CONFIDENCIAL

MINISTÉRIO DAS OBRAS PÚBLICAS, TRANSPORTES E COMUNICAÇÕES

Laboratório Nacional de Engenharia Civil

PROGRAMA MBE – PROGRAMA DE CÁLCULO AUTOMÁTICO DE BARRAGENS DE ATERRO PELO MÉTODO DOS ELEMENTOS FINITOS

RELATÓRIO 20/92 – NF

MINISTÉRIO DAS OBRAS PÚBLICAS, TRANSPORTES E COMUNICAÇÕES

Laboratório Nacional de Engenharia Civil

NÃO CONFIDENCIAL

DEPARTAMENTO DE GEOTECNIA
Núcleo de Fundações

Proc. 053/13/9212

PROGRAMA MBE – PROGRAMA DE CÁLCULO AUTOMÁTICO DE BARRAGENS DE ATERRO PELO MÉTODO DOS ELEMENTOS FINITOS

RELATÓRIO 20/92 – NF

Lisboa, Fevereiro de 1992

I&D
GEOTECNIA

Copyright © LABORATÓRIO NACIONAL DE ENGENHARIA CIVIL, I. P.
AV DO BRASIL 101 • 1700-066 LISBOA
e-mail: lnec@lnec.pt
www.lnec.pt

Digitalizado no Setor de Divulgação Científica e Técnica do LNEC

2022

PROGRAMA MBE - PROGRAMA DE CÁLCULO AUTOMÁTICO DE BARRAGENS DE ATERRO PELO MÉTODO DOS ELEMENTOS FINITOS

RESUMO

O presente relatório apresenta sumariamente a motivação para a realização dos programas MBE e MEFSG e descreve a metodologia seguida na sua elaboração. É apresentada a formulação matemática geral para os elementos finitos utilizados no programa, bem como a descrição dos modelos reológicos considerados. Descrevem-se sumariamente todas as subrotinas utilizadas nos programas e apresentam-se as instruções para a construção dos ficheiros de dados e ficheiros auxiliares necessários para o funcionamento do programa. São ainda apresentadas as listagens dos programas.

PROGRAM MBE - FINITE ELEMENT PROGRAM FOR FILL DAMS

ABSTRACT

This report describes the motivation behind the MBE and MEFSG programs and presents the methodology emploied. The finite element mathematics is explained and the reological models are described. Every subroutine is briefly related as well as the instructions needed for the data files. The programs listings are included in appendix.

PROGRAMME MBE - PROGRAMME DE CALCUL AUTOMATIQUE DES BARRAGES EN REMBLAIS PAR LA MÉTHODE DES ÉLÉMENTS FINITS

RÉSUMÉE

Dans ce rapport on présente la motivation qui a menée à l'élaboration des programmes MBE et MEFSG, bien que la methodologie suivie. La formulation des éléments finis et les modèles rhéologiques utilisés sont décrits. Chaque routine est expliquée d'une forme sommaire et on présente les instructions pour l'élaboration des fichiers de données et fichiers auxiliaires. Les programmes sont présentés en annexe.

ÍNDICE DE TEXTO

ÍNDICE DE FIGURAS

1 - INTRODUÇÃO

O programa MBE (Modelação de Barragens de Enrocamento) surge na sequência da experiência e necessidade do Núcleo de Fundações em realizar cálculos tensão-deformação pelo método dos elementos finitos de barragens de aterro, a fim de prever ou de explicar aspectos de comportamento verificados no decorrer da vida operacional das obras.

Os programas de cálculo até aqui utilizados, resultavam de sucessivas adaptações de programas consagrados na literatura da especialidade, apresentando por isso as vantagens e defeitos decorrentes desse facto.

As vantagens derivam do facto de tais programas incorporarem o resultado de vasta experiência nas técnicas de modelação de efeitos específicos das acções próprias de aterros (especialmente de barragens), nomeadamente os efeitos da sequência construtiva e do enchimento da albufeira. As deficiências são consequência do facto de serem programas de concepção antiquada, realizados por diferentes programadores, com diferentes estilos de programação resultando por consequência pouco estruturados e por isso muito sensíveis a qualquer alteração que se pretendesse introduzir.

No caso concreto dos programas que originaram a presente versão, os efeitos de construção e primeiro enchimento eram considerados em programas separados, limitando seriamente a sequência de acções a simular.

Essa foi aliás a primeira motivação para a elaboração de um novo programa genericamente designado por MBEV1.0 (Modelação de Barragens de Enrocamento Versão 1.0), cuja primeira grande alteração terá sido a consideração num só programa dos efeitos da sequência construtiva e do primeiro enchimento. Nesta versão apenas era considerado o modelo hiperbólico. Em seguida foram introduzidas diversas alterações resultando num historial do qual se apontam em seguida as fases mais importantes.

MBEV1.0 - Modelo hiperbólico, junção das sequências construtivas e de primeiro enchimento; elementos subparamétricos de 4 pontos nodais

MBEV2.0 - Modelo hiperbólico com elementos isoparamétricos quadrangulares de 8 pontos nodais

MBEV3.0 - Modelo hiperbólico com elementos isoparamétricos quadrangulares de 8 pontos nodais e elementos isoparamétricos quadrangulares de 3 pontos nodais

MBEV4.0 - MBEV3.0 + elementos isoparamétricos de junta com 6 pontos nodais e sem espessura

MBEV5.0 - MBEV4.0 + elementos isoparamétricos de junta com seis pontos nodais e com espessura finita

MBEV6.0 - MBEV5.0 + modelo EC-K0 em simultâneo ou alternativa com o modelo hiperbólico

O programa foi amplamente testado no âmbito de trabalhos e estudos que o NF tem desenvolvido, citando-se por exemplo os estudos tensão-deformação referentes as barragens de Borde Seco, Las Cuevas e Beliche.

Para complementar os resultados obtidos sob a forma numérica foi elaborado, um programa de exploração gráfica para obter um conjunto de diagramas que permitem uma melhor e mais rápida interpretação dos resultados do programa de cálculo.

O nome dado a este módulo de exploração gráfica foi MEFSG (Método dos Elementos Finitos Sistema Gráfico). Foi também alvo de diversas fases de aperfeiçoamento, correspondendo fundamentalmente a adição de novos módulos de possibilidades.

2 - DESCRIÇÃO GERAL DO PROGRAMA

2.1 - Generalidades

Uma das preocupações fundamentais sempre tida em mente no desenvolvimento do programa foi a de documentar ao máximo as instruções permitindo assim que, quer o autor quer quem no futuro possa vir a trabalhar com o programa, dispenda o mínimo de tempo para assimilar o seu funcionamento. Por outro lado a linguagem de programação (FORTRAN 77) foi utilizada procurando seguir sempre a respectiva norma, possibilitando assim garantir a portabilidade do programa a diversas máquinas.

2.2 - Elementos finitos utilizados. Formulação.

O programa contempla o uso de diversos tipos de elementos finitos de modo a fazer face a diversas situações de geometria e carregamento. Todos os elementos da presente versão são isoparamétricos com funções de forma do segundo grau, por serem reconhecidamente elementos poderosos e precisos. Além disso com os meios de cálculo de que actualmente se dispõe, não faz grande sentido o uso de elementos subparamétricos ou mesmo de elementos lineares isoparamétricos que visam diminuir o número de incógnitas sacrificando de algum modo a precisão da solução.

O programa dispõe, na presente versão de quatro tipos de elementos finitos. Assim para simulação de meios contínuos pode-se recorrer a elementos quadrangulares ou triangulares de oito e seis pontos nodais, respectivamente. Para a incorporação de descontinuidades físicas ou mecânicas (Mateus da Silva 1990), o programa dispõe de elementos finitos de junta, com ou sem espessura, ambos com seis pontos nodais.

As secções seguintes apresentam a formulação matemática geral e de cada tipo de elemento referindo-se entre parêntesis quais os nomes das subrotinas correspondentes ou directamente relacionadas com cada expressão. A formulação apresentada é desenvolvida para equilíbrios planos com base na estacionaridade da energia potencial total do sistema, partindo-se da interpolação dos deslocamentos no interior e na fronteira do elemento em função dos seus valores nos pontos nodais (Sousa, 1980). Toda formulação é apresentada em notação indicial que se manifesta extremamente conveniente para a incorporação num

programa de cálculo automático (Pedro, 1973 e 1977) permitindo a sua programação quase directa.

2.2.1 - Formulação geral dos elementos finitos utilizados no programa. Elementos finitos quadrangulares e triangulares.

Considerem-se os sistemas de eixos globais e locais (cartesianos, y_n e de área, L_n) apresentados na Fig. 1. As coordenadas dum ponto no interior ou fronteira do elementos são definidas com base nas coordenadas dos pontos nodais e em funções de interpolação (N_i) por:

$$\begin{gathered} x_k=N_i\, x_{ik}^e \\ N_i=N_i(y_n) \\ N_i=N_i(L_n) \\ i=1,2,..NP \\ k=1,2 \\ n=1,2 \quad ou \quad n=1,2,3 \end{gathered} \tag{1}$$

Porque os elementos são isoparamétricos, as funções de interpolação dos deslocamentos U_m de um ponto qualquer do elemento são idênticas às funções de interpolação das coordenadas pelo que os primeiros são definidos por:

$$\begin{gathered} U_m=N_i U_{im}^e \\ i=1,2,..NP \\ m=1,2 \end{gathered} \tag{2}$$

A transformação entre os sistemas de coordenadas locais e globais é obtida pela diferenciação (DER8, TRANJ8) da Eq. (1) ficando:

$$\begin{gathered} dx_k=\left(\frac{\partial N_i}{\partial y_n}x_{ik}^e\right)dy_n=J_{kn}dy_n \\ k,n=1,2 \\ i=1,2,..,NP \end{gathered} \tag{3}$$

no caso do elemento rectangular e, para o elemento triangular (DER6, TRANJ6) ficará:

$$\begin{gathered} dx_k=\left(\frac{\partial N_i}{\partial L_n}x_{ik}^e\right)dy_n=J_{kn}dL_n \\ k=1,2 \\ n=1,2,3 \\ i=1,2,..,NP \end{gathered} \tag{4}$$

As relações inversas são, para o elemento rectangular:

$$dy_n = J^{-1}_{nk} dx_k = Y_{nk} dx_k$$

$$k,n=1,2 \qquad (5)$$

$$Y_{nk} = \frac{(-1)^{n+k}}{\phi} J_{3-k,3-n}$$

$$\phi = J_{11}J_{22} - J_{12}J_{21}$$

e para o elemento triangular:

$$dL_n = Y_{nk} dx_k$$

$$k=1,2 \qquad (6)$$

$$n=1,2,3$$

onde:

$$Y_{nk} = \frac{(-1)^{3-k}}{\phi}(J_{3-k,m} - J_{3-k,p})$$

$$\phi = \sum_{i=1}^{i=3} (J_{1r}J_{2s} - J_{1s}J_{2r}) \qquad (7)$$

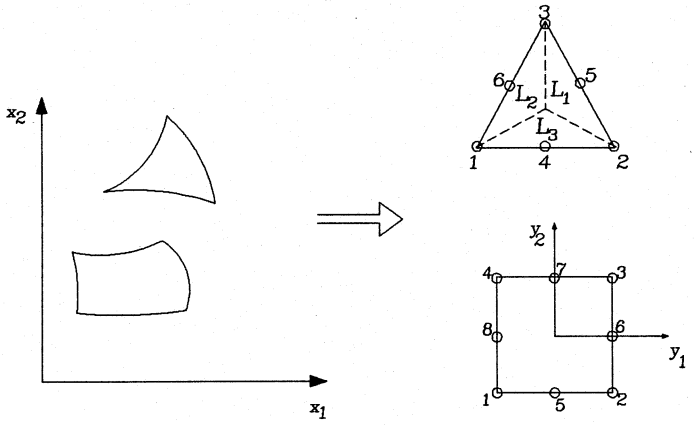
$$j=1,2,..,NP$$

$$n=1,2$$

$$p,q=1,2,3$$

$$r=2,3,1$$

$$s=3,2,1$$



Referencial global *Referenciais locais*

Fig. 1 - Elementos finitos do programa MBE. Referenciais locais e global

As deformações são obtidas pelo operador diferencial F_{pm} aplicado ao vector de deslocamentos resultando (FEM8,FEM6,FEMP):

$$\varepsilon_p = F_{pm} U_m = F_{pm} N_i U^e_{im} = A_{pim} U^e_{im}$$

$$i=1,2,..,NP \qquad (8)$$

$$p=1,2,3$$

$$m=1,2$$

O campo de tensões relaciona-se com o campo de deslocamentos, deformações iniciais e tensões iniciais por (TENSÃO):

$$\sigma_p = D_{pq}\varepsilon_q - D_{pq}\varepsilon^0_q + \sigma^0_p = D_{pq} A_{qjn} U^e_{jn} - D_{pq}\varepsilon^0_q + \sigma^0_p$$

$$j=1,2,..,NP \qquad (9)$$

$$n=1,2$$

$$p=1,2,3 \quad q=2,3,1$$

As condições de estacionaridade da energia potencial total permitem obter as equações de equilíbrio do elemento (RESOL):

$$K^e_{ijmn} U^e_{jn} = Q^e_{im} + P^e_{im} + S^e_{im} + T^e_{im}$$

$$i,j=1,2,..,NP \qquad (10)$$

$$m,n=1,2$$

nestas equações, o termo $K^e_{i\,jmn}$ representa a matriz de rigidez do elemento e, os restantes termos Q_{im}, P_{im}, S_{im} e T_{im} representam as forças nodais correspondentes a: forças mássicas, forças de superfície, deformações iniciais e tensões iniciais. As expressões gerais destes termos são (FEM8,FEM6,FEMP,FOVOL8,FOVOL6,FOVOLP,FOTE8,FOTE6,FOTEP,FOSUP8, FOSUP6,FOSUPP) :

$$K^e_{inmn} = \int_V A_{pim} D_{pq} A_{qjn} dV$$

$$Q^e_{im} = \int_V N_i Q_m dV$$

$$P^e_{im} = \int_S N_i P_m dS \qquad (11)$$

$$S^e_{im} = \int_V A_{pim} D_{pq} \varepsilon^0_q dV$$

$$T^e_{im} = \int_V A_{pim} \sigma^0_p dV$$

$$i,j=1,2,..,NP$$
$$m,n=1,2$$
$$p,q=1,2,3$$

A integração das Eq. (11) é efectuada numericamente sendo possível escolher a ordem de integração em função da complexidade do problema em estudo.

Para finalizar a formulação dos elementos utilizados apresentam-se seguidamente as funções de forma N_i para os elementos rectangulares e para os elementos triangulares. Assim, para o primeiro tipo de elementos tem-se (FUFOR8):

$$N_i=\frac{1}{4}(1-y_{i1}y_1)(1+y_{i2}y_2)(y_{i1}y_1+y_{i2}y_2-1)$$
$$i=1,2,3,4$$
$$N_i=\frac{1}{2}(1-y_{3-p}^2(1+y_{ip}y_p)y_{ip}^2 \qquad (12)$$
$$i=5,6,7,8$$
$$p=1,2$$

enquanto que, para o elemento triangular (FUFOR6) :

$$N_i=(2L_i-1)L_i$$
$$N_{i+3}=4L_iL_j \qquad (13)$$
$$i=1,2,3$$
$$j=2,3,1$$

2.2.2 - Elemento finito de junta sem espessura. Formulação.

A Fig. 2 apresenta o elemento de junta de espessura nula nos sistemas de coordenadas globais (x_1,x_2) e de coordenadas locais (y_1,y_2).

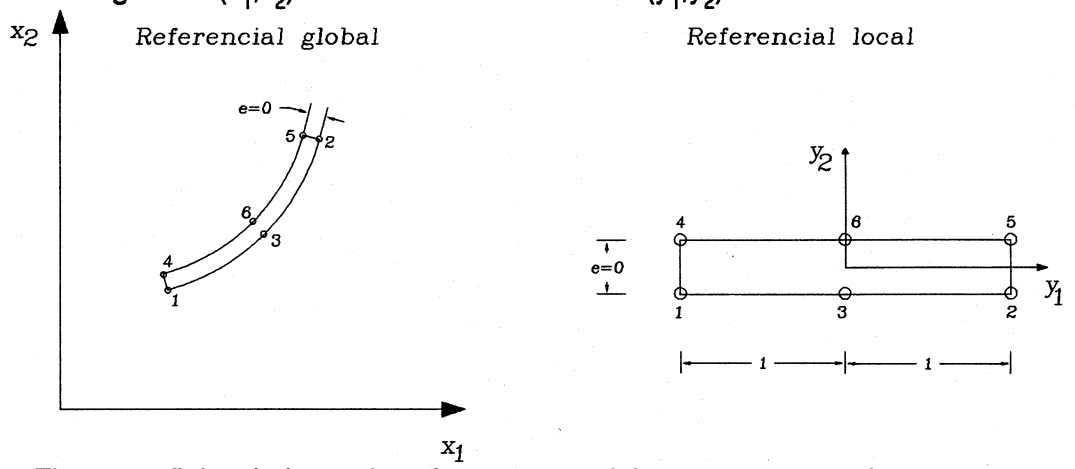


Fig. 2 - Elemento finito de junta de seis pontos nodais e espessura nula

Como as funções de interpolação são do segundo grau, a forma do elemento no sistema global pode ser curva, permitindo fazer face a geometrias complexas.

As funções de forma em coordenadas naturais são (FUFORJ):

$$N_i=\frac{1}{2}y_{i1}y_1(1+y_{i1}y_1)$$
$$i=1,2,4,5 \qquad (14)$$
$$N_i=1-y_1^2$$
$$i=3,6$$

onde y_{i1} representa as coordenadas dos pontos nodais segundo o eixo local y_1, (y_{i2}=0 em qualquer ponto) e y_1 as coordenadas do ponto onde se pretende conhecer N_i.

A posição de qualquer ponto do elemento pode ser definida em função das coordenadas globais dos pontos nodais e das funções de forma do elemento de um modo

análogo ao definido na Eq.(1) por:

$$(x_m)_k = N_i (x_{im}^e)_k \qquad i=1,..,6 \tag{15}$$

onde o índice m define o eixo coordenado, k a face considerada (face superior=1, inferior=2) e o índice i define a soma para a contribuição global dos 6 nós do elemento.

Os deslocamentos em qualquer ponto do elemento são definidos pelas mesmas funções de interpolação que foram usadas para as coordenadas dos pontos nodais (elemento isoparamétrico) obtendo-se a seguinte expressão para estes deslocamentos, expressa em termos dos deslocamentos no referencial local (U´):

$$(U'_m)_k = N_i (U'^e_{im})_k \qquad i=1,2,3 \quad m,k=1,2 \tag{16}$$

As deformações, para um elemento de espessura nula, são definidas pelos deslocamentos relativos entre os pontos nodais de iguais coordenadas da face superior e da face inferior:

$$\epsilon'_m = (U'_m)_2 - (U'_m)_1 = N_i [(U'^e_{im})_2 - (U'^e_{im})] \qquad m=1,2 \quad i=1,2,3 \tag{17}$$

onde ϵ'_m (m=1,2) representa as deformações tangencial e normal respectivamente.

De acordo com a ordenação dos pontos nodais apresentada na Fig. 2, a expressão anterior pode ser escrita de uma forma mais condensada:

$$\epsilon'_m = (-1)^r N_j U'^e_{jm} \qquad j=1,..,6 \quad m=1,2 \quad r=j/4+1 \tag{18}$$

A passagem do referencial global para o referencial local é efectuada através de uma matriz de transformação de coordenadas. Essa matriz T_{mn} é definida a partir do conhecimento da direcção dos vectores tangencial e normal à direcção do elemento. O vector tangente é definido por:

$$\vec{t} = \frac{dx_k}{dy_1}\vec{e}_k = \frac{dx_1}{dy_1}\vec{e}_1 + \frac{dx_2}{dy_1}\vec{e}_2 \tag{19}$$

As derivadas dx_k/dy_1 podem ser expressas em termos das coordenadas dos pontos nodais e das funções de forma (Eq. (15)):
nestas condições, a direcção da tangente (normalizada) será:

$$\vec{t}_1 = \frac{(J_{11}\vec{e}_1 + J_{12}\vec{e}_2)}{\phi} \tag{20}$$

$$\text{com} \quad \phi = \sqrt{J_{11}^2 + J_{12}^2}$$

A direcção da normal t_2, pode ser obtida atendendo à definição de produto interno para vectores perpendiculares, resultando:

$$\vec{t}_1=\frac{(-J_{12}\vec{e}_1+J_{11}\vec{e}_2)}{\phi} \tag{21}$$

Deste modo a matriz de rotação entre o referencial global e o referencial local é (TRANJJ):

$$T=\frac{1}{\phi}\begin{bmatrix} J_{11} & J_{12} \\ -J_{12} & J_{11} \end{bmatrix} \tag{22}$$

É agora possível descrever as deformações no referencial local do elemento de junta em função dos deslocamentos no referencial global, ficando:

$$\begin{gathered} \epsilon'_m=(-1)^r\ T_{mn}N_jU^e_{jn} \\ j=1,..,6 \\ m,n,s=1,2 \\ r=j//4+1 \end{gathered} \tag{23}$$

As tensões normal e tangencial, expressas em termos dos deslocamentos dos pontos nodais são obtidas por (TENSÃO):

$$\begin{gathered} \sigma'_m=(-1)^r\ D_{mn}T_{ns}N_jU^e_{js} \\ j=1,..6 \\ m,n,s=1,2 \\ r=j//4+1 \end{gathered} \tag{24}$$

onde D representa a matriz de constantes elásticas para o elemento de junta. Esta matriz tem, no caso de junta dilatante, a seguinte forma (MATD,ELAWJ):

$$D_{mn}=\begin{bmatrix} K_T & K_{TN} \\ K_{NT} & K_N \end{bmatrix} \tag{25}$$

e, para o caso abrangido neste trabalho, de junta não dilatante, os termos da diagonal secundária são nulos ficando simplesmente:

$$D_{mn}=\begin{bmatrix} K_T & 0 \\ 0 & K_N \end{bmatrix} \tag{26}$$

A expressão da matriz de rigidez pode agora ser obtida pela minimização da energia potencial do sistema (Principio dos Trabalhos Virtuais) ficando (FEMJ):

$$\begin{gathered} K^e_{ijmn}=(-1)^t\int_{-1}^{+1} T_{mp}N_iD_{pq}T_{qn}N_j\phi dy_1 \\ i,j=1,..,6 \\ m,n,p,q=1,2 \\ t=i//4+j//4+2 \end{gathered} \tag{27}$$

2.2.3 - Elemento finito de junta com espessura. Formulação.

O elemento finito de junta com espessura pode, regra geral, ser tratado como se de um elemento normal se tratasse havendo porém o cuidado de na discretização da malha garantir que a espessura do elemento seja pequena relativamente à dimensão maior. Este procedimento, seguido por alguns autores (Griffiths, 1985; Desai et al., 1984), pode apresentar inconvenientes de ordem numérica pois, devido à precisão limitada dos cálculos em computador podem surgir com facilidade problemas de precisão numérica que poderão originar ou uma divisão por zero ou um afastamento considerável da solução considerada

como exacta.

Por este motivo, o elemento utilizado (apresentado na Fig. 3), é um elemento finito lagrangeano do tipo quadrado com seis pontos nodais, sendo quatro nos vértices, e dois nos lados opostos de maior dimensão, utilizando-se nestes lados funções de interpolação do 2º grau. Os outros dois lados (correspondentes à espessura da junta) apenas têm nós nos vértices sendo as funções de interpolação do 1º grau.

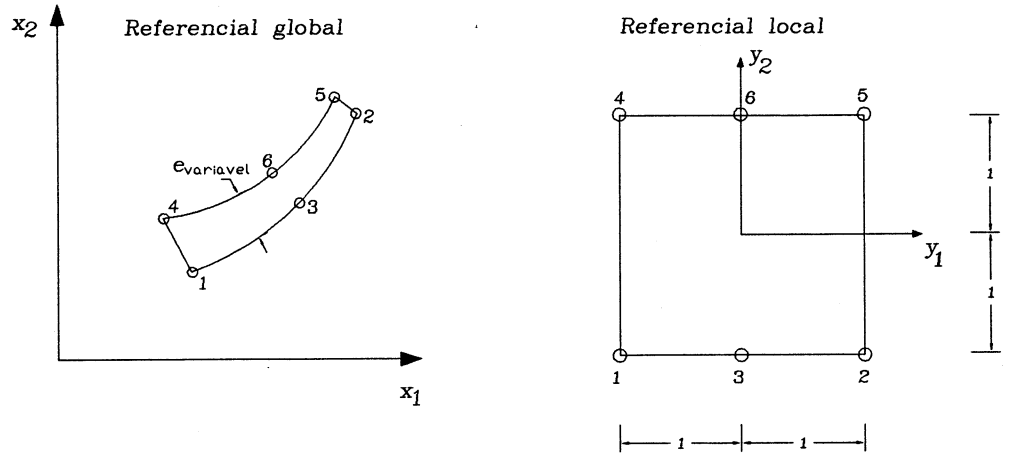


Fig. 3 - Elemento finito de junta com espessura

O facto de se reduzir o grau de interpolação visa fundamentalmente dois objectivos: em primeiro lugar, obviar os problemas de precisão numérica e, em segundo lugar, evitar o aumento desnecessário de incógnitas (não faz sentido calcular o deslocamento no ponto central já que esse deslocamento é indefinido).

As funções de forma deste elemento são (FUFORP):

$$N_i=\frac{1}{4}y_{i1}y_1(1+y_{i1}y_1)(1+y_{i2}y_2) \qquad i=1,2,4,5$$

$$N_i=\frac{1}{2}(1+y_1^2)(1+y_{i2}y_2) \qquad i=3,6 \qquad \textbf{(28)}$$

A restante formulação deste elemento (cálculo de deformações, tensões, matriz de rigidez e vector de forças) é perfeitamente análoga à formulação já apresentada para o elemento finito rectangular de 8 pontos nodais (Eqs. (1) a (11)), havendo apenas que alterar a variação dos índices em conformidade com o número de pontos nodais do elemento.

3 - MODELOS REOLÓGICOS

3.1 - Modelo hiperbólico para meio contínuo

O modelo hiperbólico, baseado numa proposta de Kondner (1963) é um dos modelos usados pelo programa para modelar o comportamento dos materiais constituintes dos aterros. Este modelo enquadra-se, como é sabido, nos modelos hipoelásticos permitindo modelar situações com dependência da trajectória de tensões.

Apresenta diversas vantagens que justificam plenamente a sua aplicação no âmbito da modelação das fases de construção e primeiro enchimento da albufeira :

- Ajuste razoável a situações de carregamento monotónico;

- Vasta experiência na sua aplicação, havendo na bibliografia inúmeros exemplos de aplicações bem sucedidas;

- A consideração de um critério de carga-descarga permite modelar o comportamento de descarga e recarga;

- Facilidade de programação e implementação num programa pelo método dos elementos finitos;

- Facilidade de obtenção de parâmetros, quer pela realização de simples ensaios triaxiais, quer pela consulta de bibliografia.

Toda a formulação do modelo se baseia no ajuste de duas curvas resultantes dos ensaios a curvas hiperbólicas do tipo :

$$y=\frac{x}{a+bx} \tag{29}$$

Surge assim a relação entre a tensão distorcional e a deformação principal máxima ε_1:

$$\sigma_1-\sigma_3=\frac{\varepsilon_1}{a+b\varepsilon_1} \tag{30}$$

e a relação entre as deformações principais máxima e mínima:

$$\varepsilon_1=\frac{\varepsilon_3}{f+d\varepsilon_3} \tag{31}$$

A obtenção dos parâmetros das curvas é efectuada pela linearização das curvas hiperbólicas e pela aplicação do método dos mínimos quadrados ou qualquer processo matemático ou gráfico equivalente. A linearização é feita reescrevendo a relação hiperbólica por:

$$\frac{x}{y}=a+bx \tag{32}$$

Partindo destas duas premissas, das relações básicas da elasticidade, de um critério de rotura, das propostas de alguns autores e ainda com base noutras simplificações obtém-se o conjunto de relações que definem na globalidade o modelo hiperbólico. Essas expressões são seguidamente descritas:

Variação do módulo de elasticidade inicial com a tensão de confinamento (Janbu, 1963):

$$E_i=K\,Pa\left(\frac{\sigma_3}{Pa}\right)^n \tag{33}$$

Critério de Mohr-Coulomb:

$$(\sigma_1-\sigma_3)_r=\frac{2c\cos\phi+2\sigma_3 sen\phi}{1-\sin\phi} \quad (34)$$

Coeficiente de rotura:

$$R_f=\frac{(\sigma_1-\sigma_3)_r}{(\sigma_1-\sigma_3)_u} \quad (35)$$

Módulo tangente para uma tensão $(\sigma_1-\sigma_3)$:

$$E_t=\left[1-\frac{(\sigma_1-\sigma_3)}{(\sigma_1-\sigma_3)_U}\right]^2 E_i \quad (36)$$

Variação do ângulo de atrito com a tensão de confinamento:

$$\phi=\phi_0-\Delta\phi\log\left(\frac{\sigma_3}{Pa}\right) \quad (37)$$

Variação do coeficiente de Poisson inicial com a tensão de confinamento:

$$\nu_i=G-F\log\left(\frac{\sigma_3}{Pa}\right) \quad (38)$$

Coeficiente de Poisson tangente:

$$\nu_t=\frac{\nu_i}{\left[1-\frac{d(\sigma_1-\sigma_3)}{E_i\left[1-\frac{(\sigma_1-\sigma_3)}{(\sigma_1-\sigma_3)_U}\right]}\right]^2} \quad (39)$$

Nas expressões anteriores e nas que se seguem, o parâmetro Pa simboliza a pressão atmosférica no sistema de unidades pretendido. A sua utilização nas expressões permite tornar os parâmetros adimensionais e por consequência, independente do sistema de unidades.

3.2 - Considerações sobre rotura

Em modelos elásticos, como no hiperbólico, existem dificuldades várias na simulação de comportamentos próximos ou em rotura. A primeira dificuldade é intrínseca ao processo de modelação pois o comportamento real do meio real nunca pode ser descrito com precisão por um conjunto de equações matemáticas mais ou menos determinísticas e usando ensaios que são por si só importantes simplificações da realidade. Por outro lado, e sob o ponto de vista da formulação matemática, às condições de rotura dos modelos elásticos correspondem parâmetros numéricos que resultam em dificuldades matemáticas de resolução de equações. Nestes modelos, o comportamento de rotura caracteriza-se por valores particulares dos parâmetros que materializam as relações constitutivas, no caso do par E,ν tem-se E->0 e/ou ν->0.5, traduzindo a situação de um material que a cada acréscimo de deformação não faz corresponder um acréscimo de tensão e/ou se passa a comportar a volume constante.

Qualquer uma das situações acima referidas pode levar nos seus limites a situações de instabilidade numérica traduzidas quer pela singularidade do sistema de equações quer

por uma considerável divergência dos valores da solução "correcta". De acordo com os problemas descritos os modelos matemáticos nas situações de rotura ou rotura eminente (SL=0.95 no programa MBE) limitam os valores dos parâmetros a intervalos que pela experiência de vários autores garantem um compromisso de estabilidade numérica com uma conveniente simulação do comportamento em rotura (dentro das limitações dos modelos elásticos).

No programa MBE esta situação é traduzida por:

$$E_t=(1-0.95\ R_f)^2 E_i$$

$$\nu_t=0.49$$

3.3 - Modelo EC-K0 para meio contínuo

O modelo EC-K0 enquadra-se, à semelhança do anterior, nos modelos elásticos não lineares. Neste caso, o ensaio que serve de base à determinação de parâmetros é o ensaio de compressão unidimensional (Veiga Pinto, 1983).

De entre as várias leis possíveis para descrever a relação tensões-deformações adoptou-se uma lei geométrica do tipo:

$$\sigma_1=AE\ Pa\ \varepsilon_1^{BE} \qquad (41)$$

esta lei, consoante o valor adoptado para BE pode assumir andamentos parabólicos relativamente ao eixo de tensões (BE=2) ou deformações (BE=1/2), ajustando-se assim a diversos tipos de comportamentos, nomeadamente para materiais de enrocamento. A determinação dos parâmetros da curva pode ser efectuada gráfica ou numericamente recorrendo-se normalmente à linearização da curva. Por diferenciação da equação anterior pode-se deduzir a variação do módulo tangente (EC) em função da tensão axial:

$$EC=\frac{d\sigma}{d\varepsilon}=AE\ BE\ Pa\ \varepsilon_1^{BE-1}=AE\ BE\ Pa\left(\frac{\sigma_1}{Pa\ AE}\right)^{\frac{BE-1}{BE}}$$

A simulação de deformações irreversíveis em ciclos de descarga-recarga é conseguida pela consideração de um módulo edométrico (ECDR) superior nestas fases de comportamento, definindo-se então um terceiro parâmetro por:

$$EC_{DR}=CE\ Pa \qquad (43)$$

Para completar a lei, falta estabelecer a relação entre as tensões σ_1 e σ_3. Em condições de confinamento. O coeficiente de impulso em repouso traduz esta relação. A lei de variação de K_0 é formalmente idêntica para as situações de carga ou de descarga--recarga, diferindo apenas nos valores numéricos dos parâmetros. A lei adoptada é para carga:

$$K_0=AK_0+BK_0\left(\frac{\sigma_1}{Pa}\right) \qquad (44)$$

e, para descarga-recarga:

$$K_0 = CK_0 + DK_0\left(\frac{\sigma_1}{Pa}\right) \qquad (45)$$

Estas equações completam a descrição do modelo, resultando um conjunto de 7 parâmetros.

3.4 - Modelos para meios descontínuos.

3.4.1 - Modelo reológico para juntas sem espessura.

A forma típica das curvas tensão tangencial-deformação obtidas dos ensaios de corte directo comparada com a forma geral de uma curva hiperbólica sugere que os modelos baseados neste tipo de curvas sejam também adequados para representar o comportamento dos elementos de junta. Esta constatação foi já apontada por diversos autores (Sharma et al.,1979) e (Desai, 1981). A adopção deste modelo no contexto do programa MBEV60 tem ainda a vantagem de permitir a sua programação no fluxo normal do programa.

Considere-se então uma série de ensaios de corte directo, cujo diagrama típico se apresenta na Fig. 4 . Nesta figura e em ordenadas estão representados os valores da tensão tangencial (τ) e em abcissas os valores da deformação de corte (medida pelo deslocamento relativo da parte superior e inferior da caixa de corte). As curvas 1,2 e 3 representam a evolução das tensões tangenciais para as tensões normais σ_{n1}, σ_{n2}, σ_{n3}.

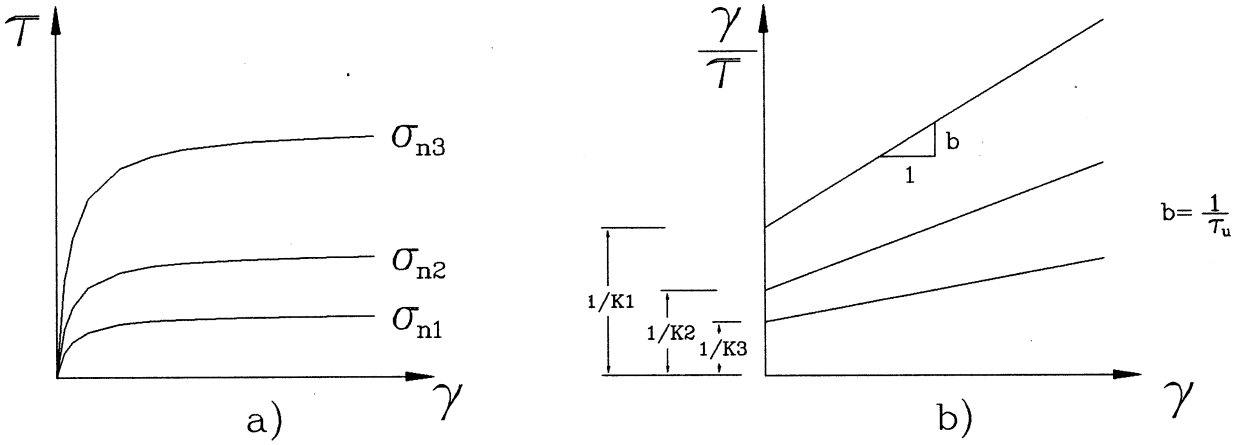


Fig. 4 - Curvas típicas obtidas num ensaio de corte directo e suas transformadas

Constata-se na prática (Mateus da Silva, 1990) que cada uma destas curvas se ajusta com razoável precisão a uma lei do tipo:

$$\tau = \frac{\gamma}{a + b\gamma} \qquad (46)$$

onde a e b representam parâmetros dependentes da tensão normal σ_n. O seu significado físico pode ser facilmente compreendido pela análise da Fig. 4 ou pela linearização da Eq. (46) na seguinte forma:

$$\frac{\gamma}{\tau} = a + b\gamma \qquad (47)$$

Assim, fazendo por mera consideração matemática a deformação tender para infinito na Eq. (46), obtém-se:

$$\lim_{\gamma\to\infty}\tau=\lim_{\gamma\to\infty}\frac{\gamma}{a+b\gamma}=\frac{1}{b} \qquad (48)$$

"1/b" representa portanto, a tensão tangencial última (habitualmente designada por τ_u) obtida para deformações infinitas. O significado do parâmetro "a" pode ser retirado da linearização da curva hiperbólica expressa na Eq. (48) e representada na Fig. 4. De acordo com esta figura, "a" representa o inverso de um módulo de deformabilidade obtido para um nível de deformação nulo, isto é, a deformabilidade inicial do material.

Assim a Eq. (46) pode agora ser reescrita com completo significado físico dos seus parâmetros da seguinte forma:

$$\tau=\frac{\gamma}{\frac{1}{K_{Ti}}+\frac{1}{\tau_u}\gamma} \qquad (49)$$

Nesta equação assenta a base do modelo hiperbólico pois permite descrever as evolução da tensão com a evolução da deformação.

À semelhança do proposto por Janbu, (1963) para a variação do módulo de elasticidade inicial no modelo hiperbólico baseado em ensaios triaxiais, a variação do módulo inicial K_{Ti} pode ser convenientemente descrita por uma relação exponencial da seguinte forma:

$$K_{Ti}=K_J\gamma_w\left(\frac{\sigma_n}{Pa}\right)^n \qquad (50)$$

na qual:

K_{Ti} - rigidez tangencial inicial;
σ_n - tensão normal ao plano de corte;
K_J,n - parâmetros do modelo;
Pa - pressão atmosférica
γ_w - peso volúmico da água.

A pressão atmosférica e o peso volúmico da água são normalmente considerados nestas relações para as tornar adimensionais. A aplicação de logaritmos à Eq. (50) permite transformar esta curva numa recta facilitando assim a obtenção dos parâmetros por aplicação de um algoritmo numérico ou por construção gráfica.

Para estabelecer a variação da tensão tangencial última (τ_u) na Eq. (50) recorre-se ao critério de rotura de Mohr-Coloumb, e a um coeficiente de rotura R_f[1]. Assim tem-se:

$$\tau_R=c+\sigma_n\tan\phi \qquad (51)$$

e

$$R_f=\frac{\tau_R}{\tau_u} \qquad (52)$$

---

[1] A designação consagrada na literatura "R_f" deriva da expressão anglo saxónica "failure ratio"

A importância do parâmetro R_f definido na Eq. (52) prende-se com a constatação de que, na maioria dos casos, a curva hiperbólica tende assimptoticamente para um valor de tensão tangencial superior ao valor da tensão tangencial máxima suportada pelo material, e dada pela Eq. (51). O parâmetro R_f é por esse motivo um indicador do nível de concordância do modelo reológico à curva real obtida no ensaio.

Para descrever a rigidez tangente em qualquer ponto da curva hiperbólica é necessário proceder à diferenciação da Eq. (46) ficando:

$$\frac{d\tau}{d\gamma}=K_T=\frac{d}{d\gamma}\left(\frac{\gamma}{a+b\gamma}\right) \tag{53}$$

que simplificando resulta em:

$$K_T=\left(1-\frac{\tau}{\tau_u}\right)^2 K_{Ti} \tag{54}$$

A Eq. (54) permite obter em qualquer momento do processo de cálculo o módulo de rigidez tangente da junta até ao momento em que é atingida uma situação de rotura incipiente. Neste modelo, a condição de rotura origina um módulo K_T nulo. Ora como matematicamente essa condição provoca erros numéricos na resolução do sistema de equações global da estrutura, é costume limitar a definição de rotura a um nível de tensão elevado (95% do valor de rotura) mas diferente da unidade. Nestas condições o módulo tangente é:

$$K_{T_{(rotura)}}=(1-0.95)^2 K_{Ti} \tag{55}$$

Para finalizar a descrição do modelo falta estabelecer o modo de variação de K_N (rigidez normal) com a tensão correspondente. Sharma (1979) estabelece um aumento arbitrário da rigidez normal com a tensão σ_n. Outros autores invocando o facto de não haver sobreposição física dos materiais adjacentes à junta, atribuem à rigidez normal um valor elevado. Foi este o procedimento adoptado para o presente modelo. Assim a rigidez normal da junta é constante excepto quando são atingidas tensões normais de tracção ($\sigma_n<0$), caso em que se assume arbitrariamente para K_T e K_N um valor baixo, correspondente à abertura da junta.

Como já se referiu, em cálculos não lineares é de extrema importância a capacidade exibida pelo modelo de memorizar os estados de tensão anteriores e condicionar o seu comportamento de acordo com esses estados.

No modelo hiperbólico aplicado a estados de compressão triaxial (com simetria radial), essa memorização é conseguida pela consideração de uma medida do afastamento do estado de tensão relativamente à rotura. Mais precisamente, essa medida designada habitualmente por SL[2] e cuja interpretação se pode fazer da análise da Fig. 5, é memorizada para cada elemento no decorrer do processo incremental de cálculo, determinando assim os comportamentos de primeiro carregamento, rotura ou descarga-recarga (Veiga Pinto, 1983).

Para ensaios de corte adopta-se um procedimento que, em traços gerais, é análogo. Assim, considere-se num diagrama T_u-σ_n, um ponto P, representativo do estado de tensão numa determinada fase do processo de cálculo (Fig. 6 e Fig. 5, Fig. 7).

---

[2] SL da expressão anglo-saxónica "stress-level"

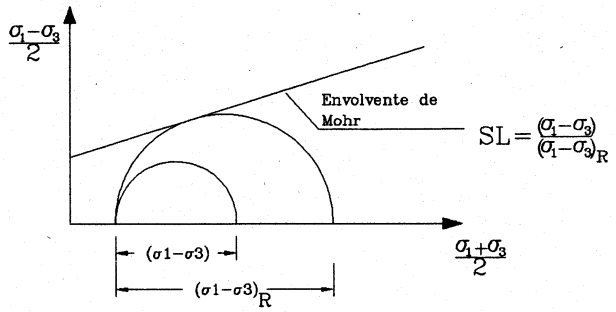


Fig. 5 - Envolvente de Mohr e definição de nível de tensão (SL) para estados de compressão triaxiais com simetria radial

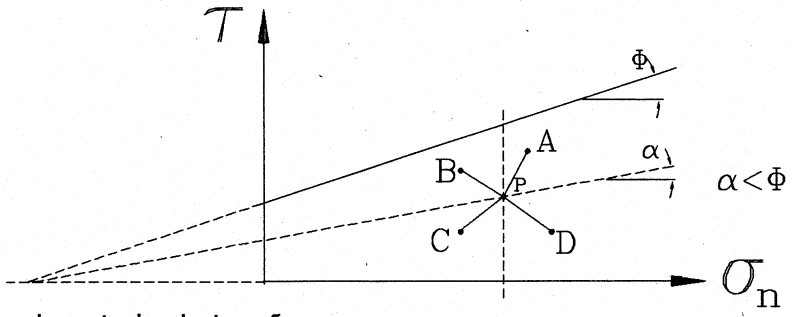


Fig. 6 - Evolução do estado de tensão no espaço τ-σ_n

Uma posterior modificação do estado de tensão pode levar o ponto P a ocupar, no espaço de tensões, uma posição qualquer A, B, C ou D seguindo qualquer uma das trajectórias indicadas na Fig. 6. Ora o comportamento exibido é uma função do percurso assumido.

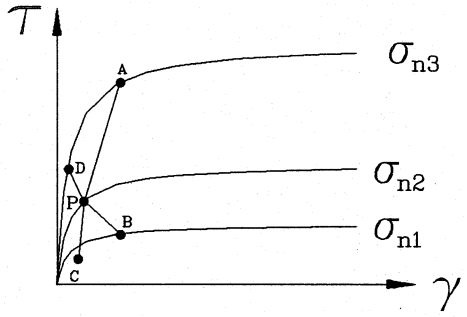


Fig. 7 - Comportamento assumido do espaço τ-π

Para os percursos P-A e P-B, o comportamento assumido é o de primeiro carregamento já que a posição final (A ou B) se encontra sempre mais perto da envolvente de rotura e portanto com um nível de tensão SL agora definido como:

$$SL = \frac{\tau}{\tau_u} \qquad (56)$$

superior ao anterior.

No caso P-A o ponto experimenta um aumento simultâneo da tensão normal e tangencial sendo que esta última aumenta mais do que $\delta\sigma_n \tan(\alpha)$, (com $\alpha<\phi$) pelo que o ponto se aproxima da situação de rotura. Na trajectória P-B a par da variação (aumento ou diminuição) da tensão tangencial há uma redução da tensão normal, situando-se o ponto

final mais próximo da envolvente.

O caso P-C corresponde a haver simultaneamente uma redução da tensão normal e tangencial sendo a redução desta última superior ao valor definido pela linha de inclinação tan(α) pelo que se processa uma descarga cuja importância é determinada pela redução da tensão normal. Neste caso o comportamento típico é o resultante de assumir um módulo de deformabilidade bastante elevado, correspondente à situação de baixamento rápido de tensão com recuperação apenas parcial e restrita da deformação processada até ao momento.

Finalmente a trajectória P-D corresponde a uma situação em que o afastamento da rotura se dá por aumento da tensão normal, a par de diminuição ou aumento (embora inferior ao valor limite definido pela inclinação tan(α)) da tensão tangencial. Nestes casos o módulo de deformabilidade assumido é o módulo inicial de carga correspondente ao novo nível de tensão normal, por se considerar que houve uma mudança na curva tensão-deformação (passagem de P para D).

Em resumo, a memorização do nível de tensão é feita à custa de duas variáveis de estado que permitem definir inequivocamente o procedimento a tomar face a uma variação no estado de tensão. As variáveis consideradas são o nível de tensão, SL conforme foi definido na Eq. (56), e a tensão normal do incremento anterior $\sigma_{n(i-1)}$.

Refira-se finalmente, que a determinação do modo de comportamento é efectuada tendo apenas em consideração a fase de carga anterior à do momento, contrariamente ao procedimento seguido para o modelo hiperbólico tradicional, que considera como parâmetro determinante do modo de comportamento o nível de tensão actual comparado com o nível máximo a que o material esteve sujeito no decorrer da prévia trajectória de tensões. Esta alteração no modelo reológico para as juntas, deriva do facto de nos ensaios efectuados se ter constatado que mesmo após se ter atingido um estado de quase-rotura, e se for aumentado o nível de tensão normal, o material exibir um comportamento quase igual ao comportamento exibido quando o corte se processou sempre com o mesmo nível de tensão normal.

Na Fig. 8 apresenta-se o fluxograma da subrotina utilizada para modelar o comportamento de um elemento de junta sem espessura (ELAWJ). Neste fluxograma distinguem-se alguns pontos de maior importância que se passam a detalhar:

a) Teste ao comportamento elástico linear. Nesta hipótese, os termos K_N e K_T são obtidos pela multiplicação dos parâmetros do modelo K_J e K_{NJ} pelo peso volúmico da água de modo a converter os sistemas de unidades.

b) Teste à abertura da junta por tracção (o indicador Rotura é activado) ou a um nível de tensão normal excessivamente reduzido, caso em que se assume um valor mínimo para o cálculo de módulos iniciais.

c) Cálculo do módulo tangente inicial, da tensão tangencial de rotura, tensão tangencial última e nível de tensão.

d) Com base no nível de tensão ou no valor do indicador de rotura por tracção, é calculado o módulo tangente correspondente à situação de rotura.

e) A rotura deu-se por abertura de junta, logo para a rigidez normal assume-se

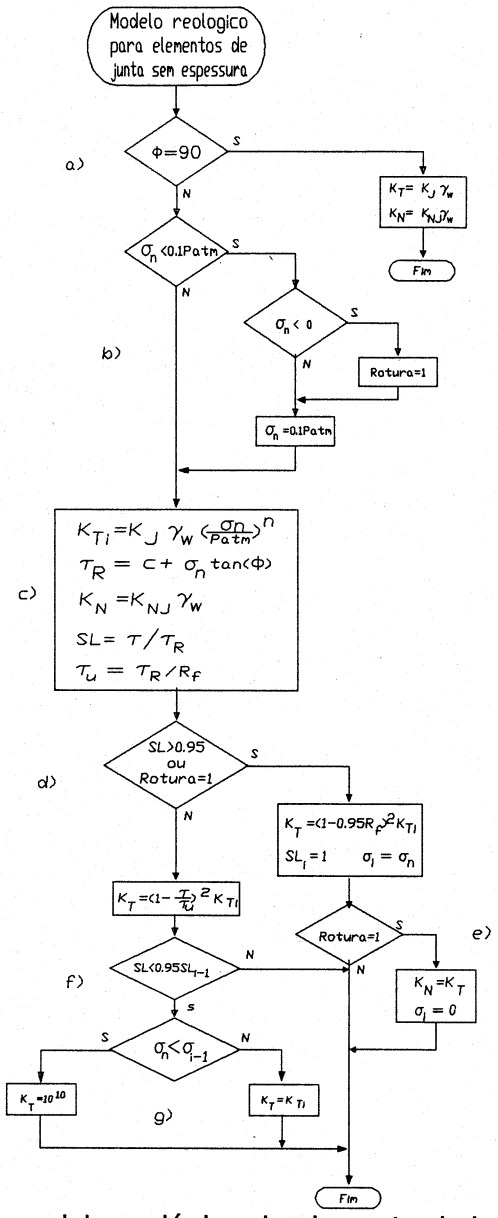


Fig. 8 - Fluxograma para o modelo reológico do elemento de junta

arbitrariamente um módulo igual ao tangente (também em rotura).

f) Teste à situação de descarga pela comparação dos níveis de tensão actual e anterior. Uma descida de 5% em SL corresponderá à situação de descarga.

g) Dentro da situação de descarga é feita a distinção se há redução da tensão tangencial ou aumento da tensão normal.

3.4.2 - Modelo reológico para juntas com espessura.

Para este elemento, não é prático descrever as relações tensão deformação em função da rigidez da junta já que, sendo a espessura variável e finita, os termos K_T e K_N são também variáveis com esta grandeza.

Assim parece ser conveniente descrever as relações constitutivas em termos de um módulo de deformabilidade confinado M e de um módulo de distorção G. Esta formulação de comportamento em termos dos módulos M e G tem como vantagem evitar que os erros na determinação do coeficiente de Poisson se repercutam gravemente nos termos da matriz tensão deformação.

A forma típica das curvas tensão deformação que se ilustram na Fig. 9 permite também o ajuste a leis hiperbólicas do tipo:

$$\tau = \frac{\gamma}{a+b\gamma} \qquad \varepsilon = \frac{\sigma_n}{c+d\sigma_n} \tag{57}$$

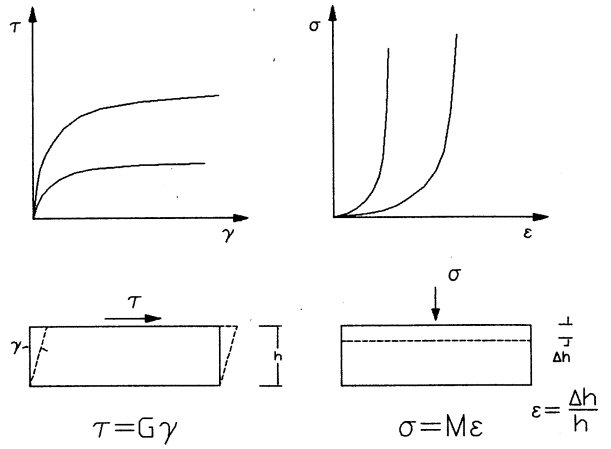


Fig. 9 - Elemento de junta com espessura, curvas tensão-deformação típicas

A interpretação dos parâmetros a, b, c e d pode ser realizada de modo perfeitamente análogo ao indicado para os ensaios de corte directo resultando:

$a=1/G_i$ - inverso do módulo de distorção inicial;

$b=1/\tau_u$ - inverso da tensão tangencial última;

$c=M_i$ - módulo edométrico inicial;

$d=1/\varepsilon_{max}$ - deformação máxima.

Em analogia com o modelo EC-K0 (Veiga Pinto, 1983), pode-se utilizar uma lei geométrica para relacionar a tensão normal com a deformação correspondente, ficando assim:

$$\sigma_n = A\ Pa\ \varepsilon^B \tag{58}$$

Note-se ainda que esta equação tem a vantagem de descrever curvas de concavidade

positiva (B>1), negativa (0<B<1) ou indiferente (B=1).

3.5 - Descrição dos módulos

Apresenta-se em seguida a lista e explicação resumida das subrotinas e funções utilizadas no programa, cuja listagem se apresenta no Anexo II.

MBEV60 - Módulo inicial do programa. Dimensionamento de variáveis. Inicialização de vectores e matrizes, gestão do fluxo do programa.

LOGON - Apresentação do programa.

LEITUR - Módulo de gestão da leitura dos dados gerais do ficheiro de dados. Coordenadas dos pontos nodais, propriedades dos materiais, condições de apoio e Incidencias nodais. Permite a opção de cálculo das coordenadas dos pontos nodais no meio dos lados de cada elemento.

LEXY - Leitura das coordenadas X e Y dos pontos nodais. Cálculo das mesmas coordenadas de pontos intermédios quando são definidos alinhamentos de pontos pelas suas extremidades.

LEMAT - Leitura das características mecânicas dos diversos tipo de materiais considerados.

POSC - Cálculo das coordenadas dos pontos no meio de cada lado de um elemento finito (opcional).

MED - Cálculo da coordenadas de um ponto com base na média dos pontos extremos. Utilizada pela subrotina POSC.

LEELE - Leitura da descrição dos elementos finitos. Opção de definir apenas os extremos de um grupo de elementos do mesmo tipo e do mesmo material.

ESCRNO - Impressão das coordenadas dos nós em 2 colunas.

LEAP - Leitura das condições de apoio.

FASE - Módulo de gestão de uma fase de carregamento. Determinação da dimensão do sistema de equações. Fases consideradas: 1)construção de uma camada de elementos, 2)aplicação de cargas, 3)colapso e aumento de peso próprio de elementos submersos, 4)impulsão e forças em fronteiras impermeáveis, 5)fluência (não implementada) 6)movimento de apoios. Gestão do número de incrementos a considerar para a simulação de cada fase. Para o caso de se tratar de um cálculo em continuação de um cálculo anterior, realiza a leitura das tensões e deslocamentos correspondentes a fase anterior. Realiza a escrita dos resultados no ficheiro de dados para o programa MEFSG e no ficheiro de resultados.

LOCATE - Localiza no ficheiro de dados para o programa MEFSG, os resultados (tensões e deslocamentos) correspondentes a uma determinada fase.

READD - Leitura de deslocamentos do ficheiro de registo (chamado por FASE)

LELEV - Leitura dos níveis de tensão dos elementos (chamado por FASE)

LETEN - Leitura das tensões dos elementos (chamado por FASE)

IMPULS - Cálculo das forças de impulsão a aplicar nos elementos que foram submersos. Definição dos elementos que ficam em descarga. Cálculo das forças a aplicar nas fronteiras impermeáveis com base na variação de altura de água nos pontos nodais.

MAXNO - Determinação do número do nó máximo pertencente a uma dada camada (necessário para a definição da dimensão do sistema de equações).

CONSTR - Cálculo das forças volúmicas resultantes da construção de uma ou mais camadas de elementos finitos.

ASSEMB - Assemblagem (junção) das matrizes de rigidez individuais de cada elemento finito na matriz de rigidez global.

ESCRF - Escrita do vector de forças (apenas para os nós carregados) no ficheiro de resultados.

ICAB - Selecciona o cabeçalho de tensão para escrita no ficheiro de resultados, consoante o tipo de elemento finito.

ESCRTE - Escrita dos valores das tensões e extensões. As extensões são calculadas através das tensões, invertendo a matriz D ($\epsilon = D^{-1}\sigma$). As tensões e extensões no centro do elemento correspondem à média dos valores em cada ponto de Gauss.

RODA - Converte a tensão do referêncial global na tensão no referêncial do elemento de junta pela lei de transformação tensorial. $\sigma' = T\ \sigma\ T^T$

MULT - Subrotina genérica de multiplicação de matrizes

TRANSP - Transposição de uma matriz

TRADD - Transfere a matriz tridimensional DD(3,3,ngauss) para uma matriz de trabalho D(3,3)

INVERT - Subrotina de inversão de uma matriz

MULTV - Multiplicação de uma matriz por um vector

MATKK - Subrotina que para cada elemento finito calcula a matriz de rigidez procedendo à chamada da subrotina adequada. Calcula também a matriz de tensão - deslocamento

TRANXE - Transfere, para cada elemento finito, as coordenadas dos pontos nodais para um vector de trabalho

KKGLOB - Monta a matriz de rigidez global. Efectua um ciclo para todos os elementos activos na fase considerada, procede as adaptações necessárias para atender aos fenómenos de descarga ou de construção, lê as tensões da fase anterior ou as tensões iniciais, estabelece os parâmetros E, ν (meio contínuo) ou K_T, K_N (elemento de junta) para cada ponto de Gauss, calcula a matriz de rigidez de cada elemento, assembla e finalmente impõe as condições de apoio.

TRASIG - Passa para um vector de trabalho as tensões de cada ponto de Gauss.

NPON - Indica para cada elemento o número de pontos de Gauss a considerar.

INDEXA - Coloca num vector a numeração dos deslocamentos correspondentes a um determinado elemento finito

CORDG - Determina a posição do centro de gravidade em coordenadas globais, a partir das coordenadas locais, para os elementos quadrangulares e triangulares.

INIC - Inicializa uma matriz a zeros.

INIC3 - Inicializa uma matriz tridimensional a zero.

INICV - Inicializa um vector a zeros.

DATA - Subrotina de escrita da data e hora

DER8 - Calcula as derivadas das funções de forma para o elemento quadrangular de 8 pontos nodais

DER6 - Calcula as derivadas das funções de forma para o elemento triangular de 6 pontos nodais

CICLO - Permuta índices

FEM8 - Calcula a matriz de rigidez e tensão-deslocamento para o elemento quadrangular de 8 nós

FEM6 - Idem para o elemento triangular de 6 nós

FEMJ - Idem para elemento de junta sem espessura.

FOSUP6 - Calcula as forças nodais equivalentes num elemento triangular de 6 pontos nodais para um carregamento (de grau inferior ou igual a 2) definido pelas suas intensidades nos pontos nodais.

FOSUP8 - Idem para o elemento quadrangular de 8 pontos nodais.

FOVOL6 - Forças nodais equivalentes a acções volúmicas (definida pela sua intensidade q_x,q_y) para o elemento triangular de 6 pontos nodais.

FOVOL8 - Idem para o elemento quadrangular de 8 pontos nodais.

FOTE8 - Calcula as forças nodais equivalentes a estados de tensão definidos pelos seus valores nos pontos de Gauss, para os elementos rectangulares de 8 pontos nodais.

FOTE6 - Idem para os elementos triangulares.

FOTEJ - Idem para os elementos de junta sem espessura.

SOMAF - Adiciona o vector de forças dum elemento ao vector global de forças

FUFOR8 - Funções de forma para o elemento quadrangular de 8 pontos nodais

FUFORJ - Idem para o elemento de junta sem espessura.

FUFOR6 - Idem para o elemento triangular.

GAUSS6 - Inicialização de vectores com as coordenadas dos pontos de gauss e respectivos pesos para integração numérica para os elementos triangulares de 6 pontos nodais.

GAUSS8 - Idem para os elementos quadrangulares de 8 pontos nodais.

MATD - Estabelece a matriz tensão-deformação dados os valores de E e ν. Para os elementos de junta sem espessura são dados os valores de K_T e K_N.

MATYM - Coordenadas locais dos pontos nodais.

DELTA - Matriz identidade.

MATYV - Estabelece a matriz YV.

CARGAS - Leitura das cargas distribuídas num lado do elemento. Gestão do acesso às rotinas adequadas para cada tipo de elemento. Leitura das forças concentradas nos pontos nodais.

DISTR - Coloca no vector global de forças, os valores resultantes da subrotina CARGAS.

BANDW - Determina a largura de banda e número de equações para a matriz do sistema global.

BANDWD - Idem, para cada fase de carregamento (em que a malha pode ainda não estar completa).

ATRIB - Transferência das características mecânicas de um tipo de elemento para um bloco COMMON.

ANULD - Subrotina que anula os deslocamentos no topo de uma camada recém construída.

PRINCI - Cálculo dos valores principais de um tensor.

ELAW - Módulo de gestão do modelo reológico. Chama ELAWHY, ELAWJ ou ELAWEC consoante se trate do modelo hiperbólico, hiperbólico para elementos de junta ou EC-K0.

ELAWJ - Modelo hiperbólico para elemento de junta sem espessura.

ELAWHY - Modelo hiperbólico para os restantes elementos.

ELAWEC - Modelo EC-K0.

RESOL - Determinação de deslocamentos, tensões etc.. Forma a matriz de rigidez global, resolve o sistema de equações. Calcula a tensão para cada elemento distinguindo os casos de elementos já existentes ou recém construídos. Para estes dois casos,

distingue ainda os cálculos de primeira ou segunda iteração. Escreve tensão, anula deslocamentos de topo de camada, escreve características elásticas correntes e calcula extensões. Adiciona os deslocamentos da fase aos deslocamentos totais.

SOLVE - Resolve pelo método de Gauss o sistema de equações tirando partido da sua simetria.

TENSÃO - Cálculo das tensões nos elementos (todos os tipos) a partir dos deslocamentos nos pontos nodais e da matriz tensão deslocamento definida em FEM8, FEM6, FEMJ OU FEMP e armazenada em ficheiro. A tensão no ponto central é calculada pela média simples dos valores das tensões nos pontos de Gauss.

AVG - Cálculo da tensão média dos pontos de Gauss.

TRANJ8 - Matriz de transformação de coordenadas para o elemento rectangular de 8 nós.

TRANJJ - Idem para o elemento de junta sem espessura.

TRANJ6 - Idem para o elemento triangular de 6 nós.

VALINI - Determinação dos valores iniciais de tensão e características mecânicas para todos os elementos. Nos elementos quadrangulares e triangulares a tensão é calculada com base na altura do elemento enquanto que nos elementos de junta a tensão é arbitrada.

TENINI - Determina as tensões iniciais para os elementos finitos. Para os elementos rectangulares e triangulares calcula-se a distância vertical entre o centro do elemento e o ponto de maior cota. Para os elementos de junta assume-se $\tau=0.01$ Pa e $\sigma=0.1$ Pa.

MODIF - Impõe condições de apoio na matriz [K] e no vector de forças {F}

ALTAP - Apoio fixo.

MODIMP - Deslocamentos impostos. Lê os deslocamentos do ficheiro de dados.

ALTDES - Chamado por MODIMP para fixar um deslocamento.

COLAPS - Lê os elementos submersos e calcula as tensões de relaxação de acordo com o modelo reológico adoptado para o material (hiperbólico ou EC-K0), calcula as forças equivalentes à libertação dessas tensões, constrói o vector de forças global. Calcula as forças volúmicas correspondentes ao aumento de peso próprio por molhagem do material. Altera o vector de forças.

RELAXEC - Determinação das tensões de relaxação para os elementos cujo comportamento segue a lei EC-K0.

RELAX - Idem para o modelo hiperbólico.

CLPS3 - Determina $(\sigma_1-\sigma_3)_w$ recorrendo ao método de Newton-Rapson para determinar σ_3 de modo a igualar as deformações volumétricas.

FUNC - Função a determinar a raiz pelo método de Newton-Rapson. Esta função devolve a diferença de deformações volumétricas entre o material seco e o material molhado. Anulando esta diferença as deformações serão obviamente iguais.

EA - Determinação da deformação axial com base em σ_1, σ_3 e nas características mecânicas do momento.

S3PAN - $(\sigma_3/Pa)^n$

ER - Determinação da deformação radial com base em $\sigma 1$, ϵ_a e nas características do material

EV - Deformação volumétrica $\epsilon_v = \epsilon_a + 2\epsilon_r$

S1 - Determinação de $\sigma 1$ com base no modelo reológico do material.

EALEAC - Deformação volumétrica de colapso e correspondente deformação axial.

FIP2 - Determinação de variáveis auxiliares.

DERP - Derivadas das funções de forma para o elemento de junta com espessura.

FUFORP - Funções de forma para o elemento de junta com espessura.

TRANJP - Matriz de transformação para o elemento de junta com espessura.

FOSUPP - Determinação de forças nodais equivalentes a cargas de superfície para elementos de junta com espessura

FOVOLP - Determinação de forças volúmicas para o elemento de junta com espessura.

FEMP - Matriz de rigidez e matriz de tensão deslocamento para o elemento de junta com espessura.

FOTEP - Forças nodais equivalentes a uma libertação de tensões no elemento de junta com espessura.

4 - PREPARAÇÃO DE DADOS

A estrutura do ficheiro de dados para o programa MBE não difere da que se usa em programas do mesmo tipo. Este facto permite que, com um mínimo de alterações, se possa aproveitar os ficheiros de dados, nomeadamente no que se refere as incidências e coordenadas dos pontos nodais, podendo então serem utilizados noutros programas de elementos finito como por exemplo em programas de cálculo dinâmico ou de percolação.

O ficheiro de dados esta dividido em três grandes blocos:

1) Informação geral
2) Descrição de geometria
3) Sequência de efeitos.

No grupo da informação geral inserem-se todos os dados que de índole geral, nomeadamente o número de pontos nodais, de elementos finitos de materiais etc.

A descrição da geometria e constituída pelas coordenadas dos pontos nodais e pela descrição das incidências dos elementos finitos.

A sequência de efeitos descreve o conjunto de dados que permite ao programa determinar as variações de malha e forças nodais que constituirão cada fase do cálculo.

Os três blocos supracitados são agora descritas em mnemonicas, esclarecendo-se a frente o seu significado:

INFORMAÇÃO GERAL:

Nome
nelem nnos nap 0 0 nummat nfases
pg8 pg6 pgj
qt tint
patm

i=1,nummat
    MATER
    pvol k kur n d g f c fi dfi rf dgama /

DESCRIÇÃO GEOMETRIA:

i=1,nnos
    no x y

i=1,nap
    apoio nó rx ry

i=1,nelem
    ele no1..no8 mat mod codmol 0 tipo

0 0

SEQUÊNCIA DE EFEITOS:

'codigo' nmax
EFEITO
nescal
ngrup
i=1,ngrup
    noi noj

Descrição das mnemónicas utilizadas

Nome - designação do caso em estudo
nelem - número de elementos finitos que constituem a malha

nnos - número de pontos nodais
nap - número de pontos nodais considerados fixos
nummat- número de materiais diferentes
nfases- número de efeitos diversos a considerar
pg8 - número de pontos de gauss para os elementos finitos rectangulares de 8 pontos nodais (2, 3) para (4, 9) pontos
pg6 - número de pontos de gauss para os elementos finitos triangulares de 6 pontos nodais (3, 7)
pgj - número de pontos de gauss para os elementos finitos de junta (deverá ser sempre superior a pg8) (2,3,5)
qt - qual a tensão a imprimir no ficheiro de resultados:
1- só a tensão no ponto médio (obtida pelas medias)
2- só a tensão nos pontos de gauss
3- ambas
tint - imprimir as tensões da iteração intermédia (1) ou não imprimir(0)
patm - valor da pressão atmosférica no sistema de unidades do caso:
1.033 kgf/cm^2
10.33 tf/m^2
103.3 kN/m^2
MATER - designação do tipo de material cujos parâmetros se seguem
pvol - peso volúmico

k,kur,n,d,g,f,c,fi,dfi,rd,dgama - parâmetros do modelo hiperbólico
Ae,Be,Ce,De,Ak0,Bk0,Ck0,Dk0 - parâmetros do modelo EC-K0

no - número do nó cujas coordenadas se seguem
x,y - abcissa e ordenada do ponto nodal "no"
ele - número do elemento cuja descrição se segue
no1..no8 - incidências nodais
mat - número do material correspondente ao elemento
mod - tipo de modelo
1- hiperbólico
2- EC-K0
codmol- comportamento do elemento face à água
0 - sem impulsão e sem colapso
1 - só impulsão
2 - só colapso
3 - impulsão + colapso
tipo - Tipo de elemento finito
1- Rectangular de 8 nós
2- triangular de 6 nós
3- junta sem espessura
4- junta com espessura

'código' - Variável de selecção do efeito a simular
'#1 ' - Fase de construção
'#2 ' - Aplicação de forças concentradas e distribuídas
'#3a' - Enchimento, fase de aplicação de aumento de peso próprio + colapso para os materiais que sofram este efeito
'#3b' - Enchimento, fase de aplicação de forças nas fronteiras e impulsão
nmax - número do elemento de maior ordem

EFEITO - Bloco de informação correspondente a descrição da fase (descrito mais a frente)

nescal - esta variável permite dividir o vector de forças resultante do efeito a simular em nescal escalões, melhorando por consequência a simulação da não linearidade

ngrup - número de grupos de nos onde se pretende que o deslocamento seja anulado (para simulação da construção)

noi,noj - nó inicial e final de cada grupo de nós onde se pretende anular o deslocamento (todos os nós intermédios serão considerados de deslocamento nulo)

EFEITO

1) Fase de construção

A informação resume-se a indicação do número da camada de elementos finitos adicionada, o elemento inicial dessa camada e o elemento final no seguinte formato:

ncamada ele_ini ele_fin

2) Aplicação de forças concentradas e distribuídas

nfd nfc
i=1,nfd
    ele lado p1y p1x p2y p2x p3y p3x

i=1,nfc
    no Fx Fy

3) Colapso e aumento de peso próprio por molhagem

numel_col
i=1,numel_col
    elei

4) Impulsão e forças nas fronteiras impermeáveis

numel_imp
i=1,numel_imp
    elei
nforças
i=1,nforças
    ele lado h1 h2

Descrição das mnemónicas correspondentes aos efeitos:

ncamada - Número da camada de elementos a adicionar
ele_ini - Número do elemento finito que define o início da camada
ele_fin - Idem para o fim da camada
ndf - Número de forças distribuídas
nfc - Número de forças concentradas nos pontos nodais

ele - Elemento
lado - Número do lado do elemento em que está aplicada a carga
pix,piy - Intensidade da forca no nó i, segundo as direcções correspondentes

5 - PROGRAMA DE EXPLORAÇÃO GRÁFICA

5.1 - Funcionamento e possibilidades.

Ao contrário do programa MBE no qual a interacção com o utilizador e reduzida ao mínimo indispensável (toda a informação necessária existe no ficheiro de dados e o aspecto fundamental é a rapidez de processamento), o programa de exploração gráfica é predominantemente interactivo e baseado na escolha de diversas opções entre os menus apresentados ao utilizador.

Existem fundamentalmente dois níveis de trabalho com este programa. No primeiro, basta existir um ficheiro de dados com os elementos da geometria do problema em estudo, permitindo nesta fase o traçado da malha de elementos finitos com ou sem numeração de nós e/ou elementos. O traçado da malha é de importância crucial na fase de preparação dos dados pois verifica-se que um grande número de erros é cometido nesta fase.

O segundo nível de utilização do programa está disponível quando o cálculo de determinado problema foi já efectuado. Nesta altura e possível traçar diversos diagramas de diversas grandezas. As diversas opções disponíveis são:

1 - Desenho da malha
2 - Desenho de deslocamentos
3 - Desenho de tensões
4 - Malha deformada
5 - Isolinhas

A cada uma das opções acima indicadas correspondem ainda diversas alternativas que seguidamente se indicam:

1 - Desenho da malha original
2 - Deslocamentos
  2.1- Deslocamento total
  2.2- Deslocamento diferencial entre duas fases

3 - Desenho dos vectores representativos do estado de tensão (para qualquer fase)
4 - Desenho da malha deformada (para qualquer fase do cálculo)
5 - Desenho de isolinhas
  5.1- Isolinhas de tensão com as seguintes opções a)σ_x b)σ_y c)τ d)σ_1 e)σ_3
  5.2- Isolinhas de deslocamento a)vertical b)horizontal
  5.3- Isolinhas de nível de tensão a)valor máximo b)valor actual

Em qualquer uma das hipóteses anteriores o utilizador tem a possibilidade de ampliar uma determinada zona do desenho, bastando para tal especificar as coordenadas (reais) da "janela" que deseja com mais pormenor. Essa zona será ampliada no sentido da sua

dimensão se ajustar a dimensao do papel préviamente escolhida[3].

Ainda dentro das opções comuns a todas as hipóteses de desenho existe a possibilidade de numerar os elementos e/ou pontos nodais, e sobrepor a malha inicial do problema ao desenho selecionado.

O funcionamento do programa para este nível de exploração baseia-se na existência de um ficheiro de extensao ".plo" criado automaticamente pelo programa de cálculo. Para auxiliar a criação de desenhos pode ainda ser definido um ficheiro auxiliar ".aux" de dados para o programa MEFSG, que contém informação auxiliar para minimizar a introdução de dados no terminal. O seu formato é seguidamente explicado. Saliente-se ainda que os dados a fornecer são opcionais e a sua ordem é arbitrária.

[3] A dimensão do papel é determinada em menú próprio existindo prédefinidos os seguintes formatos: A4,A3,A4D,A3D,MU,MUD,Manual. Onde a letra "D" significa deitado e "MU" mancha útil do LNEC.

FORMATO:	COMENTÁRIOS:
CONTORNO npts i=1,npts ponto codigo	Indicador de contorno número de pontos para todos os pontos quais os nós código=1 desenha, código=0 não desenha (para o 1° ponto codigo=0)
IDESLOCX niso (val_i) i=1,niso	Isolinhas de deslocamento horizontal, niso=quantas isolinhas val_i = quais
IDESLOCY niso (val_i) i=1,niso	Isolinhas de deslocamento vertical
ITENSAOX niso (val_i) i=1,niso	Isolinhas de tensão horizontal
ITENSAOY niso (val_i) i=1,niso	Isolinhas de tensão vertical
ITENSAOT niso (val_i) i=1,niso	Isolinhas de tensão tangencial
ITENSAO1 niso (val_i) i=1,niso	Isolinhas de tensão principal máxima
ITENSAO3 niso (val_i) i=1,niso	Isolinhas de tensão principal mínima
SL niso (val_i) i=1,niso	Isolinhas de nível de tensão (actual)
ISLMAX niso (val_i) i=1,niso	Isolinhas de nível de tensão máximo

5.2 - Descrição da modelação.

A presente secção descreve todas as subrotinas e funções do programa MEFSG cuja listagem se apresenta no Anexo III.

MEFSG - Programa principal. Leitura do cabeçalho do ficheiro de dados. Dimensionamento de variáveis. Chama a leitura de dados e chama o 1º menú do programa.

INICV - Inicialização de um vector a zero.

LEITUR - Módulo de gestão da leitura de dados. Ignora as características mecânicas, lê as coordenadas, lê as incidências.

POSC,MED,LEXY,LEAP,LEELE - Já definidos.

CORDM - Determinação dos valores máximos e mínimos das coordenadas da malha.

SKIP - "Salta" registos na unidade 8.

LOCATE - Localiza, no ficheiro onde estão guardados os resultados do cálculo, a posição de início dos dados de determinada fase.

GRAFS - Módulo operativo do programa. Determina novo nome para o ficheiro de plotter, determina as dimensões pretendidas para o desenho, verifica se existe o ficheiro com dados auxiliares para desenho, apresenta menú de possibilidades, efectua eventual ampliação, determina o factor de escala em função das escolhas anteriores e efectua o desenho.

NEWNAME - Determina o nome do desenho compondo o nome do ficheiro de dados com uma numeração sequencial.

SETSIZE - Determina a dimensão do desenho, apresentando várias hipóteses prédefinidas ou aceitando a dimensão definida pelo utilizador.

SET - Estabiliza as dimensões do desenho, de acordo com a escolha efectuada em SETSIZE

WHAT - Determina qual o desenho pretendido: desenho da malha; desenho de deslocamentos; desenho de tensões; malha deformada ou isolinhas.

AUX - Abre o ficheiro auxiliar

ZOOM - Pergunta ao utilizador se deseja zoom e no caso afirmativo, determina qual a zona a ampliar.

SETSCALEG - Determina, de acordo com as opções anteriores de dimensão da folha e zoom escolhido, o valor da escala gráfica.

DSCAG - Desenha a escala gráfica do desenho, escolhendo no vector esc() quais os limites.

FIND - Procura, no ficheiro auxiliar, uma palavra-chave.

PROCEED - De acordo com as opções estabelecidas até ao momento (zoom, dimensões, tipo de desenho), procede à realização do desenho. Se o ficheiro auxiliar existir, e consoante o desenho pretendido, lê os elementos relevantes. Para a opção de isolinhas pergunta quais as isolinhas a desenhar (de tensão, deslocamento ou nível de tensão).

PROMPT - Subrotina genérica de escolha (Sim/Não)

QFASE - Escreve no desenho qual a fase de cálculo

MALHA - Desenho da malha de elementos finitos indeformada

DEFORM - Altera matriz de coordenadas de modo a contabilizar a deformação de determinada fase de cálculo, desenha a malha deformada chamando MALHA, e repõe as coordenadas originais.

LINE - Desenha uma linha. São fornecidas as coordenadas reais, é feiro o teste às dimensões do desenho (quando há um "zoom" é necessário truncar linhas), e a linha é desenhada em coordenadas de desenho.

CIRCULO - Desenha uma circunferência para assinalar o início do vector de deslocamento. Procede também à verificação das coordenadas.

TEST - Verifica as coordenadas extremas de um determinado segmento de recta de modo a este estar sempre contido na mancha de desenho.

DNUMBER - Desenha um número (inteiro ou real) verificando se este se encontra dentro da mancha de desenho. As coordenadas fornecidas são reais.

NODENUM - Subrotina de numeração de nós da malha.

ELENUM - Subrotina de numeração dos elementos finitos.

CENTRO - Determina a posição do centro de um elemento finito pelo cálculo da média das coordenadas.

CONTOUR - Desenha o contorno do desenho de acordo com os elementos fornecidos no ficheiro auxiliar (caso exista).

IPEN - Converte a nomenclatura UP/DOWN da caneta de 3/2 para 0/1

SKIP9 - "Salta" linhas na unidade 9 (ficheiro com os resultados do cálculo).

DDESL - Prepara o desenho de deslocamentos. Alternativa de desenho de deslocamentos totais ou entre duas fases especificadas.

READD - Lê os deslocamentos de uma fase de cálculo.

PLODES - Desenha o vector de deslocamentos em cada ponto nodal da malha. Chama a subrotina de desenho da escala gráfica de deslocamentos.

DSCD - Desenha a escala gráfica para os vectores de deslocamento definida como 1 Cm

= x unidades de cálculo.

DTEN - Prepara o desenho de vectores de tensões principais. Identifica a fase de cálculo, lê as tensões correspondentes, determina a escala a utilizar e chama a subrotina que efectua o desenho.

DSCT - Desenha a escala gráfica de tensões.

LETEN - Lê na unidade 9 (ficheiro de resultados de cálculo) as tensões nos elementos finitos para uma determinada fase.

LELEV - Lê o nível de tensão (SL) actual e máximo para uma determinada fase.

ARROW - Desenha uma seta.

PRINCI - Determina valores principais.

PLOTEN - Traça vectores de tensão no centro de cada elemento finito e de acordo com a escal prédeterminada.

ISOL - Subrotina genérica de traçado de isolinhas. Recebe as coordenadas do elemento, os valores nodais da grandeza e o valor da isolinha pretendida. Cada elemento finito é dividido em triangulos. Para cada triangulo verifica se a isolinha pretendida intersecta a sua fronteira, e no caso afirmativo calcula e armazena as coordenadas da isolinha.

CONTI - Verifica se o valor pretendido para a isolinha está contido nos valores correspondentes ao lado do triangulo em consideração.

POSP - Determina as coordenadas (x,y) de intersecção da linha correspondente a um determinado isovalor com a recta definida pelos pontos extremos de um dos lados de um triangulo aos quais correspondem determinados valores da grandeza em apreciação.

RECTA - Interpolação linear para determinação da posição correspondente a um dado isovalor com base na posição e valor da grandeza dos pontos extremos (chamado por POSP).

TRANXE - Transfere as coordenadas dos pontos nodais de um determinado elementos finito para um vector de trabalho, determinando além disso a posição do ponto central pela média das coordenadas.

ISOLD - Prepara os elementos necessários para o traçado das isolinas de deslocamento (vertical ou horizontal). Gere a leitura dos deslocamentos e, caso o ficheiro auxiliar exista lê as isolinhas pretendidas. Caso contrário pergunta ao utilizador os valores pretendidos. Chama a subrotina de traçado de isolinhas.

ESCRISO - Escreve no desenho, quais as isolinhas traçadas.

ISOLT - Prepara os elementos necessários para o traçado das isolinas de tensão (vertical ou horizontal, distorcional, principal máxima ou mínima). Gere a leitura de tensões e, caso o ficheiro auxiliar exista lê as isolinhas pretendidas. Caso contrário pergunta ao utilizador os valores pretendidos. Medianiza as tensões em cada ponto nodal partindo

da extrapolação dos valores obtidos nos pontos de Gauss de cada elemento finito. Caso haja enchimento, permite considerar a pressão intersticial definida pela altura de água em cada ponto e assim calcula as tensões totais. Chama a subrotina de traçado de isolinhas.

ISOLS - Prepara os elementos necessários para o traçado das isolinas de nível de tensão (valor actual ou máximo). Gere a leitura de níveis de tensão e, caso o ficheiro auxiliar exista lê as isolinhas pretendidas. Caso contrário pergunta ao utilizador os valores pretendidos. Medianiza os níveis do nível de tensão em cada ponto nodal partindo da extrapolação dos valores obtidos nos pontos de Gauss de cada elemento finito. Chama a subrotina de traçado de isolinhas.

TENPRI - Transfere as tensões de um determinado elemento do vector geral de tensões para um vector de trabalho e chama a subrotina de cálculo de tensões principais.

TENPON - Calcula, pelo método dos mínimos quadrados e para cada elemento finito os valores das tensões nos pontos nodais partindo dos valores de tensão nos pontos de Gauss.

LEVPON - Idêntico a TENPON mas para os níveis de tensão.

INTER8 - Para o elemento quadrangular de 8 nós, determina os coeficientes do plano de mínimos quadrados ($aX + bY + c = 0$) e, para os pontos nodais, determina os valores da grandeza pretendida.

INTER6 - Idêntico a INTER8 mas para elementos finitos triangulares de 6 pontos nodais.

MAT8 - Estabelece a matriz de correlação para o elemento finito rectangular de 8 pontos nodais e para 4 ou 9 pontos de integração numérica.

MAT6 - Idêntico a MAT8 mas para os elementos triangulares.

ELEMSUB - Identifica no ficheiro de dados, quais os elementos que estão submersos e para estes, determina a altura de água sobre cada ponto nodal.

LEELESUB - Lê e guarda num vector quais os elementos submersos.

ALTURA - Determina os pontos nodais que estão submersos recorrendo às incidências. Determina a máxima cota de água até ao momento e com base nessa cota e na ordenada de cada ponto nodal, determina a altura de água sobre esse ponto.

5.3 - Aplicação à microinformática. Ligação a programas CAD

Aproveitando a capacidade actual dos microcomputadores, os programas de cálculo e desenho foram integralmente desenvolvidos em linguagem Fortran 77 standard, permitindo assim garantir a sua portabilidade entre sistemas diversos. Por outro lado e no que respeita à exploração gráfica foi levada em consideração a existência de software de processamento gráfico de muito boa qualidade nos actuais microcomputadores PC-compatíveis. Por essa

razão o programa MEFSG de exploração gráfica dos resultados pode gerar[4] ficheiros compatíveis com a exploração posterior em programas existentes nos microcomputadores PC-compatíveis. No presente caso o programa utilizado para o processamento de desenhos foi o AutoCad (Autodesk, 1986). A utilização deste tipo de programas (CAD-Computer Aided Design) permite com grande vantagem a edição dos desenhos, facilitando a colocação de legendas, retoques, ampliações, cortes etc. Além disso torna-se ainda possível incluir directamente os resultados gráficos do programa num processador de texto facilitando assim a produção de peças escritas.

Refira-se ainda que a utilização de programas de CAD não se esgota no capítulo do pós-processamento podendo pelo contrário, ser utilizado com grande vantagem na preparação de dados, nomeadamente no desenho de malhas de elementos finitos.

Para a apresentação dos resultados gráficos resultantes do programa MEFSG quando este funciona em microcomputadores, foi elaborado um programa em basic que permite visualizar no ecrã os desenhos pretendidos. Esse programa é automaticamante chamado pelo módulo MEFSG apresentando de imediato o desenho pretendido.

---

[4] Para possibilitar o uso desses programas bastará incluir as subrotinas constantes no Anexo III. A execução do programa gera um ficheiro de extensão .SCR cuja leitura é possível em AutoCad.

Lisboa e LNEC em 18 de Dezembro de 1991

VISTOS

O Chefe de Núcleo de Fundações

P. Sêco e Pinto
Investigador Principal

O Chefe do Departamento de Geotecnia

E. Maranha das Neves
Investigador Coordenador

AUTORIA

João Marcelino
Assistente de Investigação

BIBLIOGRAFIA E REFERÊNCIAS BIBLIOGRÁFICAS

Autodesk, inc. (1986) - The AutoCAD Drafting Package. Reference Manual

Desai, C. S. (1981) - Behavior of interfaces between structural and geological media. Proc. Int. Conf. on Recente Advances in Geotech. Earthquake Eng. and Soil Dynamics. St. Louis Mo.

Desai,C.S.;Zaman,M.M.;Lightner,J.G. and Siriwardane (1984) - Thin-Layer element for interfaces and joints. International Journal Numerical and Analitical Methods in Geomechanics, 8, pp.19-43

Goodman, R. E.;Taylor, R. L. and Brekke, T. L. (1968 May) - A model for the Mechanics of Jointed Rock. Journal of Soil Mechanics and Foundation Division, Vol. 94, SM 3 pp.637-659

Griffiths,D.V. (1985) - Numerical modelling of interfaces using conventional finite elements. 5th ICONMIG, Nagoya, II 837-844

Janbu, N, (1963) - Soil compressibility as determined by oedometer and triaxial test. Proc. 4th ECSMFE, Vol. 1 Wiesbaden, pp. 19-26

Kondner,R.;Zelasko,J. (1963) - A hyperbolic stress-strain formulation for sands. Proc. of 2nd Pan-American Conference on Soil Mech. and Found. Eng. Vol. 1, Rio de Janeiro, pp. 289-324

Kondner,R.L. (1963) - Hyperbolic stress-strain response: cohesive soils. Proc. ASCE, Journal of Soil Mechanics and Foundation Division, NSM1, pp.115-143

Mateus da Silva; J.M.M. (1990) - Modelação de descontinuidades em Geotecnia". Dissertação de Mestrado. Lisboa

Naylor,D.J.;Mattar,D.,Engmann F. O. (1984) - Layered analysis of embankment construction. Institute for Numerical Methods in Engineering, Swansea

Naylor,D.J.;Maranha das Neves,E.;Mattar,D.;Veiga Pinto,A.A. (1986) - Class A prediction of construction perfomance of Beliche Dam. Thomas Telford Publications, London

Naylor,D.J.;Tong,S.;Shahkarami,A.A. (1989) - Numerical modelling of saturation shirinkage. 3rd NUMOG, Niagara Falls, Canada

Nobari,E.;Duncan,J. (1972) - Effect of reservoir filling on stresses and movements in earth and rockfill dams. Report n.TE-72-1, Office of Research Services, University of California, Berkeley

Pedro,J. Oliveira, (1973) - Finite element stress analysis of plates, shells and massive structures. CEB International Course on Structural Concrete (C3-1), LNEC. Lisboa

Pande,G.N. and Sharma,K.G. (1979) - On Joint/Interface elements and associated problems of numerical ill-conditionging. International Journal Numerical and Analytical Methods in Geomechanics, 3

Sêco e Pinto,P.S. (1983) - Fracturação hidráulica em barragens de aterro zonadas. Tese para especialista, LNEC. Lisboa

Sharma,H.D.,Nayak,G.C.and Maheshwari,J.B. (1975) - Nonlinear Analysis of Rockfill Dam with Vertical and Inclined Cores. International Symposium on Criteria and Assumptions for Numerical Analysis of Dams, Univ. Wales Swansea, U.K.

Sharma,H.D.,Nayak,G.C. and J.B. Mahaeshwari (1979) - Generalization of sequential nonlinear analysis. A study of rockfill dam with joint elements. Numerical Methods in Geomechanics pp.663-685

Sousa, L. R.;Teles, M. M. (1980) - Modelo de cálculo para estudos de túneis pelo método dos elementos finitos. Relatório interno, LNEC. Lisboa

Sousa,L.R. (1976) - Métodos Modernos de Dimensionamento de Túneis-Modelos Matemáticos. Geotecnia 16

Veiga Pinto,A.A. (1983) - Previsão do comportamento estrutural de barragens de enrocamento. Tese para especialista, LNEC. Lisboa

Zienkiewicz,O.C.;Best,B.;Dullage,C. and Stagg,K.G. (1970) - Analysis of Nonlinear Problems in Rock Mechanics with Particular Reference to Jointed Rock System. 2nd International Conference on Rock Mechanics, Belgrade, III,501-509

Zienkiewicz,O.C. (1977a) - The Finite Elelemt Method in Engineering Science. 3rd Edition. Mc Graw-Hill

ANEXO I

Exemplo de aplicação

O exemplo que se apresenta refere-se a um aterro construído sobre uma fundação e é apresentado para ilustrar a utilização dos programas MBEV60 e MEFSG, não se referendo a nenhum caso real. A geometria do exemplo e respectiva discretização em elementos finitos é a que a presenta na Fig. 1.

As acções para o exemplo apresentado são as correspondentes ao peso próprio do material de aterro (Fases 1 a 4) e a uma sobrecarga uniformemente distribuída de 6 x 50 kN/m^2 no topo do aterro (Fases 5 a 10). Os valores das acções e características mecânicas dos materiais foram escolhidos de modo a poder realçar algumas características do modelo, nomeadamente a propagação de zonas em plastificação (ou SL>0.95).

O programa MEFSG permite obter um vasto conjunto de diagramas ilustrativos dos resultados obtidos para o cálculo efectuado, no entanto, foram elaborados alguns desenhos, que se apresentam, e que abordam os seguintes aspectos:

- Malha de elementos finitos considerada para a análise do problema (28 elementos finitos e 105 pontos nodais).

- Configuração da malha deformada no final da fase construtiva e no final do cálculo (fases 4 e 10).

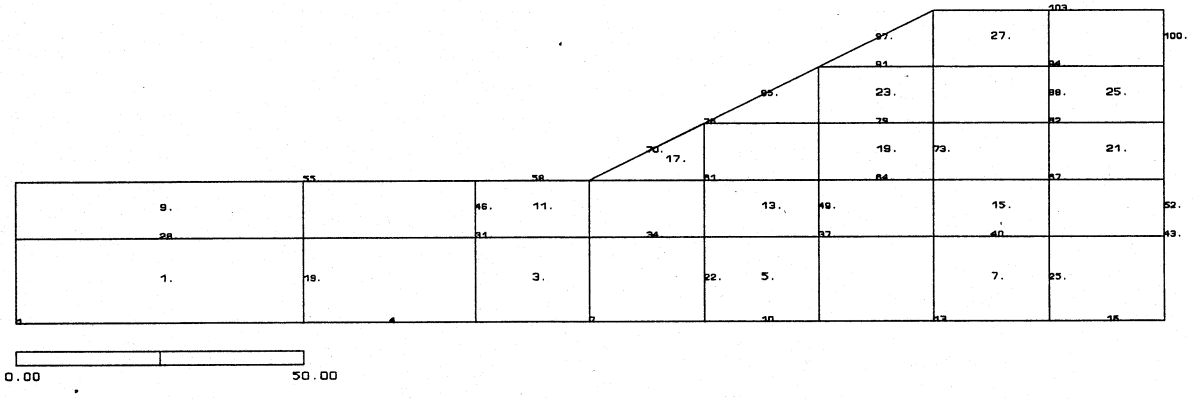
- Linhas de igual deslocamento horizontal no final do carregamento (fase 10).

- Linhas de igual deslocamento vertical no final de carregamento (fase 10).

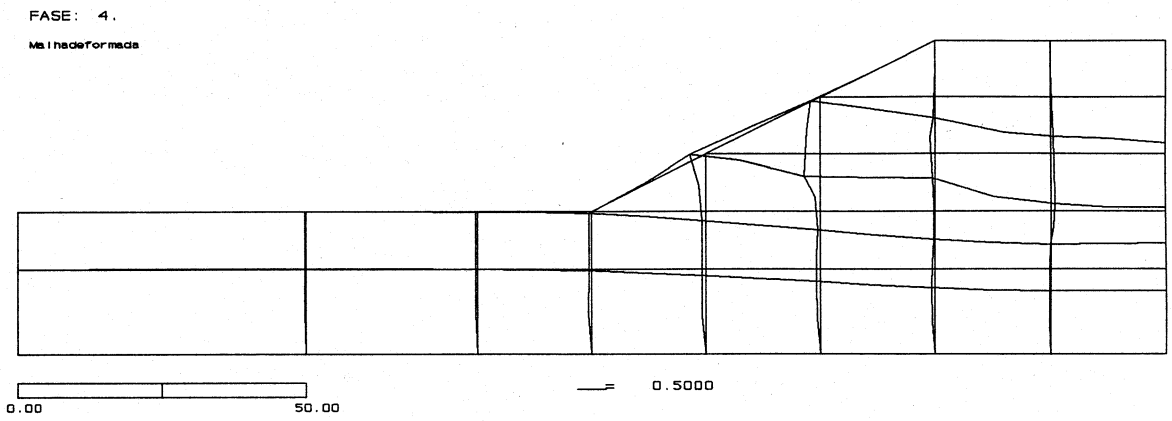
- Nível de tensão "SL" de 0,95 para a fase 10.

- Tensões principais no final do carregamento (fase 10).

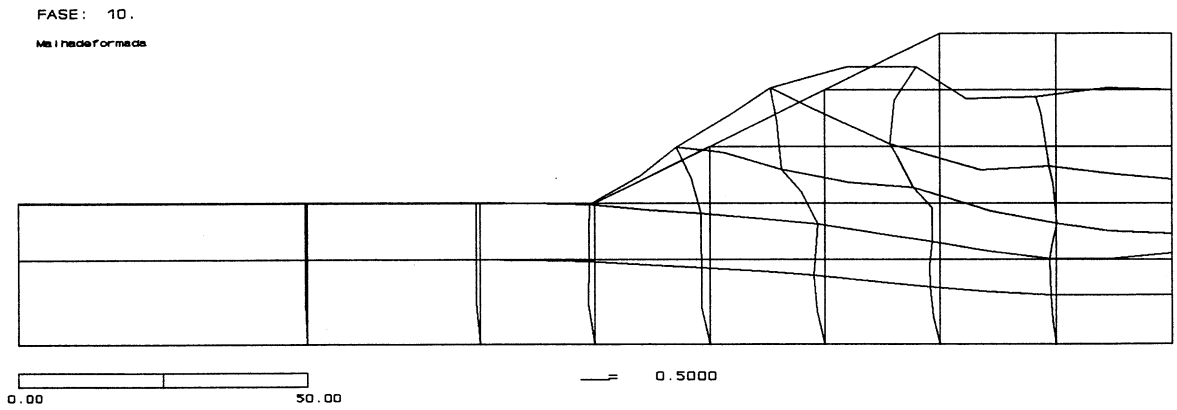
- Diferença entre os vectores de deslocamento total no final do carregamento e no final da construção (Fase 10 - Fase 4).



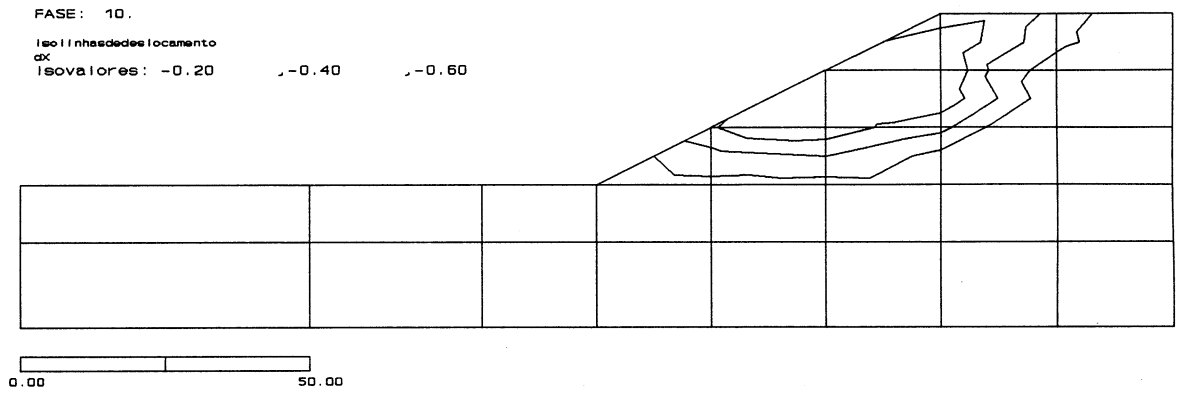
ANEXO - Fig. 1 - Malha do problema. Numeração de elementos e de nós



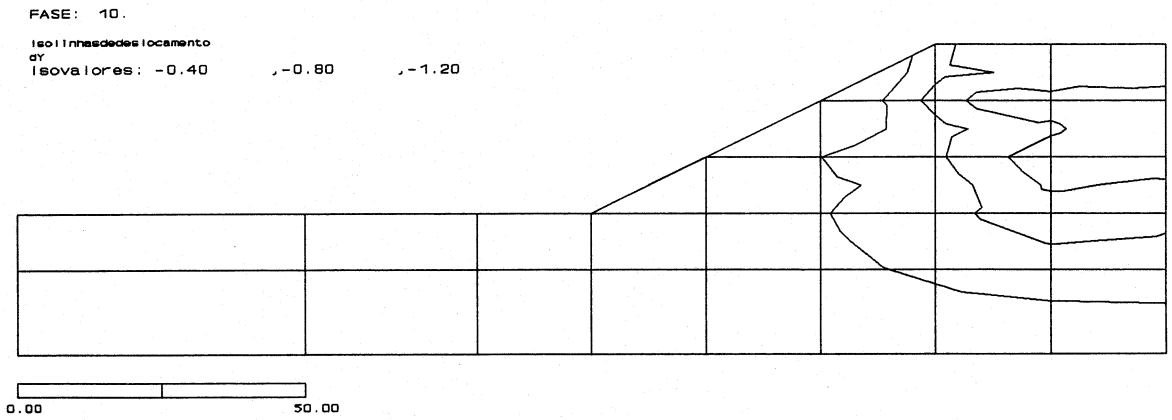
ANEXO - Fig. 2 - Malha deformada no final da construção



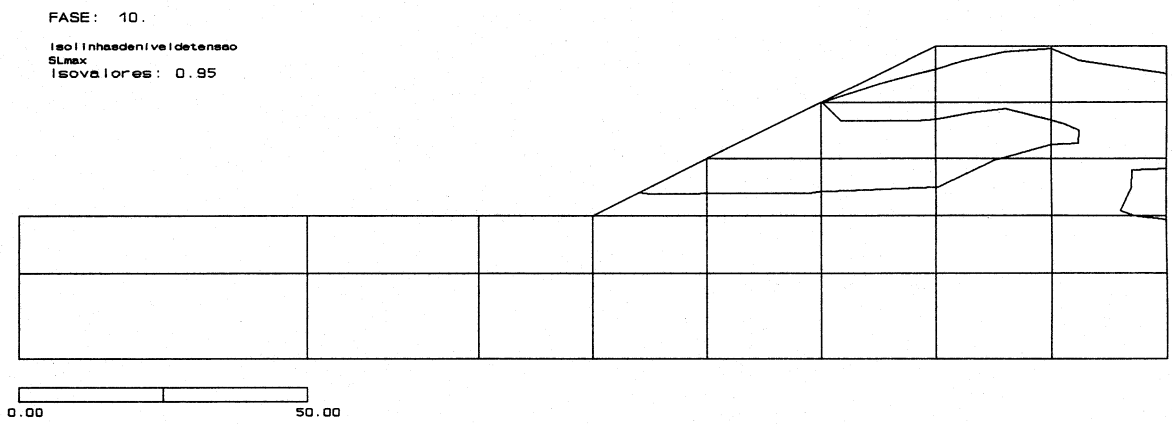
ANEXO - Fig. 3 - Malha deformada no final do carregamento



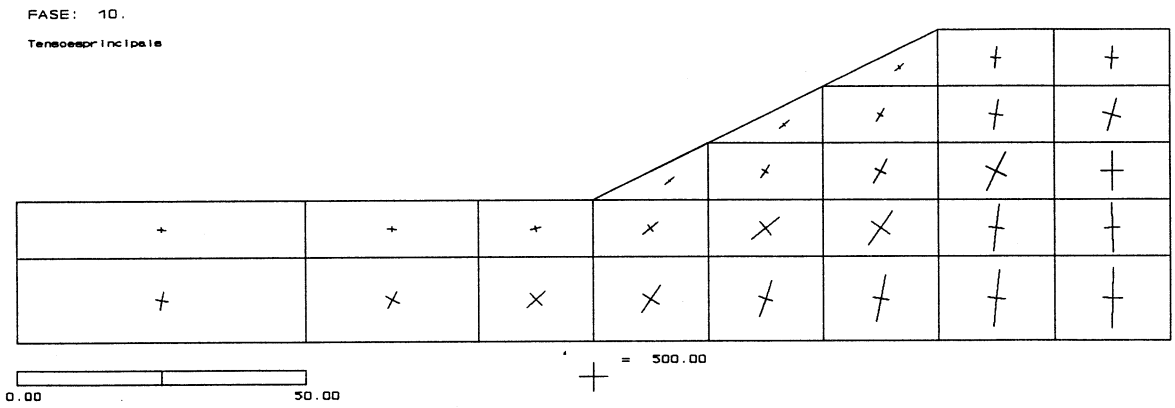
ANEXO - Fig. 4 - Isolinhas de deslocamento horizontal



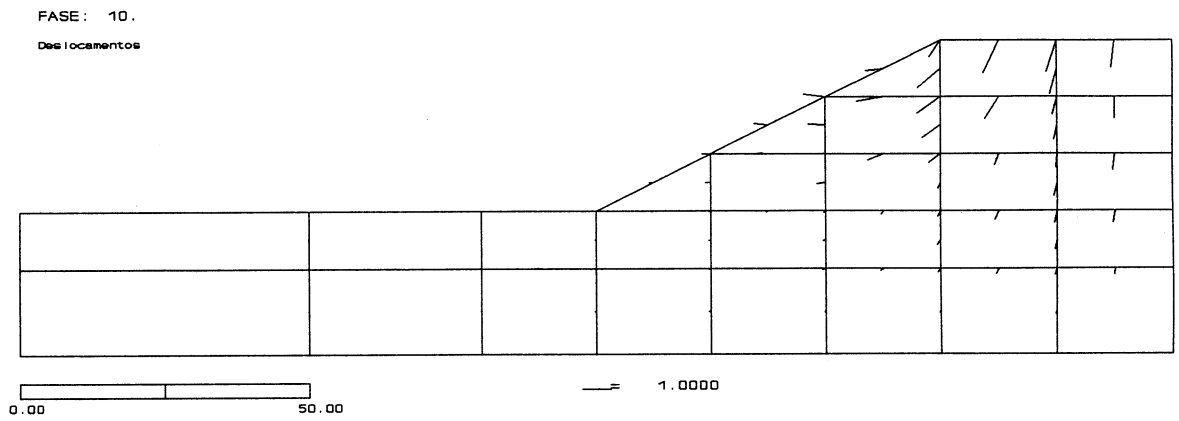
ANEXO - Fig. 5 - Isolinhas de deslocamento vertical



ANEXO - Fig. 6 - Linhas de SL=0,95



ANEXO - Fig. 7 - Vectores de tensão principal



ANEXO - Fig. 8 - Deslocamentos entre o final da construção (fase 4) e o final do carregamento (fase 10)

ANEXO II

Listagem do programa MBE

MBEV60.for

```
c*****************************************************************
c
c      Calculo de construcao e enchimento de Barragens
c       de aterro. Ver 6.* - Calculo de tensoes e formacao da
c       matriz de rigidez atendento a variacao de tensao dentro
c       do elemento, elementos isoparametricos
c       Modelo Hiperbolico e modelo EC-K0
c       rectangulares de 8 nos
c       triangulares de 6 nos
c       junta de 6 nos s/ espessura
c       junta de 6 nos c/ espessura
c
c*****************************************************************
      program MBEV60
      character*12 filein,filout,auxf
      character*80 titulo
      logical existe
      dimension a(20000)

      include 'datnum.for'
      include 'gau1.for'
      include 'gau2.for'
      include 'varios.for'
      include 'what.for'
      common/modcl/newfa
c
c       Ficheiro de dados
c
      call logon(1)

      write(*,'(a)')' Nome do ficheiro de dados ?'
      read(*,'(a12)')filein
      open(unit=1,status='old',file=filein)
      write(*,*)'Fase a repetir ou 0 p/ novo calculo -1 p/ leitura'
      read(*,*)newfa
c
c       Leitura de dados gerais
c
      read(1,'(a)')titulo
      read(1,*)numelt,numnpt,nap,nfel,numcel,nummat,nfases
      read(1,*)ng8,ng6,ngj
      read(1,*)iqt,tint
      if(ngj.le.ng8)then
            write(*,*)' Aumente a ordem de integracao'
            write(*,*)' para os elementos de junta'
            write(*,*)' Continuar ? (1) '
            read(*,*)ilixo
            if(ilixo.ne.1)stop
      endif
c
```

```
c
c       Abertura de ficheiros
c
c       4-Tensoes; 7-Resultados; 8-[D]; 9-Plotter;
c       10-Tensoes intermedias

        open(unit=4,status='unknown',form='unformatted')
        write(filout,'(a,a)')filein(1:index(filein,'.')),'lpt'
        inquire(file=filout,exist=existe)
        if(existe)then
              write(*,*)' O ficheiro de resultados ',filout,
     1  'ja existe'
              write(*,*)' apago ? (1)'
              read(*,*)iap
              if(iap.eq.1)then
                    open(unit=17,status='old',file=filout)
                    close(17,status='delete')
              else
                    stop' OK.'
              endif
        endif
        write(*,*)' Criado o ficheiro de resultados ',filout
        write(auxf,'(a,a)')filein(1:index(filein,'.')),'plo'

        open(unit=17,status='new',file=filout)
c       open(unit=8,status='scratch',access='direct',recl=90)
        open(unit=8,status='scratch',access='direct',recl=360)
        open(unit=9,status='unknown',file=auxf,form='unformatted')
        open(unit=7,status='scratch',form='unformatted')
        open(unit=10,status='scratch',access='direct',recl=3264)
c       open(unit=10,status='scratch',access='direct',recl=816)

        call logon(17)

        write(17,*)'Ficheiro de dados : ',filein
        write(17,2)titulo
        write(17,*)'MBE Ver. 6.0   de 90/10/17'
        call data(17)
        write(17,1)numelt,numnpt,nap,nfel,numcel,nummat,nfases
        write(17,3)ng8,ng6,ngj

        write(17,*)
        if(iqt.eq.1)then
              write(17,2)
     1  'Sao impressas as tensoes no centro dos elementos'
        elseif(iqt.eq.1)then
              write(17,2)
     1  'Sao impressas as tensoes nos pontos de Gauss'
        else
              write(17,'(a,a)')
     1  'Sao impressas as tensoes no centro dos ',
     1  'elementos e nos pontos de gauss'
```

```
      endif

      if(tint.eq.1)then
            write(17,2)'Sao impressas as tensoes da fase intermedia'
      endif

c     inicializar vectores

      write(*,*)'GAUSS'
      call gauss8(ng8,xg8)
      call gauss8(2,xgp)
      call gauss8(ngj,xgj)
      call gauss6(xg6)

      call matym(ym)
      call matyv(yv)
      call delta(idel)
c
c
c     Allocation
c
c     n1-coord+alt.ag.; n2-cond.front; n3-elem(4nos,mat,modelo,ncod,constr)
c     n4-forcas(x,y); n5-materiais; n6-strlev(act,max,18); n7-desl.; n8-KK
c     n9-dstr; n10-tensoes; n11-desl.tot.

      n1=1
      n2=n1+2*numnpt
      n3=n2+3*nap
      n4=n3+13*numelt
      n5=n4+2*numnpt
      n6=n5+25*nummat
      n7=n6+2*numelt*17
      n8=n7+2*numnpt
c
c
c     Leitura de dados (apoios,materiais,elementos,nos)
      write(*,*)'LEITURA DE DADOS'
      call leitur(a(n2),a(n5),a(n3),a(n1))
c
c     alocacao matriz de rigidez global
      numel=numelt
      write(*,*)'Dimensao do sistema'
      call bandw(a(n3),nbdt,net)
      n9=n8+net*nbdt
      n10=n9+3*numelt*17
      n11=n10+3*numelt*17

      write(*,'(1x,f8.2,a)')float(n11)/2000.*100.,'%'

c
c     Determinacao dos valores iniciais nos elementos (tensoes)
      write(*,*)'Valores iniciais'
```

```
      call valini(a(n5),a(n3),a(n6),a(n1),a(n7),a(n10))
c
c
c     Ciclo para cada uma das fases
      write(*,*)'FASES'
      do 10 i=1,nfases
            call fase(a(n2),a(n5),a(n7),a(n11),a(n9),a(n4),a(n3)
     1      ,a(n8),nbdt,a(n6),a(n1),a(n10),i,net)
10    continue

      call data(17)

      stop
1     format(//'  Numero de elementos                ',i3,/,
     1        ' Numero de nos                      ',i3,/,
     2        ' Numero de nos com apoio            ',i2,/,
     3        ' Numero de elementos da fundacao    ',i2,/,
     4        ' Numero de elementos pre-existentes ',i2,/,
     5        ' Numero de materiais                ',i2,/,
     6        ' Numero de fases                    ',i2)

3     format(//'  Numero de pontos de integracao'//
     1        ' Elementos rectangulares de 8 nos  ',i2/
     2        ' Elementos triangulares de 6 nos   ',i2/
     3        ' Elementos de junta de 6 nos       ',i2)

2     format(1x,a)
      end
c-------------------------------------------
      subroutine logon(unit)
      integer unit
      if(unit.eq.1)then
      write(*,*)' MM    MM   BBBBBB   EEEEEEE'
      write(*,*)' MMM  MMM   BB  BB   EE'
      write(*,*)' MM MMM MM   BBBBB    EEEEE'
      write(*,*)' MM  M  MM   BB  BB   EE'
      write(*,*)' MM    MM   BBBBBB   EEEEEEE'
      write(*,*)
      write(*,*)' Programa MBE Ver. 6.0'
      write(*,*)
      write(*,*)' J. Marcelino'
      write(*,*)
      else
      write(unit,*)' MM    MM   BBBBBB   EEEEEEE'
      write(unit,*)' MMM  MMM   BB  BB   EE'
      write(unit,*)' MM MMM MM   BBBBB    EEEEE'
      write(unit,*)' MM  M  MM   BB  BB   EE'
      write(unit,*)' MM    MM   BBBBBB   EEEEEEE'
      write(unit,*)
      write(unit,*)' Programa MBE Ver. 6.0'
      write(unit,*)
      write(unit,*)' J. Marcelino'
```

```
      write(unit,*)
      endif
      return
      end
c-----------------------------------------------
      subroutine leitur(apoio,cmat,iele,xy)
c
c     Le e escreve dados dos materiais e geometria
c
      include 'datnum.for'
      common/pat/patm
      real xy(numnpt,2),cmat(nummat,25)
      integer iele(numelt,13),apoio(nap,3)
c
c
c     le pressao atmosferica
      write(*,*)'LEITURA de :'
      read(1,*)patm
      write(17,1)patm
c
c     Le caracteristicas dos materiais
      write(*,*)'MATERIAIS'
      call lemat(cmat,25)
c
c     Le coordenadas dos nos
      write(*,*)'COORDENADAS'
      call lexy(xy)
c
c     le condicoes de apoio
      write(*,*)'APOIOS'
      call leap(apoio,xy,3)
c
c     le caracteristicas dos elementos
      write(*,*)'ELEMENTOS'
      call leele(iele,13)
c
      write(*,*)'Calcular posicao dos nos centrais (1/0)'
      read(*,*)ica
      if(ica.eq.1)then
            call posc(xy,iele)
      endif
      call escrno(apoio,xy,3)

      return
1     format(//,'  Pressao atmosferica: ',f6.2)
      end
c.....
c
      subroutine lexy(xy)
c
c
      include 'datnum.for'
```

```
      real xy(numnpt,2)
      read(1,*)no1,xy(no1,1),xy(no1,2)
      do 1 ijj=1,numnpt
            if(no1.eq.numnpt)return
            read(1,*)no2,xy(no2,1),xy(no2,2)
            if(no1+1.ne.no2)then
              deltax=(xy(no2,1)-xy(no1,1))/float(no2-no1)
              deltay=(xy(no2,2)-xy(no1,2))/float(no2-no1)
              do 10 ikk=no1+1,no2-1
                    xy(ikk,1)=xy(ikk-1,1)+deltax
                    xy(ikk,2)=xy(ikk-1,2)+deltay
10            continue
            end if
      no1=no2
1     continue
c
      return
      end


c.....
      subroutine lemat(cmat,num)
c
c     subrotina que le as carcateristicas dos materiais
c
      include 'datnum.for'
      common/mat/mattip
      real cmat(nummat,num)
      character*5 mattip(20)
c
      write(17,2)
      do 1 i=1,nummat
c
c           Le caracteristicas dos materiais
            read(1,'(a)')mattip(i)
            read(1,*)(cmat(i,j),j=1,num)
c
c           Escreve no ficheiro de resultados
            write(17,3)i,mattip(i)
            if(cmat(i,10).ne.-1.)then
                  write(17,4)(cmat(i,j),j=1,num)
            else
                  write(17,5)(cmat(i,j),j=1,num)
            endif

1     continue
c

      return
2     format(///,'   Caracteristicas dos materiais',/,
     1  ' ------------------------',/)
3     format(/,'  Material: ',i2,2x,a,//)
```

```
4       format(' Gama=',f7.2,9x,'K=',f10.1,9x,'Kur='f8.1,9x,
     1  'n=',f10.3,9x,'d=',f10.3,/,
     2  ' G=',f10.3,9x,'F=',f10.3,9x,'c=',f10.3,9x,'fi0=',f8.1,9x,
     3  'dfi=',f8.1,/,' Rf=',f9.3,9x,'S3lim=',f6.2,9x,'beta=',f7.5,
     4  9x,'S3t=',f8.2,9x,'Ko=',f9.3,/,' dgama=',f6.3,9x,'Ako=',
     5  f8.3,9x,'Bk0=',f8.3,9x,'Ck0=',f8.3,9x,'Dk0=',f8.3,/,
     6  ' Ae=',f9.1,9x,'Be=',f9.1,9x,'Ce=',f9.1,9x,'De=',f9.1,9x,
     7  'alfa=',f7.4)
5       format(' Gama=',f7.2,9x,'Kt=',f10.1,8x,'Ktur='f8.1,8x,
     1  'n=',f10.3,9x,'Kn=',f11.0,/,
     2  ' =',f10.3,9x,' =',f10.3,9x,'c=',f10.3,9x,'fi0=',f8.1,9x,
     3  'dfi=',f8.1,/,' Rf=',f9.3,9x,'S3lim=',f6.2,9x,'beta=',f7.5,
     4  9x,'S3t=',f8.2,9x,'Ko=',f9.3,/,' dgama=',f6.3,9x,'Ako=',
     5  f8.3,9x,'Bk0=',f8.3,9x,'Ck0=',f8.3,9x,'Dk0=',f8.3,/,
     6  ' Ae=',f9.1,9x,'Be=',f9.1,9x,'Ce=',f9.1,9x,'De=',f9.1,9x,
     7  'alfa=',f7.4)

        end
c.....
        subroutine posc(xy,iele)
        include 'datnum.for'
        real xy(numnpt,2)
        integer iele(numelt,13),iq(4,3),it(3,3)
        data ((iq(i,j),j=1,3),i=1,4)/5,1,2,6,2,3,7,4,3,8,1,4/
        data ((it(i,j),j=1,3),i=1,3)/4,1,2,5,2,3,6,3,1/
        do 10 i=1,numelt
                if(iele(i,13).eq.1)then
c                       quadrangular
                        do 11 j=1,4
                                call med(xy,iele,iq(j,1),iq(j,2),iq(j,3),i,1)
                                call med(xy,iele,iq(j,1),iq(j,2),iq(j,3),i,2)
11                      continue

                elseif(iele(i,13).eq.2)then
c                       triangular
                        do 12 j=1,3
                                call med(xy,iele,it(j,1),it(j,2),it(j,3),i,1)
                                call med(xy,iele,it(j,1),it(j,2),it(j,3),i,2)
12                      continue
                elseif(iele(i,13).eq.3.or.iele(i,13).eq.4)then
c               elemento de junta
                else
                        write(*,*)'erro no tipo de elemento'
                        write(*,*)'elemento ',i
                        write(*,*)(iele(i,j),j=1,13)
                endif
10      continue
        return
        end
c-----------------------------------------------------------------
        subroutine med(xy,iele,i,j,k,n,l)
        include 'datnum.for'
```

```
      real xy(numnpt,2)
      integer iele(numelt,13)
      xy(iele(n,i),l)=(xy(iele(n,j),l)+xy(iele(n,k),l))/2.
      return
      end
c-----------------------------------------------------------------
      subroutine leele(iele,num)

c     'salto' dos nos para elementos 1=8nos 2=6 nos
c
c
      include 'datnum.for'

      integer iele(numelt,num),igap(8,2)
      data ((igap(i,j),i=1,8),j=1,2)/2,2,2,2,2,1,2,1,2,2,2,2,2,2,0,0/

c     iele(n,1-8)- nos do elemento n
c     iele(n,9)  - material do el. n
c     iele(n,10)  - modelo reologico do el. n
c     iele(n,11)   - codigo de molhagem "
c     iele(,13)    - tipo de elemento finito
c                    =1 isoparam rectangular de 8 nos
c                    =2 isoparam triangular de 6 nos
c                    =3 isoparam junta de 6 nos
c                    =4 isoparam. junta 6 nos c/ espessura

c
      read(1,*)nel1,(iele(nel1,j),j=1,13)
      do 1 ijj=1,numelt
           if(nel1.eq.numelt)return
           read(1,*)nel2,(iele(nel2,j),j=1,13)
           if(nel1+1.ne.nel2)then

c            elementos intermedios
             do 10 ikk=1,nel2-nel1-1
                if(iele(nel1,13).eq.1)then
                  k=1
               else
                  k=2
                endif
c               atribuicao dos numeros dos nos
                do 20 jj=1,8
                  iele(nel1+ikk,jj)=iele(nel1+ikk-1,jj)+igap(jj,k)
20              continue

c               atribuicao de material, modelo, cod. molhagem
                do 30 jj=9,num
                  iele(nel1+ikk,jj)=iele(nel1,jj)
30              continue
10           continue
           end if
      nel1=nel2
```

```
1       continue
        return
        end
c.....
        subroutine escrno(apoio,xy,num)

        include 'datnum.for'
        integer apoio(nap,num)
        real xy(numnpt,2)
c
c       Escreve dados relativos aos nos em duas colunas de lim n"s cada
        write(17,2)
        lim=iabs(int(float(numnpt)/2.-.5))+1
c       write(*,*)'lim',lim
        do 3 i=1,2*lim,2
                no1=i
                no2=i+1
c               determina condicoes de apoio
                iapx1=0
                iapy1=0
                iapx2=0
                iapy2=0
                do 5 j=1,nap
                  if(apoio(j,1).eq.no1)then
                        iapx1=apoio(j,2)
                        iapy1=apoio(j,3)
                    elseif(apoio(j,1).eq.no2)then
                        iapx2=apoio(j,2)
                        iapy2=apoio(j,3)
                    endif
5               continue
                if(no2.le.numnpt)then
                write(17,4)no1,(xy(no1,j),j=1,2),iapx1,iapy1,
     1          no2,(xy(no2,j),j=1,2),iapx2,iapy2
                else
                write(17,4)no1,(xy(no1,j),j=1,2),iapx1,iapy1
                endif
3       continue
        return

2       format(//,'  Dados relativos aos nos',/,'----------------------',
     1  //,2(2x,'NO',6x,'COORD.  DOS NOS',10x,'COND. FRONTEIRA',11x),/
     2  2(' NUMERO',4x,'XX',8x,'YY',15x,'XX',3x,'YY',15x),/)

4       format(2(1x,i3,4x,f7.2,3x,f7.2,14x,i1,4x,i1,15x))

        end
c.....
        subroutine leap(apoio,xy,num)
c
c       subrotina para ler as condicoes de apoio e os deslocamentos impostos
c
```

```
      include 'datnum.for'
      integer apoio(nap,num)
      real xy(numnpt,2)
c
c     apoio(i,1) - numero do n'o com apoio
c     apoio(i,2) = 1. - no' apoio(i,1) tem restricao segundo xx
c     apoio(i,3) = 1. - no' apoio(i,1) tem restricao segundo yy
c
      read(1,*)iap1,(apoio(iap1,j),j=1,3)
      do 1 ijj=1,nap
            if(iap1.ne.nap)then
              read(1,*)iap2,(apoio(iap2,j),j=1,3)
              if(iap1+1.ne.iap2)then
c
c               apoios intermedios
                do 10 ikk=1,iap2-iap1-1
c                     passa para o n'o seguinte
                      apoio(iap1+ikk,1)=apoio(iap1,1)+float(ikk)
c                     mantem as mesmas condicoes de apoio e desl.
                      do 20 jj=2,num
                        apoio(iap1+ikk,jj)=apoio(iap1,jj)
20                    continue
10              continue
              end if
            end if
            iap1=iap2
1     continue
      return
      end
c-----------------------------------------------------------------
      subroutine fase(apoio,cmat,d,deslt,dstr,f,iele,k,nbdt,slev,xy
     1  ,stress,ifase,net)
c
c     Subrotina que identifica e calcula o tipo de fase a simular
c
      include'datnum.for'
      common/novel/nel1,nel2
      common/desl/deslimp
      common/camada/nomax
      common/mag/nomag
      common/modcl/newfa
      common/descarga/ndes,iel(200)

      character*3 fascod
      logical erro
      real cmat(nummat,25),d(net),f(net),stress(numelt,3,17),deslt(net)
      real k(net,nbdt),slev(numelt,2,17),xy(numnpt,2),dstr(numelt,3,17)
      integer iele(numelt,13),apoio(nap,3),anul(10,2)
c
c     inicia cod. del. imp., vector das forcas e nel1,nel2
      deslimp=0.
      nel1=0
```

```
      nel2=0
      call inicv(f,2*numnpt)
      call inic3(dstr,numelt,3,17)
      write(17,7)ifase
      write(*,7)ifase
c
c     sempre que ndes=0 nao ha elementos em descarga
      ndes=0
      read(1,*)fascod,numel

      fmag=1.
      nomag=0
      call maxno(iele)
c
c     Determina semibanda
      write(*,*)'BANDW'
      if(numel.ne.numelt)then
            call bandwd(iele)
        else
            nbd=nbdt
            ne=net
      end if
c
      if(fascod(1:2).eq.'#1')then
c           fase de construcao
            call constr(cmat,f,iele,xy)
            niter=2
            if(fascod(3:3).eq.'1')then
c                 fase de aplicacao de cargas
c                 logo apos construcao
                  nomag=1
                  call cargas(f,xy,iele)
            endif
c
      elseif(fascod.eq.'#2 ')then
c           fase de aplicacao de cargas
            call cargas(f,xy,iele)
            niter=2
c
      elseif(fascod.eq.'#3a')then
c           fase de colapso + aumento p.p.
            call colaps(cmat,stress,dstr,f,iele,xy,net)
c           so uma iteracao pois nao ha alteracao do
c           estado de tensao
            niter=2
c
      elseif(fascod.eq.'#3b')then
c           fase de impulsao + forca nas fronteiras
            call impuls(f,iele,xy,cmat,net)
            niter=2
c
      elseif(fascod.eq.'#4 ')then
```

```
c               fase de fluencia
c               call fluen(cmat,dstr,f,iele)
                niter=1
c
        else
c               deslocamento imposto
c               call deslim
                niter=2
        end if
c
c       le o numero de escaloes de aplicacao das forcas
        read(1,*)nesc
c
c       ciclo de escaloes
c       escreve e divide as forcas pelos escaloes
        call escrf(f)
        do 3 i=1,ne
                f(i)=f(i)/float(nesc)
3       continue

c       leitura dos nos a anular os deslocamentos
        read(1,*)nda
        do 25 i=1,nda
                read(1,*)anul(i,1),anul(i,2)
25      continue
        write(17,*)'Numero de escaloes para esta fase:',nesc
        if(newfa.eq.-1)goto 1101
        if(ifase.eq.newfa)then
                call locate(newfa-1,nml,mxno,erro)
                if(.not.erro)then
                        call leten(stress,nml)
                        call readd(d,deslt,net,mxno)
                        call lelev(slev,nml)
                else
                        write(*,*)'Erro na rep. de calculo'
                endif
        elseif(ifase.lt.newfa)then
                write(17,*)'Fase antiga'
                goto 1101
        else
                write(17,*)'Fase nova'
        endif

        do 1 i=1,nesc
c
c           ciclo de iteracoes
            do 2 it=1,niter
                write(17,*)' Iteracao ',it
c
                write(*,*)'RESOL'
                call resol(xy,apoio,cmat,d,deslt,dstr,f,iele,k,slev
     1          ,stress,it,niter,anul,nda,i,nesc)
```

```
2          continue
c
1       continue
c
c       Escreve tensoes e deslocamentos no ficheiro para plotter
        write(9)'FASE',ifase,numel,nomax
        do 4 n=1,numel
4       write(9)((stress(n,i,ik),i=1,3),ik=1,17)
        do 5 n=1,nomax
5       write(9)d(2*n-1),d(2*n),deslt(2*n-1),deslt(2*n)
        do 6 n=1,numel
6       write(9)((slev(n,i,ik),i=1,2),ik=1,17)

c
c       Escreve deslocamentos no ficheiro de resultados
        write(17,11)'Deslocamentos','----------',' N°',
     1  'Desl. parciais','Desl. totais','XX','YY','XX','YY','Result.'
        do 12 i=1,ne/2
                desloc=sqrt(deslt(2*i-1)**2.+deslt(2*i)**2.)
                write(17,13)i,d(2*i-1),d(2*i),deslt(2*i-1),
     1          deslt(2*i),desloc
12      continue
1101    return
c
7       format(////1x,18('.   '),/,'  .',33x,'.',/
     1                ' .   Fase no. ',i2,18x,'  .',/,
     2  '  .',33x,'.',/1x,18('.   '),////)
11      format(///1x,a13,/1x,a13,///,
     1  2x,a2,8x,a14,20x,a12,/,12x,a2,10x,a2,13x,a2,10x,a2,8x,a7,/)
13      format(1x,i3,2(5x,f7.4),3x,3(5x,f7.4))

        end
c.....
c       localiza uma fase no ficheiro de dados para plotter
        subroutine locate(ifase,numel,maxno,erro)
        character*4 txt
        logical erro
        rewind 9
c       ciclo
        erro=.false.
2       read(9,end=100)txt
        if(txt.eq.'FASE')then
                backspace 9
                goto 1
        else
                goto 2
        endif
1       read(9,end=100)txt,if,numel,maxno
        if(if.ne.ifase)goto 2
c       encontrou
        nume=numel
        return
```

```
100     write(*,*)'A fase pedida nao existe'
        erro=.true.
        return
        end
c.....
c       le deslocamentos pretendidos
        subroutine readd(d,deslt,net,mxno)
        include 'datnum.for'
        real d(net),deslt(net)
        do 5 n=1,mxno
5       read(9)d(2*n-1),d(2*n),deslt(2*n-1),deslt(2*n)
        write(4)(deslt(n),n=1,2*numnpt)
        return
        end
c.....
        subroutine lelev(slev,nml)
        include 'datnum.for'
        real slev(numelt,2,17)
        do 6 n=1,nml
6       read(9)((slev(n,i,ik),i=1,2),ik=1,17)
        return
        end
c.....
        subroutine leten(stress,nml)
        include 'datnum.for'
        real stress(numelt,3,17)
        rewind 4
        do 12 n=1,numelt
12      read(4)((stress(n,i,ik),i=1,3),ik=1,17)
        rewind 4
        do 10 n=1,nml
        read(9)((stress(n,i,ik),i=1,3),ik=1,17)
10      continue
        do 11 n=1,numelt
11      write(4)((stress(n,i,ik),i=1,3),ik=1,17)
        return
        end
c---------------------------------------------------------
        subroutine impuls(f,iele,xy,cmat,net)

        include'datnum.for'
        include 'gau1.for'
        include 'gau2.for'
        include 'varios.for'
        integer iele(numelt,13),it(3,3),iq(4,3)
        real cmat(nummat,25),f(net),xy(numnpt,2)
        real xe(8,2),pe(3,2),fe(8,2),q(2)
        common/pat/patm
        common/descarga/ndes,iel(200)
        data ((it(i,j),j=1,3),i=1,3)/1,2,4,2,3,5,1,3,6/
        data ((iq(i,j),j=1,3),i=1,4)/1,2,5,2,3,6,4,3,7,1,4,8/
```

```
      h2o=patm/10.33

      read(1,*)nsub
c     leitura dos elementos submersos
      read(1,*)(iel(i),i=1,nsub)
c     no vector iel() guardam-se quais sao os elementos que
c     devem obrigatoriamente estar em descarga
      ndes=nsub
c     impulsao = peso do volume de liquido deslocado
      q(1)=0
      write(17,*)'Elementos submersos'
      write(17,*)(iel(i),i=1,nsub)
      write(17,*)
      do 10 i=1,nsub
c           aplicacao da impulsao

            q(2)=h2o*cmat(iele(iel(i),9),1)/26.5
            write(*,*)'q(2)=',q(2)
            call tranxe(iele,xy,xe,iel(i))

            call tranxe(iele,xy,xe,iel(i))

            goto (1,2,3) iele(iel(i),13)

1           call fovol8(q,xe,idel,1.,xg8,ym,fe)
            nno=8
            goto 4

2           call fovol6(q,xe,idel,1.,xg6,fe)
            nno=6
            goto 4

c           elemento de junta n. tem implusao
3           continue

4           call distr(f,fe,iele,iel(i),nno)

10    continue

      read(1,*)nfd
      write(17,*)'Impulsao nas fronteiras'
      write(17,'(/1x,a3,3x,a3,2(3x,a10))')'Ele',' L ',' H1',' H2'
      do 20 i=1,nfd
            read(1,*)ie,lado,h1,h2
            write(17,'(1x,i3,3x,i3,2(3x,f10.5))')ie,lado,h1,h2
            if(iele(ie,13).eq.1)then
                  no1=iele(ie,iq(lado,1))
                  no2=iele(ie,iq(lado,2))
            elseif(iele(ie,13).eq.2)then
                  no1=iele(ie,it(lado,1))
                  no2=iele(ie,it(lado,2))
            else
```

```
                write(*,*)'Elemento de junta c/ forcas ?'
            endif
            x1=xy(no1,1)
            x2=xy(no2,1)
            y1=xy(no1,2)
            y2=xy(no2,2)
            if(x1-x2.ne.0)then
                alf=atan2((y2-y1),(x2-x1))
            else
                alf=1.570796327
            endif
            c=cos(alf)
            s=sin(alf)
            pe(1,1)=h1*h2o*s
            pe(2,1)=h2*h2o*s
            pe(3,1)=(h1+h2)/2.*h2o*s
            pe(1,2)=-h1*h2o*c
            pe(2,2)=-h2*h2o*c
            pe(3,2)=-(h1+h2)/2.*h2o*c
            write(17,*)no1,no2
            write(17,*)'ALF=',alf*180/3.14159
            write(17,*)' X',pe(1,1),pe(2,1),pe(3,1)
            write(17,*)' y',pe(1,2),pe(2,2),pe(3,2)
c           transferir coordenadas do elemento
            call tranxe(iele,xy,xe,ie)

c           goto de acordo com o tipo de elemento
            goto(31,32,33,35) iele(ie,13)

31          nno=8
            call fosup8(lado,pe,xg8,1.,xe,idel,ym,fe)
            goto 34

32          nno=6
            call fosup6(lado,pe,xg8,1.,xe,idel,fe)
            goto 34

33          write(*,*)' Forcas aplicadas em el de junta ?'
            stop

35          nno=6
            call fosupp(lado,pe,xg8,1.,xe,yv,xgj,fe)
            goto 34

34          call distr(f,fe,iele,ie,nno)

20      continue

        return
        end
c-------------------------------------------------
        subroutine maxno(iele)
```

```
      include'datnum.for'

      integer iele(numelt,13)
      common/camada/nomax

      nomax=0
      do 10 i=1,numel
            do 20 j=1,8
                  nomax=max0(nomax,iele(i,j))
20          continue
10    continue
c     write(*,*)' No max =',nomax
      return
      end
c-------------------------------------------------------------
      subroutine constr(cmat,f,iele,xy)
c
c     calcula as forcas devidas 'a construcao de elementos
c
      include        'datnum.for'
      include 'gau1.for'
      include 'gau2.for'
      include 'varios.for'

      common/novel/nel1,nel2
c
      real cmat(nummat,25),f(ne),xy(numnpt,2),xe(8,2),fe(8,2),q(2)
      integer iele(numelt,13)
c
c     Le o numero da camada e o 1o. e ultimo elem. a serem construidos
      read(1,*)icamada,nel1,nel2
      write(17,11)nel1,nel2
c
c     ciclo para construcao de cada elemento
c
      do 10 n=nel1,nel2

c           muda codigo de construcao
            iele(n,12)=1

c           densidade de carga
            q(1)=0.
            q(2)=-cmat(iele(n,9),1)

c           transferencia de coordenadas do elemento
            call tranxe(iele,xy,xe,n)

            goto (1,2,3,5) iele(n,13)

1           call fovol8(q,xe,idel,1.,xg8,ym,fe)
            nno=8
            goto 4
```

```
2               call fovol6(q,xe,idel,1.,xg6,fe)
                nno=6
                goto 4

c               elemento de junta n. tem peso
3               goto 10

5               call fovolp(q,xe,1.,xg8,yv,fe)
                nno=6
                goto 4
4               call distr(f,fe,iele,n,nno)

10      continue

        return
11      format(1x,'Construcao dos elementos ',i3,' a ',i3)

        end

c-------------------------------------------------------------
        subroutine assemb(k,kk,ne,nbd)
        real kk(16,16),k(ne,nbd)
        integer lin,col,lin1,col1
        common /des/  idg(16)
        do 100 ik=1,16
                do 101 j=1,16
                        lin=idg(ik)
                        col=idg(j)-lin+1
                        if(col.gt.0)then
                                lin1=ik
                                col1=j
                                if(idg(ik).ne.0.and.idg(j).ne.0)then
                                k(lin,col)=k(lin,col)+kk(lin1,col1)
                                endif
                        endif
101             continue
100     continue
        return
        end
c-------------------------------------------------------------
        subroutine escrf(f)
c
c       Escreve as forcas (nao nulas)
c
        include 'datnum.for'
c
        real f(ne),forcxy(5,2)
        integer nno(5)

        open(unit=15,form='unformatted',status='scratch')
c
c       Ciclo para os pontos
```

```
c
      fxt=0
      fyt=0
      icount=0
      do 1 i=1,ne/2
            if(f(2*i-1).ne.0. .or. f(2*i).ne.0.)then
               write(15)i,f(2*i-1),f(2*i)
               fxt=fxt+f(2*i-1)
               fyt=fyt+f(2*i)
               icount=icount+1
            end if
1     continue
c
      rewind 15

c     cabecalho
      write(17,12)'Forcas','------'
      write(17,11)' NO','Forca X','Forca Y',' NO','Forca X'
     1,'Forca Y',' NO','Forca X','Forca Y',' NO','Forca X','Forca Y'

      do 2 i=1,numnpt/4+1

            icl=4

            if(icl.gt.icount)then
                  icl=icount
                  icount=0
            else
                  icount=icount-4
            endif

            do 3 j=1,icl
                  read(15)nno(j),forcxy(j,1),forcxy(j,2)
3           continue

            write(17,10)(nno(j),(forcxy(j,k),k=1,2),j=1,icl)

            if(icount.eq.0)then
                  close(15)
                  write(17,*)' Somatorio de forcas'
                  write(17,'(2x,a,f15.5)')'S fx=',fxt
                  write(17,'(2x,a,f15.5)')'S fy=',fyt
                  return
            endif

2     continue

12    format(///,2(1x,a6,/))
10    format(1x,4(i3,2(2x,f9.2),6x))
11    format(/1x,4(a3,2(4x,a7),6x)/)

      end
```

```
c.....
c       seleciona cabecalhos
        function icab(i)
        goto (1,2,3,4),i
1       icab=1
        return
2       icab=1
        return
3       icab=3
        return
4       icab=4
        return
        end
c.....
        subroutine escrte(ext,n,stress,kk,iele,xy)
c
c       escreve valores das tensoes,extensoes
c
        include 'datnum.for'
        include 'what.for'
        common/novel/nel1,nel2
c
        real stress(numelt,3,17),ext(3),tens(3),tenspr(3),extpr(3)
        real dd(3,3,10),d(3,3),d1(3,3),t(3),xy(numnpt,2)
        integer iele(numelt,13)
c
        if(n.eq.1)then
                ihead=iele(n,13)
                write(17,1)'Tensoes','-------'
        else
                if(icab(iele(n-1,13)).eq.icab(iele(n,13)))then
c                       nao muda cabecalho
                        ihead=0
                else
c                       novo cabecalho
                        ihead=iele(n,13)
                endif
        endif
        if(ihead.eq.1.or.ihead.eq.2)then
c               escreve cabecalho
                write(17,21)'Ele','PG',' Sig X ',' Sig Y ',' Tau XY',
     1          ' Sig 1 ',' Sig 3 ',' Theta',' Eps X',' Eps Y',
     2          'GamaXY',' Eps 1',' Eps 3','Ele'
        elseif(ihead.eq.3)then
                write(17,21)'Ele','PG',' Tau ',' Sig N ',
     1          ' Gama ',' Eps N','Ele'
        elseif(ihead.eq.4)then
                write(17,21)'Ele','PG',' Sig X ',' Sig Y ',' Tau XY',
     1          ' Sig T ',' Tau ',' SIG N ',' GAMA ',' Eps N'
        else
c               nao faz nada mas guarda ihead
                ihead=iele(n,13)
```

```
      end if
c
      read(8,rec=n)dd
      e1=0.
      e2=0.
      e12=0.

      do 30 k=1,kk
            call inicv(ext,3)
            do 2 i=1,3
2           tens(i)=stress(n,i,k)
c
            if(ihead.eq.1.or.ihead.eq.2)then
                  call princi(tens,tenspr)
            elseif(ihead.eq.4)then
                  call roda(iele,xy,tens,k,n,t)
            else
            endif

            if(n.ge.nel1.and.n.le.nel2)then
c                 elementos novos, nao calcular extensoes
                  ext(1)=.9999
                  ext(2)=.9999
                  ext(3)=.9999
            else
c                 transfere a matriz D
                  call tradd(dd,d1,k)
C                elemento diagonal 3,3 =1 nos el. junta
                  if(ihead.eq.3)d1(3,3)=1
c                 inverte a matriz D, para e=D-1.S
                  call invert(d1,3,d)
c                 calcula extensoes
                  call multv(d,tens,ext,3,3)
                  if(ihead.ne.3)call princi(ext,extpr)
            endif
c           acumula para a media
            e1=e1+ext(1)
            e2=e2+ext(2)
            e12=e12+ext(3)
            if(iqt.eq.2.or.iqt.eq.3)then
                  if(ihead.eq.2.or.ihead.eq.1)then
                  write(17,3)n,k,(tens(i),i=1,3),(tenspr(i),i=1,3),
     1            (ext(i)*100.,i=1,3),(extpr(i)*100.,i=1,2),n
                  elseif(ihead.eq.3)then
                  write(17,4)n,k,(tens(i),i=1,2),
     1          (ext(i)*100.,i=1,2),n
                  else
                  write(17,3)n,k,(tens(i),i=1,3),(t(i),i=1,3),
     1            (ext(i)*100.,i=1,3),(extpr(i)*100.,i=1,2),n
                  endif
            endif
30    continue
```

```
      if(iqt.eq.1.or.iqt.eq.3)then
       k=10
c      extensao media
       ext(1)=e1/float(kk)
       ext(2)=e2/float(kk)
       ext(3)=e12/float(kk)
       do 5 i=1,3
5      tens(i)=stress(n,i,k)
c
       if(ihead.eq.2.or.ihead.eq.1)then
          call princi(tens,tenspr)
          call princi(ext,extpr)
c
          write(17,3)n,0,(tens(i),i=1,3),(tenspr(i),i=1,3),
     1    (ext(i)*100.,i=1,3),(extpr(i)*100.,i=1,2),n
       elseif(ihead.eq.3)then
          write(17,4)n,0,(tens(i),i=1,2),(ext(i)*100.,i=1,2),n
       else
          call roda(iele,xy,tens,k,n,t)
          write(17,3)n,0,(tens(i),i=1,3),(t(i),i=1,3),
     1    (ext(i)*100.,i=1,3),(extpr(i)*100.,i=1,2),n
       endif
      endif
      if(iqt.eq.2.or.iqt.eq.3)write(17,*)
      return

1     format(//,2(/1x,a7))
21    format(/,1x,a3,1x,a3,6(4x,a7),5(4x,a6),4x,a3,/)
3     format(1x,i3,1x,i3,6(4x,f7.2),5(4x,f6.2),4x,i3)
4     format(1x,i3,1x,i3,2(4x,f7.2),4x,f7.2,4x,f7.2,6x,i3)
      end

c-------------------------------------------------
      subroutine roda(iele,xy,tens,k,n,t)

      include 'datnum.for'
      include 'gau1.for'
      include 'gau2.for'
      include 'varios.for'

      real jj(2,2),y(2),yy(2,2),aa(6,2),t(3)
     1,xe(8,2),ss(2,2),r(2,2),tens(3),xy(numnpt,2)
      integer iele(numelt,13)

      call inic(ss,2,2)
      ss(1,2)=tens(3)
      ss(2,2)=tens(2)
      ss(1,1)=tens(1)
      ss(2,1)=ss(1,2)
      call tranxe(iele,xy,xe,n)
      ig=k/3+1
      jg=((-1)**k+1)/2+1
```

```
      y(1)=xgp(ig,1)
      y(2)=xgp(jg,1)
      hg=xgp(ig,2)*xgp(jg,2)
      call tranjp(y,yv,xe,jj,yy,det,aa)
c     S'=TxSxTt
      yy(1,1)=jj(1,1)
      yy(1,2)=jj(1,2)
      yy(2,1)=-jj(1,2)
      yy(2,2)=jj(1,1)

      det=sqrt(yy(1,1)**2+yy(1,2)**2)
      do 10 i=1,2
            do 10 j=1,2
10    yy(i,j)=yy(i,j)/det

      call mult(yy,ss,r,2,2,2)
      call transp(yy,2,2)
      call mult(r,yy,ss,2,2,2)
      write(*,*)ss(1,1),ss(1,2)
      write(*,*)ss(2,1),ss(2,2)
      t(1)=ss(1,1)
      t(2)=ss(1,2)
      t(3)=ss(2,2)
      return
      end
c-----------------------------------------
      subroutine mult(a,b,c,n1,n2,n3)
c     C=A x B
      real a(n1,n2),b(n2,n3),c(n1,n3)
      call inic(c,n1,n3)
      do 10 i=1,n1
            do 30 k=1,n3
                  cc=0
                  do 20 j=1,n2
                        cc=cc+a(i,j)*b(j,k)
20                  continue
                  c(i,k)=cc
30          continue
10    continue
      return
      end
c-----------------------------------------
      subroutine transp(a,n1,n2)
      real a(n1,n2)
      do 10 i=1,n1
            do 20 j=1,n2
                  x=a(i,j)
                  a(i,j)=a(j,i)
                  a(j,i)=x
20          continue
10    continue
      return
```

```
      end
c-----------------------------------------------------------
      subroutine tradd(dd,d,n)
      real dd(3,3,10),d(3,3)
      do 10 i=1,3
            do 10 j=1,3
10                d(i,j)=dd(i,j,n)
      return
      end
c-----------------------------------------------------------
      subroutine invert(a,n,b)

      real a(n,n),b(n,n)

      call inic(b,n,n)
      do 10 i=1 , 3
        b(i,i)=1.
10    continue

      do 20 i=1 , 3
            diag=a(i,i)
            do 30 j=1 , 3
                  a(i,j)=a(i,j)/diag
                  b(i,j)=b(i,j)/diag
30          continue
            do 40 j=1 , 3
                  if (i.ne.j) then
                        fact=a(j,i)
                        do 50 m=1 , 3
                              a(j,m)=a(j,m)-a(i,m)*fact
                              b(j,m)=b(j,m)-b(i,m)*fact
50                      continue
                  endif
40          continue
20    continue
      return
      end
c-----------------------------------------
      subroutine multv(a,b,c,ii,ij)
c
c     Multiplicacao de uma matriz por um vector : {C} = [A].{B}
c
      real a(ii,ij),b(ij),c(ii)
      call inicv(c,ii)
      do 1 i=1,ii
         do 2 j=1,ij
                  c(i)=c(i)+a(i,j)*b(j)
2        continue
1     continue
      return
      end
c************************************************
```

```
c
c       interface entre MBE as subs para
c       elementos isoparametricos
c
c***********************************************
        subroutine matkk(dd,kk,iele,xy,ie)

c       dd = matriz Sigma - Eps
c       kk = matriz de rigidez
c       iele = matriz dos elementos
c       xy = matriz de coordenadas dos pontos nodais
c       ie = numero do elemento a calcular a matriz kk

        real dd(3,3,10),se(8,2,3,17),xy(numnpt,2),kk(16,16)
        real xe(8,2)

        integer iele(numelt,13),tipo

        include 'datnum.for'
        include 'gau1.for'
        include 'gau2.for'
        include 'varios.for'

        tipo=iele(ie,13)

c       1=iso de 8 nos, 2=iso de 6 nos, 3=junta

c       transferir coordenadas do elemento para o
c       vector de trabalho xe()

        call tranxe(iele,xy,xe,ie)

c       calculo das matrizes
        goto (10,11,12,13) tipo

c       NOTA: todas as espessuras sao consideradas unitarias
10      call fem8(xg8,xe,dd,idel,1.,ym,kk,se)
        goto 100

11      call fem6(xg6,xe,dd,idel,1.,kk,se)
        goto 100

12      call femj(dd,xe,xgj,1.,kk,se)
        goto 100

13      call femp(1.,xgp,xe,yv,dd,kk,se)
        goto 100

c       calculadas as matrizes guarda-se a matriz de tensao

100     write(10,rec=ie)((((se(i,j,k,m),i=1,8),j=1,2),k=1,3),m=1,17)
        return
```

```
      end

c.....
c
c     transferir coordenadas p/ XE
c
c.....
      subroutine tranxe(iele,xy,xe,ie)

      include 'datnum.for'

      integer iele(numelt,13)
      real xy(numnpt,2),xe(8,2)

      goto (1,2,2,2) iele(ie,13)
1     nno=8
      goto 4
2     nno=6

4     do 5 i=1,nno
            xe(i,1)=xy(iele(ie,i),1)
            xe(i,2)=xy(iele(ie,i),2)
5     continue
      return
      end
c.....
      subroutine kkglob(xy,apoio,cmat,iele,f,slev,k,it,fmag)
c
c     calcula matriz de rigidez do sistema
c
      include 'datnum.for'

      common/desl/deslimp
      common/novel/nel1,nel2
      common/mag/nomag
      common/descarga/ndes,iel(200)
      common/moddesc/idesc

      real k(ne,nbd),tens(3,17),d(3,3),kk(16,16),f(ne),t(3),dd(3,3,10)
      real cmat(nummat,25),slev(numelt,2,17),sig(3),xy(numnpt,2)
      integer iele(numelt,13),apoio(nap,3)
c
      rewind 4
c     8 - guarda [D]
      call inic(k,ne,nbd)

c     atencao !!! alteracao exactamente para o contrario
      if(it.eq.1)then
            irot=1
c           atencao quando for nit=1 ?
      else
            irot=0
```

```
      endif

      do 1 n=1,numel
c           write(*,*)'Matriz de rigidez EL',n
c           se for uma fase de enchimento temos de verificar
c           se o elemento deve ter comportamento de descarga
            if(ndes.ne.0)then
c                 ha ndes elementos em descarga
                  do 222 ii=1,ndes
                        if(iel(ii).eq.n)then
c                             este esta em descarga
                              idesc=1
                              write(17,*)'Elemento',n,' Desc.'
                              goto 223
                        endif
222               continue
                  idesc=0
            endif

223         if(n.ge.nel1.and.n.le.nel2)then
                  fmag=.00001
c                 a var. nomag foi estabelecida em fase
c                 e possibilita a consideracao de carregamento
c                 logo apos construcao
                  if(nomag.eq.1)fmag=1.
            else
                  fmag=1.
            endif

            if(iele(n,12).eq.1)then

c               le as tensoes da iteracao anterior e calcula tens. principais
                if(it .eq. 1) then
                      read (4)((tens(j,ik),j=1,3),ik=1,17)
                   else
                      read (7)((tens(j,ik),j=1,3),ik=1,17)
                end if

c               n de pontos de gauss
                ikk=npon(iele(n,13))

c               ciclo para cada ponto de gauss
c               inicializar matriz de constantes elasticas
                call inic3(dd,3,3,10)
                do 20 ik=1,ikk
c                     transfere as tensoes do ponto para um vector
                      call trasig(t,tens,ik)
c                     calcula os valores principais
c                     do vector transferido
                      if(iele(n,13).ne.3)then
                            call princi(t,sig)
                      else
```

```
c                       sig(2)=t(2)
c                       sig(1)=t(1)
c                       para os elementos de junta c
c                       consideramos a tensao media
                        sig(2)=tens(2,10)
                        sig(1)=tens(1,10)
                    endif
c                   determina caracteristicas elasticas
                    call elaw(sig,n,ik,cmat,iele,slev,et,xniu,irot)
c                   calcula e guarda matriz [D]
                    call matd(et,xniu,d,iele(n,13))
c                   write(*,*)' E=',et,' Niu=',xniu

c                   coloca na matriz 3d
                    do 23 m=1,3
                          do 23 m1=1,3
23                  dd(m,m1,ik)=d(m,m1)*fmag
20              continue
                write(8,rec=n)dd
c
c               calcula matriz de rigidez do elemento
                call matkk(dd,kk,iele,xy,n)
c
c               assemblagem da matriz de rigidez global
                call indexa(iele,n)
            else
c               truque para resolver os elementos que ainda nao estao
c               construidos
                do 22 ii=1,16
                      kk(ii,ii)=1
22              continue
            end if

            call assemb(k,kk,ne,nbd)

1       continue
c
c       modifica [KK] para entrar com condicoes de fronteira
        if(deslimp.eq.1.)then
              call modimp(apoio,f,k)
           else
              call modif(apoio,f,k,iele)
        end if
c
        return
        end
c.....
        subroutine trasig(t,tens,k)
        real t(3),tens(3,17)
        do 1 i=1,3
              t(i)=tens(i,k)
1       continue
```

```
      return
      end
c.....
      function npon(itipo)
      include 'gau1.for'
      if(itipo.eq.1)then
            npon=ng8*ng8
      elseif(itipo.eq.2)then
            npon=ng6
      elseif(itipo.eq.3)then
            npon=ngj
      elseif(itipo.eq.4)then
            npon=4
      else
            write(*,*)'Erro no n. de PG'
            stop
      endif
      return
      end
c.....
      subroutine indexa(iele,n)
c
c     subrotine que indexa os deslocamentos no referencial global
c
      include 'datnum.for'
      common/des/idg(16)
      integer iele(numelt,13)
      do 1 i=1,8
            if(iele(n,i).ne.0)then
                  idg(2*i-1)=iele(n,i)*2-1
                  idg(2*i)=iele(n,i)*2
            else
                  idg(2*i-1)=0
                  idg(2*i)=0
            endif
1     continue
      return
      end
c.....
c
c     determinacao do centro de gravidade
c     dos elementos em coordenadas globais
c     dadas as coordenadas locais
c
c.....
      subroutine cordg(iele,xy,ie,x1,x2)

c     ie = n. do elemento
c     x1,x2 = coordenadas globais

      include'datnum.for'
      include'varios.for'
```

```
      real xy(numnpt,2),xe(8,2),y(2),x(2),l(3)

      integer iele(numelt,13)

      call inicv(x,2)

c     transfere coordenadas do elemento para
c     vector de trabalho
      call tranxe(iele,xy,xe,ie)

c     decisao para cada elemento de acordo com o tipo
      goto (1,2,3) iele(ie,13)

c     el. de 8 nos
c     coordenadas locais
1     y(1)=0
      y(2)=0

      do 10 m=1,2
            do 11 i=1,8
                  call fufor8(i,ym,y,a)
                  x(m)=x(m)+a*xe(i,m)
11          continue
10    continue
      goto 4

c     el. triangular de 6 nos
c     coordenadas locais de area
2     l(1)=1./3.
      l(2)=1./3.
      l(3)=1./3.
      do 20 m=1,2
            do 21 i=1,6
                  call fufor6(i,l,a)
                  x(m)=x(m)+a*xe(i,m)
21          continue
20    continue
      goto 4

c     elemento de junta
c     nao se calcula pois nao e' necessario
3     return

4     x1=x(1)
      x2=x(2)

      return
      end

c-------------------------------------------------

      subroutine inic(mat,a,b)
```

```
c
c       subrotina que inicializa uma matriz de dimensoes (a x b) a zero
c
        integer a,b
        real mat(a,b)
c
        do 1 i=1,a
           do 2 j=1,b
                mat(i,j)=0.
2       continue
1       continue
        return
        end
        subroutine inic3(mat,a,b,c)
c
c       subrotina que inicializa uma matriz de dimensoes (a x b x c) a zero
c
        integer a,b,c
        real mat(a,b,c)
c
        do 1 i=1,a
           do 2 j=1,b
                do 3 k=1,c
                        mat(i,j,k)=0.
3               continue
2          continue
1       continue
        return
        end
        subroutine inicv(vect,n)
c
c       subrotina que inicializa um vector de dimensao n  a zero
c
        real vect(n)
c
        do 1 i=1,n
                vect(i)=0.
1       continue
        return
        end

        subroutine data(ic)

c               VAX
        character*9 dma
        character*8 hms
c       vaxstation
c       character*24 dmahms
c       call fdate(dmahms)
        call date(dma)
        call time(hms)
        write(ic,10)'Calculo realizado em ',dma,hms
```

```
c       write(ic,10)'Calculo realizado em ',dmahms
        return
10      format(//1x,a,a,'  - ',a//)

        end

c*************************************************************
c       ficheiro c/ subs para calculo das derivadas das
c       funcoes de forma
c*************************************************************
c       elemento rectangular de 8 nos
        subroutine der8(ym,y,i,k,idel,a)
        real ym(8,2),y(2)
        integer idel(2,2)

        if(i.lt.5)then
                a=.25*ym(i,k)*(1+ym(i,3-k)*y(3-k))*(2*ym(i,k)*
     1          y(k)+ym(i,3-k)*y(3-k))
        else
                a=.5*(1-y(3-k)*y(3-k))*ym(i,k)**3-(y(k)+
     1          ym(i,3-k)*y(1)*y(2))*ym(i,3-k)*ym(i,3-k)
        endif
        return
        end
c.....
c       elemento triangular de 6 nos
        subroutine der6(l,i,k,idel,a)
        real l(3)
        integer idel(2,2)

        if(i.lt.4)then
                if(i.eq.k)then
                        a=4*l(i)-1
                else
                        a=0
                endif
        else
                m=2*i-k+1
                n=m-3*(m/3)
                if(n.eq.0)n=3
                it=i-1
                if(it.gt.3)it=it-3
                if(k.eq.it)then
                        a=0
                else
                        a=4*l(n)
                endif
        endif
        return
        end
c*************************************************************
        subroutine ciclo(i,j,k)
```

```
      j=i+1
      if(j.gt.3)j=j-3
      k=i+2
      if(k.gt.3)k=k-3
      return
      end
c*******************************************************
c     Ficheiro c/ subs p/ gerar as matrizes de
c     rigidez para os varios elementos
c     Ver 2.0
c*******************************************************
c     Elemento rectangular de 8 nos
      subroutine fem8(xg,xe,d,idel,h,ym,ke,se)

      include 'datnum.for'
      include 'gau1.for'

      integer idel(2,2)
      real jj(2,2),ke(16,16),yy(2,2),y(2),aa(8,2),se(8,2,3,17)
     1 ,xe(8,2),d(3,3,10),ym(8,2),xg(5,2)

      call inic(ke,16,16)

      do 1 ig=1,ng8
            y(1)=xg(ig,1)

            do 1 jg=1,ng8
                  y(2)=xg(jg,1)
                  hg=xg(ig,2)*xg(jg,2)

                  call tranj8(ym,y,xe,idel,jj,yy,det,aa)

                  nl=ng8*(ig-1)+jg

                  do 11 i=1,8
                    do 11 m=1,2
                      do 11 k=1,3

11                          se(i,m,k,nl)=d(k,m,nl)*aa(i,m)+
     1                      d(k,3,nl)*aa(i,3-m)

                  do 1 i=1,8
                    ii=2*(i-1)
                    do 1 j=i,8
                      ij=2*(j-1)
                      do 1 m=1,2
                        do 1 n=1,2
                          ir=ii+m
                          it=ij+n

1       ke(ir,it)=ke(ir,it)+(aa(i,m)*se(j,n,m,nl)+
     1  aa(i,3-m)*se(j,n,3,nl))*hg*det
```

```
      do 2 i=1,16
            do 2 j=1,i-1
2     ke(i,j)=ke(j,i)

      do 3 i=1,16
            do 3 j=1,16
3     ke(i,j)=ke(i,j)*h

      return
      end
c.....
c     elemento triangular de 6 nos
      subroutine fem6(xgt,xe,d,idel,h,ke,se)

      include 'datnum.for'
      include 'gau1.for'

      integer idel(2,2)
      real jj(2,3),l(3),ke(16,16),yy(3,2),y(2),aa(6,2),xgt(7,4)
     1,se(8,2,3,17),xe(8,2),d(3,3,10)

      call inic(ke,16,16)

      do 1 ig=1,ng6
            do 2 k=1,3
2     l(k)=xgt(ig,k)

      hg=xgt(ig,4)

      call tranj6(l,xe,idel,jj,yy,det,aa)

      do 11 i=1,6
        do 11 m=1,2
          do 11 k=1,3
          if(d(1,1,ig).eq.0)then
                write(*,*)' ERRO EM FEM'
            endif
11    se(i,m,k,ig)=d(k,m,ig)*aa(i,m)+d(k,3,ig)*aa(i,3-m)

      do 1 i=1,6
        ii=2*(i-1)
        do 1 j=1,6
          ij=2*(j-1)
          do 1 m=1,2
            ir=ii+m
            do 1 n=1,2
              it=ij+n
1     ke(ir,it)=ke(ir,it)+(aa(i,m)*se(j,n,m,ig)+
     1 aa(i,3-m)*se(j,n,3,ig))*hg*det

      do 3 i=1,12
        do 3 j=1,i-1
```

```
3       ke(i,j)=ke(j,i)

        do 4 i=1,12
          do 4 j=1,12
4       ke(i,j)=ke(i,j)*h

        return
        end
c.....
c       elemento de junta c/ 6 nos
        subroutine femj(d,xe,xg,h,ke,se)

        include 'datnum.for'
        include 'gau1.for'

        real jj(2,2),ke(16,16),xe(8,2),xg(5,2),d(3,3,10),se(8,2,3,17)

        call inic(ke,16,16)

        do 2 ii=1,ngj
          y=xg(ii,1)
          pe=xg(ii,2)
          call tranjj(xe,y,jj,det)
          do 2 i=1,6
            call fuforj(i,y,a)
            ir=2*(i-1)
            do 2 j=i,6
              call fuforj(j,y,b)
              it=(-1)**(i/4+j/4+2)
              is=2*(j-1)
              do 2 m=1,2
                if(i.ne.1)goto 51
                  do 7 nl=1,2
                    c=0
                    do 8 ip=1,2
8                   c=c+d(m,ip,ii)*jj(ip,nl)
7                 se(j,nl,m,ii)=((-1)**(j/4+1))*c*b/det
51              do 2 n=1,2
                    c=0
                    do 3 ip=1,2
                      e=0
                      do 4 iq=1,2
4                     e=e+d(ip,iq,ii)*jj(iq,n)
3                   c=c+e*jj(ip,m)
2       ke(ir+m,is+n)=ke(ir+m,is+n)+it*a*c*b*pe/det

        do 5 i=1,12
              do 5 j=1,i-1
5       ke(i,j)=ke(j,i)

        do 6 i=1,12
              do 6 j=1,12
```

```
6       ke(i,j)=ke(i,j)*h

        return
        end
c***********************************************
c
c       calculo das forcas nodais equivalentes
c       a cargas aplicadas na sup do elemento
c
c***********************************************
        subroutine fosup6(k,pe,xgt,h,xe,idel,fe)

        include 'datnum.for'
        include 'gau1.for'

        integer idel(2,2)

c       pe () - forcas aplicadas ao lado carregado
c       k - lado carregado
        real l(3),jj(2,3),yy(3,2),aa(6,2),pe(3,2),
     1  xgt(5,2),xe(8,2),fe(8,2)

        call ciclo(k,n,it)
        do 1 m=1,2
          do 1 i=1,6
            fe(i,m)=0
            do 2 ig=1,ng8
                y=xgt(ig,1)
                xl=.5*(y+1)
                hg=.5*xgt(ig,2)
                l(k)=0
                l(n)=xl
                l(it)=1-l(n)
                call tranj6(l,xe,idel,jj,yy,det,aa)
                c=0
                do 3 j=1,3
                  call fuforj(j,y,a)
                  c=c+a*pe(j,m)
3               continue
                call fufor6(i,l,a)
              fe(i,m)=fe(i,m)+a*c*det*sqrt(yy(k,1)*yy(k,1)+
     1          yy(k,2)*yy(k,2))*hg
2           continue
            fe(i,m)=fe(i,m)*h
1       continue
        return
        end
c.....

        subroutine fosup8(k,pe,xg,h,xe,idel,ym,fe)

        include 'datnum.for'
```

```
      include 'gau1.for'

      integer idel(2,2)

      real jj(2,2),y(2),yy(2,2),aa(8,2),pe(3,2),xg(5,2),
     1 xe(8,2),ym(8,2),fe(8,2)

      it=(-1)**(k/2+1)
      m=k/2
      i=k-2*m
      ip=i+1
      y(ip)=it
      do 1 m=1,2
        do 1 i=1,8
          fe(i,m)=0
          do 2 ig=1,ng8
c               ?
              y(3-ip)=xg(ig,1)
              hg=xg(ig,2)
              call tranj8(ym,y,xe,idel,jj,yy,det,aa)
              c=0
              do 3 j=1,3
                call fuforj(j,y(3-ip),a)
3             c=c+a*pe(j,m)
              call fufor8(i,ym,y,a)
              b=(jj(1,3-ip)*jj(1,3-ip)+jj(2,3-ip)*jj(2,3-ip))
2         fe(i,m)=fe(i,m)+a*c*sqrt(b)*hg
1     fe(i,m)=fe(i,m)*h
      return
      end
c*************************************************
c
c     calculo de forcas equivalentes a accoes volumicas
c
c*************************************************
      subroutine fovol6(q,xe,idel,h,xgt,fe)

      include 'datnum.for'
      include 'gau1.for'

      integer idel(2,2)

      real l(3),jj(2,3),yy(3,2),aa(6,2),xe(8,2),xgt(7,4),fe(8,2),q(2)

      do 1 m=1,2
        do 1 i=1,6
          fe(i,m)=0
          do 2 ig=1,ng6
              do 3 k=1,3
3             l(k)=xgt(ig,k)
            hg=xgt(ig,4)
c           write(*,*)l(1),l(2),l(3),hg
```

```
                call fufor6(i,l,a)
                call tranj6(l,xe,idel,jj,yy,det,aa)
2           fe(i,m)=fe(i,m)+a*hg*det
1       fe(i,m)=fe(i,m)*q(m)*h
        return
        end
c.....
        subroutine fovol8(q,xe,idel,h,xg,ym,fe)

        include 'datnum.for'
        include 'gau1.for'

        integer idel(2,2)
        real jj(2,2),y(2),yy(2,2),aa(8,2),q(2),xe(8,2),xg(5,2),
    1   ym(8,2),fe(8,2)

        do 1 m=1,2
          do 1 i=1,8
            fe(i,m)=0
            do 2 ig=1,ng8
                do 2 jg=1,ng8
                  y(1)=xg(ig,1)
c                        ?
                  y(2)=xg(jg,1)
                  hg=xg(ig,2)*xg(jg,2)
                  call fufor8(i,ym,y,a)
                  call tranj8(ym,y,xe,idel,jj,yy,det,aa)
2           fe(i,m)=fe(i,m)+a*hg*det
1       fe(i,m)=fe(i,m)*q(m)*h
        return
        end
c.....
c**************************************************
c
c       Forcas equivalentes a estados de tensao
c       definidos pela matriz ss(3,9) nos pontos
c       de integracao numerica
c               ISO 8 nos
c**************************************************
        subroutine fote8(fe,idel,xg,ss,ym,xe,h)

        include 'datnum.for'
        include 'gau1.for'
        integer idel (2,2)
        real jj(2,2),fe(8,2),y(2),xg(5,2),ym(8,2),ss(3,9),xe(8,2)
     1,yy(2,2),aa(8,2)

        do 1 m=1,2
          do 1 i=1,8
            fe(i,m)=0
            do 2 ig=1,ng8
              y(1)=xg(ig,1)
```

```
                do 2 jg=1,ng8
                  nl=ng8*(ig-1)+jg
                  y(2)=xg(jg,1)
                  hg=xg(ig,2)*xg(jg,2)
                  call tranj8(ym,y,xe,idel,jj,yy,det,aa)
2       fe(i,m)=fe(i,m)-(aa(i,m)*ss(m,nl)+aa(i,3-m)*ss(3,nl))*det*hg
1       fe(i,m)=fe(i,m)*h

        return
        end
c*****************************************************
c
c       Forcas equivalentes a estados de tensao
c       definidos pela matriz ss(3,9) nos pontos
c       de integracao numerica
c               ISO 6 nos
c*****************************************************
        subroutine fote6(fe,idel,xg,ss,xe,h)

        include 'datnum.for'
        include 'gau1.for'

        integer idel (2,2)
        real jj(2,3),fe(8,2),xg(7,4),ss(3,9),xe(8,2)
     1,yy(3,2),aa(6,2),l(3)

        do 1 m=1,2
          do 1 i=1,6
            fe(i,m)=0
            do 2 ii=1,ng6
              do 3 k=1,3
3               l(k)=xg(ii,k)
                  hg=xg(ii,4)
                call tranj6(l,xe,idel,jj,yy,det,aa)
2       fe(i,m)=fe(i,m)-(aa(i,m)*ss(m,ii)+aa(i,3-m)*ss(3,ii))*det*hg
1       fe(i,m)=fe(i,m)*h

        return
        end
c*****************************************************
c
c       Forcas equivalentes a estados de tensao
c       definidos pela matriz ss(3,9) nos pontos
c       de integracao numerica
c               JUNTA
c*****************************************************
        subroutine fotej(ss,xe,xg,h,fe)
        include 'datnum.for'
        include 'gau1.for'

        real jj(2,2),fe(8,2),xg(5,2),de(8,2),ss(3,9)
```

```
      do 1 m=1,2
        do 1 i=1,6
          fe(i,m)=0
            do 2 ii=1,ngj
                  y=xg(ii,1)
                  hg=xg(ii,2)
                  call fuforj(i,y,a)
                  call tranjj(xe,y,jj,det)
                  it=(-1)*n*(i/4+1)
2           fe(i,m)=fe(i,m)+a*ss(m,ii)*hg*it*h*det
1     continue

      return
      end
c.....
      subroutine somaf(forca,f,iele,n)
c
c     Soma as forcas nos 8 nos de um elemento ao vector das forcas
c
      include 'datnum.for'
c
      real f(ne),forca(8)
      integer iele(numelt,13)
      do 1 j=1,4
            no=iele(n,j)
c
c           Forcas horizontais
      write(*,*)'ERRo SUB NAO PREPARADA'
            f(2*no-1)=f(2*no-1)+forca(2*j-1)
c
c           Forcas verticais
1           f(2*no)=f(2*no)+forca(2*j)
10    continue
      return
      end


c************************************
c
c     Ficheiro c/ subs que geram
c     as funcoes de forma
c                  88/12/21
c************************************

c.....
c     Elemento rectangular isoparametrico de 8 nos
      subroutine fufor8(i,ym,y,a)
      real ym(8,2),y(2)

      if(i.lt.5)then
c           pontos nodais 1 a 4 (vertices)
            a=.25*(1+ym(i,1)*y(1))*(1+ym(i,2)*y(2))*
```

```
     1          (ym(i,1)*y(1)+ym(i,2)*y(2)-1)
          else
c                 pontos nodais 5 a 8 (lados)
                  a=0
                  do 1 m=1,2
                        a=a+.5*(1-y(3-m)*y(3-m))*(1+ym(i,m)*y(m))*
     1                  ym(i,m)*ym(i,m)
1                 continue
          endif
          return
          end
c.....
c         Elemento de junta de 6 nos
          subroutine fuforj(j,y,a)

          if(j.eq.3.or.j.eq.6)then
c                 lados
                  a=1-y*y
          else
c                 cantos
                  m=mod(j,3)
                  a=.5*y*(y+(-1)**m)
          endif
          return
          end
c.....
c         Elemento triangular de 6 nos
          subroutine fufor6(i,l,a)
          real l(3)

          if(i.lt.4)then
                  a=(2*l(i)-1)*l(i)
          else
                  k=i-3
                  j=k+1
                  if(j.gt.3)j=j-3
                  a=4*l(k)*l(j)
          endif
          return
          end
c*****************************************************
c         Ficheiro com as subs Gauss
c*****************************************************
c         Elemento triangular de 6 pontos
          subroutine gauss6(xg)

          include 'datnum.for'
          include 'gau1.for'

          real xg(7,4)

          call inic(xg,7,4)
```

```
      goto (50,52,53,52,52,52,55),ng6

c.....
c     NG6=1
50          xg(1,1)=1./3.
            xg(1,2)=xg(1,1)
            xg(1,3)=xg(1,1)
            xg(1,4)=.5
      goto 100

c.....
c     NG6=?
52    write(*,*)' Erro em Gauss6, NG6=',ng6
      stop

c.....
c     NG6=3
53          do 1 i=1,3
                  xg(i,i)=.5
                  xg(i,4)=1./6.
1           continue
            xg(1,2)=.5
            xg(2,3)=.5
            xg(3,1)=.5
      goto 100

c.....
c     NG6=7
55    do 2 i=1,3
            xg(1,i)=1./3.
            xg(i+1,4)=8./120.
            xg(4+i,4)=3./120.
            xg(4+i,i)=1.
2     continue
      xg(1,4)=27./120.
      do 3 i=1,2
            xg(2,i)=.5
            xg(3,i+1)=.5
3     continue
      xg(4,1)=.5
      xg(4,3)=.5

100   return
      end

c.....
c     Elemento rectangular de 8 pontos
      subroutine gauss8(ng,h)

      include 'datnum.for'
      include 'gau1.for'
```

```
      real aga(5,5),pga(5,5),h(5,2)


      data ((aga(i,j),j=1,5),i=1,5)
     1/0,.57735027,.77459667,.86113631,.90617984
     1,0,-.57735027,0,.33998104,.53846931
     1,0,0,-.77459667,-.86113631,0
     1,0,0,0,-.33998104,-.90617984
     1,0,0,0,0,-.53846931/

      data ((pga(i,j),j=1,5),i=1,5)
     1/0.,1.,.55555556,.34785484,.23692689
     1,0.,1.,.88888889,.65214515,.47862867
     1,0.,0.,.55555556,.34785484,.56888889
     1,0.,0.,0.,.65214515,.23692689
     1,0.,0.,0.,0.,.47862867/

      if(ng.eq.0.or.ng.eq.1)goto 100

      do 1 i=1,ng
            h(i,1)=aga(i,ng)
            h(i,2)=pga(i,ng)
1     continue

100   return
      end
c-----------------------------------------------------------------
      subroutine matd(E,xniu,D,itip)
c
c     calculo da matriz d para cada elemento
c     sigma= [D] .eps
c
      real D(3,3),K,G

c     Zero matriz D
      call inic(D,3,3)
      if(itip.eq.1.or.itip.eq.2.or.itip.eq.4)then
c           Exp. 4.9, 4.10
            K=E/(3.*(1-2.*xniu))
            G=E/(2.*(1+xniu))
            D(1,1)=K+4./3.*G
            D(1,2)=K-2./3.*G
            D(2,1)=D(1,2)
            D(2,2)=D(1,1)
            D(3,3)=G
      else
c           elemento de junta
c           Kt
            D(1,1)=xniu
c           Kn
            D(2,2)=E
      endif
```

```
      return
      end
c.....
c
c     Geracao das matrizes YM
c
c.....
      subroutine matym(ym)
      real ym(8,2)

      do 1 i=1,4
            ym(i,1)=(-1)**(i/2+1)
            ym(i,2)=(-1)**(i/3+1)
1     continue
      ym(5,1)=0
      ym(7,1)=0
      ym(6,2)=0
      ym(8,2)=0
      ym(6,1)=1
      ym(7,2)=1
      ym(8,1)=-1
      ym(5,2)=-1
      return
      end
c.....
      subroutine delta(del)
      integer del(2,2)
      del(1,1)=1
      del(2,2)=1
      del(1,2)=0
      del(2,1)=0
      return
      end
c.....
      subroutine  matyv(yv)
      real yv(6,2)
      do 1 i=1,6
            ir=i/4+1
            yv(i,2)=(-1)**ir
1     continue
      do 2 i=1,2
            yv(3*i,1)=0
            yv(i,1)=(-1)**i
            yv(3+i,1)=(-1)**i
2     continue

      return
      end
c-------------------------------------------------------
c     s/jmax0/max0/w   VAX->MICRO
      subroutine cargas(f,xy,iele)
c
```

```
c       calcula forcas
c
        include 'datnum.for'
        include 'gau1.for'
        include 'gau2.for'
        include 'varios.for'
c
        integer iele(numelt,13)

        real f(ne),xy(numnpt,2),pe(3,2),xe(8,2),fe(8,2)
c
c       le numero
        write(17,3)'Aplicacao de cargas'
        read(1,*)ncd,ncc
c
c       cargas dist.
        if(ncd.ne.0)write(17,4)ncd,'elemento','Lado','Cargas  X,Y'
        do 10 ik=1,ncd
                read(1,*)ie,lado,((pe(i,j),j=1,2),i=1,3)
                write(17,5)ie,lado,((pe(i,j),j=1,2),i=1,3)
c               transferir coordenadas do elemento
                call tranxe(iele,xy,xe,ie)

c               goto de acordo com o tipo de elemento
                goto(31,32,33,35) iele(ie,13)

31              nno=8
                call fosup8(lado,pe,xg8,1.,xe,idel,ym,fe)
                goto 34

32              nno=6
                call fosup6(lado,pe,xg8,1.,xe,idel,fe)
                goto 34

33              write(*,*)' Forcas aplicadas em el de junta ?'
                stop

35              call fosupp(lado,pe,xg8,1.,xe,yv,xgj,fe)


34              call distr(f,fe,iele,ie,nno)

10      continue
c
c
        if(ncc.ne.0)write(17,7)ncc,'No','Forca X','Forca Y'
        do 20 i=1,ncc
                read(1,*)no,forcx,forcy
                write(17,6)no,forcx,forcy
                f(2*no-1)=f(2*no-1)+forcx
                f(2*no)=f(2*no)+forcy
20      continue
```

```
      return
3     format(1x,a19)
4     format(//,1x,'Cargas  distribuidas: ',i2,//,5x,a8,3x,a4,12x,a6)
5     format(1x,i3,' - ',i3,6(5x,f10.2))
6     format(1x,i3,2(5x,f7.2))
7     format(//,1x,'Cargas  concentradas: ',i2,/,/,2x,a2,5x,a7,5x,a7,/)

      end
c*********************************************************************
      subroutine distr(f,fe,iele,n,nno)
c
c     distribui forcas pelos nos
c
      include 'datnum.for'
c
      real f(ne),fe(8,2)
      integer iele(numelt,13)
c     write(17,*)'ELEMENTO ',n
      do 20 i=1,nno
c           write(17,*)fe(i,1),fe(i,2)
            do 21 j=1,2
c               write(*,*)i,j,2*iele(n,i)-2+j
                f(2*iele(n,i)-2+j)=f(2*iele(n,i)-2+j)+fe(i,j)
21          continue
20    continue
      return
      end
c*********************************************************************
      subroutine bandw(iele,nbdt,net)
c
c     Subrotina que calcula a largura de banda da matriz de rigidez
c     assemblada
c
      include 'datnum.for'

      integer iele(numelt,13),ic8(28,2),ic6(15,2)

c     combinacoes possiveis para elemento de 8 nos
      data ((ic8(i,j),j=1,2),i=1,28)/1,2,2,3,3,4,4,1,5,6,6,7,7,8
   1,8,5,1,3,2,4,5,7,8,6,2,6,6,3,3,7,7,4,4,8,8,1,1,5,5,2,5,3,5,4
   1,6,4,6,1,7,1,7,2,8,2,8,3/

c     combinacoes para elemento de 6 nos ou juntas
      data ((ic6(i,j),j=1,2),i=1,15)/1,2,2,3,3,1,4,5,5,6,6,1,2,5
   1,5,3,3,6,6,1,1,4,4,2,4,3,2,6,5,1/

      maxdif=0
      do 100 i=1,numelt
            if(iele(i,13).eq.1)then
                  do 101 j=1,28
                        maxdif=max0(maxdif,
   1                    iabs(iele(i,ic8(j,1))-iele(i,ic8(j,2))))
```

```
101                     continue
                else
                        do 102 j=1,15
                                maxdif=max0(maxdif,
     1                          iabs(iele(i,ic6(j,1))-iele(i,ic6(j,2))))
102                     continue
                endif
100     continue

        nbdt=(maxdif+1)*2
        net=2*numnpt
        return
        end
c*******************************************************************************
        subroutine bandwd(iele)
c
c       Subrotina que calcula a largura de banda da matriz de rigidez
c                               assemblada (dinamica)
c
        include 'datnum.for'
        integer iele(numelt,13),ic8(28,2),ic6(15,2)

c       combinacoes possiveis para elemento de 8 nos
        data ((ic8(i,j),j=1,2),i=1,28)/1,2,2,3,3,4,4,1,5,6,6,7,7,8
     1,8,5,1,3,2,4,5,7,8,6,2,6,6,3,3,7,7,4,4,8,8,1,1,5,5,2,5,3,5,4
     1,6,4,6,1,7,1,7,2,8,2,8,3/

c       combinacoes para elemento de 6 nos
        data ((ic6(i,j),j=1,2),i=1,15)/1,2,2,3,3,1,4,5,5,6,6,1,2,5
     1,5,3,3,6,6,1,1,4,4,2,4,3,2,6,5,1/

        maxdif=0

        write(*,*)'Bandwd NUMEL=',numel
        write(*,*)'NUMEL,NUMELT',numel,numelt
        do 100 i=1,numel
                if(iele(i,13).eq.1)then
                        do 101 j=1,28
                                maxdif=max0(maxdif,
     1                          iabs(iele(i,ic8(j,1))-iele(i,ic8(j,2))))
101                     continue
                else
                        do 102 j=1,15
                                maxdif=max0(maxdif,
     1                          iabs(iele(i,ic6(j,1))-iele(i,ic6(j,2))))
102                     continue
                endif
100     continue

        nbd=(maxdif+1)*2

c       determinar o ponto de numeracao mais elevada ate ao momento
```

```
      ne=0

      do 103 i=1,8
            ne=max0(ne,iele(numel,i))
103   continue

      ne=2*ne

      write(*,*)'NE,NBD=',ne,nbd

      return
      end
c*********************************************************************
      subroutine atrib(itip,cmat)
c
c     subrotina que atribui a um "common" as caracteristicas de um elemento
c
      include 'datnum.for'
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,
     1 s3t,xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      character*625 array
      real cmat(nummat,25)
      write(array,'(25f25.10)') (cmat(itip,i),i=1,25)
      read(array,'(25f25.10)') gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,
     1 beta,s3t,xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      return
      end

c*********************************************************************
      subroutine anuld(anul,nda,desl)
c
c     Anula os deslocamentos da camada superior (quando esta e construida)
c
      include 'datnum.for'
      real desl(ne)
      integer anul(10,2)

      do 2 i=1,nda
            do 1 n=anul(i,1),anul(i,2)
                  neq1=2*n-1
                  neq2=neq1+1
                  desl(neq1)=0.
                  desl(neq2)=0.
1           continue
2     continue

      return
      end

c---------------------------------------------------------------
      subroutine princi(vector,pr)
c
```

```
c       Esta subrotina calcula valores principais do vector
c
        real vector(3),pr(3)
c
        centro=(vector(1)+vector(2))/2.
        b=(vector(2)-vector(1))/2.
        raio=sqrt(vector(3)**2.+b**2.)

c       write(*,*)'b=',b,' v1,v2,v3 ',(vector(i),i=1,3)

        pr(1)=centro+raio
        pr(2)=centro-raio
        if(b.eq.0)then
                alfa=0
        else
                alfa=atan2(vector(3),b)/2.
        endif

c       Passa para graus
        pr(3)=alfa*180./3.1415927

c
        return
        end
c-----------------------------------------------------------------
        subroutine elaw(sig,n,k,cmat,iele,slev,E,xniu,irot)
c
c       endereca para a subrotina caracteristica do modelo usado
c
        include 'datnum.for'
c
        real sig(3),cmat(nummat,25),slev(numelt,2,17)
        integer iele(numelt,13)
c
c       no1,no2,no3,no4,no5,no6,no7,no8,mat,mod,   ,   , t.el
c        1  2  3  4  5  6  7  8  9 10          13

c       elemento de junta
        if(iele(n,13).eq.3)then
                call elawj(sig(2),sig(1),n,k,cmat,iele,slev,e,xniu,irot)
                return
        endif

        if(iele(n,10).eq.1)then
c               modelo hiperbolico
                call elawhy(sig,n,k,cmat,iele,slev,E,xniu,irot)
          else
c               modelo EC-k0
                call elawec(sig,n,k,cmat,iele,slev,E,xniu,irot)
        end if
c
        return
```

```
      end
c.....
c
c     nova versao de elawj em 12/02/90
c     tem em atencao que o comportamento de descarga pode
c     ter duas variantes, assim quando slev diminui
c     ha que ter em atencao a variacao de sn. Caso este diminua
c     entao o modulo de descarga e muito elevado
c     caso a descarga se de por um aumento de sn considera-se o modulo
c     inicial  Kti, note-se que caso sn seja .le.0 entao
c     os modulos correspondem a tracao
c     neste caso slev(n,1,k)=slev
c                slev(n,2,k)=sigman(i-1)
c
c.....
      subroutine elawj(s,tt,n,k,cmat,iele,slev,kn,kt,irot)
      include 'datnum.for'
      common/pat/patm
      common/atribu/gama,xkt,xktur,xn,xkn,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      integer iele(numelt,13)
      real slev(numelt,2,17),cmat(nummat,25),kn,kt
c
c     atribui valores ao common atribu
      call atrib(iele(n,9),cmat)
c
c     so interessa o valor numerico
      t=abs(tt)
      h2o=patm/10.33
c     write(17,*)'elemento ' ,n
      if(fi0.eq.90.)then
c           elastico linear
            kt=xkt*h2o
            kn=xkn*h2o
            return
      endif

      ir=0
      if(s.le.0.1*patm)then
c           tracao
            if(s.le.0.)then
c                 write(17,*)'rotura por tracao'
                  ir=1
c                 limita valor minimo da tensao
            endif
            s=patm/10.
      end if
c
      sigman=s
      xkti=xkt*h2o*( s /patm)**xn
      fi=fi0*3.1415927/180.
      taur=c+s*tan(fi)
```

```
      kn=xkn*h2o
c     Carga ou descarga ou rotura do material ?
      sl=t/taur
      tauu=taur/rf
c     primeira iteracao sempre em carga
c     if(irot.eq.0)goto 2000
c     rotura
      if(sl.gt..95.or.ir.eq.1)then
c           esta em rotura, ou por tracao ou por tau excessivo
c           write(17,*)'Elemento em rotura'
            kt=(1.-.95*rf)**2.*xkti
            if(ir.eq.1)then
c                 esta em rotura por tracao
                  kn=kt
                  sigman=0
            endif
            sl=1
            goto 1000
      endif

c     carga
c     write(17,*)'Elemento em carga'
2000  kt=(1.-t/tauu)**2*xkti
      kn=h2o*xkn

c     a descarga e verificada para a it anterior
c     if(sl.lt.0.95*slev(n,1,k).and.irot.ne.0)then
      if(sl.lt.0.95*slev(n,1,k))then
            if(sigman.gt.slev(n,2,k))then
c                 elemento em recarga
                  kt=xkti
c                 assumido o modulo inicial
            else
c                 elemento em descarga
                  kt=1e8
            endif
      end if

c     guarda stress-level anterior

1000  if(irot.ne.0)then
            slev(n,2,k)=sigman
            slev(n,1,k)=sl
      endif
      return
      end

c.....
      subroutine elawhy(sig,n,k,cmat,iele,slev,Et,xniu,irot)
c
c     subrotina para calculo de E e niu (modelo hiperbolico)
c
```

```
      include 'datnum.for'

      common/moddesc/idesc
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      integer iele(numelt,13)
      real slev(numelt,2,17),cmat(nummat,25),sig(3)
c
c     atribui valores ao common atribu
      call atrib(iele(n,9),cmat)
c
c     write(17,*)'elemento ' ,n
      if(fi0.eq.90.)then
c           elastico linear
            Et=xk*patm
            xniu=g
c           write(17,*)' Elastico linear'
            return
      endif
c     trata-se de um elemento em descarga por accao da
c     impulsao ?
      if(idesc.eq.1)then
            sl=0
            xniu=g
            if(xniu.gt.0.49)xniu=0.49
            slev(n,1,k)=0.
            if(sig(2).lt..3*patm)sig(2)=.3*patm
c           assume coef. de poisson igual ao inicial
c           e salta para o calculo do modulo de descarga
            Et=xkur*patm*(sig(2)/patm)**xn
            goto 1000
      endif

      ir=0
      if(sig(2).le.0.3*patm)then
c           traccao interpretada como rotura
            if(sig(2).le.0.)then
c                 write(17,*)'rotura por tracao'
                  ir=1
            endif
c           limita o valor da tensao
            sig(2)=patm*.3
      end if

c     Exp. 4.21
      Ei=xk*patm*(sig(2)/patm)**xn
c
c     Exp. 4.30
      fi=fi0-dfi*alog10(sig(2)/patm)
      fi=fi*3.1415927/180.
c
```

```
c       Exp. 4.28
        divi=(1.-sin(fi))
        s1s3r=(2.*c*cos(fi)+2.*sig(2)*sin(fi))/divi
c
c       Exp. 4.37
        xniui=g-f*alog10(sig(2)/patm)
c
        xpart1=1.-rf*(sig(1)-sig(2))/s1s3r
        xniu=xniui/(1.-d*(sig(1)-sig(2))/Ei/xpart1)**2
        if(xniu.gt..49)xniu=.49
        if(xniu.le.0.05)xniu=.05
c
c       Carga ou descarga ou rotura do material ?
        sl=(sig(1)-sig(2))/s1s3r

        if(irot.ne.0)slev(n,1,k)=sl
c       Evitar stress-level baixos
        if(slev(n,1,k).le..01)slev(n,1,k)=.01

c       Exp. 4.27
        s1s3u=s1s3r/rf

c       esta ou esteve em rotura

        if(sl.gt..95.or.slev(n,2,k).eq.1)then
c               write(17,*)'Elemento em rotura',irot
                Et=(1.-.95*rf)**2.*Ei
                xniu=.490
                if(irot.ne.0)then
c                       write(17,*)'Memoriza rotura '
                        slev(n,1,k)=1.
                        slev(n,2,k)=1.
                endif
                goto 1000
        endif

c       carga
c       Exp. 4.33
c       write(17,*)'Elemento em carga'
        Et=(1.-(sig(1)-sig(2))/s1s3u)**2.*Ei

        if(ir.eq.1)then
                xniu=.495
                Et=(1.-.95*rf)**2.*Ei
                write(*,*)'Elemento ',n,' em tracao E=',Et
c               se entra num processo de rotura
c               o modulo tem de ser concordante
                if(irot.eq.1)then
                        slev(n,1,k)=-1.
                        slev(n,2,k)=1.
                        goto 1000
                endif
```

```
      endif
      if(sl.lt.0.95*slev(n,2,k))then
c           write(17,*)'Elemento em descarga',n
            Et=xkur*patm*(sig(2)/patm)**xn
      end if

c     guarda stress-level maximo

1000  if(irot.ne.0)then
            if(abs(slev(n,1,k)).gt.slev(n,2,k))then
                  slev(n,2,k)=abs(slev(n,1,k))
            endif
      endif
      return
      end
c.....
      subroutine elawec(sig,n,k,cmat,iele,slev,Et,xniu,irot)
c
c     subrotina para calculo de E e niu (modelo EC-k0)
c
      include 'datnum.for'
      common/pat/patm
      common/moddesc/idesc
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      integer iele(numelt,13)
      real slev(numelt,2,17),cmat(nummat,25),sig(3)
c
c     atribui valores ao common atribu
      call atrib(iele(n,9),cmat)
      slev(n,1,k)=sig(1)
c
c     Carga ou descarga ?
c
      if(slev(n,1,k).ge.0.95*slev(n,2,k))then
c           carga
c           calcula modulo de elasticidade confinado
c           Exp. 4.21
            Ec=ae*be*patm*(sig(1)/patm/ae)**((be-1.)/be)
c
c           Calcula k0
            xk0t=ak0+2.*bk0*sig(1)/patm
            slev(n,2,k)=sig(1)
        else
c           descarga
            Ec=ce*patm
            xk0t=ck0+2.*dk0*sig(1)/patm
      end if
c     calculo de niu em funcao de k0
      xniu=xk0t/(1.+xk0t)
      if(xniu.ge.0.5)xniu=0.499
c     calculo do modulo de elasticidade nao confinado
```

```
c       K+4/3G=Ec
        Et=3*Ec*(1+xniu)*(1-2*xniu)/((1+xniu)+2*(1-2*xniu))
        return
        end
c-------------------------------------------------------------
        subroutine resol(xy,apoio,cmat,desl,deslt,dstr,f,iele,k,
     1  slev,stress,it,niter,anul,nda,numesc,nesc)
c
c       determinacao de desl., tensoes, caract. elasticas, etc.
c
        include 'datnum.for'
        include 'what.for'

        common/mat/mattip
        common/novel/nel1,nel2
        common/PREEXI/npre
c
        logical ja
        character*5 mattip(20)
        real cmat(nummat,25),f(ne),slev(numelt,2,17),stress(numelt,3,17)
        real k(ne,nbd),ext(3),tens(3),desl(ne),deslt(2*numnpt)
        real dstr(numelt,3,17),sig(3),xy(numnpt,2),ss(17,3)
        integer iele(numelt,13),apoio(nap,3),anul(10,2)
c
        if(numesc.eq.nesc)then
                ja=.true.
        else
                ja=.false.
        endif
        rewind 4
        rewind 7
c       4 -tensoes; 8 -[D]
c       7 - tensoes intermedias
c
c       formacao da matriz de rigidez  {f} = [K] . {d}
        call kkglob(xy,apoio,cmat,iele,f,slev,k,it,fmag)

c       resolucao do sistema

        call solve(k,f,desl)

c       Leitura de tensoes e deslocamentos da fase anterior
        rewind 4
        rewind 7
        do 1 n=1,numelt
1       read(4)((stress(n,i,ik),i=1,3),ik=1,17)

        if(it.eq.niter)read(4)(deslt(i),i=1,ne)
c
c       determinacao do extensoes,tensoes, caract. elasticas
c
        rewind 4
```

```
      do 10 n=1,numel
c           numero de pontos onde se calcula a tensao
            ikk=npon(iele(n,13))
c
c           calculo das tensoes incrementais
            call tensao(iele,ss,desl,n)
c
c           soma tensoes (se e' el. novo nao soma)
            if((n.ge.nel1.and.n.le.nel2).and.numesc.eq.1
     1                              .and.n.gt.npre)then

c             e' elemento construido de novo (nao soma tensao)
              if(it.eq.niter)then
c               write(17,*)'elemento novo ultima iteracao'
c               ultima iteracao tensao=tensao calculada
                do 11 i=1,3
                      stress(n,i,10)=ss(10,i)
                      do 11 ik=1,ikk
11                    stress(n,i,ik)=ss(ik,i)
              else
c               write(17,*)'elemento novo 1 iteracao'
c               1. iteracao, tensao intermedia
                do 14 i=1,3
                      stress(n,i,10)=.5*ss(10,i)
                      do 14 ik=1,ikk
14                    stress(n,i,ik)=0.5*ss(ik,i)
              endif
            else
c             elemento ja existente
              if(it.ne.niter)then
c               1. iteracao, calcula tensao intermedia
                do 12 i=1,3
c                     ponto central
                      stress(n,i,10)=stress(n,i,10)+
     1                .5*(ss(10,i)-dstr(n,i,10)/nesc)
                      do 12 ik=1,ikk
12                    stress(n,i,ik)=stress(n,i,ik)+
     1                0.5*(ss(ik,i)-dstr(n,i,ik)/nesc)

              else
c               ultima iteracao, calcula tensao
                do 15 i=1,3
                      stress(n,i,10)=
     1                stress(n,i,10)+ss(10,i)-dstr(n,i,10)/nesc
                      do 15 ik=1,ikk
15                    stress(n,i,ik)=
     1                stress(n,i,ik)+ss(ik,i)-dstr(n,i,ik)/nesc

              endif
            end if
c
c           escreve resultados
```

```
c                so escreve para o ultimo escalao
                 if(.not.ja)goto 10
                 if((it.eq.niter.and.iele(n,12).eq.1).or.tint.eq.1)
     1          call escrte(ext,n,stress,ikk,iele,xy)
10      continue
c
c
c       Soma dos deslocamentos incrementais e escrita de tensoes
c
c       anula deslocamentos nos elementos novos
        if(nda.ne.0)call anuld(anul,nda,desl)
c
        if(it.ne.niter)then
          do 2 n=1,numelt
2         write(7)((stress(n,j,ik),j=1,3),ik=1,17)
        else
          if(ja)write(17,13)'Caracteristicas elasticas',
     1     '-----------------------'
           do 3 n=1,numelt
            if(n.eq.1)then
              ihead=iele(n,13)
            else
              if(icab(iele(n-1,13)).eq.icab(iele(n,13)))then
                ihead=0
              else
                ihead=iele(n,13)
              endif
            endif
            if(n.le.numel)then
                if(ihead.eq.1.or.ihead.eq.2)then
                        if(ja)write(17,9)'Ele','M. Elast','Poiss'
     1                  ,' SLEV','SLmax',' S3/S1','Mat','Ele'
                elseif(ihead.eq.3.or.ihead.eq.4)then
                        if(ja)write(17,9)'Ele',' KT ',' KN '
     1                  ,' SLEV','SigN','Mat','Ele'
                else
                        ihead=iele(n,13)
            endif
            endif
            write(4)((stress(n,j,ik),j=1,3),ik=1,17)
            if(n.le.numel)then
              if(iqt.eq.2.or.iqt.eq.3)then
                ikk=npon(iele(n,13))
                 do 20 ik=1,ikk
                  do 7 i=1,3
7                 tens(i)=stress(n,i,ik)
                  if(ihead.ne.3)then
                    call princi(tens,sig)
                  else
c                   elemento de junta, a tensao e
c                   tau e sigma
                    sig(1)=tens(1)
```

```
                sig(2)=tens(2)
              endif
              call elaw(sig,n,ik,cmat,iele,slev,E,xniu,1)
              if(ihead.eq.1.or.ihead.eq.2.or.ihead.eq.4)then
                if(ja)write(17,8)n,E,xniu,(slev(n,i,ik),i=1,2),
     1          sig(2)/sig(1),mattip(iele(n,9)),n
              else
                if(ja)write(17,18)n,xniu,E,(slev(n,i,ik),i=1,2),
     1          mattip(iele(n,9)),n
              endif
20          continue
            endif
            if(iqt.eq.1.or.iqt.eq.3)then
              do 71 i=1,3
71            tens(i)=stress(n,i,10)
              if(ihead.ne.3)then
                call princi(tens,sig)
              else
c               elemento de junta, a tensao e
c               tau e sigma
                sig(1)=tens(1)
                sig(2)=tens(2)
              endif
              call elaw(sig,n,10,cmat,iele,slev,E,xniu,1)
              ik=10
              if(ihead.eq.1.or.ihead.eq.2.or.ihead.eq.4)then
                if(ja)write(17,8)n,E,xniu,(slev(n,i,ik),i=1,2),
     1          sig(2)/sig(1),mattip(iele(n,9)),n
            else
                if(ja)write(17,18)n,xniu,E,(slev(n,i,ik),i=1,2),
     1          mattip(iele(n,9)),n
            endif
          endif
        end if

3       continue
c             adiciona e guarda deslocamentos incrementais
              do 21 i=1,ne
21            deslt(i)=desl(i)+deslt(i)
              write(4)(deslt(i),i=1,2*numnpt)
        end if
c
        return
c
9       format(/,1x,a3,6x,a9,7x,a5,2(7x,a6),7x,a6,2(7x,a3))

8       format(1x,i3,6x,f9.1,7x,f5.3,2(7x,f6.3),7x,f6.3,6x,a5,6x,i3)
18      format(1x,i3,4x,f10.1,6x,f11.0,4x,f5.3,4x,f6.0,6x,5x,a5,6x,i3)
13      format(///1x,a,/1x,a)
        end

c.....
```

```
      subroutine solve(a,b,c)
      include'datnum.for'
      integer i,j,k,neq,nbd,ax,bx
      real a(ne,nbd),b(ne),c(ne),g(3000)
      real z
      write(*,*)'SISTEMA'
      do 100 j=2,ne
c           write(*,*)'FASE 1 ',j,ne
            ax=2
            if(j.gt.nbd)ax=ax+j-nbd
            do 200 i=ax,j
                  zz=0
                  do 300 k=ax-1,i-1
                        zz=zz+a(k,i-k+1)*a(k,j-k+1)/a(k,1)
300               continue
                  a(i,j-i+1)=a(i,j-i+1)-zz
200         continue
100   continue

      g(1)=b(1)
      do 400 i=2,ne
c           write(*,*)'FASE 2 ',i,ne
            ax=1
            if(i.gt.nbd)ax=ax+i-nbd
            zz=0
            do 500 j=ax,i-1
                  zz=zz+a(j,i-j+1)*g(j)/a(j,1)
500         continue
            g(i)=b(i)-zz
400   continue

      c(ne)=g(ne)/a(ne,1)
      do 600 j=2,ne
c           write(*,*)'FASE 3 ',j,ne

            bx=ne-j+1
            ax=ne
            if(j.gt.nbd)ax=ne-j+nbd
            zz=0
            do 700 i=bx+1,ax
                  zz=zz+a(bx,i-bx+1)*c(i)
700         continue
            c(bx)=(g(bx)-zz)/a(bx,1)
600   continue
      return
      end

c*******************************************************
c
c     calculo das tensoes nos elementos
c
c     ver 3.0
```

```
c*******************************************************
      subroutine tensao(iele,ss,d,n)

      include 'datnum.for'
      include 'gau1.for'

      real s(3),d(ne),se(8,2,3,17),ss(17,3)
      integer iele(numelt,13)

c     leitura da matriz 'se' tal que Sigma=se x d
c     write(*,*)'Leitura de SE para o el ',n
      read(10,rec=n)((((se(i,j,k,m),i=1,8),j=1,2),k=1,3),m=1,17)

      call inic(ss,17,3)

      goto (1,2,3,4) iele(n,13)

c     isop rectangular de 8 nos
1     do 10 ig=1,ng8
        do 10 jg=1,ng8
          m=ng8*(ig-1)+jg
          do 10 k=1,3
            s(k)=0
            do 11 i=1,8
              do 11 j=1,2
                s(k)=s(k)-se(i,j,k,m)*d(2*(iele(n,i)-1)+j)
11          continue
            ss(m,k)=ss(m,k)+s(k)
10    continue
      goto 100

c     isop triangular de 6 nos
2     do 20 ig=1,ng6
        do 20 k=1,3
          s(k)=0
          do 21 i=1,6
            do 21 j=1,2
              s(k)=s(k)-se(i,j,k,ig)*d(2*(iele(n,i)-1)+j)
21        continue
          ss(ig,k)=ss(ig,k)+s(k)
20    continue
      goto 100

c     elementos de junta de 6 nos
3     do 30 ig=1,ngj
        do 30 k=1,2
          s(k)=0
          do 31 i=1,6
            do 31 j=1,2
              s(k)=s(k)-se(i,j,k,ig)*d(2*(iele(n,i)-1)+j)
31        continue
          ss(ig,k)=ss(ig,k)+s(k)
```

```
30      continue
        goto 100

c       junta c/ espessura
4       ngp=2
        do 40 ig=1,ngp
          do 40 jg=1,ngp
            m=ngp*(ig-1)+jg
            do 40 k=1,3
               s(k)=0
               do 41 i=1,6
                  do 41 j=1,2
                     s(k)=s(k)-se(i,j,k,m)*d(2*(iele(n,i)-1)+j)
41             continue
               ss(m,k)=ss(m,k)+s(k)
40      continue

c       tensao no ponto central calculada com base nas medias
c       dos pontos de integracao de acordo com sugestao
c       dada pelo Dr. Naylor em 89/04/14

100     npts=npon(iele(n,13))

        call AVG(ss,npts)
        return
        end
c-------------------------------------------------
        subroutine AVG(ss,n)
        real ss(17,3),s(3),pr(3),d(3)
        logical erro

        niter=0
c       calcula a tensao media
        do 10 i=1,3
               su=0.
               do 20 j=1,n
                     su=su+ss(j,i)
20             continue
               ss(10,i)=su/float(n)
10      continue
        return
        end
c*************************************************
c       matrizes de transformacao de coordenadas
c*************************************************
c       elemento rectangular de 8 nos
        subroutine tranj8(ym,y,xe,idel,jj,yy,det,aa)

        integer idel(2,2)
        real jj(2,2),ym(8,2),y(2),xe(8,2),yy(2,2),aa(8,2),a(8,2)

        do 1 j=1,2
```

```
            do 1 i=1,8
                call der8(ym,y,i,j,idel,a(i,j))
1       continue

        do 2 m=1,2
            do 2 j=1,2
                jj(m,j)=0
                do 2 i=1,8
2                   jj(m,j)=jj(m,j)+a(i,j)*xe(i,m)

        det=jj(1,1)*jj(2,2)-jj(1,2)*jj(2,1)

        do 3 m=1,2
            yy(m,m)=jj(3-m,3-m)/det
            yy(m,3-m)=-jj(3-m,m)/det
c           yy(m,3-m)=-jj(m,3-m)/det
3       continue

        do 4 i=1,8
            do 4 n=1,2
                aa(i,n)=0
                do 4 k=1,2
4                   aa(i,n)=aa(i,n)+a(i,k)*yy(n,k)

        return
        end
c.....
c       elemento de junta com 6 nos
        subroutine tranjj(xe,y,jj,det)
        real jj(2,2),a(3),xe(8,2)

        a(1)=y-.5
        a(2)=y+.5
        a(3)=-2*y

        do 2 m=1,2
            jj(1,m)=0
            do 2 i=1,3
2               jj(1,m)=jj(1,m)+a(i)*xe(i,m)

        jj(2,1)=-jj(1,2)
        jj(2,2)=jj(1,1)
        b=jj(1,1)*jj(1,1)+jj(1,2)*jj(1,2)
        det=sqrt(b)
        return
        end
c.....
c       elemento triangular de 6 nos
        subroutine tranj6(l,xe,idel,jj,yy,det,aa)

        integer idel(2,2)
        real l(3),jj(2,3),xe(8,2),yy(3,2),aa(6,2),a(6,3)
```

```
      do 1 j=1,3
            do 1 i=1,6
                  call der6(l,i,j,idel,a(i,j))
1     continue

      do 2 m=1,2
            do 2 j=1,3
                  jj(m,j)=0
                  do 2 i=1,6
2                       jj(m,j)=jj(m,j)+a(i,j)*xe(i,m)
c     do 2 m=1,2
c           do 2 j=1,3
c                 call ciclo(j,k,n)
c                 jj(m,j)=(4*l(j)-1)*xe(j,m)+4*(l(k)*xe(j+3,m)
c  1              +l(n)*xe(m+3,m))
c2    continue

      det=0

      do 3 i=1,3
            call ciclo(i,j,k)
3     det=det+jj(1,j)*jj(2,k)-jj(1,k)*jj(2,j)

      do 4 i=1,3
            call ciclo(i,j,k)
            do 4 ip=1,2
4           yy(i,ip)=((-1)**(3-ip))*(jj(3-ip,j)-jj(3-ip,k))/det

      do 5 i=1,6
            do 5 n=1,2
                  aa(i,n)=0
                  do 5 k=1,3
5                       aa(i,n)=aa(i,n)+a(i,k)*yy(k,n)

      return
      end
c-------------------------------------------------------------
      subroutine valini(cmat,iele,slev,xy,deslt,stress)
c
c     subrotina que calcula os valores iniciais nos elementos
c     (tensoes e le desl. e ext. de elementos (pre-existentes)
c
      include 'datnum.for'
      common/mat/mattip
      common/PREEXI/npre
c     este common destina-se a passar o n maximo de elementos pre-
c     -existentes para no calculo das tensoes isso ser considerado
      real cmat(nummat,25),xy(numnpt,2),tensao(3),pr(3)
      real deslt(2*numnpt),stress(numelt,3,17),slev(numelt,2,17)
      integer iele(numelt,13)
      character*5 mattip(20),tel(4)
      character*3 mod(3),cod(3)
```

```
c
      data mod/'Hyp','Ec.','    '/,cod/'  - ','I+C',' I '/
      data tel/'ISOP8','ISOP6','J  e=0','Je<>0'/
c
c
c     npre- no. do elemento pre-existente ou da fundacao
c     icod=1 ==> pre-existente; icod/= 1 ==>fundacao
      npre=0
      call inic3(stress,numelt,3,17)
c
c
      write(17,*)
      write(17,*)' Dados referentes aos elementos e valores iniciais'
      read(1,*)npre,icod
      do 1 n=1,numelt
         call inicv(tensao,3)
c        if(n.gt.npre)read(1,*)npre,icod
c        if(n.lt.npre)then
         if(n.gt.npre)then
c           este elemento nao e pre-existente nem da fundacao
            call tenini(cmat,iele,xy,tensao,n)
c
c           para garantir que vamos ter carga em todos
c           os pontos de gauss dividimos a tensao por 8
            do 9 i=1,3
            tensao(i)=tensao(i)/10.
9           continue
         else
c           este elemento e' pre-existente ou da fundacao
            iele(n,12)=1
            if(icod.eq.1)then
c              este elemento 'e preexistente e' fornecida uma tensao
               read(1,*)(stress(n,i,1),i=1,3)
c              nos restantes pontos assume-se igual
               do 61 j=2,17
                  do 61 k=1,3
61             stress(n,k,j)=stress(n,k,1)

               do 2 i=1,3
                  tensao(i)=stress(n,i,1)
2              continue
            else
c              este elemento e' da fundacao
c              call tenfun(cmat,iele,xy,n)
            end if
         end if
        write(4)((tensao(i),i=1,3),k=1,17)
c        calcula tensoes principais
         if(iele(n,13).ne.3)call princi(tensao,pr)
c        calcula modulos e coef. Poisson

c        calcula modulos iniciais e inicializa slev a zero
```

```
      kk=npon(iele(n,13))
      do 45 k=1,kk
         call elaw(pr,n,k,cmat,iele,slev,E,xniu,0)
         slev(n,1,k)=0.
         slev(n,2,k)=0.
45    continue
      slev(n,1,10)=0.
      slev(n,2,10)=0.
c
      if(n.gt.1)then
            itip=iele(n-1,13)
      else
            itip=iele(n,13)
            if(itip.eq.1.or.itip.eq.2)then
               write(17,3)'No.',' no1 no2 no3 no4 no5 no6 no7 no8'
     1        ,'MATER','REO','MOL','TIPEL','M.ELASTIC','POISS'
     1        ,'SIGM-X','SIGM-Y','TAU-XY'
            else
               write(17,4)'No.',' no1 no2 no3 no4 no5 no6 no7 no8'
     1        ,'MATER','REO','MOL','TIPEL','RIGIDEZ  T','RIGIDEZ N'
     1        ,'  TAU','SIGM-N'
            endif
      endif

c     Escreve dados referentes aos elementos
      if(iele(n,13).ne.3)then
         if(itip.eq.3)then
         write(17,3)'No.',' no1 no2 no3 no4 no5 no6 no7 no8'
     1      ,'MATER','REO','MOL','TIPEL','M.ELASTIC','POISS'
     1      ,'SIGM-X','SIGM-Y','TAU-XY'
         endif
         write(17,6)n,(iele(n,i),i=1,8),mattip(iele(n,9)),
     1      mod(iele(n,10)),cod(iele(n,11)),tel(iele(n,13)),E,xniu,
     2      (tensao(i),i=1,3)
      else
         if(itip.eq.2.or.itip.eq.1)then
         write(17,4)'No.',' no1 no2 no3 no4 no5 no6 no7 no8'
     1      ,'MATER','REO','MOL','TIPEL','RIGIDEZ  T','RIGIDEZ N'
     1      ,'  TAU','SIGM-N'
         endif
         write(17,7)n,(iele(n,i),i=1,8),mattip(iele(n,9)),
     1      mod(iele(n,10)),cod(iele(n,11)),tel(iele(n,13)),xniu,E,
     2      (tensao(i),i=1,2)
      endif

1     continue
      write(4)(deslt(i),i=1,2*numnpt)
c
      return
3     format(//1x,a3,a32,4x,a5,4x,2(a3,4x),a5,5x,a9,4x,a5,3x,3(a6,3x))
6     format(1x,i3,8(1x,i3),4x,a5,4x,2(a3,4x),a5,5x,f9.1,4x,f5.3,3x,
     1 3(f6.2,3x))
```

```
4       format(//1x,a3,a32,4x,a5,4x,2(a3,4x),a5,5x,a9,2x,a9,3x,2(a6,3x))
7       format(1x,i3,8(1x,i3),4x,a5,4x,2(a3,4x),a5,5x,f9.1,1x,f11.0,2x,
     1  2(f6.2,3x))
        end

c.....
        subroutine tenini(cmat,iele,xy,tensao,n)
c
c       Subrotina que calcula as tensoes iniciais em elementos
c               que vao ser construidos
c
        include 'datnum.for'

        common/pat/patm

        real cmat(nummat,25),xy(numnpt,2),tensao(3)
        integer iele(numelt,13)

        call inicv(tensao,3)

        if(iele(n,13).eq.3)then
c       elemento de junta=> tensao =patm
c               Sn, tau
                tensao(1)=.01*patm
                tensao(2)=patm
c               write(11,rec=n)-999,-999
                return
        endif
c       if(iele(n,13).eq.4)then
c       elemento de junta=> tensao =patm
c               Sn, tau
c               tensao(1)=.01*patm
c               tensao(2)=.1*patm
c               write(11,rec=n)-999,-999
c               return
c       endif
c
c       calcula centro de gravidade do elemento
c
        call cordg(iele,xy,n,xcg,ycg)
c       guarda coordenadas do centro de gravidade
c       write(11,rec=n)xcg,ycg
c
c       Calculo da tensao vertical
        if(iele(n,13).eq.1)then
c               el. de 8 nos
                h=xy(iele(n,7),2)-ycg
        elseif (iele(n,13).eq.2)then
c               el. triangular
                h=0
c               calcula a maxima diferenca de cotas
c               entre o CG e os pontos nodais
```

```
                do 1 i=1,6
                    h1=amax1(h,(xy(iele(n,i),2)-ycg))
                    if(h1.gt.h)then
c                       achou novo ponto
                        no=i
c                       guardou o indice do no
                        h=h1
                    endif
1               continue
c               divide por 2 ja que o elemento e triangular
                h=h/2.
        else
c               elemento de junta
                h=0
        endif

        tensao(2)=cmat(iele(n,9),1)*h

c       Calculo da tensao horizontal
        mat=iele(n,9)
        if(iele(n,10).eq.1)then
                pois=cmat(mat,6)
        else
                xxx=cmat(mat,17)+2.*cmat(mat,18)*tensao(1)/patm
                pois=xxx/(1.+xxx)
        endif
        if(pois.gt..49)pois=.49
        xk0=pois/(1.-pois)
        tensao(1)=xk0*tensao(2)

c       Calculo de Tau
c       Calculo da inclinacao

        if(iele(n,13).eq.2)then
c           elemento triangular
c           esta formula e valida para elementos em que o no 'no'
c           e o de cota mais elevada
                slope=(xy(iele(n,no),2)-ycg)/(xy(iele(n,no),1)-xcg)
                beta=1.570796327-atan(slope+.00001)
        elseif(iele(n,13).eq.1)then
c               elemento trapezoidal
                slope=(xy(iele(n,3),2)-xy(iele(n,4),2))/
    1           (xy(iele(n,3),1)-xy(iele(n,4),1))
                beta=atan(slope)
        else
c               junta
                beta=0
        endif

c       as tensoes arbitradas sao principais
c       sig=1
c       if(beta.gt.(3.14/2.))sig=-1
```

```
c       tensao(3)=0.5*tensao(2)*sin(beta)*sig

        return
        end
c-------------------------------------------------
        subroutine modif(apoio,f,kkglob,iele)
c
c       subrotina que modifica [K] e {F} para atender as condicoes de apoio
c
        include 'datnum.for'

        common/camada/nomax
        real kkglob(ne,nbd),f(ne)
        integer apoio(nap,3),iele(numelt,13)
c
        do 1 i=1,nap
                no=apoio(i,1)
c               so para os nos ja existentes
                if(no.le.nomax)then
c                       Tem desl. segundo xx impedido?
                        if(apoio(i,2).ne.0.)then
                                nequa=2*no-1
                                call altap(nequa,kkglob,f,ne,nbd)
                        endif
c                       Tem desl. segundo yy impedido ?
                        if(apoio(i,3).ne.0.)then
                                nequa=2*no
                                call altap(nequa,kkglob,f,ne,nbd)
                        endif
                endif
1       continue
        return
        end
c-------------------------------------------------
        subroutine altap(neq,k,f,ne,nbd)
        real k(ne,nbd),f(ne)
        k(neq,1)=1e15
        f(neq)=0.
        return
        end
c-------------------------------------------------
        subroutine modimp(apoio,f,k)
c
c       modifica matriz de rigidez global para deslocamentos impostos
c
        include 'datnum.for'
c
        real f(ne),k(ne,nbd)
        integer apoio(nap,3)
c       le numero de deslocamentos impostos
        read(1,*)ndi
        do 1 i=1,ndi
```

```
                read(1,*)napoio,no
                if(no.ne.apoio(napoio,1))stop'Erro nos apoios'
c               Le o valor dos deslocamentos (em metros)
                read(1,*)deslx,desly
c               Deslocamento segundo x, altera termo de kkglob correspondente
                nequa=2*no-1
                call altdes(deslx,f,k,nbd,ne,nequa)
c               Deslocamento segundo y, altera termo de kkglob correspondente
                nequa=2*no
                call altdes(desly,f,k,nbd,ne,nequa)
1       continue
        return
        end
c-------------------------------------------------------
        subroutine altdes(d,f,k,nbd,ne,neq)
        real k(ne,nbd),f(ne)
        k(neq,1)=1e15
        f(neq)=d*1e15
        return
        end
c-------------------------------------------------------
        subroutine colaps(cmat,stress,dstr,f,iele,xy,net)

        include'datnum.for'
        include 'gau1.for'
        include 'gau2.for'
        include 'varios.for'
        integer iele(numelt,13),e(200)
        real cmat(nummat,25),f(net),xy(numnpt,2),sig(3),s(3)
        real xe(8,2),pe(3,2),fe(8,2),q(2),dstr1(3),ss(3,9)
        real dstr(numelt,3,17),stress(numelt,3,17),sw(2)

        real ft(2)

        common/pat/patm
        read(1,*)nsub
c       leitura dos elementos submersos
        read(1,*)(e(i),i=1,nsub)
        write(17,*)'Elementos submersos'
        write(17,*)(e(i),i=1,nsub)
        write(17,*)
        write(*,*)'Elementos submersos'
        write(*,*)(e(i),i=1,nsub)
        write(*,*)
        do 10 i=1,nsub
        call inicv(ft,2)
                n=e(i)
c               testa se o elemento tem colapso
                if(iele(n,11).eq.2)then
                write(*,*)'Elemento ',n,' Colapso'
                call inicv(dstr1,3)
                ng=npon(iele(n,13))
```

```
c               elemento de junta
                if(iele(n,13).eq.3.or.iele(n,13).eq.4)return
c               correr pontos de gauss
                write(17,1001)'El','PG',' S1S ',' S3S ',
     1  ' Sxr ',' Syr ',' Txyr ',' Sw1 ',' Sw2 ',' Evol'
                do 20 ii=1,ng
c                       transfere a tensao dos pontos de gauss p/ vect de trab.
                        do 21 j=1,3
21                      sig(j)=stress(n,j,ii)
                        mat=iele(n,9)
c                       calcula tensao de relaxacao
                        if(iele(n,10).eq.1)then
                                call relax(sig,dstr1,mat,cmat,sw,ii,n,evol)
                        else
                                call relaxec(sig,dstr1,mat,cmat,sw,ii,n)
                        endif
                        call princi(sig,s)

                        if(sw(2).gt.s(2))then
                                write(17,*)'Erro na determinacao de dstr'
                                do 8888 nr=1,3
8888                            dstr1(nr)=0
                        endif

                        write(17,1002)n,ii,s(1),s(2)
     1          ,dstr1(1),dstr1(2),dstr1(3),sw(1),sw(2),evol
c                       guarda as tensoes calculadas
                        do 22 j=1,3
c                               guarda p/ calculo de forcas
                                ss(j,ii)=dstr1(j)
c                       guarda p/ subtrair
22                      dstr(n,j,ii)=dstr1(j)
20              continue

c               calculo da tensao media
                call inicv(dstr1,3)
                do 23 ii=1,3
                        do 24 j=1,ng
                                dstr1(ii)=dstr1(ii)+dstr(n,ii,j)
24                      continue
                        dstr(n,ii,10)=dstr1(ii)/float(ng)
23              continue

c               muda o material no fim de todos os PG
                iele(n,9)=iele(n,9)+1

c               transfere coordenadas dos pontos nodais
                call tranxe(iele,xy,xe,n)

                goto (1,2,3) iele(e(i),13)

1               call fote8(fe,idel,xg8,ss,ym,xe,1.)
```

```
                nno=8
                goto 4

2               call fote6(fe,idel,xg6,ss,xe,1.)
                nno=6
                goto 4

c               elemento de junta n. tem colapso
3               continue

4               call distr(f,fe,iele,n,nno)
                do 55 iii=1,8
                do 55 jjj=1,2
55              ft(jjj)=ft(jjj)+fe(iii,jjj)
c       write(17,*)'forcas totais no elemento ',n
c       write(17,*)ft(1),ft(2)
                endif

10      continue

c       aumento de peso proprio
        do 100 i=1,nsub
                n=e(i)
                q(1)=0
c               o aumento de pp e definido no material antigo
                q(2)=-cmat(iele(n,9)-1,16)
c               aplicacao do aumento de peso
                call tranxe(iele,xy,xe,n)

                goto (101,102,103,103) iele(n,13)

101             call fovol8(q,xe,idel,1.,xg8,ym,fe)
                nno=8
                goto 104

102             call fovol6(q,xe,idel,1.,xg6,fe)
                nno=6
                goto 104

c               elemento de junta n. tem implusao
103             write(17,*)'ATENCAO COLAPSO EM JUNTA, alterar'
                continue

104             call distr(f,fe,iele,n,nno)

100     continue
        return
1001    format(1x,2(1x,a3,1x),7(2x,a9),3x,a12)
1002    format(1x,2(1x,i3,1x),7(2x,f9.3),3x,e12.3)
        end
c-------------------------------------------------------
        subroutine relaxec(sigma,dstr,mat,cmat,sw,ii,n)
```

```
      include 'datnum.for'
      real sigma(3),sig(3),sw(2),cmat(nummat,25),pstrd(2),dstr(3)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa

      call princi(sigma,sig)
      xk0w=sig(3)/sig(1)
      call atrib(mat,cmat)
      e1=(sig(1)/ae/patm)**(1/be)
c     muda material
      call atrib(mat+1,cmat)
      sw(1)=ae*patm*e1**be
c     xk0w=ak0+bk0*(sw(1)/patm)
      sw(2)=xk0w*sw(1)

      pstrd(1)=sig(1)-sw(1)
      pstrd(2)=sig(2)-sw(2)
      alf=sig(3)
      a1=cos(alf)*cos(alf)
      a2=sin(alf)*sin(alf)
      a3=.5*sin(2*alf)
      dstr(2)=pstrd(1)*a1+pstrd(2)*a2
      dstr(1)=pstrd(1)*a2+pstrd(2)*a1
      dstr(3)=pstrd(1)*a3-pstrd(2)*a3

      return
      end
c-----------------------------------------------------
      subroutine relax(sigma,dstr,mat,cmat,sw,ii,n,evol)
      include 'datnum.for'
      real sigma(3),dstr(3),sig(3),pstrd(2),sw(2),cmat(nummat,25)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      call princi(sigma,sig)
      if(sig(2).lt.0.)then
            call inicv(dstr,3)
            return
      endif

      sig(3)=sig(3)/180.*3.1415927
      call atrib(mat,cmat)

c     calcula ea, er do material seco
      ead=ea(sig(1),sig(2))
      erd=er(sig(2),ead)
      evol=ev(ead,erd)

c     write(17,*)'Material seco'
c     write(17,*)'Ea=',ead
c     write(17,*)'Er=',erd
```

```
c       write(17,*)'Evol=',evol
c       muda material
        call atrib(mat+1,cmat)

c       calcula tensoes molhadas
        call clps3(sig,ead,erd,evd,sw)

        pstrd(1)=sig(1)-sw(1)
        pstrd(2)=sig(2)-sw(2)
        alf=sig(3)
        a1=cos(alf)*cos(alf)
        a2=sin(alf)*sin(alf)
        a3=.5*sin(2*alf)
        dstr(2)=pstrd(1)*a1+pstrd(2)*a2
        dstr(1)=pstrd(1)*a2+pstrd(2)*a1
        dstr(3)=pstrd(1)*a3-pstrd(2)*a3

        return
        end
c-------------------------------------------------
        subroutine clps3(sig,ead,erd,evd,sw)
        real sig(3),sw(2)
        common/pat/patm
        common/seco/evs
        common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1  xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa

c       ev seco
        evd=ev(ead,erd)
c       write(17,*)'Extensoes'
c       write(17,*)'Ea=',ead
c       write(17,*)'Er=',erd
c       write(17,*)'Evol=',evd
        evs=evd
        sig3w=sig(2)/1000.
        xinc=patm/500.
        fx=0
        flx=1
        nit=0

1       nit=nit+1
        if(flx.eq.0)then
                write(17,*)'Erro evitado em CLPS3'
                goto 2
        endif
        sig3w=sig3w-fx/flx
c       write(17,*)sig1w,sig3w,fx
c       write(*,*)sig1w,sig3w,fx
        if(sig3w.le.0)then
                nit=51
                goto 2
        endif
```

```
      call func(sig3w,sig1w,fx,ead)
      call func(sig3w+xinc,sig1w,fx1,ead)

c     derivada
      flx=(fx1-fx)/xinc
2     if(nit.gt.50)then
            sw(1)=sig(1)
            sw(2)=sig(2)
            write(17,*)' Erro em clps3'
            return
      endif

      if(abs(fx).lt.1e-7)then
c           convergiu
            sw(1)=sig1w
            sw(2)=sig3w
            return
      endif
      goto 1
      end
c-----------------------------------------------------
      subroutine func(sig3w,sig1w,delta,ead)
      common/seco/evs
      call ealeac(sig3w,eal,eac,ead,evc)
      sig1w=s1(sig3w,eal)
      erw=er(sig3w,eal)
      evl=eal+2*erw
      evw=evl+evc
      delta=evw-evs
c     write(17,*)'Material molhado'
c     write(17,*)'Eaw=',eal+1./3.*evc
c     write(17,*)'Eac=',eac
c     write(17,*)'Erw=',erw
c     write(17,*)'Evl=',evl
c     write(17,*)'Evw=',evw
c     write(17,*)'Delta=',delta
      return
      end
c-----------------------------------------------------
      function ea(s1,s3)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      call fip2(s1,s3,fi,p2)
      ea=(s1-s3)/xk/patm/s3pan(s3)/(1-rf*p2)**2
      end
c-----------------------------------------------------
      function s3pan(s3)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      s3pan=(s3/patm)**xn
```

```
      end
c-----------------------------------------------------------
      function er(s3,ea)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      xniu=g-f*alog10(s3/patm)
C     if(xniu.gt.0.499)xniu=0.499
      er=-xniu*ea/(1-d*ea)
      end
c-----------------------------------------------------------
      function ev(ea,er)
      ev=ea+2*er
      end
c-----------------------------------------------------------
      function s1(s3,eal)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      p1=xk*patm*s3pan(s3)*eal
      fi=(fi0-dfi*alog10(s3/patm))/57.28
      p2=rf*(1-sin(fi))/(2*c*cos(fi)+2*s3*sin(fi))
      s1=s3+p1/(1+p1*p2)
      return
      end
c-----------------------------------------------------------
      subroutine ealeac(s3,eal,eac,ead,evc)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      diff=s3-s3t
      evc=diff*beta/patm
      eac=evc/3
      eal=ead-eac
      return
      end
c-----------------------------------------------------------
      subroutine fip2(s1,s3,fi,p2)
      common/pat/patm
      common/atribu/gama,xk,xkur,xn,d,g,f,c,fi0,dfi,rf,s3l,beta,s3t,
     1 xk0,dgama,ak0,bk0,ck0,dk0,ae,be,ce,de,alfa
      fi=(fi0-dfi*alog10(s3/patm))/57.28
      p2=(s1-s3)*(1-sin(fi))/(2*c*cos(fi)+2*s3*sin(fi))
      return
      end
c-----------------------------------------------------------
      subroutine derp(i,j,y,yv,a)
      dimension yv(6,2),y(2)
      ir=i/4+1
      if(j.Ne.1)Goto 51
      if(i.Ne.3.And.I.Ne.6)Goto 61
      a=-y(1)*(1+(-1)**ir*y(2))
```

```
      goto 52
61    a=.25*Yv(i,1)*(1+(-1)**ir*y(2))*(1+2*y(1)*yv(i,1))
      goto 52
51    continue
      if(i.Ne.3.And.I.Ne.6)Goto 64
      a=.5*(-1)**Ir*(1-y(1)*y(1))
      goto 52
64    a=.25*Yv(i,1)*y(1)*(-1)**ir*(1+yv(i,1)*y(1))
52    continue
      return
      end
c-------------------------------------------------
      subroutine fuforp(i,y,yv,a)
      dimension y(2),yv(6,2)
      ir=i/4+1
      if(i.Ne.3.And.I.Ne.6)Goto 51
      a=.5*(1-Y(1)*y(1))*(1+(-1)**ir*y(2))
      goto 52
51    a=.25*Yv(i,1)*y(1)*(1+yv(i,1)*y(1))*(1+(-1)**ir*y(2))
52    return
      end
c-------------------------------------------------
      subroutine tranjp(y,yv,xe,jj,yy,det,aa)
      real jj(2,2)
      dimension a(6,2),y(2),yv(6,2),xg(5,2),yy(2,2),aa(6,2)
     1,xe(8,2)
      do 1 j=1,2
      do 1 i=1,6
1     call derp(i,j,y,yv,a(i,j))
      do 2 m=1,2
      do 2 j=1,2
      jj(m,j)=0
      do 2 i=1,6
2     jj(m,j)=jj(m,j)+a(i,j)*xe(i,m)
      det=jj(1,1)*jj(2,2)-jj(1,2)*jj(2,1)
      do 3 m=1,2
      yy(m,m)=jj(3-m,3-m)/det
3     yy(m,3-m)=-jj(3-m,m)/det
      do 4 i=1,6
      do 4 n=1,2
      aa(i,n)=0
      do 4 k=1,2
4     aa(i,n)=aa(i,n)+a(i,k)*yy(n,k)
      return
      end
c-------------------------------------------------
      subroutine fosupp ( k, pe, xg,h,xe,yv, xgj, fe )
      real fe(8,2),jj(2,2),y(2),yy(2,2),aa(8,2),pe(3,2),xg(5,2)
     1,xe(8,2),yv(6,2)
      ngp=2
      call inic(fe,8,2)
      it=(-1)**(k/2+1)
```

```
      m=k/2
      i=k-2*m
      ip=i+1
      y(ip)=it
      write(*,*)ip
      do 1 m=1,2
        do 1 i=1,6
          do 2 ig=1,ngp
            y(3-ip)=xg(ig,1)
            hg=xg(ig,2)
            call tranjp(y,yv,xe,jj,yy,det,aa)
            c=0
            if(ip.eq.1)then
              do 3 j=1,2
                a=.5*(1+(-1)**j*y(2))
                c=c+a*pe(j,m)
3             continue
            else
              do 4 j=1,3
                call fuforj(j,y(1),a)
                c=c+a*pe(j,m)
4             continue
            endif
            call fuforp(i,y,yv,a)
            b=jj(1,3-ip)**2+jj(2,3-ip)**2
2         fe(i,m)=fe(i,m)+a*c*sqrt(b)*hg
1     fe(i,m)=fe(i,m)*h
      return
      end

      subroutine fovolp(q,xe,h,xg,yv,fe)
      include 'gau1.for'
      real jj(2,3)
      dimension y(2),yy(3,2),aa(6,2),q(2),xe(8,2),xg(5,2),
     1yv(6,2),fe(8,2)
      ngp=2
      do 1 m=1,2
      do 1 i=1,6
      fe(i,m)=0
      do 2 ig=1,ngp
      do 2 jg=1,ngp
      y(1)=xg(ig,1)
      y(2)=xg(jg,1)
      hg=xg(ig,2)*xg(jg,2)
      nl= ngp*(ig-1)+jg
      call fuforp(i,y,yv,a)
      call tranjp(y,yv,xe,jj,yy,det,aa)
2     fe(i,m)=fe(i,m)+a*hg*det*h
1     fe(i,m)=fe(i,m)*q(m)
      return
      end
c-------------------------------------------------------
```

```
      subroutine femp(h,xg,xe,yv,d,ke,se)
      include'gau1.for'
      real jj(2,2),ke(16,16),y(2),yy(2,2),aa(6,2),xg(5,2)
     1,xe(8,2),yv(6,2),d(3,3,10),se(8,2,3,17)
      call inic(ke,16,16)
      ngp=2
      do 1 ig=1,ngp
       y(1)=xg(ig,1)
       do 1 jg=1,ngp
        y(2)=xg(jg,1)
        hg=xg(ig,2)*xg(jg,2)
        call tranjp(y,yv,xe,jj,yy,det,aa)
        nl=ngp*(ig-1)+jg
        do 11 i=1,6
        do 11 m=1,2
        do 11 k=1,3
11      se(i,m,k,nl)=d(k,m,nl)*aa(i,m)+d(k,3,nl)*aa(i,3-m)
        do 1 i=1,6
         ii=2*(i-1)
         do 1 j=1,6
          ij=2*(j-1)
          do 1 m=1,2
          do 1 n=1,2
           ir=ii+m
           it=ij+n
1     ke(ir,it)=ke(ir,it)+(aa(i,m)*se(j,n,m,nl)
     1+aa(i,3-m)*se(j,n,3,nl))*det*hg

      do 2 i=1,12
            do 2 j=1,i-1
2     ke(i,j)=ke(j,i)

      do 3 i=1,12
            do 3 j=1,12
3     ke(i,j)=ke(i,j)*h

      return
      end
c-------------------------------------------------------
      subroutine fotep(fe,del,xg,ss,yv,xe,h)
      include 'gau1.for'
      integer del(2,2)
      real jj(2,2),fe(8,2),y(2),xg(5,2),yv(8,2),ss(3,9),xe(8,2)
     1,yy(2,2),aa(6,2)
      call inic(fe,8,2)
      ngp=2
      do 1 m=1,2
       do 1 i=1,6
        do 2 ig=1,ngp
         y(1)=xg(ig,1)
         do 2 jg=1,ngp
          nl=ngp*(ig-1)+jg
```

```
            y(2)=xg(jg,1)
              hg=xg(ig,2)*xg(jg,2)
              call tranjp(y,yv,xe,jj,yy,det,aa)
2           fe(i,m)=fe(i,m)-(aa(i,m)*ss(m,nl)+aa(i,3-m)*ss(3,nl))*det*hg
1       fe(i,m)=fe(i,m)*h
        return
        end
```

DATNUM.for

```
      common/datnum/numelt,numnpt,nummat,nap,numel,nbd,ne,ndesimp
      common/vector/maxpg
```

GAU1.for

```
c     n. de pontos de integracao de gauss
      common/gau1/ng8,ng6,ngj
```

GAU2.for

```
c     resultados das sub's de gauss
      common /gau2/  xg8(5,2),xg6(7,4),xgj(5,2),xgp(5,2)
```

VARIOS.for

```
      common/varios/idel(2,2),ym(8,2),yv(6,2),xmove(100,3),nesc
```

WHAT.for

```
      common/what/iqt,tint
```

ANEXO III

Listagem do programa MEFSG

```
c*****************************************************************
c
c       Calculo de construcao e enchimento de Barragens de aterro
c
c       Metodo dos elementos finitos
c       sistema grafico V5.3
c
c       elementos finitos
c       rectangulares de 8 nos
c       triangulares de 6 nos
c       junta de 6 nos
c
c       João Marcelino
c
c*****************************************************************
        program MEFSGI
        character*(12)filein,auxf
        character*80 titulo
        dimension a(20000)

        include 'datnum.for'
        common/nome/filein
        common/gauss/ng8,ng6
        common/uau/  titulo
c
c       Ficheiro de dados
c
        write(*,*)' DVAX'
        write(*,*)
        write(*,*)' J. Marcelino'
        write(*,*)

        write(*,'(a)')' Nome do ficheiro de dados ? '
        read(*,'(a12)')filein
        open(unit=1,status='old',file=filein)
c
c       Leitura de dados gerais
c
        read(1,'(a)')titulo
        read(1,*)numelt,numnpt,nap,nfel,numcel,nummat,nfases
        read(1,*)ng8,ng6,ngj
        read(1,*)iqt,tint

c       ficheiro com os resultados
        write(auxf,'(a,a)')filein(1:index(filein,'.')),'plo'
        open(unit=9,status='unknown',file=auxf,form='unformatted')

c       coordenadas dos nos
        n1=1
c       apoios
```

```
      n2=n1+2*numnpt
c     elementos
      n3=n2+3*nap
c     tensoes
      n4=n3+13*numelt
c     deslocamentos
      n5=n4+2*numelt*18
c     maximo ate ao momento
      n6=n5+4*numnpt
c
c     Leitura de dados (apoios,materiais,elementos,nos)
      call leitur(a(n2),a(n3),a(n1))
c
c     execucao do programa
      call grafs(a(n1),a(n2),a(n3),a(n4),a(n5))

      end
c-------------------------------------------------
      subroutine inicv(vect,n)
c
c     subrotina que inicializa um vector de dimensao n  a zero
c
      real vect(n)
c
      do 1 i=1,n
            vect(i)=0.
1     continue
      return
      end
c.....
      subroutine leitur(apoio,iele,xy)
c
c     Le dados
c
      include 'datnum.for'
      common/atmos/patm
      real xy(numnpt,2)
      integer iele(numelt,13),apoio(nap,3)
c
c     le pressao atmosferica
      read(1,*)patm
c
c     salta caracteristicas dos materiais
      call skip(2*nummat)
c
c     Le coordenadas dos nos
      call lexy(xy)
c
c     determina limites do desenho
      call cordm(xy)
c
c     le condicoes de apoio
```

```
      call leap(apoio,xy,3)
c
c     le caracteristicas dos elementos
      call leele(iele,13)
c
      write(*,*)'Calcular posicao dos nos centrais (1/0) '
      read(*,*)ica
      if(ica.eq.1)then
            call posc(xy,iele)
      endif
      return
      end
c.....
      subroutine posc(xy,iele)
      include 'datnum.for'
      real xy(numnpt,2)
      integer iele(numelt,13),iq(4,3),it(3,3)
      data ((iq(i,j),j=1,3),i=1,4)/5,1,2,6,2,3,7,4,3,8,1,4/
      data ((it(i,j),j=1,3),i=1,3)/4,1,2,5,2,3,6,3,1/
      do 10 i=1,numelt
            if(iele(i,13).eq.1)then
c                 quadrangular
                  do 11 j=1,4
                        call med(xy,iele,iq(j,1),iq(j,2),iq(j,3),i,1)
                        call med(xy,iele,iq(j,1),iq(j,2),iq(j,3),i,2)
11                11 continue

            elseif(iele(i,13).eq.2)then
c                 triangular
                  do 12 j=1,3
                        call med(xy,iele,it(j,1),it(j,2),it(j,3),i,1)
                        call med(xy,iele,it(j,1),it(j,2),it(j,3),i,2)
12                12 continue
            elseif(iele(i,13).eq.3.or.iele(i,13).eq.4)then
c           elemento de junta
            else
                  write(*,*)'erro no tipo de elemento'
                  write(*,*)'elemento ',i
                  write(*,*)(iele(i,j),j=1,13)
            endif
10    continue
      return
      end
c-----
      subroutine med(xy,iele,i,j,k,n,l)
      include 'datnum.for'
      real xy(numnpt,2)
      integer iele(numelt,13)
      xy(iele(n,i),l)=(xy(iele(n,j),l)+xy(iele(n,k),l))/2.
      return
      end
c.....
```

```
c       calculo das coordenadas maximas e minimas
c
        subroutine cordm(xy)
        include 'datnum.for'
        real xy(numnpt,2)
        common/lim1/xm,xn,ym,yn
        xm=xy(1,1)
        xn=xm
        ym=xy(1,2)
        yn=ym
        do 10 i=2,numnpt
                xm=amax1(xm,xy(i,1))
                ym=amax1(ym,xy(i,2))
                xn=amin1(xn,xy(i,1))
                yn=amin1(yn,xy(i,2))
10      continue
        return
        end
c.....
c
        subroutine lexy(xy)
c
c
        include 'datnum.for'
        real xy(numnpt,2)
        read(1,*)no1,xy(no1,1),xy(no1,2)
        do 1 ijj=1,numnpt
                if(no1.eq.numnpt)return
                read(1,*)no2,xy(no2,1),xy(no2,2)
                if(no1+1.ne.no2)then
                  deltax=(xy(no2,1)-xy(no1,1))/float(no2-no1)
                  deltay=(xy(no2,2)-xy(no1,2))/float(no2-no1)
                  do 10 ikk=no1+1,no2-1
                        xy(ikk,1)=xy(ikk-1,1)+deltax
                        xy(ikk,2)=xy(ikk-1,2)+deltay
10                continue
                end if
        no1=no2
1       continue
        return
        end
c.....
        subroutine leap(apoio,xy,num)
c
c       subrotina para ler as condicoes de apoio e os deslocamentos
c   impostos
c
        include 'datnum.for'
        integer apoio(nap,num)
        real xy(numnpt,2)
c
c       apoio(i,1) - numero do n'o com apoio
```

```
c       apoio(i,2) = 1. - no' apoio(i,1) tem restricao segundo xx
c       apoio(i,3) = 1. - no' apoio(i,1) tem restricao segundo yy
c
        read(1,*)iap1,(apoio(iap1,j),j=1,3)
        do 1 ijj=1,nap
              if(iap1.ne.nap)then
                read(1,*)iap2,(apoio(iap2,j),j=1,3)
                if(iap1+1.ne.iap2)then
c
c                apoios intermedios
                 do 10 ikk=1,iap2-iap1-1
c                      passa para o n'o seguinte
                       apoio(iap1+ikk,1)=apoio(iap1,1)+float(ikk)
c                      mantem as mesmas condicoes de apoio e desl.
                       do 20 jj=2,num
                         apoio(iap1+ikk,jj)=apoio(iap1,jj)
20                     continue
10               continue
               end if
             end if
             iap1=iap2
1       continue
        return
        end

c.....................................................
        subroutine leele(iele,num)
c       'salto' dos nos para elementos 1=8nos 2=6 nos
        include 'datnum.for'
        integer igap(8,2),iele(numelt,num)
        data ((igap(i,j),i=1,8),j=1,2)/2,2,2,2,2,1,2,1,
     12,2,2,2,2,2,0,0/
c       iele(n,1-8)- nos do elemento n
c       iele(n,9)  - material do el. n
c       iele(n,10)  - modelo reologico do el. n
c       iele(n,11)   - codigo de molhagem "
c       iele(,13)    - tipo de elemento finito
c                        =1 isoparam rectangular de 8 nos
c                        =2 isoparam triangular de 6 nos
c                        =3 isoparam junta de 6 nos

        read(1,*)nel1,(iele(nel1,j),j=1,13)
        do 1 ijj=1,numelt
              if(nel1.eq.numelt)return
              read(1,*)nel2,(iele(nel2,j),j=1,13)
              if(nel1+1.ne.nel2)then

c               elementos intermedios
                do 10 ikk=1,nel2-nel1-1
                     if(iele(nel1,13).eq.1)then
                       k=1
                   else
```

```
                    k=2
                  endif
c                 atribuicao dos numerós dos nos
                  do 20 jj=1,8
                    iele(nel1+ikk,jj)=iele(nel1+ikk-1,jj)+igap(jj,k)
20                continue
c                 atribuicao de material, modelo, cod. molhagem
                  do 30 jj=9,num
                    iele(nel1+ikk,jj)=iele(nel1,jj)
30                continue
10              continue
              end if
      nel1=nel2
1     continue
      return
      end
c.....
      subroutine skip(n)
      character*80 lixo
c     'salta' n registos no ficheiro
      do 10 i=1,n
            read(1,'(a)')lixo
10    continue
      return
      end
c.....
c     localiza uma fase no ficheiro de dados para plotter
      subroutine locate(ifase,numel,maxno,erro)
      character*4 txt
      common/fase/if,nume
      logical erro
      rewind 9
c     ciclo
      erro=.false.
2     read(9,end=100)txt
      if(txt.eq.'FASE')then
            backspace 9
            goto 1
      else
            goto 2
      endif
1     read(9,end=100)txt,if,numel,maxno
      if(if.ne.ifase)goto 2
c     encontrou
      nume=numel
      return
100   write(*,*)'A fase pedida nao existe'
      erro=.true.
      return
      end
c.....
      subroutine grafs(xy,apoio,iele,stress,des)
```

```
      include 'datnum.for'
      character*12 nome
      logical yes,exist
      integer apoio(nap,3),iele(numelt,13)
      real xy(numnpt,2),stress(numelt,3,18),des(numnpt,4)
      common/auxe/exist
      ig=1

1     call newname(nome,ig)
c                    nome do ficheiro
      call setsize
c                    dimensoes desenho
      if(ig.eq.1)call aux
c                    ficheiro auxiliar ?
      call what(id)
c                    o que se pretende fazer
      call zoom
c                    eventual ampliacao
      call setscaleg
c                    determina escala geometria
c     ok desenha
      write(*,*)'Entra em proceed'
      call proceed(id,nome,exist,xy,iele,apoio,stress,des)
      ig=ig+1
      call prompt('Deseja continuar',yes)
      if (yes) goto 1
      return
      end
c.....
c     determina o nome a dar ao grafico
c     nomenumero.plt
      subroutine newname(name,cont)
      character*12 name,filein
      integer cont
      common/nome/filein
      ip=index(filein,'.')
      write(*,*)
      write(name,'(a,i2.2,a)')filein(1:ip-1),cont,'.plt'
      write(*,*)' Ficheiro =',name
      write(*,*)
      return
      end
c.....
c     pergunta ao utilizador qual a dimensao do desenho
c
      subroutine setsize
      integer option
      common/size/six,siy
c     isetf=formato papel
c     isetd=escala desenho
c     isett=escala tensao
      common/escalas/isetf,isetd,isett
```

```
      real a4x,a4y,a3x,a3y,mux,muy
      data a4x,a4y,a3x,a3y,mux,muy/21.,29.7,42.,29.7,16.,24.1/

1     write(*,3)'Dimensoes disponiveis'
      write(*,2)'1 - A4 ',a4x,a4y
      write(*,2)'2 - A3 ',a3x,a3y
      write(*,2)'3 - MU ',mux,muy
      write(*,2)'4 - A4R',a4y,a4x
      write(*,2)'5 - A3R',a3y,a3x
      write(*,2)'6 - MUR',muy,mux
      write(*,3)'7 - USER          - ? '
      if(isetf.ne.1)then
            read(*,*)option
            isetf=1
      endif
      goto (10,20,30,40,50,60,70),option
      goto 1
c                       erro de escolha
10    call set(a4x,a4y)
      goto 100
20    call set(a3x,a3y)
      goto 100
30    call set(mux,muy)
      goto 100
40    call set(a4y,a4x)
      goto 100
50    call set(a3y,a3x)
      goto 100
60    call set(muy,mux)
      goto 100
70    write(*,3)'Dimensao em x '
      read(*,*)dx
      write(*,3)'Dimensao em y '
      read(*,*)dy
      call set(dx,dy)
100   return
3     format(1x,a)
2     format(1x,a,3x,f6.3,',',f6.3)
      end
c.....
c     estabiliza as variaveis de dimensao do desenho
      subroutine set(x,y)
      common/size/six,siy
      six=x
      siy=y
      return
      end
c.....
c     escolha do desenho pretendido
      subroutine what(id)
      integer errsta
1     write(*,2)'Indique o desenho pretendido'
```

```
      write(*,2)'1 - Desenho da malha'
      write(*,2)'2 - Desenho de deslocamentos'
      write(*,2)'3 - Desenho de tensoes'
      write(*,2)'4 - Malha deformada'
      write(*,2)'5 - Isolinhas          ? '
      read(*,*)id
      goto (100,100,100,100,100),id
      goto 1
100   return
2     format(1x,a)
      end
c.....
c
c     verifica e abre ficheiro auxiliar
      subroutine aux
      character*12 auxf,filein
      logical exist
      common/auxe/exist
      common/nome/filein
      write(auxf,'(a,a)')filein(1:index(filein,'.')-1),'.aux'
      open(2,file=auxf,status='old',err=100)
      write(*,*)' A usar ficheiro auxiliar'
      exist=.true.
      return
100   exist=.false.
      return
      end
c.....
c     procede a ampliacao de uma zona do desenho
c     dadas as coordenadas da janela pretendida
      subroutine zoom
      logical yes
      common/lim1/xm,xn,ym,yn
      common/lim/xmax,xmin,ymax,ymin
      call prompt('Deseja ampliar alguma zona',yes)
      if (yes)then
            write(*,2)'Quais as coordenadas'
            write(*,2)'Escreva X1 X2 '
            read(*,*)xmin,xmax
            write(*,2)'Escreva Y1 Y2 '
            read(*,*)ymin,ymax
            xmin=amax1(xn,xmin)
            ymin=amax1(yn,ymin)
            xmax=amin1(xm,xmax)
            ymax=amin1(ym,ymax)
      else
            xmin=xn
            xmax=xm
            ymin=yn
            ymax=ym
      endif
      return
```

```
2       format(1x,a)
        end
c.....
c       determina a escala para a geometria de acordo
c       com as opcoes anteriores
        subroutine setscaleg
        common/lim/xmax,xmin,ymax,ymin
        common/size/six,siy
        common/scaleg/scale,x0,y0
        scale=amin1((six-2.)/(xmax-xmin),(siy-5.)/(ymax-ymin))
        x0=-xmin*scale+(six-(xmax-xmin)*scale)/2.
c       centra em altura
        y0=-ymin*scale+(siy-(ymax-ymin)*scale)/2.
        return
        end
c.....
        subroutine dscag
        real esc(11)
        common/lim/xmax,xmin,ymax,ymin
        common/size/six,siy
        common/scaleg/scale,x0,y0
        data (esc(i),i=1,11)
    1/.01,.1,.5,1.,10.,50.,100.,500.,1000.,5000.,10000./

        do 10 i=1,11
                if((xmax-xmin)/3.0.lt.esc(i))goto  20
10      continue
        write(*,*)'Nao desenho escala'
        goto 30
20      xl=esc(i-1)
        write(*,*)'Dimensao da escala grafica =',xl
c       dy=xl/20.
        dy=.5/scale
        x=xmin+xl/2.
        y=ymin-1./scale
        call line(x-xl/2.,y,x+xl/2.,y)
        call line(x+xl/2.,y-dy,x-xl/2.,y-dy)
        call line(x-xl/2.,y,x-xl/2.,y-dy)
        call line(x+xl/2.,y,x+xl/2.,y-dy)
        call line(x,y,x,y-dy)
        call dnumber(x-xl/2.-.4/scale,y-1./scale,.35,0.,0.,2)
        call dnumber(x+xl/2.-.4/scale,y-1./scale,.35,xl,0.,2)
30      return
        end
c.....
c       procura uma keyword no ficheiro auxiliar indicando
c       se foi encontrada
        subroutine find(text,found)
        common/auxe/exist
        logical found,exist
        character*(*)text
        character*20 keyw
```

```
      if(.not.exist)then
            found=.false.
            return
      endif
      rewind 2
1     read(2,'(a20)',end=100)keyw
      if(keyw.ne.text)goto 1
      found=.true.
      return
100   found=.false.
      return
      end
c.....
c     procede a execucao do desenho
      subroutine proceed(id,name,exist,xy,iele,apoio,stress,des)
      logical exist,yes,ften,fdes,yesm
      include 'datnum.for'
      common/size/six,siy
      common/fase/if,nume
      common/malhad/ja
      common /uau/titulo

      character*80 titulo
      character*30 nome(20)
      integer apoio(nap,3),iele(numelt,13)
      character*12 name
      real xy(numnpt,2),stress(numelt,3,18),des(numnpt,4)
      data (nome(i),i=1,7)
     1/'Desenho da malha','Malha deformada','Deslocamentos',
     1'Tensoes principais','Isolinhas de tensao',
     1'Isolinhas de deslocamento','Isolinhas de nivel de tensao'/

      call plots(1,six,siy,name)
      ja=0
      write(*,*)'Aberto o ficheiro plt'
      l=len(titulo)
      xlsize=siy/float(l)
      call msymbol(.5,siy-.3-xlsize,xlsize,titulo,0.,l)

      goto (1,2,2,5,6),id
1     iop=1
      if=0
      call malha(xy,iele)
      call prompt('Deseja numerar elementos',yes)
      if (yes) call elenum(xy,iele)
      call prompt('Deseja numerar nos',yes)
      if (yes) call nodenum(xy)
      goto 1000
2     if(exist)then
            call contour(xy)
      endif
      call prompt('Desenhar a malha ?',yesm)
```

```
      goto(1,3,4),id
3     iop=3
      call find('DESLOCAMENTOS',fdes)
      if(exist.and.fdes)then
c             call menud
      endif
      call prompt('Deseja numerar nos',yes)
      if (yes) call nodenum(xy)
      call ddesl(xy,des)
      if=1
      if(yesm)call malha(xy,iele)
      goto 1000
4     iop=4
      call find('TENSOES',ften)
      if(exist.and.ften)then
c             call menut
      endif
      call dten(iele,xy,stress)
      if=1
      if(yesm)call malha(xy,iele)
      goto 1000
5     iop=2
      if(exist)then
            call contour(xy)
      endif
      call prompt('Deseja desenhar malha inicial',yes)
      if=0
      if(yes)call malha(xy,iele)
      call deform(xy,iele,des)
      goto 1000
6     if(exist)then
            call contour(xy)
      endif
      call prompt('Deseja desenhar malha inicial',yes)
      if=0
      if(yes)call malha(xy,iele)
995   write(*,*)' Isolinhas de tensao.....1'
      write(*,*)' Isolinhas de deslocamentos.....2'
      write(*,*)' Isolinhas de nivel de tensao.....3'
      read(*,*)ioque
      goto (991,992,993),ioque
991         iop=5
            call isolt(if,xy,iele,stress)
            goto 1000
992         iop=6
            call isold(xy,iele,des)
            goto 1000
993         iop=7
            call isols(xy,iele,stress)
            goto 1000

1000  call msymbol(.5,siy-3.,.3,nome(iop),0.,30)
```

```
      call pltend
      return
      end
c.....
c     pergunta de escolha s/n
      subroutine prompt(text,anser)
      logical anser
      character*15 ok
      character*(*) text
      write(*,'(1x,a,2x,a)')text,'(s/n)'
      read(*,'(a)')ok
c     call lc(ok)
      if(ok(1:1).eq.'s')then
            anser=.true.
      else
            anser=.false.
      endif
      return
      end
c.....
      subroutine qfase(i)
      common/size/six,siy
      call msymbol(.5,siy-2.,.4,'FASE:  ',0.,6)
      call where(x,y)
      call number(x,y,.4,i*1.,0.,-1)
      return
      end
c.....
c     desenho da malha
      subroutine malha(xy,iele)
      include'datnum.for'
      real xy(numnpt,2)
      integer nos(3,9),iele(numelt,13),tipo
      common/size/six,siy
      common/fase/if,nume
      common/malhad/ja
      data((nos(i,j),j=1,9),i=1,3)/1,5,2,6,3,7,4,8,1,
     11,4,2,5,3,6,1,0,0,
     11,3,2,4,6,5,0,0,0/

      write(*,*)'Malha '
      if(if.eq.0)then
            nmax=numelt
      else
            nmax=nume
      endif
      if(ja.eq.0)then
            call dscag
            ja=1
      endif
      if(if.ne.0)call qfase(if)
      do 10 i=1,nmax
```

```
            tipo=iele(i,13)
            if(tipo.eq.1)then
                  npo=8
                  indic=1
            else if (tipo.eq.2)then
                  npo=6
                  indic=2
            else if (tipo.eq.4)then
                  npo=5
                  indic=3
            else
                  goto 10
            endif
            do 20 j=1,npo
                  x1=xy(iele(i,nos(indic,j)),1)
                  x2=xy(iele(i,nos(indic,j+1)),1)
                  y1=xy(iele(i,nos(indic,j)),2)
                  y2=xy(iele(i,nos(indic,j+1)),2)
                  call line(x1,y1,x2,y2)
20          continue
10    continue
      return
      end
c.....
      subroutine deform(xy,iele,des)
c     desenha a malha deformada
      include 'datnum.for'
      common/escd/dmax,dmx,dmy
      common/scaleg/scale,x0,y0
      common/fase/if,nume
      integer iele(numelt,13)
      real des(numnpt,4),xy(numnpt,2)

      if=1
      write(*,'(a)')' Numero da fase pretendida '
      read(*,*)ifase
      call locate(ifase,numel,maxno,erro)
      itipo=1
      call readd(des,maxno,itipo,numel)
      call dscd(du)
c     soma deslocamentos
      do 10 i=1,maxno
            xy(i,1)=xy(i,1)+des(i,1)/scale/du
            xy(i,2)=xy(i,2)+des(i,2)/scale/du
10    continue
      call malha(xy,iele)
c     subtrai deslocamentos
      do 20 i=1,maxno
            xy(i,1)=xy(i,1)-des(i,1)/scale/du
            xy(i,2)=xy(i,2)-des(i,2)/scale/du
20    continue
      return
```

```
      end
c.....
c     traca uma linha dentro do espaco disponivel e convertendo
c     a escala, os parametros entram em coordenadas reais
      subroutine line(x1,y1,x2,y2)
      logical ok1,ok2
      common/scaleg/scale,x0,y0
      call test(x1,y1,ok1)
      call test(x2,y2,ok2)
c     VAX tirar os comments dos ifs
c     if(ok1.and.ok2)then
            call plot(x1*scale+x0,y1*scale+y0,3)
            call plot(x2*scale+x0,y2*scale+y0,2)
c     endif
      return
      end
c.....
c     traca uma circunferencia em x,y de raio r
c     x,y sao dados em coordenadas reais e r em coordenadas
c     de desenho
      subroutine circulo(x,y,r,it)
      logical ok1
      common /scaleg/scale,x0,y0
     call test(x,y,ok1)
      if(ok1)then
c           call circul(x*scale+x0,y*scale+y0,r,it)
      endif
      return
      end
c.....
c     verificacao de coordenadas com aviso de corte de desenho
      subroutine test(xw,yw,result)
      logical result
      common/size/six,siy
      common/scaleg/scale,x0,y0
      x=xw*scale+x0
      y=yw*scale+y0
      if(x.gt.six.or.x.lt.0.0.or.y.gt.siy.or.y.lt.0.0)then
            result=.false.
            write(*,*)'Linhas truncadas'
      else
            result=.true.
      endif
      end
c.....
c     desenha um numero sendo as coordenadas reais
      subroutine dnumber(x1,y1,size,xnum,angle,ndec)
      logical ok1
      common/scaleg/scale,x0,y0
      call test(x1,y1,ok1)
      if(.not.ok1)then
            write(*,*)'Numero fora dos limites do desenho'
```

```
      endif
      call number(x1*scale+x0,y1*scale+y0,size,xnum,angle,ndec)
      return
      end
c.....
c     numera os nos
      subroutine nodenum(xy)
      include'datnum.for'
      common/size/six,siy
      real xy(numnpt,2)

      siz=amin1(six,siy)
      write(*,*)'Escreva o passo para a numeracao ou zero '
      read(*,*)ist
      if(ist.lt.1)ist=1
      do 10 i=1,numnpt,ist
            x1=xy(i,1)
            y1=xy(i,2)
            call dnumber(x1,y1,siz/120.,float(i),0.,-1)
10    continue
      return
      end
c.....
c     numera os elementos
      subroutine elenum(xy,iele)
      include'datnum.for'
      common/size/six,siy
      integer iele(numelt,13)
      real xy(numnpt,2)

      siz=amin1(six,siy)
      write(*,*)'Escreva o passo para a numeracao ou zero '
      read(*,*)ist
      if(ist.lt.1)ist=1
      do 10 i=1,numelt,ist
            call centro(iele,xy,i,sx,sy)
            sl=siz/100.
            call dnumber(sx,sy,sl,float(i),0.,-1)
10    continue
      return
      end
c.....
c     calcula aproximadamente a posicao do centro do elemento
      subroutine centro(iele,xy,i,sx,sy)
      include'datnum.for'
      integer nos(2,9),iele(numelt,13),tipo
      real xy(numnpt,2)
      data((nos(ik,j),j=1,9),ik=1,2)
     1/1,5,2,6,3,7,4,8,1,1,4,2,5,3,6,1,0,0/
      sx=0
      sy=0
      tipo=iele(i,13)
```

```
      if(tipo.eq.1)then
            npo=8
            indic=1
      else
            npo=6
            indic=2
      endif
      do 20 j=1,npo
            sx=sx+xy(iele(i,nos(indic,j)),1)
            sy=sy+xy(iele(i,nos(indic,j)),2)
20    continue
      sx=sx/npo
      sy=sy/npo
      return
      end
c.....
c     desenha o contorno de acordo com o estabelecido no ficheiro
c     auxiliar
      subroutine contour(xy)
      include 'datnum.for'
      common/scaleg/scale,x0,y0
      common/malhad/ja
      real xy(numnpt,2)
      integer icod,ponto
      logical found,ok,sim

      if(ja.eq.1)goto 100
      call dscag
      ja=1
100   call find('CONTORNO',found)
      if(found)then
            call prompt(' Desenhar o contorno',sim)
            if(.not.sim)goto 1111
            read(2,*)npts
            do 10 i=1,npts
                  read(2,*)ponto,icod
                  call test(xy(ponto,1),xy(ponto,2),ok)
                  if(ok)then
                        call plot(xy(ponto,1)*scale+x0,
   1                    xy(ponto,2)*scale+y0,ipen(icod))
                  endif
10          continue
      endif
1111  return
      end
c.....
c     conversao do codigo de estado da caneta
      function ipen(icod)
      if(icod.eq.0)then
            ipen=3
      else
            ipen=2
```

```
      endif
      end
c.....
c     salta registos na unidade 9
      subroutine skip9(n)
      do 10 i=1,n
            read(9)
10    continue
      return
      end
c.....
c     prepara desenho de deslocamentos
      subroutine ddesl(xy,des)
      include 'datnum.for'
      real xy(numnpt,2),des(numnpt,4)
      logical erro,e1,e2
      common/escd/dmax,dmx,dmy

      write(*,*)'Deslocamentos totais (1) ou incrementais (0) '
      read(*,*)itipo
      if(itipo.eq.1)then
            write(*,1)'Numero da fase pretendida '
            read(*,*)ifase
            call locate(ifase,numel,maxno,erro)
            call readd(des,maxno,itipo,numel)
      else
            write(*,1)'Indique: Fase1 Fase2 '
            read(*,*)if1,if2
            if(if2.lt.if1)then
                  ip=if2
                  if2=if1
                  if1=ip
            endif
            call locate(if1,numel,maxno,e1)
            call readd(des,maxno,1,numel)
            call locate(if2,numel1,maxno1,e2)
            call readd(des,maxno1,3,numel1)
            if(.not.e1.and..not.e2)then
                  dmax=0
                  dmx=0
                  dmy=0
c                 desloc's incrementais
                  do 10 i=1,maxno
                        do 11 j=1,2
                        des(i,j)=des(i,j+2)-des(i,j)
                       dmax=amax1(dmax,(des(i,2)**2
     1                  +des(i,1)**2)**.5)
11                      continue
                        dmx=amax1(dmx,abs(des(i,1)))
                        dmy=amax1(dmy,abs(des(i,2)))
10                2     continue
                  erro=.false.
```

```
            endif
        endif
        if(.not.erro)then
            call plodes(xy,des,maxno)
        endif
        return
1       format(1x,a)
        end
c.....
c       le deslocamentos pretendidos
        subroutine readd(des,maxno,itipo,numskip)
        include 'datnum.for'
        common/escd/dmax,dmx,dmy
        real des(numnpt,4)

c       salta o que nao interessa
        call skip9(numskip)
        dmax=0
        do 10 i=1,maxno
            read(9)x,x,des(i,itipo),des(i,itipo+1)
            dmax=amax1(dmax,(des(i,2)**2+des(i,1)**2)**.5)
            dmx=amax1(dmx,abs(des(i,1)))
            dmy=amax1(dmy,abs(des(i,2)))
10      continue
        return
        end
c.....
c       desenha deslocamentos
        subroutine plodes(xy,des,maxno)
        include'datnum.for'
        common/escd/dmax,dmx,dmy
        common/scaleg/scale,x0,y0
        real xy(numnpt,2),des(numnpt,4)
c       escala de deslocamentos
        call dscd(du)
        do 10 i=1,maxno
            x1=xy(i,1)
            x2=xy(i,1)+des(i,1)/scale/du
            y1=xy(i,2)
            y2=xy(i,2)+des(i,2)/scale/du
            call circulo(x1,y1,.06,1)
            call line(x1,y1,x2,y2)
10      continue
        return
1       format(1x,a)
        end
c.....
        subroutine dscd(du)
        common/lim/xmax,xmin,ymax,ymin
        common/size/six,siy
        common/scaleg/scale,x0,y0
        common/escd/dmax,dmx,dmy
```

```
      common/desl/dud
c     isetf=formato papel
c     isetd=escala desenho
c     isett=escala tensao
      common/escalas/isetf,isetd,isett
      x=(xmax+xmin)/2.
c     y=ymin-(ymax-ymin)/18.
      y=ymin-1.25/scale
      write(*,*)'Escala grafica:'
      write(*,*)'Maximo deslocamento=',dmax, ' em X=',dmx,' em Y=',dmy
      write(*,*)'1 Cm desenho =  x unidades de calculo'
      if(isetd.eq.1)then
            write(*,*)' 1 Cm = ',du
            write(*,*)'Escreva novo valor para x ou 0 '
            read(*,*)du1
            if(du1.ne.0.0)du=du1
      else
            write(*,*)'Qual o valor para x ? '
            read(*,*)du
            isetd=1
      endif
      dud=du
      call line(x-.5/scale,y,x+.5/scale,y)
      call where(x,y)
      call msymbol(x,y,.4,'  = ',0.,4)
      call where(x,y)
      call number(x,y,.4,du,0.,4)
      return
      end
c.....
c     desenha tensoes
      subroutine dten(iele,xy,stress)
      common /tens/  tenmax,dut
      common/esct/idt
      include 'datnum.for'
      logical erro
      integer iele(numelt,13)
      real xy(numnpt,2),stress(numelt,3,18)
      write(*,1)'Qual a fase pretendida '
      read(*,*)ifase
      call locate(ifase,numel,maxno,erro)
      if(.not.erro)then
            call qfase(ifase)
            call leten(stress,1)
            write(*,*)'Maxima tensao=',tenmax
            write(*,*)'1 cm de desenho = x unidades de calculo'
            if(idt.eq.1)then
                  write(*,*)'1 Cm=',dut
                  write(*,*)'Escreva novo valor ou 0'
                  read(*,*)du1
                  if(du1.ne.0.0)dut=du1
            else
```

```
                write(*,*)'Qual o valor para x'
                read(*,*)dut
                idt=1
            endif
            call dsct
            call ploten(iele,xy,stress)
        endif
        return
1       format(1x,a)
        end
c.....
c       desnha escala de tensoes
        subroutine dsct
        real t(8)
        common /tens/  tenmax,dut
        common/lim/xmax,xmin,ymax,ymin
        common/scaleg/scale,x0,y0
        common/esct/idt
        data t/.1,1,10,100,500,1000,5000,10000/
        do 10 i=1,8
            if(dut.le.t(i))goto 20
10      continue
        i=i-1
20      inde=i
c       escala grafica sx=posicao x  sy=posicao y
30      sx=(xmax+xmin)/2.
c       sy=ymin-(ymax-ymin)/5.
        sy=ymin-1.25/scale
        c=1
        s=0
        dx1=t(inde)*c/scale/dut
        dy1=t(inde)*s/scale/dut
        call arrow(sx,sy,dx1/2.,dy1/2.)
        call arrow(sx,sy,-dx1/2.,-dy1/2.)
        dx1=t(inde)*s/scale/dut
        dy1=-t(inde)*c/scale/dut
        call arrow(sx,sy,dx1/2.,dy1/2.)
        call arrow(sx,sy,-dx1/2.,-dy1/2.)
        call where(sx,sy)
        call msymbol(sx+t(inde)/dut,sy-dx1/2.,.35,'   = ',0.,3)
        call where(sx,sy)
        call number(sx,sy,.35,t(inde),0.,2)
        return
        end
c.....
        subroutine leten(stress,ic)
        include 'datnum.for'
        common /tens/  tenmax,dut
        real stress(numelt,3,18),pr(3)
        do 10 n=1,numel
            read(9)((stress(n,i,ik),i=1,3),ik=1,10)
            if(ic.eq.1)then
```

```
c               calcula valores principais
                call tenpri(stress,pr,n,10)
c               guarda os valores principais
                stress(n,1,10)=pr(1)
                stress(n,2,10)=pr(2)
                stress(n,3,10)=pr(3)
                tenmax=amax1(tenmax,abs(pr(1)))
                endif
10      continue
        return
        end
c.....
c       ler o nivel de tensao
        subroutine lelev(stress,numel1,maxno)
        include 'datnum.for'
        common /tens/  tenmax,dut
        real stress(numelt,3,18),pr(3)

        call skip9(numel1)
        call skip9(maxno)
        do 10 n=1,numel1
                read(9)((stress(n,i,ik),i=1,2),ik=1,10)
10      continue
        return
        end
c.....
c       desenha seta
c       x e y em coordenadas reais
c       dx,dy em cm
        subroutine arrow(x,y,dx,dy)
        s=sqrt(dx**2+dy**2)
c       o comprimento dq seta e' 1/10 de s
        if(dx.ne.0.)then
                alf=atan(dy/dx)
        else
                alf=3.14/2.
        endif
        x1=x+dx
        y1=y+dy
c       s1=s/10.
c       x2=x1-s1*cos(alf+.15)
c       y2=y1-s1*sin(alf+.15)
c       x3=x1-s1*cos(alf-.15)
c       y3=y1-s1*sin(alf-.15)

        call line(x,y,x1,y1)
c       call line(x1,y1,x2,y2)
c       call line(x1,y1,x3,y3)
        return
        end
c.....
```

```
      subroutine princi(vector,pr)
c
c     Esta subrotina calcula valores principais do vector
c
      real vector(3),pr(3)
c
      centro=(vector(1)+vector(2))/2.
      b=(vector(2)-vector(1))/2.
      raio=sqrt(vector(3)**2.+b**2.)
c     write(*,*)'b=',b,' v1,v2,v3 ',(vector(i),i=1,3)
      pr(1)=centro+raio
      pr(2)=centro-raio
      if(b.eq.0)then
            alfa=3.14/2.
      else
            alfa=atan2(vector(3),b)/2.
c           write(*,*)alfa*180./3.14
      endif
c     Passa para graus
      pr(3)=alfa
c
      return
      end
c.....
c     tracadó dos vectores de tensao
      subroutine ploten(iele,xy,stress)

      include 'datnum.for'
      integer iele(numelt,13)
      real xy(numnpt,2),stress(numelt,3,18)
      common /tens/  tenmax,dut
      common /scaleg/  scale,x0,y0

      do 10 n=1,numel
c           salta elementos de junta
            if(iele(n,13).eq.3)goto 10
            call centro(iele,xy,n,sx,sy)
            alf=3.14/2-stress(n,3,10)
            c=cos(alf)
            s=sin(alf)
            dx1=stress(n,1,10)*c/scale/dut
            dy1=stress(n,1,10)*s/scale/dut
            call arrow(sx,sy,dx1/2.,dy1/2.)
            call arrow(sx,sy,-dx1/2.,-dy1/2.)
            dx1=stress(n,2,10)*s/scale/dut
            dy1=-stress(n,2,10)*c/scale/dut
            call arrow(sx,sy,dx1/2.,dy1/2.)
            call arrow(sx,sy,-dx1/2.,-dy1/2.)
10    continue
      return
      end
c.....
```

```
      subroutine ISOL(x,s,xiso,itipo)
c     recebe as coordenadas do elemento
c     recebe o valores de uma grandeza
c     recebe o valor a tracar a isolinha
      integer lado(2,8,3,2)
c     2 tipos de elementos
c     com o maximo de 8 triangulos
c     com 3 lados cada
c     com 2 nos por lado
      real x(9,2),s(9)
      logical esta

c     neste data estao os elementos necessarios para definir
c     8 triangulos em elementos rectangulares e 6 triangulos
c     nos elementos triangulares
c     alterado em 900117 com base em sugestao de Muralha
c     em considerar apenas 6 triangulos no elemento rectangular
c     sem criar nenhum ponto adicional. de acordo com isto
c     apenas considero 4 no triangular
      data ((((lado(i,j,k,m),m=1,2),k=1,3),j=1,8),i=1,2)
     1/5,2,2,6,6,5,
     1 6,3,3,7,7,6,
     1 7,4,4,8,8,7,
     1 8,5,5,1,1,8,
     1 5,6,6,8,8,5,
     1 6,7,7,8,8,6,
     1 0,0,0,0,0,0,
     1 0,0,0,0,0,0,
     1 1,4,4,6,6,1,
     1 4,2,2,5,5,4,
     1 5,3,3,6,6,5,
     1 4,5,5,6,6,4,
     1 0,0,0,0,0,0,
     1 0,0,0,0,0,0,
     1 0,0,0,0,0,0,
     2 0,0,0,0,0,0/
c     elemento de junta, nao traca
      if(itipo.eq.3.or.itipo.eq.4)return
c     fixa n. de triangulos do elemento
      kk=6
      if(itipo.eq.2)kk=4
c     corre todos os triangulos
      do 10 i=1,kk
c           e para cada triangulo os 3 lados
            do 20 j=1,3
c                 lados a estudar
                  l1=j
                  l2=j+1
                  if(l2.gt.3)l2=l2-3
c                 acha o ponto na 1. fronteira
                  n1=lado(itipo,i,l1,1)
                  n2=lado(itipo,i,l1,2)
```

```
                    s1=s(n1)
                    s2=s(n2)
                    call conti(xiso,s1,s2,esta)
                    if(esta)then
c                       guarda coordenadas
                        x1=x(n1,1)
                        x2=x(n2,1)
                        y1=x(n1,2)
                        y2=x(n2,2)
                        call posp(x1,y1,x2,y2,xiso,s1,s2,xp1,yp1)
                    else
c                       nao encontrou vai embora para
c                       os outros lados
                        goto 20
                    endif
c                   acha o ponto na 2. fronteira
                    n1=lado(itipo,i,l2,1)
                    n2=lado(itipo,i,l2,2)
                    s1=s(n1)
                    s2=s(n2)
                    call conti(xiso,s1,s2,esta)
                    if(esta)then
c                       guarda coordenadas
                        x1=x(n1,1)
                        x2=x(n2,1)
                        y1=x(n1,2)
                        y2=x(n2,2)
                        call posp(x1,y1,x2,y2,xiso,s1,s2,xp2,yp2)
                        call line(xp1,yp1,xp2,yp2)
                    else
c                       nao encontrou vai embora para
c                       outra fronteira
                        goto 20
                    endif
20              continue
10      continue
        return
        end
c.....
c       verifica se xiso esta contido em [s1,s2]
        subroutine conti(xiso,s1,s2,esta)
        logical esta
        esta=.false.
        sm=amax1(s1,s2)
        sn=amin1(s1,s2)
        if(xiso.ge.sn.and.xiso.le.sm)esta=.true.
        return
        end
c.....
c       calcula xp,yp para o ponto xiso
        subroutine posp(x1,y1,x2,y2,xiso,s1,s2,xp2,yp2)
c       calcula as coordenadas com base numa recta
```

```
      xp2=recta(x1,s1,x2,s2,xiso)
      yp2=recta(y1,s1,y2,s2,xiso)
      return
      end
c.....
      function recta(x1,y1,x2,y2,xiso)
c     necessario verificar funcionamento
      if(x2.eq.x1)then
            recta=x1
      else
            xm=(y2-y1)/(x2-x1)
            if(xm.eq.0)then
c                 falta decidir o que fazer
                  write(*,*)' Erro. declive =0 em recta'
            endif
            recta=(xiso-y1+xm*x1)/xm
      endif
      end

c.....
c
c     transferir coordenadas p/ XE e calcular as
c     coordenadas do ponto central
c
c.....
      subroutine tranxe(iele,xy,xe,ie)
      include 'datnum.for'
      integer iele(numelt,13)
      real xy(numnpt,3),xe(9,2)
c
      sx=0
      sy=0
      goto (1,2,3) iele(ie,13)
1     nno=8
      goto 4
2     nno=6
      goto 4
3     nno=6

4     do 5 i=1,nno
            xe(i,1)=xy(iele(ie,i),1)
            xe(i,2)=xy(iele(ie,i),2)
            sx=sx+xe(i,1)
            sy=sy+xe(i,2)
5     continue
      xe(9,1)=sx/(1.*nno)
      xe(9,2)=sy/(1.*nno)
      return
      end
c.....
      subroutine isold(xy,iele,des)
c     desenha isolinhas de deslocamentos
```

```
      include 'datnum.for'
      common/size/six,siy
      common/escd/dmax,dmx,dmy
      common/scaleg/scale,x0,y0
      common/fase/if,nume
      logical found
      character*2 desloca(2)
      integer iele(numelt,13)
      real des(numnpt,4),xy(numnpt,2),xe(9,2),s(9),xisov(50)
      data desloca/'dX','dY'/

      if=1
      write(*,'(a)')' Numero da fase pretendida '
      read(*,*)ifase
      call locate(ifase,numel,maxno,erro)
      call qfase(ifase)
      itipo=1
      call readd(des,maxno,itipo,numel)
1     write(*,*)' Isolinhas X-1 ou Y-2 '
      read(*,*)ixy
      goto (2,2),ixy
c     resposta errada
      goto 1
2     write(*,*)' Maximo valor de deslocamento=',dmax
c
c     escreve qual a isolinha pretendida
      call msymbol(.5,siy-3.5,.3,desloca(ixy),0.,2)
c     procura no ficheiro auxiliar
      if(ixy.eq.1)then
            call find('IDESLOCX',found)
      else
            call find('IDESLOCY',found)
      endif
      if(found)then
            write(*,*)' Escreva o n. de isovalores ou 0 '
            read(*,*)niso
      else
            write(*,*)' Escreva o n. de isovalores '
            read(*,*)niso
      endif
      if(.not.found.or.niso.ne.0)then
            write(*,*)' Escreva os isovalores a desenhar'
            write(*,*)' Nao esqueca que os valores podem ser negativos '
            read(*,*)(xisov(i),i=1,niso)
      else
            read(2,*)niso
            read(2,*)(xisov(i),i=1,niso)
      endif
      call escriso(xisov,niso)
      do 10 i=1,nume
            call tranxe(iele,xy,xe,i)
            sum=0
```

```
                np=0
                do 11 j=1,8
                        s(j)=0
                        if(iele(i,j).ne.0)then
                                s(j)=des(iele(i,j),ixy)
                                sum=sum+s(j)
                                np=np+1
                        endif
11              continue
c               valor no ponto central
                s(9)=sum/(1.*np)
                do 12 j=1,niso
                        call isol(xe,s,xisov(j),iele(i,13))
12              continue
10      continue
        return
        end
c.....
        subroutine escriso(xi,n)
        common/size/six,siy
        real xi(n)
        y=siy-4.
        call msymbol(.5,y,.4,'Isovalores:',0.,11)
        call where(x,y)
        do 10 i=1,n
                call number(x,y,.4,xi(i),0.,2)
                call where(x,y)
                if(i.lt.n)then
                        call msymbol(x,y,.4,',',0.,1)
                        call where(x,y)
                endif
10      continue
        return
        end
c.....
        subroutine isolt(fase,xy,iele,stress)
c       desenha isolinhas de tensoes
        include 'datnum.for'
        common/size/siy,six
        common/scaleg/scale,x0,y0
        common/fase/if,nume
        common /tens/  tenmax,dut
        common/atmos/patm
        character*3 tensao(5)
        integer nossub(400),elesub(200),fase
        real alturas(400)
        integer iele(numelt,13)
        real xy(numnpt,2),xe(9,2),s(9),xisov(50)
        real stress(numelt,3,18),pr(3)
        logical found
        data tensao/'Sx  ','Sy  ','tau','S1  ','S3  '/
        if=1
```

```
      write(*,'(a)')' Numero da fase pretendida '
      read(*,*)ifase
      call locate(ifase,numel,maxno,erro)
      call qfase(ifase)
      itipo=1
      call leten(stress,0)
c     medianizar as tensoes
      open(unit=12,file='medias.dat',access='direct',
     1recl=16,status='scratch')
c     achar as tensoes nos pontos nodais
      call tenpon(stress,iele)
c     inicializar
      do 1003 i=1,numnpt
1003  write(12,rec=i)0.,0.,0.,0
      nomax=0
      do 1001 i=1,nume
            itipo=iele(i,13)
c           nao calcula medias para os elementos de
c           junta sem espessura
            if(itipo.ne.3)then
                  do 1002 j=1,8
                        no=iele(i,j)
                        nomax=max(nomax,no)
                        if(no.ne.0)then
                              read(12,rec=no)sx,sy,tau,n
                              sx=sx+stress(i,1,j+10)
                              sy=sy+stress(i,2,j+10)
                              tau=tau+stress(i,3,j+10)
                              n=n+1
                              write(12,rec=no)sx,sy,tau,n
                        else
                              goto 1001
                        endif
1002                continue
            endif
1001  continue
      sxm=0
      sym=0
      taum=0
c     escreve valores ja medianizados
      do 1010 i=1,nomax
            no=i
            read(12,rec=no)sx,sy,tau,n
            sx=sx/(1.*n)
            sy=sy/(1.*n)
            tau=tau/(1.*n)
            write(12,rec=no)sx,sy,tau,1
c           determina valores maximos
            sxm=amax1(abs(sx),sxm)
            sym=amax1(abs(sy),sym)
            taum=amax1(abs(tau),taum)
1010  continue
```

```
c       agora vamos medianizar os valores obtidos nos
c       elementos de junta mas calculados com base nos
c       outros elementos
        do 1007 i=1,nume
                itipo=iele(i,13)
                if(itipo.eq.3)then
                        do 1028 j=1,3
                                no=iele(i,j)
                                no1=iele(i,j+3)
                                read(12,rec=no)sx,sy,tau,n
                                read(12,rec=no1)sx1,sy1,tau1,n1
                                sx=(sx+sx1)/2.
                                sy=(sy+sy1)/2.
                                tau=(tau+tau1)/2.
                                write(12,rec=no)sx,sy,tau,1
                                write(12,rec=no1)sx,sy,tau,1
1028                    continue
                endif
1007    continue
c       apos as medianizacoes vamos tornar as tensoes em totais
c       1 chama-se a subroutine que verifica no ficheiro de dados
c       se ha elementos submersos
        call elemsub(xy,iele,nossub,alturas,fase,nnossub)
        if(nnossub.ne.0)then
                write(*,*)'Deseja considerar tensoes totais (1) ?'
                read(*,*)ttot
                if(ttot.ne.0.0)then
                        ttot=1.
                        do 1008 i=1,nnossub
                                no=nossub(i)
                                read(12,rec=no)sx,sy,tau,n
                                sx=sx+patm/10.33*alturas(i)
                                sy=sy+patm/10.33*alturas(i)
                                write(*,*)no,sx,sy,tau,alturas(i)
                                write(12,rec=no)sx,sy,tau,n
1008                    continue
                endif
        endif
        do 1004 i=1,nume
                if(iele(i,13).ne.3)then
                        do 1005 j=1,8
                                no=iele(i,j)
                                if(no.ne.0)then
                                        read(12,rec=no)sx,sy,tau,n
                                        stress(i,1,10+j)=sx
                                        stress(i,2,10+j)=sy
                                        stress(i,3,10+j)=tau
                                else
                                        goto 1004
                                endif
1005                    continue
                endif
```

```
1004  continue

1     write(*,*)' Isolinhas Sx-1  Sy-2  Tau-3  S1-4  S3-5 '
      read(*,*)ixy
      goto (2,2,2,2,2),ixy
      goto 1

2     write(*,'(1x,3a10/1x,3f10.4)')'Sx','Sy','Tau',sxm,sym,taum
c
c     escreve qual a isolinha pretendida
c
      call msymbol(.5,siy-3.5,.3,tensao(ixy),0.,3)
      goto (881,882,883,884,885),ixy

881   call find('ITENSAOX',found)
      GOTO 890
882   call find('ITENSAOY',found)
      GOTO 890
883   call find('ITENSAOT',found)
      GOTO 890
884   call find('ITENSAO1',found)
      GOTO 890
885   call find('ITENSAO3',found)
      GOTO 890

890   if(found)then
            write(*,*)' Escreva o n. de isovalores ou 0 '
            read(*,*)niso
      else
            write(*,*)' Escreva o n. de isovalores '
            read(*,*)niso
      endif
      if(.not.found.or.niso.ne.0)then
            write(*,*)' Escreva os isovalores a desenhar'
            write(*,*)' Nao esqueca que os valores podem ser negativos '
            read(*,*)(xisov(i),i=1,niso)
      else
            read(2,*)niso
            read(2,*)(xisov(i),i=1,niso)
      endif
      call escriso(xisov,niso)
      do 10 i=1,nume
            if(iele(i,13).ne.3)then
            call tranxe(iele,xy,xe,i)
            do 11 j=11,19
                  is=j-10
                  iie=is
                  istre=j
                  if(istre.eq.19)istre=10
                  if(iie.eq.9)iie=1
                  s(is)=0
                  if(iele(i,iie).ne.0)then
```

```
                        goto (101,102,103,104,104)ixy

c                       tensao xx
101                     s(is)=stress(i,1,istre)
                        goto 200

c                       tensao yy
102                     s(is)=stress(i,2,istre)
                        goto 200

c                       tensao tangencial
103                     s(is)=stress(i,3,istre)

c                       calcular tensoes principais
104                     call tenpri(stress,pr,i,istre)
                        goto (106,107),(ixy-3)

c                       tens. princ. max.
106                     s(is)=pr(1)
                        goto 200

c                       tens. princ.min.
107                     s(is)=pr(2)
200                     continue
                    endif
11              continue

                do 12 j=1,niso
                    call isol(xe,s,xisov(j),iele(i,13))
12              continue
                endif
10      continue
        return
        end
c.....
        subroutine isols(xy,iele,stress)
c       desenha isolinhas de nivel de tensao
        include 'datnum.for'
        common/size/six,siy
        common/scaleg/scale,x0,y0
        common/fase/if,nume
        common /tens/  tenmax,dut
        character*5 strel(2)
        integer iele(numelt,13)
        real xy(numnpt,2),xe(9,2),s(9),xisov(50)
        real stress(numelt,3,18),pr(3)
        logical found
        data strel/'SL    ','SLmax'/

        if=1
        write(*,'(a)')' Numero da fase pretendida '
        read(*,*)ifase
```

```
      call locate(ifase,numel,maxno,erro)
      call qfase(ifase)
      itipo=1
      call lelev(stress,numel,maxno)
c     medianizar as tensoes
      open(unit=12,file='medias.dat',access='direct',
     1recl=16,status='scratch')
c     achar as tensoes nos pontos nodais
      call levpon(stress,iele)
c     inicializar
      do 1003 i=1,numnpt
1003  write(12,rec=i)0.,0.,0.,0
      nomax=0
      do 1001 i=1,nume
            itipo=iele(i,13)
c           nao calcula medias para os elementos de
c           junta sem espessura
            if(itipo.ne.3)then
                  do 1002 j=1,8
                        no=iele(i,j)
                        nomax=max(nomax,no)
                        if(no.ne.0)then
                              read(12,rec=no)sl,slm,n
                              sl=sl+stress(i,1,j+10)
                              slm=slm+stress(i,2,j+10)
                              n=n+1
                              write(12,rec=no)sl,slm,n
                        else
                              goto 1001
                        endif
1002                continue
            endif
1001  continue
c     escreve valores ja medianizados
      do 1010 i=1,nomax
            no=i
            read(12,rec=no)sl,slm,n
            sl=sl/(1.*n)
            slm=slm/(1.*n)
            write(12,rec=no)sl,slm,1
1010  continue
c     agora vamos medianizar os valores obtidos nos
c     elementos de junta mas calculados com base nos
c     outros elementos
      do 1007 i=1,nume
            itipo=iele(i,13)
            if(itipo.eq.3)then
                  do 1008 j=1,3
                        no=iele(i,j)
                        no1=iele(i,j+3)
                        read(12,rec=no)sl,slm,n
                        read(12,rec=no1)sl1,slm1,n1
```

```
                        sl=(sl+sl1)/2.
                        slm=(slm+slm1)/2.
                        write(12,rec=no)sl,slm,1
                        write(12,rec=no1)sl,slm,1
1008                continue
            endif
1007    continue
        do 1004 i=1,nume
            if(iele(i,13).ne.3)then
                do 1005 j=1,8
                    no=iele(i,j)
                    if(no.ne.0)then
                        read(12,rec=no)sl,slm,n
                        stress(i,1,10+j)=sl
                        stress(i,2,10+j)=sl
                    else
                        goto 1004
                    endif
1005                continue
            endif
1004    continue
1       write(*,*)' Isolinhas Sl -1    Slmax -2 '
        read(*,*)ixy
        goto (2,2),ixy
        goto 1

c       escreve qual a isolinha pretendida
c
2       call msymbol(.5,siy-3.5,.3,strel(ixy),0.,5)
        goto (881,882),ixy

881     call find('SL',found)
        GOTO 890
882     call find('ISLMAX',found)
        GOTO 890

890     if(found)then
            write(*,*)' Escreva o n. de isovalores ou 0 '
            read(*,*)niso
        else
            write(*,*)' Escreva o n. de isovalores '
            read(*,*)niso
        endif
        if(.not.found.or.niso.ne.0)then
            write(*,*)' Escreva os isovalores a desenhar'
            write(*,*)' Nao esqueca que os valores podem ser negativos '
            read(*,*)(xisov(i),i=1,niso)
        else
            read(2,*)niso
            read(2,*)(xisov(i),i=1,niso)
        endif
        call escriso(xisov,niso)
```

```
      do 10 i=1,nume
            if(iele(i,13).ne.3)then
            call tranxe(iele,xy,xe,i)
            do 11 j=11,19
                  is=j-10
                  iie=is
                  istre=j
                  if(istre.eq.19)istre=10
                  if(iie.eq.9)iie=1
                  s(is)=0
                  if(iele(i,iie).ne.0)then
                        goto (101,102)ixy
c                       SLEV
101                     s(is)=stress(i,1,istre)
                        goto 200
c                       SLEV MAX
102                     s(is)=stress(i,2,istre)
                        goto 200
200                     continue
                  endif
11          continue
            do 12 j=1,niso
                  call isol(xe,s,xisov(j),iele(i,13))
12          continue
            endif
10    continue
      return
      end
c.....
      subroutine tenpri(stress,pr,n,ipoint)
      include 'datnum.for'
      real stress(numelt,3,18),s(3),pr(3)
      s(1)=stress(n,1,ipoint)
      s(2)=stress(n,2,ipoint)
      s(3)=stress(n,3,ipoint)
      call princi(s,pr)
      return
      end
c.....
      subroutine tenpon(stress,iele)
      include 'datnum.for'
      real stress(numelt,3,18),ss(18,3),t(4),tn(8)
      integer iele(numelt,13)
      common/gauss/ng8,ng6
      common/fase/if,nume
      call mat8(ng8)
      call mat6
      do 10 ii=1,nume
            do 20 j=1,3
                  do 30 k=1,18
                        ss(k,j)=stress(ii,j,k)
30                  continue
```

```
20          continue

            if(iele(ii,13).eq.1)then
                  do 1001 i=1,3
                        do 1002 j=1,4
1002                    t(j)=ss(j,i)
                        call inter8(t,tn)
                        do 1003 j=1,8
1003                    ss(j+10,i)=tn(j)
1001                  continue
            elseif(iele(ii,13).eq.2)then
                  call inter6(ss)
            endif
            do 201 j=1,3
                  do 301 k=1,18
                        stress(ii,j,k)=ss(k,j)
301                   continue
201         continue
10    continue
      return
      end
c.....
      subroutine levpon(stress,iele)
      include 'datnum.for'
      real stress(numelt,3,18),ss(18,3),t(4),tn(8)
      integer iele(numelt,13)
      common/gauss/ng8,ng6
      common/fase/if,nume
      call mat8(ng8)
      call mat6
      do 10 ii=1,nume
            do 20 j=1,2
                  do 30 k=1,18
                        ss(k,j)=stress(ii,j,k)
30                    continue
20          continue

            if(iele(ii,13).eq.1)then
                  do 1001 i=1,2
                        do 1002 j=1,4
1002                    t(j)=ss(j,i)
                        call inter8(t,tn)
                        do 1003 j=1,8
1003                    ss(j+10,i)=tn(j)
1001                  continue
            elseif(iele(ii,13).eq.2)then
                  call inter6(ss)
            endif
            do 201 j=1,2
                  do 301 k=1,18
                        stress(ii,j,k)=ss(k,j)
301                   continue
```

```
201             continue
10      continue
        return
        end
c-------------------------------------------------
        subroutine inter8(s,sn)
c       recebe os valores de s e calcula os de sn
c       com base num plano
        real xy(8,2),coef(3),s(4),sn(8)
        common/int8/xmat(3,4)
        data((xy(i,j),j=1,2),i=1,8)
     ./-1.,-1.,1.,-1.,1.,1.,-1.,1.,
     . 0.,-1.,1.,0.,0.,1.,-1.,0./

        call inicv(coef,3)
        call inicv(sn,8)
c       determina constantes do plano
        do 10 i=1,3
                do 20 j=1,4
                coef(i)=coef(i)+xmat(i,j)*s(j)
20              continue
10      continue
c       determina valores nos pontos nodais
        do 30 i=1,8
                x=xy(i,1)
                y=xy(i,2)
                sn(i)=coef(1)*x+coef(2)*y+coef(3)
30      continue
        return
        end
c-------------------------------------------------
        subroutine inter6(ss)
        common/gauss/ng8,ng6
        common/int6/rmat(3,3)
        real ss(18,3),coef(3),xy(3,2)
        data ((xy(i,j),j=1,2),i=1,3)/1.,0.,0.,1.,0.,0./

        do 10 k=1,3
         if(ng6.eq.3)then
           call inicv(coef,3)
           do 20 i=1,3
             do 30 j=1,3
               coef(i)=coef(i)+rmat(i,j)*ss(j,k)
30           continue
20         continue
           do 40 i=1,3
             ss(10+i,k)=coef(1)*xy(i,1)+coef(2)*xy(i,2)+coef(3)
40         continue
           do 50 i=1,3
             ss(13+i,k)=ss(i,k)
50         continue
         else
```

```
            do 60 i=1,3
60          ss(10+i,k)=ss(4+i,k)
            do 70 i=1,3
70          ss(13+i,k)=ss(i+1,k)
           endif
10        continue
          return
          end
c.....
          subroutine mat8(ng)
          common/int8/xmat(3,4)
          real mat(3,4,2)
          data(((mat(i,j,k),j=1,4),i=1,3),k=1,2)
     ./0.4330127,  0.4330127, -.4330127, -.4330127
     .,0.4330127, -.4330127, 0.4330127, -.4330127
     .,0.2500000, 0.2500000, 0.2500000, 0.2500000
     .,0.327,0.327,-0.327,-0.327
     .,0.327,-0.327,0.327,-0.327
     .,0.25,0.25,0.25,0.25/
          if(ng.eq.2)then
                  k=1
          else
                  k=2
          endif
          do 10 i=1,3
                  do 20 j=1,4
                          xmat(i,j)=mat(i,j,k)
20                continue
10        continue
          return
          end
c.....
          block data mat6
          common/int6/rmat(3,3)
          data ((rmat(i,j),j=1,3),i=1,3)
     1/2.,-2.,0.
     2,2.,0.,-2.
     3,-1.,1.,1./
          end
c-----------------------------------------------------
          subroutine elemsub(xy,iele,nossub,alturas,fase,nnossub)
          include 'datnum.for'
          integer nossub(400),fase,elesub(200)
          real alturas(400)
          character*80 linha
          integer iele(numelt,13)
          real xy(numnpt,2)
c         recomeca leitura do ficheiro de dados
          rewind 1
          ifc=0
          nelesub=0
1         read(1,'(a)',end=100)linha
```

```
      if(index(linha,'#').ne.0)then
c               encontrada uma fase
                ifc=ifc+1
                write(*,*)linha
                if(index(linha,'#3').ne.0)then
c                       e uma fase de enchimento
                        call leelesub(elesub,nelesub)
                endif
                if(ifc.eq.fase)then
c                       e a fase pretendida
                        if(index(linha,'#3').ne.0)then
c                               e uma fase com agua
                                write(*,*)'Encontrada a fase pretendida'
                                call altura(xy,iele,alturas,elesub,nelesub,
     1                          nossub,nnossub)
                                return
                        else
                                return
                        endif
                endif
        endif
        goto  1
100     write(*,*)'A fase pedida nao foi encontrada no ficheiro de dados'
        write(*,*)'Por esse motivo nao posso calcular as tensoes totais'
        return
        end
c.....
        subroutine leelesub(elesub,nelesub)
        integer elesub(200)
        read(1,*)n
        read(1,*)(elesub(i+nelesub),i=1,n)
        nelesub=nelesub+n
        return
        end
c.....
        subroutine altura(xy,iele,alturas,elesub,nelesub,nossub,nnossub)
        include 'datnum.for'
        integer nossub(400),elesub(200)
        real alturas(400)
        integer iele(numelt,13)
        real xy(numnpt,2)
        nnossub=0
        do 10 i=1,nelesub
c               corre os elementos lidos como submersos
                kel=elesub(i)
                do 20 j=1,8
c                       corre todos os nos de cada elemento
                        if(iele(kel,j).ne.0)then
                                do 30 k=1,nnossub
c                                       verifica se este no ja foi considerado
                                        if(nossub(k).eq.iele(kel,j))goto 20
30                              continue
```

```
c                               ainda nao foi, vamos considerar
                                nnossub=nnossub+1
                                nossub(nnossub)=iele(kel,j)
                        endif
20              continue
10      continue
c       determinados os nos vamos ver qual a maxima cota de agua
c       ate ao momento vendo qual o no submerso de maior cota
        cotmax=0
        do 40 i=1,nnossub
                if(xy(nossub(i),2).gt.cotmax)cotmax=xy(nossub(i),2)
40      continue
c       calcular a altura de agua pela diferença entre a cota
c       maxima e a cota do ponto
        write(*,*)'Cota maxima ',cotmax
        do 50 i=1,nnossub
                alturas(i)=cotmax-xy(nossub(i),2)
                write(*,*)i,alturas(i)
50      continue
        return
        end
c-------------------------------------------------------
c.....
c
c  Emulação para ECRÃ de :
c
c       plots,plot,pltend,number,msymbol,where
c
c.....
        subroutine plots(ilixo,sx,sy,name)
        common/xx898/xp89,yp89
        character*(*) name
        character*20 filep
        open(27,file='out.gra',status='unknown')
        close(27,status='delete')
        open(27,file='out.gra',status='new')
        write(27,*)sx,sy
        if(ilixo.eq.1)then
                call plot(0.,0.,3)
                call plot(sx,0.,2)
                call plot(sx,sy,2)
                call plot(0.,sy,2)
                call plot(0.,0.,2)
        endif
        return
        end

c-----------------------------------------
        subroutine plot(a,b,is)
        common/xx898/xp89,yp89
        if(is.eq.3)then
                xp89=a
```

```
            yp89=b
            return
        else
            write(27,1)'linha'
            write(27,2)xp89,yp89
            write(27,2)a,b
            xp89=a
            yp89=b
        endif
1       format(a)
2       format(f10.4,' ',f10.4)
3       format(f10.4)
4       format(i3)
        return
        end
c---------------------------------
        subroutine number(x,y,h,f,rot,ndec)
        common/xx898/xp89,yp89
        character*7 frt
        if(ndec.lt.0)then
            idec=0
        else
            idec=ndec
        endif
        write(frt,'(a5,i1,a1)')'(F10.',idec,')'
        write(27,1)'texto'
        write(27,frt)f
        write(27,2)x,y
        write(27,3)h
        write(27,3)rot
        xp89=x+10*h*cos(rot/57.296)
        yp89=y+10*h*sin(rot/57.296)
        return
1       format(a)
2       format(f10.4,',',f10.4)
3       format(f10.4)
        end
c-----------------------------------------------
        subroutine msymbol(x,y,h,text,rot,n)
        common/xx898/xp89,yp89
        character*(*) text
        character*60 text1
        n1=len(text)
        text1=text
        write(27,1)'texto'
        write(27,1)text
        write(27,2)x,y
        write(27,3)h
        write(27,3)rot
        xp89=x+n1*h*cos(rot/57.296)
        yp89=y+n1*h*sin(rot/57.296)
        return
```

```
1       format(a)
2       format(f10.4,',',f10.4)
3       format(f10.4)
4       format(i3)
        end
c.....
        subroutine where(x,y)
        common/xx898/xp89,yp89
        x=xp89
        y=yp89
        return
        end
c.....
        subroutine pltend
        character*80 linha,linha1
        close(27)
        call system('graphic')
        return
        end
```

```
sub leter(a$,s,a,set)
        l=len(a$)
        for i=1 to l step 3
                        n=val(mid$(a$,i+2,1))*s
                        n=int(n):if n<1 then n=1
                        set=1
                rem print s,n,mid$(a$,i,1)
                b$=mid$(a$,i,2)+str$(n)
                rem print b$            :input x$
                draw "TA"+str$(a)+b$
        next i
end sub

sub strip(a$)
        l=len(a$)
        if left$(a$,1)=" " then
                a$=right$(a$,l-1)
                call strip(a$)
        end if
end sub

sub rstrip(a$)
        l=len(a$)
        if right$(a$,1)=" " then
                a$=left$(a$,l-1)
                call rstrip(a$)
        end if
end sub

sub putlet(a$,px,py,scale,rot,set)
        pset (px,349-py)
        call leter(a$,scale,rot,set)
end sub

sub ascii(a$,s)
        s=asc(a$)
end sub

sub symbol(linha$,x,y,size,rot)
        shared l$()
        l=len(linha$)
        xf=5*l*cos(rot*3.14/180)+x
        yf=5*l*sin(rot*3.14/180)+y
        m=(yf-y)/(xf-x)
        set=0
        for i=1 to l
                letra$=mid$(linha$,i,1)
                call ascii(letra$,code)
                xpos=x+(i-1)*6*size
```

```
            if rot<>90 then
                ypos=y+(i-1)*6*size*m
            else
                ypos=y+(i-1)*5*size
                xpos=x
            end if
            call putlet(l$(code),xpos,ypos,size,rot,set)
        next i
end sub

sub escala(s,d,max)
        s=s/d*max
end sub


sub linha(x1,y1,x2,y2)
        line(x1,349-y1)-(x2,349-y2)
end sub

sub linhan(x1,y1,x2,y2,co,n)
        for i=0 to n-1
            call linha(x1,y1-i,x2,y2-i)
        next i
end sub

dim l$(126)


rem letras
data " u5 e1 r3 f1 d5 u3 l5"              :rem A
data " u6 r4 f1 d1 g1 l4 r4 f1 d1 g1 l4"   :rem B
data "Bu6Br5 l4 g1 d4 f1 r4"              :rem C
data " u6 r4 f1 d4 g1 l4"                 :rem D
data " u6 r4Bd3Bl1 l3Bd3 r4"              :rem E
data " u6 r4Bd3Bl1 l3"                    :rem F
data "Br1 r3 e1 u2 l2Bu3Br2 l4 g1 d4 f1"   :rem G
data " u6Br5 d6 u3 l5"                    :rem H
data " r4 l2 u6 l2 r4"                    :rem I
data "Bu2 d1 f1 r2 e1 u5 r1 l4"           :rem J
data " u6 d3Ne3 f3"                       :rem K
data "Nu6 r4"                             :rem L
data " u6 f3 e3 d6"                       :rem M
data " u6 f5 u5"                          :rem N
data "Br1 r3 e1 u4 h1 l3 g1 d4 f1"        :rem O
data " u6 r3 f1 d2 g1 l3"                 :rem P
data "Br1 r3 e1 u4 h1 l3 g1 d4 f1 r3Bu1 f1" :rem Q
data " u6 r3 f1 d1 g1 l3 f3"              :rem R
data " r3 e1 u1 h1 l2 h1 u1 e1 r3"        :rem S
data "Br2 u6 l2 r4"                       :rem T
data "Bu6 d5 f1 r2 e1 u5"                 :rem U
data "Bu6 d4 f2 e2 u4"                    :rem V
data "Bu6 d4 f2 e1 f1 e2 u4"              :rem W
```

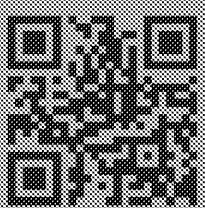
```
data " e2 u2 e2Bl4 f2 d2 f2"              :rem X
data "Br2 u4 e2BL4 F2"                    :rem Y
data "Bu6 r4 d1 g4 d1 r4"                 :rem Z
rem espaco
data ""
rem numeros
data "Br1 r2 e1 u4 h1 l2 g1 d4 f1"                    :rem 0
data " r2 u6 g1be1bd6 r2"                             :rem 1
data "Bu5 e1 r2 f1 d1 g1 l2 g1 d2 r4"                 :rem 2
data "Bu5 e1 r2 f1 d1 g1 l2 r2 f1 d1 g1 l2 h1"        :rem 3
data "Br3 u6 g4 r5"                          :rem 4
data "bu6br3 l3 d3 r3 f1 d1 g1 l3"                    :rem 5
data "bu6br3 l1 g1 g1 d3 f1 r2 e1 u1 h1 l3"           :rem 6
data "Bu5 u1 r4 d1 g2 d3"                             :rem 7
data "Br1 r2 e1 u1 h1 l2 h1 u1 e1 r2 f1 d1 g1 l2 g1 d1 f1" :rem 8
data "bu6br3 l2 g1 d1 f1 r2 e1 u1 h1 f1 d4 g1 l2 h1"       :rem 9
rem .
data " d1"


restore
rem ler definição de letras
rem maiusculas
for i=65 to 90:read l$(i):next i
rem letras minusculas
restore
for i=97 to 122:read l$(i):next i
read l$(27)
rem numeros
for i=48 to 57:read l$(i):next i
rem .
read l$(46)
screen 9
open "out.gra" for input as #1
rem ler dimensao
input #1,dx,dy
sx=639/dx
sy=349/dy
s=sx
if sy<s then s=sy

while eof(1)=0
        input#1,coma$
        if coma$="linha" then
                input#1,x1,y1,x2,y2
                call linha(x1*s,y1*s,x2*s,y2*s)
        end if
        if coma$="texto" then
                line input#1,a$
                input#1,x1,y1,si,rot
                call strip(a$)
                call rstrip(a$)
```

```
                call symbol(a$,x1*s,y1*s,si*s/4.5,rot)
        end if
wend
close#1
o$=input$(1)

screen 0
end
```



www.lnec.pt

AV DO BRASIL 101 • 1700-066 LISBOA • PORTUGAL
tel. (+351) 21 844 30 00
lnec@lnec.pt www.lnec.pt