

DESENVOLVIMENTO EM MATLAB DE UM PROGRAMA DE ELEMENTOS FINITOS PARA PERCOLAÇÃO

LUÍS FILIPE MADEIRA RIBEIRINHO SOARES

Dissertação submetida para satisfação parcial dos requisitos do grau de
MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM GEOTECNIA

Professor Doutor José Manuel Mota Couto Marques

JUNHO DE 2010

MESTRADO INTEGRADO EM ENGENHARIA CIVIL 2009/2010

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

Fax +351-22-508 1446

✉ miec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

Fax +351-22-508 1440

✉ feup@fe.up.pt

🌐 <http://www.fe.up.pt>

Reproduções parciais deste documento serão autorizadas na condição que seja mencionado o Autor e feita referência a *Mestrado Integrado em Engenharia Civil - 2009/2010 - Departamento de Engenharia Civil, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2009.*

As opiniões e informações incluídas neste documento representam unicamente o ponto de vista do respectivo Autor, não podendo o Editor aceitar qualquer responsabilidade legal ou outra em relação a erros ou omissões que possam existir.

Este documento foi produzido a partir de versão electrónica fornecida pelo respectivo Autor.

A meus Pais

*A adversidade desperta em nós capacidades que,
em circunstâncias favoráveis, teriam ficado adormecidas*

Horácio

AGRADECIMENTOS

O autor expressa os mais sinceros e profundos agradecimentos ao Prof. Doutor José Couto Marques, coordenador deste projecto, por ter possibilitado explorar um tema pelo qual o autor nutre um gosto especial, por todos os elementos fornecidos que serviram de base ao programa desenvolvido, pela clarividência com que ajudou à compreensão de algumas temáticas, mas principalmente pela disponibilidade sem limites com que abraçou este projecto.

Para além disso, o autor agradece também à sua família todo o apoio demonstrado e em especial ao seu pai, amante de programação, pelo incentivo acrescido.

RESUMO

Este trabalho tem como objectivo o desenvolvimento, no ambiente Matlab, de um programa de elementos finitos, destinado a problemas de percolação em regime permanente, sejam eles confinados ou não e dos respectivos módulos auxiliares de pré e pós-processamento. O Matlab dispõe de uma ampla variedade de técnicas para representação gráfica de dados, nomeadamente ferramentas interactivas que permitem manipular os gráficos de forma a alcançar resultados que revelem o máximo de informação possível da disponível nos dados. Para além disso, permite também animar partículas de corrente, contribuindo assim de forma significativa para melhorar a percepção do movimento descrito pelo escoamento, comparativamente com a informação fornecida pelo mapa de cores e pela representação vectorial da velocidade.

Optou-se por uma divisão em quatro capítulos. O primeiro apresenta os conceitos da Mecânica dos Solos relativos à percolação, passando pela lei de Darcy, que relaciona o potencial com a velocidade de percolação, e pela equação da continuidade, que rege o escoamento de um fluido num meio poroso numa perspectiva de conservação da massa. Para além disso, definem-se também as condições de fronteira a impor.

No segundo capítulo, abordam-se as ferramentas matemáticas que permitem resolver de forma automática as equações apresentadas no primeiro capítulo. Para isso, recorre-se ao método dos elementos finitos, fazendo uma breve referência a diferentes técnicas existentes para resolver equações diferenciais com o objectivo de justificar a escolha feita. De seguida, aplica-se o método às equações da percolação com a função de potencial como base, introduzem-se os elementos paramétricos e a integração numérica pelo método de Gauss-Legendre e, por fim, expõe-se a forma de calcular os valores da função de corrente, utilizando as velocidades calculadas.

O terceiro capítulo guia os utilizadores na introdução dos dados e descreve e explica a sequência das operações do programa desenvolvido. A terceira parte deste capítulo destina-se a explorar a visualização de resultados.

O quarto capítulo reúne um conjunto de problemas de referência que exploram as potencialidades do programa e servem para comparar as soluções obtidas com as de outros programas de forma a validar o código desenvolvido.

Com este trabalho procurou-se tirar partido de uma linguagem moderna que melhorou e simplificou as tarefas de pré e pós-processamento e que se espera que despertem interesse para futuros desenvolvimentos de novas funcionalidades que permitam auxiliar cada vez mais a compreensão dos fenómenos da percolação.

PALAVRAS-CHAVE: elementos finitos, percolação, Matlab, regime permanente, potencial.

ABSTRACT

The objective of this work is the development in Matlab of a finite element program for steady state groundwater seepage in both confined and unconfined domains, complemented by pre- and post-processing auxiliary modules. Matlab provides a wide variety of techniques to display data visually, including interactive tools for manipulating graphics in order to achieve results that reveal most information available in the data. In addition, it also allows to display stream particles in motion, thereby contributing significantly to improve the perception of groundwater flow, rather than only by colour maps and velocity vectors.

This thesis is divided into four chapters. The first presents the concepts of Soil Mechanics related to groundwater seepage, through Darcy's law, which relates the head pressure with the velocity of water flow, and the continuity equation, which controls the conservation of mass in incompressible fluid flow. In addition the boundary conditions are also defined.

In the second chapter the mathematical tools that can solve automatically the equations presented in the first one are discussed. To this end the finite element method is introduced, after a brief reference to alternative techniques for solving differential equations that justifies the choice made. The finite element method is then applied to the groundwater seepage equations in potential function formulation, parametric elements and Gauss-Legendre numerical integration are discussed and, lastly, the way to compute the stream function values, using velocity results, is established.

The third chapter guides users in the data input and describes and explains, step by step, the code operation sequence. The third part of this chapter explores the visualization of results.

The fourth chapter validates the software developed by presenting a set of reference problems that explore the capabilities of the program and compare the solutions obtained with those given by other programs.

A modern computational language has been employed to improve and simplify the pre- and post-processing tasks and also with the expectation of arousing interest in future developments of new features that may contribute to provide a sounder understanding of groundwater seepage phenomena.

KEYWORDS: Finite elements, seepage, Matlab, steady state, potencial.

ÍNDICE GERAL

AGRADECIMENTOS	i
RESUMO	iii
ABSTRACT	v
1. ÁGUA NOS SOLOS	1
1.1. INTRODUÇÃO	1
1.2. LEI DE DARCY	1
1.3. VALIDADE DA LEI DE DARCY	2
1.4. EQUAÇÕES DIFERENCIAIS GOVERNATIVAS DO ESCOAMENTO DE ÁGUA EM MEIOS POROSOS	3
1.4.1. GENERALIZAÇÃO DA LEI DE DARCY	3
1.4.2. EQUAÇÃO DA CONTINUIDADE	4
1.4.2.1. Desenvolvimento da Expressão	4
1.4.2.2. Função Potencial	5
1.4.2.3. Função de Corrente	7
2. DESENVOLVIMENTO NUMÉRICO	11
2.1. INTRODUÇÃO	11
2.2. ELEMENTOS FINITOS	12
2.2.1. MÉTODO DOS RESÍDUOS PESADOS	12
2.2.2. MÉTODO DE GALERKIN	13
2.2.3. RESÍDUO NA FRONTEIRA	13
2.2.4. PROPRIEDADES DAS FUNÇÕES DE APROXIMAÇÃO	14
2.3. APLICAÇÃO DO MÉTODO DOS ELEMENTOS FINITOS	14
2.3.1. FUNÇÃO POTENCIAL	14
2.3.2. ELEMENTOS PARAMÉTRICOS	17
2.3.3. INTEGRAÇÃO EM RELAÇÃO ÀS COORDENADAS LOCAIS	22
2.3.4. INTEGRAÇÃO NUMÉRICA	23
2.3.5. TERMO INDEPENDENTE	24
2.3.6. FUNÇÃO DE CORRENTE	25

3. PROGRAMA DE CÁLCULO AUTOMÁTICO	27
3.1. INTRODUÇÃO	27
3.2. PRÉ PROCESSAMENTO	27
3.2.1. DIMENSIONAMENTO DA PÁGINA E DEFINIÇÃO DO PROBLEMA	27
3.2.2. GERAÇÃO DO FICHEIRO DE DADOS	29
3.2.2.1. Desenho da malha	29
3.2.2.2. Atribuição de propriedades	32
3.3. CÁLCULO	35
3.3.1. INTRODUÇÃO DA TOPOLOGIA DA MALHA	35
3.3.1.1. Leitura dos Parâmetros de Controlo	35
3.3.1.2. Leitura da Topologia da Malha	35
3.3.1.3. Cálculo da Coordenadas dos Pontos de Gauss	36
3.3.2. CÁLCULO DOS POTENCIAIS NOS NÓ	36
3.3.2.1. Leitura das Condições Fronteira de Dirichlet	36
3.3.2.2. Formação da Matriz de Coeficientes Global	36
3.3.2.3. Formação do Termo Independente para a Solução em Termos de Potencial	39
3.3.2.4. Resolução do Sistema de Equações	39
3.3.2.5. Verificação do Tipo de Escoamento	40
3.3.2.6. Tratamento dos Escoamentos Não Confinados	40
3.3.2.7. Cálculo da Velocidade e dos Caudais não Equilibrados	41
3.3.3. CÁLCULO DA FUNÇÃO DE CORRENTE NOS NÓS	43
3.4. PÓS PROCESSAMENTO	43
3.4.1. BARRA DE COMANDOS	43
3.4.2. ÁREA DE DESENHO	44
4. VALIDAÇÃO	45
4.1. INTRODUÇÃO	45
4.2. CORTINA IMPERMEÁVEL	45
4.3. AQUÍFERO	49
4.4. BARRAGEM	54
4.5. BARRAGEM COM TAPETE DRENANTE	57
4.6. POÇO	60

CONSIDERAÇÕES FINAIS	61
-----------------------------	----

BIBLIOGRAFIA	63
---------------------	----

ÍNDICE DE FIGURAS

Fig. 1.1 – Experiência de Darcy	2
Fig. 1.2 – Elemento de volume	4
Fig. 1.3 – Barragem de terra homogénea.....	6
Fig. 1.4 – Condições de fronteira	7
Fig. 1.5 – Rede para um escoamento bidimensional num meio isotrópico e homogéneo	9
Fig. 2.1 – Elemento bidimensional. Referenciais geral e local	18
Fig. 2.2 – Elemento da família de Lagrange de 9 nós.....	19
Fig. 2.3 – Elementos da família Serendipity de 4 e 8 nós.....	19
Fig. 2.4 – Elemento da família de Lagrange. Geração das funções de aproximação N_k	21
Fig. 2.5 – Elemento de contorno $d\Gamma$	24
Fig. 3.1 – Dimensionamento da página.....	28
Fig. 3.2 – Definição do Problema	28
Fig. 3.3 – Geração do ficheiro de dados	29
Fig. 3.4 – Indicação da sequência de desenho	30
Fig. 3.5 – Desenho de um elemento.....	30
Fig. 3.6 – Seleccionar um elemento	31
Fig. 3.7 – Copiar um elemento.....	31
Fig. 3.8 – Mover um elemento	32
Fig. 3.9 – Atribuir potencial nos nós inferiores	33
Fig. 3.10 – Novo material	33
Fig. 3.11 – Atribuição do material.....	34
Fig. 3.12 – Menu rotativo.....	34
Fig. 3.13 – Visualização de resultados.....	44
Fig. 4.1 – Problema adaptado de Lambe & Whitman	45
Fig. 4.2 – Cortina impermeável: Esquema da malha adoptada.....	46
Fig. 4.3 – Cortina impermeável: Coloração da função potencial, versão Matlab	47
Fig. 4.4 – Cortina impermeável: Coloração da função potencial, versão Fortran	47
Fig. 4.5 – Cortina impermeável: Coloração da função de corrente com vectores de velocidade, versão Matlab ..	48

Fig. 4.6 – Cortina impermeável: Coloração da função de corrente com vectores de velocidade, versão Fortran .	48
Fig. 4.7 – Cortina impermeável: Rede de fluxo	49
Fig. 4.8 – Problema adaptado de J. N. Reddy.....	49
Fig. 4.9 – Aquífero: Esquema da malha adoptada.....	50
Fig. 4.10 – Aquífero: Coloração da função potencial, versão Matlab	51
Fig. 4.11 – Aquífero: Coloração da função potencial, versão Fortra	51
Fig. 4.12 – Aquífero: Coloração da função de corrente, versão Matlab.....	52
Fig. 4.13 – Aquífero: Coloração da função de corrente, versão Fortran	52
Fig. 4.14 – Aquífero: Rede de fluxo que combina as equipotenciais da primeira malha com as linhas de corrente da segunda	53
Fig. 4.15 – Aquífero: Pormenor entre os dois poços e representação dos vectores de velocidade	53
Fig. 4.16 – Barragem: Esquema da malha adoptada.....	54
Fig. 4.17 – Barragem: Coloração da função potencial, versão Matlab	55
Fig. 4.17 – Barragem: Coloração da função potencial, versão Fortran	55
Fig. 4.18 – Barragem: Coloração da função de corrente, versão Matlab.....	56
Fig. 4.18 – Barragem: Coloração da função de corrente, versão Fortran	56
Fig. 4.19 – Barragem: Rede de fluxo a azul e nível freático a vermelho.....	57
Fig. 4.20 – Barragem com tapete drenante: Condições geométricas do problema.....	57
Fig. 4.21 – Barragem com tapete drenante: Condições fronteira	57
Fig. 4.22 – Barragem com tapete drenante: Coloração da função potencial, versão Matlab	58
Fig. 4.23 – Barragem com tapete drenante: Coloração da função potencial, versão Fortran.....	58
Fig. 4.24 – Barragem com tapete drenante: Coloração da função corrente, versão Matlab	59
Fig. 4.25 – Barragem com tapete drenante: Coloração da função corrente, versão Fortran.....	59
Fig. 4.26 – Barragem com tapete drenante: Rede de fluxo, representação vectorial da velocidade e nível freático a verde	60
Fig. 4.27 – Poço: Descrição gráfica do problema	60

ÍNDICE DE TABELAS

Tabela 2.1 – Abcissas e factores de peso para a quadratura gaussiana	24
--	----

SÍMBOLOS E ABREVIATURAS

A - Área de uma secção

H - Função carga hidráulica

i - Gradiente hidráulico

k - Coeficiente de permeabilidade

$\vec{n}$ - Vector normal à fronteira

N_j - Função de aproximação do nó j

Q - Caudal

R_Γ - Resíduo na fronteira Γ

R_Ω - Resíduo no domínio Ω

S - Área de uma secção

$\vec{s}$ - Vector tangente à fronteira

u - Pressão intersticial da água

$\hat{u}$ - Função aproximante

v - Velocidade aparente de percolação

V - Volume total de solo

V_i - Função de peso na fronteira

x, y - Coordenadas cartesianas

Γ - Fronteira

ΔH - Perda de carga entre dois pontos

ξ, η - Coordenadas locais nos elementos paramétricos

ρ - Massa específica

ϕ - Função potencial

ψ - Função de corrente

Ω - Domínio

$[D]$ - Tensor de permeabilidade

$[f]$ - Vector fluxo

$[K]$ - Matriz de permeabilidade

- Tensor de permeabilidade

1

ÁGUA NOS SOLOS

1.1. INTRODUÇÃO

Sendo este trabalho dirigido para a resolução de problemas de percolação, importa, antes de mais, definir percolação e perceber porque é que é um problema. Chama-se percolação ao movimento da água nos solos. A necessidade do seu estudo resulta, como em qualquer obra de engenharia, da necessidade de garantir que se verificam as condições de segurança com uma probabilidade de ocorrência igual ou superior a um valor socialmente aceite. Este movimento da água deve-se a diferenças de carga hidráulica, designação atribuída no caso concreto à energia mecânica total, composta, como é sabido, pela soma da energia cinética com a energia potencial. Tendo em conta que nos maciços terrosos a velocidade com que a água se desloca é muito baixa, a energia cinética por unidade de peso é extremamente reduzida, pelo que pode ser desprezada. Em consequência, a energia potencial é praticamente igual à carga hidráulica [1].

1.2. LEI DE DARCY

Importa neste momento apresentar um conceito basilar do capítulo da percolação que é o gradiente hidráulico, que intuitivamente se compreende que seja a razão da perda de carga entre dois pontos pela distância percorrida entre eles. Henry Philibert Gaspert Darcy publicou em 1856 na obra intitulada *Les Fontaines Publiques de la Ville de Dijon*, a lei, determinada experimentalmente, que rege o movimento da água num meio poroso e que pode ser expressa da seguinte forma:

$$v = ki \quad (1.1)$$

Sendo v a velocidade de percolação e i o gradiente hidráulico. k é a constante de proporcionalidade entre ambas as grandezas e designa-se por coeficiente de permeabilidade do solo. A experiência está esquematizada na figura 1.1 [1]. Esta lei traduz, numa perspectiva física, o facto de, por um lado, a velocidade de percolação ser proporcional às forças de gravidade, e por outro, ser inversamente proporcional às forças de atrito que se geram no contacto da massa de água em movimento com a superfície das partículas sólidas do solo [2].

Deve ser notado que a velocidade em causa é aparente ou macroscópica, uma vez que resulta da divisão de um dado caudal pela secção total do solo, admitindo uma trajectória fictícia linearizada, entre montante e jusante e considerando ao longo dessa trajectória um movimento uniforme. Com

efeito, a água só atravessa uma fracção da área total pelo que a trajectória real da cada partícula através dos canaliculos formados pelos poros do solo é necessariamente bastante sinuosa. Uma partícula de água experimenta grandes variações de velocidade ao longo do seu percurso, ditadas pelos sucessivos estrangulamentos e alargamentos dos canaliculos [1].

A trajectória fictícia linearizada chama-se linha de corrente ou linha de fluxo e representa-se normalmente em conjunto com as linhas de igual carga hidráulica, denominadas equipotenciais. A representação conjunta destes dois tipos de linhas designa-se rede de fluxo ou rede de escoamento.

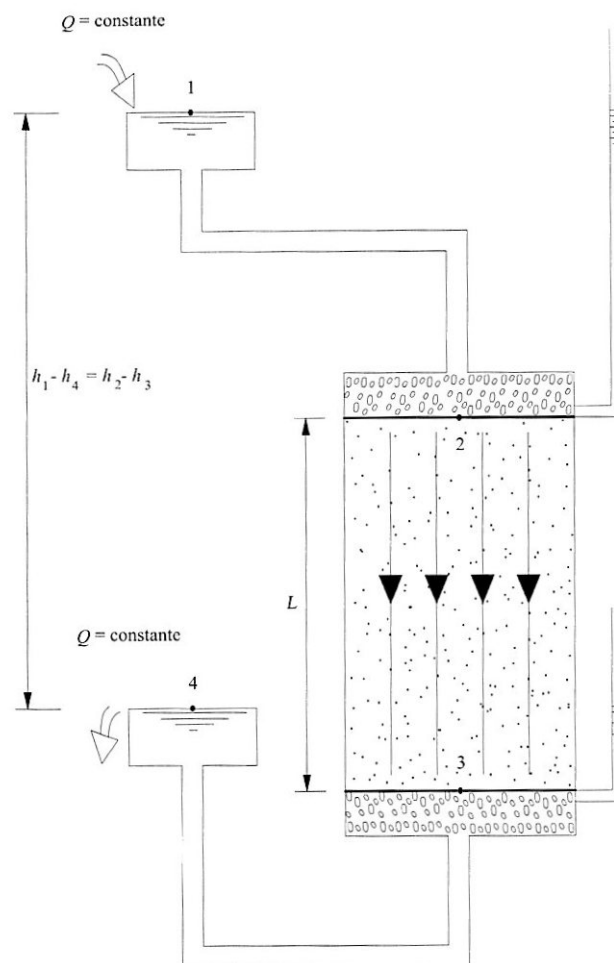


Fig. 1.1 – Experiência de Darcy

1.3. VALIDADE DA LEI DE DARCY

Sendo este trabalho orientado para o cálculo automático, é de vital importância definir o campo de aplicação desta lei, pois a qualidade dos resultados obtidos depende da dos dados introduzidos. A generalização de ferramentas poderosas de cálculo associadas a interfaces simples e apelativas tem cativado um número crescente de utilizadores, entre os quais se encontram aqueles que desconhecem os fundamentos teóricos em que se baseia o cálculo. O recurso a estas ferramentas em casos que não

se encaixam nos pressupostos admitidos, bem como a ausência de sentido crítico para avaliar os resultados, podem resultar em conclusões desastrosas.

A proporcionalidade entre a velocidade e o gradiente hidráulico estabelecida pela lei de Darcy é, como demonstraram posteriormente os estudos de Reynolds, própria dos escoamentos laminares [4]. Neste tipo de escoamento as partículas movem-se de forma ordenada, mantendo sempre a mesma posição relativa.

Para a grande maioria dos maciços terrosos os escoamentos verificam-se com velocidades muito baixas, pelo que a hipótese de escoamento laminar é inteiramente legítima. Escoamentos turbulentos podem todavia acontecer em certos solos muito grossos, como os cascalhos limpos [1]. Por outro lado, ao aplicar-se, por exemplo, um gradiente pequeno num solo fino não saturado, não só poderá deixar de ter validade a relação de proporcionalidade, como também poderá acontecer que o seu valor não seja suficiente para estabelecer escoamento [3].

1.4. EQUAÇÕES DIFERENCIAIS GOVERNATIVAS DO ESCOAMENTO DE ÁGUA EM MEIOS POROSOS

1.4.1. GENERALIZAÇÃO DA LEI DE DARCY

Esclarecido este assunto e voltando à lei de Darcy, note-se que a mesma pode ser reescrita na forma de equação diferencial, uma vez que:

$$i = \frac{\partial H}{\partial s} \quad (1.2)$$

onde ∂H representa a variação da carga hidráulica $H(x,y)$ entre os extremos do segmento infinitesimal ∂s da linha de corrente. Assim, partindo do princípio que o domínio é plano e bidimensional e que o meio é homogêneo e anisotrópico, a expressão assume a seguinte forma matricial:

$$\begin{Bmatrix} v_x \\ v_y \end{Bmatrix} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{xy} & k_{yy} \end{bmatrix} \begin{Bmatrix} -\frac{\partial H}{\partial x} \\ -\frac{\partial H}{\partial y} \end{Bmatrix} \quad (1.3)$$

Há que destacar dois aspectos da expressão anterior. O primeiro está relacionado com o sinal menos que está a afectar o vector das perdas de carga, para que a velocidade tenha o sentido do escoamento, ou seja, dos potenciais maiores para os menores. O outro está relacionado com o facto de os termos da diagonal secundária serem iguais. Na verdade, o tensor $[k]$ é simétrico e o seu elemento genérico, k_{ij} , representa a componente da velocidade na direcção i , induzida por uma componente unitária do gradiente hidráulico na direcção j [2].

1.4.2. EQUAÇÃO DA CONTINUIDADE

1.4.2.1. Desenvolvimento da Expressão

Acrescente-se agora uma nova expressão também válida para o escoamento de água em meios porosos:

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = 0 \quad (1.4)$$

Trata-se da equação da continuidade e tem como objectivo garantir que a quantidade de água que entra num elemento de volume infinitesimal é igual à que sai, uma vez que tanto a água como as partículas sólidas são consideradas incompressíveis. Para se chegar à expressão (1.4), considere-se um elemento de volume $dx \, dy \, dz$ e que o fluxo é bidimensional, uma vez que o que se passa no plano xy se repete em planos paralelos a este ao longo de z [2].

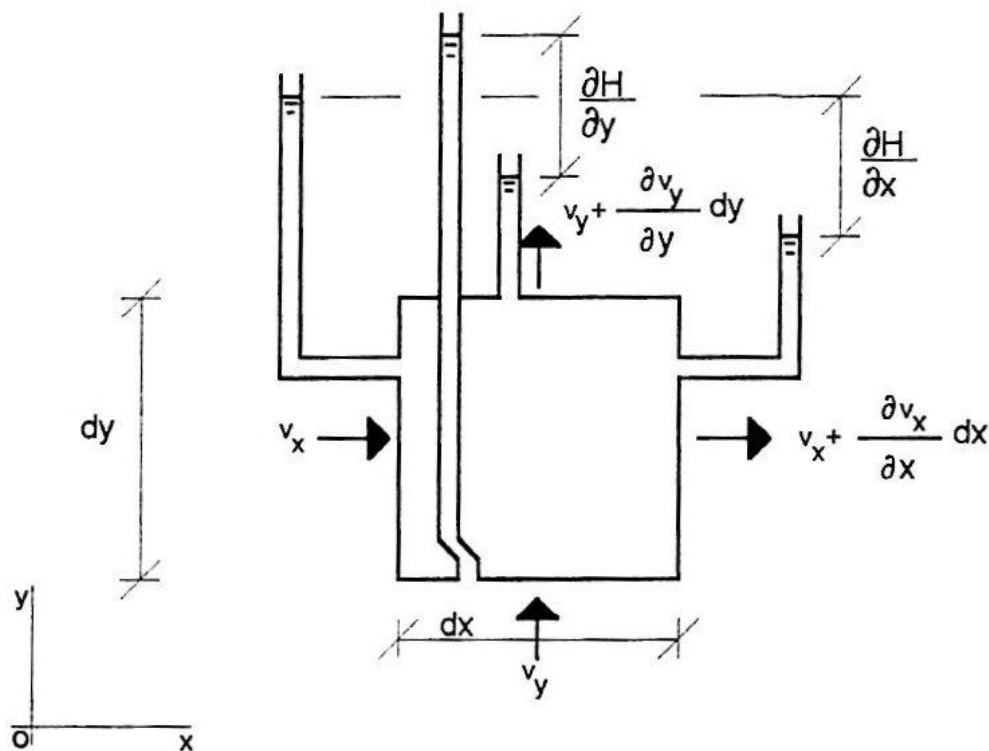


Fig. 1.2 – Elemento de volume

Neste caso, a água que entra é

$$v_x \cdot dy \cdot dz + v_y \cdot dx \cdot dz \quad (1.5)$$

e a água que sai

$$\left(v_x + \frac{\partial v_x}{\partial x} dx \right) dy \cdot dz + \left(v_y + \frac{\partial v_y}{\partial y} dy \right) dx \cdot dz \quad (1.6)$$

Igualando as duas expressões, chega-se à conclusão que

$$\frac{\partial v_x}{\partial x} dx \cdot dy \cdot dz + \frac{\partial v_y}{\partial y} dy \cdot dx \cdot dz = 0 \quad (1.7)$$

que é uma expressão equivalente à (1.4).

1.4.2.2. Função Potencial

Se se considerar a função ϕ , designada por função potencial:

$$\phi_{(x,y)} = H_{(x,y)} + C \quad (1.8)$$

em que C é uma constante cujo valor é determinado pelas condições fronteira, pode escrever-se a equação diferencial governativa do escoamento com base na função potencial substituindo a expressão (1.3) na (1.4).

$$\frac{\partial}{\partial x} \left(k_{xx} \frac{\partial \phi}{\partial x} + k_{xy} \frac{\partial \phi}{\partial y} \right) + \frac{\partial}{\partial y} \left(k_{xy} \frac{\partial \phi}{\partial x} + k_{yy} \frac{\partial \phi}{\partial y} \right) = 0 \quad (1.9)$$

Admitindo um solo isotrópico, $k_{xx} = k_{yy} = k$ e $k_{xy} = 0$, a equação (1.9) assume a forma

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \quad (1.10)$$

Uma equação do tipo da (1.10), ou seja, que assume a forma $\nabla^2 f = 0$, sendo f uma função qualquer, designa-se por equação de Laplace. Nesse caso, a função f chama-se função harmónica. No caso do solo ser anisotrópico k também é função de x e y , assim como o potencial ϕ , portanto, a função potencial é chamada quase-harmónica.

Antes de se prosseguir para a resolução da equação (1.9) importa esclarecer a noção de condição fronteira, referida a propósito da função potencial. Impor uma condição fronteira, significa obrigar certos resultados a assumirem valores conhecidos *a priori*. Note-se que existem infinitas soluções que respeitam a equação (1.9) e o que torna a solução única são as restrições impostas com base nos condicionalismos do problema em causa. As condições fronteira podem agrupar-se em dois tipos:

- i. Condição de fronteira essencial ou de Dirichlet:
 - Condição em que os valores prescritos da função são impostos em determinado trecho da fronteira do domínio

$$\phi = \bar{\phi} \quad (1.11)$$

- ii. Condição de fronteira natural ou de Neumann:
 - A derivada da função, segundo a normal à fronteira, tem valores prescritos

$$\phi_n = \left(k_{xx} \frac{\partial \phi}{\partial x} + k_{xy} \frac{\partial \phi}{\partial y} \right) n_x + \left(k_{xy} \frac{\partial \phi}{\partial x} + k_{yy} \frac{\partial \phi}{\partial y} \right) n_y \quad (1.12)$$

sendo n_x e n_y as componentes do versor $\vec{n}$ da normal exterior à fronteira.

Na figura 1.3 apresenta-se uma barragem de terra que servirá para exemplificar os diferentes tipos de condições fronteira que podem surgir, estando eles discriminados na figura 1.4 [2].

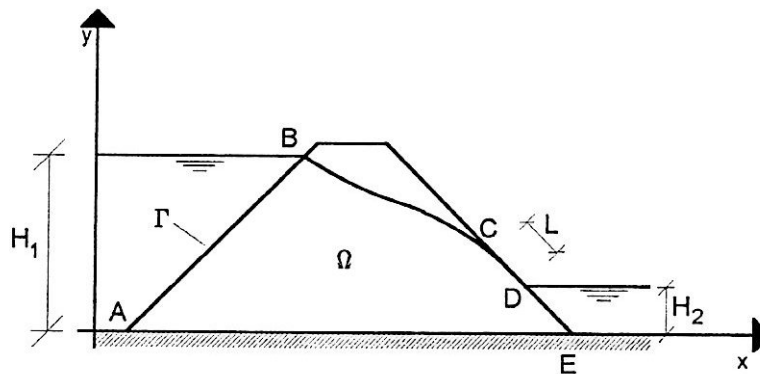


Fig. 1.3 – Barragem de terra homogênea

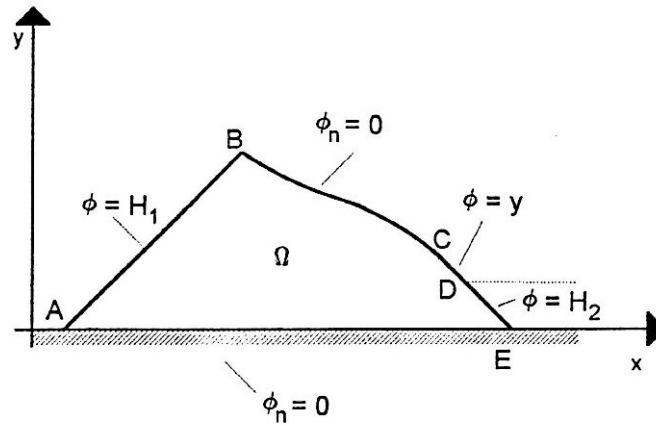


Fig. 1.4 – Condições de fronteira

1.4.2.3. Função de Corrente

Considere-se agora a função ψ tal que [1]

$$v_x = \frac{\partial \psi}{\partial y} \quad (1.13a)$$

$$v_y = -\frac{\partial \psi}{\partial x} \quad (1.13b)$$

Esta relaciona-se com a função ϕ , voltando à premissa de solo isotrópico para simplificar as expressões, através da velocidade da seguinte forma

$$k \frac{\partial \phi}{\partial x} = \frac{\partial \psi}{\partial y} \quad (1.14a)$$

$$k \frac{\partial \phi}{\partial y} = -\frac{\partial \psi}{\partial x} \quad (1.14b)$$

Derivando a equação (1.14a) em ordem a y e (1.14b) em ordem a x obtemos os seguintes resultados

$$k \frac{\partial^2 \phi}{\partial x \partial y} = \frac{\partial^2 \psi}{\partial y^2} \quad (1.15a)$$

$$k \frac{\partial^2 \phi}{\partial x \partial y} = - \frac{\partial^2 \psi}{\partial x^2} \quad (1.15b)$$

O que significa que a igualdade entre os segundos membros de ambas as equações 1.15 está assegurada e, portanto, passando os dois termos para o mesmo membro, se chega à equação (1.16), querendo isto dizer que a função ψ também satisfaz a equação de Laplace.

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0 \quad (1.16)$$

Importa agora perceber qual o significado físico da função ψ . Para isso, admita-se uma função ψ_1 constante. Nesse caso, o seu diferencial pode escrever-se:

$$d\psi = \frac{\partial \psi}{\partial x} dx + \frac{\partial \psi}{\partial y} dy \quad (1.17)$$

e será necessariamente nulo. Trabalhando então o segundo membro da equação (1.17) igual a zero, chega-se à expressão seguinte

$$\left(\frac{dy}{dx} \right)_{\psi=\psi_1} = - \frac{\frac{\partial \psi}{\partial x}}{\frac{\partial \psi}{\partial y}} = \frac{v_y}{v_x} \quad (1.18)$$

Sendo dy/dx a tangente à curva ψ_1 , pode concluir-se que essa curva coincide com a direcção da velocidade. Deste modo, as curvas que correspondem a valores constante da função ψ representam a direcção da corrente e como tal, são chamadas de linhas de corrente. Para além disso, pode já adiantar-se que a interpretação física do valor da função de corrente é a quantidade de caudal que passa abaixo dessa linha.

Repare-se que, por um raciocínio análogo ao anterior, pode concluir-se que

$$\left(\frac{dy}{dx} \right)_{\phi=\phi_1} = \frac{v_x}{v_y} \quad (1.19)$$

o que demonstra que nos meios com isotropia de permeabilidade as equipotenciais são normais às linhas de corrente. A figura 1.5 exemplifica uma rede de escoamento bidimensional num meio isotrópico e homogéneo [1].

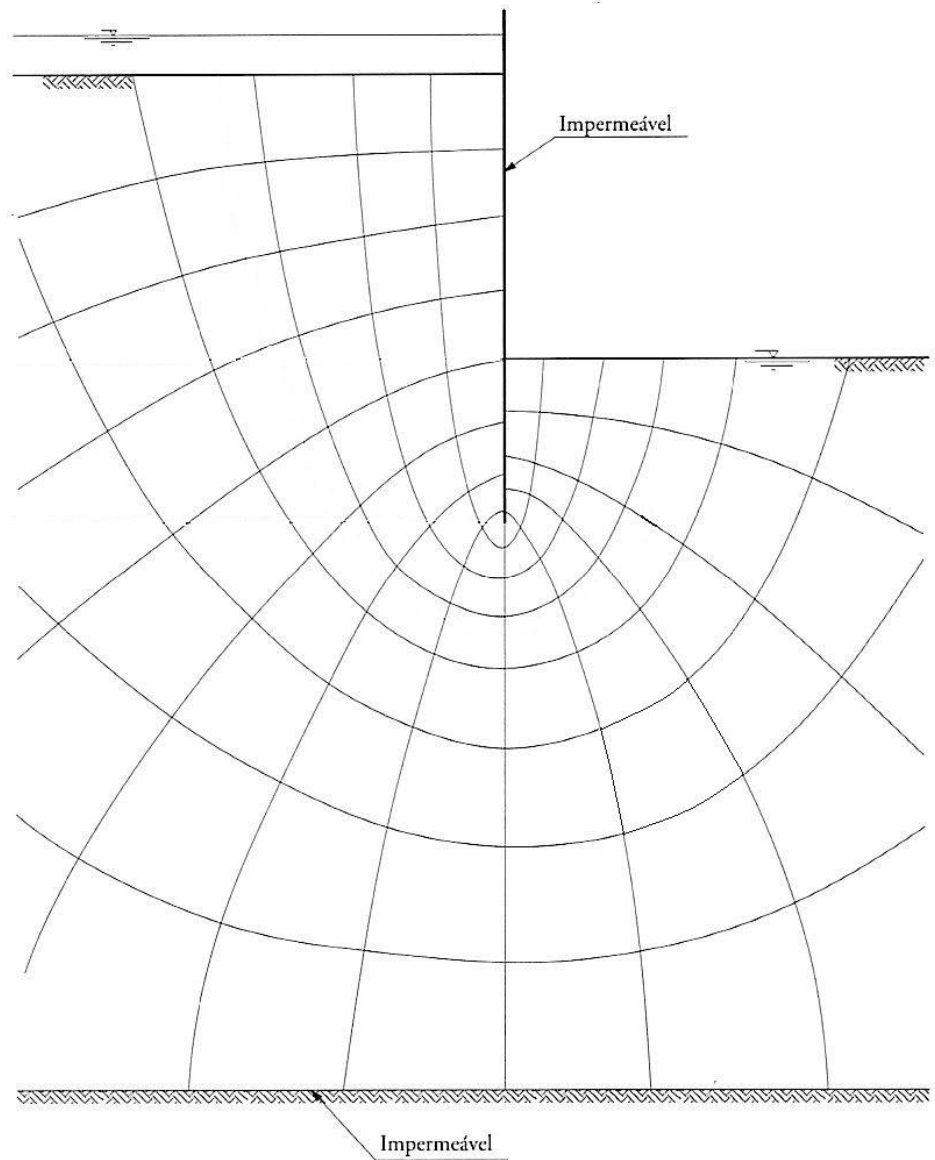


Fig. 1.5 – Rede para um escoamento bidimensional num meio isotrópico e homogêneo

2

DESENVOLVIMENTO NUMÉRICO

2.1. INTRODUÇÃO

Há diversos fenómenos na natureza que podem ser expressos através de equações diferenciais, contudo, a resolução analítica dessas equações só é possível para casos muito simples, que raramente correspondem ao problema em estudo. No sentido de ultrapassar esta dificuldade, desenvolveram-se métodos numéricos que permitem encontrar uma função capaz de aproximar a solução da equação diferencial em todo o domínio. Desses destacam-se três dignos de referência, sendo eles o método das diferenças finitas, o método variacional e o método dos elementos finitos [9].

Na aproximação por diferenças finitas de uma equação diferencial, as derivadas na equação são substituídas pelos quocientes das diferenças que envolvem os valores da solução em pontos discretos da malha do domínio. As equações discretas resultantes são resolvidas, depois de impostas as condições fronteira, para os valores da solução nos pontos da malha. Apesar do método das diferenças finitas ser simples em termos conceptuais, tem várias desvantagens. As mais notáveis são a imprecisão das derivadas da solução aproximada, a dificuldade na imposição das condições fronteira ao longo de fronteiras não lineares, a dificuldade de representar com exactidão domínios de geometria complexa e a incapacidade de recorrer a malhas não uniformes ou não rectangulares.

Na solução variacional das equações diferenciais, a equação diferencial é posta numa forma variacional equivalente e depois a solução aproximada é assumida como sendo a combinação $(\sum c_j \phi_j)$ de determinadas funções de aproximação ϕ_j . Os coeficientes c_j são determinados pela forma variacional. Os métodos variacionais têm a desvantagem de ser difícil construir as funções de aproximação para problemas com domínios arbitrários.

O método dos elementos finitos supera a dificuldade do método variacional, porque utiliza um procedimento sistemático para as funções de aproximação. O método emprega duas ferramentas básicas que atestam a sua superioridade em relação aos métodos concorrentes. Em primeiro lugar, um domínio de geometria complexa é representado como um somatório de subdomínios simples, chamados elementos finitos. Em segundo lugar, sobre cada elemento finito as funções de aproximação são derivadas recorrendo à ideia de que qualquer função contínua pode ser representada por uma combinação linear de polinómios algébricos. As funções de aproximação são deduzidas usando conceitos da teoria da interpolação e, portanto, chamadas funções de interpolação. Assim, o elemento finito pode ser interpretado como uma aplicação local do método variacional, como seja o método dos resíduos pesados, no qual as funções de aproximação são polinómios algébricos e os parâmetros indeterminados representam os valores da solução num número finito de pontos pré-seleccionados, chamados nós, na fronteira e no interior do elemento.

2.2. ELEMENTOS FINITOS

2.2.1. MÉTODO DOS RESÍDUOS PESADOS

É objectivo deste método a determinação de uma função aproximante $\hat{u}$ à solução exacta u de uma equação diferencial que governa um determinado fenómeno, garantindo o cumprimento das condições fronteira, isto é, que seja exacta na fronteira Γ do domínio Ω .

Considere-se para função aproximante $\hat{u}$, uma combinação de funções dada por [5]:

$$u \approx \hat{u} = \psi + \sum_{j=1}^M a_j \cdot N_j \quad (2.1)$$

em que:

- ψ é uma função que na fronteira Γ assume exactamente os valores de u , isto é, $\psi_\Gamma = u_\Gamma$;
- $\{N_j\}$, $j = 1, \dots, M$, é um conjunto de funções de aproximação linearmente independentes, que devem anular-se na fronteira Γ ;
- $\{a_j\}$, $j = 1, \dots, M$, é o conjunto dos coeficientes de aproximação que devem ser determinados de forma que a aproximação $\hat{u}$ seja a melhor possível.

Entende-se por resíduo de uma aproximação a diferença entre a solução exacta e a aproximação [5]:

$$R_\Omega = u - \hat{u} \quad (2.2)$$

Sendo este o método dos resíduos pesados, resta agora saber onde estão os pesos. Note-se que se fosse imposta a condição (2.3) podia haver resíduos negativos anulados com positivos, resultando em fracas aproximações.

$$\int_{\Omega} R_\Omega \cdot \partial\Omega = 0 \quad (2.3)$$

Em alternativa a este critério é sugerido um outro que consiste no anulamento de um número apropriado de integrais do resíduo estendidos ao domínio, mas agora com o resíduo pesado de diversas formas:

$$\int_{\Omega} W_i \cdot R_\Omega \cdot \partial\Omega = 0 \quad (2.4)$$

em que W_i , para $i = 1, \dots, M$, é um conjunto de pesos linearmente independentes.

2.2.2. MÉTODO DE GALERKIN

Combinando agora as expressões (2.1), (2.2) e (2.4) é possível estabelecer um sistema de M equações a M incógnitas que pode ser escrito da seguinte forma:

$$[K]\{a\} = \{f\} \quad (2.5)$$

em que:

$$k_{ij} = \int_{\Omega} W_i \cdot N_j \cdot \partial\Omega \quad i, j = 1, \dots, M \quad (2.6)$$

$$f_i = \int_{\Omega} W_i \cdot (u - \psi) \cdot \partial\Omega \quad i = 1, \dots, M \quad (2.7)$$

$\{a_j\}$ - vector dos coeficientes de aproximação

Se se adoptarem funções de peso $\{W_i\}$ iguais às funções de aproximação $\{N_j\}$, sendo este método uma variante do Método dos Resíduos Pesados denominada Método de Galerkin, as expressões (2.6) e (2.7) reescrevem-se da forma que se segue:

$$k_{ij} = \int_{\Omega} N_i \cdot N_j \cdot \partial\Omega \quad i, j = 1, \dots, M \quad (2.8)$$

$$f_i = \int_{\Omega} N_i \cdot (u - \psi) \cdot \partial\Omega \quad i = 1, \dots, M \quad (2.9)$$

A expressão (2.8) confirma que, recorrendo a este método, a matriz K é simétrica.

2.2.3. RESÍDUO NA FRONTEIRA

Note-se que até agora se tem trabalhado com a função ψ , que tal como se definiu em (2.1) implica o conhecimento de toda a fronteira e a definição de uma função que garanta todas essas condições. Uma vez que este processo se revela pouco prático, tanto menos quanto mais complexa for a geometria do problema, para além de implicar a definição de uma nova função no caso das condições fronteira se alterarem, o próximo passo será relaxar esta condição. Assim sendo, a função aproximante passará a assumir a seguinte forma:

$$u \approx \hat{u} = \sum_{j=1}^M a_j \cdot N_j \quad (2.10)$$

Consequentemente, na formulação que se vai desenvolver para além de se considerar um resíduo no domínio tem de se ter em conta um resíduo na fronteira. Assim, a expressão (2.4) terá de se completar:

$$\int_{\Omega} W_i \cdot R_{\Omega} \cdot \partial\Omega + \int_{\Gamma} V_i \cdot R_{\Gamma} \cdot \partial\Gamma = 0 \quad i = 1, \dots, M \quad (2.11)$$

em que $\{W_i\}$ e $\{V_i\}$ constituem, em princípio, diferentes conjuntos de funções de peso no domínio e na fronteira, respectivamente.

2.2.4. PROPRIEDADES DAS FUNÇÕES DE APROXIMAÇÃO

Pelo facto da aproximação $\hat{u}$ à solução exacta u poder ser construída a partir de uma combinação linear de funções associadas aos nós, a cada nó j terá que corresponder uma função N_j com as seguintes propriedades [5]:

- i. a função N_j terá que assumir um valor nulo em todo o domínio, excepto nos elementos associados ao nó j ;
- ii. a função N_j é igual à unidade no nó j a que está associada e tem valor nulo nos restantes nós.

Atendendo à propriedade ii das funções N_j , a matriz $[N]$ é uma matriz identidade, sendo por isso fácil de concluir que os coeficientes $\{a_j\}$ representam os valores da função $u(x)$ nos nós do domínio. Portanto, a expressão (2.10) assume ainda a forma:

$$u \approx \hat{u} = \sum_{j=1}^M u_j \cdot N_j \quad (2.12)$$

2.3. APLICAÇÃO DO MÉTODO DOS ELEMENTOS FINITOS

2.3.1. FUNÇÃO POTENCIAL

Neste momento já é possível aplicar a formulação teórica do método ao caso concreto em estudo, recorrendo à função potencial como base. Assim sendo, a expressão (2.12) assume a forma:

$$\phi \approx \hat{\phi} = \sum_{i=1}^M \phi_i \cdot N_i \quad (2.13)$$

e os resíduos no domínio e na fronteira definem-se, respectivamente, por:

$$R_{\Omega} = \frac{\partial}{\partial x} \left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) + \frac{\partial}{\partial y} \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \quad (2.14)$$

$$R_{\Gamma_v} = \left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_x + \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_y + v_n \quad (2.15)$$

sendo v_n o caudal normal à fronteira nos nós da malha e que se pode encarar também como o somatório do produto do caudal de cada nó pela respectiva função de interpolação.

Substituindo (2.14) e (2.15) em (2.11) obtém-se:

$$\begin{aligned} & \int_{\Omega} W_i \cdot \left[\frac{\partial}{\partial x} \left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) + \frac{\partial}{\partial y} \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \right] \cdot \partial \Omega + \\ & + \int_{\Gamma_v} V_i \cdot \left[\left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_x + \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_y + v_n \right] \cdot \partial \Gamma = 0 \\ & i = 1, \dots, M \end{aligned} \quad (2.16)$$

Fazendo agora uma transferência de diferenciação para as funções de peso W_i no integral do resíduo pesado estendido ao domínio Ω , a expressão (2.16) passa a assumir o seguinte aspecto:

$$\begin{aligned} & - \int_{\Omega} \left[\frac{\partial W_i}{\partial x} \cdot \left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) + \frac{\partial W_i}{\partial y} \cdot \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \right] \cdot \partial \Omega + \\ & + \int_{\Gamma} W_i \cdot \left[\left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_x + \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_y \right] \cdot \partial \Gamma + \\ & + \int_{\Gamma_v} V_i \cdot \left[\left(k_{xx} \frac{\partial \hat{\phi}}{\partial x} + k_{xy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_x + \left(k_{xy} \frac{\partial \hat{\phi}}{\partial x} + k_{yy} \frac{\partial \hat{\phi}}{\partial y} \right) \cdot n_y + v_n \right] \cdot \partial \Gamma = 0 \\ & i = 1, \dots, M \end{aligned} \quad (2.17)$$

A passagem da expressão (2.16) para a (2.17) foi feita à custa do teorema de Green dado por:

$$\int_{\Omega} \alpha \frac{\partial \beta}{\partial x} \partial \Omega = - \int_{\Omega} \frac{\partial \alpha}{\partial x} \beta \cdot \partial \Omega + \int_{\Gamma} \alpha \cdot \beta \cdot n_x \cdot \partial \Gamma \quad (2.18a)$$

$$\int_{\Omega} \alpha \frac{\partial \beta}{\partial y} \partial \Omega = - \int_{\Omega} \frac{\partial \alpha}{\partial y} \beta \cdot \partial \Omega + \int_{\Gamma} \alpha \cdot \beta \cdot n_y \cdot \partial \Gamma \quad (2.18b)$$

Caso se adopte o critério:

$$V_i = -W_i \quad \text{em } \Gamma_v \quad i = 1, \dots, M \quad (2.19)$$

A expressão (2.17) pode ainda ser simplificada, devido à anulação da segunda parcela com a primeira parte da terceira, ambas do primeiro membro. Antes de reescrever a expressão (2.17), vão introduzir-se já mais três alterações que, sendo fáceis de acompanhar, evitam ter de reescrever a mesma expressão mais três vezes. A primeira é a substituição da expressão (2.13), a segunda é a discretização por elementos finitos e a terceira é a aplicação do método de Galerkin, tal como anteriormente.

$$\begin{aligned} & \sum_{j=1}^M \phi_j \left\{ \sum_{e=1}^E \int_{\Omega^e} \left[\left(k_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + k_{xy} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} \right) + \left(k_{xy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} + k_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right] \cdot \partial \Omega \right\} + \\ & + \sum_{e=1}^E \int_{\Gamma_v^e} N_i \cdot v_n \cdot \partial \Gamma = 0 \\ & i = 1, \dots, M \end{aligned} \quad (2.20)$$

A partir da expressão (2.20) é possível gerar um sistema de M equações a M incógnitas na forma:

$$[K]\{\phi\} = \{f\} \quad (2.21)$$

em que:

$$\begin{aligned} K_{ij} &= \sum_{e=1}^E \int_{\Omega^e} \left[\left(k_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + k_{xy} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} \right) + \left(k_{xy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} + k_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right] \cdot \partial \Omega \\ & i, j = 1, \dots, M \end{aligned} \quad (2.22)$$

e

$$f_i = - \sum_{e=1}^E \int_{\Gamma_v^e} N_i \cdot v_n \cdot \partial \Gamma \quad i = 1, \dots, M \quad (2.23)$$

Note-se que as expressões (2.22) e (2.23) podem ainda ser escritas da seguinte forma:

$$[K^e] = \int [B]^T [D] [B] \cdot \partial\Omega \quad (2.24)$$

$$\{f^e\} = - \int_{\Gamma_v^e} [N]^T \cdot v_n \cdot \partial\Gamma \quad (2.25)$$

desde que:

$$[B] = [\{B_1\}, \{B_2\}, \dots, \{B_i\}, \dots, \{B_n\}] \quad (2.26)$$

$$\{B_i\} = \left\{ \begin{array}{c} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{array} \right\} \quad (2.27)$$

$$[N] = [N_1, N_2, \dots, N_i, \dots, N_n] \quad (2.28)$$

$$[D] = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{xy} & k_{yy} \end{bmatrix} \quad (2.29)$$

sendo n o número de nós do elemento.

2.3.2. ELEMENTOS PARAMÉTRICOS

Nas expressões (2.22) e (2.23) pode constatar-se que cada parcela se obtém pelo somatória da contribuição da cada elemento, tal como se antecipou no início do capítulo. Efectivamente, referiu-se também que cada elemento iria ser tratado individualmente e que essa era a grande vantagem do método, pois pressupõe-se que os elementos tenham geometrias muito simples. Assim, surge a distinção entre referencial geral (o, x, y) e local (ϕ, ξ, η). O referencial local deve ser tal que a origem esteja no centro do elemento e os lados do elemento correspondam às coordenadas -1 ou 1 de cada uma das direcções, tal como exemplifica a figura 2.1.

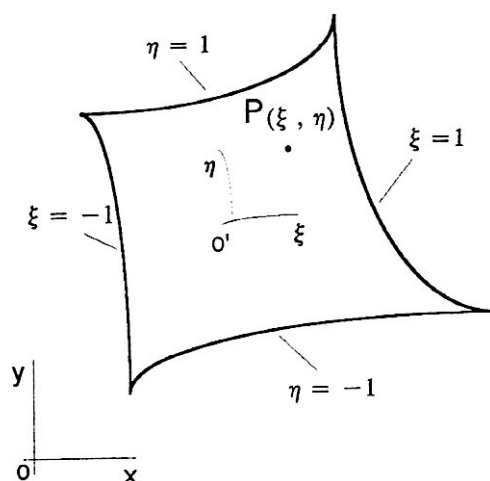


Fig. 2.1 – Elemento bidimensional. Referenciais geral e local

Note-se que a formulação por elementos finitos é independente do tipo de elemento escolhido e, portanto, a escolha do elemento apropriado deve ser feita em função da geometria do domínio a ser modelado e do grau de precisão pretendido. Em relação à modelação da geometria há a destacar dois tipos de elementos comumente utilizados: os triangulares e os rectangulares. Os primeiros são mais vantajosos para geometrias circulares, enquanto que os segundos devem ser empregues quando há necessidade de respeitar condições de simetria do problema. Contudo, o programa desenvolvido está apenas programado para elementos rectangulares.

Uma das famílias de elementos rectangulares mais conhecidas é a de Lagrange. Um elemento de ordem p da família de Lagrange tem n nós com $n = (p+1)^2$, o que significa que o elemento de quatro nós é de primeira ordem e o de nove nós de segunda. Na figura 2.2 está representado um elemento de nove nós desta família. Estes elementos, para as ordens superiores, são compostos por nós na fronteira do elemento e nós no seu interior. Contudo, uma vez que os nós internos dos elementos da família de Lagrange não contribuem para a conectividade entre elementos, eles podem ser condensados para que o tamanho das matrizes dos elementos possa ser reduzido.

Alternativamente, podem usar-se os elementos da família Serendipity para evitar os nós presentes no interior dos elementos de Lagrange. Os elementos Serendipity são os elementos rectangulares que não têm nós interiores. Por outras palavras, todos os nós estão na fronteira do elemento. Na figura 2.3 encontram-se dois exemplares desta família.

Assim sendo, os elementos adoptados no programa são de quatro, oito ou nove nós. O elemento de quatro nós é coincidente em ambas as famílias, como se poderá constatar adiante, o de oito nós pertence à família Serendipity e o de nove nós à família de Lagrange.

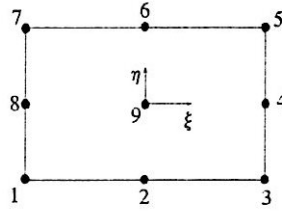


Fig. 2.2 – Elemento da família de Lagrange de 9 nós

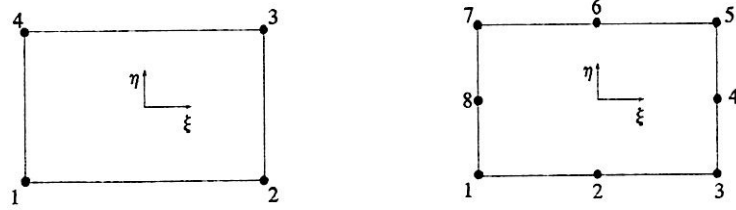


Fig. 2.3 – Elementos da família Serendipity de 4 e 8 nós

Está ainda por esclarecer como é que se faz a passagem do referencial local para o geral. Note-se que, conforme já foi explicitado, a função a aproximar pelo Método dos Elementos Finitos nos escoamentos em meios porosos é a função potencial ϕ dada pela expressão:

$$\phi \approx \hat{\phi} = \sum_{i=1}^M \phi_i \cdot N_i \quad (2.13)$$

em que ϕ_i são os potenciais nos nós da malha.

Analogamente, a definição da geometria dos elementos é realizada a partir de funções idênticas à função (2.13) utilizada na aproximação dos potenciais, isto é:

$$\begin{aligned} x &= \sum_{i=1}^M x_i \cdot N_i \\ y &= \sum_{i=1}^M y_i \cdot N_i \end{aligned} \quad (2.30)$$

em que x_i e y_i são as coordenadas dos nós da malha de elementos finitos.

Como as funções de interpolação N_i para traduzir os potenciais e definir a geometria dos elementos são as mesmas, atribui-se aos elementos em causa a designação de elementos isoparamétricos [6].

As propriedades que as funções de interpolação $N_i(\xi, \eta)$ devem respeitar foram já referidas na secção 2.2.4 e as expressões que garantem essas condições são as seguintes [2]:

i. Família de Serendipity:

- Elementos de 4 nós

$$N_i = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i) \quad i = 1, 2, 3, 4 \quad (2.31)$$

- Elementos de 8 nós

Nós de canto

$$N_i = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i)(\xi\xi_i + \eta\eta_i - 1) \quad i = 1, 3, 5, 7 \quad (2.32a)$$

Nós intermédios

$$\begin{aligned} \xi_i = 0 \quad N_i &= \frac{1}{2}(1 - \xi^2)(1 + \eta\eta_i) \\ \eta_i = 0 \quad N_i &= \frac{1}{2}(1 + \xi\xi_i)(1 - \eta^2) \end{aligned} \quad i = 2, 4, 6, 8 \quad (2.32b)$$

ii. Família de Lagrange:

$$N_k = L_k(\xi) \cdot L_k(\eta) \quad (2.33)$$

$$L_k(\xi) = \frac{\prod_{\substack{i=0 \\ i \neq k}}^n (\xi - \xi_i)}{\prod_{\substack{i=0 \\ i \neq k}}^n (\xi_k - \xi_i)} \quad (2.34)$$

$$L_k(\eta) = \frac{\prod_{\substack{i=0 \\ i \neq k}}^m (\eta - \eta_i)}{\prod_{\substack{i=0 \\ i \neq k}}^m (\eta_k - \eta_i)} \quad (2.35)$$

- Elementos de 9 nós

Nós de canto

$$N_i = \frac{1}{4}(\xi^2 + \xi\xi_i)(\eta^2 + \eta\eta_i) \quad i = 1, 3, 5, 7 \quad (2.36a)$$

Nós intermédios

$$\begin{aligned} N_i &= \frac{1}{2}\eta_i^2(\eta^2 + \eta\eta_i)(1 - \xi^2) + \frac{1}{2}\xi_i^2(\xi^2 + \xi\xi_i)(1 - \eta^2) \\ i &= 2, 4, 6, 8 \end{aligned} \quad (2.36b)$$

Nó central

$$N_i = (1 - \xi^2)(1 - \eta^2) \quad i = 9 \quad (2.36c)$$

Os valores de m e n estão definidos na figura 2.4 que auxilia a compreensão do processo de geração das funções de aproximação [2].

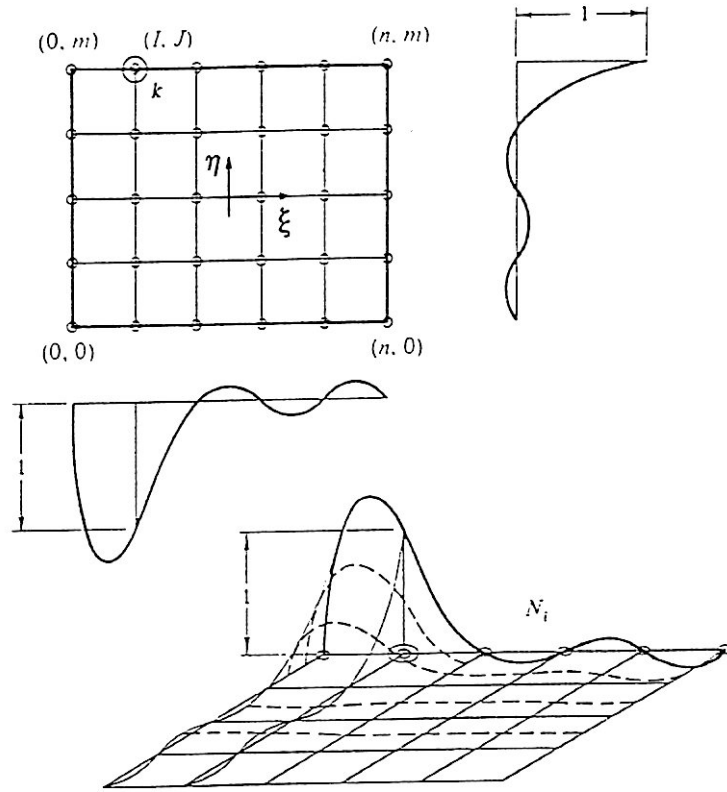


Fig. 2.4 – Elemento da família de Lagrange. Geração das funções de aproximação N_k

A título de exemplo, vai calcular-se a função N_1 para o elemento de quatro nós, recorrendo às expressões de Lagrange, com o objectivo de mostrar que se obtém a mesma expressão que se definiu para o elemento de quatro nós da família Serendipity. Assim, adoptando a numeração dos nós da figura 2.2

$$L_1(\xi) = \frac{(\xi - \xi_2)}{(\xi_1 - \xi_2)} = \frac{1 - \xi}{2} \quad (2.37)$$

$$L_1(\eta) = \frac{(\eta - \eta_4)}{(\eta_1 - \eta_4)} = \frac{1 - \eta}{2} \quad (2.38)$$

Substituindo (2.37) e (2.38) na expressão (2.33), obtemos a seguinte expressão:

$$N_1 = \frac{1}{4}(1 - \xi)(1 - \eta) \quad (2.39)$$

que corresponde à expressão (2.31) para $i = 1$.

2.3.3. INTEGRAÇÃO EM RELAÇÃO ÀS COORDENADAS LOCAIS

Neste momento já são conhecidas as funções de forma, mas na expressão (2.27) essas funções estão derivadas em ordem às coordenadas globais x e y . Como se viu na secção anterior, as funções de forma são funções das coordenadas locais ξ e η . Interessa, portanto, relacionar as duas derivadas para que se possa derivar em ordem às coordenadas locais. A matriz capaz de estabelecer essa relação é a matriz Jacobiana.

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \quad (2.40)$$

sendo $[J]$ a matriz Jacobiana que é definida por [6]:

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (2.41)$$

mas como o que se pretende é substituir as derivadas das funções de forma em relação às coordenadas globais, a matriz Jacobiana tem de passar para o outro membro, reescrevendo-se a expressão (2.40) de seguinte forma:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \quad (2.42)$$

sendo $[J]^{-1}$ a matriz inversa da matriz Jacobiana dada por:

$$[J]^{-1} = \frac{1}{|J|} \begin{bmatrix} \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \quad (2.43)$$

onde $|J|$ é o determinante da matriz Jacobiana.

Por outro lado, sabendo que $\partial\Omega = |J| \cdot \partial\xi \cdot \partial\eta$, a expressão (2.24) assume a forma final:

$$[K_{ij}] = \int_{-1}^{+1} \int_{-1}^{+1} [B_i]^T [D] [B_j] \cdot |J| \cdot \partial\xi \cdot \partial\eta \quad (2.44)$$

sendo $\{B_i\}$ calculado de acordo com a expressão (2.42).

2.3.4. INTEGRAÇÃO NUMÉRICA

Para determinar a matriz de permeabilidade $[K]$ é ainda necessário resolver a integração dupla da expressão (2.44).

Para isso, considere-se que as funções integrandas do integral (2.44) são representadas pela função genérica $f(\xi, \eta)$, tendo-se então [2]:

$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) \cdot \partial\xi \cdot \partial\eta \quad (2.45)$$

Aplicando o Método de Gauss-Legendre a expressão que permite determinar o integral (2.45) por integração numérica é dado por:

$$I \cong \sum_{i=1}^n \sum_{j=1}^n H_i \cdot H_j \cdot f(\xi_i, \eta_j) \quad (2.46)$$

sendo ξ_i e η_j as coordenadas locais dos pontos de integração, H_i e H_j os respectivos factores de peso e n a ordem de integração usada. No caso concreto, o programa permite escolher dois ou três pontos de integração em cada direcção, portanto, n será 2 ou 3, conforme seja um caso ou o outro.

A tabela 2.1 apresenta os valores das coordenadas dos pontos de integração e os respectivos factores de peso para a integração de Gauss-Legendre para os dois casos mencionados.

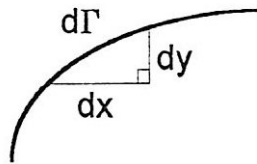
Tabela 2.1 – Abcissas e factores de peso para a quadratura gaussiana

$$\int_{-1}^{+1} f(x) dx = \sum_{j=1}^n H_j f(a_j)$$

$\pm a$		H
	$n = 2$	
0,57735 02691 89626		1,00000 00000 00000
	$n = 3$	
0,77459 66692 41483		0,55555 55555 55556
0,00000 00000 00000		0,88888 88888 88889

2.3.5. TERMO INDEPENDENTE

Em relação ao cálculo do integral da expressão (2.25), é necessário, à semelhança do que se fez para a expressão (2.24), estabelecer a relação entre as coordenadas locais e globais. Assim sendo,

Fig. 2.5 – Elemento de contorno $d\Gamma$

$$d\Gamma \approx \sqrt{dx^2 + dy^2} \quad (2.47)$$

onde:

$$dx = \frac{\partial x}{\partial \xi} d\xi + \frac{\partial x}{\partial \eta} d\eta \quad (2.48)$$

$$dy = \frac{\partial y}{\partial \xi} d\xi + \frac{\partial y}{\partial \eta} d\eta \quad (2.49)$$

Contudo, podemos simplificar estas expressões na medida em que um elemento do contorno tem de estar sempre na periferia do elemento e, portanto, pela definição imposta ao referencial local, uma das coordenadas é constante, tendo, por isso, derivada nula. Suponha-se $\eta = -1$. Nesse caso:

$$d\eta = 0 \quad (2.50)$$

Substituindo em (2.48) e (2.49) e, por sua vez, as novas expressões em (2.47), obtém-se:

$$d\Gamma = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2} \cdot d\xi \quad (2.51)$$

que pode ainda ser reescrita da seguinte forma:

$$d\Gamma = \sqrt{J_{11}^2 + J_{12}^2} \cdot d\xi \quad (2.52)$$

e portanto a expressão (2.25) assume o aspecto que se segue:

$$\{f_i^e\} = - \int_{-1}^1 N_i(\xi, -1) \cdot v_n(\xi, -1) \cdot \sqrt{J_{11}^2(\xi, -1) + J_{12}^2(\xi, -1)} \cdot d\xi \quad (2.53)$$

2.3.6. FUNÇÃO DE CORRENTE

A função de corrente é muito útil ao estudo de um determinado escoamento, traçando o caminho percorrido pela água, mas normalmente não é utilizada para a resolução do sistema, pois ao contrário da função potencial, a introdução das suas condições fronteira é problemática. Antes de mais, porque implica o conhecimento do caudal total percolado, pois, pela definição da secção 1.4.2.3 se percebe que a função assume esse valor nos pontos que limitam o problema superiormente. Contudo, vimos também nessa secção que a função de corrente se relaciona com a função potencial. Tirando partido dessa relação é possível calcular a função de corrente com os resultados obtidos para a função potencial, tendo apenas de definir a função num único ponto. Define-se então um novo sistema de equações lineares [7]

$$[K]\{\psi\} = \{f\} \quad (2.54)$$

onde $\{\psi\}$ é um vector coluna dos valores nodais da função de corrente e os elementos do vector coluna $\{f\}$ e a matriz $[K]$ definem-se assim

$$f_i = \int_{\Omega} \left(\frac{\partial N_i}{\partial x} v_y - \frac{\partial N_i}{\partial y} v_x \right) d\Omega \quad i = 1, \dots, M \quad (2.55)$$

$$K_{ij} = \int_{\Omega} \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \quad i, j = 1, \dots, M \quad (2.56)$$

No caso axissimétrico, se x for usado como coordenada radial e o y para a axial o vector dos termos independentes é calculado da mesma forma, mantendo-se válida a expressão 2.55, mas os elementos da matriz $[K]$ passam a ser obtidos a partir de

$$K_{ij} = \int_{\Omega} \frac{1}{x} \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \quad i, j = 1, \dots, M \quad (2.57)$$

3

PROGRAMA DE CÁLCULO AUTOMÁTICO

3.1. INTRODUÇÃO

Este capítulo tem como objectivo a descrição do programa QUASAR_M desenvolvido em MATLAB. Esta, por sua vez, vai ser dividida em três partes, sendo elas, o pré-processamento, o cálculo e o pós-processamento.

O desenvolvimento foi baseado no programa QUASAR_F, que, estando escrito em linguagem FORTRAN, não dispunha de pré processamento, requerendo, por isso, a existência de um ficheiro de dados. O pós-processamento era auxiliado pelo programa DRAWMESH, que, para a situação em causa, se tornava pouco ágil, uma vez que não foi concebido para este fim em particular.

O ambiente MATLAB permitiu a criação de uma interface gráfica que gera o ficheiro de dados e permite a visualização dos resultados de forma intuitiva.

3.2. PRÉ PROCESSAMENTO

Nesta secção pretende-se fazer um acompanhamento da introdução dos dados do problema, alertando para os cuidados a ter e as convenções adoptadas, servido assim de manual de utilização do programa.

Para auxiliar em termos visuais as descrições vão ser acompanhados dum exemplo muito simples composto apenas por três elementos de quatro nós que representam um solo homogéneo e isotrópico com permeabilidade 0.005 m/s, sendo o fluxo gerado pela diferença de potencial imposto nos nós que limitam a malha superior e inferiormente.

3.2.1. DIMENSIONAMENTO DA PÁGINA E DEFINIÇÃO DO PROBLEMA

Ao iniciar o programa, abre-se uma janela que pergunta se se pretende criar um ficheiro de dados ou ler um existente. Note-se que se manteve a hipótese de ler um ficheiro de dados, não só para que se pudessem utilizar os ficheiros desenvolvidos para o programa QUASAR_F, mas principalmente para que fosse possível fazer uma pequena alteração sem ter de se criar um novo ficheiro.

Ao escolher a opção “Criar”, abre-se uma nova janela onde se dimensiona a área de desenho da malha e se define as características do problema (plano vs. axissimétrico; 2 vs. 3 pontos de Gauss; confinado vs. não confinado). As janelas correspondentes encontram-se nas figuras 3.1 e 3.2, respectivamente.

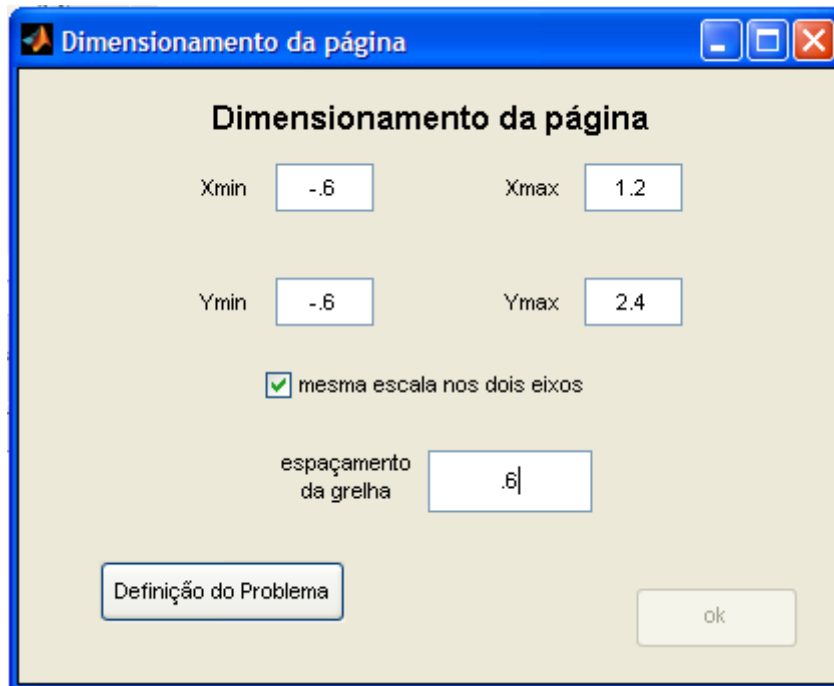


Fig. 3.1 – Dimensionamento da página

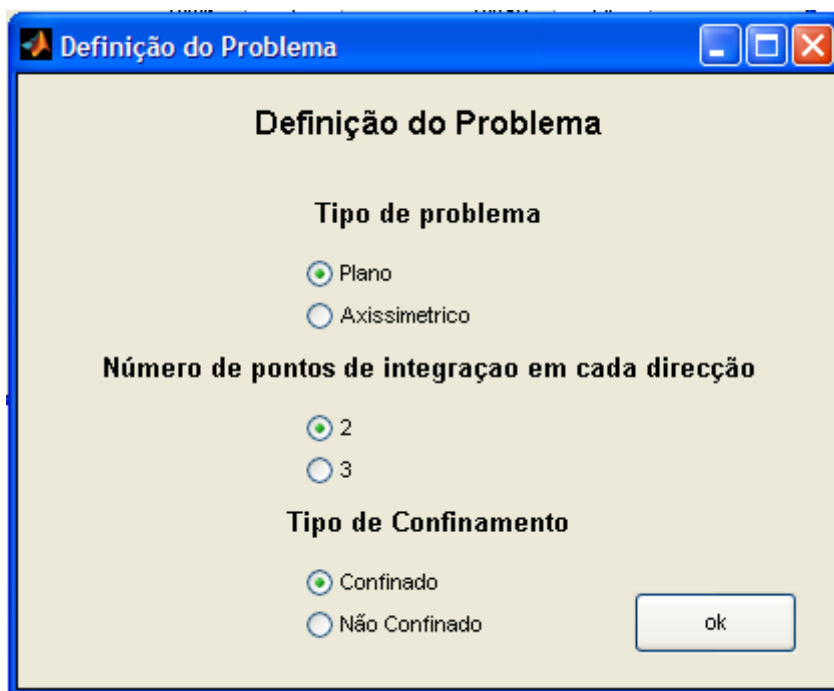


Fig. 3.2 – Definição do Problema

Posto isto, abre-se a última janela da fase de pré processamento onde se vai desenhar a malha, introduzir as condições fronteira e os caudais impostos.

3.2.2. GERAÇÃO DO FICHEIRO DE DADOS

3.2.2.1. Desenho da malha

Esta janela tem uma barra de comandos do lado direito e o restante espaço destina-se à área de desenho propriamente dita. No canto superior direito encontram-se as coordenadas do cursor, quando este está sobre a área de desenho. O número de casas decimais dos valores apresentados depende do estado da opção “Snap”, pois caso esta opção esteja activada a posição do cursor lida é a correspondente ao par de coordenadas arredondado às unidades mais próximo. Um pouco mais abaixo encontram-se seis botões que permitem desenhar a malha. Os três do lado esquerdo são “Desenhar”, “Seleccionar” e “Mover” e funcionam de forma articulada. Há sempre um seleccionado, mas não mais do que um. Por definição o botão seleccionado é o “Desenhar”. Os restantes três estão associados a acções instantâneas e são o “Undo”, “Copiar” e “Eliminar”.

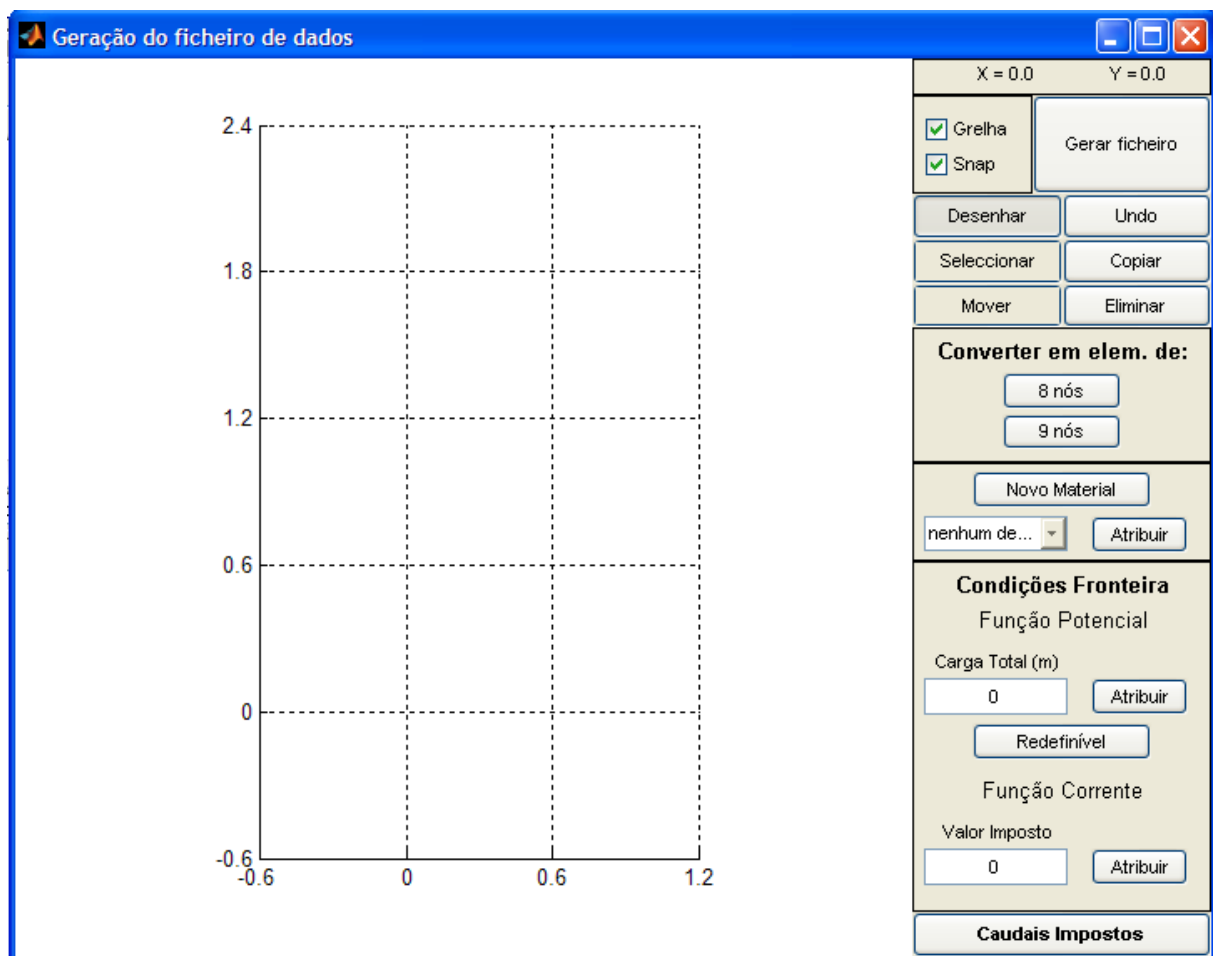


Fig. 3.3 – Geração do ficheiro de dados

O botão “Desenhar” permite, precisamente, desenhar os elementos, clicando com o rato nos pontos correspondentes às coordenadas pretendidas para os nós do elemento que se está a criar, terminando no nó inicial para dar a indicação de que o elemento está concluído. Note-se, contudo, que há certos aspectos a acautelar. Um deles prende-se com a necessidade de especificar os nós no sentido directo,

ou seja, no sentido contrário ao dos ponteiros do relógio. Esta informação aparece no visor quando se coloca o cursor por cima do botão (figura 3.4).

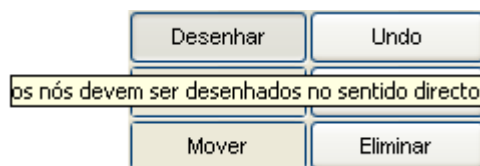


Fig. 3.4 – Indicação da sequência de desenho

Para além disso, importa referir que o programa está preparado apenas para elementos de quatro, oito e nove nós e como estão disponíveis botões que permitem converter os elementos em oito ou nove nós, ao desenhar, basta apenas indicar os nós de canto, uma vez que todos os outros podem ser calculados. Acrescente-se que a conversão de elementos só permite aumentar o número de nós, assim sendo, é possível desenhar um elemento de quatro nós e convertê-lo num de oito e posteriormente num de nove, mas não é possível voltar aos oito nós.

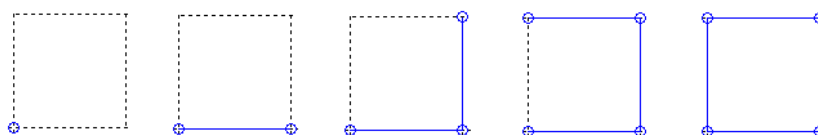
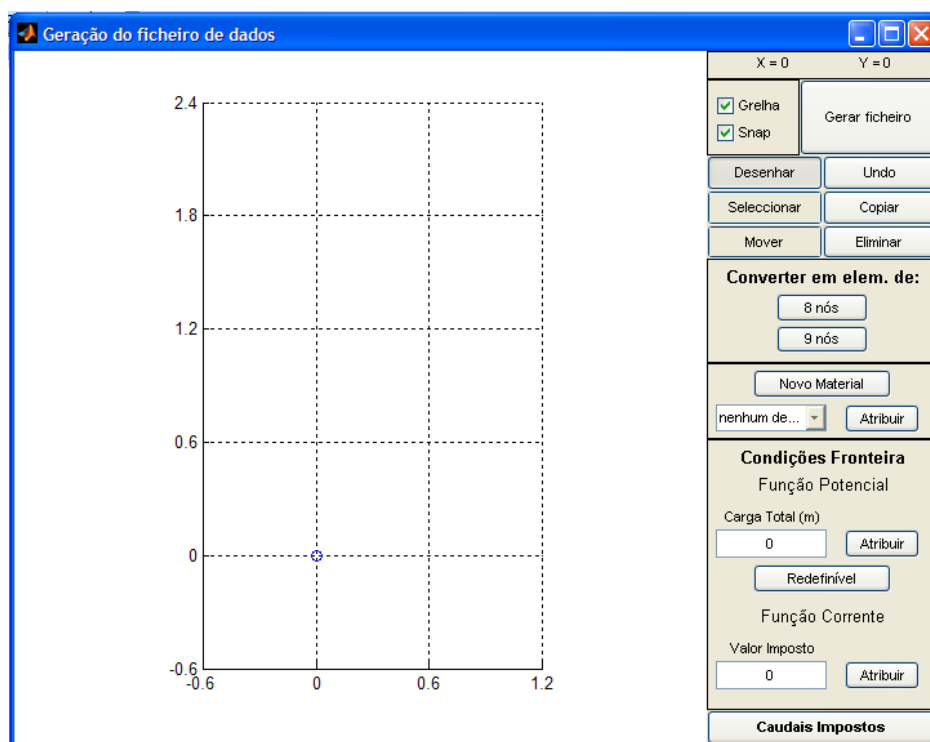


Fig. 3.5 – Desenho de um elemento

O botão “Undo” está apenas funcional em modo de desenho e tem como propósito anular o último nó desenhado. Esta operação é válida até que se tenham anulado todos os nós do elemento em edição. Posto isto, o clique seguinte não executa qualquer acção.

O botão “Seleccionar” disponibiliza duas formas de o fazer. Uma delas é clicando com o cursor por cima da entidade que se pretende seleccionar, seja ela um elemento ou um nó. Depois de seleccionada essa entidade revela o seu estado alterando a sua cor de azul para verde. Um novo clique sobre a mesma faz com que deixe de estar seleccionada e, como tal, a sua cor altera-se em conformidade. Uma outra forma de seleccionar é através da janela. Recomenda-se que se recorra a esta segunda com a opção “Snap” desactivada, para uma melhor percepção da janela. A janela é definida pelo canto inferior esquerdo e pelo superior direito, por esta ordem, e selecciona todas as entidades que se encontrem no seu interior, tornando todas as outras não seleccionadas.

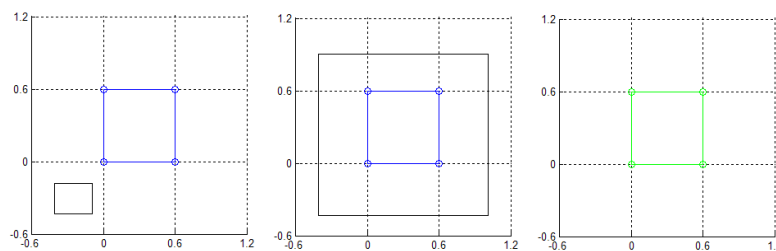
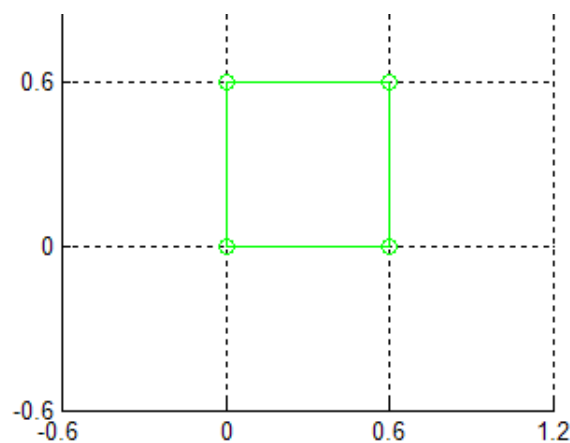


Fig. 3.6 – Seleccionar um elemento

Os botões “Copiar” e “Eliminar”, ao contrário do “Undo”, destinam-se a elementos e não a nós. Assim sendo, “Copiar” permite duplicar os elementos seleccionados, mantendo todas as propriedades originais, incluindo as coordenadas dos nós, pelo que será necessário recorrer ao botão “Mover” para deslocar os elementos copiados para as posições pretendidas. Note-se que ao copiar um elemento o original passa a não seleccionado e o duplicado a seleccionado.



Aviso: Objecto copiado!

Fig. 3.7 – Copiar um elemento

O botão “Mover” requer a introdução gráfica das coordenadas inicial e final do vector deslocamento, trabalhando em termos relativos, pelo que importa apenas a diferença das abcissas e das ordenadas entre o primeiro e o segundo ponto. Recomenda-se que após a movimentação dos elementos pretendidos se volte ao modo de selecção para evitar cliques perdidos que retirem da posição pretendida os elementos que ainda se encontrem seleccionados.

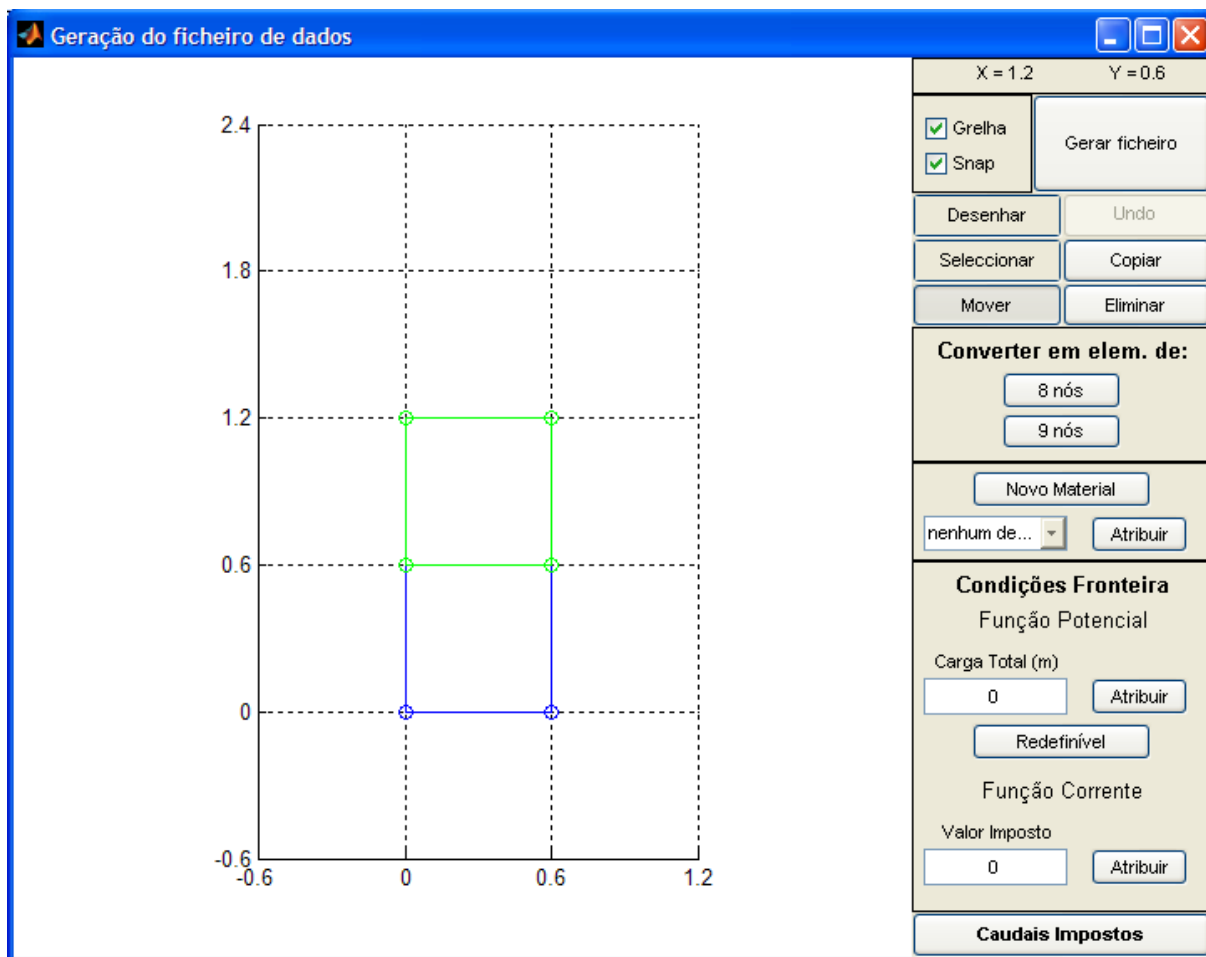


Fig. 3.8 – Mover um elemento

Esta forma que se escolheu para desenhar a malha, elemento a elemento, permite uma versatilidade sem limites, apesar de poder parecer morosa para grande malhas. Recorde-se, contudo, que no caso de se querer tirar partido da repetição dos elementos o número de elementos duplica a cada iteração, crescendo assim de forma exponencial.

3.2.2.2. Atribuição de propriedades

As ferramentas apresentadas até agora permitem o desenho integral da malha, pelo que resta apenas atribuir propriedades. Note-se que a numeração é feita de forma automática depois de se concluir a introdução dos dados. A lógica da atribuição das propriedades é sempre a mesma, apenas com duas excepções que se referem de seguida, e consiste em seleccionar a entidade a que se vai atribuir a

propriedade e, depois de escolher a propriedade ou escrever o valor no campo respectivo, carregar no botão “Atribuir” associado.

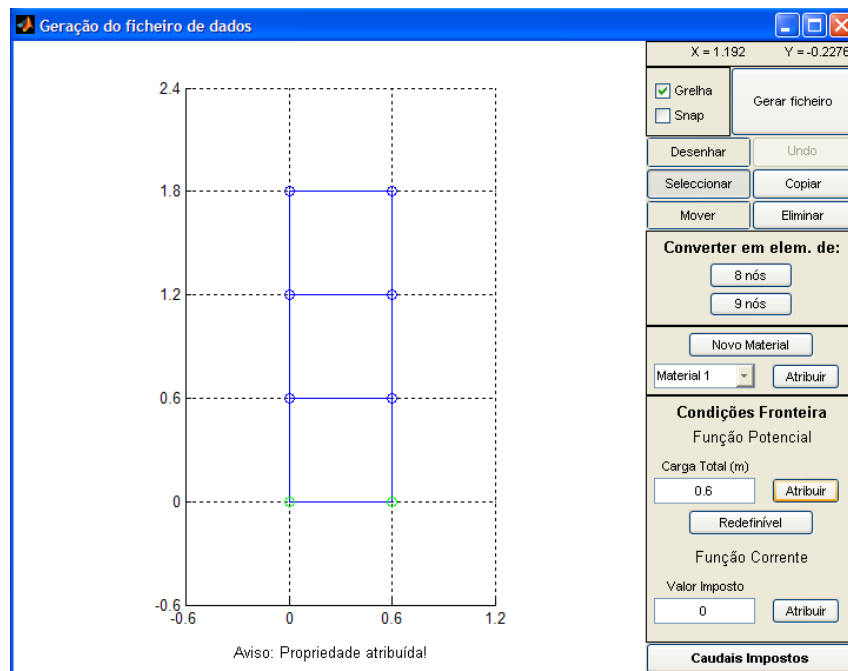


Fig. 3.9 – Atribuir potencial nos nós inferiores

No caso dos materiais, é necessário criar o material para que ele apareça na lista dos materiais disponíveis e possa ser atribuído. Para isso, basta clicar em “Novo Material”, abrindo-se de seguida uma nova janela onde se podem introduzir as quatro propriedades que o caracterizam. A numeração do material é sequencial e imposta.

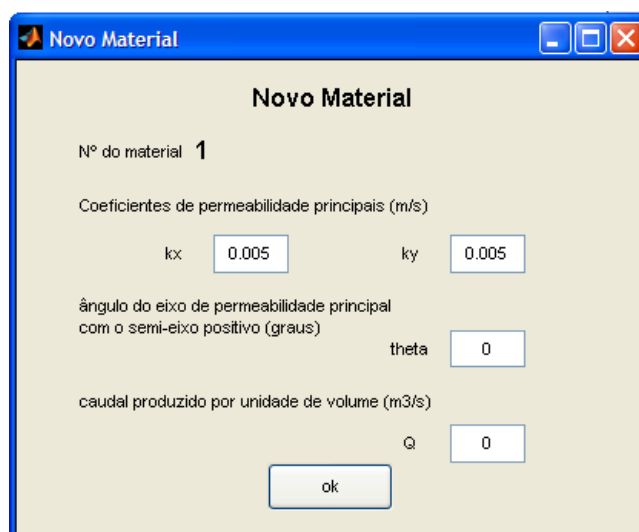


Fig. 3.10 – Novo material

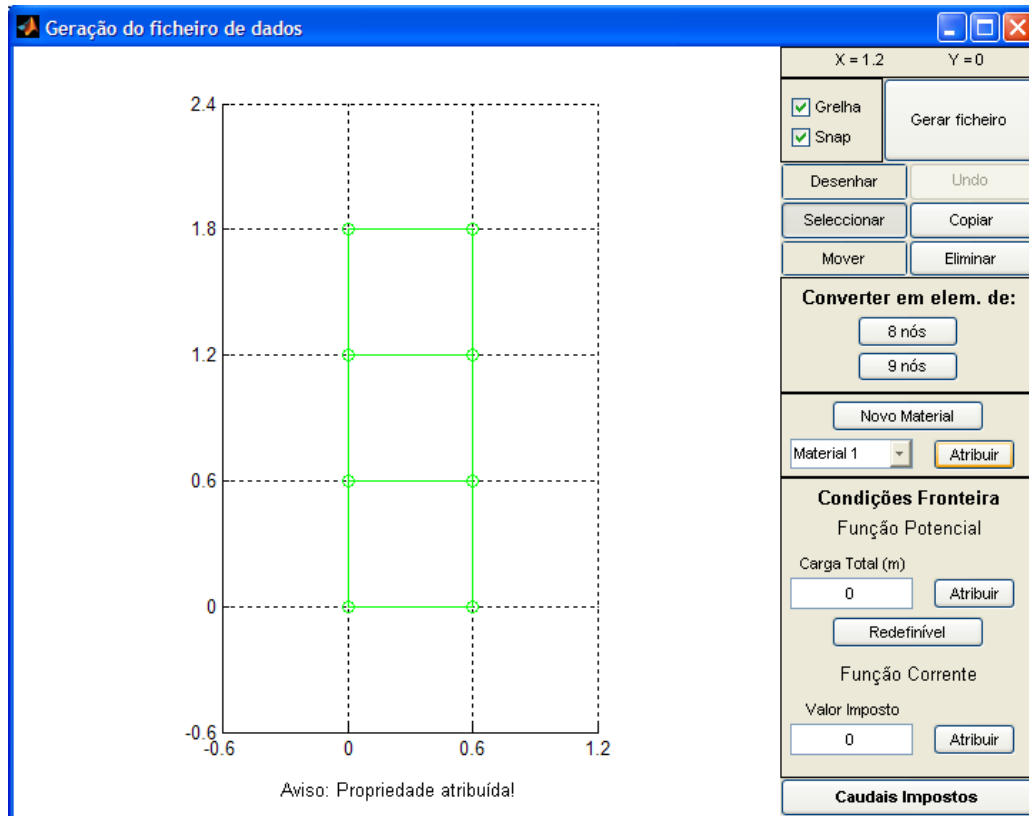


Fig. 3.11 – Atribuição do material

Os menus relativos às condições fronteira e aos caudais impostos são rotativos, alternando no botão que se encontra no final da barra de comandos.

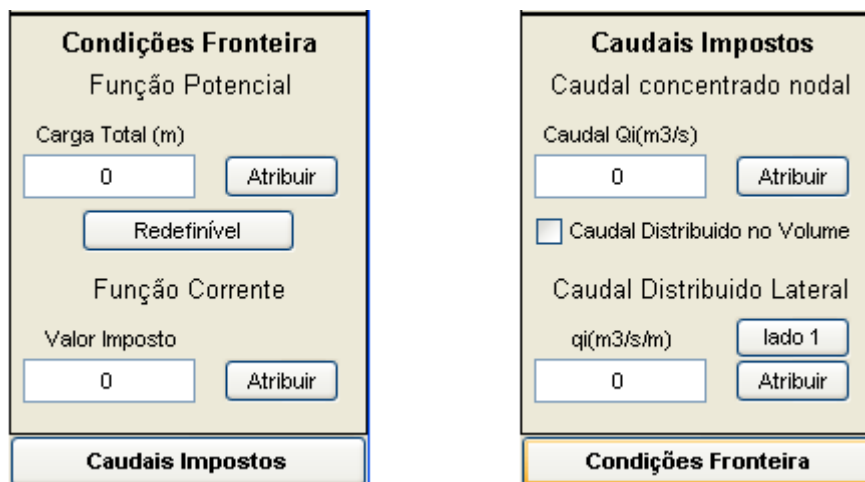


Fig. 3.12 – Menu rotativo

A primeira excepção das duas acima referidas está relacionada com a identificação dos nós redefiníveis, bastando seleccionar os nós pretendidos e clicar em “Redefinível”. A outra, não tão óbvia está relacionada com o caudal distribuído lateral. Por cima do botão “Atribuir” relativo a esta

propriedade está um outro que por defeito está legendado “lado 1”. Para atribuir o caudal distribuído lateral é necessário seleccionar o elemento a que o lado pertence, carregando em seguida no tal botão cuja legenda é a numeração sequencial dos lados e só depois atribuir o valor a cada nó desse lado, fazendo-o no sentido directo, em relação ao elemento a que pertencem. Estes três passos devem ser repetidos para cada lado. No caso dos dois ou três nós do lado em questão, consoante se trate de um elemento de 4 ou mais nós, terem todos o mesmo valor associado, correspondendo ao caso de um caudal uniforme, é possível atribuir o respectivo valor de uma só vez, seleccionando simultaneamente todos os pontos pretendidos.

Acrescente-se que no caso do caudal distribuído no volume a informação que se pretende do utilizador é apenas a existência ou não do mesmo, sendo, por isso, transmitida pelo visto, em caso afirmativo, na respectiva caixa. O valor associado já está definido nas propriedades do material.

Ao colocar o cursor sobre as unidades de cada um dos tipos de caudal, aparecem as indicações em relação aos sentidos e respectiva relação com os sinais. No caso do caudal concentrado nodal, “ Q_i positivo = fonte; Q_i negativo = poço”. No caudal distribuído lateral, “ q_i é positivo, quando dirigido para o interior”.

Depois de introduzir todos os dados do problema deve clicar-se em “Gerar ficheiro”, para que a informação gráfica seja convertida num ficheiro de texto. Assim que se solicita essa operação abre-se uma nova janela onde se pretende que se indique o nome do ficheiro de dados, que deverá ter a extensão .DAT. Note-se que o nome atribuído ao ficheiro vai constar na visualização de resultados. Logo de seguida abre-se uma janela semelhante, mas agora para indicar o ficheiro que se pretende ler. No caso de se ter escolhido a opção “Ler” na abertura do programa era a partir deste ponto que se iniciava.

3.3. CÁLCULO

3.3.1. INTRODUÇÃO DA TOPOLOGIA DA MALHA

3.3.1.1. Leitura dos Parâmetros de Controlo

A primeira fase diz respeito à introdução da topologia da malha, ou seja, da sua configuração, forma como se organiza. Note-se que à medida que o programa vai avançando o utilizador vai sendo informado das tarefas em execução na *Command Window*.

O *script* QINPT1 lê os parâmetros de controlo e chama a função QCHEK1 para identificar anomalias na introdução dos dados, como seja um número de materiais, ou de nós, negativo, ou número de pontos de Gauss diferente de dois ou três, tal como estipulado *a priori*. Estas verificações são necessárias pela possibilidade de criação manual do ficheiro de dados e consequentemente da inevitável possibilidade de erro humano.

3.3.1.2. Leitura da Topologia da Malha

O *script* QINPT2 lê as ligações nodais dos elementos, armazenando essa informação por colunas na matriz LNODES, correspondendo cada coluna a um elemento. As coordenadas dos pontos são armazenadas na matriz COORD, sendo a primeira linha destinada às abcissas e a segunda às ordenadas. No caso dos nós não estarem todos definidos ao nível das coordenadas, especificando apenas os nós de canto, é chamado o *script* NODE2D, que se encarrega de interpolar as coordenadas dos nós situados a meio dos lados de elementos de oito ou nove nós e o nó central para o segundo

caso. Note-se que o último nó numerado tem de estar especificado, uma vez que o número de nós com coordenadas definidas não é conhecido pelo código à partida e o critério de paragem adoptado é o reconhecimento pelo ciclo do último nó.

Mais uma vez é chamada uma função responsável pela verificação, neste caso da topologia da malha. A respectiva função, QCHEK2, permite também identificar nós com coordenadas idênticas, sendo esta informação escrita no ficheiro de resultados. Para além disso, o *script* QINPT2 lê o tipo de elemento, o número do material que lhe está associado e as propriedades dos materiais. Define também as constantes de integração de Gauss-Legendre. Recorde-se que o presente programa está apenas previsto para dois ou três pontos da quadratura gaussiana. O *script* GAUSLG encarrega-se de atribuir, conforme se trate de um caso ou de outro, as constantes definidas na secção 2.3.4. As abcissas ficam guardadas em POSGP e os pesos em WEIGP.

3.3.1.3. Cálculo da Coordenadas dos Pontos de Gauss

O cálculo das coordenadas dos pontos de Gauss é feito elemento a elemento em GPCORD, guardando-se para cada um as coordenadas dos nós desse elemento no vector ELCOD. Posteriormente, lêem-se as coordenadas locais ξ e η do vector POSGP anteriormente referido, sendo atribuídos os seus valores às variáveis EXISP e ETASP, respectivamente. De seguida chama-se a função SHAPE2, cujo objectivo é calcular as funções de interpolação definidas na secção 2.3.2. Note-se que na condição correspondente ao elemento de quatro nós as variáveis SM e TM correspondem aos resultados das expressões (2.37) e (2.38). A cada posição i do vector SHAPE corresponde a função de interpolação N_i , estando assim definido o vector $[N]$ da expressão (2.28). A importância de iniciar a lista de nós pertencentes a um elemento por um nó de canto e terminar no nó central prende-se com o facto das funções de interpolação serem distintas, como se viu na secção referida acima, e estarem definidas dessa forma no vector SHAPE. Se não se respeitar esta ordem, as funções não correspondem ao tipo de nó. O cálculo propriamente dito é então executado segundo a expressão (2.30) e os valores são armazenados na matriz GPCOD, correspondendo a primeira linha às abcissas e a segunda às ordenadas.

3.3.2. CÁLCULO DOS POTENCIAIS NOS NÓS

3.3.2.1. Leitura das Condições Fronteira de Dirichlet

O *script* PBOUND encarrega-se de ler as condições fronteira da função potencial, guardando o índice dos nós com carga total imposta pela atribuição do valor unitário a essa posição no vector IPFIX e à mesma o valor imposto no vector PFIXD.

3.3.2.2. Formação da Matriz de Coeficientes Global

À semelhança do que foi feito anteriormente, os cálculos são feitos elemento a elemento e as propriedades do material associado a cada elemento são guardadas temporariamente em variáveis afectas a esse propósito para economizar o tempo de localizar o ponteiro na posição correcta do vector ou matriz em causa. Assim, o número do material é guardado em IMATS, os coeficientes de permeabilidade principais são armazenados em PERMB e o ângulo do eixo de permeabilidade principal com o semi-eixo cartesiano positivo é guardado em THETA. De seguida calcula-se a matriz de permeabilidade em relação a xy , no caso de THETA ser diferente de zero e os valores das direcções principais serem distintos. Se fossem iguais significaria que o solo era isotrópico,

consequentemente a matriz que define a permeabilidade em qualquer direcção seria sempre a mesma. Admitindo então que era necessário proceder à rotação do referencial, a matriz D calculava-se do seguinte modo:

$$[D] = [T] \cdot \begin{bmatrix} k_{x'} & 0 \\ 0 & k_{y'} \end{bmatrix} \cdot [T]^T \quad (3.1)$$

sendo D a matriz definida em (2.29), e T é

$$[T] = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (3.2)$$

onde θ é o ângulo guardado em THETA.

O passo que se segue é a determinação das derivadas cartesianas das funções de forma. Para isso, invoca-se a função JACOB2, que por sua vez chama a DERIV2 que calcula as derivadas das funções de forma em relação às coordenadas locais, guardando na coluna i da matriz DERIV:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \quad (3.3)$$

Recorde-se que a matriz Jacobiana foi definida como:

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (2.41)$$

e que x e y foram definidos na expressão (2.30) do seguinte modo:

$$\begin{aligned} x &= \sum_{i=1}^M x_i \cdot N_i \\ y &= \sum_{i=1}^M y_i \cdot N_i \end{aligned} \quad (2.30)$$

Substituindo (2.30) em (2.41) obtemos:

$$[J] = \begin{bmatrix} \sum_{i=1}^M \frac{\partial N_i}{\partial \xi} \cdot x_i & \sum_{i=1}^M \frac{\partial N_i}{\partial \xi} \cdot y_i \\ \sum_{i=1}^M \frac{\partial N_i}{\partial \eta} \cdot x_i & \sum_{i=1}^M \frac{\partial N_i}{\partial \eta} \cdot y_i \end{bmatrix} \quad (3.4)$$

que corresponde ao produto de DERIV por ELCOD e é guardado em XJACM. O determinante de XJACM é guardado em DJACB e a inversa da matriz Jacobiana em XJACI. Ambos são calculados com recurso a funções disponíveis em MATLAB para o efeito. Para obter o vector $\{B_i\}$ da expressão (2.27) falta ainda multiplicar a matriz inversa da Jacobiana por (3.3). Esse produto é atribuído a CARTD. Em DVOLU fica o produto do determinante da matriz Jacobiana calculado na função JACOB2 pelos pesos definidos na expressão (2.46), tendo o cuidado de o multiplicar por $2\pi \cdot r$, sendo r o valor da primeira coordenada do nó, no caso de se tratar de uma problema axissimétrico.

Note-se que o ciclo sobre os pontos de Gauss tem dois contadores, sendo eles KGAUS e KGASP, sendo o primeiro local e o segundo global. Este tem apenas a função de identificar um eventual ponto com determinante negativo para ajudar a localizar o erro.

Todos os elementos da expressão (2.44) são agora conhecidos e, portanto, basta apenas proceder às multiplicações, sendo o integral calculado à custa dos pesos, como se referiu. Calculada a matriz K^e da expressão (2.24), que tem o nome de ESTIF, resta agora proceder ao seu espalhamento na matriz global K da expressão (2.21).

```
% No. DOS GDL E COORDENADAS DOS NOS DO ELEMENTO
NODES=LNODS(1:NNODE,IELEM);
ELCOD=COORD(:,NODES);

% CICLOS SOBRE OS PONTOS DE GAUSS
KGASP = 0;
for IGAUS = 1:NGAUS
    EXISP = POSGP(IGAUS);
    for JGAUS = 1:NGAUS
        KGAUS = KGAUS+1;
        KGASP = KGASP+1;
        ETASP = POSGP(JGAUS);

        % DETERMINACAO DAS DERIVADAS CARTESIANAS DAS FUNCOES DE FORMA E
        % DO ELEMENTO DE VOLUME
        [CARTD,DJACB]=JACOB2(EXISP,ETASP,...
            NDIME,NNODE,ELCOD,KGASP,IELEM,resultados);
        DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS);
        if NTYPE == 2
            DVOLU = DVOLU*TWOPI*GPCOD(1,KGAUS);
        end

        % CALCULO DA MATRIZ DE RIGIDEZ DO ELEMENTO
        ESTIF=ESTIF+CARTD'*PERMB*CARTD*DVOLU;
    end
end
```

3.3.2.3. Formação do Termo Independente para a Solução em Termos de Potencial

Na definição dos caudais há três tipos distintos de caudal que se pode impor, sendo eles caudal concentrado nodal; caudal distribuído lateral; caudal distribuído no volume. O *script* que se encarrega da formação do termo independente em termos de potencial é o PLOAD0.

Em relação aos caudais concentrados nodais, basta apenas colocar o respectivo valor no vector GLOAD, nome do termo independente, tendo também o cuidado de o multiplicar por $2\pi \cdot r$, no caso de se tratar de um problema axissimétrico.

No que diz respeito ao caudal distribuído lateral, é necessário proceder ao cálculo do integral da expressão (2.53), uma vez que se adoptou no código o mesmo critério que na secção 2.3.5 de assumir que $\eta = -1$. Assim, os elementos da matriz Jacobiana estão em DGAUS, calculadas tal como indicado em (3.4) e a parcela correspondente ao segundo membro da expressão (2.52) designa-se DJACB. A variável DAREA já incorpora o produto pelo peso, tal como previsto em (2.46). As mesmas precauções tiveram de ser tomadas em relação ao caso axissimétrico, sendo que desta vez, como não se trata apenas de um ponto, o raio deve ser entendido como o somatório do produto das funções de forma pelas abcissas. O cálculo de v_n tal como indicado na secção 2.3.1 está guardado em QGAUS. O valor do integral é então obtido à custa do produto de QGAUS pela respectiva função de forma e por DAREA.

Por último, falta apenas preencher o vector GLOAD com as parcelas relativas ao caudal distribuído no volume. Recorde-se que o valor do caudal está associado ao material do elemento. Esse valor é lido para a variável QVALU elemento a elemento e no caso de ser nulo os pontos de Gauss correspondentes a esse elemento são avançados. Neste caso, já não se trata do integral dum elemento de contorno, mas dum elemento de área e portanto, tal como para a matriz de permeabilidade $d\Omega = |J| \cdot d\xi \cdot d\eta$.

Este *script* contém ainda a determinação dos caudais totais prescritos positivos e negativos e escreve o GLOAD inicial num ficheiro auxiliar para uso no caso de percolação não confinada e no ficheiro de resultados no caso do utilizador o ter solicitado.

3.3.2.4. Resolução do Sistema de Equações

A rotina FORTRAN GAUSS que resolvia o sistema de equações

$$[K]\{\phi\} = \{f\} \quad (2.21)$$

efectuando a redução e retrossubstituição de Gauss, foi substituída simplesmente pelo recurso ao operador *backslash* que resolve o sistema pelo método que garantir melhores resultados.

A introdução das condições de fronteira (*boundary conditions* na língua inglesa) no sistema é feita à custa da função BDCOND. Não é mais do que identificar o elemento da diagonal em PIVOT, somar o produto dum número muito elevado por aquele e substituir o valor do termo independente pelo produto do elemento da diagonal, pelo número grande usado anteriormente e pelo valor que se pretende para o potencial. Note-se que os vectores IFFIX e FIXED correspondem aos vectores IPFIX e PFIXD, respectivamente, descritos na secção 3.3.2.1.

Ao resolver o sistema $Ax = b$, o valor x_i é calculado pelo quociente $\frac{b_i}{a_{ii}}$, sendo neste caso BIG o tal número muito elevado e FIXED o valor que se pretende para x_i

$$x_i = \frac{BIG \times PIVOT \times FIXED}{PIVOT + PIVOT \times BIG} \quad (3.6)$$

Esta expressão é equivalente a

$$x_i = \frac{FIXED}{\frac{1}{BIG} + 1} \quad (3.7)$$

que em termos computacionais é igual a FIXED uma vez que o quociente $\frac{1}{BIG}$ é aproximadamente nulo.

3.3.2.5. Verificação do Tipo de Escoamento

A função FLOWTP percorre todos os nós da malha e compara o seu potencial com a cota do ponto, fazendo a contagem do número de pontos em que o potencial é menor, o que significa que a pressão é negativa e que o nível freático se encontra abaixo desse ponto. No caso do contador ser diferente de zero o escoamento é não confinado. Esta função tem também o objectivo de verificar a informação fornecida pelo utilizador no que diz respeito ao tipo de confinamento, pois pode dar-se o caso de se julgar que se tratava de um problema não confinado, quando não verdade não o é e assim evita-se percorrer o código relativo a esta parte.

3.3.2.6. Tratamento dos Escoamentos Não Confinados

Estando confirmado que o escoamento é não confinado, recorde-se que se introduzem nos dados dois parâmetros relativos ao método da permeabilidade variável, sendo eles a razão entre os valores final e inicial da permeabilidade que é atribuída a FPERM e a tolerância de convergência, TOLCV, que regula o método iterativo.

Cada iteração é iniciada pelo cálculo da velocidade nos pontos de Gauss, que sofre alterações devido à alteração da permeabilidade, e consequentemente altera as reacções, o que se reflecte no termo independente. A diferença de potencial é calculada pela resolução do sistema composto pela matriz de coeficiente original, calculada em PSTIF0 e pelo termo independente afectado das alterações acima referidas. Note-se que os pontos com cargas totais impostas não sofrem alterações e portanto é imposto um DELTH nulo nesses pontos antes de resolver o sistema.

Depois de resolvido o sistema é verificada a convergência do processo iterativo recorrendo-se ao *script* PCONVR. Em cada ponto é somada a diferença de potencial calculada. A tolerância de convergência é comparada com a raiz quadrada do quociente dos quadrados da diferença de potencial pelo potencial final, ou seja, acrescido da diferença calculada.

Quando o *flag* CONVR passa de falso a verdadeiro, já se pode definir as condições de fronteira dos pontos identificados como redefiníveis. Os pontos em que o potencial for superior a cota passam a pertencer à lista dos pontos com condição de fronteira imposta e o valor do potencial é substituído pelo da cota desse ponto. Este tipo de condição fronteira corresponde ao troço CD da figura 1.4 e traduz o escoamento da água ao longo do paramento de jusante. Estas alterações obrigam a uma nova iteração.

3.3.2.7. Cálculo da Velocidade e dos Caudais não Equilibrados

No capítulo 1 já se tinha visto que:

$$\begin{Bmatrix} v_x \\ v_y \end{Bmatrix} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{xy} & k_{yy} \end{bmatrix} \begin{Bmatrix} -\frac{\partial H}{\partial x} \\ -\frac{\partial H}{\partial y} \end{Bmatrix} \quad (1.3)$$

e também que:

$$\phi_{(x,y)} = H_{(x,y)} + C \quad (1.5)$$

No capítulo 2 viu-se que:

$$\phi \approx \hat{\phi} = \sum_{i=1}^M \phi_i \cdot N_i \quad (2.13)$$

Relacionando as três expressões obtemos:

$$\begin{Bmatrix} v_x \\ v_y \end{Bmatrix} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{xy} & k_{yy} \end{bmatrix} \begin{Bmatrix} -\sum_{i=1}^M \frac{\partial N_i}{\partial x} \phi_i \\ -\sum_{i=1}^M \frac{\partial N_i}{\partial y} \phi_i \end{Bmatrix} \quad (3.8)$$

Os primeiros passos do QVELOC são semelhantes aos da formação da matriz de coeficientes global, uma vez que também é necessária a matriz de permeabilidade. Os potenciais dos nós do elemento são guardados em HVALU e as derivadas cartesianas das funções de forma, como se viu anteriormente, em CARTD. $\sum_{i=1}^M \frac{\partial N_i}{\partial x} \phi_i$ é guardado em DFIDX e $\sum_{i=1}^M \frac{\partial N_i}{\partial y} \phi_i$ em DFIDY. Estas operações repetem-se para cada ponto de Gauss de cada elemento e os resultados vão sendo guardados em GRADH, pois

correspondem ao gradiente do potencial. O valor do potencial nos pontos de Gauss é calculado com recurso às funções de forma e guardado em HGAUS. Em cada ciclo o valor do gradiente vertical é comparado com os anteriores para se determinar o máximo. Esta informação pode ser muito útil se o caso em estudo for susceptível a instabilidade de origem hidráulica, pois a sua comparação com o gradiente hidráulico crítico permite avaliar a possibilidade de ocorrência de *quick condition* ou levantamento hidráulico. As velocidades, segundo x e y , são por fim calculadas, multiplicando o gradiente pela matriz de permeabilidade e armazenadas em VELOC. Como se viu na secção 1.4.1, a velocidade e o gradiente têm sinais contrários para que a velocidade tenha o sentido do escoamento. No caso do escoamento ser não confinado, a velocidade é corrigida nos pontos com pressão negativa.

```
% CALCULO DE DFI/DX E DFI/DY E DO POTENCIAL NO PONTO DE GAUSS
DFIDX = sum(CARTD(1,1:NNODE).*VECTX(1:NNODE)');
DFIDY = sum(CARTD(2,1:NNODE).*VECTX(1:NNODE)');
HGAUS = sum(SHAPE(1:NNODE).*HVALU(1:NNODE));

% CALCULO DO GRADIENTE HIDRAULICO NO PONTO DE GAUSS
GRADH(1,KGAUS) = DFIDX;
GRADH(2,KGAUS) = DFIDY;
if -DFIDY > -GHVMX
    GHVMX = DFIDY;
    MXGHV = KGAUS;
end

% CALCULO DA VELOCIDADE NO PONTO DE GAUSS
PFACT=1;
if HGAUS < GPCOD(2,KGAUS)
    PFACT=FFPERM;
end
VELOC(:,KGAUS) = -(PERMB*GRADH(:,KGAUS))*PFACT;

%
% CALCULO DE B * v
for INODE = 1:NNODE
    ELRSD(INODE) = ...
        ELRSD(INODE) + (CARTD(1,INODE)*VELOC(1,KGAUS)+...
        CARTD(2,INODE)*VELOC(2,KGAUS))*DVOLU;
end
```

Este *script* serve também para calcular as reacções nodais consequentes das condições de fronteira impostas. À semelhança do que acontece com um sistema de forças e deslocamentos, quando se impõe um determinado deslocamento, desconhece-se a força nesse ponto. Do mesmo modo, quando se fixa um determinado potencial, desconhece-se à partida o caudal desse ponto. Contudo, depois de resolvido o sistema, ou seja, conhecidos todos os potenciais, passam a ser também conhecidos os caudais em todos os pontos e, portanto, também nos pontos com cargas totais impostas.

O cálculo vai então ser feito com recurso à substituição na expressão da matriz dos coeficientes local $[k^e]$ (2.24) do produto das matrizes $[D][B]$ pela velocidade efectiva desse ponto. Continuando a analogia estrutural, este passo corresponde à multiplicação da rigidez, força necessária para obter o deslocamento unitário, pelo deslocamento efectivo do ponto.

Assim sendo, o caudal é então calculado executando um ciclo sobre os elementos e somando as contribuições de cada ponto de Gauss em cada um dos elementos. O vector que guarda os resíduos, ou seja, o caudal não equilibrado, de cada elemento é o ELRSD, que no final do ciclo de cada elemento é adicionado ao vector dos caudais impostos.

3.3.3. CÁLCULO DA FUNÇÃO DE CORRENTE NOS NÓS

Os cálculos para a função de corrente seguem uma lógica muito semelhante aos da função potencial que foram até aqui detalhadamente descritos e, portanto, não será feito um acompanhamento passo a passo desse processo, visto que o código espelha com muita nitidez o cálculos dos integrais da secção 2.3.6.

```
% CALCULO DA MATRIZ DE RIGIDEZ DO ELEMENTO
ISTIF = 0;
for JNODE = 1:NNODE
    DNJDX = CARTD(1,JNODE);
    DNJDY = CARTD(2,JNODE);
    for INODE = 1:NNODE
        DNIDX = CARTD(1,INODE);
        DNIDY = CARTD(2,INODE);
        ISTIF = ISTIF+1;
        ESTIF(INODE,JNODE) = ESTIF(INODE,JNODE) ...
            + (DNIDX*DNJDX + DNIDY*DNJDY)*DVOLU;
    end
end
```

3.4. PÓS PROCESSAMENTO

3.4.1. BARRA DE COMANDOS

Depois de efectuados todos os cálculos, fica disponível um ficheiro de texto com os resultados numéricos e abre-se uma última janela destinada à visualização dos resultados. A apresentação é semelhante à da janela da “Geração do ficheiro de dados”, sendo a parte direita destinada à barra de comandos e a restante à visualização dos resultados. Apesar de ocupar o mesmo espaço esta barra é bastante mais simples, dividindo-se apenas em três partes.

A primeira destina-se a inquirir quais os elementos que se pretendem visíveis, sendo cada campo independente. Por defeito, os campos seleccionados são os “Nós”, a “Malha” e os “Vect. Veloc.” (vectores de velocidade).

A segunda parte, intitulada “Coloração”, destina-se a escolher qual o tipo de informação que se pretende ver traduzida no mapa de cores. O número de cores com que se pretende visualizar os resultados pode ser introduzido pelo utilizador na janela branca adjacente ao título deste grupo de botões, bastando, de seguida, pressionar o “ok” para que o número de cores seja actualizado. As diversas opções são:

- Potencial;
- Corrente;
- Alt. Piez;
- Sem Coloração.

Todas elas se encontram na forma de botões de rádio interligados, o que significa que há sempre um seleccionado, que por definição é a “Alt. Piez”, mas apenas um pode estar seleccionado de cada vez. Note-se que o último botão da primeira parte, correspondente ao desenho da rede de fluxo, só fica disponível quando está seleccionada a opção, “Sem Coloração”. A opção que o antecede que é o desenho do “Nível Freático” também só fica disponível quando a superfície livre se encontra no interior da malha.

A terceira parte tem apenas dois botões, sendo eles o “Save” e o “Print”. Ambos se referem apenas à imagem sem considerar a barra de comandos. Para além disso, a imagem em causa é guardada ou impressa, conforme o botão pressionado, tal qual é visualizada no ecrã.

3.4.2. ÁREA DE DESENHO

O resultado gráfico propriamente dito é encabeçado pelo título do mapa de cores que se está a visualizar, sendo o centro ocupado pela malha e respectivos resultados, enquadrados pelo sistema de eixos a uma distância de dez por cento da amplitude de valores de cada eixo. À direita encontra-se a barra de cores, que identifica o valor associado a cada cor. No caso da opção “Sem Coloração”, ou quando o valor em causa é constante ao longo de toda a malha, a barra não aparece. Para além disso, no segundo caso, aparece escrito, adjacente ao eixo das ordenadas, o valor em causa.

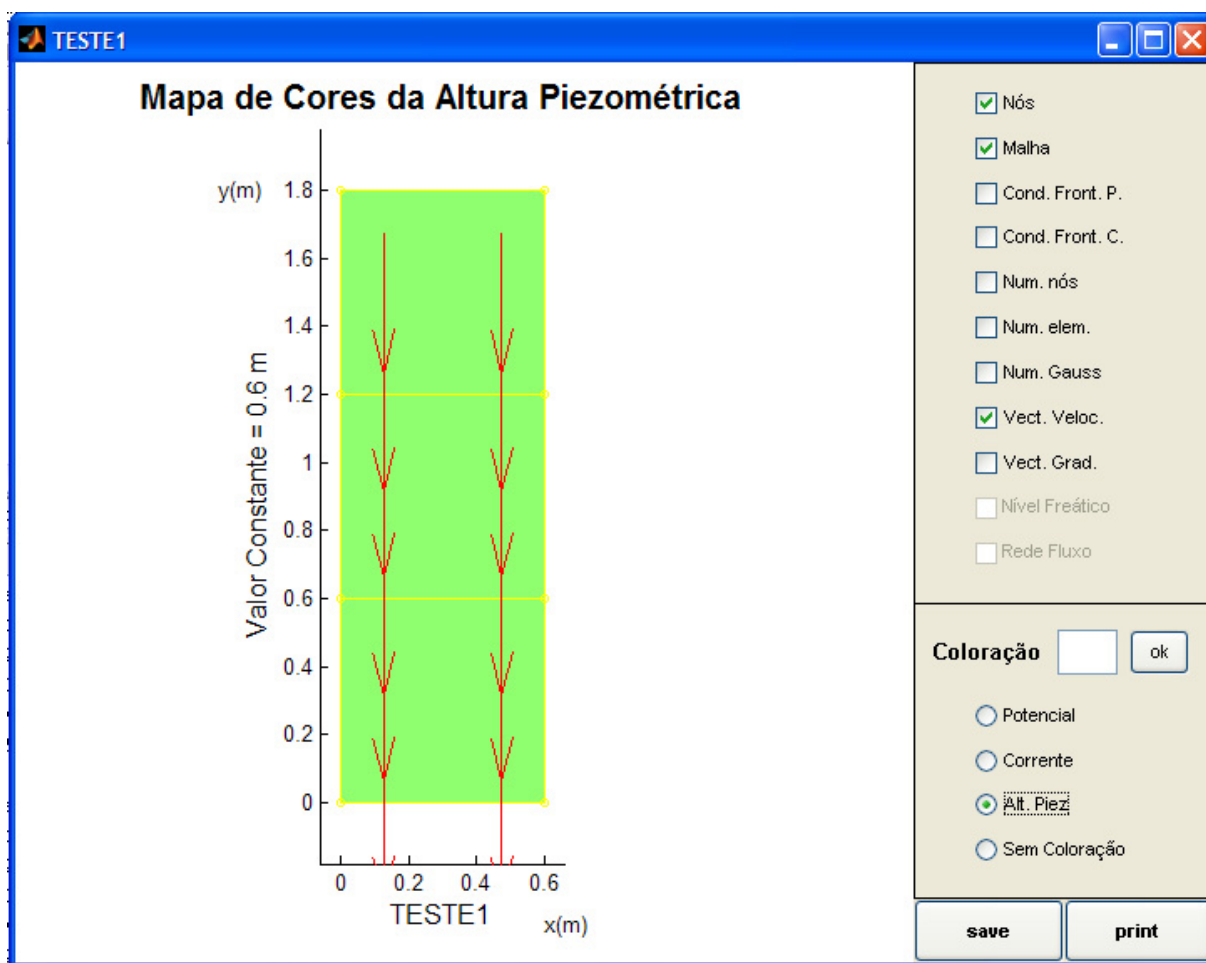


Fig. 3.13 – Visualização de resultados

4

VALIDAÇÃO

4.1. INTRODUÇÃO

Este capítulo tem o objectivo de correr problemas diversos que testem as várias opções postas à disposição pelo programa, sendo elas, o tipo de elemento, o tipo de problema, que pode ser plano ou axissimétrico, o tipo de confinamento, a existência de nós com fronteira redefinível no caso não confinado, e a imposição de caudais, nomeadamente concentrado nodal, distribuído lateral e distribuído no volume.

As soluções encontradas são apresentadas de forma colorida e alternadas com as obtidas pelo programa Quasar em versão Fortran. Houve o cuidado de escolher as mesmas cores, as mesmas divisões e a representação da malha para facilitar a comparação e a conclusão de que os resultados são os mesmos. No último problema apresentado há uma comparação com resultados analíticos.

4.2. CORTINA IMPERMEÁVEL

Este é um problema plano, confinado, que foi extraído de Lambe e Whitman [8] e consiste no fluxo que se estabelece sob uma cortina impermeável com uma altura enterrada igual a metade da altura da camada de solo até à fronteira impermeável. A espessura dessa camada é de 18 m, a altura de água a montante da cortina é de 9 m e a jusante é de 1,5 m. O solo é homogêneo e isotrópico e a permeabilidade é de 5×10^{-4} m/s.

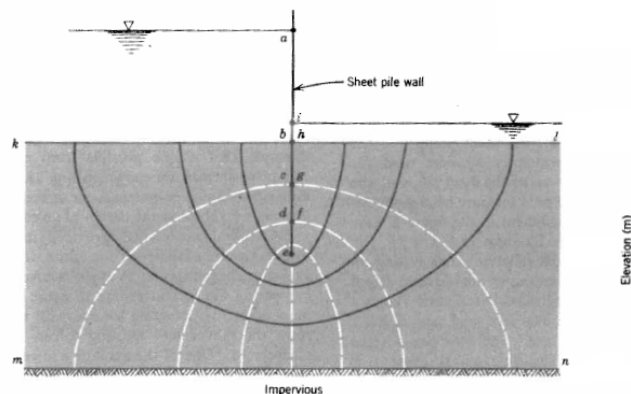


Fig. 4.1 – Problema adaptado de Lambe & Whitman

Adoptou-se uma malha em que a largura era o dobro da altura, correspondendo esta à espessura da camada de solo, composta por 72 elementos dispostos de acordo com a figura 4.2. As condições fronteira impostas encontram-se também assinaladas nessa figura, estando a verde as condições de fronteira associadas ao potencial e a azul o ponto do estrato impermeável suficiente para a solução em termo de função de corrente. A introdução da cortina foi conseguida através da duplicação dos pontos correspondentes à localização da mesma. Deste modo os elementos a montante da cortina (31, 32, e 33) não partilham nós com os elementos a jusante da cortina (37, 38 e 39).

De seguida apresentam-se os mapas das colorações do potencial e da função de corrente da versão Matlab (figuras 4.3 e 4.5) e Fortran (figuras 4.4 e 4.6). Na figura 4.6 está ainda a representação da rede de fluxo, sendo esta uma nova funcionalidade desta versão.

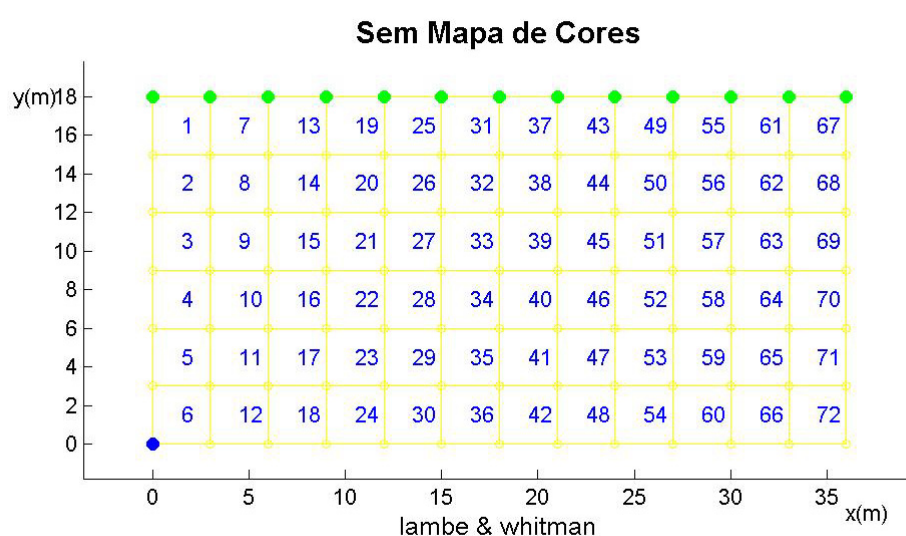


Fig. 4.2 – Cortina impermeável: Esquema da malha adoptada

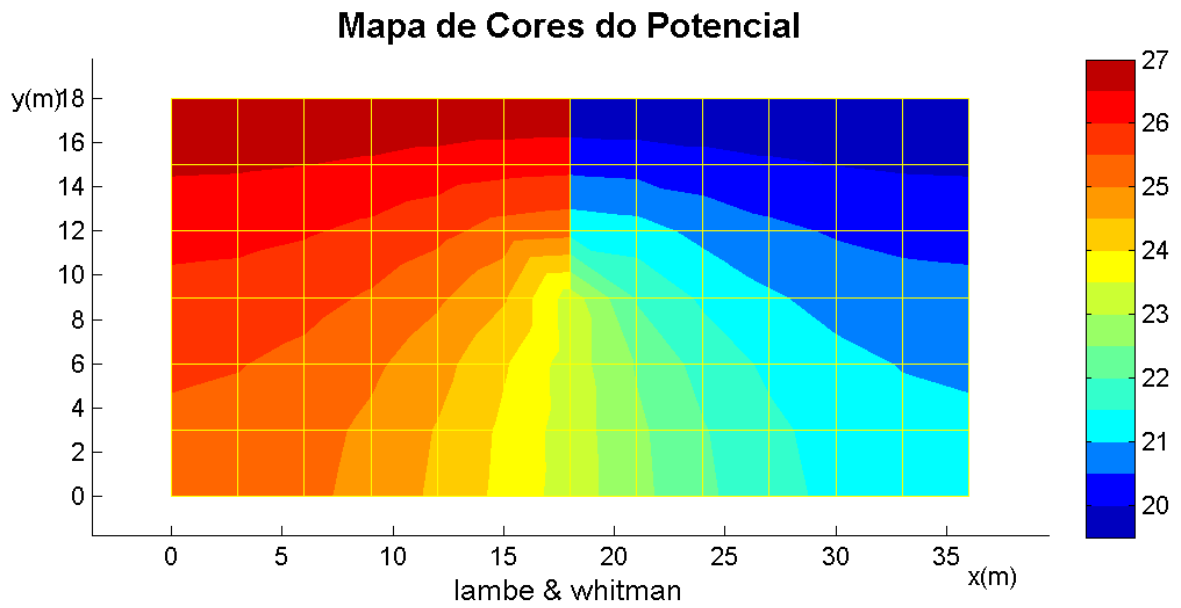


Fig. 4.3 – Cortina impermeável: Coloração da função potencial, versão Matlab

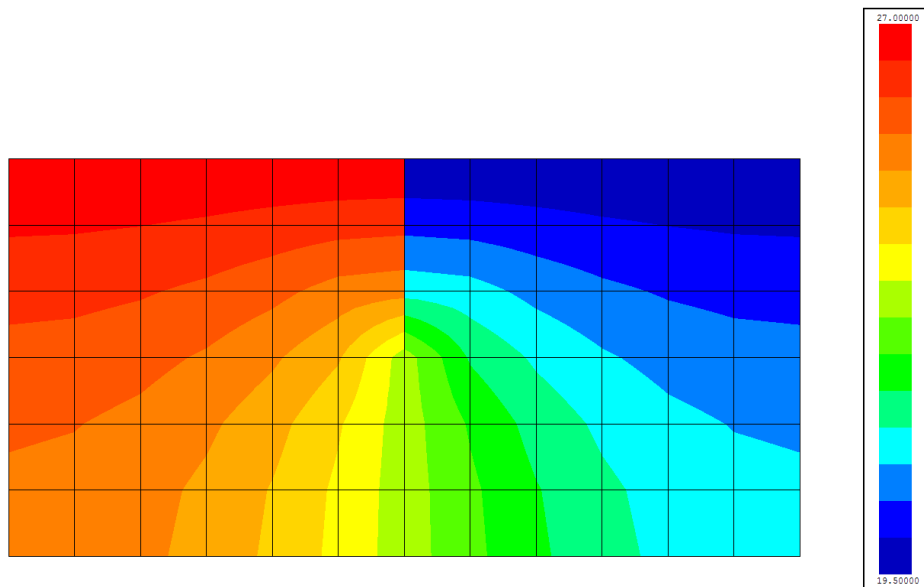


Fig. 4.4 – Cortina impermeável: Coloração da função potencial, versão Fortran

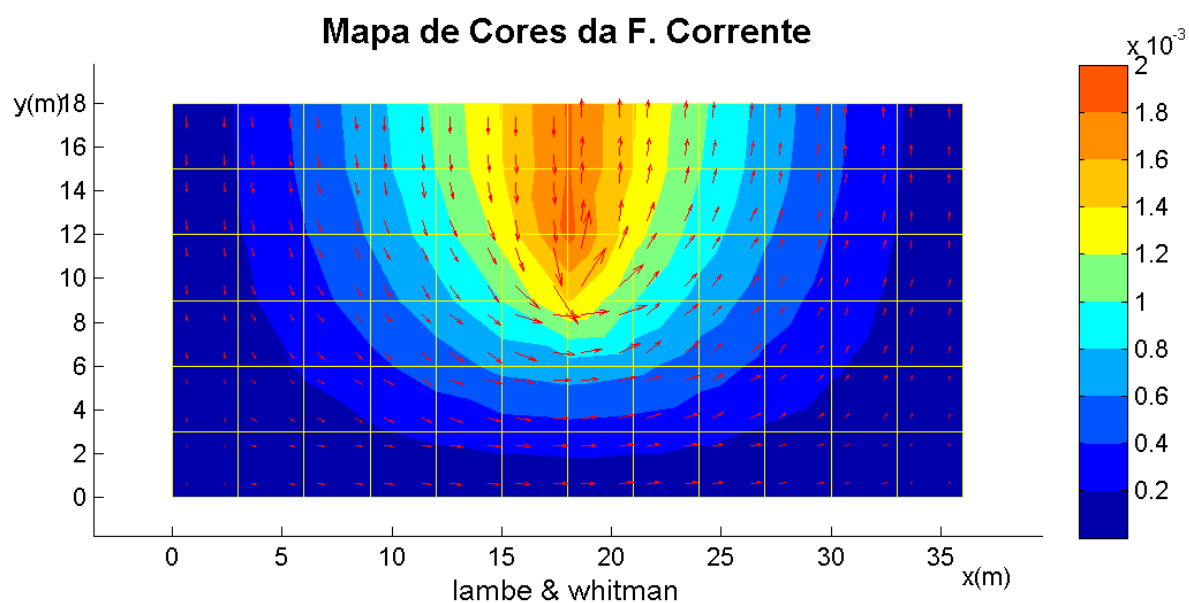


Fig. 4.5 – Cortina impermeável: Coloração da função de corrente com vectores de velocidade, versão Matlab

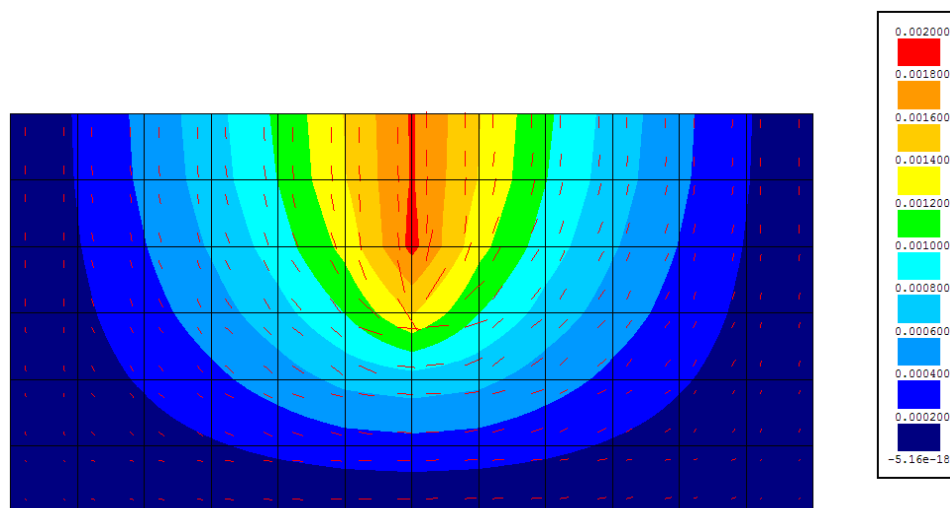


Fig. 4.6 – Cortina impermeável: Coloração da função de corrente com vectores de velocidade, versão Fortran

Sem Mapa de Cores

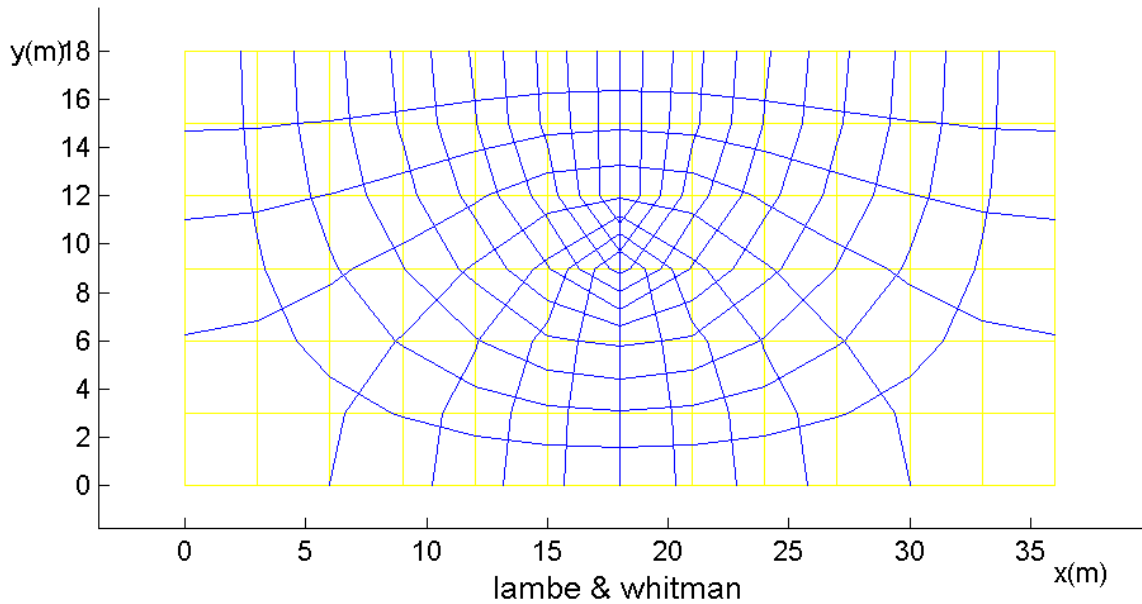


Fig. 4.7 – Cortina impermeável: Rede de fluxo

4.3. AQUÍFERO

Este problema, adaptado de Reddy [9] também ele confinado, introduz um nível de complexidade mais elevado do que o anterior. A situação que se procura representar é a de um aquífero alimentado por uma linha de água e com dois poços, um de cada lado da linha de água, sendo que o da margem esquerda tem uma capacidade de extracção de água com metade da capacidade da direita, sendo esta 2400 m³/dia. Neste problema vamos, portanto, ter caudais impostos concentrados nodais para representar os poços e distribuídos laterais para simular a linha de água. Para além disso, o solo já não é isotrópico, sendo a permeabilidade na direcção x 40 m/dia e metade na y . Note-se também que este exemplo explora o fluxo num plano horizontal e que, portanto as cargas totais são impostas ao longo das fronteiras paralelas à direcção y de valor constante de 200 m. As fronteiras superior e inferior são consideradas impermeáveis.

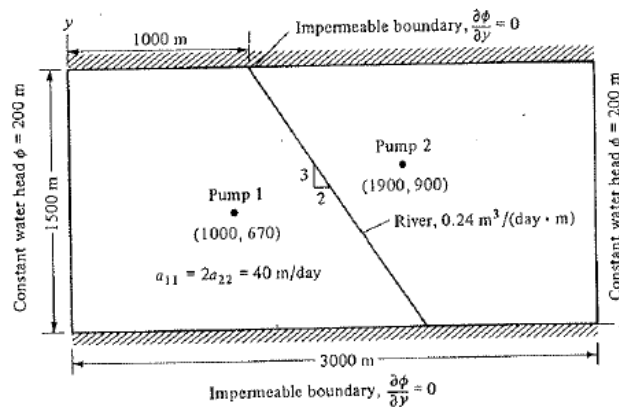


Fig. 4.8 – Problema adaptado de J. N. Reddy

A introdução dos poços levou à necessidade da criação de duas malhas, uma para a função potencial e outra para a função de corrente. A primeira é composta por 1500 elementos de 8 nós, o que perfaz um total de 4671 nós e a segunda mantém o número de elementos, mas o número de nós passa para 4713. Este aumento deve-se ao facto de os poços não poderem estar isolados no interior da malha, o que tornaria o domínio não simplesmente conexo, impossibilitando a obtenção correcta das linhas de corrente. É necessário fazer dois cortes na malha ligando o contorno exterior a cada um dos poços. Este processo, à semelhança do exemplo anterior, é conseguido à custa da duplicação dos nós que se encontram na zona onde se pretende fazer o corte, interrompendo assim a ligação dos elementos. Ao introduzir os cortes na malha passa a ser possível percorrer todo o contorno sem levantar o lápis do papel, o que caracteriza um domínio plano simplesmente conexo. A presença dos poços pode ser vista como associada à introdução no domínio rectangular de dois orifícios de raio infinitesimal. Os caudais nodais concentrados têm como contrapartida as forças pontuais na analogia estrutural

Na figura 4.9 está então representada a malha com indicação dos nós com condições de fronteira impostas, utilizando a mesma simbologia do caso anterior. Assinalaram-se também, a vermelho, a linha de água e os poços. Nas figuras 4.10 e 4.12 encontram-se os resultados da versão de Matlab para as funções potencial e de corrente, respectivamente e nas figuras 4.11 e 4.13 o mesmo, mas na versão Fortran. Na figura 4.14 está desenhada a rede de fluxo que combina as duas soluções. A de potencial da malha sem cortes, que garante que as equipotenciais são fechadas e a de corrente com cortes. A figura 4.15 mostra ainda um pormenor da zona situada entre os dois poços, recorrendo ao desenho dos vectores de velocidade, convenientemente orientados, para compreender a direcção do fluxo e sua repartição espacial.

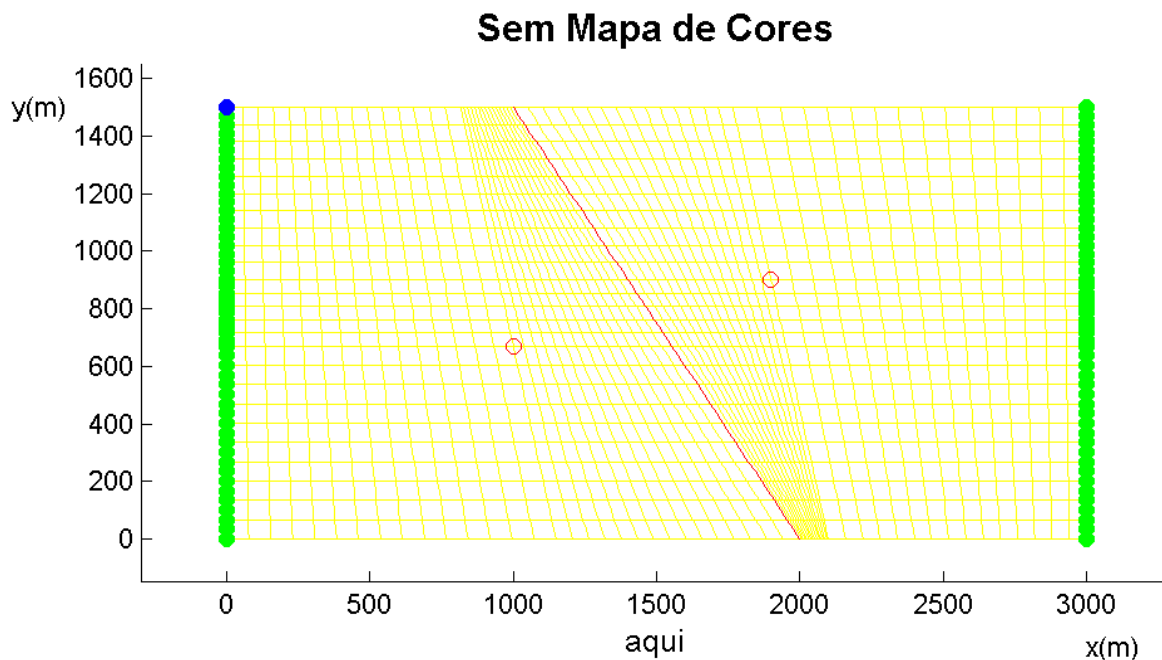


Fig. 4.9 – Aquífero: Esquema da malha adoptada

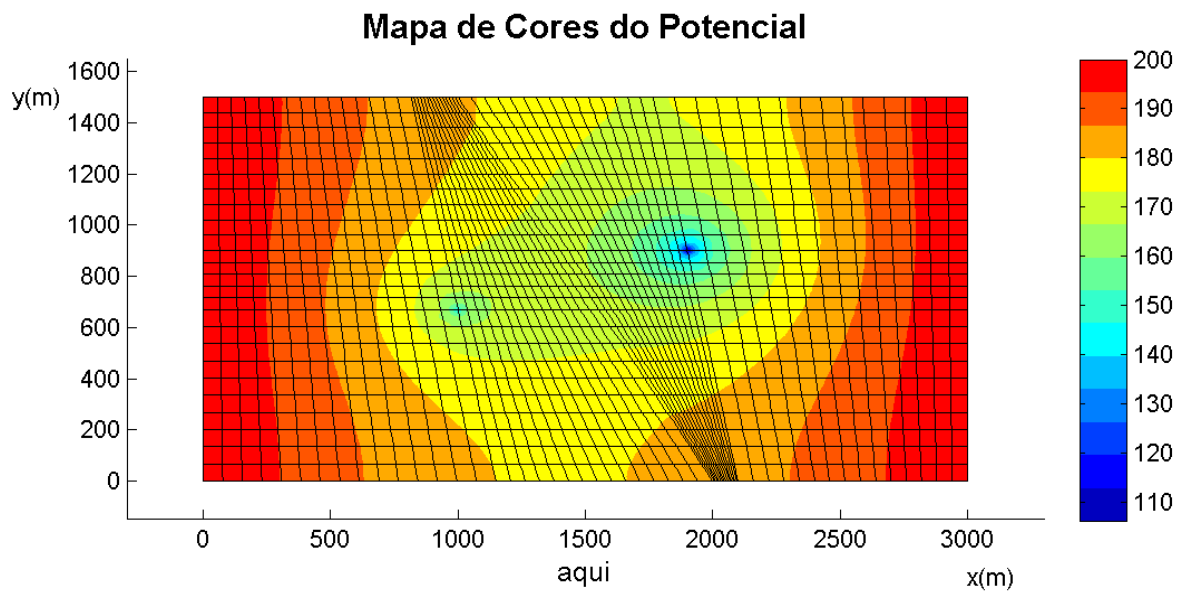


Fig. 4.10 – Aquífero: Coloração da função potencial, versão Matlab

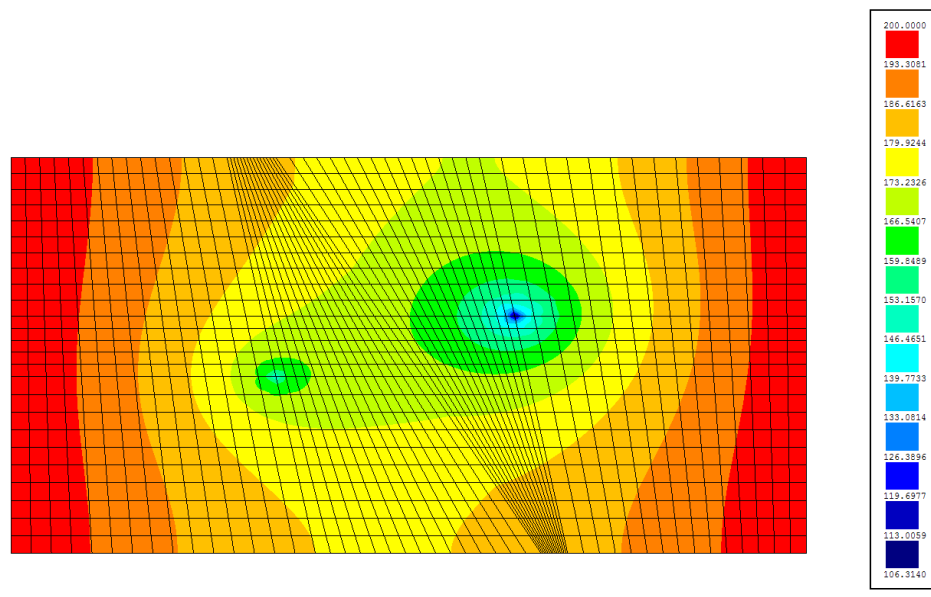


Fig. 4.11 – Aquífero: Coloração da função potencial, versão Fortran

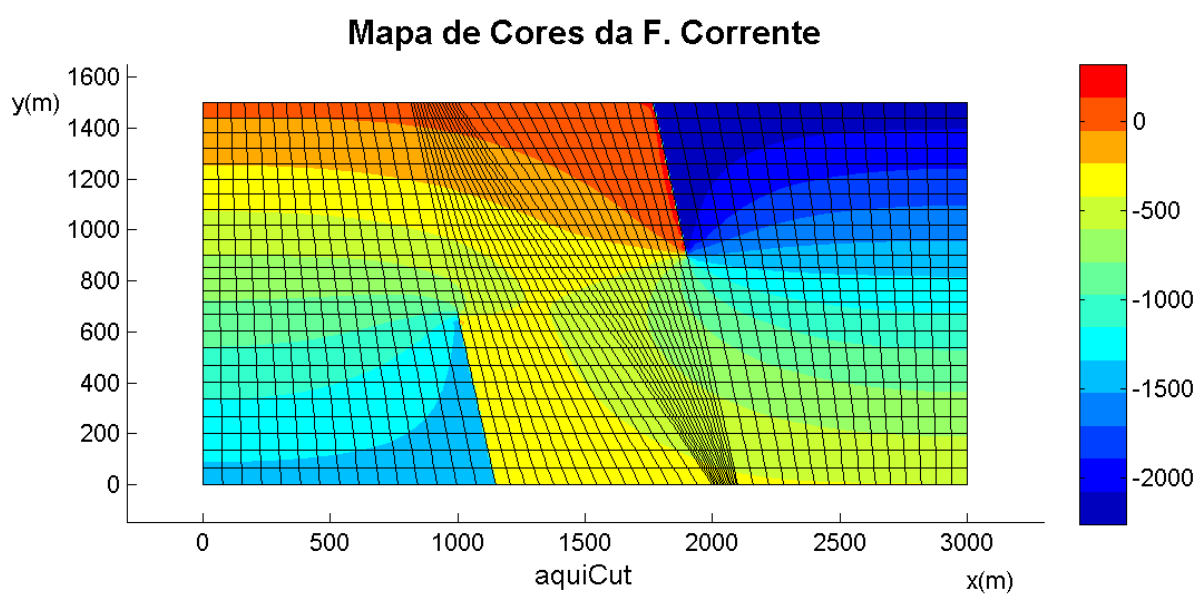


Fig. 4.12 – Aquífero: Coloração da função de corrente, versão Matlab

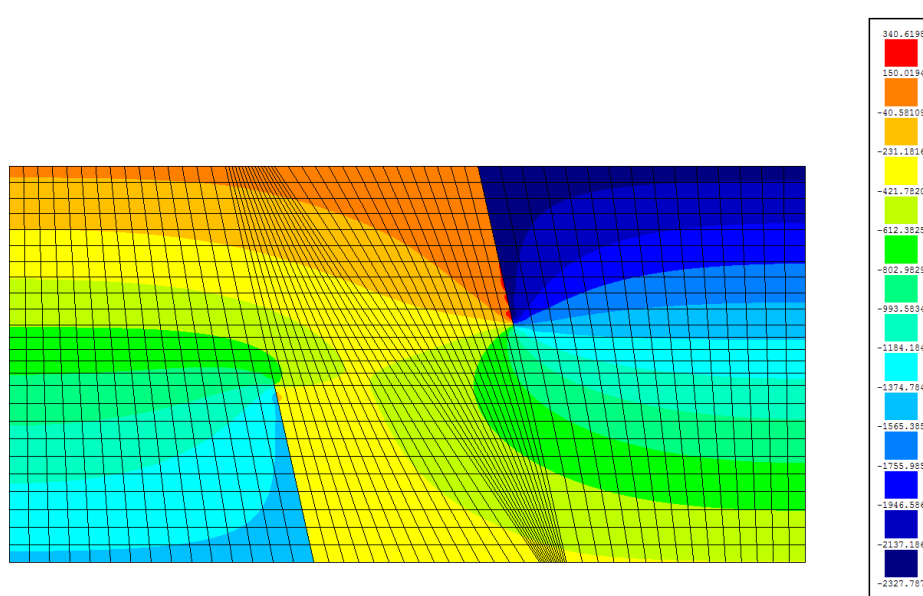


Fig. 4.13 – Aquífero: Coloração da função de corrente, versão Fortran

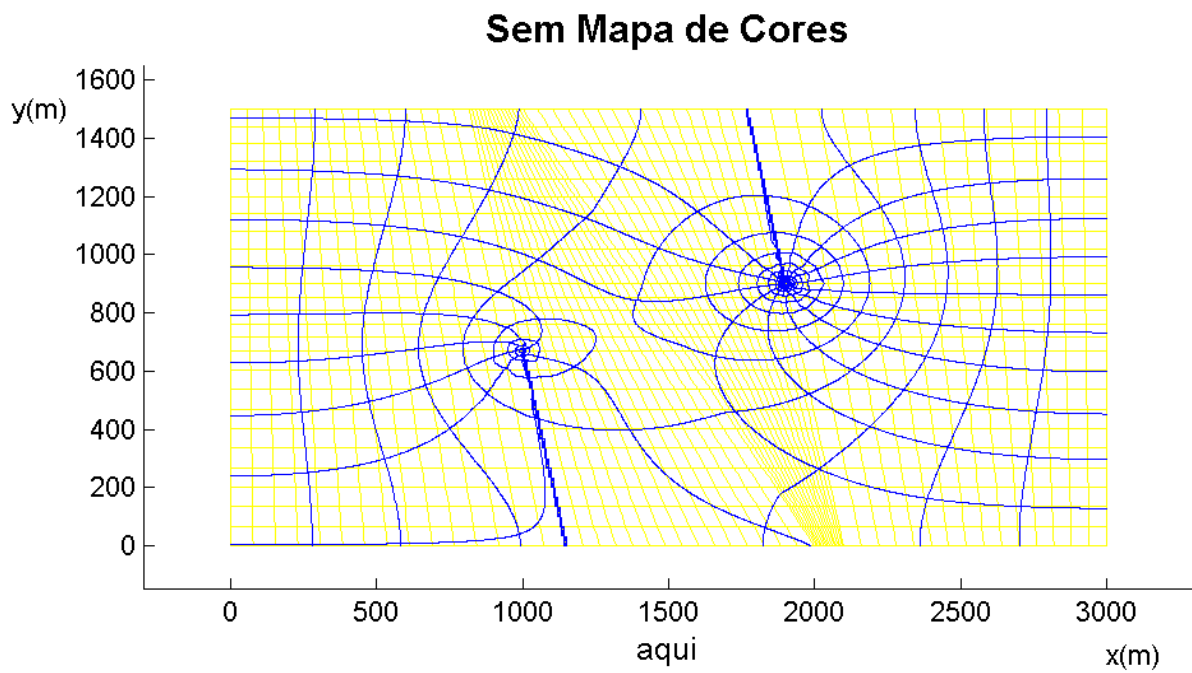


Fig. 4.14 – Aquífero: Rede de fluxo que combina as equipotenciais da primeira malha com as linhas de corrente da segunda

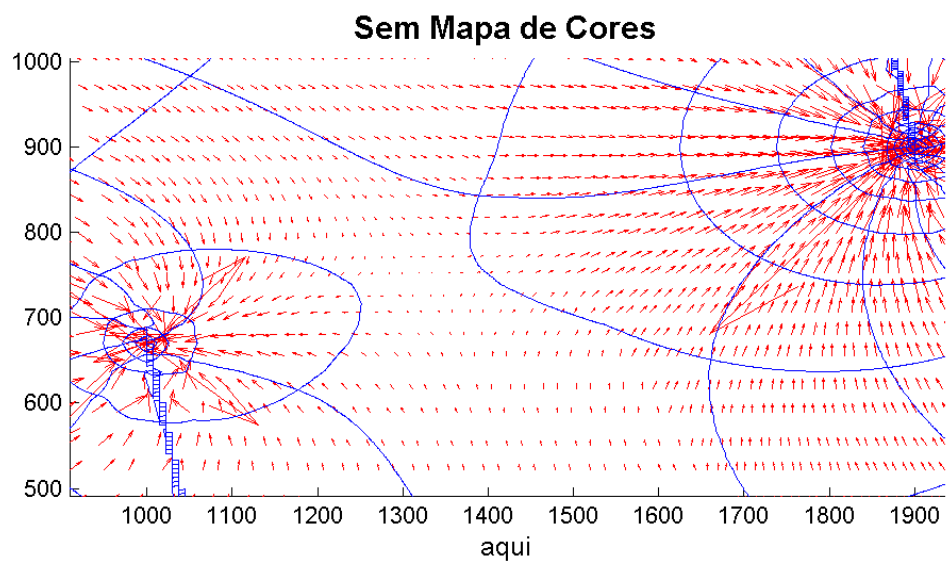


Fig. 4.15 – Aquífero: Pormenor entre os dois poços e representação dos vectores de velocidade

4.4. BARRAGEM

Este é um problema não confinado, na medida em que se desconhece a posição do nível freático à partida e portanto também as condições de fronteira nessa zona. Trata-se de uma barragem de terra com permeabilidade isotrópica de 0,9936 m/dia. A sua geometria é trapezoidal e simétrica e os paramentos têm ambos um declive de 0,5. A base é considerada impermeável e tem 126 m de desenvolvimento e a cabeceira 12 m. A altura da barragem é 28,5 m e a água a montante encontra-se à cota 24 m.

A malha que se usou para representar este problema encontra-se na figura 4.16 e é composta por 609 elementos de 8 nós, o que origina um total de 1928 nós. Devido à dimensão da malha, foi necessário adoptar uma escala reduzida o que dificulta a leitura e impossibilita a representação da numeração dos nós. Tentar-se-ão arranjar formas alternativas de referir os nós em questão.

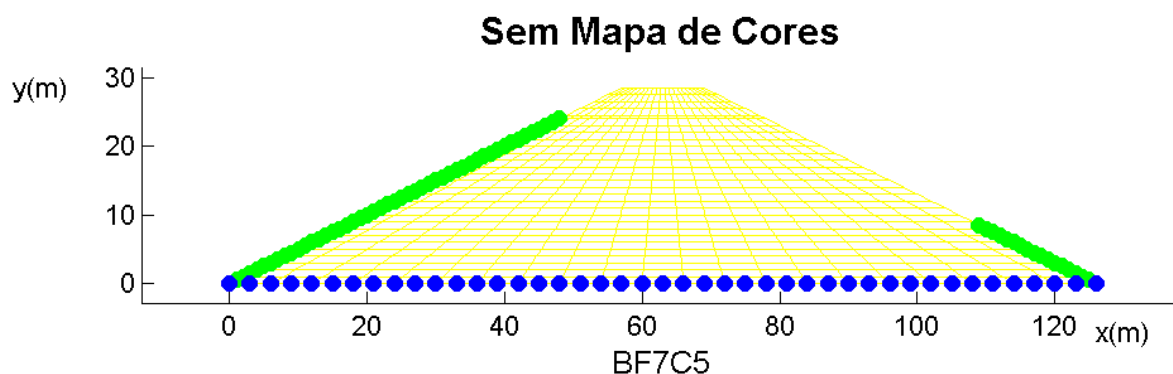


Fig. 4.16 – Barragem: Esquema da malha adoptada

As cargas totais impostas foram de 24 m a montante e a jusante à excepção do nó correspondente ao vértice inferior mais a jusante, onde se impôs uma carga nula. A altura de água a jusante é desconhecida, mas certamente inferior à de montante e portanto definiram-se os pontos assinalados à direita na figura 4.16 com esse valor e indicaram-se como redefiníveis 47 nós que são os que se encontram na fronteira de jusante abaixo da cota 24 m.

A razão adoptada entre a permeabilidade final e inicial foi de 10^{-7} e a tolerância de convergência 10^{-3} . Neste problema não há caudais impostos.

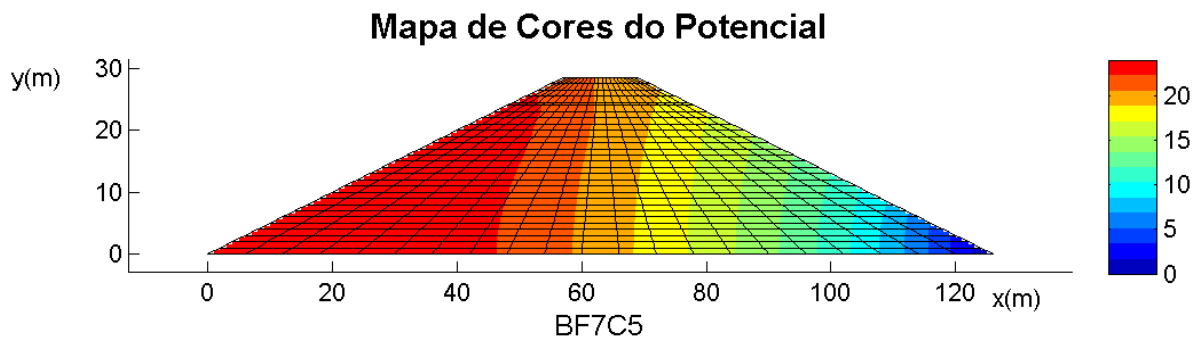


Fig. 4.17 – Barragem: Coloração da função potencial, versão Matlab

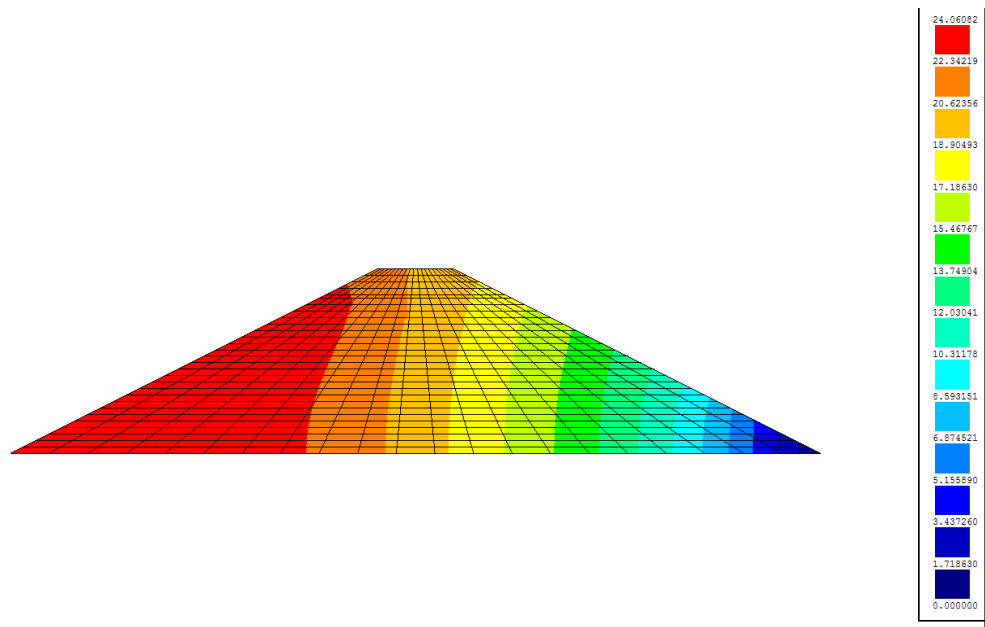


Fig. 4.17 – Barragem: Coloração da função potencial, versão Fortran

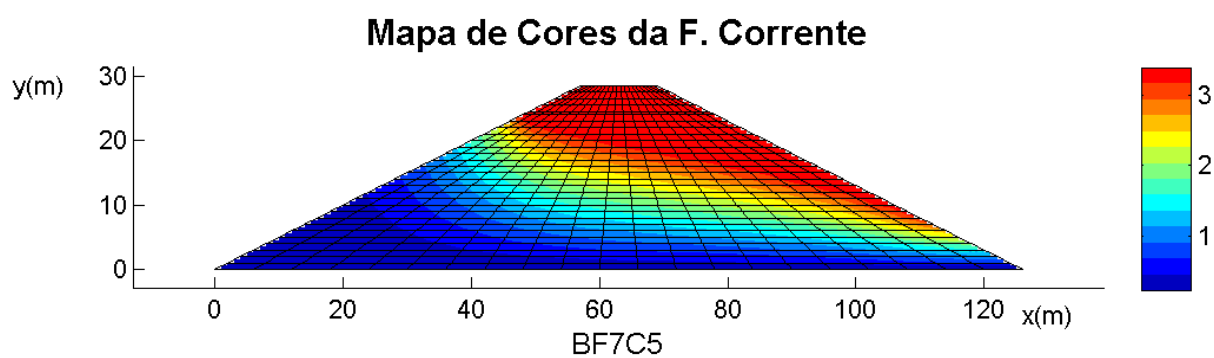


Fig. 4.18 – Barragem: Coloração da função de corrente, versão Matlab

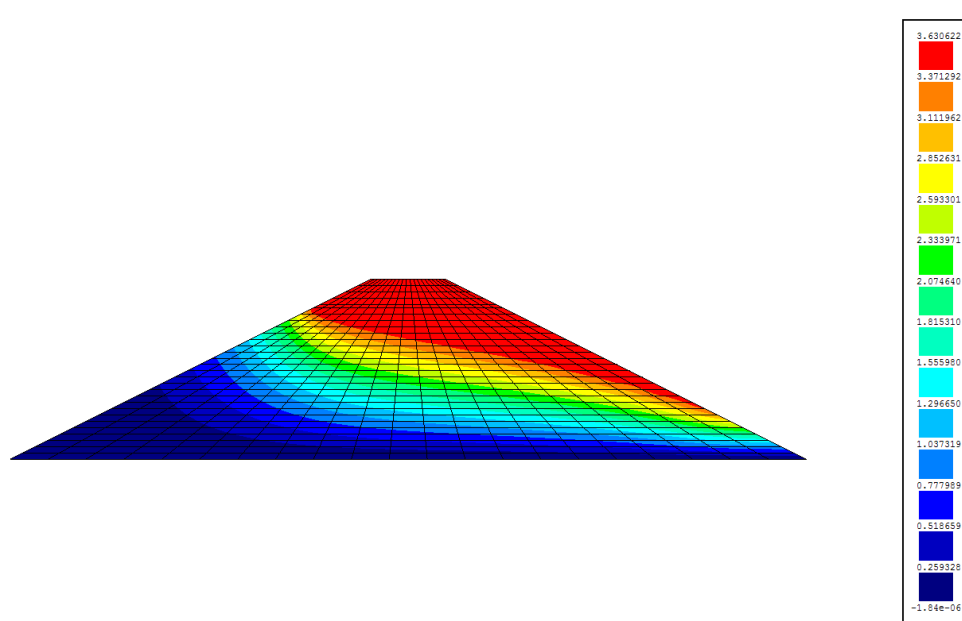


Fig. 4.18 – Barragem: Coloração da função de corrente, versão Fortran

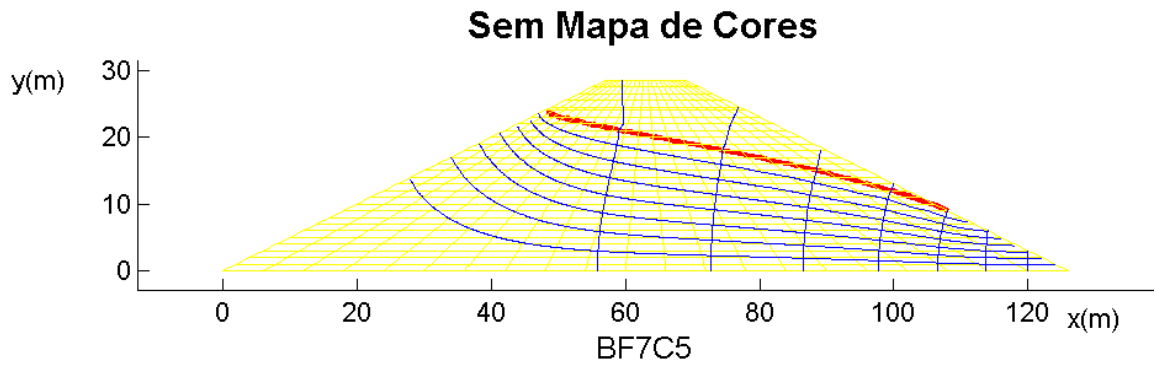


Fig. 4.19 – Barragem: Rede de fluxo a azul e nível freático a vermelho

4.5. BARRAGEM COM TAPETE DRENANTE

Apresenta-se agora uma outra barragem de terra, mas esta com a introdução de um tapete drenante. As condições geométricas são as apresentadas na figura 4.20, nomeadamente uma cabeceira de 8 m, o talude de montante com uma inclinação de 1 para 3 e o de jusante de 1 para 2,5, o que significa que a base tem 96 m de desenvolvimento. A permeabilidade mantém-se isotrópica agora de valor 10^{-5} m/s. Continua a não haver caudais impostos e as condições de fronteira a respeitar passam pela imposição de uma carga total de 14 m no paramento de montante e carga nula nos pontos que correspondem ao tapete. Também aqui se considerou que a fronteira inferior era impermeável.

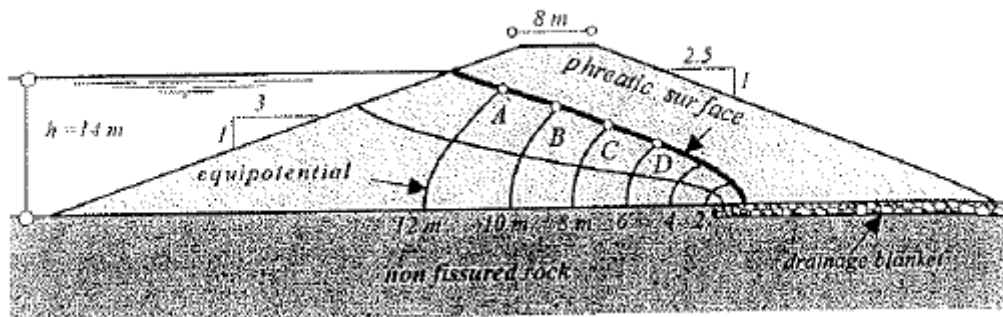


Fig. 4.20 – Barragem com tapete drenante: Condições geométricas do problema

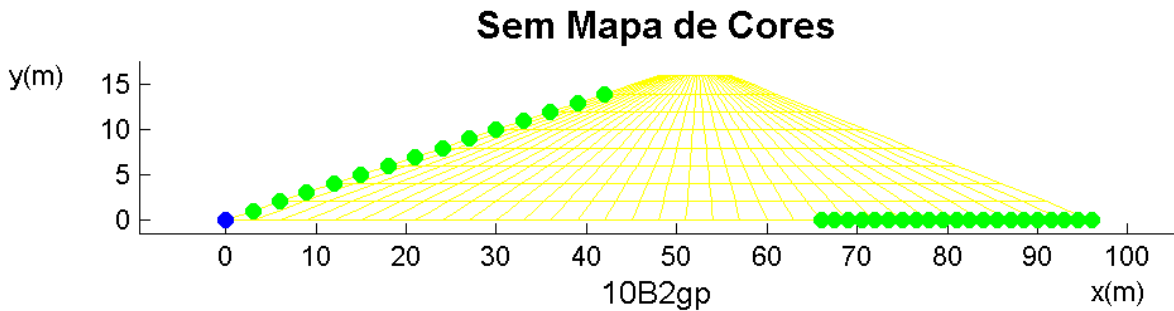


Fig. 4.21 – Barragem com tapete drenante: Condições fronteira

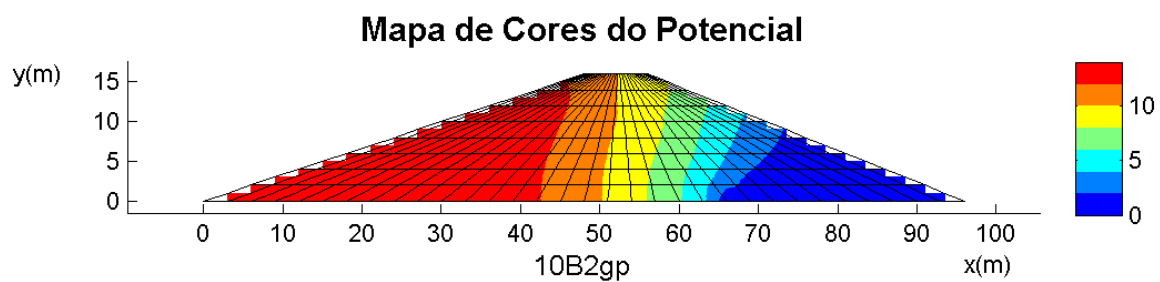


Fig. 4.22 – Barragem com tapete drenante: Coloração da função potencial, versão Matlab

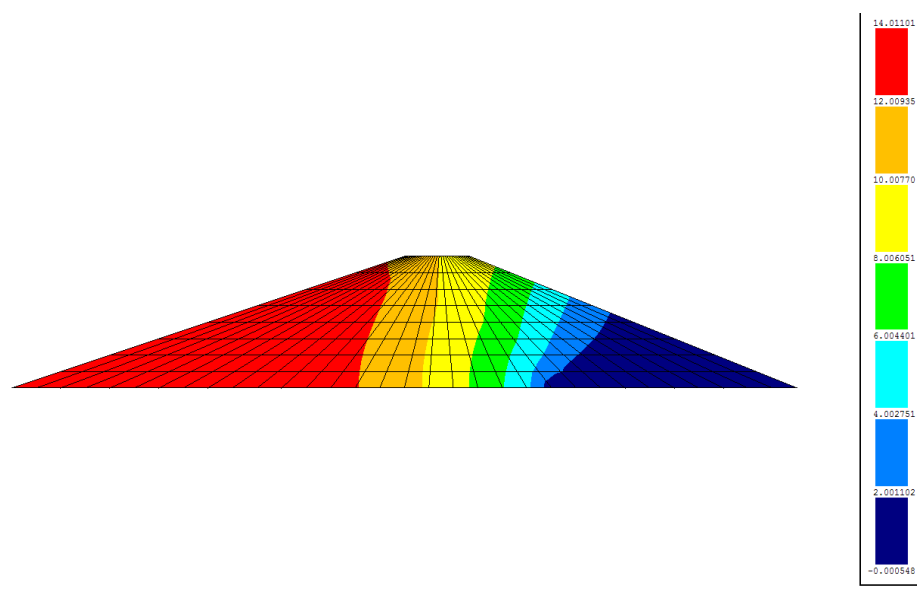


Fig. 4.23 – Barragem com tapete drenante: Coloração da função potencial, versão Fortran

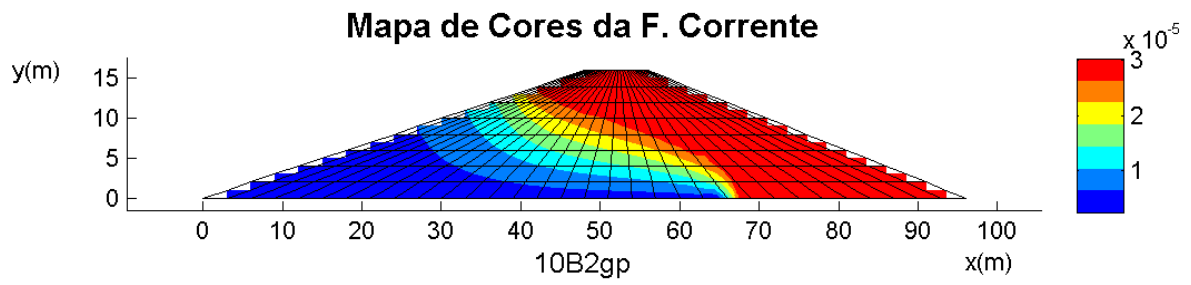


Fig. 4.24 – Barragem com tapete drenante: Coloração da função corrente, versão Matlab

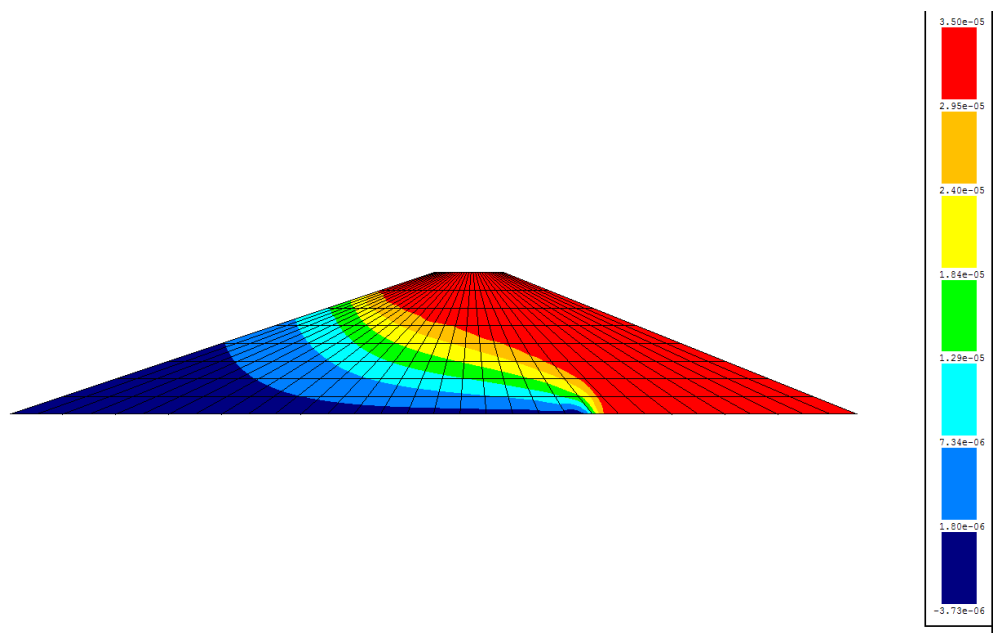


Fig. 4.25 – Barragem com tapete drenante: Coloração da função corrente, versão Fortran

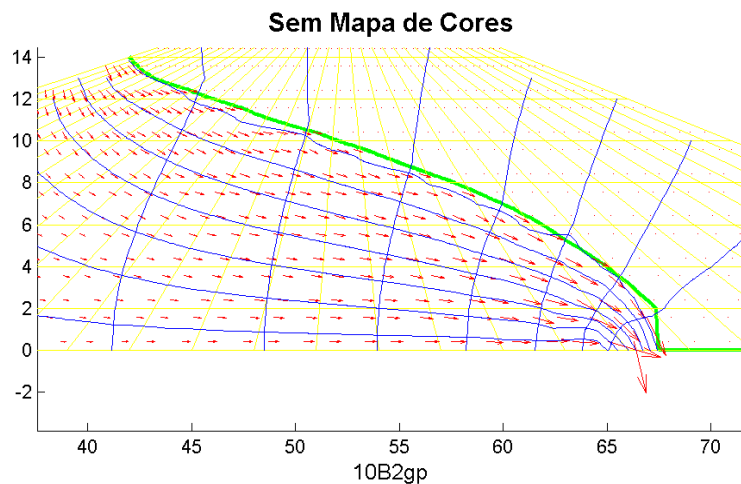


Fig. 4.26 – Barragem com tapete drenante: Rede de fluxo, representação vectorial da velocidade e nível freático a verde

4.6. Poço

Por último, apresenta-se um problema axissimétrico não confinado. Trata-se de um poço com um metro de diâmetro e uma extracção constante de caudal. O valor do caudal é desconhecido à partida, mas sabe-se que provoca um abaixamento do nível freático dos 10 m para os 8,5 m, num percurso de 20 m. A permeabilidade adoptada para o problema foi de 10^{-4} m/s. Como se antecipou na introdução este problema tem uma expressão analítica que permite determinar o caudal, que é a que se encontra em (4.1). A figura 4.27 serve de suporte às variáveis da expressão. Ao resolver este problema o somatório das reacções positivas foi 2,3537 l/s que é um valor bastante próximo do analítico, confirmando assim por uma outra via a validade do programa desenvolvido.

$$Q = \frac{k \times \pi \times (h_2^2 - h_1^2)}{\ln\left(\frac{r_2}{r_1}\right)} = \frac{0,0001 \times \pi \times (10^2 - 8,5^2)}{\ln\left(\frac{20,5}{0,5}\right)} = 0,002348 m^3 / s = 2,2348 l / s \quad (4.1)$$

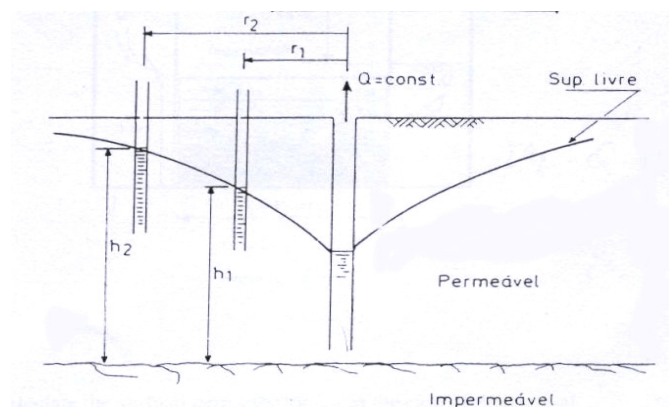


Fig. 4.27 – Poço: Descrição gráfica do problema

CONSIDERAÇÕES FINAIS

Este trabalho, consequência da sua natureza, implicou o esclarecimento dos conceitos que envolvem a percolação. À medida que se iam aprofundando as teorias subjacente ao movimento da água em meios porosos foi ficando claro que esse movimento é extremamente complexo e impossível de descrever com rigor.

Contudo, foram apresentados modelos simplificados capazes de traduzir em termos médios o comportamento do fluido. Percebeu-se que velocidade do modelo não é a velocidade a que as partículas se deslocam e que as linhas de corrente não correspondem ao caminho descrito pelas mesmas. Ainda assim, é com os resultados do modelo que os engenheiros trabalham e garantem a segurança. É importante recordar que a Engenharia não é uma ciência exacta e que não se consome com o rigor dos resultados obtidos, mas antes com a sua grandeza e interpretação.

Recorde-se que os resultados são função da permeabilidade, sendo este o parâmetro físico com maior gama de variação. Para além disso, a geotecnia em particular é uma área da Engenharia Civil que se caracteriza por um material de propriedades desconhecidas à partida, ao contrário das estruturas de aço, ou mesmo betão, cuja produção industrial lhes confere propriedades sobre as quais há um grande controle. A avaliação das propriedades do solo é com frequência deficiente e introduz à partida erros de uma magnitude que dispensam preciosismos na modelação. Dito isto, refere-se que o presente trabalho pode ser completado com elementos infinitos para representar meios semi-indefinidos e elementos especiais capazes de tratar as singularidades com melhores resultados sem necessidade de recorrer ao refinamento da malha.

Os desenvolvimentos futuros porém não se ficam por aqui. Este trabalho abordou apenas problemas em regime permanente, mas pode ser alargado para regimes variáveis, que incorporam a variável tempo, como é o caso do esvaziamento de uma barragem, ou mesmo ir além da percolação e incorporar métodos de verificação de taludes, uma vez que os dois estão normalmente associados. A importância de conhecer a posição do fluido no interior de uma barragem de terra advém da possível situação de instabilidade provocada pela diferença de velocidades de escoamento da água a montante da barragem e no seu interior, que concilia um solo com propriedades reduzidas pela presença da água sem o peso da massa de água a montante para equilibrar as forças instabilizadoras.

Uma outra hipótese seria a ampliação para a terceira dimensão, uma vez que, apesar de se ter defendido que a grande maioria dos problemas se repetem numa das direcções, o que permite que sejam reduzidos a um plano, outras haverá em que assim não acontece.

BIBLIOGRAFIA

- [1] Matos Fernandes, M. *Mecânica dos Solos – Conceitos e Princípios Fundamentais, Volume I*. FEUP Edições, Porto, 2006.
- [2] Marado, J. M. P. *Análise de Problemas de Percolação pelo Método dos Elementos Finitos*. Dissertação de Mestrado, FEUP, 1993.
- [3] Fortes, R. N. *Redes de Fluxo em Regime Variável – Implementação pelo Método dos Elementos Finitos*. Dissertação de Mestrado, Universidade Nova, Lisboa, 1988.
- [4] Novais Barbosa, J. A. *Mecânica dos Fluidos e Hidráulica Geral*, Porto Editora, 1985.
- [5] Zienkiewicz, O. C. e Morgan, K. *Finite Elements and Approximation*, John Wiley and Sons, New York, 1983.
- [6] Martins, F. F. *Aplicações de Elementos Finitos e Infinitos na Análise de Problemas de Carácter Geotécnico*. Dissertação de Mestrado, FEUP, 1988.
- [7] Aalto, J. *Finite Element Seepage Flow Nets*, International Journal for Numerical and Analytical Methods in Geomechanics, Vol. 8, 1984, 297-303, John Wiley & Sons.
- [8] Lambe, T. W., Whitman, R. V., Poulos, H. G. Two-Dimensional Fluid Flow. In *Soil mechanics, SI version*, 269, John Wiley and Sons, New York, 1979.
- [9] Reddy, J. N. *An Introduction to the Finite Element Method*, McGraw-Hill, 1984.

A1

PRÉ PROCESSAMENTO

```

%***  alternar.m  *****
%
%***  torna visível o outro menu
%
%****  2010.06.30 * LR ***** quasar_m ****

function k=alternar(cf,ci,k,h)

if k==0
    set(cf,'visible','off')
    set(ci,'visible','on')
    set(h,'string','Condições Fronteira')
    k=1;
else
    set(cf,'visible','on')
    set(ci,'visible','off')
    set(h,'string','Caudais Impostos')
    k=0;
end

```

```

%***  atribuir.m  *****
%
%***  atribui as respectivas propriedades
%
%****  2010.06.30 * LR ***** quasar_m ****

function NQEDG=atribuir(h,data)

%abertura de canal de escrita do caudal distribuido
distribuido=fopen('IQEDG','at+');

%identificar os objectos seleccionados
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
if isempty(hKids_eixos)
    aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
        'string','Aviso: Não há objectos desenhados','Units',...
        'normalized','position',[0 0 .75 .05],...
        'fontsize',10);
    pause(2)
    delete(aviso)
    return
end
todos=findobj(hKids_eixos,'color',[0 1 0]);
linhas=findobj(todos,'userdata',1);
marcadores=findobj(todos,'userdata',2);
n=size(linhas,1);
m=size(marcadores,1);
if h==1
    if n==0
        aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
            'string','Aviso: Não há elementos seleccionados','Units',...
            'normalized','position',[0 0 .75 .05],...
            'fontsize',10);
        pause(2)
    end
end

```

```

        delete(aviso)
        return
    end
    for i=1:n
        setappdata(linhas(i),'material',data)
    end
else
    if m==0 && h~=7
        aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
            'string','Aviso: Não há nós seleccionados','Units',...
            'normalized','position',[0 0 .75 .05],...
            'fontsize',10);
        pause(2)
        delete(aviso)
        return
    end
    if h==2
        for i=1:m
            setappdata(marcadores(i),'potencial',str2num(data))
        end
    elseif h==3
        for i=1:m
            setappdata(marcadores(i),'corrente',str2num(data))
        end
    elseif h==4
        for i=1:m
            setappdata(marcadores(i),'concentrado',str2num(data))
        end
    elseif h==5
        if m==0
            aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor',...
                'w','string','Aviso: Não há nós seleccionados',...
                'Units','normalized','position',[0 0 .75 .05],...
                'fontsize',10);
            pause(2)
            delete(aviso)
            return
        elseif m>1
            aviso=uicontrol(gmalha_fig,'Style','text',...
                'BackgroundColor','w',...
                'string',...
                'Aviso: Há mais do que um nós seleccionado!',...
                'Units','normalized','position',[0 0 .75 .05],...
                'fontsize',10);
            pause(2)
            delete(aviso)
            return
        end
        fprintf(distribuido,'% .14f %g ',marcadores,str2num(data));
    elseif h==6
        for i=1:m
            setappdata(marcadores(i),'redefiniveis','s')
        end
    elseif h==7
        if n==0
            aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor',...
                'w','string','Aviso: Não há elementos seleccionados',...
                'Units','normalized','position',[0 0 .75 .05],...
                'fontsize',10);

```

```

        pause(2)
        delete(aviso)
        return
    elseif n>1
        aviso=uicontrol(gmalha_fig,'Style','text',...
            'BackgroundColor','w',...
            'string',...
            'Aviso: Há mais do que um elemento seleccionado!',...
            'Units','normalized','position',[0 0 .75 .05],...
            'fontsize',10);
        pause(2)
        delete(aviso)
        return
    end
    NQEDG=data;
    if NQEDG == 1
        fprintf(distribuido,'% .14f ',linhas);
    else
        fprintf(distribuido,'\n%.14f ',linhas);
    end
    NQEDG=NQEDG+1;
    h_push=findobj(hKids_fig,'flat','tag','ci(12)');
    set(h_push,'string',['lado ',num2str(NQEDG)])
end
end
aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
    'string','Aviso: Propriedade atribuída!','Units',...
    'normalized','position',[0 0 .75 .05],...
    'fontsize',10);
pause(2)
delete(aviso)

```

```

%*** closemyfig.m *****
%
%*** abre uma caixa de diálogo a confirmar o encerramento da janela *
%
%**** 2010.06.30 * LR ***** quasar_m ****

selection = questdlg(['Quer realmente fechar a ',...
'figura: ',JNAME,'?'],...
'Fechar a representação gráfica',...
'Sim','Não','Não');
switch selection,
case 'Sim',
delete(gcf)
case 'Não'
return
end

```

```

%*** conf.m *****
%
%*** atribui o tipo de confinamento *
%
%**** 2010.06.30 * LR ***** quasar_m ****

function KONFN=conf(h_radio,button)

```

```

% garante que quando se carrega num botão este permanece activo
% e os restantes são desactivados
num_buttons=length(h_radio);;
if get(h_radio(button),'value') == 1
    set(h_radio([1:(button-1),(button+1):num_buttons]),'value',0)
else
    set(h_radio(button),'value',1)
end

if button == 1
    KONFN=1;
elseif button ==2
    KONFN=0;
end

```

```

%***  desenhar.m  ****
%
%***  une os pontos seleccionados  *
%
%****  2010.06.30 * LR **** quasar_m ****

function desenhar

gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
[m,n]=size(hKids_eixos);
if noeixo(gmalha_eixos),
    user_data=get(gmalha_fig,'UserData');
    x=get(user_data(1),'String');x=str2num(x);
    y=get(user_data(2),'String');y=str2num(y);
    xydata=get(gmalha_eixos,'userdata');
    if isempty(xydata)
        xydata=[x,y];
        h_linhas_temp=hgroup('parent',gmalha_eixos,'tag','linhas_temp');
        linha=plot(gmalha_eixos,x,y,'ob');
        set(linha,'parent',h_linhas_temp)
        hgroup('parent',gmalha_eixos,'tag','marcadores_temp')
        set(gmalha_eixos,'userdata',xydata)
        return
    end
    xdata=xydata(:,1);
    ydata=xydata(:,2);
    xdata=[xdata;x];
    ydata=[ydata;y];
    h_linhas_temp=findobj(hKids_eixos,'flat','tag','linhas_temp');
    h_marcadores_temp=findobj(hKids_eixos,'flat','tag','marcadores_temp');
    linha=plot(gmalha_eixos,xdata,ydata,'-ob');
    set(linha,'parent',h_linhas_temp)
    marcador=plot(gmalha_eixos,x,y,'ob','userdata',2);
    set(marcador,'parent',h_marcadores_temp)
    kids=get(h_linhas_temp,'children');
    delete(kids(2))
    if [x,y]==xydata(1,:)
        sombra=fill(xdata,ydata,'b','FaceAlpha',0.5);
    end
end

```

```

        pause(1)
        delete(sombra)
        set(linha,'userdata',1)
        set(gmalha_eixos,'userdata',[])
        set(h_linhas_temp,'tag','')
        set(h_marcadores_temp,'tag','')
        set(h_marcadores_temp,'parent',h_linhas_temp)
        return
    end
    set(gmalha_eixos,'userdata',[xdata,ydata])
end

```

```

%***  editavel.m  *****
%
%***  lê e associa a uma variável o valor introduzido numa janela      *
%
%****  2010.06.30 * LR ***** quasar_m ****

function x=editavel(handle)

string=get(handle,'string');
x=str2num(string);

```

```

%***  eixoslim.m  *****
%
%***  abertura da janela onde se dimensionam os eixos da pag. principal  *
%
%****  2010.06.30 * LR ***** quasar_m ****

eixos_inp=figure;
titulo_eixos_inp='Dimensionamento da página';
set(eixos_inp,'name',titulo_eixos_inp,'numbertitle','off',...
'menubar','none','units','normalized',...
'position',[0.3 0.3 .4 .4],'tag','eixos_inp')

% valores iniciais
xmin=0;xmax=10;ymin=0;ymax=10;grelha=1;

% criação dos campos de entrada de dados
uicontrol(eixos_inp,'Style','frame','Units','normalized',...
'Position',[0,0,1,1]);
uicontrol(eixos_inp,'style','text','units','normalized',...
'position',[0.01,.88,.98,.07],'string','Dimensionamento da página',...
'FontSize',12,'FontWeight','bold')

% xmim
uicontrol(eixos_inp,'style','text','units','normalized',...
'position',[.19,.765,.12,.06],'string','Xmin')
xminimo=uicontrol(eixos_inp,'style','edit','units','normalized',...
'position',[.315,.765,.12,.08],'BackgroundColor','w',...
'callback','xmin=editavel(xminimo);','string','0');

% xmax
uicontrol(eixos_inp,'style','text','units','normalized',...
'position',[.565,.765,.12,.06],'string','Xmax')
xmaximo=uicontrol(eixos_inp,'style','edit','units','normalized',...

```

```

        'position',[.69,.765,.12,.08],'BackgroundColor','w',...
        'callback','xmax=editavel(xmaximo);','string','10');
% ymim
uicontrol(eixos_inp,'style','text','units','normalized',...
    'position',[.19,.58,.12,.06],'string','Ymin')
yminimo=uicontrol(eixos_inp,'style','edit','units','normalized',...
    'position',[.315,.58,.12,.08],'BackgroundColor','w',...
    'callback','ymin=editavel(yminimo);','string','0');
% ymax
uicontrol(eixos_inp,'style','text','units','normalized',...
    'position',[.565,.58,.12,.06],'string','Ymax')
ymaximo=uicontrol(eixos_inp,'style','edit','units','normalized',...
    'position',[.69,.58,.12,.08],'BackgroundColor','w',...
    'callback','ymax=editavel(ymaximo);','string','10');
% escala
check_box_escala=uicontrol(eixos_inp,'style','checkbox','units',...
    'normalized','position',[0.3,.45,.4,.07],...
    'string','mesma escala nos dois eixos',...
    'callback','on_off(check_box_escala,'escala'),'value',1);
% grelha
uicontrol(eixos_inp,'style','text','units','normalized',...
    'position',[.3,.28,.2,.1],'string',{'espaçamento';'da grelha'})
h_grelha=uicontrol(eixos_inp,'style','edit','units','normalized',...
    'position',[.5,.28,.2,.1],'BackgroundColor','w',...
    'callback','grelha=editavel(h_grelha);','string','1');
%Definição do problema
uicontrol(eixos_inp,'units','normalized',...
    'position',[.1,.1,.3,.1],'string','Definição do Problema',...
    'callback','[NTYPE,NGAUS,KONFN]=problema;')
% ok
ok_eixoslim=uicontrol(eixos_inp,'string','ok','units',...
    'normalized','position',[.75,.06,.2,.1],...
    'callback','status_escala=ok(check_box_escala);');
set(ok_eixoslim,'enable','off')

```

```

%***  fpush.m  ****
%
%***  copiar e eleminar
%
%****  2010.06.30 * LR **** quasar_m ****

function fpush(button)

gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
if isempty(hKids_eixos)
    aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
        'string','Aviso: Não há objectos desenhados','Units',...
        'normalized','position',[0 0 .75 .05],...
        'fontsize',10);
    pause(2)
    delete(aviso)
    return
end
todos=findobj(hKids_eixos,'color',[0 1 0]);

```

```

linhas=findobj(todos,'userdata',1);
[n,m]=size(linhas);
if n==0
    antigo=[];
end
for i=1:n
    antigo(i)=get(linhas(i),'parent');
end
if isempty(antigo) && button ~= 3
    aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
        'string','Aviso: Não há objectos seleccionados','Units',...
        'normalized','position',[0 0 .75 .05],...
        'fontsize',10);
    pause(2)
    delete(aviso)
    return
end
if button == 1
    novo=copyobj(antigo,gmalha_eixos);
    for i=1:n
        set(findobj(antigo(i),'type','line'),'color',[0 0 1])
        set(findobj(novo(i),'type','line'),'color',[0 1 0])
    end
    aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
        'string','Aviso: Objecto copiado!','Units',...
        'normalized','position',[0 0 .75 .05],...
        'fontsize',10);
    pause(2)
    delete(aviso)
elseif button == 2
    delete(antigo)
elseif button == 3
    xydata=get(gmalha_eixos,'userdata');
    if isempty(xydata)
        return
    end
    xdata=xydata(:,1);
    ydata=xydata(:,2);
    xdata=xdata(1:end-1);
    ydata=ydata(1:end-1);
    h_linhas_temp=findobj(hKids_eixos,'flat','tag','linhas_temp');
    h_marcadores_temp=findobj(hKids_eixos,'flat','tag','marcadores_temp');
    linha=plot(gmalha_eixos,xdata,ydata,'-ob');
    set(linha,'parent',h_linhas_temp)
    kids=get(h_linhas_temp,'children');
    if size(kids,1)==1
        delete(kids)
        set(gmalha_eixos,'userdata',[])
        set(h_linhas_temp,'tag','')
        set(h_marcadores_temp,'tag','')
    else
        delete(kids(2))
        kids_marc=get(h_marcadores_temp,'children');
        delete(kids_marc(1))
    end
    set(gmalha_eixos,'userdata',[xdata,ydata])
end
end

```

```

%***  ftoggle.m  ****
%
%***  desenhar, seleccionar e mover
%
%****  2010.06.30 * LR **** quasar_m ****

function ftoggle(h_toggle,button)

% garante que quando se carrega num botão este permanece activo
% e os restantes são desactivados
num_buttons=length(h_toggle);
if get(h_toggle(button),'value') == 1
    set(h_toggle([1:(button-1),(button+1):num_buttons]),'value',0)
else
    set(h_toggle(button),'value',1)
end
% especifica as acções de cada botão
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
h_linhas=findobj(hKids_eixos,'type','line');
undo_push=findobj(hKids_fig,'flat','tag','undo');
%procura rectangulo
rect=findobj(hKids_eixos,'flat','tag','rectan');
if ~isempty(rect)
    delete(rect)
end
if button == 1
    set(h_linhas,'ButtonDownFcn','')
    set(gmalha_fig,'WindowButtonDownFcn','desenhar')
    set(undo_push,'enable','on')
elseif button == 2
    set(gmalha_fig,'WindowButtonDownFcn','janela')
    set(h_linhas,'ButtonDownFcn','select')
    set(undo_push,'enable','off')
elseif button == 3
    set(h_linhas,'ButtonDownFcn','')
    set(gmalha_fig,'WindowButtonDownFcn','mover')
    set(undo_push,'enable','off')
end

```

```

%***  gauss.m  ****
%
%***  atribui o número de pontos de Gauss
%
%****  2010.06.30 * LR **** quasar_m ****

function NGAUS=gauss(h_radio,button)

% garante que quando se carrega num botão este permanece activo
% e os restantes são desactivados
num_buttons=length(h_radio);
if get(h_radio(button),'value') == 1
    set(h_radio([1:(button-1),(button+1):num_buttons]),'value',0)
else
    set(h_radio(button),'value',1)
end

```

```

end

if button == 1
    NGAUS=2;
elseif button ==2
    NGAUS=3;
end

```

```

%***  gerar_ficheiro.m  ****
%
%***  cria um ficheiro de texto com os dados
%
%****  2010.06.30 * LR ****      quasar_m ****

%cria um ficheiro de texto com os dados
function gerar_ficheiro(NMATS,NTYPE,NGAUS,KONFN,ci,NQEDG)

%abertura do canal de escrita
[nome_ficheiro,pathname] = uiputfile('*.DAT','Nome do ficheiro');
dados=fopen(nome_ficheiro,'wt');
%3 linhas do titulo
fprintf(dados,'\n\n\n');

%PARAMETROS DE CONTROLE

%procurar elementos e nos
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
elementos=findobj(hKids_eixos,'userdata',1);
nos=findobj(hKids_eixos,'userdata',2);
% nos_aux=findobj(hKids_eixos,'userdata',2);
n=size(elementos,1);
m=size(nos,1);
% m=size(nos_aux,1);
% for i=1:m
%     nos(i)=nos_aux(m+1-i);
% end

%armazenar coordenadas de cada elemento
xi=get(elementos,'xdata');
yi=get(elementos,'ydata');
if n>1
    xi=cell2mat(xi);
    yi=cell2mat(yi);
end
%procurar elementos sobrepostos e apaga-los
for i=1:n-1
    for j=i+1:n
        if all(xi(i,:)==xi(j,:))
            if all(yi(i,:)==yi(j,:))
                delete(elementos(j))
            end
        end
    end
end
end

```

```

end
%nova lista sem elementos repetidos
elementos=findobj(hKids_eixos,'userdata',1);
NELEM=size(elementos,1);
fprintf(dados,'%d ',NELEM);

%n° de nos da malha
%armazenar coordenadas de cada no
xni=get(nos,'xdata');
yni=get(nos,'ydata');
if m>1
    xni=cell2mat(xni);
    yni=cell2mat(yni);
end
%procurar nos sobrepostos para que nao sejam contados duas vezes
k=0;
for i=m:-1:2
    for j=i-1:-1:1
        if xni(i)==xni(j)
            if yni(i)==yni(j)
                k=k+1;
                setappdata(nos(j),'primario',nos(i))
            end
        end
    end
end
%numeraçao dos nos
NPOIN=0;
for i=m:-1:1
    prim=getappdata(nos(i),'primario');
    if isempty(prim)
        NPOIN=NPOIN+1;
        setappdata(nos(i),'num',NPOIN)
    else
        setappdata(nos(i),'num',getappdata(prim,'num'))
    end
end
fprintf(dados,'%d ',NPOIN);
fprintf(dados,'%d ',NMATS);
%no de nos com carga total imposta
NPFIX=0;
for i=1:m
    if ~isempty(getappdata(nos(i),'potencial'))
        NPFIX=NPFIX+1;
    end
end
fprintf(dados,'%d ',NPFIX);

%n° max de nos por elemento na malha
%preenchimento do vector LNODS
NNODM=4;
LNODS=zeros(9,NELEM);
for J=1:NELEM
    MATNO(J)=getappdata(elementos(J),'material');
    pai=get(elementos(J),'parent');
    filhos=get(pai,'children');
    h_marcador=findobj(filhos,'flat','type','hggroup');
    nos_elemento_aux=get(h_marcador,'children');

```

```

        NNODE(J)=size(nos_elemento_aux,1);
        for i=1:NNODE(J)
            nos_elemento(i)=nos_elemento_aux(NNODE(J)+1-i);
        end
        if NNODE(J)>NNODM
            NNODM=NNODE(J);
        end
        for I=1:NNODE(J)
            LNODS(I,J)=getappdata(nos_elemento(I),'num');
        end
    end
end
fprintf(dados,'%d ',NNODM);
fprintf(dados,'%d ',NTYPE);
fprintf(dados,'%d ',NGAUS);
fprintf(dados,'%d ',KONFN);

%CARACTERISTICAS DOS ELEMENTOS

ITYPE=1;
for J=1:NELEM
    fprintf(dados,'\n%d %d %d %d ',J,MATNO(J),ITYPE,NNODE(J));
    for I=1:NNODE(J)
        fprintf(dados,'%d ',LNODS(I,J));
    end
end

%COORDENADAS DOS NOS

for i=m:-1:1
    if isempty(getappdata(nos(i),'primario'))
        fprintf(dados,'\n%d %.2f %.2f',...
            getappdata(nos(i),'num'),...
            get(nos(i),'xdata'),get(nos(i),'ydata'));
    end
end

%CARACTERÍSTICAS DOS MATERIAIS

materiais=getappdata(gmalha_eixos,'material');
q=size(materiais,1);
for i=1:q
    fprintf(dados,'\n%d\n%G %G %G %G',materiais(i,:));
end

%CONDIÇÕES FRONTEIRA DA FUNCAO POTENCIAL

%cargas totais impostas
for i=1:m
    if ~isempty(getappdata(nos(i),'potencial'))
        fprintf(dados,'\n%d %.2f',...
            getappdata(nos(i),'num'),getappdata(nos(i),'potencial'));
    end
end
%condições fronteira redefiniveis
if KONFN==0;
    for i=1:m
        if ~isempty(getappdata(nos(i),'redefinivel'))

```

```

        redef(i)=getappdata(nos(i),'num');
    end
end
NREVU=size(redef,2);
if NREVU~=0
    fprintf(dados,'\n%d\n',NREVU);
    fprintf(dados,'%d ',redef);
end
end
end

%DEFINICAO DE CAUDAIS IMPOSTOS

%titulo
fprintf(dados,'\n');
%parametros de controle
IQNOD=0;
NQNOD=0;
for i=1:m
    if ~isempty(getappdata(nos(i),'concentrado'))
        IQNOD=1;
        NQNOD=NQNOD+1;
    end
end
NQEDG=NQEDG-1;
if NQEDG >= 1
    IQEDG=1;
else
    IQEDG=0;
end
IQVOL=get(ci(7),'value');
fprintf(dados,'\n%d %d %d %d',IQNOD,IQEDG,IQVOL,1);
%caudais concentrados aplicados nos nós
if IQNOD == 1
    fprintf(dados,'\n%d',NQNOD);
    for i=1:m
        if ~isempty(getappdata(nos(i),'concentrado'))
            fprintf(dados,'\n%d %G',...
                getappdata(nos(i),'num'),getappdata(nos(i),'concentrado'));
        end
    end
end
end
%caudais distribuidos em lados de elementos
if IQEDG == 1
    distribuido=fopen('IQEDG');
    fseek(distribuido,0,'bof');
    for i=1:NQEDG
        linha=fgetl(distribuido);
        mat=str2num(linha);
        p=size(mat,2);
        h_elemento=mat(1);
        elemento=find(h_elemento==elementos);
        fprintf(dados,'\n%d',elemento);
        j=0;
        for i=2:2:p-1
            j=j+1;
            marcador(j)=mat(i);
            qi(j)=mat(i+1);
        end
    end
end

```

```

        k=0;
        for i=1:j
            grupo=get(marcador(i),'parent');
            pai=get(grupo,'parent');
            filhos=get(pai,'children');
            aux=findobj(filhos,'flat','type','line');
            if h_elemento==aux
                num=getappdata(marcador(i),'num');
                fprintf(dados,'\n%d %.14g',num,qi(i));
            end
        end
    end
end
end

```

%CONDIÇÕES-FRONTEIRA DA FUNÇÃO DE CORRENTE

```

NCFIX=0;
for i=1:m
    if ~isempty(getappdata(nos(i),'corrente'))
        NCFIX=NCFIX+1;
    end
end
fprintf(dados,'\n%d',NCFIX);
for i=1:m
    if ~isempty(getappdata(nos(i),'corrente'))
        fprintf(dados,'\n%d %G',...
            getappdata(nos(i),'num'),getappdata(nos(i),'corrente'));
    end
end
fclose(dados);
delete(gmalha_fig)

```

```

%*** GMalha.m *****
%
%*** script que cria o ambiente gráfico onde se vai desenhar a malha *
%
%**** 2010.06.30 * LR ***** quasar_m ****

```

```

gmalha_fig=figure;
titulo='Geração do ficheiro de dados';
set(gmalha_fig,'name',titulo,'numbertitle','off',...
'menubar','none','color','w','units','normalized',...
'position',[0.15 0.15 .7 .7],'tag','gmalha_fig')
set(gmalha_fig,'CloseRequestFcn','closemyfig')

gmalha_eixos=axes;
hold on
set(gmalha_eixos,'units','normalized',...
'outerposition',[0,0,.75,1],...
'tag','gmalha_eixos',...
'xlim',[xmin xmax],'ylim',[ymin ymax],...
'xtick',xmin:grelha:xmax,'ytick',ymin:grelha:ymax)
setappdata(gmalha_eixos,'snap',1)
grid(gmalha_eixos,'on')
if status_escala == 0
    set(gmalha_eixos,'DataAspectRatioMode','auto')
elseif status_escala == 1

```

```

        set(gmalha_eixos,'DataAspectRatio',[1 1 1])
    end

% posicao X e Y:
    uicontrol(gmalha_fig,'Style','frame','Units','normalized',...
        'Position',[0.75 0.96 .25 .04]);
    uicontrol(gmalha_fig,'Style','text',...
        'Units','normalized',...
        'Position',[.8 .965 .03 .03],...
        'String','X =');
    xh=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
        'HorizontalAlignment','left','Tag','GMX',...
        'Position',[.83 .965 .075 .03],'String','0.0');
    uicontrol(gmalha_fig,'Style','text',...
        'Units','normalized',...
        'Position',[.91 .965 .03 .03],...
        'ForegroundColor','k','String','Y =');
    yh=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
        'HorizontalAlignment','left','Tag','GMY',...
        'Position',[.94 .965 .055 .03],'String','0.0');

user_data=[xh,yh];
set(gmalha_fig,'userdata',user_data)
set(gmalha_fig,'WindowButtonMotionFcn','ponteiro')
set(gmalha_fig,'WindowButtonDownFcn','desenhar')

%snap e grelha
    uicontrol(gmalha_fig,'Style','frame','Units','normalized',...
        'Position',[0.75 0.85 .1 .11]);
    check_box_grelha=uicontrol(gmalha_fig,'style','checkbox','units',...
        'normalized','position',[.76,.9,.08,.045],'string','Grelha',...
        'callback','on_off(check_box_grelha,'grelha'),'value',1);
    check_box_snap=uicontrol(gmalha_fig,'style','checkbox','units',...
        'normalized','position',[.76,.86,.08,.045],'string','Snap',...
        'callback','on_off(check_box_snap,'snap'),'value',1);

%Gerar ficheiro de dados
    uicontrol(gmalha_fig,'Units','normalized','string','Gerar ficheiro',...
        'Position',[0.85 0.85 .15 .11],...
        'callback','gerar_ficheiro(nmat,NTYPE,NGAUS,KONFN,ci,NQEDG)');

%botoes para desenhar a malha (desenhar, seleccionar e mover)
    h_toggle(1)=uicontrol(gmalha_fig,'Style','togglebutton','Units',...
        'normalized','Position',[.75 .8 .125 .05],'string','Desenhar',...
        'callback','ftoggle(h_toggle,1)','value',1,...
        'TooltipString',...
        'os nós devem ser desenhados no sentido directo');
    h_toggle(2)=uicontrol(gmalha_fig,'Style','togglebutton','Units',...
        'normalized','Position',[.75 .75 .125 .05],'string','Seleccionar',...
        'callback','ftoggle(h_toggle,2)','tag','seleccionar');
    h_toggle(3)=uicontrol(gmalha_fig,'Style','togglebutton','Units',...
        'normalized','Position',[.75 .7 .125 .05],'string','Mover',...
        'callback','ftoggle(h_toggle,3)');

%pushbutton undo, copiar e eliminar
    uicontrol(gmalha_fig,'Units','normalized',...
        'Position',[.875 .75 .125 .05],'string','Copiar',...

```

```

'callback','fpush(1)')
uicontrol(gmalha_fig,'Units','normalized',...
'Position',[.875 .7 .125 .05],'string','Eliminar',...
'callback','fpush(2)')
uicontrol(gmalha_fig,'Units','normalized',...
'Position',[.875 .8 .125 .05],'string','Undo',...
'callback','fpush(3)','tag','undo')

%conversão dos elementos
uicontrol(gmalha_fig,'Style','frame','Units','normalized',...
'Position',[.75 .55 .25 .15])
uicontrol(gmalha_fig,'Style','text','Units','normalized',...
'Position',[.76 .64 .23 .05],'string','Converter em elem. de:',...
'FontSize',10,'FontWeight','bold')
uicontrol(gmalha_fig,'Units','normalized','string','8 nós',...
'Position',[.825 .61 .1 .04],...
'callback','NODE2D_gmalha(8)')
uicontrol(gmalha_fig,'Units','normalized','string','9 nós',...
'Position',[.825 .565 .1 .04],...
'callback','NODE2D_gmalha(9)')

%material
lista_materiais='nenhum definido';
nmat=0;
uicontrol(gmalha_fig,'Style','frame','Units','normalized',...
'Position',[.75 .44 .25 .11])
uicontrol(gmalha_fig,'Units','normalized','string','Novo Material',...
'Position',[.8 .5 .15 .04],...
'callback','material')
lista=uicontrol(gmalha_fig,'Style','popupmenu','Units','normalized',...
'Position',[.76 .45 .12 .04],'tag','lista',...
'string',lista_materiais,...
'BackgroundColor','w');
uicontrol(gmalha_fig,'Units','normalized','string','Atribuir',...
'Position',[.9 .45 .08 .04],...
'callback','atribuir(1,get(lista,'value'))')

%condicoes fronteira
cf(1)=uicontrol(gmalha_fig,'Style','frame','Units','normalized',...
'Position',[.75 .05 .25 .39]);
cf(2)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
'Position',[.76 .38 .23 .05],'string','Condições Fronteira',...
'FontSize',10,'FontWeight','bold');

cf(3)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
'Position',[.76 .34 .23 .05],'string','Função Potencial',...
'FontSize',10);
cf(4)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
'Position',[.76 .30 .12 .04],'string','Carga Total (m)');
cf(5)=uicontrol(gmalha_fig,'Style','edit','Units','normalized',...
'Position',[.76 .27 .12 .04],'string','0','BackgroundColor','w');
cf(6)=uicontrol(gmalha_fig,'Units','normalized','string','Atribuir',...
'Position',[.9 .27 .08 .04],...
'callback','atribuir(2,get(cf(5),'string'))');
cf(7)=uicontrol(gmalha_fig,'Units','normalized','string','Redefinível',...
'Position',[.8 .22 .15 .04],...
'callback','atribuir(6,'s'))');

```

```

cf(8)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .15 .23 .05],'string','Função Corrente',...
    'FontSize',10);
cf(9)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .11 .12 .04],'string','Valor Imposto');
cf(10)=uicontrol(gmalha_fig,'Style','edit','Units','normalized',...
    'Position',[.76 .08 .12 .04],'string','0','BackgroundColor','w');
cf(11)=uicontrol(gmalha_fig,'Units','normalized','string','Atribuir',...
    'Position',[.9 .08 .08 .04],...
    'callback','atribuir(3,get(cf(10),'string'))');

%substitui parcialmente o menu
k_menu=0;
alternador=uicontrol(gmalha_fig,'Units','normalized',...
    'string','Caudais Impostos',...
    'Position',[.75 0 .25 .05],'FontWeight','bold',...
    'callback','k_menu=alternar(cf,ci,k_menu,alternador);');

%caudais impostos
ci(1)=uicontrol(gmalha_fig,'Style','frame','Units','normalized',...
    'Position',[.75 .05 .25 .39]);
ci(2)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .38 .23 .05],'string','Caudais Impostos',...
    'FontSize',10,'FontWeight','bold');

ci(3)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .34 .23 .05],'string','Caudal concentrado nodal',...
    'FontSize',10);
ci(4)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .30 .12 .04],'string','Caudal Qi (m3/s)',...
    'TooltipString','Qi positivo = fonte; Qi negativo = poço');
ci(5)=uicontrol(gmalha_fig,'Style','edit','Units','normalized',...
    'Position',[.76 .27 .12 .04],'string','0','BackgroundColor','w');
ci(6)=uicontrol(gmalha_fig,'Units','normalized','string','Atribuir',...
    'Position',[.9 .27 .08 .04],...
    'callback','atribuir(4,get(ci(5),'string'))');

ci(7)=uicontrol(gmalha_fig,'Units','normalized',...
    'string','Caudal Distribuido no Volume',...
    'style','checkbox','Position',[.76 .22 .23 .04]);

ci(8)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .15 .23 .05],'string','Caudal Distribuido Lateral',...
    'FontSize',10);
ci(9)=uicontrol(gmalha_fig,'Style','text','Units','normalized',...
    'Position',[.76 .11 .12 .04],'string','qi (m3/s/m)',...
    'TooltipString','qi é positivo quando dirigido para o interior');
ci(10)=uicontrol(gmalha_fig,'Style','edit','Units','normalized',...
    'Position',[.76 .08 .12 .04],'string','0','BackgroundColor','w');
ci(11)=uicontrol(gmalha_fig,'Units','normalized','string','Atribuir',...
    'Position',[.9 .08 .08 .04],...
    'callback','atribuir(5,get(ci(10),'string'))',...
    'TooltipString','seleccionar os nós no sentido directo');
NQEDG=1;
ci(12)=uicontrol(gmalha_fig,'Units','normalized',...
    'Position',[.9 .12 .08 .04],'string',['lado ',num2str(NQEDG)],...
    'callback','NQEDG=atribuir(7,NQEDG);','tag','ci(12)',...
    'TooltipString','seleccionar o elemento a que o lado pertence');

```

```
set(ci,'visible','off')
```

```
*** janela.m *****
%
%*** selecciona os objectos que estão dentro da janela de seleccao e
% des-selecciona os restantes
%
%**** 2010.06.30 * LR ***** quasar_m ****

function janela

%handles
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
%procura rectangulo
rect=findobj(hKids_eixos,'flat','tag','rectan');
%lê as coordenadas do cursor
user_data=get(gmalha_fig,'UserData');
x=get(user_data(1),'String');x=str2num(x);
y=get(user_data(2),'String');y=str2num(y);

if noeixo(gmalha_eixos) && isempty(rect)
    rectangle('position',[x,y,1E-3,1E-3],'tag','rectan');
elseif noeixo(gmalha_eixos) && ~isempty(rect)
    linhas=findobj(hKids_eixos,'type','line');
    xi=get(linhas,'xdata');
    yi=get(linhas,'ydata');
    [m,n]=size(linhas);
    posicao=get(rect,'position');
    k=0;
    l=0;
    for i=1:m
        xii=cell2mat(xi(i));
        if (min(xii)>posicao(1)) && ((posicao(1)+posicao(3))>max(xii))
            l=l+1;
            seleccionados(l)=linhas(i);
            yii(l,:)=yi(i,:);
        else
            k=k+1;
            nao_seleccionados(k)=linhas(i);
        end
    end
    if l>0
        [o,p]=size(seleccionados);
        for i=1:p
            yiii=cell2mat(yii(i));
            if (min(yiii)>posicao(2)) &&
((posicao(2)+posicao(4))>max(yiii))
                set(seleccionados(i),'color','g')
            else
                k=k+1;
                nao_seleccionados(k)=seleccionados(i);
            end
        end
    end
end
```

```

        else
            nao_selecionados=linhas;
        end
        if k>0
            set(nao_selecionados,'color','b')
        end
        delete(rect)
    end
end

```

```

%*** material.m *****
%
%*** abre uma janela onde se introduzem as características do novo
% material
%
%**** 2010.06.30 * LR ***** quasar_m ****

%abre uma janela onde se introduzem as características do novo material
material_inp=figure;
titulo_material_inp='Novo Material';
set(material_inp,'name',titulo_material_inp,'numbertitle','off',...
'menubar','none','units','normalized',...
'position',[0.3 0.3 .4 .4],'tag','material_inp')

%valores iniciais
kx=0;ky=0;theta=0;Q=0;

uicontrol(material_inp,'Style','frame','Units','normalized',...
'Position',[0,0,1,1])
uicontrol(material_inp,'style','text','units','normalized',...
'position',[0.01,.88,.98,.07],'string','Novo Material',...
'FontSize',12,'FontWeight','bold')
%n° do material
uicontrol(material_inp,'style','text','units','normalized',...
'position',[.1,.765,.2,.06],'string','N° do material',...
'HorizontalAlignment','left')
nmat=nmat+1;
n_material=uicontrol(material_inp,'style','text','units','normalized',...
'position',[.27,.78,.05,.06],'string',nmat,...
'FontSize',12,'FontWeight','bold');

%propriedades do material

%permeabilidade
uicontrol(material_inp,'style','text','units','normalized',...
'position',[.1,.65,.6,.06],'HorizontalAlignment','left',...
'string','Coeficientes de permeabilidade principais (m/s)')
%kx
uicontrol(material_inp,'style','text','units','normalized',...
'position',[.19,.55,.12,.06],'string','kx')
k_x=uicontrol(material_inp,'style','edit','units','normalized',...
'position',[.315,.55,.12,.08],'BackgroundColor','w',...
'string','0','callback','kx=editavel(k_x);');
%ky
uicontrol(material_inp,'style','text','units','normalized',...
'position',[.565,.55,.12,.06],'string','ky')
k_y=uicontrol(material_inp,'style','edit','units','normalized',...
'position',[.69,.55,.12,.08],'BackgroundColor','w',...

```

```

    'string','0','callback','ky=editavel(k_y);');
%angulo
uicontrol(material_inp,'style','text','units','normalized',...
    'position',[.1,.4,.6,.1],'HorizontalAlignment','left',...
    'string',{'ângulo do eixo de permeabilidade principal';...
    'com o semi-eixo positivo (graus)'}))
uicontrol(material_inp,'style','text','units','normalized',...
    'position',[.565,.35,.12,.06],'string','theta')
angulo=uicontrol(material_inp,'style','edit','units','normalized',...
    'position',[.69,.35,.12,.08],'BackgroundColor','w',...
    'string','0','callback','theta=editavel(angulo);');
%caudal
uicontrol(material_inp,'style','text','units','normalized',...
    'position',[.1,.2,.6,.1],'HorizontalAlignment','left',...
    'string','caudal produzido por unidade de volume (m3/s)')
uicontrol(material_inp,'style','text','units','normalized',...
    'position',[.565,.15,.12,.06],'string','Q')
caudal=uicontrol(material_inp,'style','edit','units','normalized',...
    'position',[.69,.15,.12,.08],'BackgroundColor','w',...
    'string','0','callback','Q=editavel(caudal);');

% ok
uicontrol(material_inp,'string','ok','units',...
    'normalized','position',[.4,.05,.2,.1],...
    'callback',...
    'lista_materiais=ok_material(lista_materiais,nmat,kx,ky,theta,Q);')

```

```

%*** mover.m *****
%
%*** desloca os objectos seleccionados
%
%**** 2010.06.30 * LR ***** quasar_m ****

function mover

gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
if isempty(hKids_eixos)
    aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
        'string','Aviso: Não há objectos desenhados','Units',...
        'normalized','position',[0 0 .75 .05],...
        'fontsize',10);
    pause(2)
    delete(aviso)
    return
end
todos=findobj(hKids_eixos,'color',[0 1 0]);
linhas=findobj(todos,'userdata',1);
n=size(linhas,1);
for i=1:n
    pai=get(linhas(i),'parent');
    filhos=get(pai,'children');
    h_marcaador=findobj(filhos,'flat','type','hggroup');
    if i==1
        marcadores=get(h_marcaador,'children');
    end
end

```

```

        else
            marc=get(h_marcador,'children');
            marcadores=cat(1,marcadores,marc);
        end
    end
end
if isempty(linhas)
    aviso=uicontrol(gmalha_fig,'Style','text','BackgroundColor','w',...
        'string','Aviso: Não há objectos seleccionados','Units',...
        'normalized','position',[0 0 .75 .05],...
        'fontsize',10);
    pause(2)
    delete(aviso)
    return
end
if noeixo(gmalha_eixos),
    user_data=get(gmalha_fig,'UserData');
    x=get(user_data(1),'String');x=str2num(x);
    y=get(user_data(2),'String');y=str2num(y);
    inicial=getappdata(gmalha_eixos,'mover');
    if isempty(inicial)
        inicial=[x,y];
        setappdata(gmalha_eixos,'mover',inicial)
        return
    end
    delta=[x,y]-inicial;
    xi=get(linhas,'xdata');
    yi=get(linhas,'ydata');
    if n>1
        xi=cell2mat(xi);
        yi=cell2mat(yi);
    end
    for i=1:n
        set(linhas(i),'xdata',xi(i,:)+delta(1))
        set(linhas(i),'ydata',yi(i,:)+delta(2))
    end
    [o,p]=size(marcadores);
    q=o*p;
    for i=1:q
        xm=get(marcadores(i),'xdata');
        set(marcadores(i),'xdata',xm+delta(1))
        ym=get(marcadores(i),'ydata');
        set(marcadores(i),'ydata',ym+delta(2))
    end
    setappdata(gmalha_eixos,'mover',[])
end
end

```

```

%***  NODE2D_gmalha.m  *****
%
%***  GERA AS COORDENADAS DE:
%      * NOS SITUADOS A MEIO DOS LADOS RECTILINEOS DE ELEMENTOS PLANOS
%      DE 8 E 9 NOS
%      * NO CENTRAL DE ELEMENTOS DE 9 NOS
%
%**** 2010.06.30 * LR ***** quasar_m ****

function NODE2D_gmalha(NNODE)

```

```

NODES=1:9;
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
todos=findobj(hKids_eixos,'color',[0 1 0]);
linhas=findobj(todos,'userdata',1);
[NELEM,lixo]=size(linhas);

% CICLO SOBRE OS ELEMENTOS
k=0;
for IELEM = 1:NELEM
    COORD=ones(2,9);
    COORD=inf*COORD;
    pai=get(linhas(IELEM),'parent');
    filhos=get(pai,'children');
    h_marcador=findobj(filhos,'flat','type','hggroup');
    nos=get(h_marcador,'children');
    m=size(nos,1);
    for i=1:m
        marcadores(i)=nos(m+1-i);
    end
    j=0;
    if m==4
        for i=1:2:8
            j=j+1;
            COORD(1,i)=get(marcadores(j),'xdata');
            COORD(2,i)=get(marcadores(j),'ydata');
        end
    elseif m==8
        for i=1:m
            j=j+1;
            COORD(1,i)=get(marcadores(j),'xdata');
            COORD(2,i)=get(marcadores(j),'ydata');
        end
    elseif m==9
        return
    end
end

% NOS SITUADOS A MEIO DE LADOS RECTILINEOS (elementos de 8 e 9 nos)
for INOD2 = 2:2:8
    IPOI2 = NODES(INOD2);
    TOTAL = abs(COORD(1,IPOI2))+abs(COORD(2,IPOI2));
    if TOTAL == inf
        INOD1 = INOD2-1;
        INOD3 = INOD2+1;
        if INOD3 == 9
            INOD3 = 1;
        end
        IPOI1 = NODES(INOD1);
        IPOI3 = NODES(INOD3);
        for I = 1:2
            COORD(I,IPOI2)=(COORD(I,IPOI1)+COORD(I,IPOI3))*0.5;
        end
    end
end
end

```

```

% NO CENTRAL (elementos de 9 nos)
if NNODE == 9
    IPOI9 = NODES(9);
    TOTAL = abs(COORD(1,IPOI9))+abs(COORD(2,IPOI9));
    if TOTAL == inf
        COORD(1,IPOI9) = 0;
        COORD(2,IPOI9) = 0;
        for INODE = 1:2:7
            NCORN = NODES(INODE);
            NMIDS = NODES(INODE+1);
            for IDIME = 1:2
                COORD(IDIME,IPOI9) = COORD(IDIME,IPOI9)...
                    - COORD(IDIME,NCORN)/4 + COORD(IDIME,NMIDS)/2;
            end
        end
    end
end
delete(marcadores)
for i=1:NNODE
    marcador=plot(gmalha_eixos,COORD(1,i),COORD(2,i),'ob','userdata',2);
    set(marcador,'parent',h_marcador)
end
end

```

```

%***  noeixo.m  *****
%
%***  determina se o ponteiro se encontra dentro dos eixos
%
%****  2010.06.30 * LR ***** quasar_m ****

function onax=noeixo(handle_eixos)

xlim=get(handle_eixos,'XLim');
ylim=get(handle_eixos,'YLim');
p=get(handle_eixos,'CurrentPoint');
onax = (p(1,1) >= xlim(1) & p(1,1) <= xlim(2) &...
    p(1,2) >= ylim(1) & p(1,2) <= ylim(2));

```

```

%***  ok.m  *****
%
%***  lê o valor e encerra a janela
%
%****  2010.06.30 * LR ***** quasar_m ****

function status=ok(handle)
status=get(handle,'value');
delete(gcf)

```

```

%***  lista_materiais.m  *****
%
%***  lê o valor das propriedades do material e encerra a janela
%
%****  2010.06.30 * LR ***** quasar_m ****

```

```

function lista_materiais=ok_material(lista_materiais,nmat,kx,ky,theta,Q)
%armazenar os valores
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids=get(gmalha_fig,'Children');
h_lista=findobj(hKids,'flat','tag','lista');
gmalha_eixos=findobj(hKids,'flat','tag','gmalha_eixos');
if isappdata(gmalha_eixos,'material')
    material=getappdata(gmalha_eixos,'material');
    setappdata(gmalha_eixos,'material',[material;nmat kx ky theta Q])
else
    setappdata(gmalha_eixos,'material',[nmat kx ky theta Q])
end
if all(strcmp(lista_materiais,'nenhum definido'))
    lista_materiais=['Material ',int2str(nmat)];
else
    lista_materiais=[lista_materiais;['Material ',int2str(nmat)]];
end
set(h_lista,'string',lista_materiais)
delete(gcf)

```

```

%***  on_off.m  *****
%
%***  liga ou desliga o snap e a grelha
%
%****  2010.06.30 * LR *****  quasar_m ****

function on_off(h_check_box,accao)
status=get(h_check_box,'value');
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids,'flat','tag','gmalha_eixos');
if status == 0
    if strcmp(accao,'snap')
        setappdata(gmalha_eixos,accao,0)
    elseif strcmp(accao,'grelha')
        grid(gmalha_eixos,'off')
    end
elseif status == 1
    if strcmp(accao,'snap')
        setappdata(gmalha_eixos,accao,1)
    elseif strcmp(accao,'grelha')
        grid(gmalha_eixos,'on')
    end
end
end

```

```

%***  ponteiro.m  *****
%
%***  obtem as coordenadas da localização do ponteiro
%
%****  2010.06.30 * LR *****  quasar_m ****

function ponteiro

%handles
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids=get(gmalha_fig,'Children');

```

```

gmalha_eixos=findobj(hKids,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
ptr=get(gmalha_fig,'Pointer');
toggle=findobj(gmalha_fig,'tag','seleccionar');
%procura rectangulo
rectangulo=findobj(hKids_eixos,'flat','tag','rectan');

if noeixo(gmalha_eixos),

    p=get(gmalha_eixos,'CurrentPoint');
    [xcurr,ycurr]=snap(gmalha_eixos,p,getappdata(gmalha_eixos,'snap'));
    if all(ptr(1:2)=='ar') && ~(get(toggle,'value'))
        set(gmalha_fig,'Pointer','crosshair');
    end
    user_data=get(gmalha_fig,'UserData');
    set(user_data(1),'String',sprintf('%.4g',xcurr))
    set(user_data(2),'String',sprintf('%.4g',ycurr))
    if ~isempty(rectangulo)
        posicao=get(rectangulo,'position');
        deltax=xcurr-posicao(1);
        deltay=ycurr-posicao(2);
        if deltax<=0
            deltax=1E-3;
        end
        if deltay<=0
            deltay=1E-3;
        end
        set(rectangulo,'position',[posicao(1),posicao(2),deltax,deltay])
    end

else
    set(gmalha_fig,'Pointer','arrow')
end

```

```

%*** prg_ini.m *****
%
%*** pergunta se se pretende criar um ficheiro de dados ou ler um
% existente
%
%**** 2010.06.30 * LR ***** quasar_m ****

dados_inp=figure;
titulo_dados_inp='QUASAR';
set(dados_inp,'name',titulo_dados_inp,'numbertitle','off',...
'menubar','none','units','normalized',...
'position',[0.35 0.4 .3 .2],'tag','dados_inp')
uicontrol(dados_inp,'Style','frame','Units','normalized',...
'Position',[0,0,1,1]);
uicontrol(dados_inp,'style','text','units','normalized',...
'position',[0.01,.8,.98,.12],'string','QUASAR',...
'FontSize',12,'FontWeight','bold')
uicontrol(dados_inp,'style','text','units','normalized',...
'position',[0.01,.5,.98,.2],...
'string',...
{'Pretende criar o ficheiro de dados,','ou ler um existente?'})
uicontrol(dados_inp,'units','normalized',...
'position',[0.2,.2,.2,.15],'string','criar',...

```

```

'callback', ['opcao=' 'criar' ';' , 'delete(gcf)'])
uicontrol(dados_inp, 'units', 'normalized', ...
'position', [0.6, .2, .2, .15], 'string', 'ler', ...
'callback', ['opcao=' 'ler' ';' , 'delete(gcf)'])

```

```

%*** problema.m *****
%
%*** abertura da janela onde se dimensionam os eixos da pag. principal *
%
%**** 2010.06.30 * LR ***** quasar_m ****

function [NTYPE,NGAUS,KONFN]=problema

problema_inp=figure;
titulo_problema_inp='Definição do Problema';
set(problema_inp, 'name', titulo_problema_inp, 'numbertitle', 'off', ...
'menubar', 'none', 'units', 'normalized', ...
'position', [0.3 0.3 .4 .4], 'tag', 'problema_inp')

uicontrol(problema_inp, 'Style', 'frame', 'Units', 'normalized', ...
'Position', [0,0,1,1]);
uicontrol(problema_inp, 'style', 'text', 'units', 'normalized', ...
'position', [0.01, .88, .98, .07], 'string', 'Definição do Problema', ...
'FontSize', 12, 'FontWeight', 'bold')

%tipo de problema
NTYPE=1;
uicontrol(problema_inp, 'style', 'text', 'units', 'normalized', ...
'position', [0.3, .7, .4, .1], 'string', 'Tipo de problema', ...
'FontSize', 10, 'FontWeight', 'bold')
h_tipo(1)=uicontrol(problema_inp, 'style', 'radio', 'units', ...
'normalized', 'position', [.35,0.65, .3, .05], 'string', 'Plano', ...
'callback', 'NTYPE=tipo(h_tipo,1);', 'value', 1);
h_tipo(2)=uicontrol(problema_inp, 'style', 'radio', 'units', ...
'normalized', 'position', [.35,0.58, .35, .05], ...
'string', 'Axissimetrico', ...
'callback', 'NTYPE=tipo(h_tipo,2);');
%nº de pts de integração
NGAUS=2;
uicontrol(problema_inp, 'style', 'text', 'units', 'normalized', ...
'position', [0.01, .45, .98, .1], ...
'string', 'Número de pontos de integração em cada direcção', ...
'FontSize', 10, 'FontWeight', 'bold')
h_gauss(1)=uicontrol(problema_inp, 'style', 'radio', 'units', ...
'normalized', 'position', [.35,0.4, .3, .05], 'string', '2', ...
'callback', 'NGAUS=gauss(h_gauss,1);', 'value', 1);
h_gauss(2)=uicontrol(problema_inp, 'style', 'radio', 'units', ...
'normalized', 'position', [.35,0.33, .35, .05], ...
'string', '3', ...
'callback', 'NGAUS=gauss(h_gauss,2);');
%tipo de confinamento
KONFN=1;
uicontrol(problema_inp, 'style', 'text', 'units', 'normalized', ...
'position', [0.01, .2, .98, .1], ...
'string', 'Tipo de Confinamento', ...
'FontSize', 10, 'FontWeight', 'bold')
h_conf(1)=uicontrol(problema_inp, 'style', 'radio', 'units', ...

```

```

        'normalized','position',[.35,0.15,.3,.05],'string','Confinado',...
        'callback','NGAUS=conf(h_conf,1);','value',1);
h_conf(2)=uicontrol(problema_inp,'style','radio','units',...
        'normalized','position',[.35,0.08,.35,.05],...
        'string','Não Confinado',...
        'callback','NGAUS=conf(h_conf,2);');
% ok
uicontrol(problema_inp,'string','ok','units',...
        'normalized','position',[.75,.06,.2,.1],...
        'callback',['set(ok_eixoslim,'enable','on');','delete(gcf)'])

```

```

%***  rectangulo.m  ****
%
%***  desenha a janela de seleccao
%
%****  2010.06.30 * LR **** quasar_m ****

function rectangulo

disp('ola')
%handles
gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids_fig=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids_fig,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
%procura rectangulo
rect=findobj(hKids_eixos,'flat','tag','rectan')
%lê as coordenadas do cursor
user_data=get(gmalha_fig,'UserData');
x=get(user_data(1),'String');x=str2num(x);
y=get(user_data(2),'String');y=str2num(y);
if noeixo(gmalha_eixos) && isempty(rect)
    rectangle('position',[x,y,1,1],'tag','rectan');
elseif noeixo(gmalha_eixos) && ~isempty(rect)
    delete(rect)
end

```

```

%***  select.m  ****
%
%***  altera a cor do objecto para que seja identificado como
%      seleccionado, ou nao
%
%****  2010.06.30 * LR **** quasar_m ****

function select

gmalha_fig=findobj(get(0,'Children'),'flat','tag','gmalha_fig');
hKids=get(gmalha_fig,'Children');
gmalha_eixos=findobj(hKids,'flat','tag','gmalha_eixos');
hKids_eixos=get(gmalha_eixos,'Children');
rectangulo=findobj(hKids_eixos,'flat','tag','rectan');
delete(rectangulo)

status=get(gcbo,'color');
if all(status == [0 0 1])
    set(gcbo,'color',[0 1 0])

```

```
elseif all(status == [0 1 0])
    set(gcbo,'color',[0 0 1])
end
```

```
%*** snap.m *****
%
%*** identifica o inteiro mais próximo do ponteiro
%
%**** 2010.06.30 * LR ***** quasar_m ****

function [x,y]=snap(ax,pv,flag)

xticks=get(ax,'XTick');
yticks=get(ax,'YTick');
xlim=get(ax,'XLim');
ylim=get(ax,'YLim');
if isempty(xticks) || isempty(yticks) || flag==0,
    x=pv(1,1); y=pv(1,2);
else
    x=xticks(min(abs(xticks-pv(1,1))==abs(xticks-pv(1,1))));
    x=x(1);
    y=yticks(min(abs(yticks-pv(1,2))==abs(yticks-pv(1,2))));
    y=y(1);
end
x=sort([xlim x]); x=x(2);
y=sort([ylim y]); y=y(2);
```

```
%*** tipo.m *****
%
%*** atribui o tipo de problema
%
%**** 2010.06.30 * LR ***** quasar_m ****

function NTYPE=tipo(h_radio,button)

% garante que quando se carrega num botão este permanece activo
% e os restantes são desactivados
num_buttons=length(h_radio);
if get(h_radio(button),'value') == 1
    set(h_radio([1:(button-1),(button+1):num_buttons]),'value',0)
else
    set(h_radio(button),'value',1)
end

if button == 1
    NTYPE=1;
elseif button ==2
    NTYPE=2;
end
```

A2

CÁLCULO

```

**** BDCOND.m *****
%
%**** INTRODUZ AS CONDICAOES-FRONTIEIRA DE DIRICHLET *
%
%**** 2010.06.11 * JCM ***** quasar ****
function [GSTIF,GLOAD]=BDCOND(GSTIF,NTOTV,GLOAD,IFFIX,FIXED)
BIG=1E30;
for ITOTV = 1:NTOTV
    if IFFIX(ITOTV) == 1
        PIVOT = GSTIF(ITOTV,ITOTV);
        GSTIF(ITOTV,ITOTV) = PIVOT*(1+BIG);
        GLOAD(ITOTV) = BIG*PIVOT*FIXED(ITOTV);
    end
end
end

```

```

C***** SUBROUTINE BDCOND (GSTIF, NTOTV, GLOAD, IFFIX, FIXED) *****
C
C*** INTRODUZ AS CONDICAOES-FRONTIEIRA DE DIRICHLET
C
C**** 2010.06.24 * JCM ***** quasar *
C***** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C***** DIMENSION IFFIX(NTOTV), FIXED(NTOTV), GSTIF(NTOTV,NTOTV),
C***** . GLOAD(NTOTV)
C***** DATA BIG/ 1.0D30/
C
C DO ITOTV = 1,NTOTV
C IF (IFFIX(ITOTV).EQ.1) THEN
C PIVOT = GSTIF(ITOTV,ITOTV)
C GSTIF(ITOTV,ITOTV) = PIVOT*(1.0D0+BIG)
C GLOAD(ITOTV) = BIG*PIVOT*FIXED(ITOTV)
C END IF
C END DO
C
C RETURN
C END

```



```

%*** CLOAD0.m *****
%
%*** CALCULA TERMO INDEPENDENTE P/ A SOLUCAO EM TERMOS DA F.DE CORRENTE *
%
%**** 2010.06.30 * LR ***** quasar_m ****
%
% INICIALIZACAO
GLOAD=zeros(NPOIN,1);
%
% CICLO SOBRE OS ELEMENTOS
KGASP = 0;
for IELEM = 1:NELEM
%
% COORDENADAS DOS NOS DO ELEMENTO
NODES=LNODS(1:NODE, IELEM);
ELCOD=COORD(:, NODES);
%
% CICLOS SOBRE OS PONTOS DE GAUSS
KGASP = 0;
for IGAUS = 1:NGAUS
EXISP = POSGP(IGAUS);
for JGAUS = 1:NGAUS
ETASP = POSGP(JGAUS);
KGASP = KGASP+1;
KGASP = KGASP+1;
[ CARTD,DJACB]=JACOB2(EXISP,ETASP,...
NDIME,NNODE,ELCOD,KGASP,IELEM,resultados);
DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS);
%
% CALCULO E ESPALHAMENTO DAS CARGAS
for INODE = 1:NNODE
IPOIN = NODES(INODE);
GLODI =(CARTD(1,INODE))*VELOC(2,KGAUS)-
CARTD(2,INODE)*VELOC(1,KGAUS))*DVOLU;
GLOAD(IPOIN) = GLOAD(IPOIN)-GLODI;
end
end
end
end
end

```

```

SUBROUTINE CLOAD0 (COORD, LNODS, NELEM, NNODE, NGAUS, NDIME,
NPOIN, POSGP, WEIGP, GLOAD, NTOTG, VELOC)
C*****
C*****
C***** CALCULA TERMO INDEPENDENTE P/ A SOLUCAO EM TERMOS DA F.DE CORRENTE
C*****
C***** 2010.06.11 * JCM ***** quasar *
C*****
C***** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C***** DIMENSION POSGP(NGAUS), LNODS(NNODE, NELEM),
C***** GLOAD(NPOIN), COORD(NDIME, NPOIN),
C***** WEIGP(NGAUS), VELOC(NDIME, NTOTG)
C***** DIMENSION CARTD(2,9), NODES(9),
C***** ELCOD(2,9)
C***** INICIALIZACAO
C*****
C***** GLOAD = 0.0D0
C*****
C***** CICLO SOBRE OS ELEMENTOS
C*****
C***** KGASP = 0
C***** DO IELEM = 1, NELEM
C*****
C***** COORDENADAS DOS NOS DO ELEMENTO
C*****
C***** DO INODE = 1, NNODE
C***** IPOIN = LNODS(INODE, IELEM)
C***** NODES(INODE) = IPOIN
C***** DO IDIME = 1, NDIME
C***** ELCOD(IDIME, INODE) = COORD(IDIME, IPOIN)
C***** END DO
C***** END DO
C*****
C***** CICLOS SOBRE OS PONTOS DE GAUSS
C*****
C***** KGASP = 0
C***** DO IGAUS = 1, NGAUS
C***** EXISP = POSGP(IGAUS)
C***** DO JGAUS = 1, NGAUS
C***** ETASP = POSGP(JGAUS)
C***** KGASP = KGASP+1
C***** KGASP = KGASP+1
C***** CALL JACOB2 (CARTD, DJACB, ELCOD, KGASP, IELEM, ETASP,
C***** EXISP, NNODE, NDIME)
C***** DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
C*****
C***** CALCULO E ESPALHAMENTO DAS CARGAS
C*****
C***** DO INODE = 1, NNODE
C***** IPOIN = NODES(INODE)
C***** GLODI =(CARTD(1,INODE))*VELOC(2,KGAUS)-
C***** CARTD(2,INODE)*VELOC(1,KGAUS))*DVOLU
C***** GLOAD(IPOIN) = GLOAD(IPOIN)-GLODI
C***** END DO
C*****

```

```
END DO
END DO
END DO
RETURN
END
```

C

```

*** CSTIF0.m *****
%
%*** FORMULACAO EM TERMOS DA FUNCAO DE CORRENTE NODAL PSI
%
%*** CALCULA A MATRIZ DE RIGIDEZ "ESTIF" DE CADA ELEMENTO E PROCEDE
% AO RESPECTIVO ESPALHAMENTO NA MATRIZ DE RIGIDEZ GLOBAL "GSTIF"
%
%***** 2010.06.30 * LR ***** quasar_m ****

% INICIALIZACAO DA MATRIZ DE RIGIDEZ GLOBAL
GSTIF=zeros(NTOTV);

% CICLO SOBRE OS ELEMENTOS
KGAUS = 0;
for IELEM = 1:NELEM

    % INICIALIZACAO DA MATRIZ DE RIGIDEZ DO ELEMENTO
    ESTIF=zeros(NEVAB);

    % NO. DOS GOL E COORDENADAS DOS NOS DO ELEMENTO
    NODES=LNODS(1:NNODE,IELEM);
    ELCOD=COORD(:,NODES);

    % CICLOS SOBRE OS PONTOS DE GAUSS
    KGASP = 0;
    for IGAUS = 1:NGAUS
        EXISP = POSGP(IGAUS);
        for JGAUS = 1:NGAUS
            KGAUS = KGAUS+1;
            KGASP = KGASP+1;
            ETASP = POSGP(JGAUS);

            % DETERMINACAO DAS DERIVADAS CARTESIANAS DAS FUNCOES DE FORMA E
            % DO ELEMENTO DE VOLUME
            [CARTD,DJACB]=JACOB2(EXISP,ETASP,...
                NDIME,NNODE,ELCOD,KGASP,IELEM,resultados);
            DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS);
            if NTYPE == 2
                DVOLU = DVOLU/GPCOD(1,KGAUS);
            end

            % CALCULO DA MATRIZ DE RIGIDEZ DO ELEMENTO
            ISTIF = 0;
            for JNODE = 1:NNODE
                DNJDX = CARTD(1,JNODE);
                DNJDY = CARTD(2,JNODE);
                for INODE = 1:NNODE
                    DNIDX = CARTD(1,INODE);
                    DNIDY = CARTD(2,INODE);
                    ISTIF = ISTIF+1;
                    ESTIF(INODE,JNODE) = ESTIF(INODE,JNODE)...
                        + (DNIDX*DNJDX + DNIDY*DNJDY)*DVOLU;
                end
            end
        end
    end

% ESPALHAMENTO DA MATRIZ DE RIGIDEZ DO ELEMENTO

```

```

SUBROUTINE CSTIF0 (COORD, NDIME, LNODS, GSTIF, NTOTV, NELEM,
    POSGP, WEIGP, NGAUS, NTYPE, ESTIF, NEVAB,
    NNODE, NPOIN, GPCOD, NTOTG)
C*****
C
C*** FORMULACAO EM TERMOS DA FUNCAO DE CORRENTE NODAL PSI
C
C*** CALCULA A MATRIZ DE RIGIDEZ "ESTIF" DE CADA ELEMENTO E PROCEDE
C AO RESPECTIVO ESPALHAMENTO NA MATRIZ DE RIGIDEZ GLOBAL "GSTIF"
C
C***** 2010.06.11 * JCM ***** quasar *
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C DIMENSION GPCOD(NDIME,NTOTG), ESTIF(NEVAB,NEVAB), WEIGP(NGAUS),
    LNODS(NNODE,NELEM), GSTIF(NTOTV,NTOTV), POSGP(NGAUS),
    COORD(NDIME,NPOIN)
C
C DIMENSION ELCOD(2,9), CARTD(2,9), NODES(9)
C
C INICIALIZACAO DA MATRIZ DE RIGIDEZ GLOBAL
GSTIF = 0.0D0
C
C CICLO SOBRE OS ELEMENTOS
C
C KGAUS = 0
C DO IELEM = 1,NELEM
C
C INICIALIZACAO DA MATRIZ DE RIGIDEZ DO ELEMENTO
C
C ESTIF = 0.0D0
C
C NO. DOS GDL E COORDENADAS DOS NOS DO ELEMENTO
C
C DO INODE = 1,NNODE
C IPOIN = LNODS(INODE,IELEM)
C NODES(INODE) = IPOIN
C DO IDIME = 1,NDIME
C ELCOD(IDIME,INODE) = COORD(IDIME,IPOIN)
C
C END DO
C
C END DO
C
C CICLOS SOBRE OS PONTOS DE GAUSS
C
C KGASP = 0
C DO IGAUS = 1,NGAUS
C EXISP = POSGP(IGAUS)
C DO JGAUS = 1,NGAUS
C KGAUS = KGAUS+1
C KGASP = KGASP+1
C ETASP = POSGP(JGAUS)
C
C DETERMINACAO DAS DERIVADAS CARTESIANAS DAS FUNCOES DE FORMA E
C DO ELEMENTO DE VOLUME
C
C CALL JACOB2 (CARTD, DJACB, ELCOD, KGAUS, IELEM, ETASP,

```

```

GSTIF(NODES,NODES)=GSTIF(NODES,NODES)+ESTIF;
end

```

```

      .
      DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
      IF(NTYPE.EQ.2) DVOLU = DVOLU/GPCOD(1,KGAUS)
C*** CALCULO DA MATRIZ DE RIGIDEZ DO ELEMENTO
C
      ISTIF = 0
      DO JNODE = 1,NNODE
        DNJDX = CARTD(1,JNODE)
        DNJDY = CARTD(2,JNODE)
        DO INODE = 1,NNODE
          DNIDX = CARTD(1,INODE)
          DNIDY = CARTD(2,INODE)
          ISTIF = ISTIF+1
          ESTIF(INODE,JNODE) = ESTIF(INODE,JNODE)
            + (DNIDX*DNJDX + DNIDY*DNJDY)*DVOLU
        END DO
      END DO
      END DO
      END DO
C*** ESPALHAMENTO DA MATRIZ DE RIGIDEZ DO ELEMENTO
C
      DO JNODE = 1,NNODE
        JPOIN = NODES(JNODE)
        DO INODE = 1,NNODE
          IPOIN = NODES(INODE)
          GSTIF(IPOIN,JPOIN) =
            GSTIF(IPOIN,JPOIN) + ESTIF(INODE,JNODE)
        END DO
      END DO
      END DO
      RETURN
      END
C

```

```

%*** GMaIha.m *****
%
%*** CALCULA AS DERIVADAS DAS FUNCOES DE FORMA PARA *****
%
% * ELEMENTOS PLANOS DE 4,8 E 9 NOS
%
%***** 2010.06.30 * LR ***** quasar_m ****
function DERIV=DERIV2(S,T,NNODE)
RNAME='DERIV2';
DERIV=zeros(2,NNODE);
% ELEMENTOS DE 4 NOS
if NNODE == 4
    SM = (1-S)/2;
    SP = (1+S)/2;
    TM = (1-T)/2;
    TP = (1+T)/2;
    DM = -0.5;
    DP = +0.5;
    DERIV(1, 1) = DM*TM;
    DERIV(2, 1) = SM*DM;
    DERIV(1, 2) = DP*TM;
    DERIV(2, 2) = SP*DM;
    DERIV(1, 3) = DP*TP;
    DERIV(2, 3) = SP*DP;
    DERIV(1, 4) = DM*TP;
    DERIV(2, 4) = SM*DP;
% ELEMENTOS DE 8 NOS
elseif NNODE == 8
    SM = 0.5*(1-S);
    TM = 0.5*(1-T);
    SP = 0.5*(1+S);
    TP = 0.5*(1+T);
    SS = 0.5*(1-S*S);
    TT = 0.5*(1-T*T);
    S2 = -2*S;
    T2 = -2*T;
    AA = +SM+TM- 1.5;
    DERIV(1, 1) = - TM*(AA+SM);
    DERIV(2, 1) = -SM *(AA+TM);
    DERIV(1, 2) = +S2*TM;
    DERIV(2, 2) = -SS;
    AA = +SP+TM- 1.5;
    DERIV(1, 3) = + TM*(AA+SP);
    DERIV(2, 3) = -SP *(AA+TM);

```

```

C***** SUBROUTINE DERIV2 (DERIV, ETASP, EXISP, NNODE) *****
C
C*** CALCULA AS DERIVADAS DAS FUNCOES DE FORMA PARA
C * ELEMENTOS PLANOS DE 4,8 E 9 NOS
C
C***** 2010.06.11 * JCM *****
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C CHARACTER RNAME*6
C DIMENSION DERIV(2,9)
C DATA RNAME/ 'DERIV2' /
C
C S = EXISP
C T = ETASP
C
C*** ELEMENTOS DE 4 NOS
C
C IF (NNODE.EQ.4) THEN
C
C     SM = (1.0D0-S)/2.0D0
C     SP = (1.0D0+S)/2.0D0
C     TM = (1.0D0-T)/2.0D0
C     TP = (1.0D0+T)/2.0D0
C     DM = -0.5D0
C     DP = +0.5D0
C
C     DERIV(1, 1) = DM*TM
C     DERIV(2, 1) = SM*DM
C
C     DERIV(1, 2) = DP*TM
C     DERIV(2, 2) = SP*DM
C
C     DERIV(1, 3) = DP*TP
C     DERIV(2, 3) = SP*DP
C
C     DERIV(1, 4) = DM*TP
C     DERIV(2, 4) = SM*DP
C
C *** ELEMENTOS DE 8 NOS
C
C ELSE IF (NNODE.EQ.8) THEN
C
C     SM = 0.5D0*(1.0D0-S)
C     TM = 0.5D0*(1.0D0-T)
C     SP = 0.5D0*(1.0D0+S)
C     TP = 0.5D0*(1.0D0+T)
C     SS = 0.5D0*(1.0D0-S*S)
C     TT = 0.5D0*(1.0D0-T*T)
C     S2 = -2.0D0*S
C     T2 = -2.0D0*T
C
C     AA = +SM+TM- 1.5D0
C     DERIV(1, 1) = - TM*(AA+SM)
C     DERIV(2, 1) = -SM *(AA+TM)

```

```

DERIV(1, 4) = + TT;
DERIV(2, 4) = +SP*T2;

AA = +SP+TP- 1.5;
DERIV(1, 5) = + TP*(AA+SP);
DERIV(2, 5) = +SP *(AA+TP);

DERIV(1, 6) = +S2*TP;
DERIV(2, 6) = +SS;

AA = +SM+TP- 1.5;
DERIV(1, 7) = - TP*(AA+SM);
DERIV(2, 7) = +SM *(AA+TP);

DERIV(1, 8) = - TT;
DERIV(2, 8) = +SM*T2;

% ELEMENTOS DE 9 NOS
elseif NNODE == 9
    SSM = (S*S-S)/2;
    SSP = (S*S+S)/2;
    SS0 = -S*S+1;
    TTM = (T*T-T)/2;
    TTP = (T*T+T)/2;
    TT0 = -T*T+1;
    DSM = -0.5+S;
    DSP = +0.5+S;
    DS0 = -2*S;
    DTM = -0.5+T;
    DTP = +0.5+T;
    DT0 = -2*T;

    DERIV(1, 1) = DSM*TTM;
    DERIV(2, 1) = SSM*DTM;

    DERIV(1, 2) = DS0*TTM;
    DERIV(2, 2) = SS0*DTM;

    DERIV(1, 3) = DSP*TTM;
    DERIV(2, 3) = SSP*DTM;

    DERIV(1, 4) = DSP*TT0;
    DERIV(2, 4) = SSP*DT0;

    DERIV(1, 5) = DSP*TTP;
    DERIV(2, 5) = SSP*DTP;

    DERIV(1, 6) = DS0*TTP;
    DERIV(2, 6) = SS0*DTP;

    DERIV(1, 7) = DSM*TTP;
    DERIV(2, 7) = SSM*DTP;

    DERIV(1, 8) = DSM*TT0;
    DERIV(2, 8) = SSM*DT0;

    DERIV(1, 9) = DS0*TT0;

```

```

C
DERIV(1, 2) = +S2*TM
DERIV(2, 2) = -SS

C
AA = +SP+TM- 1.5D0
DERIV(1, 3) = + TM*(AA+SP)
DERIV(2, 3) = -SP *(AA+TM)

C
DERIV(1, 4) = + TT
DERIV(2, 4) = +SP*T2

C
AA = +SP+TP- 1.5D0
DERIV(1, 5) = + TP*(AA+SP)
DERIV(2, 5) = +SP *(AA+TP)

C
DERIV(1, 6) = +S2*TP
DERIV(2, 6) = +SS

C
AA = +SM+TP- 1.5D0
DERIV(1, 7) = - TP*(AA+SM)
DERIV(2, 7) = +SM *(AA+TP)

C
DERIV(1, 8) = - TT
DERIV(2, 8) = +SM*T2

C
C***
ELEMENTOS DE 9 NOS
ELSE IF (NNODE.EQ.9) THEN
    SSM = (S*S-S)/2.0D0
    SSP = (S*S+S)/2.0D0
    SS0 = -S*S+1.0D0
    TTM = (T*T-T)/2.0D0
    TTP = (T*T+T)/2.0D0
    TT0 = -T*T+1.0D0
    DSM = -0.5D0+S
    DSP = +0.5D0+S
    DS0 = -2.0D0*S
    DTM = -0.5D0+T
    DTP = +0.5D0+T
    DT0 = -2.0D0*T

C
DERIV(1, 1) = DSM*TTM
DERIV(2, 1) = SSM*DTM

C
DERIV(1, 2) = DS0*TTM
DERIV(2, 2) = SS0*DTM

C
DERIV(1, 3) = DSP*TTM
DERIV(2, 3) = SSP*DTM

C
DERIV(1, 4) = DSP*TT0
DERIV(2, 4) = SSP*DT0

C

```

```

DERIV(2, 9) = SS0*DT0;
% ERRO
else
    fprintf('resultados,...
    [\n\nSTOP na subrotina DERIV2 - ',...
    'elemento finito com %d nos',NNODE);
    ERRMSG(1,RNAME)
end

```

```

C
DERIV(1, 5) = DSP*TTP
DERIV(2, 5) = SSP*DTP

C
DERIV(1, 6) = DS0*TTP
DERIV(2, 6) = SS0*DTP

C
DERIV(1, 7) = DSM*TTP
DERIV(2, 7) = SSM*DTP

C
DERIV(1, 8) = DSM*TT0
DERIV(2, 8) = SSM*DT0

C
DERIV(1, 9) = DS0*TT0
DERIV(2, 9) = SS0*DT0

C
C*** ERRO
C
ELSE
    WRITE (6,910) NNODE
    CALL ERRMSG (RNAME, 1)
END IF
910 FORMAT('/', '** STOP na subrotina DERIV2 - elemento finito com ',I5,
    ' nos')
C
RETURN
END

```

```

%*** ERRMSG.m *****
%
% ESCRVE MENSAGEM DE ERRO E/OU TERMINA EXECUCAO
%
%*** 2010.06.30 * LR ***** quasar_m ****
function ERRMSG(NTASK,RNAME)

switch NTASK
case 0
    disp('A preparar visualização de resultados')
case 1
    error(['STOP - Erro na subrotina ',RNAME])
end

```

```

SUBROUTINE ERRMSG (RNAME, NTASK)
C*****
C*** ESCRVE MENSAGEM DE ERRO E/OU TERMINA EXECUCAO
C
C*** 2010.06.11 * JCM *****
C      CHARACTER RNAME*15
C
C      SELECT CASE (NTASK)
C      CASE (0)
C          WRITE (*,*) ' *** STOP - Fim de execucao do programa
',RNAME
C          PAUSE
C          STOP
C          ' *** Para terminar tecle ENTER'
C          ' *** STOP'
C      CASE (1)
C          WRITE (*,*) ' *** STOP - Erro na subrotina ',RNAME
C          PAUSE
C          STOP
C          ' *** Para terminar tecle ENTER'
C          ' *** STOP'
C      END SELECT
C
C      END

```

```

%*** FLOWTP.m *****
%
%*** IDENTIFICA O TIPO DE ESCOAMENTO (confinado ou nao confinado) *
%
%**** 2010.06.30 * LR ***** quasar_m ****
function CONFN=FLOWTP(HNODL,COORD,NPOIN,resultados)

% DETERMINACAO DO TIPO DE ESCOAMENTO
KOUNT = 0;
for IPOIN = 1:NPOIN
    HVALU=HNODL(IPOIN);
    if HVALU < COORD(2,IPOIN)
        KOUNT = KOUNT + 1;
    end
end

if KOUNT == 0
    CONFN = true;
    disp(' ***** ESCOAMENTO CONFINADO ')
else
    CONFN = false;
    fprintf('ESCOAM.NAO CONFINADO - H<z em %d nos\n',KOUNT);
    fprintf(resultados,...
        '\n\n ***** ESCOAM.NAO CONFINADO - H<z em %d nos\n',KOUNT);
end

```

```

C***** SUBROUTINE FLOWTP (COORD, NDIME, NPOIN, HNODL, CONFN) *****
C
C*** IDENTIFICA O TIPO DE ESCOAMENTO (confinado ou nao confinado)
C
C**** 2010.06.24 * JCM ***** quasar *
C***** IMPLICIT DOUBLE PRECISION (A-H, O-Z) *****
C***** LOGICAL CONFN *****
C***** DIMENSION COORD(NDIME,NPOIN), HNODL(NPOIN) *****
C
C*** DETERMINACAO DO TIPO DE ESCOAMENTO
C
C***** KOUNT = 0
C***** DO IPOIN = 1,NPOIN
C*****     HVALU = HNODL(IPOIN)
C*****     IF(HVALU.LT.COORD(2,IPOIN)) KOUNT = KOUNT + 1
C***** END DO
C
C***** IF (KOUNT.EQ.0) THEN
C*****     CONFN = .TRUE.
C*****     WRITE(*,*) ' ***** ESCOAMENTO CONFINADO '
C***** ELSE
C*****     CONFN = .FALSE.
C*****     WRITE(*,*) ' ***** ESCOAM.NAO CONFINADO - H<z em ',KOUNT,' nos'
C*****     WRITE(6,900) KOUNT
C***** ENDIF
C***** 900 FORMAT(///' ***** ESCOAM.NAO CONFINADO - H<z em ',I5,' nos')
C***** RETURN
C***** END

```

```

%*** GAUSLG.m *****
%
%*** DEFINE AS CONSTANTES DE INTEGRACAO DE GAUSS-LEGENBRE *
%
%***** 2010.06.30 * LR ***** quasar_m ****
%
% 2 PONTOS DE GAUSS
if NGAUS == 2
    POSGP(1) = -0.57735026918962576451;
    POSGP(2) = +0.57735026918962576451;
    WEIGP(1) = +1.00000000000000000000;
    WEIGP(2) = +1.00000000000000000000;
%
% 3 PONTOS DE GAUSS
elseif NGAUS == 3
    POSGP(1) = -0.77459666924148337704;
    POSGP(2) = +0.00000000000000000000;
    POSGP(3) = +0.77459666924148337704;
    WEIGP(1) = +0.55555555555555555556;
    WEIGP(2) = +0.88888888888888888889;
    WEIGP(3) = +0.55555555555555555556;
end

```

```

SUBROUTINE GAUSLG (NGAUS, POSGP, WEIGP)
C*****
C
C*** DEFINE AS CONSTANTES DE INTEGRACAO DE GAUSS-LEGENBRE
C
C***** 2002.11.11 * JCM *****
C
C*** IMPLICIT DOUBLE PRECISION (A-H, O-Z),
C      DIMENSION POSGP(NGAUS), WEIGP(NGAUS)
C
C*** 2 PONTOS DE GAUSS
C
C      IF (NGAUS.EQ.2) THEN
C          POSGP(1) = -0.57735026918962576451D0
C          POSGP(2) = +0.57735026918962576451D0
C          WEIGP(1) = +1.0000000000000000000D0
C          WEIGP(2) = +1.0000000000000000000D0
C
C*** 3 PONTOS DE GAUSS
C
C      ELSE IF (NGAUS.EQ.3) THEN
C          POSGP(1) = -0.77459666924148337704D0
C          POSGP(2) = +0.0000000000000000000D0
C          POSGP(3) = +0.77459666924148337704D0
C          WEIGP(1) = +0.55555555555555555556D0
C          WEIGP(2) = +0.88888888888888888888D0
C          WEIGP(3) = +0.55555555555555555556D0
C
C      END IF
C
C      RETURN
C      END

```

```

function count = espacos(fid,s,n)
for i = 1 : n
count=fprintf(fid,s);
end

function count = estrelas(fid,n,linha)
count=fprintf(fid,' ');
for i = 1 : n
count=fprintf(fid,'*');
if linha && i == n
count=fprintf(fid,'\n');
end
end

```

```

C***** SUBROUTINE GAUSS (A, B, N, X) *****
C
C*** EFECTUA A REDUCAO E RETROSUBSTITUICAO DE GAUSS
C * MATRIZ DOS COEFICIENTES COM ARMAZENAMENTO COMPLETO
C
C*** 2010.06.11 * JCM *****
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C CHARACTER RNAME*5
C DIMENSION A(N,N), B(N), X(N)
C DATA RNAME/'GAUSS'/
C.....
C*** CICLO SOBRE AS EQUACOES
C
C DO L = 1,N
C
C C*** VERIFICACAO DO PIVOT
C
C PIVOT = A(L,L)
C IF(PIVOT.LT.1.D-10) THEN
C WRITE (6,900) PIVOT,L
C CALL ERRMSG (RNAME, 1)
C END IF
C 900 FORMAT(/5X,' *** STOP na subrotina GAUSS : PIVOT =',E12.3,
C , ' na equacao no.',I7)
C
C C*** CICLOS SOBRE AS LINHAS E COLUNAS
C
C DO I = L+1,N
C FACTR = A(I,L)/PIVOT
C IF(FACTR.NE.0.0D0) THEN
C DO J = L+1,N
C A(I,J) = A(I,J) - FACTR*A(L,J)
C END DO
C B(I) = B(I) - FACTR*B(L)
C END IF
C END DO
C
C.....
C*** CICLO POR ORDEM INVERSA SOBRE AS EQUACOES REDUZIDAS
C
C DO I = N,1,-1
C PIVOT = A(I,I)
C XI = B(I)
C DO J = I+1,N
C XI = XI - A(I,J)*X(J)
C END DO
C X(I) = XI/PIVOT
C END DO
C
C RETURN
C END

```

```

%*** GPCORD.m *****
%
%*** CALCULA AS COORDENADAS DOS PONTOS DE GAUSS
%
%**** 2010.06.30 * LR ***** quasar_m ****
GPCOD=zeros(NDIME,NTOTG);

% CICLO SOBRE OS ELEMENTOS
KGAUS = 0;
for IELEM = 1:NELEM

    % COORDENADAS DOS NOS DO ELEMENTO
    NODES=LNODS(1:NODE,IELEM);
    ELCOD=COORD(:,NODES);

    % CICLOS SOBRE OS PONTOS DE GAUSS
    KGASP = 0;
    for IGAUS = 1:NGAUS
        EXISP = POSGP(IGAUS);
        for JGAUS = 1:NGAUS
            KGAUS = KGAUS+1;
            KGASP = KGASP+1;
            ETASP = POSGP(JGAUS);

            % CALCULO DAS COORDENADAS DO PONTO DE GAUSS
            SHAPE=SHAPE2(EXISP,ETASP,NNODE,resultados);
            GPCOD(1,KGAUS)=sum(SHAPE'.*ELCOD(1,:),2);
            GPCOD(2,KGAUS)=sum(SHAPE'.*ELCOD(2,:),2);

        end
    end
end
end

```

```

SUBROUTINE GPCORD (COORD, GPCOD, LNODS, NELEM, NNODE, NPOIN,
                  NTOTG, NDIME, NGAUS, POSGP)
C*****
C
C*** CALCULA AS COORDENADAS DOS PONTOS DE GAUSS
C
C**** 2010.06.11 * JCM *****
C
C*** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
C*** DIMENSION LNODS(NNODE,NELEM), COORD(NDIME,NPOIN),
C
C*** GPCOD(NDIME,NTOTG), POSGP(NGAUS)
C
C*** DIMENSION ELCOD(2,9), SHAPE(9)
C
C*** CICLO SOBRE OS ELEMENTOS
C
C*** KGAUS = 0
C
C*** DO IELEM = 1, NELEM
C
C*** COORDENADAS DOS NOS DO ELEMENTO
C
C*** DO INODE = 1, NNODE
C
C*** IPOIN = LNODS(INODE,IELEM)
C
C*** DO IDIME = 1, NDIME
C
C*** ELCOD(IDIME,INODE) = COORD(IDIME,IPOIN)
C
C*** END DO
C
C*** END DO
C
C*** CICLOS SOBRE OS PONTOS DE GAUSS
C
C*** KGASP = 0
C
C*** DO IGAUS = 1, NGAUS
C
C*** EXISP = POSGP(IGAUS)
C
C*** DO JGAUS = 1, NGAUS
C
C*** KGAUS = KGAUS+1
C
C*** KGASP = KGASP+1
C
C*** ETASP = POSGP(JGAUS)
C
C*** CALCULO DAS COORDENADAS DO PONTO DE GAUSS
C
C*** CALL SHAPE2 (ETASP, EXISP, NNODE, SHAPE)
C
C*** DO IDIME = 1, NDIME
C
C*** GPCOD = 0.0D0
C
C*** DO INODE = 1, NNODE
C
C*** GPCOD = GPCOD + ELCOD(IDIME,INODE)*SHAPE(INODE)
C
C*** END DO
C
C*** GPCOD(IDIME,KGAUS) = GPCOD
C
C*** END DO
C
C*** END DO
C
C*** END DO
C
C*** RETURN
C
C*** END

```

```

%*** JACOB2.m *****
%
%*** CALCULA E INVERTE A MATRIZ JACOBIANA E AS DERIVADAS CARTESIANAS
%
%
%
%***** 2010.06.30 * LR ***** quasar_m ****
function [CARTD,DJACB]=...
    JACOB2(EXISP,ETASP,NDIME,NNODE,ELCOD,KGASP,IELEM,resultados)
CARTD=zeros(2,NNODE);
XJACI=zeros(2,2);
RNAME='JACOB2';
DERIV=DERIV2(EXISP,ETASP,NNODE);
XJACM(1,:)= [sum(DERIV(1,:).*ELCOD(1,:)),sum(DERIV(1,:).*ELCOD(2,:))];
XJACM(2,:)= [sum(DERIV(2,:).*ELCOD(1,:)),sum(DERIV(2,:).*ELCOD(2,:))];
% CALCULO DO DETERMINANTE E INVERSA DA MATRIZ JACOBIANA
DJACB = det(XJACM);
if DJACB <= 0
    fprintf(resultados,['\n\nSTOP na subrotina JACOB2 - ',...
        'o ponto de Gauss no. %d ',KGASP]);
    fprintf(resultados,'do elemento %d tem area %10.3E',IELEM,DJACB);
    ERRMSG(1,RNAME)
end
XJACI=inv(XJACM);
CARTD=XJACI*DERIV;

```

```

SUBROUTINE JACOB2 (CARTD, DJACB, ELCOD, KGASP, IELEM, ETASP,
    EXISP, NNODE, NDIME)
C*****
C
C*** CALCULA E INVERTE A MATRIZ JACOBIANA E AS DERIVADAS CARTESIANAS
C
C
C***** 2010.06.11 * JCM *****
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
CHARACTER RNAME*6
DIMENSION CARTD(2,9), XJACI(2,2), DERIV(2,9),
    ELCOD(2,9), XJACM(2,2)
.
    DATA RNAME/'JACOB2'/
C
C*** CALCULO DA MATRIZ JACOBIANA
C
C
CALL DERIV2 (DERIV, ETASP, EXISP, NNODE)
DO IDIME = 1,NDIME
    DO JDIME = 1,NDIME
        XJACO = 0.0D0
        DO INODE = 1,NNODE
            XJACO = XJACO + DERIV(IDIME,INODE)*ELCOD(JDIME,INODE)
        END DO
        XJACM(IDIME,JDIME) = XJACO
    END DO
END DO
C
C*** CALCULO DO DETERMINANTE E INVERSA DA MATRIZ JACOBIANA
C
C
DJACB = XJACM(1,1)*XJACM(2,2)-XJACM(1,2)*XJACM(2,1)
C
IF (DJACB.LE.0.0D0) THEN
    WRITE (6,900) KGASP,IELEM,DJACB
    CALL ERRMSG (RNAME, 1)
END IF
900 FORMAT(/' *** STOP na subrotina JACOB2 - o ponto de Gauss no. ',I2,
    ' do elemento ',I7,' tem area ',E10.3)
C
XJACI(1,1) = +XJACM(2,2)/DJACB
XJACI(2,2) = +XJACM(1,1)/DJACB
XJACI(1,2) = -XJACM(1,2)/DJACB
XJACI(2,1) = -XJACM(2,1)/DJACB
C
C*** CALCULO DAS DERIVADAS CARTESIANAS
C
C
DO IDIME = 1,NDIME
    DO INODE = 1,NNODE
        DCART = 0.0D0
        DO JDIME = 1,NDIME
            DCART = DCART + XJACI(IDIME,JDIME)*DERIV(JDIME,INODE)
        END DO
        CARTD(IDIME,INODE) = DCART
    END DO
END DO

```

C
RETURN
END

```

*** NCONF1.m *****
%
*** TRATA ESCOAMENTOS NAO CONFINADOS PELO METODO DA PERMEABILIDADE
% VARIAVEL
%
%***** 2010.06.30 * LR ***** quasar_m ***

RNAME='NCONF1';

% VERIFICAÇÃO DO TIPO DE ESCOAMENTO
CONFN=FLOWTP(HNODL,COORD,NPOIN,resultados);
if CONFN
    return
end

% CICLOS RELATIVOS ÀS ITERAÇÕES
CONVR = false;
KREU = 0;
KITER = 0;
while ~CONVR
    KITER = KITER + 1;
    % CALCULO DA VELOCIDADE E DOS CAUDAIS NAO EQUILIBRADOS
    QVELOC

    % INTRODUCAO DAS CONDICÖES FRONTEIRA (DE DIRICHLET)
    DPFIX=zeros(NPOIN,1);
    fseek(gstif,0,'bof');
    GSTIF=zeros(NPOIN);
    for IPOIN=1:NPOIN
        GSTIF(:,IPOIN)=fread(gstif,NPOIN,'double');
    end
    [GSTIF,GLOAD]=BDCOND(GSTIF,NTOTV,GLOAD,IPFIX,DPFIX);

    % RESOLUCAO DO SISTEMA DE EQUACOES
    DELTH=GSTIF\GLOAD;
    DELTH=GAUSS(GSTIF,GLOAD,NPOIN)';

    %
    % VERIFICACAO DA CONVERGENCIA
    PCONVR

    % NOS COM CONDICÖES-FRONTEIRA REDEFINIVEIS (REVIEW BOUND.CONDITIONS)
    if NREU ~= 0 && CONVR
        IREU = KREU;
        while CONVR && IREU <= KREU
            IREU = IREU+1;
            if IREU > NREU
                fprintf(resultados,...
                    ['\n\n *** STOP na subrotina NCONF1',...
                    '- a lista de %d nos',NREU]);
                fprintf(resultados,...
                    [' com condicoes-fronreira redefiniveis',...
                    'e demasiado curta']);
                ERRMSG(1,RNAME)
            else
                IPOIN = NDRVU(IREU);
                fprintf('\nHNODL(%d)=%E',IPOIN,HNODL(IPOIN));
                fprintf(resultados,'\nHNODL(%d)=%E',IPOIN,HNODL(IPOIN));
                fprintf('\nCORD(%d)=%E\n',IPOIN,COORD(2,IPOIN));
            end
        end
    end
end

```

```

SUBROUTINE NCONF1 (MATNO, PROPS, COORD, NDIME, LNODS, GLOAD,
. NMATS, NPROP, NELEM, NNODE, NPOIN, GPCOD,
. POSGP, WEIGP, NGAUS, NTYPE, VELOC, GRADH,
. HNODL, DELTH, NTOTG, DPFIX, NREU, FPERM,
. GSTIF, NTOTV, NDRVU, IPFIX, TOLCV)
C*****
C
C*** TRATA ESCOAMENTOS NAO CONFINADOS PELO METODO DA PERMEABILIDADE
C VARIAVEL
C
C*** 2010.06.17 * JCM ***** quasar *
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C LOGICAL CONVR, CONFN
C CHARACTER RNAME*6
C DIMENSION MATNO(NELEM), GLOAD(NPOIN), PROPS(NPROP,NMATS),
. POSGP(NGAUS), NDRVU(NREU), LNODS(NNODE,NELEM),
. WEIGP(NGAUS), DPFIX(NPOIN), COORD(NDIME,NPOIN),
. IPFIX(NPOIN), HNODL(NPOIN), GPCOD(NDIME,NTOTG),
. DELTH(NPOIN), GSTIF(NTOTV,NTOTG),
. VELOC(NDIME,NTOTG),
. GRADH(NDIME,NTOTG)
C
C DATA RNAME/'NCONF1'/
C
C*** VERIFICACAO DO TIPO DE ESCOAMENTO
C
C CALL FLOWTP (COORD, NDIME, NPOIN, HNODL, CONFN)
C IF(CONFN) RETURN
C
C CICLOS RELATIVOS AS ITERACOES
C
CONVR = .FALSE.
KREU = 0
KITER = 0
DO WHILE (.NOT.CONVR)
    KITER = KITER + 1

C
C*** CALCULO DA VELOCIDADE E DOS CAUDAIS NAO EQUILIBRADOS
C
C CALL QVELOC (MATNO, PROPS, COORD, NDIME, LNODS, HNODL,
. NMATS, NPROP, NGAUS, NELEM, NNODE, NPOIN,
. POSGP, WEIGP, NTYPE, VELOC, NTOTG, GRADH,
. GPCOD, GLOAD, NTOTV, FPERM)
C
C*** INTRODUCAO DAS CONDICÖES FRONTEIRA (DE DIRICHLET)
C
DPFIX = 0.0D0
REWIND 22
READ (22) GSTIF
CALL BDCOND (GSTIF, NTOTV, GLOAD, IPFIX, DPFIX)

C*** RESOLUCAO DO SISTEMA DE EQUACOES
C
CALL GAUSS (GSTIF, GLOAD, NPOIN, DELTH)
C

```

```

fprintf(resultados, '\nCORD(%d)=%E', IPOIN, COORD(2, IPOIN));
disp(' =====> Tecle ENTER para continuar')
pause
if HNODL(IPOIN) > COORD(2, IPOIN)
    KREVU = KREVU+1;
    IPFIX(IPOIN) = 1;
    HNODL(IPOIN) = COORD(2, IPOIN);
    CONVR = false;
end
end
end

```

```

C** VERIFICACAO DA CONVERGENCIA
C
    CALL PCONVR (HNODL, DELTH, NPOIN, KITER, CONVR, TOLCV,
    .            COORD, NDIME)
C
C** NOS COM CONDICAOES-FRONTIEIRA REDEFINIVEIS (REVIEW BOUND.CONDITIONS)
C
    IF(NREVU.NE.0.AND.CONVR) THEN
        IREVU = KREVU
        DO WHILE (CONVR.AND.IREVU.LE.KREVU)
            IREVU = IREVU+1
            IF(IREVU.GT.NREVU) THEN
                WRITE (6,900) NREVU
                CALL ERRMSG (RNAME, 1)
            ELSE
                IPOIN = NDRVU(IREVU)
                WRITE(*,*) 'HNODL(', IPOIN, ')=' , HNODL(IPOIN)
                WRITE(6,*) 'HNODL(', IPOIN, ')=' , HNODL(IPOIN)
                WRITE(*,*) 'YCORD(', IPOIN, ')=' , COORD(2, IPOIN)
                WRITE(6,*) 'YCORD(', IPOIN, ')=' , COORD(2, IPOIN)
                PAUSE ' =====> Tecle ENTER para continuar'
                IF(HNODL(IPOIN).GT.COORD(2, IPOIN)) THEN
                    KREVU = KREVU+1
                    IPFIX(IPOIN) = 1
                    HNODL(IPOIN) = COORD(2, IPOIN)
                    CONVR = .FALSE.
                END IF
            END IF
        END DO
    END IF
END DO
FORMAT(/' *** STOP na subrotina NCONFI - a lista de 'I2,' nos',
.      ' com condicoes-fronteira redefiniveis e demasiado curta')
C
RETURN
END
900

```

```

%*** NODE2D.m *****
%
%*** GERA AS COORDENADAS DE:
% * NOS SITUADOS A MEIO DOS LADOS RECTILINEOS DE ELEMENTOS PLANOS
% DE 8 E 9 NOS
% * NO CENTRAL DE ELEMENTOS DE 9 NOS
%
%***** 2010.06.30 * LR ***** quasar_m ****
function COORD=NODE2D(COORD, LNODS, NDIIME, NPOIN, NELEM, NNODE)

% CICLO SOBRE OS ELEMENTOS
for IELEM = 1:NELEM

    NODES=LNODS(1:NNODE, IELEM);

    % NOS SITUADOS A MEIO DE LADOS RECTILINEOS (elementos de 8 e 9 nos)
    for INOD2 = 2:2:8
        IPOI2 = NODES(INOD2);
        TOTAL = abs(COORD(1,IPOI2))+abs(COORD(2,IPOI2));
        if TOTAL == inf
            INOD1 = INOD2-1;
            INOD3 = INOD2+1;
            if INOD3 == 9
                INOD3 = 1;
            end
            IPOI1 = NODES(INOD1);
            IPOI3 = NODES(INOD3);
            for I = 1:NDIIME
                COORD(I, IPOI2)=(COORD(I, IPOI1)+COORD(I, IPOI3))*.5;
            end
        end
    end

    % NO CENTRAL (elementos de 9 nos)
    if NNODE == 9
        IPOI9 = NODES(9);
        TOTAL = abs(COORD(1,IPOI9))+abs(COORD(2,IPOI9));
        if TOTAL == inf
            COORD(1,IPOI9) = 0;
            COORD(2,IPOI9) = 0;
            for INODE = 1:2:7
                NCORN = NODES(INODE);
                NMIDS = NODES(INODE+1);
                for IDIME = 1:NDIIME
                    COORD(IDIME, IPOI9) = COORD(IDIME, IPOI9)...
                        - COORD(IDIME, NCORN)/4 + COORD(IDIME, NMIDS)/2;
                end
            end
        end
    end
end
end

```

```

SUBROUTINE NODE2D (COORD, LNODS, NDIIME, NPOIN, NELEM, NNODE)
*****
C*****
C
C*** GERA AS COORDENADAS DE:
C * NOS SITUADOS A MEIO DOS LADOS RECTILINEOS DE ELEMENTOS PLANOS
C DE 8 E 9 NOS
C * NO CENTRAL DE ELEMENTOS DE 9 NOS
C
C***** 2010.06.11 * JCM *****
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C DIMENSION COORD(NDIIME,NPOIN), LNODS(NNODE,NELEM)
C DIMENSION NODES(9)
C DATA BIG/ 1.0D30/
C
C CICLO SOBRE OS ELEMENTOS
C***
C
C DO IELEM = 1,NELEM
C DO INODE = 1,NNODE
C   NODES(INODE) = LNODS(INODE, IELEM)
C   END DO
C
C NOS SITUADOS A MEIO DE LADOS RECTILINEOS (elementos de 8 e 9 nos)
C
C DO INOD2 = 2,8,2
C   IPOI2 = NODES(INOD2)
C   TOTAL = ABS(COORD(1,IPOI2))+ABS(COORD(2,IPOI2))
C   IF(TOTAL.GT.BIG) THEN
C     INOD1 = INOD2-1
C     INOD3 = INOD2+1
C     IF(INOD3.EQ.9) INOD3 = 1
C     IPOI1 = NODES(INOD1)
C     IPOI3 = NODES(INOD3)
C     DO I = 1,NDIIME
C       COORD(I, IPOI2)=(COORD(I, IPOI1)+COORD(I, IPOI3))*.5D0
C     END DO
C   END IF
C END DO
C
C NO CENTRAL (elementos de 9 nos)
C***
C
C IF (NNODE.EQ.9) THEN
C   IPOI9 = NODES(9)
C   TOTAL = ABS(COORD(1,IPOI9))+ABS(COORD(2,IPOI9))
C   IF(TOTAL.GT.BIG) THEN
C     COORD(1,IPOI9) = 0.0D0
C     COORD(2,IPOI9) = 0.0D0
C     DO INODE = 1,7,2
C       NCORN = NODES(INODE)
C       NMIDS = NODES(INODE+1)
C       DO IDIME = 1,NDIIME
C         COORD(IDIME, IPOI9) = COORD(IDIME, NCORN)/4.D0
C         COORD(IDIME, IPOI9) + COORD(IDIME, NMIDS)/2.D0
C       END DO
C     END DO
C   END IF
C END IF

```

```
END DO
END IF
END IF
END DO
RETURN
END
```

C

```

%*** PBOUND.m *****
%
%*** LE AS CONDICÕES-FRONTIEIRA P/ A SOLUCAO EM TERMOS DO POTENCIAL *
%
%**** 2010.06.30 * LR ***** quasar_m ****
%
% INICIALIZACAO
IPFIX=zeros(NPOIN,1);
PFIxD=zeros(NPOIN,1);

% CONDICÕES-FRONTIEIRA DE DIRICHLET
fprintf('resultados,...
        \n\n\n CONDICÕES-FRONTIEIRA DE DIRICHLET (DO POTENCIAL) EM %d NOS',...
        NPFIx);
fprintf('resultados,...
        \n\n\n No      Valor\n ');espacos(resultados,'=',19);
for IVFIx = 1:NPFIx
    IPOIN=fscanf(dados,'%d',1);
    VALUE=fscanf(dados,'%f',1);
    fprintf('resultados,\n%5d%15.5f',IPOIN,VALUE);
    IPFIx(IPOIN) = 1;
    PFIxD(IPOIN) = VALUE;
end

```

```

C***** SUBROUTINE PBOUND (NPFIx, IPFIx, PFIxD, NPOIN) *****
C
C*** LE AS CONDICÕES-FRONTIEIRA P/ A SOLUCAO EM TERMOS DO POTENCIAL
C
C**** 2010.06.11 * JCM ***** quasar *
C
C***** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C***** DIMENSION IPFIx(NPOIN), PFIxD(NPOIN)
C
C*** INICIALIZACAO
C
C      IPFIx = 0
C      PFIxD = 0.000
C
C*** CONDICÕES-FRONTIEIRA DE DIRICHLET
C
C      WRITE (6,900) NPFIx
C      DO IVFIx = 1,NPFIx
C      READ (5,* ) IPOIN,VALUE
C      WRITE (6,905) IPOIN,VALUE
C      IPFIx(IPOIN) = 1
C      PFIxD(IPOIN) = VALUE
C
C      END DO
900  FORMAT(///' *** CONDICÕES-FRONTIEIRA DE DIRICHLET (DO POTENCIAL) '
      .      ' EM ',I3,' NOS',
      .      ///' No      Valor',/1X,19('='))
905  FORMAT(I5,F15.5)
C
C      RETURN
C      END

```

```

**** PCONVR.m *****
%
%*** VERIFICA A CONVERGENCIA DO PROCESSO ITERATIVO
%
%**** 2010.06.30 * LR ***** quasar_m ****

DNORM = 0;
HNORM = 0;
HWNRM = 0;
DHMAX = 0;
for IPOIN = 1:NPOIN
    DELTA = DELTH(IPOIN);
    HNEXT = HNODL(IPOIN);
    HNEW = HNEXT+DELTA;
    HNODL(IPOIN) = HNEXT;
    DNORM = DNORM+DELTA*DELTA;
    HNORM = HNORM+HNEXT*HNEXT;
    HWNRM = HWNRM+HNEW*HNEW;
    if abs(DELTA) > abs(DHMAX)
        DHMAX = DELTA;
        IPMAX = IPOIN;
    end
end

HNORM = sqrt(DNORM/HNORM);
HWNRM = sqrt(DNORM/HWNRM);
fprintf('resultados,\n *** Iteracao- %d HNORM = %10.3E',KITER,HNORM);
fprintf('resultados,\n HWNRM = %10.3E *** DHMAX = %10.3E - NO %d',...
        HWNRM,DHMAX,IPMAX);
fprintf('\n *** Iteracao- %d HNORM = %10.3E',KITER,HNORM);
fprintf(' HWNRM = %10.3E *** DHMAX = %10.3E - NO %d',HWNRM,DHMAX,IPMAX);

if HWNRM < TOLCV
    CONVR = true;
end

```

```

SUBROUTINE PCONVR (HNODL, DELTH, NPOIN, KITER, CONVR, TOLCV,
COORD, NDIME)
C*****
C*** VERIFICA A CONVERGENCIA DO PROCESSO ITERATIVO
C
C**** 2010.06.24 * JCM *****
C
C*****
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
LOGICAL CONVR
DIMENSION DELTH(NPOIN), HNODL(NPOIN), COORD(NDIME,NPOIN)

C
DNORM = 0.0D0
HNORM = 0.0D0
HWNRM = 0.0D0
DHMAX = 0.0D0
DO IPOIN = 1,NPOIN
    DELTA = DELTH(IPOIN)
    HNEXT = HNODL(IPOIN)
    HNEW = HNEXT+DELTA
    HNODL(IPOIN) = HNEXT
    DNORM = DNORM+DELTA*DELTA
    HNORM = HNORM+HNEXT*HNEXT
    HWNRM = HWNRM+HNEW*HNEW
    IF(ABS(DELTA).GT.ABS(DHMAX)) THEN
        DHMAX = DELTA
        IPMAX = IPOIN
    ENDIF
END DO

C
HNORM = DSQRT(DNORM/HNORM)
HWNRM = DSQRT(DNORM/HWNRM)
WRITE (6,900) KITER,HNORM,HWNRM,DHMAX,IPMAX
900 WRITE (*,900) KITER,HNORM,HWNRM,DHMAX,IPMAX
FORMAT(' *** Iteracao- ',I5,' HNORM = ',1PE10.3,' HWNRM = ',E10.3,
. ' *** DHMAX = ',E10.3,' - NO ',I5)
C IF(HWNRM.LT.TOLCV) CONVR = .TRUE.
C
C
RETURN
END

```

```

%*** PLOAD0.m *****
%
%*** CALCULA TERMO INDEPENDENTE P/ A SOLUCAO EM TERMOS DO POTENCIAL *
%
%**** 2010.06.30 * LR ***** quasar_m ****

GLOAD=zeros(IPOIN,1);
QEDGE=zeros(3,1);
DERIV=zeros(2,9);
DGAUS=zeros(2,1);
QSUMS=zeros(2,1);
RNAME='PLOAD0';
TWOPI=6.283185307179586;

% TITULO
fseek(dados,2,'cof');
TITLE=fgetl(dados);
fprintf(resultados,'\n\n\n');estrelas(resultados,78,1);
fprintf(resultados,'%s\n',TITLE);estrelas(resultados,78,1);

% PARAMETROS DEFINIDORES DOS TIPOS DE CAUDAL
IQNOD=fscanf(dados,'%d',1);
IQEDG=fscanf(dados,'%d',1);
IQVOL=fscanf(dados,'%d',1);
IWRIT=fscanf(dados,'%d',1);

fprintf(resultados,'\n\n *** PARAMETROS DE CONTROLE');
fprintf(resultados,...
    '\n Caudal concentrado nodal (IQNOD)=%5d',IQNOD);
    '\n Caudal distrib. em lados de elementos (IQEDG)=%5d',IQEDG);
    '\n Caudal distribuido no volume (IQVOL)=%5d',IQVOL);
    '\n Flag de escrita do termo independente (IWRIT)=%5d',IWRIT);

% CAUDAIS CONCENTRADOS NODAIS
if IQNOD ~= 0
    NQNOD=fscanf(dados,'%d',1);
    fprintf(resultados,'\n\n\n *** CAUDAIS CONCENTRADOS NODAIS');
    fprintf(resultados,'\n No.de nos com caudal pontual =%4d',NQNOD);
    fprintf(resultados,'\n No Q(m3/s)\n ');
    espacos(resultados,'=',23);
    for JQNOD = 1:NQNOD
        IPOIN=fscanf(dados,'%g',1);
        QNODL=fscanf(dados,'%g',1);
        fprintf(resultados,'\n %5d %14.6E',IPOIN,QNODL);
        if NTYPE == 2 && COORD(1,IPOIN) ~= 0
            QNODL = QNODL*TWOPI*COORD(1,IPOIN);
        end
        GLOAD(IPOIN) = GLOAD(IPOIN) + QNODL;
    end
end

% CAUDAIS DISTRIBUIDOS APLICADOS EM LADOS DE ELEMENTOS
if IQEDG ~= 0
    NEDGE=fscanf(dados,'%d',1);
    fprintf(resultados,'\n\n\n *** CAUDAIS DISTRIBUIDOS LATERAIS');

```

```

SUBROUTINE PLOAD0 (COORD, LNODS, MATNO, NELEM, NNODE, NGAUS,
. NPOIN, NTYPE, POSGP, PROPS, NMATS, WEIGP,
. NPROP, NDIME, GLOAD, GPCOD, NTOIG, QSUMS,
. CONFN)
C*****
C
C*** CALCULA TERMO INDEPENDENTE P/ A SOLUCAO EM TERMOS DO POTENCIAL
C
C**** 2010.06.24 * JCM ***** quasar *
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
CHARACTER TITLE*78
LOGICAL CONFN
DIMENSION MATNO(NELEM), LNODS(NNODE,NELEM),
. GLOAD(NPOIN), COORD(NDIME,NPOIN),
. POSGP(NGAUS), PROPS(NPROP,NMATS),
. WEIGP(NGAUS), GPCOD(NDIME,NTOTG)
DIMENSION CARTD(2,9), NODES(9), QEDGE(3), DERIV(2,9),
. ELCOB(2,9), SHAPE(9), DGAUS(2), QSUMS(2)
DATA TWOPI/ 6.283185307179586 D0/
C
C*** INICIALIZACAO
C
C GLOAD = 0.0D0
C QSUMS = 0.0D0
C
C*** TITULO
C
READ (5,'(A)') TITLE
WRITE (6,1010) TITLE
FORMAT(/1X,78('*'),/A78,/1X,78('*'))
1010
C
C*** PARAMETROS DEFINIDORES DOS TIPOS DE CAUDAL
C
READ (5,*) IQNOD,IQEDG,IQVOL,IWRIT
WRITE (6,1030) IQNOD,IQEDG,IQVOL,IWRIT
FORMAT(/1 *** PARAMETROS DE CONTROLE',
. //' Caudal concentrado nodal (IQNOD)=' ,15,
. //' Caudal distrib. em lados de elementos (IQEDG)=' ,15,
. //' Caudal distribuido no volume (IQVOL)=' ,15,
. //' Flag de escrita do termo independente (IWRIT)=' ,15)
C
C*** CAUDAIS CONCENTRADOS NODAIS
C
IF (IQNOD.NE.0) THEN
    READ (5,*) NQNOD
    WRITE (6,1100) NQNOD
    WRITE (6,1110)
    DO JQNOD = 1,NQNOD
        READ (5,*) IPOIN,QNODL
        WRITE (6,1130) IPOIN,QNODL
        IF (NTYPE.EQ.2.AND.COORD(1,IPOIN).NE.0.0D0)
            QNODL = QNODL*TWOPI*COORD(1,IPOIN)
        GLOAD(IPOIN) = GLOAD(IPOIN)+QNODL
    END DO
END

```

```

fprintf(resultados,...
'\n No.de lados com caudal prescrito =%4d',NEDGE);
fprintf(resultados,...
'\n\n Elem. No Caudal(m3/s)\n ');
espacos(resultados,'=',35);

NDEG = 3;
if NNODE == 4
    NDEG = 2;
end
NODES=zeros(NDEG,1);
for JEDGE = 1:NDEG
    JELEM=fscanf(dados,'%d',1);
    for J=1:NDEG
        NODES(J)=fscanf(dados,'%d',1);
        QEDGE(J)=fscanf(dados,'%e',1);
    end
    fprintf(resultados,...
        '\n %5d %5d %4.6E',JELEM,NODES(1),QEDGE(1));
    for J=2:NDEG
        fprintf(resultados,...
            '\n %5d %5d %4.6E',NODES(J),QEDGE(J));
    end
end

% COORDENADAS DOS NOS DO LADO CARREGADO
ELCOD = COORD(:,NODES);

% INTEGRACAO NUMERICA AO LONGO DO LADO DO ELEMENTO
ETASP = -1;
for IGAUS = 1:NGAUS
    EXISP = POSGP(IGAUS);
    SHAPE=SHAPE2(EXISP,ETASP,NNODE,resultados);
    DERIV=DERIV2(EXISP,ETASP,NNODE);

    DGAUS(1)=sum(ELCOD(1,:).*DERIV(1,1:NDEG));
    DGAUS(2)=sum(ELCOD(2,:).*DERIV(1,1:NDEG));

    DJACB = sqrt(DGAUS(1)*DGAUS(1)+DGAUS(2)*DGAUS(2));
    DAREA = DJACB*WEIGP(IGAUS);

    if NTYPE == 2
        RADUS = sum(SHAPE(1:NDEG).*ELCOD(1,:));
        DAREA = DAREA*TWOPI*RADUS;
    end

    QGAUS = sum(QEDGE.*SHAPE(1:NDEG));

    % CALCULO E ESPALHAMENTO DOS CAUDAIS NODAIS EQUIVALENTES
    GLOAD(NODES) = GLOAD(NODES) + QGAUS*SHAPE(1:NDEG)*DAREA;

end
end

% CAUDAL DISTRIBUIDO NO VOLUME
if IQVOL ~= 0

    % CICLO SOBRE OS ELEMENTOS
    KGAUS = 0;

```

```

1100 END IF
    FORMAT('' *** CAUDAIS CONCENTRADOS NODAIS',
        /' No.de nos com caudal pontual =',I4)
1110 FORMAT( /' No',12X,'Q(m3/s)',/3X,23('='))
1130 FORMAT(2X,I5,5X,1PE14.6)
C
C*** CAUDAIS DISTRIBUIDOS APLICADOS EM LADOS DE ELEMENTOS
C
IF (IQEDG.NE.0) THEN
    READ (5,*) NEDGE
    WRITE (6,1310) NEDGE
1310 FORMAT('' *** CAUDAIS DISTRIBUIDOS LATERAIS',
        /' No.de lados com caudal prescrito =',I4,
        /' Elem. No Caudal(m3/s)',
        /5X,35('='))
C
C
NDEG = 3
IF (NNODE.EQ.4) NDEG = 2
DO JEDGE = 1,NEDGE
    READ (5,*) JELEM,(NODES(J),QEDGE(J),J=1,NDEG)
    WRITE (6,1320) JELEM,(NODES(J),QEDGE(J),J=1,NDEG)
1320 FORMAT(4X,I5,5X,I5,7X,1PE14.6,2(/14X,I5,7X,E14.6))
C
C*** COORDENADAS DOS NOS DO LADO CARREGADO
C
DO INODE = 1,NDEG
    IPOIN = NODES(INODE)
    DO IDIME = 1,NDIME
        ELCOD(IDIME,INODE) = COORD(IDIME,IPOIN)
    END DO
END DO
C
C*** INTEGRACAO NUMERICA AO LONGO DO LADO DO ELEMENTO
C
ETASP = -1.000
DO IGAUS = 1,NGAUS
    EXISP = POSGP(IGAUS)
    CALL SHAPE2 (ETASP, EXISP, NNODE, SHAPE)
    CALL DERIV2 (DERIV, ETASP, EXISP, NNODE)
    DO IDIME = 1,NDIME
        DGAUS(IDIME) = 0.000
    DO INODE = 1,NDEG
        DGAUS(IDIME) =
            DGAUS(IDIME) + ELCOD(IDIME,INODE)*DERIV(1,INODE)
    END DO
END DO
DJACB = DSQRT(DGAUS(1)*DGAUS(1)+DGAUS(2)*DGAUS(2))
DAREA = DJACB*WEIGP(IGAUS)
IF (NTYPE.EQ.2) THEN
    RADUS = 0.000
    DO INODE = 1,NDEG
        RADUS = RADUS+SHAPE(INODE)*ELCOD(1,INODE)
    END DO
    DAREA = DAREA*TWOPI*RADUS

```

```

for IELEM = 1:NELEM
    IMATS = MATNO(IELEM);
    QVALU = PROPS(4,IMATS);
    if QVALU == 0
        KGAS = KGAS+NGAS*NGAS;
    else
        % COORDENADAS DOS NOS DO ELEMENTO
        NODES = LNDS(1:NODE,IELEM);
        ELCOD = COORD(:,NODES);

        % CICLOS SOBRE OS PONTOS DE GAUSS
        KGASP = 0;
        for IGAUS = 1:NGAS
            EXISP = POSGP(IGAUS);
            for JGAUS = 1:NGAS
                ETASP = POSGP(JGAUS);
                KGAS = KGAS+1;
                KGASP = KGASP+1;
                [CARTD,DJACB]=...
                    JACOB2(EXISP,ETASP,2,NNODE,ELCOD,...
                        KGASP,IELEM,resultados);
                DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS);
                if NTYPE == 2
                    DVOLU = DVOLU*TWOPi*GPCOD(1,KGAS);
                end
            end
        end
    end
end
end

%CALCULO E ESPALHAMENTO DOS CAUDAIS NODAIS
%EQUIVALENTES
SHAPE=SHAPE2(EXISP,ETASP,NNODE,resultados);
GLOAD(NODES)=GLOAD(NODES)+QVALU*SHAPE(NODES)*DVOLU;

end
end
end

% ESCRITA DO TERMO INDEPENDENTE
if IWRT == 1
    fprintf(resultados,'\n\n\n *** TERMO INDEPENDENTE\n ');
    espacos(resultados, '=', ' ', 83);
    k=0;
    for IPOIN=1:NPOIN
        k=k+1;
        if k==1
            fprintf(resultados, '\n%5d%11.3E ', IPOIN, GLOAD(IPOIN));
        else
            fprintf(resultados, '%5d%11.3E ', IPOIN, GLOAD(IPOIN));
        end
        if k==5
            k=0;
        end
    end
end

% DETERMINACAO DOS CAUDAIS TOTAIS PRESCRITOS POSITIVOS E NEGATIVOS
QSUNS(1)=sum(GLOAD(GLOAD>0));
QSUNS(2)=sum(GLOAD(GLOAD<0));

```

```

C
END IF
QGAUS = 0.0D0
DO INODE = 1, NODEG
    QGAUS = QGAUS + QEDGE(INODE)*SHAPE(INODE)
END DO

C*** CALCULO E ESPALHAMENTO DOS CAUDAIS NODAIS EQUIVALENTES
C
DO INODE = 1, NODEG
    IPOIN = NODES(INODE)
    GLOAD(IPOIN) =
        GLOAD(IPOIN) + QGAUS*SHAPE(INODE)*DAREA
    END DO
END DO
END IF
END IF

C*** CAUDAL DISTRIBUIDO NO VOLUME
C
IF(IQVOL.NE.0) THEN
C*** CICLO SOBRE OS ELEMENTOS
C
    KGAS = 0
    DO IELEM = 1, NELEM
        IMATS = MATNO(IELEM)
        QVALU = PROPS(4,IMATS)
        IF(QVALU.EQ.0.0D0) THEN
            KGAS = KGAS+NGAS*NGAS
        ELSE
            COORDENADAS DOS NOS DO ELEMENTO
            DO INODE = 1, NNODE
                IPOIN = LNDS(INODE,IELEM)
                NODES(INODE) = IPOIN
                DO IDIME = 1,NDIME
                    ELCOD(IDIME,INODE) = COORD(IDIME,IPOIN)
                END DO
            END DO
        END DO
    END DO
    KGASP = 0
    DO IGAUS = 1,NGAS
        EXISP = POSGP(IGAUS)
        DO JGAUS = 1,NGAS
            ETASP = POSGP(JGAUS)
            KGAS = KGAS+1
            KGASP = KGASP+1
        END DO
        CALL JACOB2
            (CARTD, DJACB, ELCOD, KGASP, IELEM, ETASP,
            EXISP, NNODE, NDIME)
    END DO
END DO

```

```
% ESCRITA DO GLOAD INICIAL P/USO NO CASO DE PERCOLACAO NAO CONFINADA
if ~CONFN
    gload=fopen('GLOAD','w+');
    fwrite(gload,GLOAD,'double');
end
```

```

        DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
        IF(NTYPE.EQ.2) DVOLU = DVOLU*TWOPI*GPCOD(1,KGAUS)

C***  CALCULO E ESPALHAMENTO DOS CAUDAIS NODAIS EQUIVALENTES
C
        CALL SHAPE2
           (ETASP, EXISP, NNODE, SHAPE)
        DO INODE = 1,NNODE
            IPOIN = NODES(INODE)
            GLOAD(IPOIN) =
                GLOAD(IPOIN) + QVALU*SHAPE(INODE)*DVOLU
        END DO
    END DO
END DO
END IF
END DO
END IF
END IF

C
C***  ESCRITA DO TERMO INDEPENDENTE
C
        IF(IWRIT.EQ.1) THEN
            WRITE (6,1800)
            WRITE (6,1802) (IPOIN,GLOAD(IPOIN),IPOIN=1,NPOIN)
        END IF
1800  FORMAT('///' ***)
1802  FORMAT(1P,5(15,E11.3,1X))

C
C***  DETERMINACAO DOS CAUDAIS TOTAIS PRESCRITOS POSITIVOS E NEGATIVOS
C
        QPLUS = 0.000
        QMINS = 0.000
        DO IPOIN = 1,NPOIN
            GLODI = GLOAD(IPOIN)
            IF(GLODI.GT.0.000) QPLUS = QPLUS+GLODI
            IF(GLODI.LT.0.000) QMINS = QMINS+GLODI
        END DO
        QSUMS(1) = QPLUS
        QSUMS(2) = QMINS

C
C***  ESCRITA DO GLOAD INICIAL P/USO NO CASO DE PERCOLACAO NAO CONFINADA
C
        IF(.NOT.CONFN) THEN
            OPEN (20, FORM='UNFORMATTED', DISPOSE='DELETE')
            WRITE (20) GLOAD
        END IF

C
        RETURN
    END
```

```

**** PSTIF0.m *****
%
%*** FORMULACAO EM TERMOS DO POTENCIAL NODAL H
%
%*** CALCULA A MATRIZ DE RIGIDEZ "ESTIF" DE CADA ELEMENTO E PROCEDE
% AO RESPECTIVO ESPALHAMENTO NA MATRIZ DE RIGIDEZ GLOBAL "GSTIF"
%
%***** 2010.06.30 * LR ***** quasar_m ****

PERMB=zeros(2,2);
NODES=zeros(NNODE,1);
TWOP1=6.283185307179586;

% INICIALIZACAO DA MATRIZ DE RIGIDEZ GLOBAL
GSTIF=zeros(NTOTV);

% CICLO SOBRE OS ELEMENTOS
KGAUS = 0;
for IELEM = 1:NELEM
    ESTIF=zeros(NEVAB);

    % CARACTERISTICAS GEOMETRICAS E MATERIAIS DO ELEMENTO
    IMATS = MATNO(IELEM);
    PERMB(1,1) = PROPS(1,IMATS);
    PERMB(2,2) = PROPS(2,IMATS);
    THETA = PROPS(3,IMATS);

    % DETERMINACAO DA MATRIZ DE PERMEABILIDADE EM RELACAO A XY
    if PERMB(1,1) ~= PERMB(2,2) && THETA ~= 0
        T=[cosd(THETA) -sind(THETA);sind(THETA) cosd(THETA)];
        PERMB=T*PERMB*T';
    end

    % NO. DOS GDL E COORDENADAS DOS NOS DO ELEMENTO
    NODES=LNODS(1:NNODE,IELEM);
    ELCOD=COORD(:,NODES);

    % CICLOS SOBRE OS PONTOS DE GAUSS
    KGASP = 0;
    for IGAUS = 1:NGAUS
        EXISP = POSGP(IGAUS);
        for JGAUS = 1:NGAUS
            KGAUS = KGAUS+1;
            KGASP = KGASP+1;
            ETASP = POSGP(JGAUS);

            % DETERMINACAO DAS DERIVADAS CARTESIANAS DAS FUNCOES DE FORMA E
            % DO ELEMENTO DE VOLUME
            [CARDT,DJACB]=JACOB2(EXISP,ETASP,...
                NDIME,NNODE,ELCOD,KGASP,IELEM,resultados);
            DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS);
            if NTYPE == 2
                DVOLU = DVOLU*TWOP1*GPCOD(1,KGAUS);
            end

            % CALCULO DA MATRIZ DE RIGIDEZ DO ELEMENTO
            ESTIF=ESTIF+CARTD'*PERMB*CARTD*DVOLU;
        end
    end
end

```

```

SUBROUTINE PSTIF0 (MATNO, PROPS, COORD, NDIME, LNODS, GSTIF,
.      IMATS, NPROP, NTOTV, NELEM, NNODE, IPOIN,
.      POSGP, WEIGP, NGAUS, NTYPE, ESTIF, NEVAB,
.      GPCOD, NTOTG, CONFN)
C*****
C
C*** FORMULACAO EM TERMOS DO POTENCIAL NODAL H
C
C*** CALCULA A MATRIZ DE RIGIDEZ "ESTIF" DE CADA ELEMENTO E PROCEDE
C AO RESPECTIVO ESPALHAMENTO NA MATRIZ DE RIGIDEZ GLOBAL "GSTIF"
C
C***** 2010.06.24 * JCM ***** quasar *
C
C*** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C LOGICAL CONFN
C DIMENSION MATNO(NELEM), PROPS(NPROP,NMATS), GPCOD(NDIME,NTOTG),
.      POSGP(NGAUS), LNODS(NNODE,NELEM), GSTIF(NTOTV,NTOTV),
.      WEIGP(NGAUS), COORD(NDIME,NPOIN), ESTIF(NEVAB,NEVAB)
C DIMENSION ELCOD(2,9), CARTD(2,9), PERMB(2,2), NODES(9)
C DATA TWOP1/ 6.283185307179586 D0/
C
C*** INICIALIZACAO DA MATRIZ DE RIGIDEZ GLOBAL
C
C*** GSTIF = 0.0D0
C
C*** CICLO SOBRE OS ELEMENTOS
C
C*** KGAUS = 0
C DO IELEM = 1,NELEM
C
C*** INICIALIZACAO DA MATRIZ DE RIGIDEZ DO ELEMENTO
C
C*** ESTIF = 0.0D0
C
C*** CARACTERISTICAS GEOMETRICAS E MATERIAIS DO ELEMENTO
C
C*** IMATS = MATNO(IELEM)
C PERMB(1,1) = PROPS(1,IMATS)
C PERMB(2,2) = PROPS(2,IMATS)
C PERMB(1,2) = 0.0D0
C THETA = PROPS(3,IMATS)
C
C*** DETERMINACAO DA MATRIZ DE PERMEABILIDADE EM RELACAO A XY
C
C*** IF (PERMB(1,1).NE.PERMB(2,2).AND.THETA.NE.0.0D0)
C CALL RTMAT2 (PERMB, THETA)
C
C*** No. DOS GDL E COORDENADAS DOS NOS DO ELEMENTO
C
C*** DO INODE = 1,NNODE
C IPOIN = LNODS(INODE,IELEM)
C NODES(INODE) = IPOIN
C DO IDIME = 1,NDIME
C ELCOD(IDIME,INODE) = COORD(IDIME,IPOIN)
C END DO

```

```

end
% ESPALHAMENTO DA MATRIZ DE RIGIDEZ DO ELEMENTO
GSTIF(NODES,NODES)=GSTIF(NODES,NODES)+ESTIF;
end
% ESCRITA DO GSTIF INICIAL P/USO NO CASO DE PERCOLACAO NAO CONFINADA
if ~CONFN
    gstif=fopen('GSTIF','w+');
    fwrite(gstif,GSTIF,'double');
end

```

```

C
C***
C
END DO
CICLOS SOBRE OS PONTOS DE GAUSS
C
    KGASP = 0
    DO IGAUS = 1,NGAUS
        EXISP = POSGP(IGAUS)
        DO JGAUS = 1,NGAUS
            KGAUS = KGAUS+1
            KGASP = KGASP+1
            ETASP = POSGP(JGAUS)
        .
    C*** DETERMINACAO DAS DERIVADAS CARTESIANAS DAS FUNCOES DE FORMA E
    C DO ELEMENTO DE VOLUME
    C
        CALL JACOB2 (CARTD, DJACB, ELCOD, KGASP, IELEM, ETASP,
        .
        DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
        IF(NTYPE.EQ.2) DVOLU = DVOLU*TWOPi*GPCOD(1,KGAUS)
    C
C*** CALCULO DA MATRIZ DE RIGIDEZ DO ELEMENTO
C
    DO JNODE = 1,NNODE
        FACT1 = PERMB(1,1)*CARTD(1,JNODE)
        FACT2 = PERMB(2,2)*CARTD(2,JNODE)
        IF(PERMB(1,2).NE.0.0D0) THEN
            FACT1 = FACT1+PERMB(1,2)*CARTD(2,JNODE)
            FACT2 = FACT2+PERMB(1,2)*CARTD(1,JNODE)
        ENDIF
        DO INODE = 1,NNODE
            DNIDX = CARTD(1,INODE)
            DNIDY = CARTD(2,INODE)
            ESTIF(INODE,JNODE) = ESTIF(INODE,JNODE)
            + (DNIDX*FACT1 + DNIDY*FACT2)*DVOLU
        .
        END DO
    END DO
    END DO
END DO
C
C***
C
    ESPALHAMENTO DA MATRIZ DE RIGIDEZ DO ELEMENTO
    C
        DO JNODE = 1,NNODE
            JPOIN = NODES(JNODE)
            DO INODE = 1,NNODE
                IPOIN = NODES(INODE)
                GSTIF(IPOIN,JPOIN) =
                .
                GSTIF(IPOIN,JPOIN) + ESTIF(INODE,JNODE)
            END DO
        END DO
    END DO
END DO
C
C***
C
    ESCRITA DO GSTIF INICIAL P/USO NO CASO DE PERCOLACAO NAO CONFINADA
    IF(.NOT.CONFN) THEN

```

```
OPEN (22, FORM='UNFORMATTED',DISPOSE='DELETE')  
WRITE (22) GSTIF  
END IF  
RETURN  
END
```

C

```

%*** QCHEK1.m *****
%
% VERIFICA OS PARAMETROS DE CONTROLO
%
%**** 2010.06.30 * LR ***** quasar_m ****
function QCHEK1(NELEM,NPOIN,MNOD,MMATS,NVFIX,NTYPE,NGAUS,resultados)
RNAME='QCHEK1';
fprintf(resultados,'\n\n\n *** DIAGNOSTICO DA SUBROTINA "QCHEK1" :\n');
NERRO = 0;
if NELEM <= 0
    NERRO = 1;
end
if NPOIN <= 0
    NERRO = 1;
end
if NELEM*MNOD < NPOIN
    NERRO = 1;
end
if MMATS < 1
    NERRO = 1;
end
if NVFIX < 1 || NVFIX > NPOIN
    NERRO = 1;
end
if NTYPE < 1 || NTYPE > 2
    NERRO = 1;
end
if NGAUS < 2 || NGAUS > 3
    NERRO = 1;
end
if NERRO == 0
    fprintf(resultados,...
        ' *** Nao foram detectados erros nos parametros de controle\n');
else
    ERRMSG(1,RNAME)
end

```

```

SUBROUTINE QCHEK1 (NELEM, NPOIN, MMATS, NPFIX, MNODE, NTYPE,
    NGAUS, MELEM, MPOIN, MMATS, MNODE)
C*****
C** VERIFICA OS PARAMETROS DE CONTROLE
C
C**** 2010.06.11 * JCM ***** quasar *
CHARACTER RNAME*6
DATA RNAME/'QCHEK1'/
WRITE (6,800)
FORMAT(// ' *** DIAGNOSTICO DA SUBROTINA "QCHEK1" :')
800
C
    NERRO = 0
    IF (NELEM.LE.0) NERRO = 1
    IF (NPOIN.LE.0) NERRO = 1
    IF (NELEM*MNODE.LT.NPOIN) NERRO = 1
    IF (MMATS.LT.1.OR.MMATS.GT.NELEM) NERRO = 1
    IF (NPFIX.LT.1.OR.NPFIX.GT.NPOIN) NERRO = 1
    IF (NTYPE.LT.1.OR.NTYPE.GT.2) NERRO = 1
    IF (NGAUS.LT.2.OR.NGAUS.GT.3) NERRO = 1
C
    IF (NERRO.EQ.1) WRITE (6,810)
    FORMAT(' *** Ha erros nos parametros de controle !')
810
C
    JERRO = 0
    IF (NELEM.GT.MELEM) THEN
        WRITE (6,910) NELEM,MELEM
        JERRO = JERRO+1
    END IF
    FORMAT(' *** NELEM (= ',I7,' ) > MELEM (= ',I7,' )')
910
C
    IF (NPOIN.GT.MPOIN) THEN
        WRITE (6,920) NPOIN,MPOIN
        JERRO = JERRO+1
    END IF
    FORMAT(' *** NPOIN (= ',I7,' ) > MPOIN (= ',I7,' )')
920
C
    IF (MMATS.GT.MMATS) THEN
        WRITE (6,940) MMATS,MMATS
        JERRO = JERRO+1
    END IF
    FORMAT(' *** MMATS (= ',I7,' ) > MMATS (= ',I7,' )')
940
C
    IF (MNODE.GT.MNODE) THEN
        WRITE (6,950) MNODE,MNODE
        JERRO = JERRO+1
    END IF
    FORMAT(' *** MNODE (= ',I7,' ) > MNODE (= ',I7,' )')
950
C
    IF (JERRO.NE.0) WRITE (6,960) JERRO
    FORMAT(' *** Foram excedidas ',I3,' dimensoes maximas !')
960
C
    IF (NERRO.EQ.0.AND.JERRO.EQ.0) THEN
        WRITE (6,970)
    END IF

```

```
ELSE
  CALL ERRMSG (RNAME, 1)
END IF
FORMAT(' *** Nao foram detectados erros nos parametros de',
.      ' controle')
C
RETURN
END
```

```

**** QCHEK2.m *****
%
%*** VERIFICA A TOPOLOGIA DA MALHA
%
%**** 2010.06.30 * LR ***** quasar_m ****

function COORD=QCHEK2(NPOIN,COORD,MATNO,NMATS,NNODE,LNODS,NELEM,resultados)
RNAME='QCHEK2';
IERRO = 0;
fprintf(resultados,'\n\n *** DIAGNOSTICO DA SUBROTINA "QCHEK2" :');

% VERIFICA SE HA NOS DIFERENTES COM COORDENADAS IDENTICAS
for IPOIN = 2:NPOIN
    for JPOIN = 1:IPOIN-1
        if COORD(1,IPOIN) == COORD(1,JPOIN);
            if COORD(2,IPOIN) == COORD(2,JPOIN);
                IERRO = 1;
                fprintf(resultados,...
                    '\n *** Os nos %d e %d tem coordenadas identicas',...
                    IPOIN,JPOIN);
            end
        end
    end
end

% VERIFICA O NUMERO DO MATERIAL DE CADA ELEMENTO
for IELEM = 1:NELEM
    NUMAT = MATNO(IELEM);
    if NUMAT <= 0 || NUMAT > NMATS
        IERRO = 1;
        fprintf(resultados,...
            '\n *** 0 no.do material do elemento %d esta incorrecto',...
            IELEM);
    end
end

% VERIFICA OS NUMEROS DAS LIGACOES NODAIS DOS ELEMENTOS
for IELEM = 1:NELEM
    JERRO = 0;
    for INODE = 1:NNODE
        IPOIN = LNODS(INODE,IELEM);
        if IPOIN <= 0 || IPOIN > NPOIN
            JERRO = JERRO+1;
        end
    end
    if JERRO ~= 0
        IERRO = 1;
        fprintf(resultados,...
            '\n *** 0 elemento %d tem %d erros na lista de nos',...
            IELEM,JERRO);
    end
end

% DETECTA REPETICAO DE NOS NUM ELEMENTO
for IELEM = 1:NELEM
    for INODE = 1:NNODE-1

```

```

SUBROUTINE QCHEK2 (COORD, LNODS, MATNO, NELEM, NPOIN, NMATS,
    NNODE, NDIME)
C*****
C*** VERIFICA A TOPOLOGIA DA MALHA
C
C**** 2010.06.11 * JCM *****
C*****
C**** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C*****
C***** CHARACTER RNAME*6
C***** DIMENSION MATNO(NELEM), COORD(NDIME,NPOIN), LNODS(NNODE,NELEM)
C***** DATA RNAME/'QCHEK2'/
C
C IERRO = 0
C WRITE (6,800)
C 800 FORMAT(// ' *** DIAGNOSTICO DA SUBROTINA "QCHEK2" : ' )
C
C*** VERIFICA SE HA NOS DIFERENTES COM COORDENADAS IDENTICAS
C
C DO IPOIN = 2,NPOIN
C J_LOOP: DO JPOIN = 1,IPOIN-1
C DO IDIME = 1,NDIME
C IF(COORD(IDIME,IPOIN).NE.COORD(IDIME,JPOIN)) CYCLE J_LOOP
C END DO
C IERRO = 1
C WRITE (6,910) IPOIN,JPOIN
C 910 END DO J_LOOP
C END DO
C FORMAT(' *** Os nos ',I7,' e ',I7,' tem coordenadas identicas')
C
C*** VERIFICA O NUMERO DO MATERIAL DE CADA ELEMENTO
C
C DO IELEM = 1,NELEM
C NUMAT = MATNO(IELEM)
C IF(NUMAT.LE.0.OR.NUMAT.GT.NMATS) THEN
C IERRO = 1
C WRITE (6,920) IELEM
C END IF
C END DO
C 920 FORMAT(' *** 0 no.do material do elemento ',I7,' esta incorrecto')
C
C*** VERIFICA OS NUMEROS DAS LIGACOES NODAIS DOS ELEMENTOS
C
C DO IELEM = 1,NELEM
C JERRO = 0
C DO INODE = 1,NNODE
C IPOIN = LNODS(INODE,IELEM)
C IF(IPOIN.LE.0.OR.IPOIN.GT.NPOIN) JERRO = JERRO+1
C END DO
C IF(JERRO.NE.0) THEN
C IERRO = 1
C WRITE (6,930) IELEM,JERRO
C END IF
C END DO
C 930 FORMAT(' *** 0 elemento ',I7,' tem ',I3,' erros na lista de nos')

```

```

IPOIN = LNODS(INODE,IELEM);
for JNODE = INODE+1:NNODE
    if IPOIN == LNODS(JNODE,IELEM)
        IERRO = 1;
        fprintf('resultados,...\n *** 0 no %d esta repetido no elemento %d',...,
            IPOIN, IELEM);
    end
end
end

% VERIFICA SE CADA NO APARECE EM PELO MENOS UM ELEMENTO
for IPOIN = 1:NPOIN
    k = any(IPOIN == LNODS);
    if k == 0
        IERRO = 1;
        fprintf('resultados,...\n *** 0 no %d nao aparece em nenhum elemento',IPOIN);
    end
end

if IERRO == 0
    fprintf('resultados,...\n *** Nao foram detectados erros na topologia da malha');
else
    ERRMSG(1,RNAME)
end
end

```

```

C
C*** DETECTA REPETICAO DE NOS NUM ELEMENTO
C
DO IELEM = 1,NELEM
    I_LOOP: DO INODE = 1,NNODE-1
        IPOIN = LNODS(INODE,IELEM)
        DO JNODE = INODE+1,NNODE
            IF(IPOIN.EQ.LNODS(JNODE,IELEM)) THEN
                IERRO = 1
                WRITE (6,940) IPOIN,IELEM
                CYCLE I_LOOP
            END IF
        END DO
    END DO I_LOOP
END DO
940 FORMAT(' *** 0 no ',I7,' esta repetido no elemento ',I7)
C
C*** VERIFICA SE CADA NO APARECE EM PELO MENOS UM ELEMENTO
C
IP_LOOP: DO IPOIN = 1,NPOIN
    DO IELEM = 1,NELEM
        DO INODE = 1,NNODE
            JPOIN = LNODS(INODE,IELEM)
            IF(JPOIN.EQ.IPOIN) CYCLE IP_LOOP
        END DO
    END DO
    IERRO = 1
    WRITE (6,950) IPOIN
END DO IP_LOOP
950 FORMAT(' *** 0 no ',I7,' nao aparece em nenhum elemento')
C
IF(IERRO.EQ.0) THEN
    WRITE (6,1000)
ELSE
    CALL ERRMSG (RNAME, 1)
END IF
1000 FORMAT(' *** Nao foram detectados erros na topologia da malha')
C
RETURN
END

```

```

**** GMa lha.m *****
%
% LE OS PARAMETROS DE CONTROLE, VERIFICA AS DIMENSOES DO PROBLEMA
%
%**** 2010.06.30 * LR ***** quasar_m ****
% TITULO DO PROBLEMA

estrelas(resultado,78,1);fprintf(resultado,'\n');
for i=1:3
    tline=fgetl(dados);
    fprintf(resultado,' %s\n',tline);
end
estrelas(resultado,78,1);

% PARAMETROS DE CONTROLE
NELEM=fscanf(dados,'%d',1);
NPOIN=fscanf(dados,'%d',1);
NMATS=fscanf(dados,'%d',1);
NPFI=fscanf(dados,'%d',1);
NNODE=fscanf(dados,'%d',1);
NTYPE=fscanf(dados,'%d',1);
NGAUS=fscanf(dados,'%d',1);
KONFN=fscanf(dados,'%d',1);

NEVAB = NNODE*NDOFN;
NTOTV = NPOIN*NDOFN;
NTOTG = NELEM*NGAUS*NGAUS;
KONFN = true;
if KONFN == 0
    KONFN = false;
end

fprintf(resultado,...
'\n\n *** PARAMETROS DE CONTROLE\n');
fprintf(resultado,...
'\n\n No. total de elementos (NELEM) = %d', NELEM);
fprintf(resultado,...
'\n\n No. total de nos (NPOIN) = %d', NPOIN);
fprintf(resultado,...
'\n\n No. de materiais (NMATS) = %d', NMATS);
fprintf(resultado,...
'\n\n No. de nos com carga total fixa (NPFI) = %d', NPFI);
fprintf(resultado,...
'\n\n No. de nos/elemento (NNODE) = %d', NNODE);
fprintf(resultado,...
'\n\n Tipo de problema (NTYPE) = %d', NTYPE);
fprintf(resultado,...
'\n\n No de pontos de integracao (NGAUS) = %d', NGAUS);
fprintf(resultado,...
'\n\n Problema confinado/nao confinado (KONFN) = %d', KONFN);
fprintf(resultado,...
'\n\n No. de graus de liberdade/no (NDOFN) = %d', NDOFN);
fprintf(resultado,...
'\n\n No. de coordenadas/no (NDIME) = %d', NDIME);
fprintf(resultado,...
'\n\n No. de propriedades/material (NPROP) = %d', NPROP);

```

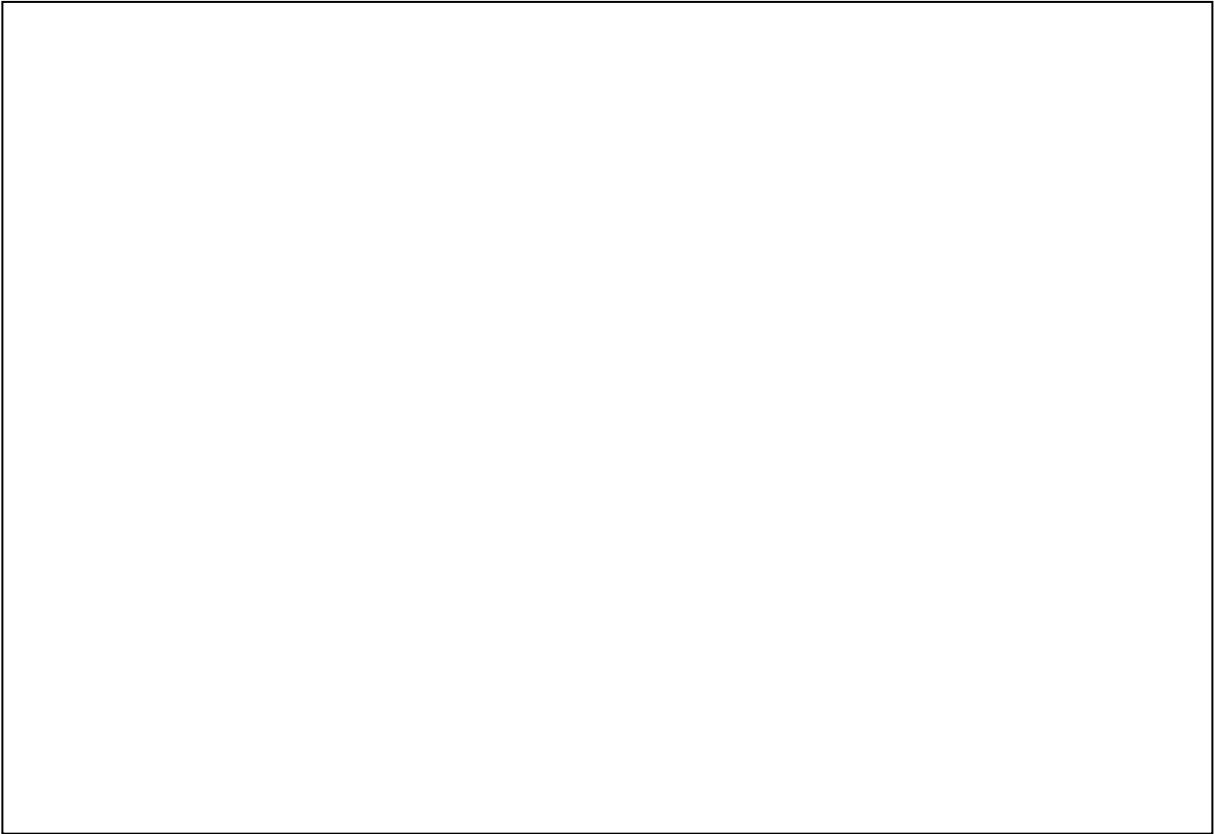
```

SUBROUTINE QINPT1 (NELEM, NPOIN, NMATS, NPFI, NTYPE, NGAUS,
NPROP, NDOFN, NDIME, NNODE, NTOTG, NEVAB,
NTOTV, MPOIN, MELEM, MMATS, MNODE, CONFN)
C*****
C
C*** LE OS PARAMETROS DE CONTROLE, VERIFICA AS DIMENSOES DO PROBLEMA
C
C**** 2010.06.11 * JCM ***** quasar *
C
CHARACTER TITLE(3)*78
LOGICAL CONFN
C
C*** TITULO DO PROBLEMA
C
WRITE (6, '(1X,78(''***''))')
READ (5, '(A)') (TITLE(I), I=1,3)
WRITE (6, '(1X,A)')(TITLE(I), I=1,3)
WRITE (6, '(1X,78(''***''))')
C
C*** PARAMETROS DE CONTROLE
C
READ (5,*) NELEM,NPOIN,NMATS,NPFI,NNODE,NTYPE,NGAUS,KONFN
C
NEVAB = NNODE*NDOFN
NTOTV = NPOIN*NDOFN
NTOTG = NELEM*NGAUS*NGAUS
KONFN = .TRUE.
IF (KONFN.EQ.0) KONFN = .FALSE.
C
WRITE (6,901) NELEM,NPOIN,NMATS,NPFI,NNODE,NTYPE,NGAUS,KONFN,
NDOFN,NDIME, NPROP,NEVAB,NTOTV,NTOTG
901 FORMAT(//' *** PARAMETROS DE CONTROLE',
//' No. total de elementos (NELEM) =',I7,
//' No. total de nos (NPOIN) =',I7,
//' No. de materiais (NMATS) =',I7,
//' No. de nos com carga total fixa (NPFI) =',I7,
//' No. de nos/elemento (NNODE) =',I7,
//' Tipo de problema (NTYPE) =',I7,
//' No de pontos de integracao (NGAUS) =',I7,
//' Problema confinado/nao confinado (KONFN) =',I7,
//' No. de graus de liberdade/no (NDOFN) =',I7,
//' No. de coordenadas/no (NDIME) =',I7,
//' No. de propriedades/material (NPROP) =',I7,
//' Max. no. de variaveis/elemento (NEVAB) =',I7,
//' No. total de variaveis (NTOTV) =',I7,
//' No. total de pontos de Gauss (NTOTG) =',I7)
C
C*** VERIFICACAO DOS DADOS
C
CALL QCHEK1 (NELEM, NPOIN, NMATS, NPFI, NNODE, NTYPE,
NGAUS, MELEM, MPOIN, MMATS, MNODE)
C
RETURN
END

```

```
fprintf(resultados,...
'\n Max. no. de variaveis/elemento (NEVAB) = %d' ,NEVAB);
fprintf(resultados,...
'\n No. total de variaveis (NTOTV) = %d' ,NTOTV);
fprintf(resultados,...
'\n No. total de pontos de Gauss (NTOTG) = %d' ,NTOTG);

% VERIFICACAO DOS DADOS
QCHECK1(NELEM,NPOIN,NNODE,NMATS,NPFX,NTRY,NGAUS,resultados)
```



```

**** QINPT2.m *****
%
% LE A TOPOLOGIA DA MALHA E AS CARACTERISTICAS DOS MATERIAIS
%***** 2010.06.30 * LR ***** quasar_m ****

MATNO=zeros(NELEM,1); LNODS=zeros(NNODE,NELEM);
POSGP=zeros(NGAUS,1); COORD=ones(NDIME,NPOIN);
WEIGP=zeros(NGAUS,1); PROPS=zeros(NPROP,NMATS);
RDIAN=57.295779513082320;

% LEITURA DAS LIGACOES NODAIS DOS ELEMENTOS E DO NUMERO DO MATERIAL
fprintf('resultados,\\n\\n ***** CARACTERISTICAS DOS ELEMENTOS\\n');
fprintf('resultados,\\n\\n Elem. Mater. No.nos Nos\\n');
for IELEM = 1 : NELEM
    J=fscanf(dados,'%d',1);
    MATNO(J)=fscanf(dados,'%d',1);
    LTYPE=fscanf(dados,'%d',1);
    NNODE=fscanf(dados,'%d',1);
    for I=1:NNODE
        LNODS(I,J)=fscanf(dados,'%d',1);
    end
    nonod(J) = NNODE;
    fprintf('resultados,\\n\\n%d\\n',J);
    fprintf('resultados,\\n\\n%d\\n',MATNO(J),NNODE);
    for I=1:NNODE
        fprintf('resultados,\\n\\n%d\\n',LNODS(I,J));
    end
end

% LEITURA (E INTERPOLACAO) DAS COORDENADAS DOS NOS
COORD=inf*COORD;

if NTYPE ~= 2
    fprintf('resultados,\\n\\n\\n ***** COORDENADAS NODAIS\\n');
    fprintf('resultados,\\n\\n No X(m) Y(m)\\n');
    espacos('resultados,\\n',17);
    fprintf('resultados,\\n No X(m) Y(m)\\n');
    fprintf('resultados,\\n ');espacos('resultados,\\n',29);
    espacos('resultados,\\n',15);espacos('resultados,\\n',29);
end
if NTYPE == 2
    fprintf('resultados,\\n\\n\\n ***** COORDENADAS NODAIS\\n');
    fprintf('resultados,\\n\\n No X(m) Z(m)\\n');
    espacos('resultados,\\n',17);
    fprintf('resultados,\\n No X(m) Z(m)\\n');
    fprintf('resultados,\\n ');espacos('resultados,\\n',29);
    espacos('resultados,\\n',15);espacos('resultados,\\n',29);
end

for KOUNT = 1:NPOIN
    IPOIN=fscanf(dados,'%d',1);
    for IDIME=1:NDIME

```

```

SUBROUTINE QINPT2 (NELEM, NPOIN, NMATS, NTYPE, COORD, NDIME,
. NNODE, NGAUS, LNODS, MATNO, NPROP, PROPS,
. POSGP, WEIGP)
C*****
C
C*** LE A TOPOLOGIA DA MALHA E AS CARACTERISTICAS DOS MATERIAIS
C
C*** 2010.06.11 * JCM ***** quasar *
C
C*** IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
C*** DIMENSION MATNO(NELEM), LNODS(NNODE,NELEM),
. POSGP(NGAUS), COORD(NDIME,NPOIN),
. WEIGP(NGAUS), PROPS(NPROP,NMATS)
C
C*** DATA
RDIAN/57.295779513082320 D0/
DATA BIG/ 1.0030/

C
C*** LEITURA DAS LIGACOES NODAIS DOS ELEMENTOS E DO NUMERO DO MATERIAL
C
C*** WRITE (6,902)
C
C*** FORMAT(//,' ***** CARACTERISTICAS DOS ELEMENTOS ',
. //,' Elem. Mater. No.nos Nos ',
. //1X,6('='),1X,19('='),1X,63('='))

C
C DO IELEM = 1,NELEM
C READ (5,* ) J,MATNO(J),LTYPE,NNODE,(LNODS(I,J),I=1,NNODE)
C WRITE (6,903) J,MATNO(J),NNODE,(LNODS(I,J),I=1,NNODE)
C END DO
C
C*** FORMAT(I6,2I9,2X,9I7)
C
C LEITURA (E INTERPOLACAO) DAS COORDENADAS DOS NOS
C
C*** COORD = BIG
C
C
C IF (NTYPE.NE.2) WRITE (6,904)
C IF (NTYPE.EQ.2) WRITE (6,905)
C
C*** FORMAT(//,' *** COORDENADAS NODAIS',
. //3X,'No',9X,'X(m)',8X,'Y(m)',17X,'No',9X,'X(m)',8X,'Y(m)',
. //1X,29('='),15X,29('='))
C
C*** FORMAT(//,' *** COORDENADAS NODAIS',
. //3X,'No',9X,'R(m)',8X,'Z(m)',17X,'No',9X,'R(m)',8X,'Z(m)',
. //1X,29('='),15X,29('='))
C
C DO KOUNT = 1,NPOIN
C READ(5,*) IPOIN,(COORD(IDIME,IPOIN),IDIME=1,NDIME)
C IF(IPOIN.EQ.NPOIN) EXIT
C END DO
C
C IF (KOUNT.NE.NPOIN)
C .CALL NODE2D (COORD, LNODS, NDIME, NPOIN, NELEM, NNODE)
C
C NROWS = (NPOIN+1)/2
C DO IROWS = 1,NROWS
C ICOL1 = IROWS
C ICOL2 = MIN(ICOL1+NROWS,NPOIN)
C WRITE (6,906)(J,(COORD(I,J),I=1,NDIME),J=ICOL1,ICOL2,NROWS)

```

```

COORD(IDIME, IPOIN)=fscanf(dados, '%f', 1);
end
if IPOIN == NPOIN
    break
end
end

if KOUNT ~= NPOIN
    COORD=NODE2D(COORD, LNODS, NDIME, NPOIN, NELEM, NNODE);
end

k=true;
NROWS = floor((NPOIN+1)/2);
for IROWS = 1:NROWS
    ICOL1 = IROWS;
    ICOL2 = min(ICOL1+NROWS, NPOIN);
    fprintf('resultados, \n%d %10.5f %10.5f', ...,
        ICOL1, COORD(1, ICOL1), COORD(2, ICOL1));
    if k
        espacos(resultados, ' ', 14);
        fprintf('resultados, %5d %10.5f %10.5f', ...,
            ICOL2, COORD(1, ICOL2), COORD(2, ICOL2));
    end
    if ICOL2==NPOIN
        k=false;
    end
end

% VERIFICACAO DA MALHA
COORD=QCHEK2(NPOIN, COORD, MATNO, NMATS, NONOD, LNODS, NELEM, resultados);

% LEITURA DAS PROPRIEDADES DOS MATERIAIS
fprintf('resultados, \n\n ***** PROPRIEDADES DOS MATERIAIS');
fprintf('resultados, ...
    \n\n Mat.      k1      k2      theta      Q');
fprintf('resultados, ...
    \n      (m/s)      (m/s)      (graus)      (m3/s)\n ');
espacos(resultados, '=', 57);

for IMATS = 1:NMATS
    NUMAT=fscanf(dados, '%d', 1);
    for I=1:NPROP
        PROPS(I, NUMAT)=fscanf(dados, '%g', 1);
    end
    if NTYPE == 2
        PROPS(3, NUMAT) = 0;
    end
    fprintf('resultados, \n%d%d%15.5E%15.5E%9.2f%15.5E', ...,
        NUMAT, PROPS(:, NUMAT));
    THETA = PROPS(3, NUMAT);
    if THETA ~= 0
        PROPS(3, NUMAT) = THETA/ROIAN;
    end
end

% DEFINICAO DAS CONSTANTES DE INTEGRACAO NUMERICA
GAUSLG

```

```

906      FORMAT(I5, 3X, F10.5, 2X, F10.5, 14X, I5, 3X, F10.5, 2X, F10.5)
C
C***  VERIFICACAO DA MALHA
C
CALL QCHEK2 (COORD, LNODS, MATNO, NELEM, NPOIN, NMATS,
    .
    NNODE, NDIME)
C
C***  LEITURA DAS PROPRIEDADES DOS MATERIAIS
C
WRITE (6, 910)
FORMAT(//' ***** PROPRIEDADES DOS MATERIAIS',
    . // ' Mat.      k1      k2      theta      Q',
    . / '      (m/s)      (m/s)      (graus)      (m3/s)',
    . / 1X, 57(' '= '))
C
DO IMATS = 1, NMATS
    READ (5, * ) NUMAT
    READ (5, * )(PROPS(I, NUMAT), I=1, NPROP)
    IF (NTYPE.EQ.2) PROPS(3, NUMAT) = 0.000
    WRITE (6, 940) NUMAT, (PROPS(I, NUMAT), I=1, NPROP)
    THETA = PROPS(3, NUMAT)
    IF (THETA.NE.0.000) PROPS(3, NUMAT) = THETA/ROIAN
END DO
940      FORMAT(I4, 1P, 2E15.5, 0P, F9.2, 1PE15.5)
C
C***  DEFINICAO DAS CONSTANTES DE INTEGRACAO NUMERICA
C
CALL GAUSLG (NGAUS, POSGP, WEIGP)
C
RETURN
END

```

```

**** QOUTPT.m *****
%
%*** ESCREVE OS RESULTADOS FINAIS
%
%**** 2010.06.30 * LR ***** quasar_m ****

function QOUTPT(NPOIN,XVECT,IFFIX,COORD,REACT,QSUMS,GPCOD,...
    VELOC,GRADH,NTYPE,NTOTG,NTASK,resultados)

% POTENCIAL
if NTASK == 1
    % POTENCIAIS E ALTURAS PIEZOMETRICAS
    fprintf(resultados,'\n\n ');estrelas(resultados,17,0);
    fprintf(resultados,' Potenciais e alturas piezometricas ');
    estrelas(resultados,16,1);
    fprintf(resultados,'\n No H(m) Hw(m) ');
    fprintf(resultados,' No H(m) Hw(m) ');
    fprintf(resultados,'\n\n ');espacos(resultados,'=',31);
    espacos(resultados,' ',7);
    espacos(resultados,'=',31);
    NROWS =int16(NPOIN/2);
    for IROWS = 1:NROWS
        I1 = IROWS;
        I2 = min(I1+NROWS,NPOIN);
        fprintf(resultados,'\n%5d %14.6E%14.6E ',...,
            I1,XVECT(I1),XVECT(I1)-COORD(2,I1));
        fprintf(resultados,'%5d %14.6E%14.6E',...,
            I2,XVECT(I2),XVECT(I2)-COORD(2,I2));
        end

    % REACOES
    fprintf(resultados,'\n\n\n ***** Reaccoes *****');
    fprintf(resultados,'\n No Q(m3/s)\n ');
    espacos(resultados,'=',21);

    RPLUS = 0;
    RMINS = 0;
    for IPOIN = 1:NPOIN
        if IFFIX(IPOIN) ~= 0
            REACP = REACT(IPOIN);
            if NTASK == 1
                fprintf(resultados,'\n%5d %14.6E',IPOIN,REACP);
            end
            if REACP > 0
                RPLUS = RPLUS+REACP;
            else
                RMINS = RMINS+REACP;
            end
        end
    end

    QPLUS = QSUMS(1);
    QMINS = QSUMS(2);
    QTOTP = QPLUS+RPLUS;
    QOTTM = QMINS+RMINS;
    QTOTL = QTOTP+QOTTM;

    fprintf(resultados,...

```

```

SUBROUTINE QOUTPT (NPOIN, XVECT, IFFIX, REACT, COORD, NDIME,
    VELOC, GPCOD, NTOTG, NTYPE, QSUMS, GRADH,
    NTASK)
C*****
C
C*** ESCREVE OS RESULTADOS FINAIS
C
C**** 2010.06.11 * JCM ***** quasar *
C
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION COORD(NDIME, NPOIN), XVECT(NPOIN),
    VELOC(NDIME, NTOTG), REACT(NPOIN),
    GPCOD(NDIME, NTOTG), IFFIX(NPOIN),
    GRADH(NDIME, NTOTG), QSUMS(2)
C
C**** POTENCIAIS E ALTURAS PIEZOMETRICAS
C
IF (NTASK.EQ.1) THEN
    WRITE (6,900)
    NROWS = (NPOIN+1)/2
    DO IROWS = 1, NROWS
        I1 = IROWS
        I2 = MIN(I1+NROWS,NPOIN)
        WRITE (6,910) (J,XVECT(J),XVECT(J)-COORD(2,J),J=I1,I2,NROWS)
900    FORMAT(/3X,17(' '), ' Potenciais e alturas piezometricas ',16(' '),
        . /' No No H(m) Hw(m) ',
        . /3X,31(' '),7X,31(' '))
910    FORMAT(2(I5,1X,1P,E14.6,E14.6,4X))
C
C**** REACOES
C
IF (NTASK.EQ.1) WRITE (6,920)
RPLUS = 0.0D0
RMINS = 0.0D0
DO IPOIN = 1,NPOIN
    IF (IFFIX(IPOIN).NE.0) THEN
        REACP = REACT(IPOIN)
        IF (NTASK.EQ.1) WRITE (6,925) IPOIN,REACP
        IF (REACP.GT.0.0D0) THEN
            RPLUS = RPLUS+REACP
        ELSE
            RMINS = RMINS+REACP
        END IF
    END IF
END DO
QPLUS = QSUMS(1)
QMINS = QSUMS(2)
QTOTP = QPLUS+RPLUS
QOTTM = QMINS+RMINS
QTOTL = QTOTP+QOTTM
QSUMS(1) = QTOTP
QSUMS(2) = QTOTM
CC
CC
WRITE (6,926) RPLUS,QPLUS,QTOTP,RMINS,QMINS,QTOTM,QTOTL

```

```

['\n\n | Somatorios | Reaccoes | ',...
'Caud.prescritos | Caudais totais |\n '];
espacos(resultados,'=',67); Positivos %16.6E%16.6E%16.6E | ',...
fprintf(resultados,'\n | RPLUS,QPLUS,QTOTP);
fprintf(resultados,'\n | RMINS,QMINS,QTOTM);
espacos(resultados,'=',67);fprintf(resultados,'\n');
espacos(resultados,' ',46);
fprintf(resultados,'TOTAL =%16.6E',QTOTL);

% VELOCIDADE NOS PONTOS DE GAUSS
if NTYPE == 1
    fprintf(resultados,'\n\n ');espacos(resultados,'*',19);
    fprintf(resultados,' Velocidade nos Pontos de Gauss ');
    espacos(resultados,'*',18);
    fprintf(resultados,'\n p.G. x(m) y(m)');
    fprintf(resultados,' Vx(m/s) Vy(m/s)');
    fprintf(resultados,'\n ');espacos(resultados,'=',69);
else
    fprintf(resultados,'\n\n ');espacos(resultados,'*',19);
    fprintf(resultados,' Velocidade nos Pontos de Gauss ');
    espacos(resultados,'*',18);
    fprintf(resultados,'\n p.G. r(m) z(m)');
    fprintf(resultados,' Vr(m/s) Vz(m/s)');
    fprintf(resultados,'\n ');espacos(resultados,'=',69);
end
for I = 1:NTOTG
    fprintf(resultados,'\n%6d %10.3f %10.3f',...
        I,GPCOD(1,I),GPCOD(2,I));
    fprintf(resultados,' %14.6E %14.6E',...
        VELOC(1,I),VELOC(2,I));
end

% GRADIENTE HIDRAULICO NOS PONTOS DE GAUSS
if NTYPE == 1
    fprintf(resultados,'\n\n ');espacos(resultados,'*',18);
    fprintf(resultados,' Gradiente Hidraulico nos P.de G. ');
    espacos(resultados,'*',18);
    fprintf(resultados,'\n p.G. x(m) y(m)');
    fprintf(resultados,' ix(m/m) iy(m/m)');
    fprintf(resultados,'\n ');espacos(resultados,'=',69);
else
    fprintf(resultados,'\n\n ');espacos(resultados,'*',18);
    fprintf(resultados,' Gradiente Hidraulico nos P.de G. ');
    espacos(resultados,'*',18);
    fprintf(resultados,'\n p.G. r(m) z(m)');
    fprintf(resultados,' ir(m/m) iz(m/m)');
    fprintf(resultados,'\n ');espacos(resultados,'=',69);
end
for I = 1:NTOTG
    fprintf(resultados,'\n%6d %10.3f %10.3f',...
        I,GPCOD(1,I),GPCOD(2,I));
    fprintf(resultados,' %14.6E %14.6E',...
        GRADH(1,I),GRADH(2,I));
end

% FUNCAO DE CORRENTE

```

```

920 FORMAT(// ' ***** Reaccoes *****',
    /' No Q(m3/s)',/3X,21('='))
925 FORMAT(15,5X,1P,E14.6)
926 FORMAT(/' | Somatorios | ',
    Reaccoes | Caud.prescritos | Caudais totais | ',
    /3X,67('='),
    /' | Positivos | ',1P,3(E16.6,' | '),
    /' | Negativos | ', 3(E16.6,' | '),
    /3X,67('='),/46X,'TOTAL =',E16.6)

C*** VELOCIDADE NOS PONTOS DE GAUSS
C
C
    IF(NTYPE.EQ.1) THEN
        WRITE (6,930)
    ELSE
        WRITE (6,931)
    END IF
    DO I = 1,NTOTG
        WRITE (6,935) I,GPCOD(1,I),GPCOD(2,I),VELOC(1,I),VELOC(2,I)
    END DO
930 FORMAT(/3X,19('='),' Velocidade nos Pontos de Gauss ',18('='),
    /' p.G. x(m) y(m) Vy(m/s)',
    /3X,69('='),
    /3X,69('='),
    /' p.G. r(m) z(m) Vz(m/s)',
    /3X,69('='),
    /3X,69('='))
935 FORMAT(16,4X,F10.3,4X,F10.3,5X,1P,E14.6,5X,E14.6)
C
C*** GRADIENTE HIDRAULICO NOS PONTOS DE GAUSS
C
C
    IF(NTYPE.EQ.1) THEN
        WRITE (6,940)
    ELSE
        WRITE (6,941)
    END IF
    DO I = 1,NTOTG
        WRITE (6,935) I,GPCOD(1,I),GPCOD(2,I),GRADH(1,I),GRADH(2,I)
    END DO
940 FORMAT(/3X,18('='),' Gradiente Hidraulico nos P.de G.',18('='),
    /' p.G. x(m) y(m) ix(m/m) iy(m/m)',
    /3X,69('='),
    /3X,69('='))
941 FORMAT(/3X,18('='),' Gradiente Hidraulico nos P.de G.',18('='),
    /' p.G. r(m) z(m) ir(m/m) iz(m/m)',
    /3X,69('='),
    /3X,69('='))
C
C*** FUNCAO DE CORRENTE
C
    ELSE
        WRITE (6,950)
        NROWS = (NPOINT+2)/3
    END IF

```

```

else
    fprintf(resultados,'\\n\\n ');espacos(resultados,'*',25);
    fprintf(resultados,' Funcao de Corrente ');espacos(resultados,'*',24);
    fprintf(resultados,'\\n No Psi(m3/s) ');
    fprintf(resultados,' No Psi(m3/s) ');
    fprintf(resultados,'\\n No Psi(m3/s) \\n ');
    espacos(resultados,'=',17);espacos(resultados,' ',9);
    espacos(resultados,'=',17);espacos(resultados,' ',9);
    NROWS = floor((NPOIN+2)/3);
    for IROWS = 1:NROWS
        I1 = IROWS;
        I2 = min(I1+2*NROWS,NPOIN);
        for J=I1:NROWS:I2
            if J==I1
                fprintf(resultados,'\\n%d %14.6E ',J,XVECT(J));
            else
                fprintf(resultados,'%5d %14.6E ',J,XVECT(J));
            end
        end
    end
end
end
end

```

```

950 DO IROWS = 1, NROWS
    I1 = IROWS
    I2 = MIN(I1+2*NROWS,NPOIN)
    WRITE (6,955)(J,XVECT(J),J=I1,I2,NROWS)
    END DO
    FORMAT(/3X,25(' '), ' Funcao de Corrente ',24(' '),
    . /' No Psi(m3/s) ',
    . ' No Psi(m3/s) ',
    . ' No Psi(m3/s) ',
    . /3X,17(' '),9X,17(' '),9X,17(' '))
955 FORMAT(3(I5,1X,1P,E14.6,0P,6X))
END IF
RETURN
END
C

```



```

disp('Leitura dos parametros de controlo')
QINPT1

disp('Leitura da topologia da malha')
QINPT2

disp('Cálculo das coordenadas dos pontos de Gauss')
GPCORD

% CALCULO DOS POTENCIAIS NOS NOS
disp('Cálculo dos potenciais nodais')

% LEITURA DAS CONDICÖES FRONTEIRA (DE DIRICHLET)
disp('Leitura das condicoes-fronreira - POTENCIAIS')
PBOUND
if ~CONFN
    RBOUND
end

% CALCULO DA MATRIZ DE COEFICIENTES GLOBAL
disp('Formacao da matriz de coeficientes global')
PSTIF0

% CALCULO DO TERMO INDEPENDENTE
disp('Formacao do termo independente')
PLOADO

% INTRODUCAO DAS CONDICÖES FRONTEIRA (DE DIRICHLET)
disp('Introdução das condições-fronreira - Potenciais')
[GSTIF,GLOAD]=BDCOND(GSTIF,NTOTV,GLOAD,IPFIX,PFIXD);

% RESOLUCAO DO SISTEMA DE EQUACOES
disp('Resolucao do sistema de equacoes')
HNODL=GSTIF\GLOAD;
% HNODL=GAUSSS(GSTIF,GLOAD,NPOIN)';

% TRATAMENTO DOS ESCOAMENTOS NAO CONFINADOS
if ~CONFN
    NCONFI
end

% CALCULO DA VELOCIDADE E DOS CAUDAIS NAO EQUILIBRADOS
disp('Calculo da velocidade nos pontos de Gauss')
QVELOC

% ESCRITA DOS RESULTADOS FINAIS - POTENCIAIS
disp('Escrita dos resultados finais - POTENCIAIS')
QOUTPT(NPOIN,HNODL,IPFIX,COORD,GLOAD,QSUMS,GPCORD,...
    VELOC,GRADH,NTYPE,NTOTG,1,resultados)

% CALCULO DA FUNCAO DE CORRENTE NOS NOS
disp('CALCULO DA FUNCAO DE CORRENTE')

% LEITURA DAS CONDICÖES FRONTEIRA (DE DIRICHLET)
disp('Leitura das condicoes-fronreira - F.DE CORRENTE')
CBOUND

% CALCULO DA MATRIZ DE COEFICIENTES GLOBAL

```

```

DO JNAML = 72,1,-1
    IF(JNAME(JNAML:JNAML).NE.' ') EXIT
END DO
OPEN (5, FILE=JNAME(1:JNAML)//'.DAT', STATUS='OLD', IOSTAT=IOCH5)
IF(IOCH5.EQ.0) THEN
    OK = .TRUE.
    OPEN (6, FILE= JNAME(1:JNAML)//'.RES', STATUS='UNKNOWN')
    ELSE
        PAUSE ' ===> Tecle DIR *.DAT para ver nomes de ficheiros '
    END IF
END DO

C***
C ESCRITA DO CABECALHO DO FICHEIRO DE RESULTADOS
C
call getdat (ano, mes, dia)
call gettim (hora, min, seg, ii00)

C
WRITE (6,9000) PNAME,VNAME,ano,mes,dia,hora,min,seg,
    JNAME(1:JNAML)//'.DAT',JNAME(1:JNAML)//'.RES'
9000 FORMAT(1X,78(' '),1X,36(' '), ' Programa ',A,1X,A,1X,3(' '),
    /1X,63(' '),1X,14.4,' ',I2.2,' ',I2.2,1X,3(' '),
    /1X,3(' '), ' Ficheiro de dados:',
    6X,35(' '),3X,I2.2,' ',I2.2,' ',I2.2,1X,3(' '),
    /1X,3(' '),1X,A,T77.3(' '),/1X,78(' '),
    /1X,3(' '), ' Ficheiro de resultados: ',50(' '),
    /1X,3(' '),1X,A,T77.3(' '))

C
C*** INTRODUCAO DA TOPOLOGIA DA MALHA
C
WRITE (*,*) ' *** Leitura dos parametros de controle'
CALL QINPT1 (NELEM, NPOIN, NMATS, NPFIX, NTYPE, NGAUS,
    NPROP, NDOFN, NDIME, NNODE, NTOTG, NEVAB,
    NTOTV, MPOIN, MELEM, MMATS, MNODE, CONFN)

C
WRITE (*,*) ' *** Leitura da topologia da malha'
CALL QINPT2 (NELEM, NPOIN, NMATS, NTYPE, COORD, NDIME,
    NNODE, NGAUS, LNODS, MATNO, NPROP, PROPS,
    POSGP, WEIGP)

C
WRITE (*,*) ' *** Calculo das coordenadas dos pontos de Gauss'
CALL GPCORD (COORD, GPCORD, LNODS, NELEM, NNODE, NPOIN,
    NTOTG, NDIME, NGAUS, POSGP)

C
C*** CALCULO DOS POTENCIAIS NOS NOS *****
C
WRITE (*,*) ' *** CALCULO DOS POTENCIAIS NODAIS *****'

C
LEITURA DAS CONDICÖES FRONTEIRA (DE DIRICHLET)

C***
C
WRITE (*,*) ' *** Leitura das condicoes-fronreira - POTENCIAIS'
CALL PBOUND (NPFIX, IPFIX, PFIXD, NPOIN)
IF(.NOT.CONFN)
    .CALL RBOUND (NREVU, NDRVU, NPOIN, FPERM, TOLCV)
WRITE (*,*) ' *** Escrita de um ficheiro de desenho'

```

```

disp('Formacao da matriz de coeficientes global')
CSTIF0

% CALCULO DO TERMO INDEPENDENTE
CLOAD0

% INTRODUCAO DAS CONDICoes FRONTEIRA (DE DIRICHLET)
[ GSTIF, GLOAD]=BDCOND(GSTIF,NTOTV,GLOAD,ICFIX,CFIXD);

% DETERMINACAO DO VALOR DA FUNCAO DE CORRENTE NOS NOS
disp('Resolucao do sistema de equacoes')
XVECT=GSTIF\GLOAD;
% XVECT=GAUSSS(GSTIF, GLOAD, NPOIN)';

% ESCRITA DOS RESULTADOS FINAIS
disp('Escrita dos resultados finais - F.DE CORRENTE')
QOUTPT(NPOIN,XVECT,ICFIX,COORD,GLOAD,QSUMS,GPCOD,...
        VELOC,GRADH,NTYPE,NTOTG,2,resultados)

% FECHO DE CANAIS DE LEITURA E ESCRITA
fclose(dados);
fclose(resultados);

ERRMSG(0,'QUASAR')

%visualização
QPLOTM2

```

```

CALL QPLOT (NPOIN, NELEM, NPFIX, LNODS, NNODE, COORD,
            NDIME, XVECT, IPFIX, VELOC, GPCOD, NTOTG,
            GRADH, PNAME, VNAME, JNAME, JNAMEL, 0,
            )
.
.
C***
C
CALCULO DO TERMO INDEPENDENTE

WRITE (*,*) ' *** Formacao do termo independente'
CALL PLOAD0 (COORD, LNODS, MATNO, NELEM, NNODE, NGAUS,
            NPOIN, NTYPE, POSGP, PROPS, NMATS, WEIGP,
            NPROP, NDIME, GLOAD, GPCOD, NTOTG, QSUMS,
            CONFN)
.
C
CALCULO DA MATRIZ DE COEFICIENTES GLOBAL

C***
C
WRITE (*,*) ' *** Formacao da matriz de coeficientes global'
CALL PSTIF0 (MATNO, PROPS, COORD, NDIME, LNODS, GSTIF,
            NMATS, NPROP, NTOTV, NELEM, NNODE, NPOIN,
            POSGP, WEIGP, NGAUS, NTYPE, ESTIF, NEVAB,
            GPCOD, NTOTG, CONFN)
.
C
INTRODUCAO DAS CONDICoes FRONTEIRA (DE DIRICHLET)

C***
C
WRITE (*,*) ' *** Introduco das condicoes-fronreira - POTENCIAIS'
CALL BDCOND (GSTIF, NTOTV, GLOAD, IPFIX, PFIxD)

C
RESOLUCAO DO SISTEMA DE EQUACOES

C***
C
WRITE (*,*) ' *** Resolucao do sistema de equacoes'
CALL GAUSS (GSTIF, GLOAD, NPOIN, HNODL)

C
TRATAMENTO DOS ESCOAMENTOS NAO CONFINADOS

C***
C
IF(.NOT.CONFN)
.
CALL NCONFI (MATNO, PROPS, COORD, NDIME, LNODS, GLOAD,
            NMATS, NPROP, NELEM, NNODE, NPOIN, GPCOD,
            POSGP, WEIGP, NGAUS, NTYPE, VELOC, GRADH,
            HNODL, DELTH, NTOTG, DPFIX, NREVV, FPERM,
            GSTIF, NTOTV, NDRVV, IPFIX, TOLCV)
.
C
CALCULO DA VELOCIDADE E DOS CAUDAIS NAO EQUILBRADOS

C***
C
WRITE (*,*) ' *** Calculo da velocidade nos pontos de Gauss'
CALL QVELOC (MATNO, PROPS, COORD, NDIME, LNODL,
            NMATS, NPROP, NGAUS, NELEM, NNODE, NPOIN,
            POSGP, WEIGP, NTYPE, VELOC, NTOTG, GRADH,
            GPCOD, GLOAD, NTOTV, FPERM)
.
C
ESCRITA DOS RESULTADOS FINAIS - POTENCIAIS

C***
C
WRITE (*,*) ' *** Escrita dos resultados finais - POTENCIAIS'
CALL QOUTPT (NPOIN, HNODL, IPFIX, GLOAD, COORD, NDIME,
            VELOC, GPCOD, NTOTG, NTYPE, QSUMS, GRADH,
            1)
.

```

```
WRITE (*,*) ' *** Escrita de um ficheiro de desenho '  
CALL QPLOT (NPOIN, NELEM, NCFIX, LNODS, NNODE, COORD,  
NDIME, XVECT, ICFIX, VELOC, GPCOD, NTOTG,  
GRADH, PNAME, VNAME, JNAME, JNAML, 1,  
2)  
.  
.  
.  
C  
C*** FECHO DE CANAIS DE LEITURA E ESCRITA  
C  
CLOSE (5)  
CLOSE (6)  
C  
CALL ERRMSG (PNAME, 0)  
END
```

```

SUBROUTINE QVELOC (MATNO, PROPS, COORD, NDIME, LNODS, HNODL,
.      NMATS, NPROP, NGAUS, NELEM, NNODE, NPOIN,
.      POSGP, WEIGP, NTYPE, VELOC, NTOTG, GRADH,
.      GPCOD, GLOAD, NTOTV, FPERM)
C*****
C
C*** CALCULA A VELOCIDADE NOS PONTOS DE GAUSS P/ A SOLUCAO EM TERMOS DO
C POTENCIAL {v} = -[k]{grad fi}
C*** CALCULA OS CAUDAIS NODAIS EQUIVALENTES
C
C*** 2010.06.24 * JCM ***** quasar *****
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C DIMENSION MATNO(NLEM), PROPS(NPROP, NMATS), VELOC(NDIME, NTOTG),
.      POSGP(NGAUS), LNODS(NNODE, NELEM), GPCOD(NDIME, NTOTG),
.      WEIGP(NGAUS), COORD(NDIME, NPOIN), GLOAD(NTOTV),
.      HNODL(NPOIN), GRADH(NDIME, NTOTG)
C DIMENSION ELCOD(2,9), PERMB(2,2), HVALU(9), ELRSD(9), NODES(9),
.      CARTD(2,9), SHAPE(9)
C DATA TWOP1/ 6.283185307179586 D0/
C
C*** INICIALIZACAO DO VETOR DE RESIDUOS (FLUXOS NAO EQUILIBRADOS)
C
C REWIND 20
C READ (20) GLOAD
C
C*** CICLO SOBRE OS ELEMENTOS
C
C GAUS = 0
C DO IELEM = 1, NELEM
C   ELRSD = 0.0D0
C
C*** VALORES DE FI E COORDENADAS DOS NOS DO ELEMENTO
C
C DO INODE = 1, NNODE
C   IPOIN = LNODS(INODE, IELEM)
C   NODES(INODE) = IPOIN
C   HVALU(INODE) = HNODL(IPOIN)
C   DO IDIME = 1, NDIME
C     ELCOD(IDIME, INODE) = COORD(IDIME, IPOIN)
C   END DO
C END DO
C
C*** CARACTERISTICAS MATERIAIS DO ELEMENTO
C
C IMATS = MATNO(IELEM)
C PERMB(1,1) = PROPS(1, IMATS)
C PERMB(2,2) = PROPS(2, IMATS)
C PERMB(1,2) = 0.0D0
C THETA
C   = PROPS(3, IMATS)
C
C*** DETERMINACAO DA MATRIZ DE PERMEABILIDADE EM RELACAO A XY
C
C IF(PERMB(1,1).NE.PERMB(2,2).AND.THETA.NE.0.0D0)
C   CALL RTMAT2 (PERMB, THETA)

```

```

if NTYPE == 2
    DVOLU = DVOLU*Twopi*GPCOD(1,KGAUS);
end

% CALCULO DE DFI/DX E DFI/DY E DO POTENCIAL NO PONTO DE GAUSS
SHAPE=SHAPE2(EXISP,ETASP,NNODE,resultados);
DFIDX = sum(CARTD(1,1:NNODE).*HVALU(1:NNODE)');
DFIDY = sum(CARTD(2,1:NNODE).*HVALU(1:NNODE)');
HGAUS = sum(SHAPE(1:NNODE).*HVALU(1:NNODE));

% CALCULO DO GRADIENTE HIDRAULICO NO PONTO DE GAUSS
GRADH(1,KGAUS) = DFIDX;
GRADH(2,KGAUS) = DFIDY;
if -DFIDY > -GHVMX
    GHVMX = DFIDY;
    MXGHV = KGAUS;
end

% CALCULO DA VELOCIDADE NO PONTO DE GAUSS
PFACT=1;
if HGAUS < GPCOD(2,KGAUS)
    PFACT=FPERM;
end
VELOC(:,KGAUS) = -(PERMB*GRADH(:,KGAUS))*PFACT;

t
% CALCULO DE B * V
for INODE = 1:NNODE
    ELRSD(INODE) = ...
        ELRSD(INODE) + (CARTD(1,INODE)*VELOC(1,KGAUS)+...
        CARTD(2,INODE)*VELOC(2,KGAUS))*DVOLU;
end
end
end

% ESPALHAMENTO DA CONTRIBUICAO DO ELEMENTO PARA O RESIDUO
GLOAD(NODES) = GLOAD(NODES) + ELRSD(1:NNODE);

end

fprintf(resultados,...
'\n\n *** Grad.Hidraulico Vertical Maximo =%15.7E(G.P. %d)',...
GHVMX,MXGHV);

```

```

C
C*** CICLOS SOBRE OS PONTOS DE GAUSS
C
    KGASP = 0
    DO IGAUS = 1,NGAUS
        EXISP = POSGP(IGAUS)
        DO JGAUS = 1,NGAUS
            KGAUS = KGAUS+1
            KGASP = KGASP+1
            ETASP = POSGP(JGAUS)

C
C*** DETERMINACAO DAS DERIVADAS CARTESIANAS DAS FUNCOES DE FORMA E
C DO ELEMENTO DE VOLUME
C
        CALL JACOB2 (CARTD, DJACB, ELCOD, KGAUS, IELEM, ETASP,
        . EXISP, NNODE, NDIME)
        DVOLU = DJACB*WEIGP(IGAUS)*WEIGP(JGAUS)
        IF(NTYPE.EQ.2) DVOLU = DVOLU*Twopi*GPCOD(1,KGAUS)

C
C*** CALCULO DE DFI/DX E DFI/DY E DO POTENCIAL NO PONTO DE GAUSS
C
        CALL SHAPE2 (ETASP, EXISP, NNODE, SHAPE)
        HGAUS = 0.0D0
        DFIDX = 0.0D0
        DFIDY = 0.0D0
        DO INODE = 1,NNODE
            HINOD = HVALU(INODE)
            DFIDX = DFIDX+CARTD(1,INODE)*HINOD
            DFIDY = DFIDY+CARTD(2,INODE)*HINOD
            HGAUS = HGAUS+SHAPE(INODE) *HINOD
        END DO

C
C*** CALCULO DO GRADIENTE HIDRAULICO NO PONTO DE GAUSS
C
        GRADH(1,KGAUS) = DFIDX
        GRADH(2,KGAUS) = DFIDY

C
C*** CALCULO DA VELOCIDADE NO PONTO DE GAUSS
C
        PFACT = 1.0D0
        IF (HGAUS.LT.GPCOD(2,KGAUS)) PFACT = FPERM
        VELOC(1,KGAUS) = -(PERMB(1,1)*DFIDX+PERMB(1,2)*DFIDY)
        . *PFACT
        VELOC(2,KGAUS) = -(PERMB(1,2)*DFIDX+PERMB(2,2)*DFIDY)
        . *PFACT
        t
C
C*** CALCULO DE B *V
C
        DO INODE = 1,NNODE
            ELRSD(INODE) =
        . ELRSD(INODE) + (CARTD(1,INODE)*VELOC(1,KGAUS)+
        . CARTD(2,INODE)*VELOC(2,KGAUS))*DVOLU
        END DO
END DO

```

```
C      END DO
C***  ESPALHAMENTO DA CONTRIBUICAO DO ELEMENTO PARA O RESIDUO
C
      DO INODE = 1, NNODE
        IPOIN = NODES(INODE)
        GLOAD(IPOIN) =
          .      GLOAD(IPOIN) + ELRSD(INODE)
          .      END DO
        END DO
      END DO
      RETURN
      END
```

C

```

**** ROUND_m *****
%
%*** LE A LISTA DE NOS A INVESTIGAR COM CONDICÕES FRONTEIRA
% REDEFINIVEIS (REVIEW BOUNDARY CONDITIONS)
%
%**** 2010.06.30 * LR ***** quasar_m ****

NDRVU=zeros(NPOIN,1);

% BOUNDARY REVIEW NODES
NREVU=fscanf(dados,'%d',1);
fprintf('resultados,...
[\'\\n\\n *** CONDICÕES FRONTEIRA REDEFINIVEIS - \' ,...
\'no. de nos a investigar = %d\\n\',NREVU);
if NREVU ~= 0
    for I=1:NREVU
        NDRVU(I)=fscanf(dados,'%d',1);
        fprintf('resultados, %5d',NDRVU(I));
    end
end

FPERM=fscanf(dados,'%f',1);
TOLCV=fscanf(dados,'%f',1);
fprintf('resultados,...
\\n *** Metodo da permeabilidade variavel - parametros');
fprintf('resultados,...
\\n FPERM (Kfinal/Kinicial) = %15.7E',FPERM);
fprintf('resultados,...
\\n TOLCV (tolerancia de convergencia)= %15.7E',TOLCV);

```

```

SUBROUTINE RBOUND (NREVU, NDRVU, NPOIN, FPERM, TOLCV)
C*****
C
C*** LE A LISTA DE NOS A INVESTIGAR COM CONDICÕES FRONTEIRA
C REDEFINIVEIS (REVIEW BOUNDARY CONDITIONS)
C
C**** 2010.06.17 * JCM ***** quasar *
C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C DIMENSION NDRVU(NPOIN)
C
C*** BOUNDARY REVIEW NODES
C
C    READ (5, * ) NREVU
C    WRITE (6,910) NREVU
C    IF (NREVU.NE.0) THEN
C        READ (5, * )(NDRVU(I), I=1,NREVU)
C        WRITE (6,915)(NDRVU(I), I=1,NREVU)
C    ENDIF
C    910  FORMAT(//' ' *** CONDICÕES FRONTEIRA REDEFINIVEIS',
C          .  FORMAT(//' ' - no. de nos a investigar = ',I3)
C
C    READ (5,*) FPERM,TOLCV
C    WRITE(6,1000) FPERM,TOLCV
C    1000  FORMAT(//' *** Metodo da permeabilidade variavel - parametros',
C          .  /' FPERM (Kfinal/Kinicial) = ',1PE15.7,
C          .  /' TOLCV (tolerancia de convergencia)= ', E15.7)
C
C    RETURN
C    END

```

```

%*** SHAPE2 *****
%
%*** CALCULA AS FUNCOES DE INTERPOLACAO PARA
% * ELEMENTOS PLANOS DE 4,8 E 9 NOS
%
%***** 2010.06.30 * LR *****
function SHAPE=SHAPE2(S,T,NNODE,resultados)

RNAME='SHAPE2';
SHAPE=zeros(NNODE,1);

% ELEMENTOS DE 4 NOS
if NNODE == 4

    SM = (1-S)/2;
    SP = (1+S)/2;
    TM = (1-T)/2;
    TP = (1+T)/2;

    SHAPE(1) = SM*TM;
    SHAPE(2) = SP*TM;
    SHAPE(3) = SP*TP;
    SHAPE(4) = SM*TP;

% ELEMENTOS DE 8 NOS
elseif NNODE == 8

    SM = 0.5*(1-S);
    TM = 0.5*(1-T);
    SP = 0.5*(1+S);
    TP = 0.5*(1+T);
    SS = 0.5*(1-S*S);
    TT = 0.5*(1-T*T);

    SHAPE(1) = +SM*TM*(SM+TM-1.5)*2;
    SHAPE(2) = +SS*TM*2;
    SHAPE(3) = +SP*TM*(SP+TM-1.5)*2;
    SHAPE(4) = +SP*TT*2;
    SHAPE(5) = +SP*TP*(SP+TP-1.5)*2;
    SHAPE(6) = +SS*TP*2;
    SHAPE(7) = +SM*TP*(SM+TP-1.5)*2;
    SHAPE(8) = +SM*TT*2;

% ELEMENTOS DE 9 NOS
elseif NNODE == 9

    SSM = (S*S-S)/2;
    SSP = (S*S+S)/2;
    SS0 = -S*S+1;
    TTM = (T*T-T)/2;
    TTP = (T*T+T)/2;
    TT0 = -T*T+1;

    SHAPE(1) = SSM*TTM;
    SHAPE(2) = SS0*TTM;
    SHAPE(3) = SSP*TTM;
    SHAPE(4) = SSP*TT0;

```

```

SUBROUTINE SHAPE2 (ETASP, EXISP, NNODE, SHAPE)
C*****
C
C*** CALCULA AS FUNCOES DE INTERPOLACAO PARA
C * ELEMENTOS PLANOS DE 4,8 E 9 NOS
C
C***** 2010.06.11 * JCM *****
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
CHARACTER RNAME*6
DIMENSION SHAPE(9)
DATA RNAME/'SHAPE2' /
C
C S = EXISP
C T = ETASP
C
C*** ELEMENTOS DE 4 NOS
C
C IF (NNODE.EQ.4) THEN
C
C SM = (1.0D0-S)/2.0D0
C SP = (1.0D0+S)/2.0D0
C TM = (1.0D0-T)/2.0D0
C TP = (1.0D0+T)/2.0D0
C
C SHAPE(1) = SM*TM
C SHAPE(2) = SP*TM
C SHAPE(3) = SP*TP
C SHAPE(4) = SM*TP
C
C
C*** ELEMENTOS DE 8 NOS
C
C ELSE IF (NNODE.EQ.8) THEN
C
C SM = 0.5D0*(1.0D0-S)
C TM = 0.5D0*(1.0D0-T)
C SP = 0.5D0*(1.0D0+S)
C TP = 0.5D0*(1.0D0+T)
C SS = 0.5D0*(1.0D0-S*S)
C TT = 0.5D0*(1.0D0-T*T)
C
C SHAPE(1) = +SM*TM*(SM+TM-1.5D0)*2.0D0
C SHAPE(2) = +SS*TM*2.0D0
C SHAPE(3) = +SP*TM*(SP+TM-1.5D0)*2.0D0
C SHAPE(4) = +SP*TT*2.0D0
C SHAPE(5) = +SP*TP*(SP+TP-1.5D0)*2.0D0
C SHAPE(6) = +SS*TP*2.0D0
C SHAPE(7) = +SM*TP*(SM+TP-1.5D0)*2.0D0
C SHAPE(8) = +SM*TT*2.0D0
C
C
C*** ELEMENTOS DE 9 NOS
C
C ELSE IF (NNODE.EQ.9) THEN
C
C SSM = (S*S-S)/2.0D0

```

```
SHAPE(5) = SSP*TTP;  
SHAPE(6) = SS0*TTP;  
SHAPE(7) = SSM*TTP;  
SHAPE(8) = SSM*TT0;  
SHAPE(9) = SS0*TT0;  
  
end
```

```
C  
  
SSP = (S*S+S)/2.0D0  
SS0 = -S*S+1.0D0  
TTM = (T*T-T)/2.0D0  
TTP = (T*T+T)/2.0D0  
TT0 = -T*T+1.0D0  
  
SHAPE(1) = SSM*TTM  
SHAPE(2) = SS0*TTM  
SHAPE(3) = SSP*TTM  
SHAPE(4) = SSP*TT0  
SHAPE(5) = SSP*TTP  
SHAPE(6) = SS0*TTP  
SHAPE(7) = SSM*TTP  
SHAPE(8) = SSM*TT0  
SHAPE(9) = SS0*TT0  
  
C  
C*** ERRO  
C  
ELSE  
  WRITE (6,910) NNODE  
  CALL ERRMSG (RNAME, 1)  
END IF  
910 FORMAT(/' *** STOP na subrotina SHAPE2 - elemento finito com ',I5,  
  .      ' nos')  
C  
RETURN  
END
```


A3

PÓS PROCESSAMENTO

```

%***  check.m  *****
%
%***  averigua a existencia de valores constantes
%
%****  2010.06.30 * LR ***** quasar_m ****

function [valores,flag]=check(valores)
maximo=max(valores);
maximo=max(maximo);
minimo=min(valores);
minimo=min(minimo);
dif=maximo-minimo;
flag=0;
if dif<1E-10
    [m,n]=size(valores);
    valores=valores(1)*ones(m,n);
    flag=1;
end

```

```

%***  check_box.m  *****
%
%***  altera o estado da check box
%
%****  2010.06.30 * LR ***** quasar_m ****

function check_box(h_check_box,handle)
status=get(h_check_box,'value');
if status==0
    set(handle,'visible','off')
else
    set(handle,'visible','on')
end

```

```

%***  fradio.m  *****
%
%***  acções relativas aos botões de rádio
%
%****  2010.06.30 * LR ***** quasar_m ****

function fradio(h_radio,button,flag,texto)
% garante que quando se carrega num botão este permanece activo
% e os restantes são desactivados
num_buttons=length(h_radio);
if get(h_radio(button),'value') == 1
    set(h_radio([1:(button-1),(button+1):num_buttons]),'value',0)
else
    set(h_radio(button),'value',1)
end
% identifica cada uma das colorações e torna visível a correspondente
% ao botão activo, bem como o respectivo título
eixos=findobj('tag','eixos');
handle=zeros(3,1);
handle(1)=findobj('tag','col_pot');
handle(2)=findobj('tag','col_corr');
handle(3)=findobj('tag','col_piez');

```

```

switch button
    case 1
        set(handle(1),'visible','on')
        set(handle([2,3]),'visible','off')
        set(eixos,'title',...
            text('string','\bfMapa de Cores do Potencial'))
    case 2
        set(handle(2),'visible','on')
        set(handle([1,3]),'visible','off')
        set(eixos,'title',...
            text('string','\bfMapa de Cores da F. Corrente'))
    case 3
        set(handle(3),'visible','on')
        set(handle([1,2]),'visible','off')
        set(eixos,'title',text('string',...
            '\bfMapa de Cores da Altura Piezométrica'))
    case 4
        set(handle,'visible','off')
        set(eixos,'title',text('string',...
            '\bfSem Mapa de Cores'))
end
set(get(eixos,'title'),'fontsize',15)
% garante que a rede de fluxo só está disponível quando não há colorações
% ativas e apaga a barra de cores nesse caso
barra_cores=findobj('tag','barracores');
children=get(barra_cores,'children');
barra_cores=[barra_cores,children];
check_box_rede=findobj('tag','check_box_rede');
rede=findobj('tag','rede');
if button == 4
    set(barra_cores,'visible','off')
    set(check_box_rede,'enable','on')
else
    set(barra_cores,'visible','on')
    barra_cores=colorbar;
    set(barra_cores,'tag','barracores')
    set(check_box_rede,'value',0)
    check_box(check_box_rede,rede)
    set(check_box_rede,'enable','off')
end
% torna a barra de cores invisível e acrescenta uma etiqueta com o valor da
% função
if flag==1
    barra_cores=findobj('tag','barracores');
    children=get(barra_cores,'children');
    barra_cores=[barra_cores,children];
    set(barra_cores,'visible','off')
    switch button
        case 1
            set(get(eixos,'ylabel'),'string',texto(1),'visible','on');
        case 2
            set(get(eixos,'ylabel'),'string',texto(2),'visible','on');
        case 3
            set(get(eixos,'ylabel'),'string',texto(3),'visible','on');
    end
    set(get(eixos,'ylabel'),'fontsize',12);
else
    set(get(eixos,'ylabel'),'visible','off');
end
end

```

```

%***  imprimir.m  *****
%
%***  acciona a caixa de diálogo de impressão e esconde os botões
%
%****  2010.06.30 * LR ***** quasar_m ****

function imprimir(figura,imagem,menu)

set(menu,'visible','off')
posicao=get(imagem,'outerposition');
set(imagem,'outerposition',[0 0 1 1])
printdlg(figura)
set(imagem,'outerposition',posicao)
set(menu,'visible','on')

```

```

%***  malha8.m  *****
%
%***  INTERPOLA AS COORDENADAS E O POTENCIAL:
%      * NO CENTRAL DE ELEMENTOS DE 8 NOS
%
%****  2010.06.30 * LR ***** quasar_m ****

function [COORD,HNODL]=malha8(COORD, HNODL, LNODS, NDIME, NELEM,
NNODE,flag)

COORD9=zeros(2,NELEM);
HNODL9=zeros(NELEM,1);

% CICLO SOBRE OS ELEMENTOS
for IELEM = 1:NELEM

    NODES=LNODS(1:NNODE,IELEM);

    if flag == 1
        % COORDENADAS
        for INODE = 1:2:7
            NCORN = NODES(INODE);
            NMIDS = NODES(INODE+1);
            for IDIME = 1:NDIME
                COORD9(IDIME,IELEM) = COORD9(IDIME,IELEM)...
                    - COORD(IDIME,NCORN)/4 + COORD(IDIME,NMIDS)/2;
            end
        end
    end

    % POTENCIAL
    for INODE = 1:2:7
        NCORN = NODES(INODE);
        NMIDS = NODES(INODE+1);
        HNODL9(IELEM) = HNODL9(IELEM)...
            - HNODL(NCORN)/4 + HNODL(NMIDS)/2;
    end
end
if flag == 1
    COORD=[COORD,COORD9];
end
HNODL=[HNODL;HNODL9];

```

```

%*** nivel0.m *****
%
%*** identifica os pontos correspondentes à altura piezométrica igual a 0*
%
%**** 2010.06.30 * LR ***** quasar_m ****

function [xdata,ydata]=nivel0(C)

n=size(C,2);
j=1;
k=0;
while j<=n
    value=C(1,j);
    dim=C(2,j);
    if value == 0
        k=k+1;
        if k==1
            xdata=C(1,j+1:j+dim);
            ydata=C(2,j+1:j+dim);
        else
            xdata=[xdata,C(1,j+1:j+dim)];
            ydata=[ydata,C(2,j+1:j+dim)];
        end
    end
    j=j+dim+1;
end
end

```

```

%*** ordenadas.m *****
%
%*** identifica o conjunto de valores distintos
%
%**** 2010.06.30 * LR ***** quasar_m ****

function single=ordenadas(coord,dim,modo)
ord=sort(coord,modo);
k=1;
single(1)=ord(1);
for i=2:dim
    if abs(ord(i-1)-ord(i))>1E-10
        k=k+1;
        single(k)=ord(i);
    end
end
end

```

```

%*** QPLOTM2.m *****
%
%*** Representação gráfica dos resultados
%
%**** 2010.06.30 * LR ***** quasar_m ****

output_graf=figure;
set(output_graf,'name',JNAME,'numbertitle','off',...
'menubar','none','color','w','units','normalized',...
'position',[0.15 0.15 .7 .7])
% set(output_graf,'CloseRequestFcn','closemyfig')

```

```

eixos=axes;
set(eixos,'units','normalized','outerposition',...
[0,0,.75,1],'tag','eixos')
axis(eixos,'equal')
xmin=min(COORD(1,:));xmax=max(COORD(1,:));
ymin=min(COORD(2,:));ymax=max(COORD(2,:));
delta_x=(xmax-xmin);delta_y=(ymax-ymin);
axis(eixos,[xmin-0.1*delta_x xmax+0.1*delta_x,...
ymin-0.1*delta_y ymax+0.1*delta_y])
hold on
texto_JNAME=xlabel(JNAME,'fontsize',12);
ytexto=ylabel('');
text(xmax,ymin-0.5*delta_y,'x(m)')
% text(xmin-0.6*delta_x,ymax,'y(m)')
text(-.25*delta_x,ymax,'y(m)')

% representaçao dos nos
nos=plot(COORD(1,:),COORD(2,),'oy','markersize',4);

% MALHA
malha=zeros(1,NELEM);
for IELEM = 1 : NELEM
    NNODE=NONOD(IELEM);
    if NNODE < 9

malha(IELEM)=line(COORD(1,[LNODS(1:NNODE,IELEM);LNODS(1,IELEM)]),...
COORD(2,[LNODS(1:NNODE,IELEM);LNODS(1,IELEM)]),'Color','y');
    else
        malha(IELEM)=line(COORD(1,[LNODS(1:8,IELEM);LNODS(1,IELEM)]),...
COORD(2,[LNODS(1:8,IELEM);LNODS(1,IELEM)]),'Color','y');
    end
end

% CONDICÕES-FRONTEIRA DA FUNÇÃO POTENCIAL
condfp=plot(eixos,COORD(1,find(IPFIX)),COORD(2,find(IPFIX)),'og',...
'MarkerFaceColor','g');
set(condfp,'visible','off')

% COND.FRONTEIRA DE DIRICHLET (DA F.DE CORRENTE)
condfc=plot(eixos,COORD(1,find(ICFIX)),COORD(2,find(ICFIX)),'ob',...
'MarkerFaceColor','b');
set(condfc,'visible','off')

% numerar os nós
num_nos=zeros(NPOIN,1);
for IPOIN=1:NPOIN
    num_nos(IPOIN)=text(COORD(1,IPOIN),COORD(2,IPOIN),int2str(IPOIN));
end
set(num_nos,'visible','off')

% numerar os elementos
elem=zeros(NELEM,1);
for IELEM=1:NELEM
    NNODE=NONOD(IELEM);
    xelem=mean(COORD(1,LNODS(1:NNODE,IELEM)));
    yelem=mean(COORD(2,LNODS(1:NNODE,IELEM)));
    elem(IELEM)=text(xelem,yelem,int2str(IELEM),'color','b');

```

```

end
set(elem,'visible','off')

% numerar os pontos de GAUSS
xg=GPCOD(1,:);yg=GPCOD(2,:);zg=sqrt(VELOC(1,:).^2+VELOC(2,:).^2);
xgmin=min(GPCOD(1,:));xgmax=max(GPCOD(1,:));
ygmin=min(GPCOD(2,:));ygmax=max(GPCOD(2,:));
delta_xg=(xmax-xmin);delta_yg=(ymax-ymin);
gauss=zeros(NTOTG,1);
for KGAUS=1:NTOTG
    gauss(KGAUS)=text(xg(KGAUS),yg(KGAUS),int2str(KGAUS),'color','r');
end
set(gauss,'visible','off')

% vector das velocidades (f.potenc.)
veloc=quiver(eixos,GPCOD(1,:),GPCOD(2,:),VELOC(1,:),VELOC(2,:), 'r');

% gradiente hidráulico
grad=quiver(GPCOD(1,:),GPCOD(2,:),GRADH(1,:),GRADH(2,:), 'g');
set(grad,'visible','off')

% criação da malha que aproxima os pontos

for IPOIN = 2:NPOIN
    for JPOIN = 1:IPOIN-1
        if COORD(1,IPOIN) == COORD(1,JPOIN);
            if COORD(2,IPOIN) == COORD(2,JPOIN);
                COORD(1,IPOIN)=COORD(1,IPOIN)+10E-3;
            end
        end
    end
end

tx=ordenadas(COORD(1,:),NPOIN,'ascend');
ty=ordenadas(COORD(2,:),NPOIN,'descend');
[XI,YI]=meshgrid(tx,ty);

linhas equipotenciais
if NNODE == 8
    [COORD,HNODL]=malha8(COORD, HNODL, LNODS, NDIME, NELEM, NNODE,1);
end
PI=griddata(COORD(1,:),COORD(2,:),HNODL,XI,YI);
[PI,flag_PI]=check(PI);
texto=cell(3,1);
if flag_PI==1
    texto(1)={['Valor Constante = ',num2str(PI(1)),' m']};
end
[C1,rede1]=contour(XI,YI,PI,7,'b');

linhas de corrente
if NNODE == 8
    [COORD,XVECT]=malha8(COORD, XVECT, LNODS, NDIME, NELEM, NNODE,2);
end

CI=griddata(COORD(1,:),COORD(2,:),XVECT,XI,YI);
[CI,flag_CI]=check(CI);

```

```

if flag_CI==1
    texto(2)={'Valor Constante = ',num2str(CI(1)), ' m^3/s'}};
end
[C2, rede2]=contour(XI,YI,CI,7,'b');
rede=[rede1, rede2];
set(rede, 'visible', 'off', 'tag', 'rede')
set(rede, 'HandleVisibility', 'callback')

% colorir os potenciais
col_pot=pcolor(XI,YI,PI);
shading interp
set(col_pot, 'tag', 'col_pot', 'visible', 'off')

% colorir a funcao de corrente
col_corr=pcolor(XI,YI,CI);
shading interp
set(col_corr, 'tag', 'col_corr', 'visible', 'off')

% colorir altura piezométrica
PZI=PI-YI;
[C3, h_nivel]=contour(XI,YI,PZI);
set(h_nivel, 'visible', 'off')

[PZI, flag_PZI]=check(PZI);
col_piez=pcolor(XI,YI,PZI);
shading interp
set(col_piez, 'tag', 'col_piez')
set(eixos, 'title', text('string',...
    '\bfMapa de Cores da Altura Piezométrica'))
set(get(eixos, 'title'), 'fontsize', 15)
shading interp
barra_cores=colorbar;
set(barra_cores, 'tag', 'barracores')

lista=get(h_nivel, 'levellist');
if sum(lista==0)>0
    [xnivel, ynivel]=nivel0(C3);
    nivel=plot(xnivel, ynivel, 'linewidth', 2, ...
        'tag', 'nivel', 'visible', 'off');
end

if flag_PZI==1
    texto(3)={'Valor Constante = ',num2str(PZI(1)), ' m'}};
    set(get(eixos, 'ylabel'), 'string', texto(3), 'visible', 'on')
    children=get(barra_cores, 'children');
    barra_cores=[barra_cores, children];
    set(barra_cores, 'visible', 'off')
end

filhos_eixos=get(eixos, 'children');
set(eixos, 'children', [filhos_eixos(5:end); filhos_eixos(1:4)])

frame_check_box=uicontrol(output_graf, 'style', 'frame', 'units', ...
    'normalized', 'position', [.75, .45-.02-.05, .25, .55+.02+.05]);
check_box_nos=uicontrol(output_graf, 'style', 'checkbox', 'units', ...
    'normalized', 'position', [.8, .95-.02, .19, .05], 'string', 'Nós', ...
    'callback', 'check_box(check_box_nos, nos)', 'value', 1);

```

```

check_box_malha=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.9-.02,.19,.05],'string','Malha',...
    'callback','check_box(check_box_malha,malha)','value',1);
check_box_condfp=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.85-.02,.19,.05],'string','Cond. Front.
P.',...
    'callback','check_box(check_box_condfp,condfp)');
check_box_condfc=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.8-.02,.19,.05],'string','Cond. Front.
C.',...
    'callback','check_box(check_box_condfc,condfc)');
check_box_num_nos=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.75-.02,.19,.05],'string','Num. nós',...
    'callback','check_box(check_box_num_nos,num_nos)');
check_box_elem=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.7-.02,.19,.05],'string','Num. elem.',...
    'callback','check_box(check_box_elem,elem)');
check_box_gauss=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.65-.02,.19,.05],'string','Num. Gauss',...
    'callback','check_box(check_box_gauss,gauss)');
check_box_veloc=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.6-.02,.19,.05],'string','Vect. Veloc.',...
    'callback','check_box(check_box_veloc,veloc)','value',1);
check_box_grad=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.55-.02,.19,.05],'string','Vect. Grad.',...
    'callback','check_box(check_box_grad,grad)');
check_box_nivel=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.5-.02,.19,.05],'string',...
    'Nível
Freático','callback','check_box(check_box_nivel,nivel)','tag',...
    'check_box_nivel','enable','off');
check_box_rede=icontrol(output_graf,'style','checkbox','units',...
    'normalized','position',[.8,.5-.02-.05,.19,.05],'string','Rede
Fluxo',...
    'callback','check_box(check_box_rede,rede)','tag','check_box_rede',...
    'enable','off');

if ~isempty(findobj(eixos,'tag','nivel'))
    set(check_box_nivel,'enable','on')
end

x=.05;

frame_radio=icontrol(output_graf,'style','frame','units',...
    'normalized','position',[.75,.07,.25,.38-x]);
texto_radio=icontrol(output_graf,'style','text','units',...
    'normalized','position',[.76,0.36-x,.1,.05],'string','Coloração',...
    'fontsize',10,'fontweight','bold');
ncores=icontrol(output_graf,'style','edit','units',...
    'normalized','position',[.87,0.37-x,.05,.05],'BackgroundColor','w',...
    'TooltipString','nº de cores da coloração');
ok_ncores=icontrol(output_graf,'Units','normalized','string','ok',...
    'Position',[.93 .37-x .05 .05],...

'callback','colormap(jet(str2num(get(ncores,'string')))),barra_cores=colo
rbar;;set(barra_cores,'tag','barracores)');
h_radio=zeros(4,1);
h_radio(1)=icontrol(output_graf,'style','radio','units',...

```

```

        'normalized','position',[.8,0.25,.19,.05],'string','Potencial',...
        'callback','fradio(h_radio,1,flag_PI,texto)');
h_radio(2)=uicontrol(output_graf,'style','radio','units',...
    'normalized','position',[.8,0.2,.19,.05],'string','Corrente',...
    'callback','fradio(h_radio,2,flag_CI,texto)');
h_radio(3)=uicontrol(output_graf,'style','radio','units',...
    'normalized','position',[.8,0.15,.19,.05],'string','Alt. Piez',...
    'callback','fradio(h_radio,3,flag_PZI,texto)','value',1);
h_radio(4)=uicontrol(output_graf,'style','radio','units',...
    'normalized','position',[.8,0.1,.19,.05],'string','Sem Coloração',...
    'callback','fradio(h_radio,4,0,0)');
save_pushbutton=uicontrol(output_graf,'string','save','units',...
    'normalized','position',[.75,0,.125,.07],'fontweight','bold',...
    'callback','salvar(output_graf,eixos,botoes)');
print_pushbutton=uicontrol(output_graf,'string','print','units',...
    'normalized','position',[.875,0,.125,.07],'fontweight','bold',...
    'callback','imprimir(output_graf,eixos,botoes)');

botoes=get(output_graf,'children');
botoes=botoes(1:22);

```

```

%***  salvar.m  ****
%
%***  grava o conteúdo da janela de resultado
%
%****  2010.06.30 * LR **** quasar_m ****

function salvar(figura,imagem,menu)
[nome,pathname] = uiputfile(...
    {'*.bmp'; '*.jpeg'; '*.pdf'},...
    'Guardar Imagem');
set(menu,'visible','off')
posicao=get(imagem,'outerposition');
set(imagem,'outerposition',[0 0 1 1])

print(nome,...
    ['-d',nome(strfind(nome,'.')+1:end)],...
    ['-f',num2str(figura)])

set(imagem,'outerposition',posicao)
set(menu,'visible','on')

```