Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia Mecânica

Projeto Mecânico II

Sistema tutorial para modelagem e simulação de sistemas dinâmicos em engenharia

Autor:
Alexandre Valgôde do Nascimento

Orientador:
Prof. Dr. José Reinaldo Silva

Coordenador:
Prof. Dr. Marcelo Godoi Simões

São Paulo, 08 de dezembro de 1997.

Índice

Agradecimentos

Agradeço a todos aqueles que colaboraram para a elaboração deste trabalho, dando sugestões, dicas e contribuições para a implementação. Agradeço, principalmente, ao Doutor José Reinaldo Silva que, com paciência, forneceu conhecimento técnico e dedicação para que este trabalho pudesse ser concluído.

1. Motivação

Poucas são as ferramentas computacionais disponíveis no mercado para estudantes de engenharia que desejem lançar mão de cálculos numéricos para resolução de suas equações e problemas. Talvez a ferramenta computacional mais poderosa nesse sentido seja o software "MatLab/Simulink" que oferece poderosos recursos para projetos de sistemas em geral de engenharia, desde estruturas para o campo da engenharia civil a controladores na área de automação.

Mas como qualquer outro software de simulação numérica, o "MatLab" é apenas uma ferramenta de cálculo, cabendo ao estudante todo o processo de modelagem do sistema.

Assim, surge a idéia da construção de um software que auxilie o projetista não só na solução numérica como na modelagem do sistema.

2. Objetivos

A idéia do projeto é, então, a construção de um software tutorial que auxilie o estudante de engenharia (projetista em geral) na modelagem de determinados tipos de sistemas. O software será um tutorial na modelagem e uma ferramenta computacional de simulação numérica após a modelagem.

Seguindo a idéia da construção de um software didático para projetistas, o programa trará também explicações e deduções dos métodos numéricos a serem utilizados para a simulação do modelo que foi construído. Para que o programa seja uma ferramenta de fácil acesso para estudantes e projetistas é necessário também que possua uma interface amigável com o usuário tanto na comunicação de ações a serem executadas pelo programa como para a entrada de dados matemáticos.

Devido a impossibilidade de se produzir um software que pudesse auxiliar um projetista em qualquer tipo de modelagem de qualquer tipo de sistema, a modelagem se restringirá a alguns tipos de sistemas em engenharia que possam ser modelados com equações diferenciais. O software terá uma biblioteca de problemas clássicos de engenharia em que, após o usuário definir o problema que deseja resolver, o software auxiliará na sua modelagem e na simulação numérica.

3. Tutorial de modelagem

3.1 Modelagem de sistemas

Um modelo é a representação do comportamento essencial do sistema em relação aos objetivos da análise que se deseja proceder. O desenho de um sistema pode ser classificado como um modelo, bem como os modelos físicos em escala, gráficos, diagramas e tabelas já que permitem a melhor compreensão do problema em estudo. Em Engenharia, qualquer sistema físico que se conheça e se deseje analisar pode e deve ser tratado através de modelos, que neste caso serão os "modelos matemáticos".

Quando se deseja realizar o controle de uma determinada planta industrial, por exemplo, as da indústria química (que requerem modelagem em sistemas contínuos) ou qualquer sistema com comportamento dinâmico, os modelos recaem em bases de equações diferenciais (Ordinárias ou a Derivadas Parciais) de grau elevado ou não que, na maioria das vezes não possuem solução analítica (fechada). Para que se possa proceder a uma simulação destas equações a fim de se determinar o comportamento do sistema modelado se faz necessária a utilização de métodos numéricos eficientes para a simulação e resolução destas equações diferenciais.

3.2 O tutorial de modelagem do software

O software deve ajudar o usuário na modelagem do sistema. Considere o exemplo abaixo e veja como o software pode fazer isto !

Considere o problema da modelagem da suspensão de um quarto de veículo de massa M representado na figura abaixo. O sistema da suspensão pode ser modelado através de um sistema simples composto de massa-mola-amortecedor.

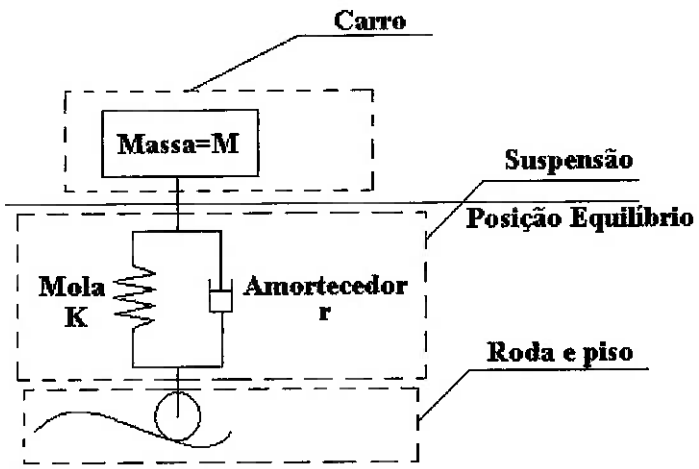


Figura 1 - Modelo gráfico da suspensão veicular

O usuário deverá então interagir com o software respondo às perguntas que lhe forem formuladas de modo que se possa obter a modelagem final do sistema com os parâmetros desejados. Na medida em que o usuário vai respondendo corretamente o software continua perguntando até o final da modelagem, caso a resposta esteja errada, o software dará dicas sobre a resposta correta de modo que o usuário possa prosseguir.

Veja abaixo uma maneira de implementar este tutorial:

P.: Quais as forças externas que estão agindo sobre o veículo ?

R.: A força da mola e do amortecedor.

P.: O carro oscilará em torno da posição de equilíbrio da figura, x(t). Como pode-se escrever a posição relativa do carro x_r (t) em relação ao referencial absoluto, levando-se em conta que o piso possui possui equação x_o (t) ?

R.: $x_r(t)=x(t)-x_o(t)$

P.: Qual a expressão da força que a mola aplica sobre o carro (F_m), sabendo-se que ela é proporcional à sua deformação relativa ? A constante de proporcionalidade da mola é K.

R.: $F_m=K.[x(t)-x_o(t)]$

P.: Qual a expressão da força que o amortecedor aplica sobre o carro (F_a), sabendo-se que ela é proporcional à sua velocidade relativa ? A constante de proporcionalidade do amortecedor é r.

R.: $F_a = r.\left(\frac{dx}{dt} - \frac{dx_o}{dt}\right)$

P.: Se as forças externas que agem sobre o sistema são da mola (F_m) e o amortecedor (F_a) use a 2°lei de Newton ($F_R=M.a$) aplicada no sistema ?

R.: $-F_m-F_a=M.a$

Substituindo-se as equações da força da mola e do amortecedor na equação acima obtida têm-se que:

$$M.\frac{d^2x}{dt^2} = -K.(x - x_o) - r\left(\frac{dx}{dt} - \frac{dx_o}{dt}\right),$$

A equação acima é a equação diferencial do sistema e representa um modelo matemático do sistema da suspensão, aproximado por um sistema massa-mola-amortecedor.

Para a resolução da equação diferencial representativa do modelo acima é necessário que estejam definidas as condições iniciais de operação.

P.: Qual a posição inicial x(0) do objeto ?

R.: x(0)=0

P.: Qual a velocidade inicial dx/dt | (x=0) ?

R.: dx/dt | (x=0) = 0

Agora que o sistema já está modelado, precisa-se determinar os parâmetros que deverão compor o sistema para que ele opere nas condições desejadas.

Sabe-se que a solução crítica da equação acima é: $\frac{r}{2.\sqrt{k.M}}$, assim, se a solução da equação for menor que este valor o sistema será subcrítico, igual será crítico e maior será super-crítico.

Após a modelagem serão determinadas as faixas dos parâmetros do sistema para que ele opere nas condições desejadas. Isto será feito com o auxílio da simulação numérica. O usuário dirá o método a ser utilizado para a resolução da equação e o programa fará a simulação devolvendo a resposta numerica e graficamente para auxiliar o usuário a enxergar o que está ocorrendo com o sistema. Através dos gráficos ficará claro o regime de operação da suspensão ao se observar os possíveis estados: crítico, sub-crítico ou super-crítico.

Dessa maneira a modelagem e simulação estarão completas. O usuário terá participado da modelagem do sistema e assistirá a resolução numérica. Os algoritmos empregados na solução poderão também ser consultados nas caixas de ajuda das deduções dos teoremas empregados nas resoluções.

3.3 Os tópicos da modelagem

Assim como foi feito acima, o tutorial poderá ajudar o usuário a modelar diversos tipos de sistemas. Alguns dos sistemas que o programa se habilitará a modelar e simular podem ser encontrados abaixo:

Equações diferenciais ordinárias

- ⇨ Modelagem e simulação da suspensão veicular.
- ⇨ Modelagem do sistema RLC de circuitos elétricos.
- ⇨ Modelagem e simulação do escoamento de calor em aletas com trocas de calor unidimensionais.

Equações diferenciais a derivadas parciais

⇨ Modelagem e simulação da defleção de chapas com carregamento qualquer dado.

⇨ Modelagem e simulação do escoamento de calor em aletas com trocas de calor bidimensionais.

Além destes outros assuntos poderão se implementados na forma de modelagem, na medida em que forem surgindo sugestões ao longo da implementação. Um dos tópicos mais importantes da Engenharia moderna - Teoria de Controle - poderá ser utilizada como tópico de modelagem e simulação para obtenção de controladores (PI,PD,PID) e compensadores.

4. Tutorial de Simulação numérica

4.1 Métodos numéricos

Seguindo a idéia da construção de um software didático para projetistas, o programa trará explicações e deduções dos métodos numéricos a serem utilizados para a simulação do modelo que foi construído. Estas explicações e deduções estarão sob a forma de um "help" que estará disponível para cada método assim que ele for requisitado como ferramenta de simulação do modelo construído. Em anexo encontram-se implementados a quase totalidade destes helps bem como figuras explicativas relativas a cada um dos métodos.

5. Interface com o Usuário

5.1 Níveis hierárquicos de comunicação do usuário com o programa

Os níveis de comunicação entre o usuário e software serão hierarquizados. Os níveis de hierarquização são, basicamente, definidos tela a tela. Em muitas telas o usuário deverá tomar decisões que o levarão a outros níveis hierárquicos que, cada vez mais definem e especificam a tarefaa ser realizada. Na fase de implementação será colocada a disposição hierárquica do software, relacionando-as com cada uma das telas implementadas.

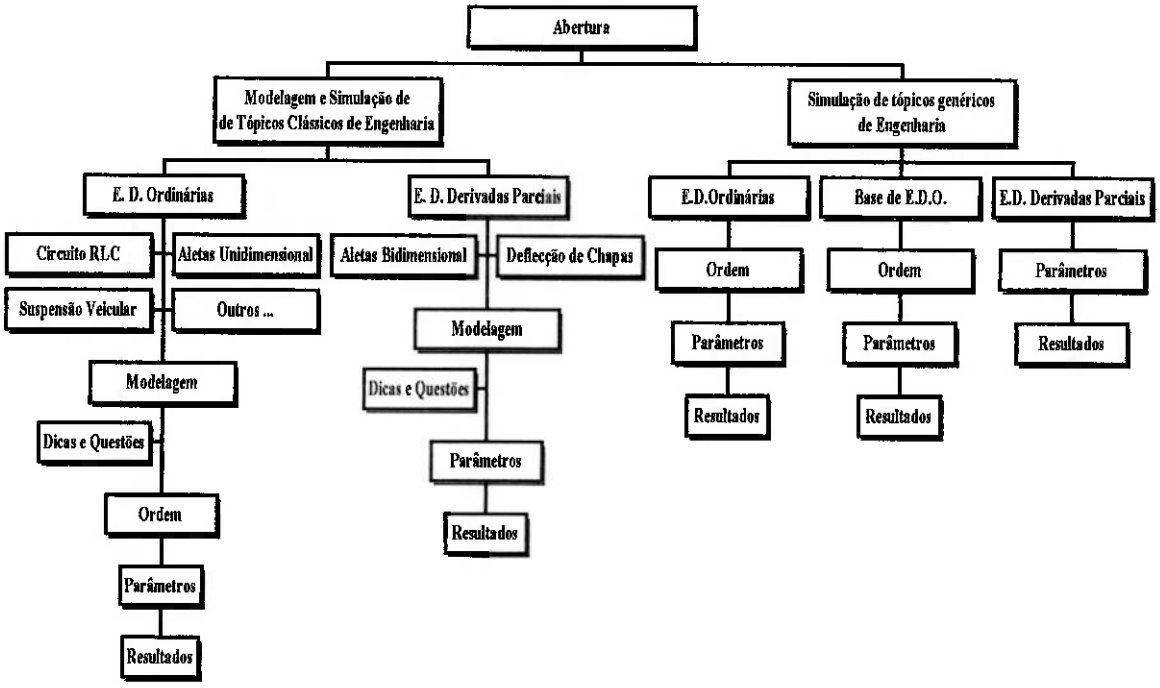
5.2 Comunicação matemática

A interface matemática entre o software e o usuário será feita através do próprio interpretador de funções matemáticas do *Turbo Pascal versão 7.0* ou do *Delphi versão 2.0*.

6. Implementação

6.1 Hierarquia do software

A divisão hierárquica subdivide o software em 7 níveis de acordo com sua classe funcional de utilização.



Organograma 1 - Divisão hierárquica do software

6.2 Manual do usuário

- Entradas: Durante a execução do software o usuário, em todas as telas de opções, se coloca em dois tipos de situação:
    - Via mouse: As entradas via mouse são entradas de opções. É através delas que o usuário define o que deseja. É nela que são definidos para qual nível hierárquica o usuário deseja passar.
    - Via teclado:
        - Linguagem matemática: Como já foi dito anteriormente, a comunação matemática com o software se dá através do próprio interpretador de texto dos softwares de desenvolvimento.
        - Texto: Durante a execução do sistema tutorial de modelagem diversas respostas devem ser dadas na forma de texto. Para uma mesma pergunta um conjunto grande de respostas é aceita, ou seja, diferentes sintaxes para uma mesma resposta podem ser dadas no tutorial de modelagem.

7. Demonstração da funcionalidade do software

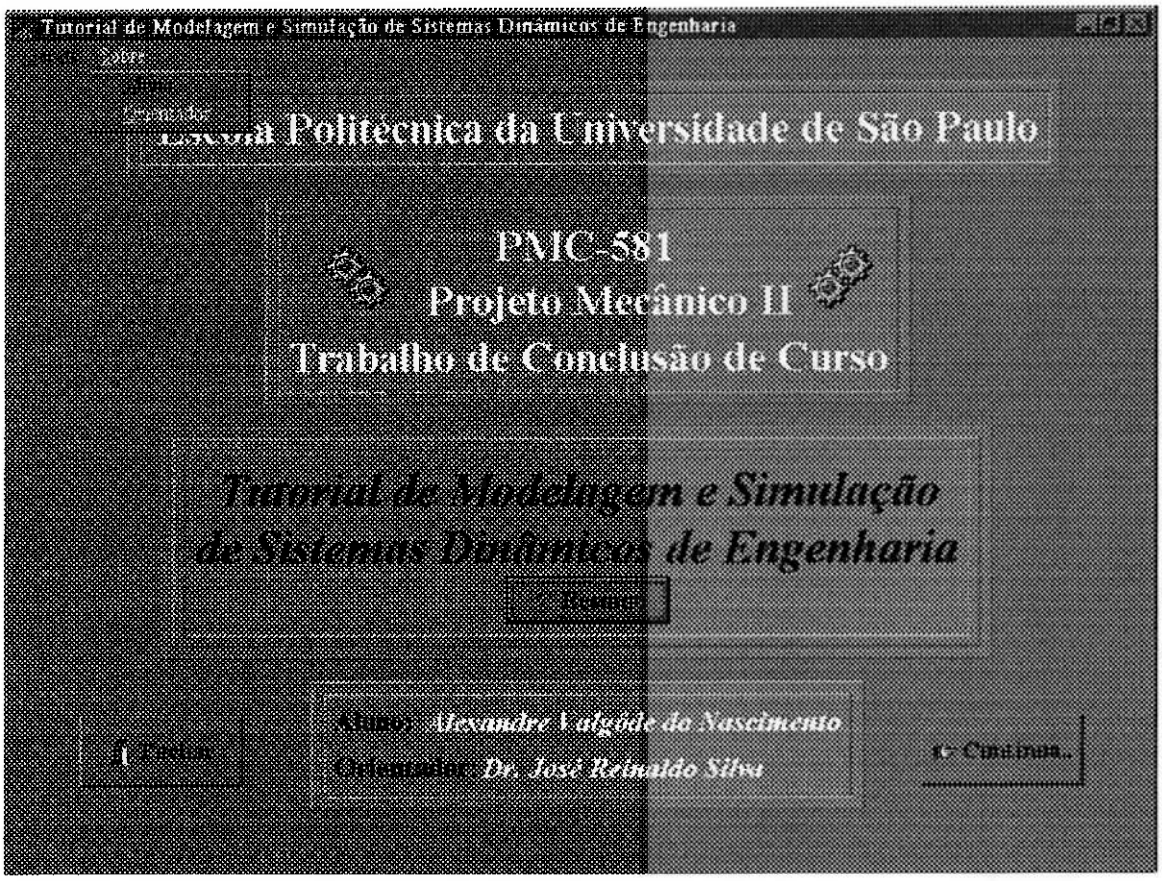
O objetivo desta seção é realizar uma simulação do software para que possa ficar explícito o seu funcionamento. A simulação é mostrada tela a tela de modo que se possa observar todas as funcionalidades e opções do software.

Ao seu executar o programa sua primeira tela é a de abertura. Através dela é possível conhecer características gerais do projeto como:

⇨ Capa com o nome da escola, nome do aluno, nome do orientador, disciplina, ...

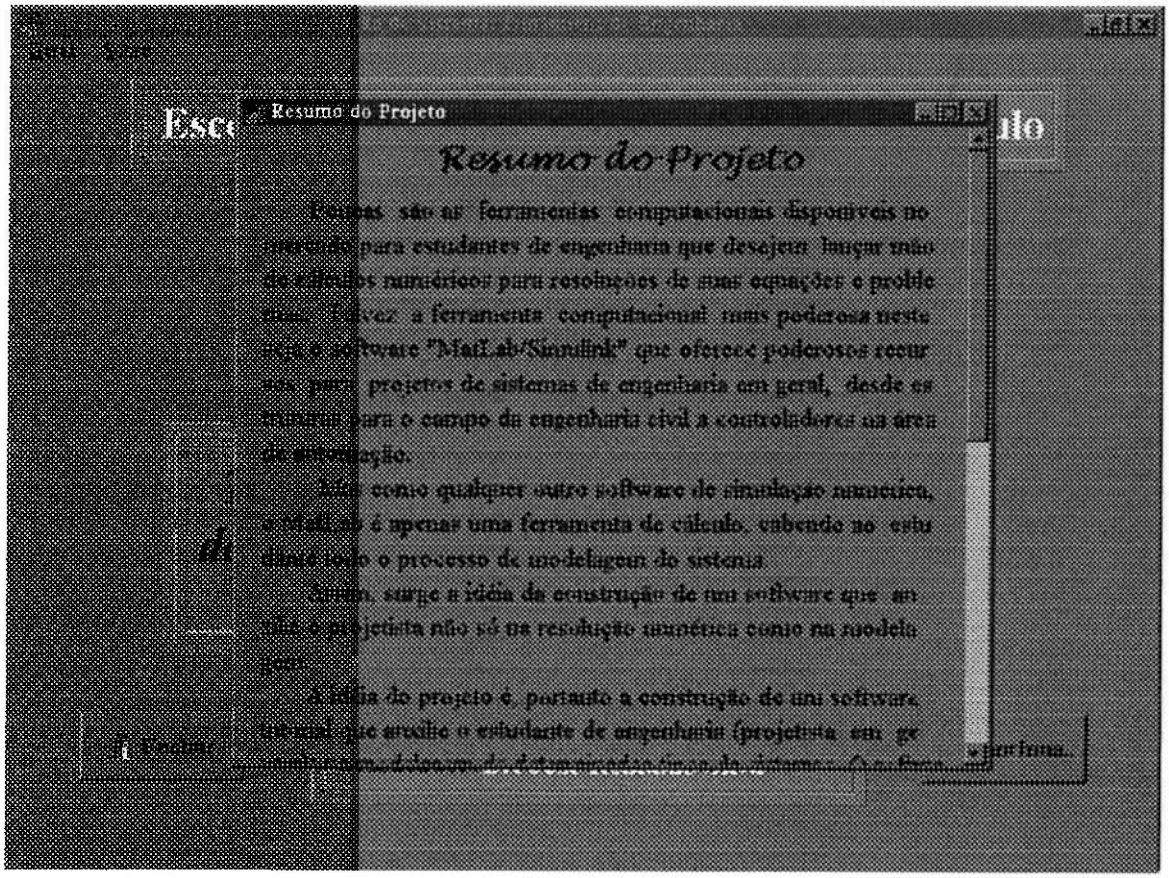
⇨ Um breve resumo sobre o projeto.

⇨ Um breve resumo sobre o orientador e sobre o aluno.



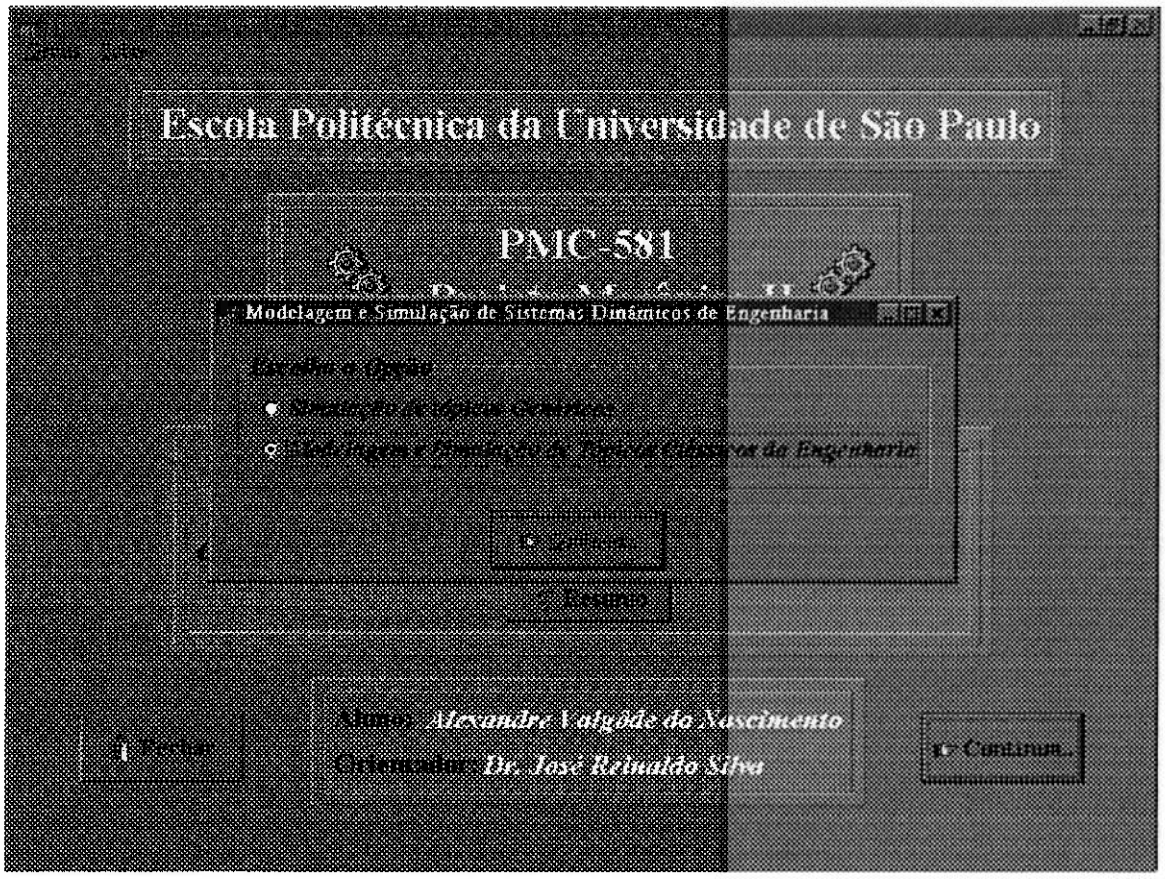
Tela 1 - Tela de abertura do software

Na primeira tela, como já foi dito é possível acessar um breve resumo sobre o projeto. A tela abaixo mostra a tela de resumo sendo acessada.



Tela 2 - Resumo do projeto

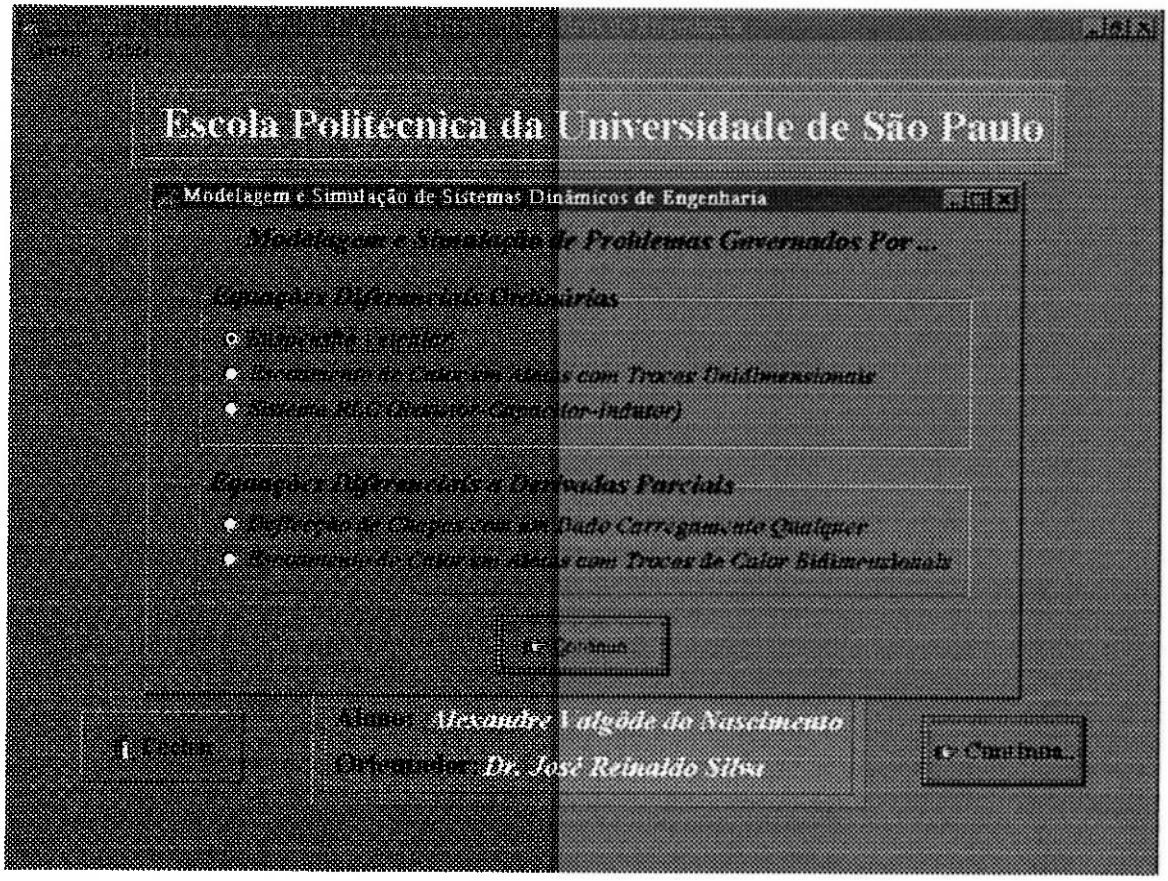
Ao clicar em "continua" na tela de abertura, o usuário encontrará uma tela de opção entre "simulação de tópicos genéricos" ou "modelagem seguido de simulação de tópicos específicos de engenharia". Desta maneira, optando-se pela simulação genérica, o aluno já deve ter o modelo a ser simulado. Caso opte por modelagem seguido de simulação o sistema auxiliará o usuário na modelagem e em seguida este modelo poderá ser simulado pelo sistema com os parâmetros definidos pelo usuário. A tela abaixo mostra este menu de opções disponível.



Tela 3 - Primeiro menu de opções do software

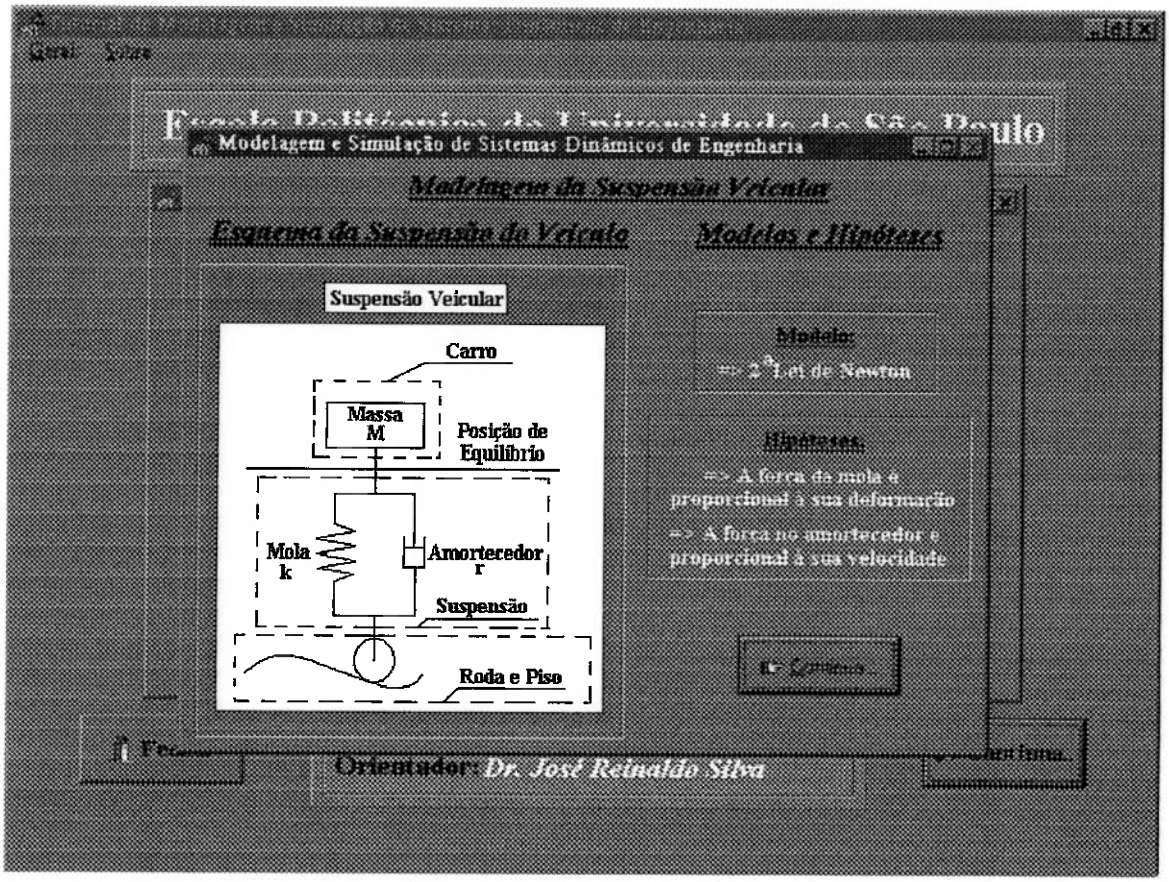
Escolha: Modelagem seguida de simulação

Primeiramente, a nível de demonstração do software, escolhe-se a opção modelagem seguido de simulação. Desta maneira em segundo menu de opções aparece ao usuário. Este segundo menu trata de uma biblioteca de modelos pré-definidos tanto no campo de equações diferenciais ordináriais quanto a derivadas parciais.



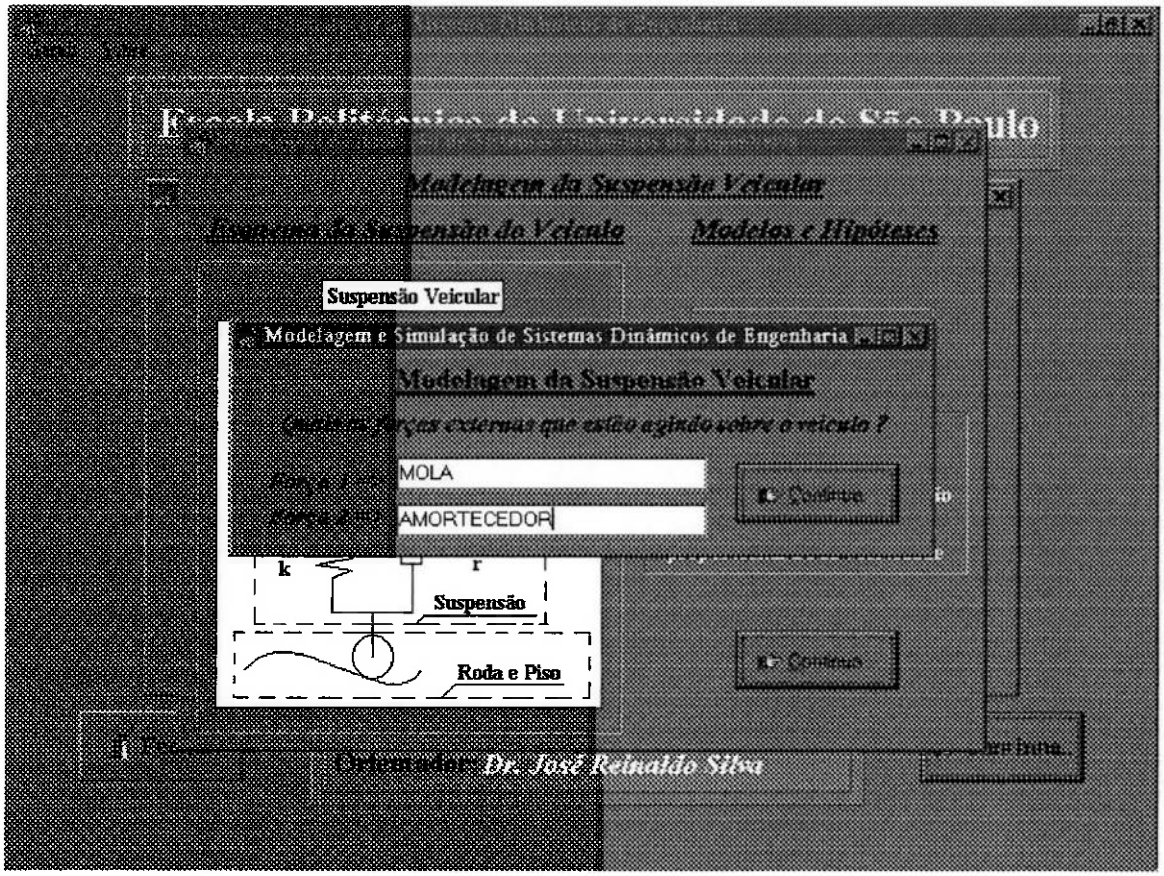
Tela 4 - Menu da biblioteca de modelos matemáticos

Para efeito de demonstração foi escolhido o tópico modelagem e simulação da suspensão veicular. A tela abaixo mostra a prmeira tela da modelagem da suspensão. Através dela o usuário pode ver o modelo gráfico e o fisico, além de algumas hipóteses a serem consideradas para efetuar a modelagem.

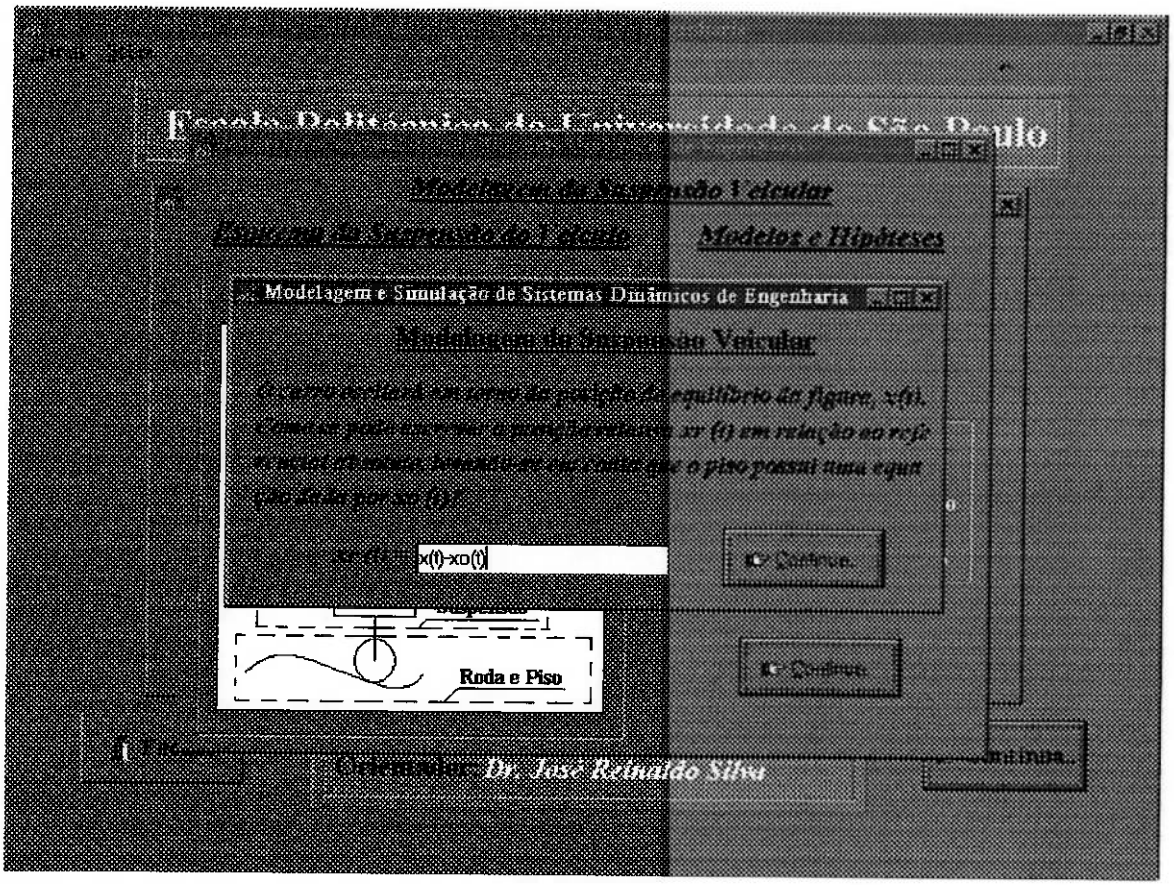


Tela 5 - Primeira tela da modelagem da suspensão veicular

A segunda tela da modelagem em diante mostra perguntas feitas ao usuário que, ao responde-las, disserne um modelo matemático (dinâmico) que será representado por equações diferenciais ordinárias.

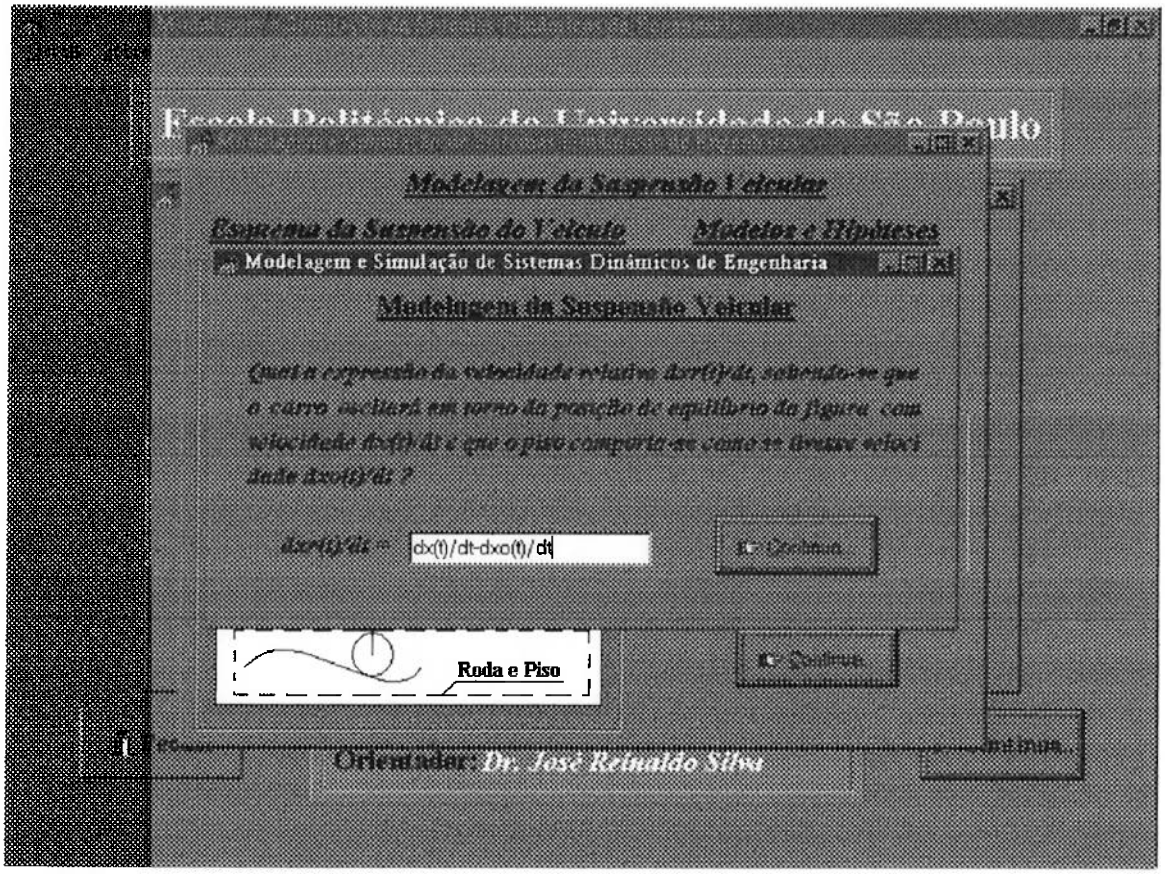


Tela 6 - Segunda tela da modelagem da suspensão veicular

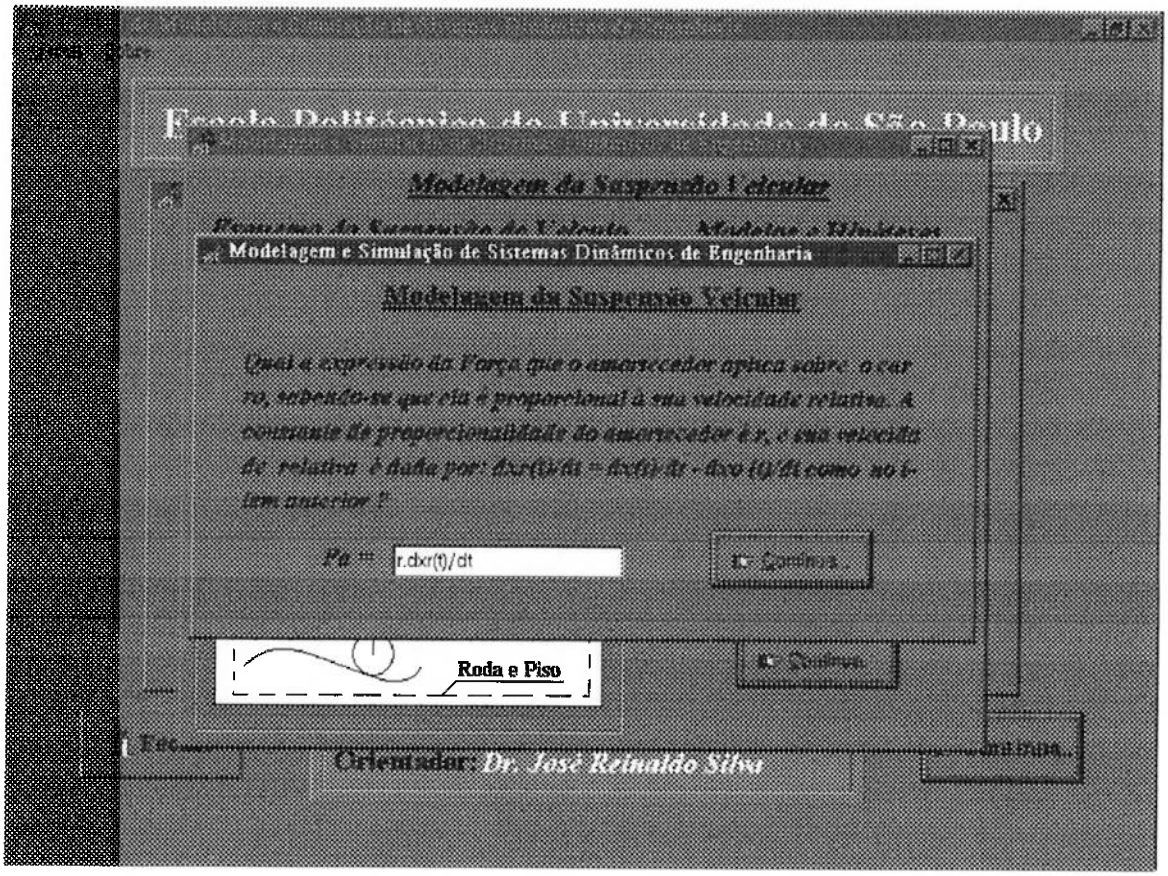


Tela 7 - Terceira tela da modelagem da suspensão veicular

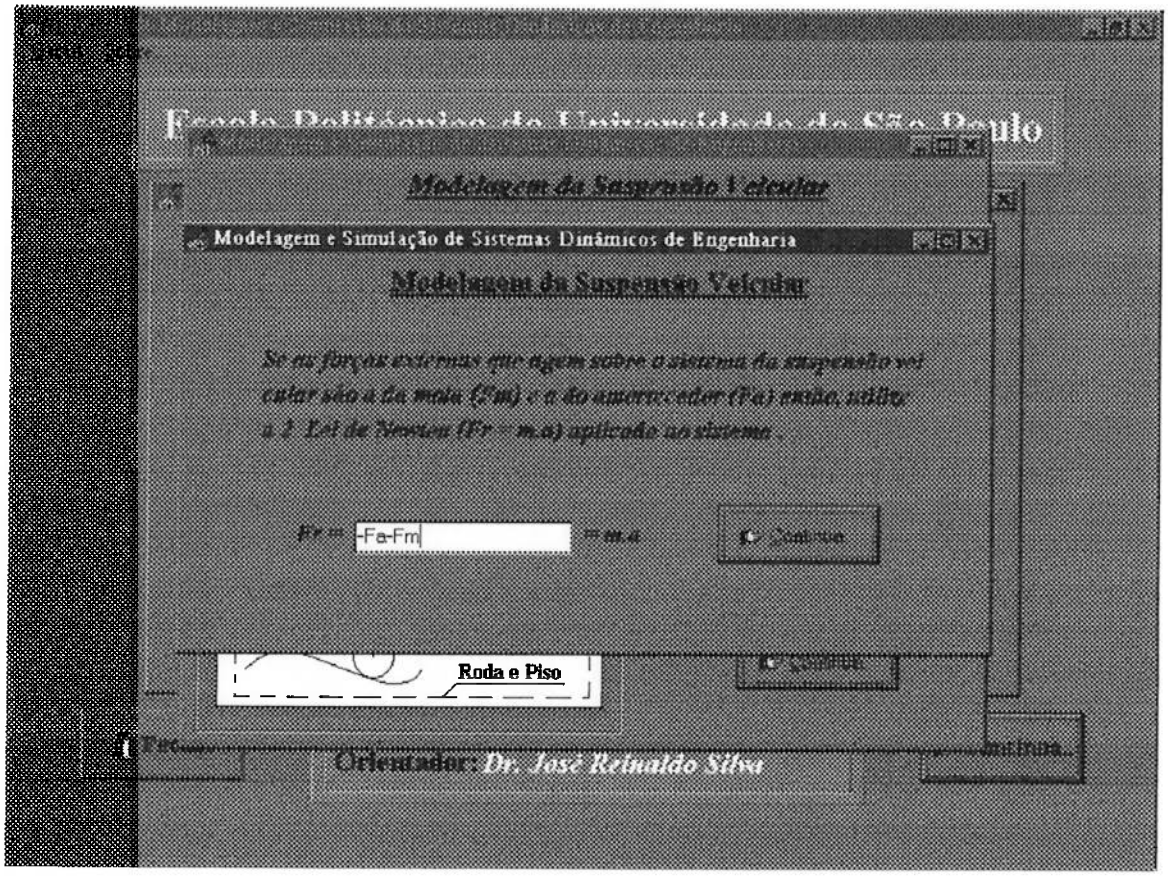
Tela 8 - Quarta tela da modelagem da suspensão veicular



Tela 9 - Quinta tela da modelagem da suspensão veicular

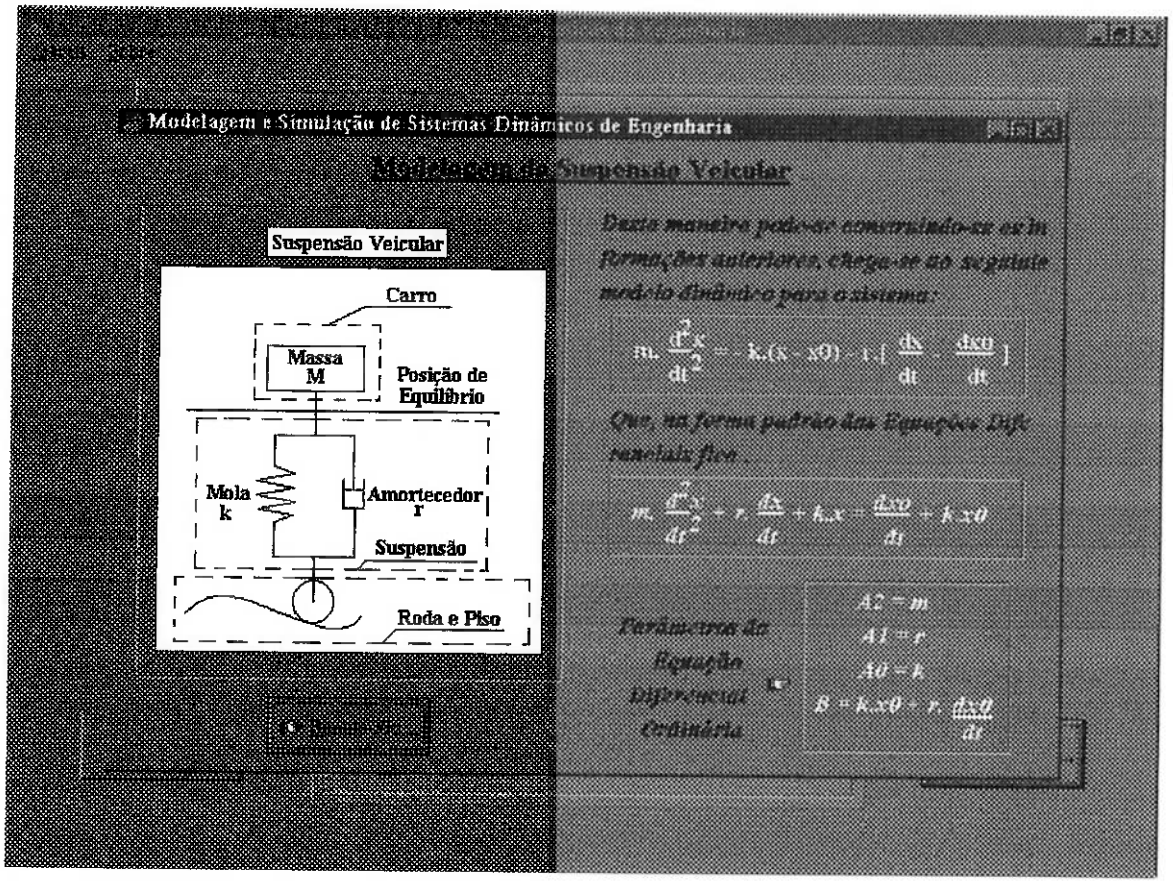


Tela 10 - Sexta tela da modelagem da suspensão veicular



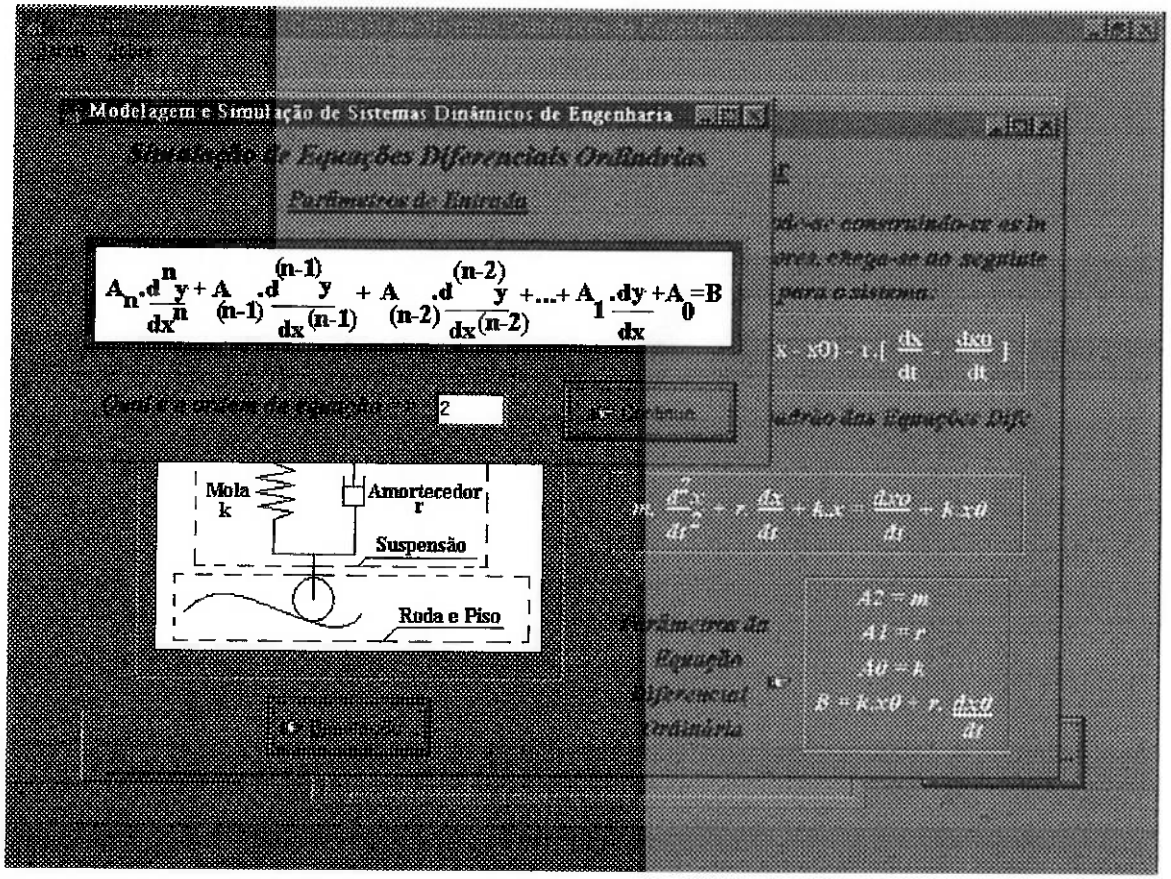
Tela 11 - Sétima tela da modelagem da suspensão veicular

Respondidas as questões acima e compondo-se as respostas, chega-se ao modelo do sistema dinâmico, que pode ser simulado em seguida.

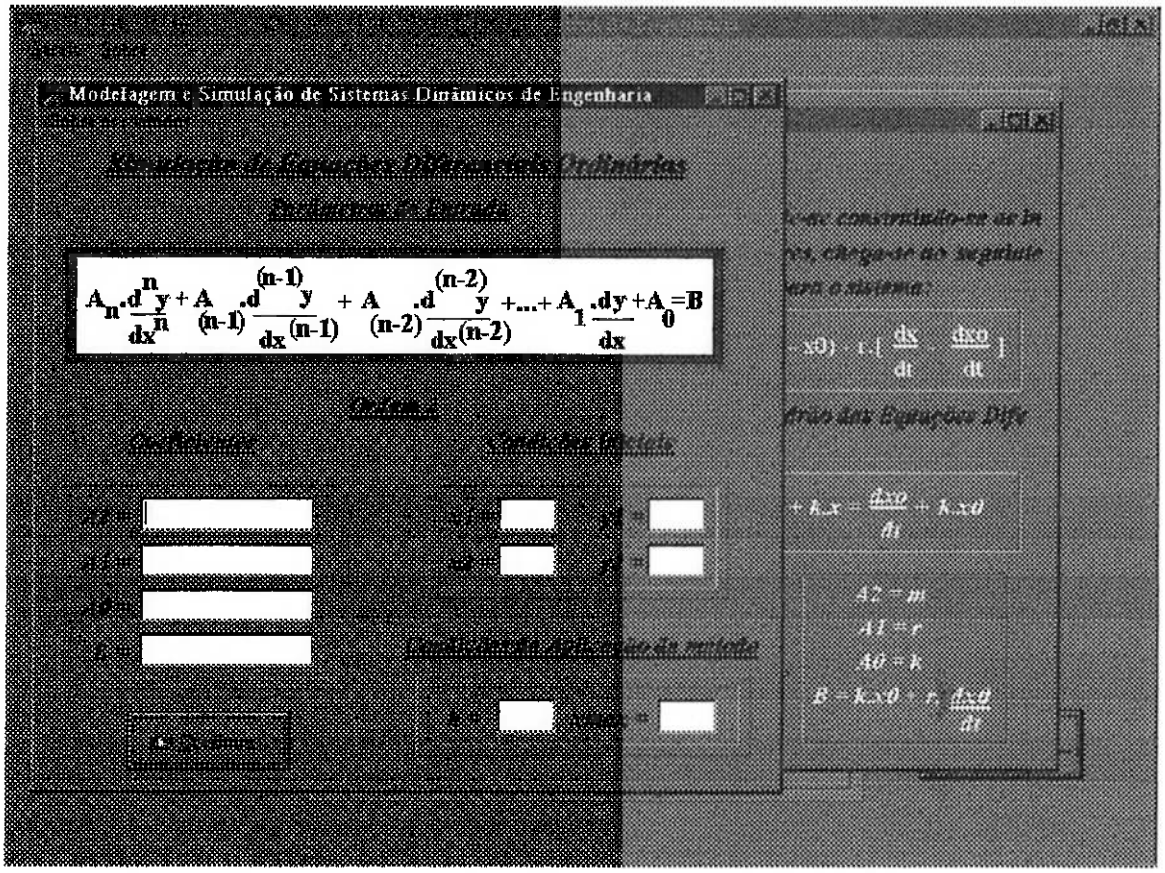


Tela 12 - Oitava tela da modelagem da suspensão veicular

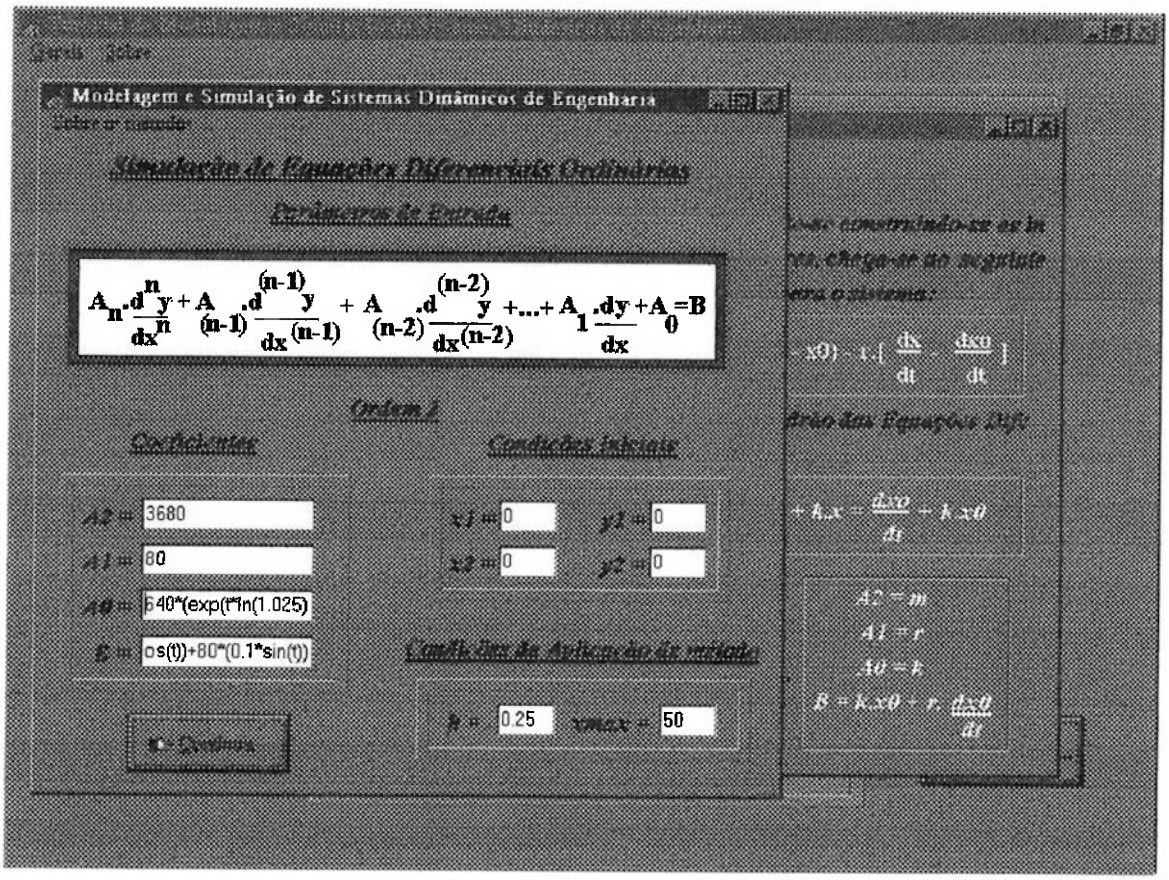
Optando-se pela simulação, o usuário entra a ordem da equação diferencial seguida de seus parâmetros. A figura abaixo mostra a tela de entrada da ordem da equação.



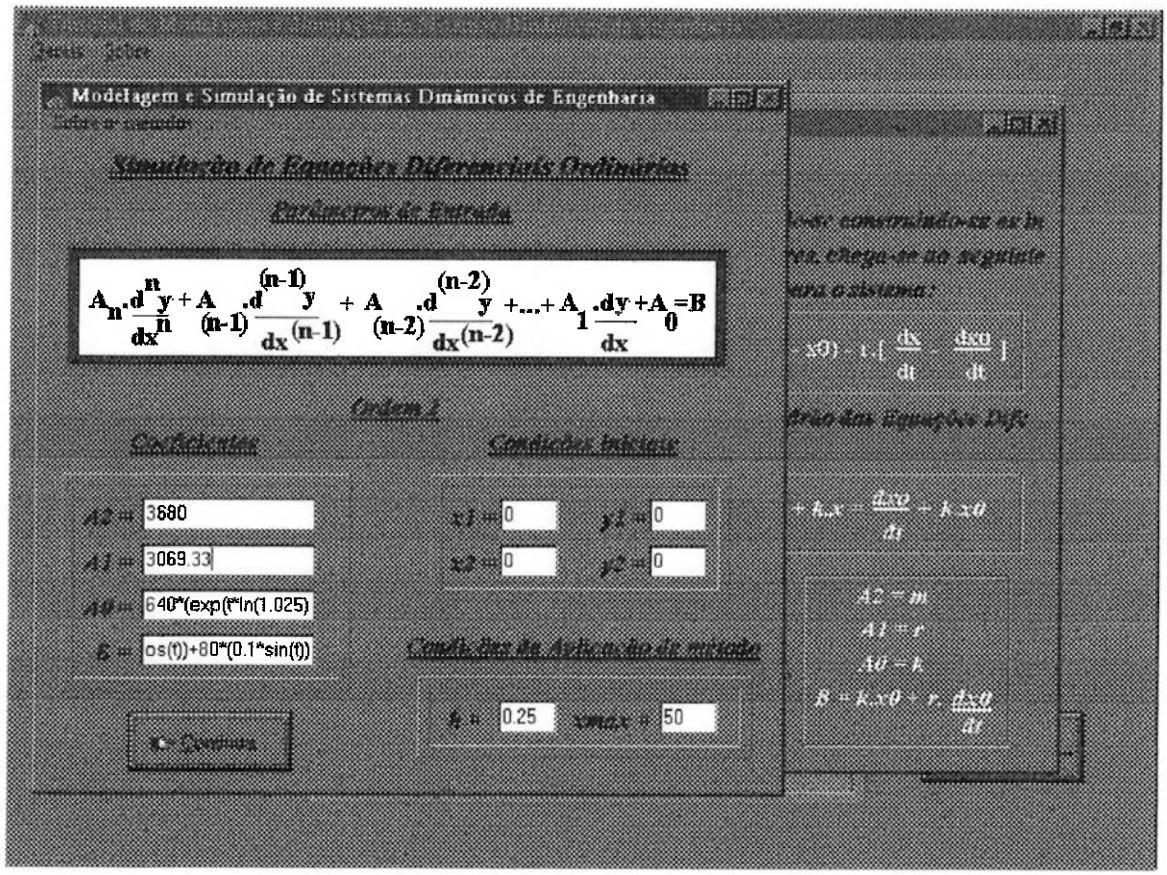
Tela 13 - Entrada da oredem da equação diferencial



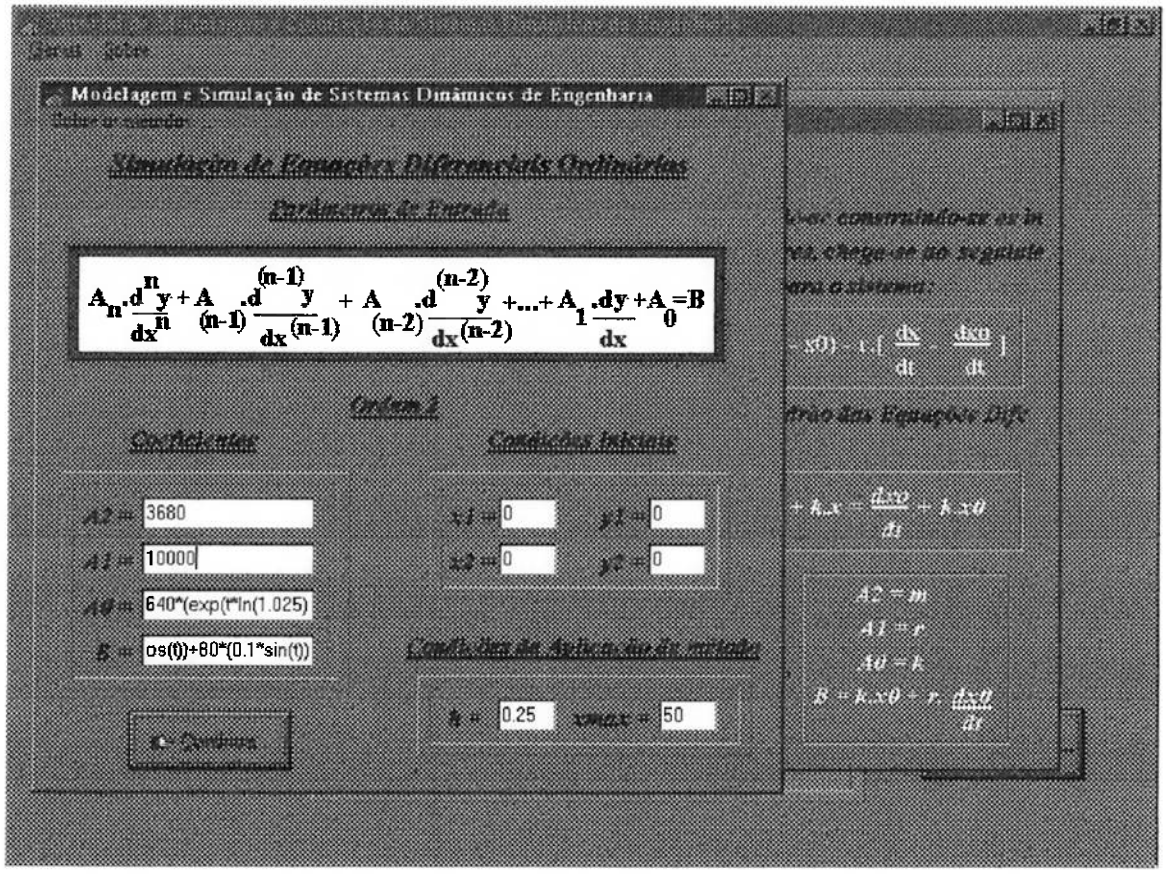
Tela 14 - Entrada de parâmetros da equação diferencial



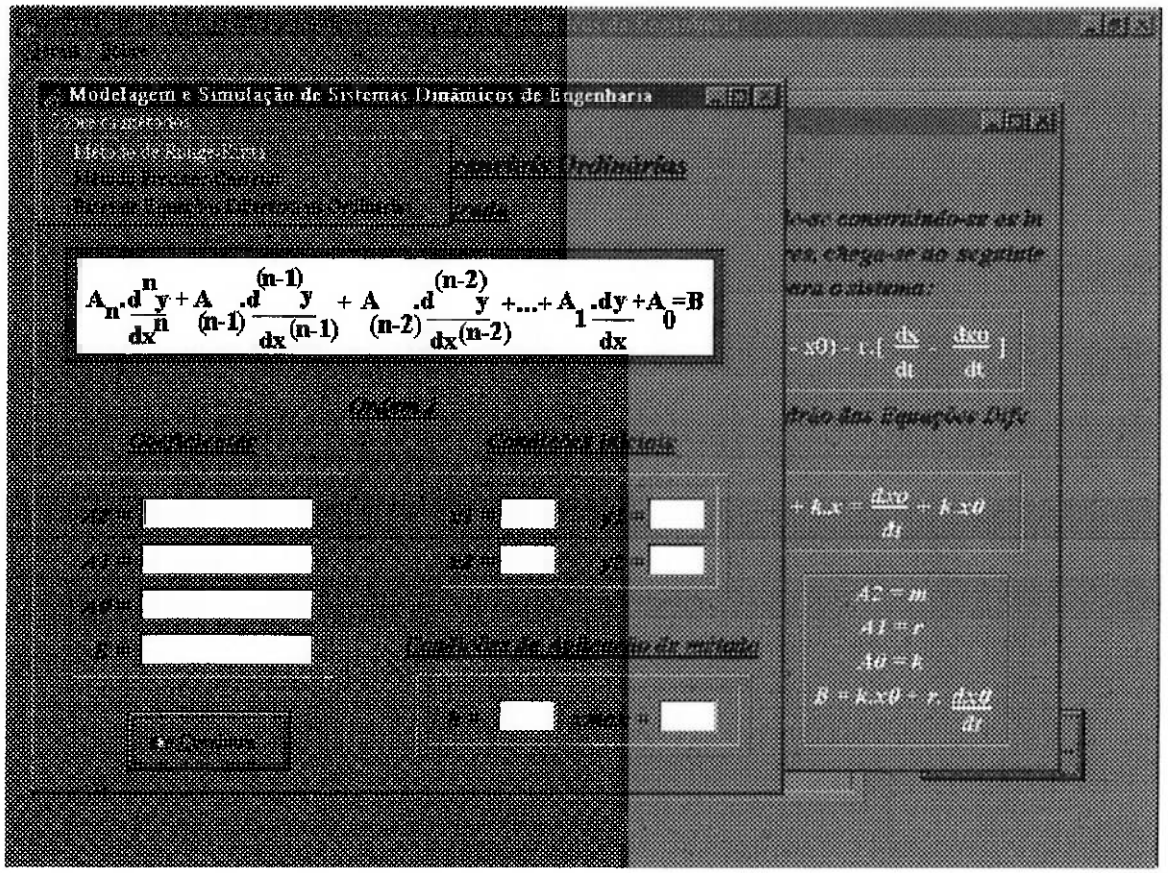
Tela 15 - Entrada de dados para simulação 1



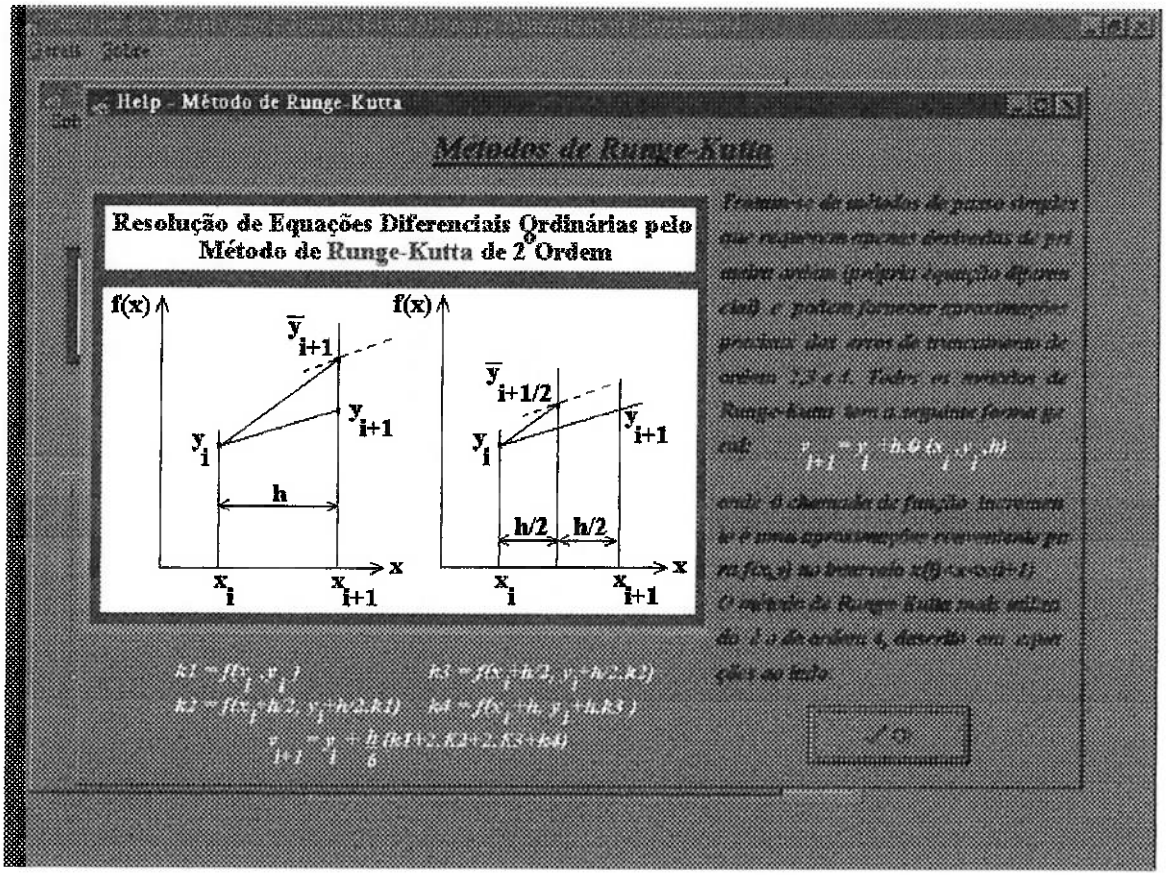
Tela 16 - Entrada de dados para simulação 2



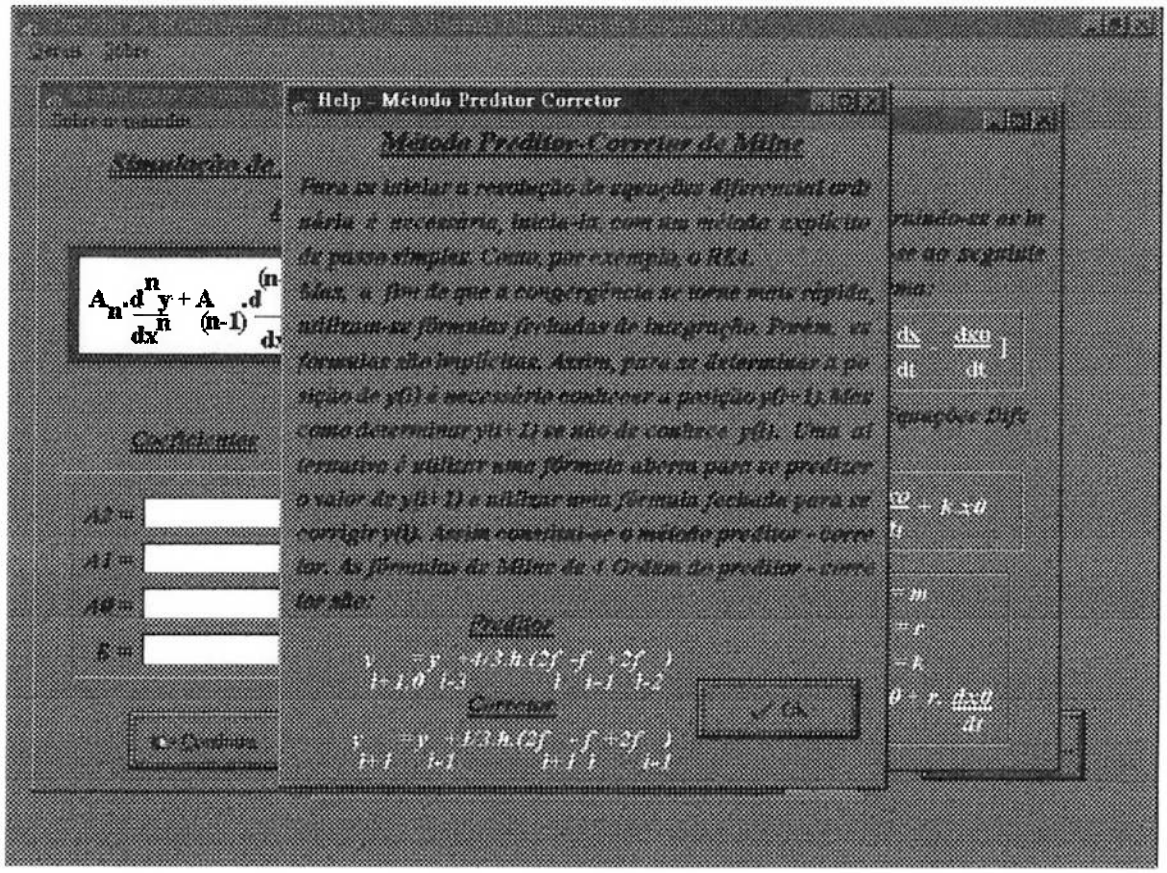
Tela 17 - Entrada de dados para simulação 3



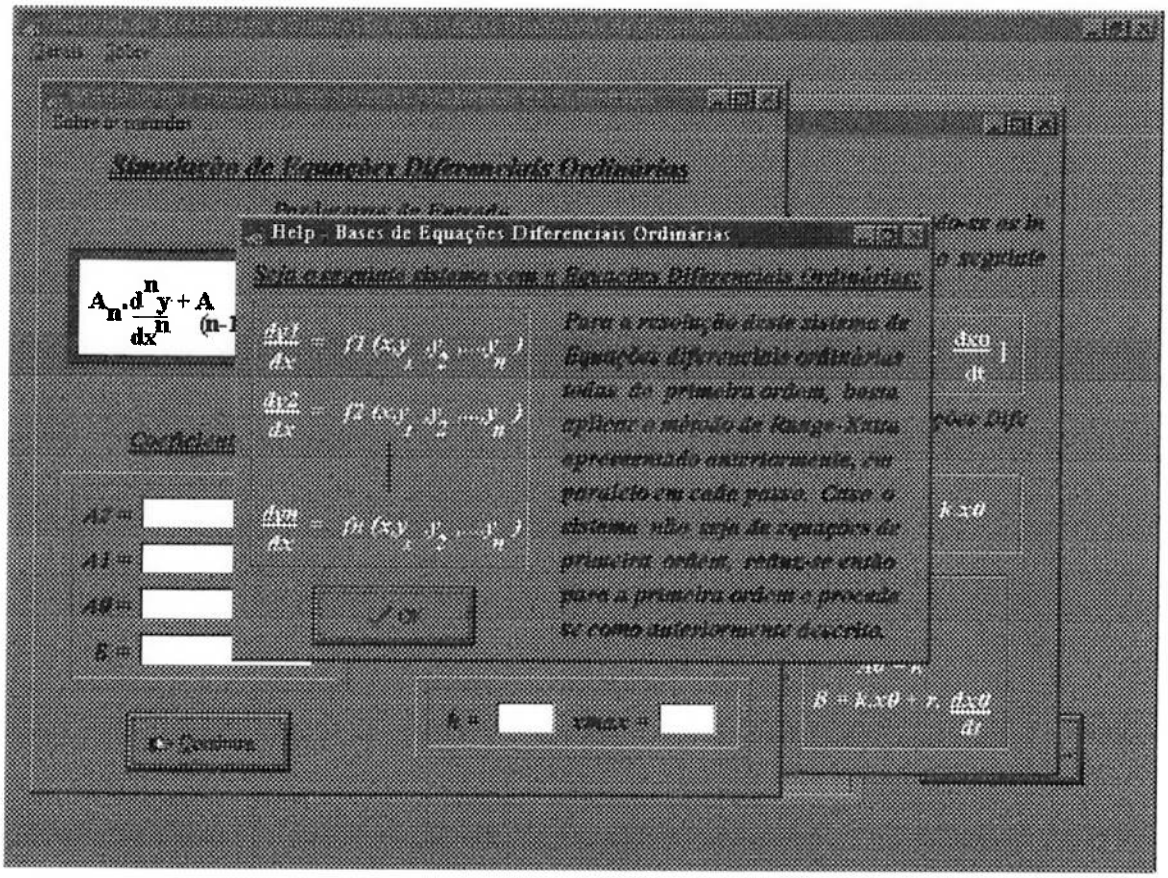
Tela 18 - Acesso aos "help's" didáticos



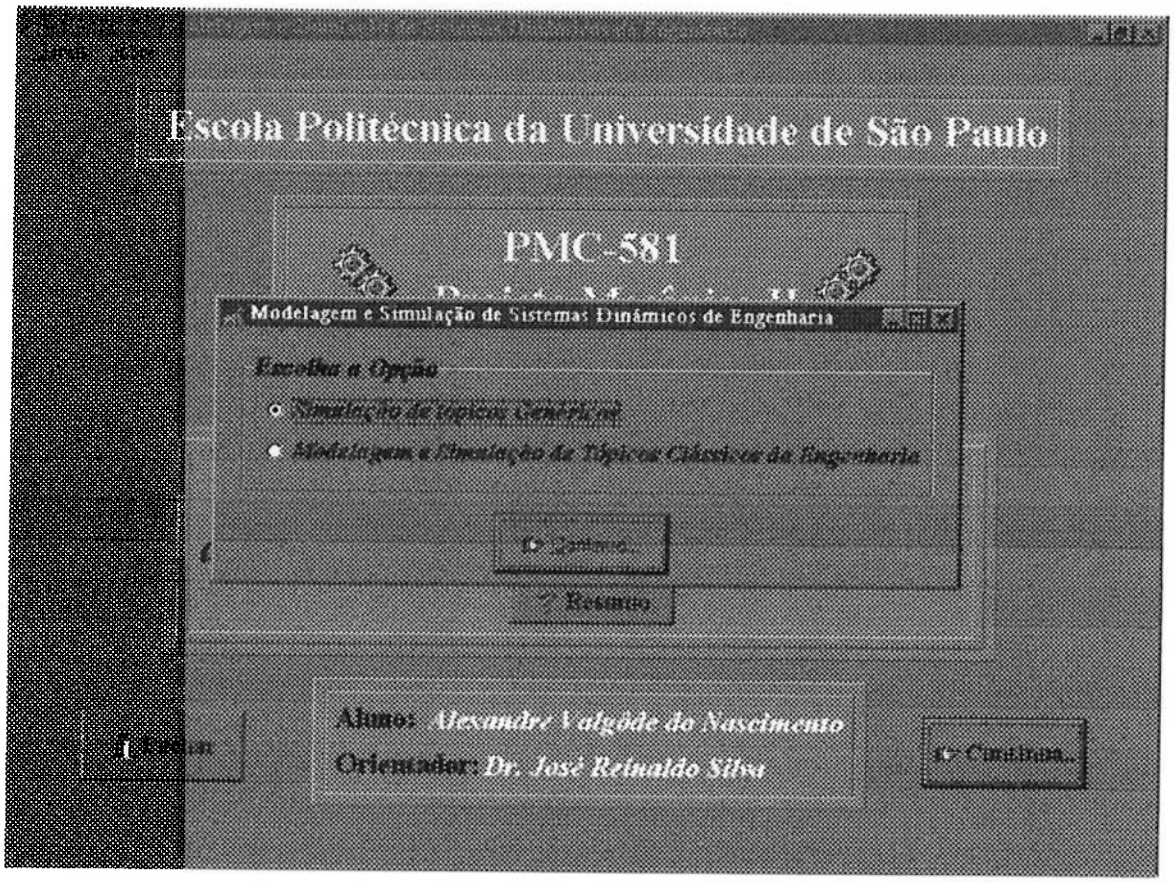
Tela 19 - Help didático - "Método de Runge-Kutta"



Tela 20 - Help didático - "Método Preditor-Corretor"

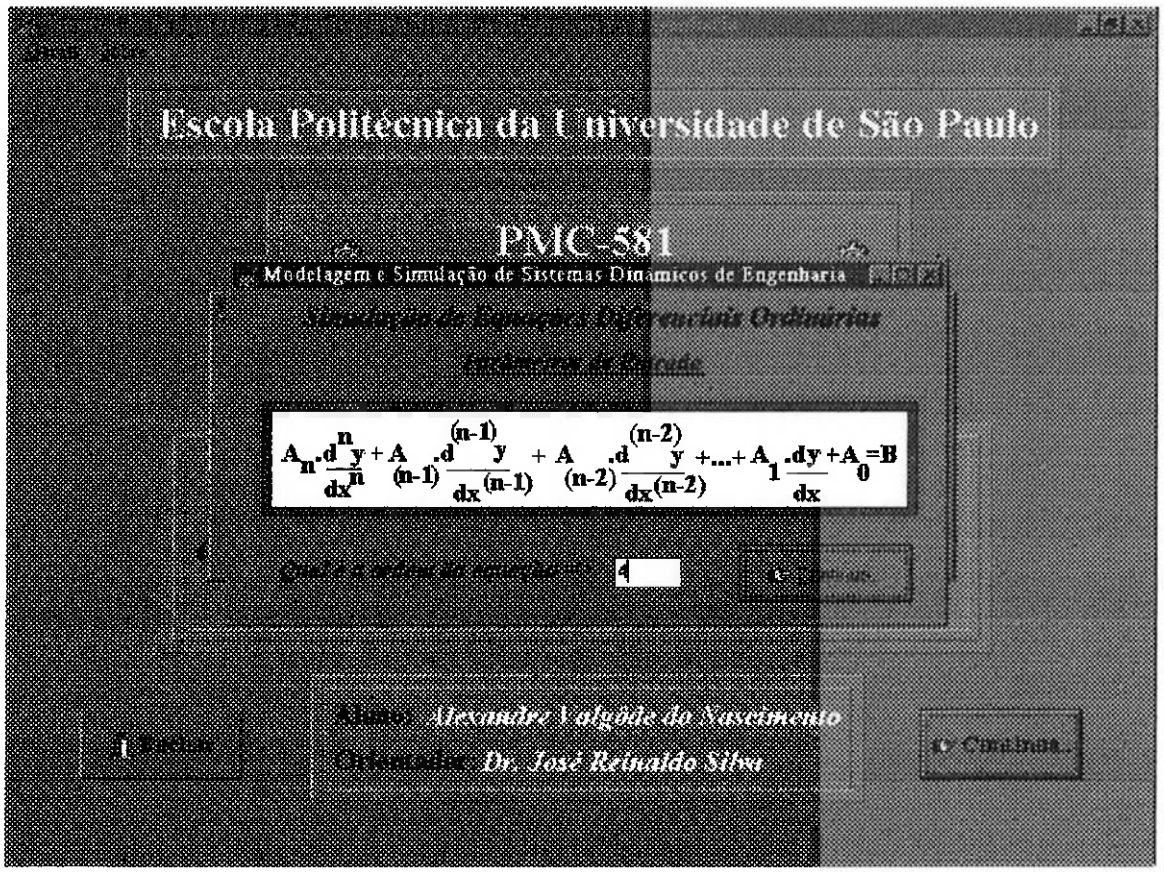


Tela 21 - Help didático - "Bases de equações diferenciais ordinárias"

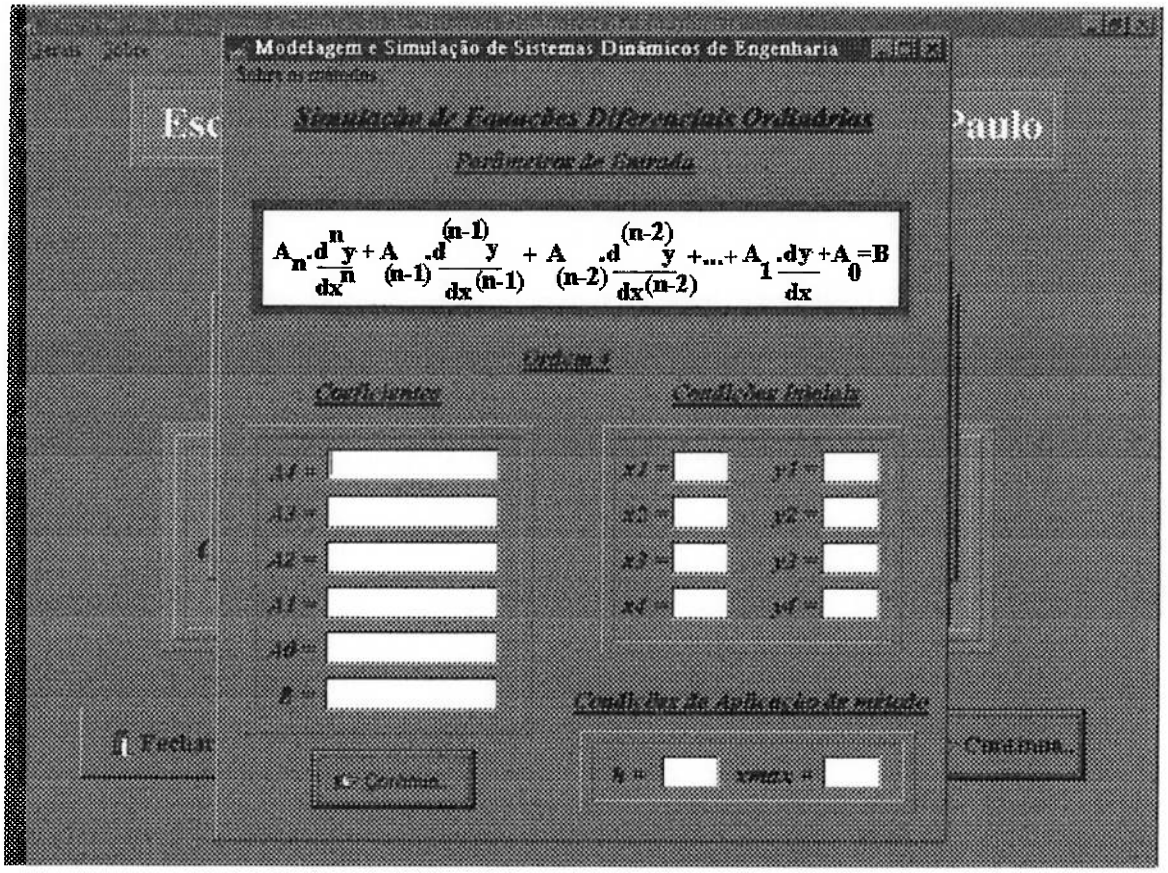


Tela 22 - Primeira tela de menu de opções

Escolha: Simulação pura



Tela 23 - Entrada da ordem da equação diferencial a ser simulada



Tela 24 - Entrada de parâmetros da equação

Tela 25 - Sair do software

8. Resultados da simulação

Na demonstração do funcionamento do software na seção 7 deste relatório foram realizadas uma modelagem (suspensão veicular) seguida de 3 simulações. Os resultados, disponibilizados pelo software em um arquivo texto podem ser visualizados graficamente:

Dados da Simulação:

- Veículo:
  - $m = 3680$
  - $r = 80$
  - $k = 640.(1{,}025)^t$
- Parâmetros da pista:
  - $xo = 0.1\ (1-\cos t)$
  - $dxo/dt = 0.1 \sin t$
- Método:
  - $h = 0{,}25$
  - $tmax = 50$
- Condições iniciais nulas.

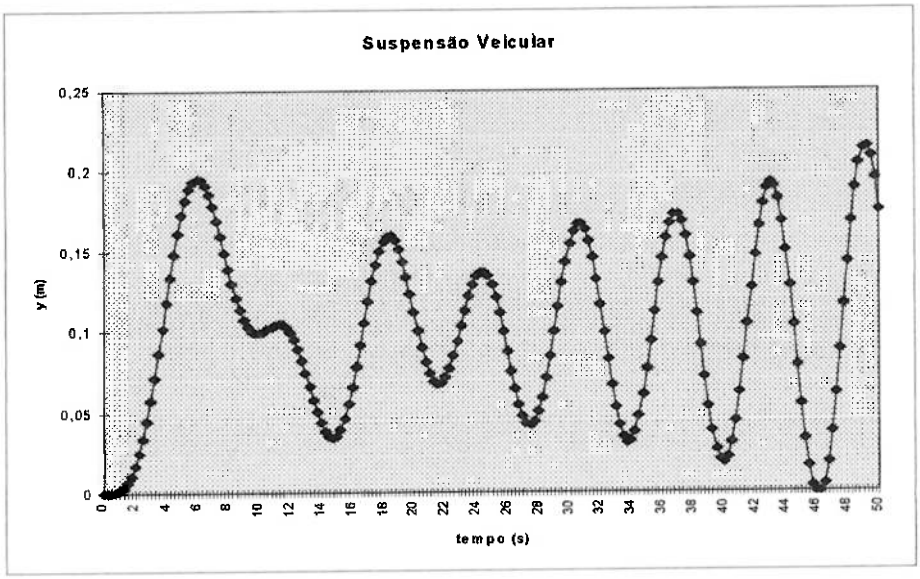


Gráfico 1 - Simulação de um amortecedor em movimento subcrítico

Na simulação acima tem-se o amortecedor efetuando um movimento subcrítico. No caso acima, considere-se um amortecedor em uma pista cossenoidal deslocada e uma mola de constante variável em função do uso. A equação acima representa o desgaste da mola. No caso acima, 1 ano é simulado em 20s.

Dados da Simulação:

- Veículo:
  - m = 3680
  - r = 3069,33
  - $k = 640.(1,025)^t$
- Parâmetros da pista:
  - xo = 0.1 (1-cos t)
  - dxo/dt = 0.1 sin t
- Método:
  - h = 0,25
  - tmax = 50
- Condições iniciais nulas.

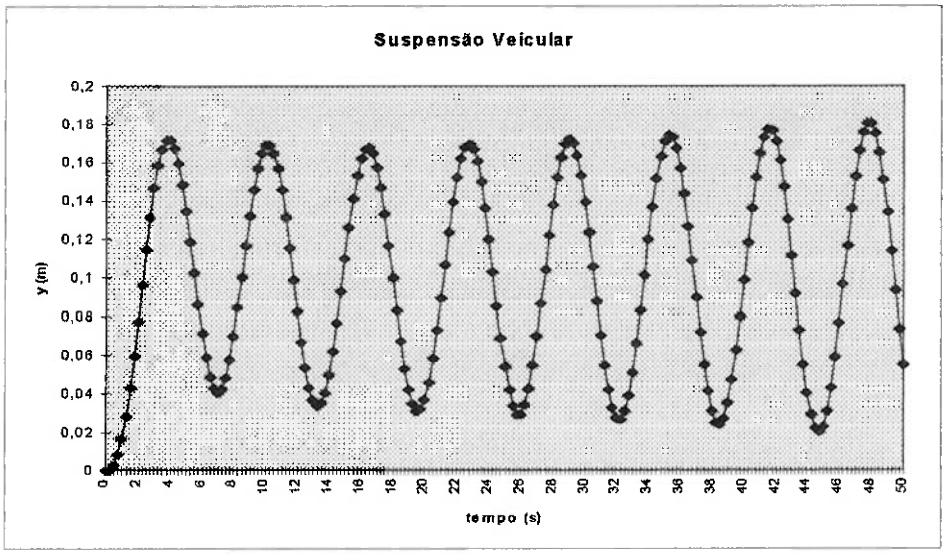


Gráfico 2 - Simulação de um amortecedor em movimento crítico

Na simulação acima tem-se o amortecedor efetuando um movimento crítico. No caso acima, considere-se um amortecedor em uma pista cossenoidal deslocada e uma mola de constante variável em função do uso. A equação acima representa o desgaste da mola. No caso acima, 1 ano é simulado em 20s.

Dados da Simulação:

- Veículo:
  - m = 3680
  - r = 10000
  - $k = 640.(1{,}025)^t$
- Parâmetros da pista:
  - xo = 0.1 (1-cos t)
  - dxo/dt = 0.1 sin t
- Método:
  - h = 0,25
  - tmax = 50
- Condições iniciais nulas.

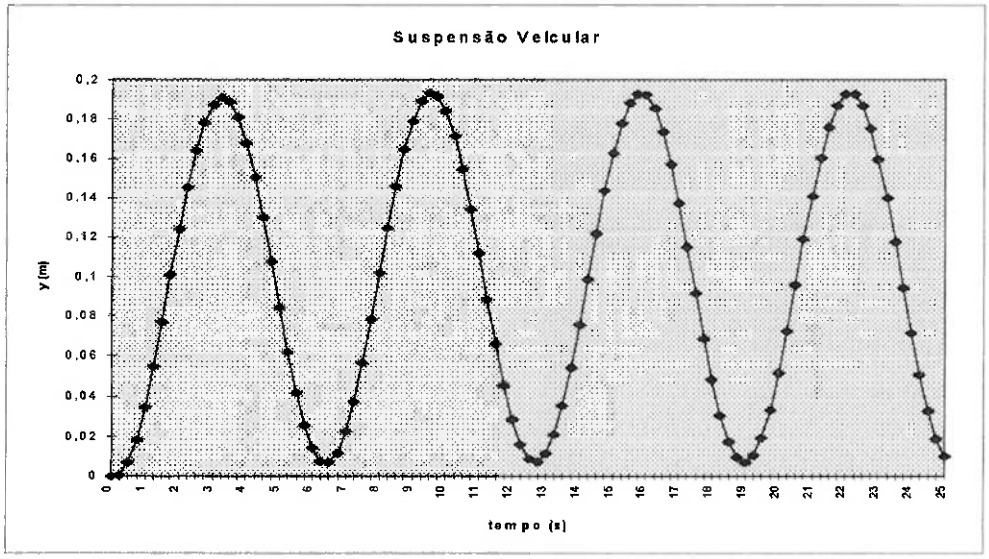


Gráfico 3 - Simulação de um amortecedor em movimento hipercrítico

Na simulação acima tem-se o amortecedor efetuando um movimento hipercrítico. No caso acima, considere-se um amortecedor em uma pista cossenoidal deslocada e uma mola de constante variável em função do uso. A equação acima representa o desgaste da mola. No caso acima, 1 ano é simulado em 20s.

9. Conclusões

O software atende satisfatoriamente aos quesitos de modelagem de sistemas de engenharia regidos por sistemas contínuos. Para que o trabalho se completa-se em termos de modelagem, poderia-se introduzir sistemas regidos por eventos discretos. Desse modo o usuário poderia dissernir quanto aos diferentes aspectos na modelagem dos 2 tipos de sistemas, bem como o ferramental competente a cada um deles em termos de posterior simulação.

Em termos de simulação, foram efetuadas simulações para diferentes parâmetros nos diferentes modelos da biblioteca de modelagem e para todos os casos ocorreu uma convergência rápida e atendendo aos resultados esperados.

10. Referências Bibliográficas

- Carnahan, Brice and Luther, H.A. and Wilkes, J.O., *Applied Numerical Methods*, John Wiley and Songs, 1969.
- Duntemann, J. and Mischel J., *Delphi - Kit do explorador*. Coriolis Group Books, 1996.
- Dumas, A., *Psicologia na interface homem máquina*. Mc Graw Hill, 1992.
- Dumas, A., *Design de softwares para Windows*. Mc Graw Hill, 1992.
- Holman, J.P., *Transferência de calor*. Mc Graw Hill, 1993.

Anexos

- Help do conteúdo didático

Abaixo encontram-se as deduções dos teoremas relativos aos métodos numéricos a serem empregados nas simulações numéricas das equações diferenciais, bem as explicações didáticas relativas a cada tópico. Os tópicos abaixo constarão dos help's didáticos dos tópicos de simulação numérica. Conforme o tópico 5.1 deste relatório, relativo aos níveis hierárquicos de comunicação do usuário com o programa, mostra onde se encaixam as deduções e explicações abaixo.

Equações Diferenciais Ordinárias

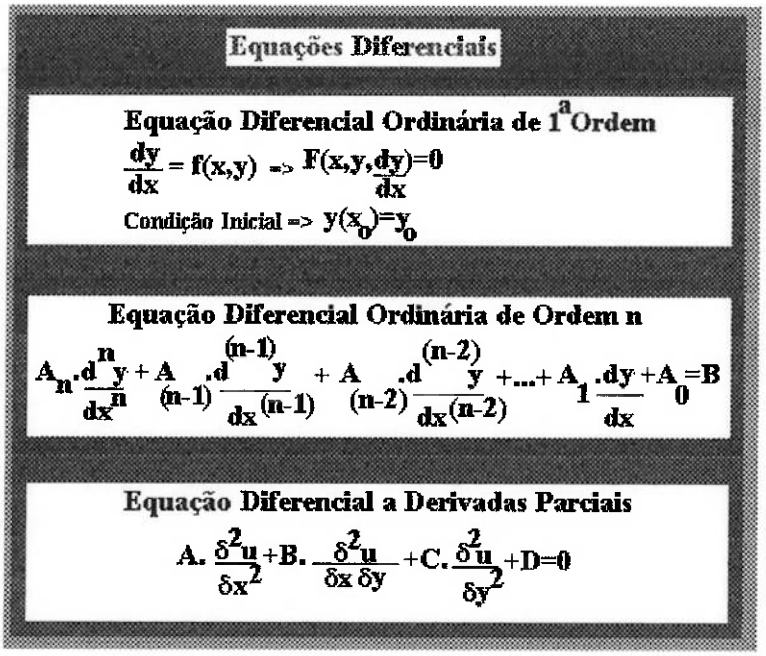
Equações diferenciais Ordinárias são aquelas que podem ser escritas sob a forma: $F\left(x, y, \frac{dy}{dx}\right) = 0$ ou $F\left(x, y, \frac{dy}{dx}\right) = f(x, y)$. Os principais métodos de resolução destas equações são:

⇨ O "Método de Taylor"

⇨ O "Método de Euler"

⇨ O "Método de Runge-Kutta"

⇨ O "Método Preditor-Corretor de Milne".



Equações Diferenciais a Derivadas Parciais

Equações diferenciais a Derivadas Parciais são aquelas que podem ser escritas sob a forma: $A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D = 0$. Os principais métodos de resolução destas equações são:

- ⇨ O Método Explícito
- ⇨ O Método Implícito

Equações Diferenciais Ordinárias

"Método de Taylor"

"Trata-se de um método exclusivamente dedicado a resolução de equações diferenciais ordinárias de primeira ordem, que podem ser definidas como abaixo:

$$F\left(x, y, \frac{dy}{dx}\right) = 0 \quad \text{ou} \quad \frac{dy}{dx} = f(x, y)$$

Para um sistema que possa ser modelado como acima, suponhamos a seguinte condição inicial $y(x_0) = y_0$.

A solução da equação diferencial referida deve satisfazer a equação e a sua condição inicial. Em geral é impossível determinar a forma analítica de y(x), ao invés disto determina-se numericamente sua solução. Se se deseja saber a solução da equação em um intervalo fechado [a,b], então divide-se este intervalo em *n partes igualmente espaçados* denominados subintervalos de resolução da equação. O valor da função y(x) pode ser aproximada, portanto, em (n+1) pontos igualmente espaçados $(x_0, x_1, x_2, \ldots, x_n)$. Ao incremento de uma posição para outra denomina-se *passo.* O passo pode ser determinado como abaixo:

$$h = \left(\frac{b-a}{n}\right) \quad \text{logo} \Rightarrow \; x_i = x_0 + i.h \;\; \text{com i=0,1,2..,n.}$$

Expandindo-se a série de Taylor de y(x) da posição inicial conhecida para uma posição imediatamente posterior (passo h), chega-se a:

$$y(x_0 + h) = y_0 + h.f(x_0, y_0) + \frac{h^2}{2!}.f''(x_0, y_0) + \frac{h^3}{3!}.f'''(x_0, y_0) + \ldots$$

Sendo que as derivadas podem ser calculadas pela regra da cadeia:

$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dx}$$

A solução da equação acima é dada portanto por uma tabela com (n+1) valores discretos de x. Estes valores são uma aproximação da função em cada ponto."

"Método de Euler"

Trata-se de um método exclusivamente dedicado a resolução de equações diferenciais ordinárias de primeira ordem, que podem ser definidas como abaixo:

$$F\left(x, y, \frac{dy}{dx}\right) = 0 \quad \text{ou} \quad \frac{dy}{dx} = f(x, y)$$

Para um sistema que possa ser modelado como acima, suponhamos a seguinte condição inicial $y(x_0) = y_0$.

Trata-se de um método de passo único, onde o conhecimento da posição posterior depende exclusivamente da última posição calculada. Para o cálculo da segunda posição necessita-se, portanto, da primeira posição, para o cálculo da terceira depende-se da segunda e assim por diante. A primeira posição é a condição inicial da equação diferencial ordinária como acima.

O método é um caso particular da aplicação do método de Taylor, para o caso em que a expansão é de primeira ordem, ou seja, o erro depende da segunda derivada da função.

Expandindo Taylor pela primeira ordem chega-se a equação abaixo:

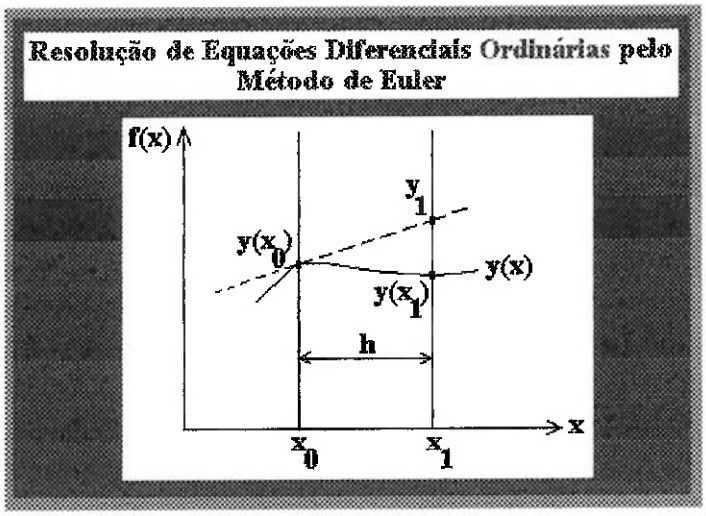
$$y_{i+1} = y_i + h.f(x_i, y_i) = y_i + h.f_i$$

O passo de resolução é definido como no Método de Taylor:

$$h = \left(\frac{b-a}{n}\right) \quad \text{logo,}$$

$$x_i = x_0 + i.h \quad \text{com i=0,1,2..,n.}$$

A figura abaixo elucida o texto acima:



"Método de Runge-Kutta"

São métodos de passo simples que requerem apenas derivadas de primeira ordem (própria equação diferencial) e podem fornecer aproximações precisas com erros de truncamento de ordem 2,3 ou 4.

Todos os Métodos de Runge-Kutta tem a seguinte forma geral:

$$y_{i+1} = y_i + h.\phi(x_i, y_i, h)$$

onde ϕ, chamado de função incremento, é uma aproximação conveniente para f(x,y) no intervalo x(i)<x<x(i+1)

Método de Runge-Kutta de 2° Ordem

Seja ϕ uma média ponderada de duas aproximações das derivadas k1 e k2 no intervalo x(i)<x<x(i+1), tal que:

$$\phi = a.k_1 + b.k_2$$

Então o algoritmo de Runge-Kutta de 2°Ordem fica:

$$y_{i+1} = y_i + h.(a.k_1 + b.k_2) \quad \text{(I)}$$

Assumindo-se:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + ph, y_i + qhf(x_i, y_i))$$

e expandindo k2 segundo Taylor:

$$f(x_i + ph, y_i + qhf(x_i, y_i)) = f(x_i, y_i) + phf_x(x_i, y_i) + qhf_y(x_i, y_i) + O(h^2) \quad \text{(II)}$$

Substituindo-se (II) em (I), chega-se a:

$$y_{i+1} = y_i + h.[af(x_i, y_i) + bf(x_i, y_i)] + h^2[bpf_x(x_i, y_i) + bqf(x_i, y_i)f_y(x_i, y_i)] + O(h^3)$$

(III)

Expandindo-se y(x) em x(i) usando a série de Taylor:

$$y(x_i + h) = y(x_i) + hf(x_i, y(x_i)) + \frac{h^2}{2!} \cdot f'(x_i, y(x_i)) + \frac{h^3}{3!} \cdot f''(\xi, y(\xi)) \quad \text{(IV)}$$

Calculando-se a derivada pela regra da cadeia:

$$f'(x_i, y(x_i)) = f_x(x_i, y(x_i)) + f_y(x_i, y(x_i)).f(x_i, y(x_i))$$

Igualando-se as potências comuns de h das equações (III) e (IV), chega-se a:

$$h = 0 \Rightarrow y(x_i) = y_i$$

$$h = 1 \Rightarrow f(x_i, y(x_i)) = (a+b) f(x_i, y(x_i))$$

$$h = 2 \Rightarrow \frac{1}{2}\left[f_x(x_i, y(x_i)) + f_y(x_i, y(x_i)).f(x_i, y(x_i))\right]$$

Assumindo-se que:

$$y_i = y(x_i)$$

então:

$$a + b = 1 \Rightarrow a = 1 - b$$

$$\left.\begin{array}{l} bp = \frac{1}{2} \\ bq = \frac{1}{2} \end{array}\right\} \Rightarrow \quad p = q = \frac{1}{2b}$$

$$\left.\begin{array}{l} b = \frac{1}{2} \Rightarrow p = q = 1 \\ b = \frac{1}{2} \Rightarrow a = \frac{1}{2} \end{array}\right\} \Rightarrow \text{Método de Heun (Família de Métodos de 2° Ordem)}$$

Logo:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h, y_i + hk_1)$$

$$y_{i+1} = y_i + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)$$

Ou se escolhermos:

$b = 1 \Rightarrow a = 0$

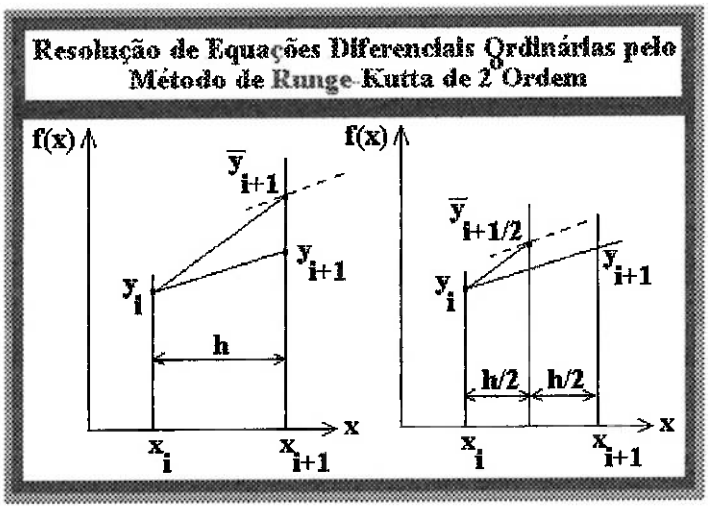
$b = 1 \Rightarrow p = q = \frac{1}{2}$ $\Rightarrow$ Método de Euler Modificado

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1\right)$$

$$y_{i+1} = y_i + hf\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}f(x_i, y_i)\right)$$

A figura abaixo elucida o método de Runge-Kutta de 2°Ordem:



⇒ Método de Runge-Kutta de 3°Ordem

Similarmente ao de segunda ordem e desenvolvendo-se as equações chega-se a:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)$$

$$k_3 = f(x_i + h, y_i + 2hk_2 - hk_1)$$

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 4k_2 + k_3)$$

⇒ Método de Runge-Kutta de 4°Ordem

Similarmente ao de segunda ordem e desenvolvendo-se as equações chega-se a:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right)$$

$$k_4 = f(x_i + h, y_i + hk_3) \quad \Rightarrow \quad y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

"Bases de Equações Diferenciais Ordinárias"

"Seja o seguinte sistema com n Equações Diferenciais Ordinárias:

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, y_3, \ldots, y_n)$$

$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, y_3, \ldots, y_n)$$

.

.

.

$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, y_3, \ldots, y_n)$$

com as seguintes condições iniciais:

$$y_1(x_0) = y_{1_0}$$
$$y_2(x_0) = y_{2_0}$$
$$y_3(x_0) = y_{3_0} \ldots$$
$$y_n(x_0) = y_{n_0}$$

Para a resolução deste sistema de equações diferenciais ordinárias todas de 1°ordem basta aplicar um dos métodos apresentados anteriormente, em paralelo em cada passo. Caso o sistemas de equações a ser resolvido não seja de equações diferenciais ordinárias de 1°Ordem, então reduz-se o sistemas para equações de 1°Ordem e aplica-se os métodos conhecidos de resolução."

"Método Preditor-Corretor"

"Para iniciar o método deve-se utilizar um método explícito. Por exemplo, um Runge-Kutta de 4°Ordem.

As fórmulas fechadas tem um erro local de truncamento menor que as fórmulas abertas. Porém as fórmulas fechadas são implícitas. Assim, para se determinar a posição y_i é necessário conhecer a posição y_{i+1}. Mas como determinar y_{i+1} se não se conhece y_i. Uma alternativa é utilizar uma fórmula aberta para se predizer o valor de y_{i+1} e utilizar uma fórmula fechada para corrigir y_i. Assim constitui-se o Método Preditor-Corretor.

As fórmulas de Milne de 4°Ordem do preditor-corretor são:

Preditor

$$y_{i+1,0} = y_{i-3} + \frac{4h}{3}.\left(2f_i - f_{i-1} + 2f_{i-2}\right)$$

Corretor

$$y_{i+1} = y_{i-1} + \frac{h}{3}\left(f_{i+1} + 4f_i + f_{i-1}\right)$$

Listagem do software

⇨ 1. Unit de implementação da Abertura

```
unit Capa;

interface

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

type
  TF_Capa = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Bevel1: TBevel;
    Label9: TLabel;
    Label10: TLabel;
    MainMenu1: TMainMenu;
    Gerais1: TMenuItem;
    Resumo1: TMenuItem;
    Agradecimentos1: TMenuItem;
    N1: TMenuItem;
    Sair1: TMenuItem;
    Sobre1: TMenuItem;
    Aluno1: TMenuItem;
    Orientador1: TMenuItem;
    BitBtn3: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn1: TBitBtn;
    Panel3: TPanel;
```

```
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Image4: TImage;
    Image5: TImage;
    Bevel5: TBevel;
    Image2: TImage;
    Image3: TImage;
    Panel4: TPanel;
    Bevel4: TBevel;
    Label1: TLabel;
    Panel5: TPanel;
    Bevel3: TBevel;
    Label5: TLabel;
    Label7: TLabel;
    Label6: TLabel;
    Label8: TLabel;
    procedure Sair1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure Resumo1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
     private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Capa: TF_Capa;
```

```
implementation
uses Resumo, Opcao;
{$R *.DFM}

procedure TF_Capa.Sair1Click(Sender: TObject);
begin
close;
end;

procedure TF_Capa.BitBtn3Click(Sender: TObject);
begin
F_Resumo.showmodal;
end;

procedure TF_Capa.Resumo1Click(Sender: TObject);
begin
F_Resumo.showmodal;
end;

procedure TF_Capa.BitBtn2Click(Sender: TObject);
begin
close;
end;

procedure TF_Capa.BitBtn1Click(Sender: TObject);
begin
F_Opcao.showmodal;
end;
end.
```

⇨ 2. Unit de implementação da Tela referente ao resumo do projeto

```
unit Resumo;

interface

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

type
  TF_Resumo = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
```

```
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    BitBtn1: TBitBtn;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Resumo: TF_Resumo;
implementation
{$R *.DFM}
end.
```

⇨ 3. Implementação da janela referente ao primeiro menu de opções

```
unit Opcao;

interface

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

type
  TF_Opcao = class(TForm)
    RadioGroup1: TRadioGroup;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    BitBtn1: TBitBtn;
    procedure close(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Opcao: TF_Opcao;

implementation

uses Opcao2, Modelo;

{$R *.DFM}
```

```
procedure TF_Opcao.close(Sender: TObject);
begin
if (RadioButton1.checked=true) then
  begin
      F_Opcao2.showmodal;
  end;
if (RadioButton2.checked=true) then
  begin
      F_Modelo.showmodal;
  end;
end;
end.
```

⇨ 4. Implementação da janela de opções dos modelos da biblioteca

```
unit Modelo;

interface

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

type
  TF_Modelo = class(TForm)
    Label1: TLabel;
    RadioGroup1: TRadioGroup;
    RadioButton1: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton5: TRadioButton;
    RadioGroup2: TRadioGroup;
```

```
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Modelo: TF_Modelo;

implementation

uses Susp1, RLC1, Opcao;

{$R *.DFM}

procedure TF_Modelo.BitBtn1Click(Sender: TObject);
begin
if (RadioButton1.checked=true) then
  begin
     F_Susp1.showmodal;
  end;
if (RadioButton3.checked=true) then
  begin
     F_RLC1.showmodal;
  end;
```

```
if (RadioButton5.checked=true) then
  begin

  end;
if (RadioButton6.checked=true) then
  begin

  end;
if (RadioButton7.checked=true) then
  begin

  end;
end;

procedure TF_Modelo.FormCreate(Sender: TObject);
begin
F_Opcao.close;
end;

end.
```

⇨ 5. Implementação dos elementos da biblioteca de modelos

```
unit RLC1;

interface

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

```
type
  TF_RLC1 = class(TForm)
    Label6: TLabel;
    Image1: TImage;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Bevel2: TBevel;
    Label9: TLabel;
    Label10: TLabel;
    Bevel3: TBevel;
    BitBtn1: TBitBtn;
    Label11: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_RLC1: TF_RLC1;

implementation
```

```
uses RLC2;

{$R *.DFM}

procedure TF_RLC1.BitBtn1Click(Sender: TObject);
begin
F_RLC2.showmodal;
end;

end.


unit RLC2;

interface

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

type
  TF_RLC2 = class(TForm)
    Label6: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
```

```
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_RLC2: TF_RLC2;

implementation

uses RLC3;

{$R *.DFM}

procedure TF_RLC2.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='C.V') or
  (edit1.text='V.C') or
  (edit1.text='CV') or
  (edit1.text='VC') then
     begin
        F_RLC3.showmodal;
     end;
end;

end.
```

```
unit RLC3;

interface

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

type
  TF_RLC3 = class(TForm)
    Label6: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
```

```
  F_RLC3: TF_RLC3;

implementation

uses RLC4;

{$R *.DFM}

procedure TF_RLC3.BitBtn1Click(Sender: TObject);
begin
if ((edit1.text='R.i') and (edit2.text='L.di(t)/dt')) then
   begin
      F_RLC4.showmodal;
   end;
if ((edit1.text='R.i') and (edit2.text='L.di/dt')) then
   begin
      F_RLC4.showmodal;
   end;
if ((edit1.text='Ri') and (edit2.text='L.di(t)/dt')) then
   begin
      F_RLC4.showmodal;
   end;
if ((edit1.text='Ri') and (edit2.text='L.di/dt')) then
   begin
      F_RLC4.showmodal;
   end;

if ((edit1.text='i.R') and (edit2.text='L.di(t)/dt')) then
   begin
      F_RLC4.showmodal;
```

```
    end;
  if ((edit1.text='iR') and (edit2.text='L.di(t)/dt')) then
    begin
      F_RLC4.showmodal;
    end;
  if ((edit1.text='i.R') and (edit2.text='L.di/dt')) then
    begin
      F_RLC4.showmodal;
    end;
  if ((edit1.text='iR') and (edit2.text='L.di/dt')) then
    begin
      F_RLC4.showmodal;
    end;
  end;

  end.


  unit RLC4;

  interface

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

  type
   TF_RLC4 = class(TForm)
     Label6: TLabel;
     Label1: TLabel;
```

```
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_RLC4: TF_RLC4;

implementation

uses RLC5;

{$R *.DFM}

procedure TF_RLC4.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='dq(t)/dt') or
   (edit1.text='dq/dt') then
      begin
         F_RLC5.showmodal;
      end;
end;
```

```
end.
unit RLC5;

interface

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

type
  TF_RLC5 = class(TForm)
    Label6: TLabel;
    Image1: TImage;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Image2: TImage;
    Label15: TLabel;
    Label16: TLabel;
```

```
    Label17: TLabel;
    Label18: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Bevel2: TBevel;
    Bevel3: TBevel;
    BitBtn1: TBitBtn;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_RLC5: TF_RLC5;

implementation

{$R *.DFM}
```

```
end.



unit Susp1;

interface

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

type
  TF_Susp1 = class(TForm)
    Label1: TLabel;
    Image1: TImage;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Bevel1: TBevel;
    Label8: TLabel;
    Label9: TLabel;
    BitBtn1: TBitBtn;
    Label10: TLabel;
    Label11: TLabel;
    Bevel2: TBevel;
```

```
    Bevel3: TBevel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Susp1: TF_Susp1;

implementation

uses Susp2;

{$R *.DFM}

procedure TF_Susp1.BitBtn1Click(Sender: TObject);
begin
F_Susp2.showmodal;
end;

end.



unit Susp2;

interface
```

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

type
  TF_Susp2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    BitBtn1: TBitBtn;
    Label4: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Susp2: TF_Susp2;

implementation

uses Susp3;

{$R *.DFM}
```

```
procedure TF_Susp2.BitBtn1Click(Sender: TObject);
begin
if ((edit1.text='MOLA') and (edit2.text='AMORTECEDOR')) then
   begin
      F_susp3.showmodal;
   end;
if ((edit2.text='MOLA') and (edit1.text='AMORTECEDOR')) then
   begin
      F_susp3.Showmodal;
   end;
end;

end.


unit Susp3;

interface

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

type
  TF_Susp3 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
```

```
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    Label6: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Susp3: TF_Susp3;

implementation

uses Susp4;

{$R *.DFM}

procedure TF_Susp3.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='x(t)-x0(t)') or (edit1.text='x-x0') or
   (edit1.text='x(t)-xo(t)') or (edit1.text='x-xo') or
   (edit1.text='-x0(t)+x(t)') or (edit1.text='-x0+x') or
   (edit1.text='-xo(t)+x(t)') or (edit1.text='-xo+x') then
       begin
          F_Susp4.showmodal;
       end;
end;
```

```
end.



unit Susp4;

interface

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

type
  TF_Susp4 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  F_Susp4: TF_Susp4;

implementation

uses Susp5;

{$R *.DFM}

procedure TF_Susp4.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='k.[x(t)-x0(t)]') or
   (edit1.text='k.[x(t)-xo(t)]') or
   (edit1.text='k.(x(t)-x0(t))') or
   (edit1.text='k.(x(t)-xo(t))') or
   (edit1.text='k.xr(t)') or
   (edit1.text='k.xr') then
      begin
         F_Susp5.showmodal;
      end;
if (edit1.text='[x(t)-x0(t)].k') or
   (edit1.text='[x(t)-xo(t)].k') or
   (edit1.text='(x(t)-x0(t)).k') or
   (edit1.text='(x(t)-xo(t)).k') or
   (edit1.text='xr(t).k') or
   (edit1.text='xr.k') then
      begin
         F_Susp5.showmodal;
      end;
end;
```

```
end.



unit Susp5;

interface

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

type
  TF_Susp5 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  F_Susp5: TF_Susp5;

implementation

uses Susp6;

{$R *.DFM}

procedure TF_Susp5.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='dx(t)/dt-dx0(t)/dt') or
   (edit1.text='dx/dt-dx0/dt') or
   (edit1.text='dx(t)/dt-dxo(t)/dt') or
   (edit1.text='dx/dt-dxo/dt') or
   (edit1.text='(dx(t)/dt)-(dx0(t)/dt)') or
   (edit1.text='(dx/dt)-(dx0/dt)') or
   (edit1.text='(dx(t)/dt)-(dxo(t)/dt)') or
   (edit1.text='(dx/dt)-(dxo/dt)') or
   (edit1.text='[dx(t)/dt]-[dx0(t)/dt]') or
   (edit1.text='[dx/dt]-[dx0/dt]') or
   (edit1.text='[dx(t)/dt]-[dxo(t)/dt]') or
   (edit1.text='[dx/dt]-[dxo/dt]') then
     begin
        F_susp6.showmodal;
     end;

if (edit1.text='-dx0(t)/dt+dx(t)/dt') or
   (edit1.text='-dx0/dt+dx/dt') or
   (edit1.text='-dxo(t)/dt+dx(t)/dt') or
```

```
    (edit1.text='-dxo/dt+dx/dt') or
    (edit1.text='-(dx0(t)/dt)+(dx(t)/dt)') or
    (edit1.text='-(dx0/dt)+(dx/dt)') or
    (edit1.text='-(dxo(t)/dt)+(dx(t)/dt)') or
    (edit1.text='-(dxo/dt)+(dx/dt)') or
    (edit1.text='-[dx0(t)/dt]+[dx(t)/dt]') or
    (edit1.text='-[dx0/dt]+[dx/dt]') or
    (edit1.text='-[dxo(t)/dt]+[dx(t)/dt]') or
    (edit1.text='-[dxo/dt]+[dx/dt]') then
      begin
         F_susp6.showmodal;
      end;
end;

end.



unit Susp6;

interface

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

type
  TF_Susp6 = class(TForm)
    Label6: TLabel;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Susp6: TF_Susp6;

implementation

uses Susp7;

{$R *.DFM}

procedure TF_Susp6.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='r.dxr(t)/dt') or
   (edit1.text='r.dx(t)/dt-dx0(t)/dt') or
   (edit1.text='r.dx(t)/dt-dxo(t)/dt') then
       begin
```

```
            F_susp7.showmodal;
        end;
    end;

    end.


    unit Susp7;

    interface

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

    type
     TF_Susp7 = class(TForm)
      Label6: TLabel;
      Label1: TLabel;
      Label2: TLabel;
      Label3: TLabel;
      Edit1: TEdit;
      Label4: TLabel;
      Label5: TLabel;
      BitBtn1: TBitBtn;
      procedure BitBtn1Click(Sender: TObject);
     private
      { Private declarations }
     public
      { Public declarations }
```

```
 end;

var
 F_Susp7: TF_Susp7;

implementation

uses Susp8;

{$R *.DFM}

procedure TF_Susp7.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='-Fm-Fa') or
  (edit1.text='-Fa-Fm') then
     begin
        F_susp8.showmodal;
     end;
end;

end.



 unit Susp8;

 interface

 uses
   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
```

```
  ExtCtrls, StdCtrls, Buttons;

type
  TF_Susp8 = class(TForm)
    Label6: TLabel;
    Image1: TImage;
    Bevel3: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Bevel1: TBevel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
```

```
    Label24: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Bevel2: TBevel;
    Label26: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Label38: TLabel;
    Label39: TLabel;
    Label40: TLabel;
    Label41: TLabel;
    Label42: TLabel;
    Label43: TLabel;
    Label44: TLabel;
    Image2: TImage;
    Bevel4: TBevel;
    BitBtn1: TBitBtn;
    Label25: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  F_Susp8: TF_Susp8;

implementation

uses Ordem;

{$R *.DFM}

procedure TF_Susp8.BitBtn1Click(Sender: TObject);
begin
F_Ordem.showmodal;
end;

end.



unit Aleta1;

interface

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

type
  TF_Aleta1 = class(TForm)
```

```
    Image1: TImage;
    Label1: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Bevel1: TBevel;
    Label10: TLabel;
    Bevel2: TBevel;
    BitBtn1: TBitBtn;
    Label7: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Aleta1: TF_Aleta1;

implementation

{$R *.DFM}

end.
```

```
unit Aleta2;

interface

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

type
  TF_Aleta2 = class(TForm)
    Label1: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Aleta2: TF_Aleta2;

implementation

{$R *.DFM}

end.
```

⇨ 6. Implementação das tabelas de parâmetros das equações diferenciais

```
unit DPOrd2;

interface

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

type
  TF_DPOrd2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label9: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  F_DPOrd2: TF_DPOrd2;

implementation

{$R *.DFM}

end.


unit Ordem;

interface

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

type
  TF_Ordem = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  F_Ordem: TF_Ordem;

implementation

uses Ordem1, Ordem2, Ordem3, Ordem4, Ordem5;

{$R *.DFM}

procedure TF_Ordem.BitBtn1Click(Sender: TObject);
begin
if (edit1.text='1') then
  begin
    F_Ordem1.showmodal;
  end;
if (edit1.text='2') then
  begin
    F_Ordem2.showmodal;
  end;
if (edit1.text='3') then
  begin
    F_Ordem3.showmodal;
  end;
if (edit1.text='4') then
  begin
    F_Ordem4.showmodal;
```

```
    end;
  if (edit1.text='5') then
    begin
       F_Ordem5.showmodal;
    end;
  end;

  end.



  unit Ordem1;

  interface

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

  type
   TF_Ordem1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    BitBtn1: TBitBtn;
    Panel1: TPanel;
    Panel2: TPanel;
```

```
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Edit3: TEdit;
    Edit2: TEdit;
    Label9: TLabel;
    Label10: TLabel;
    Edit4: TEdit;
    Edit5: TEdit;
    Bevel1: TBevel;
    Bevel2: TBevel;
    MainMenu1: TMainMenu;
    Helpdosmtodos1: TMenuItem;
    Mtododе1: TMenuItem;
    MtodoPreditorCorretor1: TMenuItem;
    BasesdeEquaesDiferenciaisOrdinrias1: TMenuItem;
    Panel3: TPanel;
    Label11: TLabel;
    Label12: TLabel;
    Bevel3: TBevel;
    Edit6: TEdit;
    Edit7: TEdit;
    Label13: TLabel;
    procedure Mtododе1Click(Sender: TObject);
    procedure MtodoPreditorCorretor1Click(Sender: TObject);
    procedure BasesdeEquaesDiferenciaisOrdinrias1Click(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

var
  F_Ordem1: TF_Ordem1;

implementation

uses Help1, Help3, Help2;

{$R *.DFM}

procedure TF_Ordem1.Mtodode1Click(Sender: TObject);
begin
F_Help1.showmodal;
end;

procedure TF_Ordem1.MtodoPreditorCorretor1Click(Sender: TObject);
begin
F_Help3.showmodal;
end;

procedure TF_Ordem1.BasesdeEquaesDiferenciaisOrdinrias1Click(
  Sender: TObject);
begin
F_Help2.showmodal;
end;

end.
```

```
unit Ordem2;

interface

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

type
  TF_Ordem2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Panel1: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Edit3: TEdit;
    Edit2: TEdit;
    Label11: TLabel;
    Edit6: TEdit;
    Bevel1: TBevel;
    Panel2: TPanel;
    Label9: TLabel;
    Label10: TLabel;
```

```
    Edit4: TEdit;
    Edit5: TEdit;
    Label12: TLabel;
    Label13: TLabel;
    Edit7: TEdit;
    Edit8: TEdit;
    Bevel2: TBevel;
    BitBtn1: TBitBtn;
    MainMenu1: TMainMenu;
    Sobreosmtodos1: TMenuItem;
    MtododeRungeKutta1: TMenuItem;
    MtodoPreditorCorretor1: TMenuItem;
    BasesdeEquaesDiferenciais1: TMenuItem;
    Panel3: TPanel;
    Label14: TLabel;
    Label15: TLabel;
    Bevel3: TBevel;
    Edit9: TEdit;
    Edit10: TEdit;
    Label16: TLabel;
    procedure MtododeRungeKutta1Click(Sender: TObject);
    procedure MtodoPreditorCorretor1Click(Sender: TObject);
    procedure BasesdeEquaesDiferenciais1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
```

```
  F_Ordem2: TF_Ordem2;

implementation

uses Help1, Help3, Help2;

{$R *.DFM}

procedure TF_Ordem2.MtododeRungeKutta1Click(Sender: TObject);
begin
F_Help1.showmodal;
end;

procedure TF_Ordem2.MtodoPreditorCorretor1Click(Sender: TObject);
begin
F_Help3.showmodal;
end;

procedure TF_Ordem2.BasesdeEquaesDiferenciais1Click(Sender: TObject);
begin
F_Help2.showmodal;
end;

end.



unit Ordem3;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, Menus;

type
  TF_Ordem3 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Panel1: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Label11: TLabel;
    Edit6: TEdit;
    Label16: TLabel;
    Edit11: TEdit;
    Bevel1: TBevel;
    Edit3: TEdit;
    Panel2: TPanel;
    Label17: TLabel;
    Label18: TLabel;
    Edit12: TEdit;
    Edit13: TEdit;
```

```
    Label19: TLabel;
    Label20: TLabel;
    Edit14: TEdit;
    Edit15: TEdit;
    Label21: TLabel;
    Label22: TLabel;
    Edit16: TEdit;
    Edit17: TEdit;
    Bevel2: TBevel;
    BitBtn1: TBitBtn;
    MainMenu1: TMainMenu;
    Sobreosmtodos1: TMenuItem;
    MtododeRungeKutta1: TMenuItem;
    MtodoPreditorCorretor1: TMenuItem;
    BasesdeEquaesDiferenciais1: TMenuItem;
    Label13: TLabel;
    Panel3: TPanel;
    Label14: TLabel;
    Label15: TLabel;
    Bevel3: TBevel;
    Edit9: TEdit;
    Edit10: TEdit;
    procedure MtododeRungeKutta1Click(Sender: TObject);
    procedure MtodoPreditorCorretor1Click(Sender: TObject);
    procedure BasesdeEquaesDiferenciais1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  F_Ordem3: TF_Ordem3;

implementation

uses Help1, Help3, Help2;

{$R *.DFM}

procedure TF_Ordem3.MtododeRungeKutta1Click(Sender: TObject);
begin
F_Help1.showmodal;
end;

procedure TF_Ordem3.MtodoPreditorCorretor1Click(Sender: TObject);
begin
F_Help3.showmodal;
end;

procedure TF_Ordem3.BasesdeEquaesDiferenciais1Click(Sender: TObject);
begin
F_Help2.showmodal;
end;

end.



unit Ordem4;
```

```
interface

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

type
  TF_Ordem4 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Panel1: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Edit3: TEdit;
    Edit2: TEdit;
    Label11: TLabel;
    Edit6: TEdit;
    Label16: TLabel;
    Edit11: TEdit;
    Label19: TLabel;
    Edit14: TEdit;
    Bevel1: TBevel;
    Panel2: TPanel;
```

Label9: TLabel;

Label10: TLabel;

Edit4: TEdit;

Edit5: TEdit;

Label12: TLabel;

Label13: TLabel;

Edit7: TEdit;

Edit8: TEdit;

Label14: TLabel;

Label15: TLabel;

Edit9: TEdit;

Edit10: TEdit;

Label17: TLabel;

Label18: TLabel;

Edit12: TEdit;

Edit13: TEdit;

Bevel2: TBevel;

BitBtn1: TBitBtn;

MainMenu1: TMainMenu;

Sobreosmtodos1: TMenuItem;

MtododeRungeKutta1: TMenuItem;

MtodoPreditorCorretor1: TMenuItem;

BasesdeEquaesDiferenciais1: TMenuItem;

Panel3: TPanel;

Label20: TLabel;

Label21: TLabel;

Bevel3: TBevel;

Edit15: TEdit;

Edit16: TEdit;

Label22: TLabel;

```
    procedure MtododeRungeKutta1Click(Sender: TObject);
    procedure MtodoPreditorCorretor1Click(Sender: TObject);
    procedure BasesdeEquaesDiferenciais1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Ordem4: TF_Ordem4;

implementation

uses Help1, Help3, Help2;

{$R *.DFM}

procedure TF_Ordem4.MtododeRungeKutta1Click(Sender: TObject);
begin
F_Help1.showmodal;
end;

procedure TF_Ordem4.MtodoPreditorCorretor1Click(Sender: TObject);
begin
F_Help3.showmodal;
end;

procedure TF_Ordem4.BasesdeEquaesDiferenciais1Click(Sender: TObject);
begin
```

```
F_Help2.showmodal;
end;

end.



unit Ordem5;

interface

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

type
  TF_Ordem5 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Label3: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Panel1: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Edit3: TEdit;
    Edit2: TEdit;
```

Label11: TLabel;
Edit6: TEdit;
Label16: TLabel;
Edit11: TEdit;
Label19: TLabel;
Edit14: TEdit;
Label22: TLabel;
Edit17: TEdit;
Bevel1: TBevel;
Panel2: TPanel;
Label9: TLabel;
Label10: TLabel;
Edit4: TEdit;
Edit5: TEdit;
Label12: TLabel;
Label13: TLabel;
Edit7: TEdit;
Edit8: TEdit;
Label14: TLabel;
Label15: TLabel;
Edit9: TEdit;
Edit10: TEdit;
Label17: TLabel;
Label18: TLabel;
Edit12: TEdit;
Edit13: TEdit;
Label20: TLabel;
Label21: TLabel;
Edit15: TEdit;
Edit16: TEdit;

```
    Bevel2: TBevel;
    BitBtn1: TBitBtn;
    MainMenu1: TMainMenu;
    Sobreosmtodos1: TMenuItem;
    MtododeRungeKutta1: TMenuItem;
    MtodoPreditorCorretor1: TMenuItem;
    BasesdeEquaesDiferenciais1: TMenuItem;
    Label23: TLabel;
    Panel3: TPanel;
    Label24: TLabel;
    Label25: TLabel;
    Bevel3: TBevel;
    Edit18: TEdit;
    Edit19: TEdit;
    procedure MtododeRungeKutta1Click(Sender: TObject);
    procedure MtodoPreditorCorretor1Click(Sender: TObject);
    procedure BasesdeEquaesDiferenciais1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Ordem5: TF_Ordem5;

implementation

uses Help1, Help3, Help2;
```

```
{$R *.DFM}

procedure TF_Ordem5.MtododeRungeKutta1Click(Sender: TObject);
begin
F_Help1.showmodal;
end;

procedure TF_Ordem5.MtodoPreditorCorretor1Click(Sender: TObject);
begin
F_Help3.showmodal;
end;

procedure TF_Ordem5.BasesdeEquaesDiferenciais1Click(Sender: TObject);
begin
F_Help2.showmodal;
end;

end.
```

⇨ 7. Implementação dos help's didáticos

```
unit Help1;

interface

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

type
  TF_Help1 = class(TForm)
```

```
Image1: TImage;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
Label14: TLabel;
Label15: TLabel;
Label16: TLabel;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Label21: TLabel;
Label22: TLabel;
Label23: TLabel;
Label24: TLabel;
Label25: TLabel;
Label26: TLabel;
Label27: TLabel;
Label28: TLabel;
Label29: TLabel;
```

```
    Label41: TLabel;
    Label43: TLabel;
    Label44: TLabel;
    Label45: TLabel;
    Label46: TLabel;
    Label47: TLabel;
    Label30: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label31: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Help1: TF_Help1;

implementation

{$R *.DFM}

procedure TF_Help1.BitBtn1Click(Sender: TObject);
begin
```

```
close;
end;

end.



unit Help2;

interface

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

type
  TF_Help2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label6: TLabel;
    Label8: TLabel;
    Label10: TLabel;
    Label15: TLabel;
    Label24: TLabel;
    Label4: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
```

```
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Shape1: TShape;
    Label5: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Bevel1: TBevel;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
```

```
  F_Help2: TF_Help2;

implementation

{$R *.DFM}

procedure TF_Help2.BitBtn1Click(Sender: TObject);
begin
close;
end;

end.



unit Help3;

interface

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

type
  TF_Help3 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
```

```
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  F_Help3: TF_Help3;

implementation

{$R *.DFM}

procedure TF_Help3.BitBtn1Click(Sender: TObject);
begin
close;
end;

end.
```