10 (dez)
20/12/98

Escola Politécnica da Universidade de São Paulo

PMC – 581 – Projeto Mecânico II

Trabalho de Formatura 1998

Desenvolvimento de um Software para Resolução de Sistemas Massa-Mola-Amortecedor

Relatório Final para avaliação - 2º Semestre de 1998

Área: Projeto e Fabricação

Prof. Responsável : Prof. Edson Gomes

Prof. Orientador : Prof. Dr. Marcílio Alves

Aluno : Bruno Luiz Assaf NUSP: 0127697

Índice

Introdução

De acordo com as propostas da disciplina PMC-581 (Projeto Mecânico II) e do escopo do projeto desenvolvido o relatório abaixo contém os seguintes tópicos :

- Motivação Mercadológica
- Requisitos de Mercado
- Síntese de Soluções
- Desenvolvimento teórico da Solução
- A estrutura do software Mechcalc 2.0
- Características do Software
- Testes Funcionais
- Análise Econômica
- Listagens

Neste relatório será discutida e provada a viabilidade de desenvolver-se o projeto, incluindo desde aspectos meramente técnicos até mesmo sua viabilidade econômica. Além disso todos os apsctos técnicos de modelagem, as características atuais do software e alguns testes com exemplos cujo resultado é conhecido estão presentes.

1. Motivação Mercadológica

O ponto de partida de qualquer projeto é sem dúvida uma necessidade. Neste capítulo são citados alguns fatos que suscitam a percepção de que o projeto têm real utilidade, necessidade e portanto valor.

A necessidade percebida é o **desenvolvimento de software capaz de solucionar sistemas massa-mola-amortecedor**. Essa necessidade advém do fato de que há um mercado muito favorável como um todo à análise computacional de projetos de forma a possibilitar a correção e otimização destes numa fase de menor prejuízo econômico. Ganha-se tempo e dinheiro com essas análises, sendo que atualmente as soluções de maior penetração atualmente são as soluções FEA. Mais do que isso, existem muito poucos softwares utilizando o enfoque massa-mola-amortecedor, o que confere certa exclusividade para o produto, com todas as vantagens associadas. Além disso, foi detectado que as mais diversas áreas têm uso para um software de tal natureza, representando possibilidades de ampla penetração no mercado. Como exemplos, posso citar os seguintes setores :

Pesquisa Acadêmica

O setor de pesquisa utiliza modelos massa-mola-amortecedor no desenvolvimento e análise de diversos trabalhos. As áreas de impacto

estrutural, acústica e vibrações dinâmicas são exemplos clássicos desses estudos. Como exemplo posso citar os Trabalhos de análise do comportamento de colapso dinâmico elasto-plástico de tubos submetidos a impacto axial, de Karagiozova e Jones(1), de grande influência no meio acadêmico.

Ensino acadêmico

O setor acadêmico poderia com facilidade utilizar tal ferramenta para viabilizar uma conexão mais estreita entre teoria e prática em matérias específicas como vibrações em laboratórios para alunos de graduação.

Aplicação Industrial

Diversos setores da indústria tem interesse em prever o comportamento de seus projetos antes da execução, possibilitando economia em recursos, ferramental e outros. A utilização de ferramentas como análise através de Elementos Finitos têm crescido, entretanto suas características incluem altíssima demanda de poder de processamento e tempo, inviabilizando uma maior disseminação dessas ferramentas em especial no cenário nacional. O software proposto tem condições de se encaixar em nichos de menor complexidade e/ou como pré-processador de projetos, otimizando-o para posterior análise através de FEA.

Mesmo em países de industrialização menos recente e mais desenvolvida, como os Estados Unidos esse enfoque é atual e utilizado, como

pode se perceber também no paper "State of the Art Review of Automobile Structural Crashworthiness" do American Iron and Steel Institute, onde entre outros assuntos é discutido um modelo massa-mola-amortecedor adequado para simular o comportamento da estrutura frontal de um veículo submetida aos testes de impacto exigidos pela legislação americana.

Todos esses setores possuem demandas diversas, mas que podem ser consolidadas de forma a possibilitar a criação de um produto com uma maior penetração no mercado.

2. *Requisitos de Mercado*

Este capítulo resume as exigências do produto para que se possa determinar objetivos de projeto mais claros. Trata-se também de certa forma de uma colocação de como encontra-se atualmente o mercado para o produto, tanto do ponto de vista de potenciais consumidores bem como da atual oferta de produtos.

De acordo com contatos feitos com empresas como a Compugraf (Que atualmente licencia os produtos MSC, como o NASTRAN), Volkswagen do Brasil, e a própria percepção do autor, posso destacar alguns requisitos necessários para assegurar maiores possibilidades de sucesso do produto:

- produto deve ter desempenho significativamente superior (cerca de 50%) em tempo de execução quando confrontado a soluções FEA, para poder ser apresentado ao mercado como pré-processador de projetos.
- Deve também ser executado em plataformas de hardware mais leves e reconhecidamente disseminadas tanto no ambiente empresarial bem como no acadêmico para ampliar o mercado potencial a ser atingido.
- A maioria das necessidades acadêmicas/industriais é atendida por um produto que analise o problema de forma unidimensional. Ainda assim, é interessante(desde que não onere em demasiado o projeto) o desenvolvimento de uma solução bidimensional pois uma grande classe de problemas é melhor analisada sob este enfoque, possibilitando não só

maior penetração de mercado bem como um maior reconhecimento do produto. Assim, deve-se analisar o compromisso de desempenho entre as soluções bi e unidimensionais.

- Deve apresentar confiabilidade acima de 30% da realidade a fim de justificar melhora nos processos de projeto e viabilizá-lo economicamente, e também características de configuração que permitam o refinamento dos modelos a fim de melhorar o seu desempenho e adaptá-lo a diversas aplicações.
- Interface amigável, minimizando os custos de adaptação e treinamento.

3. Síntese de Soluções

Esta fase compreende a discussão e melhor formulação sobre os possíveis "caminhos " a trilhar na busca da solução. Em verdade trata-se da explicitação de diversos aspectos relativos ao desempenho do produto no mercado.

3.1 Plataforma de Hardware

No mercado de hoje, temos basicamente três plataformas antagônicas: Os cada vez mais rápidos processadores e subsistemas da linha Intel, os sistemas especializados RISC de empresas como Sun e Silicon Graphics e a plataforma Alpha da Digital que é o processador de *clock* mais rápido do mundo. Em fevereiro de 1998 a revista PC Magazine publicou um levantamento de desempenho de *workstations* dessas plataformas. Hoje o desempenho da plataforma Intel é ainda melhor com sistemas multiprocessados Pentium II Xeon de 450 Mhz, enquanto as workstations RISC não tiveram o mesmo ritmo de renovação dos processadores. Assim optei por

Segue abaixo tabela comparativa de desempenho e preço:

Equipamento	Desempenho	Preço	D/P
Compaq Professional Workstation 6000	7.2	9360	77.1
Digital Personal Workstation 500a	8.2	12960	63.6
SGI 02 R10000	7.4	15500	48.0
Sun Ultra 30	7.8	17500	44.5

3.2 Plataforma de Software

Da mesma forma que o hardware, no tocante à sistemas operacionais de workstations duas plataformas sobressaem, a solução UNIX e o Windows NT. Pode-se dizer que o primeiro é teoricamente mais estável, tem uma grande penetração no nicho de *workstations*, mas é um sistema de interfaces proprietárias e de difícil implementação. Espera-se que a iniciativa Java modifique um pouco essa situação, mas ainda é cedo para apostar nisso.

A plataforma da Microsoft, o Windows NT vêm crescendo vertiginosamente nesse meio específico com o avanço impressionante de capacidade dos sistemas Intel. É reconhecidamente um *open-standard* da indústria, com penetração em todos os campos e possui a vantagem de ter uma versão para a poderosa plataforma Alpha 500 Mhz.

De acordo com o exposto anteriormente, por maior facilidade de uso, treinamento, similaridades de capacidade, e até mesmo maior disponibilidade de ferramentas para programação, optei pelo sistema Windows NT/95/98.

3.3 Ferramenta de Programação

Como ferramentas de programação analise as seguintes possibilidades:

- Visual Fortran
- Visual C++
- Visual Basic

- Delphi
- Java

O Visual Fortran, apesar de poderoso no processamento matemático, é muito pobre quanto a facilidade de operação e do seu ambiente gráfico, sendo quase uma solução "DOS" para windows.

O Visual C++ é juntamente com o Delphi, a plataforma mais poderosa de processamento de ponto flutuante, possuindo basicamente a mesma estruturação e apenas perdendo por pouco no ambiente gráfico. É provável que o aplicativo pudesse ser feito sem grandes dificuldades também nessa plataforma.

O Visual Basic é uma alternativa facilmente descartada, pela baxíssima eficiência em processamento matemático/ponto flutuante e pobreza de suas estruturas de dados.

O Delphi é sem dúvida o mais poderoso nesse contexto de processamento de ponto flutuante e aplicação gráfica, perdendo apenas na pouca disponibilidade de bibliotecas matemáticas quando comparado ao Fortran. Entretanto, como o algoritmo será plenamente desenvolvido, não se fazem necessárias essas bibliotecas. Por todas as usas vantagens, somadas ao fato de ser uma linguagem de programação muito utilizada no meio de engenharia(Pascal) e como um todo no mercado, foi a alternativa selecionada.

A iniciativa Java da Sun infelizmente ainda é meramente uma iniciativa. Apesar de suas vantagens como operar em qualquer equipamento, disponibilização remota e outras, ainda é muito pobre em recursos e desempenho em aplicações sérias de engenharia.

3.4 Método de Resolução de EDOs

Os métodos de resolução de Equações Diferenciais Ordinárias mais utilizados são Runge Kutta(3ª e 4ª ordem), Milne, Adams e Hamming

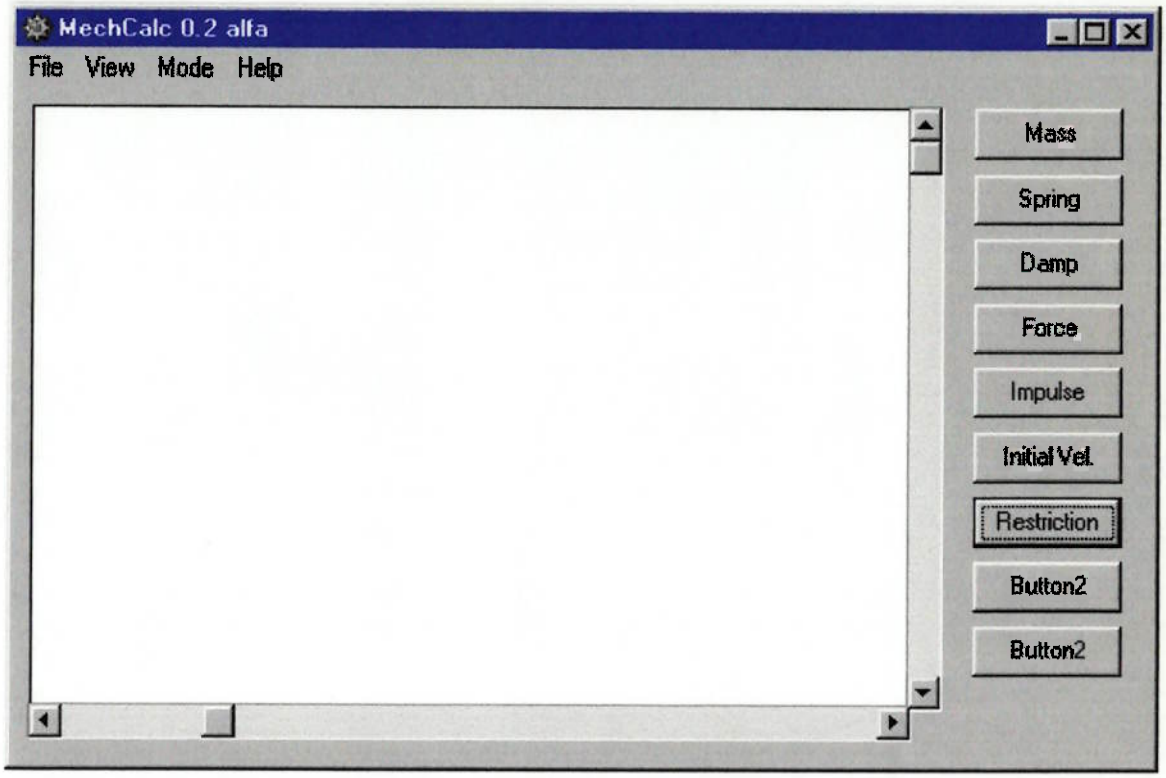
Runge Kutta é um ótimo método, porém apresenta o problema de ser utilizado somente um passo(delta t) em todo o processo, fazendo com que seja desnecessariamente lento em alguns intervalos de tempo do fenômeno estudado.

Milne, Adams e Hamming são métodos *multistep*, ou seja, o passo é corrigido de acordo com os resultados obtidos(usualmente dobrando ou dividindo por 2). O método de Milne entretanto tem problemas de estabilidade inaceitáveis para o projeto a ser desenvolvido. O de Adams é bem mais estável, mas é extremamente complexo, enquanto o de Adams apesar de não ser tão estável quanto o de Adams é superior ao de Milne e é bem simples, dentro do possível.

O enfoque de solução deste método(Hamming) é melhor apresentado no capítulo seguinte, que trata do protótipo.

3.5 Interface

A interface final desejada para o projeto consiste de um *workspace* para onde podem ser arrastados e editados os elementos de interesse, estabelecendo as suas interconexões. A partir desse *workspace* pode ser gerado um arquivo armazenando as características do sistema desenhado, e outro com o seu comportamento. Ao lado do *workspace* se encontrarão botões com os elementos e com o botão direito do mouse será possível editar suas características mecânicas. Ainda será possível obter uma listagem impressa dos elementos e suas variáveis de interesse.



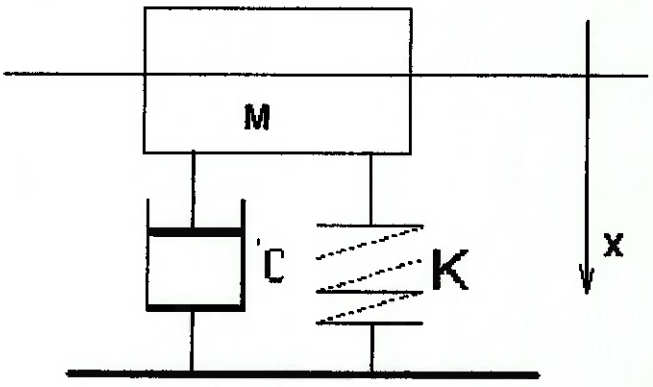
4. Desenvolvimento da Solução

Neste capítulo será discutida a formulação teórica de um sistema massa-mola-amortecedor acompanhada de um algoritmo simples de estimativa de solução, a solução através do método de Hamming implementada num protótipo do primeiro semestre(LMSS beta tester) e a diferença do ponto de visto do algoritmo com relação a solucionar um problema com somente uma massa e outro com diversas delas.

4.1 O sistema massa-mola-amortecedor

Os sistemas massa-mola-amortecedor com uma massa são regidos pela seguinte equação:

$$M \cdot \ddot{x} = -k \cdot x - c \cdot \dot{x} + F(t)$$



É facil perceber que essa equação nada mais é do que Força é igual a massa multiplicada pela aceleração.

Um método simples de tentar estimar o próximo estado da massa(posição,velocidade) a partir do estado anterior seria algo como:

$$\dot{x}(t) = \dot{x}(t-1) + \ddot{x}(t-1) \cdot \Delta t$$

$$\ddot{x}(t-1) = \frac{k \cdot x(t-1) - c \cdot \dot{x}(t-1)}{M}$$

O que é imediato pois a primeira nada mais é do que v = v0 + a . t , e a segunda nada mais é do que: a= F/M no instante anterior conhecido. Todos os método numéricos de resolução se baseiam de alguma forma nessa forma de resolver o problema: A partir de um estado anterior conhecido(ou mais de um) é possível estimar como um sistema estará no futuro. Isso se torna particularmente verdadeiro à medida em que se reduz o passo de cálculo.

4.2 O método de Hamming

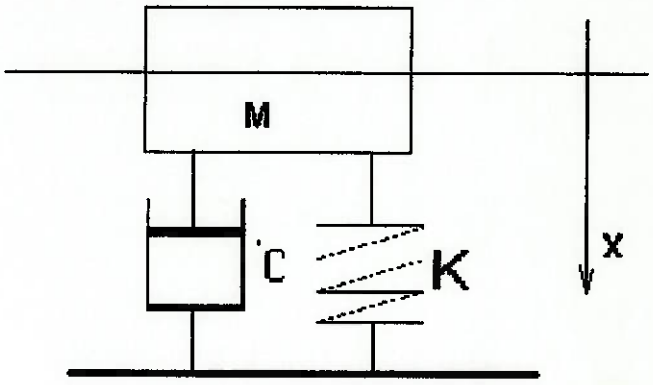
Um dos métodos de resolução de Equações Diferenciais mais populares é o método de Hamming. Todos os métodos númericos de resolução mais populares só resolvem sistemas com derivada primeira. Para resolver o problema massa-mola-amortecedor(derivada segunda em relação ao tempo) é preciso utilizar uma variável auxiliar que no exemplo será y. Assim:

$$M \cdot \ddot{x} = -k \cdot x - c \cdot \dot{x} + F(t)$$

$$\downarrow$$

$$\dot{x} = y$$

$$\dot{y} = \frac{-k \cdot x - c \cdot y + F(t)}{M}$$

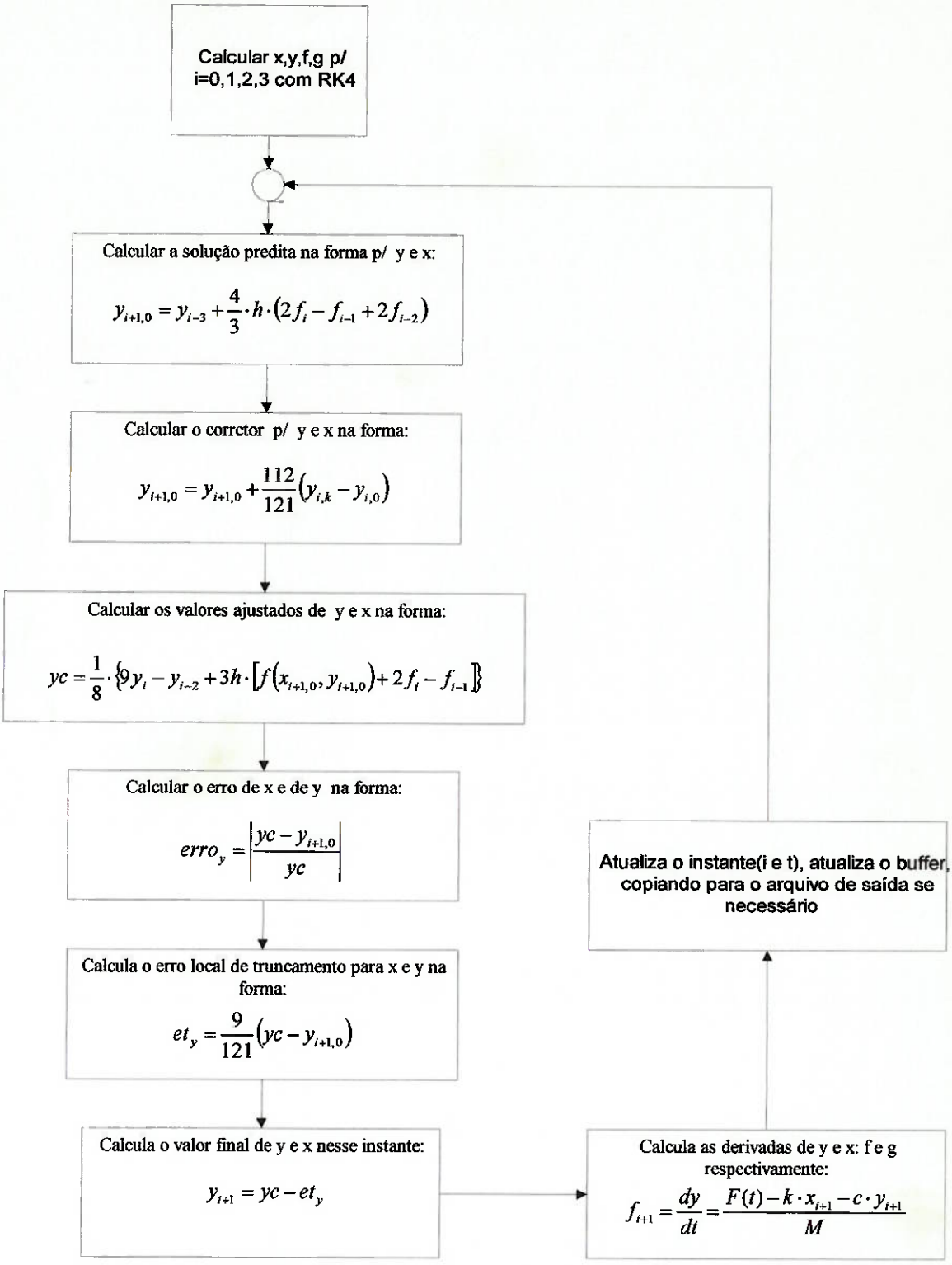


Dessa forma será resolvido um sistema de duas equações diferenciais de 1ª ordem. Uma das necessidades do método durante o cálculo é a avaliação(cálculo) do valor de $\dot{X}$ e de $\dot{Y}$, que estão expressos acima. Esses valores(que são a primeira e segunda derivada da posição em relação ao tempo) serão referenciados por g e f, respectivamente. Assim,

- X é a posição da massa
- g e y são a velocidade dela ($\dot{x}$)
- f é a aceleração ($\ddot{x}$)

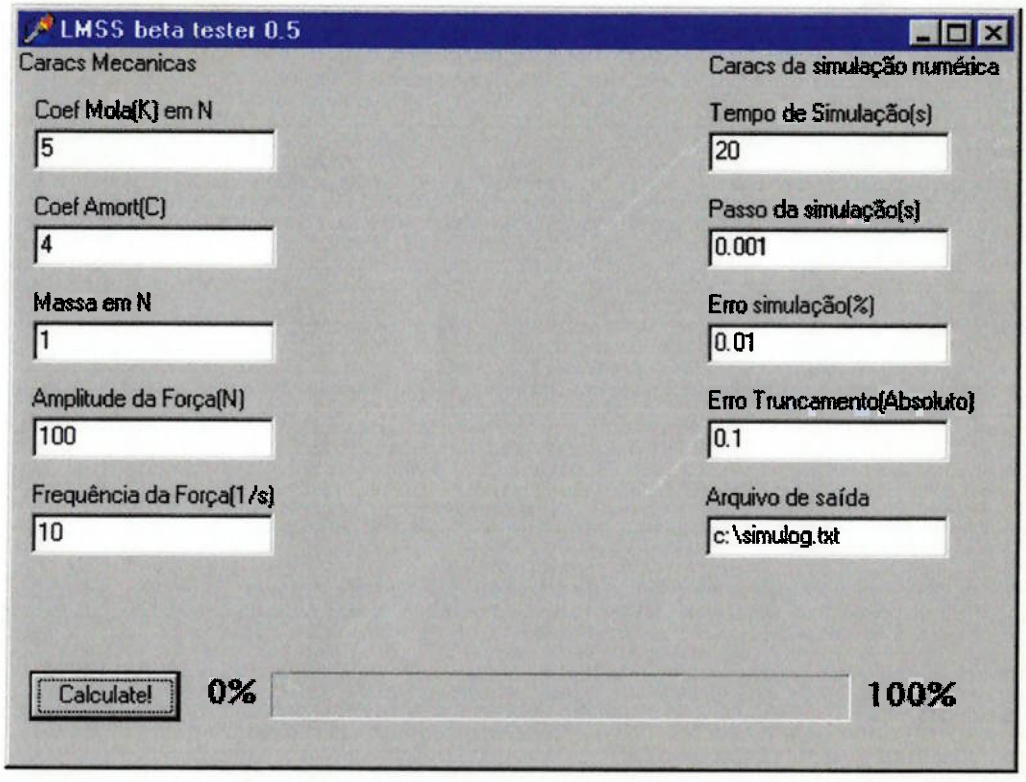
Segue abaixo fluxograma explicitando o método de Hamming para uma massa com as variáveis definidas acima. É importante notar que o Método de

Hamming se referencia a três estados anteriores do sistema para estimar o próximo, sendo necessário inicialmente estimar três estados através de outro método numérico, como Runge Kutta, por exemplo.


Calcular x,y,f,g p/ i=0,1,2,3 com RK4
Calcular a solução predita na forma p/ y e x:
$y_{i+1,0} = y_{i-3} + \frac{4}{3} \cdot h \cdot (2f_i - f_{i-1} + 2f_{i-2})$
Calcular o corretor p/ y e x na forma:
$y_{i+1,0} = y_{i+1,0} + \frac{112}{121}(y_{i,k} - y_{i,0})$
Calcular os valores ajustados de y e x na forma:
$yc = \frac{1}{8} \cdot \{9y_i - y_{i-2} + 3h \cdot [f(x_{i+1,0}, y_{i+1,0}) + 2f_i - f_{i-1}]\}$
Calcular o erro de x e de y na forma:
$erro_y = \left| \frac{yc - y_{i+1,0}}{yc} \right|$
Calcula o erro local de truncamento para x e y na forma:
$et_y = \frac{9}{121}(yc - y_{i+1,0})$
Calcula o valor final de y e x nesse instante:
$y_{i+1} = yc - et_y$
Calcula as derivadas de y e x: f e g respectivamente:
$f_{i+1} = \frac{dy}{dt} = \frac{F(t) - k \cdot x_{i+1} - c \cdot y_{i+1}}{M}$
Atualiza o instante(i e t), atualiza o buffer, copiando para o arquivo de saída se necessário

O software desenvolvido no primeiro semestre para testes, o LMSS beta tester (Lumped Mass Spring Solver) é uma aplicação direta do algoritmo acima. Ele possibilitou o teste do algoritmo numa condição simplificada e a familiarização com alguns problemas que poderiam surgir.

O LMSS cuja tela se vê abaixo foi criado com o único propósito de efetuar testes de funcionalidade e potenciais correções de rumo antes da construção do software final. Este protótipo calcula numericamente através do método de Hamming a resposta de um sistema mecânico massa-mola-amortecedor submetido a uma força externa(no caso uma força senoidal). A interface atualmente permite definir o passo do cálculo, o arquivo de saída e as características mecânicas do sistema(K,C,M,w,F).Segue abaixo figura da interface do protótipo.



4.3 Múltiplos graus de liberdade

Não há muito o que se discutir sobre o algoritmo de Hamming. Esta é a forma proposta por ele e utilizada há muitos anos. Como então utilizá-lo para um sistema de equações absolutamente interligado com diversos graus de liberdade ?

As seguintes considerações devem ser feitas sobre o algoritmo:

- Para progredir de um segmento de cálculo(box) para outro, antes é necessário ter realizado a anterior para todas as massas
- Dentro de cada box de cálculo as operações de cada massa são independentes uma das outras, fazendo com que o sistema possa ser resolvido simplesmente com um *loop* para todas as massas sequencialmente.
- Existe uma exceção ao cálculo independente. Note que no box abaixo para calcular o valor ajustado de y(dx/dt) e x é necessário

Calcular os valores ajustados de y e x na forma:

$$yc = \frac{1}{8} \cdot \{9y_i - y_{i-2} + 3h \cdot [f(x_{i+1,0}, y_{i+1,0}) + 2f_i - f_{i-1}]\}$$

conhecer os valores de "g" e "f" (velocidade e aceleração) nos dois instantes imediatamente anteriores(i e i-1) e também os calculados com base na posição e velocidade calculados originalmente no início desse passo. Para calcular essas derivadas, é necessário levar em conta todas as interações entre os elementos do sistema, fazendo com que para um sistema genérico esse box tenha que ser quebrado

em dois processos : cálculo das acelerações de todas as massas do sistema e então o cálculo desse box.

- Além disso existem todas dificuldades inerentes ao correto equacionamento de um sistema absolutamente genérico, como projeções de forças, resolver o problema de cálculodas acelerações para este sistema, que serão brevemente discutidos mais adiante.

5. A estrutura do software Mechcalc 2.0

Neste capítulo será apresentado o software MechCalc 2.0 de minha autoria. Inicialmente será discutida o que há por trás dele, ou seja, sua estrutura de dados, os módulos do programa em especial o módulo acelcalc, arquivos de sistema e de dados. Após serão apresentadas suas características na versão atual e possibilidades imediatas e futuras de expansão das capacidades do programa.

5.1 Estrutura de dados

O software possui um pré-processador de sistemas massa-mola-amortecedor integrado. Dessa forma podemos dividir razoavelmente as estruturas de dados entre estruturas *graphic* e *data*. As estruturas graphic dizem respeito ao pré-processador e as data são as relativas ao processamento dos dados.

5.1.1 Estruturas Graphic

Foi criado um dispositivo COM(Component Object Model – padrão windows) gráfico que comportasse o display de todos os elementos de interesse(Massa, mola, amortecedor, restrição e força). Este elemento tem propriedades como largura, altura e posição x e y do seu canto superior esquerdo. Essas propriedades dizem respeito ao display da tela em pixels, por isso para fazer o display na tela de um elemento é necessário converter a sua posição real(x,y) para o sistema de coordenadas da tela.

Além disso, existem outra variáveis como tipo, cor e outros que não necessitam de maior detalhamento.

Assim, temos os seguintes array(vetores) de elementos gráficos:

- GMASSA – el. Gráfico de massa e de restrição
- GMOLA – el. Gráfico de mola
- GAMORT – el. Gráfico de amortecedor
- GFORCA – el. Gráfico de força

As principais variáveis destes são height, width, top, left e teta, através das quais eles são posicionados e tem seu tamanho definido.

5.1.1 Elementos data

Os elementos data são mais complexos e os principais são Forca, Massa, Amort e Mola.

a) Forca[1..Nforca] : Cada forca tem as seguintes variáveis :

- N – índice
- El1 – Elemento a que está aplicada
- T0 e Tf – intervalo de tempo em que estará ativa.
- Teta – Angulo da direção da força em relação ao eixo x
- A, w – Coeficientes de força senoidal ou cossenoidal. ex: F=A.sen(w.t)
- P[0..29] – 29 coeficientes para força polinomial da forma F= a0+a1.t + a2. t^2...+a29.t^29

b) Amort[1..Namort]: Cada amortecedor tem as seguintes variáveis:

- N – índice
- El1, El2 – Elementos a que estão ligadas
- C – coeficiente de amortecimento

c) Mola[1..Nmola]: Cada mola tem as seguintes variáveis

- N – Índice
- El1, El2 – Elementos a que está ligada
- L – comprimento inicial
- K1, K2, K3 – Coeficientes de mola(no momento somente possível utilizar K1)

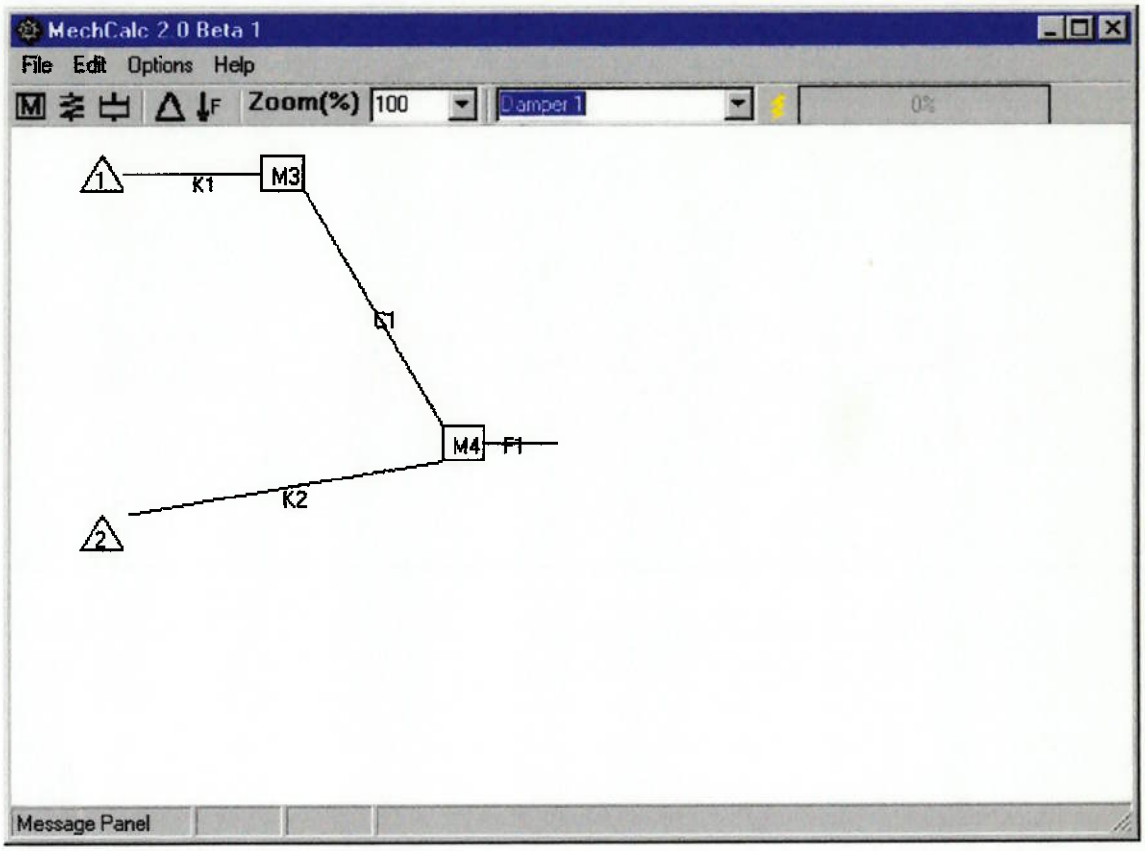
d) Massa[1..Nmassa][-4..3] : Cada massa tem um buffer de seus dados de 7 instantes(-4 a 3). É nessa estrutura que estão os principais dados do sistema. As principais variáveis são:

- N – Índice
- Massa – Massa em N
- Tipo – Massa ou Restrição
- X, Y – Posição em x e y
- VX, VY – Velocidade em x e y
- AX, AY – Aceleração em x e y
- Fxmola, Fymola – Forças em x e y devido a molas
- Fxamort, Fyamort - Forças em x e y devido a amortecedores
- Fxt, Fyt - Forças em x e y devido a forças aplicadas no elemento
- Errox, Erroy, ErroVx, ErroVy, etx, ety, etvx, etvy – Erro no cálculo de diversas variáveis.

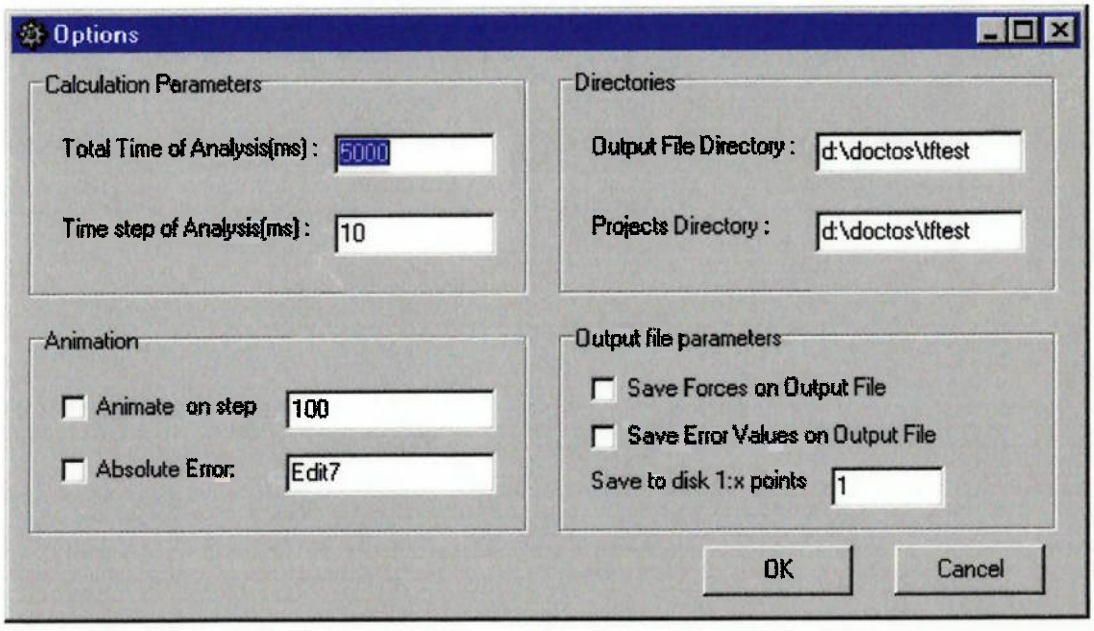
5.2 Divisão do programa

O software está dividido em 11 subprogramas. Cada um deles é responsável pelo gerenciamento de um *form*. Um form é uma tela de diálogo. Assim:

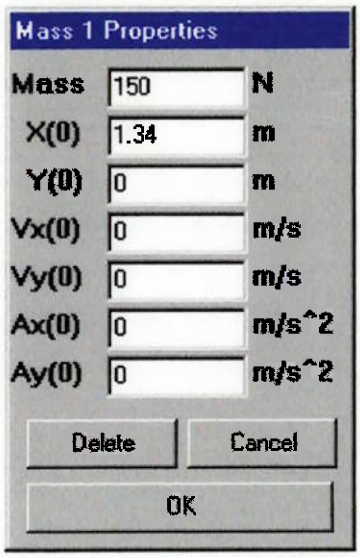
Unit1.pas – Janela principal



Unit2.pas – Janela de opções

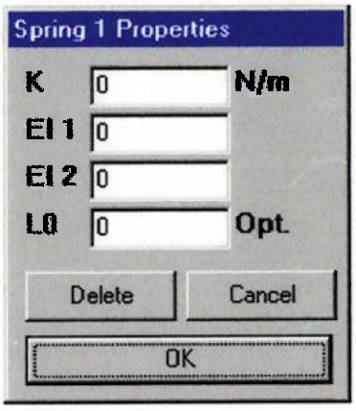


Unit3.pas – Janela das Massas



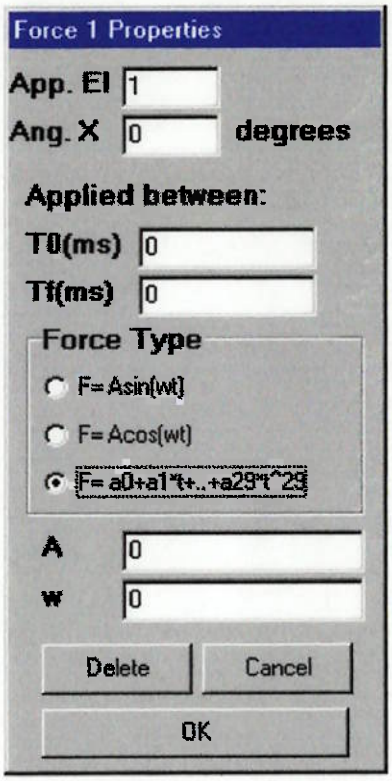
Unit4.pas- Janela dos amortecedores

Unit5.pas – Janela das molas

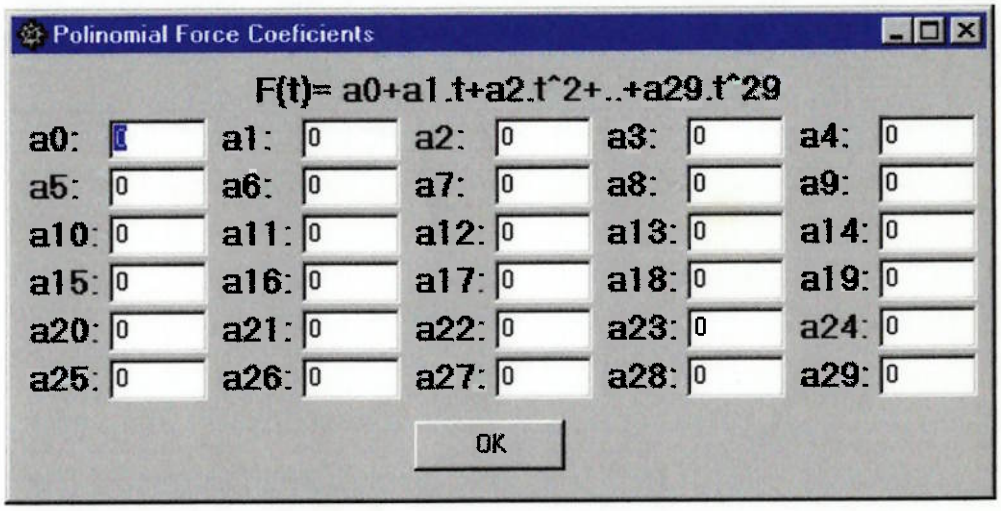


Unit6.pas – Janela das restrições

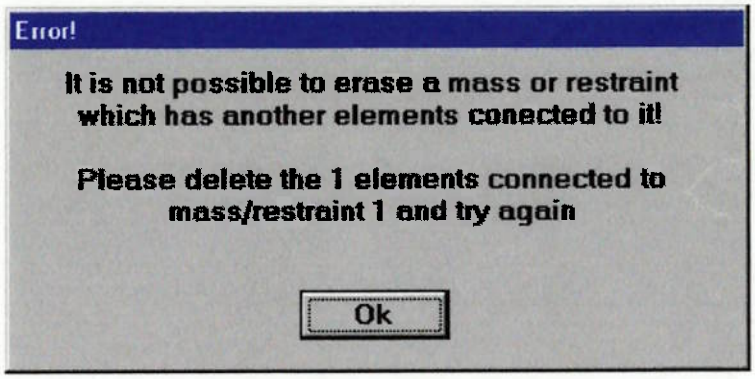
Unit7.pas – Janela das forças



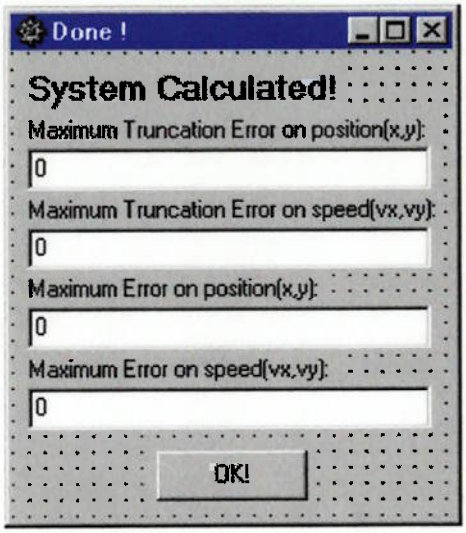
Unit8.pas – Janela dos coeficientes de força polinomial



Unit10.pas – Janela de erro devido a tentativa de remoção de massa conectada a outros elementos



Unit11.pas – Janela de final de cálculo



5.3 Módulos do programa

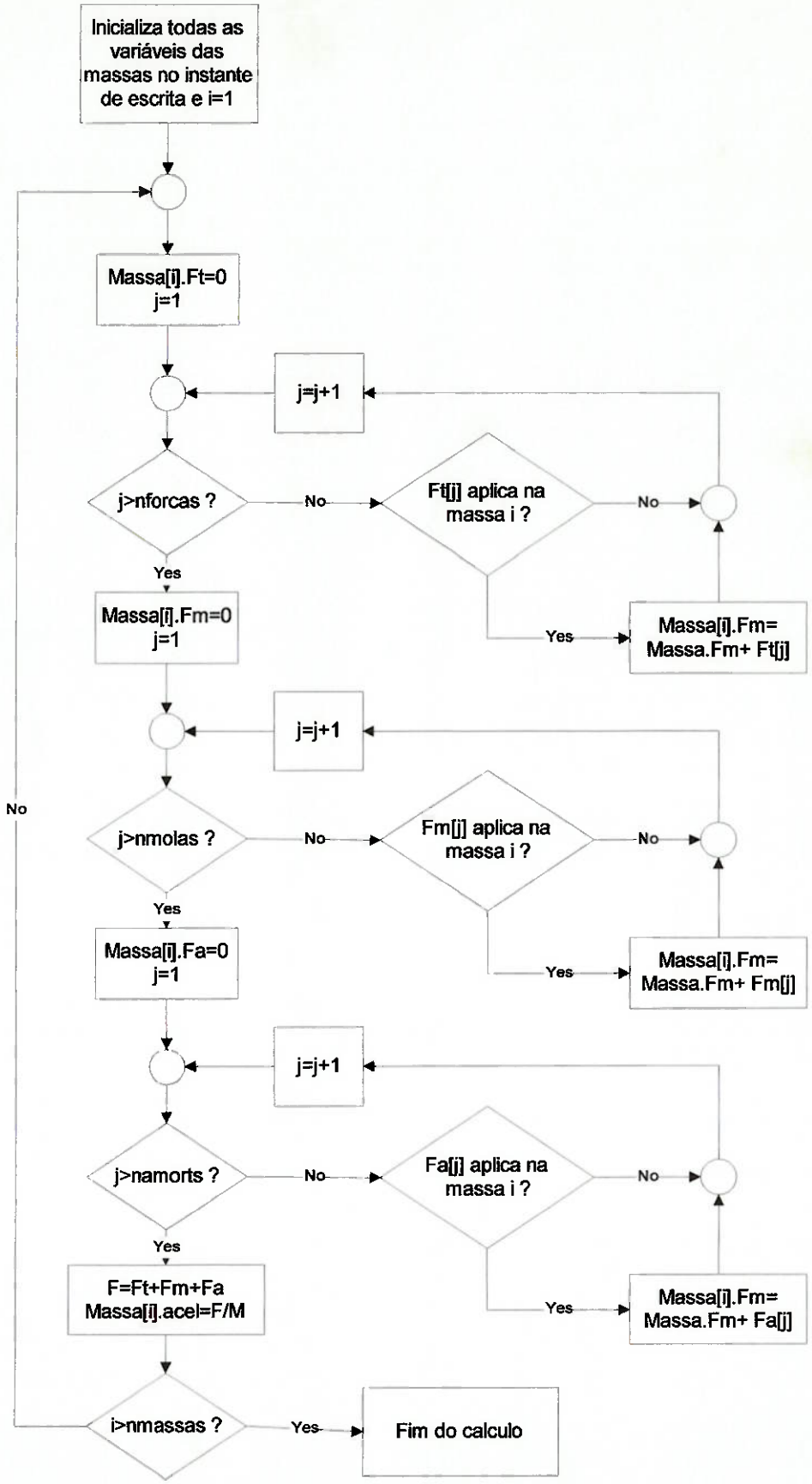
Apesar de estar dividido em 10 subprogramas, o núcleo do software, tanto do ponto de vista de processamento quanto do ponto de vista gráfico é o Unit1.pas. Poucas são as rotinas contidas nos outros units que são realmente importantes. Abaixo seguem as funções das rotinas do Unit1:

- Procedure **zera(instante)** – Zera todos os valores das massas (posição, velocidade, etc.) no instante **instante**
- Procedure **inicializa(inst)** – Zera todos os valores das massas, inclusive características mecânicas como massa no instante **inst**
- Procedure **angulocalc(x1,y1,x2,y2)** – Calcula em radianos o ângulo em relação ao eixo x da direção entre dois elementos com coordenadas **(x1,y1)** e **(x2,y2)**
- Procedure **move(inst1,inst2)** – Copia todos os valores das massas do instante **inst1** para o **inst2**
- Function **forcat(tempo,num)** – Retorna o valor da força **num** no tempo **tempo**
- Function **forcaamort(num,vel)** – Retorna o valor da força exercida pelo amortecedor **num** sob uma velocidade **vel**
- Function **forcamola(num,compr)** – Retorna o valor da força exercida pela mola **num** sob o comprimento L **compr**
- Function **conectado(num)** – Retorna o número de elementos conectados a massa/restrição **num**
- Procedure **Acelcalc(instbase,instwrite)** – Calcula os valores de aceleração(segunda derivada) com base na situação do sistema no instante **instbase** e escreve no instante **instwrite**.
- Procedure **TForm1.Calculate1Click** – Rotina Principal, faz o cálculo dos quatro primeiros instantes(-3 ,-2,-1 e 0) através de Runge Kutta de 4ª ordem e dos demais através do método de Hamming

- Procedure **TForm1.Open1Click** – Rotina que permite a escolha e leitura de um arquivo "project" ou de sistema e os gera na matriz gráfica.
- Procedure **TForm1.SaveAs1Click** – Rotina que permite a gravação de um sistema desenhado na tela para um arquivo
- Procedure **TForm1.Graphrefresh** – responsável por desenhar os elementos do sistema no *workspace* em suas posições e atualizá-las sempre que necessário.
- Procedure **TForm1.ToolButton1Click** – Mostra a janela de opções e cria o elemento de dados e gráfico de uma massa
- Procedure **TForm1.ToolButton2Click** – Mostra a janela de opções e cria o elemento de dados e gráfico de uma mola
- Procedure **TForm1.ToolButton3Click** – Mostra a janela de opções e cria o elemento de dados e gráfico de um amortecedor
- Procedure **TForm1.ToolButton9Click** – Mostra a janela de opções e cria o elemento de dados e gráfico de uma restrição vert/hor
- Procedure **TForm1.ToolButton4Click** – Mostra a janela de opções e cria o elemento de dados e gráfico de uma força

5.4 O Módulo Acelcalc

Como visto anteriormente no item 4.3, um dos maiores problema para solucionar um sistema genérico como proposto é com relação ao cálculo da segunda derivada (acelerações) do sistema. A solução pode ser melhor visualizada através do fluxograma da página seguinte.


Inicializa todas as variáveis das massas no instante de escrita e i=1
Massa[i].Ft=0
j=1
j=j+1
j>nforcas ?
No
Ft[j] aplica na massa i ?
No
Yes
Massa[i].Fm=0
j=1
Yes
Massa[i].Fm= Massa.Fm+ Ft[j]
j=j+1
j>nmolas ?
No
Fm[j] aplica na massa i ?
No
Yes
Massa[i].Fa=0
j=1
Yes
Massa[i].Fm= Massa.Fm+ Fm[j]
j=j+1
j>namorts ?
No
Fa[j] aplica na massa i ?
No
Yes
F=Ft+Fm+Fa
Massa[i].acel=F/M
Yes
Massa[i].Fm= Massa.Fm+ Fa[j]
i>nmassas ?
Yes
Fim do calculo
No

5.5 Arquivos gerados pelo Mechcalc 2.0

O Mechcalc 2.0 gera dois tipos de arquivos. Arquivos "project" (.mpr) contém um sistema massa-mola-amortecedor previamente desenhado. A saída das respostas do sistema é feita através de um arquivo texto facilmente importável para qualquer planilha eletrônica do mercado.

5.5.1 Arquivo Project

O arquivo project grava as informações do sistema da seguinte forma:

Número de massas

Número de molas

Número de amortecedores

Número de forças

Linha com informações da massa 1

Linha com informações da massa 2

Linha com informações da massa N

Linha com informações da mola 1

Linha com informações da mola 2

Linha com informações da mola N

Linha com informações do amortecedor 1

Linha com informações do amortecedor 2

Linha com informações do amortecedor N

Linha com informações da força 1

Linha com informações da força 2

Linha com informações da força N

E as informações das linha das massas sâo:

Massa x0 y0 vx0 vy0 ax0 ay0 tempor, onde tempor=1 é massa

tempor=2 é restrição vert/hor

massa é massa em N

x0,y0 são posição em t =0

vx0,vy0 são velocidade em t =0

ax0,ay0 são aceleração em t =0

As informações das linhas das molas são:

K1 EL1 EL2 , onde K1 é o coeficiente de mola(N/m)

EL1,EL2 são os elementos a que a mola está ligada

As informações das linhas dos amortecedores são:

C EL1 EL2, onde C é o coeficiente de amortecimento (N.s/m)

EL1,EL2 são os elementos a que o amortecedor está ligado

E as informações das linhas das forças são :

t0 tf teta el1 tempor a w

ou

t0 tf teta el1 tempor p0 p1 p2 p3 p4 p5 .... P30,

onde t0,tf é o intervalo de tempo de aplicação da força

teta é o ângulo da força em relação ao eixo x

tempor=1 significa força senoidal

tempor=2 significa força cossenoidal

tempor=3 significa força polinomial

a,w são os coeficientes de força senoidal/cossenoidal

p0,p1..p30 são os coeficientes da força polinomial

5.5.2 Arquivo de saída de dados

O arquivo de saída de dados grava obrigatoriamente a posição, velocidade e aceleração de cada massa ao longo do tempo e pode ainda gravar as forças aplicadas em cada massa e os erros de cálculo. Isso é gravado em colunas da seguinte forma:

Tempo dadosmassa1 errosmassa1 forçasmassa1 * dadosmassa2 ....

Observando os dados de massas diferentes estarão separados por uma coluna de asteriscos. Além disso é importante notar que tanto a gravação dos erros quanto das forças aplicadas ao longo do tempo é opcional.

Os dadosmassa são distribuídos da seguinte forma:

X VX AX Y VY AY

Os errosmassa são distribuídos assim:

Errox errovx erroy errovy etx etvx ety etvy,

onde t é de truncamento e são so erros mais importantes a observar.

As forçasmassa são gravadas da seguinte forma:

Fxt Fxmola Fxamort Fyt Fymola Fyamort

6. O Software Mechcalc 2.0

O software Mechcalc no seu estado Beta 1 proporciona os recursos abaixo :

- Resolução de sistemas bidimensionais compostos de massa, molas e amortecedores
- Obtenção de posição, velocidade, aceleração, força ativas(F(t)) e reativas(fxmola,fxamort) de cada massa ao longo do tempo
- Desenho do sistema através de interface gráfica, com facilidades de uso como menus que surgem a partir de duplo-clique nos elementos
- Possibilidade de gravar o sistema desenhado para arquivo
- Forças aplicadas senoidais, cossenoidais ou polinomiais de até 29 termos
- Interface gráfica amigável
- Monitoramento do erro de cálculo

Como melhorias a serem implementadas estão:

- Criação de rotinas de detecção de erros de operador e janelas de correção correlatas, tornando-o *idiot proof*.
- Existe espaço para evolução no método de cálculo, podendo melhorar ainda mais o seu desempenho.
- Implementação com múltiplos métodos de resolução. A modularidade do código permite que sejam implementadas soluções através de Runge Kutta ou Milne, por exemplo.

- Implementação de versão para sistemas multiprocessados
- Implementação de animação do sistema durante o cálculo
- Pós-processador para análise com máscara de elementos.
- Implementação de materiais através de uma massa, uma mola e um amortecedor, com regime plástico através de múltiplos coeficientes de mola.
- Módulo para simulação e reconstituição de acidentes.

Uma versão pre-release com mais testes efetuados e todas as implementações de características idiot-proof estará disponível na internet para download a partir de março de 1999 no site www.brunoassaf.com .

7. Testes Funcionais

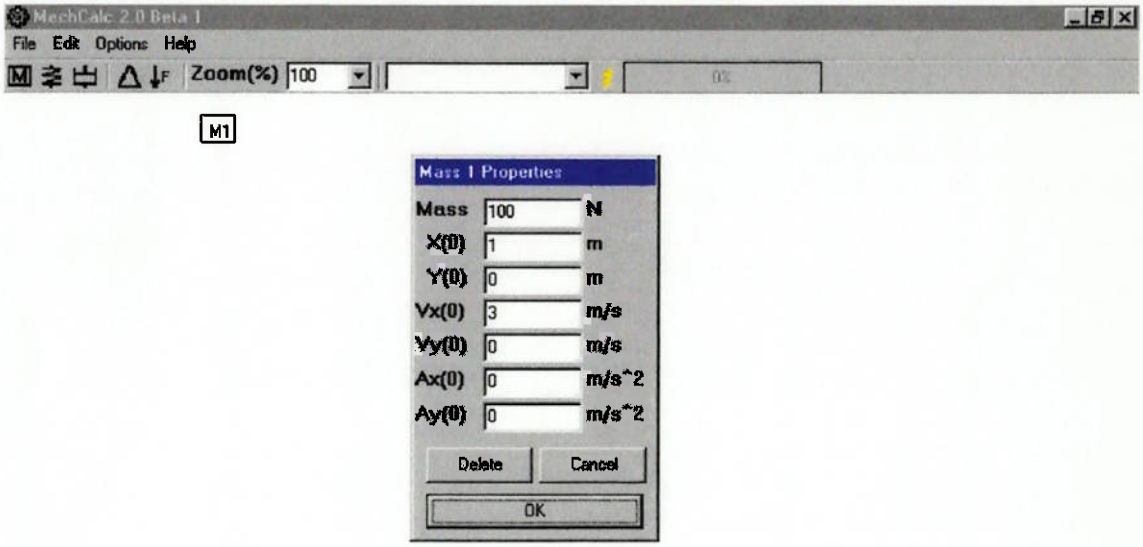
Os testes funcionais do software podem ser divididos em alguns tipos. Serão efetuados em todos os eixos até a versão final de release e os tipos são:

- Cálculo com massa livre submetida a velocidade inicial
- Cálculo com massa livre submetida a força constante
- Cálculo com mola
- Cálculo com mola e amortecedor
- Stress testing

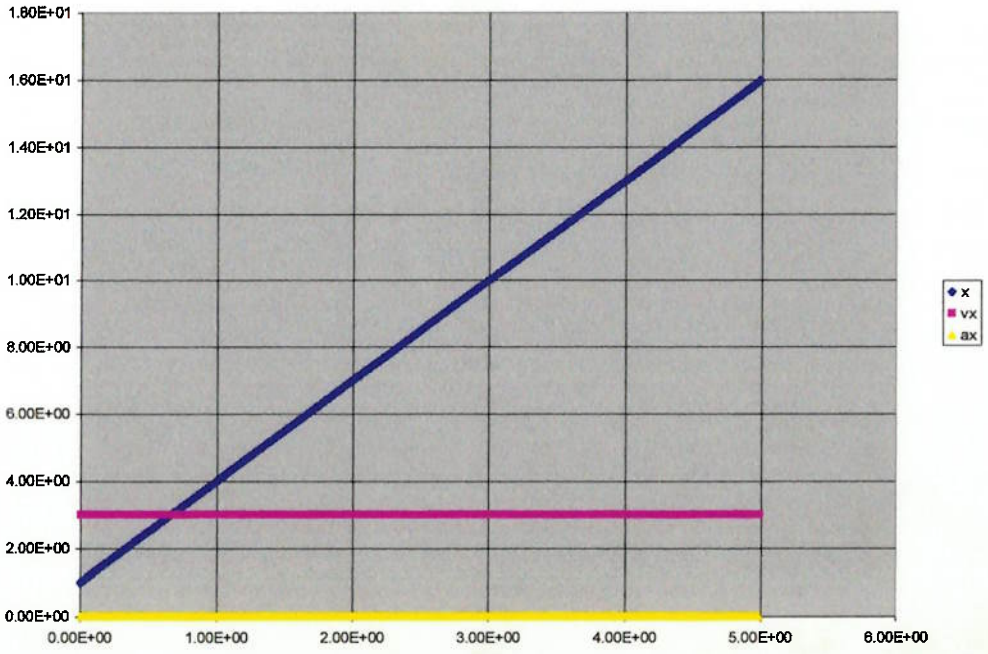
No atual estágio do software beta 1 release 2, foram efetuados os 4 primeiros tipos de teste para o eixo x, cujos resultados serão mostrados a seguir:

7.1 Cálculo com massa livre submetida a velocidade inicial

Foi calculado o sistema abaixo:



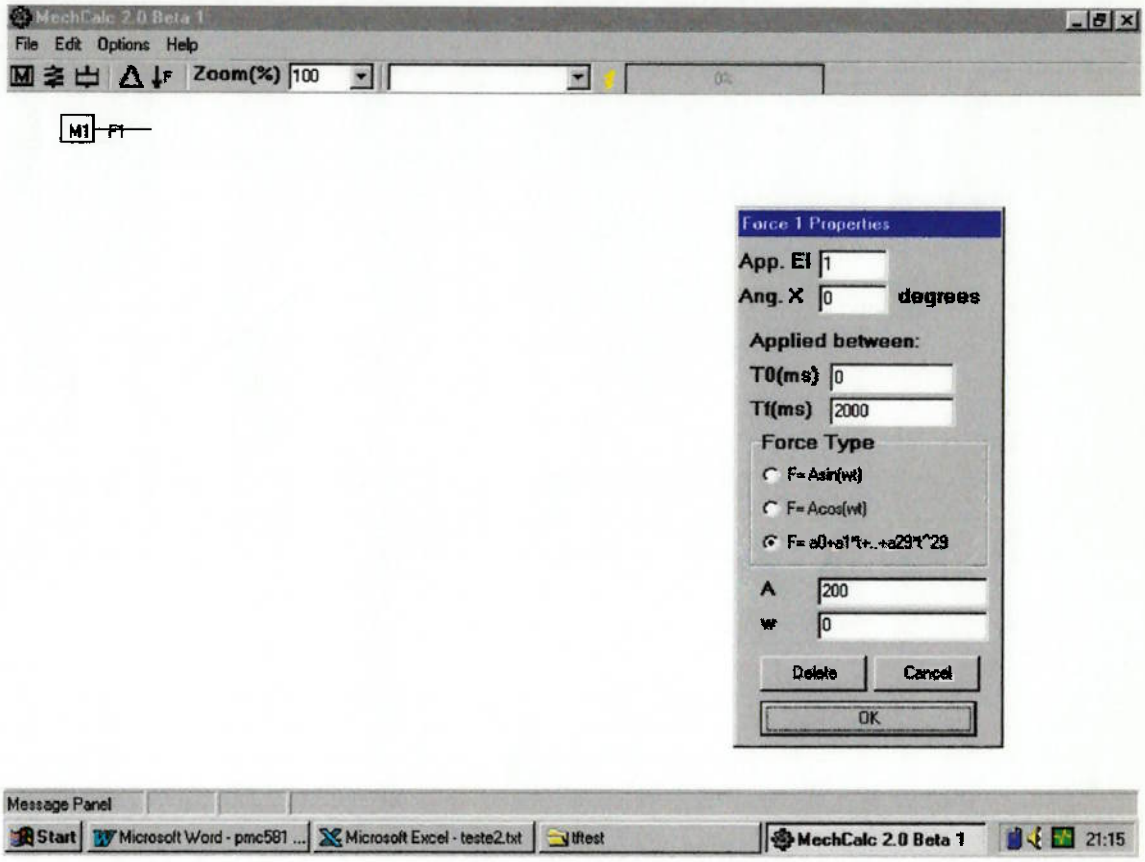
E a resposta calculada foi:



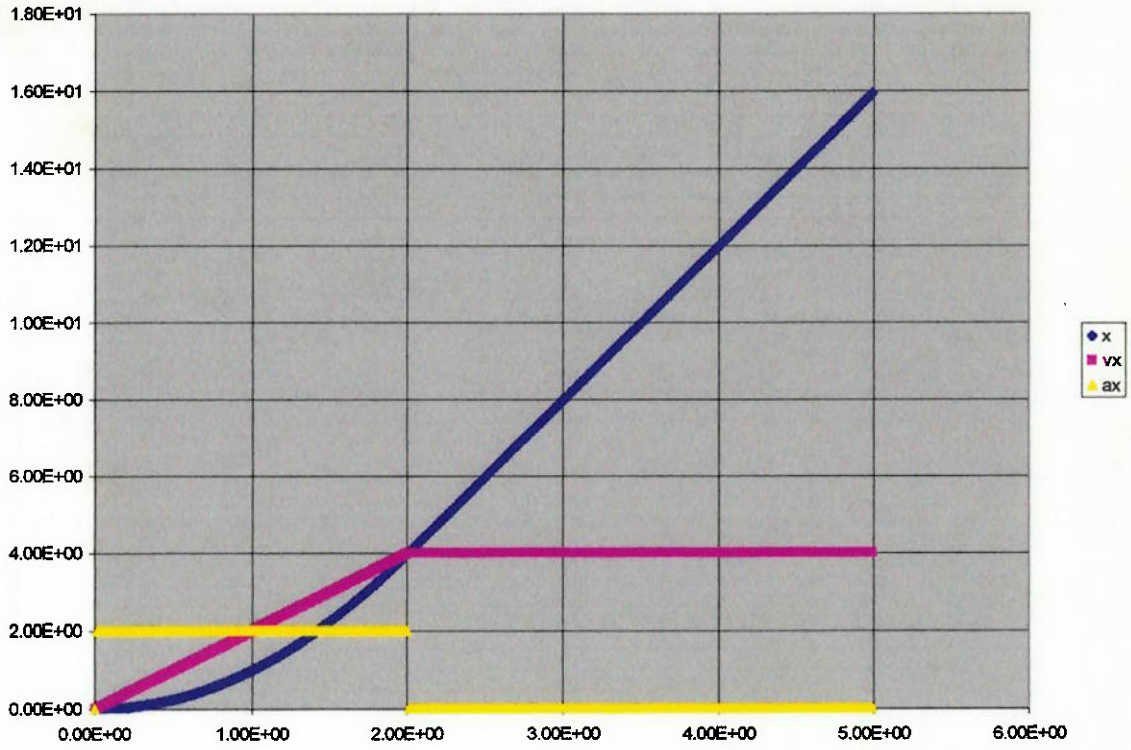
Onde pode-se ver claramente que o resultado foi correto pois a aceleração em amarelo manteve-se em zero, a velocidade em roxo em 3 m/s e a posição variou linearmente indo de x=1 m p/ t=0 para x=16 m em t=5.

7.2 Cálculo com aceleração(força) aplicada à massa

Foi calculado o sistema abaixo, com M=100 N, e F= 200 N de t=0 a t=2.



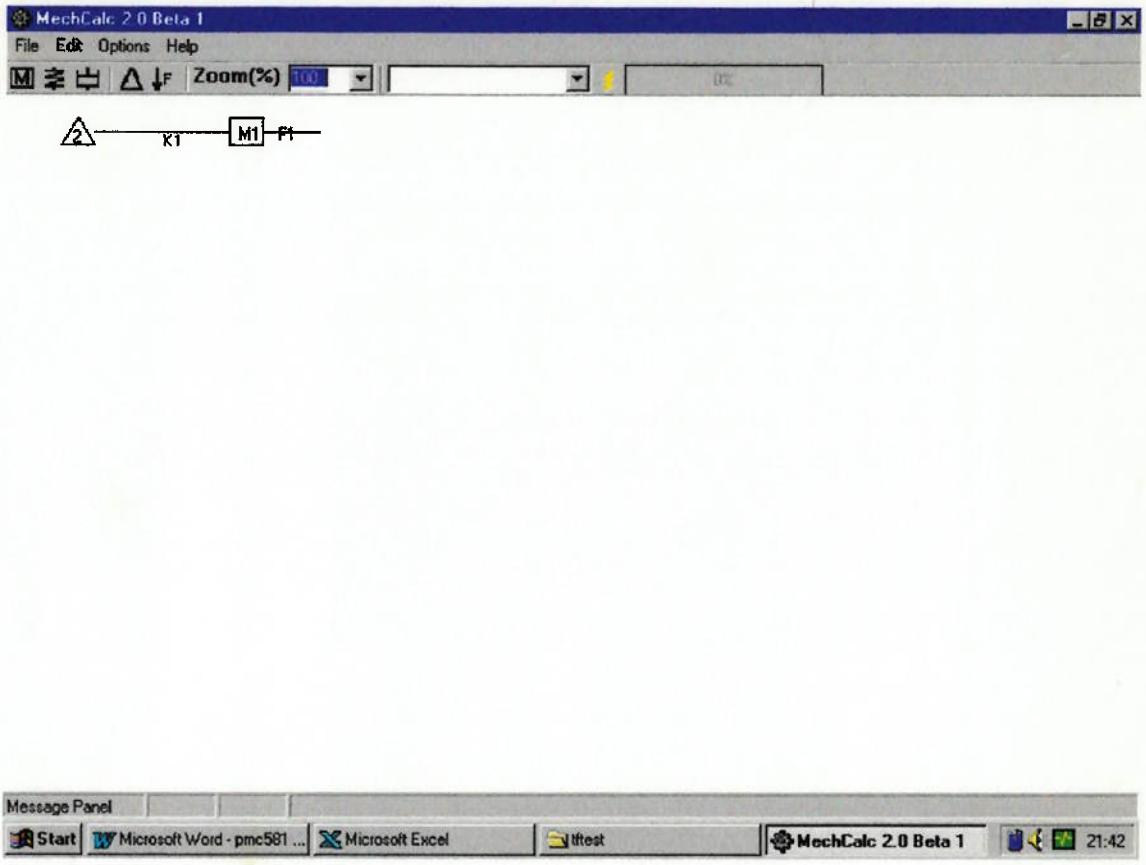
E o resultado pode ser visto abaixo:



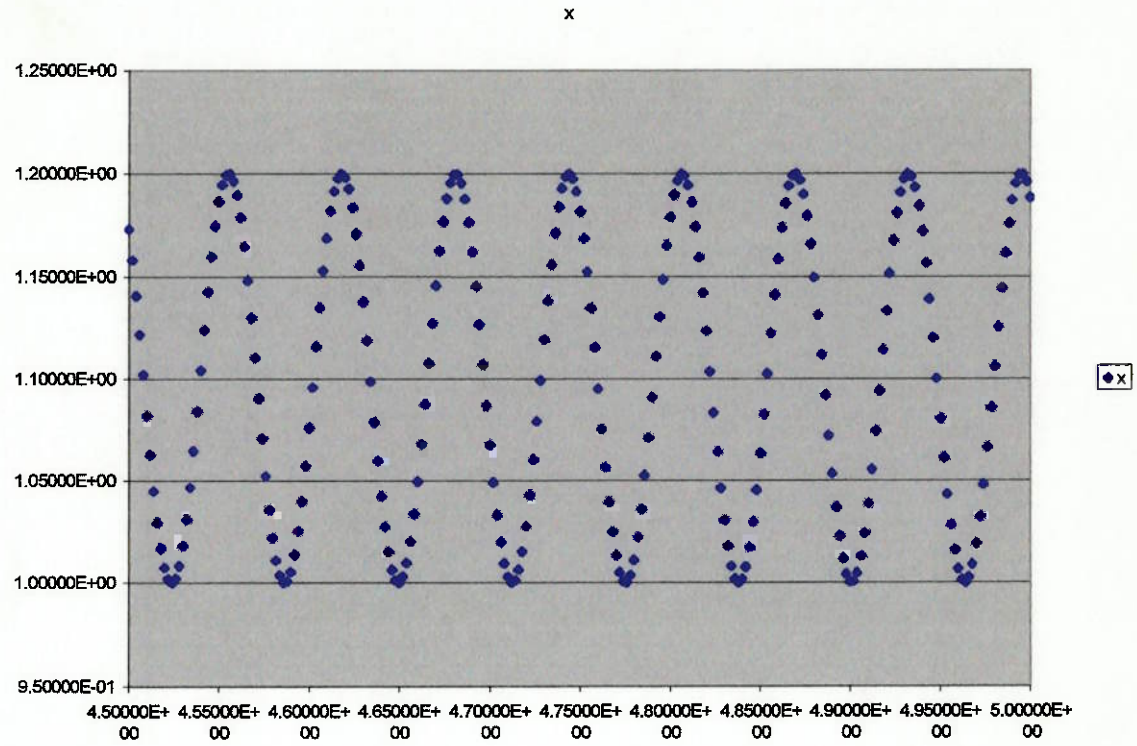
Esses resultados também mostram a perfeita correlação com a realidade pois a aceleração(amarelo) se mantém em 2 m/s^2 durante t=0s a t=2s, proporcionando um aumento linear da velocidade (roxo) de 0 para 4 m/s nesse período, mantendo-se constante depois disso. A posição aumenta de forma parabólica até t=2 e depois de forma linear. Os valores também mostram-se adequados pois $S=so+v0.t+a.t^2/2$, portanto p/ a=2 e t =2 e v0=0 temos S=4 m. Apos isso s=s0 + vt ou s= 4+4.t o que resulta em s=16m.

7.3 Cálculo com mola no sistema

Foi resolvido um sistema com uma massa de 1 N conectada a uma restrição horizontal através de uma mola de k=10kN /m e a massa submetida a uma força constante de 1 kN.



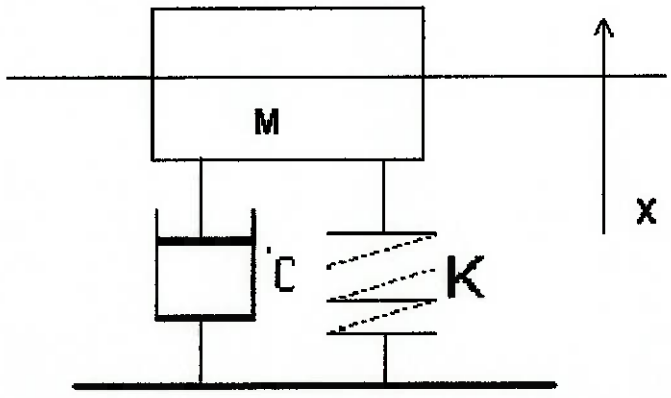
Os resultados calculados seguem abaixo:



O que mostra uma oscilação em torno do ponto de "equilíbrio"de 1.1 m o que é razoável pois não há nada que esteja dissipando a energia.Mas não iremos nos ater muito a este exmplo, passando diretamente para o caso com mola e amortecedor.

7.6 Cálculo com mola e amortecedor

Foi calculado um sistema como o abaixo:

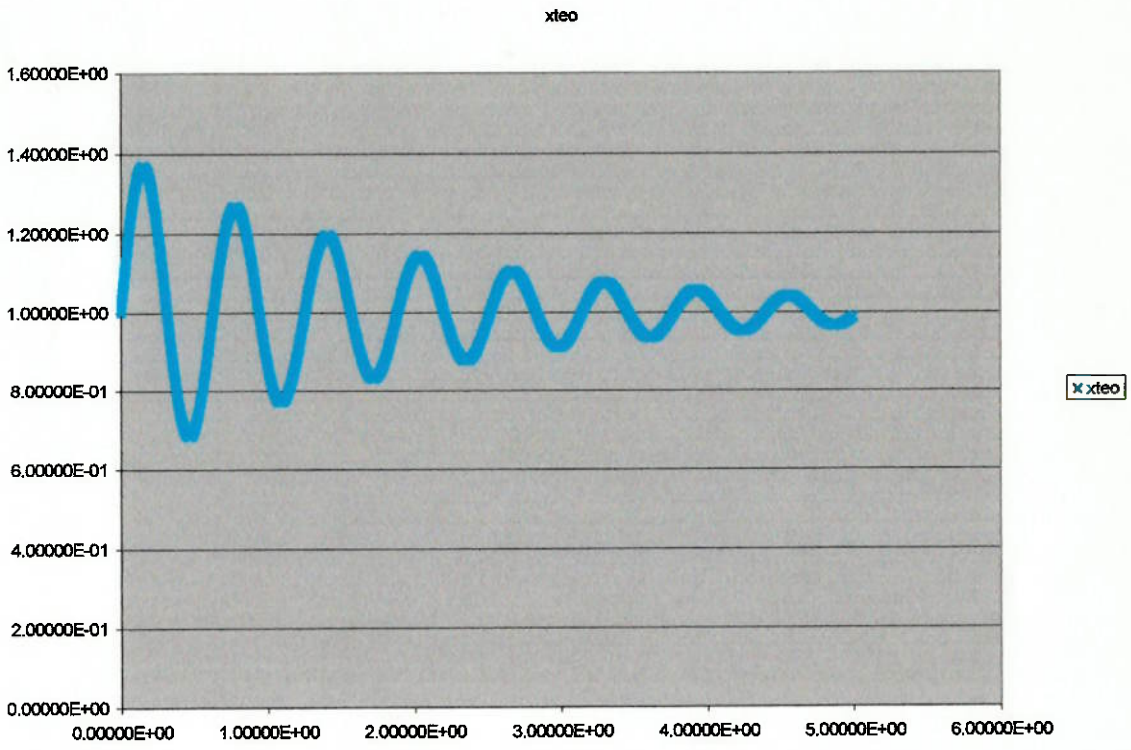


a) Com M=100, K=10000 e C=100, V(0) = 4 temos Wn=10 e γ=0,05 ou seja, sistema com amortecimento subcrítico.

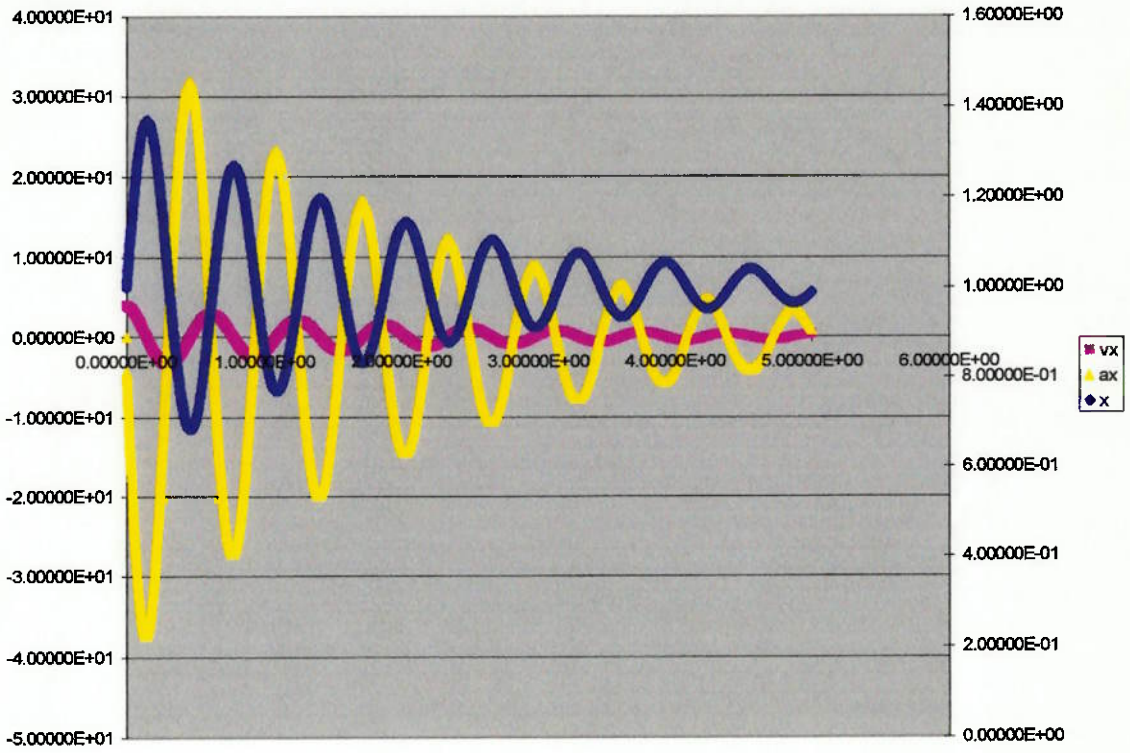
b) Com M=100, K= 10000, C= 2000, V(0) = 4 temos Wn=10 e γ=1 ou seja, sistema com amortecimento crítico.

Comparação dos resultados do software com os teóricos:

Resultado teórico para "a"

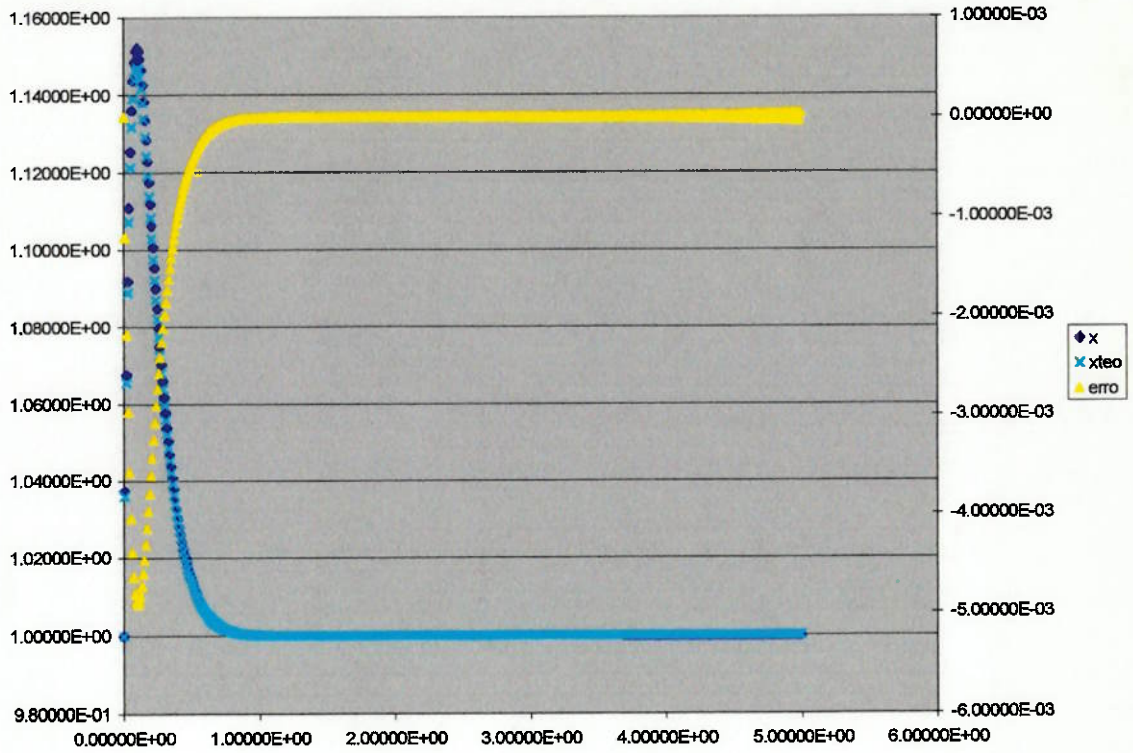


Resultado obtido com o software:



Onde pode-se observar que as curvas de posição teórica e prática são absolutamente idênticas.

Comparação no caso b temos o gráfico abaixo:



Onde é possível ver claramente que as linhas azul escura e clara(x dado pelo software e obtido teóricamente) se sobrepõem. Mais do que isso, o erro indicato em amarelo cujos valore estão indicados na segunda escala, à direita fica entre 0 e 5 x 10^{-3}, valor perfeitamente razoável. Além disso a curva qualitativamente mostra um comportamento de sistema amortecido criticamente.

7. Análise Econômica

Esta fase, muitas vezes desprezada, na verdade uma das mais importantes na definição da viabilidade do projeto. A seguinte análise foi feita tendo com base um software que seja uma pequena evolução do atual, não demandando muito tempo na sua evolução, mas apenas acertando algumas arestas que certamente estarão presentes no final deste primeiro projeto. Para estudar a viabilidade financeira é necessário passar pelas etapas de analisar os custos, as possibilidades de mercado/concorrência, estabelecer um preço razoável e verificar se o produto é viável finaceiramente.

7.1 Análise de Custos

No desenvolvimento do software estima-se a ocupação de 300 horas de programação de um engenheiro capacitado e com formação de software

- Atual remuneração deste: R$ 40/h + encargos = R$ 70/h
- Custo = 300 h x R$ 70/h = R$ 21.000

7.2 Análise de Mercado

Para aplicações e investimentos feitos no país, um tempo de retorno usual utilizado é de 1 ano. A atual concorrência é baseada principalmente em soluções FEA a partir de US$ 4.000 por licença. Levou-se em conta o custo de oportunidade(Taxa de juros no Brasil: 28 % ao ano) e uma incidência de impostos/ encargos + IR + ISS e outros impostos que onere o produto em cerca de 30 %

Além disso, estima-se uma venda de um cliente por mês, em média três licenças

7.3 Precificação

- Custo : R$ 21.000
- Se investido a 28% de juros/ano = R$ 26.680
- Adicionando encargos (30%) = R$ 34.944
- Frente a soluções FEA mais precisas, mas com maior demanda computacional, tratando-se de uma solução inovadora sem penetração de mercado, um preço competitivo seria da ordem de 25 % de soluções FEA ou R$ 1.200(US$1000).
- Estimando 3 licenças por mês temos Receita:

R = 3 x 12 x 1200 = R$ 43.200

ML= (R-C)/R = 19,1 % (Margem de Lucro)

Além disso é interessante lembrar que como vantagem de mercado também há o fato do software ser projetado para um plataforma de hardware 50% mais barata do que as *workstations* tradicionais. Tendo em vista esses resultados percebe-se que o produto em questão pode vir a ter ótimas chances no mercado.

8. *Bibliografia*

KAEAGIOZOVA, D. ; JONES, N. **Dynamic Elastic-Plastic Buckling Phenomena in Rod due to Axial Impact**. Grã-Bretanha, Int. J. Impact Engng 1996,(1)

AMERICAN IRON AND STEEL INSTITUTE, **State of the Art Review of Automobile Structural Crashworthiness**.U.S.A. , American Iron and Steel Institute, 1993

GUIDORIZZI, HAMILTON **Um Curso de Cálculo**. vol.4, Livros Técnicos e Científicos Editora, São Paulo,1988

COOPER, D. ; CLANCY, M. **Oh! Pascal**. second edition, W. W. Norton & Company , New York,1985

LONGO, M. ; SMITH JR., R. ; POLITSCHUCK, D. **Delphi 3 Total – Dominando a Ferramenta**. Brasport Livros e Multimídia Ltda., São Paulo, 1997

SETO, WILLIAM **Vibrações Mecânicas**. Ed. Mcgraw-Hill do Brasil Ltda., Rio de Janeiro, 1971

9. *Listagem do software LMSS – Protótipo do primeiro semestre*

Segue abaixo a listagem do LMSS.pas, que é o que gerou o LMSSPR.exe

```
unit LMSS;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, ComCtrls;

type
 TForm1 = class(TForm)
  Button1: TButton;
  ProgressBar1: TProgressBar;
  Label1: TLabel;
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  Edit4: TEdit;
  Edit5: TEdit;
  Edit6: TEdit;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
```

```
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Edit11: TEdit;
    Label12: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    procedure Button1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

```
  fcao : integer;
  k,c,m,tfinal,et,erro,h,w,A : extended;
  x,y,efe,ge,etx,ety,t : array [-6..1] of extended;
  arquivo : textfile;
  arq : string;
  controle : boolean;
implementation

{$R *.DFM}

function fi(tempo : extended ; tipo : integer) : extended ;
begin
if tipo = 0 then
  begin
  fi:=w*cos(w*tempo)*A;
  end;
end;

function f(x,y,tempo : extended): extended;
begin
fcao:=0;
f:=(fi(tempo,fcao)-k*x-c*y)/M;
end;

function g(y : extended) : extended;
```

```
begin
g:=y;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
    i,z : integer;
    k1,k2,k3,k4,q1,q2,q3,q4,yp,xp,yc,xc,erroy,errox : extended;

begin
{ Início do Processamento }
{ Tratamento do aruqivo de dados }
arq:=edit11.text;
assignfile(arquivo,arq);
rewrite(arquivo);

{leitura dos dados das janelas de texto }

fcao := 0;
k := Strtofloat(edit1.Text);
c := Strtofloat(edit2.Text);
M := Strtofloat(edit3.Text);
w := Strtofloat(edit5.Text);
tfinal:= Strtofloat(edit6.Text);
erro := Strtofloat(edit8.Text);
```

```
et := Strtofloat(edit9.Text);
h:= Strtofloat(edit7.Text);
A:= Strtofloat(edit4.Text);
controle:=true;

{ Inicializacao das variaveis - buffer}
for i:=-6 to 1 do
 begin
 t[i]:=0;x[i]:=0;y[i]:=0;efe[i]:=0;ge[i]:=0;etx[i]:=0;ety[i]:=0;
 end;
{inicialização da Progress Bar }
ProgressBar1.min:=0;
ProgressBar1.max:=1000;

{Cálculo dos 3 primeiros valores através de Runge Kutta 4}
for i:=-3 to -1 do
   begin
   {Calculo dos coeficientes k e q }
   k1 := f(x[i],y[i],t[i]);
   q1 := y[i];
   k2 := f((x[i]+h*q1/2),(y[i]+h*k1/2),(t[i]+h/2));
   q2 := y[i]+h*k1/2;
   k3 := f((x[i]+h*q2/2),(y[i]+h*k2/2),(t[i]+h/2));
   q3 := y[i]+h*k2/2;
   k4 := f((x[i]+h*q3),(y[i]+h*k3),(t[i]+h));
```

```
    q4 := y[i]+h*k3;
    {Calculo dos valores no instante seguinte }
    x[i+1] := x[i]+ h * (q1 + 2*q2 +2*q3 + q4)/6;
    y[i+1] := y[i]+ h * (k1 + 2*k2 +2*k3 + k4)/6;
    t[i+1] := t[i] + h;
    efe[i+1] := f(x[i+1],y[i+1],t[i+1]);
    ge[i+1] := y[i+1];
    end;
    {Fim do calculo por RK4 }
{ Inicio do calculo por Hamming }

while (t[0] < tfinal ) do
 begin
 t[1]:=t[0]+h;
 if t[1]>0.053 then
    z:=1;
 { Solucao Predita }
 yp := y[-3] + 4*h*(2*efe[0]-efe[-1]+2*efe[-2])/3;
 xp := x[-3] + 4*h*(2*ge[0]-ge[-1]+2*ge[-2])/3;

 { Solucao Predita Modificada }
 y[1] := yp + 112*ety[0]/121;
 x[1] := xp + 112*etx[0]/121;

{Solucao Corrigida com o Corretor de Hamming }
```

```
yc := 0.125*(9*y[0] - y[-2] + 3*h*(f(x[1],y[1],t[1])+2*efe[0] - efe[-1]));
xc := 0.125*(9*x[0] - x[-2] + 3*h*(g(y[1])+2*ge[0] - ge[-1]));

{Calculo do erro da iteracao }
if yc<>0 then erroy := abs((yc-y[1])/yc)
  else erroy:=0;
if xc<>0 then errox := abs((xc-x[1])/xc)
  else errox :=0;

{Calculo do erro de truncamento local }
etx[1] := 9*(yc-y[1])/121;
ety[1] := 9*(xc-x[1])/121;

{Comparacao para determinar se o passo deve ser modificado }
if ((abs(etx[1])>et) or (abs(ety[1])>et) or (erroy > erro) or (errox > erro))
then
  begin
  { Modifica step de h para h/2 }
  h:=(h/2);
  {Passa valores de -2 para -4 }
  t[-4]:=t[-2];
  x[-4]:=x[-2];
  y[-4]:=y[-2];
  efe[-4]:=efe[-2];
  ge[-4]:=ge[-2];
```

```
etx[-4]:=etx[-2];
ety[-4]:=ety[-2];
{Passa valores de -1 para -2 }
t[-2]:=t[-1];
x[-2]:=x[-1];
y[-2]:=y[-1];
efe[-2]:=efe[-1];
ge[-2]:=ge[-1];
etx[-2]:=etx[-1];
ety[-2]:=ety[-1];
{Calcula os instantes -3 e -1 por RK4 }

{Instante -3 Calculo dos coeficientes k e q }
k1 := f(x[-4],y[-4],t[-4]);
q1 := y[-4];
k2 := f((x[-4]+h*q1/2),(y[-4]+h*k1/2),(t[-4]+h/2));
q2 := y[-4]+h*k1/2;
k3 := f((x[-4]+h*q2/2),(y[-4]+h*k2/2),(t[-4]+h/2));
q3 := y[-4]+h*k2/2;
k4 := f((x[-4]+h*q3),(y[-4]+h*k3),(t[-4]+h));
q4 := y[-4]+h*k3;
{Calculo dos valores no instante seguinte }
x[-3] := x[-4]+ h * (q1 + 2*q2 +2*q3 + q4)/6;
y[-3] := y[-4]+ h * (k1 + 2*k2 +2*k3 + k4)/6;
t[-3] := t[-4] + h;
```

```
efe[-3] := f(x[-3],y[-3],t[-3]);
ge[-3] := y[-3];

{Instante -1 Calculo dos coeficientes k e q }
k1 := f(x[-2],y[-2],t[-2]);
q1 := y[-2];
k2 := f((x[-2]+h*q1/2),(y[-2]+h*k1/2),(t[-2]+h/2));
q2 := y[-2]+h*k1/2;
k3 := f((x[-2]+h*q2/2),(y[-2]+h*k2/2),(t[-2]+h/2));
q3 := y[-2]+h*k2/2;
k4 := f((x[-2]+h*q3),(y[-2]+h*k3),(t[-2]+h));
q4 := y[-2]+h*k3;
{Calculo dos valores no instante seguinte }
x[-1] := x[-2]+ h * (q1 + 2*q2 +2*q3 + q4)/6;
y[-1] := y[-2]+ h * (k1 + 2*k2 +2*k3 + k4)/6;
t[-1] := t[-2] + h;
efe[-1] := f(x[-1],y[-1],t[-1]);
ge[-1] := y[-1];

{Fim do calculo por RK4 }

{ Transfere os valores de -6 e -5 para o arquivo e zera estes no buffer }
for i:=-6 to -5 do
  begin
  if t[i]<>0 then
```

```
        begin
        write(arquivo,t[i]);{tempo}
        write(arquivo,x[i]);{posição}
        write(arquivo,y[i]);{vel}
        write(arquivo,efe[i]);{acel}
        write(arquivo,ge[i]);{g}
        write(arquivo,etx[i]);{erro truncamento pos}
        writeln(arquivo,ety[i]);{erro truncamento vel}
        ProgressBar1.position:=round(1000*(t[i]/tfinal));
        end;
    t[i]:=0;x[i]:=0;y[i]:=0;efe[i]:=0;ge[i]:=0;etx[i]:=0;ety[i]:=0;
    end;
    {zera os valores de inst +1}
    t[1]:=0;x[1]:=0;y[1]:=0;efe[1]:=0;ge[1]:=0;etx[1]:=0;ety[1]:=0;
  end
else
  begin
  {Calcula valores pois está dentro da faixa de erro exigida }
  y[1]:=yc-ety[1];
  x[1]:=xc-etx[1];
  efe[1]:=f(x[1],y[1],t[1]);
  ge[1]:=y[1];
  {Envia valores do buffer para o arquivo em disco }
  if t[-6]<>0 then
    begin
```

```
      write(arquivo,t[-6]);{tempo}
      write(arquivo,x[-6]);{posição}
      write(arquivo,y[-6]);{vel}
      write(arquivo,efe[-6]);{acel}
      write(arquivo,ge[-6]);{g}
      write(arquivo,etx[-6]);{erro truncamento pos}
      writeln(arquivo,ety[-6]);{erro truncamento vel}
      ProgressBar1.position:=round(1000*(t[-6]/tfinal));
      end;

    {move o buffer uma posição acima }
    for i:=6 downto 0 do
      begin
      t[-i]:=t[1-i];
      x[-i]:=x[1-i];
      y[-i]:=y[1-i];
      efe[-i]:=efe[1-i];
      ge[-i]:=ge[1-i];
      etx[-i]:=etx[1-i];
      ety[-i]:=ety[1-i];
      end;

    {Verifica se o passo deve ser aumentado }
    if (etx[1]<(et/100)) and (ety[1]<(et/100)) and (erroy<(erro/10)) and
(errox<(erro/10))and (t[-6]<>0)and(not controle)
```

```
      then
        begin
        {Modifica step de h para 2h }
        for i:=-1 downto -3 do  {designa os novos valores de -1 a -3 }
          begin
          t[i]:=t[2*i];
          x[i]:=x[2*i];
          y[i]:=y[2*i];
          efe[i]:=efe[2*i];
          ge[i]:=ge[2*i];
          etx[i]:=etx[2*i];
          ety[i]:=ety[2*i];
          end;
        for i:=-4 downto -6 do {zera os valores de -4 a -6 }
          begin
          t[i]:=0;x[i]:=0;y[i]:=0;efe[i]:=0;ge[i]:=0;etx[i]:=0;ety[i]:=0;
          end;
        h:=2*h;
        end;
        {zera os valores de i+1}
        t[1]:=0;x[1]:=0;y[1]:=0;efe[1]:=0;ge[1]:=0;etx[1]:=0;ety[1]:=0;
    end;{else}
end;{while}
CloseFile(arquivo);
```

```
end;{procedure}


end. {program}
```

10. Listagem do Mechcalc 2.0 Beta 1

Segue abaixo a listagem dos units que compõem o Mechcalc 2.0

```
unit Unit1;


interface


uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ComCtrls, Math, unit2, ToolWin, Gauges, StdCtrls, GraphElem,
  ExtCtrls;
const
  NMASS=50;
  NFORCE=50;
  NDAMPER=50;
  NSPRING=50;


type
  Grafmass = array[1..NMASS] of TGraphElem;
  Tipodemassa = (none,mass,frestraint,vrestraint,hrestraint);
  Tmass = record
    n : integer;
    fxmola,fymola,fxamort,fyamort,fxt,fyt : extended;
    x,y,vx,vy,ax,ay,vxc,vyc,xc,yc,t,xdif,ydif,vxdif,vydif : extended;
    errox,erroy,errovx,errovy,etx,ety,etvx,etvy,massa : extended;
```

```
    tipo : Tipodemassa;
    end;
  Tdamper = record
    n,el1,el2 : integer;
    l,c : extended;
    end;
  Tspring = record
    n,el1,el2 : integer;
    l,k1,k2,k3 : extended;
    end;
  Tipodeforca = (senoidal,cossenoidal,polinomial);
  Tforce = record
    n,el1 : integer;
    t0,tf,teta : extended;

    case tipo : tipodeforca of
    senoidal,cossenoidal :
      (A,w : extended;);
    polinomial:
      (p : array [0..30] of extended;);
    end;

amass = array[-4..3] of Tmass;
```

```
TForm1 = class(TForm)
  MainMenu1: TMainMenu;
  File1: TMenuItem;
  New1: TMenuItem;
  Open1: TMenuItem;
  Save1: TMenuItem;
  Calculate1: TMenuItem;
  StatusBar1: TStatusBar;
  SaveAs1: TMenuItem;
  Options1: TMenuItem;
  Edit1: TMenuItem;
  Help1: TMenuItem;
  OpenDialog1: TOpenDialog;
  PrintDialog1: TPrintDialog;
  PrinterSetupDialog1: TPrinterSetupDialog;
  SaveDialog1: TSaveDialog;
  ToolBar1: TToolBar;
  ToolButton1: TToolButton;
  ImageList1: TImageList;
  ToolButton2: TToolButton;
  ToolButton3: TToolButton;
  ToolButton4: TToolButton;
  ToolButton7: TToolButton;
  ToolButton8: TToolButton;
  ToolButton9: TToolButton;
```

```
ToolButton14: TToolButton;
Gauge1: TGauge;
PrinterSetup1: TMenuItem;
Print1: TMenuItem;
OutputFileAs1: TMenuItem;
SaveDialog2: TSaveDialog;
ComboBox1: TComboBox;
ToolButton15: TToolButton;
StaticText1: TStaticText;
ComboBox2: TComboBox;
procedure Calculate1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure Options1Click(Sender: TObject);
procedure OutputFileAs1Click(Sender: TObject);
procedure ToolButton1Click(Sender: TObject);
procedure Massclick(Sender : TObject);
procedure Springclick(Sender : TObject);
procedure ToolButton2Click(Sender: TObject);
procedure Graphrefresh;
procedure ToolButton3Click(Sender: TObject);
procedure Amortclick(Sender : TObject);
procedure ToolButton9Click(Sender: TObject);
procedure Frestclick(Sender : TObject);
procedure ToolButton4Click(Sender: TObject);
```

```
    procedure Forcaclick(Sender : TObject);
    procedure SaveAs1Click(Sender: TObject);
    procedure New1Click(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);

    procedure Help1Click(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }



  end;

var
  massa : array[1..NMASS] of amass;
  amort : array[1..NDAMPER] of Tdamper;
  mola : array[1..NSPRING] of Tspring;
  forca : array[1..NFORCE] of Tforce;

 gmassa : array[1..NMASS] of TGraphElem;
 gamort : array[1..NDAMPER] of TGraphElem;
```

```
  gmola : array[1..NSPRING] of TGraphElem;
  gforca : array[1..NFORCE] of TGraphElem;

  nummassa,numamort,nummola,numforca : integer;
  passo,tfinal : extended;
  arquivo,arqsyst,arqini : textfile;
  salvaerro,salvaforca,anima : boolean;
  numbruno : integer;{variavel de passagem menu}
  viewcoef : extended;
  Form1: TForm1;
  largforca,comprforca,pontosaver,videorate : integer;
  etposmax,etspdmax,eposmax,espdmax : extended;
  tmpstrg : string;

implementation

uses Unit3, Unit5, Unit4, Unit6, Unit7, Unit11, Unit9;

{$R *.DFM}

{procedures auxiliares de copia e "zeragem de instantes"}
procedure zera(inst:integer);
var plow : integer;
begin
for plow:=1 to nummassa do
```

```
begin

massa[plow][inst].fxmola:=0;
massa[plow][inst].fymola:=0;
massa[plow][inst].fxamort:=0;
massa[plow][inst].fyamort:=0;
massa[plow][inst].fxt:=0;
massa[plow][inst].fyt:=0;
massa[plow][inst].x:=0;
massa[plow][inst].y:=0;
massa[plow][inst].vx:=0;
massa[plow][inst].vy:=0;
massa[plow][inst].ax:=0;
massa[plow][inst].ay:=0;
massa[plow][inst].errox:=0;
massa[plow][inst].erroy:=0;
massa[plow][inst].errovx:=0;
massa[plow][inst].errovy:=0;
massa[plow][inst].etx:=0;
massa[plow][inst].ety:=0;
massa[plow][inst].etvx:=0;
massa[plow][inst].etvy:=0;
massa[plow][inst].vxc:=0;
massa[plow][inst].vyc:=0;
massa[plow][inst].xc:=0;
```

```
  massa[plow][inst].yc:=0;
  massa[plow][inst].t:=-1;
  massa[plow][inst].xdif:=0;
  massa[plow][inst].ydif:=0;
  massa[plow][inst].vxdif:=0;
  massa[plow][inst].vydif:=0;

  end;
end;

procedure inicializa(inst:integer);
var plow : integer;
begin
zera(inst);
for plow:=1 to nummassa do
  begin
  massa[plow][inst].tipo:=none;
  massa[plow][inst].n:=0;
  massa[plow][inst].massa:=0;
  end;
end;

procedure move(inst1,inst2 : integer);
var plow : integer;
begin
```

```
for plow:=1 to nummassa do
 begin
 massa[plow][inst2].massa:=massa[plow][inst1].massa;
 massa[plow][inst2].fxmola:=massa[plow][inst1].fxmola;
 massa[plow][inst2].fymola:=massa[plow][inst1].fymola;
 massa[plow][inst2].fxamort:=massa[plow][inst1].fxamort;
 massa[plow][inst2].fyamort:=massa[plow][inst1].fyamort;
 massa[plow][inst2].fxt:=massa[plow][inst1].fxt;
 massa[plow][inst2].fyt:=massa[plow][inst1].fyt;
 massa[plow][inst2].x:=massa[plow][inst1].x;
 massa[plow][inst2].y:=massa[plow][inst1].y;
 massa[plow][inst2].vx:=massa[plow][inst1].vx;
 massa[plow][inst2].vy:=massa[plow][inst1].vy;
 massa[plow][inst2].ax:=massa[plow][inst1].ax;
 massa[plow][inst2].ay:=massa[plow][inst1].ay;
 massa[plow][inst2].errox:=massa[plow][inst1].errox;
 massa[plow][inst2].erroy:=massa[plow][inst1].erroy;
 massa[plow][inst2].errovx:=massa[plow][inst1].errovx;
 massa[plow][inst2].errovy:=massa[plow][inst1].errovy;
 massa[plow][inst2].etx:=massa[plow][inst1].etx;
 massa[plow][inst2].ety:=massa[plow][inst1].ety;
 massa[plow][inst2].etvx:=massa[plow][inst1].etvx;
 massa[plow][inst2].etvy:=massa[plow][inst1].etvy;
 massa[plow][inst2].vxc:=massa[plow][inst1].vxc;
 massa[plow][inst2].vyc:=massa[plow][inst1].vyc;
```

```
  massa[plow][inst2].xc:=massa[plow][inst1].xc;
  massa[plow][inst2].yc:=massa[plow][inst1].yc;
  massa[plow][inst2].t:=massa[plow][inst1].t;
  massa[plow][inst2].xdif:=massa[plow][inst1].xdif;
  massa[plow][inst2].ydif:=massa[plow][inst1].ydif;
  massa[plow][inst2].vxdif:=massa[plow][inst1].vxdif;
  massa[plow][inst2].vydif:=massa[plow][inst1].vydif;
  massa[plow][inst2].tipo:=massa[plow][inst1].tipo;
  massa[plow][inst2].n:=massa[plow][inst1].n;
  end;
end;



{Funções de cálculo de forças
  de molas,amortecedores e aplicadas }

function forcat(tempo : extended; elm : integer) : extended;
var tempor : extended;plow : integer;
begin
if (tempo>=forca[elm].t0) and (tempo<=forca[elm].tf) then
 begin
 if forca[elm].tipo=senoidal then
```

```
    forcat:=forca[elm].A*(sin(forca[elm].w*(tempo-forca[elm].t0)))
    else if forca[elm].tipo=cossenoidal then
        forcat:=forca[elm].A*(cos(forca[elm].w*(tempo-forca[elm].t0)))
        else if forca[elm].tipo=polinomial then
                begin
                tempor:=forca[elm].p[0];
                for plow:=1 to 30 do
                  begin
                  tempor:=tempor+(forca[elm].p[plow]*Power(tempo,plow));
                  end;
               forcat:=tempor;
               end;
  end
  else forcat:=0;
end;{function forcat}

function forcaamort(elm : integer; vel : extended) : extended;
begin
forcaamort := -(vel * amort[elm].c);
end;

function angulocalc(x1,y1,x2,y2 : extended):extended;
var dx,dy : extended;
begin
dx:=x1-x2;
```

```
 dy:=y1-y2;

 if dx=0 then
  begin
  if dy>0 then angulocalc:=Pi/2
  else angulocalc:=3*Pi/2;
  end
 else
  if dy=0 then
    begin
    if dx>0 then angulocalc:=0
    else angulocalc:=Pi;
    end
  else
    if (dx>0) and (dy>0) then angulocalc:=arctan(dy/dx)
    else
      if (dx<0) and (dy>0) then angulocalc:=Pi+arctan(dy/dx)
      else
        if (dx<0) and (dy<0) then angulocalc:=Pi+arctan(dy/dx)
        else
          angulocalc:=2*Pi+arctan(dy/dx);
end;


function forcamola(elm : integer; compr : extended) : extended;
```

```
begin
forcamola := (mola[elm].l - compr)*mola[elm].k1;
end;

function conectado (nconec : integer) : integer;
var cci,ctempor : integer;
begin
ctempor:=0;
for cci:=1 to nummola do
   if (mola[cci].el1=nconec) or (mola[cci].el2=nconec) then ctempor:=ctempor+1;
for cci:=1 to numamort do
   if (amort[cci].el1=nconec) or (amort[cci].el2=nconec) then
ctempor:=ctempor+1;
for cci:=1 to numforca do
   if forca[cci].el1=nconec then ctempor:=ctempor+1;
conectado:=ctempor;

end;

{funcao de cáluclo de acelerações (2da derivada)}
procedure acelcalc(instbase,instwrite : integer);
var wa : integer;
   x1,x2,y1,y2,deltax,deltay,latual,teta1,teta2,F,vx1,vx2,vy1,vy2 : extended;
   Fx,Fy,deltavx,deltavy,deltav,vxp,vyp : extended;
begin
```

```
{inicializa variáveis que serão calculadas}
for wa:=1 to nummassa do
 begin
 massa[wa][instwrite].fxmola:=0;
 massa[wa][instwrite].fymola:=0;
 massa[wa][instwrite].fxamort:=0;
 massa[wa][instwrite].fyamort:=0;
 massa[wa][instwrite].fxt:=0;
 massa[wa][instwrite].fyt:=0;
 end;
{ inicializa o instante de escrita }
zera(instwrite);

{Calcula e escreve as forças de mola envolvidas }
for wa:=1 to nummola do
 begin
 {variáveis auxiliares}
 x1:=massa[mola[wa].el1][instbase].x;
 y1:=massa[mola[wa].el1][instbase].y;
 x2:=massa[mola[wa].el2][instbase].x;
 y2:=massa[mola[wa].el2][instbase].y;
 deltax:=x1-x2;
 deltay:=y1-y2;
 latual:=sqrt(power(deltax,2)+power(deltay,2));
 {rever o cálculo de teta1!!!!!!!!!!}
```

```
 teta1:=angulocalc(x1,y1,x2,y2);
 teta2:=teta1+Pi;
 F:=forcamola(wa,latual);
 {calcula e adiciona nos elementos}
 if deltax<>0 then massa[mola[wa].el1][instwrite].fxmola:=
                  massa[mola[wa].el1][instwrite].fxmola+F*cos(teta1);

 if deltay<>0 then massa[mola[wa].el1][instwrite].fymola:=
                  massa[mola[wa].el1][instwrite].fymola+F*sin(teta1);

 if deltax<>0 then massa[mola[wa].el2][instwrite].fxmola:=
                  massa[mola[wa].el2][instwrite].fxmola+F*cos(teta2);

 if deltay<>0 then massa[mola[wa].el2][instwrite].fymola:=
                  massa[mola[wa].el2][instwrite].fymola+F*sin(teta2);
 end;
{Calcula e escreve as forças de amortecimento envolvidas}
for wa:=1 to numamort do
 begin
 {designando variáveis }
 x1:=massa[amort[wa].el1][instbase].x;
 y1:=massa[amort[wa].el1][instbase].y;
 x2:=massa[amort[wa].el2][instbase].x;
 y2:=massa[amort[wa].el2][instbase].y;
 vx1:=massa[amort[wa].el1][instbase].vx;
```

```
vy1:=massa[amort[wa].el1][instbase].vy;
vx2:=massa[amort[wa].el2][instbase].vx;
vy2:=massa[amort[wa].el2][instbase].vy;

{calculando}
deltax:=x1-x2;
deltay:=y1-y2;

teta1:=angulocalc(x1,y1,x2,y2);
teta2:=teta1+Pi;
{rever o calculo do amort}
deltavx:=vx1-vx2;
deltavy:=vy1-vy2;
vxp:=deltavx*cos(teta1);
vyp:=deltavy*cos((Pi/2)-teta1);
deltav:=vxp+vyp;
F:=forcaamort(wa,deltav);

{calcula e adiciona nos elementos}
if deltax<>0 then massa[amort[wa].el1][instwrite].fxamort:=
                massa[amort[wa].el1][instwrite].fxamort+F*cos(teta1);

if deltay<>0 then massa[amort[wa].el1][instwrite].fyamort:=
                massa[amort[wa].el1][instwrite].fyamort+F*sin(teta1);
```

```
 if deltax<>0 then massa[amort[wa].el2][instwrite].fxamort:=
                    massa[amort[wa].el2][instwrite].fxamort+F*cos(teta2);

 if deltay<>0 then massa[amort[wa].el2][instwrite].fyamort:=
                    massa[amort[wa].el2][instwrite].fyamort+F*sin(teta2);
 end;
{calcula e escreve as forças aplicadas envolvidas }
for wa:=1 to numforca do
 begin
 F:=forcat(massa[1][instbase].t,wa);
 massa[forca[wa].el1][instwrite].fxt:= massa[forca[wa].el1][instwrite].fxt+
                              F*cos(forca[wa].teta);
 massa[forca[wa].el1][instwrite].fyt:= massa[forca[wa].el1][instwrite].fyt+
                              F*sin(forca[wa].teta);
 end;

{calcula a aceleração para as massas}
for wa:=1 to nummassa do
 begin
 Fx:=massa[wa][instwrite].fxmola+massa[wa][instwrite].fxamort+
                              massa[wa][instwrite].fxt;
 Fy:=massa[wa][instwrite].fymola+massa[wa][instwrite].fyamort+
                              massa[wa][instwrite].fyt;
 if massa[wa][instwrite].tipo=mass then
   begin
```

```
      massa[wa][instwrite].ax:=Fx/massa[wa][instwrite].massa;
      massa[wa][instwrite].ay:=Fy/massa[wa][instwrite].massa;
      massa[wa][instwrite].vx:=massa[wa][instbase].vx;
      massa[wa][instwrite].vy:=massa[wa][instbase].vy;
      end
    else{caso o elemento seja na verdade um restraint}
      begin
      massa[wa][instwrite].ax:=0;
      massa[wa][instwrite].ay:=0;
      massa[wa][instwrite].vx:=0;
      massa[wa][instwrite].vy:=0;
      end;
    end;
  end;{fim da procedure acelcalc}

{fim das funcoes auxiliares}


procedure TForm1.Calculate1Click(Sender: TObject);
var i,j,progresso,bcounter,videocounter : integer;
kk1,kk2,kk3,kk4,kq1,kq2,kq3,kq4,kw1,kw2,kw3,kw4,kz1,kz2,kz3,kz4 :
                                        array[1..NMASS] of extended;
begin
{verifica se foi designado arquivo de saída e se necessário designa}
```

```
{Escreve cabecalho no arquivo de saida}
rewrite(arquivo);bcounter:=0; videocounter:=0;
write(arquivo,massa[1][-4].t);{escreve o tempo na primeira coluna}
{    write(arquivo,' ');
    for j:=1 to nummassa do
      begin
      write(arquivo,massa[j][-4].x:24);write(arquivo,' ');
      write(arquivo,massa[j][-4].vx);write(arquivo,' ');
      write(arquivo,massa[j][-4].ax);write(arquivo,' ');
      write(arquivo,massa[j][-4].y);write(arquivo,' ');
      write(arquivo,massa[j][-4].vy);write(arquivo,' ');
      write(arquivo,massa[j][-4].ay);write(arquivo,' ');
      if salvaerro=true then
        begin
        write(arquivo,massa[j][-4].errox);write(arquivo,' ');
        write(arquivo,massa[j][-4].errovx);write(arquivo,' ');
        write(arquivo,massa[j][-4].erroy);write(arquivo,' ');
        write(arquivo,massa[j][-4].errovy);write(arquivo,' ');
        write(arquivo,massa[j][-4].etx);write(arquivo,' ');
        write(arquivo,massa[j][-4].etvx);write(arquivo,' ');
        write(arquivo,massa[j][-4].ety);write(arquivo,' ');
        write(arquivo,massa[j][-4].etvy);write(arquivo,' ');
        end;
      if salvaforca=true then
        begin
```

```
        write(arquivo,massa[j][-4].fxt);write(arquivo,' ');
        write(arquivo,massa[j][-4].fxmola);write(arquivo,' ');
        write(arquivo,massa[j][-4].fxamort);write(arquivo,' ');
        write(arquivo,massa[j][-4].fyt);write(arquivo,' ');
        write(arquivo,massa[j][-4].fymola);write(arquivo,' ');
        write(arquivo,massa[j][-4].fyamort);write(arquivo,' ');
        end;
      write(arquivo,'* ');
      end;{loop massas}
    writeln(arquivo);

{Fim da escrita do cabeçalho}

massa[1][-4].t:=0;
{inicializa todos os valores do buffer}
for i:=-3 to 3 do zera(i);
{inicializa os valores de monitoramento do erro de calculo}
etposmax:=0;eposmax:=0;
etspdmax:=0;espdmax:=0;

{ Runge-Kutta Calculation - i=-3,-2,-1,0 }
{ inicializa os coeficientes }
for i:=1 to nummassa do
 begin
 kk1[i]:=0;
```

```
kq1[i]:=0;
kw1[i]:=0;
kz1[i]:=0;
kk2[i]:=0;
kq2[i]:=0;
kw2[i]:=0;
kz2[i]:=0;
kk3[i]:=0;
kq3[i]:=0;
kw3[i]:=0;
kz3[i]:=0;
kk4[i]:=0;
kq4[i]:=0;
kw4[i]:=0;
kz4[i]:=0;
end;
```

```
{passa valores pertinentes da massa para todos os instantes do buffer }
For i:=-3 to 3 do
 for j:=1 to nummassa do
   begin
   massa[j][i].tipo:=massa[j][i-1].tipo;
   massa[j][i].massa:=massa[j][i-1].massa;
   massa[j][i].n:=massa[j][i-1].n;
```

```
    end;


For i:=-3 to 0 do { loop instantes }
 begin
 { Calculo do primeiro coeficiente }
 for j:=1 to nummassa do
   begin
   kk1[j]:=massa[j][i-1].ax;
   kq1[j]:=massa[j][i-1].vx;
   kw1[j]:=massa[j][i-1].ay;
   kz1[j]:=massa[j][i-1].vy;
   end;
{Calculo do coef 2 }
{zera o instante 2 e 3 (buffer)}
zera(2);
zera(3);
{coloca valores no instante 2 }
for j:=1 to nummassa do
  begin
  massa[j][2].x:=massa[j][i-1].x + ((passo/2)*kq1[j]);
  massa[j][2].vx:=massa[j][i-1].vx + ((passo/2)*kk1[j]);
  massa[j][2].t:=massa[j][i-1].t + passo/2;
  massa[j][2].y:=massa[j][i-1].y + ((passo/2)*kz1[j]);
  massa[j][2].vy:=massa[j][i-1].vy + ((passo/2)*kw1[j]);
```

```
    end;
 {calcula valores de aceleração para as cond 2}
 acelcalc(2,3);
 {copia valores p/a coefs 2}
 for j:=1 to nummassa do
    begin
    kk2[j]:=massa[j][3].ax;
    kq2[j]:=massa[j][3].vx;
    kw2[j]:=massa[j][3].ay;
    kz2[j]:=massa[j][3].vy;
    end;

{calculo coef 3}

{zera o instante 2 e 3 (buffer)}
zera(2);
zera(3);
{coloca valores no instante 2 }
for j:=1 to nummassa do
   begin
   massa[j][2].x:=massa[j][i-1].x + ((passo/2)*kq2[j]);
   massa[j][2].vx:=massa[j][i-1].vx + ((passo/2)*kk2[j]);
   massa[j][2].t:=massa[j][i-1].t + passo/2;
   massa[j][2].y:=massa[j][i-1].y + ((passo/2)*kz2[j]);
   massa[j][2].vy:=massa[j][i-1].vy + ((passo/2)*kw2[j]);
```

```
    end;
 {calcula valores de aceleração para as cond 2}
 acelcalc(2,3);
 {copia valores p/a coefs 3}
 for j:=1 to nummassa do
    begin
    kk3[j]:=massa[j][3].ax;
    kq3[j]:=massa[j][3].vx;
    kw3[j]:=massa[j][3].ay;
    kz3[j]:=massa[j][3].vy;
    end;
{calculo coef 4}

{zera o instante 2 e 3 (buffer)}
zera(2);
zera(3);
{coloca valores no instante 2 }
for j:=1 to nummassa do
   begin
   massa[j][2].x:=massa[j][i-1].x + (passo*kq3[j]);
   massa[j][2].vx:=massa[j][i-1].vx + (passo*kk3[j]);
   massa[j][2].t:=massa[j][i-1].t + passo;
   massa[j][2].y:=massa[j][i-1].y + (passo*kz3[j]);
   massa[j][2].vy:=massa[j][i-1].vy + (passo*kw3[j]);
   end;
```

```
{calcula valores de aceleração para as cond 2}
acelcalc(2,3);
{copia valores p/a coefs 4}
for j:=1 to nummassa do
  begin
  kk4[j]:=massa[j][3].ax;
  kq4[j]:=massa[j][3].vx;
  kw4[j]:=massa[j][3].ay;
  kz4[j]:=massa[j][3].vy;
  end;
{fim do calculo dos coefs }

{Calcula o valor final no instante seguinte}
for j:=1 to nummassa do
  begin
  massa[j][i].x:=massa[j][i-1].x + (passo/6)*(kq1[j]+(2*kq2[j])+(2*kq3[j])+kq4[j]);
  massa[j][i].vx:=massa[j][i-1].vx +
(passo/6)*(kk1[j]+(2*kk2[j])+(2*kk3[j])+kk4[j]);
  massa[j][i].y:=massa[j][i-1].y + (passo/6)*(kz1[j]+(2*kz2[j])+(2*kz3[j])+kz4[j]);
  massa[j][i].vy:=massa[j][i-1].vy +
(passo/6)*(kw1[j]+(2*kw2[j])+(2*kw3[j])+kw4[j]);
  massa[j][i].t:=massa[j][i-1].t + passo;
  end;
{Calcula as acelerações no instante seguinte}
acelcalc(i,3);
```

```
  for j:=1 to nummassa do
    begin
    massa[j][i].ax:=massa[j][3].ax;
    massa[j][i].ay:=massa[j][3].ay;
    massa[j][i].fxt:=massa[j][3].fxt;
    massa[j][i].fxmola:=massa[j][3].fxmola;
    massa[j][i].fxamort:=massa[j][3].fxamort;
    massa[j][i].fyt:=massa[j][3].fyt;
    massa[j][i].fymola:=massa[j][3].fymola;
    massa[j][i].fyamort:=massa[j][3].fyamort;
    end;
  zera(3);

  end;{do loop dos instantes i=-3 a i=0 (fim do rk4)}

{ Calculo geral(Hamming) }
zera(1);zera(2);zera(3);

while(massa[1][-4].t<tfinal) do
  begin
  {passa valores pertinentes da massa para o isntante seguinte }
  for j:=1 to nummassa do
    begin
    massa[j][1].tipo:=massa[j][0].tipo;
    massa[j][1].massa:=massa[j][0].massa;
```

```
    massa[j][1].n:=massa[j][0].n;
    end;

 {Fase I do calculo }
 for j:=1 to nummassa do
   begin
   massa[j][1].vx:=massa[j][-3].vx + (4/3)*passo*((2*massa[j][0].ax)
                          -massa[j][-1].ax+(2*massa[j][-2].ax));
   massa[j][1].x:=massa[j][-3].x + (4/3)*passo*((2*massa[j][0].vx)
                          -massa[j][-1].vx+(2*massa[j][-2].vx));
   massa[j][1].vy:=massa[j][-3].vy + (4/3)*passo*((2*massa[j][0].ay)
                          -massa[j][-1].ay+(2*massa[j][-2].ay));
   massa[j][1].y:=massa[j][-3].y + (4/3)*passo*((2*massa[j][0].vy)
                          -massa[j][-1].vy+(2*massa[j][-2].vy));
   end;
{Fase II do Calculo }
for j:=1 to nummassa do
   begin
   massa[j][1].vx:=massa[j][1].vx+(112/121)*massa[j][0].vxdif;
   massa[j][1].x:=massa[j][1].x+(112/121)*massa[j][0].xdif;
   massa[j][1].vy:=massa[j][1].vy+(112/121)*massa[j][0].vydif;
   massa[j][1].y:=massa[j][1].y+(112/121)*massa[j][0].ydif;
   end;

{Fase III do calculo }
```

```
{calcula valores e coloca no inst 3}
acelcalc(1,3);
{calcula os valores corrigidos}
for j:=1 to nummassa do
  begin
  massa[j][1].vxc:=(1/8)*(9*massa[j][0].vx -massa[j][-2].vx+3*passo
                *(massa[j][3].ax+(2*massa[j][0].ax)-massa[j][-1].ax));
  massa[j][1].xc:=(1/8)*(9*massa[j][0].x -massa[j][-2].x+3*passo
                *(massa[j][3].vx+(2*massa[j][0].vx)-massa[j][-1].vx));
  massa[j][1].vyc:=(1/8)*(9*massa[j][0].vy -massa[j][-2].vy+3*passo
                *(massa[j][3].ay+(2*massa[j][0].ay)-massa[j][-1].ay));
  massa[j][1].yc:=(1/8)*(9*massa[j][0].y -massa[j][-2].y+3*passo
                *(massa[j][3].vy+(2*massa[j][0].vy)-massa[j][-1].vy));
  end;
{Fase IV, V e VI do Calculo }
for j:=1 to nummassa do
  begin
  {calcula as diferenças de correção}
  massa[j][1].xdif:=massa[j][1].xc-massa[j][1].x;
  massa[j][1].ydif:=massa[j][1].yc-massa[j][1].y;
  massa[j][1].vxdif:=massa[j][1].vxc-massa[j][1].vx;
  massa[j][1].vydif:=massa[j][1].vyc-massa[j][1].vy;

  {calcula os erros (IV)}    {VAI DAR PAU NA DIVISÂO }
  if massa[j][1].xc=0 then massa[j][1].errox := 0
```

```
    else massa[j][1].errox:=abs(massa[j][1].xdif/massa[j][1].xc);
  if massa[j][1].yc=0 then massa[j][1].erroy:=0
    else massa[j][1].erroy:=abs(massa[j][1].ydif/massa[j][1].yc);
  if massa[j][1].vxc=0 then massa[j][1].errovx:=0
    else massa[j][1].errovx:=abs(massa[j][1].vxdif/massa[j][1].vxc);
  if massa[j][1].vyc=0 then massa[j][1].errovy:=0
    else massa[j][1].errovy:=abs(massa[j][1].vydif/massa[j][1].vyc);

  {calcula os erros de truncamento(V)}
  massa[j][1].etx:=(9/121)*massa[j][1].xdif;
  massa[j][1].ety:=(9/121)*massa[j][1].ydif;
  massa[j][1].etvx:=(9/121)*massa[j][1].vxdif;
  massa[j][1].etvy:=(9/121)*massa[j][1].vydif;

  {calcula os valores finais (VI)}
  massa[j][1].x:=massa[j][1].xc-massa[j][1].etx;
  massa[j][1].y:=massa[j][1].yc-massa[j][1].ety;
  massa[j][1].vx:=massa[j][1].vxc-massa[j][1].etvx;
  massa[j][1].vy:=massa[j][1].vyc-massa[j][1].etvy;
  massa[j][1].t:=massa[j][0].t +passo;

  {atualiza o erro máximo encontrado}
  if massa[j][1].etx>etposmax then etposmax:=massa[j][1].etx;
  if massa[j][1].ety>etposmax then etposmax:=massa[j][1].ety;
  if massa[j][1].etvx>etspdmax then etspdmax:=massa[j][1].etvx;
```

```
    if massa[j][1].etvy>etspdmax then etspdmax:=massa[j][1].etvy;

    if massa[j][1].errox>eposmax then eposmax:=massa[j][1].errox;
    if massa[j][1].erroy>eposmax then eposmax:=massa[j][1].erroy;
    if massa[j][1].errovx>espdmax then espdmax:=massa[j][1].errovx;
    if massa[j][1].errovy>espdmax then espdmax:=massa[j][1].errovy;
    end;


{Fase VII do calculo }
acelcalc(1,3);
for j:=1 to nummassa do
   begin
   massa[j][1].ax:=massa[j][3].ax;
   massa[j][1].ay:=massa[j][3].ay;
   massa[j][1].fxt:=massa[j][3].fxt;
   massa[j][1].fxmola:=massa[j][3].fxmola;
   massa[j][1].fxamort:=massa[j][3].fxamort;
   massa[j][1].fyt:=massa[j][3].fyt;
   massa[j][1].fymola:=massa[j][3].fymola;
   massa[j][1].fyamort:=massa[j][3].fyamort;
   end;

{Fim do calculo do instante seguinte(1)}
```

```
{Grava os valores do buffer se necessário e movimenta uma pos.}

if (massa[1][-4].t<>-1) and (bcounter=0) then
  begin
  write(arquivo,massa[1][-4].t);{escreve o tempo na primeira coluna}
  write(arquivo,' ');
  for j:=1 to nummassa do
    begin
    write(arquivo,massa[j][-4].x);write(arquivo,' ');
    write(arquivo,massa[j][-4].vx);write(arquivo,' ');
    write(arquivo,massa[j][-4].ax);write(arquivo,' ');
    write(arquivo,massa[j][-4].y);write(arquivo,' ');
    write(arquivo,massa[j][-4].vy);write(arquivo,' ');
    write(arquivo,massa[j][-4].ay);write(arquivo,' ');
    if salvaerro=true then
      begin
      write(arquivo,massa[j][-4].errox);write(arquivo,' ');
      write(arquivo,massa[j][-4].errovx);write(arquivo,' ');
      write(arquivo,massa[j][-4].erroy);write(arquivo,' ');
      write(arquivo,massa[j][-4].errovy);write(arquivo,' ');
      write(arquivo,massa[j][-4].etx);write(arquivo,' ');
      write(arquivo,massa[j][-4].etvx);write(arquivo,' ');
      write(arquivo,massa[j][-4].ety);write(arquivo,' ');
      write(arquivo,massa[j][-4].etvy);write(arquivo,' ');
      end;
```

```
        if salvaforca=true then
          begin
          write(arquivo,massa[j][-4].fxt);write(arquivo,' ');
          write(arquivo,massa[j][-4].fxmola);write(arquivo,' ');
          write(arquivo,massa[j][-4].fxamort);write(arquivo,' ');
          write(arquivo,massa[j][-4].fyt);write(arquivo,' ');
          write(arquivo,massa[j][-4].fymola);write(arquivo,' ');
          write(arquivo,massa[j][-4].fyamort);write(arquivo,' ');
          end;
        write(arquivo,'* ');
        end;{loop massas}
      writeln(arquivo);
      end;{termina a escrita no arquivo}
    { Animação do movimento }
    if (anima) and (videocounter=0) then graphrefresh;

    bcounter:=bcounter+1; if bcounter=pontosaver then bcounter:=0;
    videocounter:=videocounter+1; if videocounter = videorate then videocounter
  := 0;
    for j:=-4 to 0 do move(j+1,j);{movimenta o buffer para trás}
    progresso:=round(100*massa[1][0].t/tfinal);
    gauge1.Progress:=progresso;
    end;
  {finalizações do cálculo - escrita no arquivo }
  closefile(arquivo);
```

```
 gauge1.progress:=0;
 form11.show;



end;

procedure TForm1.FormCreate(Sender: TObject);
var i,inst,plow : integer;
begin

{zera todas as variáveis mecanicas }
{zera tudo }
for inst:=-4 to 3 do
for plow:=1 to NMASS do
 begin
 massa[plow][inst].fxmola:=0;
 massa[plow][inst].fymola:=0;
 massa[plow][inst].fxamort:=0;
 massa[plow][inst].fyamort:=0;
 massa[plow][inst].fxt:=0;
 massa[plow][inst].fyt:=0;
 massa[plow][inst].x:=0;
 massa[plow][inst].y:=0;
 massa[plow][inst].vx:=0;
 massa[plow][inst].vy:=0;
```

```
    massa[plow][inst].ax:=0;
    massa[plow][inst].ay:=0;
    massa[plow][inst].errox:=0;
    massa[plow][inst].erroy:=0;
    massa[plow][inst].errovx:=0;
    massa[plow][inst].errovy:=0;
    massa[plow][inst].etx:=0;
    massa[plow][inst].ety:=0;
    massa[plow][inst].etvx:=0;
    massa[plow][inst].etvy:=0;
    massa[plow][inst].vxc:=0;
    massa[plow][inst].vyc:=0;
    massa[plow][inst].xc:=0;
    massa[plow][inst].yc:=0;
    massa[plow][inst].t:=-1;
    massa[plow][inst].xdif:=0;
    massa[plow][inst].ydif:=0;
    massa[plow][inst].vxdif:=0;
    massa[plow][inst].vydif:=0;
    massa[plow][inst].tipo:=none;
    massa[plow][inst].n:=0;
    massa[plow][inst].massa:=0;
   end;
for i:=1 to NSPRING do
   begin
```

```
      mola[i].k1:=0;
      mola[i].el1:=0;
      mola[i].el2:=0;
      mola[i].k2:=0;
      mola[i].k3:=0;
      mola[i].n:=0;
     end;
  for i:=1 to NDAMPER do
     begin

     with amort[i] do
         begin
         c:=0;
         el1:=0;
         el2:=0;
         l:=0;
        end;
    end;
 for i:=1 to NFORCE do
    begin
    forca[i].el1:=0;
    forca[i].teta:=0;
    forca[i].tipo:=senoidal;
    forca[i].a:=0;
    forca[i].w:=0;
```

```
      forca[i].t0:=0;
      forca[i].tf:=0;
      for inst:=0 to 29 do forca[i].p[inst]:=0;
      end;
  nummassa:=0;
  nummola:=0;
  numamort:=0;
  numforca:=0;
  graphrefresh;

  {variáveis que necessitam de inicialização}
  nummassa :=0;
  numamort :=0;
  nummola :=0;
  numforca :=0;
  salvaerro:=false;
  salvaforca:=false;
  viewcoef:=0.1;
  numbruno:=-1;
  largforca:=20;
  comprforca:=2*largforca;
  {lê dados do arquivo de inicialização }
  {reset(arqini);}

end;
```

```
procedure TForm1.Open1Click(Sender: TObject);
var i,j,tempor : integer;
    deltax,deltay,anggrau : extended;
begin
{preparação da leitura}


opendialog1.FileName:='*.mpr';


if opendialog1.execute then
 begin
 assignfile(arqsyst,opendialog1.filename);
 reset(arqsyst);
 {leitura dos dados}
 readln(arqsyst,nummassa);
 readln(arqsyst,nummola);
 readln(arqsyst,numamort);
 readln(arqsyst,numforca);
 { le as massas }
 for i:=1 to nummassa do
   begin
   tmpstrg:='Mass/Restr. ' + inttostr(i);
   combobox2.items.Add(tmpstrg);
```

```
    read(arqsyst,massa[i][-4].massa);
    read(arqsyst,massa[i][-4].x);
    read(arqsyst,massa[i][-4].y);
    read(arqsyst,massa[i][-4].vx);
    read(arqsyst,massa[i][-4].vy);
    read(arqsyst,massa[i][-4].ax);
    read(arqsyst,massa[i][-4].ay);
    readln(arqsyst,tempor);
    if tempor=1 then massa[i][-4].tipo:=mass
      else if tempor=2 then massa[i][-4].tipo:=frestraint
        else if tempor=3 then massa[i][-4].tipo:=vrestraint
          else if tempor=4 then massa[i][-4].tipo:=hrestraint
            else massa[i][-4].tipo:=none;

    end;

{lê as molas }
for i:=1 to nummola do
    begin
    tmpstrg:='Spring ' + inttostr(i);
    combobox2.items.Add(tmpstrg);
    read(arqsyst,mola[i].k1);
    read(arqsyst,mola[i].el1);
    readln(arqsyst,mola[i].el2);
    {calcula o L das molas}
```

```
    deltax:=massa[mola[i].el1][-4].x-massa[mola[i].el2][-4].x;
    deltay:=massa[mola[i].el1][-4].y-massa[mola[i].el2][-4].y;
    mola[i].l:=sqrt(power(deltax,2)+power(deltay,2));
    end;

{le os amort}
for i:=1 to numamort do
    begin
    tmpstrg:='Damper ' + inttostr(i);
    combobox2.items.Add(tmpstrg);
    read(arqsyst,amort[i].c);
    read(arqsyst,amort[i].el1);
    readln(arqsyst,amort[i].el2);
    end;

{lê as forcas aplicadas }
for i:=1 to numforca do
    begin
    tmpstrg:='Force ' + inttostr(i);
    combobox2.items.Add(tmpstrg);
    read(arqsyst,forca[i].t0);
    read(arqsyst,forca[i].tf);
    read(arqsyst,anggrau);
    forca[i].teta:=anggrau*pi/180;
    read(arqsyst,forca[i].el1);
```

```
    {le o tipo de forca}
    read(arqsyst,tempor);
    case tempor of
      1 : forca[i].tipo:=senoidal;
      2 : forca[i].tipo:=cossenoidal;
      3 : forca[i].tipo:=polinomial;
      end;
   if (forca[i].tipo=senoidal) or (forca[i].tipo=cossenoidal) then
      begin
      read(arqsyst,forca[i].a);
      readln(arqsyst,forca[i].w);
      end
      else if forca[i].tipo=polinomial then
        begin
        for j:=0 to 29 do read(arqsyst,forca[i].p[j]);
        readln(arqsyst);
        end;
   end;



closefile(arqsyst);
{gera os elementos gráficos }

{massas }
```

```
 for i:=1 to nummassa do
     begin
     gmassa[i]:=Tgraphelem.Create(self);
     gmassa[i].parent:=form1;
     gmassa[i].number:=i;
     gmassa[i].ondblclick:=massclick;
     gmassa[i].width:=24;
     gmassa[i].height:=20;

     gmassa[i].top:=round((viewcoef*massa[i][-4].y*1000)-
gmassa[i].height/2+50);
     gmassa[i].left:=round((viewcoef*massa[i][-4].x*1000)-gmassa[i].width/2+50);
     if massa[i][-4].tipo=mass then
       begin
       gmassa[i].tipo:=tmassa;
       gmassa[i].ondblclick:=massclick;
       end
     else
       if massa[i][-4].tipo=frestraint then
         begin
         gmassa[i].ondblclick:=frestclick;
         gmassa[i].tipo:=tfrest;
         end;

     end;
```

```
 for i:=1 to nummola do
     begin
     gmola[i]:=TgraphElem.Create(self);
     gmola[i].parent:=form1;
     gmola[i].number:=i;
     gmola[i].ondblclick:=springclick;
     gmola[i].tipo:=tmola;
     end;
 for i:=1 to numamort do
     begin
     gamort[i]:=Tgraphelem.Create(self);
     gamort[i].parent:=form1;
     gamort[i].number:=i;
     gamort[i].ondblclick:=amortclick;
     gamort[i].tipo:=tamort;
     end;
 for i:=1 to numforca do
     begin
     gforca[i]:=tgraphelem.create(self);
     gforca[i].parent:=form1;
     gforca[i].number:=i;
     gforca[i].ondblclick:=forcaclick;
     gforca[i].tipo:=tforca;
     end;
 graphrefresh;{desenha}
```

```
  end;
end;

procedure TForm1.Options1Click(Sender: TObject);
begin
form2.show;
end;

procedure TForm1.OutputFileAs1Click(Sender: TObject);
begin
if savedialog2.execute then
  assignfile(arquivo,savedialog2.filename);
end;

procedure TForm1.ToolButton1Click(Sender: TObject);
begin
nummassa:=nummassa+1;
gmassa[nummassa]:=Tgraphelem.Create(self);
gmassa[nummassa].parent:=form1;
gmassa[nummassa].number:=nummassa;
gmassa[nummassa].ondblclick:=massclick;
gmassa[nummassa].width:=24;
gmassa[nummassa].height:=20;
gmassa[nummassa].tipo:=tmassa;
numbruno:=nummassa;
```

```
{Atualiza o combobox }
tmpstrg:='Mass/Restr. ' + inttostr(nummassa);
combobox2.items.Add(tmpstrg);
form3.show;

end;
procedure TForm1.Springclick(Sender : TObject);
begin
numbruno:=(sender as Tgraphelem).number;
form5.show;
end;

procedure TForm1.Massclick(Sender : TObject);
begin
numbruno:=(sender as TgraphElem).number;
form3.show;

end;
procedure TForm1.ToolButton2Click(Sender: TObject);
begin
nummola:=nummola+1;
gmola[nummola]:=TgraphElem.Create(self);
gmola[nummola].parent:=form1;
gmola[nummola].number:=nummola;
gmola[nummola].ondblclick:=springclick;
```

```
gmola[nummola].tipo:=tmola;
numbruno:=nummola;
{Atualiza o combobox}
tmpstrg:='Spring ' + inttostr(nummola);
combobox2.items.Add(tmpstrg);
form5.show;



end;

procedure TForm1.Graphrefresh;
var  gi : integer;
    el1x,el2x,el1y,el2y,angt : extended;
begin
{ Fase 1 - Atualização das Massas}
for gi:=1 to nummassa do
 begin
 gmassa[gi].top:=round((viewcoef*massa[gi][-4].y*1000)-
gmassa[gi].height/2+50);
 gmassa[gi].left:=round((viewcoef*massa[gi][-4].x*1000)-
gmassa[gi].width/2+50);
 {local para inserir width e height da massa se quiser alterar tamanho}
 end;

{ fase 2 - atualização das molas}
```

```
for gi:=1 to nummola do
 begin
 {calcula L0}
 el1x:=massa[mola[gi].el1][-4].x;
 el2x:=massa[mola[gi].el2][-4].x;
 el1y:=massa[mola[gi].el1][-4].y;
 el2y:=massa[mola[gi].el2][-4].y;
 mola[gi].l:=sqrt(power((el1x-el2x),2)+power((el1y-el2y),2));
 if el1x=el2x then angt:=90
   else angt:=radtograd(Arctan((el1y-el2y)/(el1x-el2x)));
 if angt>=0 then gmola[gi].teta:=angt
   else gmola[gi].teta:=angt+360;
 if (angt=0) or (angt=180) then
   begin
   gmola[gi].top:=gmassa[mola[gi].el1].top;
   gmola[gi].height:=gmassa[mola[gi].el1].height;
   if el1x>el2x then
     begin
     gmola[gi].left:=gmassa[mola[gi].el2].left + gmassa[mola[gi].el2].width;
     gmola[gi].width:=gmassa[mola[gi].el1].left - gmola[gi].left;
     end
   else
     begin
     gmola[gi].left:=gmassa[mola[gi].el1].left + gmassa[mola[gi].el1].width;
     gmola[gi].width:=gmassa[mola[gi].el2].left - gmola[gi].left;
```

```
      end
    end
  else
    if (angt=90) or (angt=270) then
      begin
      gmola[gi].left:=gmassa[mola[gi].el1].left;
      gmola[gi].width:=gmassa[mola[gi].el1].width;
      if el1y>el2y then
        begin
        gmola[gi].top:=gmassa[mola[gi].el2].top + gmassa[mola[gi].el2].height;
        gmola[gi].height:=gmassa[mola[gi].el1].top - gmola[gi].top;
        end
      else
        begin
        gmola[gi].top:=gmassa[mola[gi].el1].top + gmassa[mola[gi].el1].height;
        gmola[gi].height:=gmassa[mola[gi].el2].top - gmola[gi].top;
        end;
      end
    else
      if ((angt>0) and (angt<90)) or ((angt>180) and (angt<270)) then
        begin
        if el1x>el2x then
          begin
          gmola[gi].left:=gmassa[mola[gi].el2].left + gmassa[mola[gi].el2].width;
          gmola[gi].top:=gmassa[mola[gi].el2].top + gmassa[mola[gi].el2].height;
```

```
    gmola[gi].width:=gmassa[mola[gi].el1].left - gmola[gi].left;
    gmola[gi].height:=gmassa[mola[gi].el1].top - gmola[gi].top;
    end
  else
    begin
    gmola[gi].left:=gmassa[mola[gi].el1].left + gmassa[mola[gi].el1].width;
    gmola[gi].top:=gmassa[mola[gi].el1].top + gmassa[mola[gi].el1].height;
    gmola[gi].width:=gmassa[mola[gi].el2].left - gmola[gi].left;
    gmola[gi].height:=gmassa[mola[gi].el2].top - gmola[gi].top;
    end;
  end
 else
  {90<angt<180 ou angt>270}
  begin
  if el1x>el2x then
    begin
    gmola[gi].left:=gmassa[mola[gi].el2].left+gmassa[mola[gi].el2].width;
    gmola[gi].top:=gmassa[mola[gi].el1].top+gmassa[mola[gi].el1].height;
    gmola[gi].width:=gmassa[mola[gi].el1].left - gmola[gi].left;
    gmola[gi].height:=gmassa[mola[gi].el2].top - gmola[gi].top;
    end
  else
    begin
    gmola[gi].left:=gmassa[mola[gi].el1].left+gmassa[mola[gi].el1].width;
    gmola[gi].top:=gmassa[mola[gi].el2].top+gmassa[mola[gi].el2].height;
```

```
            gmola[gi].width:=gmassa[mola[gi].el2].left - gmola[gi].left;
            gmola[gi].height:=gmassa[mola[gi].el1].top - gmola[gi].top;
            end;
          end;
 end;{end do loop das molas}

{ fase 3 - atualização dos amorts}
for gi:=1 to numamort do
 begin
 {calcula L0}
 el1x:=massa[amort[gi].el1][-4].x;
 el2x:=massa[amort[gi].el2][-4].x;
 el1y:=massa[amort[gi].el1][-4].y;
 el2y:=massa[amort[gi].el2][-4].y;
 amort[gi].l:=sqrt(power((el1x-el2x),2)+power((el1y-el2y),2));
 if el1x=el2x then angt:=90
   else angt:=radtograd(Arctan((el1y-el2y)/(el1x-el2x)));
 if angt>=0 then gamort[gi].teta:=angt
   else gamort[gi].teta:=angt+360;
 if (angt=0) or (angt=180) then
   begin
   gamort[gi].top:=gmassa[amort[gi].el1].top;
   gamort[gi].height:=gmassa[amort[gi].el1].height;
   if el1x>el2x then
     begin
```

```
      gamort[gi].left:=gmassa[amort[gi].el2].left + gmassa[amort[gi].el2].width;
      gamort[gi].width:=gmassa[amort[gi].el1].left - gamort[gi].left;
      end
    else
      begin
      gamort[gi].left:=gmassa[amort[gi].el1].left + gmassa[amort[gi].el1].width;
      gamort[gi].width:=gmassa[amort[gi].el2].left - gamort[gi].left;
      end
    end
  else
    if (angt=90) or (angt=270) then
      begin
      gamort[gi].left:=gmassa[amort[gi].el1].left;
      gamort[gi].width:=gmassa[amort[gi].el1].width;
      if el1y>el2y then
        begin
        gamort[gi].top:=gmassa[amort[gi].el2].top + gmassa[amort[gi].el2].height;
        gamort[gi].height:=gmassa[amort[gi].el1].top - gamort[gi].top;
        end
      else
        begin
        gamort[gi].top:=gmassa[amort[gi].el1].top + gmassa[amort[gi].el1].height;
        gamort[gi].height:=gmassa[amort[gi].el2].top - gamort[gi].top;
        end;
      end
```

```
    else
      if ((angt>0) and (angt<90)) or ((angt>180) and (angt<270)) then
       begin
       if el1x>el2x then
         begin
         gamort[gi].left:=gmassa[amort[gi].el2].left +
gmassa[amort[gi].el2].width;
         gamort[gi].top:=gmassa[amort[gi].el2].top +
gmassa[amort[gi].el2].height;
         gamort[gi].width:=gmassa[amort[gi].el1].left - gamort[gi].left;
         gamort[gi].height:=gmassa[amort[gi].el1].top - gamort[gi].top;
         end
       else
         begin
         gamort[gi].left:=gmassa[amort[gi].el1].left +
gmassa[amort[gi].el1].width;
         gamort[gi].top:=gmassa[amort[gi].el1].top +
gmassa[amort[gi].el1].height;
         gamort[gi].width:=gmassa[amort[gi].el2].left - gamort[gi].left;
         gamort[gi].height:=gmassa[amort[gi].el2].top - gamort[gi].top;
         end;
       end
      else
       {90<angt<180 ou angt>270}
       begin
```

```
        if el1x>el2x then
          begin
          gamort[gi].left:=gmassa[amort[gi].el2].left+gmassa[amort[gi].el2].width;

gamort[gi].top:=gmassa[amort[gi].el1].top+gmassa[amort[gi].el1].height;
          gamort[gi].width:=gmassa[amort[gi].el1].left - gamort[gi].left;
          gamort[gi].height:=gmassa[amort[gi].el2].top - gamort[gi].top;
          end
        else
          begin
          gamort[gi].left:=gmassa[amort[gi].el1].left+gmassa[amort[gi].el1].width;

gamort[gi].top:=gmassa[amort[gi].el2].top+gmassa[amort[gi].el2].height;
          gamort[gi].width:=gmassa[amort[gi].el2].left - gamort[gi].left;
          gamort[gi].height:=gmassa[amort[gi].el1].top - gamort[gi].top;
          end;
        end;
 end;{end do loop das amort}

{Fase 4 - loop das forças }
for gi:=1 to numforca do
 begin
 angt:=radtodeg(forca[gi].teta);
 gforca[gi].teta:=angt;
 if ((angt=0) or (angt=180))then
```

```
    begin
    gforca[gi].width:=comprforca;
    gforca[gi].height:=largforca;
    gforca[gi].top:=round( gmassa[forca[gi].el1].top +
(gmassa[forca[gi].el1].height-largforca)/2);
    gforca[gi].left:=gmassa[forca[gi].el1].left + gmassa[forca[gi].el1].width;
    end
  else
    if ((angt=90) or (angt=270)) then
      begin
      gforca[gi].width:=largforca;
      gforca[gi].height:=comprforca;
      gforca[gi].top:=gmassa[forca[gi].el1].top + gmassa[forca[gi].el1].height;
      gforca[gi].left:=round(gmassa[forca[gi].el1].left +
(gmassa[forca[gi].el1].width-largforca)/2);


      end
    else
      if ( ((angt>0) and (angt<90)) or ((angt>180) and (angt<270))) then
        begin
        gforca[gi].width:=comprforca;
        gforca[gi].height:=comprforca;
        gforca[gi].top:=round(gmassa[forca[gi].el1].top +
gmassa[forca[gi].el1].height/2);
        gforca[gi].left:=gmassa[forca[gi].el1].left + gmassa[forca[gi].el1].width;
```

```
        end
      else
        begin
        gforca[gi].width:=comprforca;
        gforca[gi].height:=comprforca;
        gforca[gi].top:=round(gmassa[forca[gi].el1].top +
(gmassa[forca[gi].el1].height - comprforca)/2 );
        gforca[gi].left:=gmassa[forca[gi].el1].left + gmassa[forca[gi].el1].width;
        end;



 form1.Paint;
 end;

end;
procedure TForm1.ToolButton3Click(Sender: TObject);
begin
numamort:=numamort+1;
gamort[numamort]:=Tgraphelem.Create(self);
gamort[numamort].parent:=form1;
gamort[numamort].number:=numamort;
gamort[numamort].ondblclick:=amortclick;
gamort[numamort].tipo:=tamort;

numbruno:=numamort;
```

```
{Atualiza o combobox}
tmpstrg:='Damper ' + inttostr(numamort);
combobox2.items.Add(tmpstrg);
form4.show;
end;

procedure TForm1.Amortclick(Sender : TObject);
begin
numbruno:=(sender as TGraphelem).number;
form4.show;
end;


procedure TForm1.ToolButton9Click(Sender: TObject);
begin
nummassa:=nummassa+1;
gmassa[nummassa]:=tgraphelem.create(self);
gmassa[nummassa].parent:=form1;
gmassa[nummassa].number:=nummassa;
gmassa[nummassa].ondblclick:=frestclick;
gmassa[nummassa].tipo:=tfrest;
gmassa[nummassa].width:=24;
gmassa[nummassa].height:=20;
numbruno:=nummassa;
with massa[nummassa][-4] do
```

```
 begin
 massa:=0;
 vx:=0;
 vy:=0;
 ax:=0;
 ay:=0;
 tipo:=frestraint;
 end;
{Atualiza o combobox}
tmpstrg:='Mass/Restr. ' + inttostr(nummassa);
combobox2.items.Add(tmpstrg);

form6.show;

end;
procedure TForm1.Frestclick(Sender : TObject);
begin
numbruno:=(sender as TGraphelem).number;
form6.show;
end;
procedure TForm1.ToolButton4Click(Sender: TObject);
begin
numforca:=numforca+1;
gforca[numforca]:=tgraphelem.create(self);
gforca[numforca].parent:=form1;
```

```
gforca[numforca].number:=numforca;
gforca[numforca].ondblclick:=forcaclick;
gforca[numforca].tipo:=tforca;
numbruno:=numforca;
{Atualiza o combobox}
tmpstrg:='Force ' + inttostr(numforca);
combobox2.items.Add(tmpstrg);

form7.show;
end;

procedure TForm1.Forcaclick (Sender : TObject);
begin
numbruno:=(sender as TGraphelem).number;
form7.show;
end;

procedure TForm1.SaveAs1Click(Sender: TObject);
var i,j,tempor : integer;
   anggrau : extended;
begin
savedialog1.filename:='*.mpr';
savedialog1.initialdir:=form2.Edit5.text;

if savedialog1.execute then
```

```
begin
assignfile(arqsyst,savedialog1.filename);
rewrite(arqsyst);
{gravacao dos dados}
writeln(arqsyst,nummassa);
writeln(arqsyst,nummola);
writeln(arqsyst,numamort);
writeln(arqsyst,numforca);
{ grava as massas }
for i:=1 to nummassa do
  begin
  write(arqsyst,massa[i][-4].massa);write(arqsyst,' ');
  write(arqsyst,massa[i][-4].x);write(arqsyst,' ');
  write(arqsyst,massa[i][-4].y);write(arqsyst,' ');
  write(arqsyst,massa[i][-4].vx);write(arqsyst,' ');
  write(arqsyst,massa[i][-4].vy);write(arqsyst,' ');
  write(arqsyst,massa[i][-4].ax);write(arqsyst,' ');
  write(arqsyst,massa[i][-4].ay);write(arqsyst,' ');
  if massa[i][-4].tipo=mass then tempor:=1
    else if massa[i][-4].tipo=frestraint then tempor:=2
      else if massa[i][-4].tipo=vrestraint then tempor:=3
        else if massa[i][-4].tipo=hrestraint then tempor:=4;
  writeln(arqsyst,tempor);
```

```
    end;

 {grava as molas }
 for i:=1 to nummola do
    begin
    write(arqsyst,mola[i].k1);write(arqsyst,' ');
    write(arqsyst,mola[i].el1);write(arqsyst,' ');
    writeln(arqsyst,mola[i].el2);write(arqsyst,' ');
    end;

{grava os amort}
for i:=1 to numamort do
   begin
   write(arqsyst,amort[i].c);write(arqsyst,' ');
   write(arqsyst,amort[i].el1);write(arqsyst,' ') ;
   writeln(arqsyst,amort[i].el2);write(arqsyst,' ');
   end;

{grava as forcas aplicadas }
for i:=1 to numforca do
   begin
   write(arqsyst,forca[i].t0);write(arqsyst,' ');
   write(arqsyst,forca[i].tf);write(arqsyst,' ');
   anggrau:=forca[i].teta*180/pi;
   write(arqsyst,anggrau);write(arqsyst,' ');
```

```
    write(arqsyst,forca[i].el1);write(arqsyst,' ');
    {le o tipo de forca}

    case forca[i].tipo of
        senoidal : tempor:=1;
        cossenoidal : tempor:=2;
        polinomial : tempor:=3;
        end;
    write(arqsyst,tempor);write(arqsyst,' ');
    if (forca[i].tipo=senoidal) or (forca[i].tipo=cossenoidal) then
      begin
      write(arqsyst,forca[i].a);write(arqsyst,' ');
      writeln(arqsyst,forca[i].w);
      end
      else if forca[i].tipo=polinomial then
        begin
        for j:=0 to 29 do write(arqsyst,forca[i].p[j]);write(arqsyst,' ');
        writeln(arqsyst);
        end;
    end;



closefile(arqsyst);
end;
```

```
end;

procedure TForm1.New1Click(Sender: TObject);
var i,ibru2: integer;
begin
{zera tudo }
for i:=-4 to 3 do
   inicializa(i);
for i:=1 to nummassa do
   gmassa[i].destroy;
for i:=1 to nummola do
   begin
   gmola[i].destroy;
   mola[i].k1:=0;
   mola[i].el1:=0;
   mola[i].el2:=0;
   mola[i].k2:=0;
   mola[i].k3:=0;
   mola[i].n:=0;
   end;
for i:=1 to numamort do
   begin
   gamort[i].destroy;
   with amort[i] do
```

```
        begin
        c:=0;
        el1:=0;
        el2:=0;
        l:=0;
        end;
    end;
for i:=1 to numforca do
    begin
    gforca[i].destroy;
    forca[i].el1:=0;
    forca[i].teta:=0;
    forca[i].tipo:=senoidal;
    forca[i].a:=0;
    forca[i].w:=0;
    forca[i].t0:=0;
    forca[i].tf:=0;
    for ibru2:=0 to 29 do forca[i].p[ibru2]:=0;
    end;
nummassa:=0;
nummola:=0;
numamort:=0;
numforca:=0;
graphrefresh;
end;
```

```
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
viewcoef:=viewcoef*(strtofloat(Combobox1.text)/100);
form1.graphrefresh;
end;



procedure TForm1.Help1Click(Sender: TObject);
begin
aboutbox.show;
end;

procedure TForm1.ComboBox2Change(Sender: TObject);


begin
numbruno:=strtoint(combobox2.text[length(combobox2.text)]);
if (combobox2.text[1]='M') or (combobox2.text='R') then form3.show
else
 if combobox2.Text[1]='S' then form5.show
 else
   if combobox2.text[1]='D' then form4.show
   else
     if combobox2.text[1]='F' then form7.show;
```

```
end;

end.
```

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm2 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    GroupBox1: TGroupBox;
    GroupBox3: TGroupBox;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    GroupBox4: TGroupBox;
    CheckBox3: TCheckBox;
    Edit3: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    CheckBox5: TCheckBox;
    Button1: TButton;
    Button2: TButton;
    CheckBox1: TCheckBox;
```

```
    CheckBox2: TCheckBox;
    GroupBox2: TGroupBox;
    StaticText1: TStaticText;
    StaticText4: TStaticText;
    Edit4: TEdit;
    Edit5: TEdit;
    StaticText5: TStaticText;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Unit1;

{$R *.DFM}
```

```
procedure TForm2.Button1Click(Sender: TObject);
begin
{Fecha a janela de options e confirma valores alterados}
passo:=(strtofloat(edit2.text)/1000);
tfinal:=(strtofloat(edit1.text)/1000);
form1.opendialog1.InitialDir:=edit5.text;
form1.savedialog1.initialdir:=edit5.text;
form1.savedialog2.InitialDir:=edit4.text;
salvaforca:=checkbox1.checked;
salvaerro:=checkbox2.checked;
form2.close;
pontosaver:=strtoint(edit3.text);
anima:=checkbox3.checked;
videorate:=strtoint(edit6.text);
end;

procedure TForm2.Button2Click(Sender: TObject);
begin
{fecha a janela options e não confirma os valores }
form2.close;
end;

end.
```

```
unit Unit3;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls,unit1;

type
 TForm3 = class(TForm)
   Button1: TButton;
   Button2: TButton;
   Edit1: TEdit;
   Edit2: TEdit;
   Edit3: TEdit;
   Edit4: TEdit;
   Edit5: TEdit;
   Edit6: TEdit;
   StaticText1: TStaticText;
   StaticText2: TStaticText;
   StaticText3: TStaticText;
   StaticText4: TStaticText;
   StaticText5: TStaticText;
   StaticText6: TStaticText;
   Edit7: TEdit;
```

```
    StaticText7: TStaticText;
    StaticText8: TStaticText;
    StaticText9: TStaticText;
    StaticText10: TStaticText;
    StaticText11: TStaticText;
    StaticText12: TStaticText;
    StaticText13: TStaticText;
    StaticText14: TStaticText;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

uses Unit10;
```

```
{$R *.DFM}

procedure TForm3.Button1Click(Sender: TObject);
begin
with massa[numbruno][-4] do
 begin
 massa:=strtofloat(edit7.text);
 x:=strtofloat(edit1.text);
 y:=strtofloat(edit2.text);
 vx:=strtofloat(edit3.text);
 vy:=strtofloat(edit4.text);
 ax:=strtofloat(edit5.text);
 ay:=strtofloat(edit6.text);
 tipo:=mass;
 end;
gmassa[numbruno].top:=round((viewcoef*massa[numbruno][-4].y*1000)-
gmassa[numbruno].height/2+50);
gmassa[numbruno].left:=round((viewcoef*massa[numbruno][-4].x*1000)-
gmassa[numbruno].width/2+50);
numbruno:=-1;

form3.close;
form1.Graphrefresh;
```

```
end;

procedure TForm3.FormShow(Sender: TObject);
begin
form3.caption:='Mass '+inttostr(numbruno)+' Properties';
edit7.text:=floattostr(massa[numbruno][-4].massa);
edit1.text:=floattostr(massa[numbruno][-4].x);
edit2.text:=floattostr(massa[numbruno][-4].y);
edit3.text:=floattostr(massa[numbruno][-4].vx);
edit4.text:=floattostr(massa[numbruno][-4].vy);
edit5.text:=floattostr(massa[numbruno][-4].ax);
edit6.text:=floattostr(massa[numbruno][-4].ay);
end;

procedure TForm3.Button2Click(Sender: TObject);
begin
form3.close;
end;

procedure TForm3.Button3Click(Sender: TObject);
var ibru,cci,ligacoes : integer;
begin
{verifica ligações}
ligacoes:=0;
for cci:=1 to nummola do
```

```
    if (mola[cci].el1=numbruno) or (mola[cci].el2=numbruno) then
ligacoes:=ligacoes+1;
for cci:=1 to numamort do
    if (amort[cci].el1=numbruno) or (amort[cci].el2=numbruno) then
ligacoes:=ligacoes+1;
for cci:=1 to numforca do
    if forca[cci].el1=numbruno then ligacoes:=ligacoes+1;

{apaga ou não dependendo das ligações}

if ligacoes=0 then
  begin
  for ibru:=numbruno to (nummassa-1) do
      begin
      massa[ibru][-4].n:=massa[ibru+1][-4].n;
      massa[ibru][-4].massa:=massa[ibru+1][-4].massa;
      massa[ibru][-4].x:=massa[ibru+1][-4].x;
      massa[ibru][-4].y:=massa[ibru+1][-4].y;
      massa[ibru][-4].vx:=massa[ibru+1][-4].vx;
      massa[ibru][-4].vy:=massa[ibru+1][-4].vy;
      massa[ibru][-4].ax:=massa[ibru+1][-4].ax;
      massa[ibru][-4].ay:=massa[ibru+1][-4].ay;
      gmassa[ibru].tipo:=gmassa[ibru+1].tipo;
      end;
  gmassa[nummassa].destroy;
```

```
 with massa[nummassa][-4] do
     begin
     n:=0;
     massa:=0;
     x:=0;
     y:=0;
     vx:=0;
     vy:=0;
     ax:=0;
     ay:=0;
     end;
{atualiza ligações}
for cci:=1 to nummola do
    begin
    if mola[cci].el1>numbruno then mola[cci].el1:=mola[cci].el1-1;
    if mola[cci].el2>numbruno then mola[cci].el2:=mola[cci].el2-1;
    end;
for cci:=1 to numamort do
    begin
    if amort[cci].el1>numbruno then amort[cci].el1:=amort[cci].el1-1;
    if amort[cci].el2>numbruno then amort[cci].el2:=amort[cci].el2-1;
    end;
for cci:=1 to numforca do
    if forca[cci].el1>numbruno then forca[cci].el1:=forca[cci].el1-1;
```

```
  form1.combobox2.Items.Delete(form1.combobox2.items.indexof('Mass/Restr.
'+inttostr(nummassa)));
  nummassa:=nummassa-1;
  numbruno:=-1;
  form1.graphrefresh;
  form3.close;
  end
else
   begin
   form10.statictext2.caption:='Please delete the '+inttostr(ligacoes)+' elements
connected to mass/restraint '+inttostr(numbruno)+' and try again';
   form10.statictext2.width:=345;
   form10.statictext1.width:=345;
   form10.statictext2.height:=345;
   form10.statictext1.height:=35;

   form3.close;
   form10.show;
   numbruno:=-1;
   end;




end;
```

end.

```
unit Unit4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,unit1,graphelem,math;

type
  TForm4 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    Edit7: TEdit;
    StaticText7: TStaticText;
    StaticText8: TStaticText;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;

implementation

{$R *.DFM}


procedure TForm4.Button1Click(Sender: TObject);
var del1x,del1y,del2x,del2y : extended;
begin

amort[numbruno].c:=strtofloat(edit7.text);
amort[numbruno].el1:=strtoint(edit1.text);
amort[numbruno].el2:=strtoint(edit2.text);

del1x:=massa[amort[numbruno].el1][-4].x;
del1y:=massa[amort[numbruno].el1][-4].y;
del2x:=massa[amort[numbruno].el2][-4].x;
del2y:=massa[amort[numbruno].el2][-4].y;
```

```
amort[numbruno].l:=sqrt(power((del1x-del2x),2)+power((del1y-del2y),2));

numbruno:=-1;
form4.close;
form1.graphrefresh;

end;

procedure TForm4.Button2Click(Sender: TObject);
begin
form4.close;
end;

procedure TForm4.FormShow(Sender: TObject);
begin
form4.caption:='Damper '+inttostr(numbruno)+' Properties';
edit7.text:=floattostr(amort[numbruno].c);
edit1.text:=inttostr(amort[numbruno].el1);
edit2.text:=inttostr(amort[numbruno].el2);

end;

procedure TForm4.Button3Click(Sender: TObject);
var ibru : integer;
```

```
begin
for ibru:=numbruno to numamort do
   begin
   amort[ibru].c:=amort[ibru+1].c;
   amort[ibru].el1:=amort[ibru+1].el1;
   amort[ibru].el2:=amort[ibru+1].el2;
   amort[ibru].l:=amort[ibru+1].l;
   end;
gamort[numamort].destroy;
with amort[numamort] do
   begin
   c:=0;
   el1:=0;
   el2:=0;
   l:=0;
   end;
form1.combobox2.Items.Delete(form1.combobox2.items.indexof('Damper
'+inttostr(numamort)));
numamort:=numamort-1;
numbruno:=-1;
form4.close;
form1.graphrefresh;
end;

end.
```

```
unit Unit5;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls,unit1,math;

type
 TForm5 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Edit1: TEdit;
  Edit2: TEdit;
  StaticText1: TStaticText;
  StaticText2: TStaticText;
  Edit7: TEdit;
  StaticText7: TStaticText;
  StaticText8: TStaticText;
  Button3: TButton;
  Edit3: TEdit;
  StaticText3: TStaticText;
  StaticText4: TStaticText;
  procedure FormShow(Sender: TObject);
  procedure Button1Click(Sender: TObject);
```

```
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;

implementation

{$R *.DFM}



procedure TForm5.FormShow(Sender: TObject);
begin
form5.caption:='Spring '+inttostr(numbruno)+' Properties';
edit7.text:=floattostr(mola[numbruno].k1);
edit1.text:=inttostr(mola[numbruno].el1);
edit2.text:=inttostr(mola[numbruno].el2);
edit3.text:=inttostr(0);
end;
```

```
procedure TForm5.Button1Click(Sender: TObject);
var kel1x,kel2x,kel1y,kel2y : extended;
begin
{Inserir verificação da existência dos elementos}
mola[numbruno].k1:=strtofloat(edit7.text);
mola[numbruno].el1:=strtoint(edit1.text);
mola[numbruno].el2:=strtoint(edit2.text);
mola[numbruno].l:=strtofloat(edit3.text);

kel1x:=massa[mola[numbruno].el1][-4].x;
kel1y:=massa[mola[numbruno].el1][-4].y;
kel2x:=massa[mola[numbruno].el2][-4].x;
kel2y:=massa[mola[numbruno].el2][-4].y;
if mola[numbruno].l=0 then
 mola[numbruno].l:=sqrt(power((kel1x-kel2x),2)+power((kel1y-kel2y),2));

form1.graphrefresh;
numbruno:=-1;
form5.close;



end;

procedure TForm5.Button3Click(Sender: TObject);
```

```
var ibru : integer;
begin
for ibru:=numbruno to nummola do
   begin
   mola[ibru].k1:=mola[ibru+1].k1;
   mola[ibru].el1:=mola[ibru+1].el1;
   mola[ibru].el2:=mola[ibru+1].el2;
   mola[ibru].k2:=mola[ibru+1].k2;
   mola[ibru].k3:=mola[ibru+1].k3;
   mola[ibru].n:=mola[ibru+1].n;
   mola[ibru].l:=mola[ibru+1].l;
   end;
gmola[nummola].destroy;
mola[nummola].k1:=0;
mola[nummola].el1:=0;
mola[nummola].el2:=0;
mola[nummola].k2:=0;
mola[nummola].k3:=0;
mola[nummola].n:=0;
form1.combobox2.Items.Delete(form1.combobox2.items.indexof('Spring
'+inttostr(nummola)));
nummola:=nummola-1;
numbruno:=-1;
form1.graphrefresh;
form5.close;
```

```
end;

procedure TForm5.Button2Click(Sender: TObject);
begin
form5.close;
end;



end.
```

```
unit Unit6;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,unit1;

type
  TForm6 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText9: TStaticText;
    StaticText10: TStaticText;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button3Click(Sender: TObject);
private
  { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  Form6: TForm6;

implementation

uses Unit10;

{$R *.DFM}



procedure TForm6.Button1Click(Sender: TObject);
begin
massa[numbruno][-4].x:=strtofloat(edit1.text);
massa[numbruno][-4].y:=strtofloat(edit2.text);
gmassa[numbruno].top:=round((viewcoef*massa[numbruno][-4].y*1000)-
gmassa[numbruno].height/2+50);
gmassa[numbruno].left:=round((viewcoef*massa[numbruno][-4].x*1000)-
gmassa[numbruno].width/2+50);
numbruno:=-1;
form1.graphrefresh;
```

```
form6.close;
end;

procedure TForm6.Button2Click(Sender: TObject);
begin
form6.close
end;

procedure TForm6.FormShow(Sender: TObject);
begin
form6.caption:='Restraint '+inttostr(numbruno)+' Properties';
edit1.text:=floattostr(massa[numbruno][-4].x);
edit2.text:=floattostr(massa[numbruno][-4].y);
end;

procedure TForm6.Button3Click(Sender: TObject);
var ligacoes,cci,ibru : integer;
begin
{verifica ligações}
ligacoes:=0;
for cci:=1 to nummola do
   if (mola[cci].el1=numbruno) or (mola[cci].el2=numbruno) then
ligacoes:=ligacoes+1;
for cci:=1 to numamort do
```

```
   if (amort[cci].el1=numbruno) or (amort[cci].el2=numbruno) then
ligacoes:=ligacoes+1;
for cci:=1 to numforca do
   if forca[cci].el1=numbruno then ligacoes:=ligacoes+1;

{apaga ou não dependendo das ligações}

if ligacoes=0 then
  begin
  for ibru:=numbruno to nummassa do
     begin
     massa[ibru][-4].n:=massa[ibru+1][-4].n;
     massa[ibru][-4].massa:=massa[ibru+1][-4].massa;
     massa[ibru][-4].x:=massa[ibru+1][-4].x;
     massa[ibru][-4].y:=massa[ibru+1][-4].y;
     massa[ibru][-4].vx:=massa[ibru+1][-4].vx;
     massa[ibru][-4].vy:=massa[ibru+1][-4].vy;
     massa[ibru][-4].ax:=massa[ibru+1][-4].ax;
     massa[ibru][-4].ay:=massa[ibru+1][-4].ay;
     gmassa[ibru].tipo:=gmassa[ibru+1].tipo;
     end;
  gmassa[nummassa].destroy;
  with massa[nummassa][-4] do
     begin
     n:=0;
```

```
        massa:=0;
        x:=0;
        y:=0;
        vx:=0;
        vy:=0;
        ax:=0;
        ay:=0;
        end;
    {atualiza ligações}
    for cci:=1 to nummola do
        begin
        if mola[cci].el1>numbruno then mola[cci].el1:=mola[cci].el1-1;
        if mola[cci].el2>numbruno then mola[cci].el2:=mola[cci].el2-1;
        end;
    for cci:=1 to numamort do
        begin
        if amort[cci].el1>numbruno then amort[cci].el1:=amort[cci].el1-1;
        if amort[cci].el2>numbruno then amort[cci].el2:=amort[cci].el2-1;
        end;
    for cci:=1 to numforca do
        if forca[cci].el1>numbruno then forca[cci].el1:=forca[cci].el1-1;
    form1.combobox2.Items.Delete(form1.combobox2.items.indexof('Mass/Restr.
'+inttostr(nummassa)));
    nummassa:=nummassa-1;
    numbruno:=-1;
```

```
    form1.graphrefresh;
    form6.close;
    end
  else
    begin
    form10.statictext2.caption:='Please delete the '+inttostr(ligacoes)+' elements
 connected to mass/restraint '+inttostr(numbruno)+' and try again';
    form10.statictext2.width:=345;
    form10.statictext1.width:=345;
    form10.statictext2.height:=345;
    form10.statictext1.height:=35;

    form6.close;
    form10.show;
    numbruno:=-1;
    end;
end;

end.
```

```
unit Unit7;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls,unit1,math;

type
  TForm7 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    RadioGroup1: TRadioGroup;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    StaticText4: TStaticText;
    StaticText5: TStaticText;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    Button2: TButton;
    Button1: TButton;
```

```
    Button3: TButton;
    Edit5: TEdit;
    Edit6: TEdit;
    StaticText6: TStaticText;
    StaticText7: TStaticText;
    StaticText8: TStaticText;
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form7: TForm7;

implementation

uses Unit8;

{$R *.DFM}
```

```
procedure TForm7.Button1Click(Sender: TObject);
begin
forca[numbruno].el1:=strtoint(edit1.text);
forca[numbruno].teta:=degtorad(strtofloat(edit2.text));
forca[numbruno].A:=strtofloat(edit3.text);
forca[numbruno].w:=strtofloat(edit4.text);
forca[numbruno].t0:=strtofloat(edit5.text)/1000;
forca[numbruno].tf:=strtofloat(edit6.text)/1000;

if radiobutton1.checked=true then
  begin
  forca[numbruno].tipo:=senoidal;
  numbruno:=-1;
  end
 else if radiobutton2.checked=true then
     begin
     forca[numbruno].tipo:=cossenoidal;
     numbruno:=-1;
     end
```

```
    else if radiobutton3.checked=true then
          begin
          forca[numbruno].tipo:=polinomial;
          form8.show;
          end;
form7.close;

form1.graphrefresh;

{falta}

end;

procedure TForm7.FormShow(Sender: TObject);
begin
form7.caption:='Force '+inttostr(numbruno)+' Properties';
edit1.text:=inttostr(forca[numbruno].el1);
edit2.text:=floattostr(radtodeg(forca[numbruno].teta));
edit3.text:=floattostr(forca[numbruno].A);
edit4.text:=floattostr(forca[numbruno].w);
edit5.text:=floattostr(1000*forca[numbruno].t0);
edit6.text:=floattostr(1000*forca[numbruno].tf);
if forca[numbruno].tipo=senoidal then radiobutton1.checked:=true
 else if forca[numbruno].tipo=cossenoidal then radiobutton2.checked:=true
   else if forca[numbruno].tipo=polinomial then radiobutton3.checked:=true;
```

```
end;

procedure TForm7.Button2Click(Sender: TObject);
begin
form7.close;
numbruno:=-1;
end;

procedure TForm7.Button3Click(Sender: TObject);
var ibru,ibru2 : integer;
begin
{move uma posição }
for ibru:=numbruno to numforca-1 do
   begin
   forca[ibru].el1:=forca[ibru+1].el1;
   forca[ibru].teta:=forca[ibru+1].teta;
   forca[ibru].tipo:=forca[ibru+1].tipo;
   forca[ibru].a:=forca[ibru+1].a;
   forca[ibru].w:=forca[ibru+1].w;
   forca[ibru].t0:=forca[ibru+1].t0;
   forca[ibru].tf:=forca[ibru+1].tf;
   for ibru2:=0 to 29 do forca[ibru].p[ibru2]:=forca[ibru+1].p[ibru2];
   end;
   {apaga}
```

```
    gforca[numforca].destroy;
    forca[numforca].el1:=0;
    forca[numforca].teta:=0;
    forca[numforca].tipo:=senoidal;
    forca[numforca].a:=0;
    forca[numforca].w:=0;
    forca[numforca].t0:=0;
    forca[numforca].tf:=0;
    for ibru2:=0 to 29 do forca[numforca].p[ibru2]:=0;
    form1.combobox2.Items.Delete(form1.combobox2.items.indexof('Force
'+inttostr(numforca)));

    numforca:=numforca-1;
    numbruno:=-1;
    form7.close;

form1.graphrefresh;

end;

end.
```

```
unit Unit8;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,unit1;

type
  TForm8 = class(TForm)
    Edit1: TEdit;
    StaticText1: TStaticText;
    Edit2: TEdit;
    StaticText2: TStaticText;
    Edit3: TEdit;
    StaticText3: TStaticText;
    Edit4: TEdit;
    StaticText4: TStaticText;
    Edit5: TEdit;
    StaticText5: TStaticText;
    Edit6: TEdit;
    StaticText6: TStaticText;
    Edit7: TEdit;
    StaticText7: TStaticText;
    Edit8: TEdit;
```

```
StaticText8: TStaticText;
Edit9: TEdit;
StaticText9: TStaticText;
Edit10: TEdit;
StaticText10: TStaticText;
Edit11: TEdit;
StaticText11: TStaticText;
Edit12: TEdit;
StaticText12: TStaticText;
Edit13: TEdit;
StaticText13: TStaticText;
Edit14: TEdit;
StaticText14: TStaticText;
Edit15: TEdit;
StaticText15: TStaticText;
Edit16: TEdit;
StaticText16: TStaticText;
Edit17: TEdit;
StaticText17: TStaticText;
Edit18: TEdit;
StaticText18: TStaticText;
Edit19: TEdit;
StaticText19: TStaticText;
Edit20: TEdit;
StaticText20: TStaticText;
```

```
    Edit21: TEdit;
    StaticText21: TStaticText;
    Edit22: TEdit;
    StaticText22: TStaticText;
    Edit23: TEdit;
    StaticText23: TStaticText;
    Edit24: TEdit;
    StaticText24: TStaticText;
    Edit25: TEdit;
    StaticText25: TStaticText;
    Edit26: TEdit;
    StaticText26: TStaticText;
    Edit27: TEdit;
    StaticText27: TStaticText;
    Edit28: TEdit;
    StaticText28: TStaticText;
    Edit29: TEdit;
    StaticText29: TStaticText;
    Edit30: TEdit;
    StaticText30: TStaticText;
    Button1: TButton;
    StaticText31: TStaticText;
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form8: TForm8;

implementation

{$R *.DFM}

procedure TForm8.Button1Click(Sender: TObject);
begin
forca[numbruno].p[0]:=strtofloat(edit1.text);
forca[numbruno].p[1]:=strtofloat(edit2.text);
forca[numbruno].p[2]:=strtofloat(edit3.text);
```

```
forca[numbruno].p[3]:=strtofloat(edit4.text);
forca[numbruno].p[4]:=strtofloat(edit5.text);

forca[numbruno].p[5]:=strtofloat(edit6.text);
forca[numbruno].p[6]:=strtofloat(edit7.text);
forca[numbruno].p[7]:=strtofloat(edit8.text);
forca[numbruno].p[8]:=strtofloat(edit9.text);
forca[numbruno].p[9]:=strtofloat(edit10.text);

forca[numbruno].p[10]:=strtofloat(edit11.text);
forca[numbruno].p[11]:=strtofloat(edit12.text);
forca[numbruno].p[12]:=strtofloat(edit13.text);
forca[numbruno].p[13]:=strtofloat(edit14.text);
forca[numbruno].p[14]:=strtofloat(edit15.text);

forca[numbruno].p[15]:=strtofloat(edit16.text);
forca[numbruno].p[16]:=strtofloat(edit17.text);
forca[numbruno].p[17]:=strtofloat(edit18.text);
forca[numbruno].p[18]:=strtofloat(edit19.text);
forca[numbruno].p[19]:=strtofloat(edit20.text);

forca[numbruno].p[20]:=strtofloat(edit21.text);
forca[numbruno].p[21]:=strtofloat(edit22.text);
forca[numbruno].p[22]:=strtofloat(edit23.text);
forca[numbruno].p[23]:=strtofloat(edit24.Text);
```

```
forca[numbruno].p[24]:=strtofloat(edit25.text);

forca[numbruno].p[25]:=strtofloat(edit26.text);
forca[numbruno].p[26]:=strtofloat(edit27.text);
forca[numbruno].p[27]:=strtofloat(edit28.text);
forca[numbruno].p[28]:=strtofloat(edit29.text);
forca[numbruno].p[29]:=strtofloat(edit30.text);
form8.close;
numbruno:=-1;
end;

procedure TForm8.FormShow(Sender: TObject);
begin
edit1.text:=floattostr(forca[numbruno].p[0]);
edit2.text:=floattostr(forca[numbruno].p[1]);
edit3.text:=floattostr(forca[numbruno].p[2]);
edit4.text:=floattostr(forca[numbruno].p[3]);
edit5.text:=floattostr(forca[numbruno].p[4]);

edit6.text:=floattostr(forca[numbruno].p[5]);
edit7.text:=floattostr(forca[numbruno].p[6]);
edit8.text:=floattostr(forca[numbruno].p[7]);
edit9.text:=floattostr(forca[numbruno].p[8]);
edit10.text:=floattostr(forca[numbruno].p[9]);
```

```
edit11.text:=floattostr(forca[numbruno].p[10]);
edit12.text:=floattostr(forca[numbruno].p[11]);
edit13.text:=floattostr(forca[numbruno].p[12]);
edit14.text:=floattostr(forca[numbruno].p[13]);
edit15.text:=floattostr(forca[numbruno].p[14]);

edit16.text:=floattostr(forca[numbruno].p[15]);
edit17.text:=floattostr(forca[numbruno].p[16]);
edit18.text:=floattostr(forca[numbruno].p[17]);
edit19.text:=floattostr(forca[numbruno].p[18]);
edit20.text:=floattostr(forca[numbruno].p[19]);

edit21.text:=floattostr(forca[numbruno].p[20]);
edit22.text:=floattostr(forca[numbruno].p[21]);
edit23.text:=floattostr(forca[numbruno].p[22]);
edit24.text:=floattostr(forca[numbruno].p[23]);
edit25.text:=floattostr(forca[numbruno].p[24]);
edit26.text:=floattostr(forca[numbruno].p[25]);
edit27.text:=floattostr(forca[numbruno].p[26]);
edit28.text:=floattostr(forca[numbruno].p[27]);
edit29.text:=floattostr(forca[numbruno].p[28]);
edit30.text:=floattostr(forca[numbruno].p[29]);
end;

end.
```

```
unit Unit10;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm10 = class(TForm)
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form10: TForm10;

implementation
```

```
{$R *.DFM}

procedure TForm10.Button1Click(Sender: TObject);
begin
form10.close;
end;

end.
```

```
unit Unit11;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,unit1;

type
  TForm11 = class(TForm)
    Button1: TButton;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    StaticText4: TStaticText;
    StaticText5: TStaticText;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
private
  { Private declarations }
public
```

```
    { Public declarations }
  end;

var
  Form11: TForm11;

implementation

{$R *.DFM}

procedure TForm11.Button1Click(Sender: TObject);
begin
form11.close;
end;

procedure TForm11.FormShow(Sender: TObject);
begin
edit1.text:=floattostr(etposmax);
edit2.text:=floattostr(etspdmax);
```

```
edit3.text:=floattostr(eposmax);

edit4.text:=floattostr(espdmax);

end;


end.
```

ATENÇÃO: Todo o código reproduzido entre a partir da página 52 até a página 173, que é a implementação da resolução utilizada pelo software Mechcalc 2.0, é propriedade intelectual única e exclusivamente de Bruno Luiz Assaf - RG:27.052.627-4. A utilização total ou parcial do código em qualquer software, com funções comerciais ou não é absolutamente vedada, sendo necessária autorização prévia por escrito por parte do autor.

A utilização do código para qualquer fim não autorizado previamente pelo autor é uma violação das leis de direitos autorais, consistindo em crime pelas leis vigentes do país.

A reprodução acima destina-se somente para estudo e demonstração das possibilidades de utilização de métodos numéricos como fator de inovação no solucionamento de diversos problemas de engenharia e suas possibilidades. É absolutamente proibida qualquer tentativa de reescrever o software listado acima, compilá-lo, utilizar qualquer parte dele integrando outro software ou mesmo distribuí-lo. Qualquer interesse relativo à obtenção do software ou mesmo dúvidas, favor contatar o autor através do email mechcalc@brunoassaf.com .

A versão mais atual do software sempre poderá ser encontrada na internet para download em www.brunoassaf.com.