



Universidade de Aveiro

TP3

Aplicação do Método das Diferenças Finitas

MESTRADO INTEGRADO EM ENGENHARIA MECÂNICA

Simulação de Processos Tecnológicos
2014/15

António Almeida^{*}, Schnaider Mansoa⁺

^{}Nº.mec 64960, almeid@ua.pt*

⁺Nº.mec 41383, swgm@ua.pt

Índice

Índice de figuras	3
Índice de Tabelas	4
Índice de gráficos	4
Resumo	4
Introdução	4
Análise Estrutural	5
Apresentação do problema	5
Objetivos	6
Solução teórica	6
Resolução do problema e código	7
Construção do modelo	7
Limitações e vantagens	10
Análise em <i>Matlab</i>	11
Análise em <i>Abaqus</i>	11
Resultados	11
Análise de resultados	14
Análise Térmica	16
Apresentação do problema	16
Objetivos	17
Solução teórica	17
Resolução do problema e código	18
Construção do modelo	18
Limitações e vantagens	22
Outras observações	22
Análise em <i>Matlab</i>	23
Análise em <i>Abaqus</i>	24
Resultados e Análise de Resultados	24
Pensamentos finais	29
Agradecimentos	29

Bibliografia	29
Anexos	30
Código estrutural	30
Código térmico	38

Índice de figuras

Fig. 1 – Molécula genérica ^[1]	5
Fig. 2 - Esquema da placa	6
Fig. 3- Molécula do modelo estrutural ^[1]	7
Fig. 4 - Criação do ficheiro de dados	8
Fig. 5 - Dados do ficheiro	8
Fig. 6 - Condições de fronteira	9
Fig. 7 - Criação do intervalo entre pontos	9
Fig. 8 - Modelo de chapa antes da simetria	9
Fig. 9 - Matriz de simetria	10
Fig. 10 - Chapa simplesmente apoiada, com condições de fronteira	10
Fig. 11 - Domínio bidimensional do problema térmico	17
Fig. 12 - Condições de fronteira	17
Fig. 13- Molécula do modelo de Laplace com diferenças centradas ^[1]	18
Fig. 14 - Criação do ficheiro de dados	19
Fig. 15 – Representação da geometria do modelo e representação das condições de fronteira	19
Fig. 16 - Atribuição das temperaturas das faces	20
Fig. 17 - Matriz da posição das faces	20
Fig. 18 - Exemplo de malha de pontos	21
Fig. 19 - Numeração dos nós	21
Fig. 20 - Molécula genérica de diferenças a trás ^[1] .	22
Fig. 21 - Nós em x diferentes de y no Matlab	23
Fig. 22 - Nos diferentes no Abaqus	23

Índice de Tabelas

Tabela 1 - Resultados das simulações	11
Tabela 2 - Resultados gráficos das simulações par 6, 30 e 200 pontos.	12
Tabela 3- Distribuição de temperaturas para várias malhas	24

Índice de gráficos

Gráfico 1 - Valores de deslocamento	15
Gráfico 2 - Relação tempo de processamento vs. número de pontos	16
Gráfico 3 - Relação erro relativo vs. tempo de processamento	16

Resumo

Este trabalho consiste resolução de problemas térmicos e estruturais pelo método das diferenças finitas em cascas.

Para além da construção do código em *Matlab*, é feita a análise desses resultados face a resultados teórico (analíticos), e aos resultados obtidos pelo método dos elementos finitos. É assim possível determinar-se o a implementação é ou não viável, e quais as suas vantagens e desvantagens.

Introdução

Este trabalho é realizado no âmbito da unidade curricular Simulação de Processos Tecnológicos (SPT), de modo a testar outros métodos de resolução de problemas em cascas finas. No contexto desta disciplina, o termo “cascas finas” refere-se a sólidos tipo chapa, cuja espessura é muito inferior às restantes dimensões, é então possível aproximar estes problemas tridimensionais (3D) a problemas bidimensionais (2D).

Dentro da categoria de cascas finas pode-se encontrar, entre outros, dois tipos de problemas, condução de calor e chapa à flexão sob uma pressão em toda a sua área, que são o foco deste trabalho.

Estes tipos de fenómenos físicos em meios contínuos são modelados dos matematicamente por equações diferenciais difícil ou impossível resolução. Para superar estas dificuldades os modelos podem ser discretizados de modo a terem uma fácil resolução. Neste caso específico foi utilizada a discretização pela equação de *Laplace*.

Com o *Método das Diferenças Finitas 2D* o modelo é discretizado numa malha de pontos cujas coordenadas que representam a geometria do modelo, e o seu valor é o valor

da grandeza nesse ponto. A relação entre os pontos da malha é dada por uma molécula do tipo da representada na Fig. 1,

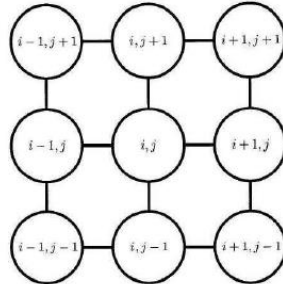


Fig. 1 – Molécula genérica^[1]

em que o valor de $M_{i,j}$ é função dos pontos vizinhos conforme a equação que descreve o fenómeno.

Para implementar este método foi usada a linguagem de programação *Matlab*. Está é uma linguagem rápida de programar e otimizada para cálculos matemáticos e utilização de matrizes.

Os *scripts Matlab*, tanto estrutural como térmico foram pensados de modo a serem rápidos, usando utilizando resolução matricial do género

$$Ax = b \quad \text{eq(1)}$$

A matriz A é obtida a partir da *molécula* característica de cada modelo, e b é o vetor com as condições essenciais. Por isto e por necessitar do mínimo de conhecimento do *script* por parte do utilizador, devido à interface via consola do *Matlab*, não só a construção do modelo é rápida como o processamento é mais eficiente.

O código para o MDF tem algumas limitações comuns aos dois tipos de problemas que neste trabalho nos é proposto resolver. Em primeiro lugar os *scripts* só conseguem resolver geometrias com faces ortogonais, uma grande limitação, nenhum dos códigos permite que as condições de fronteira variem numa dada face e, por facilidade de cálculo e limitação no tempo disponível para desenvolver o código, nenhum deles permite que dx seja diferente de dy (em que dx é distância entre pontos na direção x e dy é distância entre pontos na direção y).

Os resultados obtidos pelo *MDF* são então comparados com o resultado obtido pelo *software Abaqus*, que utiliza o *MEF* (*Método dos Elementos Finitos*), e sempre que disponível o resultado analítico. Para efeitos de análise dos resultados obtidos pelo código de *Matlab* foram obtidos resultados para várias malhas (tanto para o modelo em *Abaqus* como para o modelo *Matlab*).

Análise Estrutural

Apresentação do problema

O problema atribuído ao grupo para análise estrutural encontra-se representado na Fig. 2

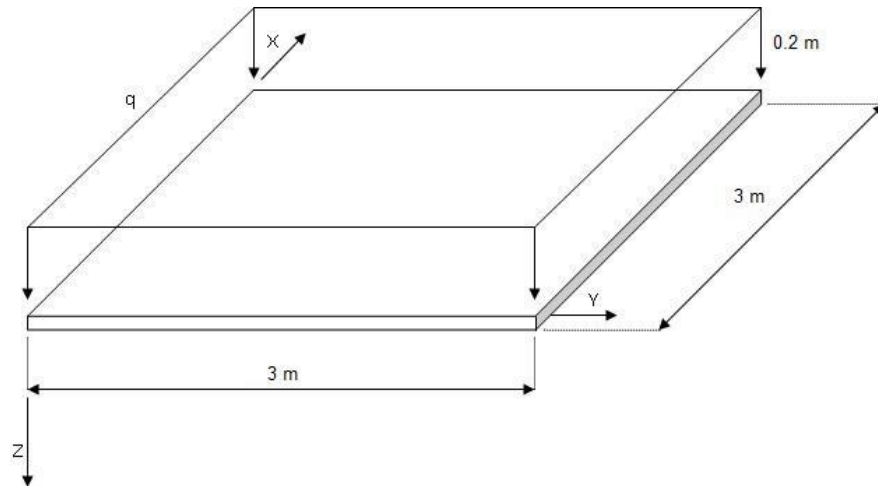


Fig. 2 - Esquema da placa

Neste caso tem-se uma carga uniformemente aplicada de 5 kN/mm^2 numa placa quadrada de 3m de lado e 0.2m de espessura simplesmente apoiada nos seus bordos. O material apresenta um módulo de Young de 210GPa e um coeficiente de Poisson de 0.3 . O objetivo do problema é calcular os deslocamentos verticais ao longo da placa recorrendo a malhas de diferenças finitas.

Objetivos

O objetivo principal é calcular os deslocamentos verticais ao longo da placa, recorrendo a uma malha de diferenças finitas que reproduza com precisão aceitável os resultados reais. Tem-se ainda como objetivo comparar os resultados obtidos pelos *softwares* *MATLAB* e *Abaqus*, bem como estudar malhas com diferentes refinamentos.

É também um objetivo testar até que densidade de malha vale a refinar para obter o resultado.

Solução teórica

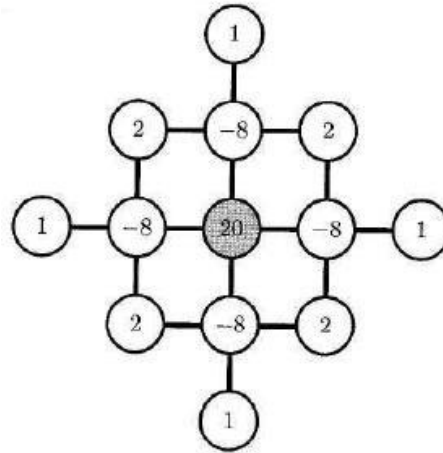
O modelo físico por trás do problema estrutural é dado por:

$$\frac{q(x,y)}{D} = \frac{\partial^4 w}{\partial x^4} + \frac{\partial^4 w}{\partial y^4} + 2 \frac{\partial^4 w}{\partial x^2 \partial y^2} \quad \text{eq(2)}^{[1]}$$

em que,

$$D = \frac{Et^3}{12(1-\nu^2)} \quad \text{eq(3)}^{[1]}$$

Ao aproximar esta equação esta equação de *Laplace* obtém-se uma *molécula* com o seguinte aspeto

Fig. 3- Molécula do modelo estrutural^[1]

No entanto este modelo é extremamente exigente de usar, pelo menos em malhas com mais de 9 pontos, a não ser que seja programado de modo a que um computador o execute autonomamente.

Os capítulos seguintes são assim o modo de utilização e aproximações tomadas no código de modo a resolver este tipo de problemas.

A solução teórica para este modelo estrutural é uma simplificação do problema que permite calcular o deslocamento máximo, que deverá ocorrer no centro da chapa.

O valor do deslocamento máximo depende de α que é função de a/b ($dimx/dimy$, da placa) e calcula-se pela eq(4).

$$\alpha \propto \frac{qa^4}{D}$$

eq(4) - Formula para calculo do deslocamento máximo^[2].

Para este problema $\alpha = 0.00406$, pois $a/b = 1$, $D = 1.5385e8$ e $q = 5000$. Temos por isso que o deslocamento máximo é $1.0688e-5$, em metros. Este valor será posteriormente utilizado como referencia para o erro das simulações.

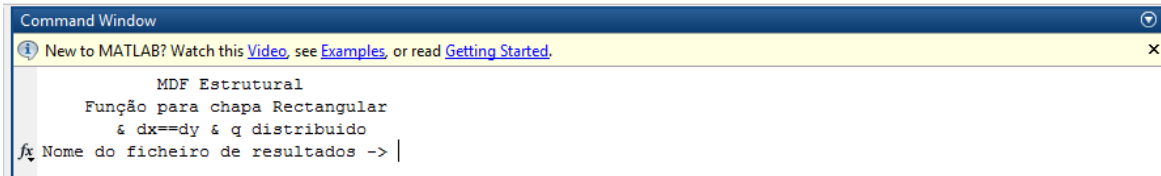
Resolução do problema e código

Construção do modelo

1) A construção do modelo

Inicia-se com a execução da secção “%- Dados - Criar Geometria”(em anexo no capítulo “Código estrutural”). Nesta secção é pedido ao utilizador que introduza o nome que quer para o ficheiro de dados do modelo (Fig. 4).

Entre cada uma das próximas fases é pedido o nome do ficheiro a alterar, não sendo possível editar apenas os atributos que se pretende testar. No entanto na primeira vez que o modelo é criado é necessário seguir esta ordem.



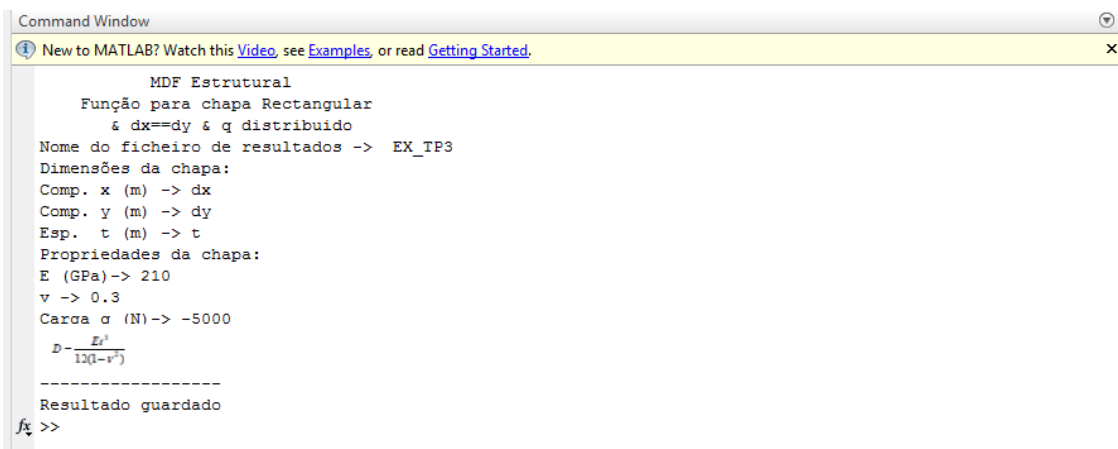
```

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
MDF Estrutural
Função para chapa Rectangular
& dx==dy & q distribuido
fx Nome do ficheiro de resultados -> |

```

Fig. 4 - Criação do ficheiro de dados

Em seguida são inseridos os dados do problema e o ficheiro é guardado (Fig. 5). O ficheiro criado tem o nome e é guardado no diretório atual como ficheiro de dados *Matlab* (.mat).



```

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
MDF Estrutural
Função para chapa Rectangular
& dx==dy & q distribuido
Nome do ficheiro de resultados -> EX_TP3
Dimensões da chapa:
Comp. x (m) -> dx
Comp. y (m) -> dy
Esp. t (m) -> t
Propriedades da chapa:
E (GPa)-> 210
v -> 0.3
Carga q (N)-> -5000
D = - (E t^3) / (12(1-v^2))
-----
Resultado guardado
fx >>

```

Fig. 5 - Dados do ficheiro

2) Criar as condições de fronteira.

Na secção “% - Dados - Criar Con. fronteira” (Fig. 6) cada uma das faces laterais é identificada por *F1* até *F4* e é-lhe atribuída uma condição das seguintes:

- e – encastrado;
- a – simplesmente apoiado;
- l – livre;

Nenhum modelo pode no entanto estar todo livre, tendo que ter pelo menos uma face encastrada ou duas apoiadas.

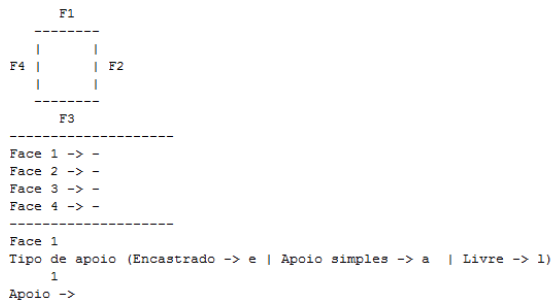


Fig. 6 - Condições de fronteira

3) Para escolher dimensão dos intervalos

Para escolher dx e dy (que nesta resolução simplificada são iguais, $dx=dy=dxy$) é necessário correr a secção “%- Dados - N^o pontos” (na Fig. 7 a interface da consola relembra quais as dimensões do modelo).

```

Nome do ficheiro de resultados -> expl
Cond. fronteira simetricas em x e y com: dimx = 100 ; dimy = 100
fx dxy (m) ->

```

Fig. 7 - Criação do intervalo entre pontos

4) Processamento

Secção “%- Processamento -“. Depois de escolhido o modelo a processar o programa segue os seguintes passos de modo a poder efetuar os cálculos.

- Verifica se o modelo é simétrico. Se as quatro faces forem do mesmo tipo (tenham as mesmas condições de fronteira “e” ou “a”) o modelo é reduzido a um quarto. Caso detete apenas duas faces opostas do mesmo tipo o modelo será reduzido a metade. Neste processo os nós são numerados de acordo com o modelo anteriormente criado. O próximo exemplo, Fig. 8, é uma chapa encastrada em todos os lados, logo simétrica em dois eixos. Na linha de fora os nós virtuais para efeitos de cálculo, na linha seguinte (azul) os nós do apoio sem deslocamento e ao centro (azul claro) as incógnitas.

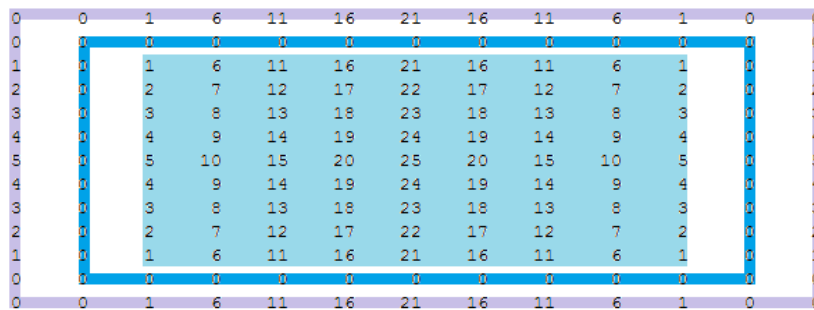


Fig. 8 - Modelo de chapa antes da simetria

0	0	1	6	11	16	21
0	0	0	0	0	0	0
1	0	1	6	11	16	21
2	0	2	7	12	17	22
3	0	3	8	13	18	23
4	0	4	9	14	19	24
5	0	5	10	15	20	25

Fig. 9 - Matriz de simetria

-Numa segunda matriz todos os pontos correspondentes a uma incógnita são substituídos por o valor 1, valor que entra na matriz de resolução. As únicas exceções são os nós virtuais de faces livres ou simplesmente apoiadas, que são 0 e -1 respetivamente, de modo a respeitar as condições de fronteira. Em anexo pode ser vista esta matriz para uma chapa simplesmente apoiada em todas as faces, como representado na *Erro! A origem da referência não foi encontrada.*

hh =	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	-1	0	0	0	0	0	0	0	0	0	0	-1
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	1	1	1	1	1	1	1	1	1	0
	-1	0	0	0	0	0	0	0	0	0	0	-1
	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Fig. 10 - Chapa simplesmente apoiada, com condições de fronteira

Depois por meio de um ciclo a posição de cada nó é associado à matriz “A” de acordo com a molécula do modelo estrutural. Neste caso “A” seria 25x25 e o vetor “b” 1x25 sendo cada elemento de “b” igual a

$$\frac{q \cdot dx \cdot y^4}{D}$$

Da operação

$$Ax = b \Leftrightarrow A \setminus b = x,$$

em que “x” é o vetor com deslocamento de cada ponto. É assim possível reconstruir a geometria do modelo com os valores de deslocamento no local onde ocorrem.

5) Resultados “%% Mostrar resultados”.

Os resultados são sempre guardados num ficheiro novo com o nome do original e com a terminação “_res_e” de *resolução estrutural*. Deste modo para um só modelo é possível criar várias resoluções sem perder a anterior.

Os resultados são representados num modelo 3D com a forma final do modelo.

Limitações e vantagens

Este é um código muito simples e daí algo limitado. Não permite forças localizadas nem refinação de malha e só permite modelos retangulares. Por outro a sua simplicidade e otimização de modelos simétricos também fazem com que esta resolução corra muito

depressa e com menor esforço computacional permitindo malhas com muitos pontos. Pode ser então uma ferramenta interessante para este tipo de exercício mesmo que não sendo versátil.

Outra desvantagem desta versão do código prende-se no facto de as fronteiras livres ainda não poderem ser usadas (não estão operacionais). No entanto estas não são necessárias para este relatório.

Análise em *Matlab*

A análise em *Matlab* foi feita seguindo os passos já apresentados. Recorrendo à secção 3) foram feitas varias simulações para diferentes dx_y .

Os valores utilizados para o número de nós por face foi 6, 10, 20, 30, 50, 100, 150, 200.

Com esta análise é possível descobrir qual o erro associado ao nosso MDF e qual o refinamento que permito o melhor resultado no menor tempo e custo de processamento.

Análise em *Abaqus*

Recorrendo ao *software Abaqus* foi também feita uma simulação. Para o MEF foram usados elementos do tipo C3D8 só com um elemento de espessura de modo a rentabilizar o número de pontos possíveis de utilizar na malha.

Esta é mais um termo de comparação para os resultados obtidos. Por apenas serem para comparação do valor do deslocamento, e por o número de nos estar limitado pela versão *Student*, só foram usados quatro espaçamentos na malha. Estes são 0.5, 0.3, 0.25, e 0.15, em metros e correspondem respetivamente a 6, 10, 20 e 30 nós.

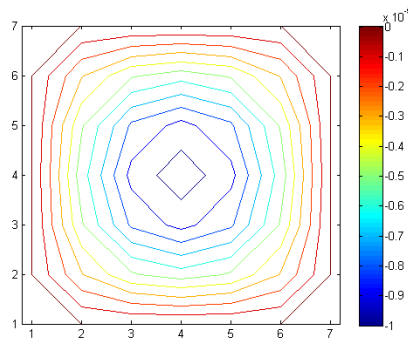
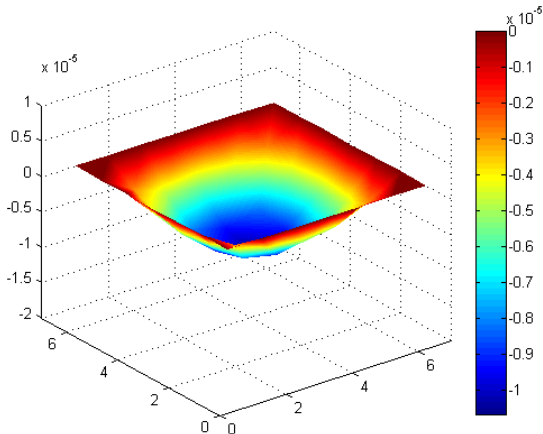
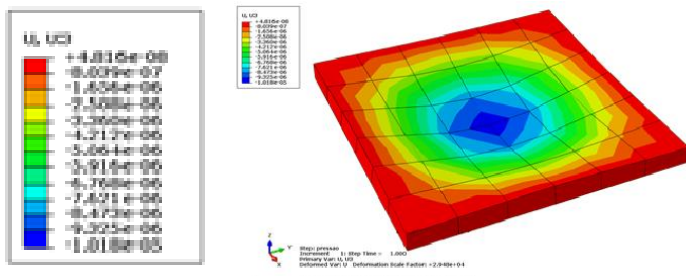
Resultados

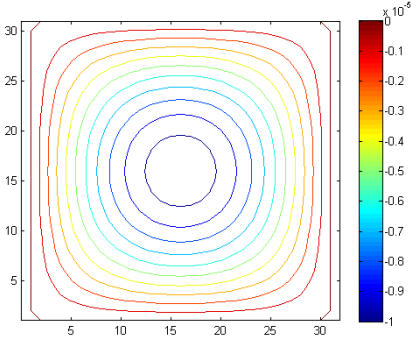
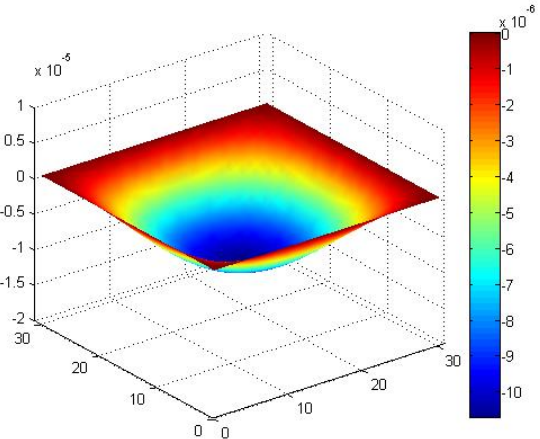
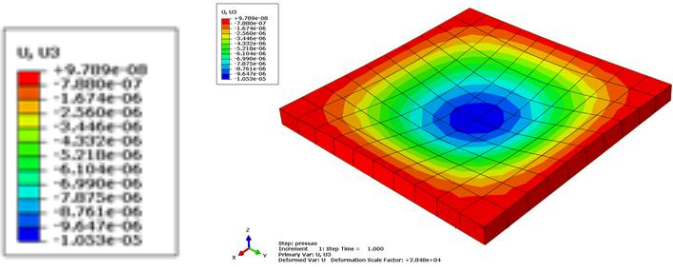
Depois de simular o modelo para as diferentes malhas nos dois *softwares* foram obtidos os valores de deformação e tempo de processamento que estão resumidos na Tabela 1, e algumas estão ilustradas na Tabela 2, a título de exemplo.

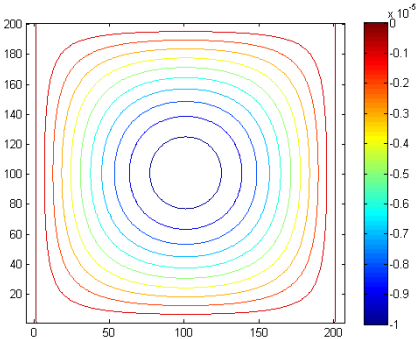
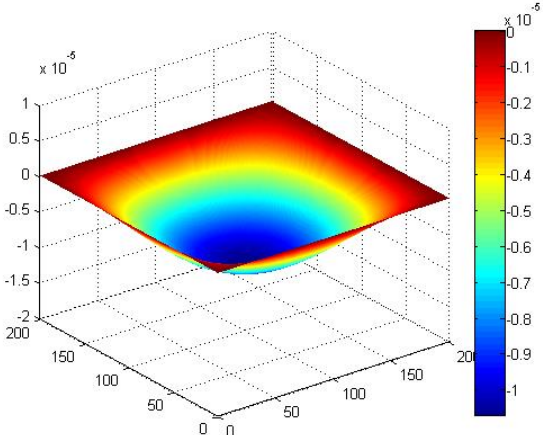
Tabela 1 - Resultados das simulações

Nº pontos por lado	Valor teórico	Valor obtido no <i>Matlab</i>	Valor obtido no <i>Abaqus</i>	Relativo <i>Matlab</i> (%)	Tempo de processamento <i>Matlab</i> (s)
200	1,069E-05	1,069E-05	-	5,614E-02	1,573E+01
150		1,069E-05	-	5,614E-02	3,957E+00
100		1,069E-05	-	5,614E-02	8,350E-01
50		1,069E-05	-	5,614E-02	2,631E-01
30		1,069E-05	1,070E-05	5,614E-02	2,242E-01
20		1,069E-05	1,059E-05	2,807E-02	2,084E-01
10		1,068E-05	1,053E-05	5,614E-02	2,031E-01
6		1,066E-05	1,018E-05	2,900E-01	2,030E-01

Tabela 2 - Resultados gráficos das simulações par 6, 30 e 200 pontos.

Nº pontos por lado = 6		
	Mapa de nível <i>Matlab</i>	
Representação 3D	<i>Matlab</i>	
	<i>Abaqus</i>	
Nº pontos por lado = 30		

	Mapa de nível <i>Matlab</i>	
Representação 3D	<i>Matlab</i>	
	<i>Abaqus</i>	
		Nº pontos por lado = 200

	Mapa de nível <i>Matlab</i>	
Representação 3D	<i>Matlab</i>	

Análise de resultados

Dos valores obtidos anteriormente é possível criar algumas relações e concluir sobre o desempenho do código.

Do Gráfico 1 é possível ver que o *MDF* converge para um valor estável, com um erro constante face ao valor teórico. Quando comparado com esta formulação do *MEF* o *MDF* converge mais depressa embora, tanto quanto estes dados nos permitem ver, com um erro maior para malhas mais refinadas.

Mesmo assim o erro relativo máximo foi 2,900E-01% para malhas grosseiras e o mínimo foi 2,807E-02%. Tendo em conta que o erro converge para 5,614E-02%, podemos dizer que esta é uma solução bastante aceitável, com valores do erro sempre inferiores a 1%.

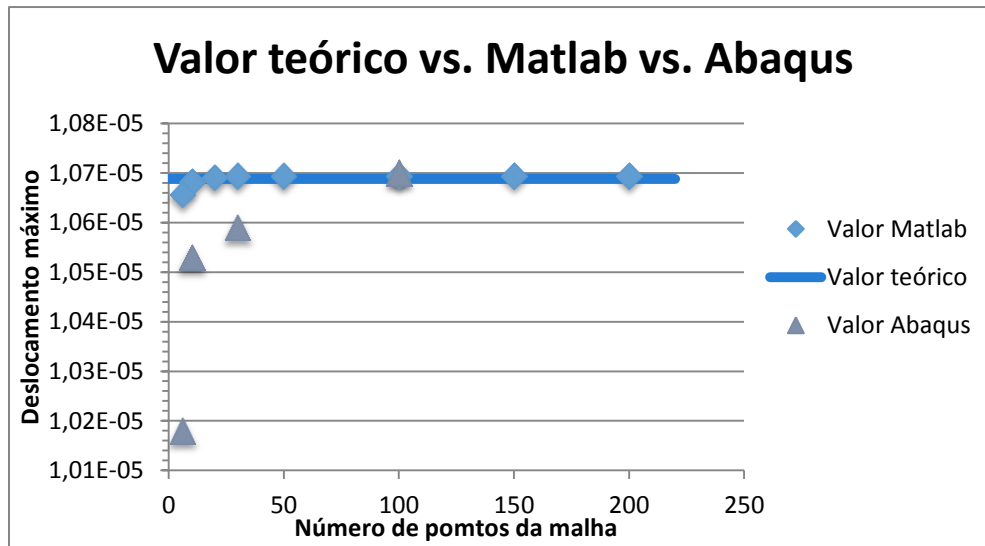


Gráfico 1 - Valores de deslocamento

Podem-se ainda tirar outras conclusões sobre o desempenho do código. Ao testar a relação entre tempo de processamento e número de pontos por face, e conjugando com essa relação os valores da relação erro relativo vs. tempo de processamento podemos estimar o número de pontos ótimo na redução do erro relativo e do tempo de processamento. Com base nos gráficos seguintes temos que:

- Tempo processamento para valor de convergência $\approx 0,3$ segundos;
- Tempo em função do número de pontos para o intervalo [6 ; 200] pontos é dado

por

$$t = 5E-06x^3 - 0,0009x^2 + 0,0409x - 0,1747 \quad \text{eq(5)}$$

Podemos concluir que o número de pontos que minimiza o tempo de processamento para o erro de convergência é ≈ 18 pontos por face, para este tipo de problema.

Embora o código não tenha grande versatilidade demonstrou que é possível obter muito boas aproximações pelo *MDF*.

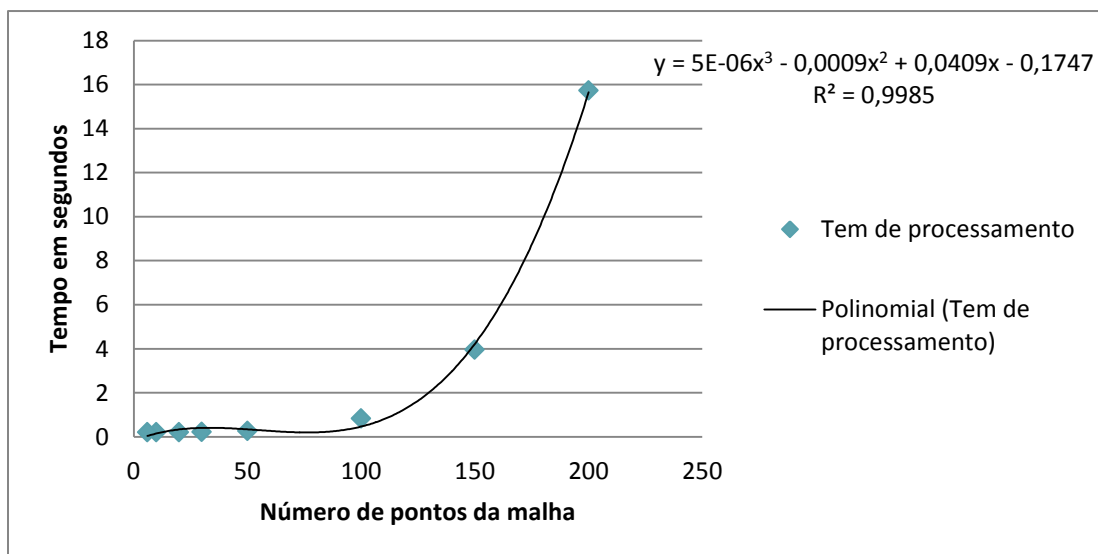


Gráfico 2 - Relação tempo de processamento vs. número de pontos

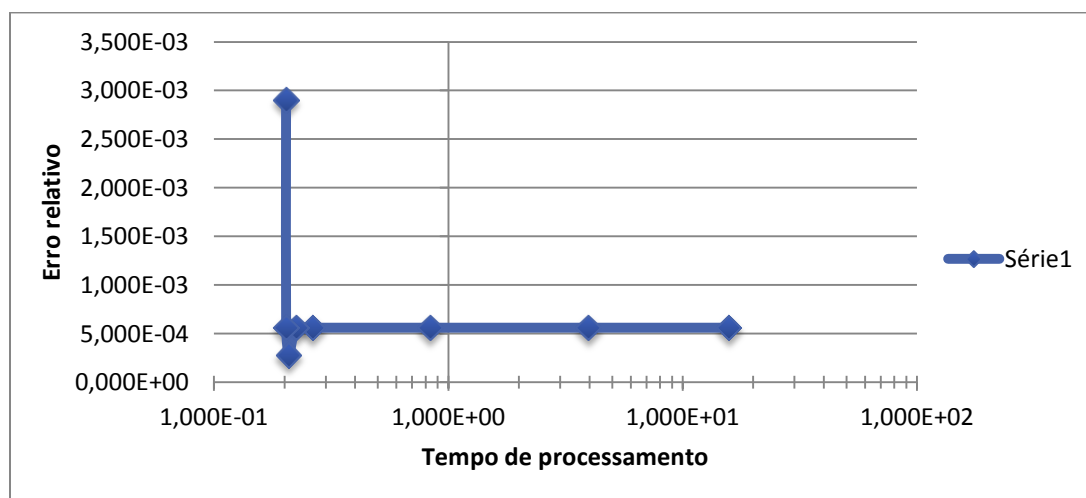


Gráfico 3 - Relação erro relativo vs. tempo de processamento

Análise Térmica

Apresentação do problema

O problema atribuído ao grupo para análise térmica encontra-se representado na Fig. 11

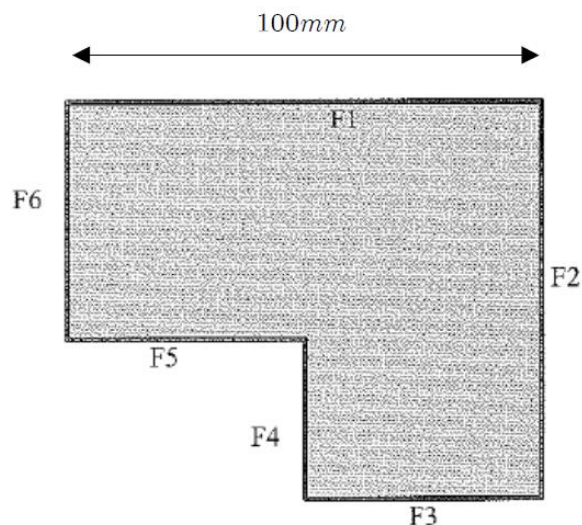


Fig. 11 - Domínio bidimensional do problema térmico

e é a condução 2D de calor na placa que tem as seguintes condições de fronteira (ver Fig. 12).

F1 - $T=40^{\circ}\text{C}$
F2 - $T=30^{\circ}\text{C}$
F3 - $\partial T / \partial y = 0$
F4 - $T=10^{\circ}\text{C}$
F5 - $T=0^{\circ}\text{C}$
F6 - $T=40^{\circ}\text{C}$

Fig. 12 - Condições de fronteira

Objetivos

O objetivo principal é calcular e representar o campo de temperaturas que ocorre na placa recorrendo a código *Matlab*.

Comparar estes resultados com os resultados obtidos no *Abaqus* a fim de os corroborar.

Solução teórica

O modelo físico por trás do problema térmico é dado por

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

, eq(6) Eq. de Laplace

que é a equação de *Laplace*. Esta equação pode ser simplificada aproximando δx^2 e δy^2 a Δx^2 e Δy^2 , e $\delta^2 T$ à variação entre T e T_{-1} , e T e T_{+1} . Desta forma tem-se que

$$\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{(\Delta y)^2} = 0 \quad \text{eq(7)}$$

Ao simplificar $\Delta x = \Delta y$ na equação de *Laplace* obtém-se

$$T_{i-1,j} - 4T_{i,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1} = 0 \quad \text{eq(8)}$$

que dá origem a uma *molécula* com o seguinte aspeto

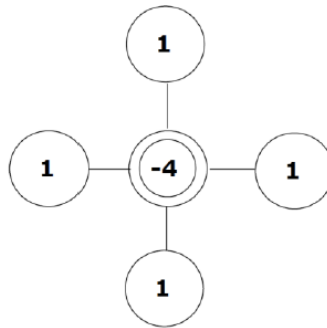


Fig. 13- Molécula do modelo de Laplace com diferenças centradas^[1]

Como já foi antes referido a resolução de um sistema desta natureza é algo muito moroso e por isso adequado para um computador.

Neste caso não existe solução analítica pelo que os resultados serão só comprados com os resultados obtidos no programa *Abaqus*.

Resolução do problema e código

Construção do modelo

1) A construção do modelo

Inicia-se com a execução da secção “% - Dados - Criar Geometria” (em anexo no capítulo “Código térmico”). Nesta secção é pedido ao utilizador que introduza o nome que quer para o ficheiro de dados do modelo (Fig. 14).

Entre cada uma das próximas fases é pedido o nome do ficheiro a alterar, não sendo possível editar apenas os atributos que se pretende testar. No entanto na primeira vez que o modelo é criado é necessário seguir esta ordem.

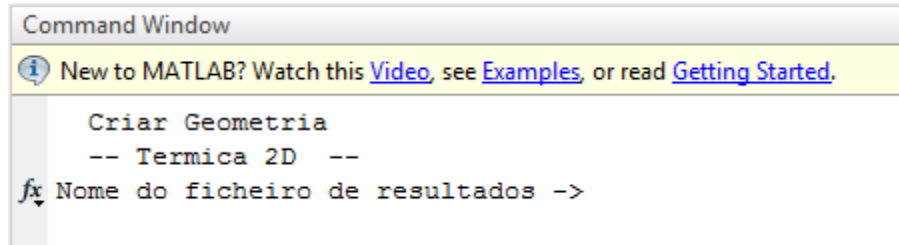


Fig. 14 - Criação do ficheiro de dados

Em seguida são inseridos os dados do problema e o ficheiro é guardado. O ficheiro criado tem o nome e é guardado no diretório atual como ficheiro de dados *Matlab* (.mat). Uma representação gráfica do modelo é criada à medida que as coordenadas deste são introduzidas.

2) Criar as condições de fronteira.

Na secção “% - Dados - Criar Con. fronteira” (Fig. 15) cada uma das faces laterais é identificada por $F1$ até $F_{n^{lados}}$ e é-lhe atribuída uma condição das seguintes:

t – se for uma temperatura;

q – se for transferência de calor;

Depois é inserido o valor respetivo à grandeza escolhida e é indicado se a face é fronteira de c - cima, b - baixo, e - esquerda ou d - direita.

A geometria é novamente representada para ser mais fácil identificar qual é cada face.

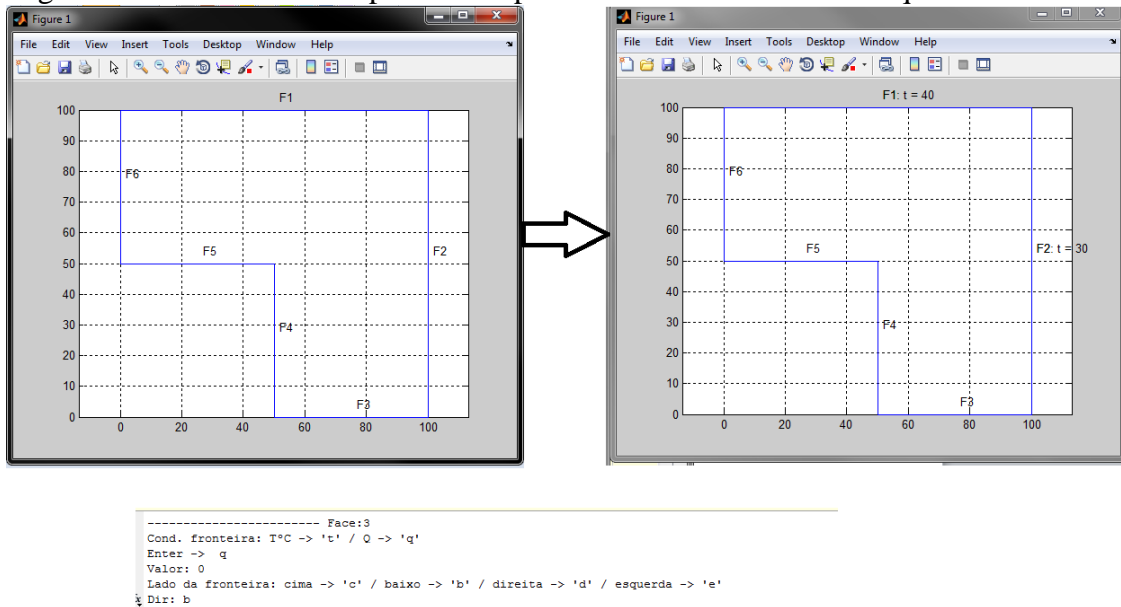


Fig. 15 – Representação da geometria do modelo e representação das condições de fronteira

3) Para escolher dimensão dos intervalos

Para escolher dx e dy (que nesta resolução simplificada são iguais) é necessário correr a secção “% - Dados - N° pontos”. Como são iguais $dx=dy=dxy$.

Como os objetos podem ter pequenas reentrâncias, ou saliências, dxy muito grandes podem fazer com que a geometria se perca. Para prevenir que tal aconteça o intervalo mínimo que o script aceita tem base na menor distância entre esquinas, (coordenadas da geometria introduzidas pelo utilizador). O dxy máximo é calculado como sendo a mínima distância a dividir por 3, de modo a deixar nós de fronteira.

4) Processamento

Secção “% - Processamento -”. Depois de escolhido o modelo a processar o programa segue os seguintes passos de modo a poder efetuar os cálculos.

- Calcula o número de pontos e converter as coordenadas reais em coordenadas da matriz discreta de pontos.

- Percorre as faces e a uma segunda matriz com as mesmas dimensões atribui o valor das mesmas (tal como na Fig. 16, onde o modelo está invertido pois o referencial vertical da matriz é de cima para baixo, ao contrário de um referencial convencional). No entanto os valores de esquina são deixados e só serão calculados depois de todo o miolo ter valores de temperatura.

0	0	0	0	1	0	0	0	0	1
0	0	0	0	10	1	1	1	1	30
0	0	0	0	10	1	1	1	1	30
0	0	0	0	10	1	1	1	1	30
1	0	0	0	1	1	1	1	1	30
40	1	1	1	1	1	1	1	1	30
40	1	1	1	1	1	1	1	1	30
40	1	1	1	1	1	1	1	1	30
40	1	1	1	1	1	1	1	1	30
1	40	40	40	40	40	40	40	40	1

Fig. 16 - Atribuição das temperaturas das faces

- Da mancha original é excluída a linha que corresponde à fronteira (ver Fig. 17).

0	0	0	0	0	3	3	3	3	0
0	0	0	0	4	0	0	0	0	2
0	0	0	0	4	0	0	0	0	2
0	0	0	0	4	0	0	0	0	2
0	5	5	5	0	0	0	0	0	2
6	0	0	0	0	0	0	0	0	2
6	0	0	0	0	0	0	0	0	2
6	0	0	0	0	0	0	0	0	2
6	0	0	0	0	0	0	0	0	2
0	1	1	1	1	1	1	1	1	0

Fig. 17 - Matriz da posição das faces

- Cria ainda uma visualização dos pontos na malha que confirma a geometria e o número de pontos usado (ver Fig. 18 - Exemplo de malha de pontos).

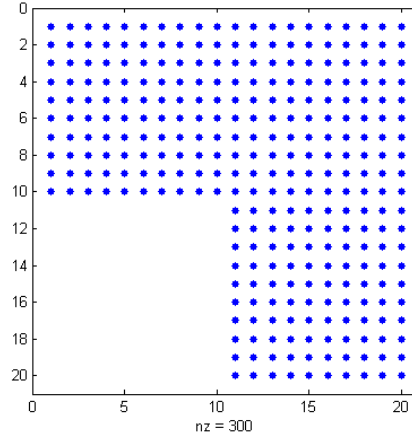


Fig. 18 - Exemplo de malha de pontos

- Numera todos os nós que são variáveis (Fig. 19) exceto os nós de fronteira com condução de calor, esses são tratados mais tarde por outra rotina.

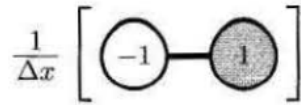
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	17	25	33	41	0
0	0	0	0	0	18	26	34	42	0
0	0	0	0	0	19	27	35	43	0
0	0	0	0	0	20	28	36	44	0
0	1	5	9	13	21	29	37	45	0
0	2	6	10	14	22	30	38	46	0
0	3	7	11	15	23	31	39	47	0
0	4	8	12	16	24	32	40	48	0
0	0	0	0	0	0	0	0	0	0

Fig. 19 - Numeração dos nós

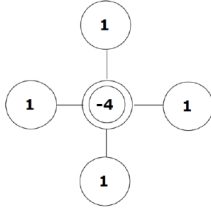
- Com recurso à molécula é criada a matriz “A”.

- O vetor b é igual ao simétrico da temperatura de fronteira que para os pontos junto à fronteira e igual a $q^* \, dx y^2$ para pontos sujeitos a condução. Nestes casos, uma vez que não são consideradas as equações dos pontos sujeitos a condução a matriz “A” é corrigida somando “1” ao coeficiente do ponto central. Desta forma são calculados todos os pontos do “miolo” do modelo tendo em conta as condições de fronteira.

- Uma vez calculado o “miolo” são revistas as faces com condução de calor e calculadas com segundo $T_{\text{fronteira}} = T_{\text{tfrenteira}} - 1 + q^* \, dx y^2$, que vem de uma molécula de diferenças atrás

Fig. 20 - Molécula genérica de diferenças a trás^[1].

- O valor dos cantos é agora a única incógnita. Estes são calculados pela média dos valores vizinhos não nulos, isto é têm uma molécula que se adapta à vizinhança, do tipo



$$* 1/\sum N_{i,j}, N_{i,j} = 1, T_{i,j} \neq 0 \wedge N_{i,j} = 0, T_{i,j} = 0$$

Com esta aproximação todos valores da matriz final são calculados em função das condições de fronteira

5) Resultados “%% Mostrar resultados”.

O resultados são sempre guardados num ficheiro novo com o nome do original e com o a terminação “_res_t” de *resolução térmica*. Deste modo para um só modelo é possível criar várias resoluções sem perder a anterior.

Os resultados são representados num modelo 2D com a forma final do modelo e escala de cores representativa da distribuição de temperaturas.

Limitações e vantagens

Não permite refinação de malha e só permite modelos com faces ortogonais. Não permite também variações de temperatura para uma mesma face. No entanto neste exercício tal não era pedido, por isso não se revelou um problema.

Por outro lado a sua simplicidade faz com que processe muito depressa e com menor esforço computacional, permitindo malhas refinadas, estando limitado pela memória RAM do computador.

Outra desvantagem do código é que não faz a interpolação dos resultados para malhas pouco refinadas, fazendo com que o aspeto da representação gráfica não seja agradável e perceptível. Para malhas refinadas esse efeito é corrigido e as linhas que formam a malha são escondidas para que o gráfico 2D seja mais fácil de ler, e mais apelativo.

Outras observações

Ao fazer a simulação térmica em *Matlab* um problema recorrente era que o número de linhas e de colunas não coincidiam, tal como na Fig. 21, no entanto $dx=dy$ e as dimensões das partes são iguais, logo deviam ambas ter 10 ou 9 pontos por face. Mas não aconteceu no *Matlab*, no *Abaqus* este fenómeno também ocorre, assim mostra a Fig. 22.

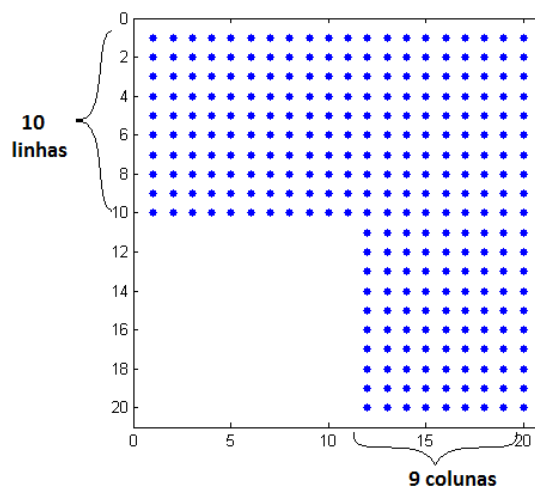


Fig. 21 - Nós em x diferentes de y no Matlab

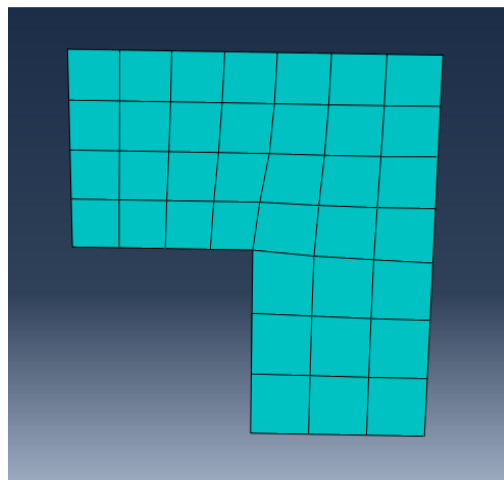


Fig. 22 - Nos diferentes no Abaqus

Em *Matlab* isto deve ocorrer ao ser feita a passagem de coordenadas reais para coordenadas da matriz. Para o *Abaqus* a razão não é conhecida mas deve ter que ver com as aproximações nas distâncias quando as divisões não resultam num número inteiro de pontos. Na imagem anterior, com sete elementos por face (face de maior comprimento), teria que ter meio elemento e por isso aproxima 4 para 3.

No código de isto foi corrigido, é acrescentada uma linha de pontos de modo a criar um número inteiro de pontos. Isto não representa problemas pois os modelos só podem faces ortogonais e assim também não afeta as aproximações e o resultado.

Análise em *Matlab*

A análise em *Matlab* foi feita seguindo os passos já apresentados. Recorrendo à secção 3)

foram feitas varias simulações para diferentes dxy .

Os valores de intervalos utilizados foram: 1, 5, 10, 15, 18, em milímetros o que dá respetivamente 7500, 300, 75, 40, 27, nós (no total, não por face).

Análise em *Abaqus*

Recorrendo ao *software Abaqus* foi também feita uma simulação. Para o MEF foram usados elementos do tipo DC3D8. Em *Abaqus* este tipo de elemento trata de condução sem radiação ou convecção, por isso é adequado para esta resolução.

Por o número de nós estar limitado pela versão *Student*, só foram usados quatro espaçamentos na malha. Estes são 5, 10, 15, e 18, em milímetros.

Resultados e Análise de Resultados

Depois de simular o modelo para as diferentes malhas nos dois softwares foram obtidas as seguintes distribuições de temperaturas (Tabela 3).

Como se pode visualizar em todos os gráficos, os tons de vermelho representam temperaturas mais altas, pois as temperaturas das fronteiras nas regiões correspondentes a essas cores (F1 e F6), são também as mais elevadas. Junto as fronteiras F4 e F5 é visível uma gama de cores mais próxima do azul (ou seja, temperaturas dos 0°C aos 10°C), que mais uma vez está interligado às baixas temperaturas destas fronteiras.

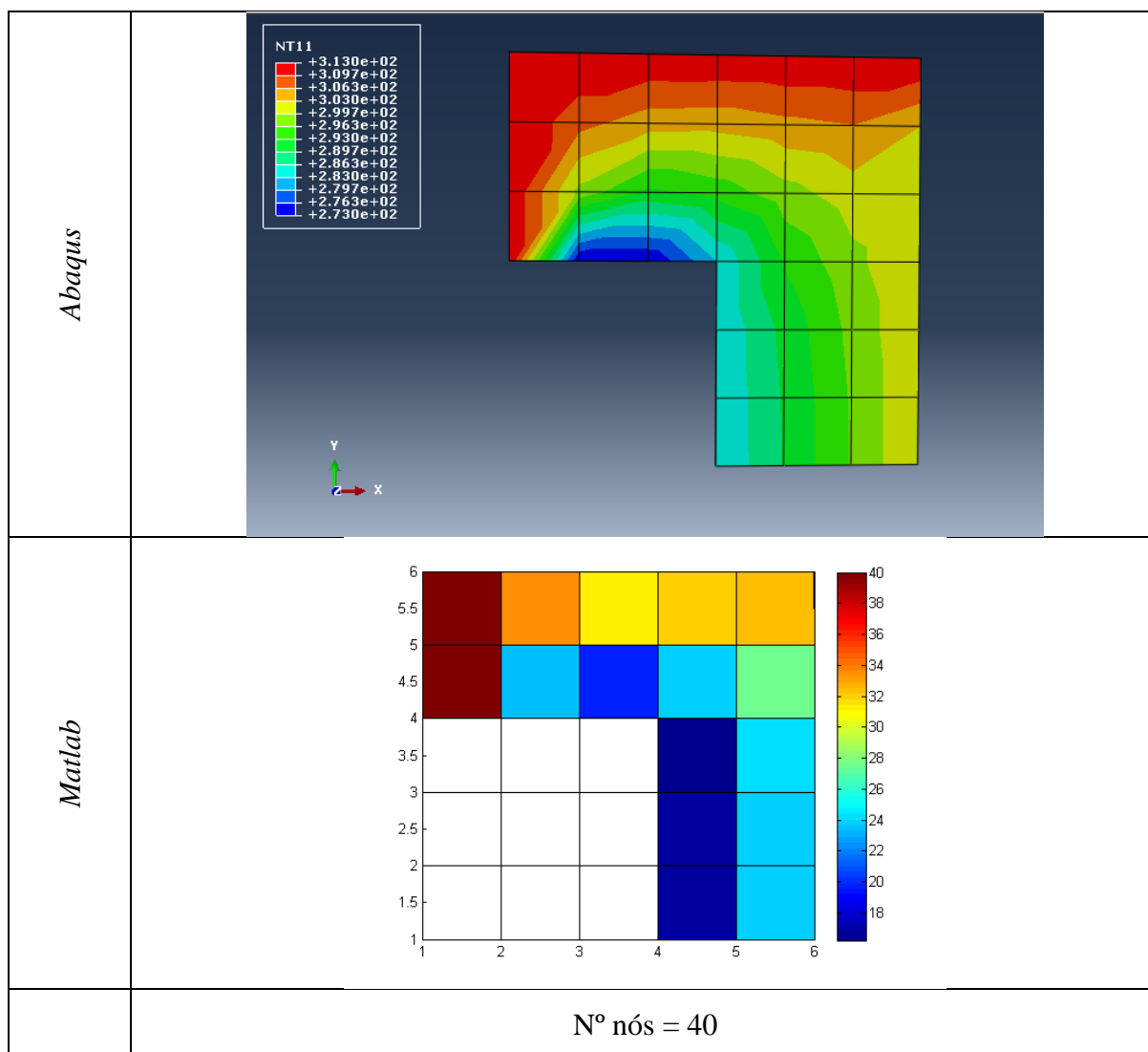
Na fronteira F3 o fluxo é nulo ($q=0$, fronteira adiabática), apresentando-se por isso uma variação de temperatura. A gama de temperaturas varia entre 40°C e 0°C, como seria de esperar.

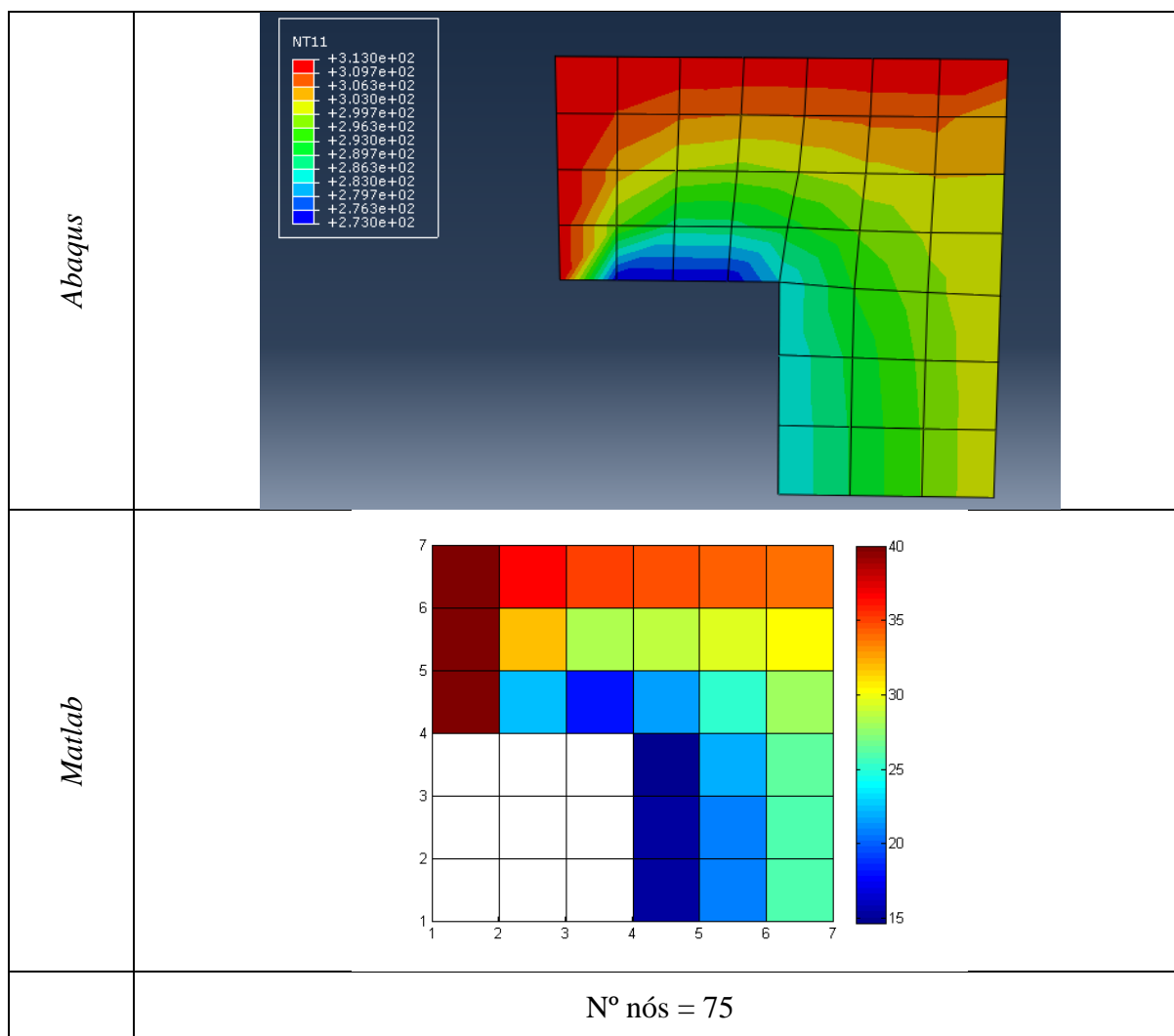
Ao avaliara os resultados com o refinamento da malha podemos concluir que a partir dos 300 nós (altura na qual entra em acção a interpolação dos valores da representação), deixa de haver grande variação na mesma.

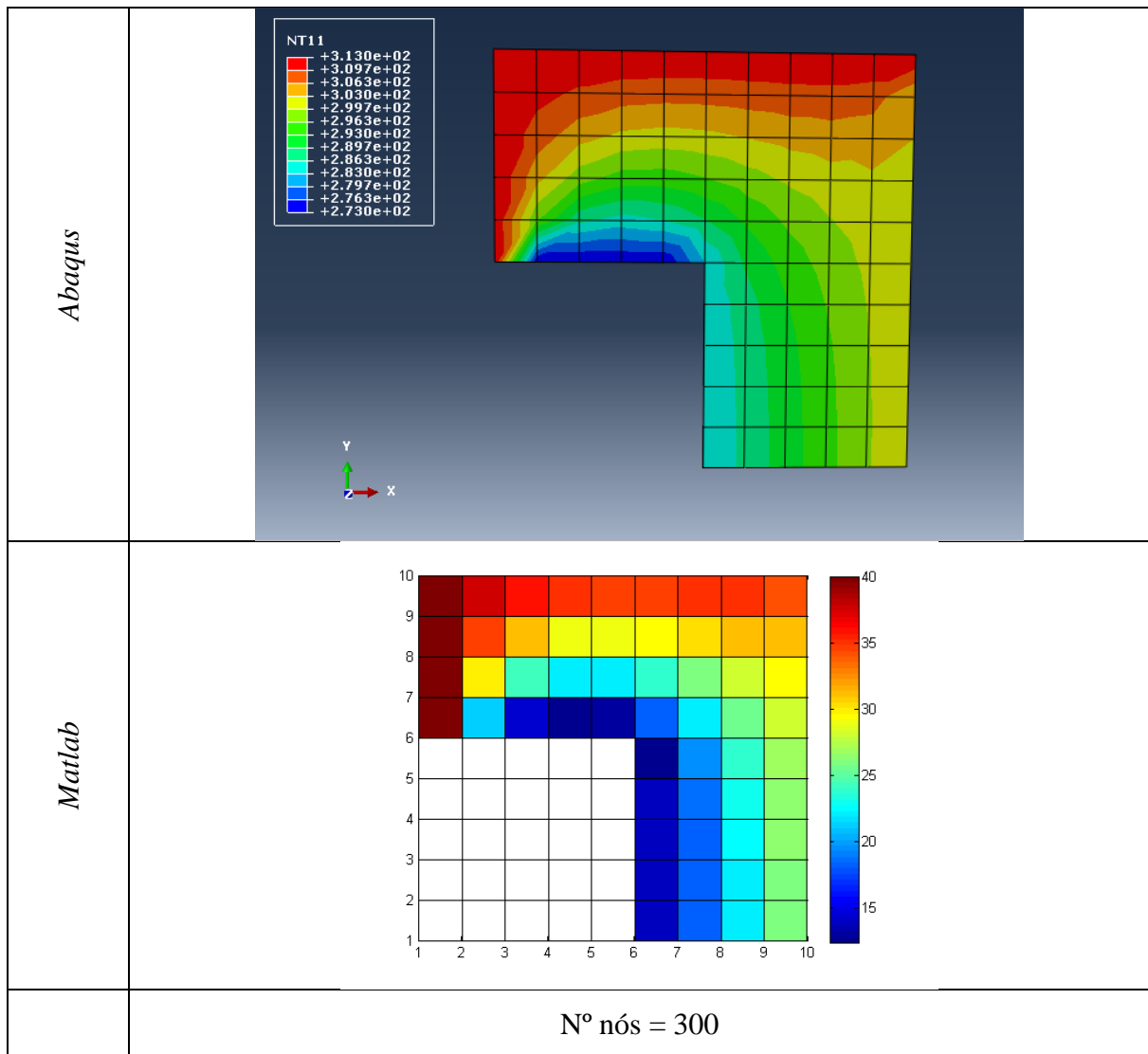
Os resultados obtidos nesta análise, utilizando dois métodos diferentes, são muito semelhantes.

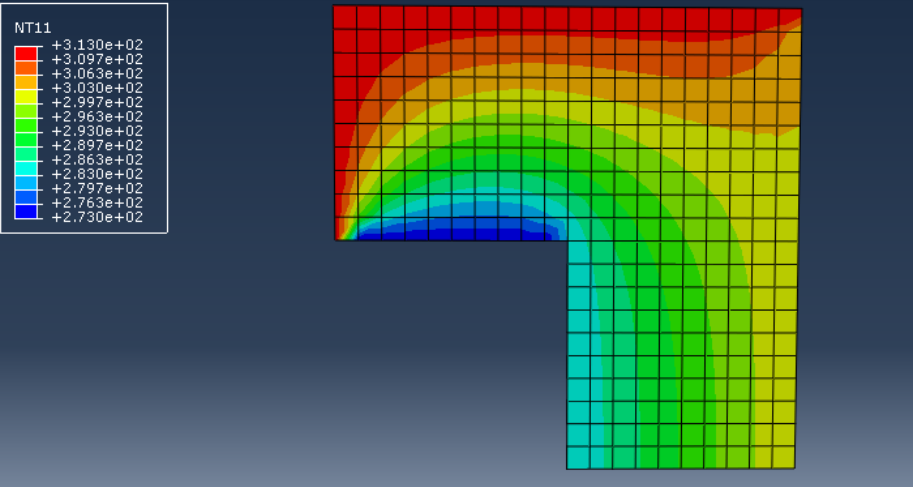
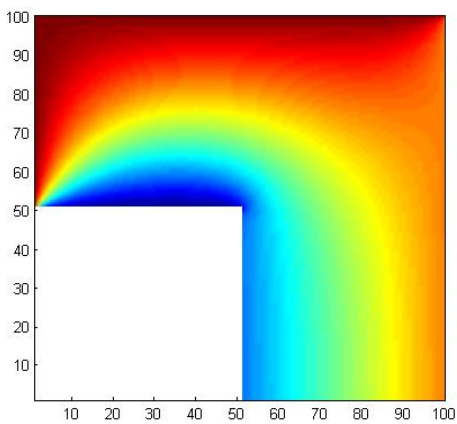
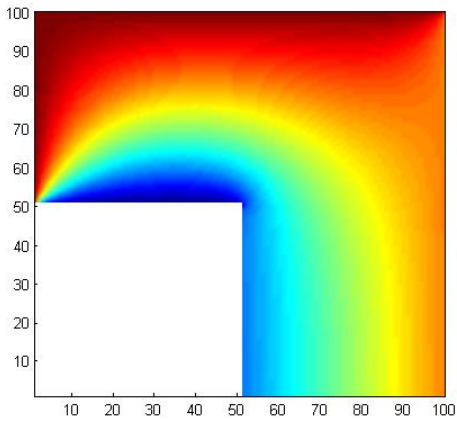
Tabela 3- Distribuição de temperaturas para várias malhas

	Nº nós = 27
--	-------------







<i>Abaqus</i>	
<i>Matlab</i>	
	<p>Nº nós = 7500</p>
<i>Matlab</i>	

Pensamentos finais

Foi com a programação do método das diferenças finitas que nos foi possível aprender como é que os conhecimentos teóricos são aplicáveis na prática. Permitiu-nos testar o quão exata é a solução por este método e que para aplicações simples é viável e não tao difícil de programar. Foi especialmente evidente no exercício estrutural onde o erro embora existe foi muito reduzido.

Podemos dizer que, tendo em conta os resultados obtidos, os objetivos foram cumpridos com sucesso e que apreendemos novos conhecimentos.

Agradecimentos

Queremos deixar um agradecimento ao professor Robert Valente por nos ajudar a recompor este trabalho de maneira a manter um código interessante e com as nossas aproximações, e ao mesmo tempo ter dado as dicas para que o relatório reflita todo o trabalho que tivemos.

Bibliografia

- [1] - Dias, F.T. & Cruz, J.P. & Valente, R.A.F. & Sousa, R.J.A., (2010), Método dos Elementos Finitos – Técnicas de Simulação Numérica em Engenharia.
- [2] - Sousa, R.J.A. & Valente, R.A.F., (2015), Guia de Trabalhos Práticos – Simulação de Processos Tecnológicos;
- [3] - Sousa, R.J.A & Valente, R.A.F., (2015), Tutoriais das aulas práticas – Simulação de Processos Tecnológicos.

Anexos

Código estrutural

```
%% -----
% - Dados - Criar Geometria
%-----
clear all;close all; clc
disp('      MDF Estrutural')
disp('  Função para chapa Rectangular ')
disp('    & dx==dy & q distribuido')
format compact;
%-----
nome = input('Nome do ficheiro de resultados -> ','s');
tipo = 'estrutural2dv2';
%-----
disp('Dimensões da chapa:')
lx = str2double(input('Comp. x (m) -> ','s'));
ly = str2double(input('Comp. y (m) -> ','s'));
t = str2double(input('Esp. t (m) -> ','s'));
disp('Propriedades da chapa:')
E = str2double(input('E (GPa)-> ','s'));
v = str2double(input('v -> ','s'));
qq = str2double(input('Carga q (N)-> ','s'));
D = (E*(10^9)*t^3)/(12*(1-v^2))
%-----
save((strcat(nome,'.mat')), 'tipo', 'nome', 'lx', 'ly', 'E', 't', 'v', 'qq', 'D')
disp('-----')
disp('Resultado guardado')
%-----

%% -----
% - Dados - Criar Con. fronteira
%-----
clear all;close all; clc
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'), '.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'estrutural2dv2'); warning('Ficheiro inválido!'); break; end
%-----
ok = 0;
ok1 = 1;
```

```

ta = '----';
while ok ~= 1
    while ok1 <= 4
        disp('    F1    ')
        disp('    ----- ')
        disp(' |      | ')
        disp('F4 |      | F2')
        disp(' |      | ')
        disp('    ----- ')
        disp('    F3    ')
        disp('-----')
        disp(sprintf('Face %d -> %s',1,ta(1)))
        disp(sprintf('Face %d -> %s',2,ta(2)))
        disp(sprintf('Face %d -> %s',3,ta(3)))
        disp(sprintf('Face %d -> %s',4,ta(4)))
        disp('-----')
        disp(sprintf('Face %d',ok1))
        disp('Tipo de apoio (Encastrado -> e | Apoio simples -> a | Livre -> l) ')
        disp(ok1)
        ta(ok1) = input('Apoio -> ','s');
        if ta(ok1) == 'e' || ta(ok1) == 'a' || ta(ok1) == 'l'
            disp(sprintf('Face %d -> %s',ok1,ta(ok1)))
            ok1 = ok1 + 1;
            pause(0.8)
            clc
            disp('-----')
        else
            disp('Repetir cond. não válida')
            ta(ok1) = '-';
        end
    end
end
if ta(1) == ta(2) && ta(1) == ta(3) && ta(1) == ta(4)
    simetria = 4; % simetrica em 4 dir
    ok = 1;
    simetria2 = 'Cond. fronteira simetricas em x e y';
    disp('Cond. fronteira simetricas em x e y')
elseif ta(4) == ta(2)
    simetria = 3; % simetria horizontal
    ok = 1;
    simetria2 = 'Cond. fronteira simetricas em y';
    disp('Cond. fronteira simetricas em y')
elseif ta(1) == ta(3)

```

```

    simetria = 2; % simetria vertical
    ok = 1;
    simetria2 = 'Cond. fronteira simetricas em x';
    disp('Cond. fronteira simetricas em x')
else
    simetria = 1; % nao simetrico
    ok = 1;
    simetria2 = 'Con. fronteira não simetricas';
    disp('Con. fronteira não simetricas')
end
pconf = 0; % variável para confirmar que os apoios estão ok
for k = 1:4
    if ta(k) == 'e'; pconf = pconf + 2; elseif ta(k) == 'a'; pconf = pconf + 1; end
end
if pconf <= 1; ok = 0;
    simetria = 0;
    clc
    disp('As extremidades não podem estar todas livres')
    disp('Pelo menos uma tem que estar encastrada ou duas apoiadas')
    disp('Repetir')
end
end
%-----
save((strcat(nome,'.mat')),'tipo','nome','lx','ly','E','t','v','qq','D','ta','simetria','simetria2')
disp('-----')
disp('Resultado guardado')
%-----

%% -----
% - Dados - N° pontos
%-----

clear all;close all; clc
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'estrutural2dv2'); warning('Ficheiro inválido!'); break; end
%-----
disp(sprintf('%s com: dimx = %d ; dimy = %d',simetria2,lx,ly))

dxy = str2double(input('dxy (m) -> ','s'));
if simetria == 4;
    npx = round((lx/dxy)/2);

```



```

    npy = round((ly/dxy)/2);
    npt = npx*npv;
    disp(sprintf('Nº elementos = %d',npt*4))
elseif simetria == 3;
    npx = round((lx/dxy)/2);
    npy = round((ly/dxy));
    npt = npx*npv;
    disp(sprintf('Nº elementos = %d',npt*2))
elseif simetria == 2;
    npx = round((lx/dxy));
    npy = round((ly/dxy)/2);
    npt = npx*npv;
    disp(sprintf('Nº elementos = %d',npt*2))
elseif simetria == 1;
    npx = round((lx/dxy));
    npy = round((ly/dxy));
    npt = npx*npv;
    disp(sprintf('Nº elementos = %d',npt))
end
%-----
save((strcat(nome,'.mat')), 'tipo','nome','lx','ly','E','t','v','qq','D','ta','simetria','dxy','npx','np
y','npt','D','qq','ta','simetria2')
disp('-----')
disp('Resultado guardado')
%-----

%% -----
% - Processamento -
%-----

clear all;close all; clc
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'estrutural2dv2'); warning('Ficheiro inválido!'); break; end
%-----
tic
%-----
% criar condições de fronteira e simetrias
tta = ones(1,4);
%-----
for k = 1:4;
    if ta(k) == 'e'

```

```

    tta(k) = 1;
elseif ta(k) == 'a'
    tta(k) = -1;
elseif ta(k) == 'l'
    tta(k) = 0;
end
end

switch simetria
case 4 %simetria 4 dir (horizontal e vertical)
    h = zeros(npy+2,npx+2);
    h(3:end,3:end) = 1;
    h1 = h;
    h0 = h;
    h(h==1)=1:sum(sum(h));

    h(3:end,1) = h(3:end,3); %fronteira 4
    h(1,3:end) = h(3,3:end); %fronteira 1

    h1(:,1) = tta(4);
    h1(1,:) = tta(1);

    h2 = fliplr(h(:,1:end-1));
    h3 = flipud(h(1:end-1,:));
    h4 = flipud(h2(1:end-1,:));

    h12 = fliplr(h1(:,1:end-1));
    h13 = flipud(h1(1:end-1,:));
    h14 = flipud(h12(1:end-1,:));

    h02 = fliplr(h0(:,1:end-1));
    h03 = flipud(h0(1:end-1,:));
    h04 = flipud(h02(1:end-1,:));

    H0 = [h0 h02; h03 h04];
    H = [h h2; h3 h4];
    hh = [h1 h12; h13 h14];

    s = sum(sum(h0));
    pos = (find(h0==1));

case 3 %simetria horizontal

```

```

h = zeros(npy+4,npx+2);
h(3:end,3:end) = 1;
h1 = h;
h0 = h;
h(h==1)=1:sum(sum(h));

h(3:end-2,1) = h(3:end-2,3); %fronteira 4
h(1,3:end) = h(3,3:end); %fronteira 1
h(end,3:end) = h(end-2,3:end); %fronteira 3

h1(:,1) = tta(4);
h1(1,:) = tta(1);
h1(end,:) = tta(3);

h2 = fliplr(h(:,1:end-1));

h12 = fliplr(h1(:,1:end-1));

h02 = fliplr(h0(:,1:end-1));

H0 = [h0 h02];
H = [h h2];
hh = [h1 h12];

s = sum(sum(H0));
pos = (find(H0==1)');

case 2 % simetria vertical
h = zeros(npy+2,npx+4);
h(3:end,3:end) = 1;
h1 = h;
h0 = h;
h(h==1)=1:sum(sum(h));

h(3:end,1) = h(3:end,3); %fronteira 4
h(1,3:end-2) = h(3,3:end-2); %fronteira 1
h(end,3:end) = h(end-2,3:end); %fronteira 2

h1(:,1) = tta(4);
h1(1,:) = tta(1);
h1(:,end) = tta(2);

```

```

h3 = flipud(h(1:end-1,:));

h13 = flipud(h1(1:end-1,:));

h03 = flipud(h0(1:end-1,:));

H0 = [h0; h03];
H = [h; h3];
hh = [h1; h13];

s = sum(sum(H0));
pos = (find(H0==1)');

case 1          % sem simetria
    h = zeros(npy+4,npx+4);
    h(3:end-2,3:end-2) = 1;
    h1 = h;
    h0 = h;
    h(h==1)=1:sum(sum(h));

    h(3:end,1) = h(3:end,3); %fronteira 4
    h(1,3:end-2) = h(3,3:end-2); %fronteira 1
    h(end,3:end) = h(end-2,3:end); %fronteira 2

    h1(:,1) = tta(4);
    h1(1,:) = tta(1);
    h1(:,end) = tta(2);
    h1(end,:) = tta(3);

    H0 = h0;
    H = h;
    hh = h1;

    s = sum(sum(h0));
    pos = (find(h0==1)');
end
%-----
figure
spy(H0)
clear H0 h0 h02 h03 h04 h1 h12 h13 h14
%-----

```

```

mat1 = [0 0 1 0 0
        0 2 -8 2 0
        1 -8 20 -8 1
        0 2 -8 2 0
        0 0 1 0 0];

%-----
res = zeros(s);
rrr = (qq*(dxy^4)/D);
resb = repmat(rrr,s,1);
%-----
A = zeros(s);
%-----
for k = 1:numel(pos)
    [I,J] = ind2sub(size(h),pos(k));
    matpos = [0      0      H(J-2,I) 0      0
              0      H(J-1,I-1) H(J-1,I) H(J-1,I+1) 0
              H(J,I-2) H(J,I-1)  H(J,I)  H(J,I+1)  H(J,I+2)
              0      H(J+1,I-1) H(J+1,I) H(J+1,I+1) 0
              0      0      H(J+2,I) 0      0];

    matval = [0      0      hh(J-2,I) 0      0
              0      hh(J-1,I-1) hh(J-1,I) hh(J-1,I+1) 0
              hh(J,I-2) hh(J,I-1)  hh(J,I)  hh(J,I+1)  hh(J,I+2)
              0      hh(J+1,I-1) hh(J+1,I) hh(J+1,I+1) 0
              0      0      hh(J+2,I) 0      0];

    pos1 = (find(matpos>0)');
    for kkk = 1:numel(pos1())
        kk = pos1(kkk);
        A(matpos(13),matpos(kk)) = A(matpos(13),matpos(kk))+mat1(kk)*matval(kk);
    end
end
disp('Resultados - RR')
RR = A\resb;
disp('Max(RR)')
R = max(abs(RR))
disp('Min(RR)')
R = min(abs(RR))
%-----
chapa = H(2:end-1,2:end-1);
for k = 1:numel(RR)
    chapa(chapa==k) = RR(k);
end

```

```

toc
%-----
tipo = 'estrutural2dv2_res';
save(strcat(nome,'_res_e','.mat'),'tipo','chapa');
disp('-----')
disp('Resultado guardado')
%-----

%% Mostrar resultados
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'estrutural2dv2_res'); warning('Ficheiro inválido!'); break; end
%-----

figure
surf(chapa)
shading interp
axis([0 size(chapa,2) 0 size(chapa,1) -3*R 0.00001])
%-----

```

Código térmico

```

%% -----
% - Dados - Criar Geometria
%-----
clear all;close all;format compact; clc
%-----
nome = input('Nome do ficheiro de resultados -> ','s');
tipo = 'termico2dv2';
%-----
disp('Criar Geometria')
nlados = str2double(input('Enter -> N° lados: ','s'));
%-----
fig1 = figure();
%-----
k = 1;
keycheck = "";
ptslim = [];
clc;
while k < nlados+1
    keycheck="";

```

```

ok1 = false;
disp('Coordenadas sequenciais dos vertices')
disp(strcat('----- Ponto: ',num2str(k)))
ptslim(1,k) = str2double(input('Enter -> Coordenada x: ','s'));
ptslim(2,k) = str2double(input('Enter -> Coordenada y: ','s'));
plot(ptslim(1,:),ptslim(2,:),'bx--')
disp('-----')
disp(strcat(['x:','y:'],num2str(ptslim)))
keycheck = input('Continue[y]/repeat[n]: ','s');
clc;
disp(strcat(['x:','y:'],num2str(ptslim)))
disp(strcat('----- Face: ',num2str(k)));
axis equal
if (keycheck == sprintf('y'))
    k = k + 1;
else
    clc
    disp('repetir/erro')
end
end
ptslim(:,nlados+1) = ptslim(:,1);
plot(ptslim(1,:),ptslim(2,:))
%axis([ (min(ptslim(1,:))-20),(max(ptslim(1,:))+20),(min(ptslim(2,:))-
20),(max(ptslim(1,:))+20)]);
for kk = 1:nlados

text(((ptslim(1,kk)+ptslim(1,kk+1))/2)+2,((ptslim(2,kk)+ptslim(2,kk+1))/2)+4,sprintf('F%d'
,kk));
end
axis auto
axis equal
grid on
pause(1)
clc;
%-----
save((strcat(nome,'.mat')),'nome','tipo','ptslim','nlados');
disp('-----')
disp('Resultado guardado')
%-----

%% -----
% - Dados - Criar Con. fronteira

```

```

%-----
clear all;close all;format compact; clc
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'termico2dv2'); warning('Ficheiro inválido!'); break; end
%-----
tface = zeros(1,nlados);
tcond = "";
tlado = "";
%-----
plot(ptslim(1,:),ptslim(2,:))
axis auto
%axis([(min(ptslim(1,:))-20),(max(ptslim(1,:))+20),(min(ptslim(2,:))-
20),(max(ptslim(2,:))+20)]);
for kk = 1:nlados

text(((ptslim(1,kk)+ptslim(1,kk+1))/2)+2,((ptslim(2,kk)+ptslim(2,kk+1))/2)+4,sprintf('F%d'
,kk));
end
axis equal
grid on

k = 1;
while k <= nlados
    keycheck="";
    clc;
    disp(strcat('----- Face: ',num2str(k)));
    disp('Cond. fronteira: T°C -> "t" / Q -> "q"');
    tcond(k) = input('Enter -> ','s');
    tface(k) = str2double(input('Valor: ','s'));
    disp('Lado da fronteira: cima -> "c" / baixo -> "b" / direita -> "d" / esquerda -> "e");
    tlado(k) = input('Dir: ','s');
    keycheck = input('Continue[y]/repeat[n]: ','s');
    if (keycheck == sprintf('y')) && ((tcond(k) == 't') || (tcond(k) == 'q')) %&&
(((tlado(k) == 'c') || (tlado(k) == 'b') || (tlado(k) == 'e') || (tlado(k) == 'd')) )

text(((ptslim(1,k)+ptslim(1,k+1))/2)+2,((ptslim(2,k)+ptslim(2,k+1))/2)+4,sprintf('F%d: %s
= %d',k,tcond(k),tface(k)));
    k = k + 1;
else
    clc

```



```

        disp('repetir/erro')
    end
end
%-----
save((strcat(nome,'.mat')),nome,'tipo','ptslim','nlados','tcond','tface','tlado');
disp('-----')
disp('Resultado guardado')
%-----
%% -----
% - Dados - Nº pontos
%-----
clear all;close all;format compact; clc
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'termico2dv2'); warning('Ficheiro inválido!'); break; end
%-----

matriz1 = [];
disp('Resolução')
ok2 = 'n';
a = 0;
b = 0;
while (a < 3) || (b < 3)
    while ok2 ~= 'y'
        disp('Dim de intervalos (dx)e(dy)')
        dx = str2double(input('dxy: ','s'));
        % disp('Dim de intervalos (dy)')
        % dy = str2double(input('dy: ','s'));
        dy=dx;
        ok2 = input('Continue/repeat [y]/[n]: ','s');
    end
    dimxx = round(max(ptslim(1,:))-min(ptslim(1,:)));
    dimyy = round(max(ptslim(2,:))-min(ptslim(2,:)));
    npx = round(dimxx/dx);
    npy = round(dimyy/dy);

    ptslimno(1,:) = round((ptslim(1,:)+dx)/dx);
    ptslimno(2,:) = round((ptslim(2,:)+dy)/dy);

    a = abs(ptslimno(1,1:end-1)-ptslimno(1,2:end));
    b = abs(ptslimno(2,1:end-1)-ptslimno(2,2:end));
end

```

```

    a = min(a(a~=0)); if a < 3;disp('!-----!'); disp('dxy Muito grande não há
pontos suficientes');ok2 = 'n';disp('!-----!');end;
    b = min(b(b~=0)); if b < 3;disp('!-----!'); disp('dxy Muito grande não há
pontos suficientes');ok2 = 'n';disp('!-----!');end;
end
h = zeros(npy,npx);
h = poly2mask(ptslimno(1,:), max(ptslimno(2,:))-ptslimno(2,:),npv,npx); %criar
matrizes de valores 1 para pontos de inconita
h = imdilate(h,[0,0,0;1,1,0;1,1,0]);
figure
spy(h)

%-----
save((strcat(nome,'.mat')),'nome','tipo','ptslim','nlados','tcond','tface','tlado','dx','dy','npx','
npv');
disp('-----')
disp('Resultado guardado')
%-----

%% -----
% - Processamento -
%-----
clear all;close all;format compact; clc
%-----
tic
%-----
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))
if exist('tipo','var')~= 1; warning('Ficheiro inválido!'); break; end;
if ~strcmp(tipo,'termico2dv2'); warning('Ficheiro inválido!'); break; end
%-----
ptslimno(1,:) = round((ptslim(1,.)+dx)/dx);
ptslimno(2,:) = round((ptslim(2,.)+dy)/dy);
h = zeros(npy,npx);
H = h; % matriz com valor dos pontos; (pi+exp(1))*10^10 para inconita, outro valor
para t
h = poly2mask(ptslimno(1,:),max(ptslimno(2,:))-ptslimno(2,:),npv,npx); %criar
matrizes de valores 1 para pontos de inconita %
h = imdilate(h,[0,0,0;1,1,0;1,1,0]);
%-----
% Mostrar figura representativa da malha de pontos
%-----
figure

```

```

spy(h)
%-----
h = flipud(h); % corrigir modelo para as coordenadas de matrix
h1 = h;
h = im2double(h);
%-----
% Corrigir coordenadas para não excederem o tamanho do modelo
%-----
ptxx = ptslimno(1,:);
ptyy = ptslimno(2,:);
ptxx(ptxx>npx/2) = ptxx(ptxx>npx/2)-1;
ptyy(ptyy>np/2) = ptyy(ptyy>np/2)-1;
%-----
% Construir as condições de fronteira
%-----
h(h==1) = (pi+exp(1))*10^10;
hh = zeros(npy,npx);
for k = 1:nlados
    xx = (min(ptxx(k),ptxx(k+1)):max(ptxx(k),ptxx(k+1)));
    yy = (min(ptyy(k),ptyy(k+1)):max(ptyy(k),ptyy(k+1)));
    h(yy,xx) = tface(k);
    hh(yy,xx) = k;
end
for k = 1:nlados
    h(ptyy(k),ptxx(k)) = 2*(pi+exp(1))*10^10;
    hh(ptyy(k),ptxx(k)) = 0;
end
H(h==(pi+exp(1))*10^10) = 1:sum(sum((h==(pi+exp(1))*10^10))); % matriz de
posições das inconitas
s = max(max(H)); % numero de elementos
h(h==(pi+exp(1))*10^10) = 1;
h(h==2*(pi+exp(1))*10^10) = 1;
%-----
% Criar vector b e matrix A ; de Ax=b
%-----
b = (zeros(s,1));
A = (zeros(s));
pos = find(H~=0); % posições das inconitas na mariz
mat1 = [0 1 0
        1 -4 1
        0 1 0]; % molecula para dx==dy

```

```

for k = 1:numel(pos)
    [J,I] = ind2sub(size(h),pos(k));
    matpos = [0      H(J-1,I) 0
              H(J,I-1) H(J,I) H(J,I+1)
              0      H(J+1,I) 0];

    matval = [0      h(J-1,I) 0
              h(J,I-1) h(J,I) h(J,I+1)
              0      h(J+1,I) 0];

    matbvl = [0      hh(J-1,I) 0
              hh(J,I-1) hh(J,I) hh(J,I+1)
              0      hh(J+1,I) 0];

    pos1 = (find(matpos>0));

    for kkk = 1:numel(pos1)
        kk = pos1(kkk);
        A(matpos(5),matpos(kk)) = A(matpos(5),matpos(kk))+ mat1(kk)*matval(kk);
    end

    for kkk = 1:9
        if matbvl(kkk) ~= 0 &&matpos(kkk)== 0
            switch tcond(matbvl(kkk))
                case 't'
                    bcond = matval(kkk);
                case 'q'
                    bcond = matval(kkk)*(dx^2);
                    A(matpos(5),matpos(5)) = A(matpos(5),matpos(5))+ 1;
            end
            b(matpos(5)) = b(matpos(5)) - bcond;
        end
    end
    end
    % pause
end
disp('Resultados Cálculados - RR')
RR = A\b;
disp('Max(RR)')
R1 = max(abs(RR))
disp('Min(RR)')
R2 = min(abs(RR))
clear A b

```

```

%-----
% Corrigir valores dos cantos e das faces com condução
%-----
chapa = h;
for k = 1:numel(RR)
    chapa(H==k) = RR(k);
end
clear H
for k = 1:nlados
    if tcond(k) == 'q'
        pos2 = find(hh==k);
        for kk = 1:numel(pos2)
            [J,I] = ind2sub(size(hh),pos2(kk));
            switch tlado(k)
                case 'c'
                    chapa(J,I) = chapa(J-1,I)+(tface(k)*(dy^2));
                case 'b'
                    chapa(J,I) = chapa(J+1,I)+(tface(k)*(dy^2));
                case 'd'
                    chapa(J,I) = chapa(J,I-1)+(tface(k)*(dx^2));
                case 'e'
                    chapa(J,I) = chapa(J,I+1)+(tface(k)*(dx^2));
            end
        end
    end
end
chapa2 = zeros(size(chapa)+2);
chapa2(2:end-1,2:end-1)=chapa;
for k = 1:nlados
    mo = [0, chapa2(ptyy(k),ptxx(k)+1), 0; chapa2(ptyy(k)+1,ptxx(k)), 0,
chapa2(ptyy(k)+1,ptxx(k)+2); 0, chapa2(ptyy(k)+2,ptxx(k)+1), 0];
    chapa2(ptyy(k)+1,ptxx(k)+1) = (mo(4)+mo(6)+mo(2)+mo(8))/(sum(sum(mo~=0)));
end

chapa = chapa2(2:end-1,2:end-1);
clear chapa2
disp('Resultados Globais - chapa')
disp('Max(chapa)')
Rmx = max(max(abs(RR)))
disp('Min(chapa)')
Rmn = min(min(abs(RR)))
chapa(h1==0) = nan;

```

```
%-----  
toc  
%-----  
tipo = 'termico2dv2_res';  
save(strcat(nome,'_res_t','.mat'),'tipo','chapa');  
disp('-----')  
disp('Resultado guardado')  
%-----  
  
%% -----  
% Mostrar resultados  
%-----  
clear all;close all;format compact; clc  
%-----  
load(strcat(input('Nome do ficheiro de resultados -> ','s'),'.mat'))  
if exist('tipo','var')== 1; warning('Ficheiro inválido!'); break; end;  
if ~strcmp(tipo,'termico2dv2_res'); warning('Ficheiro inválido!'); break; end  
%-----  
  
figure  
pcolor(chapa)  
colorbar  
if ((size(chapa,1)*size(chapa,2))>1000)  
shading interp  
end
```