Mecánica Computacional Vol. XXII
M. B. Rosales, V. H. Cortínez y D. V. Bambill (Editores)
Bahía Blanca, Argentina, Noviembre 2003.

UMA ALTERNATIVA PARA PRÉ E PÓS PROCESSADOR GRÁFICO POR MEIO DE ARQUIVOS TEXTOS PARA PROGRAMAS DE ELEMENTOS FINITOS.

José G. Maluf Soler†, José P. Moitinho de Almeida‡

† Curso de Engenharia Civil
Pontifícia Universidade Católica de Minas Gerais, *Campus* Poços de Caldas
e-mail: jgmsoler@yahoo.com, web page: http://www.pucpcaldas.br

‡ Departamento de Engenharia Civil e Arquitetura
Instituto Superior Técnico, Universidade Técnica de Lisboa.
e-mail: moitinho@civil.ist.utl.pt, web page: http://www.civil.ist.utl.pt

Palavras chave: Elementos finitos, Processador gráfico.

Resumo. *Neste trabalho será apresentada a abordagem utilizada pelos autores para a visualização dos dados e dos resultados associados a uma análise estrutural por meio do método dos elementos finitos. A representação gráfica do modelo é realizada recorrendo a um conjunto de programas de código aberto. Estes programas de visualização utilizam arquivos de dados, gerados pelo utilizador, que descrevem as características do modelo e da solução e adaptam-se facilmente às necessidades da generalidade dos métodos numéricos utilizados em Mecânica Computacional. A experiência obtida confirma que, apesar da tendência para encarar o método numérico como o aspecto fundamental do problema, a visualização pode ser mais do que uma simples ferramenta, ajudando a interpretar e validar os resultados obtidos.*

1 INTRODUÇÃO

A crescente capacidade de cálculo dos meios computacionais disponíveis possibilita o desenvolvimento de técnicas de modelação numérica capazes de aproximar a solução dos problemas em estudo com uma precisão crescente, às quais estão associados valores numéricos em quantidades que impossibilitam a sua interpretação direta. Torna-se por isso imprescindível ter acesso a técnicas que permitam compreender o comportamento do modelo, através de uma análise global dos resultados, bem como a detecção dos pontos críticos no espaço ou no tempo.

A visualização das soluções é a abordagem mais importante utilizada neste contexto, sendo aliás uma opção hoje em dia indispensável em qualquer código comercial. No entanto, quando no âmbito do desenvolvimento de um programa se pretende proceder à visualização dos resultados obtidos há que adaptar a saída de resultados ao formato utilizado pelo programa de visualização disponível. Além disso há que "esperar" que o programa de visualização seja capaz de representar o tipo de informação produzido na forma desejada, o que nem sempre se passa, não sendo possível adaptar programas que são fornecidos unicamente em versão binária.

Do ponto de vista do desenvolvimento da técnica numérica há por isso que encontrar o ponto de equilíbrio entre a capacidade e a adaptabilidade do programa a utilizar, não deixando de ter em conta a necessidade de não se desejar dar maior atenção aos aspectos da visualização do que à modelação numérica.

Apresenta-se neste artigo a experiência dos autores com a utilização do programa *mayavi*[5], o qual fornece uma interface facilmente configurável sobre a biblioteca *vtk*. As capacidades destas ferramentas e a forma como se faz a interligação entre elas serão explicadas, procurando dar ênfase aos aspectos, tanto positivos como negativos, que maior importância tiveram na implementação referida. Ela poderá servir como orientação em projetos semelhantes.

2 A BIBLIOTECA *VTK*

A biblioteca *vtk* (*Visualization ToolKit*)[2] é a base de um sistema, literalmente um conjunto de ferramentas de visualização, desenvolvido em código aberto, para a representação gráfica de modelos tridimensionais, processamento de imagem e visualização.

Os algoritmos de visualização implementados possibilitam diversos tipos de visualização de campos escalares, vetoriais e tensoriais. São também possíveis operações de modelação geométrica que permitem, entre outras, a representação implícita de funções, operações sobre modelos poliédricos, cortes e triangulações de Delaunay. O núcleo do sistema é uma biblioteca de classes programadas em C++, as quais foram desenvolvidas de forma a também ser possível e fácil utilizá-lo a partir de linguagens interpretadas, nomeadamente Tcl/Tk, Java e Python[6].

As classes/objetos são imbuídas de significado, atributos e métodos, de acordo com o seu comportamento, representando o papel que desempenham no funcionamento do sistema. Os seus atributos representam propriedades ou parâmetros das entidades, enquanto os seus

métodos alteram os atributos e determinam as suas ações.

Todo o processo de visualização é baseado no conceito do "*visualization pipeline*", o qual implica que cada objeto pode ser simultaneamente um produtor e um consumidor de informação sobre a visualização. Por exemplo um objeto do tipo vtkCubeSource cria uma representação poligonal de um cubo, mas não trata dos aspectos diretamente relacionados com a visualização, para isso há que criar (entre outros) um vtkActor e um vtkRenderer. Assim, enquanto as propriedades geométricas do cubo são definidas a nível da fonte (vtkCubeSource), a cor, textura e/ou a transparência desse cubo são propriedades do vtkActor que recebe como *input* o *output* que veio da vtkCubeSource, enquanto que o tipo de projeção utilizada e o ponto de vista do observador são definidos pelo objeto de tipo vtkRenderer, o qual pode receber com *input* o *output* de diversos atores.

A conduta de visualização pode conter informação sobre a geometria e a topologia do modelo representado, sendo a geometria definida pelas coordenadas dos pontos e a topologia pela incidência através da indicação de quais os pontos que definem cada uma das células - pontos, linhas, figuras planas ou tridimensionais. Além disso a cada ponto ou a cada célula podem estar associados atributos escalares, vetoriais e/ou tensoriais.

Este modelo possibilita a criação de objetos que funcionam como filtros, alterando as propriedades da informação que processam. Por exemplo um objeto do tipo vtkWrapVector pode ler como *input* uma geometria qualquer e adicionar um vetor a cada um dos seus nós. A consistência do modelo é garantida pela verificação que é feita da concordância entre o tipo de *output* produzido a montante e o tipo de *input* requerido a jusante.

A versatilidade do sistema resulta da facilidade com que se desenvolvem interfaces utilizando uma linguagem interpretada, os quais utilizam o núcleo da biblioteca que executa os algoritmos de visualização numa linguagem compilada. Esta combinação possibilita o rápido desenvolvimento de interfaces, sem perda de eficiência.

A licença de utilização, o código fonte, versões binárias e exemplos de aplicação deste *toolkit* podem ser obtidas a partir de http://www.vtk.org/.

3 ARQUIVOS DE DADOS

Apesar de ser possível programar diretamente os dados referentes a um dado modelo utilizando as bibliotecas do sistema *vtk*, essa abordagem é pouco prática, sendo regra geral mais simples e genérico criar um arquivo de dados com a informação desse modelo escrita num formato que possa ser lido por forma a criar um objeto do tipo **vtkSource**. Estão predefinidas classes que permitem ler arquivos em formatos utilizados por outros programas, nomeadamente: Wavefront (.obj), movie.byu (.byu), digital elevation models (.dem) e PLOT3D. Além disso a biblioteca define um formato próprio, em formato texto ou binário, adaptado às suas capacidades e às necessidades. Este formato (.vtk) será utilizado no presente trabalho.

Este arquivo contém informação sobre o tipo de modelo utilizado (STRUCTURED_POINTS, STRUCTURED_GRID, UNSTRUCTURED_GRID, POLYDATA ou RECTILINEAR_GRID), a geometria (as coordenadas dos pontos), a topologia (tipo e incidência das células) e os atributos (escalares, vetores ou tensores

associados aos pontos ou às células).

No contexto do presente trabalho iremos operar com modelos do tipo UNSTRUCTURED_GRID os quais são constituídos por células sem organização, geométrica ou topológica, pré-definida. Apesar de estarem definidos outros tipos de células o VTK_HEXAHEDRON (um hexaedro genérico) e o VTK_TETRA (um tetraedro genérico) são suficientes para definir qualquer malha de elementos finitos tridimensional.

Note-se que, na versão corrente da biblioteca, tanto a geometria como as propriedades variam linear ou bilinearmente nas faces, pelo que elementos quadráticos ou de maior grau devem ser subdivididos, por forma a poder representar com precisão os seus campos. No entanto estão sendo desenvolvidas classes que permitirão representar células não lineares.

4 O PROGRAMA MAYAVI

O programa *mayavi* foi desenvolvido, igualmente num modelo de código aberto, como uma interface para a visualização de dados científicos por meio da biblioteca *vtk*. Está escrito em *python*, uma linguagem interpretada, "object oriented", que permite facilmente conjugar as potencialidades da biblioteca vtk, utilizadas em rotinas compiladas, com a facilidade de desenvolvimento inerente a uma linguagem interpretada.

A utilização deste programa não requer quaisquer conhecimentos específicos de programação e a sua curva de aprendizagem é muito rápida. Para se proceder à visualização há que indicar pelo menos um conjunto de dados, os quais podem, na versão atual, estar guardados num arquivo em formato ".vtk" ou ".plot3d". A esse nível escolhe-se igualmente quais os campos escalares, vetoriais e tensoriais a utilizar nessa visualização. Pode-se escolher um ou nenhum campo de cada tipo.

A visualização propriamente dita recorre a dois conceitos de operação: os módulos e os filtros. Um módulo é um conjunto de rotinas com capacidade para representar um determinado arquivo de dados, por exemplo a representação da superfície de um dado volume, colorida de acordo com o valor do escalar selecionado, enquanto que um filtro opera sobre os dados no *visualization pipeline*, por exemplo efetuando um corte segundo um plano definido pelo utilizador por forma a representar a intersecção de um volume com esse plano.

A escolha do ponto de vista do utilizador é feita interativamente por meio da movimentação do cursor, associada a toques do *mouse* ou teclas, sendo também possível obter informação das coordenadas e do valor numérico dos campos em qualquer célula ou vértice indicado pelo cursor (*picking*).

Para cada módulo e para cada filtro existe um menu de configuração, o qual permite escolher a forma como esse elemento atua, por exemplo qual a posição e orientação do plano de corte ou qual a escala de valores utilizada na coloração.

Os módulos e filtros abaixo indicados serão referidos no contexto da presente comunicação, pelo que importa dar uma indicação básica do seu funcionamento.

SurfaceMap: módulo que representa a superfície exterior do modelo definido, colorida a partir do escalar escolhido ou com uma cor pré-definida;

VelocityVector: módulo que representa em cada vértice do modelo um símbolo (uma seta

ou um cone), correspondente ao campo vetorial escolhido;

TensorGlyphs: módulo que representa em cada vértice um elipsóide ou um conjunto de eixos, orientados e escalados de acordo com o campo tensorial escolhido;

ScalarCutPlane: módulo que realiza um corte num modelo, segundo um plano determinado pelo utilizador;

Threshold: filtro que seleciona as células cujo escalar está entre valores máximo e mínimo definidos pelo utilizador;

WarpVector: filtro que modifica as coordenadas dos vértices adicionado-lhes um vetor, o qual pode ser multiplicado por um fator de escala;

ExtractVectorComponents e *ExtractTensorComponents*: filtros que permitem selecionar componentes de um campo vetorial ou tensorial.

A licença de utilização, o código fonte, pacotes de instalação e manuais de utilização podem ser obtidos a partir de http://mayavi.sourceforge.net/.

5 UM MODELO NUMÉRICO PARA A ANÁLISE DE PÓRTICOS ESPACIAIS DE CONCRETO ARMADO

O modelo numérico utilizado na análise estrutural é baseado no trabalho desenvolvido no programa de doutoramento de Soler, J.G.M, "Análise não-linear de pórticos espaciais de concreto armado"[4], o qual incorpora os efeitos da não linearidade física e geométrica, considerando que o betão se comporta como um material não linear, seguindo as equações constitutivas definidas pela NBR 6118 ou pelo CEB-FIP MC90. A não linearidade geométrica é considerada utilizando um modelo em que a matriz de rigidez é corrigida, tendo em conta os deslocamentos no ponto de equilíbrio anterior[3].

Sendo as lajes aproximadas por um modelo de grelha, todos os elementos estruturais são discretizadas como elementos finitos unidimensionais, com seis graus de liberdade por nó. Apesar de, em termos de visualização, este modelo ser muito simples, quando se pretende representar as peças como elementos tridimensionais, há que ter em conta os principais aspectos que teriam de estar presentes para a visualização dos resultados de uma análise tridimensional.

O programa foi escrito em Fortran[1], sendo os seus resultados fornecidos unicamente como valores numéricos, sendo por isso um bom modelo para testar a adaptabilidade do programa de visualização.

6 VISUALIZAÇÃO DOS DADOS DE ENTRADA (GEOMETRIA, CONDIÇÕES DE FRONTEIRA, AÇÕES)

A primeira visualização que pode ser feita serve para conferir se as coordenadas e as incidências foram introduzidas corretamente. É possível ter uma vista simples do eixo da estrutura (Figura 1) ou da estrutura completa, com as dimensões de vigas, pilares e lajes (Figura 2). Para as obter há que criar um arquivo de dados com a definição da estrutura e utilizar o módulo SurfaceMap para a visualizar.

A definição destas geometrias por meio de um arquivo ".vtk" requer a definição do número de vértices, as suas coordenadas, o número de células (linhas ou hexaedros), a sua incidência e a definição do seu tipo. Por exemplo para definir uma consola usando o modelo mais simples basta criar o seguinte arquivo:

```
# vtk DataFile Version 3.0
Consola de 10 metros (segundo y) representada em wireframe
ASCII
DATASET UNSTRUCTURED_GRID.
POINTS 2 float.
0. 0. 0. 0. 10. 0.
CELLS 1 3
2 0 1
CELL_TYPES 1
3
```

Este arquivo é simples de interpretar, sendo no entanto útil realçar os seguintes aspectos:

- a primeira linha serve para identificar o tipo de arquivo e qual a versão utilizada;
- a segunda linha é um título à escolha do utilizador;
- este é um arquivo em formato texto (ASCII). Existe também um formato binário que não iremos considerar;
- neste artigo consideraremos sempre modelos constituídos por malhas não estruturadas (unstructured grid), mas para modelos garantidamente regulares, tanto geometricamente como apenas em termos da topologia, pode haver vantagem em utilizar outros tipos de malhas;
- as coordenadas dos vértices devem ser introduzidas seqüencialmente, não sendo relevante a divisão em linhas. Ao primeiro vértice é atribuído o número 0 (zero);
- quando se indica o número de células é também necessário indicar a dimensão da lista que as vai representar, o qual é igual ao número de células mais a soma do número de vértices em cada célula;
- a única célula deste modelo é do tipo 3 (**VTK_LINE**) e tem dois vértices, com os números 0 e 1.

A definição do modelo tridimensional é um pouco mais complexa:

```
# vtk DataFile Version 3.0
Consola de 10 metros, com seção 1x1, representada em 3D
ASCII
DATASET UNSTRUCTURED_GRID.
POINTS 8 float.
0.5  0.0  0.5  0.5  0.0 -0.5 -0.5  0.0 -0.5 -0.5  0.0  0.5
0.5 10.0  0.5  0.5 10.0 -0.5 -0.5 10.0 -0.5 -0.5 10.0  0.5
CELLS 1 9
```

8 0 1 2 3 4 5 6 7
CELL_TYPES 1
12

Neste caso há novas situações a realçar:

- a única célula deste modelo é do tipo 12 (**VTK_HEXAHEDRON**), a qual tem oito vértices e é necessário introduzir suas incidências de acordo com a seqüência definida pelo sistema;
- a numeração dos vértices deixa de corresponder à numeração dos nós do modelo estrutural, os quais deixam aliás de estar presentes no modelo visualizado.

Optou-se no presente caso por gerar para cada elemento os vértices necessários para o representar, os quais passam a ter uma numeração múltipla do número de nós gerados na discretização da seção transversal.

A implementação que foi feita só permite visualizar corretamente barras de seção retangular, mas, recorrendo a células do tipo VTK_HEXAEDRON e eventualmente VTK_WEDGE, é possível representar barras cuja seção transversal possa ser aproximada por quadriláteros e triângulos.

A visualização em modelo tridimensional permite, à custa de mais informação, uma visualização mais correta do modelo, nomeadamente quando existam problemas na orientação das seções. Não se considera relevante, no presente contexto, refinar a representação dos vértices em que convergem barras com seções diferentes. Como o modelo é baseado em peças uni ou bidimensionais considera-se mais correto utilizar o resultado por ele calculado, o qual não é totalmente consistente nos nós de ligação.

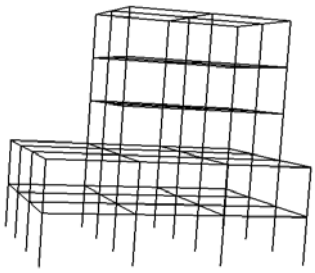

Figura 1: Representação dos eixos das barras do pórtico espacial.

Figura 2 : Representação das barras e lajes do pórtico espacial.

Como o modelo numérico exige a definição de condições de fronteira em termos de deslocamentos lineares e de rotações a sua visualização não é imediata. Uma solução possível é utilizar símbolos diferentes consoante o tipo de restrição aplicada no nó. No entanto essa abordagem não é simples de implementar diretamente, tendo-se optado em vez disso por

definir em cada nó dois vetores, correspondentes às condições de fronteira, os quais são nulos nos vértices livres (Figura 3). A visualização dos elementos representado os vetores (linhas, setas ou cones) é feita recorrendo ao módulo VelocityVector.

Estes vetores são definidos como campos associados aos vértices dos dados visualizados. Para os exemplos da consola apresentados anteriormente é necessário acrescentar ao arquivo ".vtk" a informação dos vetores correspondentes aos vértices, os quais devem valer (1;1;1) nos nós com coordenada y=0 e (0;0;0) nos nós com coordenada y=10.

Tem-se então:

POINT_DATA 2
VECTORS apoio_linear float
1. 1. 1. 0. 0. 0.
VECTORS apoio_rotação float
1. 1. 1. 0. 0. 0.

para o primeiro exemplo e

POINT_DATA 8
VECTORS apoio_linear float
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
VECTORS apoio_rotação float
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

para o segundo.

Esta parte do arquivo de dados, referente a atributos dos vértices, deve ser definida tendo em conta seguintes aspectos:

- a dimensão de POINT_DATA deve ser igual ao número de vértices (POINTS);
- podem ser definidos vários vetores;
- para definir atributos escalares deve utilizar-se a instrução "SCALARS nome tipo 1", em que o tipo é regra geral float (um número real), mas pode também ser, por exemplo, int (um número inteiro). Neste caso é também necessário indicar a tabela de cores a utilizar na representação desse escalar, por meio da instrução adicional "LOOKUP_TABLE nome". Regra geral basta utilizar a tabela padrão do sistema, indicando o nome default. O número de valores a indicar é igual ao número de vértices;
- podem também ser definidos campos tensoriais usando a instrução "TENSORS nome tipo". Neste caso há que indicar nove valores por vértice.

A definição de atributos das células é feita de um modo semelhante, sendo iniciada por meio da indicação “CELL_DATA tamanho”, sendo tamanho igual ao número de células.

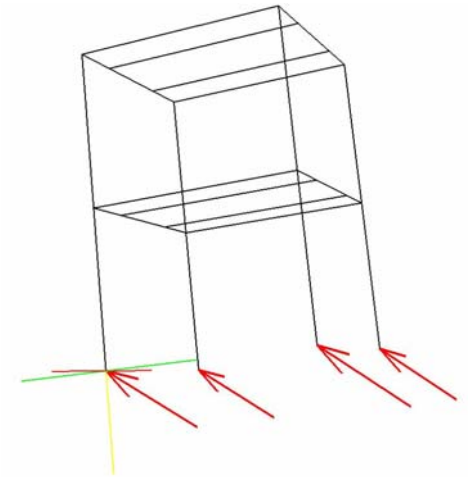

Figura 3 : Visualização das condições de apoio.

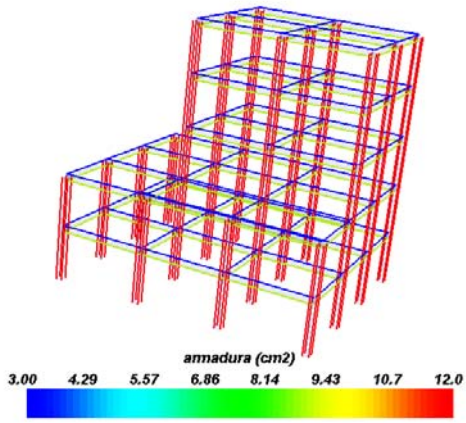

Figura 4 : Localização das armaduras da estrutura espacial.

Como se referiu, recorrendo ao picker sobre os nós de fronteira é possível validar as ligações. Por exemplo, no caso de se solicitar informação sobre um dos apoios na figura 3, obtém-se a informação apresentada abaixo, a qual informa que o apoio possui restrições em relação a x, y e z.

Picked Point number:18
Picked position: 400.000000, 0.000000, 400.000000
Scalar: None
Vector: (1.0, 1.0, 1.0)
Tensor: None

É também possível visualizar a armadura, como exemplificado na Figura 4 e verificar se o valor da área de armadura está correto. Para isso prepara-se um arquivo separado, em que os varões são definidos como linhas, fazendo-lhes corresponder a área como um atributo escalar.

A visualização simultânea dos arquivos com a definição da estrutura e da armadura, fazendo a estrutura parcialmente transparente ou usando planos de corte, permite uma fácil inspeção visual dos dados introduzidos.

A visualização das ações a que a estrutura é submetida, está exemplificada na Figura 5, onde as forças aplicadas estão representadas como cones, orientados segundo a direção da resultante, cujo tamanho e cor varia em função da sua intensidade. Neste caso as forças são definidas como um campo vetorial nos vértices, à semelhança do que foi apresentado para os

apoios. Os momentos aplicados são definidos como um novo campo vetorial. Na Figura 5 foi também ativado o módulo Axes, o qual permite representar o sistema de eixos.

Por meio dos procedimentos apresentados podem ser facilmente detectados erros grosseiros na introdução de dados, os quais são os mais comuns, como por exemplo a introdução de coordenadas ou de incidência trocadas. É também possível detectar muitos erros mais subtis, como por exemplo na definição de orientações ou de atributos, sendo neste caso exigida alguma atenção por parte do utilizador.

Há também recursos que podem auxiliar na interpretação da forma e da sua discretização. Pode-se, por exemplo verificar a seção transversal discretizada, como representado na Figura 6.

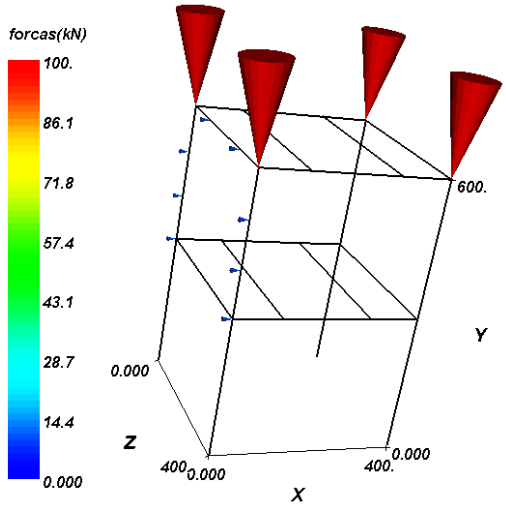

Figura 5 : Representação por meio de cones, das forças aplicadas à estrutura e indicação do sistema de eixos.

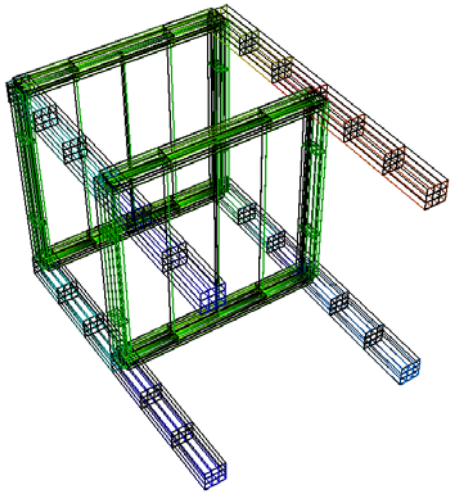

Figura 6 : Representação da discretização das barras e laje do pórtico espacial.

7 VISUALIZAÇÃO DE RESULTADOS

Visualizando os esforços e deslocamentos fornecidos pela análise é possível, de uma forma simples, efetuar uma validação preliminar desses valores. É também possível dar início ao processo de interpretação que permite ao analista entender, a partir dos valores que obtém, se a sua estrutura funciona da forma que idealizou.

Como os esforços são grandezas vetoriais uma forma de proceder à sua visualização seria representar em cada secção, para cada esforço, o vetor correspondente. Apesar desta solução poder ter vantagens em determinadas situações, como por exemplo quando há dúvida acerca de qual a orientação escolhida, optou-se por representar os esforços locais como escalares cuja intensidade determina a coloração das células em que se subdividem as barras, conforme se exemplifica na Figura 7. Assim as seções onde está instalado o menor esforço (regra geral o maior esforço negativo) são coloridas a vermelho, sendo as seções onde está instalado o maior esforço (regra geral o maior esforço positivo) coloridas a azul. É possível conhecer o

valor do esforço numa determinada seção por inspeção visual do modelo e da escala de cor ou recorrendo do picker.

A introdução destes dados no arquivo de dados é feita de uma forma semelhante à descrita no contexto da definição das condições de apoio, tendo em conta que neste caso interessa definir escalares em células, sendo portanto necessário utilizar as instruções "CELL_DATA ..." e "SCALARS nome ...".

Sobre a representação de um campo escalar é possível aplicar o filtro Threshold, o qual permite eliminar as células em que o campo escalar considerado se situa fora de um intervalo definido pelo utilizador, conforme representado na Figura 8, onde se representa um campo de momentos e se eliminaram as células de esforço positivo.

Diversas representações são possíveis para visualizar o campo de deslocamentos correspondente à solução: desenhando os vetores correspondentes ao deslocamento de cada vértice, colorindo o modelo de acordo com a intensidade de uma componente ou da resultante do deslocamento e adicionando a cada coordenada o valor do deslocamento, por forma a representar a estrutura deformada.

Os vetores correspondentes ao deslocamento de cada vértice podem ser desenhados diretamente recorrendo ao módulo VelocityVector, conforme foi indicado para a representação das condições de apoio.

Para efetuar a coloração de acordo com a intensidade do deslocamento há que extrair dos vetores que definem o deslocamento em cada vértice a informação escalar desejada. Para isso utiliza-se o filtro ExtractVectorNorm caso se deseje utilizar a intensidade da resultante do deslocamento ou o filtro ExtractVectorsComponents se desejar uma coloração de acordo com uma das componentes do deslocamento, a qual tem de ser selecionada pelo utilizador. Esta última situação está exemplificada na Figura 9, para o deslocamento horizontal segundo o eixo x. Para obter esta figura foi aplicado o filtro ExtractVectorComponents, escolhido a opção X-component, com o módulo SurfaceMap ativado.

Como foi referido o filtro WarpVector permite adicionar à coordenada de cada vértice um campo vetorial, o qual pode ser multiplicado por um fator de escala. Dessa forma a geometria da modelo passa a ser a da estrutura deformada, sendo possível representar sobre ela um campo escalar, conforme indicado na Figura 10, onde um dos campo de momentos fletores está representado sobre a estrutura deformada, com os deslocamentos ampliados 36 vezes. É também possível representar em simultâneo a estrutura deformada e indeformada.

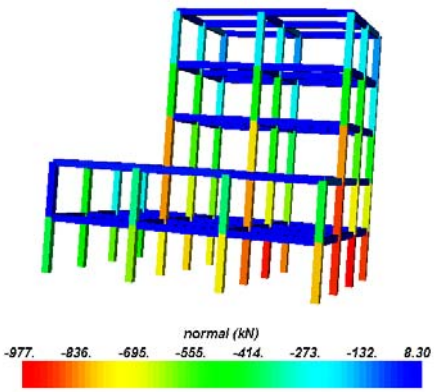

Figura 7 : Distribuição dos momentos segundo x, com eliminação das células com esforço positivo.

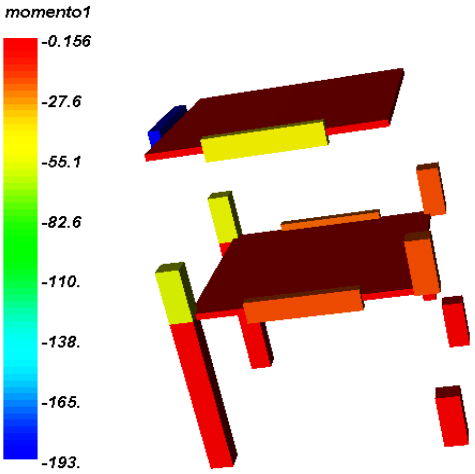

Figura 8 : Distribuição dos deslocamentos segundo x.

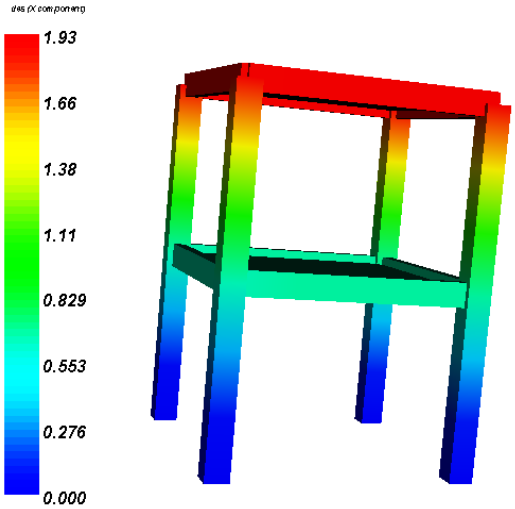

Figura 9 : Distribuição do esforço normal.

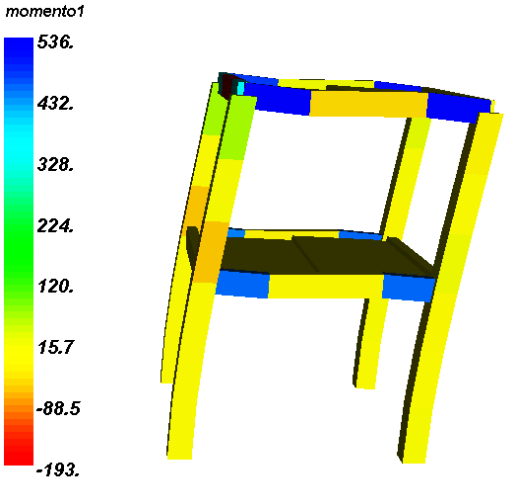

Figura 10 : Distribuição do momento segundo x, representado na figura deformada (com os deslocamentos ampliados 36 vezes).

8 CONCLUSÕES

A utilização do programa MayaVi permitiu visualizar, de uma maneira muito simples, os dados de entrada e os resultados de um programa que tinha sido concebido sem ter em conta essa necessidade. Por meio dessa visualização é possível verificar os dados de entrada, corrigindo, pelo menos, os erros mais grosseiros.

Como a informação é transmitida por meio de um arquivo em formato texto, com uma

sintaxe simples e fácil de aprender e utilizar, a linguagem de programação em que foi desenvolvido o programa de análise torna-se irrelevante para o utilizador[7].

Os módulos e filtros existentes cobrem na generalidade as necessidades sentidas, mas a modularidade do programa permite a sua expansão, o que é particularmente facilitado pelo fato de ser criado num modelo de código aberto, existindo uma comunidade de utilizadores que cooperam para o seu desenvolvimento. A linguagem de programação utilizada é simples, compacta e fácil de aprender, pelo que é fácil introduzir novas opções com base nos elementos existentes.

Pelas razões expostas julgamos que esta é uma boa solução de visualização para projetos de investigação centrados no desenvolvimento de modelos numéricos.

9 AGRADECIMENTOS

Este trabalho foi realizado no âmbito das atividades de investigação do ICIST, Instituto de Engenharia de Estruturas, Território e Construção, tendo sido possível graças à bolsa BEX2248/01-8, concedida pela CAPES no âmbito do projeto CAPES/ICCTI 075/01.

10 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ellis, T. M. R., Fortran 77 Programming, second edition, (with an introduction to the fortran 90 standard), Addilson Wesley,1990.

[2] Schroeder, W.; Martin,K.; Lorensen, B.; The visualization Toolkit, An Object-Oriented Appoach to 3D Graphics, Prentice Hall, second edition, 646 p . , 1997.

[3] Soler, J.G.M, Análise não linear de pórticos planos de concreto armado, Dissertação de Mestrado, EPUSP/USP,1989.

[4] Soler, J.G.M, Análise não linear de pórticos espaciais de concreto armado, Tese de Doutoramento, EPUSP/USP,1995.

[5] Ramachandran, P., MayaVi: A free tool for CFD data visualization, 4rd Annual CFD Symposium, Aeronautical Society of India, 2001.

[6] Jr. Drake, F.L: Rossum, G. , Python Reference Manual, Release 2.1.2, 2002.

[7] J.G.M. Soler, J.P.M. Almeida, P.M. Pimenta "Pré e pós processador para utilização em programas de elementos finitos", First South-Americam Congress on Computational Mechanics, III Brazilian Congress on Computational Mechanics and VII Argentine Congress on Computational Mechanics, MECOM2002, 28-31 de Outubro de 2002, Santa Fé-Paraná, Argentina.