MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

INTERFACE DE CÁLCULO MODAL E VISUALIZAÇÃO PARA ESTRUTURAS FLEXÍVEIS TIPO EULER-BERNOULLI

por

Alcindo José Dal Piva

Dissertação para a obtenção do Grau de
Mestre em Matemática Aplicada

Porto Alegre

Junho de 2003

INTERFACE DE CÁLCULO MODAL E VISUALIZAÇÃO PARA ESTRUTURAS FLEXÍVEIS TIPO EULER-BERNOULLI

por

Alcindo José Dal Piva

Dissertação submetida ao Corpo Docente do Programa de Pós-Graduação em Matemática Aplicada, PPGMAp, do Instituto de Matemática da Universidade Federal do Rio Grande do Sul, como parte dos requisitos necessários para a obtenção do Grau de

Mestre em Matemática Aplicada

Linha de Pesquisa: Vibrações, Controle e Sinais

Orientador: Prof. Dr.Julio Cesar Ruiz Claeyssen

Aprovada por:

Prof. Dr.Ausberto Silvério Castro Vera (UNASP)

Prof. Dr.Rudnei Dias da Cunha (PPGMAp/UFRGS)

Prof. Dr.Vilmar Trevisan (PPGMAp/UFRGS)

Prof. Dr. Vilmar Trevisan

Coordenador do PPGMAp

Porto Alegre, Junho de 2003

RESUMO

Neste trabalho, foram analisadas, implementadas e mostradas as características de uma interface com estrutura de programação genérica para o ambiente Windows, criando facilidades de rapidez, confiabilidade e a apresentação de resultados aos usuários do sistema matemático Maple na criação e uso de aplicações matemáticas.

A interface utilizou como modelo de implementação cálculos modais de vigas clássicas de Euler-Bernoulli. O usuário encontra, em um único sistema, cálculo para vigas clássicas, terá uma entrada de dados facilitada de variáveis que serão substituídas automaticamente no programa fonte da viga e a geração de resultados em um ambiente amigável com dados e gráficos mostrados de forma organizados.

ABSTRACT

TITLE: "INTERFACE OF MODAL CALCULATION AND VIEW FOR STRUCTURES FLEXIBLE TYPE EULER-BERNOULLI"

In this work, we analise, implement and shown the characteristics of an interface with generic programming structure for the Windows Operating System, creating means of speed, reliability and result presentation for users of the mathematical system Maple in the development and use of mathematical applications.

The interface used as implementation model is modal calculations of classic beams of Euler-Bernoulli. The user finds, in a single system, classic beams, he will also the imput of data of variables which will be replecent automatically in the program source of the beam and the generation of results in a friendly atmosphere with data and visualization graphs in organized way.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1 INTRODUÇÃO

O usuário do sistema matemático Maple, quando desenvolve uma nova aplicação ou manipula aplicações matemáticas existentes, freqüentemente vivencia a seguinte situação:

Os programas fontes das aplicações, que são linhas de comandos em formato texto contendo instruções ao computador para fazer algo, geralmente estão soltos ou perdidos no disco rígido do computador.

A cada novo cálculo que desejar é necessário localizar o programa fonte, abri-lo e executar. Quando desejar usar novos valores de variáveis para uma nova situação, é necessário editorar o programa fonte e substituir manualmente as variáveis específicas e rodar novamente para recalcular a nova situação desejada.

Após o cálculo ter sido efetuado a visualização dos resultados se torna embaraçada, sendo geralmente necessária a rolagem de tela para a procura dos dados de interesse do cálculo.

Para resolver as situações apresentadas, foi desenvolvida neste trabalho uma interface que possui uma estrutura genérica, podendo ser adaptada a qualquer situação de aplicação matemática do Maple desde aplicações com a realização de cálculos simbólicos, tais como derivação, integração, simplificação, etc ou, simplesmente, o cálculo numérico. Em ambos os casos, a interface utiliza a automação na substituição das variáveis genéricas pré-fixadas nos programas fontes das aplicações por constantes numéricas lidas nos formulários de entrada de dados.

Neste estudo foi demonstrada a viabilidade e os recursos da interface através da implementação de um modelo, sendo escolhido o problema de cálculo modal de estruturas flexíveis clássicas de Euler-Bernoulli, por tratar-se de uma aplicação importante na matemática, engenharia civil e engenharia mecânica e pelo

fato de que os problemas vivenciados pelos usuários destas áreas são muito aparentes nestas situações.

Para a implementação da interface foi necessária uma análise detalhada do funcionamento do aplicativo matemático Maple, no que se refere a manuseio de arquivos e à lógica de programação dos programas fontes das aplicações. Utilização de uma linguagem de programação com programação em nível de objetos e manipulação de arquivos facilitada e no tratamento com o Sistema Operacional Windows através das funções da API - (Application Program Interface), que é o conjunto de funções que gerenciam o relacionamento entre a interface criada e o Sistema Operacional Windows.

A seguir, far-se-á uma breve descrição dos capítulos subseqüentes que formam este trabalho.

No capítulo 2, é descrito o problema; é contextualizado o Maple; é mostrada a necessidade da criação de uma interface para que a relação homem-máquina seja facilitada; métodos de conversão do programa fonte; proposta da alteração automatizada do programa fonte das vigas.

No capítulo 3, é apresentado o sistema através do Diagrama de Fluxo de Dados, da descrição de seus fluxos; de seus arquivos; das suas variáveis de entrada e saída e seus gráficos.

No capítulo 4, é apresentado o cálculo modal em vigas mono-segmentadas, a equação de Euler-Bernoulli; estudo da equação modal; resolução da equação modal e vibração forçada.

No capítulo 5, é mostrada a configuração do sistema; simulações e programas fontes das vigas e, por fim, far-se-ão as conclusões pertinentes.

2 ANÁLISE DO PROBLEMA

O usuário do sistema matemático Maple, ao desenvolver ou utilizar programas fontes de aplicações matemáticas, depara-se com problemas, tais como os programas criados ficam soltos ou perdidos no disco rígido do microcomputador. Tem-se que, para uma nova situação de cálculo, editorar o programa fonte da aplicação, fazer alterações de valores de variáveis e recalcular; os resultados mostrados dos cálculos ficam desorganizados, sendo necessária a rolagem de tela para poder visualizar melhor os resultados.

2.1 O Maple

O Maple é um sistema matemático que suporta uma linguagem de programação funcional própria e um mecanismo de construção e distribuição de pacotes de procedimentos e funções. É um software largamente utilizado para a computação científica, nas áreas da matemática, física, computação etc, por apresentar cálculos simbólicos, numéricos e gráficos. Por ele ser um sistema matemático que além de resolver problemas numéricos, realiza operações simbólicas tais como: derivação, integração, fatoração, etc e trata seus programas de aplicações como programas interpretados para a execução de uma instrução.

Entende-se que o Maple tem um duplo papel educacional, instrumento do projeto e objeto de pesquisa. Do ponto de vista instrumental, o Maple pode ser considerado como um ambiente para o exercício e a exploração computacional dos temas matemáticos clássicos, tratados em disciplinas como Cálculo, Geometria Analítica, Álgebra Linear etc.

Neste sentido o Maple é uma ferramenta central para a integração curricular da coleção de matemática e fundamentos teóricos, porque possibilita aos

professores e alunos a realização de um imenso conjunto de práticas educacionais e laboratoriais em um ambiente computacional único.

Quanto ao Maple como objeto de pesquisa, enfoca-se a possibilidade da utilização cooperativa do mesmo. Observa-se que o Maple foi concebido para a utilização individual, praticamente sem nenhum suporte para o uso cooperativo. O reconhecimento desta deficiência por parte do fabricante começa a ocorrer nas versões mais recentes do Maple já oferecem a possibilidade de geração automática de páginas em HTML, contendo resultados das sessões realizadas, para a distribuição na WEB.

2.2 Necessidade de uma Interface

Para que a relação homem-máquina ocorra, é indispensável o uso das interfaces e da interatividade. Sem estes dois fundamentos, é impossível haver qualquer tipo de relação homem-máquina em um ambiente computacional. Esta visão é representada na figura 2.1 que mostra o Maple carregado na memória do computador em um nível mais elevado e disponível para ser utilizado por um usuário. O sistema operacional apresenta-se em um nível mais baixo, pois tem a função de fazer o gerenciamento e a comunicação entre o Maple e o hardware,

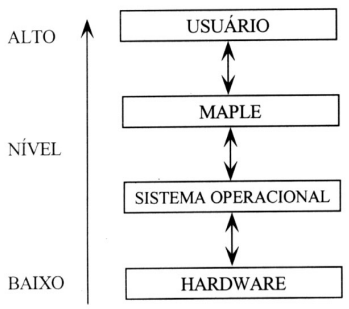

Figura 2.1 *Visão do Maple na Memória do Computador*

Este trabalho tem a finalidade de criar, no mesmo nível de memória de trabalho do computador, uma interface capaz de facilitar o cálculo modal e visualização para estruturas flexíveis do tipo Euler-Bernoulli, que será cálculada no Maple, mostrada na figura 2.2.

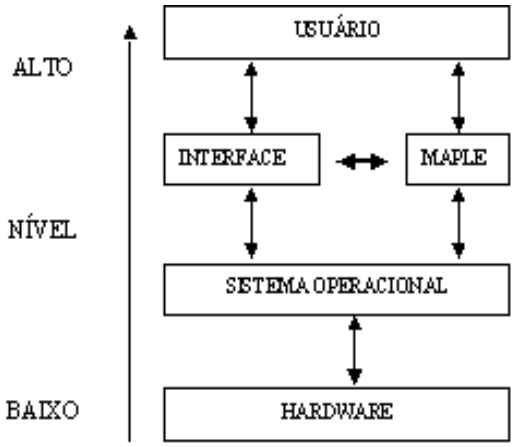

Figura 2.2 *Visão da Interface na Memória do Computador*

O computador deve converter os comandos dados em linguagem de alto nível para linguagem de máquina (códigos binários). Esta tarefa de conversão é feita por um programa especial de computador, isto é, um programa que recebe as instruções em linguagem de alto nível e dá como saída outro programa constituído de instruções binárias. Ao programa original, em linguagem de alto nível, dá-se o nome de programa fonte, e ao resultado, em linguagem de máquina, de programa objeto.

2.3 Métodos de Conversão

Existem dois métodos básicos de abordagem na conversão de linguagem de alto nível para linguagem de máquina, sendo o fluxo mostrado na figura 2.3.

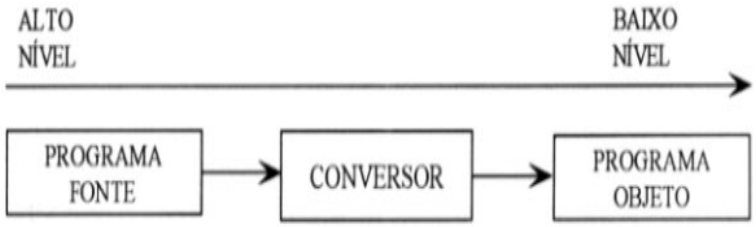

Figura 2.3 *Fluxo de Conversão*

2.3.1 Interpretador

Neste método o programa conversor é chamado de interpretador e recebe a primeira instrução do programa fonte, confere para ver se está escrita corretamente, converte-a em linguagem de máquina, e então, ordena ao computador que execute esta instrução. Depois repete o processo para a segunda instrução e, assim, sucessivamente, até a última instrução do programa fonte. Quando a segunda instrução é trabalhada, a primeira é perdida, isto é, apenas uma instrução fica na memória em cada instante. Se este programa fonte for executado uma segunda vez, novamente haverá uma nova tradução, comando por comando, pois os comandos em linguagem de máquina não ficam armazenados para futuras execuções. O aplicativo matemático Maple utiliza o método de interpretação para a execução de suas instruções.

2.3.2 Compilador

Neste método, o programa conversor é chamado de compilador e recebe a primeira instrução do programa fonte, confere-a para ver se está escrita corretamente, converte-a para linguagem de máquina em caso afirmativo e passa para a próxima instrução, repetindo o processo, sucessivamente, até a última instrução do programa fonte. Caso tenha terminado a transformação da última instrução do programa fonte e nenhum erro tenha sido detectado, o computador volta à primeira

instrução, já transformada para linguagem de máquina e executa-a. Passa à instrução seguinte, executa-a etc., até a última. Se este programa for executado uma segunda vez, não haverá necessidade de uma nova tradução, uma vez que todos os comandos em linguagem binária foram memorizados em um novo programa completo. A vantagem, neste processo, é que a execução fica mais rápida em relação à anterior, pois se economiza o tempo de retradução de cada instrução a cada nova execução e a desvantagem é que a cada modificação introduzida no programa fonte é necessária uma nova tradução completa para obter um novo programa objeto, o que torna o processo mais dificultoso na fase de desenvolvimento, quando muitas modificações são feitas. Pode-se citar algumas linguagens de programação tais como: Pascal, Cobol, C, Fortran que utilizam compiladores na geração de seu programa objeto.

2.4 Alteração Automatizada

Neste estudo, mostra-se a facilidade que a interface irá desempenhar frente ao usuário, pois o Maple é um interpretador e, a cada vez que se queira executar um programa fonte de cálculo de vigas, é necessário abrir o programa, executá-lo com suas variáveis fixadas ou substituir por novas.

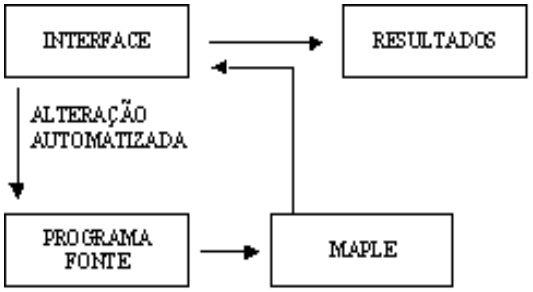

Figura 2.4 *Alteração Automatizada do Programa Fonte*

Com a utilização da interface erros de editoração dos programas fontes das aplicações são eliminados, pois a função da alteração automatizada é fazer as sub-

stituições das variáveis predefinidas pelo sistema automaticamente, conforme figura 2.4, evitando que o usuário as faça, conseqüentemente, aumentando a confiabilidade.

3 DESCRIÇÃO DO SISTEMA

Esta imagem representa o fluxo integrado do sistema, representando em alto nível a utilização da interface pelo usuário desde a entrada de dados até a saída dos resultados.

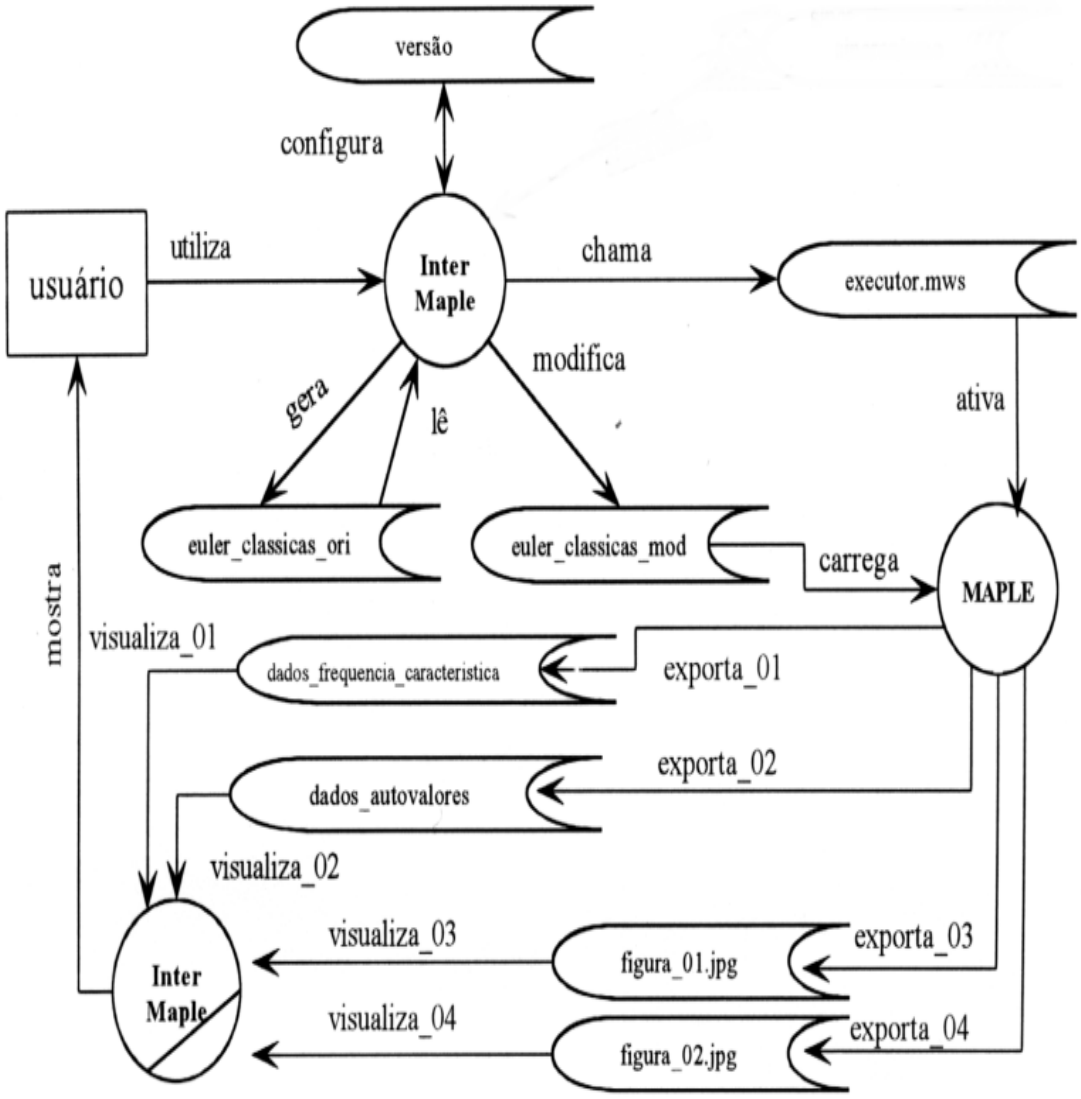

Figura 3.1 *Diagrama de Fluxo de Dados*

3.1 Descrição dos Fluxos de Dados

A lista a seguir mostra o nome do fluxo com sua descrição, objetivando o entendimento do sistema.

- Fluxo Utiliza

  O usuário está carregando na memória a interface para uso, onde selecionará o tipo de viga desejada e dará a entrada nas variáveis e mandará calcular no Maple.

- Fluxo Gera

  O processo intermaple gera em disco um arquivo fonte no formato texto chamado `euler_classicas_ori`, para cada tipo de viga selecionada que contém variáveis implantadas.

- Fluxo Lê

  Após ter sido gerado o arquivo `euler_classicas_ori` o processo intermaple começa a ler este arquivo.

- Fluxo Modifica

  O processo intermaple faz a geração do arquivo no formato texto chamado `euler_classicas_mod`, a partir do arquivo `euler_classicas_ori` substituindo as variáveis inicialmente implantadas por valores numéricos lidos no formulário de entrada do processo intermaple.

- Fluxo Chama

  O processo intermaple faz a leitura do programa fonte chamado executor.mws gerado pelo Maple.

- Fluxo Ativa

  No sistema windows arquivos chamam seu gerador pela sua extensão e, neste caso, automaticamente, o Maple é carregado na memória. Com o

arquivo executor.mws carregado, que contém instruções para a execução do arquivo `euler_classicas_mod`.

- Fluxo Carrega

  Após o Maple ter sido ativado o arquivo carregado executor.mws é rodado, automaticamente, iniciando a execução do arquivo `euler_classica_mod` calculando a viga.

- Fluxo `Exporta_01`

  O processo maple faz a exportação de um arquivo texto chamado `dados_frequencia_caracteristica`, que contém as freqüências características calculadas.

- Fluxo `Exporta_02`

  O processo maple faz a exportação de arquivo texto chamado `dados_autovalores`, que contém os autovalores calculados.

- Fluxo `Exporta_03`

  O processo maple faz a exportação do gráfico chamado `figura_01.jpg`, que contém os modos calculados.

- Fluxo `Exporta_04`

  O processo maple faz a exportação do gráfico chamado `figura_02.jpg`, que contém a vibração forçada.

- Fluxo `Visualiza_01`

  O processo intermaple fará a abertura do arquivo chamado `dados_frequencia_característica`.

- Fluxo `Visualiza_02`

  O processo intermaple fará a abertura do arquivo `dados_autovalores`.

- Fluxo `Visualiza_03`

  O processo intermaple fará a abertura do arquivo `figura_01.jpg`

- Fluxo `Visualiza_04`

  O processo intermaple fará a abertura do arquivo `figura_02.jpg`.

- Fluxo Visualiza

  Após ter sido calculada a viga com os valores numéricos definidos no programa fonte e exportados os arquivos necessários, o processo intermaple faz a leitura de todos os arquivos exportados pelo Maple e os mostra em um formulário de saída ao usuário para a sua análise.

- Fluxo Configura

  Pelo processo intermaple o usuário pode definir a versão mais atualizada do Maple instalada em seu computador, para o correto funcionamento do sistema.

3.2 Descrição dos Arquivos

A lista a seguir tem a finalidade de descrever os arquivos utilizados no sistema, mostrando seu formato, nome e descrição.

- Arquivo texto `euler_classicas_ori` contendo instruções do Maple com as variáveis implantadas.

- Arquivo texto `euler_classicas_mod` contendo instruções do Maple já com constantes numéricas nos locais das variáves implantadas que foram lidas no formulário de entrada de dados.

- Arquivo texto `dados_frequencia _caracteristica` contém as freqüências características calculadas e exportado pelo Maple.

- Arquivo texto `dados_autovalores` contendo os autovalores calculados e exportados pelo Maple.

- Arquivo executor.mws é um programa fonte do Maple que contém instruções para carregar o arquivo `euler_classicas_mod` pela interface.
- Arquivo de imagem `figura_01.jpg` no formato jpg, contendo os modos calculados pelo Maple.
- Arquivo de imagem `figura_02.jpg` no formato jpg, contendo a vibração forçada.
- Arquivo texto versão.ini, contém a versão mais recente instalada e configurada pelo usuário da interface.

3.3 Descrição das Variáveis

3.3.1 Variáveis de Entrada

O sistema foi programado para a entrada de nove variáveis, que já vêm com valores em default para facilitar a digitação que são descritas a seguir.

- O número de modos que está diretamente relacionado com a quantidade de freqüências características e os autovalores resultantes. Há possibilidade de simular com até 7 modos.
- O comprimento da viga deverá ser maior que zero.
- A rigidez flexural é a elasticidade do material, é a capacidade de voltar ao seu estado normal. O valor deverá ser maior que zero.
- A massa linear deverá ter seu valor maior que zero.
- A freqüência de entrada é em radianos, sendo a freqüência inicial que a viga irá sofrer no início do cálculo. Poderá ter seu valor igual a zero.
- A magnitude da força externa é a força em nexton que a viga irá receber na simulação, podendo ter seu valor igual a zero.

- A posição da carga será o local da viga que a força externa irá exercer a sua força, podendo o valor ser zero ou até o comprimento da viga.
- A força axial pode ser de compressão ou tensão, podendo ter seu valor igual a zero.
- O tempo de vibração que a viga irá sofrer no tempo em segundos, visa a facilitar a visualização da simulação no tempo. O valor deverá ser maior que zero.

3.3.2 Variáveis de Saída

O sistema produz como resultado as freqüências características e os autovalores que são resultados da simulação efetuada, utilizando as variáveis de entrada definidas pelo usuário.

3.4 Gráficos de Saída

Para a melhor visualização do resultado da simulação são apresentados dois gráficos.

- Gráfico dos Modos, que mostra o comportamento dos modos simulados no tempo.
- Gráfico da Vibração Forçada, que mostra o comportamento de um dos modos no tempo, sendo que este modo assume o padrão da freqüência resultante com o valor mais próximo da freqüência de entrada.

3.5 Implantando e Substituindo Variáveis

A interface gera um arquivo texto chamado `euler_classicas_ori`, o qual teve seu nome definido por se tratar das vigas clássicas de Euler-Bernoulli e ser o arquivo original. Neste arquivo foram implantadas variáveis que estão substituindo valores numéricos do programa original do Maple em condições normais. Estas variáveis que substituíram esses valores numéricos, são exatamente as que iriam influenciar em uma substituição manual pelo usuário para ser possível efetuar um novo cálculo do Maple.

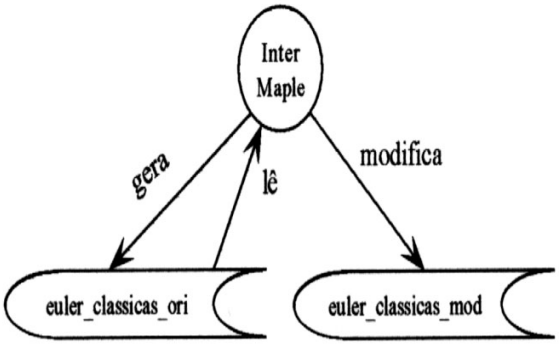

Figura 3.2 *Substituição das Variáveis e Geração do Arquivo Modificado*

O arquivo gerado `euler_classicas_ori` é aberto pela interface e lido linha por linha, procurando as variáveis implantadas e substituindo-as exatamente por valores lidos a partir de um formulário de entrada de dados do sistema, gerando outro arquivo texto chamado de `euler_classicas_mod`, conforme mostrado na figura 3.2.

Este exemplo é parte do arquivo `euler_classicas_ori` do tipo de viga fixa livre nas linhas de comando que faz a criação dos arquivos, calcula os autovalores e a freqüência característica e faz a gravação nos arquivos criados e possuem variáveis implantadas podendo ser visualizadas, tendo sua representação na tabela 3.1.

Nome da Variável	Representação
INT_01	Número de Modos ou Quantidade de Gráficos
INT_02	Comprimento da Viga
INT_03	Rigidez Flexural
BOL_01	Massa Linear
BOL_03	Frequência de Entrada
BOL_04	Magnitude da Força Externa
BOL_05	Posição da Carga
BOL_06	Força Axial
BOL_07	Tempo de Vibração

Tabela 3.1 *Descrição das Variáveis Implantadas*

```
>FD1 := fopen("C:/VIGAS/DADOS_AUTOVALORES",WRITE,TEXT);
>FD2 := fopen("C:/VIGAS/DADOS_FREQUENCIA_CARACTERISTICA",WRITE,TEXT);
>for n from 1 to INT_01 do
>bbeta[n]:=fsolve(eq_car/beta^4,beta=(2*n-1)*Pi/2/INT_02);
>vetor_01 := array([bbeta[n]]);
>writedata(FD1,vetor_01);
>ww[n]:=evalf(bbeta[n]^2*sqrt( INT_03 / mL ));
>vetor_02 := array([ww[n]]);
>writedata(FD2,vetor_02);
>modo_d[n] := subs(beta=bbeta[n],modos);
>norma_d[n]:= evalf(sqrt(simpson(modo_d[n]^2,x=0..INT_02,200)));
>modo_d[n] := modo_d[n]/norma_d[n];
>od:
>fclose(FD1);
>fclose(FD2);
```

3.6 Ativando o Maple

Para a inicialização do Maple, a partir da interface, é necessária a utilização da função ShellExecute da API - (Application Program Interface) do Win-

dows, que tem a função de abrir um arquivo relacionado com a função, que automaticamente abre o programa Maple. O arquivo relacionado é o executor.mws e pode ser visto na figura 3.3.

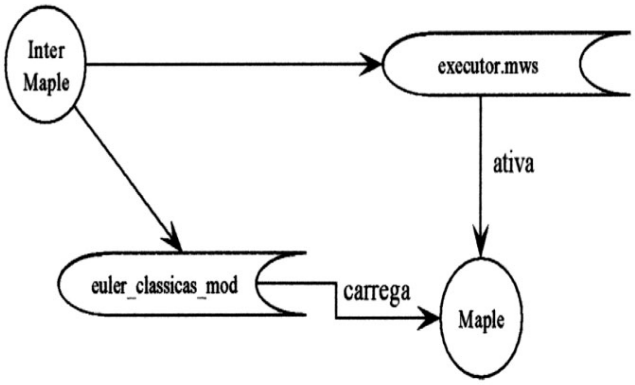

Figura 3.3 *Ativando o Maple*

3.7 Funções do Maple Utilizadas

Para ser possível o funcionamento e a integração da interface com o Maple, foi necessário inserir funções nos programas fontes do Maple como funções de exportações de dados em forma de arquivos textos e figuras, para, posteriormente, serem lidas pela interface e mostrado o seu resultado dos cálculos ao usuário final.

3.7.1 Função Read

A seguir é mostrado o conteúdo do programa fonte do Maple executor.mws.

```
>restart;
>read "c:/vigas/EULER_CLASSICAS_MOD";
```

Na primeira linha são zeradas todas as variáveis de sistema e na segunda linha a função read que tem a finalidade de ler o nome do arquivo em disco que está como parâmetro da função onde os comandos do arquivo são lidos e executados até a última linha de comando.

Neste caso, o arquivo que vai ser lido é o arquivo `euler_classica_mod`, que já foi inserido os valores numéricos e está pronto para ser executado pelo Maple.

3.7.2 Função Open

A utilização da função open tem a finalidade de abrir um arquivo com o nome especificado em disco em modo de leitura e gravação. Caso o arquivo esteja sendo aberto em modo leitura "read" e o mesmo não exista, a função open vai gerar um erro de abertura. Se o arquivo que está sendo aberto em modo de escrita "write" e o mesmo não exista ele será criado, sendo esta forma adotada no sistema.

```
>FD1:= fopen("C:/VIGAS/DADOS_AUTOVALORES",WRITE,TEXT);
>FD2:=fopen("C:/VIGAS/DADOS_FREQUENCIA_CARACTERISTICA",WRITE,TEXT);
```

Na primeira linha de comando está sendo aberto um arquivo texto no disco c: na pasta vigas com o nome de `DADOS_AUTOVALORES`, onde será armazenados os autovalores gerados pelo cálculo efetuado pelo Maple, na segunda linha está sendo aberto o arquivo texto no disco c: na pasta vigas, com o nome de `DADOS_FREQUENCIA_CARACTERISTICA`, onde serão armazenados os valores das freqüências características efetuados pelo Maple.

A criação de arquivos é feita a cada novo tipo de cálculo efetuado, pelo motivo de não ser necessário o armazenamento destes dados gerados, eles somente servem para serem mostrados em um formulário de saída.

3.7.3 Função Writedata

A função writedata escreve dados numéricos ou textos em um vetor, matriz ou lista, onde os dados estão em um vetor, cada valor é impresso em linhas separadas.

```
>vetor_01 := array([bbeta[n]]);
>writedata(FD1,vetor_01);
>ww[n]:=evalf(bbeta[n]^2*sqrt( INT_03 / mL ));
>vetor_02 := array([ww[n]]);
>writedata(FD2,vetor_02);
```

Geralmente os números consistem em números inteiros ou de ponto flutuante. A função writedata planeja simplificar o trabalho para escrever vetores e matrizes dentro de um arquivo. Por default, a função writedata cria um novo arquivo cada vez que é chamada a menos que o arquivo já esteja aberto pela função Fopen. Se o arquivo já está aberto o writedata escreve na posição dentro do arquivo.

Os dois arquivos foram abertos anteriormente pela função Open, sendo agora possível trabalhar com eles. No trecho do programa mencionado pode-se constatar o cálculo dos autovalores e da freqüência característica. Após cada passagem pelo laço é calculado um autovalor e uma freqüência característica, exportado através da função writedata para o arquivo direcionado na variável de arquivo FD1 e FD2 e gravado linha a linha e no formato texto.

3.7.4 Função Fclose

Sempre que um arquivo é aberto pela função Fopen, é necessário fechá-lo através do comando Fclose, o qual tem uma variável direcionada na abertura pela função.

```
>fclose(FD1);
>fclose(FD2);
```

Na primeira linha de comando acontece o fechamento do arquivo com o nome de `dados_autovalores` que está na unidade de disco c: e na pasta vigas que é assumido pela variável de arquivo FD1. Na segunda linha de comando acontece o fechamento do arquivo com o nome de `dados_freqüência_característica` que está na unidade de disco c: e na pasta vigas que é assumido pela variável de arquivo FD2.

3.7.5 Função Plotsetup

Como mostra a figura 3.4 o Maple tem que exportar duas figuras do cálculo efetuado para, posteriormente, serem lidas pela interface e mostradas ao usuário, mas para serem possíveis estas exportações foi utilizada a função plotsetup no programa fonte.

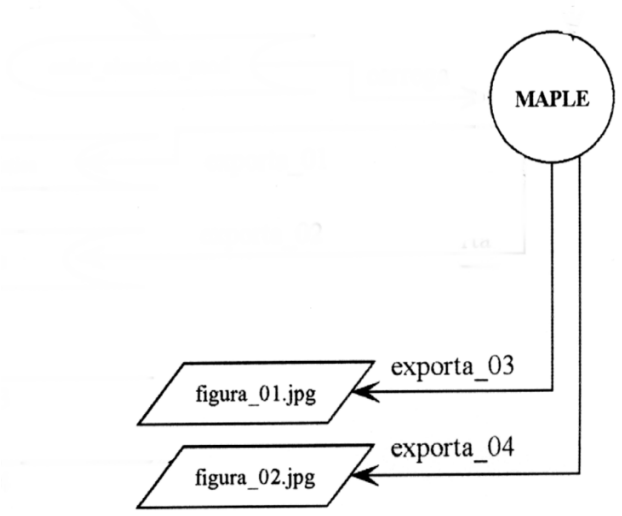

Figura 3.4 *Exportando Figuras*

Foram utilizadas 3 variáveis de interface, que controlam os dispositivos usados para a execução.

3.7.5.1 Variável Plotoutput

Variável Plotoutput foi utilizada na exportação das duas figuras geradas pelo sistema. Esta variável tem a função de criar o arquivo no endereço de disco especificado.

O trecho do programa descrito mostra a criação dos arquivos `figura_01.JPG` e do arquivo `figura_02.JPG`, que representam o arquivo de modos e da vibração forçada, no disco c: na pasta vigas.

```
Plotoutput=‘C:/VIGAS/FIGURA_01.JPG‘
Plotoutput=‘C:/VIGAS/FIGURA_02.JPG‘
```

3.7.5.2 Variável Plotoptions

Está vinculado ao comando, sobrepondo ao arquivo anteriormente gerado, que são as opções que são atribuídas à imagem que foi gerada, sua altura, largura e com bordas.

plotoptions=‘portrait,height=216,width=288,border‘

3.7.5.3 Variável Jpeg

É um dispositivo que é conhecido pelo Maple e produz uma versão de 24 bits coloridos de imagem e por default apresenta uma altura de 360 linhas e largura de 480 colunas, gerando uma imagem de 172.800 pontos. O drive JPEG permite especificar a altura e a largura do quadro usando o string Plotoptions e, neste caso, foi utilizada uma altura de 216 linhas e largura de 288 colunas, gerando uma imagem de 62.208 pontos, a qual tem um tamanho ideal para a visualização do usuário e se enquadra bem no formulário de saída de dados.

3.8 Definição da Linguagem de Programação

A interface foi programada na linguagem de programação Delphi, por apresentar programação em nível de objetos, manipulação de arquivos facilitada e tratamento com o sistema operacional Windows através das funções da API - (Application Program Interface), que é um conjunto de funções que gerenciam o relacionamento entre a aplicação criada e o Windows.

3.9 Tipos de Dados da API

3.9.1 Handles

A maioria das funções da API do windows usa um tipo de dados chamado de Handle. O windows usa handles como indexadores para se referenciar a objetos que são armazenados em uma tabela como ponteiros. Desta forma, uma aplicação windows pode se referir a uma janela ou dispositivo de contexto de seu handle.

As janelas e os dispositivos de contexto têm um tipo de handle específico. Normalmente, os handles dos componentes que o possuem é conseguido através da propriedade HANDLE.

3.9.2 Pchar

São ponteiros para caracteres. Um ponteiro é uma variável que contém um endereço de memória. Esse endereço é, normalmente a posição de uma outra varíável na memória, ou de um determinado dado. Suponha que a varíavel var1 esteja armazenada no endereço 1000 da memória. Se var2 for um ponteiro para var1, terá como conteúdo o endereço de memória var1, ou seja 1000. Note que na

declaração da variável que aponta para algum outro dado, precisa-se dar a ela o tipo de dado apontado, se var var2 é do tipo char, var1 tem que ser ponteiro para char.

3.10 Funções da API de Manuseio de Janelas

3.10.1 ShellExecute

A função tem a finalidade de abrir um determinado arquivo, nesta aplicação ela abre o arquivo executor.mws, que pertence ao Maple e, conseqüentemente, irá ativar o Maple.

3.10.2 ShowWindow

Mostra ou esconde uma janela no windows de uma certa maneira. Esta função pode minimizar, maximizar ou restaurar uma janela, nesta aplicação, após a utilização da função shellexecute ter sido executada a interface é sobreposta à janela da Maple.

3.10.3 SetForegroundWindows

Faz uma específica janela do windows ficar corrente no foco, acompanha a função ShowWindow para sobrepor a janela da interface sobre o Maple.

3.10.4 Application.ProcessMessages

Esta função tem a finalidade de dar um tempo para o processador jogar a janela que entrara em foco e pintá-la - recompor na tela.

3.11 Funções de Manuseio de Janela Utilizadas

3.11.1 GetWindowText

Faz uma cópia do texto da barra de títulos da janela especificada, para um buffer de memória.

3.11.2 GetClassName

Copia o texto do nome da classe de uma janela filha de uma classe.

3.11.3 PostMessage

Após o Maple ter sido carregado na memória com o arquivo executor.mws, ele fica aguardando um enter do teclado para o início do cálculo para a solução do problema. Para facilitar este procedimento automatizou-se este "ENTER" passado da interface para o Maple. Utilizou-se esta função que tem a finalidade de enviar uma mensagem entre aplicações o comando "ENTER" é enviado para a janela Maple onde está aberto o documento executor.mws.

Esta função também é utilizada no fechamento do Maple após ele ter sido utilizado pela interface. Na configuração do sistema pode-se optar pela versão mais recente instalada no computador do Maple, que influenciará diretamente no fechamento do mesmo, pois a interface utilizará as informações de configuração setadas.

3.12 Sincronismo entre Aplicações

Para o bom funcionamento entre a interface e o Maple foram desenvolvidos mecanismos de programação que possibilitam o perfeito sincronismo entre essas duas aplicações.

Um dos mecanismo desenvolvido é o reconhecimento do processador do computador onde está instalada a interface, que faz o auto-ajuste entre as aplicações através de temporizadores. Outro mecanismo é que não seja lido pela interface nenhum arquivo gerado pelo Maple até que todos sejam gerados e fechados.

4 CÁLCULO MODAL EM VIGAS MONO-SEGMENTADAS

Neste capítulo, consideram-se vigas com distribuições uniformes de massa e rigidez, sujeitas a uma força axial resultante, governadas pela equação de Euler-Bernoulli.

4.1 Equação de Euler-Bernoulli

A equação de Euler-Bernoulli é estabelecida a partir do balanço de forças e momentos em um elemento diferencial de uma viga, desprezando os efeitos de força de cisalhamento e inércia de rotação. O enunciado matemático de tal equação com seção transversal uniforme é:

$$\boxed{\rho A\frac{\partial^2 u(t,x)}{\partial t^2} + N\frac{\partial^2 u(t,x)}{\partial x^2} + EI\frac{\partial^4 u(t,x)}{\partial x^4} = p(t,x), \quad 0 < x < L, \quad t > 0,} \tag{4.1}$$

onde

- t = tempo, $t \geq 0$,
- x = posição de um ponto da viga, $0 \leq x \leq L$,
- $u(x,t)$ = deslocamento da viga no tempo t e posição x,
- ρA = massa por unidade de comprimento da viga,
- EI = rigidez flexural da viga,
- $p(x,t)$ = força aplicada,
- L = comprimento da viga,
- N = força axial da viga, que pode ser de compressão ou de tensão

e as unidades físicas estão dadas no sistema *MKS*.

As diversas grandezas envolvidas podem ser observadas na figura 4.1

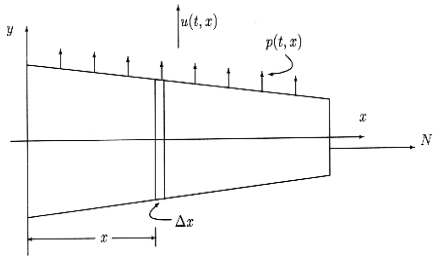

Figura 4.1 *Viga de Euler-Bernoulli com força axial*

4.1.1 Estudo da Equação Modal

A procura de soluções de tipo harmônico

$$u(t,x) = e^{i\omega t}X(x) \tag{4.2}$$

para

$$\rho A\frac{\partial^2 u(t,x)}{\partial t^2} + N\frac{\partial^2 u(t,x)}{\partial x^2} + EI\frac{\partial^4 u(t,x)}{\partial x^4} = 0, \tag{4.3}$$

resulta na *equação modal* da viga,

$$\boxed{EI\frac{d^4X(x)}{dx^4} + N\frac{d^2X(x)}{dx^2} - \rho A\omega^2 X(x) = 0,} \tag{4.4}$$

onde o parâmetro ω é denominado a *freqüência característica* da estrutura.

Na tabela 4.1 são mostradas as expressões matemáticas para as condições de contorno clássicas, bem como o esquema físico que representa tal condição.

Cada configuração de viga fornece quatro condições de contorno como é mostrado na tabela 4.3, levando em conta as siglas utilizadas na tabela 4.2.

Contorno	Equações	Esquema Físico
Fixo	$u(t,0) = 0, \quad \frac{\partial u}{\partial x}(t,0) = 0$ deslocamento e giro nulos	
Apoiado	$u(t,0) = 0, \quad K(x)\frac{\partial^2 u}{\partial x^2}(t,0) = 0$ deslocamento e momento fletor nulos	
Deslizante	$\frac{\partial u}{\partial x}(t,0) = 0, \quad \frac{\partial}{\partial x}\left(K(x)\frac{\partial^2 u}{\partial x^2}\right)(t,0) = 0$ giro e força de cisalhamento nulos	○ ○ ○ ○
Livre	$K(x)\frac{\partial^2 u}{\partial x^2}(t,0) = 0, \quad \frac{\partial}{\partial x}\left(K(x)\frac{\partial^2 u}{\partial x^2}\right)(t,0) = 0$ momento fletor e força de cisalhamento nulos	

Tabela 4.1 *Condições de Contorno Clássicas*

Condição	Fixa	Apoiada	Deslizante	Livre
Sigla	F	A	D	L

Tabela 4.2 *Siglas para as Condições de Contorno Clássicas*

Caso	Extremo $x = 0$	Extremo $x = L$
F-F	$X(0) = 0, \quad X'(0) = 0$	$X(L) = 0, \quad X'(L) = 0$
F-A	$X(0) = 0, \quad X'(0) = 0$	$X(L) = 0, \quad \mathcal{M}(L) = 0$
F-D	$X(0) = 0, \quad X'(0) = 0$	$X'(L) = 0, \quad \mathcal{V}(L) = 0$
F-L	$X(0) = 0, \quad X'(0) = 0$	$\mathcal{M}(L) = 0, \quad \mathcal{V}(L) = 0$
A-A	$X(0) = 0, \quad \mathcal{M}(0) = 0$	$X(L) = 0, \quad \mathcal{M}(L) = 0$
A-D	$X(0) = 0, \quad \mathcal{M}(0) = 0$	$X'(L) = 0, \quad \mathcal{V}(L) = 0$
A-L	$X(0) = 0, \quad \mathcal{M}(0) = 0$	$\mathcal{M}(L) = 0, \quad \mathcal{V}(L) = 0$
D-D	$X'(0) = 0, \quad \mathcal{V}(0) = 0$	$X'(0) = 0, \quad \mathcal{V}(0) = 0$
D-L	$X'(0) = 0, \quad \mathcal{V}(0) = 0$	$\mathcal{M}(0) = 0, \quad \mathcal{V}(0) = 0$
L-L	$\mathcal{M}(0) = 0, \quad \mathcal{V}(0) = 0$	$\mathcal{M}(0) = 0, \quad \mathcal{V}(0) = 0$

Tabela 4.3 *Condições de Contorno para a Equação Modal para uma Viga com Condições de Contorno Clássicas*

onde

- $X(0)$ e $X(L)$ são os deslocamentos em $x = 0$ e $x = L$, respectivamente;
- $X'(0)$ e $X'(L)$ são os giros em $x = 0$ e $x = L$, respectivamente;
- $\mathcal{M}(0)$ e $\mathcal{M}(L)$ são os momentos fletores em $x = 0$ e $x = L$, respectivamente; e
- $\mathcal{V}(0)$ e $\mathcal{V}(L)$ são as forças de cisalhamento em $x = 0$ e $x = L$, respectivamente.

4.1.2 Resolução da Equação Modal

A equação (4.4) pode ser escrita na forma paramêtrica como

$$\frac{d^4X(x)}{dx^4} + g^2\frac{d^2X}{dx^2}(x) - \beta^4 X(x) = 0 \tag{4.5}$$

onde

$$g^2 = \frac{N}{EI} \tag{4.6}$$

e o parâmetro β, denominado *autovalor* esta definido como

$$\beta^4 = \omega^2\frac{\rho A}{EI}. \tag{4.7}$$

Tem-se que a solução da equação modal, junto com a condição de contorno impulsiva

$$\begin{aligned} &\frac{d^4X(x)}{dx^4} + g^2\frac{d^2X(x)}{dx^2} - \beta^4 X(x) = 0, \\ &X(0) = X'(0) = X''(0) = 0, \quad X'''(0) = 1 \end{aligned} \tag{4.8}$$

está dada por [SOD 00]

$$\varphi(x) = \frac{-\frac{\text{sen}(\delta x)}{\delta} + \frac{\text{senh}(\varepsilon x)}{\varepsilon}}{\varepsilon^2 + \delta^2}, \tag{4.9}$$

onde

$$\varepsilon = \sqrt{\sqrt{\beta^4 + \frac{g^4}{4}} - \frac{g^2}{2}} \tag{4.10}$$

e

$$\delta = \sqrt{\varepsilon^2 + g^2}. \tag{4.11}$$

A vantagem desta solução é que a partir dela podem ser obtidas, facilmente outras soluções linearmente independentes da equação modal (4.4), simplesmente derivando a equação (4.9). Portanto, o conjunto

$$\left\{\varphi(x), \varphi'(x), \varphi''(x), \varphi'''(x)\right\} \tag{4.12}$$

forma uma base de funções para o espaço de soluções da equação modal (4.4). Isto é, toda solução da equação modal, $X(x)$ pode ser escrita como

$$X(x) = c_1\varphi(x) + c_2\varphi'(x) + c_3\varphi''(x) + c_4\varphi'''(x). \tag{4.13}$$

As quatro condições de contorno podem ser expressas mediante o sistema linear

$$\mathbf{B\Phi c} = \mathbf{0} \tag{4.14}$$

onde

$$\mathbf{B} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & A_{24} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & B_{11} & B_{12} & B_{13} & B_{14} \\ 0 & 0 & 0 & 0 & B_{21} & B_{22} & B_{23} & B_{24} \end{pmatrix}, \tag{4.15}$$

$$\mathbf{\Phi} = \begin{pmatrix} \varphi(0) & \varphi'(0) & \varphi''(0) & \varphi'''(0) \\ \varphi'(0) & \varphi''(0) & \varphi'''(0) & \varphi^{(iv)}(0) \\ \varphi''(0) & \varphi'''(0) & \varphi^{(iv)}(0) & \varphi^{(v)}(0) \\ \varphi'''(0) & \varphi^{(iv)}(0) & \varphi^{(v)}(0) & \varphi^{(vi)}(0) \\ \varphi(L) & \varphi'(L) & \varphi''(L) & \varphi'''(L) \\ \varphi'(L) & \varphi''(L) & \varphi'''(L) & \varphi^{(iv)}(L) \\ \varphi''(L) & \varphi'''(L) & \varphi^{(iv)}(L) & \varphi^{(v)}(L) \\ \varphi'''(L) & \varphi^{(iv)}(L) & \varphi^{(v)}(L) & \varphi^{(vi)}(L) \end{pmatrix}, \tag{4.16}$$

e

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} \tag{4.17}$$

sendo A_{ij} (B_{ij}) os coeficientes da i-ésima condição de contorno do lado esquerdo (direito) envolvendo a j-ésima grandeza: deslocamento, giro, momento fletor e força de cisalhamento.

Este sistema linear é homogêneo e, portanto, o determinante deve ser nulo para conseguir soluções não nulas:

$$\boxed{\det(\mathbf{B\Phi}) = 0,} \tag{4.18}$$

igualdade que é denominada *equação característica* da viga e cujas raízes fornecem os valores do parâmetro β. Na verdade, os valores de β podem ser considerados em seqüência $\{\beta_n\}_{n=1}^{\infty}$, correspondendo a cada valor β_n o n-ésimo modo da viga

$$X_n(x) = c_1(\beta_n)\varphi(x) + c_2(\beta_n)\varphi'(x) + c_3(\beta_n)\varphi''(x) + c_4(\beta_n)\varphi'''(x). \tag{4.19}$$

4.2 Vibrações forçadas com a influência da força axial

Seguindo [GIA 00], as vibrações forçadas podem ser obtidas pelo método espectral como segue. A equação(4.1) por ter coeficientes constantes pode ser escrita na forma

$$M\frac{\partial^2 v}{\partial t^2} + Kv = f(t,x) \tag{4.20}$$

onde M é o operador $m\mathcal{I}$, onde $\mathcal{I}$ denota o operador identidade e K é o operador diferencial linear espacial de quarta ordem

$$K = EI\frac{d^4}{dx^4} + N\frac{d^2}{dx^2}, \tag{4.21}$$

que satisfaz as condições iniciais $v(0,x) = v_0(x)$ e $v_t(0,x) = \dot{v}_0(x)$ e as condiçoes de contorno $B_1 v(0) = 0$,$B_2 v(L) = 0$.

4.2.1 O método espectral para vibrações forçadas com a influência da força axial

Supõe-se que o problema de contorno simétrico possua um conjunto ortonormal completo de autofunções [BIR 62] com respeito ao produto interno funcional

$$\langle \phi, \psi \rangle = \int_0^L \phi\psi dx \tag{4.22}$$

para ϕ, ψ, funções contínuas no intervalo [0,L]. Denota-se por

$$\|\psi\| = \left[\int_0^L \psi^2 dx\right]^{1/2} \tag{4.23}$$

a norma da uma função ψ.

A solução geral da equação (4.20), pelo método espectral, escreve-se na forma:

$$v(t,x) = \sum_{n=1}^{\infty} v_n(t) X_n(x) \tag{4.24}$$

onde $X_n(x)$ são os modos relativos às condições de contorno do problema e os coeficientes temporais $v_n(t)$ obtêm-se da seguinte maneira:

Substitui-se a equação (4.24) na equação (4.20):

$$\sum_{n=1}^{\infty} m\ddot{v_n}X_n(x) + \sum_{n=1}^{\infty} v_n K X_n(x) = f(t,x), \tag{4.25}$$

Da equação (4.5) vem:

$$\sum_{n=1}^{\infty}[m\ddot{v}_n(t) + m\omega^2 v_n(t)]X_n(x) = f(t,x). \tag{4.26}$$

Efetua-se o produto interno com o modo $X_p(x)$ nos dois termos da equação

$$\langle X_p(x), \sum_{n=1}^{\infty}[\ddot{v}_n(t) + \omega^2 v_n(t)]\rangle = \frac{1}{m}\langle X_p(x), f(t,x)\rangle, \tag{4.27}$$

e assim,

$$\sum_{n=1}^{\infty}[\ddot{v}_n(t) + \omega^2 v_n(t)]\int_0^L X_p(x)X_n(x)dx = \frac{1}{m}\int_0^L f(t,x)X_p(x)dx. \tag{4.28}$$

Pela ortogonalidade dos modos,

$$\int_0^L X_p(x)X_n(x)dx = \begin{cases} 0, p \neq n \\ \|X_n\|^2, \ se\ p = n \end{cases} \tag{4.29}$$

decorre que:

$$\ddot{v}_n(t) + \omega^2 v_n(t) = Q_n(t) \tag{4.30}$$

em que

$$Q_n(t) = \frac{1}{m\|X_n\|^2}\int_0^L f(t,x)X_n(x)dx \tag{4.31}$$

Busca-se encontrar a solução da equação (4.30) , para tanto, considera-se

$$h(t) = \frac{\text{sen}(\omega_n t)}{\omega_n}, \tag{4.32}$$

que satisfaz o problema de valor inicial

$$\ddot{h} + \omega^2 h = 0 \tag{4.33}$$

onde

$$h(0) = 0 \; e \; \dot{h}(0) = 1 \tag{4.34}$$

Escreve-se a equação(4.30) em termos de τ

$$\ddot{v}_n(\tau) + \omega^2 v_n(\tau) = Q_n(\tau) \tag{4.35}$$

e multiplicam-se ambos os lados da equação (4.35) por

$$h(t-\tau) = \frac{\text{sen}\omega_n(t-\tau)}{\omega_n} \tag{4.36}$$

decorre que

$$\int_0^t \ddot{v}_n(\tau)\frac{\text{sen}w_n(t-\tau)}{w_n}d\tau + \omega^2 \int_0^t v_n(\tau)\frac{\text{sen}w_n(t-\tau)}{w_n}d\tau = \int_0^t Q_n(\tau)\frac{\text{sen}w_n(t-\tau)}{w_n}d\tau \tag{4.37}$$

Integram-se duas vezes por partes o primeiro termo e resolve-se para $v_n(t)$,tem-se:

$$v_n(t) = \cos(w_n t)v_n(0) + \frac{\text{sen}(w_n t)}{w_n}\dot{v}_n(0) + \int_0^t Q_n(\tau)\frac{\text{sen}w_n(t-\tau)}{w_n}d\tau \tag{4.38}$$

Da equação(2.41) e das condições iniciais prescritas no problema ,$v(0,x) = v_0(x)$ e $v_t(0,x) = \dot{v}_0(x)$, obtêm-se os valores iniciais $v_n(0)$ e $\dot{v}_n(0)$. Mais precisamente, da relação

$$v(0,x) = v_0(x) = \sum_{n=1}^{\infty} v_n(0)X_n(x)dx; \tag{4.39}$$

Aplica-se a ortogonalidade dos modos para obter

$$v_n(0) = \frac{\int_0^L v_0(x)X_n(x)dx}{\|X_n\|^2}. \tag{4.40}$$

Igualmente, com a derivada inicial, decorre

$$\dot{v}_n(x) = \frac{\int_0^L \dot{v}_0(x)X_n(x)dx}{\|X_n\|^2}. \tag{4.41}$$

Utilizando (4.40) e (4.41), chega-se à expressão:

$$v(t,x) = \sum_{n=1}^{\infty}[\cos(w_n t)(\frac{\int_0^L v_0(x)X_n(x)dx}{\|X_n\|^2})+\frac{\mathrm{sen}(w_n t)}{w_n}(\frac{\int_0^L \dot{v}_0(x)X_n(x)dx}{\|X_n\|^2})+\sum_{n=1}^{\infty}(\int_0^t Q_n(\tau)\frac{\mathrm{sen}w_n(t-\tau)}{w_n}d(\tau))]X_n(x) \tag{4.42}$$

Troca-se a ordem do somatório pela integral em (4.42), vem

$$\boxed{v(t,x) = \int_0^L [\frac{\partial \mathbf{h}}{\partial t}(t,x,s)mv_0(s) + \mathbf{h}(t,x,s)m\dot{v}_0(s) + \int_0^t \mathbf{h}(t-\tau,x,s)f(\tau,s)d\tau\ ]ds} \tag{4.43}$$

onde

$$\mathbf{h}(t,x,s) = \sum_{n=1}^{\infty}(\frac{X_n(s)X_n(x)}{\|X_n\|^2})\frac{\mathrm{sen}w_n t}{w_n}\frac{1}{m} \tag{4.44}$$

vem ser a Função de Green de valor inicial ou resposta impulso do sistema distribuído, onde $\|X_n\|$ denota a norma integral quadrática do modo X_n. Na equação (4.43) tem-se nos dois primeiros termos a vibração livre e no terceiro termo a vibração forçada de uma viga sujeita à força axial e à carga externa f(t,x)

Para condições iniciais nulas,isto é ,$v(0) = 0$ e $\dot{v}(0) = 0$ a equação (4.43) pode ser escrita na forma compacta

$$v(t,x) = \int_0^t \mathbf{h}(t-\tau)\mathrm{F}(\tau)d\tau, \tag{4.45}$$

onde **h** é abreviação do operador

$$\mathbf{h}(t)\phi(x) = \int_0^L \mathbf{h}(t,x,s)\phi(s)ds. \tag{4.46}$$

Para cada τ, $F(\tau)$ denota a função $f(\tau, s)$, no intervalo $0 \leq s \leq L$.

4.2.1.1 A Carga f(t,x)

Neste trabalho será considerada uma carga harmônica temporal conectada num ponto da viga. Sobre um ponto a da viga aplicar-se-á uma carga cuja equação e os parâmetros de entrada estão indicados na tabela a seguir:

Valores de Entrada			**Carga**
P_0	$\hat{\omega}$	a	$f(t,x) = P_0\mathrm{sen}(\hat{\omega}_n t)\delta(x-a)$

Tabela 4.4 Valores de entrada e carga pontual

5 SIMULAÇÕES NO SISTEMA INTERMAPLE

Para iniciar as simulações no sistema é necessário fazer ajustes de configurações para o correto funcionamento. A primeira seleção é ajustar a versão do Maple mais atualizada instalada no sistema, sendo disponível para a versão 5 ou 7.

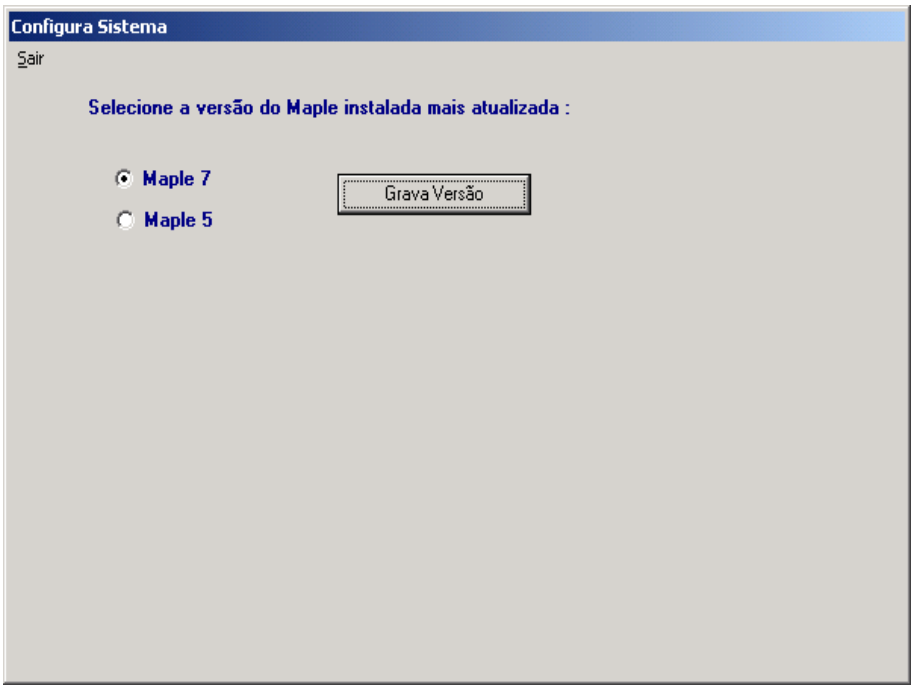

Figura 5.1 *Formulário de Configuração do Sistema*

Quando o Maple está sendo carregado pela primeira vez, não está na memória cache, a interface irá esperar 30 segundos para o Maple ser carregado adequadamente e carregar o arquivo executor.mws e enviar o enter para iniciar o cálculo. Se o usuário não sair da interface e efetuar um segundo cálculo ele será mais rápido, pois o Maple já se encontra na memória cache e, este tempo de espera será decrementado automaticamente. A interface faz o reconhecimento do processador existente na máquina para melhor sincronizar-se com o Maple.

O formulário sobre o sistema descreve informações a respeito do sistema tais como : Versão do sistema; Universidade propicionadora do Mestrado; Instituto; Orientador e Autor da dissertação.

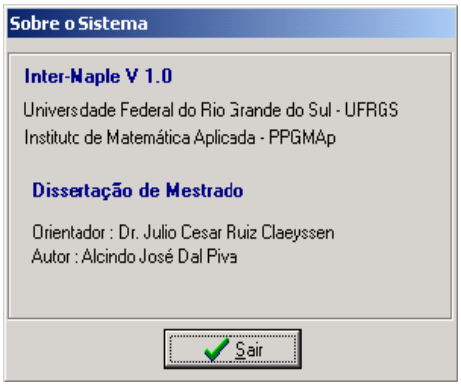

Figura 5.2 *Formulário Sobre o Sistema*

A interface foi programada para efetuar cálculos de vigas de Euler Bernoulli de condições clássicas do tipo : Apoiada Deslizante, Apoiada Livre, Biapoiada, Deslizante Deslizante, Deslizante Livre, Fixa Apoiada, Fixa Livre, Fixa Deslizante, Fixa Fixa, Livre Livre.

Na figura 5.3 é apresentado o menu principal do sistema Intermaple versão 1.0 - para cálculos simbólicos de vibrações, onde pode-se optar por qualquer uma das vigas apresentadas.

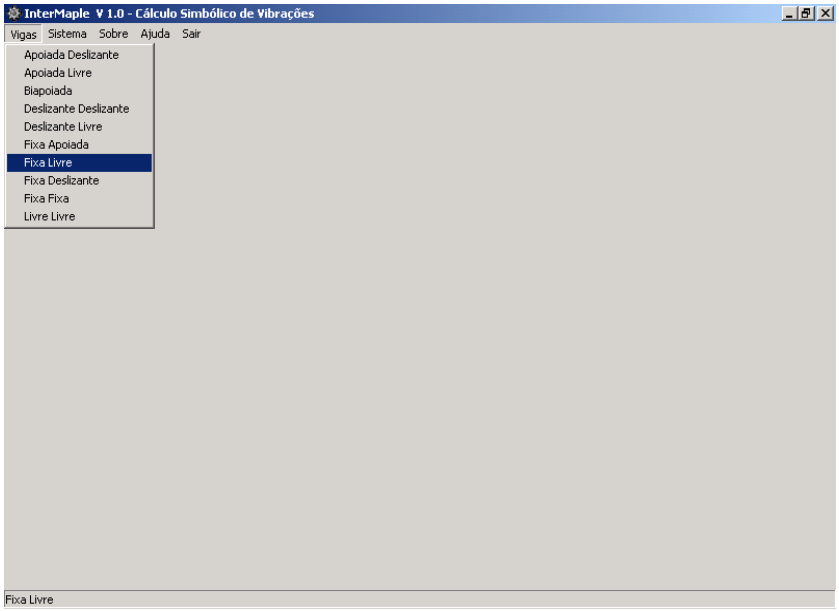

Figura 5.3 *Menu Principal da Interface*

Nas simulações subseqüentes se utilizar-se-á a viga do tipo Fixa Livre para a comprovação de que se a freqüência de entrada tiver o seu valor próximo de uma das freqüências características resultantes acontecerá a ativação do modo e o gráfico da vibração forçada terá o padrão do modo ativado.

5.1 Simulação 1 com a Viga Fixa Livre

Far-se-ão simulações da viga do tipo Fixa Livre conforme a seleção do menu principal da interface conforme figura 5.3.

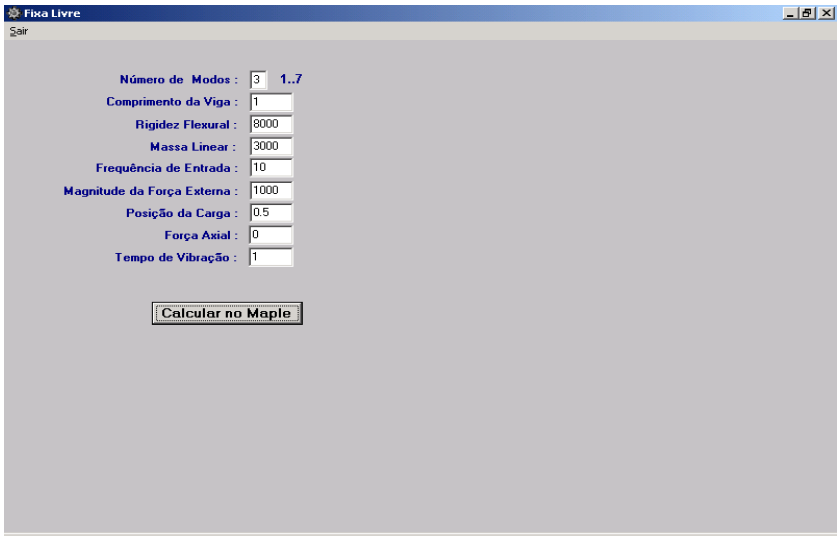

Figura 5.4 *Tela de Entrada de Dados da Viga Fixa Livre - Primeira Simulação*

Nesta primeira simulação utilizar-se-ão os seguintes valores de entrada de dados, conforme tela de entrada de dados figura 5.4, destacando a freqüência de entrada com o valor 10.

Após os dados terem sido confirmados pelo usuário o Maple será ativado e efetuará o cálculo com os valores dados. Serão mostrados os valores resultantes deste cálculo em uma tela de saída conforme figura 5.5 mostrada a seguir.

Pode-se comprovar, visualmente, que quando se tem uma freqüência de entrada próxima a uma freqüência resultante, o modo desta freqüência resultante

será ativado e o gráfico da vibração forçada assumirá o padrão desta freqüência, ativando, neste caso o primeiro modo. A freqüência de entrada é 10 e a resultante 5,74.

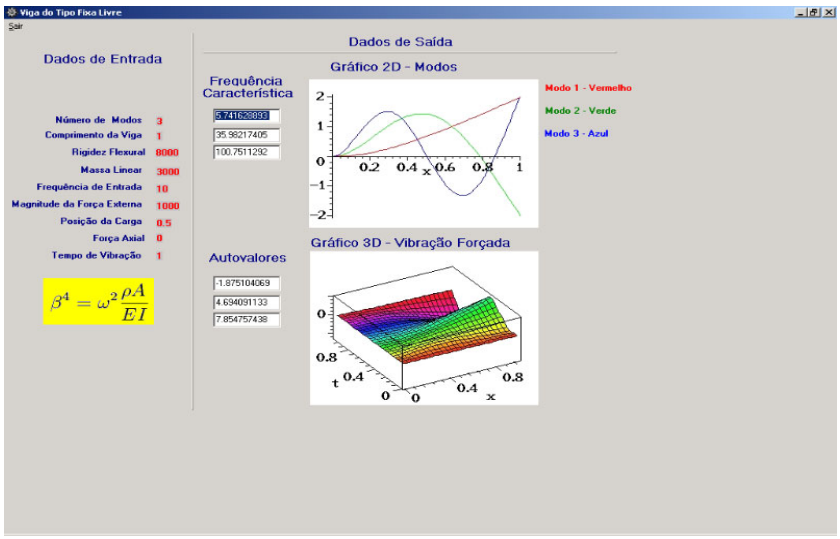

Figura 5.5 *Resultados da Primeira Simulação*

5.2 Simulação 2 com a Viga Fixa Livre

Far-se-ão simulações da viga do tipo Fixa Livre conforme a seleção do menu principal da interface conforme figura 5.3.

Nesta segunda simulação utilizar-se-ão os seguintes valores de entrada de dados, conforme tela de entrada de dados figura 5.6, destacando a frequência de entrada com o valor 35.

Após os dados terem sido confirmados pelo usuário o Maple será ativado e efetuará o cálculo com os valores dados. Serão mostrados os valores resultantes deste cálculo em uma tela de saída conforme figura 5.7 mostrada a seguir.

Pode-se comprovar, visualmente, que quando se tem uma freqüência de entrada próxima a uma freqüência resultante, o modo desta freqüência resultante será ativado e o gráfico da vibração forçada assumirá o padrão desta freqüência,

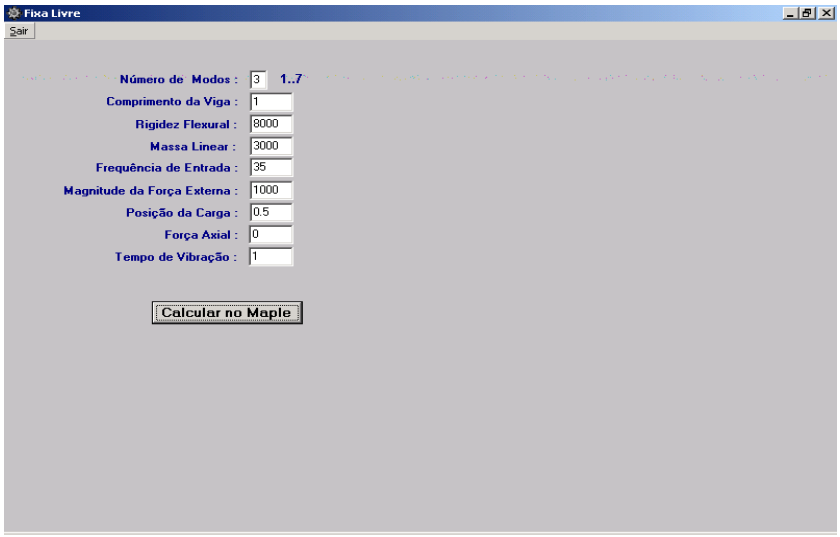

Figura 5.6 *Tela de Entrada de Dados da Viga Fixa Livre - Segunda Simulação*

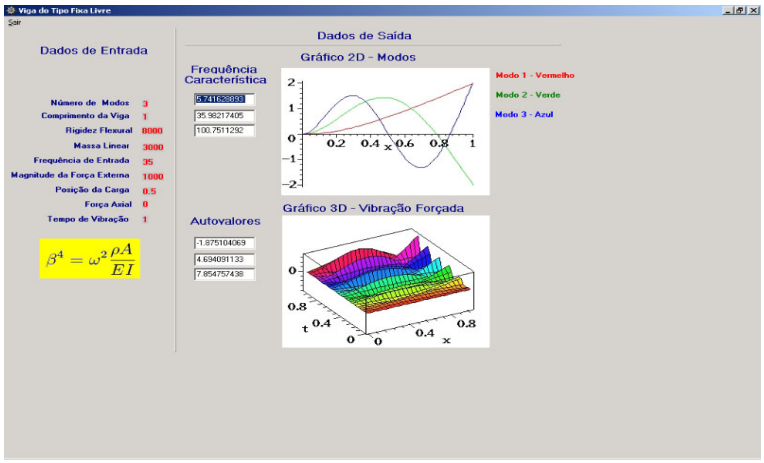

Figura 5.7 *Resultados da Segunda Simulação*

ativando, neste caso o segundo modo. A freqüência de entrada é 35 e a resultante é 35,98.

5.3 Programa Fonte da Viga Fixa Livre com Variáveis Implantadas

Programa texto com variáveis implantadas do caso da simulação 1 apresentado na seção 5.1, onde se pode visualizara implantação das variáveis BOL_01, BOL_03, BOL_04, BOL_05, BOL_06, BOL_07 e INT_01, INT_02, INT_03.

```
>restart:
>with(linalg):
>phi[1] := x -> (1/(epsilon^2+delta^2) *
(-1/delta * sin(delta*x) + (1/epsilon) * sinh(epsilon*x))):
>phi[1](x);
>phi[2] := unapply(simplify(diff(phi[1](x),x)),x):
>phi[2](x);
>phi[3] := unapply(simplify(diff(phi[2](x),x)),x):
>phi[3](x);
>phi[4] := unapply(simplify(diff(phi[3](x),x)),x):
>phi[4](x);
>Phi:=matrix(8,4,[phi[1](0),phi[2](0),phi[3](0),phi[4](0),D(phi[1])(0),
      D(phi[2])(0),D(phi[3])(0),D(phi[4])(0),(D@@2)(phi[1])(0),(D@@2)(phi[2])(0),
      (D@@2)(phi[3])(0),(D@@2)(phi[4])(0),(D@@3)(phi[1])(0),(D@@3)(phi[2])(0),
      (D@@3)(phi[3])(0),(D@@3)(phi[4])(0),phi[1](L),phi[2](L),phi[3](L),phi[4](L),
      D(phi[1])(L),D(phi[2])(L),D(phi[3])(L),D(phi[4])(L),(D@@2)(phi[1])(L),
      (D@@2)(phi[2])(L),(D@@2)(phi[3])(L),(D@@2)(phi[4])(L),(D@@3)(phi[1])(L),
      (D@@3)(phi[2])(L),(D@@3)(phi[3])(L),(D@@3)(phi[4])(L)]):
>B:=matrix(4,8,[1,0,0,0, 0,0,0,0, 0,1,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0,
               0,0,0,0, 0,0,0,1]):
>B_Phi:=evalm(B&*Phi):
>eq_car := simplify(det(B_Phi));
>U:=LUdecomp(B_Phi);
>U[2,2]:=0;
>U:=evalm(U):
```

```
>det(U);
>f := matrix(4,1,[0,0,0,0]):
>c:=linsolve(U,f,'r',v);
>funcoes:= matrix(1,4,[phi[1](x),phi[2](x),phi[3](x),phi[4](x)]);
>modos:=simplify(multiply(funcoes,c)[1,1]/v[1][1]);
>L := INT_02:
>mL := BOL_01 * INT_02 :
>a := BOL_05/10 * INT_02:
>g := evalf(sqrt(BOL_06/ INT_03 ));
>with(student):
>epsilon := sqrt(sqrt(beta^4+g^4/4)-g^2/2);
>delta := sqrt(epsilon^2+g^2);
>FD1 := fopen("C:/VIGAS/DADOS_AUTOVALORES",WRITE,TEXT);
>FD2 := fopen("C:/VIGAS/DADOS_FREQUENCIA_CARACTERISTICA",WRITE,TEXT);
>for n from 1 to INT_01 do
>bbeta[n]:=fsolve(eq_car/beta^4,beta=(2*n-1)*Pi/2/INT_02);
>vetor_01 := array([bbeta[n]]);
>writedata(FD1,vetor_01);
>ww[n]:=evalf(bbeta[n]^2*sqrt( INT_03 / mL ));
>vetor_02 := array([ww[n]]);
>writedata(FD2,vetor_02);
>modo_d[n] := subs(beta=bbeta[n],modos);
>norma_d[n]:= evalf(sqrt(simpson(modo_d[n]^2,x=0..INT_02,200)));
>modo_d[n] := modo_d[n]/norma_d[n];
>od:
>fclose(FD1);
>fclose(FD2);
>with(plots):
>AAd1:=plot(modo_d[1],x=0..INT_02,color=red):
```

```
>AAd2:=plot(modo_d[2],x=0..INT_02,color=green):
>AAd3:=plot(modo_d[3],x=0..INT_02,color=blue):
>AAd4:=plot(modo_d[4],x=0..INT_02,color=brown):
>AAd5:=plot(modo_d[5],x=0..INT_02,color=black):
>AAd6:=plot(modo_d[6],x=0..INT_02,color=yellow):
>AAd7:=plot(modo_d[7],x=0..INT_02,color=gray):
>plotsetup(jpeg,plotoutput=‘C:/VIGAS/FIGURA_01.JPG‘,
 plotoptions=‘portrait,height=216,width=288,noborder‘);
>display({AAd1,AAd2,AAd3,AAd4,AAd5,AAd6,AAd7},thickness=3);
>for n from 1 to INT_01 do
>C[n]:=subs(x=a,modo_d[n])/(ww[n]*(BOL_03 ^ 2-ww[n]^2));
>T[n]:= BOL_03 * sin(ww[n]*t)-ww[n]*sin(BOL_03*t);
>od;
>SERIE:= 2 * BOL_04 /mL * INT_02*sum('C[n]*T[n]*modo_d[n]','n'=1..INT_01);
>TT := BOL_07:
>plotsetup(jpeg,plotoutput=‘C:/VIGAS/FIGURA_02.JPG‘,
 plotoptions=‘portrait,height=216,width=288,border‘);
>plot3d(SERIE,t=0..TT,x=0..INT_02,color=t,numpoints=625,axes=boxed,
 orientation=[-120,37]);
>quit;
```

5.4 Programa Fonte da Viga Fixa Livre com Variáveis Substituídas

Programa fonte do Maple com as variáveis implantadas, já substituídas por valores numéricos lidos na tela de entrada de dados da viga do tipo Fixa Livre na primeira simulação no item 5.1.

```
>restart:
>with(linalg):
>phi[1] := x -> (1/(epsilon^2+delta^2) *
```

```
          (-1/delta * sin(delta*x) + (1/epsilon) * sinh(epsilon*x))):
>phi[1](x);
>phi[2] := unapply(simplify(diff(phi[1](x),x)),x):
>phi[2](x);
>phi[3] := unapply(simplify(diff(phi[2](x),x)),x):
>phi[3](x);
>phi[4] := unapply(simplify(diff(phi[3](x),x)),x):
>phi[4](x);
>Phi:=matrix(8,4,[phi[1](0),phi[2](0),phi[3](0),phi[4](0),D(phi[1])(0),
      D(phi[2])(0),D(phi[3])(0),D(phi[4])(0),(D@@2)(phi[1])(0),(D@@2)(phi[2])(0),
      (D@@2)(phi[3])(0),(D@@2)(phi[4])(0),(D@@3)(phi[1])(0),(D@@3)(phi[2])(0),
      (D@@3)(phi[3])(0),(D@@3)(phi[4])(0),phi[1](L),phi[2](L),phi[3](L),phi[4](L),
      D(phi[1])(L),D(phi[2])(L), D(phi[3])(L),D(phi[4])(L),(D@@2)(phi[1])(L),
      (D@@2)(phi[2])(L), (D@@2)(phi[3])(L),(D@@2)(phi[4])(L),(D@@3)(phi[1])(L),
      (D@@3)(phi[2])(L),(D@@3)(phi[3])( L ),(D@@3)(phi[4])( L )]):
>B := matrix(4,8,[1,0,0,0,  0,0,0,0, 0,1,0,0,  0,0,0,0, 0,0,0,0,
                  0,0,1,0,  0,0,0,0,  0,0,0,1]):
>B_Phi:=evalm(B&*Phi):
>eq_car := simplify(det(B_Phi));
>U:=LUdecomp(B_Phi);
>U[2,2]:=0;
>U:=evalm(U):
>det(U);
>f := matrix(4,1,[0,0,0,0]):
>c:=linsolve(U,f,'r',v);
>funcoes:=matrix(1,4,[phi[1](x),phi[2](x),phi[3](x),phi[4](x)]);
>modos:=simplify(multiply(funcoes,c)[1,1]/v[1][1]);
>L := 1:
>mL := 3000 * 1 : a := 0.5/10 * 1:
```

```
>g := evalf(sqrt(0/ 8000 ));
>with(student):
>epsilon := sqrt(sqrt(beta^4+g^4/4)-g^2/2);
>delta := sqrt(epsilon^2+g^2);
>FD1 := fopen("C:/VIGAS/DADOS_AUTOVALORES",WRITE,TEXT);
>FD2 := fopen("C:/VIGAS/DADOS_FREQUENCIA_CARACTERISTICA",WRITE,TEXT);
>for n from 1 to 3 do
>bbeta[n]:=fsolve(eq_car/beta^4,beta=(2*n-1)*Pi/2/1);
>vetor_01 := array([bbeta[n]]);
>writedata(FD1,vetor_01);
>ww[n]:=evalf(bbeta[n]^2*sqrt( 8000 / mL ));
>vetor_02 := array([ww[n]]);
>writedata(FD2,vetor_02);
>modo_d[n] := subs(beta=bbeta[n],modos);
>norma_d[n] := evalf(sqrt(simpson(modo_d[n]^2,x=0..1,200)));
>modo_d[n] := modo_d[n]/norma_d[n];
>od:
>fclose(FD1);
>fclose(FD2);
>with(plots):
>AAd1:=plot(modo_d[1],x=0..1,color=red):
>AAd2:=plot(modo_d[2],x=0..1,color=green):
>AAd3:=plot(modo_d[3],x=0..1,color=blue):
>AAd4:=plot(modo_d[4],x=0..1,color=brown):
>AAd5:=plot(modo_d[5],x=0..1,color=black):
>AAd6:=plot(modo_d[6],x=0..1,color=yellow):
>AAd7:=plot(modo_d[7],x=0..1,color=gray):
>plotsetup(jpeg,plotoutput=‘C:/VIGAS/FIGURA_01.JPG‘,
 plotoptions=‘portrait,height=216,width=288,noborder‘);
```

```
>display({AAd1,AAd2,AAd3,AAd4,AAd5,AAd6,AAd7},thickness=3);
>for n from 1 to 3 do
>C[n]:=subs(x=a,modo_d[n])/(ww[n]*(6 ^ 2-ww[n]^2));
>T[n]:= 6 * sin(ww[n]*t)-ww[n]*sin(6*t);
>od;
>SERIE:= 2 * 1000 /mL * 1*sum('C[n]*T[n]*modo_d[n]','n'=1..3);
>TT := 2:
>plotsetup(jpeg,plotoutput=`C:/VIGAS/FIGURA_02.JPG`,
 plotoptions=`portrait,height=216,width=288,border`);
>plot3d(SERIE,t=0..TT,x=0..1,color=t,numpoints=625,axes=boxed,
 orientation=[-120,37]);
>quit;
```

6 CONCLUSÕES

O usuário do sistema matemático Maple, ao desenvolver uma nova aplicação ou quando manipula outras aplicações matemáticas existentes, freqüentemente depara-se com a situação de ter seus programas fontes soltos ou perdidos no disco rígido do computador. A cada novo cálculo que desejar, é necessário localizar o programa fonte, abri-lo e executar. Quando desejar usar novos valores de variáveis para uma nova situação, é necessário editorar o programa fonte e substituir manualmente as variáveis específicas e rodar novamente para recalcular a nova situação desejada. Após o cálculo ter sido efetuado a visualização dos resultados se torna embaraçada, sendo geralmente necessária a rolagem de tela para a procura dos dados de interesse do cálculo.

Neste trabalho descreveu-se a análise, a implementação e as características da interface, onde a situação vivenciada pelo usuário do aplicativo matemático Maple é resolvida, não sendo mais necessário localizar seus programas fontes de aplicações no disco rígido, os programas de interesse afins estarão em um único sistema.

A editoração de programas fontes de aplicações para uma nova entrada de constantes numéricas é suprimida, pois é tarefa da interface fazer a substituição automatizada. A entrada de dados é feita na interface, onde são lidas todas as variáveis de interesse do cálculo e substituídas automaticamente no programa fonte da aplicação. A substituição automatizada evita erros de digitação que possam ocasionar erros de sintaxe na execução, conseqüentemente aumentando a rapidez e a confiabilidade em efetuar várias vezes o mesmo ou diferentes cálculos disponíveis pela interface.

Após o cálculo ter sido efetuado, somente os dados de valores e gráficos pré-definidos pelo sistema serão mostrados ao usuário de forma organizada em uma ambiente amigável.

Para demonstrar a capacidade da interface, implementou-se como modelo de aplicação o cálculo modal clássico de estruturas flexíveis de Euler-Bernoulli, sendo que esta aplicação irá facilitar os usuários das vigas das áreas da matemática e engenharias.

A interface foi implementada na linguagem de programação Delphi, que apresenta, em seu módulo principal, a seleção para dez tipos diferentes de vigas clássicas para cálculo modal e visualização.

Como trabalhos futuros, estuda-se a implementação de vigas não clássicas de Euler-Bernoulli, equações de Timoshenko e outras aplicações da matemática aplicada.

REFERÊNCIAS BIBLIOGRÁFICAS

[BIR 62] BIRKHOFF, G & ROTA, G. Ordinary Differential Equations.*Introduction Higher Mathematics*,Ed. Ginn and company,1962.

[CAN 96] CANTÚ, M. *Delphi API´s Sockets*, Visual,2000.

[CHI 00] CHIWIACOWSKY, L. Cálculo Simbólico doas Modos Vibratórios no Modelo de Kuchoff-Love para placas. *Dissertação de Mestrado*. UFRGS/CPGMap, Porto Alegre,2000.

[CLA 90a] CLAEYSSEN, J. C. R. On Predicting the Response of Non-Conservative Linear Vibrating Systems by Using Dynamical Matrix Solutions. *Journal of Sound and Vibration*, 140(1): 73-84, 1990.

[CLA 90b] CLAEYSSEN, J. C. R., TSUKAZAN, T. Dynamical Solutions of Linear Matrix Differential Equations. *Quarterly of Applied Mathematics*, vol. XLVIII, n^0 1, 1990.

[CLA 99a] CLAEYSSEN, J. C. R., CANAHUALPA, G., JUNG, C. A Direct Approach to Second-Order Matrix Non-Classical Vibrating Equations, Applied Numerical Mathematics, vol.30,1999.

[CLO 93] CLOUGH, R. W. & PENZIEN, J. Dynamics of Structures. Prentice Hall, 1993.

[CUD 89] CUDNEY, H. H., INMAN, D. J. Determining Damping Mechanisms in a Composite Beam. *International Journal of Analytical and Experimental Modal Analysis*. vol 4., no. 4, 138-143, 1989.

[DAT 95] DATTA, B. N. *Numerical Linear Algebra and Applications*. Brooks/Cole Publishing Company, Pacific Grove, California, 1995.

[GIA 00] GIARETA, K. M. Vibrações Forçadas com forças axial num Modelo de Euler-Bernoulli para Vigas. *Dissertação de Mestrado*, UFRGS/CPGMAP, Porto Alegre, 2000.

[INM 89] INMAN, D. *Vibration, with Control, Measurement, and Stability.* Prentice Hall, Englewood Cliffs, 1989.

[LIS 00] LISCHNER, R. *Delphi: O Guia Essencial*, Campus,Rio de Janeiro,2000.

[MOS 99] MOSCHEN, I. D. Cálculo Simbólico de Modos Vibratórios no Modelo Euler-Bernoulli para vigas. *Dissertação fe Mestrado* UFRGS/CPGMAp, Porto Alegre,1999.

[SHI 96] SHIRAISHI, K. *Conhecendo e Trabalhando com Delphi 4*, Érica,São Paulo,2000.

[SOD 00] SODER, R. A. L Modos Flexurais sob a Influência de uma Força Axial. *Dissertação de Mestrado*. UFRGS/CPGMAP, Porto Alegre, 2000.

[SWA 96] SWAN, T. *Delphi:Biblia do Programador*, Berkeley,São Paulo,1996.

[TEI 00] TEIXEIRA, S. *Delphi 5,Guia do Desenvolvedor*,Campus,Rio de Janeiro,2000.