotal = 10;
    l4 = Ltotal - (l1+l2+l3); %último tramo
    l = [l1,l2,l3,l4];

%     k1 = xi(4); % efectuar o mesmo procedimento dos comprimentos para a
%     k2 = xi(5); % a rigidez dos apoios. Caso se considerem os apoios
%     k3 = xi(6); % como rígidos meter em comentário

    kmola = []; % caso os apoios sejam flexíveis, colocar os respectivos
                % valores neste vector à semelhança dos comprimentos



    %%%%%%%%%%%%        Propriedades dos materiais      %%%%%%%%%%


    syms x L E I P % assume objectos simbólicos

    E= 210e6;       % Módulo de Young 210 GPa
    I= 27.72e-6;    % Momento de Inércia HEB 200

    P=20;          % Carga distribuida (kN/m)

    ntramos=length(l);          % nº de tramos
    napoios=ntramos-1;          % nº de apoios

    ndiv = 1;       % numero de EF por tramo
```

```
    %%%%%%%%%%            COORDENADAS DOS PONTOS         %%%%%%%%%%

    e = 1;
    laux = 0;
    xx(e) = laux;
    for i = 1:ntramos
        for j = 1:ndiv

            laux = l(i)/ndiv + laux;

            e = e + 1;

            xx(e) = laux;

        end
    end

%     xx; % vector das coordenadas dos nós



    %%%%%%%%%%%%       Numeração dos elementos e nós    %%%%%%%%%%%%

    nelem = ndiv*ntramos; %numero de elementos
    element = zeros(nelem,2);

    for i = 1:nelem

        element(i,1) = i;
        element(i,2) = i+1;

    end

%     element;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            CÁLCULO DA MATRIZ DE RIGIDEZ E VECTOR DAS CARGAS          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    numnode = length(xx); %numero de nos
    ndofs = 2*numnode; % numero de graus de liberdade

    U=zeros(ndofs,1); %vector dos deslocamentos nodais
    f=zeros(ndofs,1); %vector das solicitações
    K=zeros(ndofs,ndofs); %matriz de rigidez

    for i = 1:nelem

        index=element(i,:);

        Le = xx(index(2))-xx(index(1));
```

```
        % Matriz funções de forma (polinómio de hermite)
        N=[1-3*x^2/Le^2+2*x^3/Le^3;x-2*x^2/Le+x^3/Le^2;
            3*x^2/Le^2-2*x^3/Le^3;-x^2/Le+x^3/Le^2];

        % Cálculo da matriz de rigidez de cada elemento
        Ndd=diff(diff(N),x);

        Ke=int(E*I*Ndd*(Ndd'),x,0,Le);

        %Cálculo do vector de carga para carga uniformemente distribuida
        Fe=-int(P*N,x,0,Le);

        % Assemblagem na matriz de rigidez e no vector das cargas
        posgdl=[2*i-1,2*i,2*i+1,2*i+2];
        K(posgdl,posgdl)=K(posgdl,posgdl)+Ke;
        f(posgdl,1)=f(posgdl,1)+Fe;

    end

%     K;
%     f;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          CONDIÇÕES DE FRONTEIRA                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%          APOIOS FLEXIVEIS           %%%%%%%%%%%%%

    Kmola = zeros(ndofs,ndofs);

    % código para que todos os apoios sejam flexiveis
    e = 1;
    for i = 1:length(l)-1
        mofs(e) = ndiv*e+1;
        e = e + 1;
    end

    mofs;
    dmofs = 2*mofs-1;

    for i = 1:length(kmola)

        Kmola(dmofs(i),dmofs(i)) = Kmola(dmofs(i),dmofs(i)) + kmola(i);

    end

    Kmola;

    K = K + Kmola;
```

```
%%%%%%%%%%%%%%%            APOIOS RIGIDOS           %%%%%%%%%%%%%%%

% código para que todos os apoios sejam rigidos
e = 1;
for i = 1:length(l)-1
    lofs(e) = ndiv*e+1;
    e = e + 1;
end

lofs;
dofs = 2*lofs-1;

% dofs = []

activos=setdiff([1:ndofs]',[dofs]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            SOLUÇÃO DO PROBLEMA                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

U=K(activos,activos)\f(activos);
U1=zeros(ndofs,1);
U1(activos)=U;

disp('deslocamentos ')
U = U1

disp('reaccoes ')
R=K*U-f

% disp('reaccoes nas molas')
% Rm = kmola'.*U(dmofs)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             ESFORÇOS INTERNOS                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:nelem

    index=element(i,:);

    Le = xx(index(2))-xx(index(1));

    % Matriz funções de forma (polinómio de hermite)
    N=[1-3*x^2/Le^2+2*x^3/Le^3;
        x-2*x^2/Le+x^3/Le^2;
        3*x^2/Le^2-2*x^3/Le^3;
        -x^2/Le+x^3/Le^2];

    % Cálculo da matriz de rigidez de cada elemento
    Ndd=diff(diff(N),x);
```

```
    Ke=int(E*I*Ndd*(Ndd'),x,0,Le);

    %Cálculo do vector de carga para carga uniformemente distribuida
    Fe=-int(P*N,x,0,Le);

    fe(:,i)= Ke*U1([2*i-1,2*i,2*i+1,2*i+2])-Fe;

end

fe = double(fe);

l=1;
for i=1:nelem
    for j=1:2

        V(l)=fe(2*j-1,i); % valor do esf transverso V
        if j==2
            V(l)=-V(l); %correcção do sinal para V positivo
                        %(atenção ao referencial do EF! O sinal +
                        % deve ser para baixo no nó 2 do EF)
        else
        end
        M(l)=fe(2*j,i);  %colhe valor do Momento flector M


        if j==1
            M(l)=-M(l); %correcção do sinal para M positivo
                        %(atenção ao referencial do EF! O sinal +
                        %deve ser horário no nó 1 do EF)
        else
        end
        l=l+1;
    end
end

V;
M;

% verificação se existem maximos momentos interiores do EF

pp=1;

for i=1:nelem

   index=element(i,:); %nós do elemento

   L = xx(index(2))-xx(index(1));

   V0=V(2*i-1);
   M0=M(2*i-1);
   xint=abs(V0/P);

   if xint<L
       Mint=M0+V0*xint-P*(xint^2)/2;
```

```
        else
            Mint=0;
        end

        Mf(pp)=M(2*i-1);
        Mf(pp+1)=Mint;
        Mf(pp+2)=M(2*i);

        pp=pp+3;

    end

    Mf; %vector do M com os valores a meio vão dos elementos

    Mmax = max(abs(Mf)) %valor do máximo valor em módulo do M

    % guardar os resultados em ficheiros .dat
    parametros=[xi(1),xi(2),xi(3)];
    dlmwrite('Parametro4.dat',parametros,'-append','newline', 'pc')

    vobjectivo=[Mmax];
    dlmwrite('obj4.dat',vobjectivo,'-append','newline', 'pc')

end
```

Apêndice B

Algortimo para a análise de grelhas

B.1 Programa: fobjectivo_tipoIa.m

```
function Mmax = fobjectivo_tipoIa(xi)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%                   Programa para a análise estrutural                          %
%                de grelhas do tipo I.a com apoios rígidos                      %
%                            baseado no MEF                                     %
%                                                                               %
%                           Guilherme Martins                                   %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%           Variaveis do problema            %%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Espaçamentos dos apoios segundo x
    lx1 = xi(1);
    lx2 = xi(2);
    lx3 = xi(3);
    lx4 = xi(2);
    lx5 = xi(1);

    % lxtot = 10;


    lxt = [lx1,lx2,lx3,lx4,lx5]; %vector com os espaçamentos em x


    % Espaçamentos dos apoios segundo y
    ly1=xi(4);
    ly2=xi(5);
    ly3=xi(6);
    ly4=xi(7);
    ly5=xi(7);
    ly6=xi(6);
    ly7=xi(5);
    ly8=xi(4);

    % lytot = 16;


    lyt = [ly1,ly2,ly3,ly4,ly5,ly6,ly7,ly8]; %vector com os espaçamentos
em y


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%          Propriedades dos materiais       %%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    E=210e6;     % Módulo de Young (GPa)
    G=84e6;      % Módulo de distorção (GPa)
    I=5696e-8;   % Inércia da secção (m^4)
```

```
J=59.28e-8; % Factor de torção (m^4)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%           Carga aplicada           %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Parea = 10;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%      Numeração dos elementos e nós      %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ndivx = length(lxt);         %nº de divisões em x
ndivy = length(lyt);         %nº de divisões em y

nelemy = ndivy*(ndivx-1);    %nº de elementos na direcção y
nelemx = ndivx*(ndivy-1);    %nº de elementos na direcção x
nelem = nelemx+nelemy;       %nº de elementos finitos da estrutura

element = zeros(nelem,2)     %matriz com a numeração dos nós em cada
                             %elemento finito


for i = 1:ndivx-1

    for j = 1:ndivy

    element((i-1)*ndivy+j,1)=(ndivy+1)*(i-1)+j;
    element((i-1)*ndivy+j,2)=(ndivy+1)*(i-1)+j+1;

    end
end

for j = 1:ndivy-1

    element((nelemy+1)+(ndivx)*(j-1),1) = (ndivy+1)*(ndivx-1)+1 ...
+2*(j-1);
    element((nelemy+1)+(ndivx)*(j-1),2) = (1+j);

    element((nelemy+ndivx)+(ndivx)*(j-1),1) = (ndivy+1)* ...
(ndivx-2)+1+j;
    element((nelemy+ndivx)+(ndivx)*(j-1),2) = (ndivy+1)* ...
(ndivx-1)+2+2*(j-1);

    for i = 1:ndivx-2

        element((nelemy+1)+(ndivx)*(j-1)+i,1) = (1+j)+(ndivy+1)* ...
(i-1);
        element((nelemy+1)+(ndivx)*(j-1)+i,2) = (1+j)+(ndivy+1)*(i);

    end
```

```
end

element


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%            COORDENADAS DOS PONTOS            %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% coordenadas em x

lxaux = 0;
e = 1;
for i = 0:ndivx

     if i == 0

        for j = 1:ndivy-1

            xx((ndivy+1)*(ndivx-1)+1+2*(j-1))=0;

        end

     elseif i <= ndivx-1 && i>=1

        lxaux = lxt(e)+lxaux;
        for j = 1:ndivy+1

            xx((ndivy+1)*(i-1)+j)=lxaux;

        end
        e=e+1;

     else

        for j = 1:ndivy-1

            xx((ndivy+1)*(ndivx-1)+2+2*(j-1))= lxt(e)+lxaux;

        end
     end

end


% coordenadas em y

for i = 0:ndivx

     if i == 0
     e = 1;
     lyaux = 0;
```

```
        for j = 1:ndivy-1

            lyaux = lyt(e)+lyaux;
            yy((ndivy+1)*(ndivx-1)+1+2*(j-1)) = lyaux;
            e = e + 1;
        end

    elseif i <= ndivx-1 && i>=1

        e = 1;
        lyaux = 0;

        yy((ndivy+1)*(i-1)+1) = 0;

        for j = 1:ndivy
            lyaux = lyt(e)+lyaux;
            yy((ndivy+1)*(i-1)+j+1)=lyaux;
            e=e+1;
        end

    else
    e = 1;
    lyaux = 0;
        for j = 1:ndivy-1

            lyaux = lyt(e)+lyaux;
            yy((ndivy+1)*(ndivx-1)+2+2*(j-1)) = lyaux;
            e = e + 1;

        end
    end

end

xx;
yy;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%    Carga unif distribuída para cada elemento    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Pelememt = zeros(nelem,1);  %vector com a carga distribuida em cada
                            %elemento finito

Ainf = zeros(nelem,1);   % vector com a area de influencia de cada EF


eee = 1;
for i = 1:ndivx-1

    if i == 1
```

```
        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)+lxt(i+1)/2;
            eee = eee+1;

        end

    elseif i < ndivx-1 && i>1

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)/2+lxt(i+1)/2;
            eee = eee+1;

        end

    else

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)/2+lxt(i+1);
            eee = eee+1;
        end

    end

end

for i = 1:ndivy-1

    if i == 1

        for j = 1:ndivx

            Ainf(eee,1) = lyt(i)+lyt(i+1)/2;
            eee = eee+1;

        end

    elseif i < ndivy-1 && i>1

        for j = 1:ndivx

            Ainf(eee,1) = lyt(i)/2+lyt(i+1)/2;
            eee = eee+1;

        end

    else

        for j = 1:ndivx
```

```
            Ainf(eee,1) = lyt(i)/2+lyt(i+1);
            eee = eee+1;
        end

    end

end

Pelement = Ainf*Parea/2


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%    Cálculo da rigidez e vector de forças    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% iniciação das matrizes
numnode=length(xx);             % numero de nós
numelem=size(element,1);

U=zeros(3*numnode,1);
f=zeros(3*numnode,1);
K=zeros(3*numnode,3*numnode);


for e=1:numelem
    index=element(e,:); %nós do elemento
    %gdls respectivos do elemento
    indexB=[ (index(1)-1)*3+1 (index(1)-1)*3+2 (index(1)-1)*3+3 ...
        (index(2)-1)*3+1 (index(2)-1)*3+2 (index(2)-1)*3+3];
    xa=xx(index(2))-xx(index(1));
    ya=yy(index(2))-yy(index(1));
    L=sqrt(xa*xa+ya*ya);                        %comprimento do EF
    C=xa/L;
    S=ya/L;
    w1 = 12*E*I/(L*L*L);
    w2 = 6*E*I/(L*L);
    w3 = G*J/L;
    w4 = 4*E*I/L;
    w5 = 2*E*I/L;

    k = [w1 0 w2 -w1 0 w2 ; 0 w3 0 0 -w3 0 ;  %matriz de rigidez
        w2 0 w4 -w2 0 w5 ; -w1 0 -w2 w1 0 -w2 ; %local
        0 -w3 0 0 w3 0 ; w2 0 w5 -w2 0 w4];

    R = [1 0 0 0 0 0 ; 0 C S 0 0 0 ; 0 -S C 0 0 0 ;%matriz de rotação
        0 0 0 1 0 0 ; 0 0 0 0 C S ; 0 0 0 0 -S C];

    kr = R'*k*R;                % matriz de rigidez nos eixos globais

    P = Pelement(e,1);          % carga do EF

    fe=-(P*L/12)*[6 0 L 6 0 -L]';   % vector das cargas local
```

```
    fr = R'*fe;                 % vector das cargas nos eixos globais

    % Assemblagem na matriz de rigidez e no vector das cargas
    K(indexB,indexB)=K(indexB,indexB)+kr;
    f(indexB,1) = f(indexB,1)+fr;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%    Graus de liberdade restringidos   %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = 1;
for i = 1:ndivx-1
    lofs(e) = (ndivy+1)*(i-1)+1;
    e = e + 1;
end

for i = 1:ndivx-1
    lofs(e) = (ndivy+1)*(i-1)+ndivy+1;
    e = e + 1;
end

for i = 1:ndivy-1
    lofs(e) = (ndivy+1)*(ndivx-1)+1+2*(i-1);
    e = e +1;
end

for i = 1:ndivy-1
    lofs(e) = (ndivy+1)*(ndivx-1)+2+2*(i-1);
    e = e +1;
end

lofs;                   %vector com os nós restringidos

dofs = (lofs-1)*3+1;    %vector com os gdl restringidos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%          Solução do problema          %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

activos=setdiff([1:3*numnode]',[dofs]);

U=K(activos,activos)\f(activos);
U1=zeros(3*numnode,1);
U1(activos)=U;

disp('deslocamentos ')
U=U1;

disp('reaccoes ')
F=K*U-f;
```

```
    for e = 1:numelem
       index=element(e,:); %nós do elemento
       %gdls respectivos do elemento
       indexB = [ (index(1)-1)*3+1 (index(1)-1)*3+2 (index(1)-1)*3+3 ...
(index(2)-1)*3+1 (index(2)-1)*3+2 (index(2)-1)*3+3];%gdls respectivos do
                                                    %elemento

       xa = xx(index(2))-xx(index(1));
       ya = yy(index(2))-yy(index(1));
       L = sqrt(xa^2+ya^2);                 %comprimento do EF
       C = xa/L;
       S = ya/L;

       w1 = 12*E*I/(L^3);
       w2 = 6*E*I/(L^2);
       w3 = G*J/L;
       w4 = 4*E*I/L;
       w5 = 2*E*I/L;

       k = [w1 0 w2 -w1 0 w2 ; 0 w3 0 0 -w3 0 ;     % matriz de rigidez
           w2 0 w4 -w2 0 w5 ; -w1 0 -w2 w1 0 -w2 ;  % local
           0 -w3 0 0 w3 0 ; w2 0 w5 -w2 0 w4];

       R = [1 0 0 0 0 0 ; 0 C S 0 0 0 ; 0 -S C 0 0 0 ; %matriz de rotação
           0 0 0 1 0 0 ; 0 0 0 0 C S ; 0 0 0 0 -S C];

       %Cálculo do vector de carga para carga uniformemente distribuida

       P = Pelement(e,1);

       fe=-(P*L/12)*[6 0 L 6 0 -L]';

       % Força nos elementos finitos
       Fe = k*R*U(indexB);

       EF(:,e) = Fe-fe;

    end

    disp('forças nos elementos')
    EF;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%             Esforços internos              %%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    l=1;
    for i=1:numelem
        for j=1:2

            V(l)=EF(3*j-2,i); %colhe valor do esf transverso V
            if j==2
                V(l)=-V(l); %correcção do sinal para V positivo
                            %(atenção ao referencial do EF! O sinal + do
```

```
                        %esforço transverso deve ser para baixo no nó
                        %2 do EF)
        end

        M(l)=EF(3*j,i);  %colhe valor do Momento flector M
        if j==1
            M(l)=-M(l); %correcção do sinal para M positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %momento flector deve ser horário no nó 1 do
                        %EF)
        end

        T(l) = EF(3*j-1,i); %colhe valor do Momento torsor T
        if j==1
            T(l)=-T(l); %correcção do sinal para T positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %momento torsor deve ser no sentido do eixo
                        %do EF no nó 2)
        end
        l=l+1;
    end
end

% vectores dos esforços
V;
T;
M;

% verificação se existem maximos momentos interiores do EF
pp=1;
for i=1:numelem

   index=element(i,:); %nós do elemento

   xa = xx(index(2))-xx(index(1));
   ya = yy(index(2))-yy(index(1));
   L = sqrt(xa^2+ya^2);

   P = Pelement(i,1);

   %verificar se existe variação do sinal de V
   V0=V(2*i-1);
   M0=M(2*i-1);
   xint=abs(V0/P);

   if xint<L
       Mint=M0+V0*xint-P*(xint^2)/2;
   else
       Mint=0;
   end

   Mf(pp)=M(2*i-1);
   Mf(pp+1)=Mint;
   Mf(pp+2)=M(2*i);
```

```
   pp=pp+3;

end

Mf              %vector do M com os valores a meio vão dos elementos


% criação de uma matriz auxiliar com a distribuição dos momentos
% flectores em cada elemento finito

for j = 1:length(lxt)*3+length(lxt)-1

    for i = 1:length(lyt)*3+length(lyt)-1

        Mstress(i,j) = NaN;

    end

end



ee = 1;
    for j = 1:length(lxt)-1

        e = 0;


        for jj = 1:length(lyt)

            Mstress(3*jj-2+e,4*j) = Mf(ee);
            Mstress(3*jj-1+e,4*j) = Mf(ee+1);
            Mstress(3*jj+e,4*j) = Mf(ee+2);

            ee = ee +3;
            e = e + 1;
        end


    end

for j = 1:length(lyt)-1

        e = 0;


        for jj = 1:length(lxt)

            Mstress(4*j,3*jj-2+e) = Mf(ee);
            Mstress(4*j,3*jj-1+e) = Mf(ee+1);
            Mstress(4*j,3*jj+e) = Mf(ee+2);
```

```
                ee = ee +3;
                e = e + 1;
            end


    end

    Mstress %Matriz com a disposição dos valores de momentos flectores
nos
            %nós de extremidade do EF e a meio-vão


    Mmax = max(abs(Mf)) %valor do máximo valor em módulo do Mf

end
```

B.2 Programa: fobjectivo_tipoIb.m

```
function Mmax = fobjectivo_tipoIb(xi)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%                   Programa para a análise estrutural                          %
%                de grelhas do tipo I.b com apoios rígidos                      %
%                            baseado no MEF                                     %
%                                                                               %
%                           Guilherme Martins                                   %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%           Variaveis do problema            %%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Espaçamentos dos apoios segundo x
    lx1 = xi(1);
    lx2 = xi(2);
    lx3 = xi(3);
    lx4 = xi(2);
    lx5 = xi(1);

    % lxtot = 10;


    lxt = [lx1,lx2,lx3,lx4,lx5]; %vector com os espaçamentos em x


    % Espaçamentos dos apoios segundo y
    ly1=xi(4);
    ly2=xi(5);
    ly3=xi(6);
    ly4=xi(7);
    ly5=xi(7);
    ly6=xi(6);
    ly7=xi(5);
    ly8=xi(4);

    % lytot = 16;


    lyt = [ly1,ly2,ly3,ly4,ly5,ly6,ly7,ly8]; %vector com os espaçamentos
em y


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%          Propriedades dos materiais       %%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    E=210e6;     % Módulo de Young (GPa)
    G=84e6;      % Módulo de distorção (GPa)
    I=5696e-8;   % Inércia da secção (m^4)
```

```
J=59.28e-8; % Factor de torção (m^4)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%             Carga aplicada             %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Parea = 10;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%      Numeração dos elementos e nós       %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ndivx = length(lxt);         %nº de divisões em x
ndivy = length(lyt);         %nº de divisões em y

nelemy = ndivy*(ndivx-1);    %nº de elementos na direcção y
nelemx = ndivx*(ndivy-1);    %nº de elementos na direcção x
nelem = nelemx+nelemy;       %nº de elementos finitos da estrutura

element = zeros(nelem,2)     %matriz com a numeração dos nós em cada
                             %elemento finito


for i = 1:ndivx-1

    for j = 1:ndivy

    element((i-1)*ndivy+j,1)=(ndivy+1)*(i-1)+j;
    element((i-1)*ndivy+j,2)=(ndivy+1)*(i-1)+j+1;

    end
end

for j = 1:ndivy-1

    element((nelemy+1)+(ndivx)*(j-1),1)  =  (ndivy+1)*(ndivx-1)+1...
+2*(j-1);
    element((nelemy+1)+(ndivx)*(j-1),2)  =  (1+j);

    element((nelemy+ndivx)+(ndivx)*(j-1),1)  =  (ndivy+1)* ...
(ndivx-2)+1+j;
    element((nelemy+ndivx)+(ndivx)*(j-1),2)  =  (ndivy+1)* ...
(ndivx-1)+2+2*(j-1);

    for i = 1:ndivx-2

        element((nelemy+1)+(ndivx)*(j-1)+i,1)  =  (1+j)+(ndivy+1)* ...
(i-1);
        element((nelemy+1)+(ndivx)*(j-1)+i,2)  =  (1+j)+(ndivy+1)*(i);

    end
```

```
end

element


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%            COORDENADAS DOS PONTOS            %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% coordenadas em x

lxaux = 0;
e = 1;
for i = 0:ndivx

     if i == 0

        for j = 1:ndivy-1

            xx((ndivy+1)*(ndivx-1)+1+2*(j-1))=0;

        end

     elseif i <= ndivx-1 && i>=1

        lxaux = lxt(e)+lxaux;
        for j = 1:ndivy+1

            xx((ndivy+1)*(i-1)+j)=lxaux;

        end
        e=e+1;

     else

        for j = 1:ndivy-1

            xx((ndivy+1)*(ndivx-1)+2+2*(j-1))= lxt(e)+lxaux;

        end
     end

end


% coordenadas em y

for i = 0:ndivx

     if i == 0
     e = 1;
     lyaux = 0;
```

```
        for j = 1:ndivy-1

            lyaux = lyt(e)+lyaux;
            yy((ndivy+1)*(ndivx-1)+1+2*(j-1)) = lyaux;
            e = e + 1;
        end

    elseif i <= ndivx-1 && i>=1

        e = 1;
        lyaux = 0;

        yy((ndivy+1)*(i-1)+1) = 0;

        for j = 1:ndivy
            lyaux = lyt(e)+lyaux;
            yy((ndivy+1)*(i-1)+j+1)=lyaux;
            e=e+1;
        end

    else
    e = 1;
    lyaux = 0;
        for j = 1:ndivy-1

            lyaux = lyt(e)+lyaux;
            yy((ndivy+1)*(ndivx-1)+2+2*(j-1)) = lyaux;
            e = e + 1;

        end
    end

end

xx;
yy;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%    Carga unif distribuída para cada elemento    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Pelememt = zeros(nelem,1);  %vector com a carga distribuida em cada
                            %elemento finito

Ainf = zeros(nelem,1);   % vector com a area de influencia de cada EF


eee = 1;
for i = 1:ndivx-1

    if i == 1
```

```
        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)+lxt(i+1)/2;
            eee = eee+1;

        end

    elseif i < ndivx-1 && i>1

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)/2+lxt(i+1)/2;
            eee = eee+1;

        end

    else

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)/2+lxt(i+1);
            eee = eee+1;
        end

    end

end

for i = 1:ndivy-1

    if i == 1

        for j = 1:ndivx

            Ainf(eee,1) = lyt(i)+lyt(i+1)/2;
            eee = eee+1;

        end

    elseif i < ndivy-1 && i>1

        for j = 1:ndivx

            Ainf(eee,1) = lyt(i)/2+lyt(i+1)/2;
            eee = eee+1;

        end

    else

        for j = 1:ndivx
```

```
            Ainf(eee,1) = lyt(i)/2+lyt(i+1);
            eee = eee+1;
        end

    end

end

Pelement = Ainf*Parea/2


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%    Cálculo da rigidez e vector de forças   %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% iniciação das matrizes
numnode=length(xx);             % numero de nós
numelem=size(element,1);

U=zeros(3*numnode,1);
f=zeros(3*numnode,1);
K=zeros(3*numnode,3*numnode);


for e=1:numelem
    index=element(e,:); %nós do elemento
    %gdls respectivos do elemento
    indexB=[ (index(1)-1)*3+1 (index(1)-1)*3+2 (index(1)-1)*3+3 ...
        (index(2)-1)*3+1 (index(2)-1)*3+2 (index(2)-1)*3+3];
    xa=xx(index(2))-xx(index(1));
    ya=yy(index(2))-yy(index(1));
    L=sqrt(xa*xa+ya*ya);                        %comprimento do EF
    C=xa/L;
    S=ya/L;
    w1 = 12*E*I/(L*L*L);
    w2 = 6*E*I/(L*L);
    w3 = G*J/L;
    w4 = 4*E*I/L;
    w5 = 2*E*I/L;

    k = [w1 0 w2 -w1 0 w2 ; 0 w3 0 0 -w3 0 ;      %matriz de rigidez
        w2 0 w4 -w2 0 w5 ; -w1 0 -w2 w1 0 -w2 ;   %local
        0 -w3 0 0 w3 0 ; w2 0 w5 -w2 0 w4];

    R = [1 0 0 0 0 0 ; 0 C S 0 0 0 ; 0 -S C 0 0 0 ;%matriz de rotação
        0 0 0 1 0 0 ; 0 0 0 0 C S ; 0 0 0 0 -S C];

    kr = R'*k*R;                % matriz de rigidez nos eixos globais

    P = Pelement(e,1);          % carga do EF

    fe=-(P*L/12)*[6 0 L 6 0 -L]';   % vector das cargas local
```

```
    fr = R'*fe;                 % vector das cargas nos eixos globais

    % Assemblagem na matriz de rigidez e no vector das cargas
    K(indexB,indexB)=K(indexB,indexB)+kr;
    f(indexB,1) = f(indexB,1)+fr;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%    Graus de liberdade restringidos   %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = 1;
for i = 1:ndivx-1
    lofs(e) = (ndivy+1)*(i-1)+1;
    e = e + 1;
end

for i = 1:ndivx-1
    lofs(e) = (ndivy+1)*(i-1)+ndivy+1;
    e = e + 1;
end

for i = 1:ndivy-1
    lofs(e) = (ndivy+1)*(ndivx-1)+1+2*(i-1);
    e = e +1;
end

for i = 1:ndivy-1
    lofs(e) = (ndivy+1)*(ndivx-1)+2+2*(i-1);
    e = e +1;
end

el = 1;
for i = 1:length(lofs)

    dofs(el) = (lofs(i)-1)*3+1;
    dofs(el+1) = (lofs(i)-1)*3+2;
    dofs(el+2) = (lofs(i)-1)*3+3;

    el = el + 3;

end


lofs      %vector com os nós restringidos

dofs      %vector com os gdl restringidos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%          Solução do problema          %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    activos=setdiff([1:3*numnode]',[dofs]);

    U=K(activos,activos)\f(activos);
    U1=zeros(3*numnode,1);
    U1(activos)=U;

    disp('deslocamentos ')
    U=U1;

    disp('reaccoes ')
    F=K*U-f;

    for e = 1:numelem
       index=element(e,:); %nós do elemento
       %gdls respectivos do elemento
       indexB = [ (index(1)-1)*3+1 (index(1)-1)*3+2 (index(1)-1)*3+3 ...
(index(2)-1)*3+1 (index(2)-1)*3+2 (index(2)-1)*3+3]; %gdls respectivos do
                                                      %elemento

       xa = xx(index(2))-xx(index(1));
       ya = yy(index(2))-yy(index(1));
       L = sqrt(xa^2+ya^2);                  %comprimento do EF
       C = xa/L;
       S = ya/L;

       w1 = 12*E*I/(L^3);
       w2 = 6*E*I/(L^2);
       w3 = G*J/L;
       w4 = 4*E*I/L;
       w5 = 2*E*I/L;

       k = [w1 0 w2 -w1 0 w2 ; 0 w3 0 0 -w3 0 ;     % matriz de rigidez
           w2 0 w4 -w2 0 w5 ; -w1 0 -w2 w1 0 -w2 ;  % local
           0 -w3 0 0 w3 0 ; w2 0 w5 -w2 0 w4];

       R = [1 0 0 0 0 0 ; 0 C S 0 0 0 ; 0 -S C 0 0 0 ; %matriz de rotação
           0 0 0 1 0 0 ; 0 0 0 0 C S ; 0 0 0 0 -S C];

       %Cálculo do vector de carga para carga uniformemente distribuida

       P = Pelement(e,1);

       fe=-(P*L/12)*[6 0 L 6 0 -L]';

       % Força nos elementos finitos
       Fe = k*R*U(indexB);

       EF(:,e) = Fe-fe;

    end

    disp('forças nos elementos')
    EF;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%              Esforços internos                %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

l=1;
for i=1:numelem
    for j=1:2

        V(l)=EF(3*j-2,i); %colhe valor do esf transverso V
        if j==2
            V(l)=-V(l); %correcção do sinal para V positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %esforço transverso deve ser para baixo no nó
                        %2 do EF)
        end

        M(l)=EF(3*j,i);  %colhe valor do Momento flector M
        if j==1
            M(l)=-M(l); %correcção do sinal para M positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %momento flector deve ser horário no nó 1 do
                        %EF)
        end

        T(l) = EF(3*j-1,i); %colhe valor do Momento torsor T
        if j==1
            T(l)=-T(l); %correcção do sinal para T positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %momento torsor deve ser no sentido do eixo
                        %do EF no nó 2)
        end
        l=l+1;
    end
end

% vectores dos esforços
V;
T;
M;

% verificação se existem maximos momentos interiores do EF
pp=1;
for i=1:numelem

   index=element(i,:); %nós do elemento

   xa = xx(index(2))-xx(index(1));
   ya = yy(index(2))-yy(index(1));
   L = sqrt(xa^2+ya^2);

   P = Pelement(i,1);

   %verificar se existe variação do sinal de V
   V0=V(2*i-1);
```

```
    M0=M(2*i-1);
    xint=abs(V0/P);

    if xint<L
        Mint=M0+V0*xint-P*(xint^2)/2;
    else
        Mint=0;
    end

    Mf(pp)=M(2*i-1);
    Mf(pp+1)=Mint;
    Mf(pp+2)=M(2*i);

    pp=pp+3;

end

Mf              %vector do M com os valores a meio vão dos elementos


% criação de uma matriz auxiliar com a distribuição dos momentos
% flectores em cada elemento finito

for j = 1:length(lxt)*3+length(lxt)-1

    for i = 1:length(lyt)*3+length(lyt)-1

        Mstress(i,j) = NaN;

    end

end



ee = 1;
    for j = 1:length(lxt)-1

        e = 0;


        for jj = 1:length(lyt)

            Mstress(3*jj-2+e,4*j) = Mf(ee);
            Mstress(3*jj-1+e,4*j) = Mf(ee+1);
            Mstress(3*jj+e,4*j) = Mf(ee+2);

            ee = ee +3;
            e = e + 1;
        end


    end
```

```
    for j = 1:length(lyt)-1

            e = 0;


            for jj = 1:length(lxt)

                Mstress(4*j,3*jj-2+e) = Mf(ee);
                Mstress(4*j,3*jj-1+e) = Mf(ee+1);
                Mstress(4*j,3*jj+e) = Mf(ee+2);

                ee = ee +3;
                e = e + 1;
            end


    end

    Mstress %Matriz com a disposição dos valores de momentos flectores
            %nos nós de extremidade do EF e a meio-vão


    Mmax = max(abs(Mf)) %valor do máximo valor em módulo do Mf

end
```

B.3 Programa: fobjectivo_tipoII.m

```
function Mmax = fobjectivo_tipoII(xi)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%                   Programa para a análise estrutural                          %
%                de grelhas do tipo II com apoios rígidos                       %
%                            baseado no MEF                                     %
%                                                                               %
%                           Guilherme Martins                                   %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%           Variaveis do problema            %%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Comprimentos dos tramos segundo x
    lx1 = xi(1);
    lx2 = xi(2);
    lx3 = xi(3);
    lx4 = xi(2);
    lx5 = xi(1);

    % lxtot = 10;


    lxt = [lx1,lx2,lx3,lx4,lx5]; %vector com os comprimentos em x


    % Comprimentos dos tramos segundo y
    ly1=xi(4);
    ly2=xi(5);
    ly3=xi(6);
    ly4=xi(7);
    ly5=xi(7);
    ly6=xi(6);
    ly7=xi(5);
    ly8=xi(4);

    % lytot = 16;


    lyt = [ly1,ly2,ly3,ly4,ly5,ly6,ly7,ly8]; %vector com os comprimentos
                                              %em x



    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%          Propriedades dos materiais       %%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    E=210e6;     % Módulo de Young (GPa)
    G=84e6;      % Módulo de distorção (GPa)
```

```
I=5696e-8;  % Inércia da secção (m^4)
J=59.28e-8; % Factor de torção (m^4)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%              Carga aplicada              %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Parea = 10; % (kN/m^2)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%       Numeração dos elementos e nós       %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ndivx = length(lxt);         %nº de divisões em x
ndivy = length(lyt);         %nº de divisões em y

nelemy = ndivy*(ndivx+1);    %nº de elementos na direcção y
nelemx = ndivx*(ndivy+1);    %nº de elementos na direcção x
nelem = nelemx+nelemy;       %nº de elementos finitos da estrutura

element = zeros(nelem,2);    %matriz com a numeração dos nós em cada
                             %elemento finito

for i = 1:ndivx+1

    for j = 1:ndivy

    element((i-1)*ndivy+j,1)=(ndivy+1)*(i-1)+j;
    element((i-1)*ndivy+j,2)=(ndivy+1)*(i-1)+j+1;

    end
end

for j = 1:ndivy+1

    element((nelemy+1)+(ndivx)*(j-1),1) = j;
    element((nelemy+1)+(ndivx)*(j-1),2) = (ndivy+1)+j;

    element((nelemy+ndivx)+(ndivx)*(j-1),1) = (ndivy+1)*(ndivx-1)+j;
    element((nelemy+ndivx)+(ndivx)*(j-1),2) = (ndivy+1)*(ndivx)+j;

    for i = 1:ndivx-2

        element((nelemy+1)+(ndivx)*(j-1)+i,1) = (j)+(ndivy+1)*(i);
        element((nelemy+1)+(ndivx)*(j-1)+i,2) = (j)+(ndivy+1)*(i+1);

    end

end

element
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%            COORDENADAS DOS PONTOS                %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% coordenadas em x

lxaux = 0;
e = 1;
ex = 1;
for i = 1:ndivx+1

     if i == 1

        for j = 1:ndivy+1

            xx(ex)=0;
            ex = ex + 1;

        end

     elseif i <= ndivx && i>1

        lxaux = lxt(e)+lxaux;
        for j = 1:ndivy+1

            xx(ex)=lxaux;
            ex = ex + 1;

        end
        e=e+1;

     else

        for j = 1:ndivy+1

            xx(ex)= lxt(e)+lxaux;
            ex = ex +1;

        end
     end

end


% coordenadas em y

ey = 1;
for i = 1:ndivx+1

     if i == 1
     e = 1;
```

```
        lyaux = 0;

           for j = 1:ndivy

               yy(ey) = lyaux;
               lyaux = lyt(e)+lyaux;
               e = e + 1;
               ey = ey + 1;
           end

           yy(ey) = lyaux;

        elseif i <= ndivx && i>1

           e = 1;
           lyaux = 0;
           ey = ey + 1;

           for j = 1:ndivy

               yy(ey) = lyaux;
               lyaux = lyt(e)+lyaux;
               e = e + 1;
               ey = ey + 1;

           end

           yy(ey) = lyaux;

        else

        ey = ey +1;
        e = 1;
        lyaux = 0;

           for j = 1:ndivy

               yy(ey) = lyaux;
               lyaux = lyt(e)+lyaux;
               e = e + 1;
               ey = ey + 1;

           end

        yy(ey) = lyaux;

        end

    end

    xx                  % vector com as coordenadas nodais em x
    yy                  % vector com as coordenadas nodais em y
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%    Carga unif distribuída para cada elemento    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Pelememt = zeros(nelem,1);  %vector com a carga distribuida em cada
                            %elemento finito

Ainf = zeros(nelem,1);   % vector com a area de influencia de cada EF


eee = 1;
for i = 1:ndivx+1

    if i == 1

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)/2;
            eee = eee+1;

        end

    elseif i <= ndivx && i>1

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i)/2+lxt(i-1)/2;
            eee = eee+1;

        end

    else

        for j = 1:ndivy

            Ainf(eee,1) = lxt(i-1)/2;
            eee = eee+1;
        end

    end

end

for i = 1:ndivy+1

    if i == 1

        for j = 1:ndivx
```

```
            Ainf(eee,1) = lyt(i)/2;
            eee = eee+1;

        end

    elseif i <= ndivy && i>1

        for j = 1:ndivx

            Ainf(eee,1) = lyt(i)/2+lyt(i-1)/2;
            eee = eee+1;

        end

    else

        for j = 1:ndivx

            Ainf(eee,1) = lyt(i-1)/2;
            eee = eee+1;
        end

    end

end

Pelement = Ainf*Parea/2


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%    Cálculo da rigidez e vector de forças    %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% iniciação das matrizes

numnode=length(xx);             % numero de nós
numelem=size(element,1);

U=zeros(3*numnode,1);
f=zeros(3*numnode,1);
K=zeros(3*numnode,3*numnode);


for e=1:numelem
    index=element(e,:); %nós do elemento
    %gdls respectivos do elemento
    indexB=[ (index(1)-1)*3+1 (index(1)-1)*3+2 (index(1)-1)*3+3 ...
        (index(2)-1)*3+1 (index(2)-1)*3+2 (index(2)-1)*3+3];
    xa=xx(index(2))-xx(index(1));
    ya=yy(index(2))-yy(index(1));
    L=sqrt(xa*xa+ya*ya);                            %comprimento do EF
    C=xa/L;
    S=ya/L;
```

```
    w1 = 12*E*I/(L*L*L);
    w2 = 6*E*I/(L*L);
    w3 = G*J/L;
    w4 = 4*E*I/L;
    w5 = 2*E*I/L;

    k = [w1 0 w2 -w1 0 w2 ; 0 w3 0 0 -w3 0 ;       %matriz de rigidez
        w2 0 w4 -w2 0 w5 ; -w1 0 -w2 w1 0 -w2 ;     %local
        0 -w3 0 0 w3 0 ; w2 0 w5 -w2 0 w4];

    R = [1 0 0 0 0 0 ; 0 C S 0 0 0 ; 0 -S C 0 0 0 ;%matriz de rotação
        0 0 0 1 0 0 ; 0 0 0 0 C S ; 0 0 0 0 -S C];

    kr = R'*k*R;              % matriz de rigidez nos eixos globais

    P = Pelement(e,1);        % carga do EF

    fe=-(P*L/12)*[6 0 L 6 0 -L]';  % vector das cargas local

    fr = R'*fe;               % vector das cargas nos eixos globais

    % Assemblagem na matriz de rigidez e no vector das cargas
    K(indexB,indexB)=K(indexB,indexB)+kr;
    f(indexB,1) = f(indexB,1)+fr;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%    Graus de liberdade restringidos   %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = 1;
for i = 1:ndivy+1
    lofs(e) = i;
    e = e + 1;
end

for i = 1:ndivy+1
    lofs(e) = ndivx*(ndivy+1)+i;
    e = e + 1;
end

for i = 1:ndivx-1
    lofs(e) = (ndivy+1)*(i)+1;
    e = e +1;
end

for i = 1:ndivx-1
    lofs(e) = (ndivy+1)*(i)+1+ndivy;
    e = e +1;
end

lofs                    %vector com os nós restringidos
```

```
dofs = (lofs-1)*3+1     %vector com os gdl restringidos



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%          Solução do problema          %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

activos=setdiff([1:3*numnode]',[dofs]);

U=K(activos,activos)\f(activos);
U1=zeros(3*numnode,1);
U1(activos)=U;

disp('deslocamentos ')
U=U1

disp('reaccoes ')
F=K*U-f

for e = 1:numelem
   index=element(e,:); %nós do elemento
   %gdls respectivos do elemento
   indexB = [ (index(1)-1)*3+1 (index(1)-1)*3+2 (index(1)-1)*3+3 ...
       (index(2)-1)*3+1 (index(2)-1)*3+2 (index(2)-1)*3+3];

   xa = xx(index(2))-xx(index(1));
   ya = yy(index(2))-yy(index(1));
   L = sqrt(xa^2+ya^2);                          %comprimento do EF
   C = xa/L;
   S = ya/L;

   w1 = 12*E*I/(L^3);
   w2 = 6*E*I/(L^2);
   w3 = G*J/L;
   w4 = 4*E*I/L;
   w5 = 2*E*I/L;

   k = [w1 0 w2 -w1 0 w2 ; 0 w3 0 0 -w3 0 ;     % matriz de rigidez
       w2 0 w4 -w2 0 w5 ; -w1 0 -w2 w1 0 -w2 ;  % local
       0 -w3 0 0 w3 0 ; w2 0 w5 -w2 0 w4];

   R = [1 0 0 0 0 0 ; 0 C S 0 0 0 ; 0 -S C 0 0 0 ; %matriz de rotação
       0 0 0 1 0 0 ; 0 0 0 0 C S ; 0 0 0 0 -S C];

   %Cálculo do vector de carga para carga uniformemente distribuida

   P = Pelement(e,1);

   fe=-(P*L/12)*[6 0 L 6 0 -L]';

   % Força nos elementos finitos
   Fe = k*R*U(indexB);
```

```
   EF(:,e) = Fe-fe;

end

disp('forças nos elementos')
EF;




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%              Esforços internos              %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

l=1;
for i=1:numelem
    for j=1:2

        V(l)=EF(3*j-2,i); %colhe valor do esf transverso V
        if j==2
            V(l)=-V(l); %correcção do sinal para V positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %esforço transverso deve ser para baixo no nó
                        %2 do EF)
        end

        M(l)=EF(3*j,i);  %colhe valor do Momento flector M
        if j==1
            M(l)=-M(l); %correcção do sinal para M positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %momento flector deve ser horário no nó 1 do
                        %EF)
        end

        T(l) = EF(3*j-1,i); %colhe valor do Momento torsor T
        if j==1
            T(l)=-T(l); %correcção do sinal para T positivo
                        %(atenção ao referencial do EF! O sinal + do
                        %momento torsor deve ser no sentido do eixo
                        %do EF no nó 2)

        End

        l=l+1;

    end
end

% vectores dos esforços
V
T
M
```

```
% verificação se existem maximos momentos interiores do EF
pp=1;
for i=1:numelem

   index=element(i,:); %nós do elemento

   xa = xx(index(2))-xx(index(1));
   ya = yy(index(2))-yy(index(1));
   L = sqrt(xa^2+ya^2);

   P = Pelement(i,1);

   %verificar se existe variação do sinal de V
   V0=V(2*i-1);
   M0=M(2*i-1);
   xint=abs(V0/P);

   if xint<L
       Mint=M0+V0*xint-P*(xint^2)/2;
   else
       Mint=0;
   end

   Mf(pp)=M(2*i-1);
   Mf(pp+1)=Mint;
   Mf(pp+2)=M(2*i);

   pp=pp+3;

end

Mf          %vector do M com os valores a meio vão dos elementos


% criação de uma matriz auxiliar com a distribuição dos momentos
% flectores em cada elemento finito

for j = 1:length(lxt)*3+length(lxt)+1

    for i = 1:length(lyt)*3+length(lyt)+1

        Mstress(i,j) = NaN;

    end

end

ee = 1;
    for j = 1:length(lxt)+1

        e = 0;
```

```
            for jj = 1:length(lyt)

                Mstress(3*jj-1+e,4*(j-1)+1) = Mf(ee);
                Mstress(3*jj+e,4*(j-1)+1) = Mf(ee+1);
                Mstress(3*jj+1+e,4*(j-1)+1) = Mf(ee+2);

                ee = ee + 3;
                e = e + 1;
            end


        end

    for j = 1:length(lyt)+1

            e = 0;


            for jj = 1:length(lxt)

                Mstress(4*(j-1)+1,3*jj-1+e) = Mf(ee);
                Mstress(4*(j-1)+1,3*jj+e) = Mf(ee+1);
                Mstress(4*(j-1)+1,3*jj+1+e) = Mf(ee+2);

                ee = ee +3;
                e = e + 1;
            end


    end

    Mstress %Matriz com a disposição dos valores de momentos flectores
            %nos nós de extremidade do EF e a meio-vão


    Mmax = max(abs(Mf)) %valor do máximo valor em módulo do Mf

end
```