Mecánica Computacional, Volumen XVIII - pp. 545-554
Compilado por: E. Dari, C. Padra y R. Saliba
San Carlos de Bariloche, noviembre de 1997

GERADOR DE MALHA DE ELEMENTOS FINITOS COMO UM APLICATIVO WINDOWS

Edgard Sousa Junior , João Batista de Paiva
Universidade de São Paulo, Escola de Engenharia de São Carlos, Departamento de Engenharia de Estruturas, Av. Dr. Carlos Botelho, 1465, São Carlos-SP, Brasil

RESUMO

Os geradores de malha são a interface entre o engenheiro e os programas de cálculo estrutural baseados em elementos finitos. É fundamental que essa interface seja bem elaborada para agilizar a entrada de dados e minimizar os erros tanto de modelagem quanto de digitação. Como os recursos oferecidos pelo ambiente Windows favorecem ao desenvolvimento de aplicativos gráficos e também facilitam a comunicação entre aplicativos, foi desenvolvido para este ambiente, um pré-processador para programas de elementos finitos voltados para a análise estrutural. O aplicativo possui recursos para gerar malhas de elementos finitos de placa pelo processo de macro elementos. A malha de elementos finitos é gerada por partes, através de comandos que geram malhas quadrilaterais. Depois, estes macro elementos são unidos, formando a malha final. Existem ainda, recursos para gerar elementos de barra. A malha de elementos finitos é desenhada à proporção que se entra com os dados. O usuário pode editar uma malha eliminando elementos ou alterando suas propriedades, utilizar o recurso de zoom e várias opções de visualização da malha, como numeração de elementos e pontos nodais, e ainda, ocultar tipos de elementos para facilitar a análise da malha. Após a malha ter sido gerada, poderá ser salva em disco em um arquivo com o formato definido pelo usuário, porque geralmente o formato de arquivo de dados é específico para cada software de processamento. Desta forma, pode-se gerar arquivos de dados para serem utilizados por diferentes programas de cálculo ou aplicativos gráficos.

ABSTRACT

The preprocessors are the interface between the engineer and the programs used in finite elements analysis. This interface need to be very well constructed to minimize the input errors. Thus, it is presenting a Windows preprocessor for building slab analysis by Finite Element Method. It is an interactive program, allowing problems to be identified while the model is generated. It has graphical resources to define the structure. The program output is defined by the user; therefore it is possible to use the preprocessor with many other finite element programs. This allows the engineer to choose the appropriate program depending on the kind of analysis is wanted. The program can define frame and shell finite elements. As slabs usually need large numbers of these elements to be properly analyzed, they are generated using automatic meshing

INTRODUÇÃO

Como o método dos elementos finitos vem sendo muito utilizado por engenheiros na análise de pavimentos de edifícios, torna-se necessário o desenvolvimento de pré-processadores específicos que facilitem e agilizem a geração de malhas de elementos finitos que representam pavimentos de edifícios. Nestes pré-processadores, as atenções são centradas nos geradores de malhas porque delas depende a melhor ou pior representação do pavimento, o que interfere diretamente no resultado do cálculo estrutural.

Já foram desenvolvidas várias técnicas de geração automática de malha. Os primeiros trabalhos a esse respeito datam da década de setenta. Até hoje são apresentadas novas técnicas e aperfeiçoadas outras. A escolha da técnica de gerar malhas depende da estrutura a ser analisada. Em certos casos, para uma malha ser gerada com eficiência, podem ser empregados combinações de técnicas. As técnicas empregadas até então nos geradores de malhas podem ser classificadas em quatro: Triangulação Automática, Transformação de Coordenadas, Procedimentos de Suavização e Funções Combinadas. Os geradores de malha devem oferecer um equilíbrio entre tempo de processamento e qualidade da malha. Normalmente um processo que gera uma malha bem distribuída demora muito para ser elaborado, e em alguns casos não compensa o tempo perdido no processamento quando uma malha simples pode dar resultados equivalentes aos de uma malha bem trabalhada. Na análise de problemas de engenharia mecânica onde encontram-se variações muito grandes de tensões em regiões muito próximas, as malhas devem ter algumas áreas bem mais refinadas que outras. Já na análise de pavimentos de edifícios as malhas podem ser mais uniformes e raramente algumas regiões do projeto exigem uma densidade maior de elementos.

Hoje os computadores pessoais já têm capacidade de executar um aplicativo de cálculo estrutural baseado no método dos elementos finitos, sendo assim, desenvolveu-se um aplicativo para o ambiente Windows 95 que gera malhas de elementos finitos pela técnica Triangulação Automática. O gerador é otimizado para gerar malhas de pavimentos de edifícios de construção civil. A técnica de Triangulação Automática foi escolhida para ser utilizada no aplicativo porque ela exige pouco tempo de processamento e define elementos bem distribuídos. O aplicativo foi desenvolvido em C++, ele possui grande versatilidade na modelagem de pavimentos de edifícios de diferentes concepções arquitetônicas. O gerador de malha consegue gerar malhas de pavimentos com curvas, vigas em diagonal e lajes nervuradas.

A PLATAFORMA WINDOWS

Optou-se em fazer um aplicativo para a plataforma Windows porque essa está se tornando o padrão mais aceito e usado no mercado da microinformática. Isso se dá pelas facilidades que um ambiente gráfico multitarefa pode oferecer. Um sistema com uma interface gráfica exige pouco treinamento e pouco esforço do usuário pois, esse tipo de sistema é tão intuitivo que usando apenas a lógica e associações, é possível operá-lo. A multitarefa, capacidade de executar mais de um aplicativo ao mesmo tempo, facilita a troca de dados entre programas, é possível transferir dados de uma forma simples com a utilização do *Clipboard* - Área de Transferência - , ou através de métodos mais sofisticados, como o DDE - *Dinamic Data Exchange* - e o OLE - *Object Linking and Embedding*.

O programador para Windows não precisa se preocupar em que equipamento seu programa será executado. A comunicação entre aplicativo e máquina é feita através do Windows que utiliza *drivers* - controladores de dispositivos - para cada periférico. Desta forma o Windows consegue proporcionar gráficos independentes de dispositivos. O programador deve utilizar as rotinas da GDI - *Graphic Device Independent* - do Windows para que o programa possa desenhar em qualquer dispositivo utilizando o mesmo conjunto de chamadas, por exemplo, a rotina **Arc** da GDI é chamada para desenhar um arco na tela assim como em impressoras, independente se o monitor for VGA, SVGA ou outro e do modelo da impressora.

O Windows é um ótimo gerenciador de memória. Quando se executa mais de uma cópia ou instância de um mesmo aplicativo, o Windows faz com que o seguimento de código seja compartilhado entre elas. Isto reduz muito a memória gasta. Os seguimentos de um programa Windows são classificados como descartáveis, móveis e fixos. Quando o Windows precisa em

um determinado instante de uma certa quantidade de memória e não encontra, ele libera os seguimentos descartáveis - aqueles que podem ser recuperados a partir do executável em disco - que não estão sendo usados no momento. Depois, o Windows defragmenta a memória movendo os seguimentos classificados como móveis e assim ele consegue um seguimento continuo de memória livre. Além disso, o Windows consegue criar memória virtual em disco de maneira transparente para o programador. Outro recurso de gerenciamento de memória utilizado pelo Windows são as DLL's - *Dinamic Link Libraries* - que são um formato especial de arquivo executável para exportar e importar funções. Quando um programa é compilado, as sub-rotinas ou funções são vinculadas ao código do programa executável, isso se chama vinculação estática. Desta forma se programas diferentes que utilizam funções iguais mas, que foram compiladas utilizando a vinculação estática, forem executados ao mesmo tempo, haverá um desperdício de memória pois, duas funções iguais estarão na memoria ocupando espaços diferentes. Com o uso de DLL's, ou seja, bibliotecas de vinculação dinâmica, as funções são compiladas separadas do código de programa executável e a vinculação é feita em tempo de execução. Com as DLL's os aplicativos podem compartilhar tanto funções como recursos, o que representa mais uma economia, e quando nenhum programa as estiver usando elas são descartadas.

O AMBIENTE DE DESENVOLVIMENTO C++

A linguagem de computador adotada foi C++ porque é uma linguagem que oferece recursos para a Programação Orientada a Objetos. Isso ajuda a elaborar projetos grandes e complexos. A manutenção de projetos orientados a objetos não exige grande esforço. É importante salientar que o próprio Windows foi projetado em C, sendo também, um dos motivos para a escolha dessa linguagem. Utilizou-se compilador Borland C++ 5.0. Esse compilador oferece OWL 5.0 - *Object Windows Library 5.0* -, uma biblioteca orientada a objetos que simplifica e agiliza o desenvolvimento de aplicativos para Windows 95. Ou seja, é possível desenvolver aplicativos de 32 bits. Isso significa que pode-se utilizar matrizes com dimensões bem maiores do que em aplicativos de 16 bits que são limitados a blocos de 64 KB.

APESENTAÇÃO DO APLICATIVO

O aplicativo intitulado PEC - Projeto Estrutural por Computador - é dividido em módulos por uma questão de organização e também para um melhor aproveitamento dos recursos computacionais. Porém, o usuário final não identifica essa divisão, ele vê o aplicativo como se fosse um único módulo. Os módulos executáveis do aplicativo são: PEC.EXE, DADOS.DLL, LISTAS.DLL, MALHA.DLL, GERADOR.DLL, EDITA.DLL e RELAT.DLL. O módulo principal, PEC.EXE, que é responsável pelo gerenciamento da janela principal. Para se ativar o aplicativo basta executar este módulo e os demais são executados automaticamente. Os outros módulos contém funções específicas, o módulo DADOS.DLL contém as classes ou objetos responsáveis pelos dados gerais da estrutura como nome do projeto, coeficiente de Poisson e módulo de Young; o módulo LISTAS.DLL é composto pelas classes responsáveis pela visualização e edição dos dados da malha como os pontos nodais, elementos de placa e elementos de viga; o módulo MALHA.DLL possui as classes responsáveis pela entrada de dados para a geração automática das malhas de elementos finitos de placa e de elementos de viga; o módulo GERADOR.DLL possui as classes responsáveis pela geração automática da malha de elementos finitos de placa e de elementos de viga; o módulo EDITA.DLL contém as classes de edição dos dados diretamente na janela gráfica, essas funções são chamadas pelo menu flutuante que será detalhado mais adiante; e finalmente o módulo RELAT.DLL, as

funções utilizadas para exportar os dados de uma malha, aquelas responsáveis pela geração dos relatórios de saída de dados.

Janela principal

A comunicação do aplicativo com o usuário é através de uma interface MDI - *Multiple Document Interface* -, ou seja, ele suporta trabalhar com mais de um arquivo de dados abertos ao mesmo tempo, Fig. 1. Os comandos do aplicativo podem ser chamados de várias formas: através do menu localizado no topo da janela principal, onde encontra-se a maioria dos comandos, através das barras de botões e ainda através do toque do *mouse* sobre a janela gráfica, o que faz surgir perto das coordenadas do cursor do *mouse* o menu flutuante.

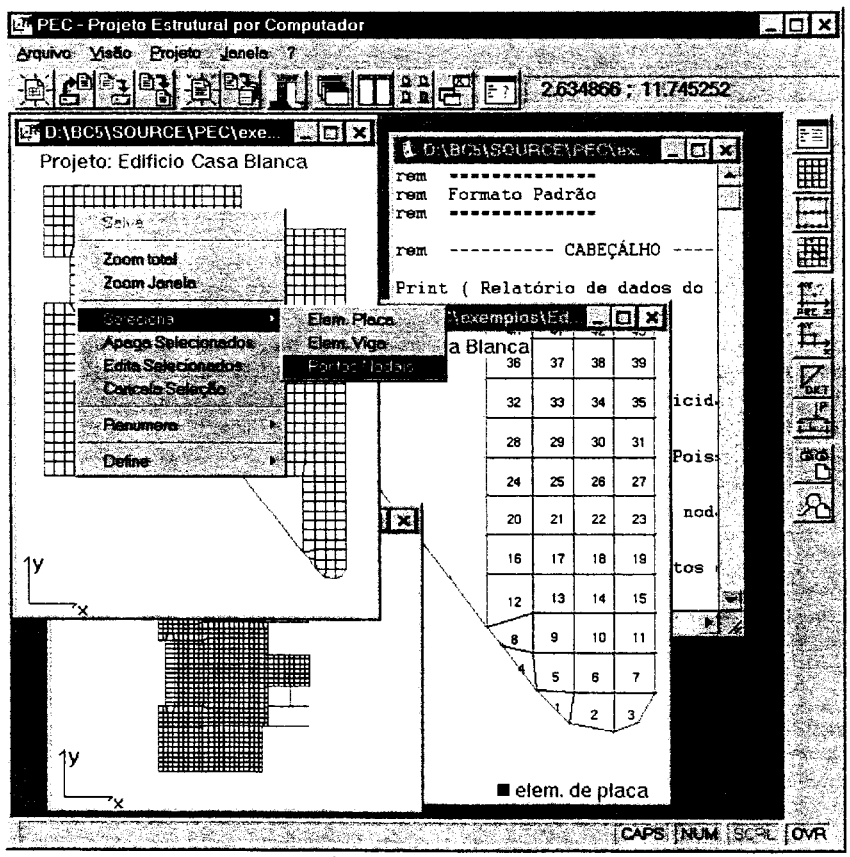

Fig 1 - Janela principal

O aplicativo trabalha com dois tipos de visualização dos dados: gráfica e texto. Os dados da janela gráfica podem ser vistos em forma de relatórios em que o formato é definido pelo próprio usuário. Desta forma, estes relatórios podem ser utilizados como arquivos de entrada de dados para qualquer programa de cálculo estrutural baseado no método dos elementos finitos. Os formatos dos relatórios são definidos em arquivos de texto puro, como os gerados

pelo NOTEPAD.EXE do Windows, contendo comandos básicos como: `print` para imprimir uma *string* ou valor de variáveis, `Do - EndDo` para se fazer laços, e `If - EndIf` para se definir condições, evitando que dados não desejados sejam impressos no relatório. Os formatos podem ser salvos, para posterior utilização. Não há limite para a quantidade de formatos. Para facilitar a manipulação dos relatórios foi incorporado no aplicativo um editor de texto similar ao NOTEPAD.EXE, mas com a capacidade de abrir arquivos grandes, Fig 2.

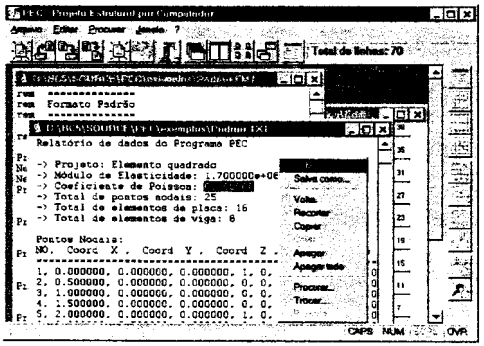

Fig. 2 - Janela texto ativa

Entrada e edição dos dados

Os dados do aplicativo são: poligonais de contorno, pontos pré-definidos, pontos nodais e elementos finitos de placa e de barra. A entrada de cada um destes elementos pode ser por uma caixa de diálogo específica em que é possível visualizar uma lista dos elementos existentes, editá-los, apagá-los e inserir novos elementos. Além desta forma de entrada de dados, o aplicativo permite a entrada, edição e remoção individual de um dos elementos diretamente na janela gráfica com o uso do *mouse* e do menu flutuante. Como os elementos finitos de placa e de barra, normalmente são de grande volume, foi desenvolvido um gerador automático para cada um destes.

As poligonais de contorno e os pontos pré-definidos são linhas e pontos que definem a geometria da malha a ser gerada, essas linhas e pontos podem ser a planta baixa de um pavimento gerada no formato DXF, e servem principalmente como orientação na geração da malha de um pavimento.

Os elementos finitos de placa e de barra, juntamente com os pontos nodais, são utilizados na discretização da estrutura. Esses elementos podem ser selecionados com o *mouse* e depois serem editados ou removidos. Como exemplo da edição de um grupo de pontos nodais diretamente na janela gráfica tem-se a figura 3. É importante notar que somente os campos preenchidos desta caixa de diálogo é que são alterados, os campos em branco permanecem com seus valores inalterados.

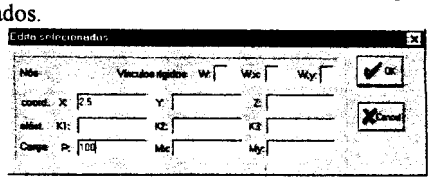

Fig. 3 - Edição de alguns pontos nodais através da janela gráfica

Como exemplo, a caixa de diálogo Pontos Nodais é mostrada na figura 4. Cada um dos outros elementos (poligonais de contorno, pontos pré-definidos, elementos finitos de placa e de barra) possuem uma caixa de diálogo similar a da figura 3.

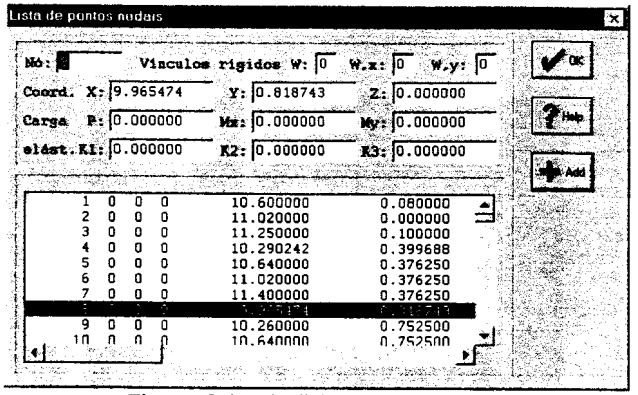

Fig. 4 - Caixa de diálogo Pontos Nodais

Existe ainda uma caixa de diálogo chamada Dados Gerais em que são armazenadas as propriedades dos materiais utilizados pelos elementos finitos e dados específicos como o nome do projeto.

O comando Malha gera uma malha de elementos finitos quadrilateral. É possível gerar várias malhas quadrilaterais e depois fazer a junção delas, formando assim a malha final, figura 5. Os elementos de viga, como dito anteriormente, também podem ser gerados automaticamente como mostrado na figura 6. Os parâmetros do gerador de vigas são apenas dois pontos, inicial e final da viga, e as propriedades dos elementos finitos de barra como inércia e área da seção transversal. Uma viga pode ser gerada em qualquer direção e os elementos de placa que estiverem na trajetória do eixo da viga, têm as coordenadas dos seus pontos nodais ajustadas automaticamente para coincidirem com o eixo da viga.

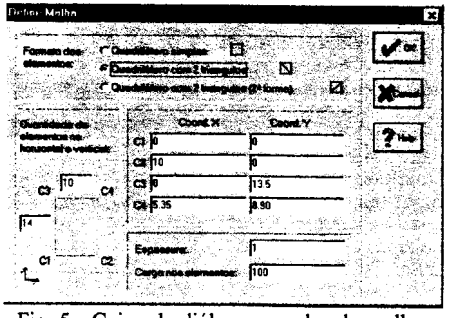

Fig. 5 - Caixa de diálogo gerador de malhas

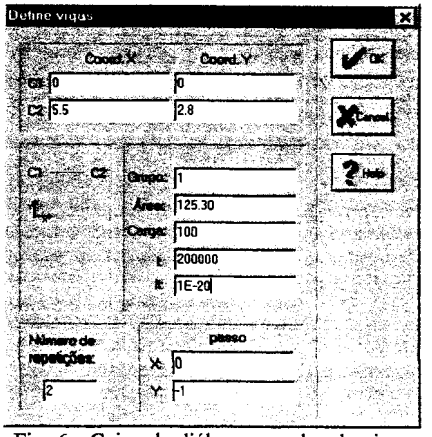

Fig. 6 - Caixa de diálogo gerador de vigas

Para gerar as vigas o primeiro passo do gerador de vigas é identificar o ponto nodal mais próximo do ponto *C1*, que é o ponto inicial da viga. Depois as coordenadas do ponto mais próximo são substituídas pelas de *C1*. O mesmo é feito para o ponto *C2*, que é o ponto final da viga. É importante salientar que os pontos *C1* e *C2* não são gerados e sim é feito um ajuste em pontos já existentes. Logo, devem existir pelo menos dois pontos antes de se gerar uma viga.

O segundo passo é ajustar os pontos nodais da malha entre os pontos *C1* e *C2* para que coincidam com o eixo da reta definida por esses dois pontos. O procedimento utilizado é o seguinte:

A inclinação *m* da reta que passa pelos pontos *C1* (x_1,y_1) e *C2* (x_2,y_2) (figura 7) é calculada pela expressão (1).

$$m = \frac{y_2 - y_1}{x_2 - x_1} \qquad (1)$$

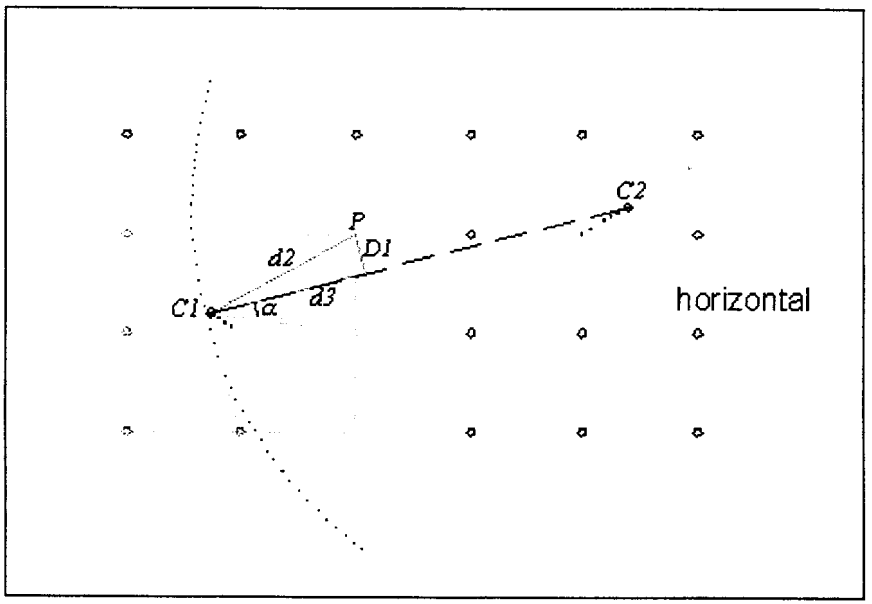

Fig. 7 - Esquema representativo da geração de uma viga

O angulo α que a reta faz com a horizontal é obtido de (2).

$$tg\alpha = m \qquad (2)$$

Entre os pontos nodais pertencentes aos elementos de placa que convergem para o ponto *C1* são selecionados aqueles em que a distância a *C2* é menor que a distância de *C1* a *C2*. Dentre estes selecionados (os pontos mais claros da figura 7) é escolhido o que estiver mais próximo da reta que passa por *C1* e *C2* e nomeado de ponto *P*. São utilizadas duas expressões de distância, a primeira (3) indica a distância *d* entre dois pontos $P_1(x_{P1},y_{P1})$ e $P_2(x_{P2},y_{P2})$ e a

segunda (4), a distância D entre um ponto $P(x_P,y_P)$ e uma reta $Ax+By+C=0$. Estas expressões estão indicadas a seguir:

$$d = \sqrt{(x_{P2} - x_{P1})^2 + (y_{P2} - y_{P1})^2} \quad (3)$$

$$D = \frac{A \cdot x_p + B \cdot y_p + C}{\pm\sqrt{A^2 + B^2}} \quad (4)$$

sendo que, os coeficientes A, B e C, para a reta que passa por $C1(x_1,y_1)$ e $C2(x_2,y_2)$ (figura 7) são dados por:

$$A = y_2 - y_1 \quad (5)$$

$$B = x_1 - x_2 \quad (6)$$

$$C = (x_2 \cdot y_1) - (x_1 \cdot y_2) \quad (7)$$

Sendo D_1 a distância entre P e a reta que passa por $C1$ e $C2$, d_2 a distância entre os pontos P e $C1$, é calculado a distância d_3 (figura 7).

$$d_3 = \sqrt{D_1^2 + d_2^2} \quad (8)$$

Para que o ponto $P(x_P, y_P)$ seja movido para o eixo da reta que passa por $C1$ e $C2$ a sua coordenada X passa a ser a do ponto $C1(x_1,y_1)$ mais a projeção de d_3 no eixo X e a coordenada Y de P passa a ser a de $C1$ mais a projeção de d_3 no eixo Y, isto é:

$$x_P = x_1 + d_3 \cos \alpha \quad (9)$$

$$y_P = y_1 + d_3 \operatorname{sen} \alpha \quad (10)$$

É então gerado um elemento de viga que inicia em $C1$ e termina em P. Se o elemento de viga estiver passando sobre um elemento de placa, este é subdividido em dois elementos. O nó P passa a ser o novo ponto $C1$ e o processo se repete para encontrar o novo ponto P e gerar o próximo elemento de viga. O processo vai se repetindo até que o ponto $C1$ coincida com $C2$, neste ponto a viga já está gerada. Se o procedimento escolhido for o de gerar vigas em conjunto, o valor do ponto $C1$ inicial e o do ponto $C2$ são incrementados com o valor do passo e o processo inteiro se repete para gerar a próxima viga.

Geração de relatórios

Após uma malha de elementos finitos ter sido gerada, é possível utilizar o comando Gerador de Relatórios para gerar arquivos de dados que possam ser lidos por programas de cálculos ou programas gráficos. Primeiramente deve ser definido um formato de arquivo de dados. Este formato é um arquivo texto contendo comandos de impressão como o exemplo da listagem 1. Normalmente o usuário irá defrontar-se com os comandos de formato somente uma vez na definição de um arquivo de formato para um programa gráfico ou de cálculo. Por exemplo, é possível definir um arquivo de formato para gerar um relatório que seja um arquivo DXF composto pelos pontos nodais e os elementos finitos de placa e barra. Para qualquer malha gerada pelo aplicativo PEC, pode-se utilizar este arquivo de formato na geração dos arquivos DXF. A figura 8 mostra a caixa de diálogo do gerador de relatórios. A listagem 2 mostra o relatório gerado pela listagem 1.

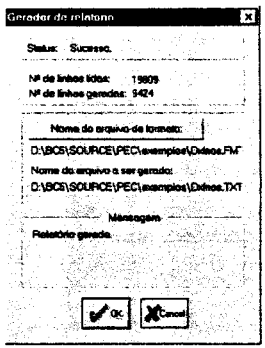

Fig. 8 - Caixa de diálogo do gerador de relatórios

Listagem 1 - Exemplo de um arquivo de formato

```
rem  ---------- CABEÇALHO -----------
Print ( Relatório de dados do Programa
PEC)
NewLine
NewLine
Print ( -> Projeto: )
  Project
  NewLine
Print ( -> Módulo de Elasticidade: )
  Elastic
  NewLine
Print ( -> Coeficiente de Poisson: )
  Poisson
  NewLine
Print ( -> Total de elementos de viga:
)
  TotBeam
  NewLine

rem  LISTA DE ELEMENTOS FINITOS DE VIGA

if (TotBeam > 0)
  NewLine
  Print ( Elementos de Viga:)
  NewLine
  Print (Grup, Elem, Nó ini., Nó final,
In.Flexão, In.Torção, Área, Carg)
  NewLine
  Print ( )

  loop 1, 53
    Print (=)
  Endloop
  NewLine


  rem Guarda em VAR1 o total de grupos de vigas
  var1 (0)
  do 1, totbeam
    if (var1 < BeamGroup)
      var1(BeamGroup)
    endif
  enddo
  IF (var1 < 1)
    NEWLINE
    print (ERRO: TOTAL DE GRUPOS DE VIGAS = 0)
    NEWLINE
  ENDIF
  loop 1, var1
    Do 1, TotBeam
      if (BeamGroup = Loop)
        Print (  )
        BeamGroup
        Print (, )
        BeamNum
        Print (, )
        BeamIni
        Print (, )
        BeamEnd
        Print (, )
        BeamI
        Print (, )
        BeamIT
        Print (, )
        BeamArea
        Print (, )
        BeamLoad
        NewLine
      endif
    EndDo
  endloop
EndIf
```

Listagem 2 - Exemplo de um relatório

```
Relatório de dados do Programa PEC

 -> Projeto: Elemento quadrado
 -> Módulo de Elasticidade: 1.700000e+06
 -> Coeficiente de Poisson: 0.300000
 -> Total de pontos nodais: 25
 -> Total de elementos de placa: 16
 -> Total de elementos de viga: 8

Elementos de Viga:
Grup, Elem, No ini., No final, In.Flexão, In.Torção,Área,Carg
-------------------------------------------------------------
1, 5, 21, 22, 0.002200, 0.000001, 0.072000, 9.000000
1, 6, 22, 23, 0.002200, 0.000001, 0.072000, 9.000000
1, 7, 23, 24, 0.002200, 0.000001, 0.072000, 9.000000
1, 8, 24, 25, 0.002200, 0.000001, 0.072000, 9.000000
2, 1, 1, 2, 0.002200, 0.000001, 0.072000, 9.000000
2, 2, 2, 3, 0.002200, 0.000001, 0.072000, 9.000000
2, 3, 3, 4, 0.002200, 0.000001, 0.072000, 9.000000
2, 4, 4, 5, 0.002200, 0.000001, 0.072000, 9.000000
```

CONCLUSÃO

O volume de dados que o programa consegue trabalhar depende do equipamento em que estiver sendo executado. Em um computador pessoal Pentium 100 MHz com 16 MB de RAM uma malha retangular composta por 10.000 elementos finitos de placa quadrilaterais e 10.201 pontos nodais demora 1 minuto e 15 segundos para ser gerada. No mesmo equipamento, uma malha retangular composta por 20.000 elementos finitos de placa triangulares e 10.201 pontos nodais demora 1 minuto e 44 segundos para ser gerada. Isto porque o aplicativo pode utilizar toda a memória convencional disponível no equipamento em que estiver sendo executado e o limite do tamanho das matrizes que forem utilizadas no aplicativo estará relacionado à quantidade de memória RAM disponível e ao tamanho do arquivo de troca que o Windows estiver utilizando.

Em Sousa Junior [2] a interface do aplicativo PEC com o usuário demostrou ser de fácil utilização além de oferecer as ferramentas necessárias para definir e manipular malhas de elementos finitos. A eficiência do Gerador de Malha foi comprovada com a geração da malha de vários pavimentos de formas variadas que continham curvas, vigas em diagonal e lajes nervuradas.

Como continuidade deste trabalho, pode-se sugerir o desenvolvimento de um assistente para o Gerador de Relatórios, ou seja, um programa ou uma janela que possa gerar um arquivo de formato contendo comandos do Gerador de Relatórios em que o usuário não precise saber o nome das variáveis do sistema e utilize apenas o *mouse* para definir a ordem e a posição dos campos na folha do relatório. Uma outra sugestão, seria aperfeiçoar o Gerador de Relatórios de tal forma que ele pudesse gerar arquivos de formato de leitura de dados, para serem utilizados pelo aplicativo PEC, para a leitura de arquivos de dados gerados por outros *softwares*. Isso permitiria um aproveitamento de dados de pavimentos já analisados. E que fossem acrescentados ao PEC recursos de rotação do objeto na tela gráfica para facilitar sua visualização em três dimensões, outros tipos de geradores de malha e ainda implementá-lo para a aquisição de dados para a análise de chapas.

REFERÊNCIAS

1 SOUSA JUNIOR, E.; PAIVA, J. B. de (1996)- *Gerador de Malha de Elementos Finitos como um Aplicativo Windows*. In: CONGRESSO TÉCNICO-CIENTÍFICO DE ENGENHARIA CIVIL, Florianópolis, *Anais*. Florianópolis, UFSC, v.4, p.718-727

2 SOUSA JUNIOR, E. (1996). *Um aplicativo para o ambiente Windows para aquisição de dados para análise de pavimentos de edifícios via Método dos Elementos Finitos*. São Carlos. Dissertação (Mestrado). EESC, USP.