

# Informática no Ensino de Física

Carlos Eduardo Aguiar

Instituto de Física, Universidade Federal do Rio de Janeiro



# Computadores no ensino de Física

AULA

1

## Metas da aula

Discutir como os computadores podem ser úteis ao ensino e à aprendizagem de Física. Apresentar as principais formas de utilização da Informática no ensino de Física.

## objetivos

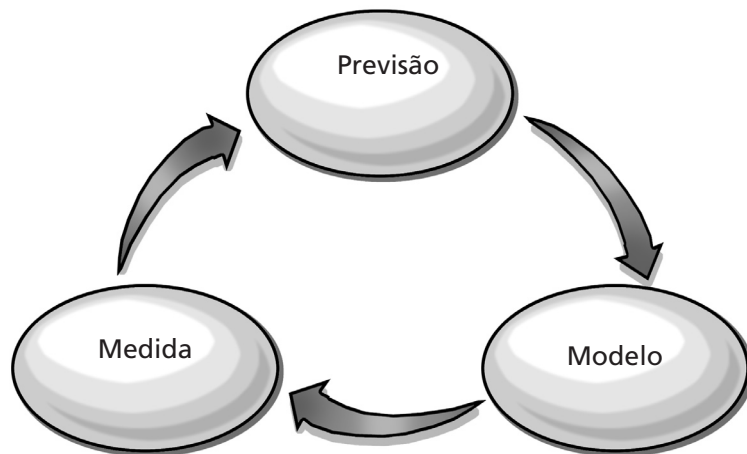
Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- compreender o impacto que os computadores podem ter no ensino de Física;
- distinguir as diferentes formas de utilização dos computadores no ensino de Física, avaliando seus méritos e deficiências.

## INTRODUÇÃO

Microcomputadores tornaram-se presença quase obrigatória nos ambientes escolares brasileiros, embora muitos professores e alunos não tenham idéia de como ou por que devem utilizá-los no processo de ensino-aprendizagem. Isso é uma pena, pois, como veremos, os computadores podem ser instrumentos pedagógicos extraordinariamente eficientes. Nosso interesse vai estar restrito ao ensino de Física e, para entender a relevância dos computadores nessa área, talvez seja bom discutir brevemente como “funciona” a Física. Um esquema de como o conhecimento físico se desenvolve está na **Figura 1.1**.

Errata:  
Na Figura 1.1, as setas devem apontar no sentido oposto.



**Figura 1.1:** Ciclo de desenvolvimento das teorias físicas.

A figura mostra um ciclo com três etapas: a formulação da teoria ou modelo, a obtenção de previsões a partir desta teoria, e a comparação dessas previsões com observações ou medidas. A passagem do modelo básico a previsões específicas é necessária porque os princípios fundamentais de uma teoria física raramente podem ser colocados diretamente a teste. O que geralmente se faz é usar esses princípios para obter uma previsão que possa ser verificada e, em seguida, realizar um experimento para determinar se tal resultado é verdadeiro ou falso. Caso o experimento não esteja de acordo com a previsão, a teoria deve estar errada ou incompleta, e é abandonada ou aperfeiçoada. Se o experimento confirmar o resultado previsto, isto não quer dizer que a teoria esteja correta – ela passou por um teste e ganhou credibilidade, mas ainda pode estar errada. Mais testes são necessários. E aí o ciclo começa novamente: outra previsão é feita e comparada com dados experimentais e, a cada sucesso, a teoria vai ganhando mais e mais credibilidade. Teorias que passam ilesas por

muitos desses testes acabam virando “verdades científicas” ou “leis físicas”. Mesmo nesses casos, nada impede que uma nova previsão venha a revelar-se incompatível com fatos experimentais ou que experimentos mais precisos venham a refutar uma previsão até então tida como confirmada. Basta um resultado negativo – se bem estabelecido, é claro – para que se considere que a teoria deva ser modificada ou até deixada de lado.

O que está (rudimentarmente) descrito acima é o conceito científico de “verdade”. Ele não se restringe à Física, e aplica-se a todas as ciências naturais. E não é um conceito qualquer. A idéia de que observação e experimento são os juízes de nossas teorias sobre a natureza levou séculos para ser elaborada e aceita, e é um dos grandes patrimônios da humanidade. Como tal, deve fazer parte da educação de todos os cidadãos – as escolas deveriam comparar o procedimento científico a outras maneiras muito comuns de se escolher o que é verdadeiro ou falso: aceitar a opinião da autoridade, concordar com a maioria, acreditar em revelações místicas etc. Esses últimos “critérios de verdade” podem ter relevância social ou pessoal, mas é importante que membros de uma sociedade moderna percebam que tais métodos não devem ser chamados científicos nem são base para a compreensão de fenômenos naturais.

Dada a importância que o conhecimento sobre a natureza da Ciência tem para a formação do cidadão, parece razoável esperar que o ensino de Física explore, de alguma forma, o procedimento esboçado anteriormente. As idéias apresentadas aos alunos devem ser justificadas pelo fato de estarem de acordo com observações e experimentos, e não porque foram ditadas pelo professor, que “sabe mais”, ou porque estão no livro-texto, ou porque “caem na prova”. Essas últimas justificativas contrariam a própria noção de conhecimento científico a ser ensinada. Entretanto, é fácil constatar, nas salas de aula, que tais práticas “anticientíficas” são quase regra quando se trata do ensino de ciências e, em particular, de Física. Esse paradoxo tem muitas causas: culturais (explicações não científicas são aceitas naturalmente pela sociedade), materiais (pode ser caro fazer um ensino diferente) ou, simplesmente, a dificuldade de imaginar e implementar alternativas.

Um dos maiores obstáculos para se ensinar em sala de aula algo semelhante à Física está na linguagem matemática. Extrair resultados de uma lei física normalmente exige o domínio de ferramentas matemáticas pouco acessíveis aos alunos da escola média. Nesses casos, é difícil fazer de forma lógica e inteligível a passagem da teoria para previsões que possam ser testadas; há uma “barreira matemática” interrompendo o ciclo da **Figura 1.1**. É por isso

que a maioria dos estudantes pensa que a Física é um conjunto de fórmulas a serem decoradas, sem que se saiba ao certo de onde saíram nem onde se aplicam. Como veremos mais à frente, este quadro pode mudar bastante com o auxílio de computadores e programas de modelagem matemática. Tais ferramentas deixam a matemática avançada muito mais acessível a estudantes e professores, facilitando a obtenção de resultados práticos a partir das leis físicas básicas. Com isso, fica viável percorrer o ciclo da **Figura 1.1** em várias situações de interesse real, tornando possível ensinar Física de modo que ela se pareça mais com ciência e menos com dogma.

Há outro componente do ensino de Física que pode ser positivamente influenciado pelos computadores: o laboratório didático. O ensino das ciências no Brasil é tradicionalmente teórico, com pouca atenção a experimentos e demonstrações, e a maioria das escolas sequer dispõe de laboratórios. Esse é mais um fator que aumenta a distância entre a física das escolas e a ciência física. Numa situação tão precária, pode parecer estranho falar em laboratórios didáticos com computadores, pois certamente é possível fazer demonstrações e experimentos com material muito mais simples e barato. Mas devemos notar que o computador já está presente (subutilizado) em muitas escolas, e é uma ferramenta de laboratório tremendamente poderosa e versátil. Ele pode substituir sozinho uma grande variedade de instrumentos, alguns até mais caros, como o osciloscópio. Uma escola que tem computadores à disposição de alunos e professores já dispõe de um bom laboratório didático, talvez até sem sabê-lo. Não teremos oportunidade de discutir em detalhe o uso dos computadores como instrumento de medida, mas existem muitos textos descrevendo experimentos que podem ser realizados com essas máquinas. Duas boas fontes de artigos interessantes são a *Revista Brasileira de Ensino de Física* e a *Física na Escola*, ambas editadas pela Sociedade Brasileira de Física e disponíveis gratuitamente na internet.

A **Figura 1.2** ilustra um pouco do que foi dito anteriormente, mostrando onde os computadores podem ajudar a fazer um ensino de Física em que esta é mais bem tratada como ciência. Não por acaso, o impacto é muito semelhante ao que eles têm atualmente na pesquisa em Física.

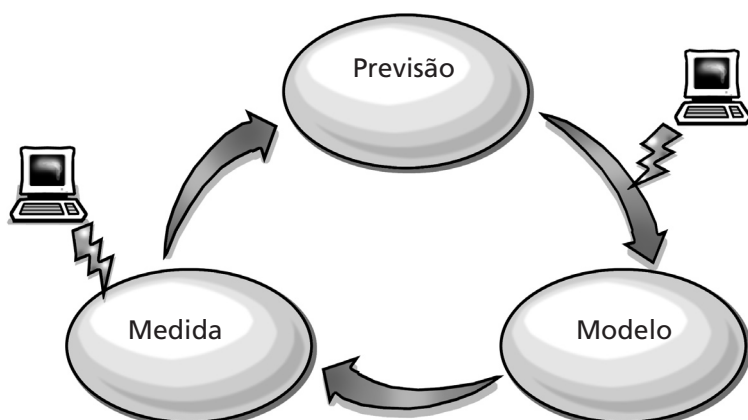


Figura 1.2: Impacto dos computadores no ensino de Física.

Errata:  
Na Figura 1.2, as setas  
devem apontar no sentido  
oposto.

Até aqui, discutimos como o computador pode ajudar a transformar o ensino de Física. Deve ficar claro que ele não é o único componente necessário a esta transformação; sem uma mudança na visão do que seja ensinar Física e a elaboração de material instrucional apropriado, a mera presença de computadores terá pouco ou nenhum efeito na direção que apontamos. Também é importante notar que o emprego do computador como ferramenta de modelagem ou como instrumento de laboratório não são os únicos a possuir valor pedagógico: há muitas outras maneiras de se usar a Informática no ensino de Física. Na próxima seção apresentaremos uma visão mais abrangente do potencial didático dos computadores, descrevendo as diferentes formas de utilização que têm sido identificadas no ensino de Física.

## COMO OS COMPUTADORES TÊM SIDO USADOS NO ENSINO DE FÍSICA

Computadores têm sido usados para ensinar Física há bastante tempo – desde a década de 1960, senão antes. Pelo menos quatro formas de utilização podem ser esquematicamente identificadas:

### Instrução assistida por computador

Foi uma das primeiras aplicações da Informática ao ensino: o computador é programado para agir de forma tutorial, apresentando conteúdos ao aluno, fazendo perguntas e verificando as respostas.

O computador é usado pelo estudante como uma espécie de “livro eletrônico”, com algumas vantagens sobre os livros de papel. Entre estas, estão a possibilidade de apresentar filmes e animações, e de ser organizado na forma de hipertexto. Nos dias de hoje, a facilidade para se distribuir material deste tipo via internet também é um ponto favorável. A instrução assistida por computador foi muito criticada por educadores (já a chamaram até de “virador eletrônico de páginas”) e praticamente foi abandonada após a década de 1980, pelo menos no que diz respeito ao ensino de Física. Seu maior defeito, provavelmente, é o de deixar os estudantes no papel passivo de espectadores ou leitores. Entretanto, é possível que esta forma de uso dos computadores volte a ganhar importância, principalmente em iniciativas de ensino a distância.

### Programas de simulação

São programas que simulam o comportamento de sistemas físicos a partir de modelos predeterminados. Os resultados da simulação geralmente são apresentados em formatos de grande apelo visual, como animações, gráficos etc. Conceitos pouco intuitivos e de difícil visualização tornam-se mais acessíveis aos estudantes, que têm uma oportunidade melhor de compreendê-los corretamente. Hoje em dia, os programas de simulação constituem, provavelmente, o modo mais popular de aplicação de computadores ao ensino de Física. Uma enorme quantidade desses programas está disponível na internet, na forma de *applets* escritos em Java e Flash. Coleções de *applets* interessantes estão em muitos locais, como, por exemplo, em:

- [http://dmoz.org/Science/Physics/Education/Interactive\\_Animations/](http://dmoz.org/Science/Physics/Education/Interactive_Animations/)
- [http://directory.google.com/Top/Science/Physics/Education/Interactive\\_Animations/](http://directory.google.com/Top/Science/Physics/Education/Interactive_Animations/)
- [http://physicsweb.org/resources/Education/Interactive\\_experiments/](http://physicsweb.org/resources/Education/Interactive_experiments/)
- <http://webphysics.davidson.edu/Applets/Applets.html>
- <http://lectureonline.cl.msu.edu/~mmp/applist/applets.htm>

Se você tiver conexão à internet e um navegador capaz de executar programas em Java, dê uma olhada em algumas simulações nesses endereços. Não é difícil convencer-se de que elas têm grande potencial

didático e podem ser muito úteis ao ensino de Física. Em contraste com o que ocorre na instrução assistida por computador, o estudante tende a adotar uma postura mais ativa frente a uma simulação, já que se espera que ele use o programa para explorar o modelo físico de interesse. Quase sempre, isto é feito variando os parâmetros e condições iniciais que são dados de entrada do programa. Neste contexto, o aluno faz perguntas ao computador, do tipo “se eu escolher estes parâmetros, o que acontece?”. Há uma mudança significativa em relação à situação anterior, onde, no máximo, ele respondia a perguntas formuladas pelo programa de instrução.

Um problema com esses programas é que, na maioria das vezes, o modelo físico subjacente à simulação não pode ser modificado pelo estudante ou professor. Isso limita o alcance das “perguntas” que podem ser feitas ao computador: não vale perguntar “e se eu mudar o modelo, o que acontece?”. Outro problema é que, em alguns casos, o aluno nem percebe que por trás da simulação está um modelo, e passa a ver no programa um substituto à realização de experimentos. É óbvio que uma confusão entre teoria e realidade pode trazer sérios problemas à aprendizagem de Física (ou à de qualquer outra ciência). Mas, de qualquer forma, deve-se reconhecer que os programas de simulação representam um avanço na utilização do computador como ferramenta pedagógica inovadora, indo muito além do livro ou de aula virtual.

### **Ferramenta de modelagem**

Esta é uma proposta antiga (provavelmente surgiu com a linguagem Logo, nos anos 1960), mas pouco adotada na prática. Aqui o estudante usa o computador para *criar e explorar* modelos de sistemas físicos. Ao contrário do que ocorre com os programas de simulação, o aluno pode participar ativamente da definição e desenvolvimento do modelo, e é capaz de modificá-lo se julgar apropriado. Este é um passo à frente em relação a simulações que já são recebidas prontas, pois, ao construir os modelos, o estudante pode explorá-los de uma forma muito mais ampla e significativa. A facilidade com que técnicas matemáticas relativamente avançadas são tratadas no computador torna possível abordar de forma consistente muitos temas importantes da Física, considerados inacessíveis ao Ensino Médio. É este uso do computador que pode abrir espaço ao ensino preconizado na seção anterior, ajudando os estudantes não só



a aprender Física, mas também a compreender melhor a natureza da ciência e do conhecimento científico. Existe uma razoável quantidade de programas que podem ser usados como ferramenta de modelagem no Ensino Médio. Alguns foram desenvolvidos com esta finalidade, como o *Modellus* e as linguagens de programação *Logo*, *Squeak* e *Stella*. Sobre os dois primeiros falaremos muito ao longo deste curso. Outros, como as planilhas eletrônicas, não foram planejados especificamente para aplicações ao ensino, mas têm sido usados com sucesso nesta área.

### **Instrumento de laboratório**

Computadores conectados a sensores são encontrados em qualquer laboratório moderno de pesquisa e desenvolvimento. Nos laboratórios didáticos, sua utilização é crescente. Computadores permitem fazer experimentos que dificilmente seriam realizados com os instrumentos usuais de um laboratório de ensino, e podem apresentar os resultados quase imediatamente, na forma de gráficos e tabelas, reduzindo o tempo gasto em tarefas repetitivas associadas ao tratamento de dados. Existem empresas especializadas em material para laboratórios escolares que vendem interfaces e sensores prontos para utilização em experimentos didáticos, embora os preços sejam altos para os padrões típicos das escolas brasileiras. Uma alternativa mais barata é utilizar interfaces e sensores já presentes na maioria dos computadores, como a placa de som e o microfone, ou a porta de jogos e o *joystick* (veja, por exemplo, <http://www.if.ufrj.br/~carlos/joystick/joystick.html>). Muitos experimentos interessantes têm sido desenvolvidos com esses meios, reduzindo o custo da montagem praticamente a zero (no caso de o computador já estar disponível). Outra possibilidade é a análise de vídeos – com *webcams* relativamente baratas pode-se, por exemplo, filmar o movimento de objetos e analisar o resultado com programas especializados, como o SAM, produzido pela USP (<http://educar.sc.usp.br/sam/>).

### **Informação e conhecimento**

Existem outras maneiras de se classificar a aplicação de computadores ao ensino de Física. Podemos, por exemplo, distinguir apenas dois tipos de uso: como (1) *máquina de fornecer informação* ou (2) *ferramenta de construção do conhecimento*. A perspectiva de “máquina de informação” permeia as propostas de instrução assistida por computador

e suas derivadas. A internet, vista como uma biblioteca virtual, também se encaixa nesse modelo. Embora numa escala menor, esta concepção também está por trás do uso dos programas de simulação. O ponto de vista alternativo, de “ferramenta cognitiva”, é o que prevalece quando se utiliza o computador em atividades de modelagem e laboratório. Os educadores mais informados tendem a defender esta última perspectiva, enquanto os órgãos administrativos oficiais quase sempre pregam que as escolas usem as máquinas de informação, colocando-as inclusive como solução para muitos dos problemas educacionais brasileiros. Não há dúvida de que o acesso à informação é essencial para o ensino efetivo de qualquer disciplina. Mas informação e conhecimento são coisas distintas. O conhecimento é algo que é construído, muitas vezes arduamente, e não simplesmente adquirido. Por isso, é importante reconhecer no computador uma ferramenta que pode ajudar essa construção, e não apenas uma fonte de informação.

### **Como será este curso**

O objetivo principal deste curso é mostrar como o computador pode ser utilizado para construir novos conhecimentos. Vamos tratar especialmente do seu uso como ferramenta de modelagem matemática, explorando um programa de modelagem (*Modellus*) e uma linguagem de programação (*Logo*) que podem ser empregados com muito proveito no ensino médio de Física. Será um curso essencialmente prático – é fundamental que todas as aulas sejam desenvolvidas frente a um computador, com os programas necessários devidamente instalados.

### **INFORMAÇÕES SOBRE A PRÓXIMA AULA**

Na próxima aula, discutiremos nossa primeira ferramenta de modelagem matemática: o programa *Modellus*. Vamos mostrar como obter e instalar o programa e apresentaremos suas principais características.

## O *Modellus*

AULA

# 2

### Metas da aula

Apresentar e discutir as principais características do programa *Modellus* e dar alguns exemplos de sua utilização.

## objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- compreender a concepção geral do programa *Modellus* e suas formas de uso;
- obter e instalar o *Modellus*;
- reconhecer e operar as janelas do programa;
- rodar simulações pré-programadas;
- utilizar o programa de ajuda.

## INTRODUÇÃO

O *Modellus* é um programa de *modelagem matemática*, desenvolvido especialmente para ser uma ferramenta de ensino-aprendizagem. Com ele, alunos e professores podem criar e explorar modelos matemáticos aplicáveis a muitos fenômenos naturais. Os modelos podem ser formulados de muitas maneiras – relações funcionais, equações diferenciais, equações iterativas – e são introduzidos no programa utilizando-se a mesma linguagem empregada nos livros e salas de aula. Para usar o *Modellus*, os estudantes não precisam aprender uma linguagem de programação nem familiarizar-se com metáforas computacionais pouco comuns.

Uma das principais características do *Modellus* é que ele permite explorar *múltiplas representações* do objeto que está sendo estudado. Num único ambiente, pode-se apresentar o mesmo objeto sob diferentes perspectivas. Fórmulas, gráficos, vetores e animações são algumas possibilidades. A capacidade de apresentar e manipular visões diferentes e complementares de uma mesma idéia dá ao usuário do *Modellus* a oportunidade de desenvolver uma intuição sobre o que está sendo estudado, facilitando a criação e fixação de modelos mentais apropriados.

Com o *Modellus* também é possível analisar fotos e vídeos armazenados no computador. O programa dispõe de ferramentas para fazer medidas sobre imagens colocadas na tela, o que transforma fotos e filmes em fonte importante e acessível de dados experimentais. A comparação desses dados com modelos criados no próprio programa pode ser feita diretamente, superpondo-se os resultados dos cálculos matemáticos às imagens analisadas.

Existem duas maneiras de se usar o *Modellus* em atividades de ensino-aprendizagem: a *exploratória* e a *expressiva*. Na primeira, os estudantes utilizam modelos e representações desenvolvidos por outras pessoas (seus professores, por exemplo) para estudar o assunto de interesse. Nesse tipo de atividade, o *Modellus* é usado basicamente como um *programa de simulação*, com o qual os alunos interagem apenas por meio da escolha de dados de entrada. No modo expressivo, os estudantes constroem seus próprios modelos e determinam a maneira de representar seus resultados. Aqui, o *Modellus* assume o papel de *ferramenta de modelagem*, que dá ao estudante amplo espaço de exploração e intervenção. Também é possível adotar uma combinação dos dois métodos, por exemplo, propondo que os alunos modifiquem modelos criados pelos professores, adaptando-os a novas situações.

Uma visão panorâmica das possibilidades de uso do *Modellus* e de sua concepção geral está apresentada na **Figura 2.1**. Esta contém um mapa conceitual, criado

por E. A. Veit e V. D. Teodoro (veja a sugestão de leitura 1 ao final desta aula), que mostra alguns dos aspectos do programa já comentados, e vários outros que merecem atenção. Por exemplo, um ponto ressaltado no mapa é que o *Modellus* trabalha com objetos que são representações concretas (ainda que apenas no computador) de idéias matemáticas abstratas. A manipulação direta desses objetos “concreto-abstratos” é um recurso pedagógico poderoso que facilita a compreensão das construções matemáticas e conceitos físicos que estão sendo estudados.

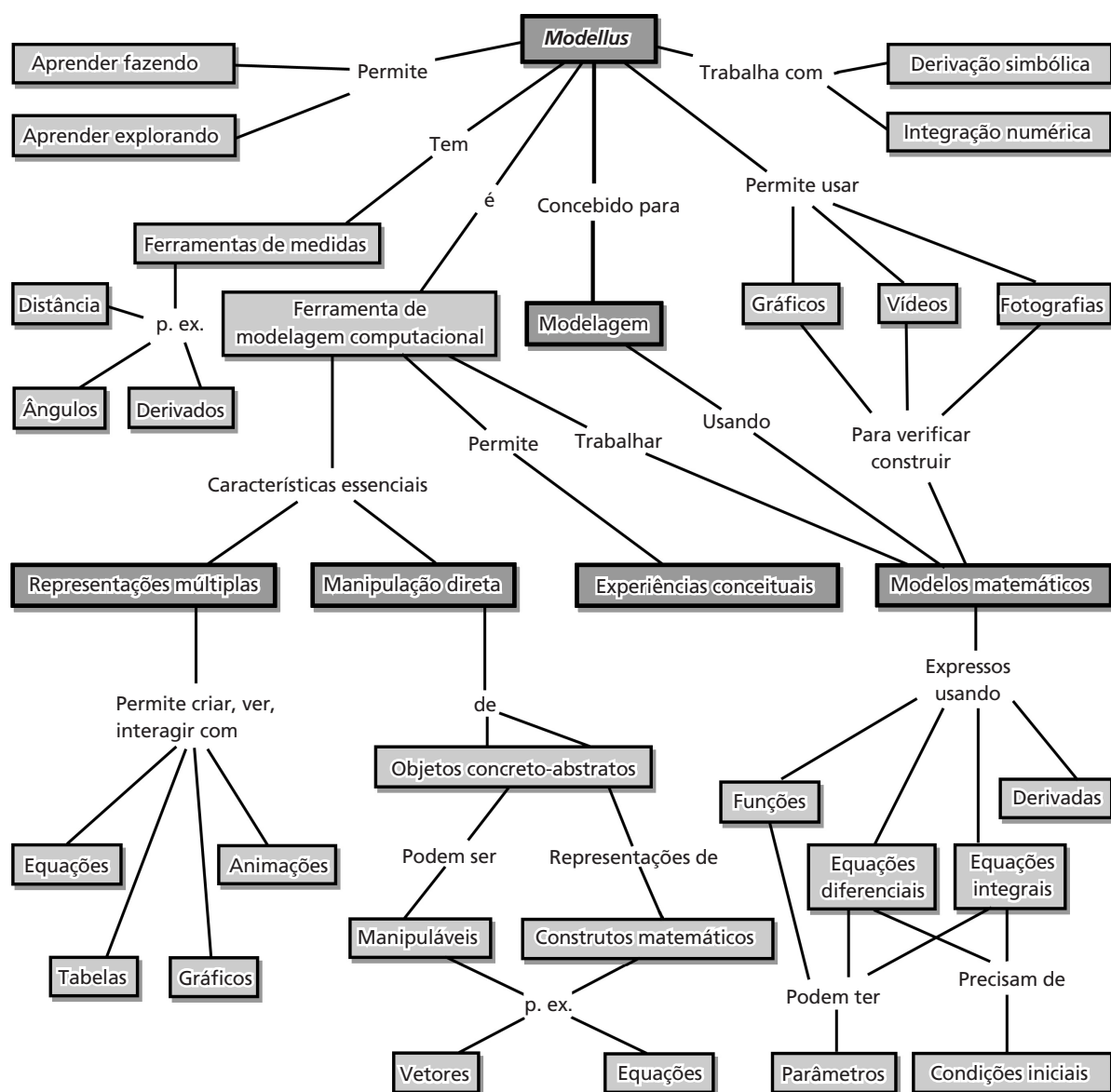


Figura 2.1: Mapa conceitual do *Modellus*. Uma versão online deste mapa, com recursos de hipertexto, pode ser encontrada em <http://phoenix.sce.fct.unl.pt/modellus/>

## OBTENDO E INSTALANDO O *MODELLUS*

*Modellus* é um programa gratuito, que pode ser obtido via internet no endereço <http://phoenix.sce.fct.unl.pt/modellus>

A versão atual do programa é o *Modellus 2.5*, que só roda em sistemas Windows. Esta é a versão que usaremos em nossas discussões. Para obtê-la, você deve antes fazer um rápido registro, informando seu nome, *e-mail* e endereço. As versões mais antigas do *Modellus*, que também são apenas para Windows, não necessitam do registro para serem baixadas. Uma vez registrado, você pode fazer o *download* do *Modellus 2.5*. Note que se deve escolher a língua usada no programa (e no material de ajuda): as opções vão do inglês ao polonês, passando pelo português do Brasil. Escolha esta última e baixe o programa correspondente, que tem pouco mais de 2MB.

Se tudo deu certo, o programa que você obteve chama-se *Modellus\_Setup\_2.5\_br.EXE*. Para instalar o *Modellus* em seu computador, basta executar este programa. Terminada a instalação, você poderá rodar o *Modellus* indo para *Iniciar / Todos os Programas / Modellus 2.5 br* (ou, se o seu sistema for em inglês, *Start / All Programs / Modellus 2.5 br*). Dependendo de suas opções de instalação, também poderá haver um atalho para o *Modellus* na área de trabalho da tela do computador. Clicando-se no ícone do atalho, o programa é iniciado.

## AS JANELAS DO *MODELLUS*

O aspecto do *Modellus*, ao ser iniciado, está mostrado na **Figura 2.2**. Uma janela é aberta, intitulada *Modellus – Modelo Sem Nome*. Esta janela principal contém outras janelas:

- *Modelo*, onde você escreve o modelo matemático que deseja estudar.
- *Controlo*, usada para executar a simulação baseada em seu modelo.
- *Gráfico*, para fazer gráficos das quantidades definidas no modelo.
- *Animação*, onde são criadas as animações associadas ao modelo. Também permite a inserção de figuras, fotos e vídeos.
- *Notas*, para escrever comentários sobre o modelo e a simulação.

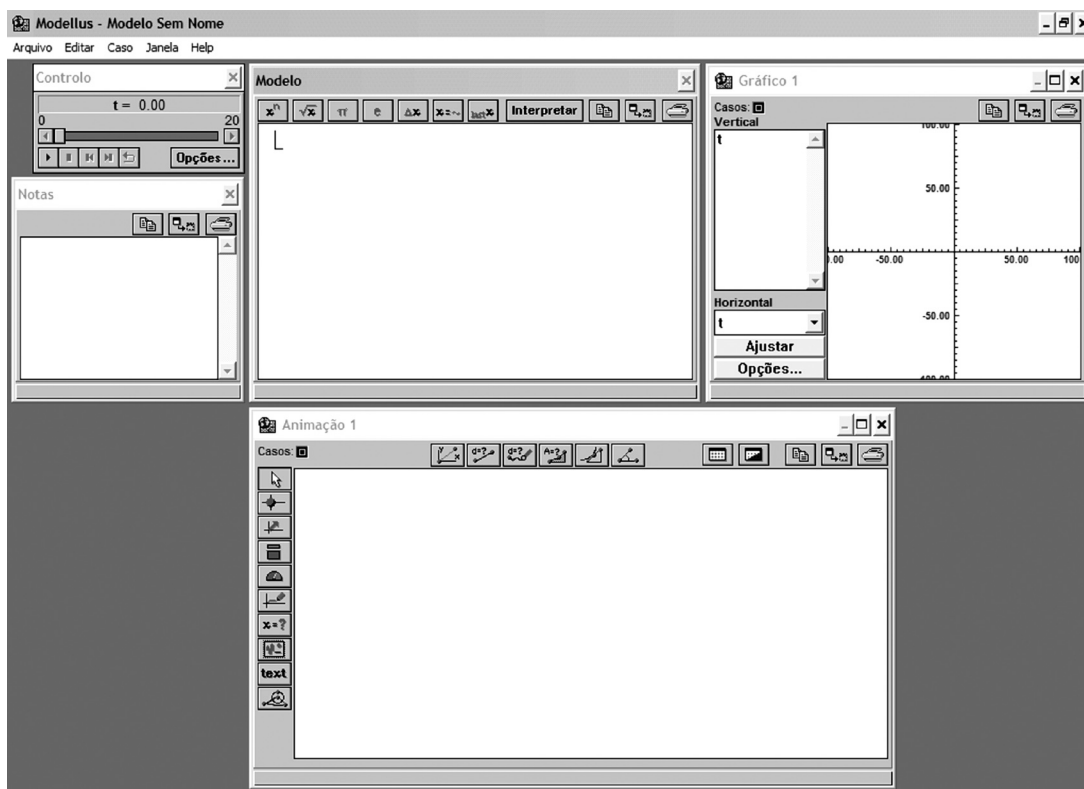


Figura 2.2: O Modellus e algumas de suas janelas.

Existem outras janelas importantes no Modellus, que não aparecem na Figura 2.2:

- *Condições Iniciais*, onde os parâmetros e condições iniciais são especificados. Ela só é aberta quando o modelo possui parâmetros ou condições iniciais “livres”.
- *Tabela* mostra tabelas numéricas de todas as quantidades utilizadas no modelo.

É possível abrir até três janelas *Gráfico* simultaneamente; basta ir à barra de menu e escolher *Janela / Novo Gráfico*. O mesmo pode ser feito com as janelas do tipo *Animação* e *Tabela*.

As janelas do Modellus podem ser deslocadas e redimensionadas com o cursor, na maneira usual do Windows. Já o processo de minimização é um pouco diferente. Embora o botão de minimização do Windows apareça em algumas janelas, ele não tem função alguma nesses casos, e é substituído pelo botão *Esconder Janela* colocado logo abaixo. A Figura 2.3 mostra esse botão – os outros ao seu lado servem para imprimir e copiar o conteúdo da janela. Os botões do Windows para

maximizar e fechar janelas também podem ser vistos, mas em alguns casos ficam desabilitados. Note que fechar uma janela é diferente de escondê-la. No primeiro caso, ela é destruída; se quisermos tê-la de volta, teremos de recriar todo o seu conteúdo. Por sua vez, uma janela escondida não foi destruída, está apenas oculta. Para vê-la de novo, basta ir ao menu *Janela* e clicar sobre o nome da janela oculta.

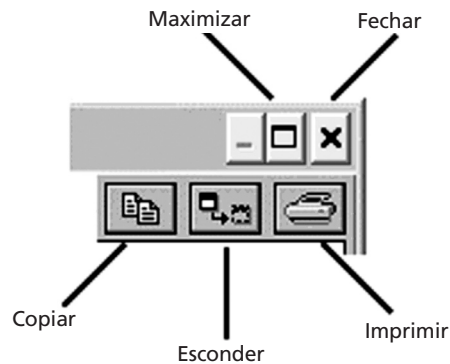


Figura 2.3: Botões de controle de janela no *Modellus*.

Quando uma janela é maximizada, ela passa a ocupar toda a área da tela disponível no *Modellus*. Para inverter o processo, reduzindo o espaço tomado pela janela, deve-se ir ao menu *Janela* e escolher uma das opções de arrumação: *Normal*, *Cascata* ou *Arrumar*.

## RODANDO SIMULAÇÕES PRONTAS

A melhor maneira de se ganhar alguma familiaridade com o *Modellus* é rodando simulações que já estão prontas. Junto com o programa vem um conjunto de simulações pré-programadas, que ilustram o que pode ser feito com o *Modellus* e servem como uma introdução ao seu uso.

Para abrir uma dessas simulações, vá à barra de menu e escolha *Arquivo / Abrir*. Uma caixa de diálogo vai aparecer, e deve mostrar vários arquivos com extensão *mdl*, que identifica as simulações programadas em *Modellus* (veja a Figura 2.4). Normalmente, os arquivos estarão em *c:\arquivos de programa\Modellus 2.5 br*, mas isto pode mudar, dependendo do seu sistema operacional e de como o *Modellus* foi instalado. No mesmo local, junto com os arquivos *mdl*, deve estar uma pasta chamada *Tutorial* (veja novamente a Figura 2.4), na qual está um conjunto de simulações que servem como um pequeno curso de *Modellus*.



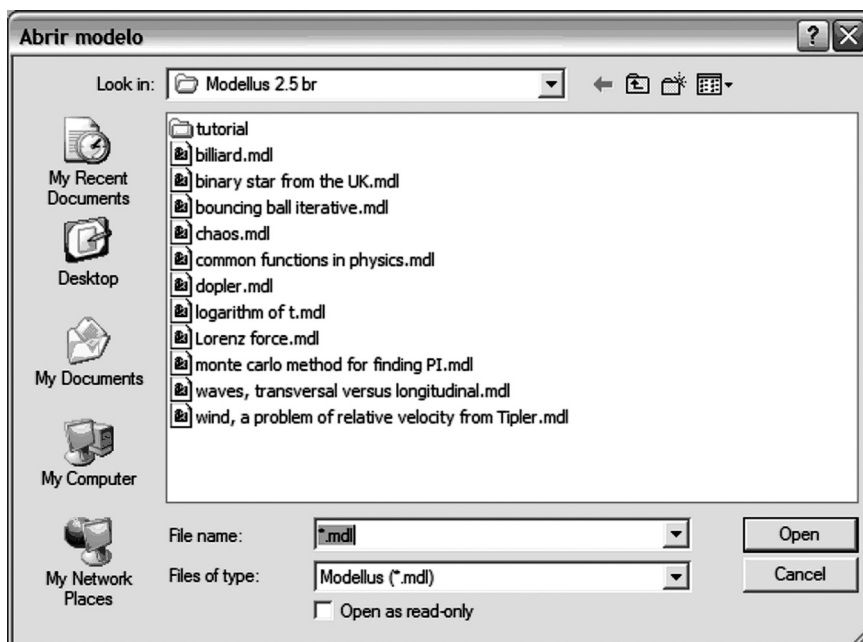


Figura 2.4: Abertura de simulações gravadas em arquivo.

Vá para a pasta *Tutorial* e escolha a primeira simulação, que trata do movimento acelerado de um carro. Após abri-la, o aspecto do *Modellus* ficará semelhante ao mostrado na Figura 2.5:

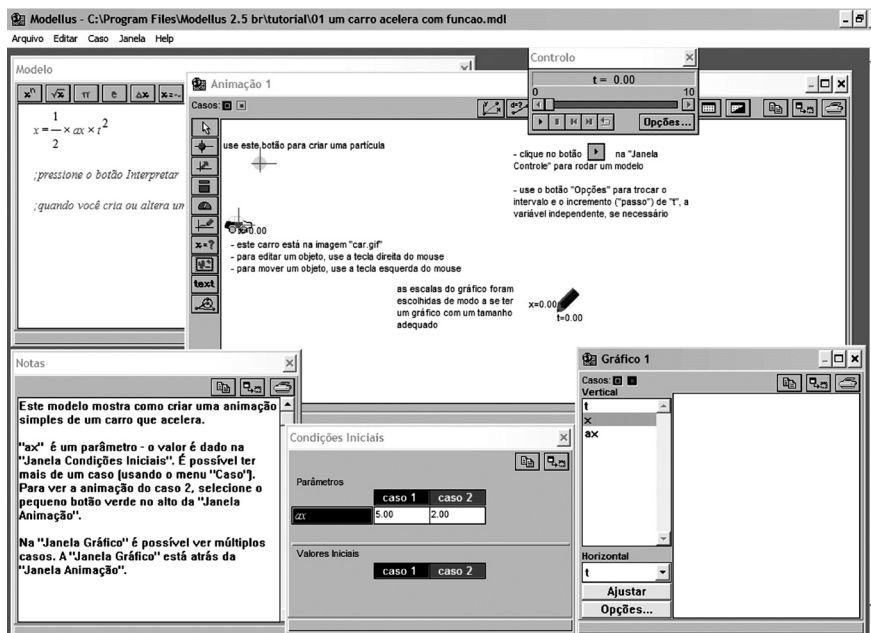


Figura 2.5: Simulação do movimento acelerado de um carro.

A simulação é executada pressionando-se o botão com a seta vermelha na janela *Controlo*. Ao fazer isso, observe que várias coisas acontecem. Na janela *Animação*, o carro começa a mover-se, e um gráfico da sua posição  $x$  em função do tempo  $t$  é desenhado. A função que descreve esse movimento está definida na janela *Modelo*, e pode-se ver que o carro tem aceleração constante  $ax$ . O valor de  $ax$  é dado na janela de *Condições Iniciais*, e dois casos estão definidos:  $ax = 5$  e  $ax = 2$ . A janela de animação está mostrando apenas o primeiro caso. Já a janela *Gráfico* apresenta o gráfico de  $x \times t$  para os dois casos. No alto das duas janelas há botões que permitem escolher quais casos serão mostrados (a janela de animação só permite um de cada vez). Para mais informações, veja os textos nas janelas de notas e de animação.

Após explorar este primeiro modelo, repita o procedimento com outras animações gravadas na pasta *Tutorial*. Note que, ao abrir uma nova simulação, o *Modellus* perguntará se você deseja salvar a que está carregada no momento. Responda *Não*; do contrário, você poderá modificar a simulação armazenada no computador. Ao rodar as próximas simulações, não se assuste se não compreender alguma coisa (mas faça um esforço!); nas aulas seguintes, você terá a oportunidade de estudar com calma o funcionamento do *Modellus*.

## **AJUDA NO MODELLUS**

O *Modellus* vem com um manual muito útil, que pode ser acessado diretamente da barra de menu, em *Help / Workshops e Help*. Aí você encontra informações sobre os diversos aspectos do programa, além de muitas atividades práticas. Dê uma olhada neste material e familiarize-se com ele: no futuro, ao usar o *Modellus*, você irá consultá-lo freqüentemente.

## INFORMAÇÃO SOBRE A PRÓXIMA AULA

Na próxima aula, iniciaremos um estudo mais detalhado do *Modellus*, mostrando como definir funções matemáticas e desenhar seus gráficos.

### LEITURAS RECOMENDADAS

A *Revista Brasileira de Ensino de Física* ([www.sbfisica.org.br/rbef](http://www.sbfisica.org.br/rbef) ou [www.scielo.br/rbef](http://www.scielo.br/rbef)) tem vários artigos interessantes sobre o *Modellus*:

1. E. A. Veit e V. D. Teodoro, *Modelagem no ensino-aprendizagem de Física e os Novos Parâmetros Curriculares Nacionais para o ensino médio*, *Revista Brasileira de Ensino de Física*, 2002, vol. 24, nº. 2, p. 87-96 (2002).
2. E. A. Veit, P. M. Mors e V. D. Teodoro, *Ilustrando a segunda lei de Newton no século XXI*, *Revista Brasileira de Ensino de Física*, vol. 24, nº. 2, p.176-184 (2002).
3. I. S. Araújo, E. A. Veit e M. A. Moreira, *Atividades de modelagem computacional no auxílio à interpretação de gráficos da cinemática*, *Revista Brasileira de Ensino de Física*, v. 26, nº. 2, p. 179-184 (2004).

Outra leitura recomendada é a tese de doutorado (em inglês) do autor do *Modellus*:

4. V. D. Teodoro, *Modellus: Learning Physics with Mathematical Modelling*, Tese de Doutorado, Universidade Nova de Lisboa (2003), disponível em <http://phoenix.sce.fct.unl.pt/modellus>

# Funções e Gráficos

AULA

# 3

## Meta da aula

Usar o *Modellus* para estudar funções matemáticas e seus gráficos.

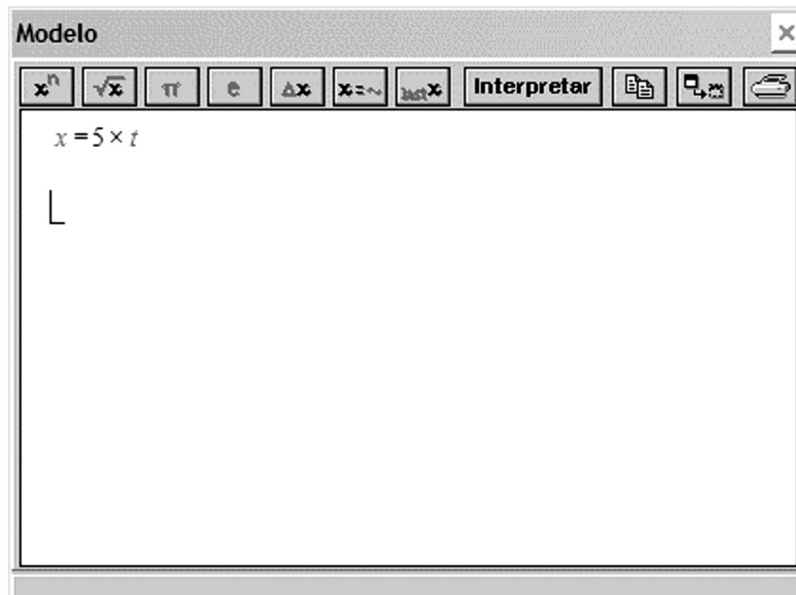
## objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- definir funções no *Modellus* e fazer seus gráficos;
- utilizar os recursos e opções da janela *Gráfico*;
- dar valores aos parâmetros das funções na janela *Condições Iniciais*;
- criar conjuntos alternativos de parâmetros (*Casos*);
- mudar o nome, limites e passo da variável independente;
- alterar a forma de apresentação dos números no *Modellus*;
- usar as funções definidas no *Modellus*, em particular, as trigonométricas;
- fazer múltiplos gráficos para casos diferentes da mesma função ou para funções distintas.

## GRÁFICOS

Gráficos de funções matemáticas são um bom ponto de partida para a aprendizagem do *Modellus*. Considere, por exemplo, uma partícula movendo-se com velocidade constante de 5 m/s. Seu deslocamento  $x$  após um tempo  $t$  é dado pela função  $x = 5t$ . Para representar este movimento em um gráfico, abra o *Modellus* e escreva a relação entre  $x$  e  $t$  na janela *Modelo*, como mostrado na **Figura 3.1**.



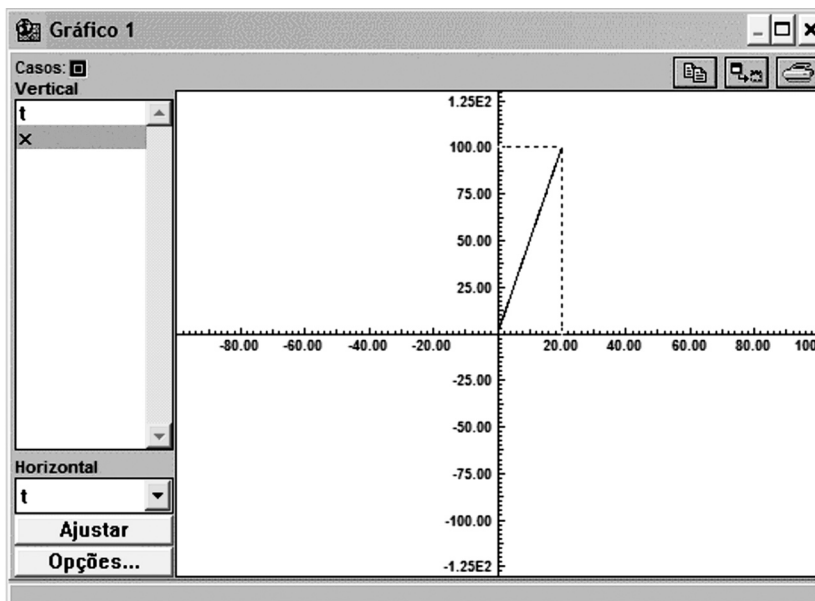
**Figura 3.1:** A janela *Modelo* com a definição da função  $x(t)$ .

O sinal de multiplicação ( $\times$ ) que aparece na expressão matemática da **Figura 3.1** pode ser inserido de duas maneiras: usando a tecla de espaço em branco ou o asterisco (\*).

O próximo passo é fazer o *Modellus* “ler e compreender” nossa função. Isto é feito clicando o botão *Interpretar*, que está no alto da janela *Modelo*. Se tudo der certo, ou seja, se o *Modellus* tiver entendido o que foi escrito, a mensagem “modelo interpretado” aparecerá na parte inferior da janela.

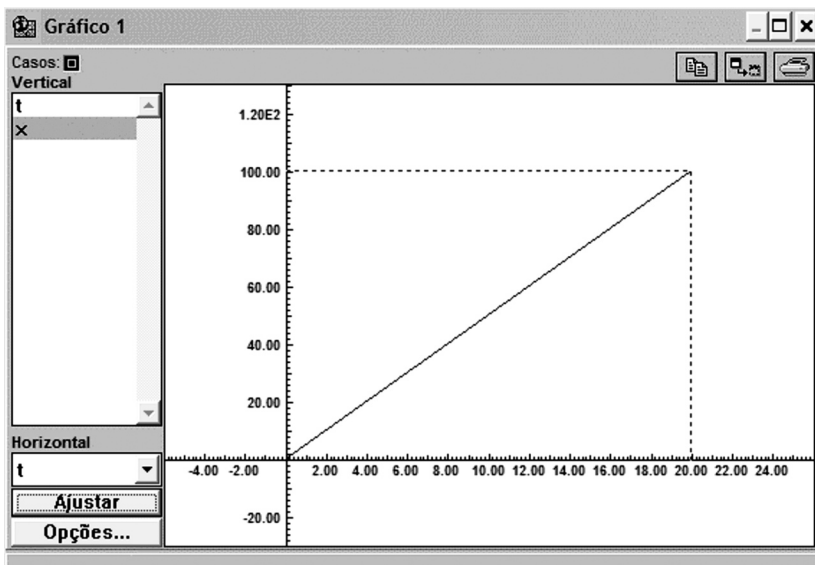
Ao lado da janela *Modelo*, deve estar outra janela, com o nome *Gráfico 1*. É nela que será traçado o gráfico de  $x(t)$ . Seu tamanho e posição podem ser modificados com o mouse, conforme vimos na aula anterior. Para começar o desenho do gráfico, vá para a janela *Controlo* e clique o botão de início (aquele com a seta vermelha). Com isso,  $t$  começa a variar, indo gradativamente de 0 até 20. Esses limites são predeterminados

pelo programa; mais à frente veremos como eles podem ser mudados. À medida que  $t$  aumenta, o gráfico de  $x(t)$  vai sendo desenhado na janela *Gráfico 1*. O resultado final está mostrado na *Figura 3.2*.



**Figura 3.2:** Gráfico da função  $x(t)$ .

Podemos ver, na *Figura 3.2*, que as escalas escolhidas pelo *Modellus* para fazer o gráfico não são muito boas, pois a linha traçada ocupa apenas uma parte pequena da área disponível. Você pode acertar automaticamente as escalas clicando o botão *Ajustar*, que está ao lado da área do gráfico. O resultado está mostrado na *Figura 3.3* – note como o gráfico ficou bem melhor distribuído.



**Figura 3.3:** Gráfico de  $x(t)$  após o ajuste automático de escalas com o botão *Ajustar*.

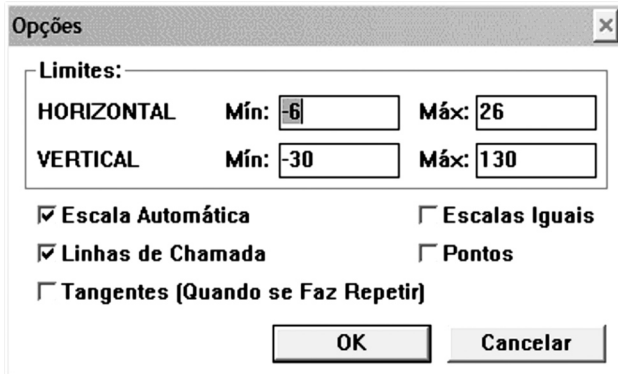


Figura 3.4: Opções da janela *Gráfico*.

O botão *Opções*, logo abaixo do botão *Ajustar*, permite fazer outras mudanças no gráfico. Ao clicá-lo, abre-se uma caixa de diálogo como a mostrada na **Figura 3.4**. Nela, você pode, por exemplo, determinar se as linhas de chamada (as linhas pontilhadas na **Figura 3.3**) serão exibidas ou não. Ou fazer com que as escalas horizontais e verticais sejam iguais. Ou ainda alterar a área do gráfico que é mostrada, mudando os limites dos eixos horizontal e vertical. Mais detalhes sobre os outros itens na caixa de opções podem ser encontrados no material de ajuda (na barra de menu, clique em *Help* → *Workshops e Help* → *Janela Gráfico*).

## PARÂMETROS

Podemos escrever a equação de movimento da nossa partícula de uma forma genérica, como  $x = x_0 + v t$ , onde  $x_0$  é a posição em  $t = 0$  e  $v$  é a velocidade. Escreva esta relação na janela *Modelo*, no lugar da fórmula anterior  $x = 5 t$ . O resultado deve ser algo como o que está na **Figura 3.5**.

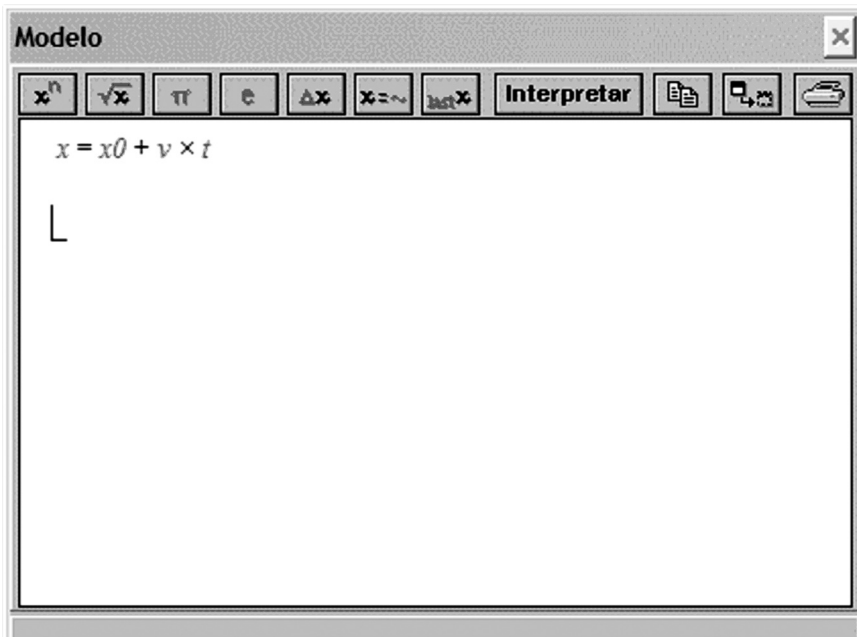
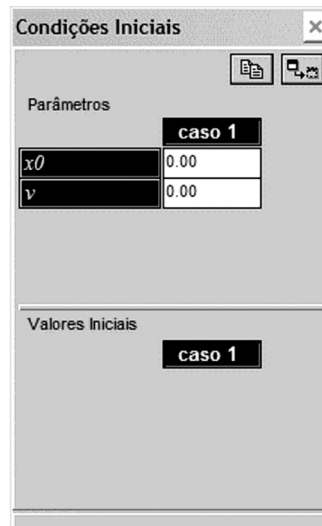
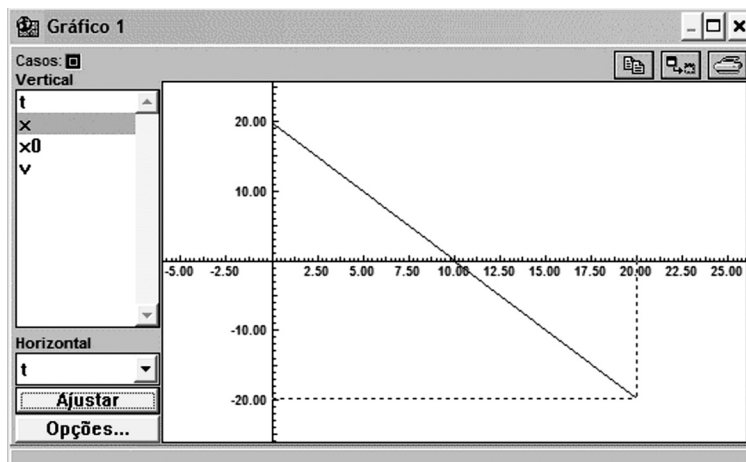


Figura 3.5: Modelo com parâmetros não especificados  $x_0$  e  $v$ .

Aperte agora o botão *Interpretar*. Observe que uma nova janela, intitulada *Condições Iniciais*, é criada pelo *Modellus*. O aspecto dessa janela está mostrado na **Figura 3.6**. Nela, podemos especificar os parâmetros  $x_0$  e  $v$  (note que, no início, todos os valores são 0). Coloque, por exemplo,  $x_0 = 20$  e  $v = -2$  nas caixas correspondentes e trace o gráfico (lembre-se: janela *Controlo*, botão com a seta vermelha). O resultado está mostrado na **Figura 3.7**.



**Figura 3.6:** Janela *Condições Iniciais*, onde parâmetros podem ser especificados.



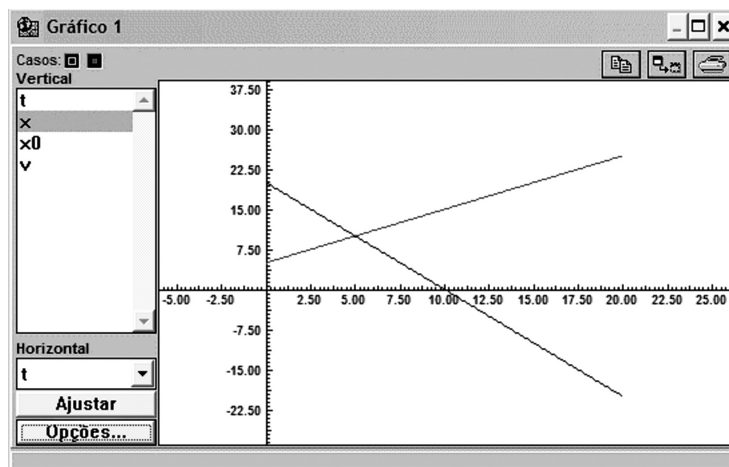
**Figura 3.7:** Gráfico da função  $x = x_0 + v t$ , com os parâmetros  $x_0 = 20$  e  $v = -2$ .

## CASOS

Uma das vantagens de se usar a janela *Condições Iniciais* é que não precisamos reinterpretar o modelo a cada mudança em um parâmetro. Outra vantagem é que podemos criar várias instâncias do modelo, com valores diferentes dos parâmetros. Essas instâncias são chamadas *Casos* pelo *Modellus*. Observe que os valores que já escolhemos,  $x_0 = 20$  e  $v = -2$ , estão identificados como *caso 1* na janela *Condições Iniciais*. Para criar um novo caso, vá para a barra de menu e clique *Caso/Adicionar*. Uma nova coluna, marcada como *caso 2*, aparece na janela *Condições Iniciais*. Inicialmente, os parâmetros do *caso 1* são colocados automaticamente no *caso 2*. Mude os valores do novo caso para, por exemplo,  $x_0 = 5$  e  $v = 1$ . Note também que a janela *Gráfico 1* foi modificada: no alto, à esquerda, onde está escrito “Casos”, existem agora duas caixinhas (antes, só havia uma). Se a primeira está marcada, o gráfico do *caso 1* é feito. Escolhendo a segunda, o *caso 2* é desenhado. Marque as duas caixas, de modo que os dois casos apareçam simultaneamente no gráfico, e inicie o desenho. O resultado deve ser parecido com a **Figura 3.8**.



Figura 3.8: Gráfico para dois conjuntos diferentes de parâmetros (casos). Note que optamos por não exibir as linhas de chamada.



Desmarque uma das caixas de caso na janela *Gráfico 1* e veja como a linha correspondente desaparece. Se você prestar atenção, verá que cada caso é identificado por uma cor, que é a mesma nas colunas da janela *Condição Inicial*, nas caixinhas de *Gráfico 1* e nas linhas que são desenhadas. Para praticar, crie mais um caso (com parâmetros de sua escolha) e faça um gráfico de tudo. Até cinco casos podem ser definidos em um modelo. Se desejar apagar o último caso criado, vá à barra de menu e clique *Caso/Remover o Último*.

## MOVIMENTO COM ACELERAÇÃO CONSTANTE

Podemos generalizar o modelo descrito na seção anterior e tratar o movimento uniformemente acelerado. A *Figura 3.9* mostra como a equação de movimento é escrita na janela *Modelo*.

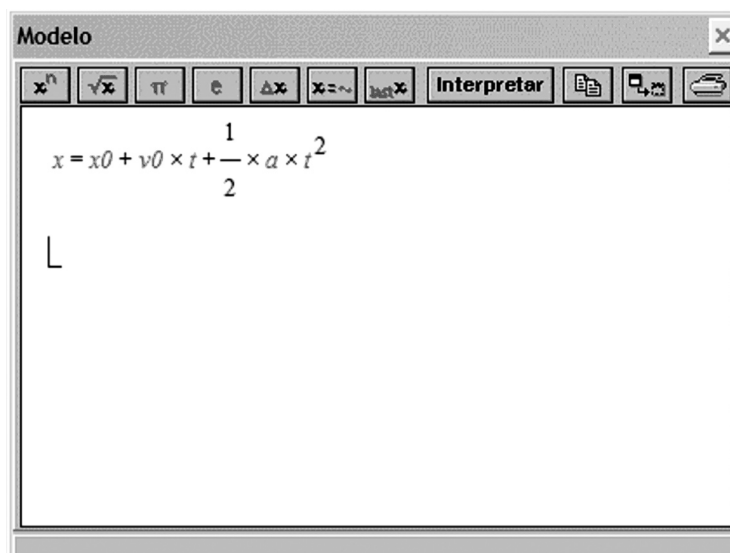


Figura 3.9: Modelo do movimento com aceleração constante.

A fração que aparece na fórmula da Figura 3.9 é obtida escrevendo-se  $1/2$ ; o *Modellus* coloca automaticamente o numerador sobre o denominador. Para escrever o expoente 2 em  $t^2$ , pode-se teclar  $t^2$  ou usar o botão  $x^n$ , que está no alto da janela *Modelo*. Faça gráficos de  $x(t)$  para diferentes casos dos parâmetros  $x_0$ ,  $v_0$  e  $a$ . Por exemplo, na Figura 3.10, estão os resultados para dois casos: em ambos  $x_0 = 0$  e  $a = -10$ , mas  $v_0 = 90$ , no caso 1, e  $v_0 = 70$ , no caso 2.

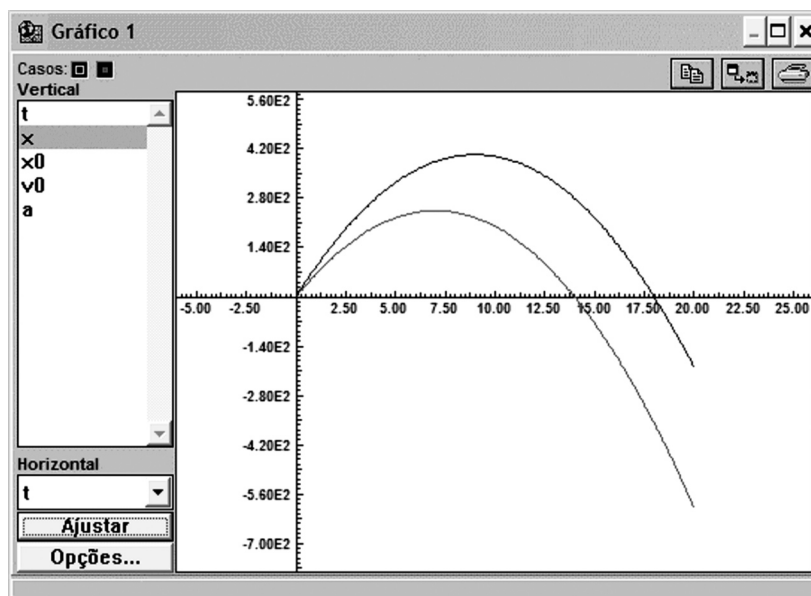


Figura 3.10: Dois casos de movimento uniformemente acelerado.

## ZOOM

É possível usar o mouse para ampliar uma região qualquer do gráfico. Pressione o botão esquerdo do mouse e marque a área de interesse com um retângulo – você verá que o gráfico é redesenhado automaticamente, mostrando apenas o que está dentro do retângulo. Isto é equivalente a redefinir os limites do gráfico com o botão *Opções*. Outro truque útil com o mouse é clicar duas vezes sobre um ponto na área do gráfico – isto coloca a origem dos eixos sobre este ponto. O botão *Ajustar* desfaz todas essas mudanças, desenhando novamente o gráfico completo na tela. Tente produzir algo como a Figura 3.11, que mostra um “zoom” sobre a parte superior ( $x > 0$ ) do gráfico na Figura 3.10.

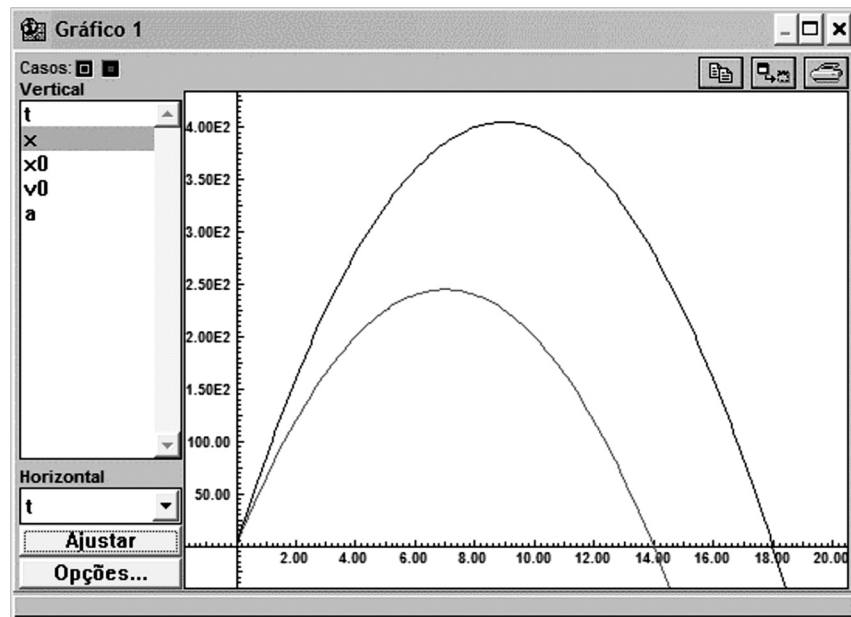
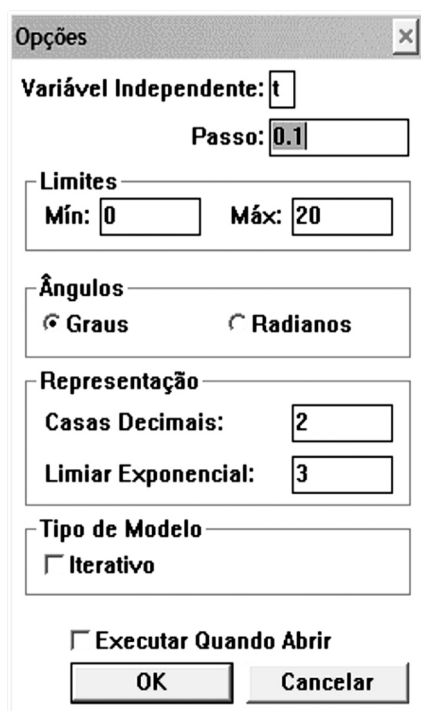


Figura 3.11: Ampliação de parte do gráfico da Figura 3.10.

## FUNÇÕES TRIGONOMÉTRICAS

As funções trigonométricas básicas estão definidas no *Modellus*. Por exemplo, para fazer o gráfico da função seno entre 0 e 360 graus, escreva  $y = \sin(t)$  na janela *Modelo* e aperte o botão *Interpretar*. Note que, para o *Modellus*, a grafia da função é *sin*, e não *sen*. Antes de desenhar o gráfico, temos que mudar os limites da variável independente  $t$ ; do contrário, só veremos o trecho entre 0 e 20 graus. Para fazer o ajuste, aperte o botão *Opções* da janela *Controlo*, que abre uma caixa de diálogo como a mostrada na Figura 3.12. Bem no alto da caixa está o nome da variável independente, que o *Modellus* define inicialmente como  $t$ . Qualquer outro nome (com uma letra) mais apropriado para o seu modelo pode ser colocado ali:  $x$ ,  $z$ , etc. Mais abaixo estão os limites da variável independente e o tamanho dos passos com que ela vai do mínimo ao máximo. Como padrão inicial, o *Modellus* define que  $t$  vai de 0 a 20 em passos de 0.1 – ou seja, em 200 passos. Mude o limite máximo para 360 e o passo para 1.0 (com 0.1, o programa dará 3.600 passos, o que pode demorar um pouco), e feche a caixa de diálogo clicando o botão OK. Rode a simulação e, se tudo der certo, você verá a curva da função seno ser desenhada na janela *Gráfico* (não se esqueça de usar o botão *Ajustar*).



Há uma opção importante quando tratamos de funções trigonométricas: como são medidos os ângulos? Na Figura 3.12, logo abaixo das informações sobre a variável independente, vê-se que é possível escolher se os argumentos das funções trigonométricas serão dados em graus ou radianos. Quando é iniciado, o *Modellus* determina que os ângulos serão lidos em graus. Se desejamos trabalhar com radianos, temos que marcar a opção correspondente na caixa mostrada na Figura 3.12. Em geral, este é o caso quando tratamos de fenômenos periódicos – relações usuais como  $\omega = 2\pi/T$ , entre período e frequência angular, são válidas apenas se os ângulos (mais exatamente, fases) estão em radianos.

Figura 3.12: Opções da janela *Controlo*.

Outra opção mostrada na Figura 3.12 diz respeito a como os números são apresentados pelo *Modellus*. Por exemplo, você pode escolher com quantas casas decimais os números serão mostrados em todo o programa. O “limiar exponencial” indica quão grande (ou pequeno) um número deve ser para que a notação exponencial seja usada (4.20E4 no lugar de 42000, ou 1.03E-2 em vez de 0.00103). Troque os valores definidos pelo *Modellus* para “casas decimais” = 1 e “limiar exponencial” = 4, e veja como fica a numeração dos eixos no seu gráfico da função seno.

## BATIMENTOS

A superposição de duas oscilações de frequências semelhantes costuma gerar “batimentos”, um fenômeno importante em diversas áreas da física e útil em muitas aplicações tecnológicas. Podemos estudar os batimentos criando o modelo mostrado na Figura 3.13.

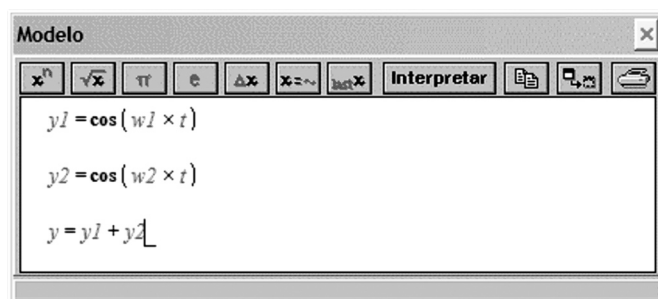
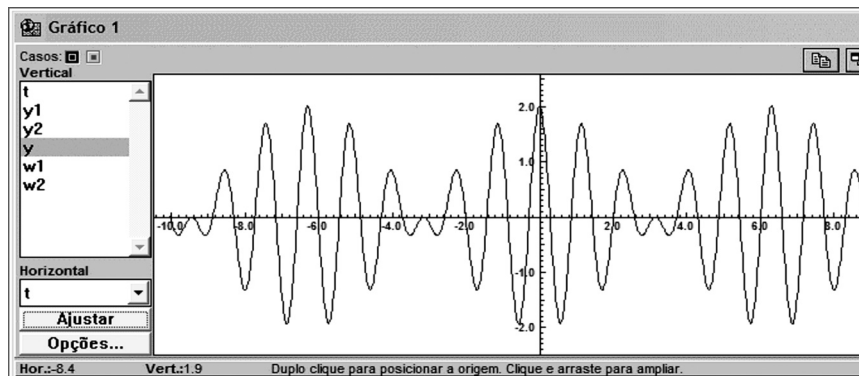


Figura 3.13: Modelo de superposição de oscilações.

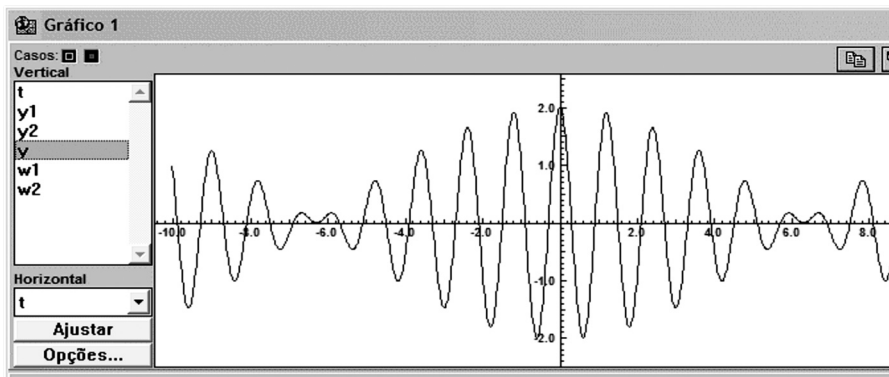
Na janela *Condições Iniciais*, defina as freqüências angulares  $w1 = 5$  e  $w2 = 6$ , e rode a simulação. O resultado deveria ficar parecido com o que está na **Figura 3.14**, mas ainda faltam alguns ajustes. Primeiro, escolha *radianos* como a medida de ângulos (veja a discussão anterior). Depois, note que queremos fazer o gráfico de  $y$ , a soma das oscilações  $y1$  e  $y2$ . Portanto, marque esta variável no painel ao lado do gráfico, como está indicado na **Figura 3.14**. Observe ainda que  $t$  vai de -10 a 10 no gráfico. Também fizemos o passo de  $t$  igual a 0.02, para obter curvas mais suaves. Faça estas mudanças e tente reproduzir o que está na figura.

Errata:  
Na legenda da Figura 3.14, o correto é  $w2 = 6$ .



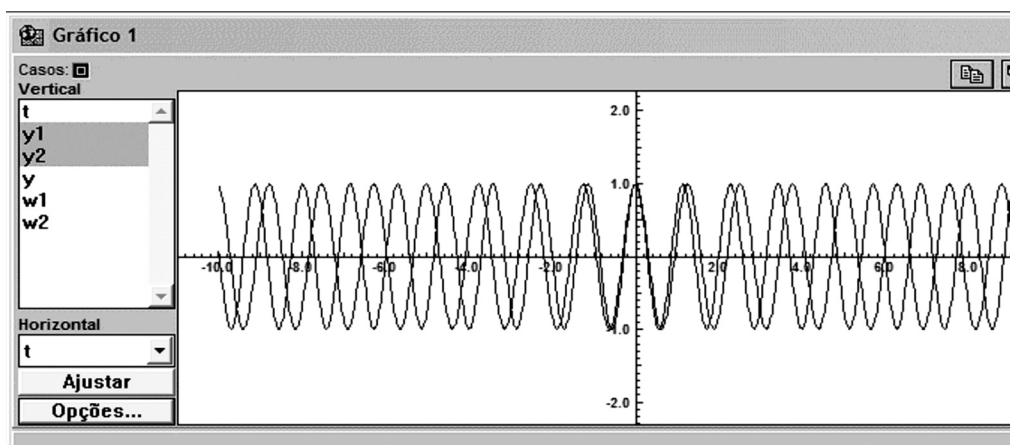
**Figura 3.14:** Superposição de oscilações com  $w1 = 5$  e  $w2 = 5$ .

Se tudo funcionou bem, você terá observado que, embora as duas oscilações sejam semelhantes, sua soma não resulta em nada parecido com elas: surgem modulações, os “batimentos”, que fazem com que a amplitude total aumente e diminua periodicamente. O período desta modulação é inversamente proporcional à diferença de freqüências  $w2 - w1$ . Para ver isso, mude o valor de  $w2$  para 5.5, reduzindo a diferença de freqüências à metade do que estava antes. Ao refazer o gráfico, você vai observar que o período da modulação dobrou, como está mostrado na **Figura 3.15**.



**Figura 3.15:** Batimentos obtidos com  $w1 = 5$  e  $w2 = 5.5$ .

O motivo para o aparecimento dos batimentos é simples. Em  $t = 0$ , as duas oscilações estão em fase e interferem construtivamente. Como as frequências são diferentes, à medida que o tempo passa, as oscilações vão ficando fora de fase, até que a interferência se torna destrutiva e uma cancela a outra. É assim que aparecem os “nós” e “barrigas” no gráfico de  $y(t)$ . Podemos utilizar o *Modellus* para verificar se esta explicação é correta, superpondo os gráficos de  $y_1(t)$  e  $y_2(t)$ , como está na **Figura 3.16**. Para fazer esses gráficos, temos que marcar as duas variáveis  $y_1$  e  $y_2$  no painel da esquerda (veja a **Figura 3.16**). Uma forma de realizar isso é marcar primeiro  $y_1$  e, depois, mantendo apertada a tecla *Ctrl*, marcar também  $y_2$ . (O *Ctrl* evita que a primeira seja desmarcada quando a segunda for marcada.)



**Figura 3.16:** As oscilações que formam o sinal da **Figura 3.15**.

Os resultados do *Modellus* mostram que nossa explicação dos batimentos é válida: note como em  $t = 0$  as oscilações estão em fase, reforçando-se quando somadas, e vão saindo de fase até cancelarem-se em torno de  $t = 6$  e  $-6$ , os mesmos pontos onde aparecem os “nós” na **Figura 3.16**.

## FIGURAS DE LISSAJOUS

Os batimentos aparecem quando somamos oscilações semelhantes, que têm a mesma “direção”. Resultados interessantes também surgem quando as oscilações estão em direções perpendiculares. A **Figura 3.17** mostra o que acontece com as mesmas oscilações usadas para gerar

as Figuras 3.15 e 3.16, se colocamos  $y_1(t)$  no eixo vertical e  $y_2(t)$  no horizontal. Note que isso é feito marcando  $y_1$  no painel intitulado *Vertical* (o que temos usado até agora), e  $y_2$  no painel *Horizontal*, que até agora mostrava sempre a variável independente  $t$ . Usando a seta ao lado deste painel, você verá que é possível colocar no eixo horizontal qualquer variável definida no modelo, e não apenas  $t$ .

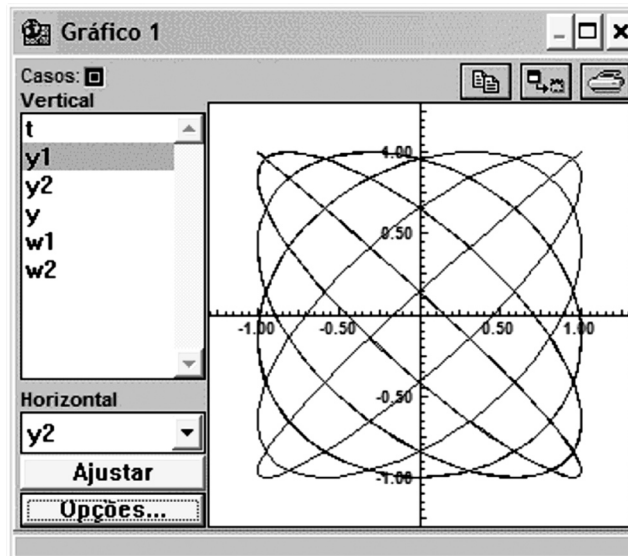


Figura 3.17: Figura de Lissajous.

O padrão obtido na Figura 3.17 é uma “figura de Lissajous”. A forma dessas figuras depende da razão entre as frequências das oscilações ( $w_2/w_1 = 1.1$  na figura mostrada) e da fase entre elas. Por exemplo, veja o que ocorre com  $w_1 = 3$  e  $w_2 = 2$ . Para ver o efeito da fase, temos que modificar um pouco o modelo. A Figura 3.18 mostra como aumentar a fase de  $y_2(t)$  em  $\pi/4$ . Observe que há um botão na janela *Modelo* que insere o número  $\pi$  nas fórmulas, de modo que não precisamos escrever 3.14159... .

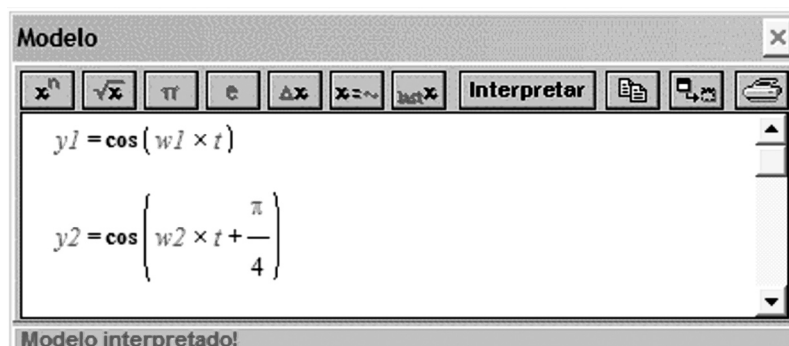


Figura 3.18: Modelo para figuras de Lissajous com defasagem  $\pi/4$ .

Rode o modelo, usando  $w_1 = 3$  e  $w_2 = 2$ . O resultado está na Figura 3.19. Compare com o que você obteve antes para as mesmas frequências.

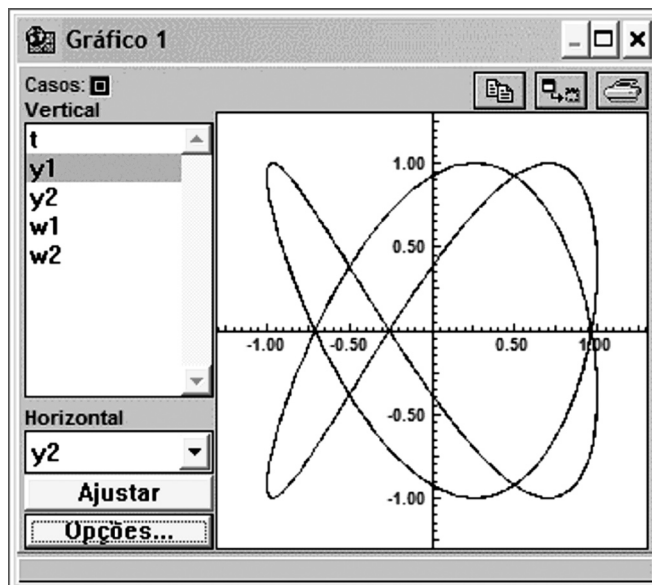


Figura 3.19: Figura de Lissajous com  $w_1/w_2 = 3/2$  e defasagem  $\pi/4$ .

Como exercício, modifique seu modelo, fazendo da defasagem um parâmetro a ser determinado na janela *Condições Iniciais*. Trace figuras de Lissajous para diferentes razões de frequência e defasagens.



### ATIVIDADE

#### 1. Função exponencial e logaritmo

O *Modellus* calcula muitas outras funções, além das trigonométricas – por exemplo, a função exponencial e o logaritmo. Faça um gráfico da exponencial  $e^t$ , usando o modelo da Figura 3.20.



Figura 3.20: Função exponencial.



Observe que existem várias maneiras de se escrever  $e^t$  no *Modellus*. Você pode apertar o botão  $e$  na janela *Modelo* (ao lado do  $\pi$ ), em seguida apertar o botão  $x^n$ ,  $e$ , por fim, a tecla  $t$ . Ou pode simplesmente escrever “ $e^t$ ” no teclado – a letra  $e$  no *Modellus* representa o número neperiano  $e = 2.71818\dots$  (por isso, nenhuma variável pode ter o nome  $e$ ). Se o expoente for uma expressão matemática, ela deve ficar entre parênteses. Por exemplo,  $e^{a+b}$  deve ser escrito como  $e^{(a+b)}$  ou “ $e^{(a+b)}$ ” – sem os parênteses, o *Modellus* entenderá isso como  $e^{a+b}$ .

Faça também gráficos da função logarítmica. O *Modellus* calcula logaritmos em duas bases, a decimal (função *log*) e a natural (função *ln*).

## ATIVIDADE



### 2. Outras funções matemáticas

Faça gráficos das seguintes funções, nos intervalos determinados:

- $\sqrt{t}$ ,  $0 < t < 20$  (a raiz quadrada é criada com um dos botões da janela *Modelo*);
- $\sqrt{1-t^2}$ ,  $-1 < t < 1$ ;
- $\sin^2(t)$ ,  $0 < t < 360^\circ$  (escreva  $\sin(t)^2$  no *Modellus*);
- $\cos^2(t)$ ,  $0 < t < 360^\circ$ ;
- $\sin^2(t) + \cos^2(t)$ ,  $0 < t < 360^\circ$ ;
- $\arctan(t)$ ,  $-10 < t < 10$ ;
- $\text{abs}(t)$ ,  $-1 < t < 1$  (esta função dá o valor absoluto de  $t$ );
- $\text{sign}(t)$ ,  $-1 < t < 1$  (o sinal de  $t$ );
- $\text{int}(t)$ ,  $0 < t < 10$  (a parte inteira de  $t$ );
- $\text{abs}(\sin(t))$ ,  $0 < t < 720^\circ$ ;
- $\text{sign}(\cos(t))$ ,  $-720^\circ < t < 720^\circ$ ;

Uma lista das funções predefinidas no *Modelus* está em *Help/Workshops* e *Help/Sintaxe*. Faça mais alguns gráficos, usando essas funções.

## INFORMAÇÕES SOBRE A PRÓXIMA AULA

Nosso estudo do *Modellus* continua na próxima aula, onde veremos como o programa calcula derivadas de funções e aprenderemos a usar a janela de animações. Esses dois recursos serão utilizados para tratar problemas de cinemática.

# Cinemática

AULA

# 4

## Meta da aula

Usar as ferramentas de cálculo e animação do *Modellus* para estudar cinemática.

# objetivos

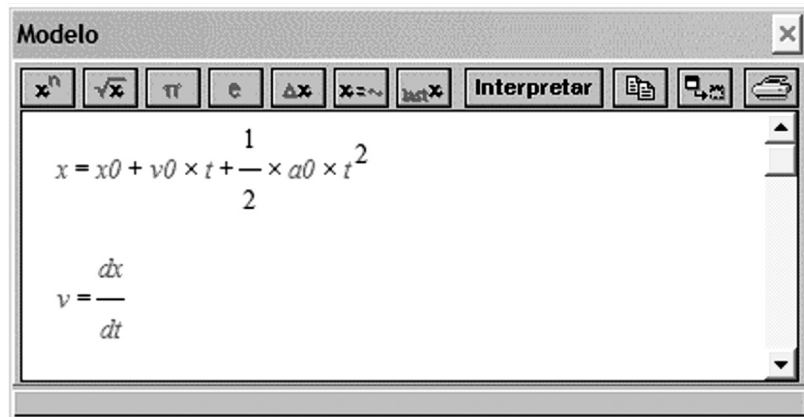
Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- calcular taxas de variação (derivadas) de funções definidas no *Modellus*;
- aplicar o cálculo de derivadas ao estudo de cinemática;
- empregar conceitos de cálculo diferencial no Ensino Médio;
- construir animações com o *Modellus*;
- usar animações para ilustrar conceitos de cinemática vetorial.

## VELOCIDADE E ACELERAÇÃO

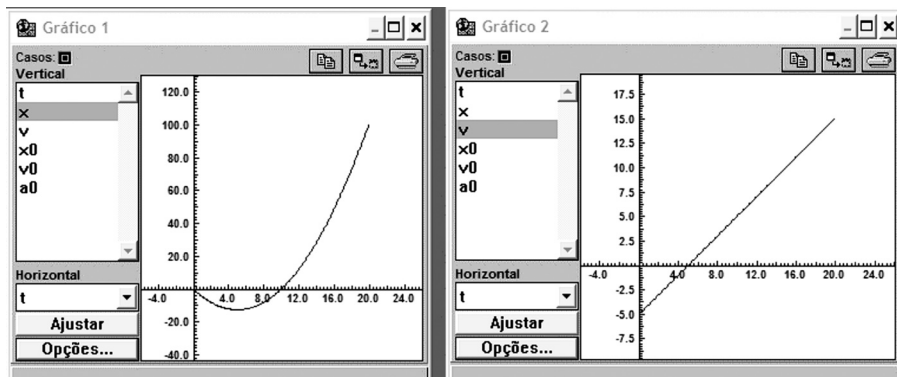
Velocidade e aceleração são conceitos fundamentais da Mecânica. São também difíceis de entender e aplicar, principalmente para os alunos de cursos introdutórios. Neste capítulo, vamos ver como o *Modellus* pode ser útil ao ensino e aprendizagem de cinemática e, com isso, estudaremos mais alguns recursos importantes do programa, como o cálculo de derivadas e as animações.

Para começar, crie o modelo mostrado na **Figura 4.1**. A primeira linha define como a posição  $x$  de uma partícula depende do tempo  $t$ . A segunda linha calcula a velocidade  $v$  dessa partícula. Note que isto foi feito com a definição de velocidade instantânea,  $v = dx/dt$ , e não escrevendo uma função específica para  $v(t)$ .



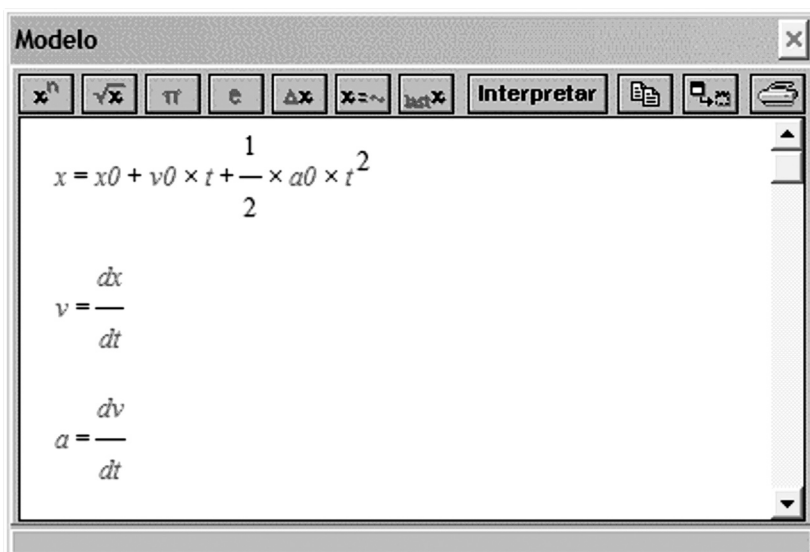
**Figura 4.1:** Cálculo da velocidade  $v$  a partir da posição  $x(t)$ .

Interprete o modelo definido na figura, dê valores aos parâmetros  $x_0$ ,  $v_0$  e  $a_0$ , e faça gráficos de  $x(t)$  e  $v(t)$ . A **Figura 4.2** mostra o que ocorre para  $x_0 = 0$ ,  $v_0 = -5$  e  $a = 1$ . Note que é conveniente fazer os gráficos em janelas separadas, dada a diferença de escalas.



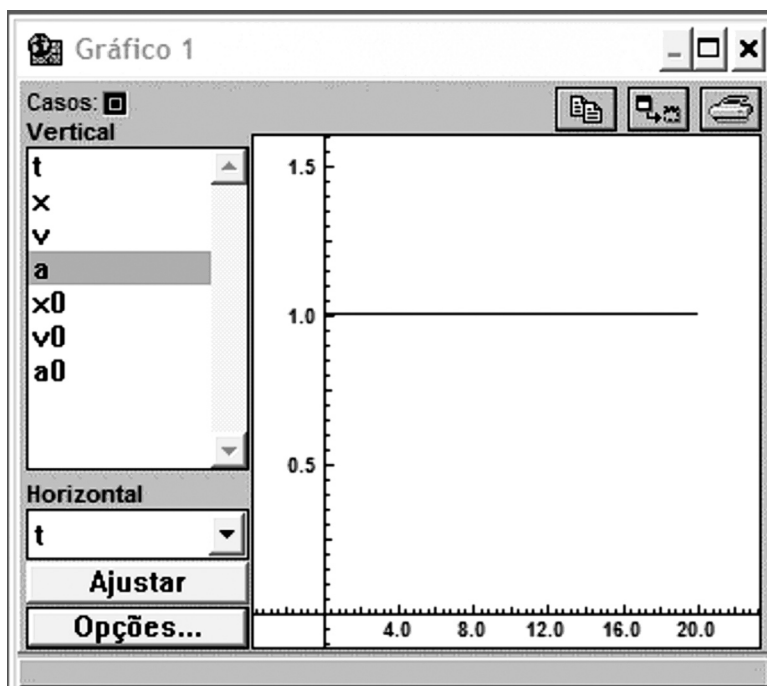
**Figura 4.2:** Gráficos da posição  $x(t)$  (à esquerda) e velocidade  $v(t)$  (à direita).

O resultado mostra que a velocidade aumenta linearmente com o tempo, ou seja, o movimento é uniformemente acelerado (como você já deve ter suspeitado). A aceleração da partícula pode ser calculada da mesma forma que a velocidade – basta usar a definição de aceleração instantânea,  $a = dv/dt$ . O modelo fica como o que está na **Figura 4.3**.



**Figura 4.3:** Definição da velocidade e aceleração a partir de  $x(t)$ .

Faça o gráfico da aceleração *vs.* tempo. Com os parâmetros usados anteriormente, o resultado fica como o mostrado na **Figura 4.4**. O valor constante  $a = 1$  não deve ser surpresa (esperamos!).

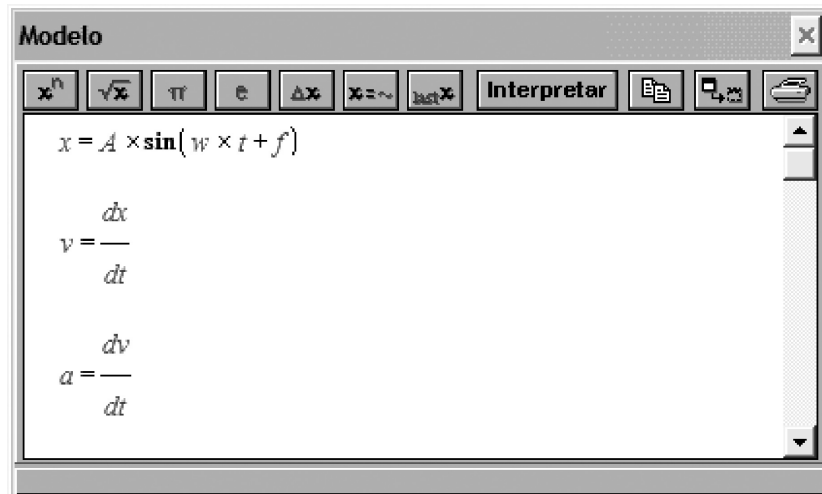


**Figura 4.4:** Gráfico da aceleração  $a(t)$ .

## MOVIMENTO HARMÔNICO SIMPLES

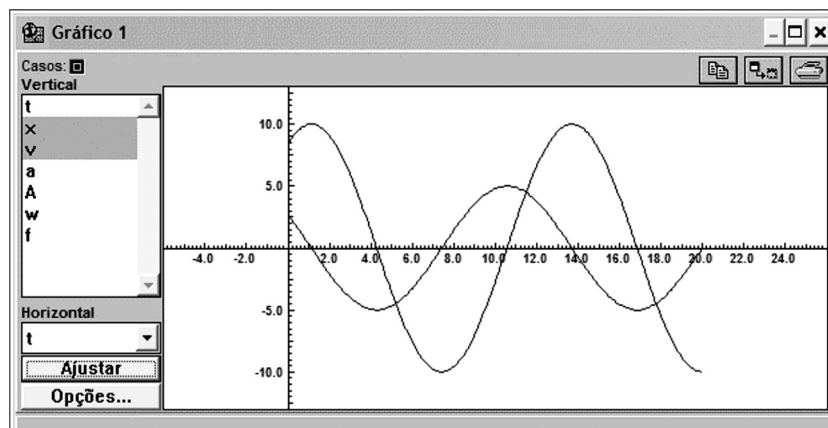
O cálculo da velocidade e aceleração a partir de  $x(t)$  pode ser realizado em situações menos triviais que a explorada na seção anterior. Um caso de interesse é o movimento harmônico simples, dado por  $x(t) = A \sin(\omega t + \varphi)$ , onde a frequência angular  $\omega$ , a amplitude  $A$  e a fase inicial  $\varphi$  são constantes. A Figura 4.5 mostra como calcular a velocidade e aceleração desse movimento. Compare com o modelo anterior e veja que mudamos apenas a primeira linha, onde a função  $x(t)$  é definida.

Figura 4.5: Cálculo da velocidade e aceleração no movimento harmônico.



Escolha valores para os parâmetros  $A$ ,  $\varphi$ ,  $\omega$  e trace os gráficos de  $x(t)$ ,  $v(t)$  e  $a(t)$ . Não se esqueça de marcar *Radianos* na opção de ângulos. As figuras a seguir mostram o resultado para  $A = 10$ ,  $\varphi = 1$ ,  $\omega = 0,5$ . Na Figura 4.6, temos  $x(t)$  (a curva de maior amplitude) e  $v(t)$  (a de menor amplitude). Vemos que a velocidade oscila com a mesma frequência do deslocamento, mas defasada de um quarto de período – a velocidade é zero quando o deslocamento é máximo, e máxima quando o deslocamento é zero.

Figura 4.6: Gráficos de  $x(t)$  e  $v(t)$  para o movimento harmônico.



A Figura 4.7 mostra os gráficos de  $x(t)$  (novamente a curva mais alta) e  $a(t)$  (a curva mais baixa). Note que a aceleração oscila com a mesma frequência do deslocamento, e que a defasagem entre  $x(t)$  e  $a(t)$  é de meio período.

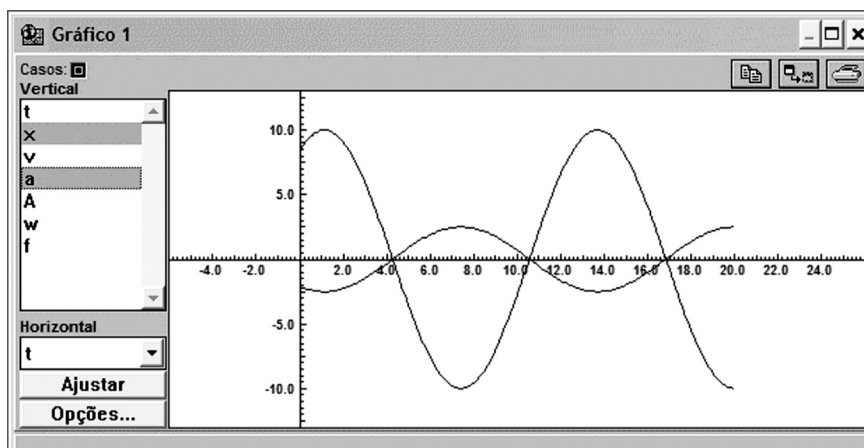


Figura 4.7: Gráficos de  $x(t)$  e  $a(t)$  para o movimento harmônico.

## DERIVADA NO ENSINO MÉDIO?

Pode parecer estranho que o *Modellus*, uma ferramenta de ensino-aprendizagem desenvolvida principalmente para a escola média, tenha recursos de cálculo diferencial como os que vimos anteriormente. Mas é importante lembrar que, embora derivadas e integrais não sejam ensinadas nas escolas médias brasileiras, muitos conceitos associados ao cálculo infinitesimal são tratados na disciplina de Física. O exemplo mais familiar é o de velocidade instantânea. A abordagem tradicional começa com a discussão de velocidade média,

$$v_{\text{média}} = \frac{\Delta x}{\Delta t},$$

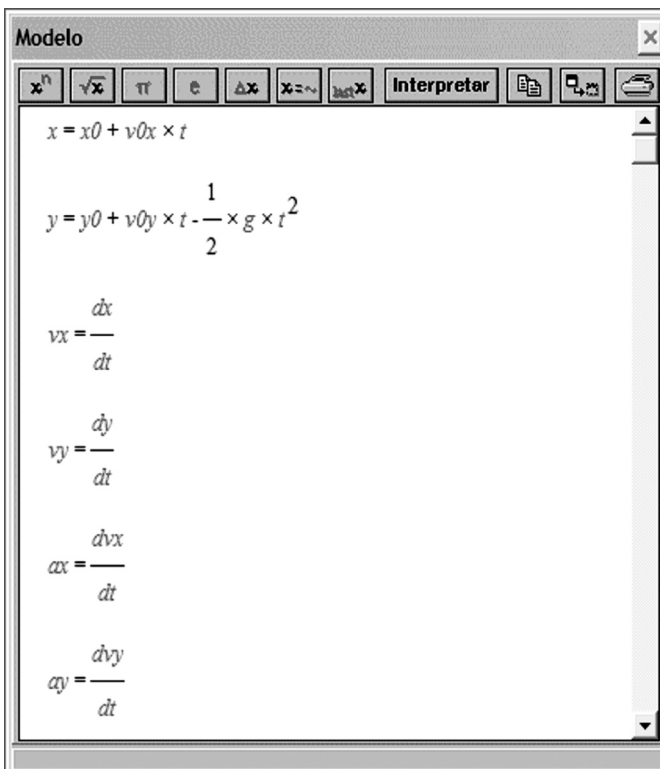
e segue com a definição de velocidade instantânea como sendo a velocidade média no limite em que o intervalo de tempo  $\Delta t$  e, conseqüentemente, o deslocamento  $\Delta x$  tornam-se muito pequenos. Quem já aprendeu cálculo diferencial reconhece imediatamente que a velocidade instantânea é a derivada de  $x$  em relação a  $t$ ,

$$v = \frac{dx}{dt},$$

e pode usar o que sabe sobre diferenciação para aplicar o conceito de velocidade em diferentes situações. Como os alunos da escola média não têm idéia do que seja uma derivada, a discussão sobre velocidade instantânea costuma terminar logo após a definição, e praticamente nenhum uso desse conceito é feito daí em diante. Já vimos que o *Modellus* permite dar alguns passos novos – o conceito de velocidade instantânea pode ser explorado e aplicado sem que os alunos tenham conhecimento de cálculo. Note que não é necessário dar o nome de derivada ao  $dx/dt$  que se escreve na janela *Modelo*. Pode-se, por exemplo, dizer que  $dt$  é um intervalo de tempo  $\Delta t$  muito pequeno, e  $dx$  um deslocamento  $\Delta x$  muito pequeno (e que o uso de  $d$  no lugar de  $\Delta$  serve para lembrar isso). Assim, mesmo alguém não familiarizado com o cálculo diferencial pode entender o conceito de velocidade instantânea e utilizá-lo em muitas aplicações.

## ANIMAÇÕES E O MOVIMENTO EM DUAS DIMENSÕES

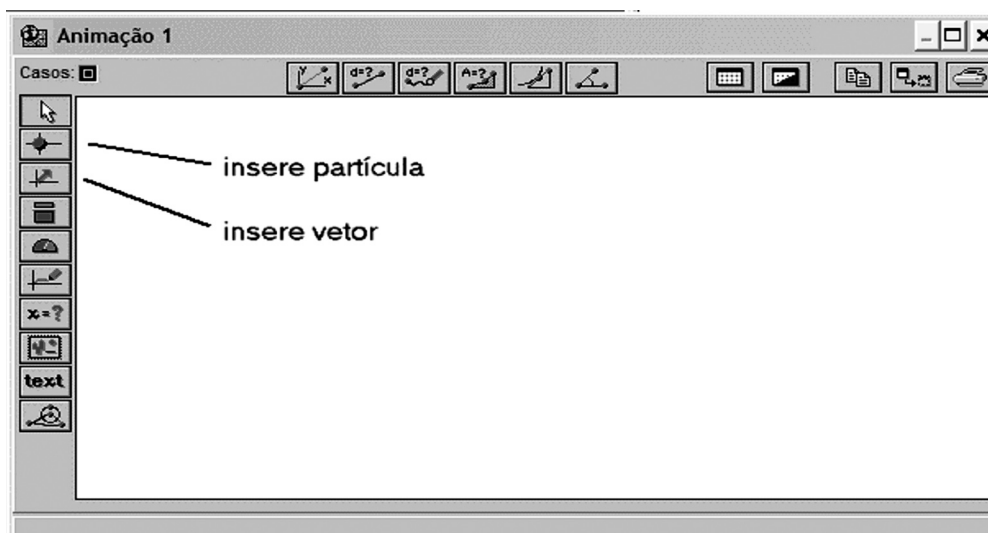
Velocidade e aceleração são vetores, o que cria ainda mais dificuldades ao aprendizado desses conceitos. Por exemplo, a possibilidade de aceleração e velocidade apontarem em direções diferentes é fonte de muita confusão em cursos introdutórios. Esse e outros problemas podem ser abordados com as ferramentas de animação do *Modellus*. Para ver como isso é feito, vamos criar um modelo para o movimento de um projétil em duas dimensões,  $x$  e  $y$ , com a aceleração da gravidade na direção  $y$ . Esta é uma generalização simples do modelo unidimensional de que tratamos anteriormente. A **Figura 4.8** mostra como definir cada componente da posição, velocidade e aceleração.



**Figura 4.8:** Cinemática do movimento bidimensional uniformemente acelerado.



Vamos agora produzir uma animação que mostre o movimento do projétil. Para tal, precisamos aprender a usar a janela *Animação*. Nela podemos fazer com que diferentes objetos se comportem de maneira determinada pelo modelo. Os objetos são criados com os botões que estão colocados do lado esquerdo da janela. O projétil cujo movimento definimos na **Figura 4.8** será representado por uma *partícula*, um dos vários tipos de objeto disponíveis. Para criar uma partícula na janela de animações, use o botão com uma bolinha desenhada, mostrado na **Figura 4.9**. Note que a função do botão é informada quando passamos o cursor sobre ele.



**Figura 4.9:** Janela de animações.

Após apertar o botão, leve o cursor para o meio da janela (repare que ele ficou diferente) e clique no ponto onde deseja criar a partícula. Imediatamente, abre-se uma caixa de diálogo como a mostrada na **Figura 4.10**. Nela são definidas as propriedades da partícula. A mais importante é a sua posição, dada por duas coordenadas, uma horizontal e outra vertical. Defina que a posição horizontal será dada pela variável  $x$  e a vertical por  $y$ , marcando estas grandezas nos painéis correspondentes, como está mostrado na **Figura 4.10**. Mude também o nome do objeto criado, de “Objeto n°. xx” para algo mais fácil de lembrar, como “partícula”. Clique o botão OK; a caixa de diálogo se fechará, deixando na janela de animação uma partícula verde.

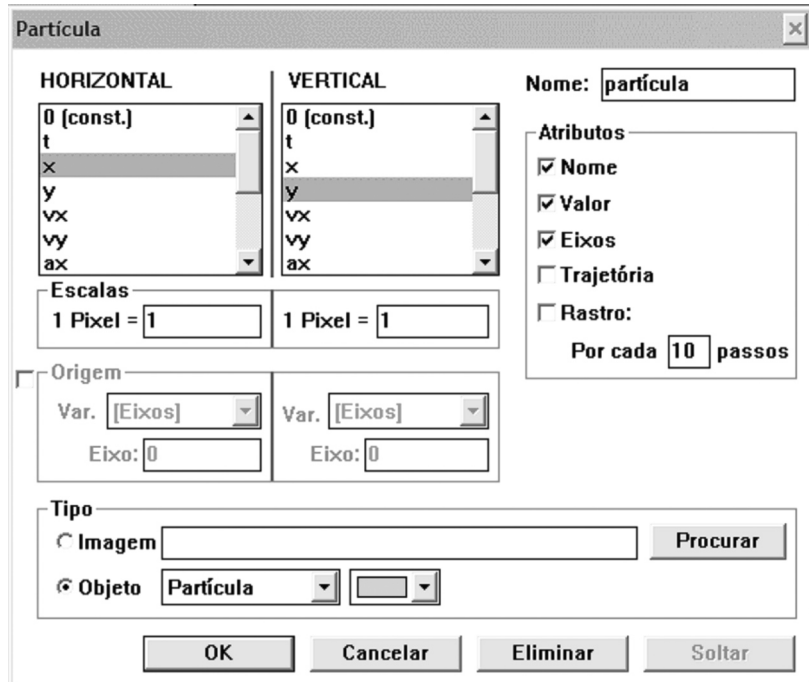


Figura 4.10: Propriedades da partícula criada.

Para rodar a animação, temos que dar valores aos parâmetros do modelo. Na janela de parâmetros, use  $x_0 = y_0 = 0$ ,  $v_{0x} = 20$ ,  $v_{0y} = 25$  e  $g = 9.8$ ; nas opções da janela de controle, faça o tempo máximo igual a 4. Agora rode o programa – se tudo der certo, você verá a partícula mover-se como um projétil balístico. O resultado final está mostrado na Figura 4.11.

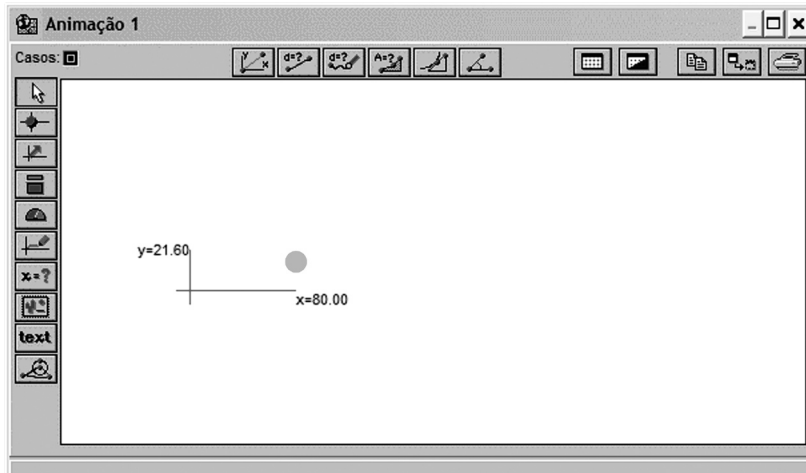


Figura 4.11: A partícula na janela de animação.

A animação que fizemos tem um defeito: ela ficou muito pequena. Para mudar a escala do desenho, temos que abrir novamente a caixa de diálogo com as propriedades da partícula. Isso é feito colocando o cursor sobre a partícula ou na origem do sistema de eixos (você notará que ele muda de aspecto) e clicando o botão direito do mouse. A caixa com os atributos da partícula vai aparecer, e você poderá editar seu conteúdo. O tamanho do desenho é mudado na área identificada como *Escalas*. Ali se determina a transformação de escala dos *pixels* na tela do computador para as correspondentes unidades de  $x$  e  $y$ . Para entender o que isso quer dizer, você tem que saber que as imagens que se vêem na tela do computador são formadas por minúsculos pontinhos luminosos muito próximos uns dos outros, os *pixels*. Você pode vê-los com uma lupa ou colocando uma gotinha de água sobre a tela. A escolha de escalas corresponde a determinar a distância entre dois *pixels* sucessivos nas unidades usadas para medir  $x$  e  $y$  (para fixar as idéias, digamos que  $x$  e  $y$  estejam em metros). Observe, na **Figura 4.10**, que o valor inicialmente atribuído pelo *Modellus* às escalas é 1, ou seja, andar um pontinho na horizontal ou vertical corresponde a andar 1 metro. É por isso que a animação ficou pequena: os 80 metros que a partícula percorreu (veja o valor final de  $x$  na **Figura 4.11**) correspondem a 80 *pixels*. Num monitor com resolução de 1024x768 *pixels*, comum hoje em dia, isso dá menos de 10% da extensão horizontal da tela. Para a figura ficar cinco vezes maior, por exemplo, temos que mudar as escalas horizontal e vertical de modo que 1 *pixel* corresponda a 0,2 metro. A **Figura 4.12** mostra como fazer isso. Na mesma figura, podem-se ver outras mudanças que melhoram o aspecto do desenho: as caixinhas ao lado dos atributos Nome e Valor foram desmarcadas, e a associada a *Trajectoria* agora está marcada. Com isso, o nome ( $x$  e  $y$ ) e o valor numérico das coordenadas da partícula não mais aparecerão, e a trajetória será desenhada. O desenho obtido ao final da animação está mostrado na **Figura 4.13**. Repare na mudança de escala e no traçado da trajetória.

A origem do sistema de eixos que é desenhado durante a animação marca o ponto onde  $x = y = 0$ . Este é o local onde você clicou ao criar a partícula. O sistema de referência pode ser deslocado com o mouse – basta colocar o cursor sobre a origem (ou a partícula), apertar o botão direito e carregar tudo para outro ponto da janela.

Errata:  
No texto ao lado, onde se lê "botão direito" leia-se "botão esquerdo".

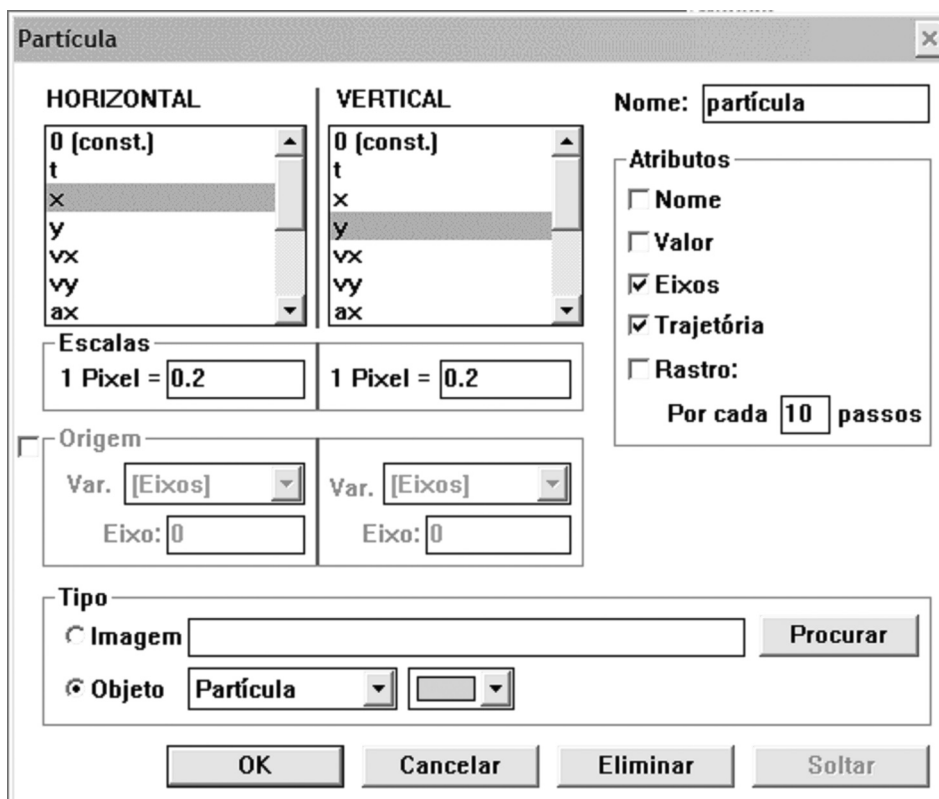


Figura 4.12: Caixa de propriedades da partícula, com novas escalas e atributos.

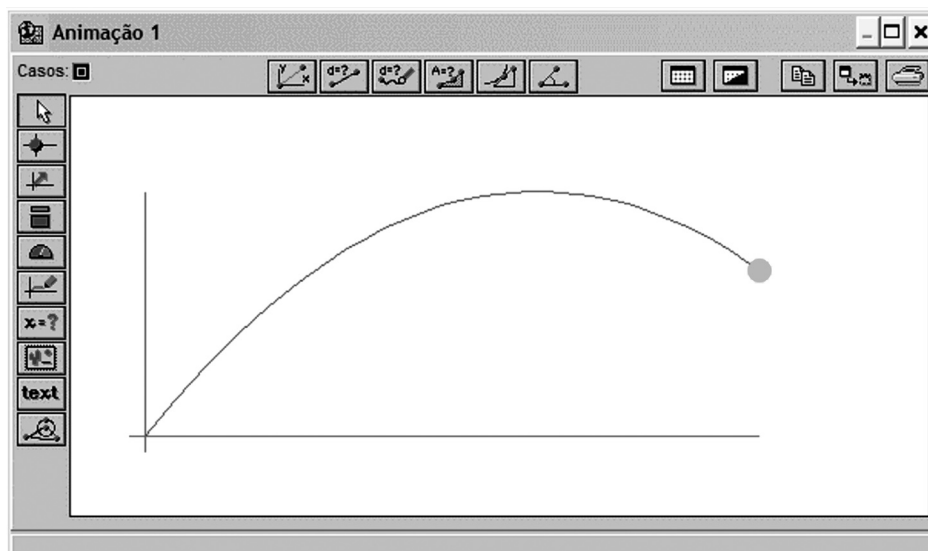
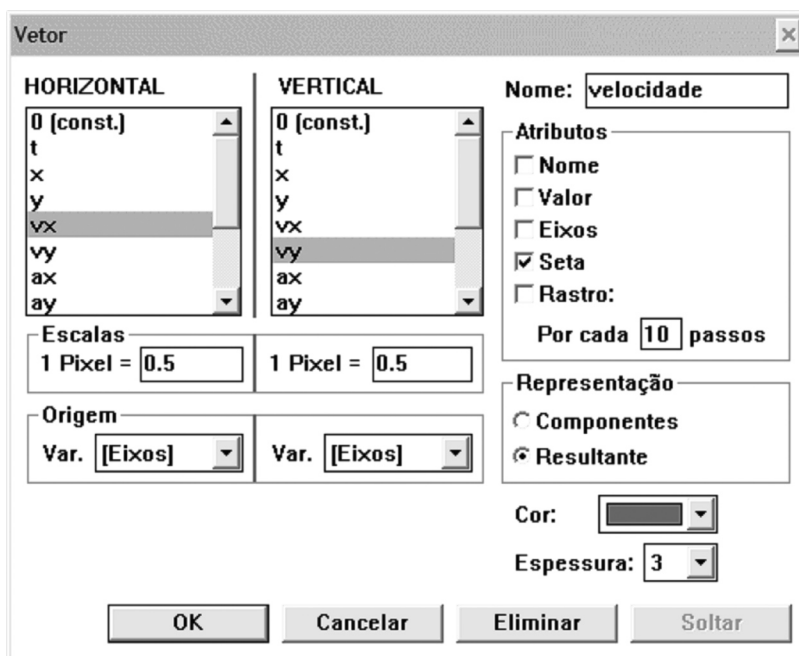


Figura 4.13: Trajetória da partícula.

Vamos agora mostrar o vetor velocidade da partícula. Para isto, clique o botão que cria vetores – é aquele com uma seta desenhada, logo abaixo do botão de partícula (veja a **Figura 4.10**) – e, em seguida, clique em algum ponto no interior da janela de animação. Você verá abrir-se uma caixa de diálogo, como a mostrada na **Figura 4.14**. É nela que são definidas as propriedades do vetor (note a semelhança com a caixa das partículas). A **Figura 4.14** mostra as opções a serem adotadas: marque  $v_x$  e  $v_y$  como as componentes horizontal e vertical do vetor; mude as escalas para 0.5; desmarque os atributos *Nome*, *Valor* e *Eixos*, deixando selecionado apenas *Seta* (do contrário, o vetor não será desenhado); e aumente a espessura da linha que representa o vetor para 3. Mude também o nome do vetor, de “Vetor n. xx” para “velocidade”. Aperte o botão *OK*, feche a caixa de diálogo, e você verá o vetor na janela de animação.



**Figura 4.14:** Propriedades do vetor velocidade.

Podemos “prender” o vetor velocidade à partícula, fazendo com que ele a acompanhe ao longo da trajetória. Para fazer isso, use o mouse para colocar o vetor sobre a partícula; neste ponto, o cursor ganhará a forma de um nó, e uma caixa de diálogo surge perguntando se você quer ligar o vetor à partícula. Responda *Sim* e execute novamente a animação. Você verá que o vetor velocidade segue agora junto com a partícula, como está na **Figura 4.15**.

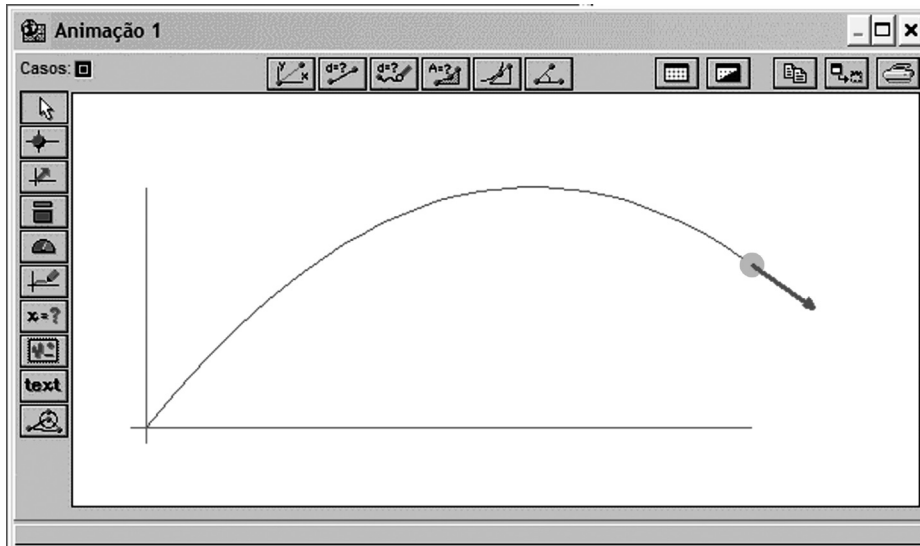


Figura 4.15: O vetor velocidade "preso" à partícula.

O mesmo procedimento pode ser seguido para exibir o vetor aceleração. Crie um vetor com componentes  $ax$  e  $ay$ , coloque as escalas em 0.2 e ligue-o à partícula. Execute a animação e observe como os dois vetores, velocidade e aceleração, se comportam durante o movimento da partícula. O resultado final está mostrado na Figura 4.16.

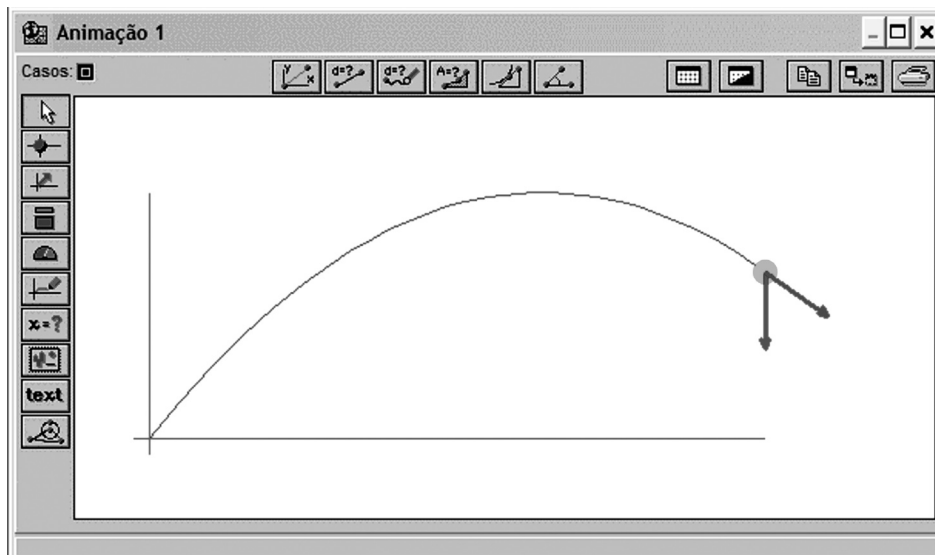
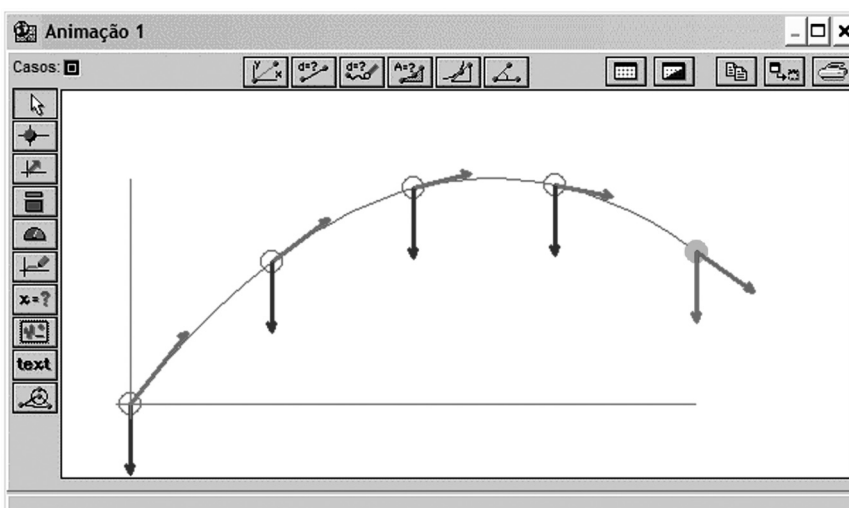


Figura 4.16: Velocidade e aceleração da partícula.

As animações do *Modellus* podem deixar uma espécie de rastro, marcando os locais por onde passam os objetos criados. Para fazer isso, abra novamente a caixa de propriedades da partícula (note que o *Modellus* lhe perguntará se quer editar a partícula ou algum dos vetores ligados a ela). Marque a opção *Rastro* e execute a animação. Você verá que a partícula deixa uma “pegada” após dar alguns passos. O número de passos entre duas marcas é fixado logo abaixo da opção *Rastro*. A mesma coisa pode ser feita para a velocidade e a aceleração – a **Figura 4.17** mostra os rastros deixados pela partícula e pelos vetores.



**Figura 4.17:** Rastros da partícula e dos vetores velocidade e aceleração.



## ATIVIDADE

### 1. Movimento circular uniforme

Uma modificação simples do modelo anterior pode ilustrar outro movimento importante: o circular uniforme. Para isto, basta mudar as definições de  $x(t)$  e  $y(t)$  para

$$x = R \cos(\omega t),$$

$$y = R \sin(\omega t).$$

O resultado está na **Figura 4.18**.

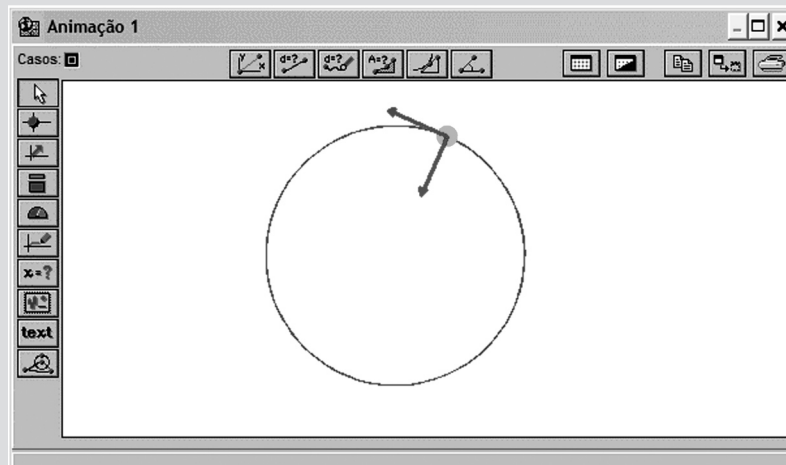


Figura 4.18: Movimento circular uniforme.

## INFORMAÇÕES SOBRE A PRÓXIMA AULA

A derivada de funções não é o único aspecto do cálculo infinitesimal que pode ser abordado com o *Modellus*. Como veremos na próxima aula, o programa também é capaz de resolver equações diferenciais, uma característica particularmente útil em aplicações à Física.



## Equações diferenciais com o *Modellus*

### Metas da aula

Discutir como equações diferenciais podem ser resolvidas com o *Modellus*. Apresentar simulações de sistemas físicos que são descritos por equações diferenciais.

# objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- resolver equações diferenciais ordinárias com o *Modellus*;
- definir a condição inicial de uma equação diferencial;
- desenvolver simulações de sistemas físicos modelados por equações diferenciais;
- comparar o resultado de simulações com resultados experimentais.

## RESOLVENDO EQUAÇÕES DIFERENCIAIS COM O *MODELLUS*

O *Modellus* não calcula apenas derivadas – ele também resolve equações diferenciais ordinárias. Esta capacidade o torna extraordinariamente útil ao ensino de Física, já que muitas leis físicas são expressas matematicamente como equações diferenciais.

Uma equação diferencial ordinária de primeira ordem tem a forma geral

$$\frac{dy}{dt} = f(y, t)$$

onde  $f(y, t)$  é uma função conhecida. Resolver esta equação significa encontrar a função  $y(t)$  cuja derivada em relação a  $t$  seja igual a  $f(y(t), t)$ . Em geral, existem muitas (infinitas) funções com essa propriedade. Para obter uma solução única, é necessário especificar a *condição inicial*

$$y(t_0) = y_0$$

ou seja, devemos indicar o valor de  $y(t)$  em um dado  $t = t_0$ .

Por exemplo, consideremos uma equação diferencial particularmente simples

$$\frac{dy}{dt} = t$$

cujas soluções gerais são

$$y(t) = \frac{1}{2}t^2 + c$$

onde  $c$  é uma constante arbitrária. Para cada valor dessa constante (há um número infinito deles) temos uma solução diferente da equação diferencial. Se especificarmos uma condição inicial, por exemplo,

$$y(0) = 50$$

a constante  $c$  ficará determinada ( $c = 50$ ) e só uma solução será possível:

$$y(t) = \frac{1}{2}t^2 + 50$$

Errata:  
A fórmula correta  
é  $y(t) = 1/2 t^2 + 50$

Como resolvemos esta mesma equação com o *Modellus*? Basta escrevê-la na janela *Modelo*, da maneira mostrada na Figura 5.1.

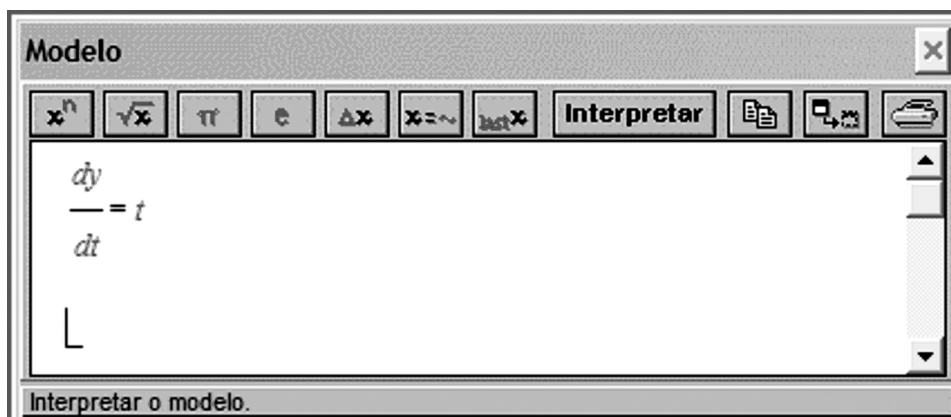


Figura 5.1: Equação diferencial definida na janela *Modelo*.

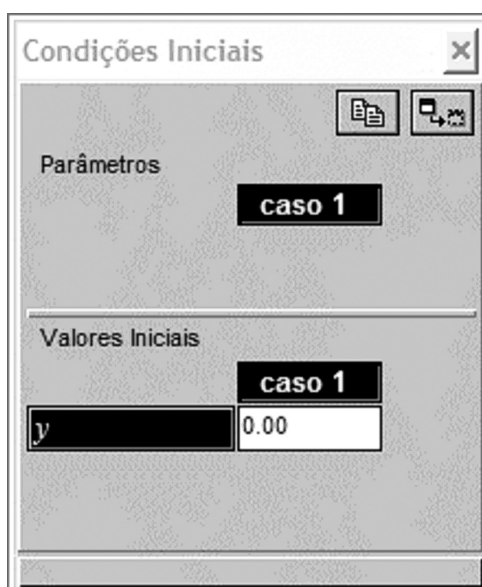


Figura 5.2: Condição inicial da equação diferencial.

Ao apertar o botão *Interpretar*, a janela *Condições Iniciais* é criada, mas com uma novidade: em vez de pedir valores dos parâmetros do modelo, ela solicita que a condição inicial da equação diferencial seja especificada, como se vê na Figura 5.2.

Note que um valor  $y_0 = 0$  já vem especificado quando a janela é aberta. Mude este valor para  $y_0 = 50$  e execute a simulação. O gráfico de  $y(t)$  deve aparecer na janela *Gráfico* e, se tudo deu certo, ele é semelhante ao que está na Figura 5.3.

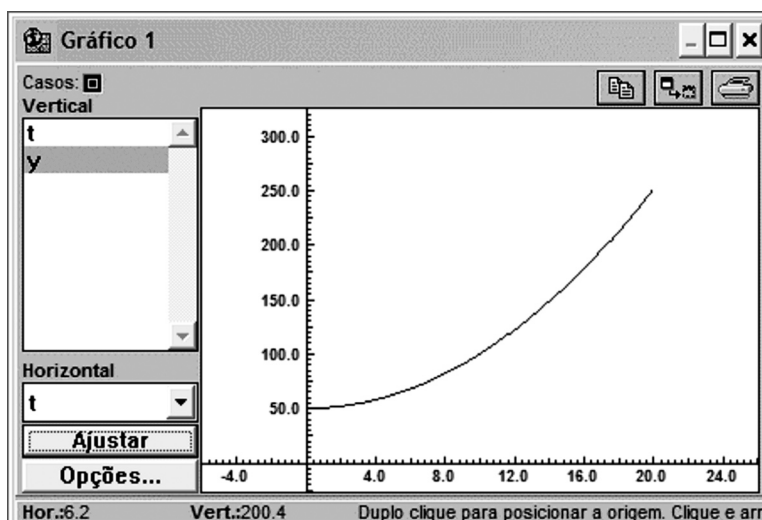


Figura 5.3: Gráfico da solução da equação diferencial.

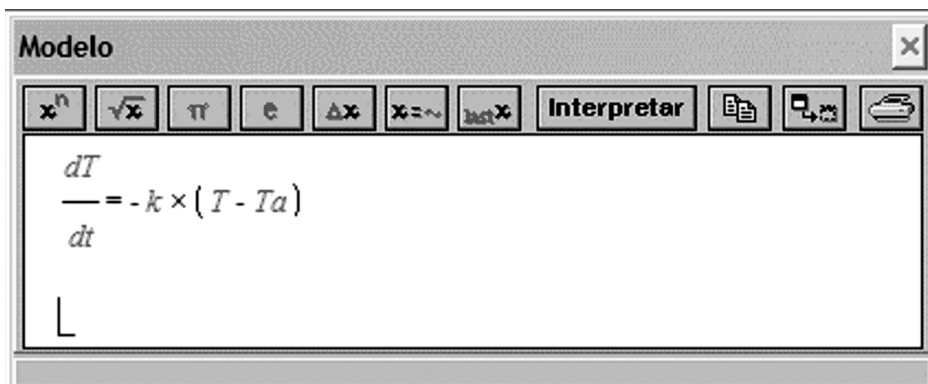
O gráfico tem a forma de uma parábola passando por  $y(0) = 50$ , como esperado (veja a solução analítica encontrada). É importante perceber que o valor de  $t_0$  não é escolhido na janela *Condições Iniciais* – ele é dado pelo limite inferior de  $t$  que está especificado na janela *Controlo*. O valor  $t = 0$  é fixado quando o *Modellus* tem início e, como já vimos, essa escolha pode ser alterada com o botão *Opções* da janela *Controlo*.

## O MODELO DE NEWTON PARA O RESFRIAMENTO

Imagine um prato de sopa quente colocado sobre a mesa: todos sabemos que ele esfria até atingir a temperatura ambiente. De que maneira isso ocorre? Como a temperatura da sopa varia com o tempo? A resposta é dada pela *lei de Newton do resfriamento*. Segundo esta “lei” – que, na verdade, é apenas um modelo aproximado para a condução térmica – quanto maior for a diferença entre a temperatura  $T$  de um corpo e a temperatura ambiente  $T_a$ , mais rapidamente o corpo irá esfriar (se  $T_a < T$ ) ou esquentar (se  $T_a > T$ ). Mais exatamente, a lei de Newton do resfriamento diz que a taxa de mudança da temperatura  $T$  com o tempo é diretamente proporcional à diferença  $T_a - T$ , ou seja

$$\frac{dT}{dt} = -k(T - T_a)$$

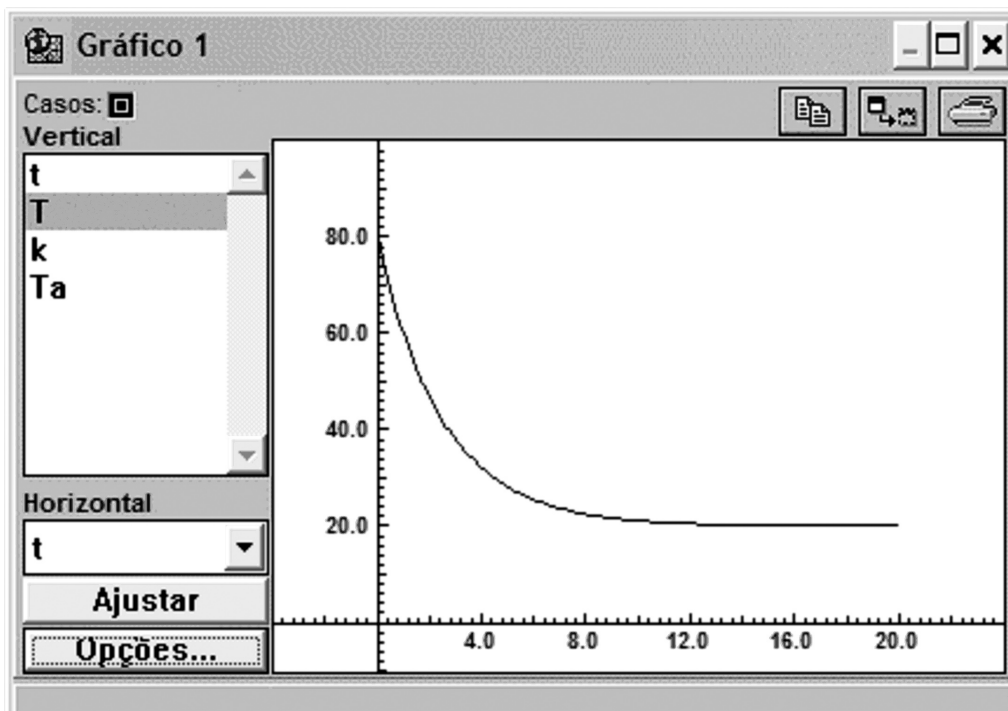
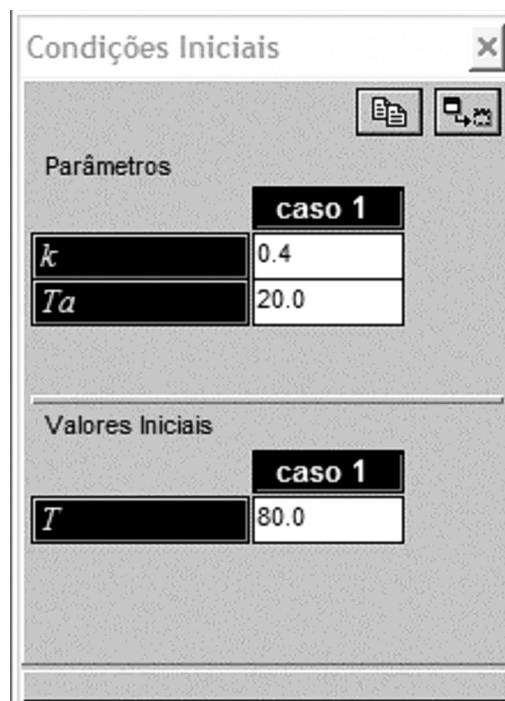
onde  $k$  é uma constante positiva. Portanto, a lei do resfriamento é uma equação diferencial. Podemos usar o *Modellus* para resolver essa equação e encontrar  $T(t)$ . Para isso, escreva a lei de Newton na janela *Modelo*, como mostrado na **Figura 5.4**.



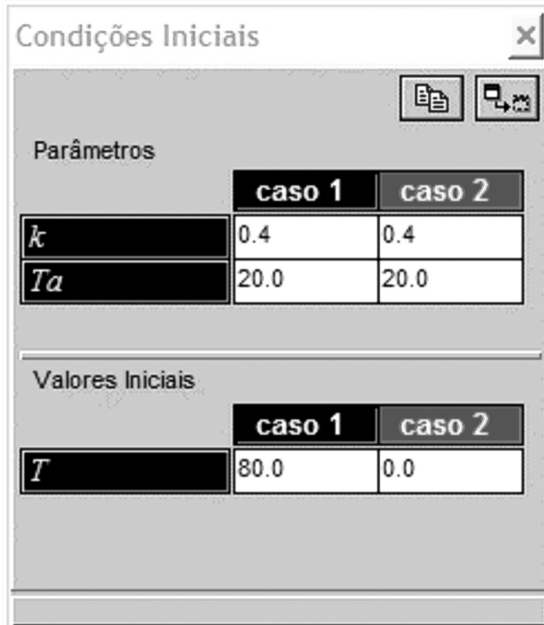
**Figura 5.4:** Modelo newtoniano de resfriamento.

Ao interpretar o modelo, a janela *Condições Iniciais* vai solicitar os parâmetros  $k$  e  $T_a$  e, também, a condição inicial  $T(0)$ . Dê os valores indicados na **Figura 5.5**:  $k = 0.4$ ,  $T_a = 20$  e  $T(0) = 80$ . Rode a simulação e faça o gráfico de  $T \times t$ . Você deve obter um gráfico como o mostrado na **Figura 5.6**, onde se pode acompanhar como a temperatura diminui até o objeto entrar em equilíbrio com o ambiente.

**Figura 5.5:** Parâmetros e condição inicial para o modelo de resfriamento.

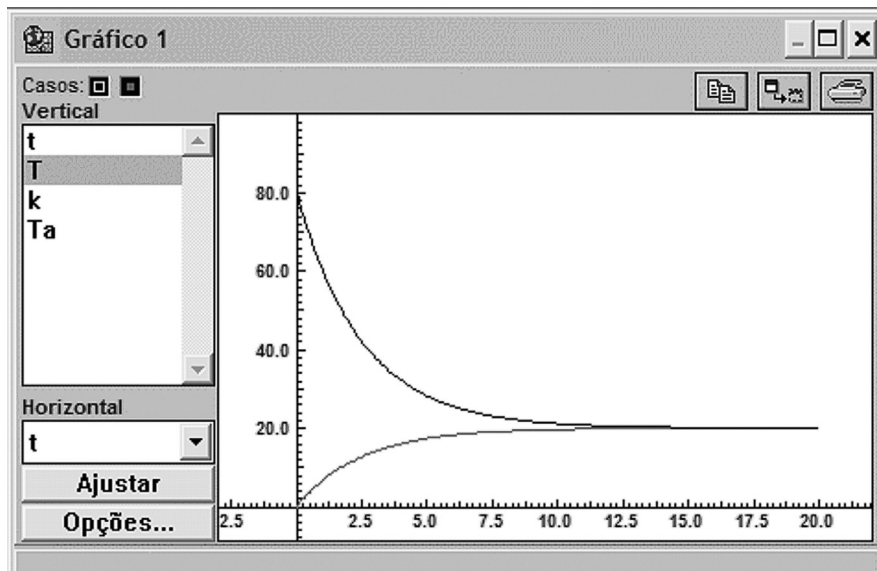


**Figura 5.6:** A temperatura como função do tempo.



É possível criar casos com diferentes condições iniciais. Clicando em *Caso / Adicionar* um novo caso é colocado na janela *Condições Iniciais*, e podemos alterar seus parâmetros e temperatura inicial. Coloque  $T(0) = 0$  no *caso 2*, como está na **Figura 5.7**, e execute a simulação. O gráfico conjunto dos dois casos está mostrado na **Figura 5.8** (lembre-se de marcar as duas caixinhas de *Casos* no alto da janela *Gráfico*).

**Figura 5.7:** Dois casos de condições iniciais.



**Figura 5.8:** Evolução a partir de temperaturas acima e abaixo da ambiente.

Observe que, no segundo caso, a temperatura inicial está abaixo da ambiente, de modo que o objeto esquenta com o passar do tempo.

## COMPARAÇÃO COM DADOS EXPERIMENTAIS

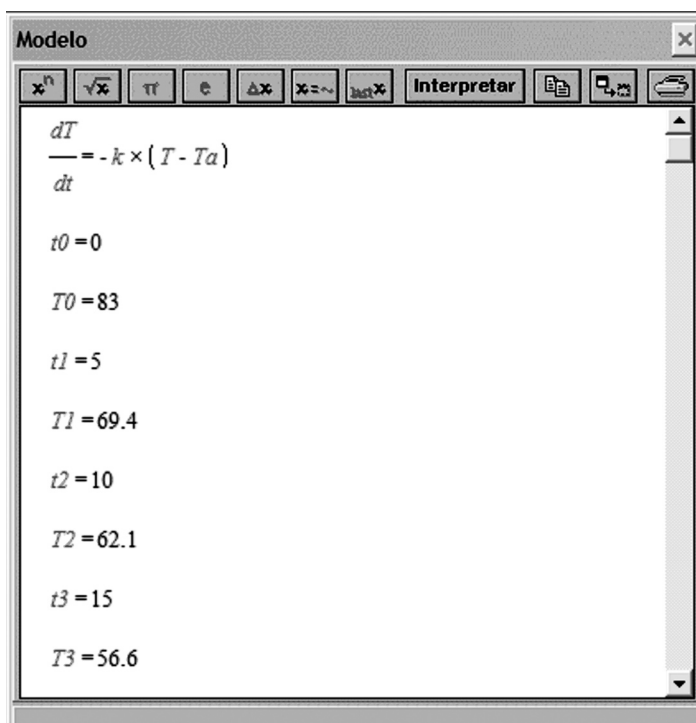
A validade do modelo de resfriamento de Newton pode ser testada comparando suas previsões a medidas de temperatura e tempo. Como veremos, a comparação pode ser feita no próprio *Modellus*, usando os

recursos da janela *Animação*. Nossos dados experimentais vêm de uma caneca de café quente, que esfria em contato com o ar, a uma temperatura ambiente  $T_a = 22^\circ \text{C}$ . O resultado das medidas está na **Tabela 5.1**.

**Tabela 5.1:** Medidas feitas em uma caneca de café esfriando sobre a mesa. A temperatura ambiente é  $22^\circ \text{C}$ .

| Tempo (minutos) | Temperatura ( $^\circ\text{C}$ ) |
|-----------------|----------------------------------|
| 0               | 83,0                             |
| 5               | 69,4                             |
| 10              | 62,1                             |
| 15              | 56,6                             |

A comparação entre teoria e experimento pode ser feita colocando-se os dados diretamente na janela *Modelo*. É claro que isso é viável apenas se a quantidade de medidas for relativamente pequena, como é o caso agora. Copie esses dados para a janela *Modelo*, colocando-os logo depois da equação diferencial, como está mostrado na **Figura 5.9**. Note que a primeira linha de dados da tabela foi escrita como  $t_0 = 0$  e  $T_0 = 83.0$ , a segunda como  $t1 = 5$  e  $T1 = 69.4$  e assim sucessivamente (Não há nada de especial na escolha desses nomes para os dados – poderíamos ter usado qualquer outra identificação.)



**Figura 5.9:** O modelo de Newton e os dados experimentais.

Não é possível colocar os dados experimentais na janela de gráficos, mas podemos fazer isso na janela de animações. Para tanto, vamos associar cada medida a um objeto do tipo *partícula*, que vai representar o ponto experimental ( $t$ ,  $T$ ). Clique o botão de criação de partícula na janela *Animação* (lembre-se da aula anterior), leve o cursor para o interior da janela e clique novamente: a caixa de diálogo que já conhecemos bem é aberta, solicitando a definição das propriedades da partícula criada. Coloque algo como o mostrado na Figura 5.10.

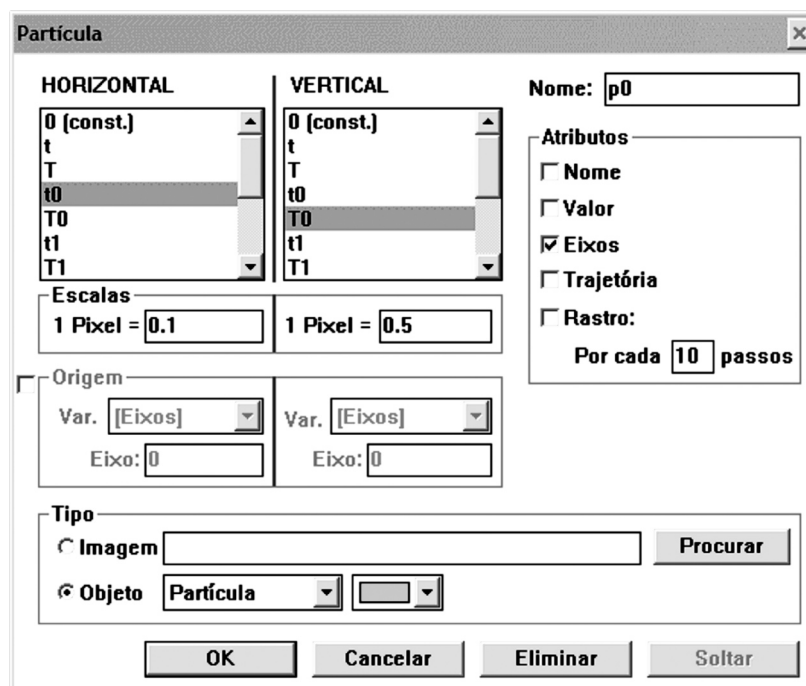
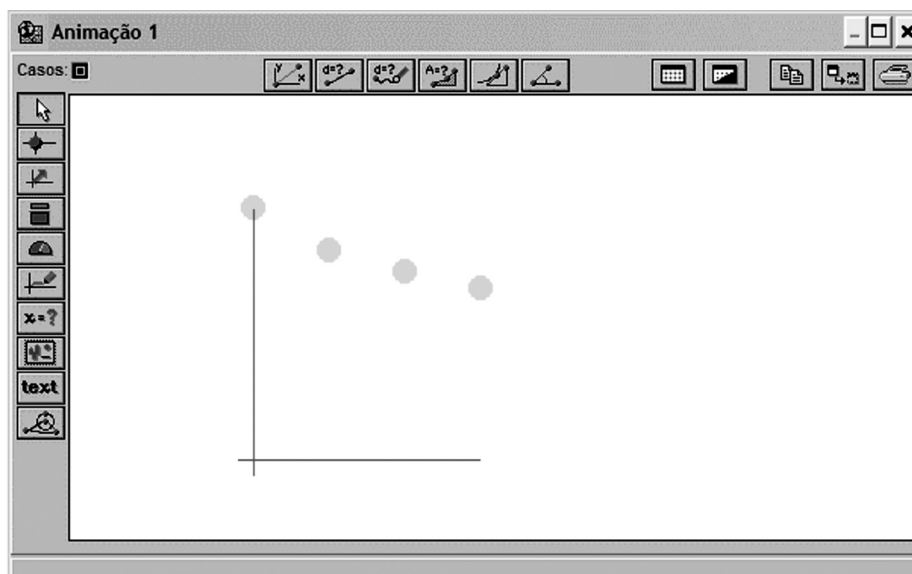


Figura 5.10: Partícula que representa o primeiro par de dados experimentais.

Repare que definimos as coordenadas da partícula como sendo  $t_0$  (horizontal) e  $T_0$  (vertical), ou seja, ela vai representar o primeiro par de dados. As escalas foram escolhidas para que os dados fiquem bem distribuídos pela janela *Animação*, mas elas podem mudar dependendo do tamanho da janela e da resolução da tela. Todos os atributos da partícula foram desativados, com exceção de *Eixos*, que nos ajudará a alinhar os diferentes pontos experimentais. Finalmente, em vez de “Objeto no. xxx”, demos à partícula o nome  $p_0$  (ponto 0). Clicando OK, a caixa de diálogo é fechada e uma partícula verde aparece na janela *Animação*, marcando o primeiro ponto experimental. Repita



o procedimento anterior, criando uma nova partícula (chamada  $p1$ ) para o próximo ponto experimental ( $t1, T1$ ), e assim por diante. Cuide para que, em todas as partículas, as escalas sejam sempre as mesmas. Quando os quatro pontos medidos estiverem inseridos na janela, use o mouse para carregar as partículas e alinhar seus eixos, colocando todas as origens exatamente no mesmo ponto (ou seja, umas sobre as outras). Caso o *Modellus* pergunte se você deseja prender uma partícula à outra, responda Não. O resultado final dessa arrumação deve ficar parecido com o que está na **Figura 5.11**.



**Figura 5.11:** Os dados experimentais na janela de animações.

O que fizemos na **Figura 5.11** foi construir um gráfico dos resultados experimentais, mostrando como a temperatura do café diminui com o tempo. Agora vamos comparar esses dados com o modelo de resfriamento de Newton. Para isso, podemos fazer um gráfico da previsão do modelo,  $T(t)$ , e colocá-lo sobre os dados experimentais. Para inserir um gráfico na janela *Animação*, clique o botão que tem o desenho de um lápis traçando uma linha (quando o cursor está sobre ele, o texto “Inserir um novo gráfico” aparece na base da janela). Leve o cursor para o interior da janela e clique sobre a origem dos eixos das partículas. Uma caixa de diálogo vai aparecer, pedindo as propriedades do gráfico. Coloque o tempo  $t$  no eixo horizontal e a temperatura  $T$  no eixo vertical, usando as mesmas escalas adotadas para os pontos experimentais. Ainda

como no caso dos pontos, deixe todos os atributos desmarcados, menos *Eixos*. Escolha também a cor e espessura da linha a ser traçada no gráfico. O resultado final deve ficar parecido com o que está na **Figura 5.12**. Ao clicar *OK*, a caixa de diálogo se fecha e o gráfico é criado. Se os eixos do gráfico não coincidirem com os eixos das partículas, use o mouse para colocar as origens no mesmo local.

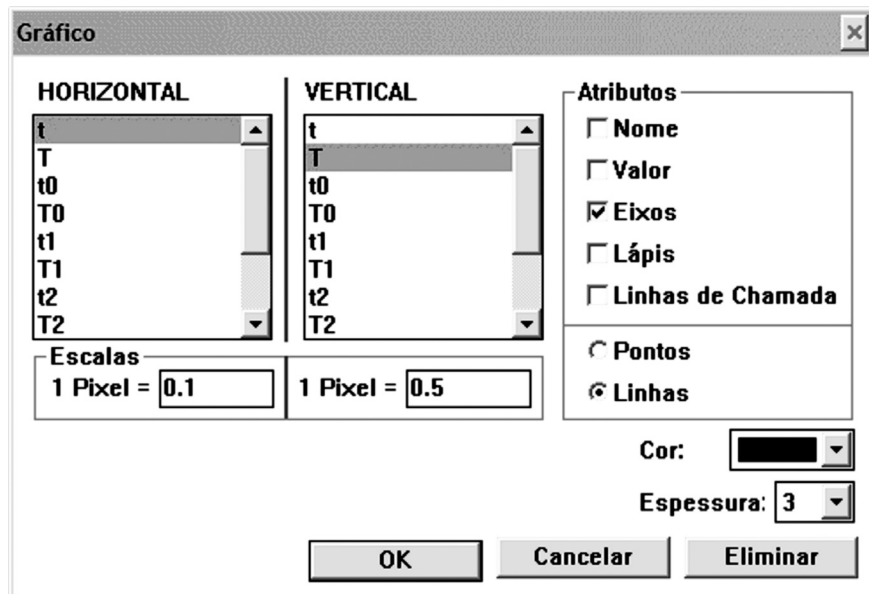
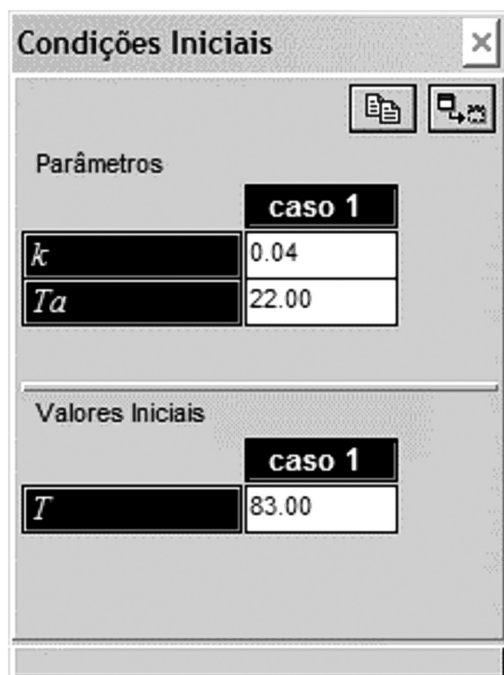


Figura 5.12: Propriedades do gráfico na janela de animações.



Antes de rodar a simulação, ainda temos de escolher os parâmetros do modelo e a condição inicial. A temperatura ambiente é conhecida,  $T_a = 22^\circ \text{C}$ , e a temperatura inicial faz parte dos dados experimentais,  $T(0) = 83^\circ \text{C}$ . Portanto, falta apenas conhecer o parâmetro  $k$ . Como não temos nenhuma informação sobre ele, temos de tentar vários valores e verificar se, com algum deles, é possível descrever os resultados experimentais. Use  $k = 0.04$ , como está na **Figura 5.13**, e execute a simulação.

Figura 5.13: Parâmetros para a comparação do modelo de Newton com o experimento da caneca de café.

O gráfico produzido pela simulação está mostrado na **Figura 5.14**. Vemos que, com o valor escolhido para  $k$ , a previsão do modelo de Newton (a linha contínua) concorda bastante bem com os dados experimentais. Note as identificações colocadas nos eixos – elas foram inseridas com o botão de inserção de texto (aquele onde se lê *text*), da maneira usual à janela *Animação* (que já deve lhe ser familiar a esta altura).

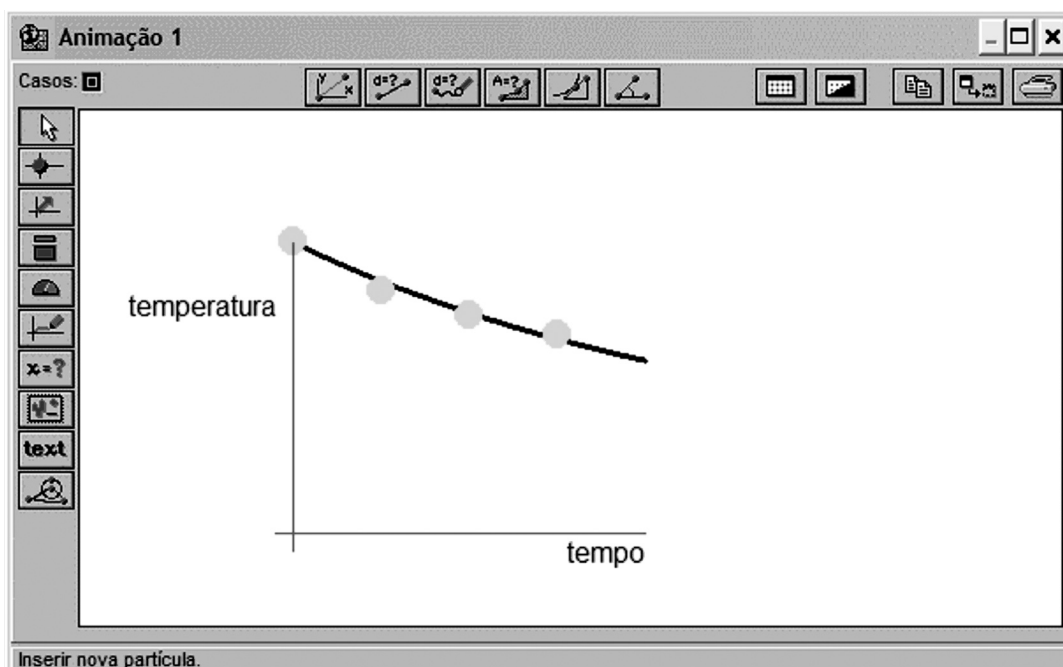


Figura 5.14: Comparação entre o modelo de Newton e os resultados experimentais.

### ATIVIDADE

1. Mude o valor de  $k$  e verifique se o acordo com os dados experimentais continua aceitável. Quando  $k$  aumenta, o resfriamento previsto fica mais rápido ou mais lento? A nossa caneca de café tem  $k = 0,04$ . Em que unidade está esse valor?

---



---



---



---



## INFORMAÇÕES SOBRE A PRÓXIMA AULA

Continuaremos nosso estudo do *Modellus* na próxima aula, analisando sistemas bem diferentes de uma caneca de café, mas que também podem ser modelados por equações diferenciais: as populações.

## Modelos populacionais

### Metas da aula

Utilizar o *Modellus* para formular e analisar modelos de crescimento populacional, aplicando os recursos de solução de equações diferenciais.

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

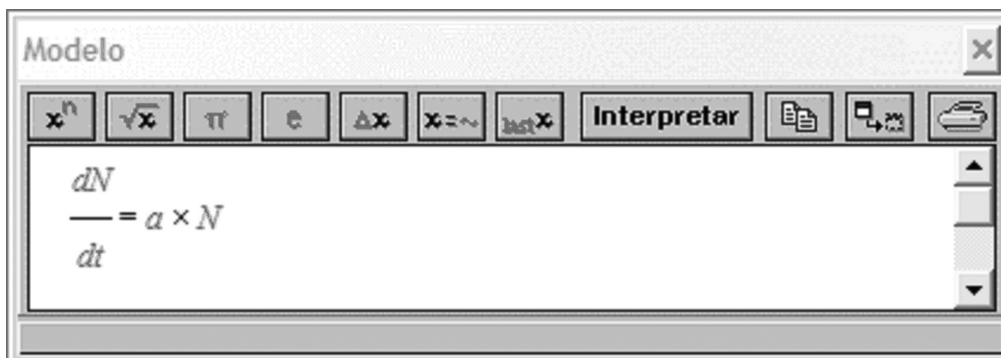
- identificar os modelos básicos de crescimento populacional;
- formular esses modelos como equações diferenciais;
- resolver essas equações com o *Modellus*;
- comparar as previsões dos modelos com dados e observações.

## MALTHUS E O CRESCIMENTO EXPONENCIAL

O primeiro modelo de crescimento populacional foi proposto por Thomas Malthus em 1798. Ele observou que, na ausência de restrições ambientais, a população humana aumentaria numa proporção fixa. Em termos matemáticos, se  $N(t)$  é o número de pessoas em uma certa área geográfica, no instante  $t$ , a hipótese de Malthus é escrita como

$$\frac{dN}{dt} = aN$$

onde  $a$  é uma constante, a taxa (relativa) de crescimento populacional. A idéia básica do modelo é simples: quanto mais gente existir, mais rapidamente a população vai aumentar. Que tipo de crescimento isso gera? Vamos usar o *Modellus* para encontrar a resposta. A hipótese de Malthus tem a forma de uma equação diferencial, que pode ser resolvida com os métodos que estudamos na aula anterior. Para encontrar a solução, escreva a equação na janela *Modelo*, como na **Figura 6.1**.



**Figura 6.1:** Modelo de crescimento malthusiano.

Interprete o modelo e dê valores ao parâmetro malthusiano  $a$  e à condição inicial  $N(0)$ . A **Figura 6.2** mostra o que acontece com  $a = 0.2$  e  $N(0) = 1$ . Note que não há nada de errado com a população inicial:  $N(0) = 1$  não quer dizer, necessariamente, um indivíduo – podem ser mil, um milhão ou qualquer outro número de pessoas, dependendo da “unidade” populacional adotada.

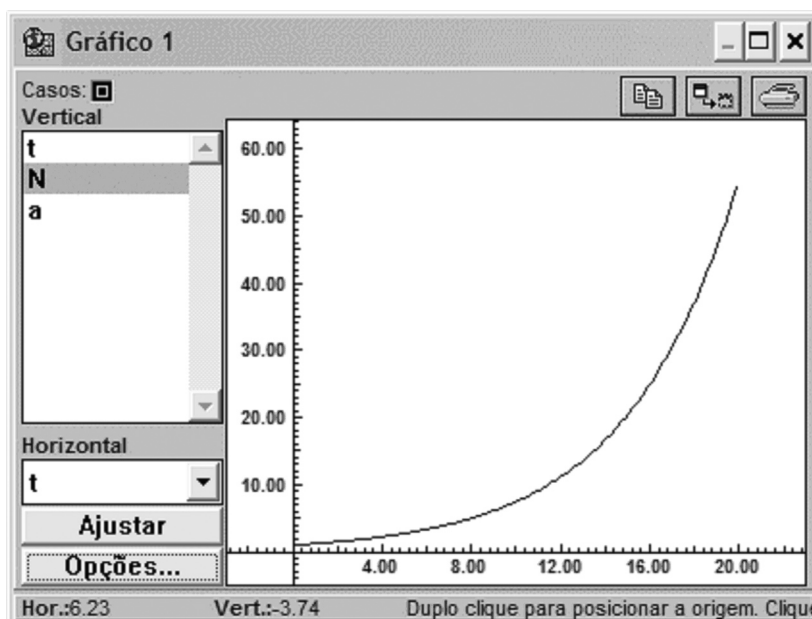


Figura 6.2: Crescimento populacional no modelo de Malthus.

Como vemos na Figura 6.2, o crescimento populacional previsto pelo modelo é explosivo (demograficamente falando) – de fato, este é um exemplo clássico de *crescimento exponencial*. A denominação é usada porque a solução analítica do modelo de Malthus é dada por uma função exponencial,

$$N(t) = N_0 e^{at}$$

ou seja, a população cresce exponencialmente com o tempo. Com o *Modellus*, mesmo quem não sabe cálculo pode verificar facilmente que a evolução malthusiana é exponencial. Basta colocar a solução analítica na janela *Modelo*, como mostrado na Figura 6.3, e comparar seu gráfico com o da solução encontrada pelo *Modellus*. Será impossível distinguir um gráfico do outro (mas não se esqueça de fazer o parâmetro  $N_0$  igual à condição inicial  $N(0)$ ).

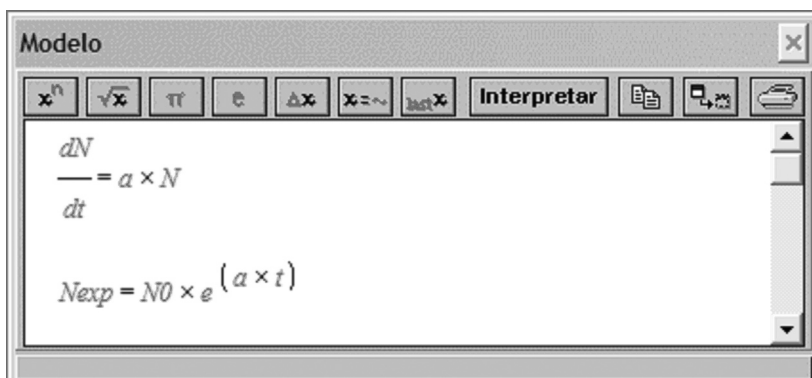


Figura 6.3: Duas formas do modelo de Malthus.

## CRESCIMENTO POPULACIONAL NO BRASIL

A Tabela 6.1 mostra o crescimento demográfico do Brasil nos últimos 100 anos. É interessante verificar se o aumento da população segue o modelo de Malthus. Para isso, vamos seguir um procedimento semelhante ao utilizado na aula anterior, colocando os dados diretamente na janela *Modelo*, como mostrado na Figura 6.4. Note que nos dados inseridos no *Modellus* o tempo está em anos, mas  $t = 0$  corresponde ao ano 1900. A população está dada em milhões de habitantes, como na Tabela 6.1:

Tabela 6.1: População do Brasil no século XX.

| Ano  | População (milhões de hab.) |
|------|-----------------------------|
| 1900 | 17                          |
| 1920 | 31                          |
| 1940 | 41                          |
| 1960 | 70                          |
| 1980 | 119                         |
| 2000 | 166                         |

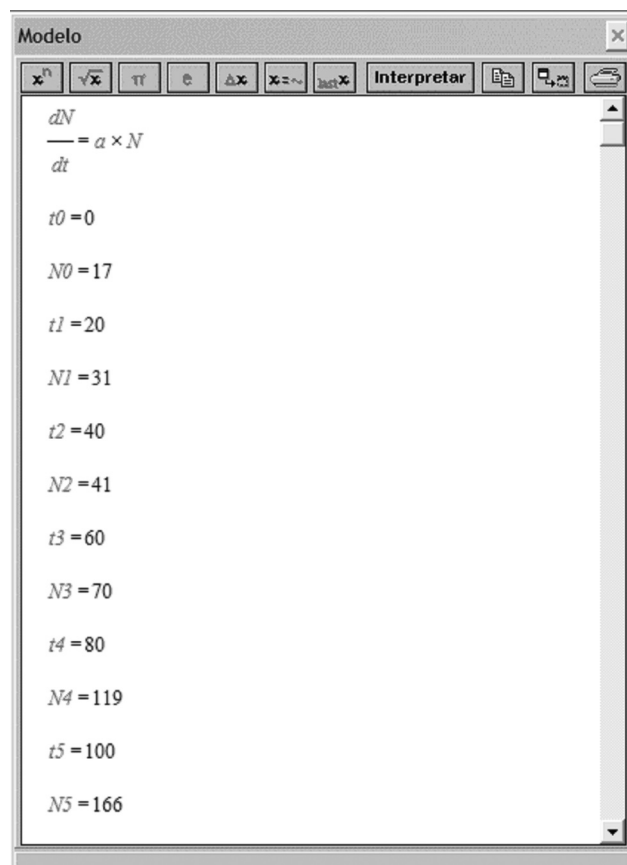
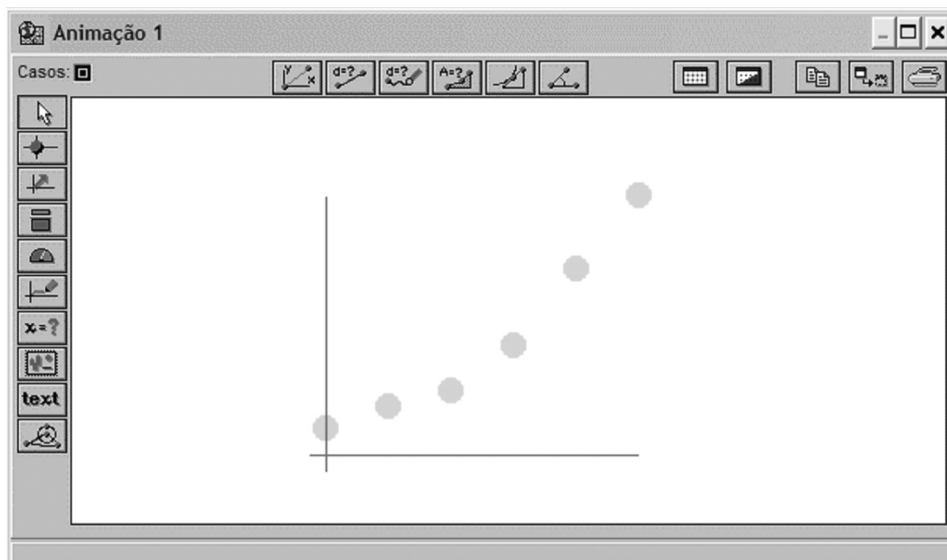


Figura 6.4: Dados demográficos inseridos na janela *Modelo*.



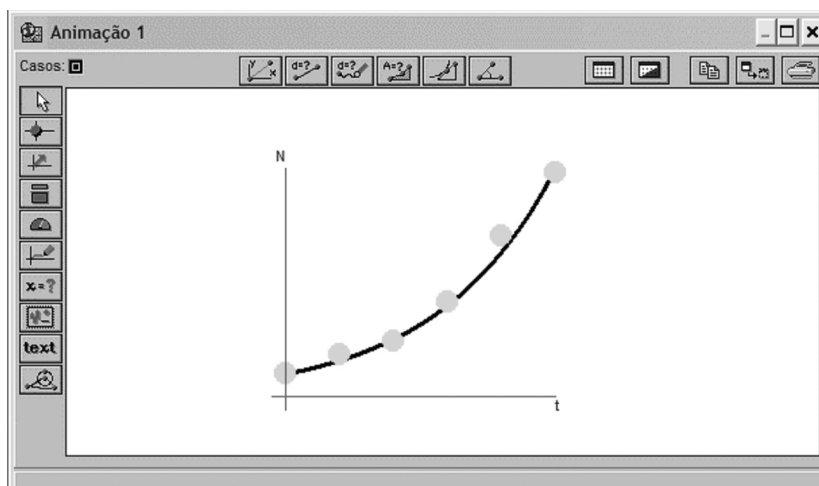
Podemos usar a janela *Animação* para fazer o gráfico desses dados. Um procedimento análogo ao da aula anterior leva ao resultado mostrado na **Figura 6.5**.



**Figura 6.5:** Crescimento populacional brasileiro no século XX.

Para comparar o crescimento populacional brasileiro ao modelo de Malthus, vamos superpor os dados demográficos e a curva prevista pelo modelo, também como fizemos na última aula. Para isso, precisamos escolher o parâmetro  $a$  (a condição inicial  $N(0)$  será o primeiro ponto da tabela). A **Figura 6.6** mostra que, com  $a = 0.023$  (encontrado após um pouco de tentativa e erro), o modelo malthusiano dá uma descrição bem razoável do crescimento demográfico brasileiro. Esse valor da taxa de crescimento  $a$  corresponde a um aumento da população de, em média, 2,3% ao ano durante o século XX.

**Figura 6.6:** Comparação entre o modelo de Malthus e o crescimento demográfico brasileiro.



O acordo entre os dados demográficos e o modelo de Malthus é razoável, mas não perfeito, como se pode observar na **Figura 6.6**. As discrepâncias mostram que a taxa de crescimento populacional não é constante como supôs Malthus. Apesar desse problema, o modelo é largamente utilizado como uma primeira aproximação para a dinâmica populacional.

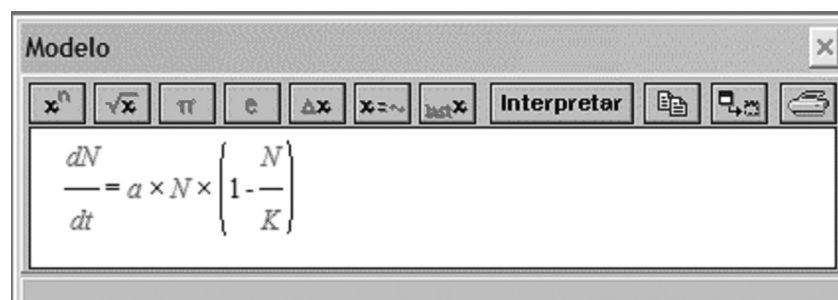
## O MODELO DE VERHULST

É claro que uma população não pode crescer indefinidamente: mais cedo ou mais tarde, o esgotamento dos recursos disponíveis imporá limites à expansão. O matemático Pierre Verhulst propôs, em 1838, uma generalização do modelo de Malthus que leva em conta essas restrições “ambientais”. Segundo Verhulst, a taxa relativa de crescimento demográfico diminui com o aumento da população, chegando a zero se uma dada população-limite (determinada pelos recursos disponíveis ou outras restrições) for alcançada. A expressão matemática do modelo de Verhulst é a equação diferencial

$$\frac{dN}{dt} = aN(1 - N/K)$$

A equação de Verhulst, também chamada *equação logística*, difere da de Malthus pelo fator  $1 - N/K$ , que elimina a explosão demográfica. Note que esse termo faz com que a taxa de crescimento populacional torne-se zero quando a população  $N = K$  é atingida. O parâmetro  $K$  é a população máxima que pode ser sustentada pelo meio ambiente ( $K$  é chamado, às vezes, de *capacidade de suporte*). Note que o modelo de Malthus corresponde à situação em que  $K = \infty$ .

Vamos implementar o modelo de Verhulst no *Modellus* e ver como suas previsões diferem das de *Malthus*. A **Figura 6.7** mostra como fica a janela *Modelo* com a equação de Verhulst.



**Figura 6.7:** Modelo de Verhulst.

Adotando  $K = 40$  para a capacidade populacional, e  $a = 0.2$  e  $N(0) = 1$  (como na Figura 6.2), o crescimento demográfico previsto pelo modelo de Verhulst é o que está mostrado no gráfico da Figura 6.8. Observe como a população tende a se estabilizar no limite fixado pela capacidade  $K$ , em contraste com a explosão demográfica malthusiana vista na Figura 6.2.

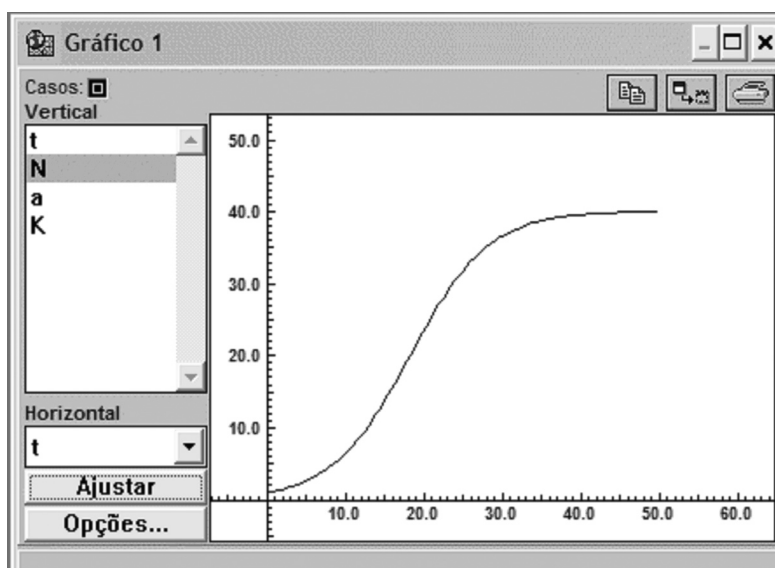


Figura 6.8: Crescimento demográfico no modelo de Verhulst. A capacidade populacional é  $K = 40$ . Note que a simulação vai até  $t = 50$ .

O que ocorre quando a população inicial excede a capacidade  $K$ ? Isso poderia acontecer, por exemplo, se uma população muito grande fosse levada para um território que não é capaz de sustentá-la. O resultado de uma simulação semelhante à anterior, mas com condição inicial  $N(0) = 100$  (portanto, maior que  $K = 40$ ), está mostrado na Figura 6.9. Como se pode observar, a população diminui rapidamente até atingir a capacidade de suporte  $K$ .

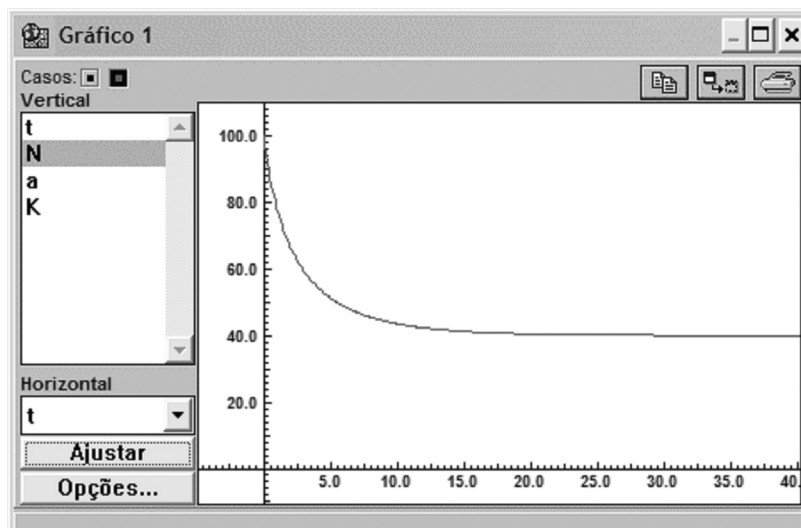
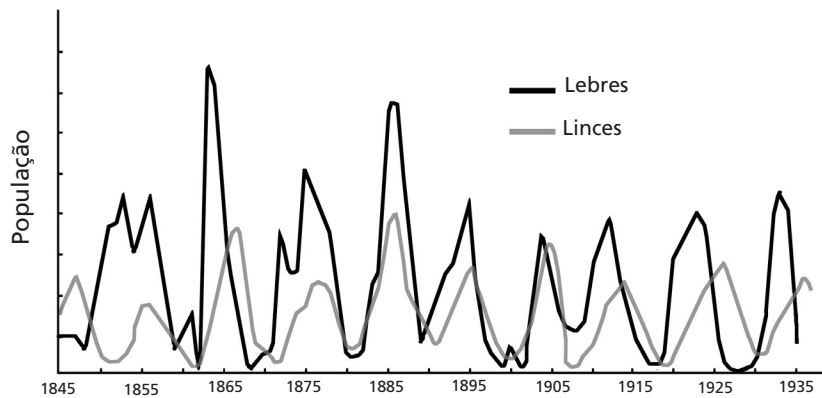


Figura 6.9: Diminuição de uma população superior à capacidade de suporte  $K$ .

## PREDADORES E PRESAS

Predador é um organismo que come outro organismo. Presa é o organismo que é comido. Alguns exemplos de predadores e presas são leões e zebras, tubarões e sardinhas, e lincos e lebres. Com relação aos últimos, a **Figura 6.10** mostra a evolução, ao longo de quase 100 anos, do número de lebres e lincos na região dos Grandes Lagos, no Canadá. Observe como a população de ambas as espécies oscila com um período de cerca de dez anos. Note também que a oscilação no número de lincos está um pouco defasada em relação às lebres: os máximos e mínimos dos lincos ocorrem quase sempre um pouco depois dos máximos e mínimos das lebres.



**Figura 6.10:** Lebres e lincos na região dos Grandes Lagos, Canadá.

Os modelos populacionais de Malthus e Verhulst não podem explicar o comportamento mostrado na **Figura 6.10**. A primeira descrição matemática razoavelmente bem-sucedida de um sistema predador-presa foi proposta por A. Lotka e V. Volterra na década de 1920. O modelo de Lotka-Volterra consiste em *duas* equações diferenciais acopladas:

$$\begin{aligned}\frac{dx}{dt} &= Ax - Bxy \\ \frac{dy}{dt} &= -Cy + Dxy\end{aligned}$$

onde  $x(t)$  é o número de presas,  $y(t)$ , o de predadores, e  $A$ ,  $B$ ,  $C$  e  $D$  são constantes positivas. O termo  $Ax$  na primeira equação corresponde a um crescimento malthusiano das presas: na ausência de predadores, as presas multiplicam-se indefinidamente. O termo  $-Bxy$  limita essa expansão – quanto maior o número  $y$  de predadores, menor será a taxa de crescimento das presas. Para um número alto de predadores,  $y > A/B$ , a taxa de crescimento da população de presas ficará negativa, ou seja, as presas vão diminuir. Na segunda equação, o termo negativo  $-Cy$  produz uma extinção exponencial do número de predadores (“crescimento” malthusiano negativo): na ausência de presas, os predadores morrem de fome. O termo positivo  $Dxy$  compensa essa tendência – quanto mais presas existirem para serem devoradas, maior será a taxa de crescimento dos predadores (menos morrem de fome). Se o número de presas é alto,  $x > C/D$ , a taxa de crescimento fica positiva e o número de predadores aumenta com o tempo. Observe que se  $x = C/D$  e  $y = A/B$ , as taxas de crescimento de ambas as espécies são nulas; essas populações são estáveis, não aumentam nem diminuem com o passar do tempo.

Vamos usar o *Modellus* para verificar se esse modelo de predador-presa é capaz de descrever os aspectos gerais da população de lincês e lebres, que descrevemos anteriormente. A **Figura 6.11** mostra as equações de Lotka-Volterra escritas na janela *Modelo*.

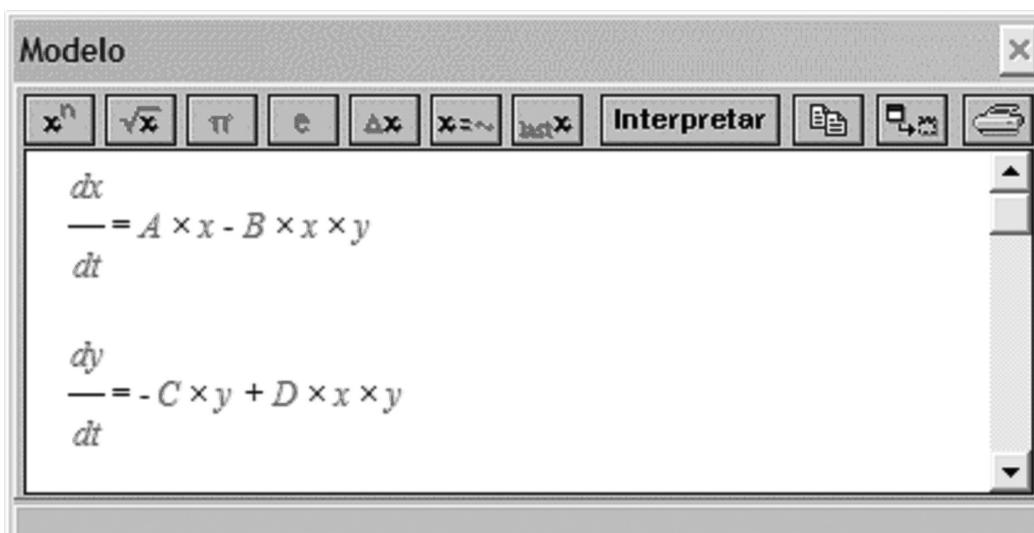
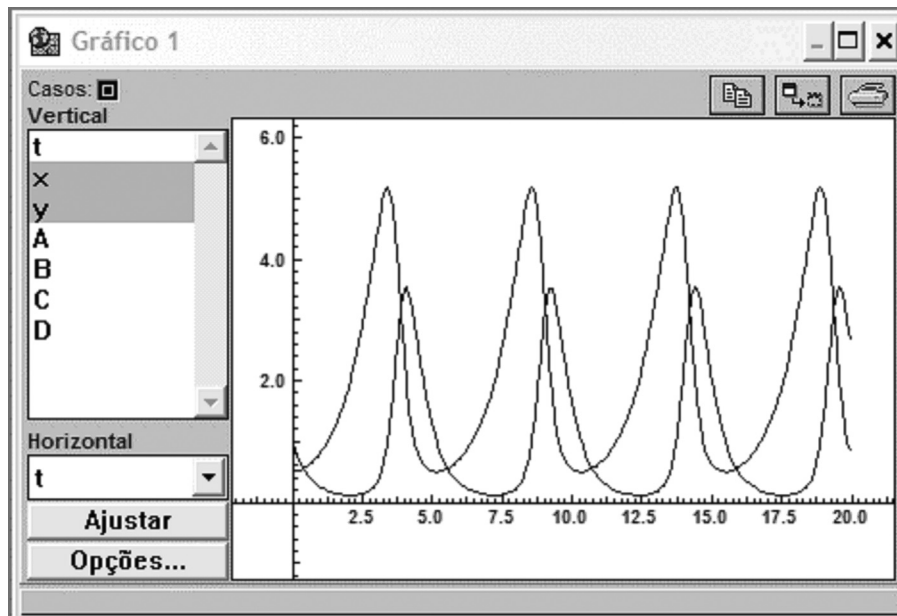

$$\frac{dx}{dt} = A \times x - B \times x \times y$$
$$\frac{dy}{dt} = -C \times y + D \times x \times y$$

Figura 6.11: Equações de Lotka-Volterra.

Para fazer uma simulação, escolha  $A = 1$ ,  $B = 1$ ,  $C = 2$  e  $D = 1$  (não há nada de muito especial nesses valores). Use as condições iniciais  $x(0) = 0.5$  e  $y(0) = 1$  (lembre que as unidades de  $x$  e  $y$  não precisam ser um indivíduo). Ao executar a simulação, você deverá encontrar algo semelhante ao que está na **Figura 6.12**, que mostra os gráfico de  $x(t)$  e  $y(t)$  (as curva de maior e menor amplitude, respectivamente). Observe a semelhança qualitativa entre os resultados do modelo e o que ocorre com os linces e lebres da **Figura 6.10**: as populações oscilam com um período bem definido, e os predadores estão um pouco “atrasados” em relação às presas.



**Figura 6.12:** Predadores e presas no modelo de Lotka-Volterra.

Os recursos gráficos do *Modellus* permitem apresentar os resultados da simulação de uma forma muito interessante: no *espaço de fase*, onde os eixos são as populações  $y$  e  $x$ . Nesse “espaço”, a evolução do sistema predador-presa dá-se sobre um ciclo fechado, como está mostrado na **Figura 6.13**. É fácil obter o gráfico no espaço de fase: basta escolher  $y$  para o eixo vertical e  $x$  para o horizontal, como se vê do lado esquerdo da **Figura 6.13**. Ao executar a simulação, você verá o sistema percorrer várias vezes o ciclo no espaço de fase, sempre no sentido anti-horário.

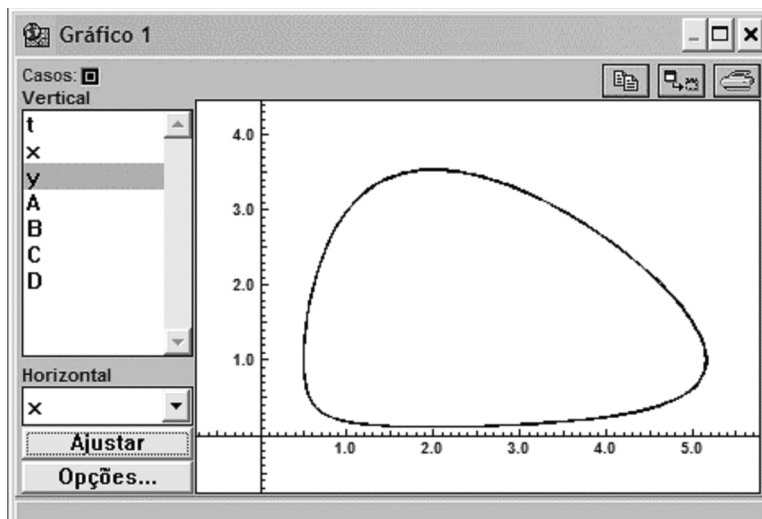


Figura 6.13: Evolução do sistema predador-presa no espaço de fase  $y \times x$ .

A Figura 6.14 mostra três ciclos diferentes, construídos adicionando novos *Casos* ao primeiro, correspondentes a um número inicial de presas  $x(0) = 0.5, 1.0$  e  $1.5$ . O número inicial de predadores é  $y(0) = 1$  nos três casos. Observe como todos os ciclos giram em torno de um mesmo ponto. Esse é justamente o ponto que discutimos anteriormente, de coordenadas  $(x, y) = (C/D, A/B)$ , no qual as taxas de crescimento se anulam e as populações permanecem fixas. Os matemáticos chamam isso de um *ponto fixo* e, no nosso caso, ele está em  $(x, y) = (2, 1)$ .

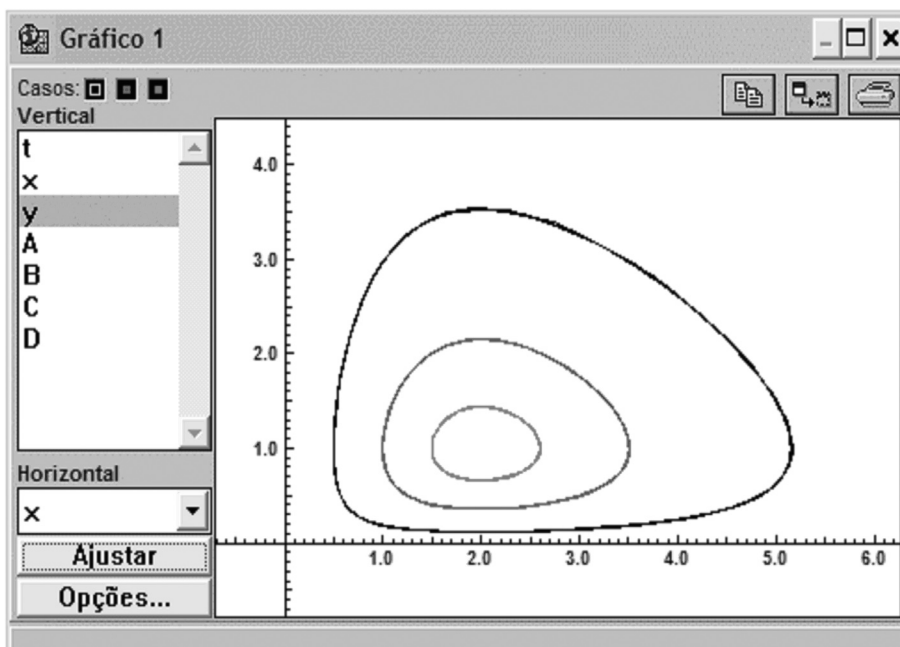


Figura 6.14: Diferentes ciclos do mesmo sistema predador-presa.

## LOTKA, VOLTERRA E VERHULST

O modelo de Lotka-Volterra não considera as restrições ambientais ao crescimento das espécies. A única restrição ao crescimento das presas vem da presença dos predadores, e a única restrição ao crescimento dos predadores é a falta de presas. Podemos mudar um pouco essa situação colocando um termo de Verhulst no sistema de equações de Lotka-Volterra. Para simplificar, vamos fazer isso apenas com as presas, já que são essas que tendem a crescer de maneira malthusiana. Um modelo de Lotka-Volterra-Verhulst seria então:

$$\frac{dx}{dt} = Ax(1 - x/K) - Bxy$$

$$\frac{dy}{dt} = -Cy + Dxy$$

onde  $K$  é, como antes, a capacidade de suporte do território ocupado pelas presas. O efeito que essa modificação tem sobre a dinâmica é surpreendente. Para ver isso, programe essas equações no *Modellus*, escolha os mesmos parâmetros que usamos anteriormente ( $A = 1$ ,  $B = 1$ ,  $C = 2$  e  $D = 1$ ) e as mesmas condições iniciais ( $x(0) = 0.5$  e  $y(0) = 1$ ). Use o valor  $K = 10$  para a capacidade de suporte. Ao rodar a simulação, você vai encontrar algo semelhante ao que está na **Figura 6.15**. Note como as oscilações típicas do sistema de Lotka-Volterra ficam amortecidas e como o sistema tende para um ponto fixo, com populações estáveis. O gráfico no espaço de fase (**Figura 6.16**) também é interessante, mostrando como o sistema atinge o equilíbrio em uma trajetória espiral.

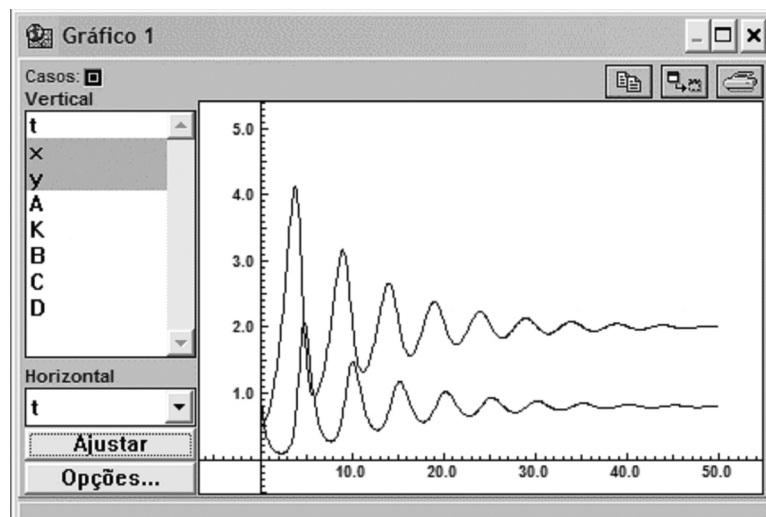


Figura 6.15: Sistema predador-presa com capacidade de suporte finita.



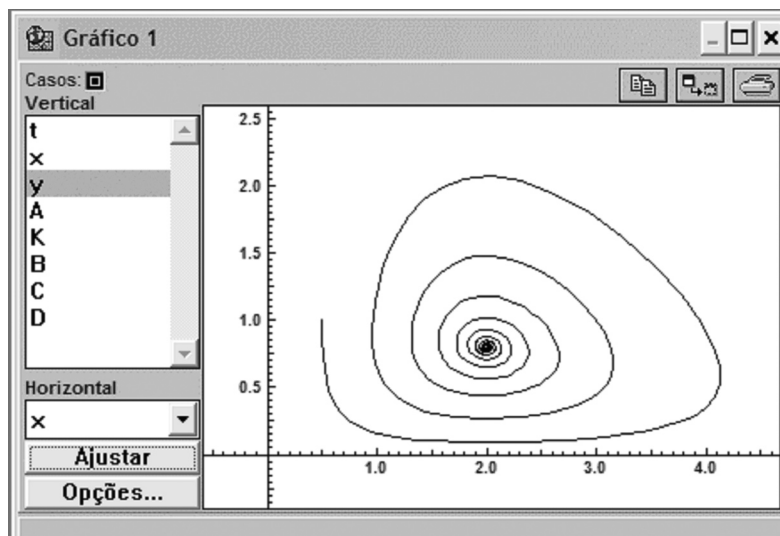


Figura 6.16: O mesmo sistema da figura anterior, mostrado no espaço de fase.

### INFORMAÇÃO SOBRE A PRÓXIMA AULA

Na próxima aula, vamos usar o que aprendemos sobre o *Modellus* e equações diferenciais para estudar um pouco de mecânica newtoniana.

## O oscilador harmônico

### Meta da aula

Mostrar como resolver problemas de mecânica clássica utilizando o *Modellus*, tendo como exemplo o oscilador harmônico.

# objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- resolver equações de movimento de sistemas clássicos de uma partícula com o *Modellus*;
- representar graficamente a trajetória da partícula, inclusive no espaço de fase;
- produzir animações mostrando o movimento da partícula.

## DINÂMICA NEWTONIANA NO *MODELLUS*

Na Mecânica clássica, a dinâmica de uma partícula é determinada pela segunda lei de Newton,

$$F = ma ,$$

onde  $F$  é a força que atua sobre a partícula, e  $m$  e  $a$  são a sua massa e aceleração, respectivamente. A força  $F$  pode depender da posição  $x$  da partícula, da velocidade  $v = dx/dt$  e do tempo  $t$ ; em linguagem matemática,  $F = F(x, v, t)$ . Para simplificar, estamos supondo que o movimento se dá em uma dimensão de coordenada  $x$ . Como a aceleração é  $a = dv/dt = d^2x/dt^2$ , a lei de Newton é uma equação diferencial de segunda ordem (a *equação de movimento* da partícula):

$$\frac{d^2x}{dt^2} = \frac{1}{m} F\left(x, \frac{dx}{dt}, t\right)$$

A solução dessa equação é  $x(t)$ , a função que dá a posição da partícula como função do tempo. O *Modellus* não opera diretamente com segundas derivadas, de modo que não podemos usá-lo para resolver a equação de movimento – pelo menos não na forma como ela está escrita. Entretanto, é fácil reescrever a segunda lei de Newton como um par de equações diferenciais de primeira ordem:

$$\begin{aligned} \frac{dv}{dt} &= \frac{1}{m} F(x, v, t) \\ \frac{dx}{dt} &= v \end{aligned}$$

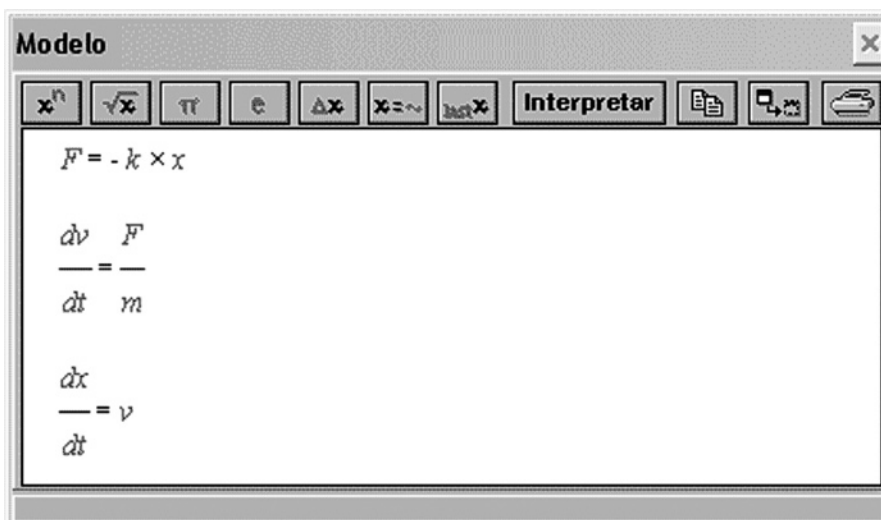
Essa forma é equivalente à anterior e usa apenas derivadas simples – portanto, ela pode ser programada e resolvida no *Modellus*. Vamos ver como isso é feito em um caso muito importante: o oscilador harmônico.

### O oscilador harmônico simples

Osciladores harmônicos são bastante comuns. Se um sistema está vibrando com pequena amplitude em torno de um ponto de equilíbrio, ele provavelmente executa um movimento harmônico. O protótipo de oscilador harmônico é o sistema massa-mola, em que uma partícula de massa  $m$  está presa a uma mola ideal, perfeitamente elástica. A força que a mola faz sobre a partícula é dada pela lei de Hooke,

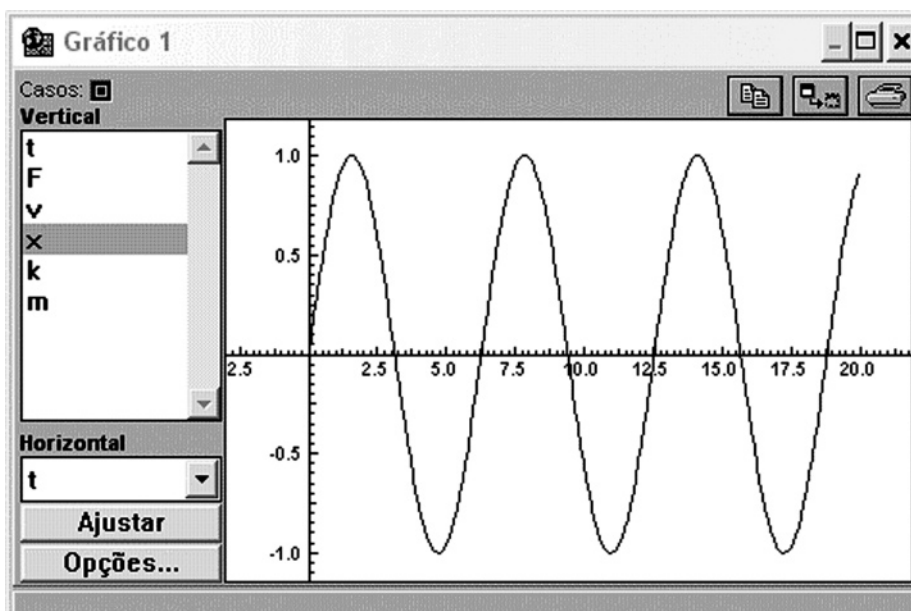
$$F = -kx,$$

em que  $x$  é o deslocamento da partícula (relativo ao ponto de equilíbrio, onde  $F = 0$ ) e  $k$  é a “constante elástica” da mola. Para estudar o oscilador harmônico com o *Modellus*, vamos primeiro escrever as equações de movimento na janela *Modelo*, na maneira mostrada na **Figura 7.1**.



**Figura 7.1:** Modelo de oscilador harmônico.

Interprete o modelo e, na janela *Condições Iniciais*, defina  $m = 1$ ,  $k = 1$ ,  $x(0) = 0$  e  $v(0) = 1$ . Execute a simulação e observe o gráfico de  $x(t)$ , que deve ser semelhante ao que está na **Figura 7.2**.



**Figura 7.2:** Gráfico de  $x(t)$ .

Note como o sistema oscila com período e amplitude bem definidos. Uma pergunta importante é: O período das oscilações depende da amplitude? Para investigar essa questão, crie novos *Casos* com os mesmos  $k$  e  $m$  (ou seja, é o mesmo oscilador) e condições iniciais que gerem amplitudes diferentes. Um exemplo está na **Figura 7.3**, na qual definimos dois novos casos mudando a velocidade inicial. A **Figura 7.4** mostra o movimento da partícula nos três casos. Observe que, embora a amplitude de oscilação mude bastante de um caso para outro, o período permanece igual.

| Condições Iniciais |        |        |        |
|--------------------|--------|--------|--------|
| Parâmetros         |        |        |        |
|                    | caso 1 | caso 2 | caso 3 |
| $k$                | 1.0    | 1.0    | 1.0    |
| $m$                | 1.0    | 1.0    | 1.0    |
| Valores Iniciais   |        |        |        |
|                    | caso 1 | caso 2 | caso 3 |
| $v$                | 1.0    | 2.0    | 3.0    |
| $x$                | 0.0    | 0.0    | 0.0    |

Figura 7.3: Três casos com diferentes condições iniciais.

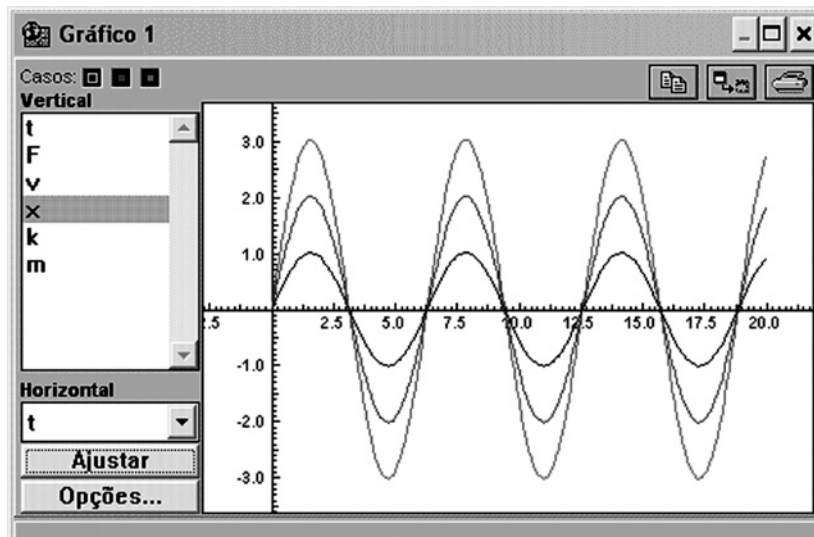
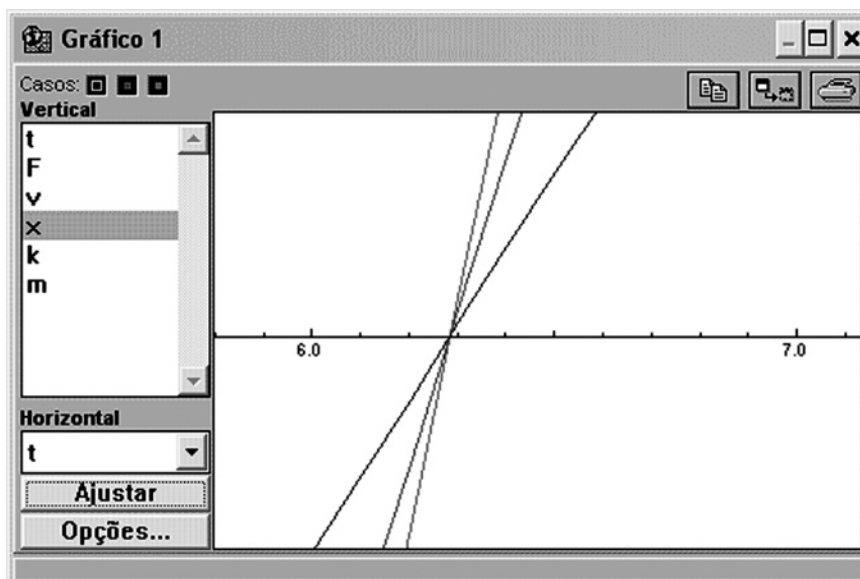


Figura 7.4: Movimentos nos três casos definidos na Figura 7.3.

Os resultados da **Figura 7.4** mostram um resultado fundamental sobre o oscilador harmônico: o período não depende da amplitude do movimento. É claro que a observação de três exemplos não constitui uma demonstração formal, mas a conclusão de que o período independe da amplitude é bem razoável. Se o oscilador harmônico tem um período característico, a próxima pergunta relevante é: Quanto vale esse período? A resposta está na **Figura 7.5**, em que fizemos um *zoom* sobre o final do primeiro ciclo mostrado na **Figura 7.4**. Vemos que as três curvas saem de  $x = 0$  e, após uma oscilação completa, passam novamente por esse ponto exatamente no mesmo instante. Esse é o período da oscilação, e a **Figura 7.5** mostra que ele vale  $T \approx 6.28$ .



**Figura 7.5:** Zoom sobre a **Figura 7.4**, mostrando que o período de oscilação é  $T \approx 6.28$ .

O oscilador que estamos considerando tem  $k = 1$  e  $m = 1$ . Qual seria o período de um oscilador diferente? A resposta é obtida com um pouco de análise dimensional. O período  $T$  é dado em segundos (vamos usar as unidades do SI) e deve ser determinado pelas características do oscilador, ou seja, por  $k$  e  $m$ . A unidade de  $k$  é  $\text{kg/s}^2$ , e a de massa é o kg. A única combinação possível dessas quantidades que tem unidade de tempo é  $\sqrt{m/k}$ . Portanto, o período do oscilador harmônico deve ser dado por

$$T = c\sqrt{m/k},$$

em que  $c$  é uma constante sem dimensão. Para determinar essa constante, basta lembrar que o oscilador com  $k = 1$  e  $m = 1$  que tratamos anteriormente tem  $T \approx 6.28$ . Portanto,

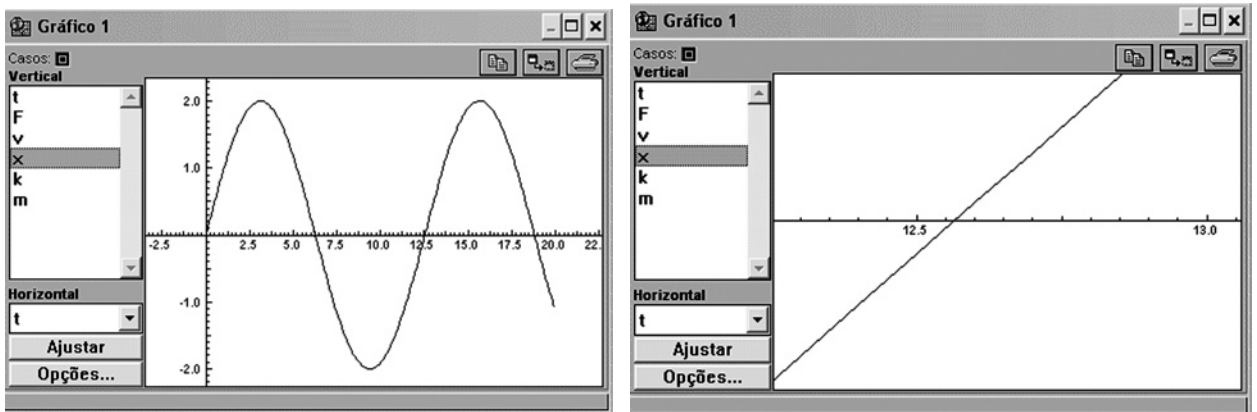
$$c \approx 6.28.$$

O resultado exato para o período do sistema massa-mola é bem conhecido:

$$T = 2\pi\sqrt{m/k},$$

ou seja, o valor da constante adimensional é  $c = 2\pi = 2 \times (3.14159\dots) = 6.28318\dots$ , bem próximo do que obtivemos com o *Modellus*.

A discussão que acabamos de fazer mostra o quanto a análise dimensional pode ser útil quando estudamos um sistema físico com auxílio do computador; ao calcular o período de um oscilador específico, obtivemos o período de todos os outros. Se você ainda não acredita que isso seja possível, teste o resultado que encontramos mudando os valores de  $k$  e/ou  $m$ . Por exemplo, se  $T \propto \sqrt{m/k}$ , fazendo  $m = 4$ , devemos ter um período duas vezes maior que o de  $m = 1$ . A **Figura 7.6** mostra que isso realmente ocorre: o período agora é  $T \approx 12.56$ .



**Figura 7.6:** Oscilador harmônico com  $k = 1$  e  $m = 4$ . À esquerda está o gráfico do movimento da partícula; do lado direito está um zoom sobre o final da primeira oscilação, mostrando que o período é  $T \approx 12.56$ .

Podemos também investigar a velocidade do movimento harmônico. O gráfico de  $v(t)$  é facilmente obtido e comparado ao de  $x(t)$ , como mostra a Figura 7.7. Observe que a velocidade apresenta uma oscilação semelhante à da posição, mas “adiantada” em  $1/4$  de período.

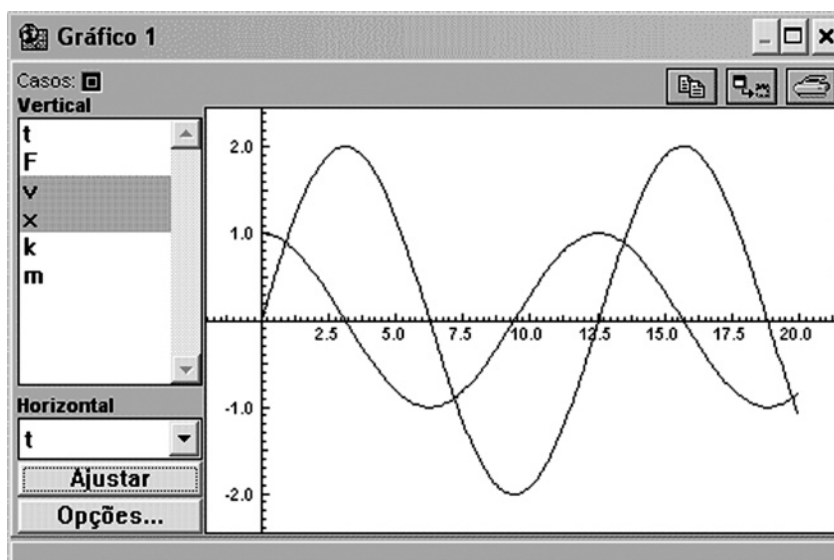


Figura 7.7: Gráficos de  $x$  e  $v$  (as curvas de maior e de menor amplitude, respectivamente).

Outra representação muito útil do movimento harmônico é obtida no *espaço de fase*, ou seja, fazendo-se o gráfico de  $v \times x$ . A Figura 7.8 mostra o mesmo movimento representado na Figura 7.7, desta vez no espaço de fase (note as escolhas dos eixos vertical e horizontal). A trajetória do oscilador harmônico no espaço  $v \times x$  tem a forma de uma elipse, com centro no ponto de equilíbrio  $x = v = 0$ , percorrida no sentido horário à medida que o tempo passa. Movimentos com amplitudes diferentes correspondem a diferentes elipses no espaço de fase, como pode ser facilmente verificado criando novos *Casos* no *Modellus*.



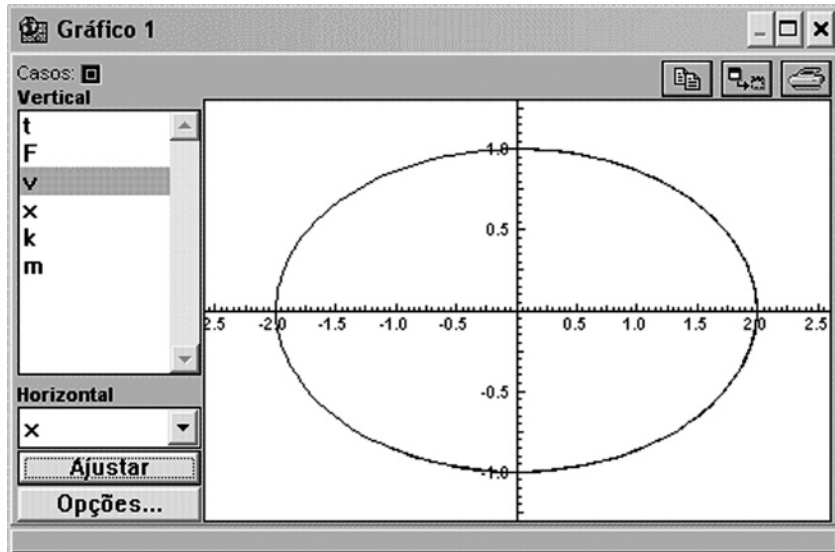


Figura 7.8: O oscilador harmônico no espaço de fase.

### Energia do oscilador harmônico

A energia do oscilador harmônico é a soma das energias cinética,  $E_{cin} = \frac{1}{2}mv^2$ , e potencial,  $E_{pot} = \frac{1}{2}kx^2$ . Vamos incluir o cálculo dessas energias no nosso modelo, como está na Figura 7.9, e verificar como elas se comportam.

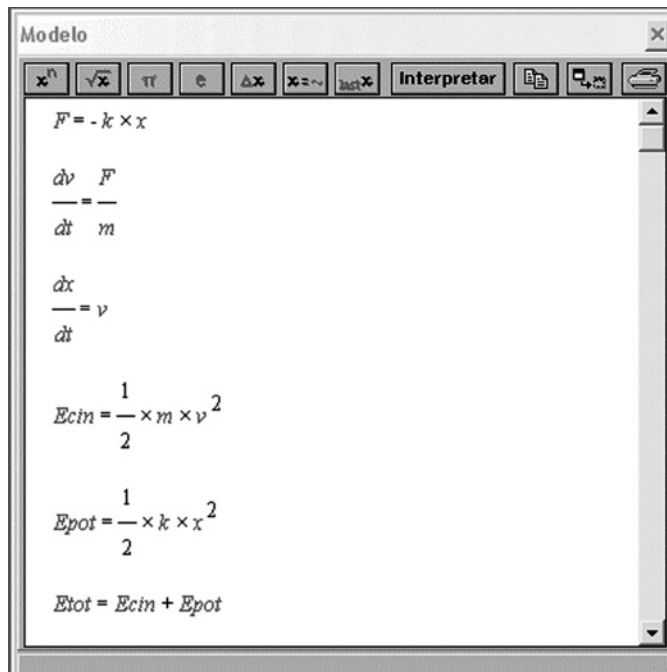


Figura 7.9: Cálculo da energia no modelo de oscilador harmônico.

Com essa mudança, podemos fazer os gráficos das energias cinética e potencial em função do tempo. Os resultados (com  $k = 1$  e  $m = 4$ ) estão na Figura 7.10. Vemos que nenhuma das duas energias é constante: elas aumentam e diminuem à medida que o tempo passa. Por outro lado, como mostra a Figura 7.11, a energia total mantém-se inalterada – no jargão dos físicos, ela é uma “constante de movimento”.

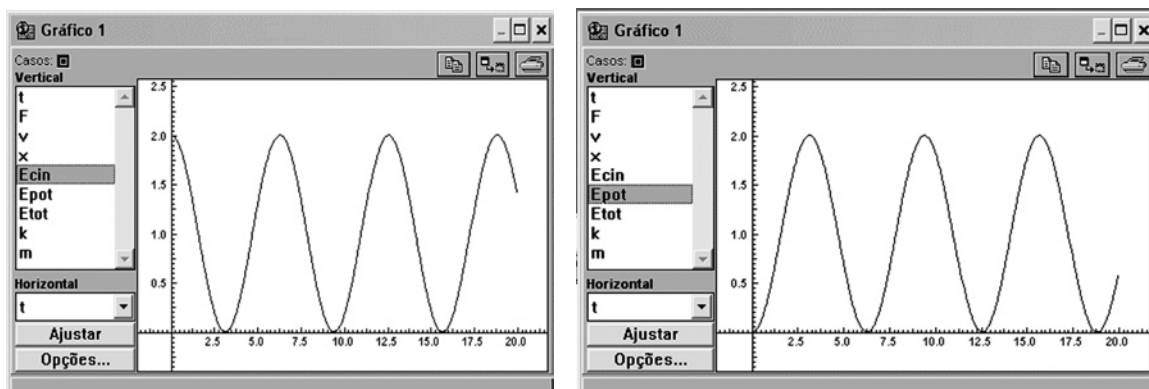


Figura 7.10: Energias cinética (esquerda) e potencial (direita) do oscilador harmônico.

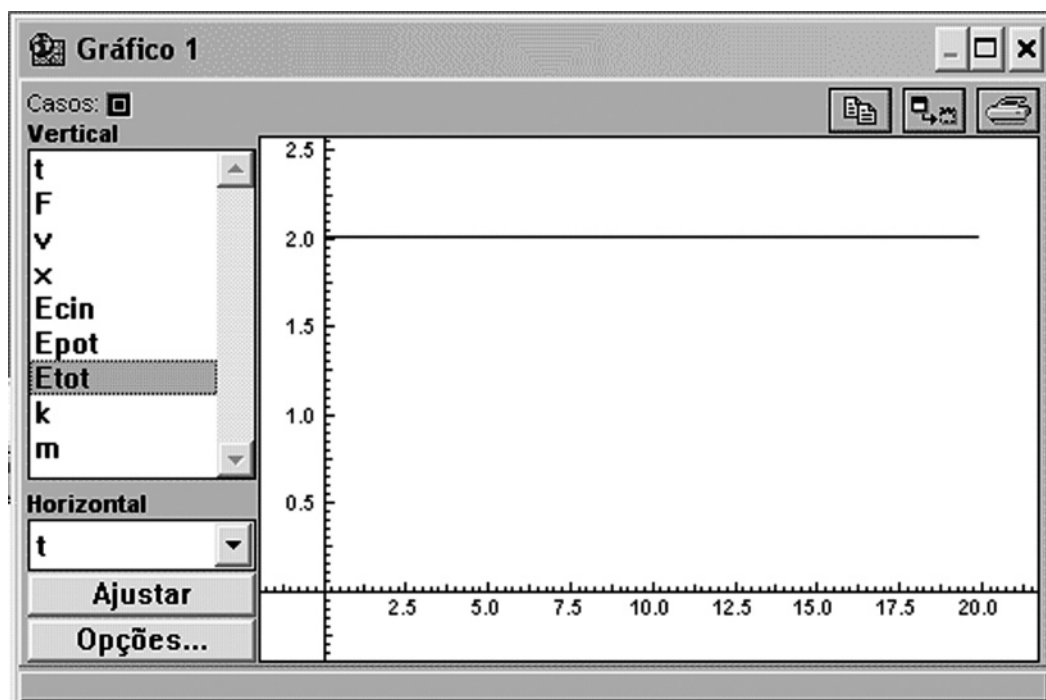


Figura 7.11: Conservação da energia no movimento harmônico.

### Animando o movimento harmônico...

Podemos usar os recursos de animação do *Modellus* para criar representações alternativas dos resultados anteriores. Por exemplo, a conservação de energia pode ser apresentada de forma muito interessante, usando-se os objetos do tipo *Barra* da janela *Animação*. A *Barra* é um “medidor de nível” que permite ilustrar as transformações da energia de maneira muito sugestiva. Para colocar uma barra na janela de animações, aperte o botão correspondente (logo abaixo do botão de vetores) e, em seguida, clique em um ponto do interior da janela. Um objeto *Barra* será criado aí, e uma (já familiar) caixa de diálogo se abrirá, pedindo as propriedades desse objeto. Defina algo semelhante ao que está na **Figura 7.12** – o nível da barra é dado pela energia cinética  $E_{cin}$ , os níveis mínimo e máximo são 0 e 3 (no gráfico da **Figura 7.10**, a energia cinética varia de 0 a 2) etc. – e clique em *OK*. Você verá surgir na janela *Animação* uma barra cuja altura é determinada pela energia cinética do oscilador. Note que a posição e a altura dessa barra podem ser mudadas com o *mouse*. Repetindo o procedimento anterior, crie mais duas barras: uma representando a energia potencial  $E_{pot}$  e a outra, a energia total  $E_{tot}$ . O resultado deve ficar parecido com o que está na **Figura 7.13**. Ao executar a simulação, os “níveis” de energia cinética e potencial sobem e descem, enquanto a energia total permanece estável.

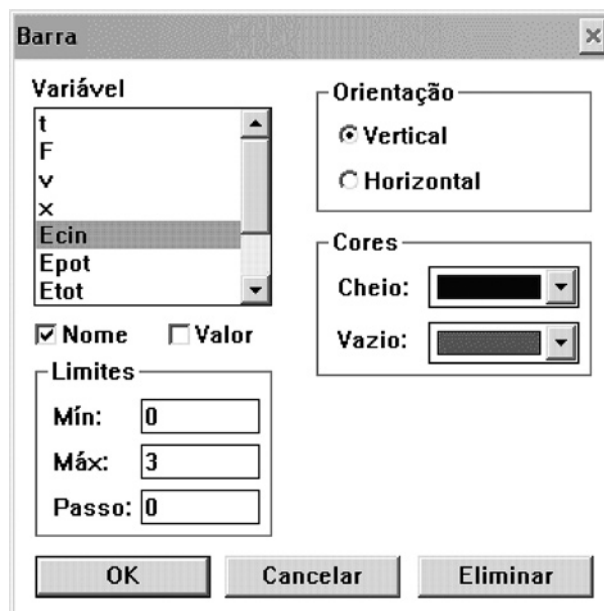
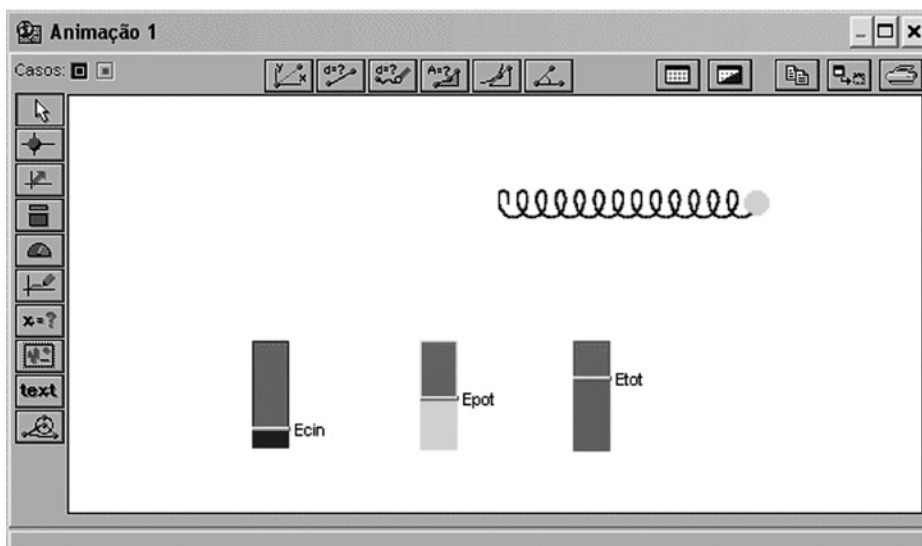


Figura 7.12: Propriedades do objeto *Barra*.



**Figura 7.13:** Animação do movimento harmônico. As barras representam as energias cinética, potencial e total. A mola é uma figura que segue o movimento do oscilador.

Além das barras, a **Figura 7.13** mostra o desenho de uma mola na janela *Animação*. Quando a simulação é executada, o comprimento da mola aumenta e diminui, acompanhando o movimento do oscilador harmônico. Para construir esse tipo de representação “realista” do sistema massa-mola, temos de aprender a importar e manipular imagens com o *Modellus*.

A primeira coisa a fazer é encontrar a figura que será colocada na janela de animação. O programa pode importar para essa janela qualquer imagem em formato *bmp* ou *gif*. Na página do *Modellus*, <http://phoenix.sce.fct.unl.pt/modellus>, estão disponíveis dezenas de figuras que podem ser utilizadas em animações (procure em *Downloads*). O arquivo com essas imagens está em:

```
http://phoenix.sce.fct.unl.pt/modellus/download/modellus2/images.zip
```

Baixe esse arquivo e extraia as figuras com um utilitário de descompressão (note a terminação *zip*). Coloque todas as imagens numa pasta com nome (*imagens\_modellus*, por exemplo) e localização fáceis de lembrar. Dê uma olhada no conteúdo da pasta. Você encontrará desenhos

de carros, bolas, planetas, circuitos elétricos etc. Lá está, inclusive, a mola que mostramos na **Figura 7.13**, no arquivo *springh.bmp*.

O próximo passo será colocar o desenho da mola na janela de animações. Para isso, use o botão de “inserir nova imagem” (é o terceiro, de baixo para cima). Aperte o botão e, em seguida, clique no interior da janela. Ao abrir-se a caixa de diálogo com as propriedades da imagem (veja a **Figura 7.14**), clique no botão *Procurar* e localize o arquivo que tem o desenho da mola (lembre: *springh.bmp*, na pasta de imagens que você acabou de criar). Com esse arquivo escolhido, feche a caixa de diálogo clicando em *OK*. Se tudo der certo, você verá o desenho da mola na janela *Animação*.

O desenho que acabamos de inserir é estático: seu tamanho e posição não mudam com o tempo. Como queremos que a mola represente o oscilador, temos de fazer com que o comprimento da imagem acompanhe o movimento do sistema, aumentando e diminuindo à medida que o tempo passa. Para isso, vamos editar as propriedades da imagem, clicando sobre ela com o botão direito do *mouse*. Quando a caixa de diálogo da imagem for aberta, marque a opção *Dimensão* no painel *Varição*; isso fará com que o tamanho da figura passe a ser determinado pelas variáveis do modelo. Coloque a variável  $x$  na dimensão *Horizontal*, com escala de 0.01 por *pixel* (quando  $x = 1$ , a mola terá 100 *pixels* de comprimento). Tudo isso está mostrado na **Figura 7.14**.

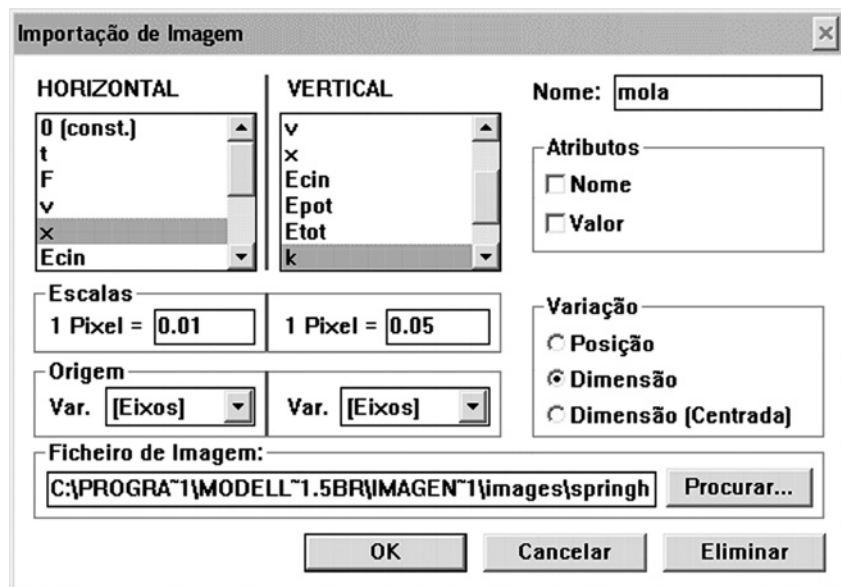


Figura 7.14: Propriedades do objeto *Imagem*.

Também temos de fixar a dimensão vertical do desenho, pois, se deixarmos o valor nulo colocado inicialmente pelo *Modellus*, a figura da mola terá altura zero. Como o diâmetro da mola deve ficar inalterado durante a animação, podemos escolher qualquer grandeza constante já definida no modelo para representar a altura da imagem. Na **Figura 7.14**, fixamos a dimensão *Vertical* da imagem com a constante  $k$  e escala 0.05. Como  $k = 1$ , isso dá uma altura de 20 *pixels* à imagem (outra possibilidade seria criar, no modelo, uma variável  $d = 20$  e colocá-la na dimensão *Vertical* da imagem, com escala igual a 1). Defina as outras propriedades da imagem como está na **Figura 7.14** e clique em **OK**. Execute a simulação e, se tudo deu certo, observe como a mola é esticada e comprimida, acompanhando o movimento do oscilador.

Para completar a animação, vamos colocar a partícula de massa  $m$  na extremidade da mola. Já sabemos fazer isso; crie um objeto *Partícula* cuja posição horizontal seja dada pela variável  $x$ . Use a mesma escala empregada na dimensão horizontal da imagem: 0.01 por *pixel*. Desmarque todos os atributos (*Nome*, *Eixos* etc.) e dê um nome ao objeto (por exemplo, “massa”). Ao executar a simulação, você verá a massa e a mola oscilarem em conjunto. Use o *mouse* para colocar a partícula sobre a extremidade da mola, de modo que elas pareçam ligadas, como está na **Figura 7.13**. Execute novamente a simulação e observe o sistema massa-mola construído por você oscilar de acordo com as equações do modelo. É instrutivo acompanhar o que ocorre com as energias cinética e potencial (indicadas nas barras de nível) enquanto a mola oscila de um lado para o outro.

O procedimento anterior pode parecer trabalhoso (com um pouco de prática, ele deixa de ser), mas ilustra bem os recursos de que o *Modellus* dispõe para a construção de múltiplas representações do mesmo objeto. Com o que fizemos até agora, o oscilador ficou descrito pelas equações de movimento da janela *Modelo*, pelos gráficos de  $x(t)$  e  $v(t)$  na janela *Gráfico* e pela construção “realista” da massa e mola na janela *Animação*, isso sem falar nas descrições baseadas na energia ou no espaço de fase. Todas essas representações são, em certo sentido, equivalentes, mas cada uma revela um aspecto distinto e importante do objeto estudado (o oscilador harmônico). Não é fácil encontrar ferramentas didáticas que permitam reunir, em um mesmo espaço, tantas visões diferentes e complementares sobre um mesmo assunto.

## Oscilador harmônico amortecido

Os osciladores encontrados no mundo real não oscilam indefinidamente. Efeitos dissipativos (forças de atrito, resistência do ar etc.) estão sempre em ação, “amortecendo” o movimento até que este cesse por completo. Se a força dissipativa for proporcional à velocidade, a força resultante sobre o oscilador harmônico será

$$F = -kx - bv.$$

Vamos usar o *Modellus* para estudar os efeitos da dissipação sobre o oscilador harmônico. É fácil incluir a força dissipativa no modelo que temos usado até agora: basta somar o termo  $-bv$  à força de Hooke, na primeira linha mostrada na **Figura 7.1**. Interpretando esse modelo e usando  $b = 0.5$  para o “coeficiente de atrito”, obtemos o movimento mostrado na **Figura 7.15** (ainda estamos usando  $k = 1$  e  $m = 4$ , e note que o tempo vai agora até  $t = 100$ ). Observe como a dissipação amortece as oscilações e o sistema aproxima-se gradativamente do ponto de equilíbrio.

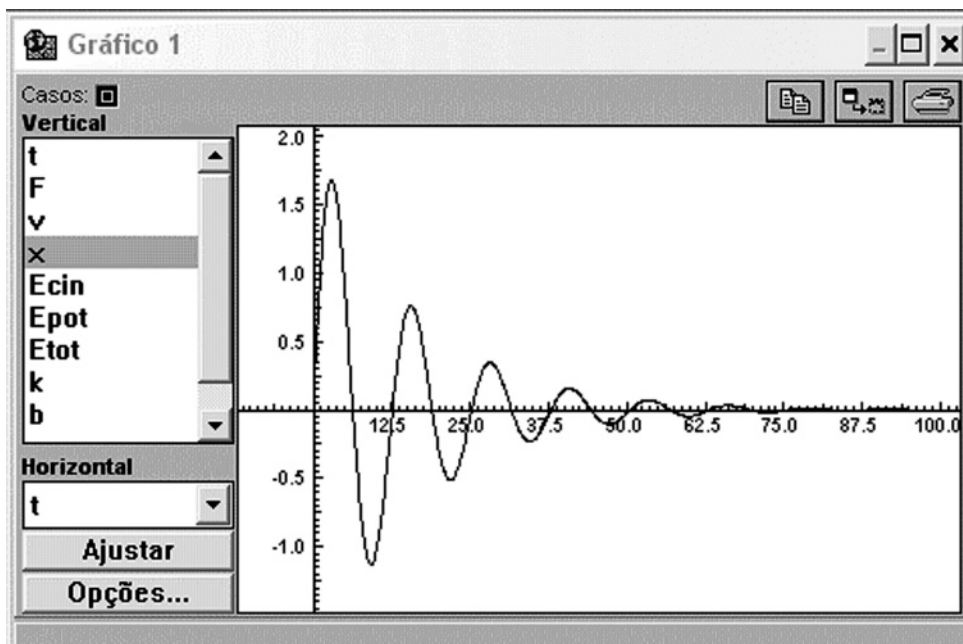
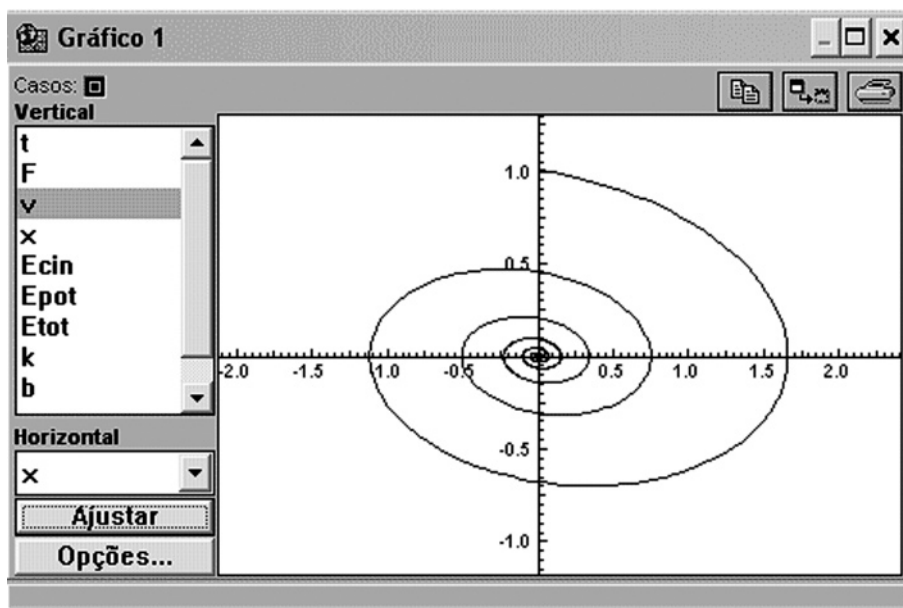


Figura 7.15: Movimento harmônico amortecido.

O gráfico do movimento amortecido no espaço de fase  $v \times x$  também é interessante. A **Figura 7.16** mostra que, nesse caso, a trajetória é uma espiral que converge para o ponto  $(x, v) = (0, 0)$ . É instrutivo comparar esse resultado com o da **Figura 7.8**, que mostra o movimento não amortecido no mesmo espaço.



**Figura 7.16:** Movimento no espaço de fase de um oscilador harmônico amortecido.

Na presença de efeitos dissipativos, a energia do oscilador não é mais conservada. A **Figura 7.17** mostra como a energia diminui com o tempo. Podemos notar que a queda se dá em “degraus”: a energia diminui rapidamente em alguns instantes e, em outros, ela varia muito pouco. Embora esse comportamento possa parecer estranho, não é difícil compreendê-lo, basta lembrar que a energia é dissipada pelo trabalho da força de atrito. Quando  $v = 0$ , o trabalho é nulo (nesse caso, tanto a força de atrito quanto o deslocamento são nulos) e, portanto, a energia não diminui. É isso que cria os “patamares” da **Figura 7.17**: eles coincidem com os pontos de retorno, quando o estiramento da mola cessa e tem início a contração. As regiões de queda rápida da energia ocorrem nos pontos de alta velocidade, ou seja, quando a partícula passa próximo à posição de equilíbrio ( $x = 0$ ).



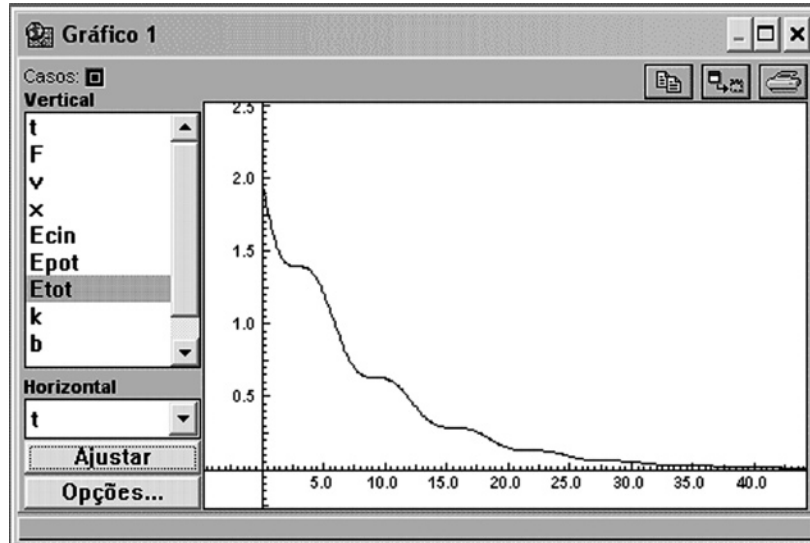


Figura 7.17: Dissipação da energia no movimento harmônico amortecido.

### Atividade: superamortecimento

Se o amortecimento for muito intenso, o movimento do sistema massa-mola deixa de ser oscilatório. Investigue o que acontece com o sistema que já estudamos ( $k = 1$  e  $m = 4$ ) para  $b = 4$  e  $b = 8$ . Os resultados estão na Figura 7.18. Note como a massa volta à posição de equilíbrio sem oscilar. O caso com  $b = 8$  é um exemplo de movimento superamortecido, que ocorre quando  $b^2 > 4mk$ . Com  $b = 4$ , o sistema tem *amortecimento crítico*, que ocorre quando  $b^2 = 4mk$ . Sistemas com  $b^2 < 4mk$  são chamados subamortecidos e apresentam os movimentos oscilatórios que já estudamos.

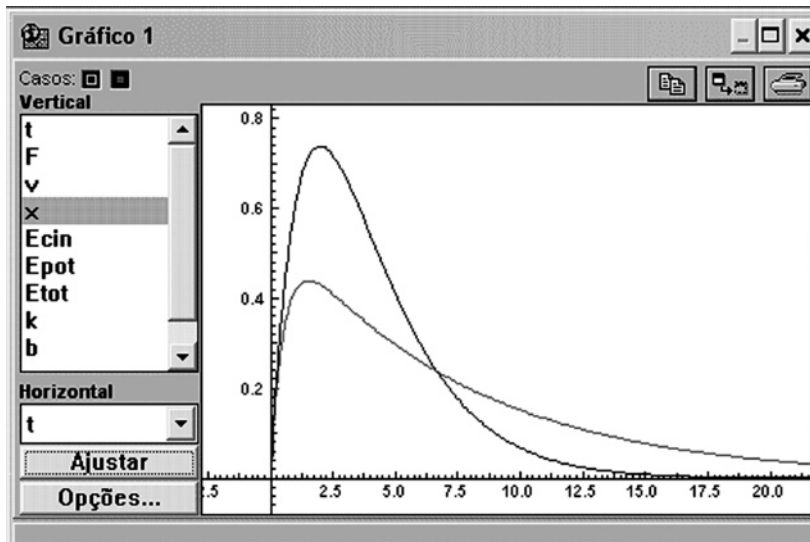


Figura 7.18: Movimento com amortecimento crítico ( $b = 4$ , curva mais estreita) e superamortecimento ( $b = 8$ , curva mais larga). Nos dois casos,  $k = 1$  e  $m = 4$ .

## INFORMAÇÃO SOBRE A PRÓXIMA AULA

Na próxima aula, vamos mudar nosso enfoque: em vez de usar o *Modellus* para fazer cálculos, vamos empregá-lo para realizar medidas.

## Medidas com o *Modellus*

AULA

8

### Meta da aula

Apresentar as ferramentas de medida disponíveis no *Modellus*.

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- fazer medidas sobre objetos colocados na janela *Animação*;
- usar as ferramentas de medida do *Modellus* para analisar fotos, gráficos e vídeos.

## O *MODELLUS* COMO INSTRUMENTO DE MEDIDA

Você já deve ter reparado que a janela *Animação* tem alguns botões, colocados na sua parte superior, que não utilizamos até agora. Eles estão mostrados na **Figura 8.1** e têm uma função muito interessante: servem para fazer medidas. Com esses botões, podemos medir coordenadas, distâncias, áreas e ângulos no interior da janela *Animação*.



**Figura 8.1:** Botões de medida.

A função de cada um dos seis botões de medida pode ser vista colocando o cursor sobre eles – uma descrição do que é medido com o botão vai aparecer na parte de baixo da janela *Animação*.

Para ganhar experiência com as ferramentas de medida do *Modellus*, vamos aprender a medir as coordenadas cartesianas de um ponto. Para isso, aperte o botão *Medir coordenadas* (é o primeiro à esquerda, com o desenho dos eixos  $x$ - $y$ ), leve o cursor até o interior da janela e clique com o botão esquerdo do *mouse*. Em seguida, mova o *mouse*, deslocando o cursor para outro local dentro da janela. Você notará que um sistema de eixos aparece, com origem no ponto em que você clicou. Observe como as coordenadas do ponto onde se encontra o cursor são dadas nos respectivos eixos cartesianos. Mude o cursor de posição e veja como as coordenadas mudam. Para fixar um ponto, clique com o botão *direito* do *mouse*; daí em diante, os movimentos não serão mais seguidos pelo sistema de medida, que permanecerá apontando o ponto clicado e suas coordenadas. A **Figura 8.2** mostra o resultado típico de uma medida de coordenadas.

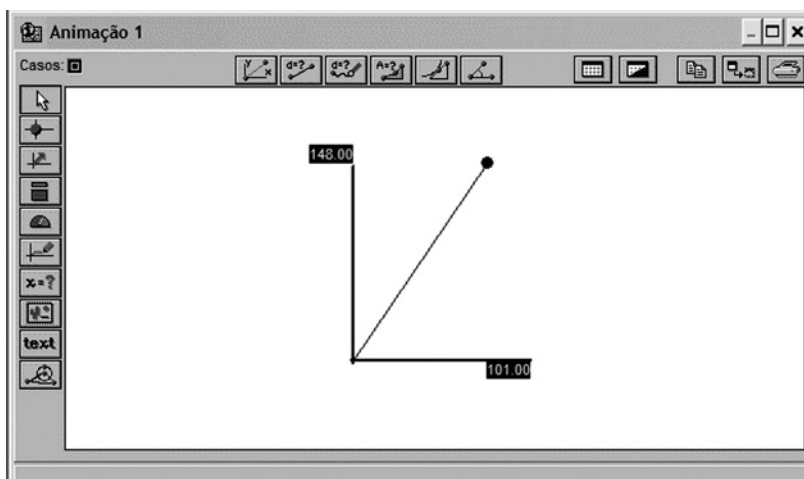


Figura 8.2: Medida das coordenadas de um ponto.

Podemos mudar facilmente a origem do sistema de eixos ou o ponto cujas coordenadas queremos medir. Coloque o cursor sobre a origem (note como ele fica diferente) e aperte o botão direito do *mouse*, sem soltá-lo; você verá que pode “carregar” o sistema de eixos para outro local da janela. Faça o mesmo com o ponto cujas coordenadas estão sendo medidas; você poderá carregá-lo para qualquer outro lugar dentro da janela.

As coordenadas mostradas pela ferramenta de medida estão dadas em *pixels*. Para mudar de escala, coloque o cursor sobre a origem ou sobre o ponto medido e clique com o botão direito do *mouse*. Isso vai abrir a caixa de diálogo mostrada na Figura 8.3, onde as escalas dos eixos horizontal e vertical podem ser mudadas. Outras opções estão disponíveis, como as cores em que as coordenadas são escritas. Clicando no botão *Eliminar*, a ferramenta de medida é apagada na janela *Animação*.

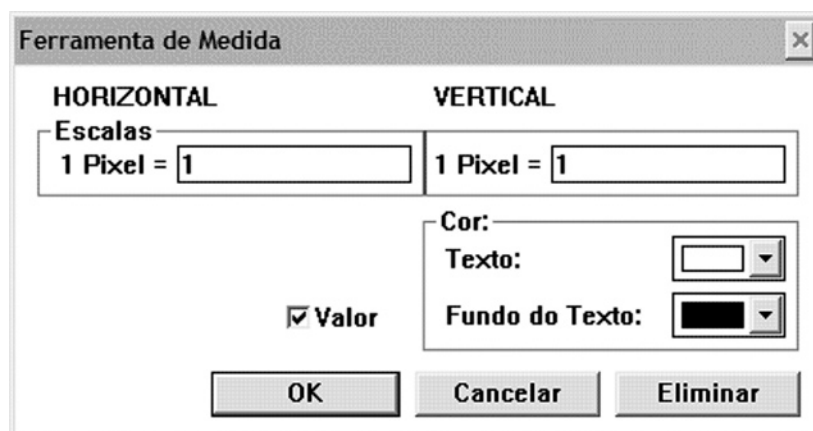
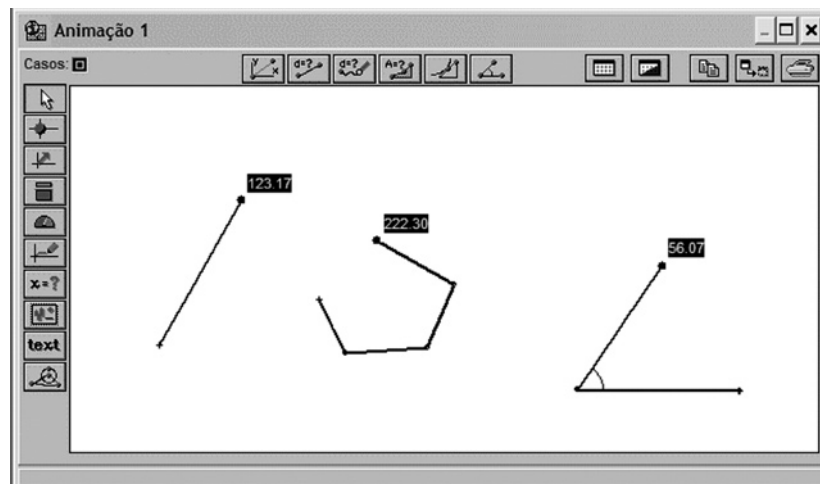


Figura 8.3: Propriedades da *Ferramenta de Medida*.

As demais ferramentas de medida do *Modellus* são utilizadas de maneira semelhante. Por exemplo, aperte o botão *Medir a distância entre dois pontos* (está logo à direita do que mede coordenadas) e, em seguida, clique em algum ponto da janela *Animação*. Movendo o *mouse*, você verá a medida da distância entre o cursor e o ponto onde clicou. Usando o botão direito do *mouse*, você pode fixar um segundo ponto e medir a distância entre este e o primeiro. A **Figura 8.4** mostra o resultado de uma dessas medidas: a linha reta conecta os pontos escolhidos e a distância entre eles aparece na etiqueta. Novamente, é possível mover os pontos marcados, carregando-os com o *mouse*.

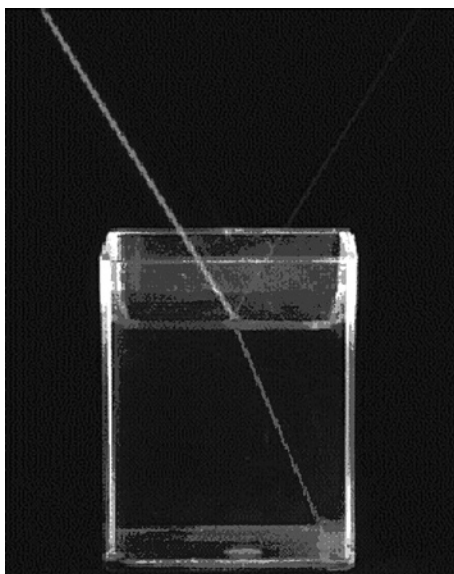


**Figura 8.4:** Medidas de distância, comprimento e ângulo.

O botão seguinte, *Medir distância sobre uma curva*, permite determinar o comprimento de uma linha poligonal. O procedimento é semelhante aos anteriores: aperte o botão de medida, escolha os pontos que definem a poligonal (usando o botão esquerdo do *mouse*) e marque o ponto final com o botão direito. A etiqueta vai fornecer o comprimento total da linha traçada. Um resultado típico está na **Figura 8.4**. A mesma figura mostra, no desenho mais à direita, uma medida de ângulo. Como nos casos anteriores, os pontos que definem o ângulo são marcados com o *mouse* e, no último (o terceiro ponto), utiliza-se o botão direito. Note que os ângulos são medidos em graus. As escalas de comprimento (ou a opção graus/radianos para ângulos) podem ser alteradas clicando-se com o botão direito do *mouse* sobre um dos pontos marcados, exatamente como fizemos no caso da medida de coordenadas.

## MEDIDA DE UM ÍNDICE DE REFRAÇÃO

Vamos usar o que aprendemos na seção anterior para estudar um pouco de ótica. A **Figura 8.5** mostra a refração sofrida por um raio luminoso ao passar do ar para a água (o raio refletido também pode ser notado). Vamos usar essa foto para medir o índice de refração da água.

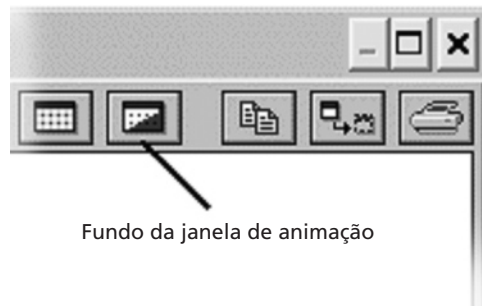


**Figura 8.5:** Refração de um raio de luz.

Para fazer a medida, precisamos primeiro obter a foto. Ela pode ser encontrada na página do *Modellus*, <http://phoenix.sce.fct.unl.pt/modellus>. Procure em *Downloads* pelo arquivo *photos.zip* ou baixe-o diretamente de

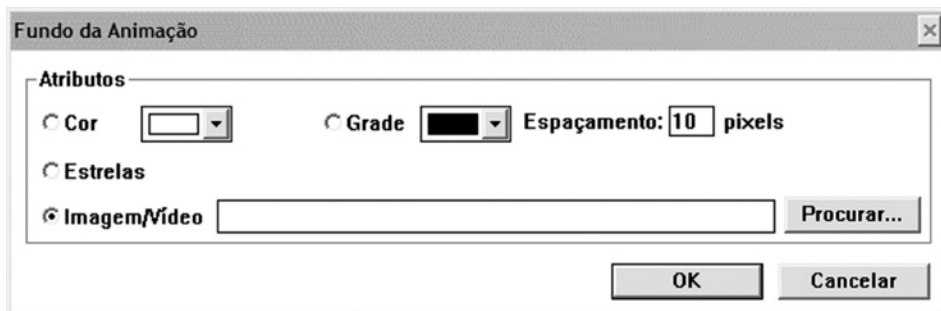
```
http://phoenix.sce.fct.unl.pt/modellus/  
download/modellus2/photos.zip
```

Esse arquivo (note que ele está comprimido) contém uma pasta com a **Figura 8.5** e muitas outras fotos. Após baixar e descomprimir o arquivo, salve a pasta com as fotos em um local fácil de lembrar. A foto com a refração do raio de luz está no arquivo *refract1.gif* dentro dessa pasta. O próximo passo é colocar a foto na janela de animação. A melhor maneira de fazer isso é tornar a foto parte do fundo da janela, usando o botão mostrado na **Figura 8.6**.



**Figura 8.6:** Botão para definir o fundo da janela *Animação*.

Quando você aperta esse botão, a caixa de diálogo mostrada na **Figura 8.7** é criada. Nela você pode modificar o fundo da janela de animação, que originalmente é branco. É possível mudar a cor do fundo ou desenhar uma grade sobre ele ou, ainda, torná-lo um céu estrelado. Também podemos colocar no fundo da janela uma imagem ou vídeo que esteja disponível no computador – basta apertar o botão *Procurar* e localizar o arquivo desejado. Faça isso com a foto em *refract1.gif* e feche a caixa de diálogo clicando em *OK*.



**Figura 8.7:** Propriedades do fundo da janela *Animação*.

Se nada der errado, você verá a foto aparecer no fundo da janela de animação, como mostrado na **Figura 8.8**.



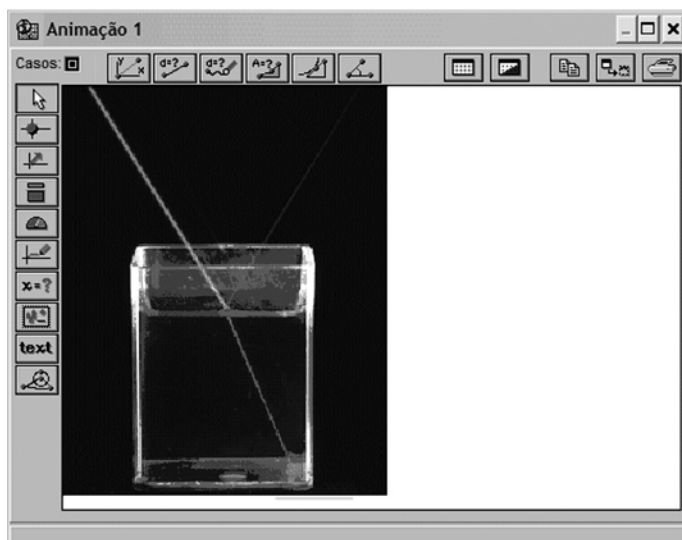


Figura 8.8: Foto colocada no fundo da janela *Animação*.

A próxima etapa é medir os ângulos que os raios incidente e refratado fazem com a superfície da água. Para isso, crie dois “medidores de ângulo” e obtenha os ângulos de incidência e refração, como mostrado na Figura 8.9. Os valores que encontramos são  $58,65^\circ$  e  $67,05^\circ$  para a incidência e a refração, respectivamente.

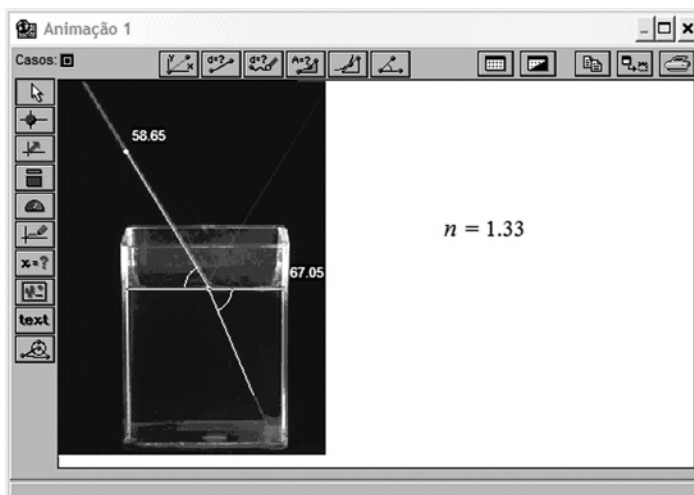


Figura 8.9: Medida dos ângulos de incidência e refração da luz.

O índice de refração da água (em relação ao ar) pode ser calculado a partir da lei de Snell,

$$\text{sen}(\theta_{inc}) = n \text{sen}(\theta_{refr}),$$

em que  $n$  é o índice de refração. Os ângulos de incidência e refração  $\theta_{inc}$  e  $\theta_{refr}$  que aparecem na fórmula de Snell são relativos à *normal* à superfície do líquido. Como medimos os ângulos a partir da superfície, e não de sua normal, temos de subtrair nossas medidas de  $90^\circ$  para obter os valores de  $\theta_{inc}$  e  $\theta_{refr}$ . Conhecidos esses ângulos, fica fácil calcular o índice de refração  $n$ . Tudo isso pode ser feito no próprio *Modellus*, como mostra a Figura 8.10.

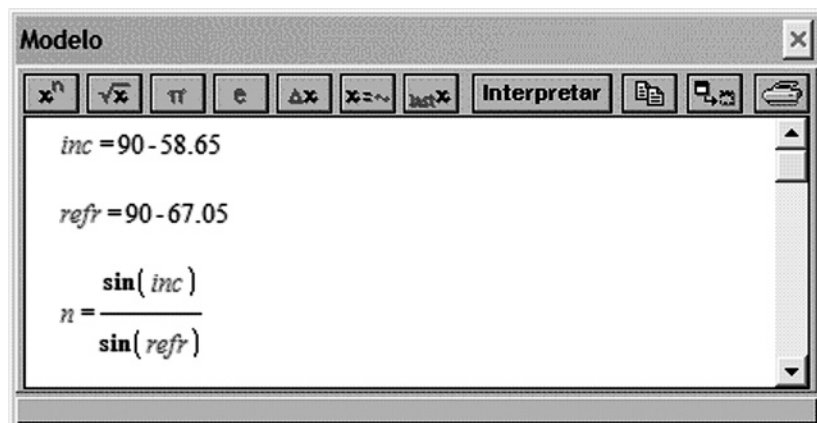
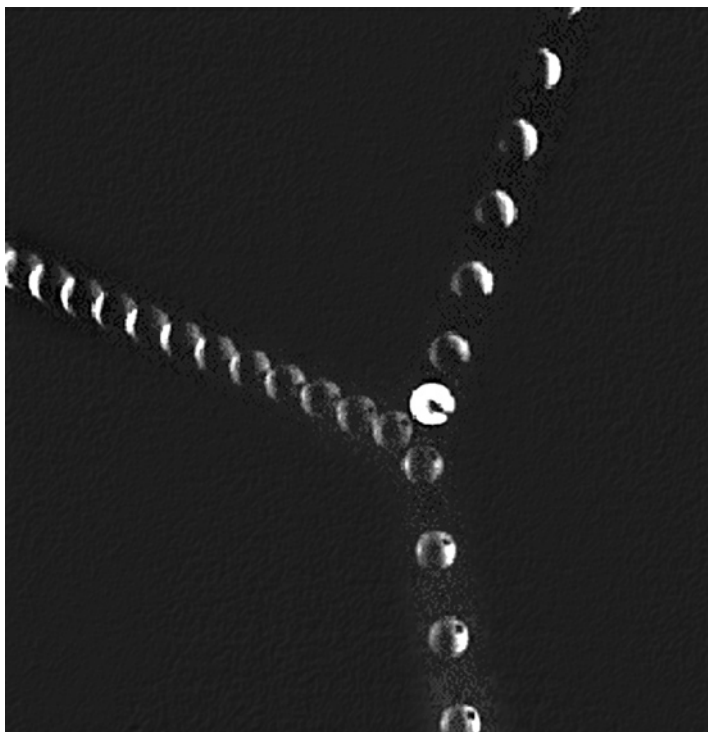


Figura 8.10: Cálculo do índice de refração na janela *Modelo*.

O resultado do cálculo pode ser colocado ao lado da foto, na janela de animação. Para isso, é necessário criar um “medidor digital”, o que é feito com o botão onde se lê  $x = ?$ , situado no lado esquerdo da janela *Animação*. Aperte esse botão e, em seguida, clique em algum lugar da janela; uma caixa de diálogo vai se abrir, na qual você pode escolher que variável será escrita na janela (e outros detalhes, como tipo de letra etc.). Escolha o índice de refração  $n$  e clique no botão *OK*. Se nada deu errado, o valor calculado do índice de refração será escrito na janela *Animação*, como está mostrado na Figura 8.9. O resultado obtido,  $n = 1,33$ , é muito próximo do valor conhecido para esse índice.

## CONSERVAÇÃO DE ENERGIA E MOMENTUM

Outra foto interessante está no arquivo *colis1.gif* (que está na mesma pasta da foto da refração) e está mostrada na **Figura 8.11**.



**Figura 8.11:** Colisão de dois corpos.

A foto mostra a colisão de dois corpos, um originalmente em repouso e outro vindo da parte de baixo da figura. Esta é uma foto *estroboscópica*, o que significa que ela foi tirada com o obturador da câmera aberto enquanto uma luz piscava a intervalos de tempo iguais. Com as ferramentas de medida do *Modellus*, é possível analisar a foto e estudar a conservação da energia e *momentum* no processo de colisão.

Como no caso anterior, comece por colocar a foto da colisão no fundo da janela *Animação*. O passo seguinte é medir as velocidades dos corpos antes e depois da colisão. Para isso, temos de medir a distância que cada um deles percorre em um determinado tempo, ou seja, precisamos usar o “medidor de distância” do *Modellus*. Para começar, crie um medidor e meça a distância entre duas posições do corpo incidente (o que parte da parte de baixo da figura) antes da colisão.

A **Figura 8.12** mostra como isso pode ser feito: existem quatro registros da posição do corpo incidente antes da colisão – do primeiro ao último a partícula deslocou-se aproximadamente 109,3 *pixels*. Para calcular a velocidade, temos de dividir essa distância pelo tempo transcorrido. Vamos usar o intervalo entre duas piscadas da luz estroboscópica como nossa unidade de tempo, e um *pixel* como unidade de distância. Nessas unidades, a velocidade do objeto incidente (vamos chamá-lo de corpo 1) é

$$u_1 = \frac{109,3}{3} = 36,4 .$$

Note que a divisão por 3 vem do fato de que existem três intervalos de tempo separando os quatro registros da posição do corpo 1. O segundo objeto está em repouso inicialmente (ele aparece brilhante no centro da foto por isso), e portanto a sua velocidade é  $u_2 = 0$ .

As velocidades finais dos corpos 1 e 2 são obtidas de maneira semelhante. Crie mais dois medidores de distância e use-os para medir os deslocamentos após a colisão. A **Figura 8.12** mostra um exemplo disso – note que não é necessário usar o mesmo intervalo de tempo em diferentes medidas. Com os dados da **Figura 8.12**, as velocidades finais são

$$v_1 = \frac{139,7}{9} = 15,5$$

$$v_2 = \frac{123,7}{4} = 30,9 .$$

Note que usamos dez registros para medir o deslocamento da partícula 1 e cinco para a partícula 2 – daí vêm as divisões por 9 e 4, respectivamente, no cálculo das velocidades.

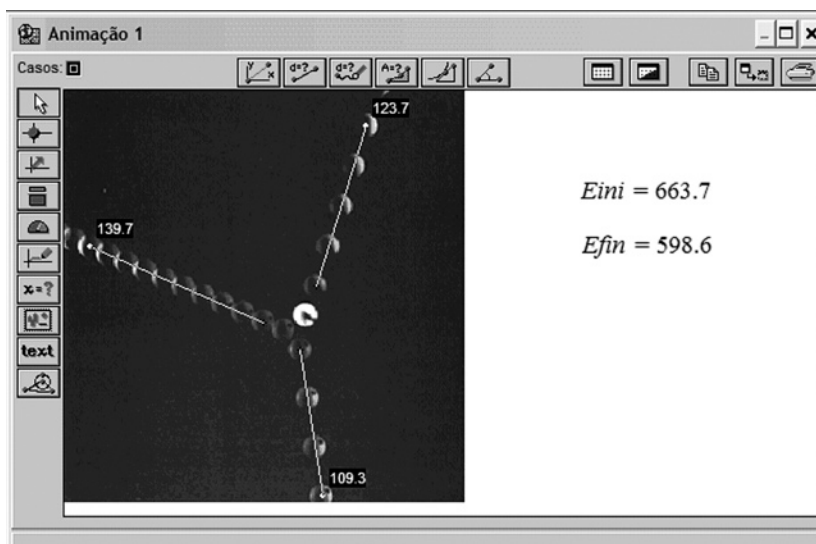


Figura 8.12: Medidas sobre a foto de colisão.

Com essas medidas, podemos calcular as energias cinéticas inicial e final e verificar se a colisão foi elástica ou inelástica. Nós não conhecemos as massas dos corpos (sabemos apenas, pela foto, que parecem iguais), portanto vamos simplificar as coisas e dizer que elas valem  $m = 1$ . Em outras palavras, adotaremos a massa das partículas em colisão como a nossa unidade de massa (nada demais para quem está medindo distância em *pixels* e tempo em piscada de estroboscópio). Com isso, as energias cinéticas inicial e final são:

$$E_{ini} = \frac{1}{2}u_1^2 + \frac{1}{2}u_2^2 = 664$$

$$E_{fin} = \frac{1}{2}v_1^2 + \frac{1}{2}v_2^2 = 599.$$

Esses resultados mostram que a colisão foi bastante inelástica; a perda de energia foi muito grande (cerca de 10%) para ser explicada apenas por incertezas nas medidas. Como já fizemos anteriormente, podemos realizar todos esses cálculos no próprio *Modellus* e escrever o resultado ao lado da foto analisada. A Figura 8.13 mostra como ficam as contas na janela *Modelo*; faça esses cálculos na sua simulação e escreva o resultado na janela de animação (veja a Figura 8.12).

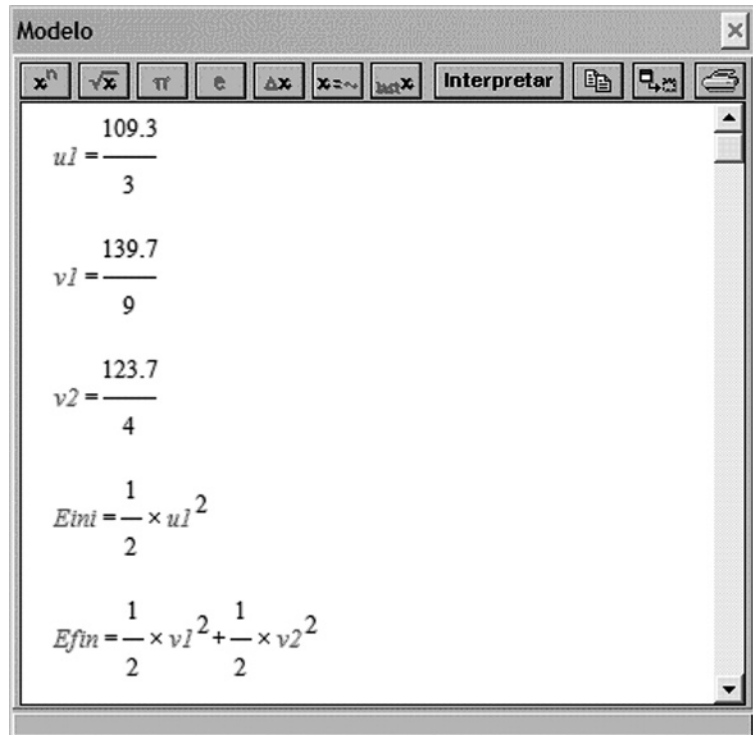


Figura 8.13: Cálculo das velocidades e energias na janela *Modelo*.

Da mesma forma como estudamos a (não) conservação da energia cinética, podemos investigar a conservação do momento linear. Ao contrário da energia cinética, que pode ou não se manter constante, o momento linear é sempre conservado numa colisão. Para verificar isso, precisamos medir não apenas o módulo das velocidades, mas também suas direções, já que o momento linear é um vetor. O ângulo que cada velocidade final faz com a direção da partícula incidente pode ser medido da maneira mostrada na **Figura 8.14**. (Note que apagamos as medidas de distância para não sobrecarregar a figura.)

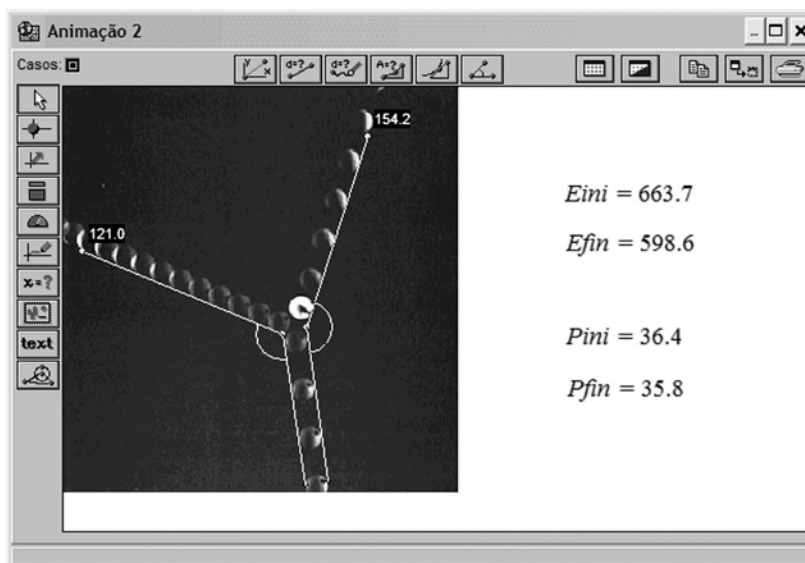


Figura 8.14: Medida dos ângulos entre as velocidades finais e a direção de incidência.

Vamos usar essas medidas para analisar a conservação do *momentum* na direção paralela à trajetória da partícula incidente. O *momentum* inicial nessa direção é  $P_{ini} = mu_1$ , já que a partícula 2 está em repouso. Como nas nossas unidades a massa é  $m = 1$ , o *momentum* inicial é  $P_{ini} = u_1$ . O *momentum* final é  $P_{fin} = v_1 \cos(\theta_1) + v_2 \cos(\theta_2)$ , onde  $\theta_1$  e  $\theta_2$  são o ângulos que as velocidades finais fazem com a direção de incidência. Estes não são exatamente os ângulos medidos na Figura 8.14, mas seus *ângulos suplementares* (ou seja,  $\theta$  é igual a  $180^\circ$  menos o ângulo medido).

Todos esses cálculos podem ser colocados na janela *Modelo*, em seguida à determinação das energias (veja a Figura 8.15). Também como antes, podemos escrever os resultados na janela *Animação*. A Figura 8.14 mostra os valores encontrados para  $P_{ini}$  e  $P_{fin}$ . Embora os momentos lineares inicial e final não sejam exatamente iguais, eles estão muito próximos: a diferença é menor que 2%. Essa discrepância pode ser perfeitamente explicada pelas incertezas do processo de medida, de modo que é seguro dizer que o momento linear se conserva nessa colisão.

Na verdade, só verificamos a conservação do *momentum* na direção paralela ao movimento da partícula incidente. Na direção perpendicular, o *momentum* inicial é zero e o *momentum* final é  $v_1 \sin(\theta_1) - v_2 \sin(\theta_2)$ . Portanto, o *momentum* na direção perpendicular à de incidência será conservado se  $v_1 \sin(\theta_1) = v_2 \sin(\theta_2)$ . Modifique os cálculos que estão

na janela *Modelo* para verificar se isso ocorre e escreva seus resultados na janela de animação.

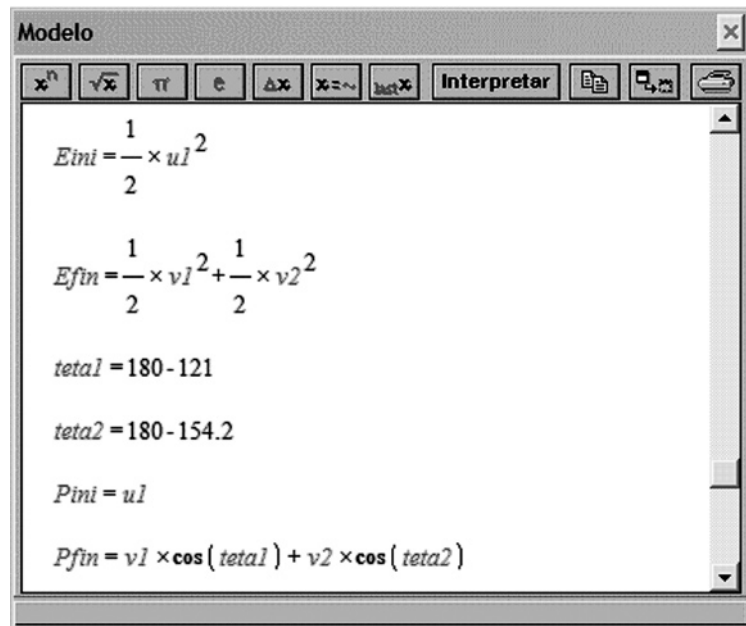


Figura 8.15: Cálculos da conservação de *momentum*.

## MOVIMENTO DE PROJÉTEIS

Como última aplicação das ferramentas de medida do *Modellus*, vamos investigar o movimento da bola mostrada na **Figura 8.16**. A foto está no arquivo *ball1.gif*, na mesma pasta das fotos que já analisamos.

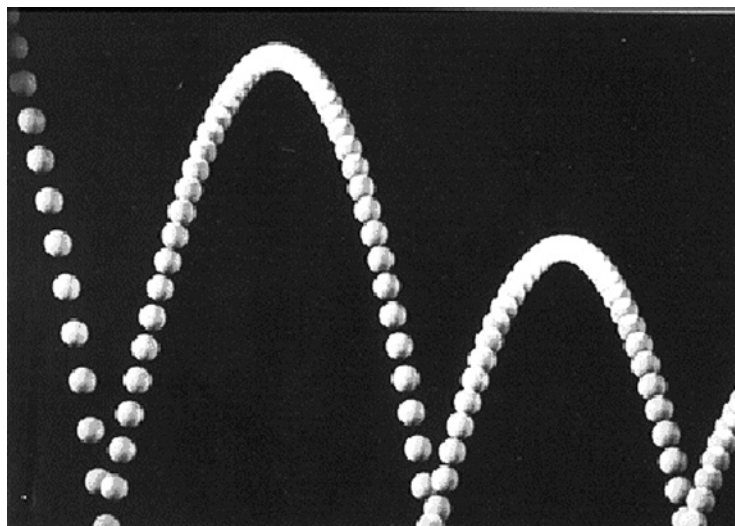


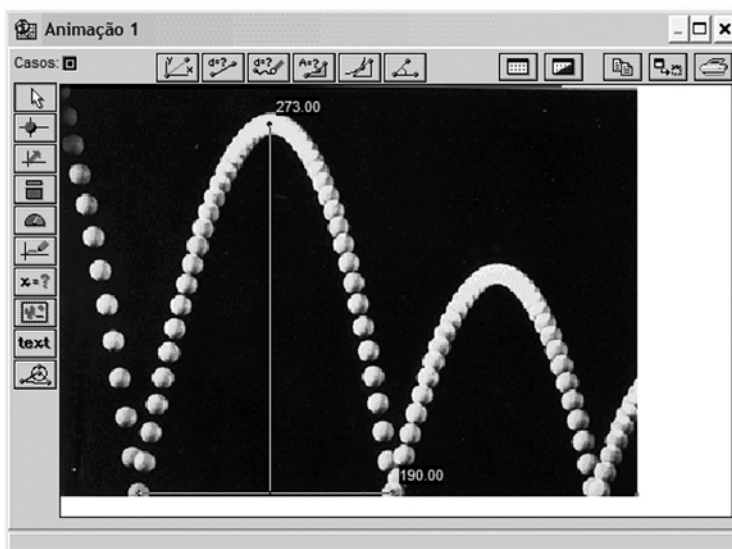
Figura 8.16: Movimento de uma bola quicando no chão.



Se a resistência do ar é desprezível, a trajetória deve ser uma parábola entre dois quiques da bola. Se conhecemos a altura  $h$  alcançada pela bola e a distância  $a$  entre dois quiques sucessivos, a forma dessa parábola é dada por

$$y = \frac{4h}{a^2} x(a-x).$$

Podemos verificar se o movimento mostrado na **Figura 8.16** é realmente dado por essa fórmula, medindo os valores de  $h$  e  $a$  e, em seguida, traçando a parábola correspondente sobre a foto. As medidas podem ser vistas na **Figura 8.17**, que você pode reproduzir utilizando o que já aprendeu nas seções anteriores.



**Figura 8.17:** Medida da altura máxima e alcance.

Com os resultados obtidos (no caso da **Figura 8.17**,  $h = 273$  e  $a = 190$ ), podemos definir a parábola que descreve o primeiro salto da bola. A **Figura 8.18** mostra como isso é feito na janela *Modelo*.

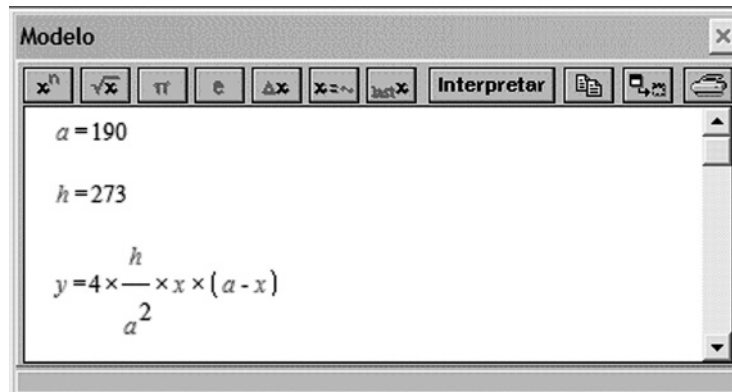


Figura 8.18: Modelo de trajetória parabólica.

Com o modelo escrito e interpretado, vamos agora desenhar a parábola sobre a foto e verificar se a trajetória realmente tem essa forma. Para isso, temos de criar um gráfico de  $y(x)$  na janela de animação, o que é feito com o botão contendo um sistema de eixos e um lápis, encontrado à esquerda da janela. Aperte o botão e, em seguida, clique sobre o local onde ocorreu o primeiro quique. Uma caixa de diálogo vai se abrir, com as propriedades do gráfico a ser criado. Como está mostrado na Figura 8.19, coloque as variáveis  $x$  e  $y$  na horizontal e na vertical, desative as opções de desenhar eixos, lápis etc. (que só irão sobrecarregar a figura) e, finalmente, feche a caixa clicando em *OK*.

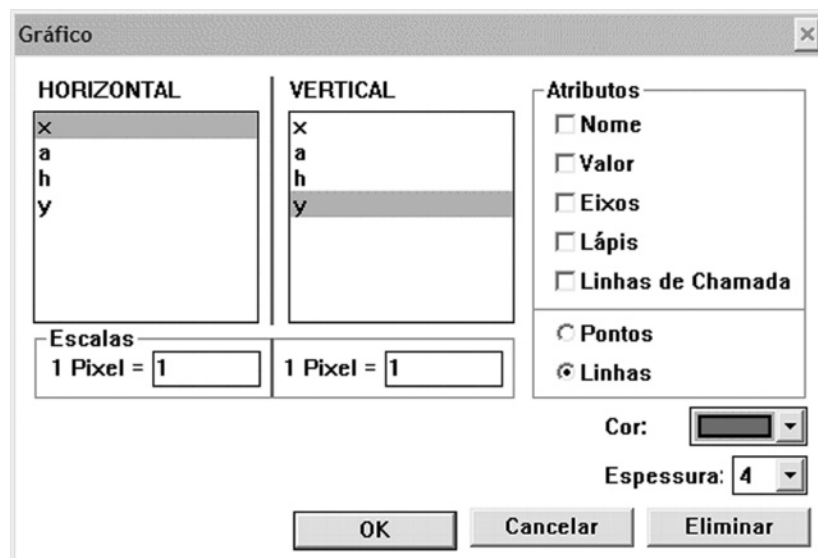
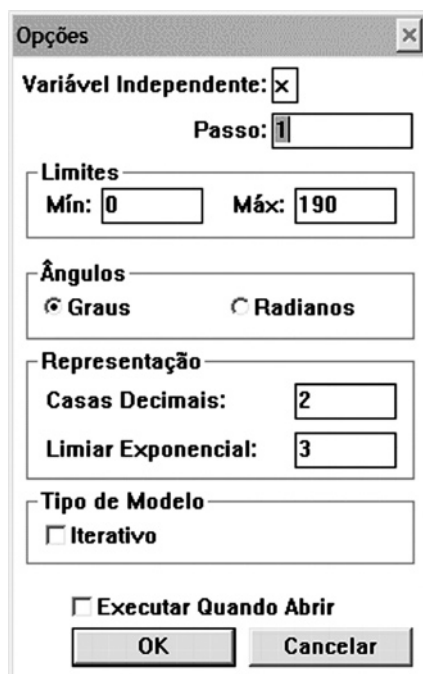


Figura 8.19: Propriedades do objeto *Gráfico*.

Para executar a simulação e traçar o gráfico, é necessário fazer mais uma coisa: mudar a variável independente do modelo. Lembre-se de que essa variável é, por *default*, chamada de  $t$ . Mas, no modelo de parábola que acabamos de construir, a variável independente é  $x$ , que varia entre  $x = 0$  e  $x = a = 190$ . Tudo isso pode ser mudado com o botão *Opções* da janela *Controlo*, tal como está mostrado na **Figura 8.20**.



**Figura 8.20:** Redefinição da variável independente.

Feito tudo isso, execute a simulação da trajetória a partir da janela *Controlo*. Se nada deu errado, você verá algo parecido com o que está na **Figura 8.21**. Podemos observar que a trajetória da bola é bastante bem descrita pela parábola, embora a concordância não seja perfeita.

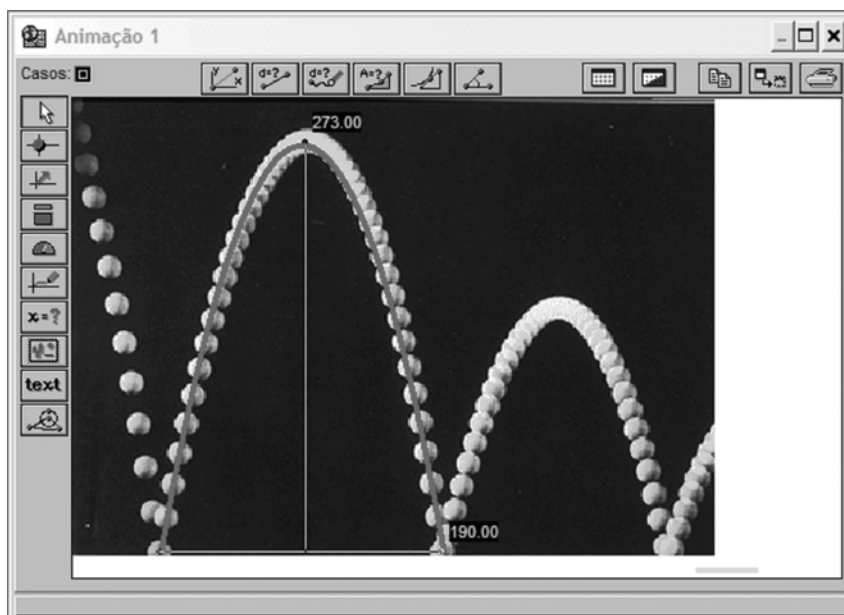


Figura 8.21: Comparação entre a trajetória parabólica e o movimento da bola.

## INFORMAÇÃO SOBRE A PRÓXIMA AULA

Na próxima aula, veremos como produzir simulações com as quais podemos interagir, modificando suas propriedades enquanto elas são executadas.

## Controlando simulações no *Modellus*

### Meta da aula

Apresentar os métodos de controle de simulações disponíveis no *Modellus*.

# objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- desenvolver no *Modellus* simulações que podem ser controladas interativamente;
- utilizar instruções de controle *if-then* (*se-então*) em simulações no *Modellus*.

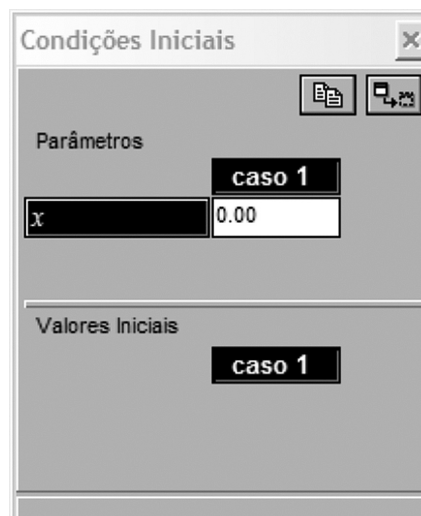
## SIMULAÇÕES INTERATIVAS NO *MODELLUS*

*Videogames* estão entre as aplicações mais populares dos computadores, para o bem ou para o mal das novas gerações. Uma característica interessante do *Modellus* é que podemos usá-lo para criar ambientes semelhantes a esses jogos, ou seja, simulações com as quais podemos interagir enquanto elas estão sendo executadas. Isso abre novas possibilidades para o uso do *Modellus*, que exploraremos nesta aula. Vamos começar criando uma simulação interativa bem simples: na janela *Modelo*, escreva  $x$  (isso mesmo, apenas a letra “ $x$ ”), como está mostrado na **Figura 9.1**.



**Figura 9.1:** Modelo com parâmetro ( $x$ ) a ser controlado interativamente.

Agora interprete o modelo. Você verá que a janela *Condição Inicial* é criada, e  $x$  aparece como um parâmetro a ser especificado (veja a **Figura 9.2**). O valor  $x = 0$  é colocado por *default* na janela e pode ser alterado, mas não há problema em deixá-lo assim. Como veremos, esse será apenas o valor inicial de  $x$ , que você aprenderá a mudar de forma interativa.

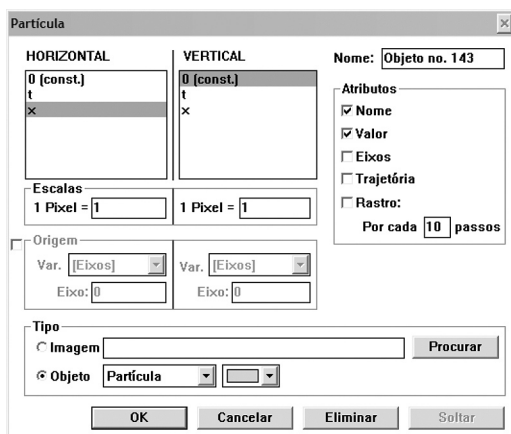


**Figura 9.2:** Janela com o valor inicial do parâmetro  $x$ .

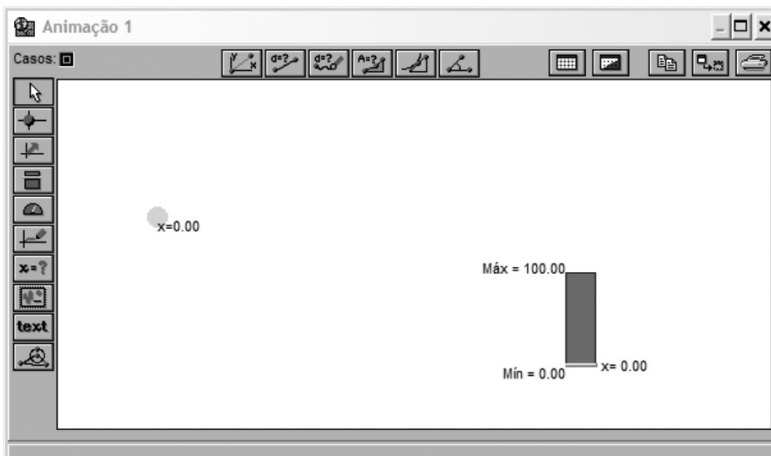
O próximo passo é colocar na janela de animação um objeto com o qual é possível interagir e, por seu intermédio, controlar o valor da variável  $x$ . Para isso, crie um objeto *Barra* com valor dado por  $x$ , tal como está mostrado na **Figura 9.3**. Note que o valor máximo de  $x$  foi fixado em 100. Em seguida, crie um objeto *Partícula* na mesma janela e defina sua posição horizontal como sendo  $x$  (veja a **Figura 9.4**). Com esses dois objetos criados, a janela *Animação* deve ficar parecida com o que está na **Figura 9.5**.



**Figura 9.3:** Propriedades do objeto *Barra*.



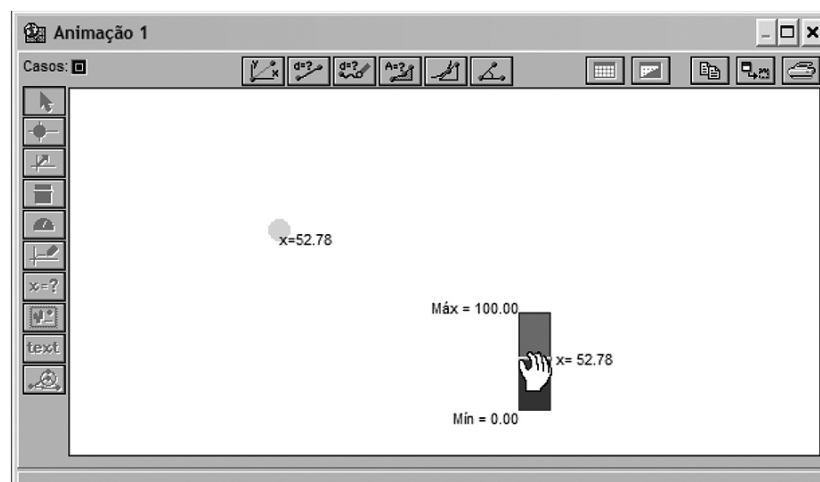
**Figura 9.4:** Propriedades do objeto *Partícula*.



**Figura 9.5:** A janela de animação com a barra e a partícula.

Para que a simulação não termine muito rapidamente, aumente o limite máximo da variável independente  $t$  para um valor bem grande, como 1.000 (lembre que isso é feito com o botão *Opções* da janela de controle). Isso nos dará tempo suficiente para explorar os controles introduzidos. Finalmente, execute a simulação. Como você já deve ter imaginado, nada acontece, já que o modelo interpretado não determina como  $x$  depende de  $t$ . Entretanto, há uma forma *interativa* de mudar o valor de  $x$ . Enquanto a simulação estiver rodando, coloque o cursor sobre a barra que foi criada na janela de animação – você verá que o cursor muda de aspecto, ganhando uma forma de “mão”, como mostrado na **Figura 9.6**. Use esse “cursor/mão” para levantar e baixar a barra, e note como isso muda o valor de  $x$  (lembre que o nível da barra está associado a  $x$ , veja a **Figura 9.3**).

O mais interessante é que a posição da partícula também muda quando movemos a barra. É fácil entender isso: basta lembrar que o valor da variável  $x$  também determina o deslocamento horizontal da partícula (veja a **Figura 9.4**). Observe que esta simulação é bem diferente de todas as que já criamos com o *Modellus* – em vez de uma definição matemática de como a posição  $x$  da partícula depende do tempo  $t$ , agora controlamos à vontade a posição da partícula, usando para isso o nível da barra. É importante ressaltar que essa interação só pode ocorrer enquanto a simulação estiver sendo executada – foi por isso que sugerimos aumentar muito o valor máximo de  $t$ . Se o tempo acabar ou a simulação for interrompida, não será mais possível controlar interativamente a variável  $x$ .

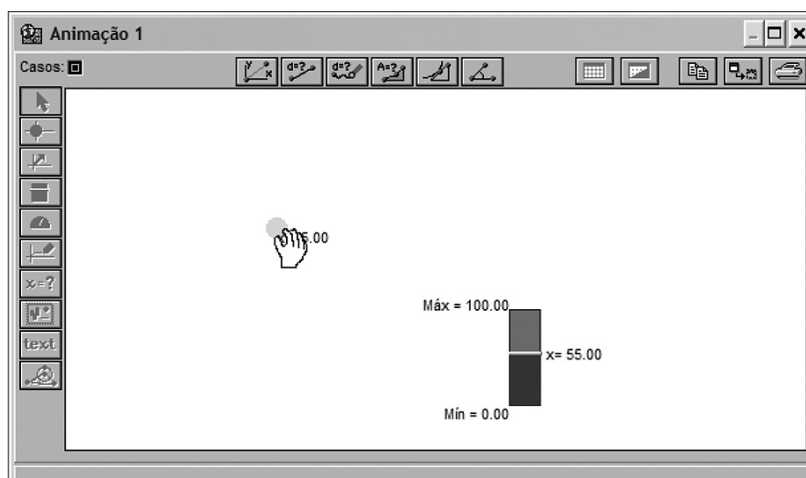


**Figura 9.6:** Controle da posição da partícula com o nível da barra.



Também é possível usar a partícula para controlar o nível da barra. Ao colocar o cursor sobre a partícula, enquanto a simulação estiver rodando, você verá novamente a “mão” aparecer, tal como está na **Figura 9.6**. Como no caso da barra, podemos então deslocar a partícula, mudando o valor de  $x$ . Com isso, o indicador da barra também muda, acompanhando o movimento que damos à partícula.

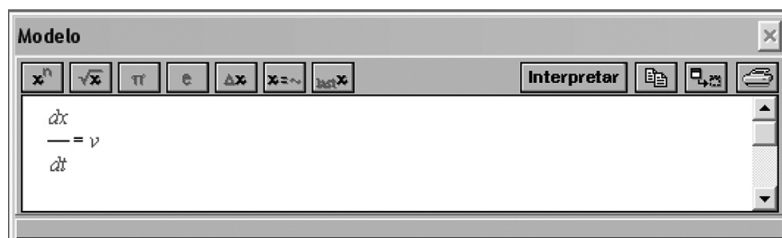
Você pode interromper a simulação a qualquer momento, usando o botão apropriado na janela de controle. A interrupção “desliga” a interatividade, que só retorna quando a execução do programa é retomada.



**Figura 9.7:** Controle da barra com a partícula.

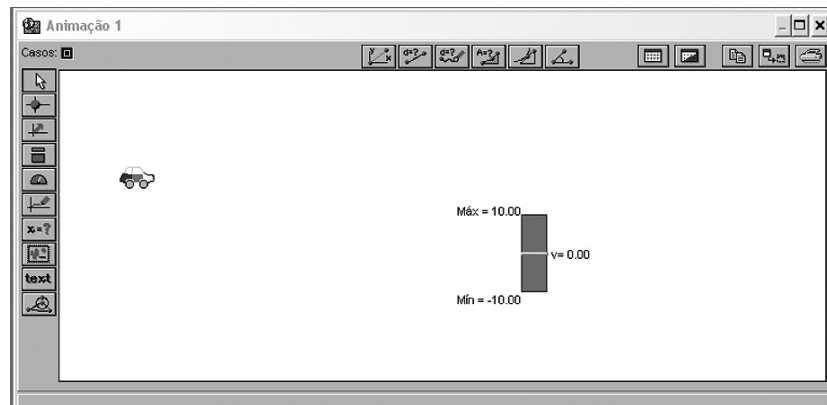
## CONTROLANDO A VELOCIDADE DE UM CARRO

Vamos agora fazer um programa interativo mais interessante. Em vez de mudar a posição da partícula, vamos controlar a sua velocidade. Para isso, comece um novo programa e escreva na janela *Modelo* a equação  $dx/dt = v$ , tal como está na **Figura 9.8**. Essa é uma equação diferencial como as que já estudou em outras aulas – sabemos que o *Modellus* vai resolvê-la e calcular a posição  $x(t)$  da partícula. Mas, ao contrário dos casos anteriores, a velocidade  $v$  não será dada por uma constante, ou uma função, ou ainda por outra equação diferencial – ela será definida interativamente à medida que o programa for sendo executado.



**Figura 9.8:** Modelo para controle da velocidade.

Quando o modelo é interpretado, a janela *Condições Iniciais* é criada pedindo o valor da velocidade  $v$  e a condição inicial para  $x$ . Novamente, não há problema em deixar inalterados os valores definidos pelo próprio *Modellus*. O próximo passo é produzir um controle da velocidade  $v$ . Para isso, vamos usar, de novo, um objeto *Barra* da janela *Animação*. Os passos para a criação de uma barra associada à variável  $v$  já são bem familiares, e não vamos repeti-los. O resultado está mostrado na **Figura 9.9**. Note que escolhemos os limites máximo e mínimo dos valores de  $v$  definidos na barra como sendo 10 e -10, respectivamente. A **Figura 9.9** também mostra a partícula, cuja posição é dada pela variável  $x$ . Em vez de representar a partícula por uma bolinha, como no exemplo anterior, desta vez usamos um carro. A figura do carrinho está no arquivo *Car.bmp* que se encontra entre as imagens fornecidas pelo *Modellus*, já utilizadas na Aula 7 (se você não lembra onde salvou as imagens, não se preocupe – pode continuar a usar uma bolinha para representar a partícula).



**Figura 9.9:** Partícula (carro) com velocidade controlada pela barra.

Os próximos passos já são conhecidos: aumente para um valor bem grande o limite máximo do tempo  $t$  (usando as opções da janela de controle), interprete o modelo e execute-o. Durante a execução, use o *mouse* para levantar e abaixar a barra, alterando assim a velocidade do carrinho. Se tudo deu certo, você será capaz de fazer o carro andar para frente e para trás, controlando seu movimento à vontade. Os gráficos da **Figura 9.10** mostram a velocidade e a posição do carrinho após uma simulação interativa típica. É instrutivo discutir a relação entre os gráficos da velocidade e do deslocamento durante esse tipo de simulação.

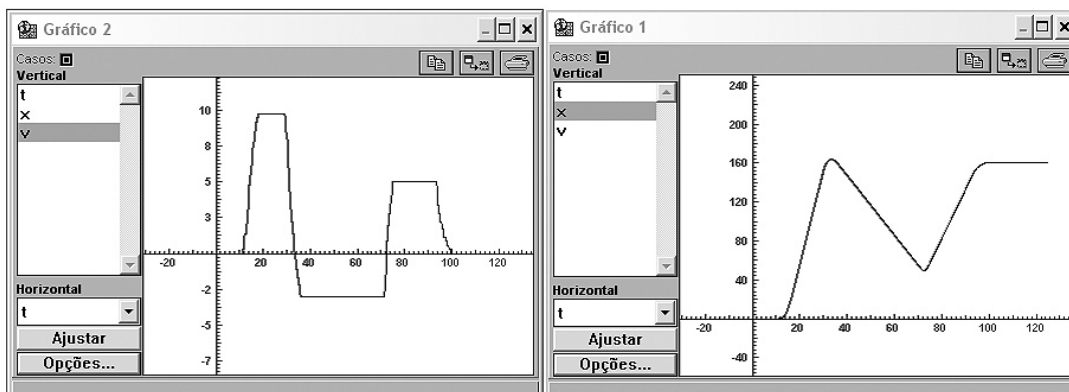


Figura 9.10: Gráficos da velocidade (à esquerda) e posição (à direita) do carro.

## CONTROLANDO A ACELERAÇÃO DO CARRO

É fácil fazer um programa em que a aceleração do carro é controlada, em vez da velocidade. Basta escrever, na janela *Modelo*,  $dx/dt = v$  e  $dv/dt = a$ , como mostrado na Figura 9.11. Note que apenas adicionamos uma equação ao modelo anterior. Esta segunda equação define a velocidade, que deixa de ser um parâmetro livre e não pode mais ser controlada interativamente. O que está livre para ser alterado agora é a aceleração  $a$ . Podemos usar a barra que já está na janela de animações para controlar a aceleração; basta mudar suas propriedades e associá-la ao valor de  $a$ , como se vê na Figura 9.12. Observe que colocamos os valores máximo e mínimo da barra em  $a = 2$  e  $a = -2$ .



Figura 9.11: Modelo para controle da aceleração.

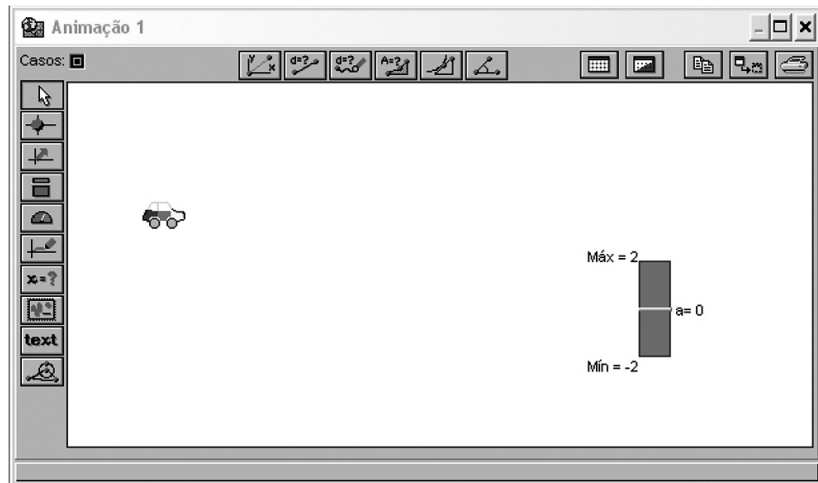


Figura 9.12: Carro com controle de aceleração.

Gráficos da aceleração, velocidade e posição do carro obtidos numa simulação em que a aceleração foi controlada interativamente estão mostrados na Figura 9.13.

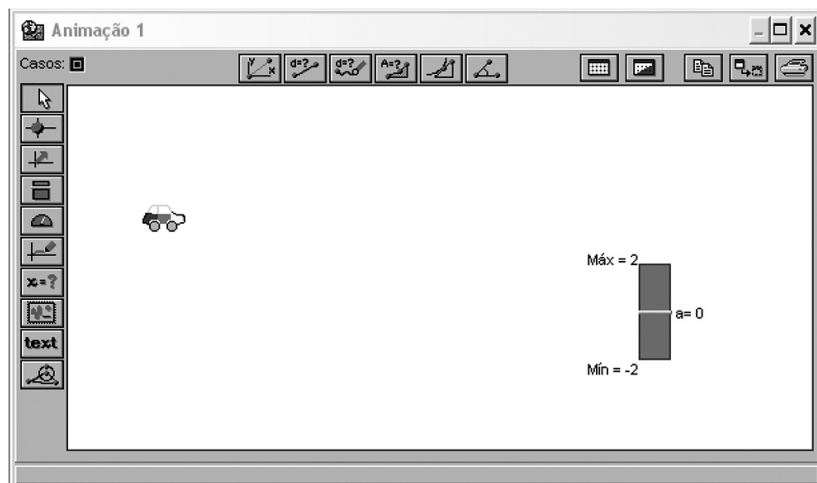


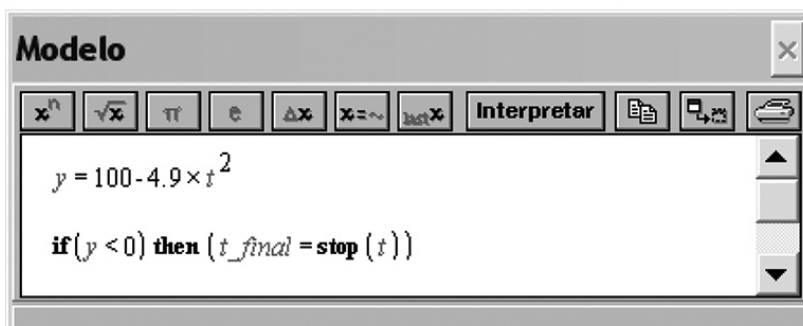
Figura 9.13: Aceleração, velocidade e posição do carro.

## INSTRUÇÕES DE CONTROLE: O COMANDO *IF-THEN* (*SE-ENTÃO*)

Nas simulações interativas, o usuário do *Modellus* resolve que rumos o programa toma durante a execução. Existe uma forma de dar poder de “decisão” semelhante ao próprio programa. Por exemplo, podemos fazer com que uma simulação resolva sozinha a hora em que deve parar. Para ver um exemplo concreto disso, você vai estudar um corpo que cai de uma altura de 100 metros. Desprezando a resistência do ar e considerando  $g = 9,8 \text{ m/s}^2$ , o movimento do corpo será dado por  $y = 100 - 4,9t^2$ , onde a altura  $y$  e o tempo  $t$  estão em metros e segundos, respectivamente. Essa fórmula só vale enquanto o corpo está acima do solo, e por isso gostaríamos de terminar a simulação assim que a altura  $y$  se torne negativa. O modelo mostrado na **Figura 9.14** mostra como fazer tal coisa. A primeira linha define a queda livre do corpo, e a segunda determina que a simulação seja interrompida no instante em que a condição  $y < 0$  for satisfeita. O comando que implementa essa decisão tem uma estrutura *se-então*: se a condição A ocorrer, *então* realize a ação B. No *Modellus*, o “se” e o “então” são escritos em inglês, como *if* e *then*. Assim, escrevemos “*if* (A) *then* (B)”, em vez de “se (A) então (B)”. No modelo mostrado na **Figura 9.14**, temos a instrução

$$\text{if } (y < 0) \text{ then } (t\_final = \text{stop } (t)),$$

ou seja, a condição A é  $y < 0$ , e a ação B é a interrupção do programa. Esta última é feita com a instrução “ $t\_final = \text{stop}(t)$ ”, que faz duas coisas: termina a execução do programa (para isso serve o *stop*, que significa “pare” em inglês) e atribui à variável  $t\_final$  o valor de  $t$  em que ocorre a interrupção.



**Figura 9.14:** Simulação de queda livre. A execução do programa termina quando a altura torna-se negativa.

Interpretando e executando o modelo acima, encontramos o resultado mostrado no gráfico da **Figura 9.15**. A trajetória da partícula é traçada apenas até o instante em que ela cruza o eixo horizontal, ou seja, até que  $y$  fique negativo. Nesse instante (algo entre 4 e 5 segundos), a execução do programa termina, embora o tempo máximo fixado na janela de controle seja 20 segundos. Se quiser saber exatamente o ponto em que o programa parou, você pode escrever o valor da variável  $t\_final$  na janela de animações. A **Figura 9.16** mostra como isso é feito usando um medidor digital (criado com o botão que tem um “ $x = ?$ ” desenhado). Ao final da simulação, o medidor mostra que a execução foi interrompida em  $t\_final = 4,6$  segundos.

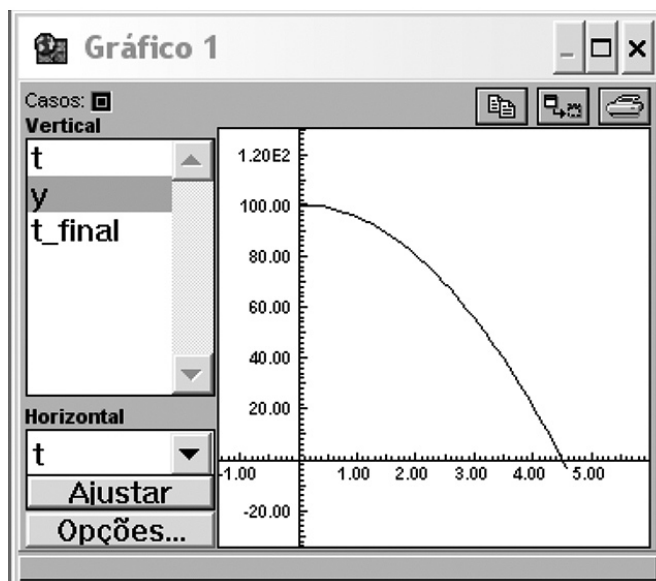


Figura 9.15: Queda livre interrompida quando  $y < 0$ .

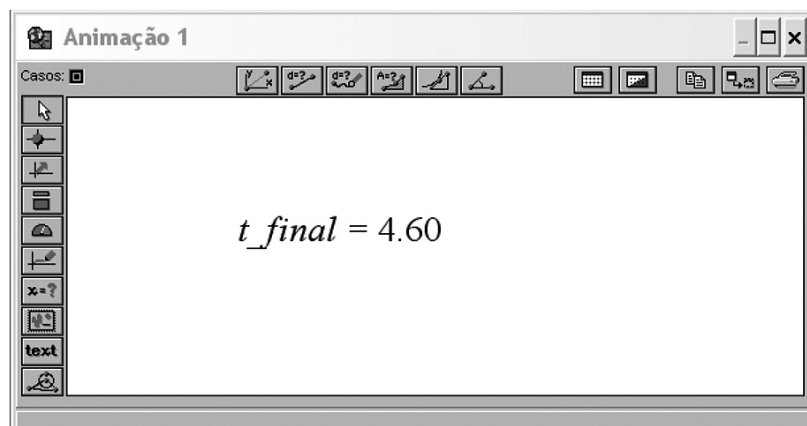


Figura 9.16: O tempo final escrito no medidor digital.

É importante notar que o corpo atinge o solo em  $t = \sqrt{100/4,9} = 4,52$  segundos, ou seja,  $t_{final}$  não é exatamente o tempo de queda. O programa pára no primeiro passo em que  $y < 0$ , não quando  $y = 0$ . É fácil ver que a diferença entre o tempo de queda e  $t_{final}$  é menor que a duração de um passo da simulação. Portanto, para fazer com que  $t_{final}$  represente melhor o tempo de queda, basta diminuir o tamanho do passo (janela de controle, botão Opções). Podemos tentar parar o programa quando  $y = 0$ , usando “*if* ( $y = 0$ ) *then* ( $t_{final} = stop(t)$ )”. Entretanto, isso dificilmente funciona, pois é muitíssimo improvável que um dos passos do programa caia exatamente em  $y = 0$  (com todas as casas decimais necessárias). Repare também na forma como a condição  $y = 0$  é escrita no comando *if-then*:  $y = = 0$ , com duplo “=”. Para ver mais sobre a sintaxe do comando *if-then*, procure o programa de ajuda do *Modellus*.

## SALTO DE PÁRA-QUEDAS

O comando *if-then* não serve apenas para terminar a simulação: ele pode ser usado para atribuir valores a parâmetros. Vamos fazer isso estudando o movimento de um pára-quedista. A resistência do ar sentida pelo pára-quedista é, em boa aproximação, proporcional ao quadrado da velocidade de queda. A força total sobre o pára-quedista é então

$$F = mg - kv^2,$$

onde  $mg$  é o peso do pára-quedista,  $v$  é a sua velocidade e  $k$  é uma constante que caracteriza a importância da resistência do ar. Antes de o pára-quedas abrir-se, o valor de  $k$  é pequeno, correspondendo apenas à resistência encontrada pelo corpo do pára-quedista. Um valor típico nesse caso é  $k = 0,5$  kg/m. Quando o pára-quedas é aberto, a resistência do ar aumenta muito, e o valor de  $k$  sobe para algo como  $k = 40$  kg/m. Vamos supor que o pára-quedista abra o pára-quedas após um tempo  $t_0$  de queda livre. Um modelo para o cálculo da velocidade desse pára-quedista está mostrado na **Figura 9.17**. Note como a abertura do pára-quedas foi modelada com as duas instruções de controle *if-then*.

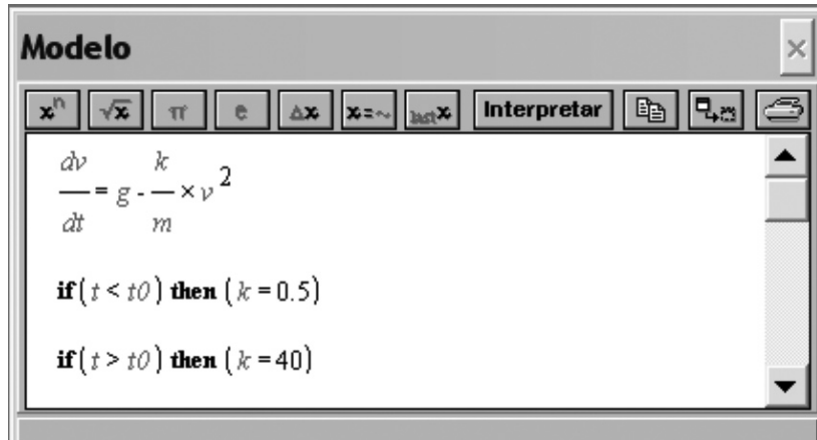


Figura 9.17: Modelo de um pára-quedista que abre o pára-quedas após um tempo  $t_0$  em queda livre.

A velocidade do pára-quedista está mostrada na Figura 9.18. O tempo de queda livre foi fixado em  $t_0 = 6$  s. A massa do pára-quedista é  $m = 80$  kg e a aceleração da gravidade é  $g = 9,8$  m/s<sup>2</sup>. No cálculo mostrado na Figura 9.18, o tempo  $t$  varia em passos de 0.01 s (lembre que isso é fixado nas *Opções* da janela *Controle*). Passos muito maiores, como o fixado inicialmente pelo *Modellus* (0.1 s), não conseguem acompanhar a rápida desaceleração causada pelo pára-quedas e levam a resultados pouco precisos.

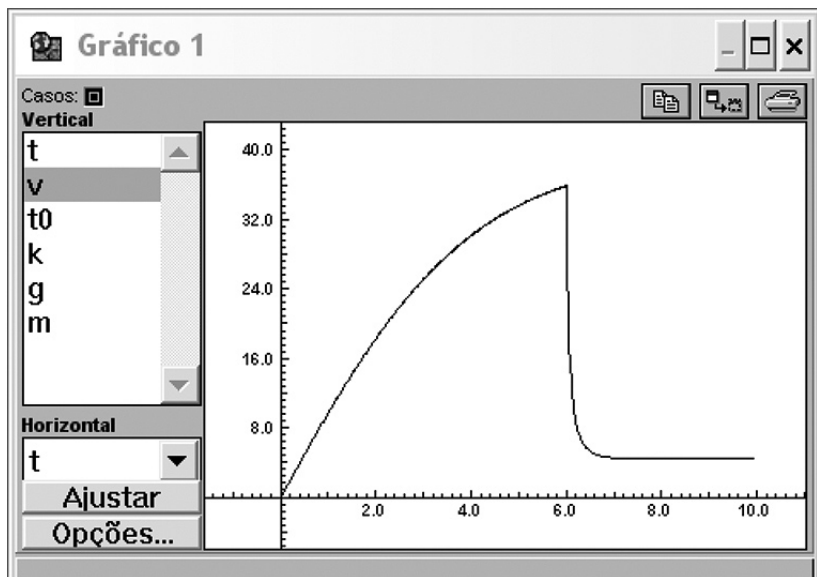


Figura 9.18: Velocidade do pára-quedista antes e depois da abertura do pára-quedas.



Existem muitas outras maneiras de se usar o comando *if-then*. É possível combinar duas condições, usando as operações “e” e “ou”, que são escritas em inglês no *Modellus*, como *and* e *or*. Também é possível usar o *and* para executar mais de uma ação. O programa de ajuda mostra como isso tudo pode ser feito.

### INFORMAÇÃO SOBRE A PRÓXIMA AULA

Esta foi a última aula sobre o *Modellus*. Na próxima aula, iniciaremos nosso estudo da linguagem de programação Logo.

# Introdução ao Logo

# AULA 10

## Meta da aula

Apresentar os principais aspectos da linguagem de programação Logo.

## objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- utilizar o ambiente de programação do *SuperLogo*;
- compreender alguns comandos básicos da linguagem Logo e usá-los em modo interativo.

*A Rainha abandonou então a partida, quase sem fôlego, e perguntou a Alice:*

*– Você já viu a Falsa Tartaruga?*

*– Não – respondeu Alice. – Nem sei o que é uma Falsa Tartaruga.*

*– É aquilo de que se faz a falsa sopa de tartaruga – informou a Rainha.*

*– Nunca vi, e nunca ouvi falar – confessou Alice.*

*– Venha então – disse a Rainha. – Ela lhe contará sua história.*

L. Carrol, *Alice no País das Maravilhas*

## LOGO E A TARTARUGA

A linguagem de programação Logo foi projetada para ser um instrumento de aprendizagem –, algo que estudantes usam para aprender outras coisas. Ela é uma linguagem poderosa (é um dialeto do LISP) e fácil de usar, criada nos anos 60 por Wallace Feurzeig, Daniel Bobrow e Seymour Papert. Este último, um matemático que colaborara com Piaget, tornou-se o principal inspirador do uso do Logo como meio de aprendizagem. Em suas primeiras versões, o Logo podia controlar os movimentos de um pequeno robô, chamado “tartaruga” devido à sua forma, e que acabou tornando-se a marca registrada da linguagem. Com o desenvolvimento de terminais gráficos de baixo custo, a tartaruga mudou-se para a tela do computador, onde pode mover-se de forma mais rápida e precisa.

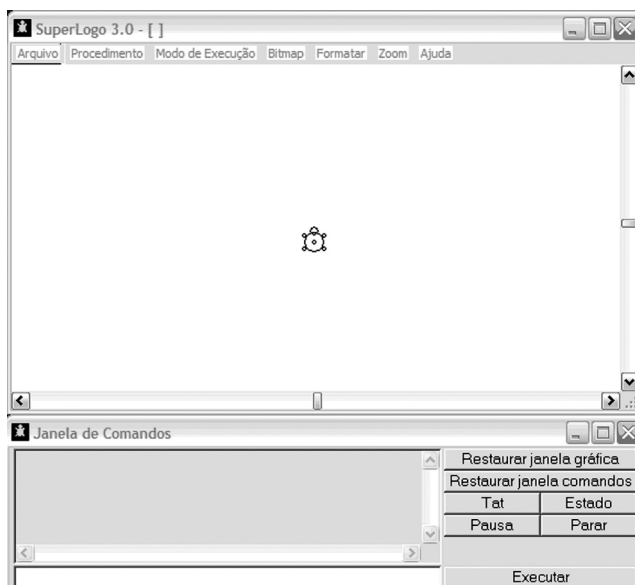
## O SUPERLOGO

Existem muitas implementações da linguagem Logo, em diferentes idiomas e para diversos sistemas operacionais. A versão que usaremos é o *SuperLogo*, um Logo para Windows que foi adaptado para o português pelo Núcleo de Informática Aplicada à Educação, da Unicamp. O *SuperLogo* é distribuído gratuitamente pela internet e pode ser encontrado em

<http://www.nied.unicamp.br/publicacoes/software/slogo30.zip>

O programa não exige máquinas de grande performance nem versões muito atuais do Windows (basicamente, Win98 ou posterior). A instalação é simples: basta descompactar o arquivo *slogo30.zip* obtido no endereço citado e executar o programa *setup.exe*. Se tudo correr

bem, o *SuperLogo* será instalado e um atalho para sua execução aparecerá em *Iniciar / Todos os Programas* (procure pelo ícone azul com uma tartaruga). Dependendo da instalação, o ícone com o atalho também será colocado na área de trabalho. Para iniciar o *SuperLogo*, clique sobre um desses atalhos. Você verá abrir-se o “ambiente de programação” do *SuperLogo*, cujo aspecto deve ser semelhante ao mostrado na **Figura 10.1**. Note que as partes superior e inferior do ambiente são janelas independentes, que podem ser movidas e redimensionadas individualmente.



**Figura 10.1:** O ambiente de programação do *SuperLogo*.

## O AMBIENTE LOGO

Logo é uma linguagem interativa e interpretada. Um comando Logo pode ser executado isoladamente, sem necessidade de fazer parte de um programa completo. À medida que as linhas de comando são escritas, o ambiente Logo as “interpreta” (traduz) para o computador e ordena a sua execução.

A maioria dos ambientes Logo divide a tela do computador em, pelo menos, duas regiões: aquela onde são escritos os comandos e outra onde a tartaruga se move. Como estas áreas estão arrançadas, depende da implementação de Logo que estamos usando. A área de comandos do *SuperLogo* fica na janela normalmente colocada na parte de baixo da tela e contém duas caixas de texto e vários botões. A caixa inferior é a caixa de entrada, na qual são digitados os comandos a serem executados. Em cima dela está a caixa de saída, onde o Logo mantém uma lista dos comandos já executados e escreve suas mensagens de erro. A caixa de saída também é usada por comandos que escrevem alguma coisa. Por exemplo, digite na caixa de entrada

**escreva [alô mundo]**

e aperte a tecla <Enter> ou o botão *Executar*. Note que você deve escrever o “escreva”. Se nada der errado, na caixa de saída vão aparecer o comando que foi digitado e o resultado da sua execução: a frase “alô mundo”. O resultado deve ficar como o que está na **Figura 10.2**.



**Figura 10.2:** Janela de comandos com o resultado da instrução **escreva [alô mundo]**.

Veja também o que acontece com as instruções

**escreva 1+1**

**escreva raizq 9**

## **MOVIMENTOS DA TARTARUGA**

Uma das características do Logo é a existência de um objeto – a tartaruga – cujos movimentos são controlados por comandos da linguagem. A tartaruga é, geralmente, um pequeno desenho que se desloca pela tela do computador, mas nada impede que ela seja, como nas primeiras versões do Logo, um robô que anda pelo chão.

Para ter uma idéia dos comandos Logo que movimentam a tartaruga, digite na caixa de entrada

**parafrente 100**

Com isto, a tartaruga vai andar 100 passos (pontos da tela) para a frente. O resultado deve ser o mostrado na **Figura 10.3**.

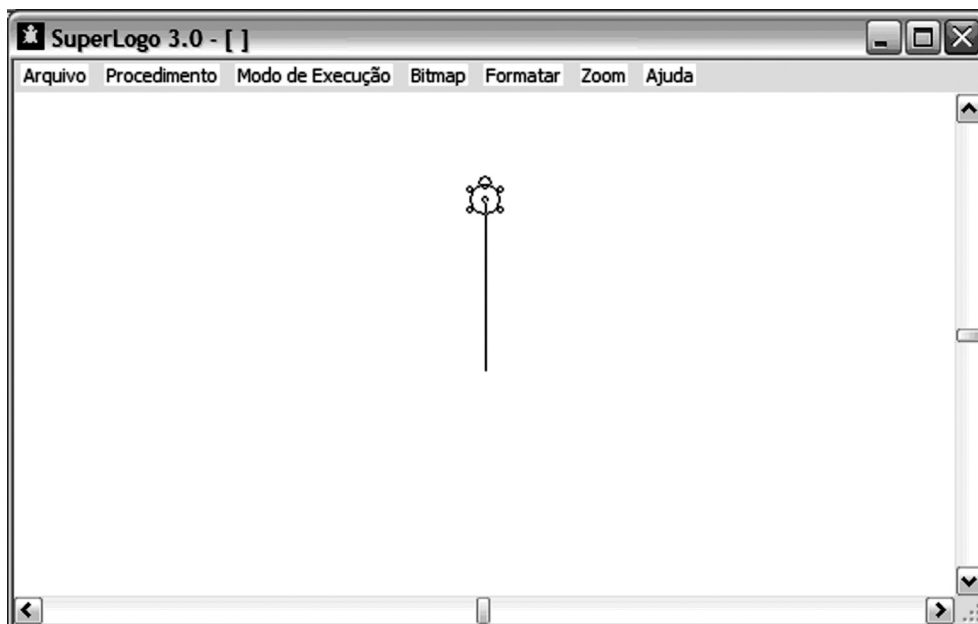


Figura 10.3: A tartaruga após a instrução **parafrente 100**.

Escreva também

```
paraesquerda 90
```

Note que a tartaruga girou 90 graus para a esquerda. Tente ainda

```
paratrás 80
```

```
paradireita 270
```

Estes são os comandos básicos para mover a tartaruga. Como muitos outros comandos Logo, eles podem ser escritos de forma abreviada:

```
parafrente = pf
```

```
paratrás = pt
```

```
paraesquerda = pe
```

```
paradireita = pd
```

A tartaruga deixa um rastro por onde passa, como se estivesse usando um lápis. Se quisermos movê-la sem desenhar, empregamos o comando **usenada**. Verifique o que acontece com

```
usenada  
pf 50
```

A tartaruga deve ter andado sem deixar rastros. Se quiser voltar a desenhar sobre a tela, utilize o comando **uselápis**. Por exemplo, faça

```
uselápis  
pt 70
```

A tartaruga pode ficar invisível. Para isto, basta executar o comando

```
desapareçatat
```

Para torná-la visível, execute

```
apareçatat
```

Para limpar a tela, apagando todos os desenhos, e recolocar a tartaruga na sua posição inicial, use o comando

```
tartaruga
```

que pode ser abreviado para **tat**, ou simplesmente aperte o botão *Tat* que está na janela de comandos.

## REPETIÇÕES

Podemos fazer a tartaruga desenhar um quadrado com as instruções (note como é possível escrever mais de um comando Logo por linha)

```
pf 50 pe 90  
pf 50 pe 90  
pf 50 pe 90  
pf 50 pe 90
```

Entretanto, é bem mais simples usar o comando `repita`,

```
repita 4 [pf 50 pe 90]
```

Um triângulo pode ser traçado com

```
repita 3 [pf 50 pe 120]
```

e um retângulo é obtido por meio de

```
repita 2 [pf 50 pe 90 pf 100 pe 90]
```

O comando **contevezes** (ou **cv**) fornece o número de repetições já realizadas pelo `repita`. Tente, por exemplo,

```
repita 3 [escreva contevezes]
```

que deve escrever os números 1, 2 e 3 na janela de comandos. Tente também

```
repita 200 [pf contevezes pe 90]
```

ou ainda

```
repita 720 [pf 10 pe contevezes]
```

e

```
repita 1800 [pf 10 pe contevezes+0.1]
```

Dentro da lista de comandos do `repita` podem estar outros `repita`. Veja o que acontece com

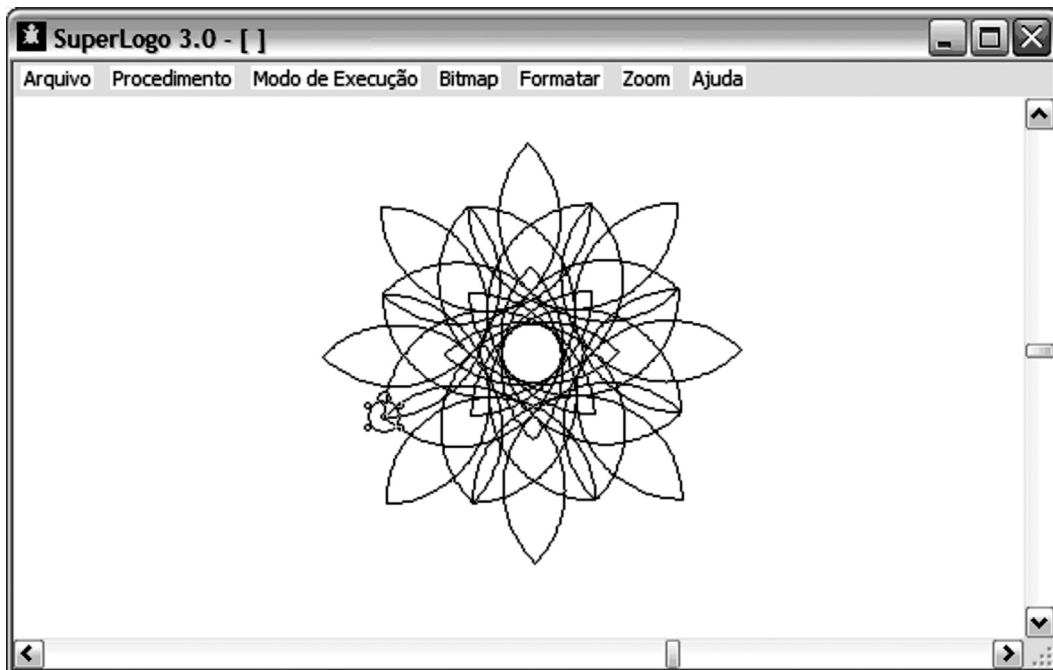
```
repita 6 [pe 60 repita 4 [pf 50 pe 90]]
```



Um resultado surpreendente, mostrado na **Figura 10.4**, é obtido com

```
repita 8 [pd 45 repita 6 [repita 90 [pf 2 pd 2]
pd 90]]
```

Mude o “6” por 1, 2... 7, para criar outras figuras interessantes.



**Figura 10.4:** Construção com vários comandos `repita` enlaçados.

## INFORMAÇÕES SOBRE A PRÓXIMA AULA

Até aqui temos usado o Logo no *modo interativo*, em que cada instrução é escrita e executada logo em seguida. Na próxima aula, começaremos a *programar* em Logo, criando procedimentos que contêm várias instruções, que são cumpridas quando o procedimento é executado.

# Programação em Logo

AULA

# 11

## Meta da aula

Apresentar alguns conceitos básicos de programação em Logo.

## objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- criar e editar procedimentos em Logo;
- definir e utilizar variáveis na linguagem Logo;
- utilizar funções matemáticas em programas Logo;
- compreender e criar procedimentos recursivos.

## PROCEDIMENTOS

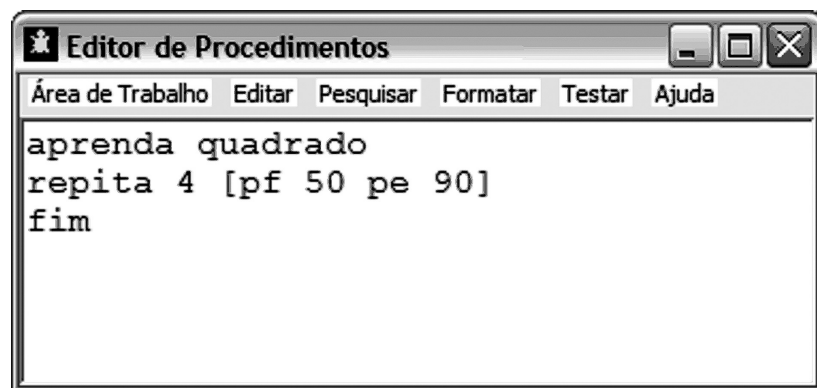
Uma das características mais importantes do Logo é a facilidade com que podemos estender a linguagem, criando novos comandos e operações (ou *procedimentos*). Por exemplo, vamos definir um comando chamado **quadrado**, que desenha um **quadrado** de lado 50. Antes, vamos verificar se já não há um procedimento com este nome. Escrevendo

**quadrado**

na caixa de entrada, obtemos como resposta a mensagem

Ainda não aprendi quadrado

ou seja, a tartaruga (ou melhor, o Logo) não conhece tal procedimento. Para definir o comando, clique *Procedimento / Editar* na barra de menu. Na janela que se abrir, escreva o nome do procedimento que deseja definir – no caso, “quadrado” – e clique no botão *OK*. A janela do editor de procedimentos aparecerá, então, com o início e o fim da definição do procedimento **quadrado** já escritos, como mostrado na **Figura 11.1**.



**Figura 11.1:** Janela do Editor de Procedimentos.

A definição propriamente dita do procedimento, **repita 4 [pf 50 pe 90]**, deve ser inserida entre o **aprenda** e o **fim**, como está na **Figura 11.2**.

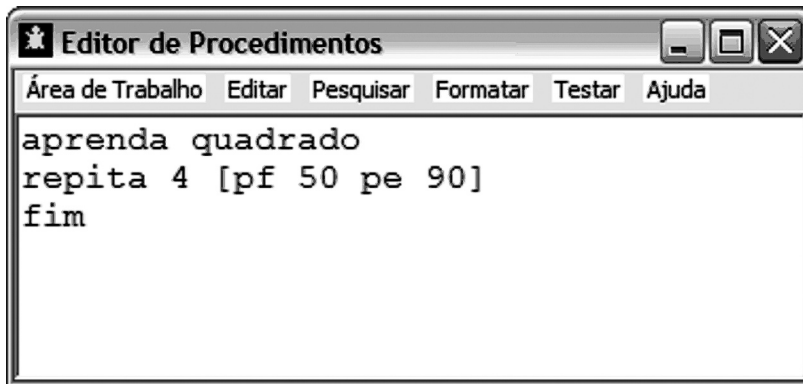


Figura 11.2: A definição do procedimento **quadrado**.

Escolha, em seguida, o item *Área de Trabalho / Atualizar* na barra de menu do editor e salve a definição do comando na memória do computador. Clicando *Área de Trabalho / Sair*, você fecha o editor de procedimentos. Escreva agora

#### **quadrado**

na janela de comandos e mande executar. Em vez de enviar uma mensagem de erro como antes, a tartaruga agora desenhará um quadrado de lado 50.

Esta definição de quadrado não é muito útil, pois obtemos sempre um quadrado de mesmo tamanho (50 pontos). Não podemos desenhar um quadrado de lado 100 ou 10. Seria melhor que **quadrado** fosse semelhante, por exemplo, ao comando *parafrente*, que tem um parâmetro que determina o tamanho do deslocamento. Assim, quadrado 100 faria um quadrado de lado 100. É fácil criar esta versão aperfeiçoada do comando. Para isto, volte ao Procedimento / Editar da barra de menu e escolha o procedimento **quadrado** na janela de opções. O procedimento definido anteriormente vai aparecer na janela do editor. Mude-o para

```
aprenda quadrado :lado
```

```
repita 4 [pf:lado pe 90]
```

```
fim
```

e salve o resultado. Agora, se tentarmos

```
quadrado 100
```

ou

```
quadrado 10
```

obteremos quadrados de lados 100 e 10, respectivamente. Veja também o que acontece com

```
repita 50 [quadrado 2*contevezes]
```

Note que o número que segue o comando quadrado é atribuído à variável *lado* definida pelo procedimento. Os dois pontos que são escritos antes da palavra lado indicam que estamos interessados no valor da variável *lado*, e não em um comando de nome *lado*. Podemos ler : **x** como “o valor da variável X” ou “o conteúdo de X” ou, ainda, “a coisa que estamos chamando X”.

## VARIÁVEIS

Já que entramos no assunto de variáveis, vamos aprender como elas podem ser criadas e receber valores. Por exemplo, para definir uma variável de nome *xy* com valor 2, podemos usar

```
atribua "xy 2
```

Se mandarmos escrever o valor de *xy*,

```
escreva :xy
```

obteremos o número 2 na caixa de saída. Observe que, se tentarmos

```
escreva "xy
```

teremos as letras *xy* como resposta. O papel das aspas é avisar ao Logo que o que vem depois delas deve ser considerado como uma palavra e não um procedimento. Se omitirmos as aspas e fizermos

```
escreva xy
```

ou

```
atribua xy 2
```

obteremos a mensagem de erro

*Ainda não aprendi xy*

ou seja, o Logo pensa que *xy* é um procedimento e avisa que ele não está definido.

Vamos usar o comando `atribua` para definir um procedimento que fornece o fatorial de um número inteiro. O procedimento **fat**, mostrado a seguir, calcula  $n!$  fazendo o produto  $1 \times 2 \times 3 \dots \times n$

```
aprenda fat :n
```

```
atribua "f 1
```

```
repita :n [atribua "f :f*contevezes]
```

```
envie :f
```

```
fim
```

Se fizermos em seguida o teste

```
escreva fat 5
```

vamos obter  $5! = 120$ . Observe o comando **envie**, que usamos no procedimento anterior: ele faz com que **fat** retorne como resultado o valor da variável *f*.

## OPERAÇÕES MATEMÁTICAS

Várias operações matemáticas estão definidas na linguagem Logo. Por exemplo, a exponenciação é feita com a operação **potência** `:a :b`, que retorna  $a^b$ . Veja o que acontece com

```
escreva potência 2 3
```

A raiz quadrada é dada por **raizq**, como já vimos. As funções trigonométricas seno, co-seno e tangente também estão definidas (**sen**, **cos**, **tan**), assim como as suas funções inversas (**arcsen**, **arccos**, **arctan**). Se executarmos a instrução

**escreva sen 30**

obtemos 0.5 como resposta. Vemos daí que os argumentos das funções trigonométricas são lidos em graus, não em radianos. Da mesma forma,

**escreva arctan 1**

resulta em 45 (graus) e não em  $\pi/4$  (radianos). Se quisermos trabalhar com radianos, em vez de graus, devemos usar as funções **senrad**, **cosrad**, **tanrad** e as inversas **arcsenrad**, **arccosrad**, **arctanrad**. Por exemplo,

**escreva senrad pi/2**

resulta em 1. Note que usamos, no último comando, o procedimento **pi**, que retorna o valor de  $\pi$ . Veja o que ocorre com

**escreva pi**

Figuras de Lissajous podem ser facilmente desenhadas com as funções trigonométricas. Por exemplo, execute a instrução

```
repita 360 [mudexy (100*sen 3*cv) (100*sen 4*cv)]
```

Observe que nessa instrução usamos um novo comando para mover a tartaruga – **mudexy :a :b** move a tartaruga até o ponto da tela de coordenadas  $(a, b)$ . O ponto  $(0, 0)$  corresponde ao centro da tela.

Um procedimento particularmente útil é **sorteienúmero**, que pode ser abreviado para **sortnum**. Executando **sorteienúmero:n**, obtemos um número inteiro entre 0 e  $n-1$ , escolhido ao acaso. Veja o que acontece com

```
repita 5 [escreva sortnum 10]
```

ou faça a tartaruga andar ao acaso com

```
repita 500 [pf 10 pe sortnum 180]
```

O logaritmo natural e a função exponencial também estão definidos no Logo. Eles são dados pelos procedimentos **ln** e **exponencial**, respectivamente. Uma lista completa das funções matemáticas definidas na *SuperLogo* pode ser encontrada no item *Ajuda* da barra de menu.

## Recursão

Procedimentos Logo podem ser recursivos. Isto significa que eles são capazes de chamar a si próprios. Veja como podemos usar um procedimento recursivo para calcular o fatorial de  $n$ , usando apenas que  $n! = n \times (n-1)!$  e  $0! = 1$ .

```
aprenda fatorial:n
```

```
se:n=0 [envie 1 pare]
```

```
envie:n*fatorial (:n-1)
```

```
fim
```

Note também o uso do comando **se**, que executa a lista de instruções dentro dos colchetes se a condição, no caso : **n=0**, for satisfeita. O comando **pare** termina a execução de um procedimento. Teste seu programa executando

```
escreva fatorial 5
```

Um procedimento recursivo que envolve desenhos com a tartaruga está mostrado a seguir:

```
aprenda arvore :x
```

```
se :x<1 [pare]
```

```
pf :x
```

```
pe 20
```

```
arvore :x/1.5
```

```
pd 40
```

```
arvore :x/1.5
```



```
pe 20
```

```
pt :x
```

```
fim
```

Executando a instrução

```
arvore 50
```

obtemos o resultado mostrado na Figura 11.3.

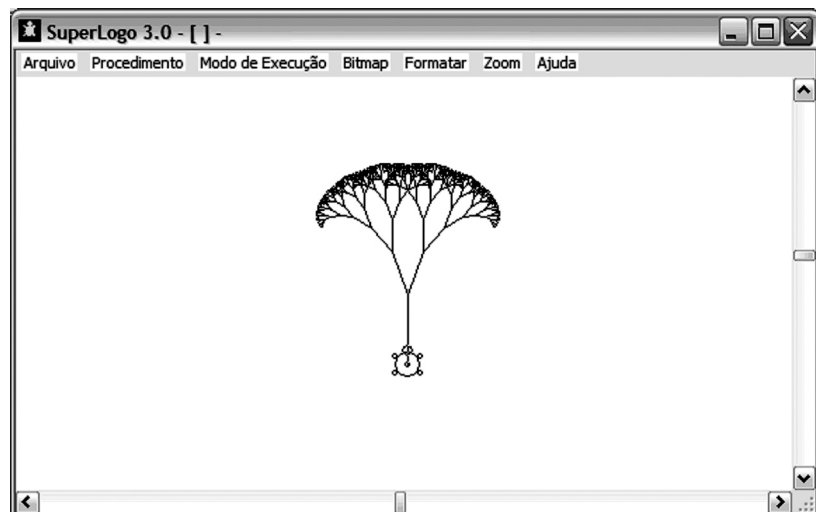


Figura 11.3: Desenho feito pelo procedimento `arvore`.

Outro procedimento recursivo é:

```
aprenda tri:x
```

```
se:x<3 [pare]
```

```
repita 3 [tri:x/2 pf:x pd 120]
```

```
fim
```

Executando

```
tri 200
```

a tartaruga desenha o fractal famoso, mostrado na Figura 11.4.

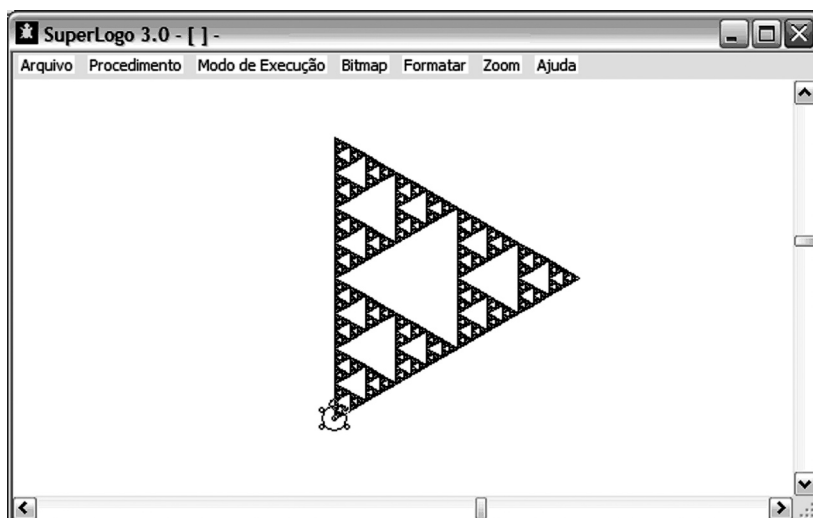


Figura 11.4: Fractal desenhado com o procedimento `tri`.

## SALVANDO SEUS PROCEDIMENTOS

Os procedimentos que você definiu após iniciar a sua sessão Logo estão guardados apenas na memória do computador. Se você terminar o programa Logo, todos eles serão perdidos. Para salvá-los em um arquivo em disco, vá para a barra de menu e selecione *Arquivo / Salvar como...*. Na caixa de diálogo que se abrir, escolha o disco, a pasta onde deseja salvar seus procedimentos e o nome do arquivo que irá contê-los. Note que todos os procedimentos que estão na memória irão para esse arquivo. Se você não deseja guardar algum procedimento, tem de apagá-lo da memória antes de salvar o resto. Isto é feito indo para *Procedimento / Apagar* na barra de menu e selecionando, na caixa de diálogo que se abrir, os procedimentos que serão descartados. Em uma próxima sessão Logo, os procedimentos salvos no arquivo podem ser carregados novamente para a memória, usando o menu *Arquivo / Abrir*.

## O QUE MAIS?

Há muito mais no Logo do que foi dito nas últimas aulas. Essas cobriram apenas uma pequena parte das características e recursos da linguagem. Muitos aspectos importantes, como a manipulação de listas (em certo sentido, o “coração” do Logo) e comandos de controle, não foram abordados. Uma visão mais completa da linguagem pode ser obtida com o programa de *Ajuda* do *SuperLogo*.

## INFORMAÇÕES SOBRE A PRÓXIMA AULA

Na próxima aula, vamos estudar um pouco de cálculo numérico e desenvolver um programa Logo que calcula o movimento de projéteis utilizando métodos numéricos.

# AULA 12

## Movimento de projéteis

### Meta da aula

Estudar o movimento de projéteis com métodos numéricos programados em linguagem Logo.

## objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- resolver numericamente equações de movimento pelo método de Euler;
- criar programas Logo que resolvam problemas de Mecânica, utilizando o método de Euler;
- analisar a precisão de soluções numéricas de equações de movimento.

## INTRODUÇÃO

O problema básico da dinâmica é determinar como se move uma partícula que está sob a ação de uma força. Para forças “simples”, o problema tem solução exata, e o movimento pode ser descrito a partir de expressões matemáticas envolvendo funções conhecidas (polinômios, senos, exponenciais...). Na maioria dos casos, entretanto, não é possível encontrar tais soluções. Se queremos estudar quantitativamente o movimento gerado por uma força “complicada”, devemos usar *métodos numéricos*. Estes métodos podem ser úteis mesmo quando soluções exatas existem, pois obtê-las analiticamente costuma ser uma tarefa difícil – acima das possibilidades de um aluno do Ensino Médio, por exemplo. Por outro lado, é relativamente fácil ensinar métodos numéricos de solução de equações de movimento – mesmo a alunos com pouca experiência matemática. Com isto, eles passam a ser capazes de discutir problemas físicos interessantes, que antes pareciam ser muito complicados.

Tomemos como exemplo o movimento unidimensional (sobre o “eixo  $x$ ”) de uma partícula. A força  $F(x, v, t)$  que age sobre a partícula pode depender, a princípio, da sua posição  $x$  e velocidade  $v$ , assim como do tempo  $t$ . A equação de movimento é dada pela segunda lei de Newton

$$F(x, v, t) = ma$$

onde  $m$  é a massa e  $a$  a aceleração da partícula. Resolver esta equação significa encontrar como a posição e a velocidade dependem do tempo, ou seja, determinar as funções  $x(t)$  e  $v(t)$ . Por exemplo, no caso de uma força constante  $F$ , temos

$$v(t) = v_0 + \frac{F}{m}t$$

$$x(t) = x_0 + v_0t + \frac{F}{2m}t^2$$

onde  $x_0$  e  $v_0$  são a posição e velocidade no instante  $t = 0$ .

Como já mencionamos, para forças mais complicadas, a solução da equação de movimento fica mais difícil, se não impossível, e métodos numéricos tornam-se úteis. O cálculo numérico de uma trajetória consiste em obter a posição e velocidade da partícula em um conjunto de instantes  $t_0, t_1, t_2, \dots, t_N$ , geralmente separados por um intervalo de duração fixa  $h$ :

$$t_n = t_0 + nh, \quad n = 0, 1, \dots$$

Chamando  $x_n$ ,  $v_n$  e  $a_n$  aos valores da posição, velocidade e aceleração no instante  $t_n$ , temos:

$$v_n \approx \frac{x_{n+1} - x_n}{h}$$

$$a_n \approx \frac{v_{n+1} - v_n}{h}$$

O que está escrito nessas equações é, basicamente, que a velocidade e aceleração no instante  $t_n$  são aproximadamente iguais à velocidade e aceleração médias no intervalo  $[t_n, t_{n+1}]$ . É claro que a aproximação só será razoável se  $h$  for pequeno, e ficará tanto melhor quanto menor for  $h$ . Rearranjando estas expressões, podemos escrevê-las como

$$x_{n+1} = x_n + v_n h$$

$$v_{n+1} = v_n + a_n h$$

onde, pela segunda lei de Newton,

$$a_n = F(x_n, v_n, t_n) / m$$

O procedimento numérico para calcular o movimento da partícula consiste em iterar essas equações a partir das condições iniciais. Dados  $x_0$  e  $v_0$  em  $t_0$ , obtemos  $x_1$  e  $v_1$  em  $t_1$ , e daí  $x_2$  e  $v_2$  em  $t_2$ , e assim por diante. Se o movimento é em duas ou três dimensões, o método continua o mesmo – devemos apenas escrever as equações na forma vetorial:

$$\vec{x}_{n+1} = \vec{x}_n + \vec{v}_n h$$

$$\vec{v}_{n+1} = \vec{v}_n + \vec{a}_n h$$

com

$$\vec{a}_n = \vec{F}(\vec{x}_n, \vec{v}_n, t_n) / m$$

Este é o *método de Euler* para resolver equações de movimento (ou qualquer equação diferencial). Embora ele não seja muito preciso, a sua simplicidade o torna muito atraente para fins didáticos. Mais adiante veremos como se pode melhorá-lo.

Vamos usar o método de Euler para obter a trajetória de um projétil sujeito à ação da gravidade e da resistência do ar. Supondo que essa resistência, também chamada *força de arrasto*, seja proporcional à velocidade do corpo em relação ao ar (o que nem sempre é realista), a força total é dada por

$$\vec{F} = m\vec{g} - b\vec{v}$$

onde  $g$  é a aceleração da gravidade e a constante  $b$  determina a intensidade do arrasto. O programa Logo listado a seguir mostra como se pode calcular a trajetória do projétil e traçá-la na tela do computador:

```

aprenda projetil :v :teta
atribua "g 9.8           ;aceleração da gravidade
atribua "m 1             ;massa
atribua "b 1.0           ;constante de arrasto
atribua "h 0.01          ;intervalo de tempo
atribua "s 10            ;escala(pixel / unid. compr.)
atribua "x -20           ;condições iniciais
atribua "y 0
atribua "vx :v * cos :teta
atribua "vy :v * sen :teta
atribua "t 0
desapareçatata          ;apaga a tartaruga
;desenha a linha do solo
usenada
mudexy -400 0
uselápis
mudexy 400 0
;coloca a tartaruga na posição inicial
mudexy (:x*:s) (:y*:s)
;calcula e desenha a trajetória
façaenquanto [passo] [:y>0]
fim

aprenda passo
força                   ;calcula a força
atribua "ax :fx/:m      ;calcula a aceleração
atribua "ay :fy/:m
atribua "x :x + :vx*:h   ;passo pelo método
de Euler
atribua "y :y + :vy*:h
atribua "vx :vx + :ax*:h
atribua "vy :vy + :ay*:h

```

```

atribua `t :t + :h
mudexy (:x*:s) (:y*:s) ;move a tartaruga
fim

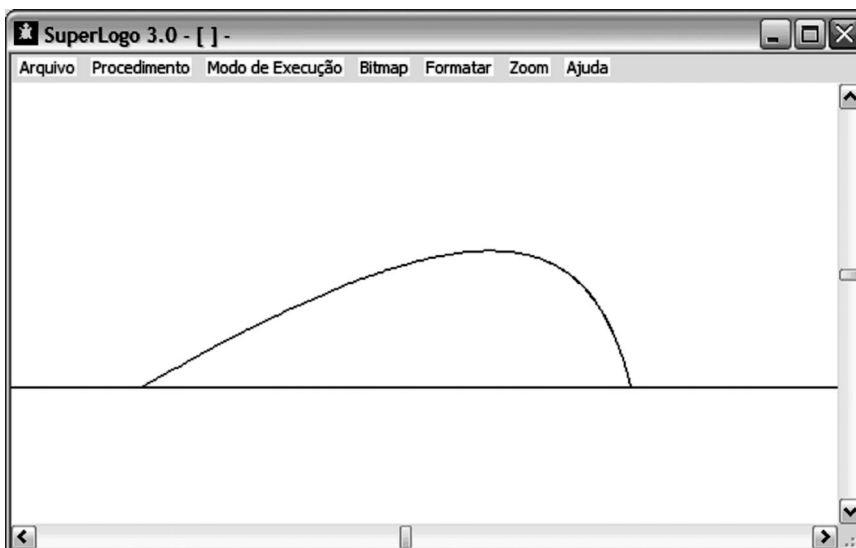
aprenda força
atribua `fx -:b*:vx
atribua `fy -:b*:vy - :m*:g
fim

```

Note que o procedimento principal **projetil** tem, como parâmetros de entrada, a velocidade inicial e o ângulo de lançamento. Ele define as constantes necessárias para o cálculo, determina as condições iniciais, desenha o chão e posiciona a tartaruga. O procedimento **passo** usa o método de Euler para calcular a posição e velocidade a cada passo  $h$  e move a tartaruga para as novas coordenadas. Ele é chamado repetidas vezes por **projetil**, enquanto a partícula estiver acima do solo ( $y > 0$ ). Por sua vez, **passo** chama o procedimento **aceleracao**, em que são calculadas as duas componentes da aceleração da partícula. Executando a instrução

```
projetil 40 30
```

você pode ver o que ocorre com um projétil lançado com velocidade inicial  $v = 40$  e ângulo  $\theta = 30^\circ$ . O resultado está na **Figura 12.1**:



**Figura 12.1:** Trajetória calculada com o programa **projetil**.



Uma questão importante neste programa (e outros) diz respeito ao sistema de unidades. É obviamente importante saber em que unidades devemos ler os números que entram e saem do programa. O fato de termos usado  $g = 9.8$  no programa não quer dizer necessariamente que estamos utilizando o sistema MKS; poderíamos estar estudando um planeta onde a aceleração da gravidade é  $9.8 \text{ cm/s}^2$ , ou até mesmo ter inventado um sistema de unidades específico para o problema.

Outro aspecto relacionado à escolha de unidades é a escala do gráfico mostrado na tela do computador. Devemos definir os fatores de escala, ou seja, quanto “mede” (nas unidades do programa) a distância entre dois pontos na tela. No programa, isto é dado pela variável  $s$ , que determina quantos “pixels” (pontos) adjacentes equivalem a uma unidade de comprimento:  $x*s$  e  $y*s$  são as coordenadas do projétil em número de pixels ou “unidades da tela”. Mudanças em  $s$  correspondem a um *zoom* sobre a trajetória.

### Precisão do cálculo

Quando se usa um método numérico, é fundamental estar atento à precisão dos resultados. Como vimos, o método de Euler fornece apenas uma aproximação para a posição e velocidade da partícula. Portanto, ao calcular uma trajetória, devemos verificar se o resultado aproximado que obtemos é suficientemente bom para os nossos propósitos. A questão é como fazer este teste. O primeiro passo é notar que o erro no método de Euler surgiu quando aproximamos a velocidade e aceleração médias no intervalo de tempo  $h = t_{n+1} - t_n$  pela velocidade e aceleração instantâneas no instante inicial  $t_n$ . Esta aproximação só é razoável para intervalos de tempo pequenos, e melhora à medida que  $h$  diminui. Entretanto, não é uma boa idéia adotar um valor de  $h$  demasiadamente pequeno no programa, pois isto o tornaria muito lento, devido ao grande número de passos necessários para calcular a trajetória. Existe um compromisso entre a precisão e o tempo de processamento. Um bom valor para o salto  $h$  é pequeno o suficiente para que os resultados tenham uma precisão aceitável, e grande o bastante para que o programa rode em um tempo razoável. Uma forma simples e eficiente de encontrar este  $h$  é fazermos o cálculo com um salto  $h_1$  que pareça razoável (por exemplo, menor que todos os tempos característicos do sistema) e, em seguida, refazermos tudo com um valor

bem menor,  $h_2 = h_1/10$  digamos. O segundo cálculo é mais preciso que o primeiro. Se, dentro da precisão que nos interessa, as duas trajetórias forem indistinguíveis (os seus gráficos parecem idênticos na tela, por exemplo), a melhoria obtida com o uso de  $h_2$  é irrelevante, ou seja,  $h_1$  é um “bom” valor para o salto de tempo. Por outro lado, se a diferença entre os dois cálculos for grande demais, isto significa que a utilização de  $h_2$  melhorou apreciavelmente o resultado e, portanto,  $h_1$  não dá uma boa precisão. O valor  $h_2$  é melhor; mas como saber se a precisão que ele confere ao cálculo já é suficiente? Basta começar tudo de novo com um salto  $h_3 = h_2/10$  e comparar com o resultado de  $h_2$ . Repetindo este procedimento, acabaremos por encontrar um valor satisfatório para  $h$ . A Figura 12.2 mostra um exemplo: vemos que, usando  $h = 0.01$ , é possível obter com boa precisão a trajetória pretendida com **projeti1 40 30**.

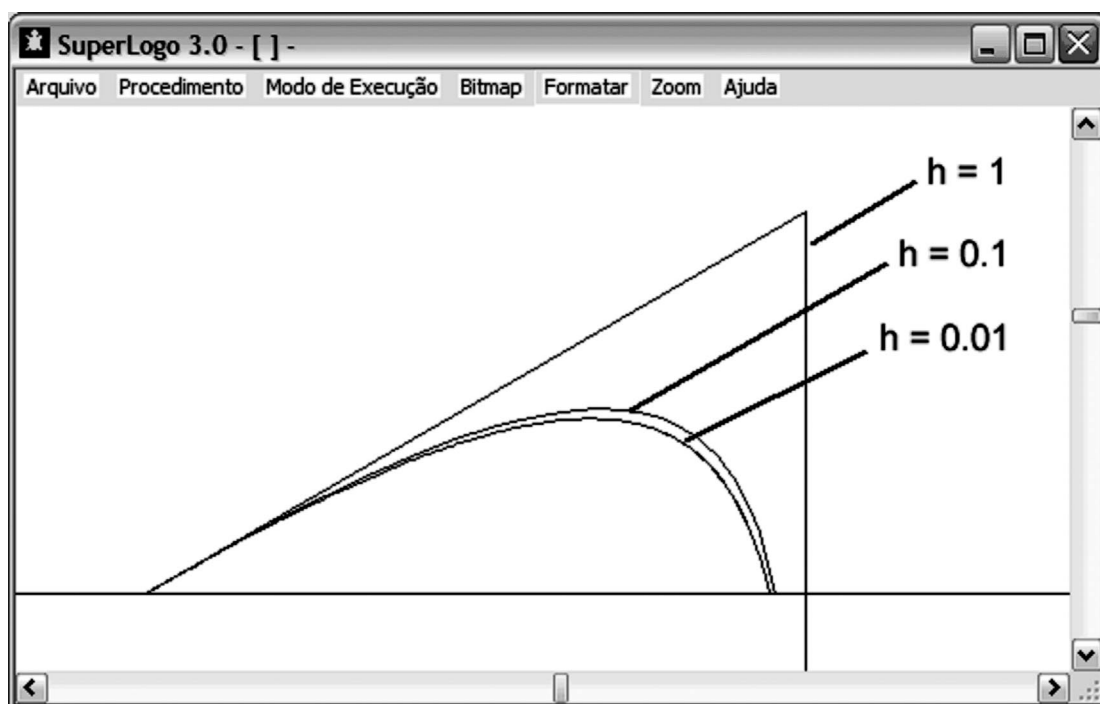


Figura 12.2: Cálculo da trajetória para diferentes valores de  $h$ .

Note que este valor de  $h$  não é “universal” – ele pode não levar a bons resultados em outras situações. Por exemplo, mostre que ele não dá uma boa precisão quando a constante de arrasto for  $b = 10$  e a velocidade inicial for  $v_0 = 4$  (use um fator de escala  $s = 1.000$  para ver as trajetórias). Determine um bom valor para  $h$  neste caso.

### INFORMAÇÃO SOBRE A PRÓXIMA AULA

Na próxima aula, vamos aplicar o programa que desenvolvemos a problemas de movimento de projéteis.

## Exercícios sobre o movimento de projéteis

# AULA 13

### Meta da aula

Aplicar o programa desenvolvido na aula anterior a diversos problemas físicos de interesse.

## objetivo

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- resolver problemas de movimento balístico com programas Logo.

## ALCANCE MÁXIMO

Sem a resistência do ar, o alcance máximo de um projétil é atingido quando o ângulo de lançamento é de 45 graus. A resistência do ar aumenta ou diminui o ângulo de alcance máximo? Você pode encontrar a resposta com o programa **projétil**, variando o ângulo de lançamento e mantendo a velocidade inicial fixa. Veja, por exemplo, o que acontece ao executar a instrução **repita 8 [projétil 40 (10\*contevezes)]**. Neste caso (velocidade inicial = 40), qual é aproximadamente o ângulo de maior alcance?

## BALÍSTICA MEDIEVAL

Antes do século XVII, pensava-se que um projétil subia em linha reta, fazia uma curva no alto da trajetória e, em seguida, caía verticalmente. A Figura 13.1, de um livro de 1561 (Daniele Santbech, *Problematum Astronomicorum*), ilustra bem essa noção pré-galileana de movimento balístico. Mostre que a idéia de trajetórias “quase-triangulares” é aproximadamente correta, se a velocidade inicial ou o coeficiente de atrito forem grandes o suficiente. Note que, para ver melhor a trajetória, você terá que mudar um pouco a escala **s** em ambos os casos.

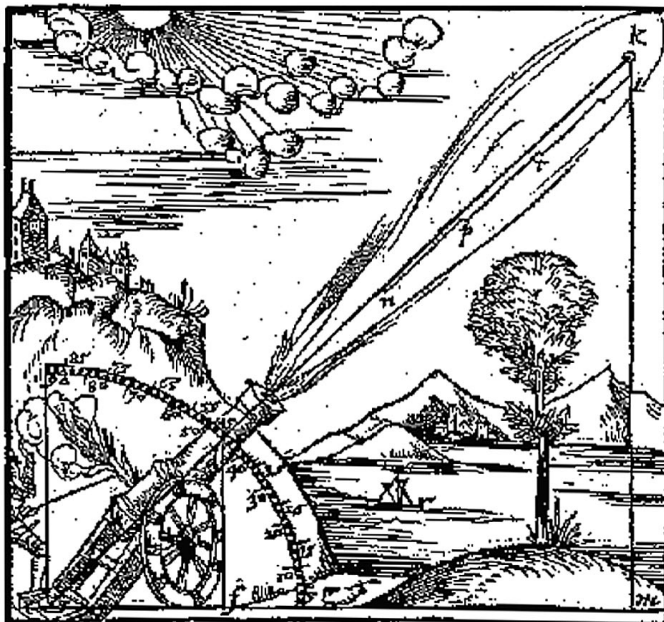


Figura 13.1: A noção medieval de trajetória balística.

## EFEITO DO VENTO

A força de arrasto depende da velocidade do corpo em relação ao ar. Se soprar um vento com velocidade  $\vec{u}$ , a resistência do ar será  $\vec{F} = -b(\vec{v} - \vec{u})$ . Suponha que o projétil seja lançado quando houver um vento de velocidade  $u$  na direção horizontal. Modifique o programa **projetil**, para levar em conta o efeito deste vento. Calcule algumas trajetórias com vento “contra” e “a favor”.

## ATRITO NÃO-LINEAR

A força de atrito que temos usado aumenta linearmente com a velocidade do projétil, o que só é realista para um corpo pequeno que se move lentamente. Uma aproximação melhor, válida para corpos macroscópicos em uma grande faixa de velocidades, é obtida supondo uma dependência quadrática na velocidade:

$$\vec{F} = -c |\vec{v}| \vec{v}$$

onde  $|\vec{v}| = \sqrt{v_x^2 + v_y^2}$  é o módulo do vetor velocidade. Para uma esfera de raio  $r$ , medidas da força de arrasto mostram que  $c$  é, aproximadamente,

$$c \approx 0,7 \rho r^2$$

onde  $\rho$  é a densidade do meio (aproximadamente  $1,2\text{kg/m}^3$  para o ar). Modifique o programa **projetil**, para que a força de atrito tenha a forma dada acima. Calcule algumas trajetórias de uma bola de futebol (11cm de raio e 430g de massa, segundo a FIFA) chutada a 20m/s. Faça o programa escrever o alcance e o tempo de vôo da bola. Compare com os resultados obtidos quando se ignora a resistência do ar.

## INFORMAÇÕES SOBRE A PRÓXIMA AULA

Na próxima aula, veremos que o método de Euler tem limitações e desenvolveremos uma técnica mais eficiente para resolver equações diferenciais: o método de Euler-Cromer. O movimento planetário será estudado com este último método.

# AULA 14

## Movimento orbital

### Meta da aula

Desenvolver métodos numéricos e programas Logo para estudar o movimento planetário.

## objetivos

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- reconhecer em que situações a pouca precisão do método de Euler pode comprometer a solução numérica de equações de movimento;
- utilizar o método de Euler-Cromer como alternativa ao método de Euler;
- fazer programas Logo que calculem numericamente movimentos orbitais.

## INTRODUÇÃO

O movimento de um planeta em torno do Sol é determinado pela equação de movimento

$$m \frac{d^2 \vec{r}}{dt^2} = \vec{F} = -GMm \frac{\vec{r}}{r^3}$$

onde  $r$  é a distância do Sol ao planeta,  $M$  é a massa do Sol (suposta ser muito maior que a massa  $m$  do planeta) e  $G$  é a constante gravitacional. A partir dessa equação, Newton demonstrou que as órbitas planetárias têm a forma de elipses, com o Sol em um dos focos, tal como fora observado por Kepler. Ele também mostrou que trajetórias parabólicas e hiperbólicas são possíveis, correspondendo a movimentos que não ficam limitados a uma região finita em torno do Sol. De maneira geral, as trajetórias em um campo gravitacional têm a forma de seções cônicas. Em coordenadas polares, elas são dadas pela fórmula

$$r = \frac{d}{1 - e \cos \theta}$$

Os parâmetros  $d$  e  $e$  são dados por

$$d = \frac{r_0^2 v_0^2}{GM}$$

$$e = \left| \frac{r_0 v_0^2}{GM} - 1 \right|$$

onde  $r_0$  e  $v_0$  são o raio e velocidade em um ponto de retorno da órbita (onde a velocidade radial se anula; veja a **Figura 14.1** para o caso da elipse).

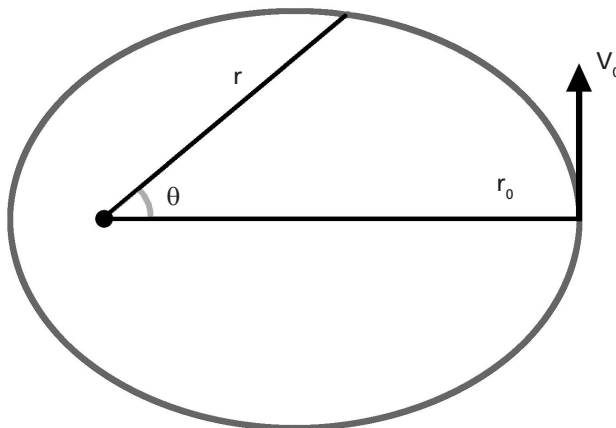


Figura 14.1: Órbita elíptica.



O parâmetro  $e$  é chamado de *excentricidade*. Se  $e = 0$ , a trajetória é circular; para  $0 < e < 1$ , ela é uma elipse;  $e = 1$  corresponde a uma parábola;  $e > 1$  a uma hipérbole.

Os resultados acima não são difíceis de obter, mas o procedimento usual exige um conhecimento de cálculo acima das possibilidades dos alunos da escola média. A consequência disso é que o estudo do movimento planetário costuma ser reduzido às órbitas circulares. É possível, contudo, usar métodos numéricos para fugir dessas limitações. Resolvendo numericamente as equações de movimento, podemos estudar as órbitas não-circulares de uma forma que é fácil de ser compreendida e explorada pelos alunos. Com isso, abrimos espaço para abordar tópicos importantes, como as leis de Kepler e o movimento dos cometas.

O ponto de partida da solução numérica pode ser o método de Euler, que já discutimos em detalhe ao tratar do movimento de projéteis. Por esse método, se  $\vec{r}_n$  e  $\vec{v}_n$  são a posição e velocidade do planeta no instante  $t_n$ , teremos no instante  $t_{n+1} = t_n + h$

$$\begin{aligned}\vec{r}_{n+1} &\approx \vec{r}_n + \vec{v}_n h \\ \vec{v}_{n+1} &\approx \vec{v}_n + \vec{a}_n h\end{aligned}$$

onde

$$\vec{a}_n = -GM \frac{\vec{r}}{r^3}$$

O programa mostrado a seguir é um procedimento Logo, **kepler**, que usa o método de Euler para traçar na tela do computador a trajetória de um “planeta” que é atraído por uma “estrela”. O procedimento **kepler** tem dois parâmetros, **r** e **v**, que são a distância inicial  $r_0$  entre o planeta e a estrela, e a velocidade inicial  $v_0$  do planeta. No programa, a posição inicial é um ponto de retorno, ou seja, o vetor velocidade é perpendicular ao vetor posição (veja a **Figura 14.1**). Note que as unidades que escolhemos são tais que  $GM = 1$  e  $m = 1$ .

```
aprenda kepler :r :v
atribua "GM 1           ;cte. gravitacional * massa solar
atribua "m 1           ;massa do planeta
atribua "h 0.01        ;intervalo de tempo
atribua "s 100         ;escala (pixel / unid. compr.)
```

```

atribua "tmax 20 ;tempo máximo
atribua "x :r ;condições iniciais
atribua "y 0
atribua "vx 0
atribua "vy :v
atribua "t 0
desapareçatat ;apaga a tartaruga
;desenha a estrela

usenada
mudexy 0 0
uselápis
arco 360 6
pinte
;coloca a tartaruga na posição inicial
usenada
mudexy (:x*:s) (:y*:s)
uselápis
;calcula e desenha a trajetória
façaenquanto [passo] [:t<:tmax]
fim

aprenda passo
força ;calcula a força
atribua "ax :fx/:m ;calcula aceleração
atribua "ay :fy/:m
atribua "x :x + :vx*:h ;passo pelo método de Euler
atribua "y :y + :vy*:h
atribua "vx :vx + :ax*:h
atribua "vy :vy + :ay*:h
atribua "t :t + :h
mudexy (:x*:s) (:y*:s) ;move a tartaruga
fim

aprenda força
atribua "r3 potência raizq(:x*:x + :y*:y) 3
atribua "fx -:GM*:m*:x/:r3
atribua "fy -:GM*:m*:y/:r3
fim

```

Podemos ver que a estrutura do programa é essencialmente a mesma do procedimento **projctil**, que usamos para estudar o movimento de projéteis. Note que o programa só termina quando  $t > t_{\max}$  ou, então, quando pressionamos o botão *Parar* na janela de comando.

Teste o programa executando o comando **kepler 1 1**. Isso corresponde às condições iniciais  $r_0 = 1$  e  $v_0 = 1$  que, nas unidades que utilizamos ( $GM = 1$ ), devem produzir uma trajetória circular. Entretanto, o que obtemos com o programa não é um círculo, mas uma espiral cujo raio aumenta com o tempo, como mostrado na **Figura 14.2**. A causa desse erro já foi discutida anteriormente: a solução numérica é uma solução aproximada. Ela será tanto melhor quanto menor for o salto de tempo  $h$ , mas nunca será exata. Se diminuirmos o valor de  $h$ , o erro no cálculo da trajetória deve ser reduzido. Verifique isto usando  $h = 0.001$  no programa, no lugar do  $h = 0.01$  do primeiro cálculo. Você encontrará uma trajetória mais próxima da circular, ou seja, o raio da órbita não aumenta tanto a cada revolução. No entanto, note que esta maior precisão foi obtida à custa de tornar o programa muito mais lento. Se calcularmos uma trajetória elíptica, executando **kepler 2 0.4**, por exemplo, veremos que, mesmo com  $h = 0.001$ , estamos longe de obter uma elipse. É claro que, reduzindo ainda mais o valor de  $h$ , poderemos obter uma órbita suficientemente precisa. Mas também é fácil perceber que precisões razoáveis exigirão tempos de cálculo muito longos.

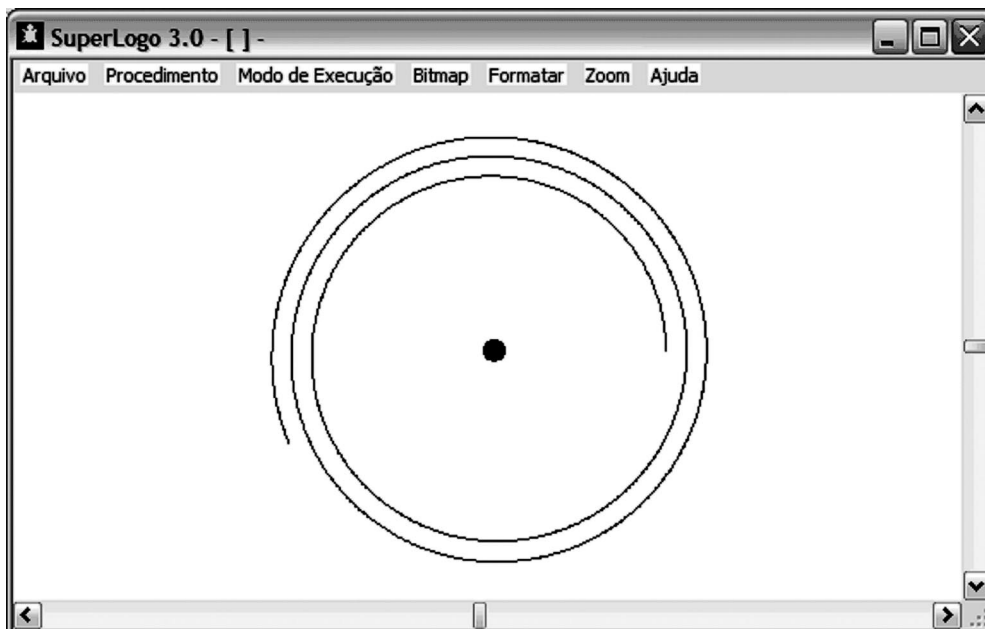


Figura 14.2: Órbita calculada com o método de Euler.

## O MÉTODO DE EULER-CROMER

O problema de precisão pode ser amenizado usando-se métodos numéricos mais eficientes que o de Euler. Para movimentos periódicos, é possível obter uma melhoria significativa na precisão do cálculo sem muito esforço, usando o *método de Euler-Cromer*.

$$\begin{aligned}\vec{v}_{n+1} &\approx \vec{v}_n + \vec{a}_n h \\ \vec{r}_{n+1} &\approx \vec{r}_n + \vec{v}_{n+1} h\end{aligned}$$

A única diferença em relação ao método de Euler original é o uso da velocidade  $\vec{v}_{n+1}$ , em vez de  $\vec{v}_n$ , no cálculo da posição  $\vec{r}_{n+1}$ . Isto equivale a escrever

$$\vec{v}_{n+1} \approx \frac{\vec{r}_{n+1} - \vec{r}_n}{h},$$

que é uma aproximação tão boa quanto a que foi utilizada para obter o método de Euler,

$$\vec{v}_n \approx \frac{\vec{r}_{n+1} - \vec{r}_n}{h}.$$

O método de Euler modificado é tão fácil de programar quanto o original – basta trocar a ordem das linhas que atualizam posição e velocidade. O programa a seguir mostra como isso pode ser feito no procedimento **passo** usado em **kepler**.

```

aprenda passo
força                                ;calcula a força
atribua "ax :fx/:m                    ;calcula aceleração
atribua "ay :fy/:m
atribua "vx :vx + :ax*:h ;passo pelo método
                                de Euler-Cromer
atribua "vy :vy + :ay*:h
atribua "x :x + :vx*:h
atribua "y :y + :vy*:h
atribua "t :t + :h
mudexy (:x*:s) (:y*:s) ;move a tartaruga
fim
    
```

Modifique o programa **kepler**, de forma a calcular a trajetória com o método de Euler-Cromer. Tome  $h = 0.01$  e execute **kepler 1 1**. Note como a precisão do cálculo melhorou. Verifique se uma trajetória elíptica é obtida com **kepler 2 0.4**.

### INFORMAÇÃO SOBRE A PRÓXIMA AULA

Na próxima aula, usaremos o programa kepler para estudar vários aspectos do movimento planetário.

# AULA 15

## Exercícios sobre movimento orbital

### Meta da aula

Aplicar o programa Logo desenvolvido na aula anterior a diversos problemas físicos de interesse.

## objetivo

Esperamos que, após o estudo do conteúdo desta aula, você seja capaz de:

- resolver problemas de movimento orbital com programas Logo.

## TRAJETÓRIAS EXCÊNTRICAS

Trace alguns exemplos de trajetórias elípticas, parabólicas e hiperbólicas. Modifique o programa **kepler**, fazendo com que ele calcule a excentricidade da órbita a partir das condições iniciais e escreva o resultado. Faça-o escrever também a energia e o momento angular correspondentes.

## TRAJETÓRIA NO “ESPAÇO DE VELOCIDADES”

Modifique o programa **kepler**, para que este mostre na tela a evolução do vetor velocidade em vez do vetor posição. Que formas têm as órbitas no “espaço de velocidades”?

## ATRITO COM A ATMOSFERA

Considere um satélite em órbita de um planeta com atmosfera. Se a órbita é baixa o suficiente para penetrar na atmosfera, os efeitos da resistência do ar devem ser considerados. Estude o efeito que uma força de atrito proporcional a  $v$  tem sobre as órbitas. Mostre que a velocidade do satélite *umenta* devido ao atrito com a atmosfera. Tente explicar como isso pode ocorrer. (Pense: o atrito diminui a energia cinética ou a energia total?)

## GRAVITAÇÃO NÃO-NEWTONIANA

O que aconteceria se a força gravitacional fosse proporcional a  $r^{-2,1}$ , em vez de  $r^{-2}$ ? Mostre que, nesse caso, as órbitas elípticas teriam um movimento de precessão. E se a força fosse proporcional a  $r$  (um oscilador harmônico)? Mostre que as órbitas seriam elipses, como no caso newtoniano, mas com o centro (e não o foco) na origem da força.