



DÉBORA KASSIANA DE ALMEIDA GOMES
Licenciada em Ciências da Engenharia Mecânica

DESENVOLVIMENTO DE UM PROGRAMA DE ELEMENTOS FINITOS PARA OTIMIZAÇÃO DE FORMA DE ESTRUTURAS PLANAS

MESTRADO EM ENGENHARIA MECÂNICA
Universidade NOVA de Lisboa
Novembro, 2021



DESENVOLVIMENTO DE UM PROGRAMA DE ELEMENTOS FINITOS PARA OTIMIZAÇÃO DE FORMA DE ESTRUTURAS PLANAS

DÉBORA KASSIANA DE ALMEIDA GOMES

Licenciada em Ciências da Engenharia Mecânica

Orientador: Prof. Doutor João Mário Burguete Botelho Cardoso,
Professor Auxiliar, Universidade NOVA de Lisboa

Coorientadores: Prof. Doutor Pedro Samuel Gonçalves Coelho,
Professor Auxiliar, Universidade NOVA de Lisboa

Júri:

Presidente: Prof. Doutor Tiago Alexandre Narciso da
Silva,
Professor Auxiliar, Universidade NOVA de Lisboa

Vogais: Prof. Doutor José Manuel Cardoso Xavier,
Professor Auxiliar, Universidade NOVA de Lisboa
Prof. Doutor João Mário Burguete Botelho
Cardoso,
Professor Auxiliar, Universidade NOVA de Lisboa

Orientador: Prof. Doutor João Mário Burguete Botelho
Cardoso,
Professor Auxiliar, Universidade NOVA de Lisboa

Desenvolvimento de um programa de elementos finitos para otimização de forma de estruturas planas

Copyright © Débora Kassiana de Almeida Gomes, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Para a minha querida mãe.

Agradecimentos

Primeiramente gostaria de agradecer ao Prof. Dr. João Cardoso, professor da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa (FCT-UNL) e orientador desta dissertação, por me ter dado um voto de confiança e ter sempre acreditado em mim. Estou-lhe muito grata por toda a disponibilidade e dedicação que tornou possível a realização deste trabalho bem como todo o apoio dado ao longo destes anos do meu percurso académico.

Ao Prof. Dr. Pedro Coelho, professor da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa (FCT-UNL), agradeço todo o interesse demonstrado pela minha dissertação, o tempo que me concedeu da sua atenção e as ajudas pontuais quando estas surgiram.

Gostaria de agradecer também a todos os professores do Departamento de Engenharia Mecânica e Industrial da FCT-UNL, pela excelente formação que me foi dada.

Agradeço ao meu pai, por todo o investimento que fez em mim, pelo grande voto de confiança e todo o apoio dado durante toda a minha vida. Às minha amigas, por sempre terem sido um grande sistema de suporte para mim e por acreditarem em mim, mesmo quando eu própria não o fazia.

Por último, gostaria de agradecer a mim mesma por ter conseguido completar esta caminhada longa apesar de todas as dificuldades e obstáculos que surgiram.

Resumo

Esta dissertação procura desenvolver ferramentas de otimização para obter materiais celulares de microestrutura periódica. Para isso, considera-se o problema da placa com furo, de maneira a simular as condições pretendidas, onde a fronteira do furo corresponde à fronteira da inclusão que constitui a heterogeneidade em materiais celulares.

Na realização desta dissertação foram estudadas várias microestruturas ótimas obtidas através da resolução de um problema de otimização de forma. Este problema é formulado como a minimização da área da placa tendo em conta um constrangimento de tensão. Neste trabalho são consideradas duas equações distintas para descrever a forma das inclusões. A primeira parametrização utilizada considera uma descrição geométrica da fronteira através de uma *spline* de 5 nós. Posteriormente é utilizada uma parametrização mais flexível onde as curvas das inclusões são descritas pela equação da superelipse. São utilizadas as condições de fronteira de periodicidade para simular a periodicidade do material que está a ser considerado.

Demonstra-se o sucesso da implementação numérica da utilização da equação da superelipse para descrever a forma das inclusões, permitindo obter microestruturas ótimas que aproximam as condições de otimalidade com uma qualidade de resultados superior à da parametrização com a *spline*. Além disso confirma-se o sucesso da implementação numérica das condições de fronteira de periodicidade.

Palavras-chave: Otimização de forma, material celular, microestruturas periódicas, superelipse, condições de fronteira de periodicidade

Abstrat

This dissertation seeks to develop optimization tools to obtain cellular materials of periodic microstructure. For this, the problem of the plate with a hole is considered, in order to simulate the intended conditions, where the hole boundary corresponds to the inclusion boundary that constitutes the heterogeneity in cellular materials.

In carrying out this dissertation, several optimal microstructures obtained by solving a shape optimization problem were studied. This problem is formulated as the minimization of the plate area taking into account a stress constraint. In this work, two distinct equations are considered to describe the form of the inclusions. The first parameterization used, considers a geometric description of the boundary through a 5-node spline. Subsequently, a more flexible parameterization is used where the inclusion curves are described by the superellipse equation. Periodic boundary conditions are used to simulate the periodicity of the material under consideration.

The success of the numerical implementation of the use of the superellipse equation to describe the shape of the inclusions is demonstrated, allowing for the optimal microstructures to approximate the optimality conditions with a superior quality of results than the parameterization with the spline. Furthermore, the success of the numerical implementation of the periodic boundary conditions is confirmed.

Keywords: Shape optimization, cellular material, periodic microstructures, superellipse, periodic boundary conditions

Índice

Agradecimentos.....	IV
Resumo.....	V
Abstrat.....	VII
Índice de figuras.....	XI
Índice de tabelas.....	XIII
Lista de símbolos.....	XIV
Lista de acrónimos.....	XV
1. Introdução.....	1
1.1. Motivação.....	1
1.2. Objetivos da dissertação.....	1
1.3. Estrutura da dissertação.....	2
2. Conceitos teóricos.....	3
2.1. Método dos elementos finitos.....	3
2.2. Elementos bidimensionais.....	3
2.3. Elementos isoparamétricos.....	5
2.4. Cálculo de sensibilidades.....	9
2.5. Método contínuo de análise de sensibilidades.....	10
2.6. Placas sujeitas a esforços de membrana.....	17
2.7. Otimização estrutural.....	21
2.7.1. Otimização dimensional.....	22
2.7.2. Otimização de forma.....	22
2.7.3. Parametrização da fronteira em otimização de forma.....	23
3. Implementação computacional.....	24
3.1. Definição do problema.....	24
3.2. Visão geral do código.....	26
3.3. Geração de malha.....	26
3.4. Algoritmo de otimização.....	27
3.4.1. MMA.....	28

3.4.2.	Avaliação do erro	28
3.5.	Validação e análise de resultados	29
3.6.	Estudo de convergência de malha.....	35
4.	Parametrização da fronteira.....	37
4.1.	Superelipse.....	37
4.2.	Resultados e análise	39
4.2.1.	<i>Spline</i> vs Superelipse.....	39
4.2.2.	Validação dos resultados obtidos na otimização com a superelipse	40
5.	Aplicação das condições de fronteira de periodicidade	44
5.1.	Condições de fronteira de periodicidade	45
5.2.	Solução numérica da equação	46
5.3.	Implementação numérica	47
5.4.	Validação e resultados finais	49
6.	Conclusões	52
	Bibliografia	53
	Anexo	54

Índice de figuras

Figura 1 - Funções de forma utilizadas no elemento, extraído de [3].....	4
Figura 2 - Elemento isoparamétrico e referencial natural e global do mesmo, extraído de [3].....	5
Figura 3 - Função de forma N_1s, t	
Figura 4 - Valor das duas funções de forma são idênticos para pontos correspondentes, extraído de [3].....	7
Figura 5 - Viga em consola	9
Figura 6 - Viga à flexão	10
Figura 7 - Variação de forma do domínio Ω	11
Figura 8 - Placa no estado de tensão plano.	17
Figura 9 - Categorias de otimização estrutural associadas às várias etapas do projeto, extraído de [5]	21
Figura 10 - Exemplo de otimização dimensional, extraído de [1]	22
Figura 11 - Exemplo de otimização de forma, extraído de [1]	23
Figura 12 - Fluxograma do algoritmo de otimização desenvolvido	24
Figura 13 - Geometria do 1/4 de placa a ser analisado	25
Figura 14 - Exemplo de malha gerada pelo programa em OCTAVE, com $N_r=30$ e $N_{teta}=20$	27
Figura 15 - Formulação do problema no ANSYS para malha (30,20).	29
Figura 16 - Resultados das tensões obtidos através da análise da malha (30,20) .	30
Figura 17 - Resultados das tensões relativamente à dimensão da malha para o 1º elemento.	31
Figura 18 - Resultados das tensões relativamente à dimensão da malha para o último elemento.	32
Figura 19 - Diferenças finitas progressivas.....	33
Figura 20 - Teste de convergência de malha para o 1º elemento da fronteira da inclusão.....	35
Figura 21 - Teste de convergência de malha para o último elemento da fronteira da inclusão.....	35

Figura 22 - Parametrização da forma de uma inclusão, utilizando três variáveis de projeto (a, b, η), extraído de [1]	38
Figura 23 - Resultados da otimização com a parametrização da Superelipse VS parametrização com <i>Spline</i> de 5 nós, para carregamento biaxial ($\sigma_x = 2\sigma_y$).....	40
Figura 24 - Representação 2-D de uma estrutura periódica sem eixo ortogonal de simetria: (a) forma não deformada com duas alternativas de RCUs; (b) Deformada sob carga global. Extraído de [10].	45
Figura 25 – Exemplo de célula discretizada 3x3, extraído de [11].....	47
Figura 26 - Exemplo de deformadas da placa com furo para os diferentes casos de carga; a) Extensão em x; b)Distorção; c)Extensão em y	

Índice de tabelas

Tabela 1: Dimensões das malhas consideradas para a análise.	30
Tabela 2: Resultados da análise de tensões para o 1º elemento.	31
Tabela 3: Resultados da análise de tensões para o último elemento.	31
Tabela 4: Demonstração da validade do cálculo das sensibilidades das tensões de von Mises; A coluna "Diferenças finitas" indica os valores calculados pela equação (59); A coluna "df/dx" indica os valores calculados pelo método contínuo, através do programa desenvolvido.	34
Tabela 5: Solução do problema de minimização da área da placa com ($\sigma_x = \sigma_y$), utilizando a equação da superelipse para definição da fronteira.	41
Tabela 6: Distribuição do campo de tensões das soluções do problema de minimização da área da placa com ($\sigma_x = \sigma_y$), utilizando a equação da superelipse para definição da fronteira.	41
Tabela 7: Solução do problema de minimização da área da placa com ($\sigma_x = 2\sigma_y$), utilizando a equação da superelipse para definição da fronteira.	42
Tabela 8: Distribuição do campo de tensões das soluções do problema de minimização da área da placa com ($\sigma_x = 2\sigma_y$), utilizando a equação da superelipse para definição da fronteira.	42
Tabela 9: Análise do cálculo de sensibilidades das tensões de von Mises, após implementação das condições de fronteira de periodicidade; A coluna "Diferenças finitas" indica os valores calculados pela equação (59); A coluna "df/dx" indica os valores calculados pelo método contínuo, através do programa desenvolvido.	49
Tabela 10: Resultados da otimização de forma considerando as condições de fronteira de periodicidade para o caso de carga ($\epsilon_x = \epsilon_y$)	50
Tabela 11: Resultados da otimização de forma considerando as condições de fronteira de periodicidade para o caso de carga ($\epsilon_x = 2\epsilon_y$)	51

Lista de símbolos

σ_{ij}	Tensor das tensões
ε_{ij}	Tensor das deformações infinitesimais
$\bar{\varepsilon}_{ij}$	Tensor das deformações infinitesimais virtuais
K_{ij}	Matriz rigidez
u_j	Deslocamentos na direção i
\bar{u}_i	Deslocamentos virtuais na direção i
ρ	Densidade mássica
b_i	Forças atuando por unidade de volume, na direção i
N_j	Função de forma
E_{ijnm}	Matriz das constantes elásticas
J	Matriz Jacobiana
w_{IS}	Peso dos dois pontos de Gauss segundo s
w_{IT}	Peso dos dois pontos de Gauss segundo t
ν	Coefficiente de Poisson
e	Espessura
E	Módulo de elasticidade

Lista de acrónimos

<i>NLPQL</i>	Non-Linear Programming by Quadratic Lagrangian
<i>SQP</i>	Sequential Quadratic Programming
<i>MMA</i>	Method of Moving Asymptotes

1. Introdução

1.1. Motivação

A otimização de estruturas continua a ser uma ferramenta de enorme importância em engenharia mecânica, desempenhando um papel importante no projeto de estruturas. Por apresentar uma enorme variedade de aplicações, é uma área que se encontra em constante desenvolvimento. Esta desempenha um papel importante no projeto de estruturas, constituindo um auxiliar precioso no projeto de estruturas complexas, com grande número de parâmetros e constrangimentos, e contribuindo para a sistematização da atividade de projeto.

No projeto em engenharia, recorre-se a modelos analíticos, numéricos ou experimentais, de maneira a se verificar a aptidão das estruturas para suportar determinados carregamentos, tendo em conta os requisitos do projeto. Os modelos analíticos apresentam soluções exatas para os problemas, uma vez que se baseiam na descrição analítica de fenómenos físicos. Estes são apenas utilizados para resolver problemas descritos por equações acessíveis, descrevendo domínios de geometria simples, isto devido a sua dependência na complexidade das equações que regem o fenómeno físico em questão.

A utilização de métodos numéricos na resolução de problemas de análise estrutural, tem sido cada vez mais frequente devido aos avanços na área computacional. Entre estes, destaca-se o método dos elementos finitos (MEF). Este constitui uma ferramenta importante em engenharia, capaz de resolver problemas reais, envolvendo geometrias, comportamentos físicos e condições de fronteira complexas.

1.2. Objetivos da dissertação

Esta dissertação procura desenvolver ferramentas de otimização para obter materiais celulares de microestrutura periódica. Para isso, considera-se o problema da placa com furo, de maneira a simular as condições pretendidas, onde a fronteira do furo corresponde à fronteira da inclusão que constitui a heterogeneidade em materiais celulares.

Para o conseguir, é necessário desenvolver:

a) Um programa de elementos finitos escrito em linguagem Octave que calcule os deslocamentos dos nós e as tensões de von Mises nos elementos.

b) Para realizar a otimização de forma eficiente, o cálculo das derivadas das tensões de von Mises em ordem às variáveis que definem a forma do domínio através do método contínuo de análise de sensibilidades.

c) Para otimizar a forma de materiais celulares de microestrutura periódica, as condições de fronteira de periodicidade.

1.3. Estrutura da dissertação

A presente dissertação encontra-se dividida em seis capítulos. O primeiro faz uma introdução ao tema a ser abordado e são descritas as motivações que levaram a realização da mesma.

No segundo capítulo é feita uma revisão teórica dos conceitos que serão abordados tal como o tipo de elemento finito a ser utilizado nas análises e a análise de sensibilidades analíticas.

O terceiro capítulo aborda a implementação computacional dos conceitos apresentados no capítulo anterior. Corresponde a uma explicação do funcionamento do algoritmo e é feito um teste de convergência de malha para determinar a melhor malha a ser utilizada para a otimização.

No quarto capítulo é apresentada uma nova equação para a parametrização da fronteira da inclusão. São comparados os resultados obtidos pela nova parametrização com a anterior bem como os resultados da otimização com a superelipse com os resultados obtidos por David Negrão [1].

O quinto capítulo são introduzidas as condições de fronteira de periodicidade de maneira a se poder recriar os resultados obtidos em [1] utilizando o modelo de célula periódica.

Por último, no sexto capítulo é feita uma retrospectiva de todo o trabalho realizado e são apresentadas as conclusões desta dissertação.

2. Conceitos teóricos

2.1. Método dos elementos finitos

Diversos problemas com importância para a engenharia podem ser descritos em termos de equações com derivadas parciais, dos quais nem sempre é possível obter uma solução analítica exata. O método dos elementos finitos é, atualmente, o método numérico mais utilizado para obter soluções aproximadas para este tipo de problemas [2].

Na presente dissertação, a discretização do problema é feita utilizando elementos finitos quadriláteros, de 4 nós devido à sua versatilidade.

2.2. Elementos bidimensionais

Dado um problema físico, a sua solução, uma vez conhecida a equação diferencial que rege determinado fenómeno, consiste em obter as funções que são solução dessa equação diferencial. O método dos elementos finitos permite obter estas funções sob a forma de campos discretizados [3].

Os elementos bidimensionais constituem o tipo de elementos utilizados para a resolução de vários problemas físicos, nomeadamente problemas de transmissão de calor ou de cálculo de tensões e deformações pela teoria da elasticidade, sempre que se verifica que a solução é independente em relação a um dos eixos do referencial. É o caso dos problemas de elasticidade plana - tensão plana ou deformação plana, que são abordados nesta dissertação.

Considere-se um problema de elasticidade plana, envolvendo um corpo elástico de volume V e superfície F sob a ação de forças mássicas ρb_i . Recorrendo à forma fraca da equação diferencial de equilíbrio formula-se o problema pelo método dos elementos finitos. Após a integração por partes e tendo em conta a simetria do tensor das tensões, pode-se escrever a seguinte equação:

$$\int_V \sigma_{ij} \bar{\epsilon}_{ij} dV + \int_{F_3} K_{ij} u_j \bar{u}_i dF_3 = \int_V \rho b_i \bar{u}_i dV + \int_{F_2} \sigma_i^* \bar{u}_i dF_2 + \int_{F_3} K_{ij} u_j^{ext} \bar{u}_i dF_3 \quad (1)$$

onde σ_{ij} e $\bar{\epsilon}_{ij}$ são o tensor das tensões associado aos deslocamentos reais e o tensor das deformações associado aos deslocamentos virtuais, u_j é o vetor dos deslocamentos reais e \bar{u}_i o vetor dos deslocamentos virtuais.

Esta expressão corresponde à equação integral procurada. Qualquer campo de deslocamentos cinematicamente admissível que a verifique, verifica simultaneamente as equações diferenciais em todo o domínio V e as condições de fronteira naturais.

Em seguida, passa-se à discretização da equação integral começando pela discretização do domínio V . Este será dividido num número de elementos finitos distintos, sendo cada um identificado por um número. Cada elemento contém um determinado número de nós, que são numerados e cujas coordenadas são identificadas.

Segue-se a discretização das funções de deslocamento e deslocamento virtual, cada uma delas com duas componentes, u_i e \bar{u}_i com $i = 1, 2$.

Considerando elementos com a forma de quadriláteros de 4 nós, um em cada vértice, u_i corresponde a solução aproximada do problema e pode ser descrita pelo somatório,

$$u_i = \sum_{J=1}^4 u_{iJ} N_J = u_{iJ} N_J \quad (2)$$

onde as funções N_J correspondem as funções de forma representadas na figura 1,

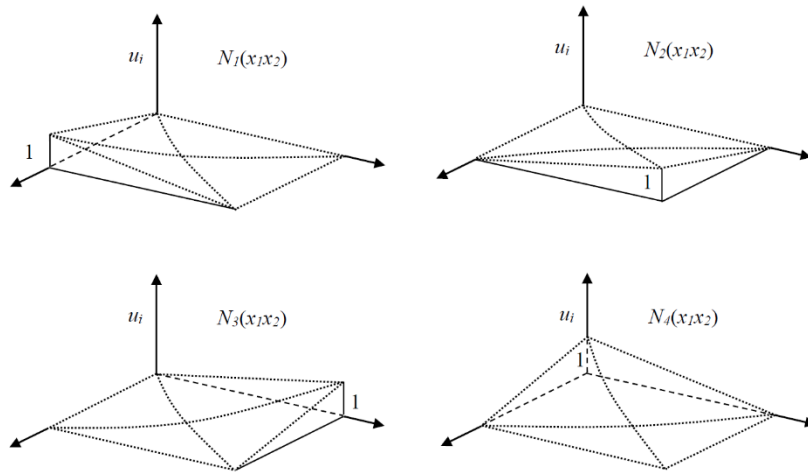


Figura 1 - Funções de forma utilizadas no elemento, extraído de [3]

O cálculo aproximado das componentes dos deslocamentos ou deslocamentos virtuais pode ser feito utilizando estas funções, a partir dos valores nodais desses deslocamentos e em qualquer ponto do domínio do elemento finito considerado.

$$u_i = u_{iJ} N_J \quad \bar{u}_i = \bar{u}_{iK} N_K \quad (3)$$

É importante referir que as funções de forma N_J dependem do grau do polinómio utilizado para aproximar a função exata e por outro lado que o número de nós escolhido para o elemento está associado ao número de constantes que ocorrem nesse polinómio. Portanto, se o polinómio escolhido tiver 4 constantes, obrigatoriamente utilizam-se 4 nós no elemento.

Os elementos finitos usados para resolver os problemas de elasticidade plana podem ter uma geometria triangular ou quadrangular. Cada um deste tipo de elementos pode ter um número variável de nós. Destacam-se duas famílias de elementos distintos, a família dos elementos *lagrangeanos* e a família dos elementos *serendipity*, que se distinguem pelo fato dos primeiros conterem nós interiores, enquanto que os segundos apenas contêm nas arestas ou vértices [3].

2.3. Elementos isoparamétricos

Os elementos isoparamétricos utilizam uma transformação de coordenadas entre referencial auxiliar s, t e o referencial x_1, x_2 para cada elemento resultante da discretização do domínio [3]. A formulação isoparamétrica possibilita a geração de elementos que não sejam retangulares e elementos curvos. A figura 2 demonstra a transformação aplicada no elemento quadrilátero de 4 nós, em que os pontos do referencial auxiliar de coordenadas $(-1, -1)$, $(+1, -1)$, $(+1, +1)$ e $(-1, +1)$ correspondem respectivamente aos nós 1, 2, 3 e 4 do elemento definido no referencial $x_1 x_2$.

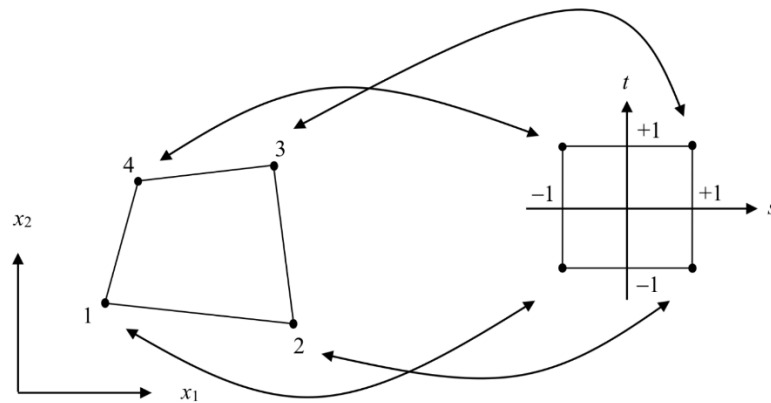


Figura 2 - Elemento isoparamétrico e referencial natural e global do mesmo, extraído de [3].

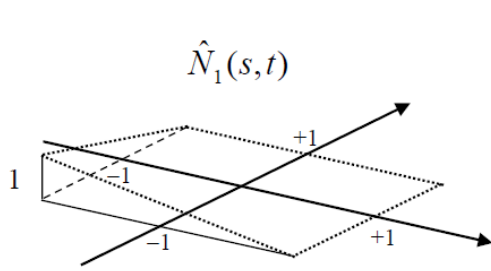
A transformação de coordenadas é definida através das funções de forma que são usadas para aproximar os deslocamentos, isto é, as coordenadas x_1 e x_2 de qualquer ponto no interior do elemento são funções das coordenadas s e t dos seus 4 nós, X_{1j} e X_{2j} [3].

$$x_1 = X_{1j} \hat{N}_j(s, t)$$

$$x_2 = X_{2j} \hat{N}_j(s, t)$$

(4)

As funções de forma $N_j(s, t)$ são iguais para todos os elementos pois o referencial s, t é sempre igual, e são facilmente deduzidas, obtendo-se:



$$\hat{N}_1(s, t) = 0,25(1 - s)(1 - t)$$

$$\hat{N}_2(s, t) = 0,25(1 + s)(1 - t)$$

$$\hat{N}_3(s, t) = 0,25(1 + s)(1 + t)$$

$$\hat{N}_4(s, t) = 0,25(1 - s)(1 + t)$$

Figura 3 - Função de forma $\hat{N}_1(s, t)$.

A generalidade de elementos finitos utilizados em problemas de elasticidade utiliza este tipo de transformação de coordenadas, que possibilita realizar as integrações necessárias para calcular a matriz de rigidez no referencial s, t .

A matriz de rigidez do elemento resulta da discretização do primeiro termo da equação (1). Convertendo o integral no domínio do problema, numa soma de integrais realizados nos NEL elementos finitos em que está dividido o domínio e utilizando as relações (3) para representar os campos de deslocamento e deslocamento virtual que existem em cada integral tendo em conta a lei de Hooke, obtém-se:

$$\sum_{N=1}^{NEL} \int_{V_N} E_{ijnm} u_{n,m} \bar{u}_{i,j} dV = \sum_{N=1}^{NEL} \bar{u}_{i,j} \int_{V_N} E_{ijnm} N_{j,j} N_{k,m} dV u_{nK} = \sum_{N=1}^{NEL} \bar{u}_{i,j} K_{ijnK}^N u_{nK} \quad (6)$$

onde,

$$K_{ijnK}^N = \int_{V_N} E_{ijnm} N_{j,j} N_{k,m} dV \quad (7)$$

corresponde à matriz de rigidez do elemento. Esta tem dimensões 8×8 e o integral é realizado no domínio do elemento definido no referencial $x_1 x_2$.

A relação fundamental dos elementos isoparamétricos estabelece que as funções $N_j(x_1, x_2)$ e $\hat{N}_j(s, t)$ apresentam o mesmo valor para pontos correspondentes, isto é, para pontos de coordenadas x_1, x_2 e s, t que verificam as equações (4) [3]. Esta relação pode escrever-se da seguinte forma:

$$N_j(x_1, x_2) = \hat{N}_j(s, t) = N_j(X_{1K} \hat{N}_K(s, t), X_{2K} \hat{N}_K(s, t)) \quad (8)$$

A figura 4 demonstra esta relação.

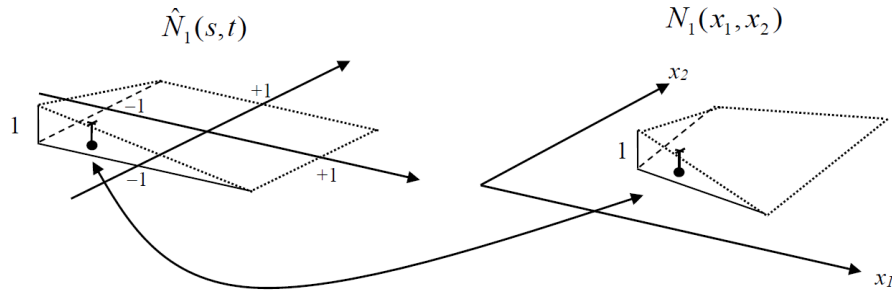


Figura 4 - Valor das duas funções de forma são idênticos para pontos correspondentes, extraído de [3]

Estabelecida esta relação procede-se a determinação da matriz Jacobiana da transformação de coordenadas, que através do seu determinante permite relacionar a área de um elemento infinitesimal nos dois referenciais. Logo, a matriz rigidez dada pela equação (7) pode ser descrita por:

$$K_{iJnK}^N = \int_{\hat{V}_N} E_{ijnm} N_{J,j} N_{K,m} J d\hat{V} \quad (9)$$

em que J corresponde a matriz Jacobiana da transformação de coordenadas,

$$J = \begin{bmatrix} \frac{dx_1}{ds} & \frac{dx_2}{ds} \\ \frac{dx_1}{dt} & \frac{dx_2}{dt} \end{bmatrix} \quad (10)$$

e as funções $N_{J,j}$ e $N_{K,m}$ são agora funções de s, t e a integração é feita no domínio s, t .

A integração numérica de uma função permite obter o valor aproximado do integral dessa função, através do cálculo da função num número discreto de pontos [3].

Utilizando a integração de Gauss com 2×2 pontos, a expressão (9), passa a:

$$K_{iJnK}^N = \sum_{IS=1}^2 \sum_{IT=1}^2 (E_{ijnm} N_{J,j} N_{K,m} J) w_{IS} w_{IT} \quad (11)$$

Prosseguindo com a discretização dos restantes termos da equação integral (1), esta pode escrever-se da seguinte forma:

$$\sum_{N=1}^{NEL} \bar{u}_j K_{ijnK}^N u_K + \sum_{N=1}^{NELF3} \bar{u}_j K_{ijnK}^{N,3} u_K = \sum_{N=1}^{NEL} \bar{u}_j F_{ij}^{1,N} + \sum_{N=1}^{NELF2} \bar{u}_j F_{ij}^{2,N} + \sum_{N=1}^{NELF3} \bar{u}_j F_{ij}^{3,N} \quad (12)$$

onde,

$$\begin{aligned} K_{ijnK}^{N,3} &= \int_{F_3^N} K_{ij} N_j N_K dF \\ F_{ij}^{1,N} &= \int_{V_N} \rho b_i N_j dV \\ F_{ij}^{2,N} &= \int_{F_2^N} \sigma_i^* N_j dF \\ F_{ij}^{3,N} &= \int_{F_3^N} K_{ij} u_j^{ext} N_j dF \end{aligned} \quad (13)$$

e $NELF2$ e $NELF3$, correspondem respetivamente ao número de elementos utilizados nas fronteiras F_2 e F_3 . A esta equação é sempre necessário juntar as condições de fronteira em F_1 :

$$u_j = u_j^* \quad (14)$$

Construída a expressão discretizada da equação integral, procede-se a assemblagem da qual resulta uma matriz de rigidez global e um vetor de forças global. Considerando que os valores nodais dos vetores u_K e \bar{u}_j estão contidos nos vetores U e \bar{U} , correspondentes ao vetor dos deslocamentos globais e dos deslocamentos virtuais globais respetivamente, e tendo em conta a condição de fronteira (14) a equação integral (12) toma a forma:

$$K_{Global} U = F_{Global} \quad (15)$$

Este sistema de equações lineares pode ser resolvido de forma a obter o valor das incógnitas, U , que são os deslocamentos nos nós da malha de elementos finitos. A partir dos deslocamentos nos nós, podem obter-se os deslocamentos no interior dos elementos, utilizando as funções de forma. A partir destes últimos podem calcular-se as tensões e as deformações em qualquer ponto do domínio [3].

2.4. Cálculo de sensibilidades

A análise de sensibilidades em Mecânica Estrutural, dedica-se fundamentalmente ao cálculo de variações do comportamento das estruturas que decorrem da alteração dos parâmetros que condicionam esse comportamento.

Esta área ganhou relevância com o desenvolvimento da otimização estrutural, a partir da década de 80 do século XX. Com efeito os algoritmos usados em otimização estrutural nessa altura requeriam o cálculo dos gradientes para funcionar, e a análise de sensibilidades ganhou importância porque disponibilizava métodos para calcular esses gradientes. Em otimização estrutural a função objetivo e os constrangimentos são obtidos a partir de medidas do desempenho ou da performance estrutural, como por exemplo: peso, volume, deslocamentos ou tensões. Os parâmetros que definem a estrutura, e que são escolhidos pelo algoritmo de otimização são designados variáveis de projeto. Se designarmos uma performance estrutural por ψ e as variáveis de projeto por X_i , a análise de sensibilidades permite obter os gradientes da função objetivo ou dos constrangimentos em ordem às variáveis de projeto, $\partial\psi/\partial X_i$. Exemplos de algoritmos que utilizam gradientes são o NLPQL, o SQP ou o MMA [5].

Como exemplo, pode considerar-se a viga em consola de comprimento L , secção transversal de momento de 2ª ordem I , sujeita a uma carga concentrada P na extremidade como representado na figura 5. Se a performance a analisar for a flecha na extremidade, $\psi = \delta$, e o parâmetro a modificar for o momento de 2ª ordem, $X = I$, a derivada de δ em relação a I será:

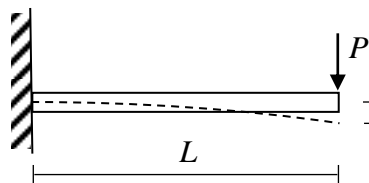

$$\delta = \frac{PL^3}{3EI} \quad \Rightarrow \quad \frac{d\psi}{dX} = \frac{d\delta}{dI} = -\frac{PL^3}{3EI^2}$$

Figura 5 - Viga em consola

Neste caso simples é possível obter uma expressão analítica para $\partial\psi/\partial X$, mas para uma estrutura real de maiores dimensões e complexidade, onde o cálculo das performances normalmente requer a utilização de um modelo de elementos finitos, existem métodos específicos para obter gradientes, que serão abordados em seguida.

2.5. Método contínuo de análise de sensibilidades

Este método deriva da equação de equilíbrio na sua forma contínua. A grande desvantagem das sensibilidades contínuas é a sua complexidade teórica, enquanto que as suas principais vantagens são a utilização de uma formulação matemática rigorosa e uma relação entre gradientes e quantidades físicas não dependente da discretização.[4]

Para obter a formulação contínua, considera-se a equação de equilíbrio estático na sua forma variacional, obtida a partir do princípio dos trabalhos virtuais (ou do princípio da energia potencial total estacionária), que é expresso pela equação (16).

$$a_x(u, \bar{u}) = l_x(\bar{u}) \quad (16)$$

onde $a_x(u, \bar{u})$ é o termo bilinear que contabiliza o trabalho virtual das forças internas (ou a variação da energia de deformação do sistema) e $l_x(\bar{u})$ é o termo linear relacionado com o trabalho virtual realizado pelas forças externas aplicadas (ou a variação da energia potencial das forças externas). Na equação (16) os termos u e \bar{u} representam o campo de deslocamentos e o campo de deslocamentos virtuais, respetivamente, que são considerados contínuos.

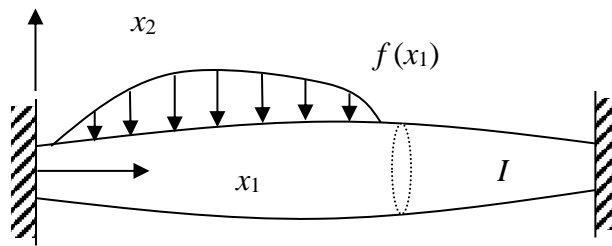


Figura 6 - Viga à flexão

Para o caso de uma viga sujeita à flexão, como a representada na figura 6, com a carga aplicada distribuída $f(x_1)$ e com um momento de segunda ordem variável $I(x_1)$, os 2 termos da equação (16) são:

$$a_x(u, \bar{u}) = \int_0^L EI \frac{\partial^2 u_2}{\partial x_1^2} \frac{\partial^2 \bar{u}_2}{\partial x_1^2} dx_1 \quad (17)$$

$$l_x(\bar{u}) = \int_0^L f \bar{u} dx_1 \quad (18)$$

O subscrito X em (16), (17) e (18) representa a dependência do parâmetro X que existe nos dois termos, em todas essas equações. Para obter a formulação contínua a partir de (16) é conveniente considerar separadamente o caso das variáveis afetarem apenas a rigidez (E e I na equação (17)) e o caso das variáveis afetarem a forma (L nas equações (17) e (18)) ou a orientação do referencial local de cada elemento em relação ao referencial global.

Para variáveis de forma ou de orientação é útil considerar o conceito de derivada material da Mecânica dos Meios Contínuos. Considere-se um domínio Ω em 1, 2 ou 3 dimensões, e um único parâmetro τ que define a passagem da configuração inicial Ω para uma configuração modificada Ω_τ . As coordenadas x_τ de qualquer ponto na nova configuração podem ser obtidas a partir de uma transformação T , expressa pela equação (19).

$$x_\tau = T(x, \tau) \quad (19)$$

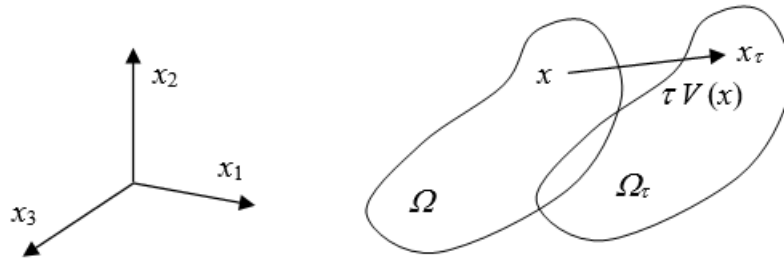


Figura 7 - Variação de forma do domínio Ω

Pode-se considerar essa transformação de Ω em Ω_τ como um processo dinâmico. No instante inicial $\tau = 0$, o domínio é Ω . Com a mudança de forma, os pontos que o constituem movem-se para a posição $x_\tau = T(x, \tau)$, em que τ é um parâmetro semelhante ao tempo. Considere-se um campo de velocidades associado à variação de forma, definido por:

$$V(x) = \frac{\partial x}{\partial \tau} = \frac{\partial T(x, \tau)}{\partial \tau} \quad (20)$$

Então os deslocamentos na configuração final, $u_\tau(x_\tau)$, dependem de τ de duas maneiras. Em primeiro lugar são a solução de um problema definido no domínio Ω_τ e em segundo lugar são calculados num ponto que se move com τ . A derivada material no ponto $x \in \Omega$, se existir, é definida por:

$$\dot{u}(x) = \frac{d}{d\tau} u_\tau(x + \tau V(x)) \Big|_{\tau=0} = \lim_{\tau \rightarrow 0} \frac{u_\tau(x + \tau V(x)) - u(x)}{\tau} \quad (21)$$

Se u_τ tem uma extensão regular a uma vizinhança de Ω_τ demonstra-se que:

$$\begin{aligned}\dot{u}(x) &= \lim_{\tau \rightarrow 0} \frac{u_\tau(x + \tau V(x)) - u(x)}{\tau} \\ \dot{u}(x) &= \lim_{\tau \rightarrow 0} \frac{u_\tau(x_\tau) - u_\tau(x) + u_\tau(x) - u(x)}{\tau} \\ \dot{u}(x) &= \lim_{\tau \rightarrow 0} \frac{u_\tau(x) - u(x)}{\tau} + \lim_{\tau \rightarrow 0} \frac{u_\tau(x_\tau) - u_\tau(x)}{\tau} \\ \dot{u}(x) &= u'_\nu(x) + \frac{\partial u}{\partial x} \frac{\partial x}{\partial \tau} \\ \dot{u}(x) &= u'_\nu(x) + \nabla u^T V(x)\end{aligned}\tag{22}$$

onde u'_ν é a derivada parcial de u em ordem a τ , avaliada no ponto x devido à variação de forma V e o operador Nabla, $\nabla = \left\{ \frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \quad \frac{\partial}{\partial x_3} \right\}$, permite calcular o gradiente dos deslocamentos ∇u .

$$\nabla u = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} & \frac{\partial u_1}{\partial x_3} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} & \frac{\partial u_2}{\partial x_3} \\ \frac{\partial u_3}{\partial x_1} & \frac{\partial u_3}{\partial x_2} & \frac{\partial u_3}{\partial x_3} \end{bmatrix}\tag{23}$$

O termo $\nabla u^T V$ será então igual a (24)

$$\nabla u^T V = \begin{Bmatrix} \frac{\partial u_1}{\partial x_1} V_1 + \frac{\partial u_1}{\partial x_2} V_2 + \frac{\partial u_1}{\partial x_3} V_3 \\ \frac{\partial u_2}{\partial x_1} V_1 + \frac{\partial u_2}{\partial x_2} V_2 + \frac{\partial u_2}{\partial x_3} V_3 \\ \frac{\partial u_3}{\partial x_1} V_1 + \frac{\partial u_3}{\partial x_2} V_2 + \frac{\partial u_3}{\partial x_3} V_3 \end{Bmatrix}\tag{24}$$

O conceito de derivada material também pode ser aplicado a um funcional,

$$\psi = \int_{\Omega} f(x) d\Omega\tag{25}$$

Obtendo-se:

$$\psi' = \int_{\Omega} [f'_v(x) + \nabla f(x)^T V(x) + f(x) \text{div} V(x)] d\Omega \quad (26)$$

E esta expressão pode agora ser aplicada à equação de equilíbrio (16), que poderá tomar a forma:

$$\int_{\Omega} c(u, \bar{u}) d\Omega = \int_{\Omega} f^T \bar{u} d\Omega \quad (27)$$

O que resulta de considerar os dois termos de (16) iguais a,

$$a_x(u, \bar{u}) = \int_{\Omega} c(u, \bar{u}) d\Omega \quad \text{e} \quad l_x(\bar{u}) = \int_{\Omega} f^T \bar{u} d\Omega \quad (28)$$

Aplicando a equação (26) ao primeiro termo de (27), e considerando que a variável X está agora associada ao campo de velocidades V , obtém-se:

$$a_v(u, \bar{u})' = \int_{\Omega} [c(u'_v, \bar{u}) + c(u, \bar{u}'_v) + \nabla c(u, \bar{u})^T V + c(u, \bar{u}) \text{div} V] d\Omega \quad (29)$$

Sabendo que $\dot{u}(x) = u'_v(x) + \nabla u^T V$, obtém-se

$$a_v(u, \bar{u})' = \int_{\Omega} [c(\dot{u} - \nabla u^T V, \bar{u}) + c(u, \dot{\bar{u}} - \nabla \bar{u}^T V) + \nabla c(u, \bar{u})^T V + c(u, \bar{u}) \text{div} V] d\Omega \quad (30)$$

Considerando agora que a derivada material do campo de deslocamentos virtuais é zero, isto é, que

$$\dot{\bar{u}}(x) = \bar{u}'_v(x) + \nabla \bar{u}^T V = 0 \quad (31)$$

Obtém-se

$$a_v(u, \bar{u})' = \int_{\Omega} [c(\dot{u}, \bar{u}) - c(\nabla u^T V, \bar{u}) - c(u, \nabla \bar{u}^T V) + \nabla c(u, \bar{u})^T V + c(u, \bar{u}) \text{div} V] d\Omega \quad (32)$$

De igual modo, aplicando a equação (26) ao segundo termo de (27), obtém-se

$$\begin{aligned}
l'_v(\bar{u}) &= \int_{\Omega} \left[f'_v{}^T \bar{u} + f^T \bar{u}'_v + \nabla(f^T \bar{u})^T V + f^T \bar{u} \operatorname{div} V \right] d\Omega \\
l'_v(\bar{u}) &= \int_{\Omega} \left[f'_v{}^T \bar{u} + f^T (\dot{\bar{u}} - \nabla \bar{u}^T V) + \nabla(f^T \bar{u})^T V + f^T \bar{u} \operatorname{div} V \right] d\Omega \\
l'_v(\bar{u}) &= \int_{\Omega} \left[f'_v{}^T \bar{u} - f^T (\nabla \bar{u}^T V) + \nabla(f^T \bar{u})^T V + f^T \bar{u} \operatorname{div} V \right] d\Omega
\end{aligned} \tag{33}$$

Igualando (32) e (33) pode escrever-se a equação de equilíbrio (16) na seguinte forma,

$$\begin{aligned}
\int_{\Omega} \left[c(\dot{u}, \bar{u}) - c(\nabla u^T V, \bar{u}) - c(u, \nabla \bar{u}^T V) + \nabla c(u, \bar{u})^T V + c(u, \bar{u}) \operatorname{div} V \right] d\Omega \\
= \int_{\Omega} \left[f'_v{}^T \bar{u} - f^T (\nabla \bar{u}^T V) + \nabla(f^T \bar{u})^T V + f^T \bar{u} \operatorname{div} V \right] d\Omega
\end{aligned} \tag{34}$$

ou

$$a'_v(\dot{u}, \bar{u}) = l'_v(\bar{u}) - a'_v(u, \bar{u}) \tag{35}$$

Com

$$a'_v(\dot{u}, \bar{u}) = \int_{\Omega} c(\dot{u}, \bar{u}) d\Omega \tag{36}$$

$$l'_v(\bar{u}) = \int_{\Omega} \left[f'_v{}^T \bar{u} - f^T (\nabla \bar{u}^T V) + \nabla(f^T \bar{u})^T V + f^T \bar{u} \operatorname{div} V \right] d\Omega \tag{37}$$

$$a'_v(u, \bar{u}) = \int_{\Omega} \left[-c(\nabla u^T V, \bar{u}) - c(u, \nabla \bar{u}^T V) + \nabla c(u, \bar{u})^T V + c(u, \bar{u}) \operatorname{div} V \right] d\Omega \tag{38}$$

A equação (35) representa a derivada material de (16) para variações de forma e de orientação expressas pelo campo de velocidades V .

Considere-se agora um funcional genérico, escrito na forma integral,

$$\psi = \int_{\Omega} g(u, \nabla u) d\Omega \tag{39}$$

A derivada material do funcional, de acordo com (26) é

$$\psi' = \int_{\Omega} [g_u u'_v + g_{\nabla u} \nabla u'_v + \nabla g^T V + g \cdot \text{div} V] d\Omega \quad (40)$$

Com $g_u = \frac{\partial g(u, \nabla \tilde{u})}{\partial u}$ e $g_{\nabla u} = \frac{\partial g(\tilde{u}, \nabla u)}{\partial (\nabla u)}$ onde (\sim) significa que a quantidade é

mantida constante durante a diferenciação. Considerando que $\dot{u}(x) = u'_v(x) + \nabla u^T V$ obtém-se:

$$\psi' = \int_{\Omega} [g_u \dot{u} + g_{\nabla u} \nabla \dot{u} - g_u (\nabla u^T V) - g_{\nabla u} \nabla (\nabla u^T V) + \nabla g^T V + g \cdot \text{div} V] d\Omega \quad (41)$$

Para obter a derivada material do funcional ψ explicitamente em função do campo de velocidades, é necessário calcular a derivada material dos deslocamentos explicitamente em função do campo de velocidades associado à variação de forma e orientação, \dot{u} . O método direto e o método adjunto podem ser usados para esse efeito. Como para o cálculo das derivadas utilizando o método discreto, a escolha depende do número de variáveis e do número de performances. Como o número de performances é habitualmente menor que o número de variáveis, apresenta-se apenas o método adjunto.[4]

Uma equação adjunta é obtida substituindo \dot{u} por $\bar{\lambda}$ em (41) e igualando o termo energético bilinear da equação (16) aos termos da equação (41) que contêm $\bar{\lambda}$.

$$a(\lambda, \bar{\lambda}) = \int_{\Omega} [g_u \bar{\lambda} + g_{\nabla u} \nabla \bar{\lambda}] d\Omega \quad (42)$$

A equação (42) tem uma solução única λ designada variável adjunta associada ao constrangimento ψ . Para tirar partido da equação anterior, substitua-se $\bar{\lambda}$ por \dot{u} ,

$$a(\lambda, \dot{u}) = \int_{\Omega} [g_u \dot{u} + g_{\nabla u} \nabla \dot{u}] d\Omega \quad (43)$$

De igual modo pode-se substituir \bar{u} por λ na equação (35) obtendo-se,

$$a_v(\dot{u}, \lambda) = l'_v(\lambda) - a'_v(u, \lambda) \quad (44)$$

Como o termo da energia de deformação é simétrico, os dois termos do lado esquerdo nas equações (43) e (44) são iguais. Igualando os termos do lado direito obtém-se,

$$\int_{\Omega} [g_u \dot{u} + g_{\nabla u} \nabla \dot{u}] d\Omega = l'_v(\lambda) - a'_v(u, \lambda) \quad (45)$$

Utilizando o método adjunto, a derivada material do funcional ψ para variações de forma ou orientação do domínio Ω , definidas pelo campo de velocidades V pode ser calculado pela equação (46),

$$\psi' = l'_v(\lambda) - a'_v(u, \lambda) - \int_{\Omega} [g_u (\nabla u^T V) - g_{\nabla u} \nabla (\nabla u^T V) + \nabla g^T V + g \cdot \text{div} V] d\Omega \quad (46)$$

Para proceder ao cálculo pelo método adjunto é necessário obter primeiro a carga adjunta para a performance em estudo, isto é, o segundo termo da equação (42). Em seguida é calculado o campo dos deslocamentos adjuntos λ através da solução de um sistema com o mesmo termo energético usado na análise de equilíbrio inicial, mas agora com a carga adjunta. Este campo de deslocamentos adjunto só depende da performance e é independente da variável de projeto.[4]

Conhecida a solução do problema inicial, u , e do problema adjunto, λ , é ainda necessário obter o campo de velocidades, V , para poder calcular (46). O cálculo desta equação é feito substituindo em cada termo u , λ e V avaliados anteriormente.

O cálculo de λ depende da performance e do componente estrutural escolhido. O campo V e os vários termos de (46) dependem da geometria de cada componente estrutural.

Resumidamente, para cálculo de (46) é necessário:

1. Obter a solução de equilíbrio para o problema, u , resolvendo (16).
2. Determinar o vetor de forças adjunto para a performance e o componente estrutural escolhidos e resolver (42) para obter λ .
3. Definir o campo de velocidades V .
4. Conhecidos u , λ e V determinar $l'_v(\lambda)$ e $a'_v(u, \lambda)$. Se a variável não tem influência sobre o carregamento aplicado então $l'_v(\lambda)$ é zero. O cálculo do termo $a'_v(u, \lambda)$ depende do componente estrutural e é apresentado em seguida para placas com esforços de membrana.
5. Finalmente é necessário obter o termo $\int_{\Omega} [g_u (\nabla u^T V) - g_{\nabla u} \nabla (\nabla u^T V) + \nabla g^T V + g \cdot \text{div} V] d\Omega$, que depende do componente estrutural e da performance selecionados.

2.6. Placas sujeitas a esforços de membrana

Considere-se uma placa de espessura constante t num estado plano de tensão como representado na figura 7.

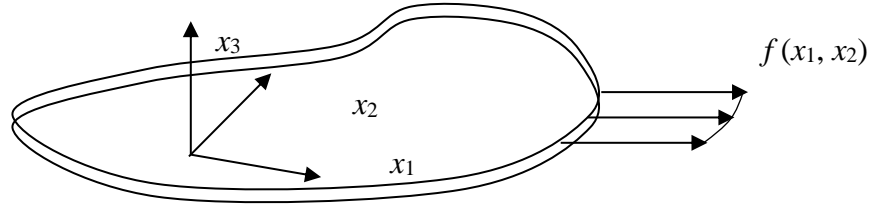


Figura 8 - Placa no estado de tensão plano.

Quando a placa se deforma sob a ação de esforços de membrana, o termo energético bilinear é igual a:

$$a_x(u, \bar{u}) = \int_{\Omega} c(u, \bar{u}) d\Omega = \int_A \sigma_{ij} \bar{\varepsilon}_{ij} t dA \quad (47)$$

onde σ_{ij} e $\bar{\varepsilon}_{ij}$ são respetivamente o tensor das tensões associado ao campo de deslocamentos real u e o tensor das deformações associado ao campo de deslocamentos virtual \bar{u} . Os vetores de deslocamento u e \bar{u} apenas contêm dois componentes em cada nó,

$$u = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \quad \bar{u} = \begin{Bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{Bmatrix} \quad (48)$$

Considerando o estado plano de tensão, as componentes não nulas dos tensores σ_{ij} e $\bar{\varepsilon}_{ij}$ são,

$$\sigma_{11} = \frac{E}{1-\nu^2} \left(\frac{\partial u_1}{\partial x_1} + \nu \frac{\partial u_2}{\partial x_2} \right) \quad \bar{\varepsilon}_{11} = \frac{\partial \bar{u}_1}{\partial x_1}$$

$$\sigma_{22} = \frac{E}{1-\nu^2} \left(\nu \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \right) \quad \bar{\varepsilon}_{22} = \frac{\partial \bar{u}_2}{\partial x_2}$$

$$\sigma_{12} = \frac{E}{1-\nu^2} \frac{(1-\nu)}{4} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) \quad \bar{\epsilon}_{12} = \frac{1}{2} \left(\frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_1} \right) \quad (49)$$

Com E representando o módulo de Young e ν o coeficiente de Poisson. Expandindo (47) obtém-se,

$$\int_{\Omega} c(u, \bar{u}) d\Omega = \int_A \frac{E}{1-\nu^2} \left[\frac{\partial u_1}{\partial x_1} \frac{\partial \bar{u}_1}{\partial x_1} + \nu \frac{\partial u_2}{\partial x_2} \frac{\partial \bar{u}_1}{\partial x_1} + \nu \frac{\partial u_1}{\partial x_1} \frac{\partial \bar{u}_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial \bar{u}_2}{\partial x_2} \right. \\ \left. + \frac{1-\nu}{4} \left(\frac{\partial u_1}{\partial x_2} \frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial u_1}{\partial x_2} \frac{\partial \bar{u}_2}{\partial x_1} + \frac{\partial u_2}{\partial x_1} \frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \frac{\partial \bar{u}_2}{\partial x_1} \right) \right] t dA \quad (50)$$

Assumindo que apenas poderão ocorrer alterações de forma da placa, isto é, que cada ponto apenas se poderá mover no plano da placa, o campo de velocidades apenas terá dois componentes $V = \begin{Bmatrix} V_1 \\ V_2 \end{Bmatrix}$ e o termo $\nabla u^T V$ apresentado em (24) será neste caso igual a,

$$\nabla u^T V = \begin{Bmatrix} \frac{\partial u_1}{\partial x_1} V_1 + \frac{\partial u_1}{\partial x_2} V_2 \\ \frac{\partial u_2}{\partial x_1} V_1 + \frac{\partial u_2}{\partial x_2} V_2 \end{Bmatrix} \quad (51)$$

Os 4 termos que constituem a equação (38) são então,

$$c(\nabla u^T V, \bar{u}) = \frac{E}{1-\nu^2} \left\{ \left(\frac{\partial \bar{u}_1}{\partial x_1} + \nu \frac{\partial \bar{u}_2}{\partial x_2} \right) \left(\frac{\partial^2 u_1}{\partial x_1^2} V_1 + \frac{\partial u_1}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial^2 u_1}{\partial x_2 \partial x_1} V_2 + \frac{\partial u_1}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) \right. \\ \left. + \left(\nu \frac{\partial \bar{u}_1}{\partial x_1} + \frac{\partial \bar{u}_2}{\partial x_2} \right) \left(\frac{\partial^2 u_2}{\partial x_1 \partial x_2} V_1 + \frac{\partial u_2}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial^2 u_2}{\partial x_2^2} V_2 + \frac{\partial u_2}{\partial x_2} \frac{\partial V_2}{\partial x_2} \right) \right. \\ \left. + \frac{1-\nu}{4} \left[\left(\frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_1} \right) \left(\frac{\partial^2 u_2}{\partial x_1^2} V_1 + \frac{\partial u_2}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial^2 u_2}{\partial x_2 \partial x_1} V_2 + \frac{\partial u_2}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) \right. \right. \\ \left. \left. + \left(\frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_1} \right) \left(\frac{\partial^2 u_1}{\partial x_1 \partial x_2} V_1 + \frac{\partial u_1}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial^2 u_1}{\partial x_2^2} V_2 + \frac{\partial u_1}{\partial x_2} \frac{\partial V_2}{\partial x_2} \right) \right] \right\} \quad (52)$$

$$\begin{aligned}
c(u, \nabla \bar{u}^T V) &= \frac{E}{1-\nu^2} \left\{ \left(\frac{\partial u_1}{\partial x_1} + \nu \frac{\partial u_2}{\partial x_2} \right) \left(\frac{\partial^2 \bar{u}_1}{\partial x_1^2} V_1 + \frac{\partial \bar{u}_1}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial^2 \bar{u}_1}{\partial x_2 \partial x_1} V_2 + \frac{\partial \bar{u}_1}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) \right. \\
&\quad + \left(\nu \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \right) \left(\frac{\partial^2 \bar{u}_2}{\partial x_1 \partial x_2} V_1 + \frac{\partial \bar{u}_2}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial^2 \bar{u}_2}{\partial x_2^2} V_2 + \frac{\partial \bar{u}_2}{\partial x_2} \frac{\partial V_2}{\partial x_2} \right) \\
&\quad + \frac{1-\nu}{4} \left[\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) \left(\frac{\partial^2 \bar{u}_2}{\partial x_1^2} V_1 + \frac{\partial \bar{u}_2}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial^2 \bar{u}_2}{\partial x_2 \partial x_1} V_2 + \frac{\partial \bar{u}_2}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) \right. \\
&\quad \left. \left. + \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) \left(\frac{\partial^2 \bar{u}_1}{\partial x_1 \partial x_2} V_1 + \frac{\partial \bar{u}_1}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial^2 \bar{u}_1}{\partial x_2^2} V_2 + \frac{\partial \bar{u}_1}{\partial x_2} \frac{\partial V_2}{\partial x_2} \right) \right] \right\} \quad (53)
\end{aligned}$$

$$\begin{aligned}
\nabla c(u, \bar{u})^T V &= \frac{E}{1-\nu^2} \left\{ \left(\frac{\partial^2 u_1}{\partial x_1^2} V_1 + \frac{\partial^2 u_1}{\partial x_1 \partial x_2} V_2 \right) \left(\frac{\partial \bar{u}_1}{\partial x_1} + \nu \frac{\partial \bar{u}_2}{\partial x_2} \right) \right. \\
&\quad + \left(\frac{\partial^2 u_2}{\partial x_2 \partial x_1} V_1 + \frac{\partial^2 u_2}{\partial x_2^2} V_2 \right) \left(\nu \frac{\partial \bar{u}_1}{\partial x_1} + \frac{\partial \bar{u}_2}{\partial x_2} \right) \\
&\quad + \left(\frac{\partial u_1}{\partial x_1} + \nu \frac{\partial u_2}{\partial x_2} \right) \left(\frac{\partial^2 \bar{u}_1}{\partial x_1^2} V_1 + \frac{\partial^2 \bar{u}_1}{\partial x_1 \partial x_2} V_2 \right) \\
&\quad + \left(\nu \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \right) \left(\frac{\partial^2 \bar{u}_2}{\partial x_2 \partial x_1} V_1 + \frac{\partial^2 \bar{u}_2}{\partial x_2^2} V_2 \right) \\
&\quad + \frac{1-\nu}{4} \left[\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) \left(\frac{\partial^2 \bar{u}_1}{\partial x_2 \partial x_1} V_1 + \frac{\partial^2 \bar{u}_1}{\partial x_2^2} V_2 \right) \right. \\
&\quad + \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) \left(\frac{\partial^2 \bar{u}_2}{\partial x_1^2} V_1 + \frac{\partial^2 \bar{u}_2}{\partial x_1 \partial x_2} V_2 \right) \\
&\quad + \left(\frac{\partial^2 u_1}{\partial x_2 \partial x_1} V_1 + \frac{\partial^2 u_1}{\partial x_2^2} V_2 \right) \left(\frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_1} \right) \\
&\quad \left. \left. + \left(\frac{\partial^2 u_2}{\partial x_1^2} V_1 + \frac{\partial^2 u_2}{\partial x_1 \partial x_2} V_2 \right) \left(\frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_1} \right) \right] \right\} \quad (54)
\end{aligned}$$

$$\begin{aligned}
c(u, \bar{u}) \operatorname{div} V &= \frac{E}{1-\nu^2} \left[\frac{\partial u_1}{\partial x_1} \frac{\partial \bar{u}_1}{\partial x_1} + \nu \frac{\partial u_2}{\partial x_2} \frac{\partial \bar{u}_1}{\partial x_1} + \nu \frac{\partial u_1}{\partial x_1} \frac{\partial \bar{u}_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial \bar{u}_2}{\partial x_2} \right. \\
&\quad \left. + \frac{1-\nu}{4} \left(\frac{\partial u_1}{\partial x_2} \frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial u_1}{\partial x_2} \frac{\partial \bar{u}_2}{\partial x_1} + \frac{\partial u_2}{\partial x_1} \frac{\partial \bar{u}_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \frac{\partial \bar{u}_2}{\partial x_1} \right) \right] \left(\frac{\partial V_1}{\partial x_1} + \frac{\partial V_2}{\partial x_2} \right)
\end{aligned} \tag{55}$$

E, utilizando as relações (52) a (55), a eq. (38) pode ser escrita sob a forma seguinte para uma placa num estado plano de tensão,

$$\begin{aligned}
a'_\nu(u, \bar{u}) &= - \int_A \left\{ \sigma_{11} \left(\frac{\partial \bar{u}_1}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial \bar{u}_1}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) + \sigma_{22} \left(\frac{\partial \bar{u}_2}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_2} \frac{\partial V_2}{\partial x_2} \right) \right. \\
&\quad + \sigma_{12} \left(\frac{\partial \bar{u}_1}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial \bar{u}_1}{\partial x_2} \frac{\partial V_2}{\partial x_2} + \frac{\partial \bar{u}_2}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial \bar{u}_2}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) \\
&\quad + \bar{\sigma}_{11} \left(\frac{\partial u_1}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) + \bar{\sigma}_{22} \left(\frac{\partial u_2}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial u_2}{\partial x_2} \frac{\partial V_2}{\partial x_2} \right) \\
&\quad + \bar{\sigma}_{12} \left(\frac{\partial u_1}{\partial x_1} \frac{\partial V_1}{\partial x_2} + \frac{\partial u_1}{\partial x_2} \frac{\partial V_2}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \frac{\partial V_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \frac{\partial V_2}{\partial x_1} \right) \\
&\quad \left. - (\sigma_{11} \bar{\epsilon}_{11} + \sigma_{22} \bar{\epsilon}_{22} + 2\sigma_{12} \bar{\epsilon}_{12}) \left(\frac{\partial V_1}{\partial x_1} + \frac{\partial V_2}{\partial x_2} \right) \right\} t \, dA
\end{aligned} \tag{56}$$

2.7. Otimização estrutural

A otimização estrutural tem constituído um tópico de interesse por mais de 100 anos. Os primeiros trabalhos analíticos sobre otimização estrutural foram publicados por Maxwell (1890) e Michell (1904). No entanto, devido ao surgimento dos primeiros computadores digitais no início da década de 1950, esta área teve um grande desenvolvimento. Estes, deram um forte impulso à aplicação dos métodos numéricos de Programação Linear, como por exemplo o algoritmo SIMPLEX. Estes métodos foram aplicados à resolução de problemas de otimização estrutural, envolvendo estruturas reticuladas. Para além do avanço tecnológico ao nível da informática durante esta década, verificaram-se avanços teóricos significativos na área dos métodos numéricos aplicados à mecânica estrutural. A descoberta do método dos elementos finitos possibilitou pela primeira vez, a realização de análises a estruturas realmente complexas. O trabalho desenvolvido por Schmit (1960) introduziu pela primeira vez a ideia de combinar a análise estrutural por elementos finitos com os métodos numéricos de otimização, demonstrando a viabilidade deste processo na resolução de problemas reais. Esta combinação revolucionou a metodologia do projeto ótimo de estruturas. Os métodos de Programação Não Linear, tiveram desenvolvimentos substanciais nas décadas de 1970 e 1980, de entre estes métodos destacam-se o método da Programação Quadrática Sequencial (SQP) e o Método das Assíptotas Móveis (MMA) [5].

A otimização estrutural, no seu conceito básico, consiste em determinar um conjunto de parâmetros, denominados de variáveis de projeto, de modo a minimizar ou maximizar uma determinada função objetivo sem violar os respetivos constrangimentos. A metodologia do projeto ótimo está associada a um processo iterativo controlado por um algoritmo de otimização, que pesquisa uma solução ótima no espaço das variáveis de projeto limitado pelos constrangimentos. A otimização estrutural pode ser classificada em três categorias principais: otimização dimensional, otimização de forma e otimização topológica. A figura 8 mostra como se relacionam as principais categorias da otimização estrutural com as várias etapas da atividade de projeto.

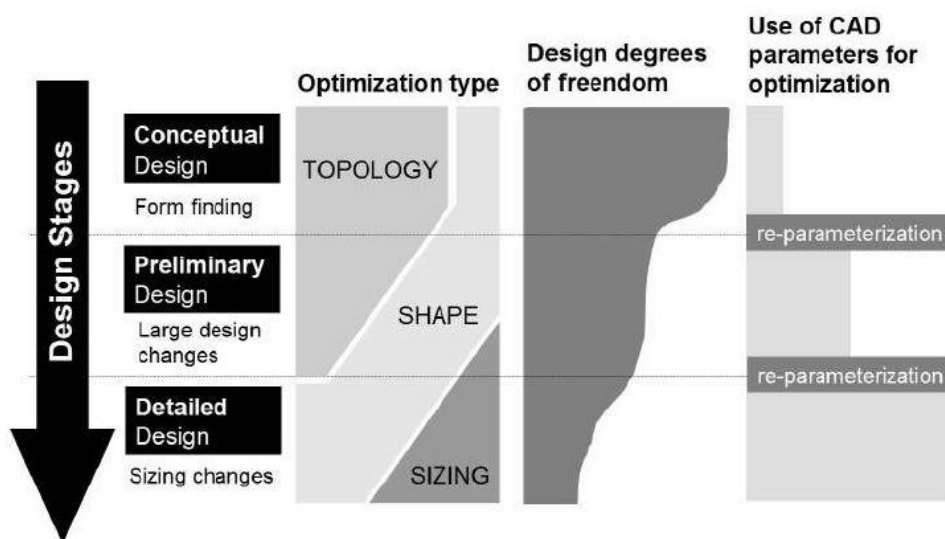


Figura 9 - Categorias de otimização estrutural associadas às várias etapas do projeto, extraído de [5]

Tem-se uma ideia do domínio a ser ocupado por uma determinada estrutura e quais as condições de fronteira que se pretendem satisfazer, na fase conceptual de um projeto. Assim sendo, a primeira categoria da otimização estrutural a ser utilizada, é a otimização topológica que consiste num procedimento que distribui racionalmente o material disponível numa área ou volume fixos, através da remoção gradual de pequenas porções de material. No entanto, na fase do projeto de pormenor, resta apenas fazer a otimização dimensional, que tem como objetivo especificar as dimensões dos membros da estrutura.

2.7.1. Otimização dimensional

No início do período da otimização estrutural, esta estava sobretudo orientada para a otimização dimensional. A otimização dimensional é utilizada para encontrar as dimensões ótimas de uma estrutura, de modo a torná-la o mais eficiente possível para uma determinada função objetivo. Neste tipo de otimização as variáveis de projeto são as dimensões dos elementos estruturais que podem variar de forma contínua ou discreta. Este processo é vastamente utilizado, por exemplo, para encontrar as dimensões das secções transversais de barras em treliças. A figura ilustra um exemplo de otimização dimensional onde as variáveis de projeto correspondem aos diâmetros das barras.

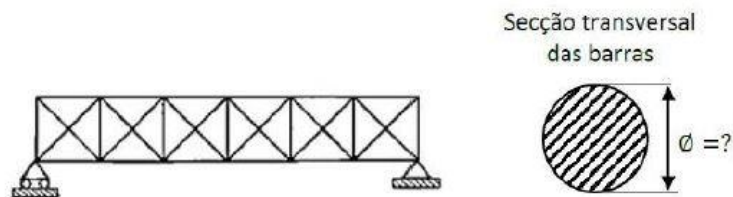


Figura 10 - Exemplo de otimização dimensional, extraído de [1]

Neste tipo de otimização a função objetivo não apresenta nenhuma melhoria do ponto de vista da forma e da topologia da estrutura, durante todo o processo de otimização, o que constitui uma desvantagem.

2.7.2. Otimização de forma

A otimização de forma começa a surgir em várias publicações no início da década de 1970, como sendo a otimização de uma estrutura cuja topologia é fixa e considera-se variável a fronteira delimitadora do domínio ocupado pela estrutura. A fronteira do domínio pode ser definida por um conjunto de pontos, uma linha ou uma superfície. Esta linha ou superfície pode ser modelada através de funções matemáticas como splines. Neste tipo de otimização as variáveis de projeto são variáveis contínuas e podem ser as coordenadas dos pontos, através dos quais se geram as splines ou os parâmetros de uma determinada equação analítica que descreve a fronteira do problema. A otimização de forma tem como objetivo encontrar o conjunto dos valores das variáveis de projeto, ou seja, a forma ótima da fronteira, que minimizam a função objetivo do problema. A figura mostra o exemplo de otimização de forma de uma estrutura.

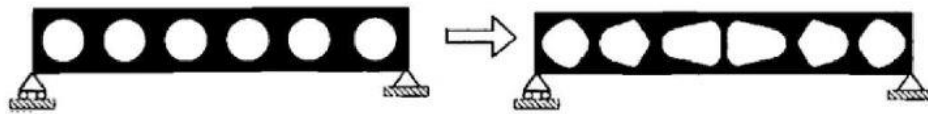


Figura 11 - Exemplo de otimização de forma, extraído de [1]

2.7.3. Parametrização da fronteira em otimização de forma

Na presente dissertação, pretende-se determinar a forma ótima de inclusões existentes num domínio retangular, que constituem a mais pequena heterogeneidade de um material celular. A forma ótima da inclusão é determinada através da resolução de um problema de otimização de forma, onde a parametrização utilizada consiste em descrever a fronteira da inclusão por uma expressão analítica que controla a posição dos nós da fronteira. Neste trabalho foram consideradas duas equações distintas para descrever a curva da fronteira da inclusão no plano. A primeira parametrização consistiu em modelar a curva da inclusão através de uma *spline* de 5 nós. Posteriormente utilizou-se uma parametrização através da equação da superelipse. Esta é definida por 3 parâmetros apenas. A descrição da fronteira da inclusão através da equação da superelipse já foi feita por alguns autores como se pode verificar em [6], [7].

3. Implementação computacional

O programa *otimizacao_pcf* em linguagem OCTAVE, permite realizar a otimização de forma da placa com furo, recorrendo a um processo iterativo. Neste processo, utiliza-se o algoritmo de otimização *mmasub* disponibilizado por Svanberg [12], do qual constam as subrotinas *mmasub* e *subsolv*. Bem como, o algoritmo *PROAES_membrana* para realizar o cálculo das tensões no bordo do furo e das respectivas sensibilidade.

Na figura 11 apresenta-se o fluxograma do programa elaborado.

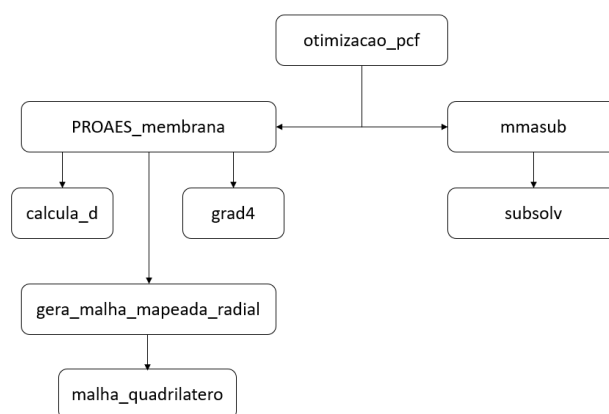


Figura 12 - Fluxograma do algoritmo de otimização desenvolvido

3.1. Definição do problema

Como primeira abordagem, considera-se o problema da placa com furo, já conhecido da literatura. Considera-se para esta análise apenas $\frac{1}{4}$ de placa, devido a geometria da mesma apresentar simetria. O furo da placa é modelado através de uma *spline* radial com 5 pontos, que se encontram inicialmente a uma distância de 1 unidade do centro do referencial xy .

Pretende-se minimizar da área da placa, variando a geometria da fronteira do furo, tendo em conta as tensões $\sigma_x = 200 Pa$ e $\sigma_y = 100 Pa$ que se encontram aplicadas nos bordos da placa, bem como, um constrangimento de tensão de $\sigma_{ADM} = 300 Pa$ admissível nos elementos constituintes do contorno do furo. A figura 9 ilustra a geometria inicial da placa considerada.

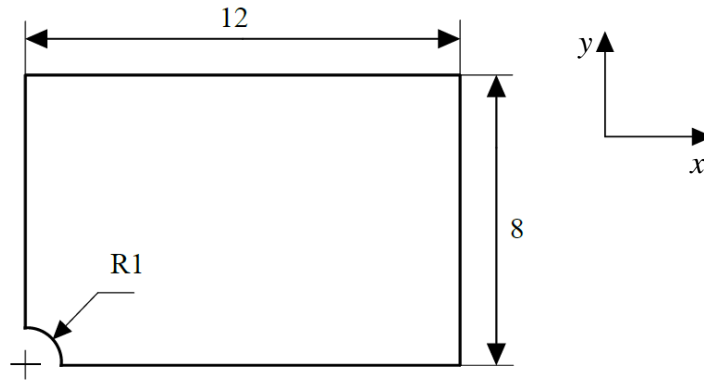


Figura 13 - Geometria do 1/4 de placa a ser analisado

Considerou-se o referencial xy alinhado com a origem do furo da placa. A placa contém as seguintes propriedades: em que e corresponde à espessura da placa:

- $E = 210 \text{ GPa}$
- $e = 1 \text{ unidade}$
- $\nu = 0,3$

O problema de otimização analisado nesta dissertação tem como objetivo, a minimização da função objetivo que corresponde a área da placa, sujeito a um constrangimento de tensão. Este problema é formulado da seguinte maneira:

$$\begin{aligned}
 & \min_x \quad A(x) \\
 & \text{Sujeito a} \quad \sigma \leq \sigma_{max} \\
 & \quad \quad \quad x_i^l \leq x \leq x_i^u \quad i = 1,2,3
 \end{aligned} \tag{56}$$

3.2. Visão geral do código

O programa *otimizacao_pcf*, começa por inicializar as variáveis necessárias à resolução do problema, nomeadamente os valores iniciais das distâncias radiais da origem do referencial até ao $nó_i$ que define a *spline* da fronteira do furo, ou seja, as variáveis de projeto, bem como, o valor inicial da área da placa, os valores iniciais das tensões nos elementos do contorno do furo e as suas sensibilidades recorrendo ao programa *PROAES_membrana*.

Estas variáveis são utilizadas pelo algoritmo de otimização *mmsub*, que retorna ao utilizador um vetor com as variáveis de projeto otimizadas. Este vetor volta a ser utilizado pelo *PROAES_membrana* de maneira a obter novos valores da função objetivo (área da placa), tensões nos elementos do contorno do furo e as suas sensibilidades.

Esta análise é feita de forma iterativa, sendo que o utilizador pode definir o número máximo de iterações a serem realizadas. Atingido o valor máximo de iterações, o programa retorna ao utilizador o número máximo de iterações, o valor otimizado da função objetivo, os valores das tensões nos elementos situados no contorno do furo, bem como, o valor otimizado das variáveis de projeto. É importante referir que o número de iterações, definido pelo utilizador, tem de ser suficientemente elevado, para que o programa possa convergir.

3.3. Geração de malha

O programa *PROAES_membrana* recorre as funções *gera_malhada_mapeada_radial* e *malha_quadrilatero*, para executar a geração da malha mapeada do $\frac{1}{4}$ do domínio no exemplo da placa com furo definido por uma *spline*. A função *gera_malhada_mapeada_radial* tem como parâmetros o vetor linha com as variáveis de projeto, utilizado para a definição da *spline* radial, bem como, os números de elementos na direção radial e circunferencial, N_r e N_{teta} respetivamente. Onde N_{teta} é sempre um número par. É importante referir que o vetor das variáveis de projeto, deve conter um número ímpar de elementos, devido às condições da função *spline* disponibilizada pelo OCTAVE.

O programa *gera_malhada_mapeada_radial* funciona em duas partes. A primeira parte consiste em delimitar a fronteira do $\frac{1}{4}$ de placa a malhar. Começa por definir as dimensões da área do $\frac{1}{4}$ de placa a malhar e guarda os valores da distância radial do primeiro e último nó que definem a *spline* do furo. Seguidamente são calculadas as coordenadas x, y dos nós da *spline* que definem o furo e são definidas as restantes fronteiras da placa através das coordenadas dos vértices da mesma.

A segunda parte começa por fazer a divisão da área em dois quadriláteros de 8 nós, através da introdução de um segmento de reta definido pelo nó correspondente a metade da *spline* e o vértice do canto superior direito da placa e tendo em conta as coordenadas da fronteira delimitada anteriormente. Calculam-se as coordenadas x, y de cada nó e definem-se as conectividades dos elementos pelo programa *malha_quadrilatero*. Pretende-se obter uma malha constituída por elementos finitos de 4 nós. Para isso o programa *malha_quadrilatero* utiliza a relação estabelecida na equação (8) fazendo o uso das funções de forma do elemento de 8 nós da família *serendipity* para

realizar o cálculo das coordenadas x, y de cada nó e realiza a conectividade entre 4 nós para definir um elemento. O programa *malha_quadrilatero* corresponde então à transformação de coordenadas entre o referencial natural do elemento s, t e o referencial x_1x_2 resultante da discretização do domínio que foi descrita anteriormente. Tem como comandos uma matriz 8×2 das coordenadas x, y dos 8 nós que definem a área a malhar, o número de nós na direção s , o número de nós na direção t e o número do 1º nó a ser considerado. A figura 13 mostra um exemplo de malha gerada pelo programa com $Nr = 30$ e $Nteta = 20$.

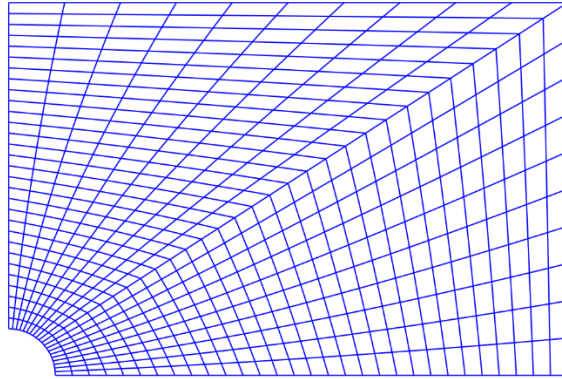


Figura 14 - Exemplo de malha gerada pelo programa em OCTAVE, com $Nr=30$ e $Nteta=20$.

Calculadas as coordenadas dos nós e definida a topologia dos elementos, o programa retorna um gráfico da malha gerada e por fim cria um ficheiro APDL com a informação dos nós e dos elementos. Este ficheiro é posteriormente utilizado para a realização da análise no ANSYS para fim de comparação e validação dos resultados obtidos pelo programa *PROEAS_membrana*.

3.4. Algoritmo de otimização

Um algoritmo corresponde a um processo que envolve um número finito de instruções e tem como objetivo a resolução de uma classe de problemas. Geralmente, os algoritmos de otimização iniciam-se com uma solução inicial, que representa uma estimativa inicial do ponto ótimo e por meio de um processo iterativo, até que as condições de otimalidade sejam satisfeitas, as soluções subsequentes vão sendo melhoradas. Tendo em conta que os problemas de otimização, podem apresentar um número elevado de variáveis de projeto e constrangimentos, que a função objetivo de um problema e as respetivas funções dos constrangimentos podem ser funções não lineares, como também podem ser funções implícitas das variáveis de projeto, torna-se conveniente desenvolver métodos numéricos para resolverem problemas de otimização. Todos os algoritmos de otimização conduzem a um processo iterativo que realiza uma procura organizada de ponto para ponto no espaço de projeto.

3.4.1. MMA

Na presente dissertação foi utilizado o método das assíntotas móveis (MMA) para resolver problemas de programação não linear constrangidos. A utilização deste algoritmo é feita através do código disponibilizado por Svanberg [12]. Este é formulado a seguinte maneira:

$$\begin{aligned}
 \min_x \quad & f_0(x) + a_0 z + \sum_{i=1}^m \left(c_i y_i + \frac{1}{2} d_i y_i^2 \right) \\
 \text{Sujeito a} \quad & f_i(x) - a_i z - y_i \leq 0, \quad i = 1, \dots, m \\
 & x_j^{\min} \leq x \leq x_j^{\max}, \quad j = 1, \dots, n \\
 & y_i \geq 0, \quad i = 1, \dots, m \\
 & z \geq 0.
 \end{aligned} \tag{57}$$

onde x_1, \dots, x_n correspondem às variáveis de projeto “reais”, enquanto y_1, \dots, y_m e z correspondem às variáveis “artificiais” de otimização. f_0, f_1, \dots, f_m correspondem a funções contínuas e diferenciáveis. x_j^{\min} e x_j^{\max} são valores reais que satisfazem a condição $x_j^{\min} < x_j^{\max}$. a_0 e a_i são valores reais que satisfazem as condições $a_0 > 0$ e $a_i \geq 0$. c_i e d_i são valores reais que satisfazem as condições $c_i \geq 0$, $d_i \geq 0$ e $c_i + d_i > 0$.

3.4.2. Avaliação do erro

Para avaliar a qualidade dos resultados obtidos através dos algoritmos de otimização, é necessário quantificar a proximidade da distribuição de tensões de von-Mises à condição de otimalidade, ou seja, *Equi-Stress Principle* (ESP), dos elementos da fronteira da inclusão. Na presente dissertação é usado o erro relativo, RE, para avaliar a qualidade dos resultados obtidos.

$$RE[\%] = \left| \frac{\sigma_{max} - \sigma_{min}}{\sigma_{max} + \sigma_{min}} \right| \times 100 \tag{58}$$

onde σ_{max} e σ_{min} correspondem à tensão máxima e mínima de von-Mises ao longo da fronteira da inclusão, respetivamente. Quando o RE se aproxima de zero a distribuição de tensão de von-Mises torna-se uniforme ao longo da fronteira, satisfazendo-se o ESP.

3.5. Validação e análise de resultados

De maneira a se poder verificar o bom funcionamento do programa desenvolvido, procede-se à verificação dos resultados do código *PROAES_membrana* recorrendo ao software de análise de elementos finitos ANSYS. Utilizando o exemplo da placa com furo, pretende-se verificar primeiro os valores das tensões calculados através da implementação numérica, seguido da verificação das sensibilidades.

A verificação das tensões consistiu em realizar uma análise a 4 malhas distintas para um *xval* fixo, vector que contém as variáveis de projeto no problema de otimização formulado em (56). Neste caso, *xval* corresponde ao estado inicial da placa em que o raio do furo é de 1 unidade. Recorre-se aos dois meios de obtenção de resultados para a verificação das tensões ($xval = [1, 1, 1, 1, 1]$). As malhas em questão diferem na quantidade de elementos, que foram incrementados a cada análise. A análise no ANSYS foi feita recriando as condições e todos os pressupostos utilizados na implementação numérica, nomeadamente o modo de aplicação das forças nos bordos da placa. Considerou-se que nos vértices dos bordos da placa estariam apenas aplicadas forças correspondentes a metade da força total aplicada nos nós entre os vértices. Isto para evitar valores de tensão demasiado elevados nos vértices da placa. As figuras 14 e 15 representam respetivamente, um exemplo da formulação do problema recorrendo ao ANSYS e os resultados das tensões obtidos através da análise de uma das malhas.

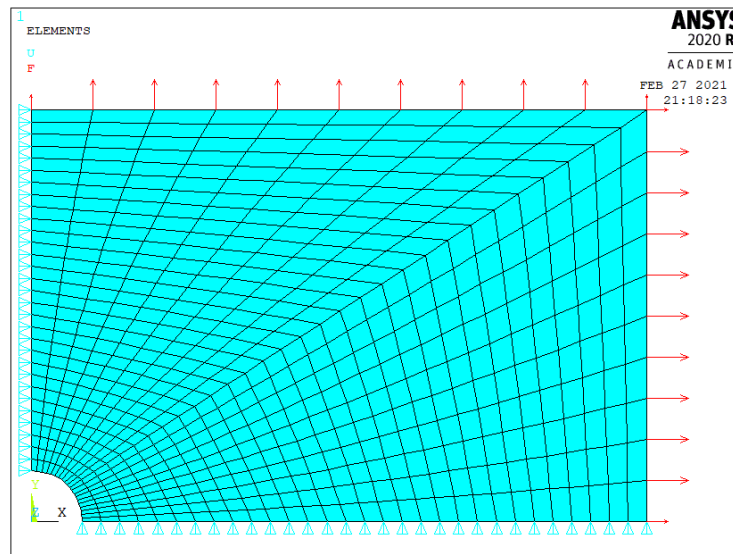


Figura 15 - Formulação do problema no ANSYS para malha (30,20).

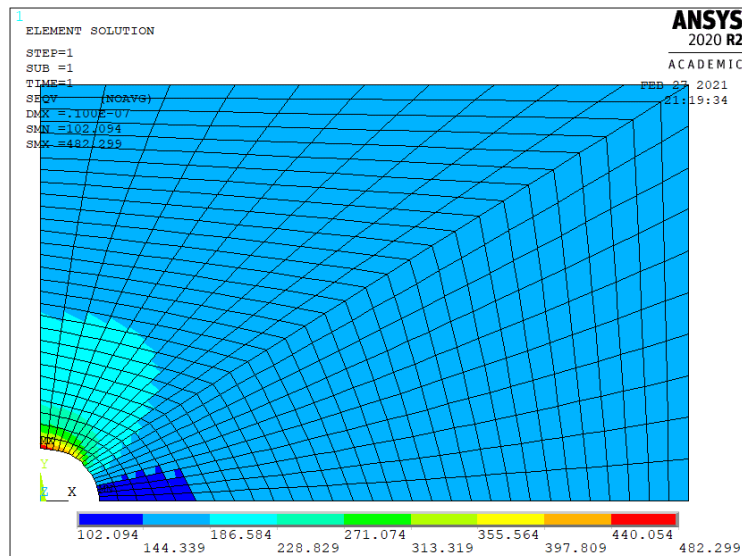


Figura 16 - Resultados das tensões obtidos através da análise da malha (30,20)

Para fazer a comparação dos resultados, teve-se em conta apenas o 1º elemento, relativo ao elemento no bordo do furo que se encontra alinhado com o eixo x , e o último elemento, que se encontra no bordo do furo alinhado ao eixo y .

Os resultados das análises para o 1º e último elemento, podem ser consultados na tabela 2 e 3 respetivamente, bem como em formato gráfico. Os gráficos 1 e 2, representam a evolução do valor da tensão relativamente à dimensão da malha, resultante da análise feita em OCTAVE e ANSYS, do 1º e do último elemento, respetivamente. Neste exemplo consideram-se as seguintes dimensões da malha indicadas na tabela 1.

Tabela 1: Dimensões das malhas consideradas para a análise.

Malha	($Nr, Nteta$)
1	(30,10)
2	(30,20)
3	(30,40)
4	(30,60)

Tabela 2: Resultados da análise de tensões para o 1º elemento.

Malha	OCTAVE	ANSYS	erro (%)
1	114,24	113,98	0,2319
2	112,48	112,19	0,2600
3	112,02	111,72	0,2678
4	111,93	111,63	0,2657

Tabela 3: Resultados da análise de tensões para o último elemento.

Malha	OCTAVE	ANSYS	erro (%)
1	397,59	401,72	-1,0381
2	403,06	406,13	-0,7617
3	406,12	408,96	-0,6999
4	407,21	410,04	-0,6937

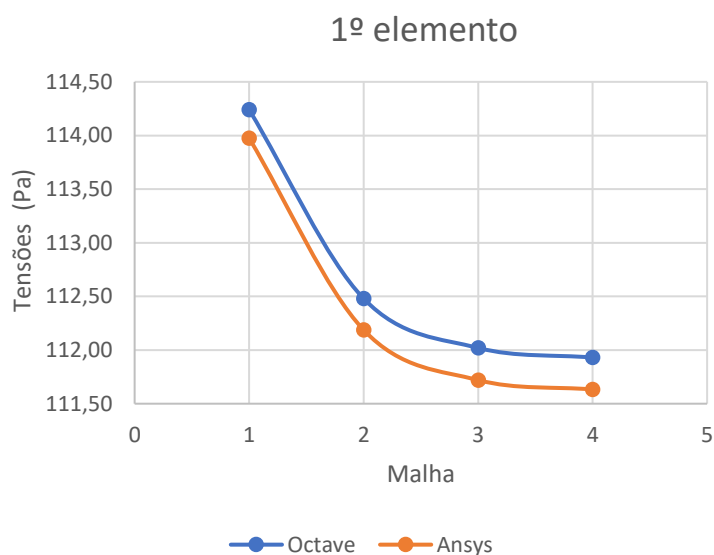


Figura 17 - Resultados das tensões relativamente à dimensão da malha para o 1º elemento.

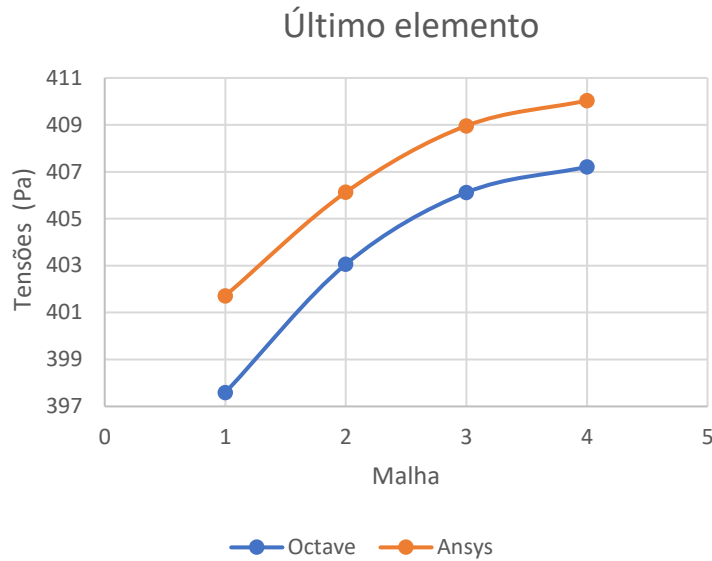


Figura 18 - Resultados das tensões relativamente à dimensão da malha para o último elemento.

Pode-se logo reparar que o aumento da dimensão da malha (refinamento da malha) não conduz a uma convergência do valor da tensão, tanto na análise feita em OCTAVE como na análise feita no ANSYS. No entanto, nota-se que as curvas apresentam o mesmo “percurso”, o que sugere que a margem de erro dos resultados entre as duas análises é constante.

No entanto, é importante referir que devido ao fenómeno de retenção ao corte (*shear locking*) que acontece neste tipo de elementos quando sujeitos a um estado de deformação associado à flexão pura, os valores de deslocamentos calculados podem ser inferiores aos reais, devido a uma rigidez excessiva resultante deste fenómeno. Para contornar este problema, podem usar-se funções de forma quadráticas ou integração reduzida juntamente com um método para controlar os modos de ampulheta. Outra alternativa consiste em usar integração completa e incluir funções de forma adicionais que permitam ao elemento assumir a posição deformada característica da flexão pura, onde não existem tensões e deformações ao corte [3].

No ANSYS a opção de integração completa (opção por omissão) utiliza na verdade uma integração reduzida seletiva, em que as extensões são integradas com 2 pontos de Gauss e as distorções com apenas um ponto [3]. Isto leva a que os resultados entre as duas análises não sejam comparáveis, uma vez que o código utiliza a integração completa de forma diferente relativamente ao ANSYS. Contudo, os resultados apresentam erros relativos muito baixos o que sugere que o problema da integração não seja crítico neste exemplo.

Para verificar as sensibilidades, recorreu-se ao método das diferenças finitas progressivas. Utilizou-se este método por apresentar a vantagem de ser genérico, ou seja, aplicável a todo o tipo de performances e variáveis. Este permite calcular uma aproximação da derivada perturbando a variável X , por adição de um valor ΔX e calculando novamente a performance ψ , como representado pela equação (59) e pela figura 17.

$$\frac{d\psi}{dX} \approx \frac{\psi(X + \Delta X) - \psi(X)}{\Delta X}$$

(59)

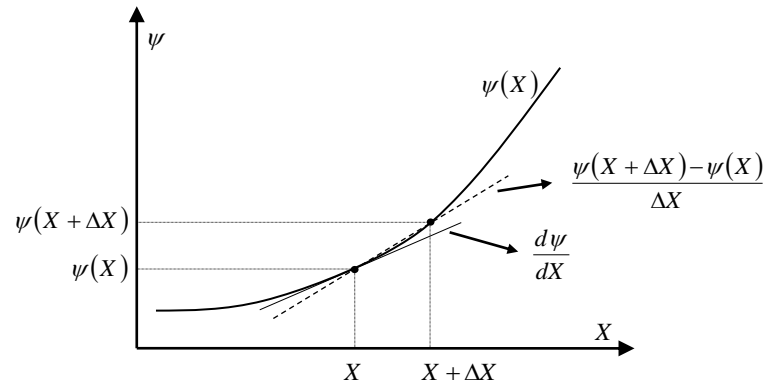


Figura 19 - Diferenças finitas progressivas

Considerando uma perturbação de $\Delta X = 0,001$, para o mesmo *xval* e para a malha mais refinada (neste caso $Nr = 30$ e $Nteta = 60$), procede-se a perturbação de cada valor de *xval* e são registadas as tensões relativas a cada perturbação. Utilizando o método das diferenças finitas, são obtidos os seguintes resultados que se encontram listados na tabela seguinte. Aqui refere-se à 1ª perturbação como sendo a perturbação aplicada ao primeiro elemento do vetor *xval* e assim consecutivamente.

Tabela 4: Demonstração da validade do cálculo das sensibilidades das tensões de von Mises; A coluna "Diferenças finitas" indica os valores calculados pela equação (59); A coluna "df/dx" indica os valores calculados pelo método contínuo, através do programa desenvolvido.

Perturbação	Elemento	sem perturbação	com perturbação	Diferenças Finitas	df/dx	erro (%)
1	1°	111,9268	112,2073	280,5369	280,4741	-0,02237
	último	407,2085	407,1886	-19,86649	-19,8543	-0,06116
2	1°	111,9268	111,7889	-137,8604	-137,547	-0,22794
	último	407,2085	407,3315	123,0009	123,1371	0,110634
3	1°	111,9268	111,8534	-73,38566	-73,1525	-0,31872
	último	407,2085	406,6306	-577,8674	-577,475	-0,06788
4	1°	111,9268	111,8576	-69,19369	-69,0954	-0,14223
	último	407,2085	407,303	94,56781	94,91648	0,36735
5	1°	111,9268	111,9246	-2,128773	-2,1149	-0,65584
	último	407,2085	407,6847	476,1809	476,2776	0,020308

Como se pode verificar na tabela 4, os erros entre os valores das derivadas calculado pelo algoritmo através das sensibilidades analíticas e os valores das derivadas calculadas pelas diferenças finitas, são muito pequenos. O que valida os cálculos realizados pelo mesmo, podendo-se prosseguir para a otimização.

3.6. Estudo de convergência de malha

Com o objetivo de selecionar a malha a ser utilizada para a realização da otimização, recorreu-se a um estudo de convergência de malha. Este constituiu em aumentar de forma proporcional, o número de elementos da malha e analisar os valores de tensão obtidos no primeiro e último elemento, comparando-os com os valores obtidos no ANSYS. Os gráficos 3 e 4 representam os valores de tensão obtidos para cada malha considerada de forma incremental, relativos ao primeiro e último elemento da fronteira do furo. Foram consideradas de forma incremental, as malhas (16,10), (30,20), (46,30), (60,40), (76,50), (90,60), (106,70), (120,80) e (136,90).

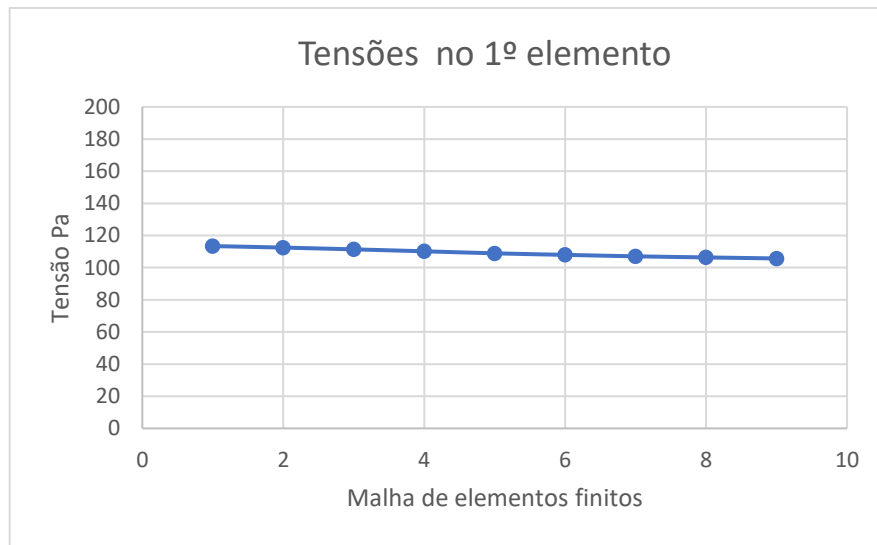


Figura 20 - Teste de convergência de malha para o 1º elemento da fronteira da inclusão.

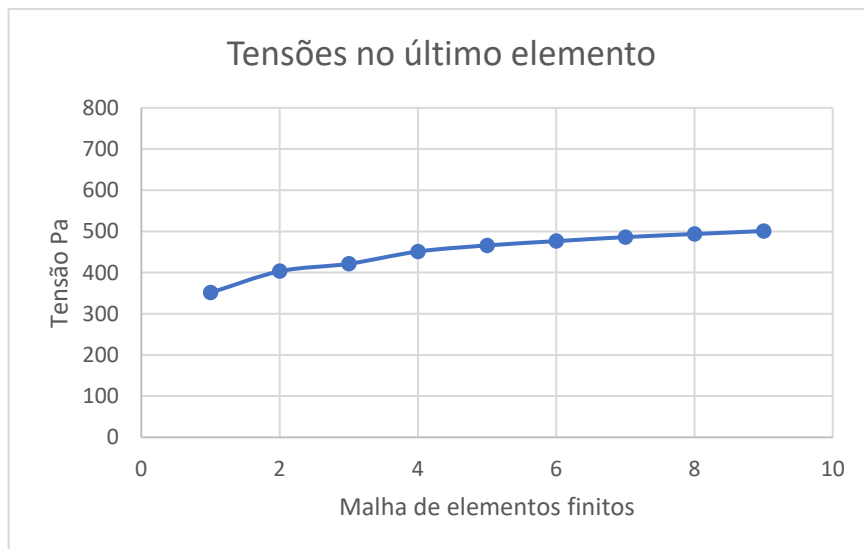


Figura 21 - Teste de convergência de malha para o último elemento da fronteira da inclusão.

Através da análise dos gráficos verifica-se que para o último elemento, o valor da tensão não chega a convergir, contrariamente ao que acontece para o primeiro elemento da fronteira do furo. Isto porque este corresponde a um ponto de concentração de tensões.

Desta análise podemos concluir que a malha mais indicada para proceder à otimização corresponde a malha (60,40). No entanto, com o objetivo de minimizar o tempo de processamento do algoritmo de otimização, recorreu-se a malha (30,20) para realizar a otimização do caso em que a placa se encontra sob carregamento biaxial ($\sigma_x = \sigma_y$). Para o caso mais crítico do carregamento biaxial ($\sigma_x = 2\sigma_y$) optou-se por utilizar a malha (60,40). Posteriormente, aquando da aplicação das condições de fronteira de periodicidade, optou-se por uma malha mais grosseira de (6,8) de maneira a reduzir o tempo de processamento dos algoritmos.

4. Parametrização da fronteira

Na presente dissertação pretende-se desenvolver um algoritmo que otimize a forma da fronteira de inclusões que constituem a mais pequena heterogeneidade de um material celular. Para isso, consideram-se dois métodos distintos para descrever a curva da inclusão no plano. Primeiro considera-se a parametrização da fronteira da inclusão através de uma *spline* de 5 nós. Esta consiste em uma função seccionalmente polinomial que, dados os seus parâmetros, interpola a função que define a fronteira do furo da placa. Embora os resultados das tensões obtidos no capítulo anterior sejam satisfatórios, comparativamente aos resultados obtidos no ANSYS, esta formulação não é a mais interessante, por se tratar da aproximação de uma função.

A segunda parametrização considera que a inclusão é modelada através da equação da superelipse. Esta é uma equação matemática que engloba as primitivas geométricas, e é definida por apenas três parâmetros. A utilização desta parametrização para descrever a curva de inclusões já foi feita por alguns autores como se pode ver em [6], [8]. Verifica-se na solução ótima, que o rácio dos dois raios da forma ótima, equivale ao rácio das tensões aplicadas nas extremidades da placa ($a/b = \sigma_1/\sigma_2$) para σ_1 e σ_2 com o mesmo sinal [6].

Para a realização da otimização, procede-se à parametrização da fronteira do furo utilizando a equação da superelipse. São comparados os resultados da otimização considerando as duas formas de parametrização, bem como, validados os resultados da mesma utilizando a equação da superelipse. Para este último objetivo, foram consideradas as soluções do problema de minimização da *compliance*, obtidos em [1]. É importante referir que as formulações e métodos utilizados, para resolver os problemas de otimização aqui considerados são diferentes. Enquanto nesta dissertação o problema de otimização procura minimizar a área da placa tendo em conta um constrangimento de tensão, o problema de otimização abordado em [1], procura minimizar a *compliance* considerando um constrangimento de fração volúmica. Embora as formulações sejam diferentes, adotando na formulação do problema de otimização desta dissertação, como tensão admissível a tensão teórica calculada por $\sigma_{max}^{teo} = Tr\{\sigma_0\}/V$, obtém-se um valor otimizado de área da placa muito próximo do valor de fração volúmica, considerada na formulação do problema de otimização considerado em [1]. Tornando os resultados perfeitamente comparáveis.

4.1. Superelipse

Também conhecida por curva de Lamé, esta equação foi introduzida por Gabriel Lamé em 1818 [9]. Consiste na descrição de todas as formas geométricas incluídas nas primitivas geométricas (círculos, elipse, quadrados e retângulos). Em coordenadas cartesianas a equação da superelipse é definida por:

$$\left(\frac{x}{a}\right)^\eta + \left(\frac{y}{b}\right)^\eta = 1 \quad (60)$$

onde a , b e η são números reais positivos, em que a e b representam os raios maior e menor da superelipse e η controla a curvatura da forma da mesma. Utilizando a equação da superelipse para descrever da fronteira do furo da placa, esta passa a ser controlada apenas por três variáveis de projeto (a , b e η). A figura mostra a parametrização utilizada para descrever um furo em uma placa através da equação da superelipse.

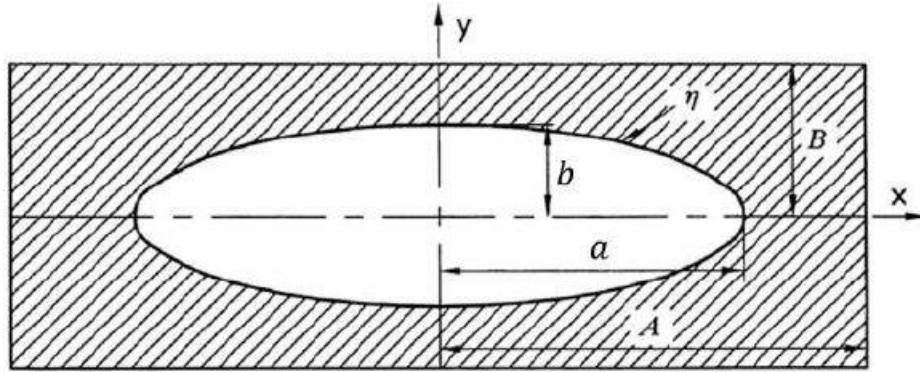


Figura 22 - Parametrização da forma de uma inclusão, utilizando três variáveis de projeto (a , b , η), extraído de [1]

De maneira a simplificar a implementação numérica da parametrização da fronteira, recorre-se à equação fundamental da trigonometria e comparando com a equação (60) obtém-se:

$$\begin{aligned} \left(\frac{x}{a}\right)^\eta &= \sin^2(\theta) \Rightarrow x = a \cdot \sin^{\frac{2}{\eta}}(\theta) \\ \left(\frac{y}{b}\right)^\eta &= \cos^2(\theta) \Rightarrow y = b \cdot \cos^{\frac{2}{\eta}}(\theta) \end{aligned} \quad (61)$$

4.2. Resultados e análise

São apresentados nos seguintes gráficos 5 e 6, os valores de tensão obtidos para o problema de otimização utilizando a parametrização com a *spline de 5 nós*, comparativamente à parametrização com a superelipse. Foram considerados dois tipos de carregamentos e registados os valores de tensão na fronteira do furo. Tendo em conta o problema de otimização proposto, que impõe um valor de tensão máximo nos elementos da fronteira do furo, quanto mais uniforme é essa distribuição de valores, melhor é a qualidade dos resultados obtidos.

De maneira a testar e a validar os resultados do problema de otimização que está na equação (51) correspondente à minimização da área com constrangimentos de tensão de von Mises, obtidos através da implementação numérica da superelipse, recorreu-se a comparação dos mesmos com os resultados apresentados em [1]. Tomaram-se como referência os valores teóricos e os valores obtidos pela análise no ANSYS apresentados em [1], e para motivos de comparação foi considerado o mesmo material fictício. Assume-se uma placa de 20x20 mm de um material que apresenta um comportamento linear elástico e isotrópico, com um módulo de elasticidade unitário, $E = 1 \text{ GPa}$ e um coeficiente de poisson de $\nu = 0,3$. Esta comparação foi realizada apenas para as frações volúmicas de 90%, 80% e 70%.

4.2.1. Spline vs Superelipse

4.2.1.1. Carregamento biaxial ($\sigma_x = \sigma_y$)

Nos gráficos 5 e 6, podemos verificar a evolução do valor da tensão nos diferentes elementos da fronteira do furo, tendo em conta as duas parametrizações consideradas. Pode-se notar que a parametrização com a superelipse apresenta valores de tensão mais uniformes comparativamente aos valores obtidos pela parametrização com a *spline* de 5 nós. Estes resultados estão coerentes com aquilo que se esperava, uma vez que se passou a uma análise feita com uma expressão analítica ao invés de uma aproximação da mesma.

4.2.1.2. Carregamento biaxial ($\sigma_x = 2\sigma_y$)

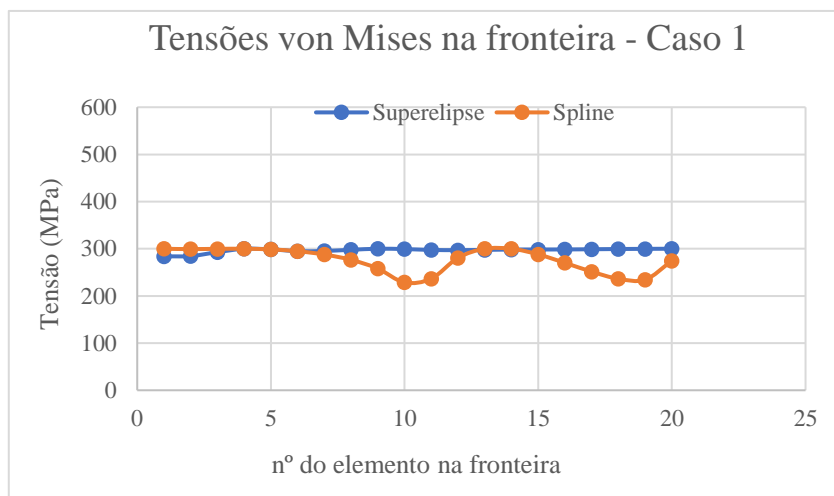


Figura 23 - Resultados da otimização com a parametrização da Superelipse VS parametrização com *Spline* de 5 nós, para carregamento biaxial ($\sigma_x = \sigma_y$)

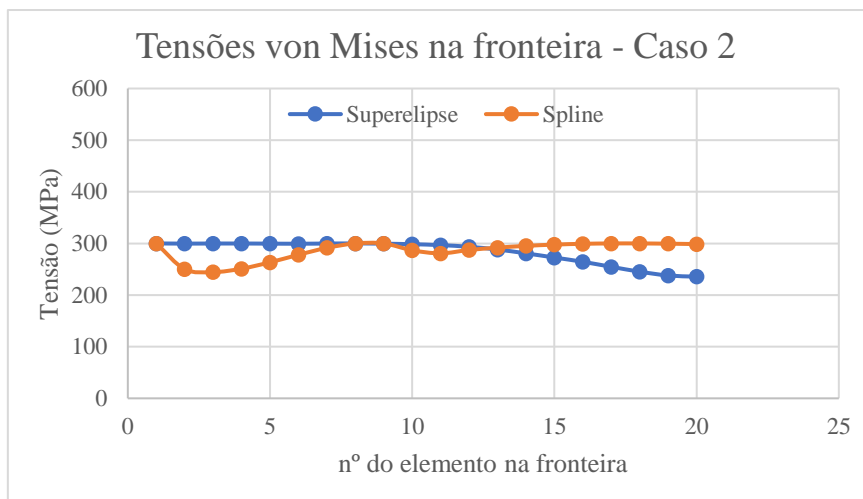


Figura 23 - Resultados da otimização com a parametrização da Superelipse VS parametrização com *Spline* de 5 nós, para carregamento biaxial ($\sigma_x = 2\sigma_y$)

4.2.2. Validação dos resultados obtidos na otimização com a superelipse

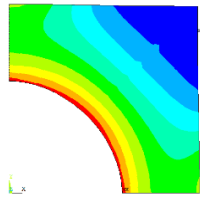
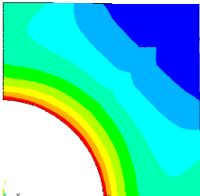
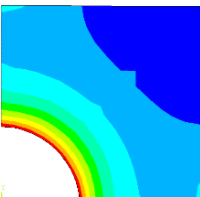
4.2.2.1. Carregamento biaxial ($\sigma_x = \sigma_y$)

Nas tabelas seguintes, apresentam-se os resultados obtidos através da resolução do problema de otimização, utilizando a parametrização com a superelipse, tendo em conta cada fração volúmica considerada para os dois tipos de carregamento. São também apresentados, para motivos de comparação, os valores máximos da tensão de von-Mises calculados através da expressão analítica $\sigma_{max}^{teo} = Tr\{\sigma_0\}/V$ e os valores máximos da tensão máxima de von-Mises obtidos na análise de elementos finitos $\sigma_{max}^{ANSYS DN}$, calculados em [1]. É importante lembrar que, os problemas de otimização aqui considerados não foram formulados da mesma maneira, isto é, o problema de otimização abordado em [1] é formulado tendo em conta constrangimentos de fração volúmica distintos, enquanto na presente dissertação o problema de otimização é formulado tendo em conta um constrangimento de tensão para os elementos da fronteira do furo.

Tabela 5: Solução do problema de minimização da área da placa com ($\sigma_x = \sigma_y$), utilizando a equação da superelipse para definição da fronteira.

V_{comp} [%]	V_{Medido} [%]	σ_{max}^{teo} [MPa]	$\sigma_{max}^{ANSYS DN}$ [MPa]	$\sigma_{max}^{código}$ [MPa]	σ_{max}^{ANSYS} [MPa]	Desvio código [%]	Desvio ANSYS [%]	RE [%]
90	89.51	2.22	2.22	2.219	2.307	-0.045	3.92	0.3540
80	81.694	2.5	2.523	2.499	2.574	-0.040	2.02	0.8098
70	74.81	2.857	2.912	2.85699	2.914	0.0004	-0.07	2.0773

Tabela 6: Distribuição do campo de tensões das soluções do problema de minimização da área da placa com ($\sigma_x = \sigma_y$), utilizando a equação da superelipse para definição da fronteira.

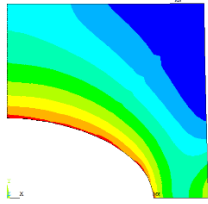
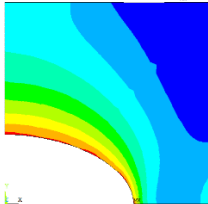
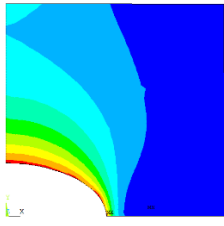
V_{comp} [%]	Campo de tensões
70	
80	
90	

4.2.2.2. Carregamento biaxial ($\sigma_x = 2\sigma_y$)

Tabela 7: Solução do problema de minimização da área da placa com ($\sigma_x = 2\sigma_y$), utilizando a equação da superelipse para definição da fronteira.

V_{comp} [%]	V_{Medido} [%]	σ_{max}^{teo} [MPa]	$\sigma_{max}^{ANSYS DN}$ [MPa]	$\sigma_{max}^{código}$ [MPa]	σ_{max}^{ANSYS} [MPa]	Desvio código [%]	Desvio ANSYS [%]	RE [%]
90	91.78	3.33	3.367	3.32998	3.492	-0.0006	-3.731	4.58
80	84.53	3.75	3.824	3.74998	3.825	-0.0005	-0.026	5.11
70	78.03	4.285	4.426	4.28498	4.382	-0.0005	0.994	6.18

Tabela 8: Distribuição do campo de tensões das soluções do problema de minimização da área da placa com ($\sigma_x = 2\sigma_y$), utilizando a equação da superelipse para definição da fronteira.

V_{comp} [%]	Campo de tensões
70	
80	
90	

Como se pode verificar nas tabelas 5 e 7 os valores de fração volúmica obtidos pelo algoritmo não correspondem aos valores exatos obtidos em [1], devido precisamente à diferença entre as formulações dos problemas. Verifica-se também que para a fração volúmica intermédia de 70% o algoritmo não apresenta resultados muito bons, isto devido a parametrização com a superelipse funcionar melhor para valores de fração volúmica mais elevados. No entanto, o algoritmo obtém bons valores de tensão máxima comparativamente ao valor teórico esperado, com um RE de até 0,8098% para as frações volúmicas mais elevadas e um RE de 12,9778% para a fração volúmica de 70%. Este último pressupõe que se iria beneficiar de mais uma análise considerando uma malha mais refinada e aumentando o número de iterações do algoritmo de otimização. Para o caso do carregamento de tensões macroscópicas não unitárias, verifica-se valores de RE superiores, isto devido a utilização da parametrização pela superelipse ser pouco flexível para estes casos. No entanto o código apresenta valores de tensão máxima muito similares aos valores teóricos esperados. É importante referir que os desvios dos valores das tensões obtidas no ANSYS têm como base o facto de em [1] ser considerada uma matriz de 5x5 células de maneira a simular o material celular periódico.

5. Aplicação das condições de fronteira de periodicidade

Com o objetivo de verificar a influência da periodicidade do material na solução do problema de otimização, são aplicadas as condições de fronteira de periodicidade considerando uma única célula. Desta forma é possível evitar estudar uma matriz de 5 x 5 células, tal como foi realizado em [1]. Contudo, estas são condições fronteira de deslocamento e por isso o problema da otimização deve ser reformulado. Consideram-se estados de extensão uniformes e são aplicadas condições fronteira de periodicidade. As soluções desta nova formulação são novamente comparadas com os resultados obtidos em [1], com o objetivo de avaliar e verificar a eficácia deste método.

Os materiais compósitos são amplamente utilizados em várias áreas tecnológicas, desde a aeronáutica, a indústria automóvel, petroquímica entre outras indústrias, devido às suas propriedades superiores relativamente aos materiais convencionais utilizados em engenharia. Nas últimas décadas, muitos pesquisadores têm dedicado esforços consideráveis para avaliar propriedades macromecânicas de compósitos usando o método de modelagem micro-mecânica [10]. O método de modelagem micro-mecânico, permite prever o comportamento geral dos compósitos de propriedades conhecidas da fase de reforço e da fase da matriz, através de uma análise de um elemento de volume representativo (RVE) ou um modelo de célula unitária. Em compósitos têxteis, a macroestrutura pode ser vista como uma matriz periódica de uma unidade repetida de célula (RUC), assim como, em outros compósitos como compósitos de lâmina unidirecional e compósitos reforçados com partículas, assumindo uma distribuição uniforme e a mesma geometria para a fase de reforço. Portanto, na maioria das análises micro-mecânicas, recorre-se ao RUC ao invés do RVE para os compósitos.

O método dos elementos finitos tem sido vastamente utilizado na literatura para analisar um RUC, de maneira a determinar as propriedades mecânicas e os mecanismos de falha dos compósitos. Os avanços computacionais, têm contribuído para o desenvolvimento desta área, possibilitando a aplicação deste método a vários tipos de materiais compósitos. Assim, tornou-se relativamente simples fazer a aplicação do MEF a RCUs sólidos, independentemente do seu grau de complexidade.

5.1. Condições de fronteira de periodicidade

Considere-se uma estrutura periódica que consiste na repetição periódica de uma célula unitária com mostra a figura 19. O campo de deslocamentos é dado por:

$$u_i(x_1, x_2, x_3) = \bar{\varepsilon}_{ik}x_k + u_i^*(x_1, x_2, x_3) \quad (62)$$

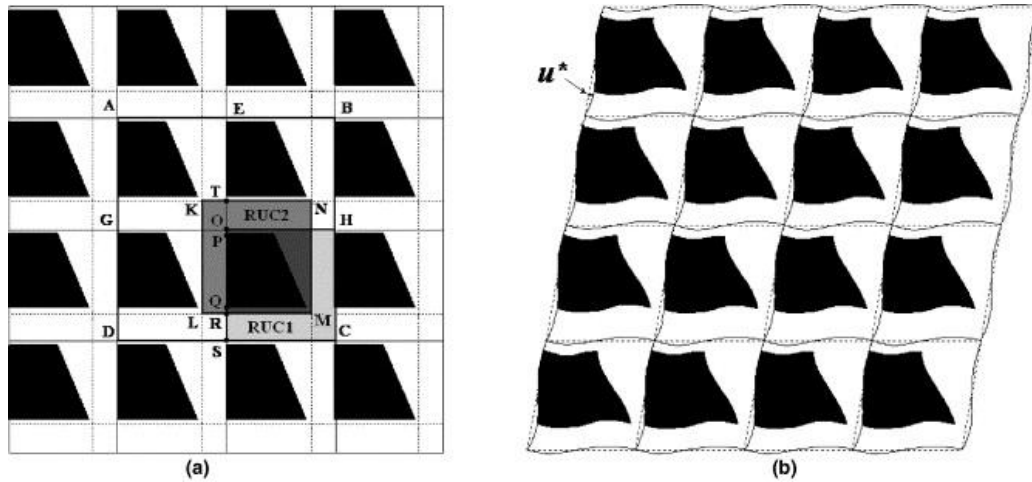


Figura 24 - Representação 2-D de uma estrutura periódica sem eixo ortogonal de simetria: (a) forma não deformada com duas alternativas de RCUs; (b) Deformada sob carga global. Extraído de [10].

onde $\bar{\varepsilon}_{ik}$ representa o tensor das extensões (aproximadas) global da estrutura periódica, u_i representa a distribuição linear do campo de deslocamentos e u_i^* representa a variação do campo de deslocamentos devido a heterogeneidade da estrutura do compósito. Esta expressão não pode ser aplicada diretamente na fronteira pois a parte periódica da equação é geralmente desconhecida. A expressão geral precisa ser transformada em um número explícito de constrangimentos, entre pares de nós correspondentes em faces opostas da fronteira da célula unitária [10]. Assim, o deslocamento entre pares de nós em faces opostas da fronteira da célula unitária, podem ser escritos como:

$$\begin{aligned} u_i^{j+} &= \bar{\varepsilon}_{ik}x_k^{j+} + u_i^* \\ u_i^{j-} &= \bar{\varepsilon}_{ik}x_k^{j-} + u_i^* \end{aligned} \quad (63)$$

onde “j +” e “j -” indicam o par das duas faces opostas paralelas da fronteira do RCU, orientadas perpendicularmente a j-ésima direção. O termo periódico pode ser eliminado através da diferença dos deslocamentos.

$$u_i^{j+} - u_i^{j-} = \bar{\varepsilon}_{ik}(x_k^{j+} - x_k^{j-}) = \bar{\varepsilon}_{ik}\Delta x_k^j \quad (64)$$

Sendo que Δx_k^j é constante para cada par de nós em faces opostas da fronteira da célula unitária, especificando $\bar{\varepsilon}_{ik}$ o lado direito da equação torna-se constante e pode ser aplicado facilmente na análise de elementos finitos como constrangimentos de deslocamentos nos nós. Esta forma de condições de fronteira, garante a solução única quando da análise de um RCU utilizando MEF baseados no deslocamento.

$$u_i^{j+} - u_i^{j-} = w_i^j \quad (65)$$

5.2. Solução numérica da equação

A solução da equação anterior é obtida através da eliminação dos termos desconhecidos. Separando o vetor dos deslocamentos global \mathbf{U} em 4 partes: \bar{U}_1 corresponde aos deslocamentos impostos, U_2 corresponde aos deslocamentos desconhecidos nos nós interiores da célula, U_3 e U_4 correspondem aos deslocamentos desconhecidos dos nós localizados em faces opostas da célula unitária, que satisfazem $U_4 = U_3 + \bar{W}$, onde \bar{W} corresponde ao termo da equação, dado um $\bar{\varepsilon}_{ik}$. A equação de equilíbrio pode ser dada por:

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{bmatrix} \bar{U}_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \quad (66)$$

onde F_1 é um vetor desconhecido e igual as reações nos nós com deslocamentos impostos, $F_2 = 0$, e $F_3 + F_4 = 0$ devido a periodicidade. K é simétrica, logo, eliminando a primeira coluna, adicionando a terceira e quarta linha e utilizando a relação $U_4 = U_3 + \bar{W}$ reduz-se a:

$$\begin{bmatrix} K_{22} & K_{23} + K_{24} \\ sym. & K_{33} + K_{34} + K_{43} + K_{44} \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \end{bmatrix} = - \begin{bmatrix} K_{21} \\ K_{31} + K_{41} \end{bmatrix} \bar{U}_1 - \begin{bmatrix} K_{24} \\ K_{34} + K_{44} \end{bmatrix} \bar{W} \quad (67)$$

permitindo a resolução do sistema de equações.

5.3. Implementação numérica

A implementação computacional das condições de periodicidade, teve como base um programa já existente, *topX.m* extraído de [11].

É importante referir que até agora, o problema de otimização foi resolvido tendo em conta um quarto de célula. Para a aplicação das condições de fronteira de periodicidade, foi necessário passar-se à análise da célula completa. Este constituiu o primeiro passo da implementação computacional destas novas condições. Seguindo a mesma metodologia para a geração da malha, adotada anteriormente, o programa *gera_malha_mapeada_radial* é alterado e passa a gerar uma célula completa com um furo definido pela superelipse.

Uma vez que esta forma de condições de fronteira, garante a solução única aquando da análise de um RCU utilizando MEF baseados no deslocamento, o problema de otimização teve de ser reformulado. Deixam-se de se aplicar tensões na fronteira e passam-se a aplicar deslocamentos. O programa *PROAES_membrana* é modificado tendo em conta estas novas alterações e seguindo a metodologia apresentada em [11]. Com propósito ilustrativo, a célula considerada na figura 20 é discretizada em elementos 3x3. Os graus de liberdade são divididos em quarto partes, tendo em conta (66). Para a resolução de problemas considerando condições de fronteira de periodicidade, é necessário fixar pelo menos um ponto para evitar movimento de corpo rígido [11]. Assim, quando por exemplo, o ponto A é fixo aos pontos B, C e D é imposto um deslocamento Δx , correspondente ao caso de carga pretendido.

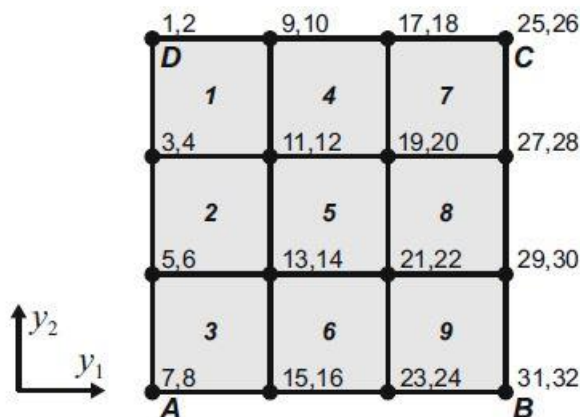


Figura 25 – Exemplo de célula discretizada 3x3, extraído de [11]

Nesta implementação numérica, foram considerados separadamente os casos de carga, contrariamente ao que acontece no algoritmo em que foi baseado. Assim, fazendo variar no código o valor de *j*, entre 1 e 3, obtêm-se os casos de carga de extensão segundo *x*, extensão segundo *y* e de distorção gama, respetivamente. A figura 21 demonstra os exemplos obtidos com esta aplicação.

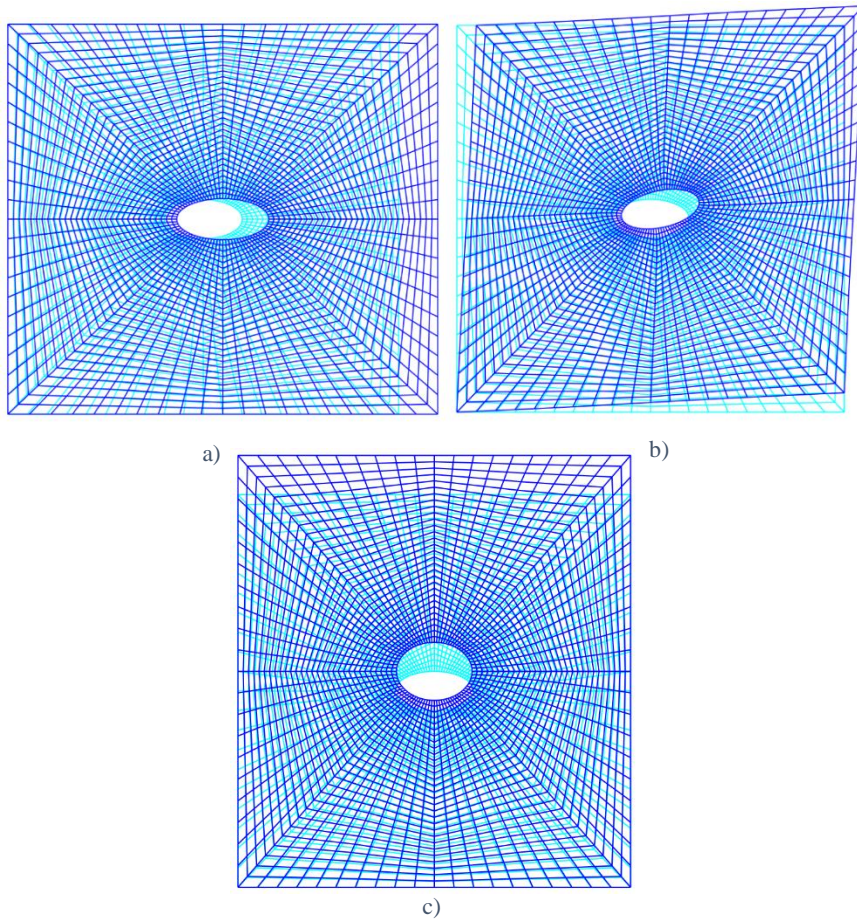


Figura 26 - Exemplo de deformadas da placa com furo para os diferentes casos de carga; a) Extensão em x; b)Distorção; c)Extensão em y

5.4. Validação e resultados finais

Implementadas as condições de fronteira, é necessário verificar a qualidade dos cálculos realizados pelo programa, nomeadamente das sensibilidades analíticas. Seguindo a mesma metodologia abordada anteriormente, comparam-se os valores das derivadas calculadas pelo programa com as diferenças finitas. Devido à existência de simetria na placa, serão apenas analisados novamente os valores na fronteira do $\frac{1}{4}$ de placa. A tabela 10 demonstra a validade das sensibilidades calculadas pelo programa, tendo em conta a implementação das condições de fronteira de periodicidade. Esta análise foi feita considerando uma malha mais grosseira $(Nr, Nteta) = (6,8)$, de maneira a minimizar o tempo de processamento do código, e teve em conta todos os casos de carga. No entanto, na tabela são demonstrados os resultados da análise para o primeiro caso de carga, ou seja, extensão em x. Para este caso de carga, com propósito demonstrativo, considera-se uma extensão de 10% da largura da placa, referente à figura 21.

Tabela 9: Análise do cálculo de sensibilidades das tensões de von Mises, após implementação das condições de fronteira de periodicidade; A coluna "Diferenças finitas" indica os valores calculados pela equação (59); A coluna "df/dx" indica os valores calculados pelo método contínuo, através do programa desenvolvido.

Perturbação	Elemento	sem perturbação	com perturbação	Diferenças Finitas	df/dx	erro (%)
1	1°	1.2523E+10	1.2526E+10	2.4171E+09	2.4173E+09	-0.01%
	último	2.8585E+10	2.8584E+10	-1.2368E+09	-1.2364E+09	0.03%
2	1°	1.2523E+10	1.2515E+10	-8.1610E+09	-8.1633E+09	-0.03%
	último	2.8585E+10	2.8592E+10	7.2305E+09	7.2316E+09	-0.01%
3	1°	1.2523E+10	1.2522E+10	-1.8224E+09	-1.8224E+09	0.00%
	último	2.8585E+10	2.8584E+10	-7.5690E+08	-7.5674E+08	0.02%

Os resultados da tabela verificam a validade dos cálculos das sensibilidades das tensões de von Mises, considerando as condições de fronteira de periodicidade. As sensibilidades analíticas demonstram erros significativamente pequenos em relação às diferenças finitas.

Como mencionado anteriormente, com a implementação das condições de fronteira de periodicidade, é necessário reformular o problema de otimização. Este passa a ter deslocamentos aplicados na fronteira ao invés de tensões. No entanto, de maneira a se poder recriar e comparar com os resultados obtidos em [1], foi necessário ajustar os valores de deslocamentos aplicados de maneira que estes correspondessem aos deslocamentos, segundo os quais, o valor da tensão mínima na fronteira corresponde ao valor teórico dado para cada fração volúmica. A reformulação do problema pode ser escrita na seguinte forma standard.

$$\begin{aligned}
 & \min_x && A(x) \\
 & \text{Sujeito a} && \sigma \geq \sigma_{min} \\
 & && x_i^l \leq x \leq x_i^u \quad i = 1,2,3
 \end{aligned} \tag{68}$$

Os resultados deste novo problema de otimização encontram-se listados na tabela 10. Nesta pode-se encontrar os valores de fração volúmica e os valores mínimos da tensão na fronteira, correspondentes, bem como os valores.

Tabela 10: Resultados da otimização de forma considerando as condições de fronteira de periodicidade para o caso de carga ($\varepsilon_x = \varepsilon_y$)

V_{comp} [%]	V_{Medido} [%]	σ_{max}^{teo}	$\sigma_{min}^{código}$	Desvio Tensões [%]	RE [%]
90	89.535	2.22	2.2234	-0.153	0.5097
80	75.9	2.5	2.5054	-0.216	1.1120
70	59.758	2.857	2.8626	-0.196	2.1454

Tabela 11: Resultados da otimização de forma considerando as condições de fronteira de periodicidade para o caso de carga ($\epsilon_x = 2\epsilon_y$)

V_{comp} [%]	V_{Medido} [%]	σ_{max}^{teo}	$\sigma_{min}^{código}$	Desvio Tensões [%]	RE [%]
90	84.39	3.33	3.333	-0.09	0.7522
80	74.135	3.75	3.771	-0.56	1.8272
70	59.748	4.285	4.291	-0.14	10.6234

Como se pode verificar nas tabelas, os valores de fração volúmica obtidos afastam-se significativamente dos valores esperados. No entanto, o código apresenta bons resultados de tensão na fronteira da inclusão, com RE bastante pequenos para frações volúmicas elevadas, satisfazendo a condição de otimalidade. Pode-se notar também uma discrepância nos dois casos de carga, para a fração volúmica de 70%, com um RE bastante superior relativamente as outras duas frações volúmicas.

6. Conclusões

O principal objetivo da presente dissertação é o desenvolvimento de um algoritmo para a otimização de forma de inclusões em materiais celulares de microestrutura periódica, de modo que estas verifiquem uma determinada condição de otimalidade que pode ser definida pelo *Equi-Stress Principle*.

Para resolver o problema de otimização primeiramente foi considerada uma formulação que utilizava como parametrização da fronteira da inclusão, uma spline de 5 nós. Verificou-se que para esta parametrização da fronteira as soluções da otimização afastam-se das condições de otimalidade. Para resolver este problema passou-se a considerar a equação da superelipse como parametrização da fronteira. Verificou-se que esta nova parametrização apresenta melhores resultados relativamente a anterior. Percebeu-se também que as formas ótimas das inclusões obtidas através desta parametrização afastam-se da condição de otimalidade definida pelo *Equi-Stress Principle* para frações volúmicas de material intermédias, ou seja 70%. Quando se consideram carregamentos constituídos por rácios de tensões macroscópicas não unitários a utilização da superelipse apresenta bons resultados para frações volúmicas elevadas. Posteriormente, reformulou-se o problema de otimização, de maneira a considerar as condições de fronteira de periodicidade. Verificou-se que os resultados obtidos correspondem a aproximações razoáveis tendo em conta as diferenças nas formulações dos problemas.

Bibliografia

- [1] D. R. Negrão, “Otimização de forma de inclusões em microestruturas de material celular utilizando a Superformula”, Dissertação de Mestrado, FCT/UNL, 2018, [Online]. Available: <https://run.unl.pt/handle/10362/56416>.
- [2] Orlando J. B. A. Pereira, “Introdução ao Método dos Elementos Finitos na Análise de Problemas Planos de Elasticidade”, Apontamentos da Disciplina, IST, 2004.
- [3] J. Cardoso, “Métodos Computacionais em Engenharia Mecânica”, Apontamentos da Disciplina, FCT/UNL, 2018.
- [4] V. Haug, E.J., Choi, K.K., Komkov, *Design sensitivity analysis of structural systems*. New York, 1992.
- [5] P. Coelho, “Tópicos avançados em mecânica estrutural”, Apontamentos da Disciplina, FCT/UNL, 2020.
- [6] P. Pedersen, “On optimal shapes in materials and structures,” *Struct. Multidiscip. Optim.*, vol. 19, no. 3, pp. 169–182, 2000, doi: 10.1007/s001580050100.
- [7] Z. Wu, “Optimal hole shape for minimum stress concentration using parameterized geometry models,” *Struct. Multidiscip. Optim.*, vol. 37, no. 6, pp. 625–634, Feb. 2009, doi: 10.1007/s00158-008-0253-4.
- [8] G. I. N. Rozvany, M. Zhou, and T. Birker, “Generalized shape optimization without homogenization,” *Struct. Optim.*, vol. 4, no. 3–4, pp. 250–252, 1992, doi: 10.1007/bf01742754.
- [9] J. Gielis, "The Geometrical Beauty of Plants", Atlantis Press Paris, 2017.
- [10] Z. Xia, C. Zhou, Q. Yong, and X. Wang, “On selection of repeated unit cell model and application of unified periodic boundary conditions in micro-mechanical analysis of composites,” *Int. J. Solids Struct.*, vol. 43, no. 2, pp. 266–278, 2006, doi: 10.1016/j.ijsolstr.2005.03.055.
- [11] L. Xia and P. Breitkopf, “Design of materials using topology optimization and energy-based homogenization approach in Matlab,” *Struct. Multidiscip. Optim.*, vol. 52, no. 6, pp. 1229–1241, 2015, doi: 10.1007/s00158-015-1294-0.
- [12] K. Svanberg, "MMA and GCMMA, versions September 2007," [Online]. Available: <http://www.math.kth.se/~krille/gcmma07.pdf>.

Anexo

Programa PROAES_membrana_condicoes_periodicidade.m

```
function [f0val,df0dx,df0dx2,fval,dfdx,dfdx2] = PROAES_membrana_condicoes_periodicidade(xval)

%
% Otimizacao de uma placa com furo, minimização da área
% com constrangimentos de tensão (versao para placa completa e
% condicoes de periodicidade)
%
GRAFICO_INI= 1;
GRAFICO_DEF= 0;
SENSIBILIDADES= 1;
DELTA_V= 0.001;
SIGMA_ADM= 2.857e6;
temp= size(xval);
nvar= temp(1);
%
Nr= 6; % Numero de elementos na direcção radial (6/30)
Nteta= 8; % Numero elementos na direcção circunferencial (tem de ser divisivel por 4) (8/20)
df0dx= zeros(nvar,1);
df0dx2= zeros(nvar,1);
fval= zeros(Nteta,1); % O constrangimento é g(x) <= 0
dfdx= zeros(Nteta,nvar);
dfdx2= zeros(Nteta,nvar);
%
if nvar == 3
%
% Chama a funcao gera_malha_mapeada_radial para obter os dados do problema
% [X,Y,elementos]= gera_malha_mapeada_radial([1 0.9 0.77 0.65 0.6],30,20);
%
[X,Y,conetividades]= gera_malha_mapeada_sup_elipse_placa_completa(xval',Nr,Nteta);
if GRAFICO_INI
[m,n]=size(X);
XX=zeros(4*Nteta,Nr+1);
XX(1:4*Nteta,:)= X;
XX(4*Nteta+1,:)= X(1,:);
```

```

YY=zeros(4*Nteta,Nr+1);

YY(1:4*Nteta,:)= Y;

YY(4*Nteta+1,:)= Y(1,:);

% Plot grid

axis off

box on

hold on

% Plot internal grid lines

if GRAFICO_DEF

    for i=1:4*Nteta+1

        plot(XX(i,:),YY(i,:), 'c', 'linewidth',1);

    end

    for j=1:Nr+1

        plot(XX(:,j),YY(:,j), 'c', 'linewidth',1);

    end

else

    for i=1:4*Nteta+1

        plot(XX(i,:),YY(i,:), 'b', 'linewidth',1);

    end

    for j=1:Nr+1

        plot(XX(:,j),YY(:,j), 'b', 'linewidth',1);

    end

end

axis equal

M= getframe(gcf);

end

%

% Define os dados do problema

nnos= (Nr+1)*(4*Nteta);

nos= zeros(2,nnos);

ino= 1;

for i= 1:4*Nteta

    for j= 1:Nr+1

        nos(1,ino)= X(i,j);

        nos(2,ino)= Y(i,j);

        ino= ino + 1;

    end

end

```



```

%
nelementos= Nr*4*Nteta;
elementos= zeros(5,nelementos);
for i= 1:nelementos
    elementos(1:4,i)= conectividades(i,1:4);
    elementos(5,i)= 1; % Todos os elementos sao do material 1
end
%
n materiais= 1;
materiais= [1e9; 0.3]; % modulo young e coef Poisson
%
% Definicao das Nteta (Tensoes) + 1 (Area) performances
%
performance_tensao= zeros(Nteta,1);
for i= 1:Nteta
    performance_tensao(i)= 1+(i-1)*Nr; % Número do elemento onde
end % é calculada a tensao
else
    fprintf('*** xval deve conter 5 valores\n');
end
%
% Inicializa as matrizes e vectores necessarios para o calculo
%
K= zeros(nnos*2);
x= zeros(nnos*2,1);
f= zeros(nnos*2,1);
% Guarda a área dos elementos
area= zeros(nelementos,1);
%
% Inicializa os vectores contendo as coordenadas
% e os pesos dos pontos de Gauss
%
gst= [ -0.577350269189626 +0.57735026918926 ];
gw= [ +1 , +1 ];
%
% Calcula a matriz de rigidez de cada elemento e
% adiciona-a a matriz global
%

```

```

for i=1:nelementos
    % inicializa a matriz de rigidez do elemento
    k= zeros(8);
    % define o material do elemento
    mat= elementos(5,i);
    % xye guarda as coordenadas x,y dos 4 nós
    xye= zeros(4,2);
    for j= 1:4
        ino= elementos(j,i);
        xye(j,1)= nos(1,ino);
        xye(j,2)= nos(2,ino);
    end
    % cálculo de matriz de constantes elásticas
    [D]= calcula_d(materiais(1,mat),materiais(2,mat));
    %
    % integração numérica
    %
    for is= 1:2 % 2 pontos de Gauss na direcção s
        s= gst(is);
        ws= gw(is);
        for it= 1:2 % 2 pontos de Gauss na direcção t
            t= gst(it);
            wt= gw(it);
            % cálculo das derivadas das funções de forma
            % em ordem a x,y e do determinante da transformação
            % de coordenadas no ponto de gauss
            [gdn,det]= grad4(xye, s, t);
            % cálculo da matriz B
            B= zeros(3,8);
            for ino=1:4
                B(1,ino*2-1)= gdn(ino,1);
                B(2,ino*2)= gdn(ino,2);
                B(3,ino*2-1)= gdn(ino,2);
                B(3,ino*2)= gdn(ino,1);
            end
            k= k + B' * D * B * det * ws * wt ;
            area(i)= area(i) + det * ws * wt ;
        end
    end
end

```

```

end

%
% assemblagem

for j= 1:4

    for m= 1:4

        ino= elementos(j,i);

        jno= elementos(m,i);

        uino= ino*2-1;

        vino= ino*2;

        ujno= jno*2-1;

        vjno= jno*2;

        K(uino:vino,ujno:vjno)= K(uino:vino,ujno:vjno) + ...

            k(j*2-1:j*2,m*2-1:m*2);

    end

end

%

end % for i=1:nelementos

%

%% PERIODIC BOUNDARY CONDITIONS

%

e0 = eye(3);

%e0(1,1) = 2;

e0(2,1) = 1;

ufixed = zeros(8,1);

alldofs = (1:2*(Nteta*4)*(Nr+1));

fprintf('alldofs= %d\n',alldofs);

%

n1 = [(Nr+1)*(5*Nteta/2+1) (Nr+1)*(7*Nteta/2+1) (Nr+1)*(Nteta/2+1) (Nr+1)*(3*Nteta/2+1) ];

d1 = reshape([(2*n1-1);2*n1],1,8);

%

n3 = [(Nr+1)*(3*Nteta/2+2):(Nr+1):(Nr+1)*(5*Nteta/2) ...

        (Nr+1)*(5*Nteta/2+2):(Nr+1):(Nr+1)*(7*Nteta/2)];

d3 = reshape([(2*n3-1);2*n3],1,2*(2*Nteta-2));

%

n4 = [(Nr+1)*(1*Nteta/2):- (Nr+1):(Nr+1) ...

        (Nr+1)*(8*Nteta/2):- (Nr+1):(Nr+1)*(7*Nteta/2+2) ...

        (Nr+1)*(3*Nteta/2):- (Nr+1):(Nr+1)*(1*Nteta/2+2)];

d4 = reshape([(2*n4-1);2*n4],1,2*(2*Nteta-2));

```

```

%
d2 = setdiff(alldofs, [d1,d3,d4]);
%
% Escolher o caso de carga, j=1 -> Epsilon_xx, j=2 -> Epsilon_yy, j=3 -> Gama_xy
%
delta_x= 0.00027; % Deslocamento imposto segundo x
delta_y= delta_x; % Deslocamento imposto segundo y
j=1;
ufixed(3:4) = [e0(1,j),e0(3,j)/2;e0(3,j)/2,e0(2,j)]*[delta_x;0];
ufixed(7:8) = [e0(1,j),e0(3,j)/2;e0(3,j)/2,e0(2,j)]*[0;delta_y];
ufixed(5:6) = ufixed(3:4)+ufixed(7:8);
%
wfixed = [repmat(ufixed(3:4),Nteta-1,1); repmat(ufixed(7:8),Nteta-1,1)];
%
%% FE-ANALYSIS
%
Kr = [K(d2,d2), K(d2,d3)+K(d2,d4); K(d3,d2)+K(d4,d2), K(d3,d3)+K(d4,d3)+K(d3,d4)+K(d4,d4)];
x(d1) = ufixed;
x([d2,d3]) = Kr\(-[K(d2,d1); K(d3,d1)+K(d4,d1)]*ufixed-[K(d2,d4); K(d3,d4)+K(d4,d4)]*wfixed);
x(d4) = x(d3)+wfixed;
%
% Grafico da posicao deformada
%
X_def=zeros(4*Nteta,Nr+1);
X_def(1:4*Nteta,:) = X;
X_def(4*Nteta+1,:) = X(1,:);
Y_def=zeros(4*Nteta,Nr+1);
Y_def(1:4*Nteta,:) = Y;
Y_def(4*Nteta+1,:) = Y(1,:);
%
m_nos= transpose(reshape(1:(Nr+1)*(4*Nteta),Nr+1,4*Nteta));
for i=1: 4*Nteta
    for j=1:Nr+1
        No_n= m_nos(i,j);
        ux= x(2*No_n-1);
        uy= x(2*No_n);
        X_def(i,j)= X_def(i,j)+ux;
        Y_def(i,j)= Y_def(i,j)+uy;
    end
end

```

```

    end
end
for j=1:Nr+1
    No_n= m_nos(1,j);
    ux= x(2*No_n-1);
    uy= x(2*No_n);
    X_def(4*Nteta+1,j)= X_def(1,j);
    Y_def(4*Nteta+1,j)= Y_def(1,j);
end
%
if GRAFICO_DEF
    [m,n]=size(X_def);
    for i=1:m
        plot(X_def(i,:),Y_def(i,:), 'b', 'linewidth', 1);
    end
    for j=1:n
        plot(X_def(:,j),Y_def(:,j), 'b', 'linewidth', 1);
    end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Calcula a funcao objectivo, os constrangimentos e os vectores de
% deslocamentos adjuntos para todas as Nteta performances tensao
%
f0val= 0;
for i=1:nelementos
    f0val= f0val + area(i);
end
%
x_adj= zeros(nnos*2,Nteta);
%
for iperf= 1:Nteta
    elvm= performance_tensao(iperf);
    f_adj= zeros(nnos*2,1);
    %
    mat= elementos(5,elvm);
    A= area(elvm);
    E= materiais(1,mat);

```

```

niu= materiais(2,mat);

[D]= calcula_d(E,niu);

xye= zeros(4,2);

ue= zeros(8,1);

for j= 1:4

    ino= elementos(j,elvm);

    xye(j,1)= nos(1,ino);

    xye(j,2)= nos(2,ino);

    ue(j*2-1)= x(ino*2-1);

    ue(j*2)= x(ino*2);

end

%

% Calcula a contribuição de cada ponto de Gauss para o vector adjunto

%

for is= 1:2 % 2 pontos de Gauss na direcção s

    s= gst(is);

    ws= gw(is);

    for it= 1:2 % 2 pontos de Gauss na direcção t

        t= gst(it);

        wt= gw(it);

        % cálculo das derivadas das funções de forma em

        % ordem a x,y e do determinante da transformação

        % de coordenadas no ponto de gauss

        [gdn,det]= grad4(xye, s, t);

        B= zeros(3,8);

        for ino=1:4

            B(1,ino*2-1)= gdn(ino,1); % dNdx

            B(2,ino*2)= gdn(ino,2); % dNdy

            B(3,ino*2-1)= gdn(ino,2);

            B(3,ino*2)= gdn(ino,1);

        end

        sigma= D * B * ue ;

        %

        fval(iperf) = fval(iperf) + sqrt( sigma(1)*sigma(1) + ... % CALCULO DA TVM

            sigma(2)*sigma(2) - ...

            sigma(1)*sigma(2) + ...

            3*sigma(3)*sigma(3) ) * ...

        det * ws * wt / A ;

```

```

%
if SENSIBILIDADES

    const= 1 / (2*sqrt(sigma(1)*sigma(1)+sigma(2)*sigma(2)-...
        sigma(1)*sigma(2)+3*sigma(3)*sigma(3)));

    dsigmaxx= const * ( 2* sigma(1) - sigma(2) );
    dsigmayy= const * ( 2 *sigma(2) - sigma(1) );
    dsigmaxy= const * 6 * sigma(3);

    dsigma = [dsigmaxx dsigmayy dsigmaxy];

%

    f_adj_e= dsigma * D * B * det * ws * wt / A ;

%

% Assemblagem

%

for i= 1:4

    ino= elementos(i,elvm);

    f_adj(ino*2-1:ino*2)= f_adj(ino*2-1:ino*2) + f_adj_e(i*2-1:i*2)';

end

end

end

end

%

if SENSIBILIDADES

    x_adj([d2,d3],iperf)=Kr\([f_adj(d2);f_adj(d3)]);

    x_adj(d4,iperf)= x_adj(d3,iperf);

    fprintf('iperf(%d)=%30.20e\n', iperf,fval(iperf));

end

%

end

%
%
% Calcula as sensibilidades

%

if SENSIBILIDADES

    for ivar= 1:nvar

        %

        % Calcula a velocidade nos nós para cada

        % variável, gerando a nova posição dos nós

        %

```

```

nos_mov= zeros(2,nnos);

nos_spline = xval;

nos_spline(ivar)= nos_spline(ivar)+DELTA_V;

[X,Y,nao_faz_falta]=
gera_malha_mapeada_sup_elipse_placa_completa(nos_spline',Nr,Nteta);

ino= 1;

for j= 1:4*Nteta

    for k= 1:Nr+1

        nos_mov(1,ino)= X(j,k);

        nos_mov(2,ino)= Y(j,k);

        ino= ino + 1;

    end

end

vel= nos_mov - nos;

%

% Calcula o gradiente do volume

%

vprime= 0;

for i=1:nelementos

    xye= zeros(4,2);

    ve= zeros(8,1);

    for j= 1:4

        ino= elementos(j,i);

        xye(j,1)= nos(1,ino);

        xye(j,2)= nos(2,ino);

        ve(j*2-1)= vel(ino*2-1);

        ve(j*2)= vel(ino*2);

    end

    %

    % Calcula a contribuição do elemento para vprime

    %

    for is= 1:2 % 2 pontos de Gauss na direcção s

        s= gst(is);

        ws= gw(is);

        for it= 1:2 % 2 pontos de Gauss na direcção t

            t= gst(it);

            wt= gw(it);

            [gdn,det]= grad4(xye, s, t);

            grad= zeros(4,8);

```



```

        for ino=1:4

            grad(1,ino*2-1)= gdn(ino,1); % dNdx
            grad(2,ino*2-1)= gdn(ino,2); % dNdy
            grad(3,ino*2)= gdn(ino,1); % dNdx
            grad(4,ino*2)= gdn(ino,2); % dNdy

        end

        grad_vel= grad * ve ;

        if norm(ve) ~= 0

            vprime= vprime + det * ws * wt * ...

                (grad_vel(1)+grad_vel(4));

        end

    end

end

end % i= 1:nelementos

df0dx(ivar)= vprime/DELTA_V;

%

for iperf= 1:Nteta

    %

    aprime= 0;

    %

    for i=1:nelementos

        % define o material do elemento

        mat= elementos(5,i);

        % xye guarda as coordenadas x,y; ue os deslocamentos u,v; ue_adj os

        % deslocamentos adjuntos e ve as velocidades dos 4 nós do elemento

        xye= zeros(4,2);

        ue= zeros(8,1);

        ue_adj= zeros(8,1);

        ve= zeros(8,1);

        % define os vectores com as coordenadas, os deslocamentos,

        % os deslocamentos adjuntos e as velocidades do elemento

        for j= 1:4

            ino= elementos(j,i);

            xye(j,1)= nos(1,ino);

            xye(j,2)= nos(2,ino);

            ue(j*2-1)= x(ino*2-1);

            ue(j*2)= x(ino*2);

            ue_adj(j*2-1)= x_adj(ino*2-1,iperf);

```

```

    ue_adj(j*2)= x_adj(ino*2,iperf);

    ve(j*2-1)= vel(ino*2-1);

    ve(j*2)= vel(ino*2);

end

% cálculo de matriz de constantes elásticas

[D]= calcula_d(materiais(1,mat),materiais(2,mat));

%

% Calcula as tensões em cada ponto de Gauss

% e a contribuição do elemento para a prime

%

for is= 1:2 % 2 pontos de Gauss na direcção s

    s= gst(is);

    ws= gw(is);

    for it= 1:2 % 2 pontos de Gauss na direcção t

        t= gst(it);

        wt= gw(it);

        % cálculo das derivadas das funções de forma em

        % ordem a x,y e do determinante da transformação

        % de coordenadas no ponto de gauss

        [gdn,det]= grad4(xye, s, t);

        % cálculo da matriz B

        B= zeros(3,8);

        grad= zeros(4,8);

        for ino=1:4

            B(1,ino*2-1)= gdn(ino,1); % dNdx

            B(2,ino*2)= gdn(ino,2); % dNdy

            B(3,ino*2-1)= gdn(ino,2);

            B(3,ino*2)= gdn(ino,1);

            %

            grad(1,ino*2-1)= gdn(ino,1); % dNdx

            grad(2,ino*2-1)= gdn(ino,2); % dNdy

            grad(3,ino*2)= gdn(ino,1); % dNdx

            grad(4,ino*2)= gdn(ino,2); % dNdy

        end

        % grad_des, grad_des_adj e grad_vel são vectores com

        % quatro elementos,

        % dUx/dx = Ux,x = grad(1)

        % dUx/dy = Ux,y = grad(2)

```

```

% dUy/dx = Uy,x = grad(3)

% dUy/dy = Uy,y = grad(4)

grad_des= grad * ue ;

grad_des_adj= grad * ue_adj ;

grad_vel= grad * ve ;

% sigma e sigma_adj são vectores com tres elementos

% Sigma XX= sigma(1)

% Sigma YY= sigma(2)

% Tau XY= sigma(3)

sigma= D * B * ue ;

sigma_adj= D * B * ue_adj ;

%

% Calcula a contribuição deste elemento para aprime

%

if norm(ve) ~= 0

    %fprintf('vai calcular aprime no elemento %d\n',i);

    aprime= aprime - det * ws * wt * ...

        ( sigma(1)*(grad_des_adj(1)*grad_vel(1)+...
grad_des_adj(2)*grad_vel(3))+...
sigma(2)*(grad_des_adj(3)*grad_vel(2)+...
grad_des_adj(4)*grad_vel(4))+...
sigma(3)*(grad_des_adj(1)*grad_vel(2)+...
grad_des_adj(2)*grad_vel(4)+...
grad_des_adj(3)*grad_vel(1)+...
grad_des_adj(4)*grad_vel(3))+...
sigma_adj(1)*(grad_des(1)*grad_vel(1)+...
grad_des(2)*grad_vel(3))+...
sigma_adj(2)*(grad_des(3)*grad_vel(2)+...
grad_des(4)*grad_vel(4))+...
sigma_adj(3)*(grad_des(1)*grad_vel(2)+...
grad_des(2)*grad_vel(4)+...
grad_des(3)*grad_vel(1)+...
grad_des(4)*grad_vel(3))-...
(sigma(1)*grad_des_adj(1)+...
sigma(2)*grad_des_adj(4)+...
sigma(3)*(grad_des_adj(2)+grad_des_adj(3)))*...
(grad_vel(1)+grad_vel(4)));

%

```

```

        end
    end
end

end % for i=1:nelementos
%
% Calcula VonMisesPrime
%
elvm= performance_tensao(iperf);
mat= elementos(5,elvm);
A= area(elvm);
E= materiais(1,mat);
niu= materiais(2,mat);
[D]= calcula_d(E,niu);
%
% xye guarda as coordenadas x,y dos 4 nós
xye= zeros(4,2);
ue= zeros(8,1);
ve= zeros(8,1);
for j= 1:4
    ino= elementos(j,elvm);
    xye(j,1)= nos(1,ino);
    xye(j,2)= nos(2,ino);
    ue(j*2-1)= x(ino*2-1);
    ue(j*2)= x(ino*2);
    ve(j*2-1)= vel(ino*2-1);
    ve(j*2)= vel(ino*2);
end
%
% Calcula a contribuição de cada ponto de Gauss para von Mises Prime
%
c1= 0;
c2= 0;
c3= 0;
c4= 0;
for is= 1:2 % 2 pontos de Gauss na direcção s
    s= gst(is);
    ws= gw(is);
    for it= 1:2 % 2 pontos de Gauss na direcção t

```

```

t= gst(it);

wt= gw(it);

% cálculo das derivadas das funções de forma em
% ordem a x,y e do determinante da transformação
% de coordenadas no ponto de gauss

[gdn,det]= grad4(xye, s, t);

% cálculo da matriz B

B= zeros(3,8);

grad= zeros(4,8);

for ino=1:4

    B(1,ino*2-1)= gdn(ino,1);    % dNdx

    B(2,ino*2)= gdn(ino,2);     % dNdy

    B(3,ino*2-1)= gdn(ino,2);

    B(3,ino*2)= gdn(ino,1);

    %

    grad(1,ino*2-1)= gdn(ino,1); % dNdx

    grad(2,ino*2-1)= gdn(ino,2); % dNdy

    grad(3,ino*2)= gdn(ino,1);  % dNdx

    grad(4,ino*2)= gdn(ino,2);  % dNdy

end

% grad_des e grad_vel são vectores com quatro elementos,
% dUx/dx = Ux,x = grad(1)
% dUx/dy = Ux,y = grad(2)
% dUy/dx = Uy,x = grad(3)
% dUy/dy = Uy,y = grad(4)

grad_des= grad * ue ;

grad_vel= grad * ve ;

% Sigma XX= sigma(1)

% Sigma YY= sigma(2)

% Tau XY= sigma(3)

sigma= D * B * ue ;

%

const= 1 / (2*sqrt(sigma(1)*sigma(1)+sigma(2)*sigma(2)-...
    sigma(1)*sigma(2)+3*sigma(3)*sigma(3)));

dsigmaxx= const * ( 2* sigma(1) - sigma(2) );

dsigmayy= const * ( 2 *sigma(2) - sigma(1) );

dsigmaxy= const * 6 * sigma(3);

dsigma = [dsigmaxx dsigmayy dsigmaxy];

```

```

%
dudx= [grad_des(1)    0    grad_des(2)    0    ;...
       0    grad_des(3)    0    grad_des(4) ;...
       grad_des(3) grad_des(1) grad_des(4) grad_des(2)];

c1= c1 + det * ws * wt * dsigma * D * dudx * grad_vel ;
%
c2= c2 + det * ws * wt * ( 0.5 / const ) * ( grad_vel(1) + grad_vel(4) ) ;
%
c3= c3 + det * ws * wt * ( 0.5 / const ) ;
%
c4= c4 + det * ws * wt * ( grad_vel(1) + grad_vel(4) ) ;
%
end

end

%
vprime= -c1 / A + c2 / A - c3 * c4 / ( A * A ) ;
%
dfdx(iperf,ivar)= -((-aprime+vprime)/DELTA_V)/SIGMA_ADM;
%
end % for iperf=1:Nteta

end % for ivar= 1:nvar

end % if SENSIBILIDADES

%
% Normaliza o valor dos constrangimentos
%
for iperf= 1:Nteta
    fval(iperf)= (SIGMA_ADM - fval(iperf))/SIGMA_ADM ; %substrair a tensao de vm a tensao adm;
end
fprintf('FIM\n');
end % function

% fim

```

Programa gera_malha_mapeada_sup_elipse_placa_completa.m

```
%  
  
% Programa que gera uma malha mapeada para todo o dominio  
  
% no exemplo placa com furo definido pela equacao da super elipse  
  
%  
  
% Parâmetros: x = Vector linha com as 3 variáveis de projecto (a b n)  
  
%  
  
% Variáveis internas, a definir para cada problema de optimização:  
  
%  
  
%           A,B = Dimensões da placa segundo X,Y  
  
%           Nr,Nteta = Número de elementos nas direcções radial e circunferencial  
  
%           (RESTRICAO: Nteta deve ser um numero PAR)  
  
%  
  
% Utilização:  
  
% [X,Y,elementos,xx,yy]= gera_malha_mapeada_sup_elipse_placa_completa([2 1 2],30,20);  
function [XX,YY,elementos,xx,yy]= gera_malha_mapeada_sup_elipse_placa_completa(x,Nr,Nteta)  
  
%  
  
GRAFICO = 0;  
  
OUTPUT_ANSYS = 0;  
  
%  
  
A= 0.1;  
  
B= 0.1;  
  
%  
  
% Valores das variáveis de projecto  
  
%  
  
a= x(1);  
  
b= x(2);  
  
n= x(3);  
  
%  
  
% Equacao da Super Elipse  
  
%  
  
N=9;           % n° de pontos
```

```

m=200;

xdata = zeros(1,N); % vetor das coordenadas x
ydata = zeros(1,N);

t= 0; % angulo teta
ang_incr = pi/(2*(m-1)); % angulo de incremento

for i = 1 : m % pi/24 = 13 pontos

    xdata(i) = real(a*(cos(t))^(2/n));
    ydata(i) = real(b*(sin(t))^(2/n));
    t = t + ang_incr;

end

p=[xdata' , ydata'];
q = curvspace(p,N); % retorna os pontos igualmente espaçados

xx=q(:,1);
yy=q(:,2);

%
% Gera a malha para toda a placa
%
X= zeros(4*Nteta+1, Nr+1);
Y= X;
nos= zeros(4*Nteta, Nr+1);
Nelm= Nteta*Nr; % Número de elementos em 1/4 de placa
NTelm= 4*Nelm; % Número total de elementos
elementos= zeros(NTelm,4); % 4 nós (conectividade dos nós)

% Gera 16 quadrilateros

% bloco 1
coord= [ a 0;...
        (A+a)/2 0;...
        A 0;...
        A B/4;...
        A B/2;...

```



```

(2*A+3*xx(3))/5 (2*B/2+3*yy(3))/5;... % Ponto 6

xx(3) yy(3);...

xx(2) yy(2)];

[X(1:Nteta/4+1,:),Y(1:Nteta/4+1,:),nos(1:Nteta/4+1,:),elementos(1:Nelm/4,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1);

% bloco 2

coord= [ xx(3) yy(3);...

(2*A+3*xx(3))/5 (2*B/2+3*yy(3))/5;... % Ponto 2 = ponto 6 antigo

A B/2;...

A B*3/4;...

A B;...

(2*A+4*xx(5))/6 (2*B+4*yy(5))/6;... % ponto 6

xx(5) yy(5);...

xx(4) yy(4)];

[X(Nteta/4+1:Nteta/2+1,:),Y(Nteta/4+1:Nteta/2+1,:),nos(Nteta/4+1:Nteta/2+1,:),elementos(Nelm/4+1:Nelm/2,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta/4);

% bloco 3

coord= [ xx(5) yy(5);...

(2*A+4*xx(5))/6 (2*B+4*yy(5))/6;... %ponto 2

A B;...

A*3/4 B;...

A/2 B;...

(2*A/2+3*xx(7))/5 (2*B+3*yy(7))/5;...

xx(7) yy(7);...

xx(6) yy(6)];

[X(Nteta/2+1:Nteta*3/4+1,:),Y(Nteta/2+1:Nteta*3/4+1,:),nos(Nteta/2+1:Nteta*3/4+1,:),elementos(Nelm/2+1:Nelm*3/4,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta/2);

% bloco 4

coord= [ xx(7) yy(7);...

(2*A/2+3*xx(7))/5 (2*B+3*yy(7))/5;...

A/2 B;...

A/4 B;...

0 B;...

```

```

0 (B+b)/2;...
0 b;...
xx(8) yy(8)];

[X(Nteta*3/4+1:Nteta+1,:),Y(Nteta*3/4+1:Nteta+1,:),nos(Nteta*3/4+1:Nteta+1,:),elementos(Nelm*3/4+1:Nelm, :)] = ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*3/4);

% bloco 5
coord= [ 0 b;...
0 (B+b)/2;...
0 B;...
-A/4 B;...
-A/2 B;...
-(2*A/2+3*xx(7))/5 (2*B+3*yy(7))/5;...
-xx(7) yy(7);...
-xx(8) yy(8)];

[X(Nteta+1:Nteta*5/4+1,:),Y(Nteta+1:Nteta*5/4+1,:),nos(Nteta+1:Nteta*5/4+1,:),elementos(Nelm+1:Nelm*5/4, :)] = ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta);

% bloco 6
coord= [ -xx(7) yy(7);...
-(2*A/2+3*xx(7))/5 (2*B+3*yy(7))/5;...
-A/2 B;...
-A*3/4 B;...
-A B;...
-(2*A+4*xx(5))/6 (2*B+4*yy(5))/6;...
-xx(5) yy(5);...
-xx(6) yy(6)];

[X(Nteta*5/4+1:Nteta*6/4+1,:),Y(Nteta*5/4+1:Nteta*6/4+1,:),nos(Nteta*5/4+1:Nteta*6/4+1,:),elementos(Nelm*5/4+1:Nelm*6/4, :)] = ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*5/4);

% bloco 7
coord= [ -xx(5) yy(5);...
-(2*A+4*xx(5))/6 (2*B+4*yy(5))/6;...
-A B;...

```

```

-A B*3/4;...

-A B/2;...

-(2*A+3*xx(3))/5 (2*B/2+3*yy(3))/5;...

-xx(3) yy(3);...

-xx(4) yy(4)];

[X(Nteta*6/4+1:Nteta*7/4+1,:),Y(Nteta*6/4+1:Nteta*7/4+1,:),nos(Nteta*6/4+1:Nteta*7/4+1,:),elementos(N
elm*6/4+1:Nelm*7/4,:)] = ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*6/4);

% bloco 8

coord= [ -xx(3) yy(3);...

-(2*A+3*xx(3))/5 (2*B/2+3*yy(3))/5;...

-A B/2;...

-A B/4;...

-A 0;...

-(A+a)/2 0;...

-a 0;...

-xx(2) yy(2)];

[X(Nteta*7/4+1:Nteta*8/4+1,:),Y(Nteta*7/4+1:Nteta*8/4+1,:),nos(Nteta*7/4+1:Nteta*8/4+1,:),elementos(N
elm*7/4+1:Nelm*8/4,:)] = ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*7/4);

% bloco 9

coord= [ -a 0;...

-(A+a)/2 0;...

-A 0;...

-A -B/4;...

-A -B/2;...

-(2*A+3*xx(3))/5 -(2*B/2+3*yy(3))/5;...

-xx(3) -yy(3);...

-xx(2) -yy(2)];

[X(Nteta*8/4+1:Nteta*9/4+1,:),Y(Nteta*8/4+1:Nteta*9/4+1,:),nos(Nteta*8/4+1:Nteta*9/4+1,:),elementos(N
elm*8/4+1:Nelm*9/4,:)] = ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*8/4);

% bloco 10

coord= [ -xx(3) -yy(3);...

```

```

-(2*A+3*xx(3))/5 -(2*B/2+3*yy(3))/5;...
-A -B/2;...
-A -B*3/4;...
-A -B;...
-(2*A+4*xx(5))/6 -(2*B+4*yy(5))/6;...
-xx(5) -yy(5);...
-xx(4) -yy(4)];

[X(Nteta*9/4+1:Nteta*10/4+1,:),Y(Nteta*9/4+1:Nteta*10/4+1,:),nos(Nteta*9/4+1:Nteta*10/4+1,:),elementos(Nelm*9/4+1:Nelm*10/4,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*9/4);

% bloco 11
coord= [ -xx(5) -yy(5);...
-(2*A+4*xx(5))/6 -(2*B+4*yy(5))/6;...
-A -B;...
-A*3/4 -B;...
-A/2 -B;...
-(2*A/2+3*xx(7))/5 -(2*B+3*yy(7))/5;...
-xx(7) -yy(7);...
-xx(6) -yy(6)];

[X(Nteta*10/4+1:Nteta*11/4+1,:),Y(Nteta*10/4+1:Nteta*11/4+1,:),nos(Nteta*10/4+1:Nteta*11/4+1,:),elementos(Nelm*10/4+1:Nelm*11/4,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*10/4);

% bloco 12
coord= [ -xx(7) -yy(7);...
-(2*A/2+3*xx(7))/5 -(2*B+3*yy(7))/5;...
-A/2 -B;...
-A/4 -B;...
0 -B;...
0 -(B+b)/2;...
0 -b;...
-xx(8) -yy(8)];

[X(Nteta*11/4+1:Nteta*12/4+1,:),Y(Nteta*11/4+1:Nteta*12/4+1,:),nos(Nteta*11/4+1:Nteta*12/4+1,:),elementos(Nelm*11/4+1:Nelm*12/4,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*11/4);

```

```

% bloco 13

coord= [ 0 -b;...

        0 -(B+b)/2;...

        0 -B;...

        A/4 -B;...

        A/2 -B;...

        (2*A/2+3*xx(7))/5 - (2*B+3*yy(7))/5;...

        xx(7) -yy(7);...

        xx(8) -yy(8)];

[X(Nteta*12/4+1:Nteta*13/4+1,:),Y(Nteta*12/4+1:Nteta*13/4+1,:),nos(Nteta*12/4+1:Nteta*13/4+1,:),elemen
ntos(Nelm*12/4+1:Nelm*13/4,:)] = ...

        malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*12/4);

% bloco 14

coord= [ xx(7) -yy(7);...

        (2*A/2+3*xx(7))/5 - (2*B+3*yy(7))/5;...

        A/2 -B;...

        A*3/4 -B;...

        A -B;...

        (2*A+4*xx(5))/6 - (2*B+4*yy(5))/6;...

        xx(5) -yy(5);...

        xx(6) -yy(6)];

[X(Nteta*13/4+1:Nteta*14/4+1,:),Y(Nteta*13/4+1:Nteta*14/4+1,:),nos(Nteta*13/4+1:Nteta*14/4+1,:),elemen
ntos(Nelm*13/4+1:Nelm*14/4,:)] = ...

        malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*13/4);

% bloco 15

coord= [ xx(5) -yy(5);...

        (2*A+4*xx(5))/6 - (2*B+4*yy(5))/6;...

        A -B;...

        A -B*3/4;...

        A -B/2;...

        (2*A+3*xx(3))/5 - (2*B/2+3*yy(3))/5;...

        xx(3) -yy(3);...

        xx(4) -yy(4)];

[X(Nteta*14/4+1:Nteta*15/4+1,:),Y(Nteta*14/4+1:Nteta*15/4+1,:),nos(Nteta*14/4+1:Nteta*15/4+1,:),elemen
ntos(Nelm*14/4+1:Nelm*15/4,:)] = ...

```

```

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*14/4);

% bloco 16
coord= [ xx(3) -yy(3);...
        (2*A+3*xx(3))/5 - (2*B/2+3*yy(3))/5;...
        A -B/2;...
        A -B/4;...
        A 0;...
        (A+a)/2 0;...
        a 0;...
        xx(2) -yy(2)];

[X(Nteta*15/4+1:Nteta*16/4+1,:),Y(Nteta*15/4+1:Nteta*16/4+1,:),nos(Nteta*15/4+1:Nteta*16/4+1,:),elementos(Nelm*15/4+1:Nelm*16/4,.)]= ...

malha_quadrilatero(coord,Nr+1,Nteta/4+1,1+(1+Nr)*Nteta*15/4);

% Merge dos nós
XX= X(1:4*Nteta,:);
YY= Y(1:4*Nteta,:);

for i=1:NTelm
    for j= 1:4
        no= elementos(i,j);
        if 4*Nteta*(Nr+1)+1 <= no && no <= (4*Nteta+1)*(Nr+1)
            elementos(i,j) = no - 4*Nteta*(Nr+1);
            fprintf('no %d substituido pelo no %d\n', no, no-4*Nteta*(Nr+1));
        end
    end
end

if GRAFICO
    plotgrid(X,Y);
end

%
if OUTPUT_ANSYS
    fprintf('!Vai escrever a malha no ficheiro para o ANSYS... ');
    %
    ficheiro= 'malha_APDL.txt';
    fid= fopen(ficheiro,'w');

```

```

%
fprintf(fid, '! \r\n');
fprintf(fid, '! Malha gerada pelo programa OCTAVE\r\n');
fprintf(fid, '! \r\n');
fprintf(fid, '! gera_malha_mapeada.m\r\n');
fprintf(fid, '! \r\n');
fprintf(fid, '/PREP7 \r\n');
fprintf(fid, '! \r\n');
fprintf(fid, '! Elementos\r\n');
fprintf(fid, '! \r\n');
fprintf(fid, 'ET,1,PLANE182\r\n');
fprintf(fid, '! \r\n');
fprintf(fid, '! Material\r\n');
fprintf(fid, '! \r\n');
fprintf(fid, 'MP,EX,1,210e9\r\n');
fprintf(fid, 'MP,PRXY,1,0.3\r\n');
fprintf(fid, '! \r\n');
fprintf(fid, '! Nos\r\n');
ino= 1;
for i= 1:4*Nteta
    for j= 1:Nr+1
        fprintf(fid, 'N,%6d,%9.5f,%9.5f\r\n', ino, XX(i,j), YY(i,j));
        ino= ino + 1;
    end
end
fprintf(fid, '! \r\n');
fprintf(fid, '! Elementos\r\n');
for i= 1:NTelm
    fprintf(fid, 'EN,%6d,%6d,%6d,%6d,%6d\r\n', i, elementos(i,1), elementos(i,2), ...
        elementos(i,3), elementos(i,4));
end
fclose(fid);
fprintf(' terminou\r\n');
end

```



```

iter = iter+1;
itte = itte+1;

[xmma,ymma,zmma,lam,xsi,eta,mu,zet,s,low,upp] = ...
mmasub(m,n,iter,xval,xmin,xmax,xold1,xold2, ...
f0val,df0dx,df0dx2,fval,dfdx,dfdx2,low,upp,a0,a,c,d);

xold2 = xold1;
xold1 = xval;
xval = xmma;

[f0val,df0dx,df0dx2,fval,dfdx,dfdx2] = PROAES_membrana_condicoes_periodicidade(xval);
outvector = [iter f0val fval' xval']'
fprintf(fp,'\nIteracao %d\n',iter);
fprintf(fp,'      Obj= %f\n',f0val);
for i=1:m
    fprintf(fp,'      Constr(%3d)= %f\n',i,fval(i));
end
for i=1:n
    fprintf(fp,'      Var(%2d)= %f\n',i,xval(i));
end
end
fclose(fp);
% end

```