SOFTWARE NO ENSINO E NO PROJECTO DE ESTRUTURAS

Álvaro Azevedo[1]; Joaquim Barros[2]; José Sena Cruz[3]; Ventura Gouveia[4]

[1]Prof. Auxiliar, Univ. Porto, Fac. Engª, Dep. Engª Civil, alvaro@fe.up.pt
[2]Prof. Auxiliar, Univ. Minho, Esc. Engª, Dep. Engª Civil, barros@civil.uminho.pt
[3]Assistente, Univ. Minho, Esc. Engª, Dep. Engª Civil, jsena@civil.uminho.pt
[4]Assistente, Inst. Sup. Polit. Viseu, Esc. Sup. Tec., Dep. Engª Civil, ventura@dcivil.estv.ipv.pt

RESUMO

No presente trabalho descrevem-se os aspectos essenciais do software desenvolvido para a análise linear e não linear de estruturas, baseado nas técnicas do MEF e concebido de forma a ser simples e eficiente a implementação de novos tipos de elementos finitos e de novos modelos constitutivos. A estrutura da informação que serve de base a uma análise linear ou não linear é descrita neste artigo, bem como as características de algumas técnicas computacionais. No final são apresentados alguns exemplos ilustrativos das ferramentas desenvolvidas.

1. INTRODUÇÃO

O método dos elementos finitos (MEF) é uma ferramenta de grande utilidade na interpretação e compreensão de muitos fenómenos de engenharia (Zienkiewicz e Taylor, 1989). Para tal é necessário dispor, não somente de algoritmos de análise linear e não linear material e geométrica robustos e fiáveis, como também de software de pós-processamento que permita representar graficamente os aspectos mais relevantes das análises efectuadas (Azevedo *et al.*, 1997).

Os códigos existentes têm-se revelado demasiado complexos e monolíticos, dificultando o ensaio de novas ideias, bem como a modificação de alguns dos seus componentes. Estas dificuldades motivaram o desenvolvimento, de raiz, de software baseado em módulos que incluem todo o código necessário ao processamento do correspondente bloco de informação. Esta arquitectura torna simples e flexível a introdução de novos elementos finitos e novos modelos de análise não linear de estruturas. O software desenvolvido foi utilizado na simulação do comportamento de estruturas porticadas de betão armado e de lajes de betão reforçado com fibras de aço e apoiadas em solo.

2. CONCEPÇÃO DO CÓDIGO COMPUTACIONAL

O código computacional encontra-se dividido em dois grandes grupos de módulos: os que necessitam de interface gráfica (e.g., pré/pós-processamento) e os que envolvem apenas cálculo numérico. Uma das plataformas mais vantajosas para o desenvolvimento de software eficiente e com interface gráfica amigável é a que se baseia nas *Microsoft Foundation Classes* (MFC), que se encontram implementadas na linguagem de programação Visual C++ (Kruglinski *et al.*, 1998). Por este motivo, nos módulos gráficos foi utilizada apenas esta linguagem e a referida biblioteca de classes. Ao utilizar uma única plataforma para toda a

aplicação evitam-se as dificuldades, as limitações e as ineficiências associadas à combinação de módulos escritos em distintas linguagens.

No caso dos módulos que envolvem exclusivamente cálculo numérico e leitura/escrita em disco, optou-se pela linguagem de programação ANSI-C. Deste modo são evitadas as complexidades associadas ao C++, diminuindo significativamente a duração da fase de adaptação, sempre que é adicionado um novo elemento à equipa de programadores. Na estruturação do código escrito em ANSI-C foram utilizados muitos conceitos inspirados na linguagem C++.

2.1 Biblioteca de funções nucleares (*cutil*)

O desenvolvimento de software que envolve essencialmente cálculos numéricos (e.g., software relativo ao método dos elementos finitos) requer a repetida realização de operações elementares, de que são exemplo as operações de álgebra linear com base em vectores e matrizes, a alocação de memória para estruturas de dados, a leitura em ficheiro seguida de uma validação exaustiva, a interacção com o utilizador em modo de consola de texto, etc. Em correspondência com estas tarefas desenvolveram-se funções genéricas, que foram em seguida englobadas numa biblioteca designada *cutil*. Uma vez que esta biblioteca contém já um número elevado de funções (cerca de 170), seria impraticável descrevê-las aqui exaustivamente, sendo aconselhada a consulta de (Azevedo, 1996). Apresenta-se em seguida apenas um fragmento de código escrito em ANSI-C (Kernighan e Ritchie, 1988), que ilustra o modo como se utilizam algumas das funções da biblioteca *cutil* (ver a Tabela 1).

Tabela 1 - Cálculo dos valores e vectores próprios de uma matriz recorrendo à biblioteca *cutil*.

```
...
double** a;
n = get_pos_int_1(f, &i, &l);
dimed2(&a, n, n);
import_mat(a, n, n, f, "Table", "NoNumbers", "Semicolons", &i);
mat_eigen_jacobi(a, n, d, q)
...
freed2(a);
```

As linhas de código indicadas na Tabela 1 efectuam as seguintes operações: declaração da matriz *a* como sendo um apontador para apontador para *double*, leitura no ficheiro *f* de um inteiro *n* que indica o número de linhas/colunas da matriz, alocação de memória para a matriz (*n* x *n*), leitura com validação da matriz no ficheiro *f*, cálculo dos valores e vectores próprios da matriz pelo método de Jacobi e libertação da memória correspondente à matriz *a*. Com este exemplo pretende-se evidenciar a grande simplicidade de utilização das funções da biblioteca *cutil.* Quando são utilizadas em grandes projectos de software, estas vantagens tornam-se ainda mais relevantes, não sendo notória qualquer degradação da performance do programa.

2.2 Organização do programa de cálculo

O programa de cálculo automático baseado no método dos elementos finitos, designado FEMIX, está estruturado nos seguintes módulos:

prefemix - leitura/validação de um ficheiro de texto, com extensão *_gl.dat*, contendo todos os dados, seguida da gravação de ficheiros binários com a informação já validada;

femix - leitura dos referidos ficheiros binários, cálculo dos resultados com base no método dos elementos finitos e gravação de ficheiros binários contendo todos os resultados desta fase, que pode ser linear ou não linear;

posfemix - geração de diversos tipos de resultados com base na informação proveniente dos módulos anteriores.

Para facilitar o acesso a este programa por parte de utilizadores menos experientes, pretende-se vir a esconder estas três fases por trás de uma interface gráfica intuitiva. De qualquer forma, será sempre possível que utilizadores mais experientes possam agrupar em macros um número arbitrário de execuções do programa para fazer, por exemplo, estudos paramétricos.

2.3 Tipos de elementos

A principal motivação para a total reescrita do programa FEMIX foi a de passar a ser possível incluir diferentes tipos de elementos na mesma análise, bem como distintos modelos de não linearidade material. Com este fim em vista, o ficheiro de dados, com a extensão *_gl.dat*, foi concebido de forma a possibilitar a descrição de problemas contemplando todos os tipos de combinações de elementos, condições fronteira, cargas e modelos de não linearidade. O ficheiro de dados é constituído por um conjunto de blocos, em que a sua ordem não é pré-definida, sendo apenas necessário que os blocos que definem propriedades sejam colocados antes de essas propriedades serem referidas. Associado a cada bloco de dados existe um *struct* que engloba toda a informação relevante. De um modo uniforme e semelhante ao que é habitual na programação orientada por objectos, foram desenvolvidas funções destinadas à manipulação dos dados de cada *struct*.

Na presente fase de desenvolvimento do programa encontram-se já implementados os seguintes elementos: sólidos tridimensionais, casca de Mindlin e de Ahmad, laje, estado plano de tensão e diversos elementos de barra com ou sem deformação por esforço transverso. Foram também implementados elementos isoparamétricos de interface de linha 2D e de superfície 3D. Na formulação admitiu-se que os elementos de interface têm espessura nula.

Todos estes elementos são definidos no espaço tridimensional, sendo livre a sua combinação na mesma análise, com excepção de certos tipos de situações fisicamente incompatíveis.

O bloco que define os nós de cada elemento está descrito na Tabela 2. Para cada elemento são especificados os nós, independentemente das propriedades do elemento.

Tabela 2 - Exemplificação do bloco da conectividade dos nós de cada elemento.

```
<ELEMENT_NODES>
## Nodes defining the elements
   COUNT = 4 ; # N. of elements

## Content of each column:
#  A -> Counter; # B -> N. of nodes of the element; # C -> Nodes of the element
# A  B    C
  1  3    5 9 13 ;
  2  2   22 33 ;
  3  4   22 33 47 49 ;
  4 20   21 22 23 24 41 42 43 44 51 52 53 54 55 10 11 12 13 14 15 16 ;

</ELEMENT_NODES>
```

O bloco das propriedades dos elementos, apresentado na Tabela 3, identifica para cada elemento ou conjunto de elementos, o tipo de elemento finito, as características mecânicas e geométricas e o tipo de integração numérica utilizada no cálculo da matriz de rigidez desses elementos. Se algum tipo de elemento não necessitar de uma dada propriedade, deve-se colocar "_NONE" na coluna correspondente. Se o utilizador não pretender especificar uma determinada propriedade que é necessária à definição do elemento, pode escrever "_DEFAULT" para que sejam considerados os correspondentes valores por defeito.

Tabela 3 - Exemplificação do bloco das propriedades dos elementos.

```
<ELEMENT_PROPERTIES>
## Specification of the element properties
   COUNT = 3 ; # N. of specifications

# Content of each column:
# A -> Counter; # B -> Group name; # C -> Phase (or phase range); # D -> Element (or
  element range); # E -> Element type; # F -> Material type; # G -> Material name; # H ->
  Geometry type; # I -> Geometry name; # J -> Integration type (stiffness matrix); # K ->
  Integration name (stiffness matrix)
# A B        C     D E                    F         G    H           I         J      K
  1 FL_1 [2-4] [3-6] _FRAME_3D            _LIN_ISO C25  _CS_3D     30x50     _NONE  _NONE ;
  2 FL_1 [5-6]    19 _TIMOSHENKO_3D       _FIBROUS BEAM _NONE      _NONE     _GLEG  INT_BEAM ;
  3 FL_1 [7-9]    22 _MINDLIN_SHELL_QUAD  _LIN_ISO C25  _THICKNESS THICK_50 _GLEG  _DEFAULT ;
```

2.4 Técnicas de integração numérica

Para a avaliação dos integrais resultantes da formulação matemática dos modelos utilizados no código computacional, recorre-se à integração numérica. Dentro de cada elemento, a posição dos pontos onde se avaliam esses integrais (pontos de integração) varia conforme a técnica de integração utilizada. No presente código computacional estão disponíveis as seguintes regras de integração: Gauss-Legendre, Gauss-Lobatto e Newton-Cotes (Schellekens, 1990). Um exemplo do bloco de integração numérica encontra-se na Tabela 4.

No caso de matrizes de rigidez compostas por submatrizes associadas a distintos modos de deformação (e.g., membrana, flexão e corte, nas cascas) o cálculo de cada uma destas submatrizes pode ser efectuado com um número de pontos de integração que se julgue mais apropriado. Esta estratégia permite utilizar a integração completa, selectiva ou reduzida, de forma a evitar fenómenos indesejados, de que o bloqueamento é exemplo (Oñate, 1995).

Tabela 4 - Exemplificação do bloco das técnicas de integração numérica.

```
<NUMERICAL_INTEGRATION>
## Keywords: _GLEG, _GLOB or _NCOTES
## Numerical integration definition
   COUNT = 2 ; # N. of layouts of integration points

## Content of each column:
# A -> Counter; # B -> Name of the layout; # C -> N. of keywords
  defining the layout; # D -> Keywords defining the type of term and
  the n. of points in each direction
# Valid syntax for D is the following:
#     _i_Sj_k
#     i: type of terms
#       G -> General use (stiffness matrix, springs and load vector)
#       M -> Membrane terms (stiffness matrix)
#       S -> Shear terms (stiffness matrix)
#       T -> Torsional terms (stiffness matrix)
#       B -> Bending terms (stiffness matrix)
#     j: direction
#       1 -> direction s1
#       2 -> direction s2
#       3 -> direction s3
#     k: n. of points in the specified direction
#     Note: k must be in the range [1,10]
#     Examples: _G_S1_5, _M_S2_1, _S_S3_2, _T_S1_7, _B_S2_10
#
# A B              C D
  1 INT_PS          2 _M_S1_2 _M_S2_1 ;
  2 INT_SHELL       6 _M_S1_2 _M_S2_2 _S_S1_2 _S_S2_2 _B_S1_3 _B_S2_3 ;
</NUMERICAL_INTEGRATION>
```

2.5 Apoios elásticos

O código computacional permite simular apoios elásticos localizados nos nós da malha, num bordo ou na superfície de um elemento. Os apoios elásticos podem ter comportamento linear

ou não linear, apenas funcionar quando forem solicitados num determinado sentido, terem diversas direcções e serem do tipo translação ou rotação. Na Tabela 5 apresenta-se um exemplo de um bloco de dados correspondente a apoios elásticos distribuídos na área do elemento.

Tabela 5 - Exemplificação do bloco de apoios elásticos por unidade de superfície.

```
<SURFACE_SPRINGS>
## Surface with springs
   COUNT = 4 ; # N. of surface springs

## Content of each column:
#  A -> Counter (or counter range); # B -> Group name; # C -> Phase (or phase range);
   # D -> Element number (or element number range); # E -> Face number; # F -> Direction
   vector: - Global: _XG1, _XG2 or _XG3; - Solid/Shell: _L1, _L2 or _N; - Specified:
   vector from the <AUXILIARY_VECTORS> block; #  G -> Spring vector orientation (+ or -)
   (Note: positive means from structure to ground); # H -> Dof type (_TRANS-Translational;
   _ROT – Rotational); # I -> Integration name (in the Gauss Legendre table); #  J -> N. of
   nodes of the face of the element; # K -> Name(s) of the spring(s) in the
   <SPRING_PROPERTIES> block; #  Note: when J is equal to 1 the spring is constant;
#  Direction vector keywords:
#     _L1 - tangential to the face and following the l1 local axis
#     _L2 - tangential to the face and following the l2 local axis
#     _N  - normal to the face
#
#    A B          C       D E F    G  H       I       J  K
     1 FL_1 [1-100]      25 6 _L1   +  _TRANS  INT_PS  4  WEAK STRONG WEAK WEAK ; # Solid
 [2-4] BASE [25-31] [22-24] 1 VECT1 -  _ROT    INT_PS  1  WEAK ; # Shell
</SURFACE_SPRINGS>
```

Criou-se um bloco comum a todos os tipos de apoios elásticos de forma a definir algumas propriedades do apoio (ver Tabela 6).

Tabela 6 - Exemplificação do bloco das propriedades dos apoios elásticos.

```
<SPRING_PROPERTIES>
## Properties of the springs that link the structure with the ground
   COUNT = 2 ; # N. of types of springs

## Content of each column:
#  A -> Counter; # B -> Name of the spring; #  C -> Stiffness of the spring
#  A   B           C
   1   STRONG    2.5 ;
   2   WEAK      3.5 ;
</SPRING_PROPERTIES>
```

2.6 Assemblagem e introdução de deslocamentos prescritos

A assemblagem da matriz de rigidez global é efectuada com base nas seguintes pressupostos: calcular a matriz de rigidez de cada elemento uma só vez e não a guardar, nem na memória, nem em disco; ter em memória apenas a matriz de rigidez do elemento corrente; espalhar a matriz de rigidez do elemento corrente por todas as linhas da matriz de rigidez global para as quais ele contribui; alocar espaço suficiente para que cada linha da matriz de rigidez global possa receber termos potencialmente não nulos sem existirem constantes realocações de memória; só alocar espaço relativo a um determinado grau de liberdade quando aparecer o primeiro elemento que para ele contribui; quando se conclui que uma linha da matriz de rigidez global já não vai receber mais nenhuma contribuição é efectuada a remoção de todos os termos nulos, seguida de uma realocação de memória. Com este procedimento e num computador com 512 MBytes de RAM foi possível efectuar, em memória, uma análise com um milhão de elementos finitos de quatro nós e dois graus de liberdade por nó.

A introdução dos deslocamentos prescritos é efectuada por substituição, i.e., sem ser necessário adicionar termos de valor elevado à diagonal da matriz de rigidez. Desta forma,

numa estrutura com um número elevado de graus de liberdade prescritos estes não contribuem para a dimensão do sistema de equações que é necessário resolver.

2.7 Ligações internas entre graus de liberdade

De forma a possibilitar a ligação de graus de liberdade entre pares de pontos de uma estrutura (e.g., simulação de uma rótula num pórtico plano), existe um bloco que identifica os nós de ligação e os graus de liberdade a ligar, ver Tabela 7.

Tabela 7 - Exemplificação do bloco das ligações internas.

```
<INTERNAL_POINT_LINKS>
## Points with linked degrees of freedom
   COUNT = 11 ; # N. of pairs of points with linked degrees of freedom

## Content of each column:
#  A -> Counter (or counter range); # B -> Group name; # C -> Phase (or
   phase range); # D -> Point number - M (Master) (or point number
   range); # E -> Point number - S (Slave) (or point number range);
   # F -> N. of linked degrees of freedom in the current pair of points;
   # G -> Linked degrees of freedom: - Available keywords: _D1, _D2,_D3,
   _R1, _R2 or _R3
#          A  B              C         D         E   F    G
           1  FLOOR_1  [1-100]       34        30   3    _D1 _D3 _R1 ;
      [2-11]  BASEMENT [22-33]   [1-10]  [21-30]   2    _D2 _R3 ;
</INTERNAL_POINT_LINKS>
```

2.8 Resolução do sistema de equações

Após as operações referidas nas secções anteriores, é efectuada a resolução do sistema de equações lineares em memória, i.e., sem efectuar explicitamente trocas de informação entre a memória RAM e o disco. Estas trocas de informação poderão eventualmente ser efectuadas pelo sistema operativo caso a memória RAM se esgote. Uma vez que se dispõe de dois métodos de resolução do sistema de equações lineares, aconselha-se o seguinte critério de selecção do método: em sistemas bem ou mal condicionados até cerca de 10000 equações sugere-se a utilização do método de eliminação de Gauss; em sistemas com um número mais elevado de equações e que não sejam mal condicionados recomenda-se o método dos gradientes conjugados (Azevedo e Barros, 1990).

No caso da análise não linear é utilizado o método de Newton-Raphson, sendo possível resolver o sistema de equações lineares correspondente a cada iteração pelo método de eliminação de Gauss ou pelo método dos gradientes conjugados.

2.9 Combinações de acções

A matriz das combinações é definida no ficheiro de dados, uma vez que tem de ser conhecida antes de iniciado o processo incremental/iterativo numa eventual análise não linear. Na ausência desta informação é gerada uma matriz identidade. Na análise não linear material e geométrica a evolução do carregamento é simulada por intermédio das combinações, sendo um incremento de carga o resultado da variação do carregamento entre duas combinações consecutivas. No caso da análise dinâmica (linear ou não linear), para cada combinação será também necessário fornecer o número de intervalos de tempo e correspondente duração.

3. MODELAÇÃO DO COMPORTAMENTO NÃO LINEAR DE ESTRUTURAS

Na concepção do presente código computacional teve-se em conta o facto de se pretender mantê-lo aberto à implementação, simples e eficaz, de novos modelos de análise não linear material. Para tal, o programador deve elaborar um bloco de dados que inclua toda a

informação relativa ao modelo a implementar e escrever o código dos módulos que processam essa informação.

A informação que permite ter em conta a evolução do comportamento da estrutura durante o seu carregamento fica guardada em vectores de *struct*'s, cujo conteúdo depende do modelo em causa. Por exemplo, no caso de estruturas laminares, em que cada elemento finito é discretizado num determinado número de camadas e possui um conjunto de pontos de amostragem, o acesso à informação requer a definição de três índices, *iElement, jSampPoint* e *kLayer*, em que *kLayer* é o número da camada, no ponto de amostragem *jSampPoint*, situado no elemento *iElement*. Desta forma, quando uma estrutura é discretizada com distintos tipos de elementos finitos e se utilizam diversos modelos de análise não linear, o formato das bases de dados que armazenam a informação colectada durante o processo incremental, adapta-se às especificidades dos elementos e dos modelos utilizados. A informação correspondente a cada incremento de carga é mantida em memória, sendo descarregada para disco no fim de cada incremento de carga convergido.

3.1 Estruturas reticuladas

Na simulação do comportamento não linear de estruturas reticuladas de betão armado a utilização de um modelo de fibras tem algumas vantagens, uma vez que permite efectuar uma discretização pormenorizada da secção transversal ao nível de cada barra (Ventura Gouveia *et al.*, 2000). Na Figura 1 representa-se de forma esquemática a discretização de um elemento de betão armado segundo o modelo de fibras desenvolvido (Ventura Gouveia, 2000). O elemento de betão armado é discretizado num elemento de viga de Timoshenko 3D de dois ou três nós. Cada nó tem seis graus de liberdade. Cada elemento finito 3D é dividido em fibras. Ao nível da secção, essas fibras são representadas por elementos finitos planos. As deformações e as tensões são avaliadas nos pontos de amostragem desses elementos planos, com base na lei constitutiva do material correspondente. A armadura longitudinal é também tida em conta, sendo discretizada por elementos finitos de barra 3D (fibras) ao nível do elemento. A posição das armaduras fica definida ao nível da secção transversal e o seu comportamento é governado por leis constitutivas não lineares.

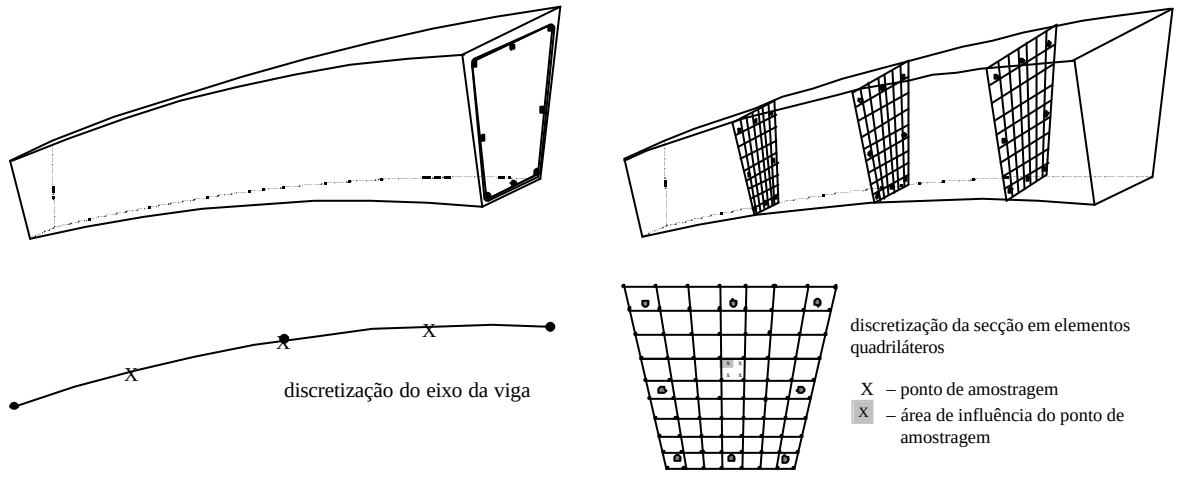

Figura 1 - Discretização de um elemento de betão armado segundo o modelo de fibras desenvolvido.

3.2 Estruturas laminares

Na Tabela 8 inclui-se a informação (parcial, por limitação de espaço) relativa a um modelo de análise não linear de estruturas laminares de betão armado, em que o betão não fendilhado é tratado segundo a teoria da elasto-plasticidade e o betão fendilhado é simulado por intermédio da mecânica da fractura não linear (Barros, 1995).

Tabela 8 - Bloco de dados (parcial) para o modelo de análise não linear NLMM200205.

```
<NLMM200205>
## Keyword: _NLMM200205
## Properties defining the material constitutive law:
##  CEB-FIP 1993 law for compression behavior
##  Linear-parabola diagram for tensile up-to-peak behaviour
##  Trilinear diagram for tensile post-peak behavior
##  Kinematic hardening rule; associated flow rule; strain hardening; Owen yield criteria
##  Multi-fixed smeared crack model

   COUNT = 1 ; # N. of materials modeled by the NLMM200205 law

## Content of each column:
#  A -> Counter
#  B -> Name of the constitutive law

... etc. ...

#  W -> Threshold angle (zero for the first criteria of the new crack opening);
#  X -> Parameter to define the mode I fracture energy available for the new crack (zero
for the initial Gf)
#  A  B                 ...     W      X
   1  C25_30_UTSO       ...     30.0   2 ;

</NLMM200205>
```

As propriedades dos tipos de camadas existentes numa determinada estrutura laminar são definidas num bloco de dados exemplificado na Tabela 9. Assim, na estrutura em causa existem dois tipos de camadas. O primeiro é designado PC_CONST, tem comportamento material simulado pelo modelo não linear material _NLMM200205, apresenta as propriedades atribuídas ao C25_30_UTSO (ver Tabela 8) e possui propriedades geométricas atribuídas no bloco exemplificado na Tabela 10. Este último define a geometria da camada. O segundo tipo de camada difere do anterior apenas nas propriedades geométricas.

Tabela 9 - Exemplificação do bloco de dados relativos aos tipos de camadas

```
<LAYER_PROPERTIES>
## Layer properties
   COUNT = 2 ; # N. of layer properties

## Content of each column:
#  A -> Counter
#  B -> Layer name
#  C -> Material type
#  D -> Material name
#  E -> Geometry pattern name
#  A  B         C           D            E
   1  PC_CONST  _NLMM200205  C25_30_UTSO  THICK_CONST ;
   2  PC_VARIA  _NLMM200205  C25_30_UTSO  THICK_VARIA ;
</LAYER_PROPERTIES>
```

Tabela 10 - Exemplificação do bloco de dados relativo à geometria dos elementos e camadas.

```
<GEOMETRY_PATTERNS>
## Keyword: _PATTERN
## Geometry patterns of several types of properties
   COUNT = 2 ; # N. of geometry patterns

## Content of each column:
#  A -> Counter
#  B -> Pattern name
#  C -> Type of geometry description
#       - Available keywords: _CS_AREA, _CS_2D, _CS_3D or _THICKNESS
#  D -> N. of nodes of the pattern
#  E -> Names of geometry descriptions
#  A  B           C           D  E
   1  THICK_CONST  _THICKNESS  1  TH_20 ;
   2  THICK_VARIA  _THICKNESS  8  TH_40  TH_30  TH_20  TH_20  TH_20  TH_30  TH_40 TH_40 ;
</GEOMETRY_PATTERNS
```

Na Tabela 11 apresenta-se o bloco de dados que define os distintos grupos de camadas existentes numa estrutura. Neste caso, existem dois agrupamentos de camadas, o primeiro designado por LAYER_PAT_CONST, composto por 5 camadas do tipo PC_CONST (ver Tabela 9). O segundo arranjo de camadas designa-se por LAYER_PAT_VARIA, sendo constituído por um grupo de 7 camadas do tipo PC_VARIA (ver Tabela 9).

Tabela 11 - Exemplificação do bloco de dados relativos aos tipos de agrupamentos de camadas

```
<LAYER_PATTERNS>
## Keyword: _PATTERN
## Layer patterns
   COUNT = 2 ; # N. of layer patterns

## Content of each column:
#  A -> Counter; # B -> Pattern name; # C -> N. of groups of layers
#  D -> Name of the layer properties in the <LAYER_PROPERTIES> block
#  E -> N. of layers composing a group
#
#  A  B                C  D          E
   1  LAYER_PAT_CONST  1  PC_CONST   5 ;
   2  LAYER_PAT_VARIA  1  PC_VARIA   7 ;
</LAYER_PATTERNS>
```

4. VISUALIZAÇÃO

De acordo com o que foi referido na secção 2.2, após o cálculo efectuado com o módulo femix, é possível gerar diversos tipos de informação destinada à visualização gráfica. Esta informação é armazenada em ficheiros compatíveis com o programa DRAWMESH, que consiste num visualizador genérico de modelos tridimensionais. Este programa possui interface MS-Windows, tendo sido desenvolvido em C++ com recurso às Microsoft Foundation Classes. Com o programa DRAWMESH é possível fazer a visualização dos modelos como fio de arame, com eliminação de linhas e superfícies escondidas e com a sobreposição opcional de um campo escalar representado com um código de cores. Na Figura 2 encontra-se o aspecto geral da interface gráfica do programa DRAWMESH, sendo representado o campo escalar correspondente às tensões normais s_x numa passagem superior solicitada pelo veículo tipo excêntrico.

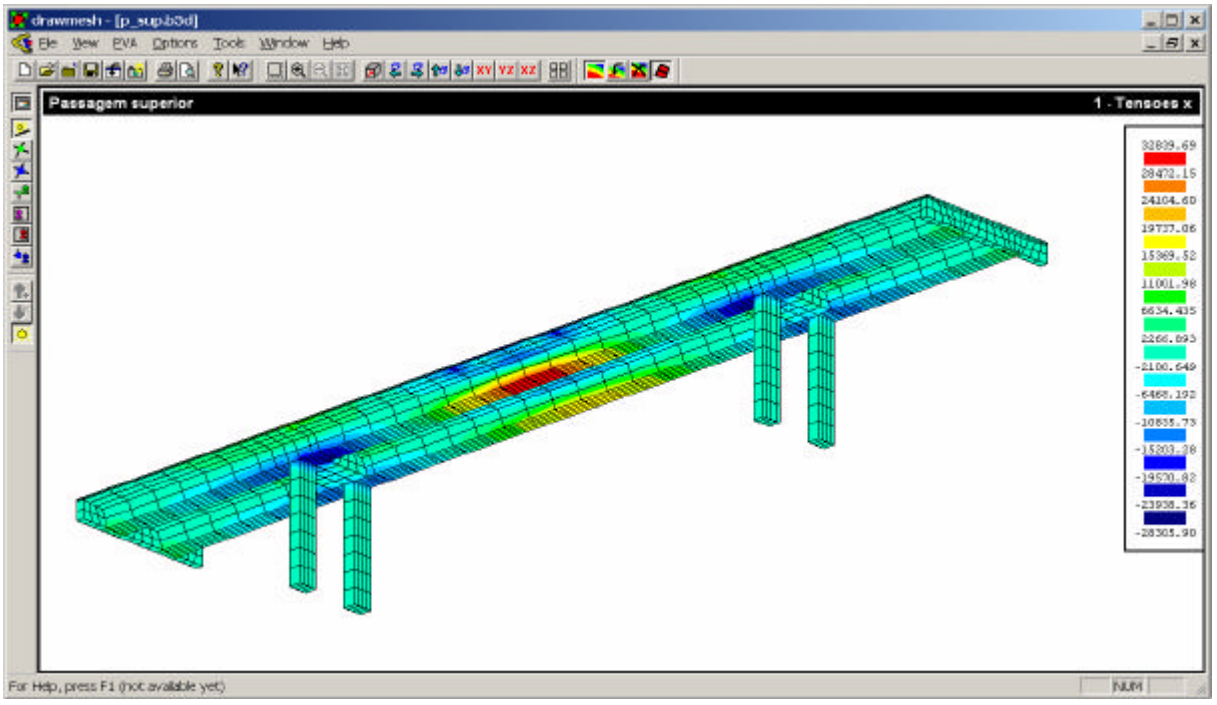

Figura 2 - Visualização do campo de tensões numa passagem superior usando o programa DRAWMESH.

5. EXEMPLOS DE APLICAÇÃO

Nesta secção são referidos dois exemplos que ilustram as características do código computacional, quer em termos de qualidade de simulação do comportamento observado experimentalmente em estruturas, quer quanto às suas capacidades de visualização gráfica dos resultados obtidos. Ambos os exemplos correspondem a análises não lineares de estruturas de betão.

5.1 Vigas de betão armado

Oito vigas de betão armado, de 2.90 m de comprimento, foram ensaiadas em laboratório até à sua rotura (ver Figura 3), tendo sido submetidas a esforços de flexão, corte e torção (Ventura Gouveia, 2000). A parte central da viga, com um comprimento de 2.00 m, apresentava uma secção rectangular oca e as extremidades, com um comprimento de 0.45 m, uma secção rectangular cheia. As vigas foram reforçadas com diferentes percentagens de armadura longitudinal e transversal (por exemplo v1_l10_t75 é a primeira viga cuja armadura longitudinal é constituída por varões em aço de 10 mm de diâmetro e com armadura transversal afastada de 75 mm).

A modelação do comportamento à compressão uniaxial do betão simples foi efectuada segundo a lei proposta pelo código modelo CEB-FIP 1990. Para simular a resistência residual do betão fendilhado foram adoptados modelos de amolecimento ("softening") e de endurecimento ("stiffening") descritos em outro trabalho (Barros, 1995). O comportamento do betão ao corte foi simulado por uma lei tensão tangencial-distorção constituída por ramos lineares. Na modelação do comportamento das armaduras foi utilizada uma lei constitutiva multi-linear. Todos os dados referentes às leis constitutivas estão apresentados noutro trabalho (Ventura Gouveia *et al.*, 2000).

Na Figura 4 são comparados os resultados da simulação numérica com os obtidos experimentalmente. Esta comparação corresponde à relação força-deslocamento vertical (*F-d*), medidos na secção central.

Da análise das curvas *F-d*, verifica-se que a aproximação entre as respostas experimentais e numéricas é bastante boa.

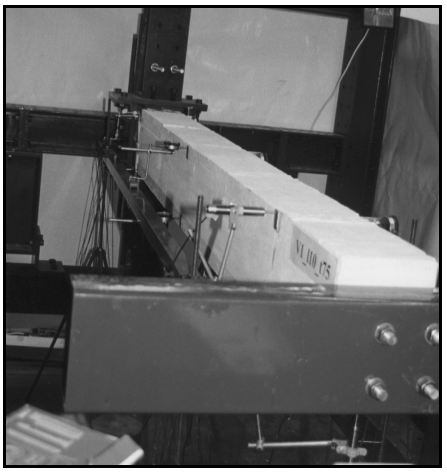

Figura 3 - Sistema de ensaio.

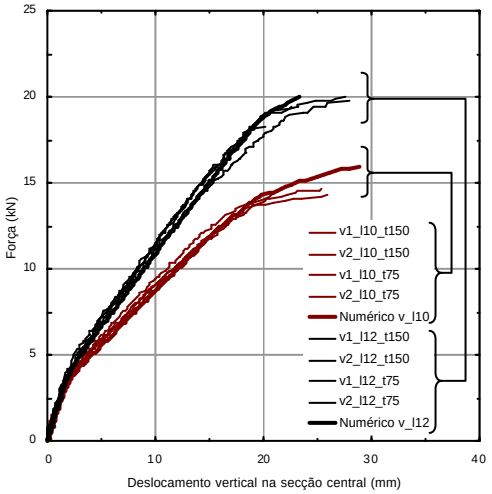

Figura 4 - Diagrama *F-d.*

5.2 Lajes apoiadas em solo

A formulação do modelo numérico utilizado está descrita noutro trabalho (Barros, 1995). Esse modelo foi utilizado para simular lajes de betão reforçado com fibras, apoiadas em solo. As

propriedades dos materiais intervenientes e os valores dos parâmetros do modelo constitutivo encontram-se em (Barros e Figueiras, 2001). Na Figura 5 apresenta-se a relação, experimental e numérica, entre a força aplicada no centro da laje e o correspondente deslocamento. Constata-se que o modelo prevê com rigor suficiente a resposta registada experimentalmente. O padrão de fendilhação para uma carga pontual num canto da laje está representado na Figura 6, bem como a correspondente linha de rotura. A fenda é representada por um rectângulo, cuja largura e direcção reproduzem a abertura da fenda e sua orientação. O diferente estado da fenda (abertura, fecho, reabertura, completamente fechada, completamente aberta) pode ser destinguido atribuindo-se-lhe uma cor em correspondência com o seu estado.

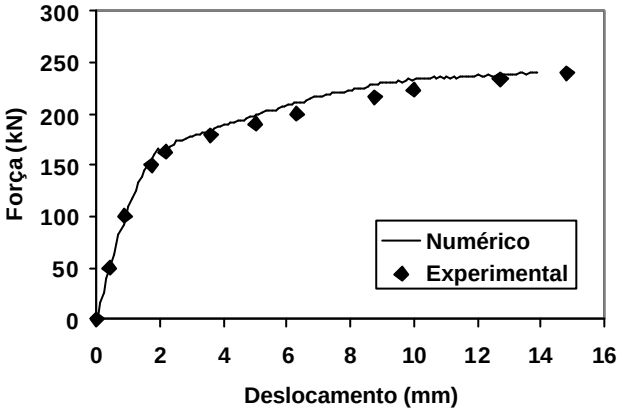

Figura 5 - Relação força-deslocamento no centro da laje.

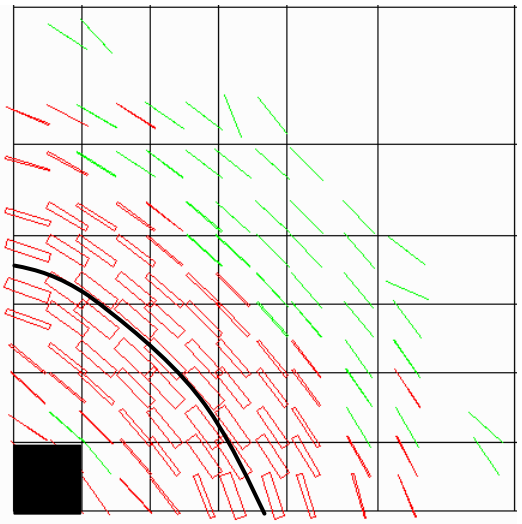

Figura 6 - Padrão de fendilhação na camada superior da laje para carga pontual num canto.

6. CONCLUSÕES

Todas as operações descritas para o caso de uma análise linear são aplicáveis, quer na análise não linear, quer na análise dinâmica, uma vez que este tipo de estudos corresponde a uma sucessão de análises lineares.

Apesar da elevada complexidade dos fenómenos simulados, verifica-se que o software desenvolvido é uma ferramenta muito útil, quer do ponto de vista pedagógico, quer no âmbito científico e profissional.

REFERÊNCIAS

Azevedo, A. F. M.; Barros, J. A. O., Análise comparativa de métodos directos e iterativos na resolução de grandes sistemas de equações lineares, JPEE 90 - 2. Jornadas Portuguesas de Engenharia de Estruturas, LNEC - Lisboa, pp. A.91-A.102, 1990.

Azevedo, A. F. M.; Barros, J. A. O.; Marques, E. R. B.; Branco, P. S. O., Object oriented programming: data preparation and visualization of FEM models, V Encontro Nacional de Mecânica Computacional, Universidade do Minho, Guimarães, 20 a 22 de Outubro de 1997.
URL: http://civil.fe.up.pt/pub/people/alvaro/pdf/1997_Mec_Comp_paper_fem.pdf

Azevedo, A. F. M., Cutil: biblioteca de apoio ao desenvolvimento de software em linguagem C, Versão 3.0, Faculdade de Engenharia da Universidade do Porto, Julho de 1996.
URL: http://civil.fe.up.pt/Software/Cutil_3.0/Cutil.pdf

Barros, J.A.O., Comportamento do betão reforçado com fibras – análise experimental e simulação numérica, Tese de doutoramento, FEUP 1995.

Barros, J. A. O., Figueiras, J. A., Nonlinear analysis of steel fibre reinforced concrete slabs on grade, Computers & Structures, Vol.79, No.1, pp. 97-106, January 2001.

CEB-FIP Model Code 1990, Design Code, Comité Euro-International du Beton, Thomas Telford, 1990.

Kernighan, B. W.; Ritchie, D. M., The C programming language-ANSI C, Second Edition, Prentice-Hall, 1988.

Kruglinski, D. J.; Wingo, S.; Shepherd, G., Programming Visual C++, Microsoft Press, 1998.

Oñate, E., Cálculo de estructuras por el método de elementos finitos - Análisis estático lineal, CIMNE, UPC, 1995.

Schellekens, J. C. J., Interface elements in finite elements analysis, Report 25.2-90 2 17, Delft University of Technology, Delft, 1990.

Ventura Gouveia, A.; Barros, J. A. O.; Azevedo, A. F. M, Modelo de análise não linear material de pórticos de betão armado – calibração das relações constitutivas, Encontro Nacional Betão Estrutural 2000, Porto, 22 a 24 de Novembro de 2000.

Ventura Gouveia, A., Análise experimental e simulação numérica de elementos de barra de pórtico tridimensional de betão armado, Dissertação para obtenção do grau de Mestre em Engenharia Civil, opção de Estruturas, Geotecnia e Fundações, pela Escola de Engenharia da Universidade do Minho, Junho 2000.

Zienkiewicz, O. C.; Taylor, R. L., The Finite Element Method, 4th Edition, McGraw-Hill, 1989.