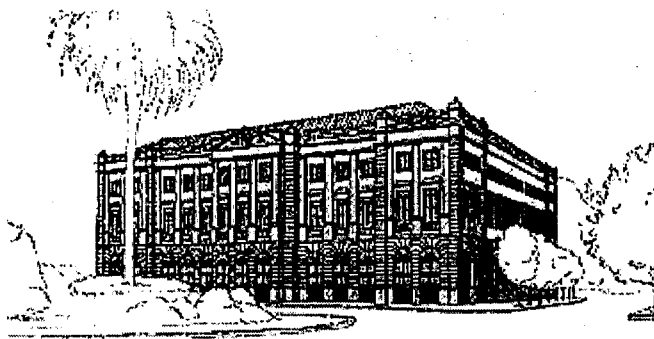


UNIVERSIDADE DO PORTO
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA PURA

Reg. S09080
Cota Tese N° 116 E2



MAPLETS

- Modelos Interactivos no Ensino da Matemática -

Faculdade de Ciências do Porto
MATEMÁTICA

Avelino José Rodrigues de Sousa



DEZEMBRO 2003



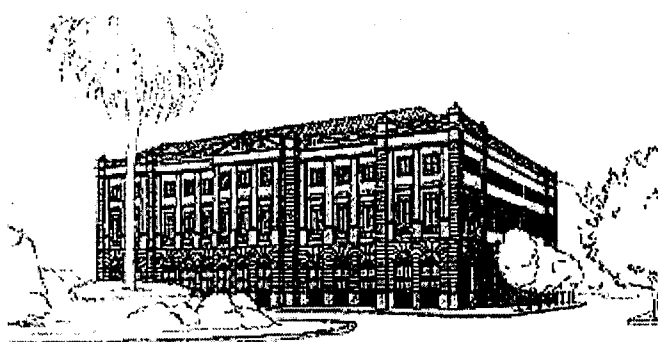
FC

Biblioteca
Faculdade de Ciências
Universidade do Porto



0000101115

UNIVERSIDADE DO PORTO
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA PURA



MAPLETS

- Modelos Interactivos no Ensino da Matemática -

Tese submetida à Faculdade de Ciências da Universidade do Porto para
obtenção do grau de Mestre em Ensino da Matemática.

Orientada por **Fernando Jorge Soares Moreira**

Realizada por **Avelino José Rodrigues de Sousa**



DEZEMBRO 2003

Martim Delgado

Conteúdo

1	Introdução	3
1.1	Utilização dos computadores no ensino	4
1.2	O Maple no ensino da Matemática	6
1.3	Acerca deste trabalho	8
2	Noções Básicas do Maple	25
2.1	Introdução	25
2.2	Iniciar uma sessão de Maple	26
2.3	Área de trabalho	28
2.4	Ajuda interactiva	31
2.5	A Linguagem do Maple	31
2.5.1	Números	31
2.5.2	Atribuição de um nome a uma expressão	37
2.5.3	Sequências e Listas	42
2.5.4	Gráficos	46
2.5.5	Cálculo Diferencial e Integral	62
2.5.6	Programação e Procedimentos	65
3	Maplets	81
3.1	O que é uma Maplet ?	81
3.2	Maplets Simples	83
3.3	Maplets para Matemática	85

3.4	Outras Maplets	101
4	Construção de Maplets	105
4.1	Maplets - Um módulo do Maple	105
4.2	Construção de uma Maplet simples	106
4.3	Terminologia	108
4.4	Maplets[Elements]	109
4.5	Maplets[Examples]	143
4.6	Maplets[Tools]	149
5	Uma Maplet para o estudo gráfico de funções	155
5.1	A janela principal	155
5.2	A janela de opções	165
5.3	As janelas do Zoom e do Trace	177
5.4	A janela sobre o estudo do gráfico	194
5.5	A Maplet actual	216
6	Exemplos de Maplets e suas aplicações	221
6.1	Exemplos úteis no Ensino	221
6.1.1	Integração	222
6.1.2	Derivação	228
6.1.3	Cálculo de limites	231
6.2	Gráficos de Funções com mais de uma variável	233
6.3	Maplets para Sistemas Dinâmicos	239
A	Códigos	253
	Bibliografia	259

Capítulo 1

Introdução

Durante muitos anos, o ensino da Matemática pautou-se pelo método expositivo tradicional, onde o papel dos alunos era meramente passivo. De há uns anos para cá temos assistido a uma mudança do ensino da Matemática, não só ao nível dos programas, mas também ao nível dos métodos de ensino. Cada vez mais, os professores utilizam o computador e outras tecnologias no processo de ensino/aprendizagem, pois isso permite que determinados conteúdos programáticos sejam abordados de uma forma diferente, motivando os alunos para disciplina de Matemática.

O professor confronta-se no seu dia a dia, perante uma grande diversidade de alunos, com a necessidade de criar condições para uma aprendizagem bem sucedida. Uma das principais tarefas que tem pela frente é ganhar o interesse dos alunos pelos conteúdos e a seguir manter essa motivação latente, propondo tarefas, desafios, animando discussões, gerindo o trabalho individual, de pequeno e grande grupo. Para o efeito, recorre a vários mediadores na planificação do seu trabalho: o manual escolar, fichas de trabalho, materiais manipulativos e, cada vez mais, tecnologia. Embora se reconheça que relativamente ao uso das tecnologias, as que se encontram mais divulgadas são as calculadoras (básicas, científicas e gráficas), é sobre a utilização de computadores no ensino da Matemática que se vai inserir este trabalho.

1.1 Utilização dos computadores no ensino

A sociedade moderna encontra-se hoje muito dependente da tecnologia informática. Em todos os ramos da sociedade pode-se constatar a utilização e consequente influência destes novos instrumentos, ou seja, os computadores.

No que diz respeito ao ensino da Matemática, a utilização de computadores, acompanhada de *software* adequado, pode contribuir para um enriquecimento dos ambientes de aprendizagem, facilitando a abordagem informal de conceitos de Matemática e tornando-a mais "experimental".

A relação que o aluno estabelece com o computador pode provocar diversas reacções no plano afectivo. Devido à fácil interactividade com o computador é criado um novo ambiente de aprendizagem. Deste modo, e como a relação entre o aluno e o computador é impessoal, para alguns alunos os erros deixam de ser embaraçosos e passam a ser algo que serve para novas aprendizagens. Nesta perspectiva, o computador é utilizado pelo aluno para explorar, modificar e desenvolver o conhecimento que se encontra localizado na sua mente. Este processo de aquisição de conhecimentos enquadra-se perfeitamente no modelo construtivista da aprendizagem.

A utilização de meios computacionais na aprendizagem dos conteúdos matemáticos pode proporcionar aos alunos uma melhor adesão ao ambiente de trabalho, na medida em que estes, mesmo depois de resolvidos os problemas, tendem a ficar mais motivados para novas experiências. A fácil interactividade com o computador permite aos alunos um novo meio de aprendizagem, onde persiste a deliberação, o debate e a concentração, bem como o trabalho de grupo e a cooperação. Por outro lado, se os professores não explorarem devidamente o uso desta tecnologia, os computadores podem levar à individualização e isolamento dos alunos. Além disso, podem surgir dificuldades por parte dos alunos, pelo facto do computador ser menos flexível e menos capaz de responder às suas questões do que o professor. É por isso que o professor desempenha um papel relevante, contribuindo

para colocar questões e lançar desafios, que valorizam o ambiente de aprendizagem.

O computador oferece à educação matemática possibilidades que ainda não foram totalmente exploradas. A sua utilização está relacionada com realidades muito diversas que podem ser influenciadas pela concepção que o professor tem sobre o ensino e o uso do computador. Quanto a este aspecto, os professores têm diversas opiniões: alguns consideram o contributo do computador na sala de aula importante, pois pode permitir o entusiasmo e a motivação dos alunos; outros consideram que a utilização do computador apenas facilita a rápida e mais rigorosa realização de algumas tarefas normalmente realizadas à mão (ex.: gráficos e cálculos morosos) e outros consideram que o computador é um instrumento privilegiado para a realização de actividades difíceis de realizar de outra forma (ex.: modelação de alguns fenómenos reais).

A integração das tecnologias no processo de ensino/aprendizagem levou a algumas transformações no papel do professor: ele deixa de ser o centro da aula para passar a ser o organizador e coordenador das várias tarefas, tornando-se as aulas em centros de criação e investigação, onde se recorre frequentemente à experimentação, discussão e reflexão. A partir de uma situação inicial em que os alunos exploram o programa e colocam as suas conjecturas, o professor tem como função moderar a discussão entre os elementos da turma, durante a qual se realiza uma troca de saberes, fazendo-se no final uma reflexão sobre os conhecimentos adquiridos durante a aula.

Com a utilização do computador, o professor deixa de ser aquele que detém todo o conhecimento e passa a ser alguém que, na presença de um problema concreto, pode ter tantas dificuldades quanto os alunos. O professor passa igualmente a ser um factor de coesão e animação de grupo, bem como um exemplo de espírito crítico e um entusiasmado companheiro de descoberta.

No entanto, neste contexto, o professor pode-se deparar com diversas dificuldades tais

como: insuficiência de equipamento nas salas de aula, a obtenção de *software* adequado aos conteúdos a leccionar, a gestão do tempo e o acompanhamento dos diversos grupos de trabalho, descobertas imprevistas que os alunos fazem, relacionar o trabalho dos alunos com o computador e fora dele.

É importante referir que a utilização de computadores na educação matemática, pode influenciar positivamente a atitude do aluno face ao trabalho proposto pelo professor e face à própria Matemática. É também de salientar que o uso de computadores pode proporcionar uma forma interessante e divertida de aprender, minimizando a atitude de desânimo manifestada por grande parte dos alunos.

Deste modo, é importante que o professor disponha de programas, onde seja possível transmitir e interagir os conteúdos matemáticos com os alunos. Neste trabalho veremos como o Maple pode ser utilizado para esses fins.

1.2 O Maple no ensino da Matemática

O Maple, a par com o Mathematica, é um dos mais conhecidos e utilizados programas de *software* de computação numérica e simbólica. Este programa está apetrechado desde há longa data de rotinas que permitem modelar e introduzir de uma forma mais interactiva conteúdos programáticos dos mais diversos ramos da matemática. Tanto no ensino superior como no ensino secundário, o uso deste programa pode trazer mais valias ao método de ensino "tradicional" baseado em manuais e com pouco recurso à experimentação e simulação.

A utilização no ensino de programas como o Maple ou o Mathematica, pressupunha até há bem pouco tempo que, o professor e o aluno conhecessem, pelo menos a um nível elementar, o funcionamento deste *software*. O tempo de aprendizagem da sintaxe dos programas era e é uma das razões para grande número de professores e alunos prescindirem

das mais valias que os programas deste tipo podiam trazer. De facto, para usar todas as potencialidades do Maple, era necessário saber o seu código, isto é, os utilizadores precisavam saber a linguagem de programação deste sistema, para trabalhar em Maple. Um utilizador tinha de ser também um conhecedor de linguagem Maple. Este impeditivo, foi bastante minimizado com a introdução em 2002, da possibilidade de elaborar, na linguagem Maple, *interfaces* gráficas "amigáveis" para os utilizadores finais - As Maplets¹.

A figura 1.1 é um exemplo de uma Maplet, na sua forma final.

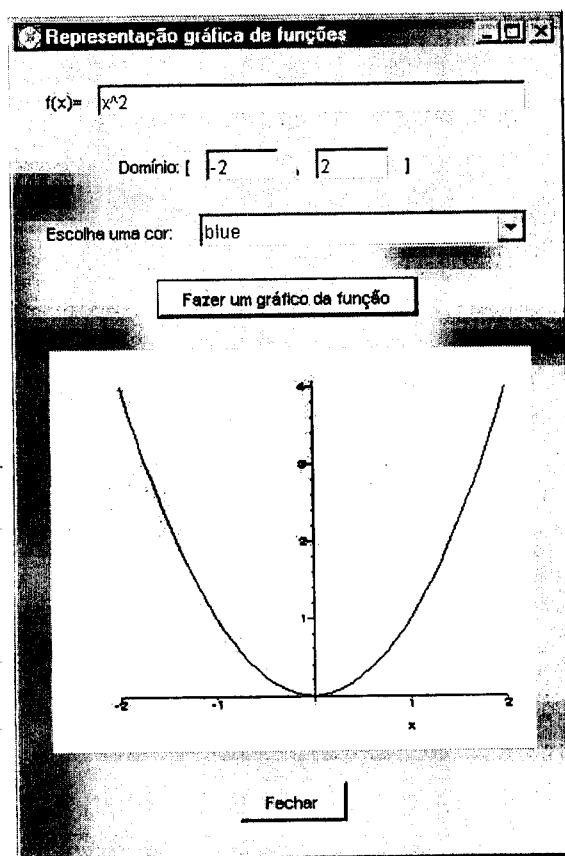


Figura 1.1: Gráfico simples

¹Maplets - nome atribuído pela empresa *MapleSoft* e que aqui usaremos para identificar esta tecnologia de *interfaces* gráficas.

É conveniente referir que na versão *on-line* deste trabalho, o utilizador pode executar qualquer Maplet pressionando a figura que lhe faz referência. Além disso, nos capítulos seguintes, pressionando a própria legenda pode-se aceder ao código Maple produzido para obter a Maplet.

Neste exemplo, o utilizador ou aluno pode visualizar o gráfico de uma função, introduzindo apenas na caixa de texto a expressão geral da função, com a notação usual existente nas calculadoras científicas, e clicando no botão 'Esboçar gráfico'. Se não fosse esta tecnologia das Maplets, no programa Maple, o utilizador teria de escrever um comando, respeitando a linguagem do Maple, para visualizar o gráfico. Como este, existem variados exemplos que podem ser utilizados pelos professores nas aulas, tirando partido de toda a potencialidade, ao nível da resolução de problemas nos vários campos da matemática, que o Maple proporciona.

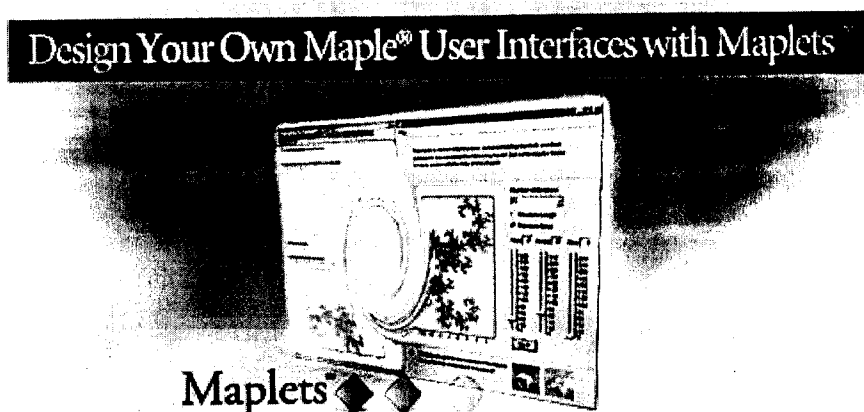
Neste trabalho, serão descritas, construídas e utilizadas várias Maplets que se integram nos programas do ensino secundário e no primeiro ano do ensino superior.

1.3 Acerca deste trabalho

Os alunos têm actualmente à sua disposição calculadoras que os ajudam a resolver problemas matemáticos ao nível do cálculo numérico, mas existe uma grande lacuna ao nível algébrico. O Maple é uma das ferramentas mais poderosas a nível algébrico, numérico e gráfico, mas como já referimos, para que pudesse ser utilizado por alunos e professores, era necessário que houvesse uma formação ao nível da linguagem que o Maple requer para a execução dos seus comandos. Com as Maplets, podem ser utilizadas todas as potencialidades do Maple sem que seja necessário os alunos/utilizadores saberem a linguagem própria do Maple, pois através de um *interface* interactivo que lhes é familiar (janelas com caixas de texto, botões, etc.) podem obter as respostas aos seus problemas sem precisarem sequer de correr o programa Maple. O recurso destas janelas interactivas ao

programa Maple é feito por um processo invisível para o utilizador final.

Este trabalho destina-se, principalmente, aos professores que queiram aproveitar as potencialidades do Maple no processo ensino /aprendizagem. Os professores podem utilizar e modificar as Maplets aqui construídas, bem como o vasto reportório existente na internet (por exemplo, em [22]), mas sobretudo, podem construir as suas próprias Maplets adequadas ao que pretendem fazer com elas. Uma Maplet pode ser construída de raiz, ou seja, pode ser totalmente idealizada e construída desde o início, ou pode-se aproveitar alguma Maplet que já esteja construída e adicionar ou modificar os elementos de forma a torná-la uma Maplet à imagem que cada um pretende.



Assim, no Capítulo 2, é feita uma abordagem ao funcionamento do Maple, isto é, são dadas as noções básicas necessárias para se conhecer a linguagem Maple, para que se possa construir as Maplets. De facto, para ser um autor de Maplets, é necessário que se conheça um pouco da linguagem Maple. Neste capítulo, fica-se a saber como é que se inicializa uma sessão de Maple, como é que se pode recorrer à ajuda interactiva que o Maple dispõe e, acima de tudo, compreender a linguagem Maple, que é abordada através

de exemplos práticos.

No Capítulo 3 são apresentadas grande parte das Maplets construídas neste trabalho, aí veremos na perspectiva do utilizador, o funcionamento de Maplets desde as mais simples às mais apetrechadas, descrevendo-se as janelas e o modo como se utilizam. A abordagem feita terá sempre como objectivo explicitar as funcionalidades que as Maplets podem proporcionar ao nível do ensino, mais precisamente do ensino da Matemática. Por exemplo, ao nível do ensino básico, é mencionada neste capítulo uma Maplet que permite estudar algumas características de um número inteiro, mais especificamente, saber se ele é ou não primo, quais os seus divisores e a sua decomposição em factores primos (fig. 1.2)².

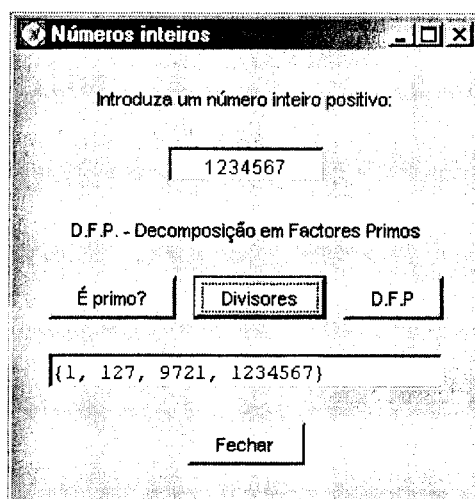


Figura 1.2: Números inteiros

Nesta Maplet, o utilizador só tem de introduzir um número inteiro positivo na primeira caixa de texto que aparece na janela. Depois, ao pressionar numa das três opções disponíveis através dos três botões alinhados no meio da janela, vai visualizar o resultado na última caixa de texto da janela. Quando o utilizador quiser fechar a janela, só tem de pressionar no botão 'Fechar'.

²Na versão *on-line*, se pressionar na imagem, poderá executar esta Maplet.

Para o ensino secundário, exibiremos entre outras uma Maplet que permita obter a função derivada de uma função a uma variável (fig. 1.3).

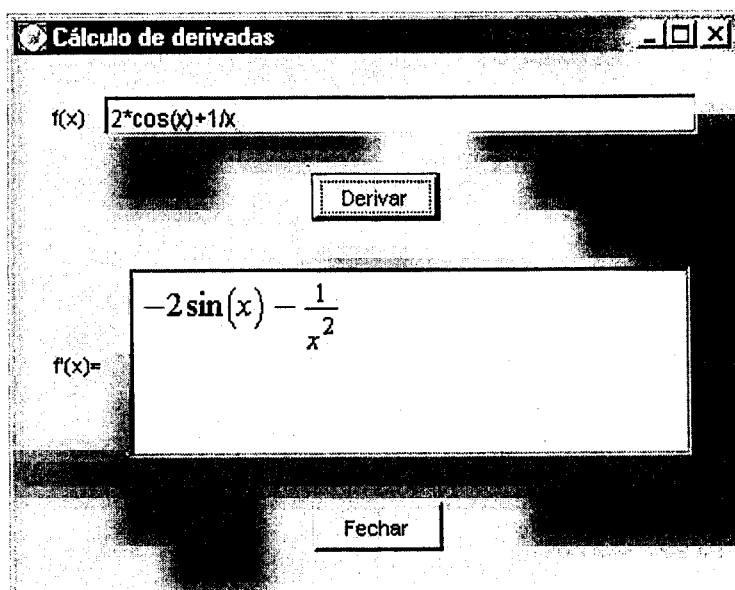


Figura 1.3: Cálculo de derivadas

O utilizador fica com um formulário interativo, onde só tem de introduzir uma função de variável x e pressionar no botão 'Derivar' para obter a função derivada no formato usual da linguagem matemática. Ao pressionar no botão 'Fechar', a janela fecha-se.

Ao nível do ensino superior, na iniciação à integração pode, por exemplo, ser explorada uma Maplet (fig. 1.4), onde o utilizador deve introduzir a expressão que deseja integrar, a variável de integração e os limites de integração, no caso de ser um integral definido. Depois, só tem de pressionar no botão 'Integrar' e aparecerá o resultado que o Maple forneceria utilizando os comandos de integração.

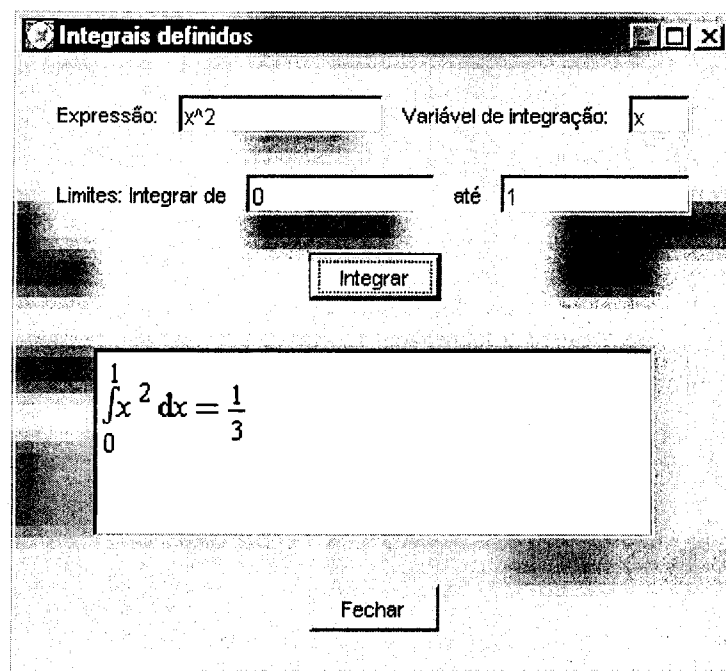


Figura 1.4: Cálculo de integrais definidos

Esta tecnologia das Maplets é tão versátil, que permite realizar jogos de lógica e raciocínio, como por exemplo, o conhecido "Mastermind" (fig. 1.5) .

Neste jogo, o principal objectivo para o utilizador, é adivinhar qual a sequência de 4 cores que o computador gera aleatoriamente.

Em todo o caso, neste capítulo, não se faz referência ao código subjacente a estas Maplets, apenas são descritas na perspectiva do utilizador final, nomeadamente o que é permitido fazer e o modo de as utilizar.

Só no Capítulo 4 se explicará como se constroem as Maplets, definindo-se todos os elementos que lhe estão subjacentes. Neste capítulo é possível ver como é que estes elementos geradores podem ser utilizados, apresentando-se casos simples exemplificativos da função de cada um.

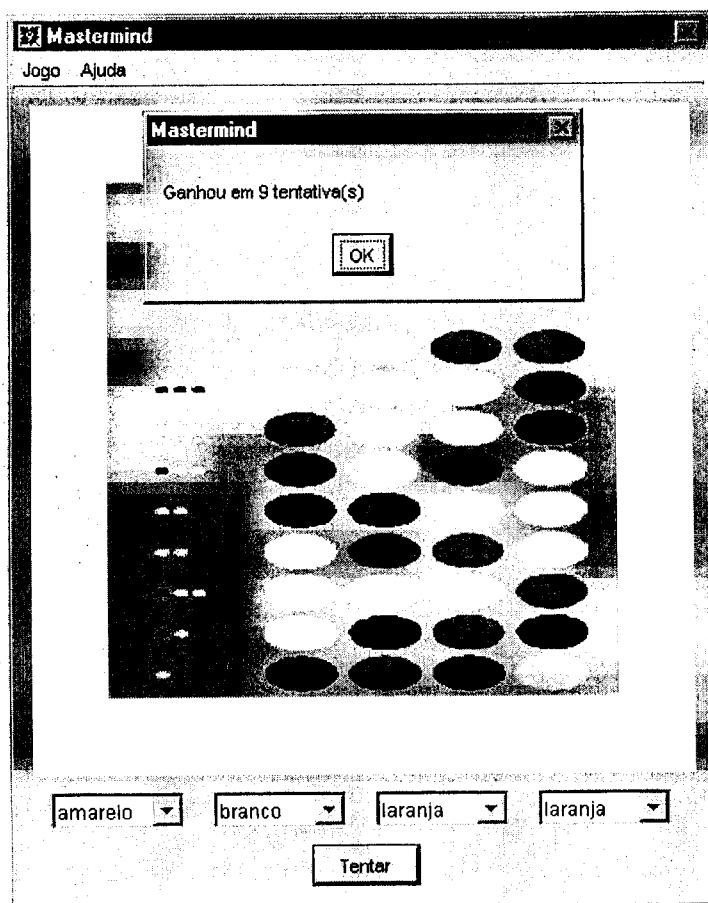


Figura 1.5: Mastermind

Para se perceber como pode ser construída uma Maplet mais complexa com várias funcionalidades, fez-se no Capítulo 5, uma explicação detalhada da construção de uma Maplet para o estudo gráfico de funções de uma variável real. Aí, começou-se por construir uma Maplet básica onde fosse possível inserir a expressão de uma função e os valores limite do domínio e do eixo das ordenadas, procedendo ao esboço gráfico dos dados introduzidos. Depois foi-se melhorando a Maplet, até ser possível modificar as opções do gráfico, fazer o *zoom* desejado, poder movimentar um ponto sobre ele e, até fornecer um estudo da função quanto aos seus zeros, monotonia e concavidades (fig. 1.6).

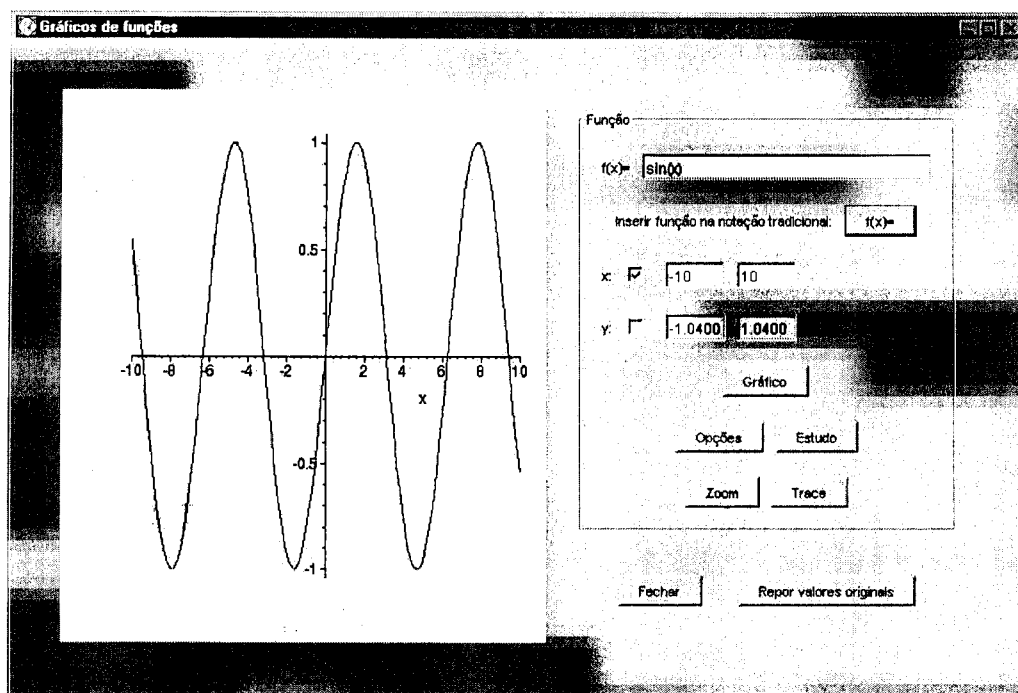


Figura 1.6: Última versão da Maplet 'Gráficos de funções'

Finalmente, no Capítulo 6, são apresentados alguns exemplos de Maplets construídas neste trabalho que podem ser úteis tanto num curso de Cálculo Infinitesimal de funções a uma variável, como num ano introdutório de Sistemas Dinâmicos que trate de aplicações unidimensionais.

Ao nível da utilidade das Maplets no ensino, temos neste capítulo, ao nível do ensino superior, os exemplos de Maplets que poderão ser úteis tanto para os alunos como para os docentes, uma vez que estas Maplets tratam problemas de cálculo de limites, derivação e integração passo-a-passo, ou seja, o utilizador pode resolver um problema visualizando todos os passos intermédios até à solução final. O utilizador/aluno pode, através de várias tentativas aplicar as regras que mais se ajustem à resolução do problema, podendo obter sugestões sempre que necessário. O utilizador/docente tem nestas Maplets uma excelente ferramenta de criação de exercícios para propor aos alunos e sobretudo uma boa

ferramenta para a correcção desses exercícios.

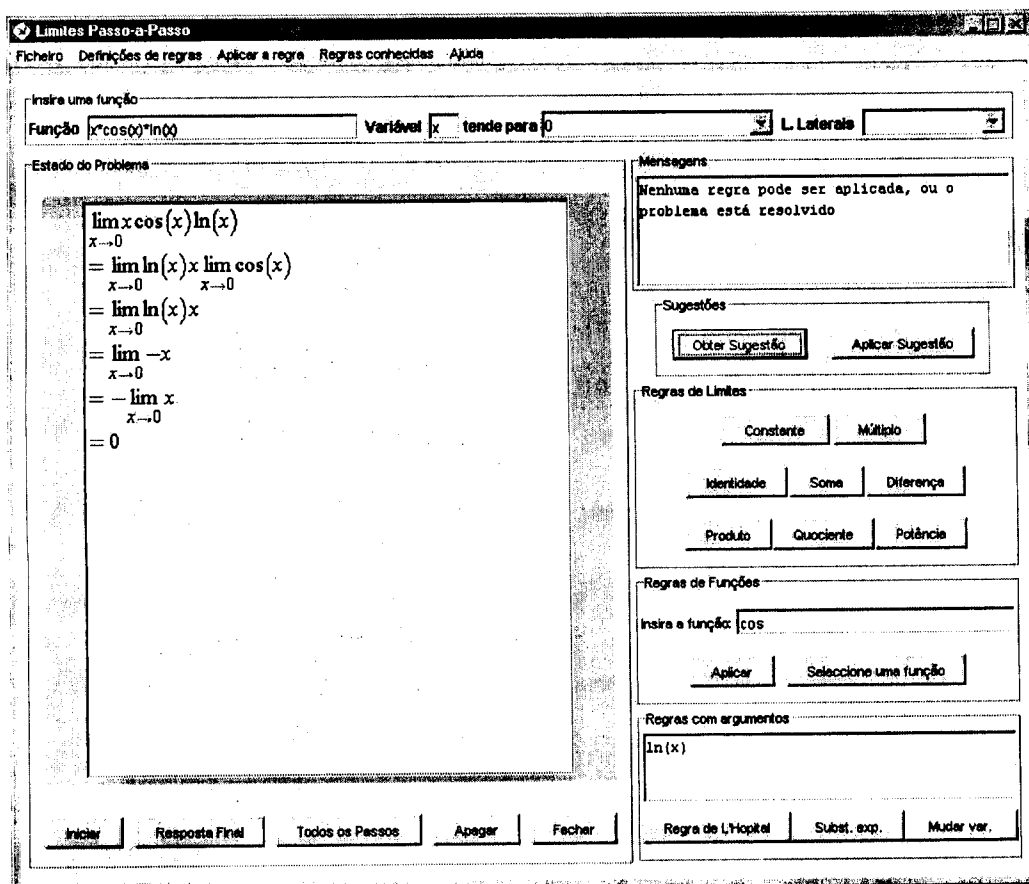


Figura 1.7: Cálculo de Limites passo a passo

Na Maplet que resolve problemas de cálculo de limites (fig. 1.7), o utilizador deverá inserir a expressão da qual deseja saber o limite, a respectiva variável, o valor para o qual tende a variável e, se for o caso de querer calcular limites laterais, indicar se é um limite à esquerda ou à direita. Depois é só pressionar no botão 'Iniciar' e começar a resolução do problema. O utilizador deve aplicar as regras que achar convenientes usar e, se não souber que regra aplicar, pode sempre pedir uma sugestão através do botão 'Obter Sugestão'.

Na Maplet que resolve problemas de derivação (fig. 1.8), o utilizador só tem de inserir a

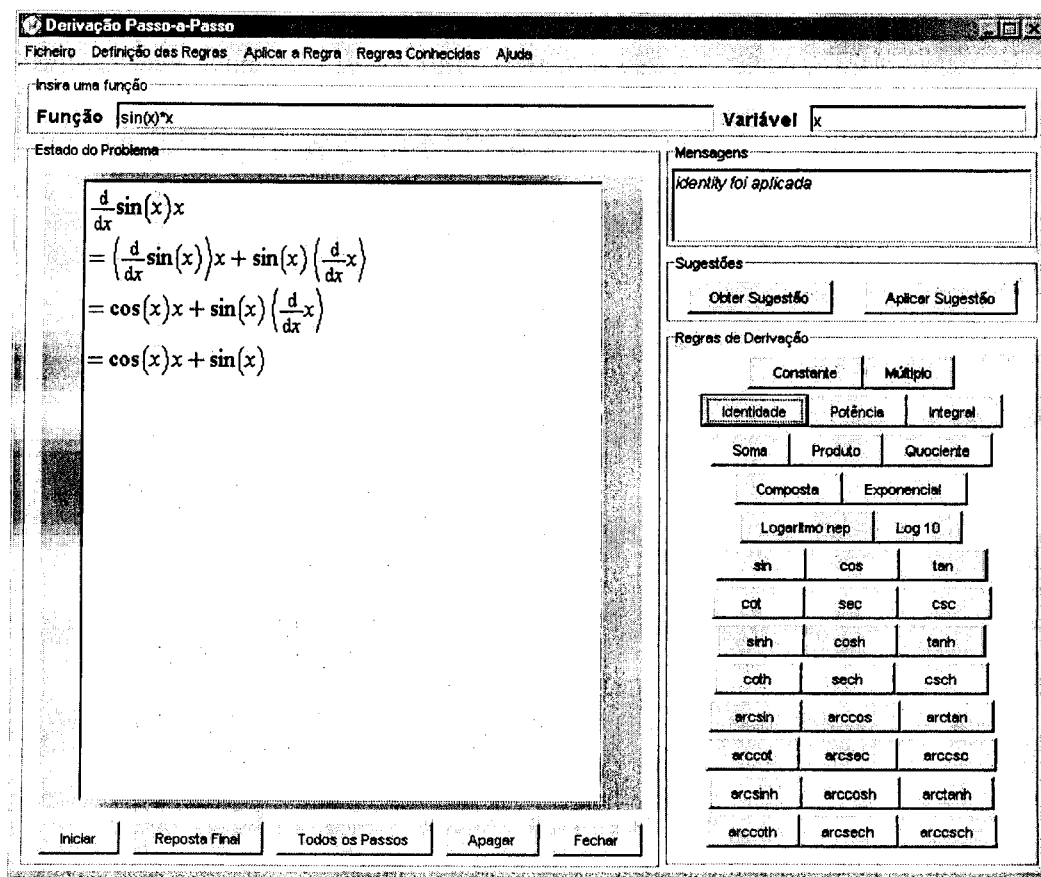


Figura 1.8: Derivação passo a passo

expressão que deseja derivar e a variável de derivação. Depois, pressionando em 'Iniciar', pode começar a resolver o problema.

Na Maplet que resolve problemas de integração (fig. 1.9), o utilizador tem de inserir a expressão que deseja integrar, a variável de integração e, no caso de pretender um integral definido, precisa de inserir também os valores limites da integração, ou seja, se o utilizador quiser resolver o seguinte integral:

$$\int_a^b f(x)dx$$

deve inserir a expressão $f(x)$, a variável x e os valores a e b . Se estes dois últimos valores não forem introduzidos, esta Maplet resolve o problema como um caso de integrais

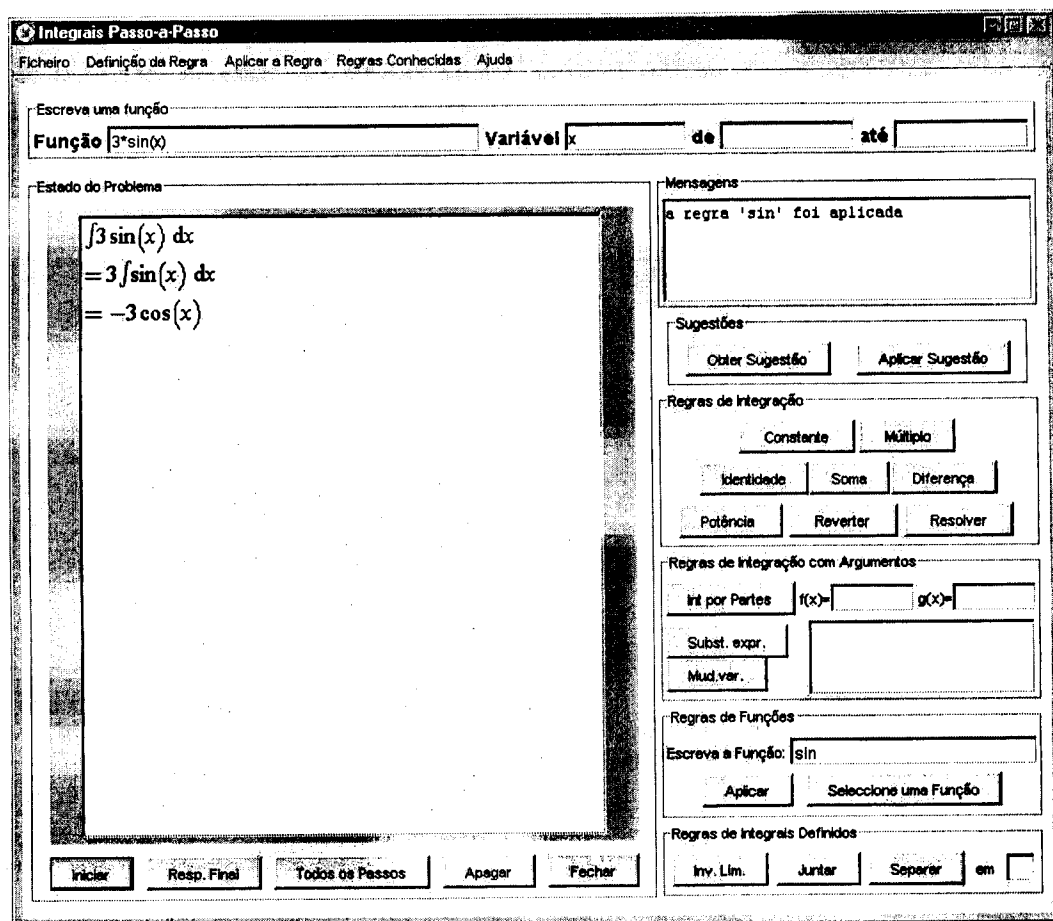


Figura 1.9: Integração passo a passo

indefinidos.

Com o aparecimento do Maple 9, em 2003, foi possível criar Maplets mais interactivas com gráficos tridimensionais, pelo que foi importante adicionar a este capítulo uma secção onde são mostradas Maplets com gráficos de funções com mais de uma variável. Um dos exemplos que se pode ver neste capítulo, é uma Maplet construída através do Maple 9, que permite visualizar e interagir com gráficos tridimensionais do tipo $z = f(x, y)$ (fig. 1.10). A grande particularidade desta Maplet, é poder "mexer" no gráfico girando-o para a posição que se pretender apenas com o auxílio do rato. Isto só é possível com o recurso

do Maple 9. No Maple 8, não era permitida essa interactividade em Maplets.

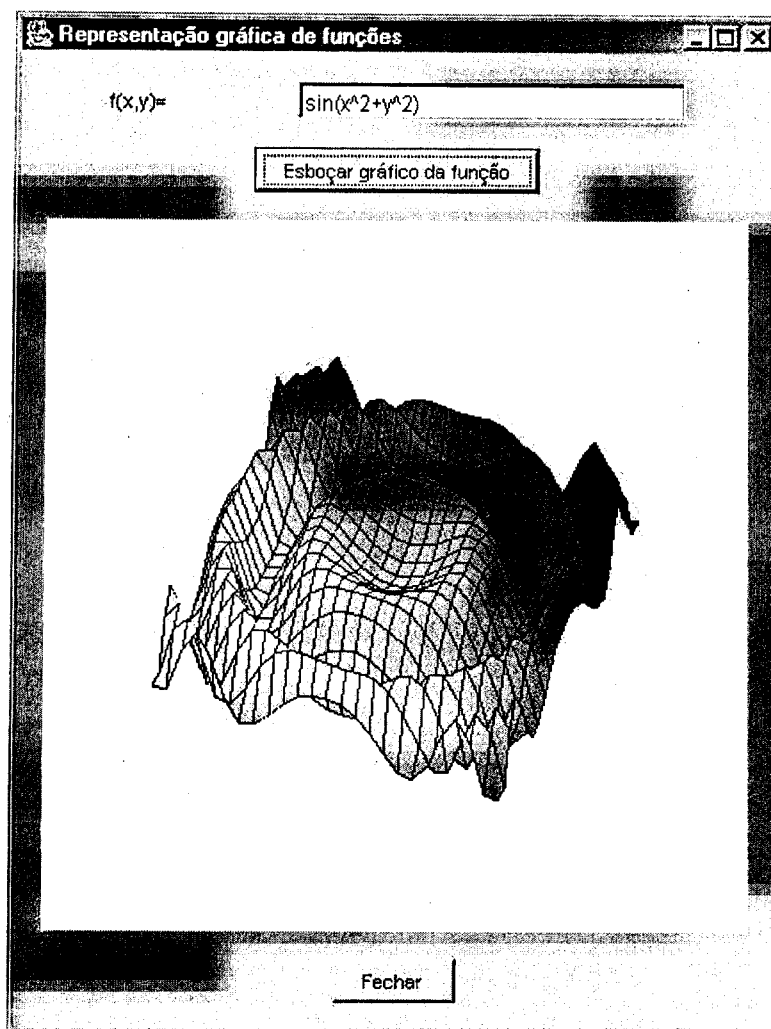


Figura 1.10: Representação de uma função tridimensional do tipo $z = f(x, y)$

Outros exemplos de gráficos tridimensionais, são as superfícies definidas por equações paramétricas. Neste capítulo também é apresentada uma Maplet com este tipo de gráficos (fig. 1.11).

O utilizador precisa de inserir o domínio da superfície e as expressões de $f(u, v)$, $g(u, v)$

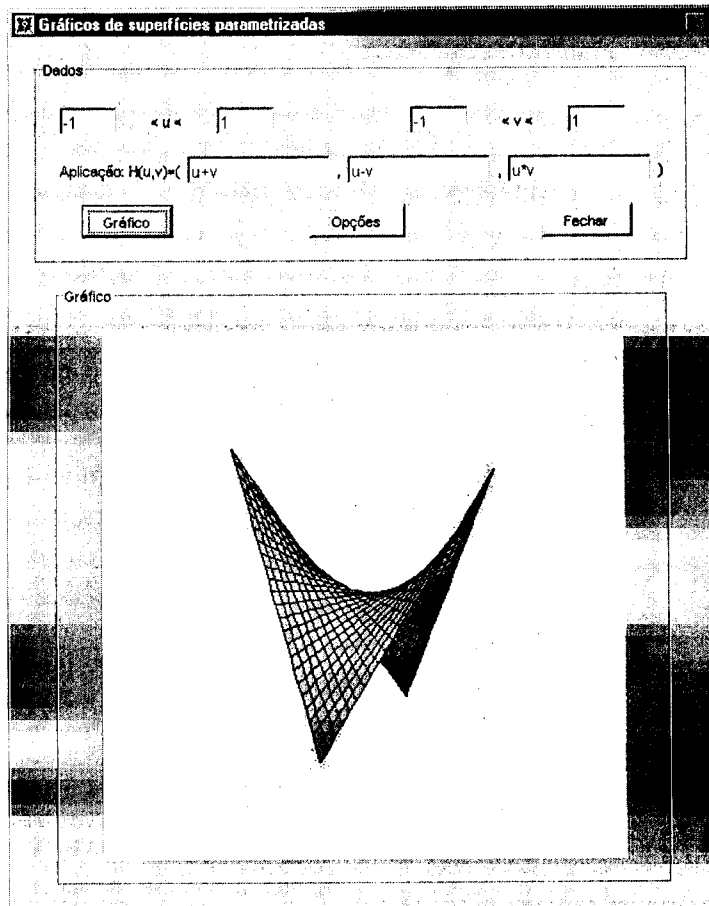


Figura 1.11: Gráfico tridimensional definido por equações paramétricas

e $h(u, v)$ e pressionar o botão 'Gráfico', para que a superfície definida pelas equações

$$\begin{cases} x = f(u, v) \\ y = g(u, v) \\ z = h(u, v) \end{cases}$$

seja mostrada. Depois, o utilizador pode alterar algumas opções de visualização do gráfico.

Baseados ainda nas superfícies parametrizadas, podemos ver também o exemplo de uma Maplet usada para a visualização de funções do plano no plano (fig. 1.12). Dada uma região do plano, como por exemplo, um rectângulo, através do seu preenchimento com

faixas de diferentes cores, podemos ver nesta Maplet, o esboço gráfico da imagem por F dessa região e observar como é que essas faixas coloridas são transformadas.

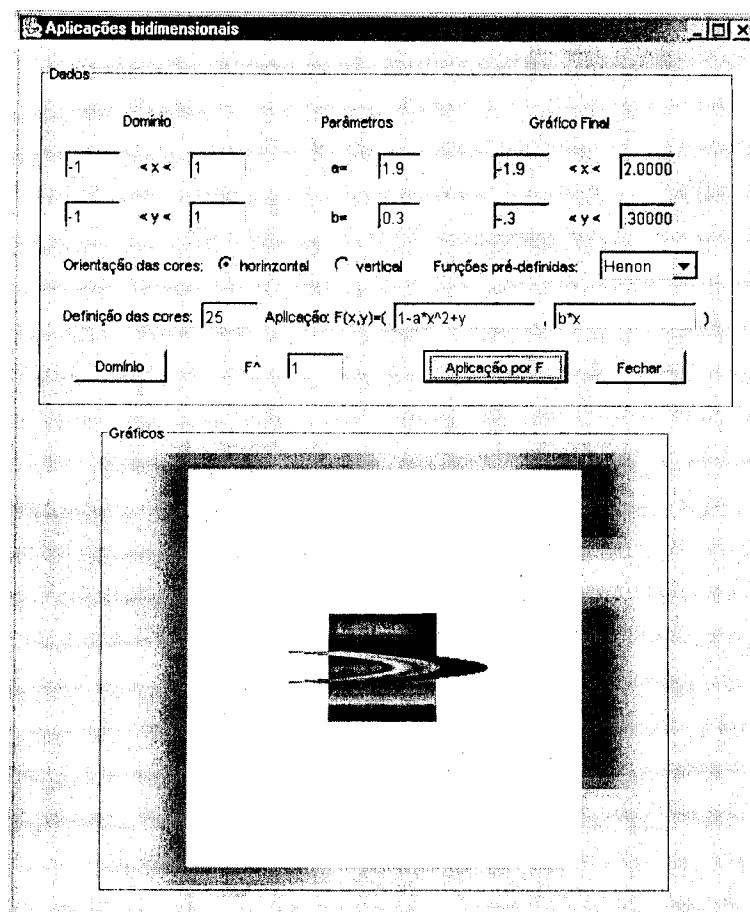


Figura 1.12: Aplicações bidimensionais

Relativamente à área dos Sistemas Dinâmicos, podemos ver neste capítulo, por exemplo, uma Maplet que serve para visualizar os pontos fixos de uma função f , os pontos x satisfazendo $f(x) = x$ e das compostas $f^{(n)}$ (fig. 1.13).

Nesta Maplet, o utilizador pode também definir funções dependendo de um parâmetro r e depois, não só visualizar o gráfico, como também assistir a uma animação do mesmo.

De facto, pode ver a variação da representação gráfica da função f^n quando o parâmetro r varia num certo intervalo definido. A visualização dos pontos fixos é possível, quando o gráfico da função intersecta a recta $y = x$ que é também representada nesta janela com outra cor. Podemos, portanto, dizer que há já uma maior interactividade entre o utilizador e esta Maplet.

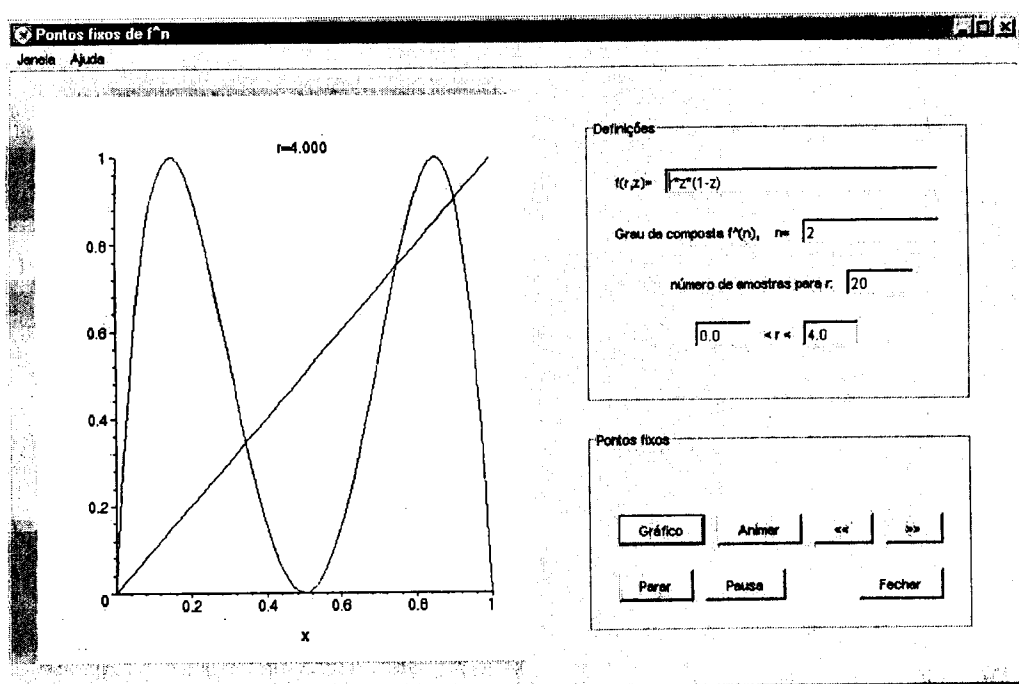


Figura 1.13: Pontos fixos de f^n

Também nesta secção, é apresentada a Maplet relativa à análise gráfica da órbita de um ponto para funções reais de variável real (fig. 1.14).

Esta Maplet pode ser usada para ilustrar pedaços $(x_0, x_1, \dots, x_n)$ da órbita³ de x_0 pela aplicação f . De facto, recorrendo à identificação $a \mapsto (a, a)$, podemos analisar a dinâmica de $a \mapsto f(a)$ pela dinâmica de $(a, a) \mapsto (f(a), f(a))$. Se partirmos de um ponto (a, a) ,

³por órbita de x_0 , designa-se a sequência $(x_n)_{n \in \mathbb{N}}$ em que $x_n = f^n(x_0)$

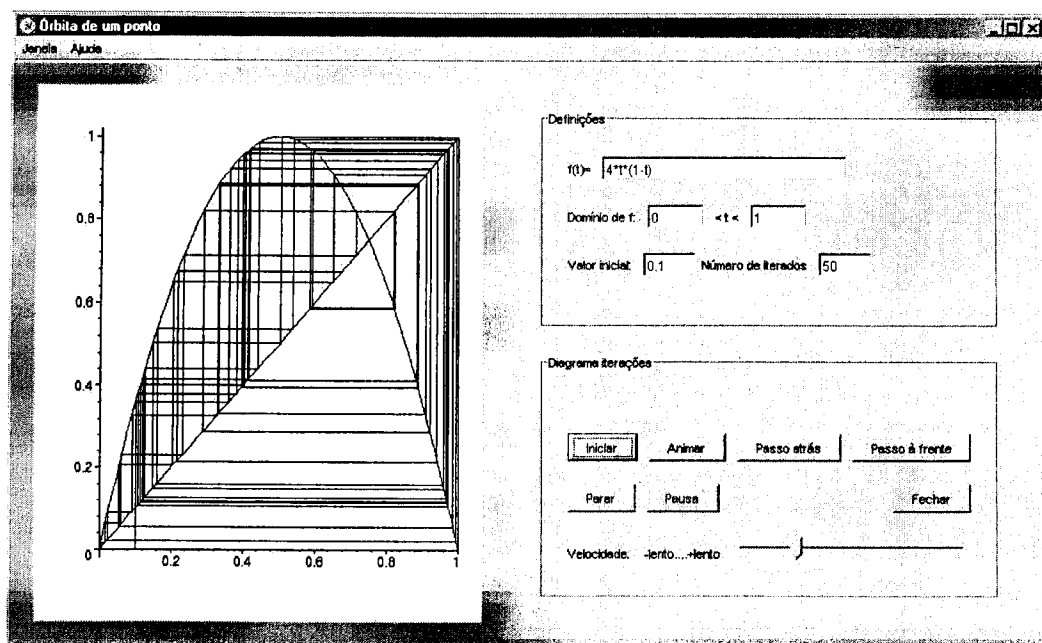


Figura 1.14: Órbita de um ponto

traçarmos a recta vertical que passa por (a, a) , essa recta intersecta o gráfico de f no ponto $(a, f(a))$. De seguida, traça-se uma recta horizontal que passa por $(a, f(a))$. Esta recta intersecta a diagonal no ponto $(f(a), f(a))$. Temos assim o modo de ilustrar a passagem de (a, a) para $(f(a), f(a))$ e consequentemente, projectando no eixo do x , temos a passagem de a para $f(a)$. Repetindo o processo, a recta vertical que passa no ponto $(f(a), f(a))$ vai intersectar o gráfico de f no ponto $(f(a), f^2(a))$. A recta horizontal que passa por $(f(a), f(a))$ intersecta a diagonal no ponto $(f^2(a), f^2(a))$ e assim sucessivamente, obtendo o conjunto de pontos $\{a, f(a), f^2(a), \dots\}$. Esta Maplet tem a vantagem de permitir ver o que acontece em cada iteração e ver a progressão das iterações de uma forma sequencial rápida ou mais lenta.

Na sequência da Maplet anterior, é também apresentada uma Maplet que faz um estudo estatístico da órbita de x_0 por uma aplicação $f_r : [a, b] \rightarrow \mathbb{R}$ (fig. 1.15). Para $n, N \in \mathbb{N}$, considerando $I_k = [a + k \frac{b-a}{N}, a + (k+1) \frac{b-a}{N}]$, a Maplet fornece em forma de histograma o

número de iterados x_i que estão em I_k , para $k = 0, \dots, N - 1$. De facto, a Maplet esboça um gráfico de barras em que cada uma tem por base I_k e área igual a $\#\{x_i : x_i \in I_k\}$.

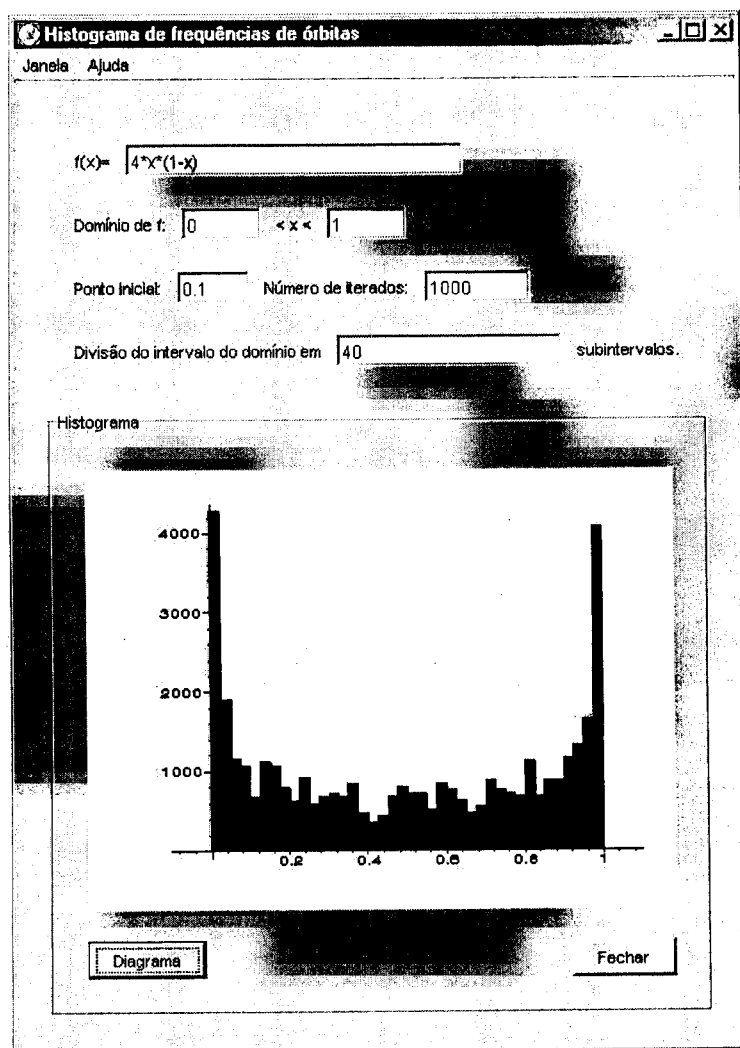


Figura 1.15: Histograma de frequências

Assim, esta Maplet pode funcionar como uma boa oportunidade para o professor do ensino secundário relacionar conteúdos programáticos de unidades distintas (Funções e Estatística).

Por último, apresentamos a Maplet da figura 1.16, que permite, para uma função real $f_r(x) = f(r, x)$ que depende de um parâmetro r e um ponto inicial $x_0 \in \mathbb{R}$, esboçar os pontos $(r, f_r^k(x_0))$ para $n < k \leq N$, $n, N \in \mathbb{N}$ e para um intervalo $[r_0, r_1]$ de amostragem do parâmetro r . Esta Maplet fornece indicações acerca da variação do comportamento assintótico de $f_r^k(x_0)$ quando n é elevado. No caso de $f(r, x) = rx(1 - x)$, obtemos o "clássico" diagrama de bifurcações exibido na figura 1.16. (Ver secção 1.17 em [6])

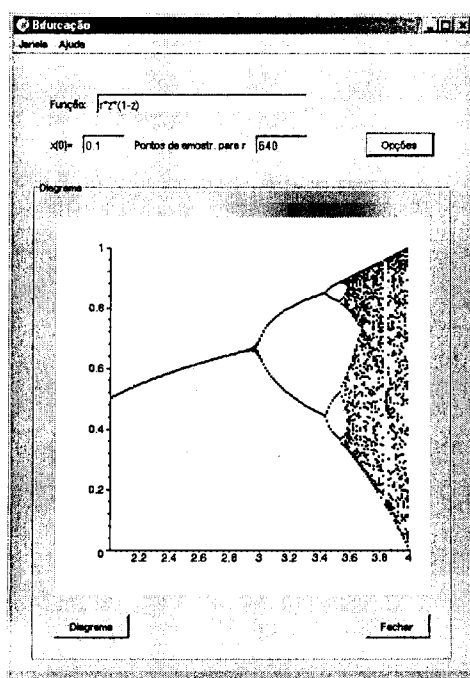


Figura 1.16: Bifurcação

Capítulo 2

Noções Básicas do Maple

Começamos por uma breve introdução ao programa Maple, para nos capítulos posteriores analisarmos as características da linguagem própria do Maple, que permitam a construção dos objectos fulcrais deste trabalho: As Maplets.

2.1 Introdução

O Maple é uma linguagem de computação que possui quatro aspectos gerais que são:

- computação algébrica
- cálculo numérico
- manipulação gráfica
- programação

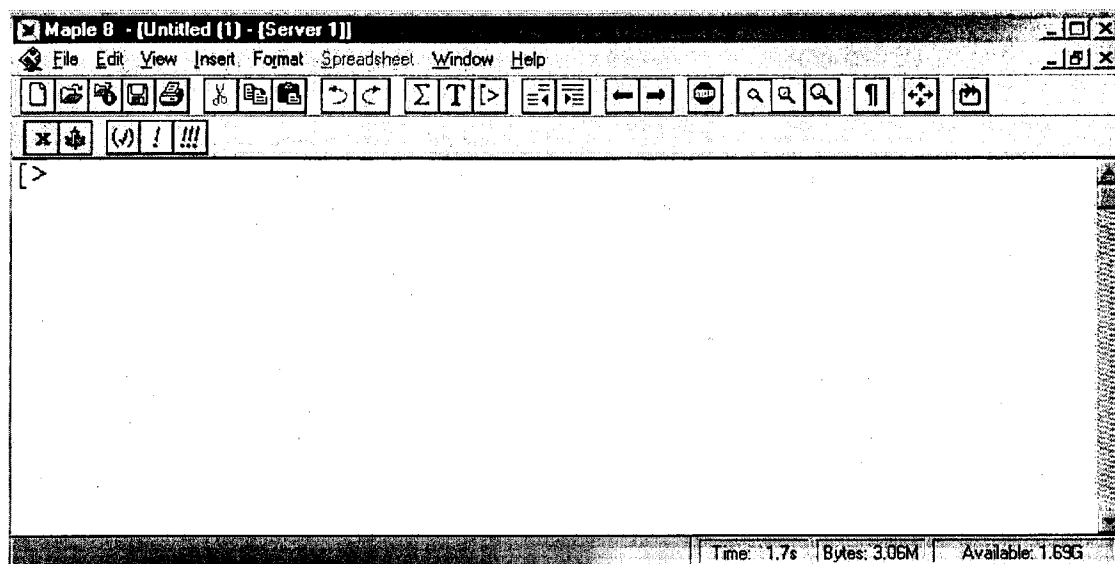
Todos estes aspectos estão integrados e formam um corpo único. Por exemplo, a partir de um resultado algébrico, pode ser feita imediatamente uma análise numérica ou gráfica. Um dos factos que tornam o Maple um programa bastante elogiado e reconhecido no

mundo inteiro é as facilidades que ele dá ao utilizador que tem por objectivo lidar com computação algébrica.

A grande quantidade de funções disponíveis para os mais diversos fins na área de Matemática, Estatística, etc., poupam muitas linhas de código (e consequentemente, muito tempo e erro) quando comparamos o Maple a outras linguagens (C, C++, Pascal, etc.).

2.2 Iniciar uma sessão de Maple

No sistema Windows, a acessibilidade ao Maple é como para a maior parte dos programas para Windows. Basta fazer um duplo-clique¹ no símbolo do Maple se ele estiver no ambiente de trabalho, através do menu Iniciar ou através do ficheiro executável Maple8.exe. Depois de se abrir o programa, aparece uma janela com menus, alguns botões e a folha de trabalho (*worksheet*) para edição dos comandos Maple.



¹clicar=pressionar

A primeira barra contém os menus File, Edit, View, Insert, Format, Spreadsheet, Window e Help. Por baixo dessa barra está um conjunto de 29 botões. Os usuais botões de um processador de texto:

-  Abre uma nova folha de trabalho.
-  Abre uma worksheet já existente.
-  Abre uma URL específica.
-  Guarda a folha de trabalho activa.
-  Imprime a folha de trabalho activa.
-  Corta a selecção feita.
-  Copia a selecção feita.
-  Cola a selecção feita.
-  Anula a última operação.
-  Anula a última anulação.

Os botões próprios do Maple:

-  Escreve texto em linguagem matemática.
-  Insere texto.
-  Insere uma linha de entrada para comandos Maple, logo a seguir ao cursor.
-  Remove da sub-secção o que estiver seleccionado.
-  Cria uma sub-secção na folha de trabalho.

-  Volta ao anterior hyperlink.
-  Avança para o próximo hyperlink.
-  Interrompe a computação que está a ser feita.
-  Zoom a 100%.
-  Zoom a 150%.
-  Zoom a 200%.
-  Redimensiona a janela.
-  Apaga o que está em memória virtual (restart).

Estes botões estão abaixo da primeira linha de botões:

-  Troca entre notação matemática e notação Maple.
-  Troca entre texto e comandos executáveis de Maple.
-  Auto-correcção da sintaxe de uma expressão.
-  Executa a expressão corrente.
-  Executa toda a worksheet.

Nesta fase inicial, não nos devemos preocupar demasiado com estes botões.

2.3 Área de trabalho

Na maior parte das referências que abordam o Maple, a designação mais usual para área ou folha de trabalho é *worksheet*. É, portanto, natural que ao longo deste trabalho seja usada a palavra *worksheet* para designar a área de trabalho onde é efectuada uma sessão de Maple. A área de trabalho é onde são escritos e executados os comandos do Maple. Ela é o principal meio para gravar e ler os trabalhos desenvolvidos no Maple.

Na área de trabalho, um utilizador pode desenvolver a solução de um problema usando o Maple, fazer comentários, colocar gráficos e gravar este conjunto de coisas num arquivo para ser lido e eventualmente modificado posteriormente. A *worksheet* pode ser imprimida seleccionando-se a opção "print" ou pode ser convertida num ficheiro *Latex*.

A área de trabalho é um caderno virtual de anotações de cálculos. A vantagem do caderno virtual é que qualquer coisa já escrita pode ser modificada sem necessidade de fazer outras alterações. O resto do trabalho ajusta-se automaticamente às mudanças. A *worksheet* funciona de maneira satisfatória como processador de texto com a vantagem de a mesma folha conter resoluções matemáticas algébricas, numéricas, gráficas ou mesmo animações interactivas. O '>restart:' pode encabeçar o trabalho para apagar da memória valores atribuídos em *worksheets* anteriores. Depois de gravar a *worksheet*, se o programador sair do Maple, no momento em que ela é lida novamente, os resultados que aparecem no monitor não estão na memória activa do Maple. É necessário processar os comandos novamente para activar os resultados.

A *worksheet* tem quatro tipos de linhas que são:

- as linhas de entrada para execução de comandos;
- as linhas de saída dos comandos executados;
- os comentários de erro;
- as linhas de texto.

As linhas de entrada para execução de comandos são por defeito de cor vermelha, precedidas do símbolo '>' e devem terminar com os caracteres ':' ou ';'. Se os comandos de entrada terminarem com ';', o Maple executa e mostra na linha seguinte o resultado dessa execução. Se terminar com ':', o Maple executa, mas não mostra o resultado dessa execução na *worksheet*. Nestas linhas de entrada, é usada a conhecida linguagem das

calculadoras programáveis. Por exemplo, para escrever a expressão $\frac{x^2-2\sqrt{x}}{3}$ em linguagem maple, temos de escrever:

```
>(x^2-2*sqrt(x))/3;
```

As linhas de saída dos comandos executados aparecem centradas horizontalmente logo a seguir a uma linha de entrada terminada com ';'. Estas linhas podem ser expressões escritas na notação usual da matemática (por defeito são de cor azul) ou gráficos.

```
> (x^2-2*sqrt(x))/3;
```

$$\frac{x^2}{3} - \frac{2\sqrt{x}}{3}$$

Os comentários de erro aparecem por defeito num tamanho de letra mais pequeno e na cor lilás. Estas linhas só aparecem se na linha de entrada dos comandos existir algum erro a nível sintáctico, estrutural ou matemático, que se traduz numa instrução não decifrável pelo Maple.

```
> 2+*2;
```

```
Error, '*' unexpected
```

As linhas de texto, são inseridas na *worksheet*, pressionado no botão que tem a letra T. O texto aparece sem ser precedido do sinal '>' e por defeito tem cor preta. Estas linhas são usadas para fazer comentários sobre o que se está a fazer ou simplesmente para processar um documento (uma aula, um exame, etc.). É possível criar secções e sub-secções. As letras podem ter diversos tamanhos e estilos, podem ser em itálico, negrito, etc.. É possível criar *hyperlinks* que conectam diversas *woksheets*.

A *worksheet* é gravada no formato *.mws* e pode ser exportada em formatos para utilizar na *web* (HTML) ou para serem reconhecidas por editores de texto (RTF, Latex, etc.).

2.4 Ajuda interactiva

O Maple inclui um sistema de ajuda interactivo conhecido pela sua designação inglesa *help on line*. Uma das formas mais simples de obter ajuda sobre uma função ou outro assunto pertinente, é clicar no menu *Help* do Maple e seleccionar a opção 'Topic Search...'. Depois só tem de escrever o assunto ou a package sobre a qual quer obter ajuda.

Também podemos aceder ao help, a partir da *worksheet*. Basta, para isso, seleccionar o comando e logo a seguir premir a tecla 'F1' do teclado do computador ou preceder a função ou o assunto com um sinal de interrogação. Por exemplo:

```
>?maple
```

A página de ajuda contém várias partes incluindo uma descrição detalhada sobre a função, e no final, contém palavras-chave relacionadas com o assunto. A partir dessas palavras-chave, pode-se navegar pelo sistema de ajuda até que a informação desejada seja encontrada.

2.5 A Linguagem do Maple

2.5.1 Números

Normalmente, o Maple trabalha com os números inteiros de maneira exacta, isto é, não faz nenhuma aproximação:

```
>(32*5+8/13)^2;
```

$$\frac{4359744}{169}$$

Podemos ver que o resultado é um número racional. Para poder obter uma aproximação decimal, devemos usar o comando *evalf* (*evaluate in floating point*):

```
> evalf(%);
```

25797.30178

O símbolo % refere-se ao resultado anterior. Dois símbolos de % seguidos (%%) referem-se ao resultado que está antes do resultado anterior. Por exemplo:

```
>2+3;
```

5

```
>4+%;
```

9

```
>%%;
```

5

É preciso ter em atenção que quando usamos %, os valores não são necessariamente os que estão imediatamente acima, pois na *worksheet* podemos processar os comandos numa ordem diferente da que aparece no écran.

As operações básicas no Maple são as usuais: a adição (+), a subtracção (-), a multiplicação (*), a divisão (/) e a exponenciação (^ ou **). O Maple tem também as funções matemáticas básicas (e muitas mais) que estão disponíveis em qualquer calculadora científica (valor absoluto - **abs(x)**, raiz quadrada - **sqrt(x)**, factorial - **n!**, funções trigonométricas, logaritmo natural - **log(x)**, logaritmo neperiano - **ln(x)**, exponencial - **exp(x)**, etc.). Para ver a lista completa de funções que o Maple tem, basta executar **>?index[functions]** na *worksheet* ou consultar o menu *Help*.

Por exemplo, para calcular a

$$\tan\left(\frac{\pi}{6}\right)$$

escrevemos:

```
>tan(Pi/6);
```

$$\frac{\sqrt{3}}{3}$$

```
> evalf(%);
```

0.5773502693

Reparemos que o **evalf** encontrou um valor aproximado de $\tan\left(\frac{\pi}{6}\right)$ com 10 casas decimais. Tal acontece, porque, por defeito, o Maple dá o resultado com 10 algarismos significativos. No entanto, podemos aproximar o resultado com quantos algarismos quisermos². Existem duas formas diferentes de alterar o número de dígitos que o Maple dá por defeito:

```
> Pi;
```

π

```
> evalf(%);
```

3.141592654

```
> evalf(Pi,5);
```

3.1416

```
> evalf(Pi,30);
```

3.14159265358979323846264338328

```
> Digits:=20;
```

Digits:=20

```
> evalf(Pi);
```

3.1415926535897932385

De facto, se no comando **evalf** indicarmos o número de algarismos significativos, obtemos a aproximação desejada (**evalf(Pi,5)**; deu um número aproximado de π com 5 algarismos significativos). O outro método é usar a variável global **Digits**, que por defeito vale 10

²com as limitações naturais do hardware (equipamento) que se usa

(Depois de atribuir a esta variável o valor 20, o `evalf(Pi)`; deu um valor aproximado de π com 20 algarismos significativos).

O Maple é capaz de fazer ao contrário, isto é, transformar um número decimal numa fracção irredutível. Por exemplo:

```
>2-.5;
```

1.5

```
> convert(%,rational);
```

$\frac{3}{2}$

A função `convert` é usada para converter expressões de um tipo para outro. Por exemplo, podíamos substituir o comando `evalf(Pi)` por:

```
>convert(Pi,float);
```

3.141592654

Este tipo de expressão (`float`), refere-se a todos os valores numéricos aproximados com casas decimais.

No Maple podemos, também, trabalhar com outras bases (base binária, hexadecimal, etc.).

```
>convert(234,binary);
```

11101010

```
>convert(1234,hex);
```

4D2

Para além destas há muitas outras conversões que podem ser feitas no Maple. Para consultar todos os tipos possíveis de conversão, basta escrever `?convert` na *worksheet* ou consultar no menu help '*convert*'.

O Maple tem muitos comandos que trabalham com números inteiros, como por exemplo, achar o máximo divisor comum entre dois inteiros, saber se um número é ou não primo, etc.

```
>igcd(18,45);
```

9

```
>ilcm(18,45);
```

90

```
>irem(22,4);
```

2

```
>iquo(22,4);
```

5

```
>ifactor(90);
```

(2)(3)²(5)

```
>isprime(1234567891);
```

true

```
>rand();
```

427419669881

O comando '**igcd(18,45);**' calcula o máximo divisor comum entre os números inteiros 18 e 45 ($\text{m.d.c.}(18,45)=9$) e o comando '**ilcm(18,45);**', dá-nos o mínimo múltiplo comum entre 18 e 45 ($\text{m.m.c.}(18,45)=90$). Agora imaginemos que temos um dividendo, um divisor e queremos saber o quociente e o resto. O comando '**irem(22,4);**' dá-nos o resto da divisão

de 22 por 4 e o comando `'iquo(22,4);'` dá-nos o quociente dessa divisão. Assim, sabemos que $22 = 4 \times 5 + 2$. O comando `'ifactor(90);'` decompõe o número 90 em factores primos. O resultado $(2)(3)^2(5)$ significa que $90 = 2 \times 3^2 \times 5$. O comando `'isprime(1234567891);'`, é um teste para saber se um número é ou não primo. Neste caso ficamos a saber que o número 1234567891 é primo, pois o resultado do teste foi `'true'`, se o resultado fosse `'false'`, isto significava que o número era composto. Finalmente uma alusão ao comando `'rand'`, que fornece números aleatórios. Ao escrever `'rand()'`, o Maple dá-nos, por default, um número aleatório de 12 algarismos, mas podemos definir o intervalo de números aleatórios pretendido.

Mas, como já vimos, no Maple podemos trabalhar com números irracionais:

```
> ((1+sqrt(7))/2)^2;
```

$$\left(\frac{1}{2} + \frac{\sqrt{7}}{2}\right)^2$$

```
> expand(%);
```

$$2 + \frac{\sqrt{7}}{2}$$

Repare-se que no início a expressão não foi simplificada. Só depois, através do comando **expand** é que obtivemos uma simplificação (note-se que o **expand** faz a simplificação no sentido de tirar os parêntesis e obter a expressão exacta mais simples). Esta é uma das regras do Maple. As expressões não são simplificadas a menos que o utilizador o peça. Só as simplificações mais elementares é que são feitas, como por exemplo, as operações aritméticas básicas. O comando **simplify** também simplifica expressões, mas no sentido de tentar factorizar, de modo a obter uma expressão simples. Por exemplo, se em vez de **expand** tentássemos simplificar a expressão anterior com o comando **simplify** obteríamos:

```
> ((1+sqrt(7))/2)^2;
```

$$\left(\frac{1}{2} + \frac{\sqrt{7}}{2}\right)^2$$

```
> simplify(%);
```

$$\frac{(1 + \sqrt{7})^2}{4}$$

Para conhecer todas as possibilidades dos comandos **expand**, **simplify** e outros comandos de simplificação(**combine**, **convert**, **normal**, etc.) pode consultar-se no *Help* com a palavra '*simplify*' obtendo-se a ajuda para o comando **simplify** e para os outros comandos de simplificação, através das palavras-chave disponibilizadas no final da página, na secção '*See Also*'.

2.5.2 Atribuição de um nome a uma expressão

Em geral, uma expressão não precisa de ter um nome. Podemos referir-nos a ela usando o símbolo %. Mas na maior parte das situações é necessário fazer referência a uma expressão diversas vezes. Nestes casos, o melhor é atribuir um nome à expressão. Isto pode ser feito com o comando de atribuição ':='. Por exemplo:

```
>dois:=2;
```

dois := 2

```
>dois;
```

2

Atribuímos à variável '*dois*', o valor 2. A partir de agora sempre que for utilizada esta variável, o Maple atribui-lhe o valor 2.

```
>dois+3;
```

5

Se quisermos retirar a atribuição de uma variável, isto é, que o Maple esqueça a atribuição, basta fazer '**>unassign('dois')**:', ou simplesmente:

```
>dois:='dois';
```

dois := *dois*

```
>dois;
```

dois

Vamos analisar agora uma série de atribuições em cadeia:

```
>a:=b;
```

$a := b$

```
>b:=c;
```

$b := c$

```
>c:=7;
```

$c := 7$

Observemos o valor que está agora atribuído à letra 'a':

```
>a;
```

7

Neste caso, a 'a' atribuiu-se 'b', a 'b' atribuiu-se 'c' e a 'c' atribuiu-se o valor 7. Inicialmente foi atribuído a 'a' uma outra variável, que mais à frente tomou o valor 7.

Também podemos atribuir nomes a equações:

```
>eqn:=x^2+3*x-10=0;
```

$eqn := x^2 + 3x - 10 = 0$

```
>solve(eqn);
```

2, -5

À variável 'eqn' foi atribuída uma equação. Com o comando 'solve' resolveu-se a equação 'eqn' tendo-se encontrado duas soluções (2 e -5).

Quando procuramos soluções numéricas, devemos usar 'fsolve' em vez de 'solve'. Por exemplo, para achar os zeros de uma função com expressão $f(x) = x^3 - 3x^2 + 2$, podemos fazer:

```
>solve(x^3-3*x^2+2=0,x);
```

$$1, 1 + \sqrt{3}, 1 - \sqrt{3}$$

e obtemos estas três soluções exactas, mas se fizermos:

```
>fsolve(x^3-3*x^2+2=0,x);
```

$$-.7320508076, 1., 2.732050808$$

obtemos soluções numéricas.

No entanto, como habitualmente, este comando pode conduzir a resultados pouco correctos.

Por exemplo, para achar os zeros numéricos aproximados de $\sin(x)$, no intervalo $[-4,4]$, devemos fazer:

```
>fsolve(sin(x)=0,x=-4..4);
```

$$0$$

no entanto, este resultado não é satisfatório, uma vez que sabemos que no intervalo $[-4,4]$, $\sin(x) = 0$ tem três soluções. De facto, devemos ter cuidado com o uso do **fsolve**, uma vez que, na resolução de equações trigonométricas, o uso do **fsolve**, por defeito, conduz a uma única solução.³ Isto não se passa na resolução de equações polinomiais, pois o Maple contém algoritmos que encontram todas as raízes.⁴

³No entanto, é possível encontrar as outras soluções, usando a opção 'avoid'. Mesmo assim, só é possível ver uma de cada vez. Esta opção pode ser consultada no *Help*, executando na *worksheet* o comando '?fsolve'.

⁴É claro que, como todos os comandos, está sujeito às limitações do sistema, nomeadamente ao nível da falta de memória.

Os nomes aceites pelo Maple incluem caracteres alfa-numéricos e *underscores*. No entanto, não aceita nomes que comecem por um número. Também se devem evitar os *underscores* no início do nome, pois o Maple usa variáveis com nomes a começar por *underscore* a nível interno. Por exemplo, o Maple considera inválido: '1nome' (começa por número) e 'x&y' (o carácter '&' não é alfa-numérico).

É também com o símbolo de atribuição ':=' que devemos definir funções. Para além disso, é necessário também usar uma "seta" (->) logo a seguir à variável da função que queremos definir. Esta seta "constrói-se" com o "sinal menos" (tecla '-' do teclado) logo seguida do "maior que" (tecla '>').

```
>f:=x->x^2+3*x-10;
```

$$f := x \rightarrow x^2 + 3x - 10$$

```
>f(x);
```

$$x^2 + 3x - 10$$

Podemos, assim, calcular a imagem de qualquer objecto do domínio de f .

```
>f(0);
```

$$-10$$

```
>f(2)
```

$$0$$

```
>f(z+y);
```

$$(z + y)^2 + 3(z + y) - 10$$

Outra forma de definir uma função é através do comando '**unapply**'. Assim, para definir por exemplo a função com expressão $g(x) = \sin(x) + \cos(x)$, poderíamos fazer:

```
> g:=unapply(sin(x)+cos(x),x);
```

$$g := x \rightarrow \sin(x) + \cos(x)$$

```
>g(Pi);
```

$$-1$$

```
>g(0);
```

$$1$$

No comando 'unapply', precisamos de ter a expressão geral da função e a variável em que estamos a definir a função.

Vamos ver o que acontece se tentarmos definir uma função apenas com uma simples atribuição:

```
>h:=cos(x);
```

$$h := \cos(x)$$

Neste comando, $\cos(x)$ é uma expressão que foi atribuída à variável h . Não podemos considerar h como uma função de x , pois não obteríamos o resultado desejado com o comando $h(0)$:

```
>h(0);
```

$$\cos(x)(0)$$

O resultado desejado seria 1 ($= \cos(0)$).

Um dos erros mais comuns é tentar definir, por exemplo a função $a(x) = x^2$ da seguinte maneira:

```
>a(x):=x^2;
```

$$a(x) := x^2$$

Este comando faz com que o Maple guarde na memória que o valor da função para x é x^2 . Porém, esta função não está definida para nenhum outro argumento. Por exemplo:

```
>a(2);
```

$$a(2)$$

Podemos observar que o comando `a(2)` não deu 4.

Nem todos os nomes estão disponíveis para serem variáveis. O Maple tem alguns nomes pré-definidos e reservados (`Pi`, `set`, ...). Se tentarmos atribuir um valor a um desses nomes, é mostrada uma mensagem de erro a avisar que é um nome protegido.

2.5.3 Sequências e Listas

No Maple, as sequências têm a seguinte forma:

$$expr1, expr2, expr3, \dots, exprn$$

```
>a:=1,2,3;
```

$$a := 1, 2, 3$$

```
>b:=4,5,6;
```

$$b := 4, 5, 6$$

```
>a,b;
```

$$1, 2, 3, 4, 5, 6$$

Reparemos que ao escrever-se '`a,b`', juntaram-se as sequências '`a`' e '`b`' numa só.

Na construção de sequências, precisamos de ter em atenção dois comandos do Maple: o '`seq`' e o operador de repetição '`$`'.

```
>seq(expr(i),i=1..4);
```

$$expr(1), expr(2), expr(3), expr(4)$$

```
>seq(i!,i=1..5);
```

$$1, 2, 6, 24, 120$$

```
>seq(i^2,i=1..6);
```

1, 4, 9, 16, 25, 36

```
>r$7;
```

r, r, r, r, r, r, r

```
>2$3;
```

2, 2, 2

Em geral, 'seq(f(i),i=1..n)', produz a sequência

$$f(1), f(2), \dots, f(n)$$

e 'x\$n' produz uma sequência com n elementos

$$x, x, \dots, x$$

Por exemplo, seja s a sequência dos primeiros seis múltiplos de 5:

```
>s:=5,10,15,20,25,30;
```

$s := 5, 10, 15, 20, 25, 30$

```
>s[2];
```

10

Nestes casos, para calcular o n -ésimo termo de uma sequência s , podemos fazer 's[n];'.

As sequências preservam a ordem e a repetição dos seus elementos.

O Maple representa as listas na forma de vector, correspondendo à forma mais natural para trabalhar com sequências de objectos. As listas têm a seguinte forma:

$$[expr1, expr2, \dots, exprn]$$

O Maple também preserva a ordem e a repetição dos elementos de uma lista.

```
> restart:5
```

```
> l1 := [a, b, c];
```

$l1 := [a, b, c]$

```
> l2 := [1, 2, 3];
```

$l2 := [1, 2, 3]$

```
> L := [op(l1), op(l2)];
```

$L := [a, b, c, 1, 2, 3]$

```
> L[4];
```

1

```
> op(4, L);
```

1

```
> nops(L);
```

6

O comando '**op(L)**', dá-nos todos os elementos da lista '**L**', mas se escrevermos '**op(i,L)**', é-nos dado o *i*-ésimo elemento da lista '**L**'. O comando '**nops(L)**', dá-nos o número de elementos da lista '**L**'. Podemos observar que as listas '**l1**' e '**l2**' são ligadas pelo comando '**[op(l1),op(l2)]**'. Atribuiu-se a esta nova lista o nome '**L**'. Para ver qual o quarto elemento da lista **L**, pode fazer-se de duas maneiras: '**L[4]**' ou '**op(4,L)**' (neste caso, 1 é o quarto elemento da lista **L**). O número total de elementos da lista **L** é 6 e, portanto, '**nops(L)**' deu 6.

⁵Fez-se **restart** para apagar da memória as atribuições já feitas até ao momento.

Também podemos criar listas a partir de sequências.

```
>s:=seq(5*i,i=1..6);
```

$$s := 5, 10, 15, 20, 25, 30$$

```
>M:=[s];
```

$$M := [5, 10, 15, 20, 25, 30]$$

```
>M[2..5];
```

$$[10, 15, 20, 25]$$

Em geral, 'M[i..j]' dá os elementos da lista M desde o i-ésimo até ao j-ésimo em forma de lista.

No Maple, os conjuntos têm a seguinte forma:

$$\{expr1, expr2, \dots, exprn\}$$

Portanto, podemos dizer que um conjunto é uma sequência entre chavetas ($\{\}$). Um conjunto em Maple é como um conjunto no sentido usual, isto é, não preserva a ordem nem a repetição.

```
>{1,3,2,1};
```

$$\{1, 2, 3\}$$

O Maple assegura as operações básicas dos conjuntos, como a reunião, intersecção, etc.

```
>A:={1,2,3,4};
```

$$A := \{1, 2, 3, 4\}$$

```
>B:={3,4,5,6};
```

$$B := \{3, 4, 5, 6\}$$

```
>A union B;
```

```
{1,2,3,4,5,6}
```

```
>A intersect B;
```

```
{3,4}
```

```
>A minus B;
```

```
{1,2}
```

Esta última linha de comandos faz ' $A \setminus B$ '.

No Maple é ainda possível verificar se um elemento pertence ou não a um conjunto.

```
>member(2,A);
```

```
true
```

```
>member(2,B);
```

```
false
```

Com o comando '**member**', podemos verificar que $2 \in A$ e que $2 \notin B$. O comando '**member**', também é aplicável em listas.

Os comandos '**op**' e '**nops**' também funcionam com conjuntos.

Para ver mais características e exemplos de sequências, listas e conjuntos, deve-se consultar o *Help*. Na *worksheet* podemos escrever '?' logo seguido do nome em que pretendemos ajuda, por exemplo, para ter ajuda nas listas, podemos escrever '>?lists' e executar.

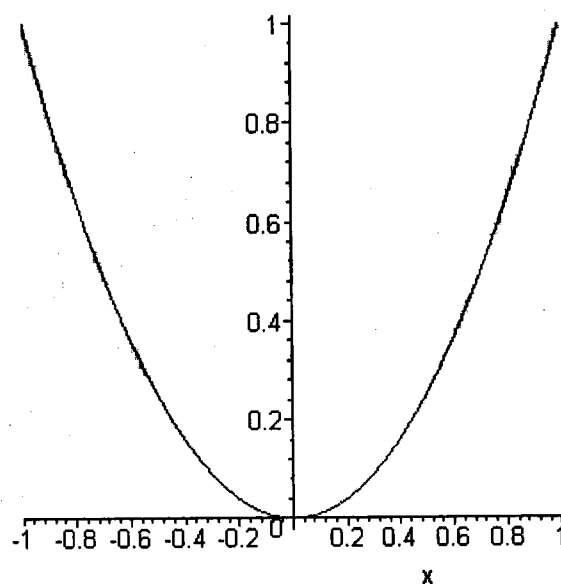
2.5.4 Gráficos

O Maple reúne um conjunto de comandos e opções para construção de gráficos, desde os gráficos bidimensionais mais simples até ao gráficos tridimensionais mais complexos,

passando por manipulação e animação gráfica.

A sintaxe para construir gráficos bidimensionais é baseada no comando `'plot'`. Por exemplo, para construir o gráfico da função $f(x) = x^2$ no intervalo $[-1, 1]$, devemos escrever:

```
>plot(x^2,x=-1..1);
```



De um modo geral, se quisermos construir o gráfico de uma função definida pela expressão algébrica $y = f(x)$, devemos digitar:

```
plot(f(x), x=a..b, y=c..d, op1, op2, ...);
```

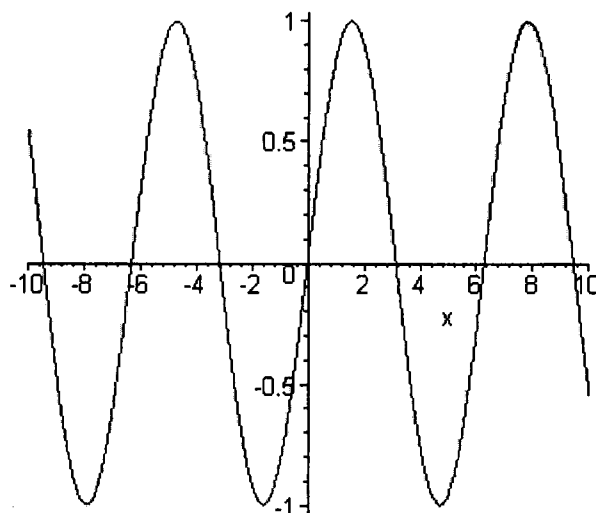
onde

- $f(x)$ é uma função real de variável real
- $x = a..b$ é o domínio da função ou a variação do x
- $y = c..d$ é a variação do y (isto é opcional)
- $op1, op2, \dots$ são opções do gráfico

Enquanto que a expressão algébrica e a variável x são fundamentais para fazer o 'plot', o y e as restantes opções do gráfico são opcionais.

No exemplo que se segue, não será referenciado o domínio que se pretende. No entanto, temos de indicar a variável independente ' x '. Por defeito, o Maple faz variar o domínio entre -10 e 10:

```
>plot(sin(x),x);
```



Podemos ver o mesmo exemplo, delimitando a variação de x e y e acrescentando a opção de ter cor azul:

```
> plot(sin(x),x=-Pi..Pi,y=-2..2,color=blue);
```

Se nada dissermos sobre a variação do y , o Maple atribui por defeito um y máximo e um y mínimo, de forma a que sejam visíveis as maiores oscilações do gráfico no domínio desejado.

Normalmente, quando queremos construir o gráfico de uma função, apenas indicamos a variação do x (variável do domínio), mas há funções em que a indicação da variação do y é bastante útil. Por exemplo:

```
> plot(tan(x),x=-2*Pi..2*Pi);
```

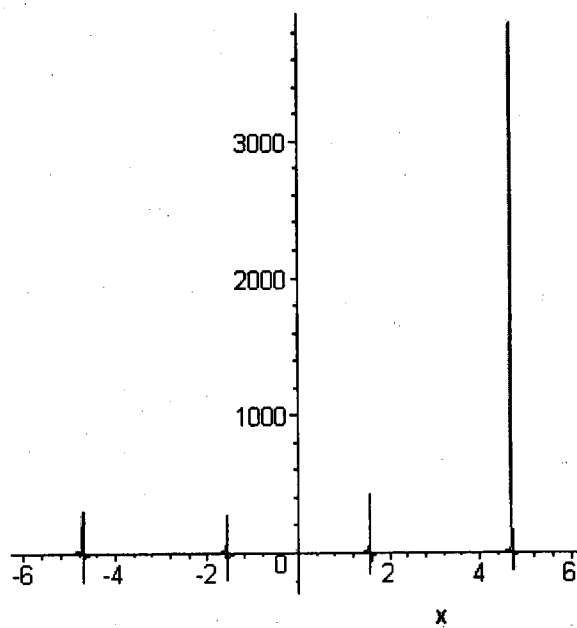
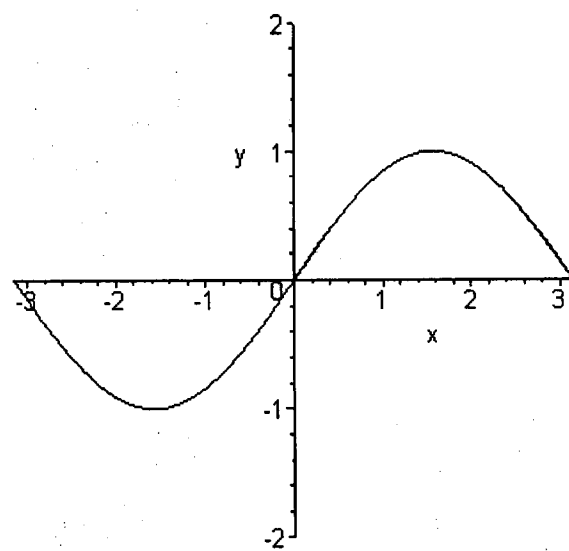


Figura 2.1: plot4

O gráfico é mostrado na figura 2.1

O resultado não é muito satisfatório. No entanto, se variarmos o y , o resultado é melhor.

```
> plot(tan(x), x=-2*Pi..2*Pi, y=-10..10);
```

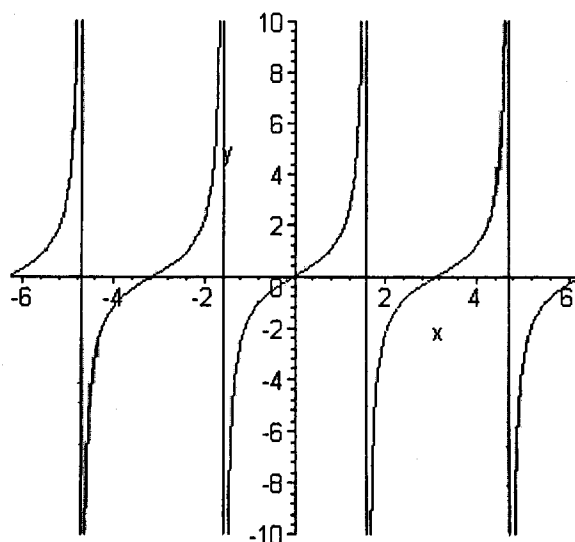


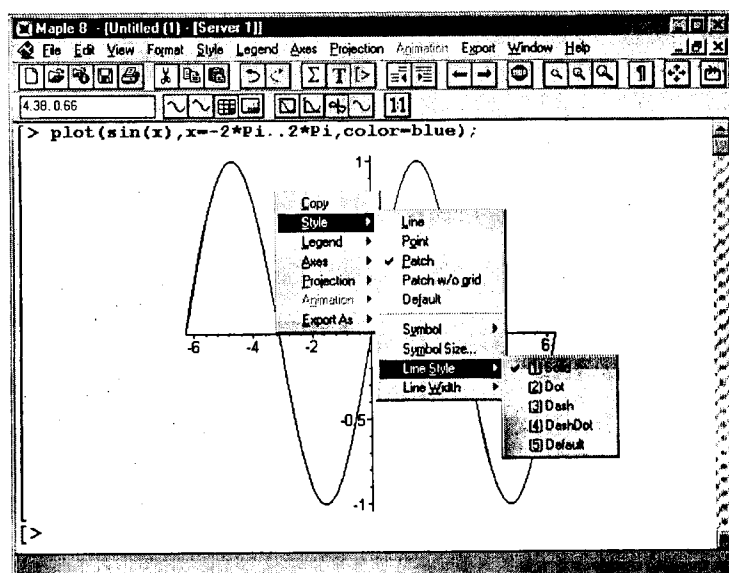
Figura 2.2: plot5

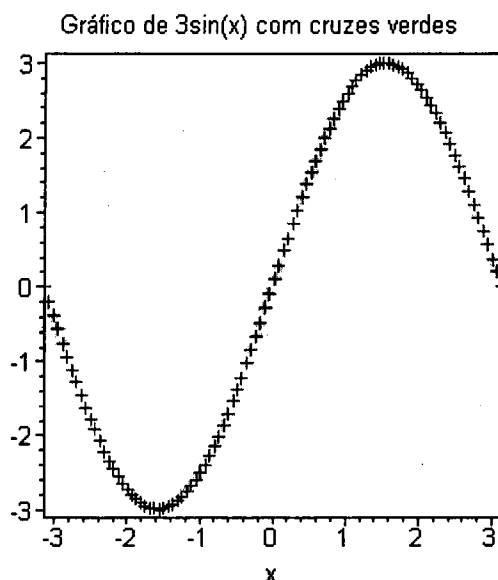
O gráfico é mostrado na figura 2.2

As opções do **plot** ⁶ inserem-se com igualdades do tipo opção = valor, como no exemplo anterior 'color=blue'. Algumas das opções do plot podem ser: 'axes' - tipo de eixos, 'color' - cor do gráfico, 'style' - pode ser linha ou ponto, 'linestyle' - tipo de linha, 'thickness' - espessura da linha, 'symbol' - símbolo dos pontos, 'symbolsize' - tamanho do símbolo, 'numpoints' - número de pontos utilizados para construir o gráfico, 'scaling' - proporção entre as escalas dos eixos, 'legend' - legenda do gráfico, 'title' - título do gráfico, etc. Estas e outras opções podem ser consultadas no menu *Help* pelo tópico de procura (*Topic Search*): 'plot,options', ou a partir da *worksheet* com ?plot,options.

⁶Muitas vezes, neste trabalho, usaremos a designação inglesa e do Maple "plot" para gráfico

Também podemos modificar algumas opções do gráfico depois de ele já ter sido esboçado. Se clicarmos no gráfico com o botão esquerdo do rato, aparecerá um rectângulo a delimitar o gráfico com marcas para redimensioná-lo. Podemos também reparar que o menu do Maple fica diferente, pois inclui mais menus e outros botões que nos permitem explorar algumas opções do plot (estes novos menus estão também disponíveis se clicarmos no botão direito do rato na região do gráfico).





forma de cruz (`symbol=cross`), cruces no tamanho 15 (`symbolsize=15`) num total de 100 cruces (`numpoints=100`) com uma cor esverdeada (`color=COLOR(RGB,0,0.7,0)`). Este último comando `COLOR(RGB, r, g, b)` permite ao utilizador escolher o tom de cor que quer, baseado no sistema de cores RGB (Red, Green, Blue), onde $0 < r < 1$ é a intensidade da cor vermelha, $0 < g < 1$ é a intensidade da cor verde e $0 < b < 1$ é a intensidade da cor azul. Há também um conjunto de 25 cores já predefinidas pelo Maple que podem ser encontradas no menu *Help / Topic Search* com '`plot,color`', ou executando '`?plot,color`' na *worksheet*.

Também podemos construir um gráfico a partir de uma função definida anteriormente:

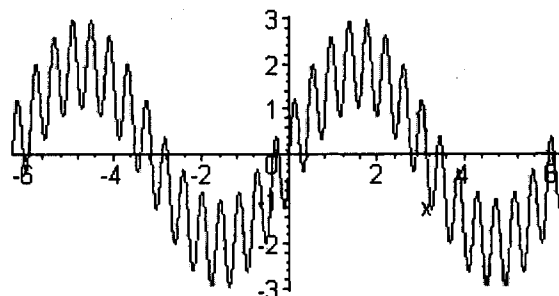
```
> restart;
```

```
> f:=x->2*sin(x)+sin(15*x);
```

$$f := x \rightarrow 2 \sin(x) + \sin(15x)$$

```
> plot(f(x), x=-2*Pi..2*Pi, scaling=constrained);
```

A opção '`scaling=constrained`', faz com que os eixos vertical e horizontal tenham a mesma escala.



Para fazer o gráfico de uma função definida por partes, devemos primeiro, definir uma função por partes. Para isso, usa-se o comando 'piecewise'. Por exemplo, para fazer o gráfico da função

$$g(x) = \begin{cases} x^2 - 2x, & \text{se } x \leq 1 \\ 1, & \text{se } 1 < x \leq 2 \\ 2, & \text{se } x > 2 \end{cases}$$

definimos a função g, e depois fazemos o plot:

```
> g:=x->piecewise(x<=1,x^2-2*x,1<x and x<=2,1,2<x,2);
```

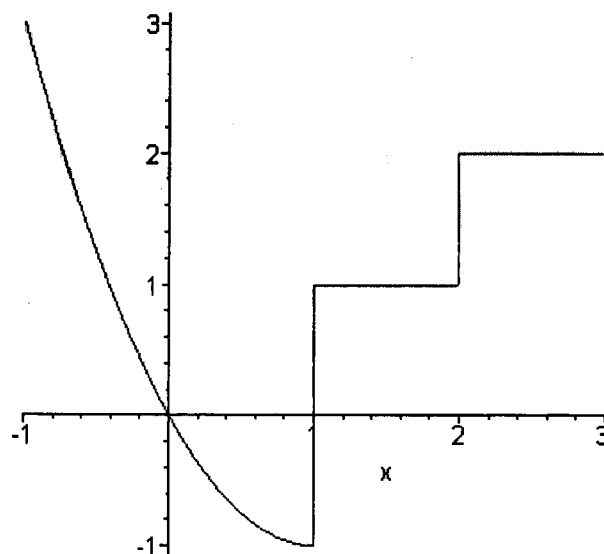
$$g := x \rightarrow \text{piecewise}(x \leq 1, x^2 - 2x, 1 < x \text{ and } x \leq 2, 1, 2 < x, 2)$$

```
> plot(g(x), x=-1..3);
```

Para que os segmentos verticais desapareçam, precisamos adicionar a opção 'discont=true'.

Esta opção evita que as discontinuidades sejam ligadas entre si com segmentos verticais.

```
> plot(g(x), x=-1..3, discont=true);
```



O gráfico é mostrado na figura 2.3

Para construir vários gráficos no mesmo sistema de eixos coordenados, devemos usar uma lista de funções $[f_1, f_2, \dots]$. Nestes casos, as funções devem ter um domínio comum e as opções devem também ser dadas na forma de lista, havendo correspondência entre a opção e a função.

```
> plot([sin(x), cos(x)], x=-Pi..Pi, color=[red, blue], legend=["sin", "cos"]);
```

O gráfico é o que foi mostrado na figura 2.4

Um método alternativo para fazer vários gráficos num só sistema de eixos, é usar a função 'display' que faz parte do package 'plots'. Para que o Maple reconheça esta função, é necessário "chamar" o package 'plots' com o comando 'with'.

```
> with(plots):
```

```
> p1:=plot(exp(x), x=-4..4, y=-4..4):
```

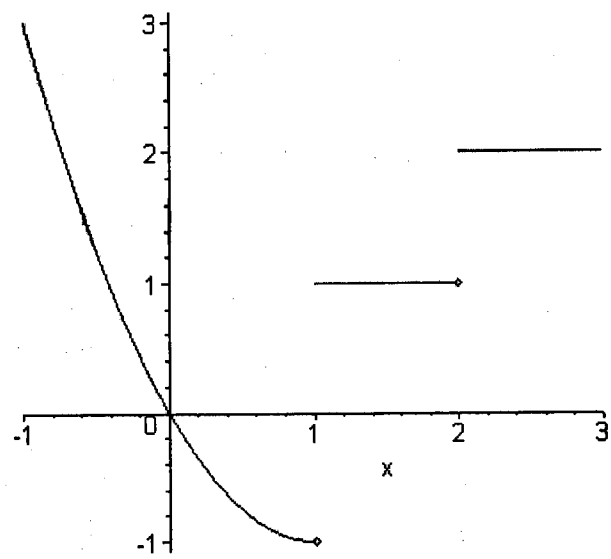


Figura 2.3: plot9

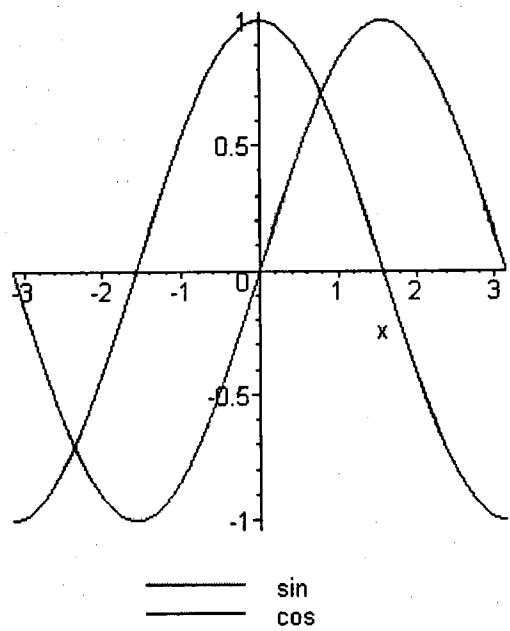
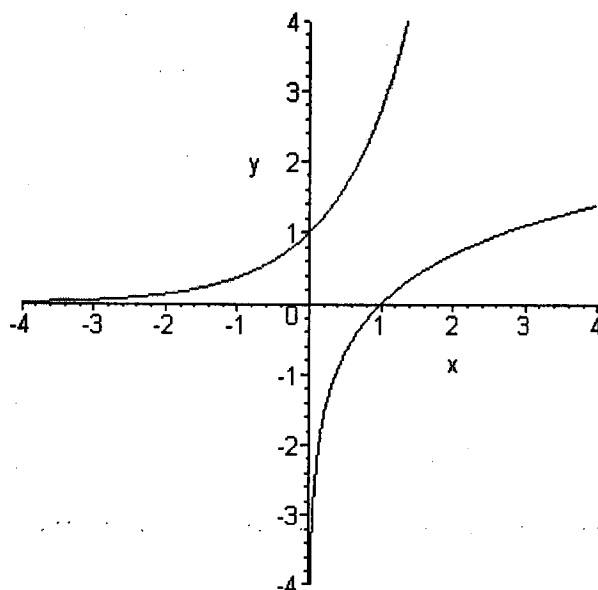


Figura 2.4: plot10

```
> p2:=plot(ln(x),x=-4..4,y=-4..4,color=blue):
> display(p1,p2);
```



A vermelho está o gráfico de e^x (exponencial) e a azul está o gráfico de $\ln(x)$ (logaritmo neperiano).

Quando atribuímos o plot à variável 'p1', não terminamos esse comando com ponto e vírgula (';'), pois seria mostrada uma lista contendo várias opções e uma longa lista de todos os pontos com que Maple constrói o gráfico.

```
> p1:=plot(exp(x),x=-4..4,y=-4..4);
p1 := PLOT(CURVES([[[-4., 0.0183156388887341822],
[-3.82562276666666668, 0.0218048519668097470],
[-3.67389806833333310, 0.0253773542313570876],
[-3.50326777666666667, 0.0300988661712307572],
[-3.33150643666666646, 0.0357392256149483904],
[-3.16056144833333352, 0.0424019279044508466],
[-3.00207356166666672, 0.0496839387711416960],
[-2.83796853499999990, 0.0585444762986677830],
```

```
[-2.66825088166666680, 0.0693734616436968710],  
(...)  
[3.91308840750000009, 50.0532984907124074],  
[4., 54.5981500331442362]], COLOUR(RGB, 1.0, 0., 0.)),  
AXESLABELS("x", "y"), VIEW(-4. .. 4., -4. .. 4.))
```

Uma das novidades desde o Maple 8, é a inclusão neste package da função '**interactive**'⁷. Esta função tem uma sintaxe mais simples que o '**plot**', pois não é necessário colocar o domínio e as opções do gráfico. Ao activar esta função, aparece em primeiro lugar, uma janela para escolhermos o tipo de gráfico que queremos, depois aparece uma outra janela, onde é possível escolher variadas opções para o gráfico que queremos construir, incluindo o domínio e a variação de outras variáveis. Só quando clicamos, nessa janela, no botão '**Plot**', é que o gráfico é mostrado na *worksheet*. A sintaxe desta função é '**interactive(expressão)**'. Por exemplo, para fazer o gráfico da função x^2 no intervalo $[-10,10]$ e com as opções que aparecem por defeito na segunda janela interactiva, devemos fazer:

```
> with(plots):  
> interactive(x^2);
```

A primeira janela que aparece deverá ser ⁸:

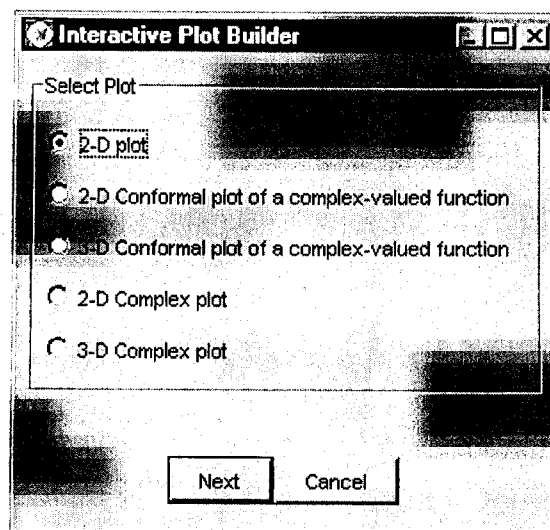
Nesta janela, deveremos escolher '**2-D plot**' e clicar no botão '**Next**' (Próxima). Se clicássemos no botão '**Cancel**', a construção deste gráfico seria anulada.

Logo a seguir aparecerá a janela mostrada na figura 2.5.

Nesta janela, como vamos aceitar as opções que estão por defeito, basta clicar no botão '**Plot**', e o gráfico aparecerá na *worksheet*.

⁷Usa a tecnologia das Maplets

⁸Um pouco diferente no Maple 9



No entanto, poderíamos nesta janela fazer as alterações que pretendêssemos, inclusive podíamos alterar a expressão da função logo na primeira caixa de texto. Nas caixas de texto que estão á frente de 'Domain of x from', devemos escrever o domínio que pretendemos para a função. Como já queríamos o domínio no intervalo $[-10,10]$, não precisamos de mudar nada. Pela janela, podemos ver que está escolhido para o gráfico, o estilo de linha sólida e fina (*line, solid, thin*), a cor é vermelha (*red*) e os eixos normais (*normal*). Poderíamos ter escrito o título do gráfico na caixa de texto por baixo de 'Title'. Em 'View', se escolhêssemos 'Constrained scaling', o eixo dos xx e dos yy apareceriam com a mesma escala. Para além de outras opções, sabemos que o gráfico foi definido com 50 pontos e que o sistema de coordenadas é o cartesiano. O botão 'Back', faz-nos voltar à janela anterior, o botão 'Reset', desfaz as alterações feitas e o botão 'Cancel', aborta a construção deste gráfico. O botão 'Command', fecha a janela e mostra na linha de output da *worksheet* a sintaxe do comando `plot` com todas as opções que forem alteradas na janela. O botão 'Plot', como já foi dito, fecha a janela e mostra o gráfico na *worksheet*.

Portanto, o gráfico seria o seguinte:

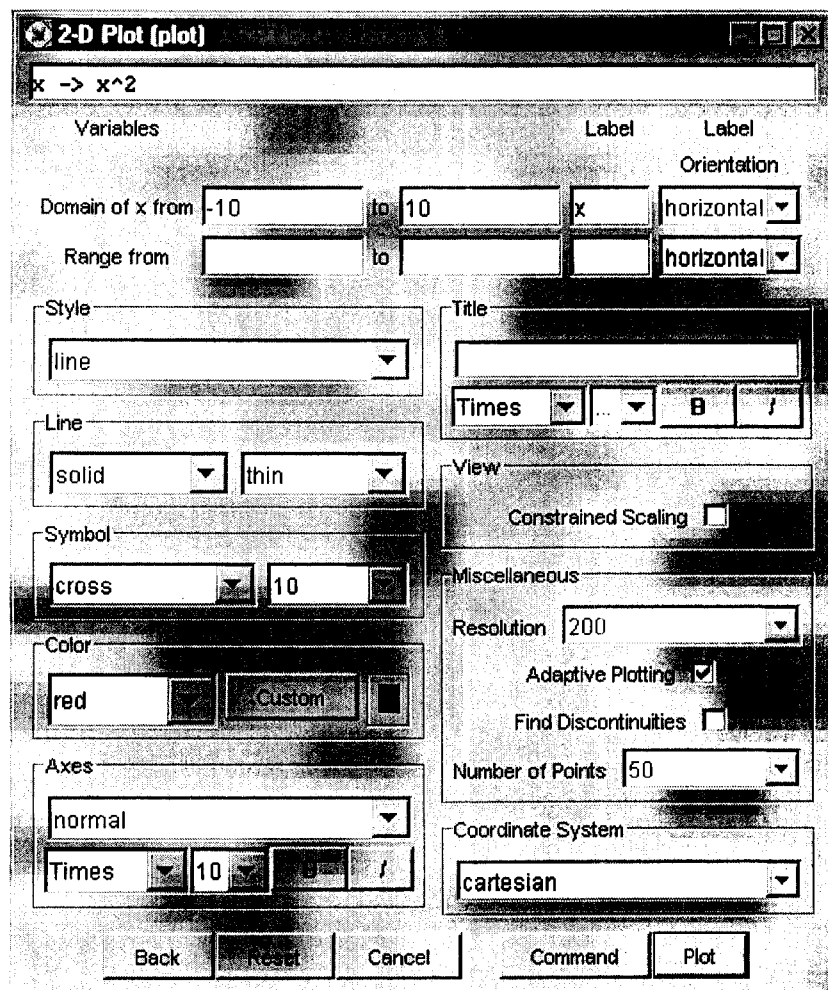
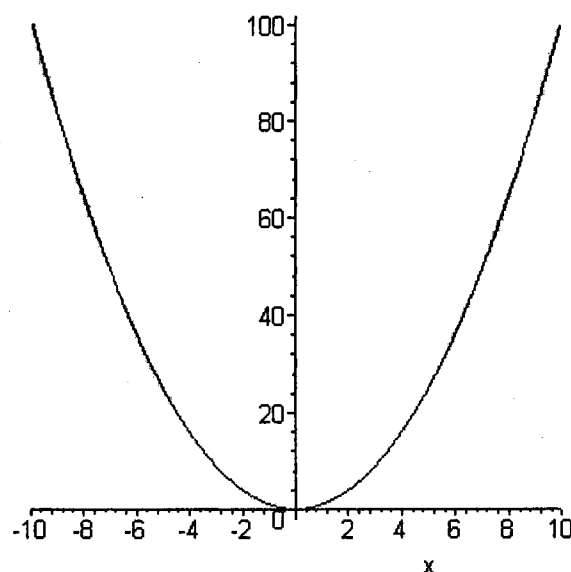


Figura 2.5: interactive

O Maple permite que a partir de uma sequência de gráficos seja possível obter uma animação. Isto pode ser feito com o comando `'animate'`. A sintaxe deste comando é `'animate(f(x,t),x=a..b,t=c..d,opções)'`, onde $f(x,t)$ é uma função em ordem a x , x é o domínio em que queremos ver a função, t é o parâmetro de variação da função e as opções são as do plot mais a opção "frames", que é o número de gráficos que aparece na sequência. Por defeito `frames=16`.

O `'animate'` está dentro do package `'plots'`, portanto é necessário fazer `'with(plots):'`.



```
> with(plots):
```

```
> animate(sin(x^2+t), x=-Pi..Pi, t=0..2*Pi, frames=9, numpoints=100);
```

O gráfico é mostrado na figura 2.6

A sequência de gráficos é mostrada na figura 2.7:

Esta sequência começa em $\sin(x^2)$ e acaba em $\sin(x^2 + 2\pi)$, isto significa que os nove gráficos da sequência correspondem aos gráficos de $\sin(x^2 + t)$ com t a variar no conjunto $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}, 2\pi\}$. Como a passagem de um gráfico para o seguinte é relativamente rápida, dá a ideia de movimento. Quantos mais "frames" tivermos, mais perfeita é a animação.

Para iniciar a animação, devemos clicar em cima do gráfico inicial com o botão esquerdo do rato. Ao fazer isso, a barra de menus altera-se aparecendo uns botões parecidos com os de um leitor de cassetes.

Para dar início à animação, devemos clicar no botão '*Play the animation*'. O último

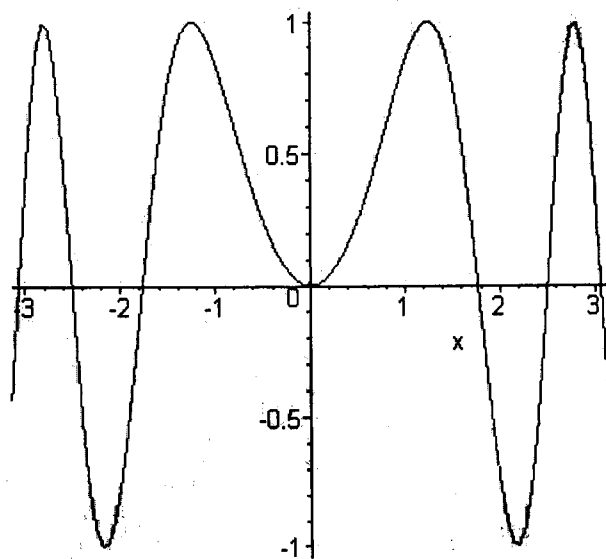


Figura 2.6: animate

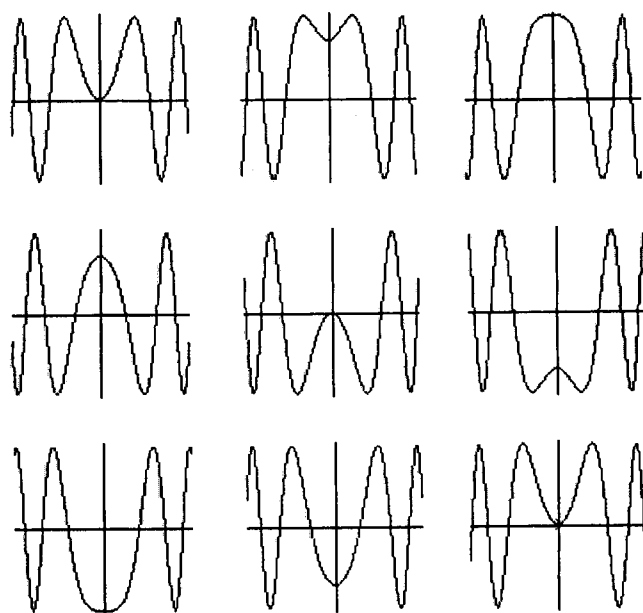


Figura 2.7: sequência



botão, permite que a animação seja repetida indefinidamente, podendo parar-se com o botão '*Stop the animation*'. Outra maneira de activar a animação é clicar com o botão direito do rato no gráfico. Ao fazer isso, aparecerá um menu, em que uma das opções é '*Animation*'. Nessa opção é possível fazer o mesmo que os botões já faziam. Uma das opções é '*Export as*'. Esta opção permite gravar a figura em vários formatos, sendo um deles o formato *gif* que é animado.

Em termos de gráficos, o Maple é bastante completo. Isto são apenas algumas noções básicas a nível bidimensional. Para ver estes comandos com maior detalhe, mais exemplos e todas as outras possibilidades, inclusive a nível tridimensional, basta consultar o '*Help*', na barra de menus, com o tópico '*Plotting*', ou executando '*?plotting,overview*', na *worksheet*.

2.5.5 Cálculo Diferencial e Integral

Esta secção contém apenas alguns exemplos de resolução de problemas de cálculo (limites, derivadas, integrais, etc.). É de referenciar que o Maple não é o programa para usar sem um acompanhamento crítico dos resultados por ele obtidos. Os algoritmos usados nem sempre dão as soluções (por vezes induzem em erro).

O limite de uma função quando a variável tende para um certo valor, é calculado com o comando '*limit*'. Se escrevermos '*Limit*' com L grande, não é calculado o limite, mas é apresentada a expressão do limite.

```
> L:=Limit((x^2-1)/(x-1),x=1);
```

$$L := \lim_{x \rightarrow 1} \frac{x^2 - 1}{x - 1}$$

```
> l:=limit((x^2-1)/(x-1),x=1);
```

```
l:= 2
```

Portanto, podemos dizer que:

```
> L=1;
```

$$\lim_{x \rightarrow 1} \frac{x^2 - 1}{x - 1} = 2$$

Também podem ser calculados limites que se aproximem do infinito.

```
> restart;
```

```
> f:=x->(x+4)/(x-3);
```

$$f := x \rightarrow \frac{x + 4}{x - 3}$$

```
> Limit(f(x),x=3,left)=limit(f(x),x=3,left);
```

$$\lim_{x \rightarrow 3^-} \frac{x + 4}{x - 3} = -\infty$$

```
> Limit(f(x),x=infinity)=limit(f(x),x=infinity);
```

$$\lim_{x \rightarrow \infty} \frac{x + 4}{x - 3} = 1$$

Usou-se 'left', pois foi calculado um limite lateral à esquerda. Se calculássemos um limite lateral à direita, usaríamos 'right'.

O Maple pode facilmente derivar funções de uma ou mais variáveis. A sintaxe para derivar uma função $f(x)$ é 'diff(f(x),x)'.

```
> diff(x^2,x);
```

```
2x
```

```
> p:=x^4-2*x^3+x^2-5*x+1;
```

$$p := x^4 - 2x^3 + x^2 - 5x + 1$$

```
> diff(p,x);
```

$$4x^3 - 6x^2 + 2x - 5$$

```
> diff(p,x$2);
```

$$12x^2 - 12x + 2$$

Nesta última linha foi calculada a segunda derivada do polinómio p .

O comando 'Diff', não calcula a derivada, mas apresenta a expressão que indica a derivada.

```
> Diff(sin(x),x)=diff(sin(x),x);
```

$$\frac{d}{dx} \sin(x) = \cos(x)$$

O Maple também tem o operador diferencial 'D'. Se f é uma função derivável de uma variável, então Df é a derivada f' . Para calcular a função $g'(x)$ da função g , fazemos:

```
> g:=x->cos(x);
```

$$g := x \rightarrow \cos(x)$$

```
> D(g);
```

$$x \rightarrow -\sin(x)$$

O Maple pode ser usado como um formulário de derivadas e de integrais.

Se f é uma função em ordem a x , então a sintaxe para calcular o integral $\int_a^b f dx$ é ' $\text{int}(f(x), x=a..b)$ '. Para calcular o integral indefinido usa-se ' $\text{int}(f(x), x)$ '. Há também formas de representar o integral sem o calcular, usando ' $\text{Int}(f(x), x=a..b)$ ' e ' $\text{Int}(f(x), x)$ '.

> $\text{int}(x, x)$;

$$\frac{x^2}{2}$$

> $\text{Int}(\cos(x), x) = \text{int}(\cos(x), x)$;

$$\int \cos(x) dx = \sin(x)$$

> $\text{int}(2*x, x=1..2)$;

$$3$$

> $\text{Int}(\exp(x), x=0..1) = \text{int}(\exp(x), x=0..1)$;

$$\int_0^1 e^x dx = e - 1$$

Para ter acesso a mais exemplos e mais detalhes sobre limites, derivação e integração, podemos consultar o '*Help*', com os seguintes tópicos: '*limit*', '*diff*', '*int*', '*student*', '*calculus*', etc. Outra forma de obter mais informação sobre este assunto, é executar na *worksheet* os tópicos anteriores precedidos de '?'.

2.5.6 Programação e Procedimentos

Para além das potencialidades a nível de cálculo simbólico interactivo, o Maple é também uma linguagem de programação. Um utilizador que tenha tido experiência anterior com outras linguagens de programação, como 'C', 'PASCAL', 'FORTRAN', ou 'BASIC', não achará estranha esta linguagem.

Quando estamos a construir um algoritmo, uma das situações mais comuns é necessitarmos de impor condições para que certos comandos sejam executados. A declaração condicional 'if', está incluída na linguagem de programação do Maple.

A forma mais simples de usarmos a declaração condicional é:

```
if condição then  
    seqcom;  
end if;
```

onde '*condição*' é uma expressão lógica que é avaliada como verdadeira ou falsa, e '*seqcom*' é uma sequência de comandos que são executados, se a '*condição*' for verdadeira. O **end if** termina a declaração condicional.

Os comandos '*seqcom*', devem ser sempre separados por ';', ou ':'. O último destes comandos pode terminar ou não com ';', ou ':'.⁹

Portanto, esta forma de declaração condicional acaba por se traduzir no seguinte:

Se a *condição* for verdadeira, **então**

executa este(s) comando(s);

fim:

Por exemplo:

```
> restart;
```

```
> x:=1;
```

```
> if x>0 then 9
```

⁹No Maple pode mudar-se de linha sem executar o comando. Basta premir ao mesmo tempo as teclas 'Shift' e 'Enter' do teclado.

```
y:=sqrt(x);  
z:=ln(x);  
end if;  
> y;
```

1

```
> z;
```

0

Neste exemplo, só são atribuídos valores às variáveis 'y' e 'z', se o 'x' for positivo. Repare-se que a seguir ao 'end if', está ':'. Se estivesse ';', apareceria uma linha de saída com as atribuições a 'y' e a 'z'. Vamos ver o que aconteceria se o 'x' fosse negativo:

```
> restart;  
> x:=-1;  
> if x>0 then  
    y:=sqrt(x);  
    z:=ln(x);  
end if;  
> y;
```

y

```
> z;
```

z

Neste caso não seriam atribuídos quaisquer valores a 'y' e a 'z'. Repare-se que mesmo com o símbolo ';' a seguir ao 'end if', não foi mostrada qualquer atribuição ao 'y' e ao 'z'.

Uma forma mais completa da declaração condicional é:

```
if condição then
    seqcom1;
else
    seqcom2;
end if;
```

Neste caso, se a *condição* for verdadeira, **então** será executada a sequência de comandos *seqcom1*, **senão**, é executada a sequência de comandos *seqcom2*.

Por exemplo, uma forma de calcular o valor absoluto de um número através de uma declaração condicional é:

```
> restart;
> x:=-2;
> if x>=0 then
    y:=x;
else;
    y:=-x;
end if;
```

$y := 2$

A forma mais completa da declaração condicional é:

```
if condição1 then
    seqcom1;
elif condição2 then
    seqcom2;
```

```
elif condição3 then
    seqcom3;
(...)
else
    seqcomN;
end if;
```

Neste caso, se a *condição1* for verdadeira, então *seqcom1* é executada e o 'if' encerra. Senão, é verificada a *condição2* e, se for verdadeira, *seqcom2* é executada e o 'if' encerra. E assim vão sendo testadas todas as condições até chegar à condição verdadeira. Se todas as condições forem falsas, então é executada *seqcomN*.

Nas condições podem ser usados os operadores de relação '<' - menor que ($<$), '>' - maior que ($>$), '<=' - menor ou igual a ($\leq$), '>=' - maior ou igual a ($\geq$) e '<>' - diferente ($\neq$), para além de outros operadores.

As estruturas de repetição, são muito usadas na programação. O Maple inclui os comandos 'for' e 'while' que são bastante eficientes para a repetição de determinadas acções.

A sintaxe para o ciclo 'for' (para), é:

```
for variável from valor1 to valor2 by step do
    seqcom;
end do;
```

A parte 'by step', relativa ao incremento da variável é opcional. Isto significa que se não escrevermos esta parte, a variável é aumentada de 1 em 1. O comando ou sequência de comandos *seqcom* que estão entre o 'do' e o 'end do', é executado para cada valor que a variável assumir no intervalo [*valor1*, *valor2*]. A variável inicia com *valor1* e vai sendo

incrementada por *step* unidades em cada ciclo, e enquanto o valor da variável for menor ou igual a *valor2*, *seqcom* vai sendo executada. Quando a variável assume um valor maior que *valor2*, o ciclo 'for' encerra. Por exemplo:

```
> restart:
> soma:=0:
> for i from 1 to 5 do
    soma:=soma+i;
end do;
```

soma := 1

soma := 3

soma := 6

soma := 10

soma := 15

Esta estrutura de repetição calcula a soma dos 5 primeiros números naturais. O ';' a seguir a 'end do', permitiu que fossem mostradas as somas parciais feitas em cada ciclo. Antes de começar o ciclo, a variável 'soma' tinha o valor 0. No primeiro ciclo, *soma* toma o valor $0+1=1$, pois $i=1$. No segundo ciclo, $soma=1+2=3$, pois *soma* tinha já o valor 1 atribuído no primeiro ciclo e desta vez $i=2$. No terceiro ciclo, $soma=3+3=6$, no quarto ciclo, $soma=6+4=10$ e no quinto ciclo, $soma=10+5=15$. Ou seja,

```
> 1+2+3+4+5;
```

15

Podemos também fazer uma coisa tão simples, como escrever os números entre 3 e 4 incrementados de 0.2 em 0.2:

```
> restart:
> for i from 3 to 4 by 0.2 do
    print(i);
```

```
end do;
```

3

3.2

3.4

3.6

3.8

4.0

Outra forma de criar uma estrutura de repetição é usando o comando **'while'** (enquanto).

```
while condição do  
    seqcom  
end do
```

A *condição* é avaliada, e enquanto for verdadeira, o comando ou sequência de comandos *seqcom* é executado. O ciclo acaba quando a *condição* é falsa.

Esta estrutura pode ajudar-nos a descobrir qual o primeiro número primo maior que 1000:

```
> restart:  
> j:=1000:  
> while not isprime(j) do  
    j:=j+1  
end do:  
j;
```

1009

A variável 'j' começa por ter o valor 1000. Depois faz-se o ciclo enquanto o j não é primo.

Vai-se acrescentando 1 valor a j , até que ele seja primo. Quando j for primo o ciclo acaba e é escrito o j . Neste caso $j=1009$ é o primeiro primo maior que 1000.

Estes são alguns exemplos simples do uso da declaração condicional **if** e das estruturas de repetição **for** e **while**. É claro que ao idealizar um programa de execução, é possível combinar estas três estruturas, que podem ser incluídas em procedimentos.

A elaboração de procedimentos (*procedures*), constitui uma das partes mais importantes da programação com o Maple. Escrever um programa em Maple pode ser muito simples. Ao colocar um grupo de comandos do Maple entre **proc()** e **end proc**, estamos a elaborar um procedimento ou subrotinas. Ao dar um nome a esse procedimento, basta executar esse nome, para executarmos todo o grupo de comandos que incluímos no procedimento.

O primeiro exemplo de um procedimento, mostra apenas uma mensagem:

```
> restart;  
> pep:=proc()  
    "este é o Primeiro Exemplo de um Procedimento";  
end proc;  
  
pep := proc() "este é o Primeiro Exemplo de um Procedimento" end proc
```

O nome deste procedimento é 'pep'. Para executá-lo, basta fazer:

```
> pep();  
    "este é o Primeiro Exemplo de um Procedimento"
```

São os parêntesis a seguir ao nome que definem pep como um procedimento. De facto, este parêntesis significam que este procedimento não usa variáveis ou valores externos ao procedimento para serem usadas internamente. É natural que hajam procedimentos que usem internamente valores de variáveis que lhes sejam fornecidos externamente. Nesses

casos, teríamos de colocar as variáveis que fossem necessárias dentro desses parêntesis.

Vejamos qual seria o resultado se não escrevêssemos os parêntesis:

```
> pep;
```

pep

O próximo exemplo, usa o comando "evalf":

```
> metade:=proc(x)
```

```
    evalf(x/2);
```

```
end proc;
```

Para evitar uma linha de saída com a descrição do procedimento, usou-se dois pontos (:) em vez de ponto e vírgula (;) a seguir a 'end proc'. Este procedimento espera por um valor de x para calcular um valor aproximado da sua metade. Quando um procedimento é executado, o resultado obtido é relativo ao último cálculo feito pelo Maple nesse procedimento. Como `evalf(x/2)` foi o último cálculo a ser efectuado no procedimento 'metade' (na verdade, foi o único cálculo feito neste procedimento), o resultado vai ser uma aproximação à metade do x .

```
> metade(3);
```

1.500000000

```
> metade(123456);
```

61728.

```
> metade(a);
```

0.5000000000a

```
> metade(7/3);
```

1.166666667

Um procedimento pode ser visto como uma função. Por exemplo:

```
> F:=proc(x)
```

```
  y:=x^2-1;
```

```
end proc;
```

Warning, 'y' is implicitly declared local to procedure 'F'

```
> F(5);
```

24

Neste procedimento foi usada uma variável 'y'. Apareceu um aviso, porque a variável não foi declarada no início do procedimento. Quando isto acontece, o Maple declara automaticamente a variável como local. Uma variável local é uma variável que só é válida dentro do procedimento. Para evitar o aviso, deveríamos ter declarado esta variável:

```
> F:=proc(x)
```

```
  local y;
```

```
  y:=x^2-1;
```

```
end proc;
```

Os avisos devem servir para verificar se foi ou não esquecida a declaração de alguma variável. As variáveis podem ser declaradas como locais ou como globais. Se 'y', fosse declarada variável global, era válida como $x^2 - 1$ mesmo fora do procedimento. O seguinte exemplo, mostra a diferença em declarar uma variável como local ou global:

```
> g:=proc()
```

```
  local a;
```

```
  a:=exp(2);
```

```
  evalf(a);
```

```
end proc;  
> g();  
  
7.389056099  
  
> a;  
  
a
```

Como foi declarada local, fora do procedimento 'a' não tem nenhuma atribuição.

```
> h:=proc()  
  global a;  
  a:=exp(2);  
  evalf(a);  
end proc;  
> h();  
  
7.389056099  
  
> a;  
  
e2
```

Se 'a' for declarada global, fora do procedimento ela toma o valor atribuído dentro do procedimento.

Exemplo de um procedimento com duas entradas:

```
> pitagorascat:=proc(c1,c2)  
  local h,h2;  
  h2:=c1^2+c2^2;
```

```

h:=hipotenusa=evalf(sqrt(h2));
end proc:

```

Este procedimento, calcula a hipotenusa, dados os catetos de um triângulo rectângulo, usando o teorema de Pitágoras.

```
> pitagorascat(3,4);
```

hipotenusa = 5.

É recomendável que num procedimento se tenha cuidado com o tipo de valores que são usados. Por exemplo, pode gerar resultados pouco animadores utilizar números fraccionários num procedimento que efectua cálculos com números inteiros. Nos procedimentos, podemos restringir as variáveis ao tipo desejado. Podemos assim evitar que o Maple execute todo o procedimento, para no final nos dar indicação de erro. Se forem introduzidos valores que não se adequam ao cálculos do procedimento, ao restringirmos as variáveis, o Maple não chega a executar o procedimento avisando que as variáveis não são do tipo desejado.

```

> mdc:=proc(a::posint,b::posint)
    igcd(a,b);
end proc:
> mdc(123,456);

```

3

```
> mdc(-3,6);
```

Error, invalid input: mdc expects its 1st argument, a, to be of type posint, but received -3

Neste procedimento, só serão aceites valores positivos e inteiros (*posint*). Se forem dados outros valores, o procedimento não chega a ser efectuado. Isto faz-nos ganhar tempo, se o procedimento for relativamente grande.

Para ver a lista dos tipos de valores que podem ser usados no Maple, pode consultar-se o *Help*, pelo tópico '*type*', ou executar *?type* na *worksheet*.

Um procedimento pode fazer referência a outros procedimentos e pode mesmo fazer referência a si próprio.

```
> Fib:=proc(n::nonnegint)
  if n<2 then
    n;
  else
    Fib(n-1)+Fib(n-2);
  end if;
end proc;
```

O parâmetro n , foi restringido ao tipo dos inteiros não negativos (*nonnegint*).

Este procedimento calcula os números de Fibonacci, que são definidos recursivamente. A sequência de Fibonacci (f_n) é, então assim definida:

$$f_0 = 0, f_1 = 1$$

e para $n \geq 2$,

$$f_n = f_{n-1} + f_{n-2}.$$

A sequência dos primeiros 20 números de Fibonacci é:

```
> seq(Fib(n), n=0..19);
```

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181

Normalmente, os procedimentos recursivos levam o Maple a fazer muitos cálculos, pelo que se podem tornar pouco eficientes. Por exemplo, para fazer `Fib(30)`, o Maple gasta cerca de 20 segundos. Para fazer `Fib(200)`, demoraria bastante tempo, pois eram esquecidos e repetidos bastantes cálculos. Para resolver estes problemas, existe a opção **remember** que é utilizada nos procedimentos para recordar os cálculos feitos.

```
> Fib:=proc(n::nonnegint)
  option remember;
  if n<2 then
    n;
  else
    Fib(n-1)+Fib(n-2);
  end if;
end proc;
```

Assim, este procedimento fica mais rápido. Vamos ver o tempo que o Maple demora a calcular `Fib(n)` para alguns valores com o comando **time** (tempo).

```
> time(Fib(30));
```

0.

```
> time(Fib(200));
```

0.001

Assim, não demorou nada calcular `Fib(30)` e demorou apenas um milésimo de segundo a calcular `Fib(200)`.

Para saber mais sobre opções para procedimentos, consultar o *Help*, pelo tópico 'procedure', ou executando '?remember' na *worksheet*.

Neste capítulo tentou-se falar um pouco sobre o Maple, dando algumas noções que se consideram básicas para trabalhar e compreender o que se vai falar mais á frente - Maplets. Para saber mais sobre o Maple, para além do Help interactivo disponível no próprio programa, pode-se também consultar informação nas referências [19] '*Maple8 - Learning Guide*', [17] '*Maple8 - Getting Started Guide*', [18] '*Maple8 - Introductory Programming Guide*', [16] '*Maple8 - Advanced Programming Guide*', [8] '*The Maple Book de Frank Garvan*', [2] '*Introdução à Computação Algébrica com Maple de Lenimar Nunes de Andrade*' e na internet, nos sites [24] 'www.maplesoft.com', [21] 'www.mapleapps.com' e [25] 'www.maple4students.com'.

Capítulo 3

Maplets

Depois de introduzida a linguagem Maple, descrevemos aqui as funcionalidades disponibilizadas por esta linguagem através da tecnologia das Maplets.

3.1 O que é uma Maplet ?

As Maplets comportam-se para o utilizador final como qualquer programa de um sistema operativo. De facto, as Maplets aparecem como janelas interactivas, onde é possível realizar computação algébrica, numérica e gráfica. Com recurso à tecnologia das Maplets, disponibilizam-se todas as potencialidades do Maple, mesmo para quem não sabe o que é o Maple ou nunca ouviu falar deste programa. Através de janelas com botões, caixas de texto, menus, etc., pode ser recolhida informação do utilizador, tratar internamente essa informação recorrendo a todas as potencialidades do Maple, executando qualquer comando disponibilizado pelo Maple e, de seguida apresentar na mesma janela resultados numéricos, algébricos ou gráficos.

Para um professor pode ser bastante útil apresentar aos seus alunos uma ferramenta como as Maplets, onde é possível trabalhar e manipular interactivamente com os conteúdos programáticos. Os alunos não necessitam de aprender a linguagem Maple, porque a partir das janelas só precisam de inserir dados e clicar em botões. A partir daí, as Maplets vão

"buscar" todos os comandos necessários do Maple para responderem ao pretendido. Os professores, apenas com algumas noções básicas de Maple, podem construir as Maplets que lhes forem úteis, começando desde o início, ou a partir de Maplets já feitas e que estão disponíveis na internet e Maplets que estarão incluídas neste trabalho. Depois de ver uma Maplet e como ela foi construída, é possível modificá-la melhorando alguns aspectos e adaptando ao que se quer fazer.

Na criação de Maplets, o programador escreve os comandos que põe em funcionamento a Maplet que quer implementar, e depois grava essa *worksheet* como um ficheiro do tipo 'maplet'.

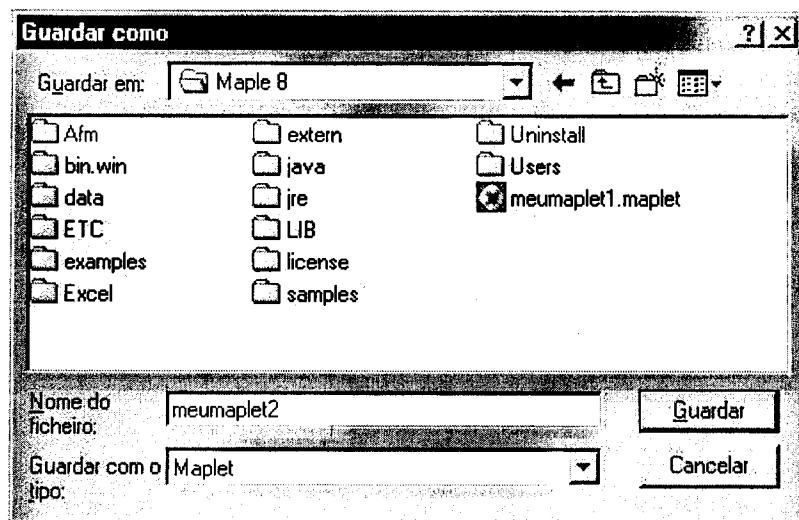


Figura 3.1: Gravação da *worksheet* num ficheiro do tipo 'maplet'

A partir daqui, o que resulta é um ficheiro que o aluno, ou outro utilizador, pode "executar" pelo simples clique do rato, tal como faz para abrir outro tipo de ficheiros como documentos de texto ('Word'), folhas de cálculo ('Excel'), etc.

Neste capítulo, não nos vamos preocupar com a programação das Maplets, mas através de exemplos, observar o que é possível obter com esta tecnologia.

Na versão escrita deste trabalho, são apresentadas apenas as imagens das janelas produzidas por estas Maplets. Na versão em CD-Rom, clicando numa imagem, o utilizador tem acesso à Maplet seleccionada, e se tiver curiosidade em ver a linguagem Maple que lhe deu origem, clicando na legenda da figura, terá acesso a esse código.

3.2 Maplets Simples

Uma das Maplets mais simples que se pode construir, é aquela em que apenas se mostra uma mensagem. Por exemplo:

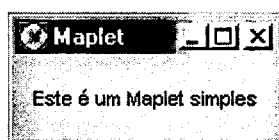


Figura 3.2: maplet simples 1

Na parte superior desta janela aparece uma barra em cima, que inclui o título e os botões:



Se não for especificado nenhum título na construção da Maplet, o título por defeito é 'Maplet'. Os botões já referidos que aparecem à frente do título, servem respectivamente para minimizar, restaurar e fechar a janela.

Ao espaço onde aparece a mensagem (que foi a única coisa que incluímos nesta Maplet), chamamos o corpo da janela (*window body*).

Para além da barra de título e do corpo da janela, uma Maplet pode ter por opção uma barra de menu.

No exemplo seguinte (fig. 3.3), temos uma Maplet com 2 menus, e o que acontece quando seleccionamos um desses menus:

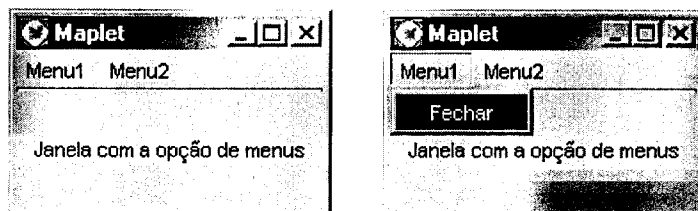


Figura 3.3: menu 1

Nesta Maplet aparece para além do que já se viu na primeira, uma barra de menus. Neste caso, no 'Menu1', tínhamos a opção de fechar a janela, clicando em 'Fechar'.

A parte principal de uma Maplet é o corpo da janela. Nesse espaço é possível incluir, por exemplo, botões para executar alguma operação. Quando se clica no botão, o Maple executa o comando especificado para obter um resultado coerente com o que se espera.

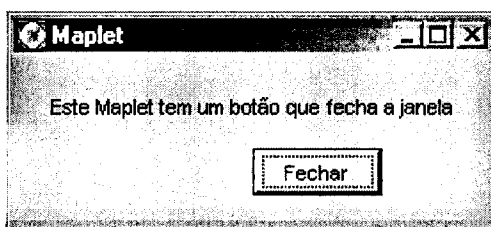


Figura 3.4: botão

Nesta Maplet (fig. 3.4) o botão 'Fechar', faz o que se espera. Quando clicamos no botão, a janela fecha-se. A cada botão de uma Maplet associa-se uma acção. Neste caso, ao pressionar o botão, a Maplet acciona o comando `Shutdown()`.

Uma Maplet também pode incluir uma caixa de texto para entrada de dados. Neste espaço, podemos inserir expressões, valores, nomes, etc.

No exemplo seguinte (fig.3.5), mostra-se uma Maplet que inclui uma caixa para entrada de dados (neste caso texto), e um botão que fecha a Maplet:

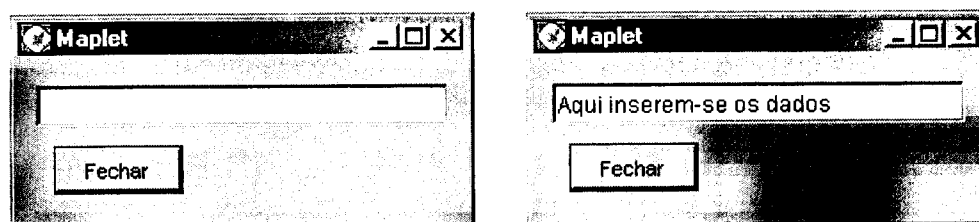


Figura 3.5: linha de texto 1

Quando se abre esta Maplet, a caixa de texto aparece em branco com um cursor a piscar. Isto significa que o utilizador pode inserir qualquer tipo de dados, mediante o que for pedido. Para isso, basta clicar com o botão esquerdo do rato em cima da caixa de texto e escrever os dados que se pretendem inserir. Cada caixa de texto pode ter o tamanho que o autor da Maplet quiser, bem como a cor e o tipo de letra. A validação do que está na caixa de texto pode ser feita de duas maneiras diferentes, mediante a opção do autor da Maplet: ou clicando num botão da janela que sirva para esse efeito ou clicando em qualquer parte da janela (propriedade "lose focus").

O resultado de uma operação (*o output*) pode ser visto numa caixa de texto. Neste caso, podemos ter uma caixa de texto para introdução de dados (*input*) e uma outra para mostrar os dados (*output*), mas pode tudo ser feito na mesma caixa de texto.

3.3 Maplets para Matemática

Nesta secção, veremos como algumas Maplets podem ser úteis desde o cálculo ou simplificação de expressões numéricas até à análise gráfica de funções, passando pela derivação e integração. Como já foi referido, de facto é possível usar nas Maplets todo e qualquer comando existente no programa Maple.

Recorrendo às Maplets "simples" apresentadas na secção anterior, podemos ver uma Maplet que calcula (simplifica) expressões numéricas. O utilizador introduz na caixa de texto a expressão na notação usual à que se introduz numa calculadora. No primeiro exemplo (fig. 3.6), o resultado é dado na caixa de texto onde é inserida a expressão (é uma caixa de *input* e *output*) e no segundo exemplo (fig. 3.7) o resultado é exibido numa caixa diferente (só de *output*).

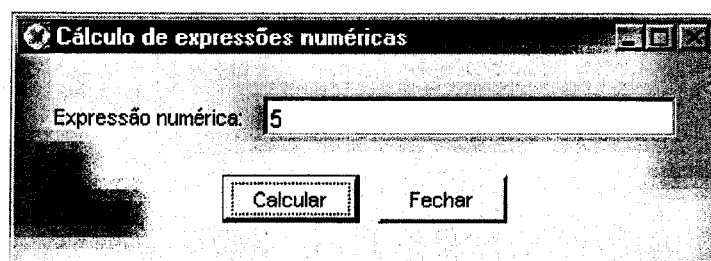


Figura 3.6: expressões numéricas 1

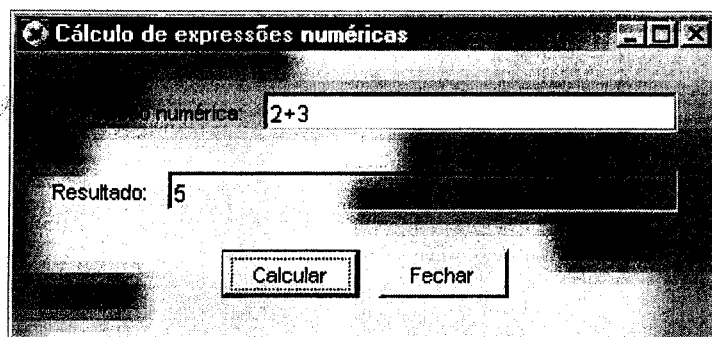


Figura 3.7: expressões numéricas 2

Nestas Maplets, acrescentou-se o título 'Cálculo de expressões numéricas'. No segundo exemplo, a segunda caixa não é editável, isto é, não é possível escrever, pois é uma caixa exclusiva para *output*. É possível para o autor da Maplet escolher se uma caixa é de *input*, de *output* ou de ambos. Quando a janela é activada, o utilizador deve escrever uma expressão numérica, usando o teclado, na caixa de *input*, que neste caso é a caixa à frente

de 'Expressão numérica:'. Neste exemplo a expressão inserida foi '2+3', ao pressionar o botão 'Calcular', o Maple faz 'evalf(2+3);' e apresenta o resultado (5.) na caixa de *output*. Ao pressionar o botão 'Fechar', a janela fecha-se.

No exemplo anterior, a expressão usada era bastante simples. Mas podemos querer ver o que se passa com uma expressão mais complicada. Para isso, é possível usar uma notação próxima da notação usual da matemática com caixas de *input* e *output* próprias para essa notação. O autor da Maplet tem assim a possibilidade de escolher uma caixa de *input*, tecnicamente designada por MathMLEditor ou *output* (MathMLViewer) que permite assim, utilizar a "tradicional" simbologia, como raízes quadradas, fracções, etc.

O próximo exemplo (fig. 3.8), calcula (simplifica) uma expressão numérica usando uma Maplet que contém uma caixa de *input*, onde o utilizador pode usar a notação "tradicional" da matemática.

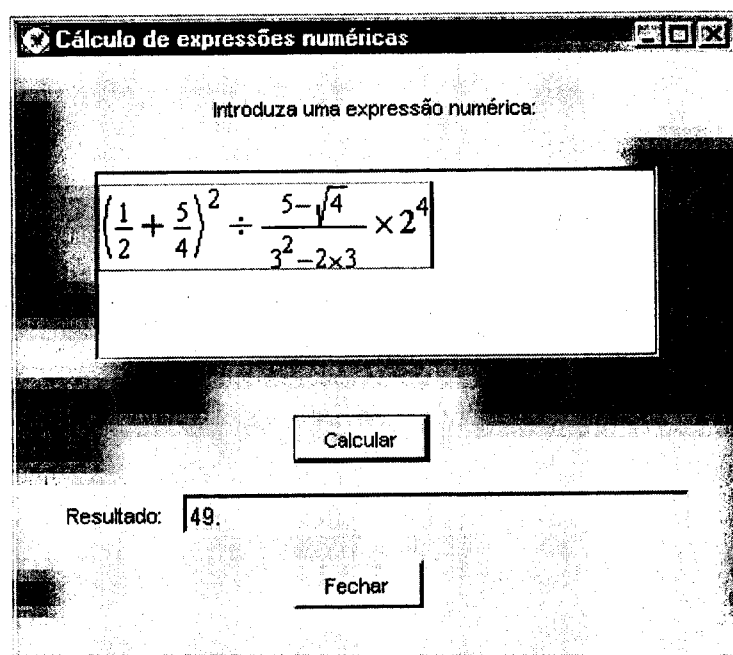
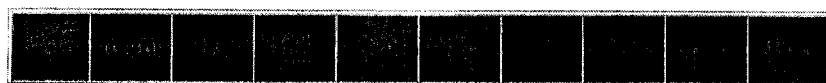


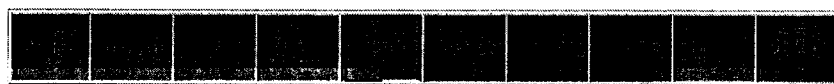
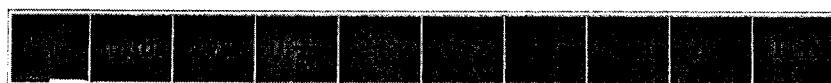
Figura 3.8: expressões numéricas 3

Quando a janela é activada, o utilizador deve inserir a expressão na caixa de *input*. Neste caso, é mesmo necessário pressionar no botão esquerdo do rato em cima da caixa, para se poder iniciar a introdução da expressão. Os algarismos e os símbolos disponíveis no teclado, podem ser obtidos pressionando a respectiva tecla, mas para obter outros símbolos é necessário aceder a um menu que fica disponível pressionando o botão direito do rato em cima da caixa. Nesse menu encontram-se vários icons que permitem a introdução de uma grande variedade de entidades matemáticas e que a própria imagem sugere.



Ao pressionar o botão esquerdo do rato em cada um dos rectângulos que constituem a barra, abre-se um submenu com vários símbolos relacionados com os que estão desenhados no respectivo rectângulo. Pressiona-se o botão esquerdo do rato no símbolo que desejamos inserir e ele aparece na caixa de *input*.

Só a título de exemplo, mostra-se aqui alguns dos símbolos que foram usados para inserir a expressão da Maplet anterior:



Há também "atalhos" no teclado para se obterem alguns símbolos. Por exemplo, para obter a raiz quadrada pode pressionar-se ao mesmo tempo as teclas 'Ctrl' e 'Q'. Na tabela 3.1 estão alguns exemplos de "teclas de atalho":

Ctrl+Q	$\sqrt{}$	
Ctrl+P	$\square\square$	Shift+^
Ctrl+B	$\square\square$	Shift+_
Ctrl+F	$\square$	

Tabela 3.1: Teclas de atalho para obter alguns símbolos

Depois de inserir a expressão, clica-se no botão 'Calcular' e o resultado aparece na caixa de *output*. Como o resultado esperado é um número decimal, a caixa de *output* é normal.

A utilização deste tipo de entrada de expressões nas Maplets, embora seja um método com uma visualização mais "amigável", muitas vezes traduz-se num maior tempo para a introdução da expressão que se pretende. A maior parte das vezes, opta-se apenas pelo uso de caixas de texto normais, contudo, existe sempre a possibilidade do utilizador poder escolher entre que tipo de caixa de entrada prefere, como se pode ver na Maplet da secção 5.5.

Na Maplet anterior (fig. 3.8), tínhamos uma caixa de entrada com MathMLEditor e uma caixa de saída na notação usual das calculadoras. Mas podemos ter uma Maplet em que tanto a caixa de entrada como a de saída (*output*) usem a simbologia usual associada à escrita de expressões matemáticas, ou seja, teremos uma caixa de entrada MathMLEditor e uma caixa de saída MathMLViewer.

A próxima Maplet (fig. 3.9), simplifica uma expressão numérica utilizando o comando **simplify**, e apresenta o resultado numa caixa do tipo MathMLViewer (notação "tradicional" da matemática):

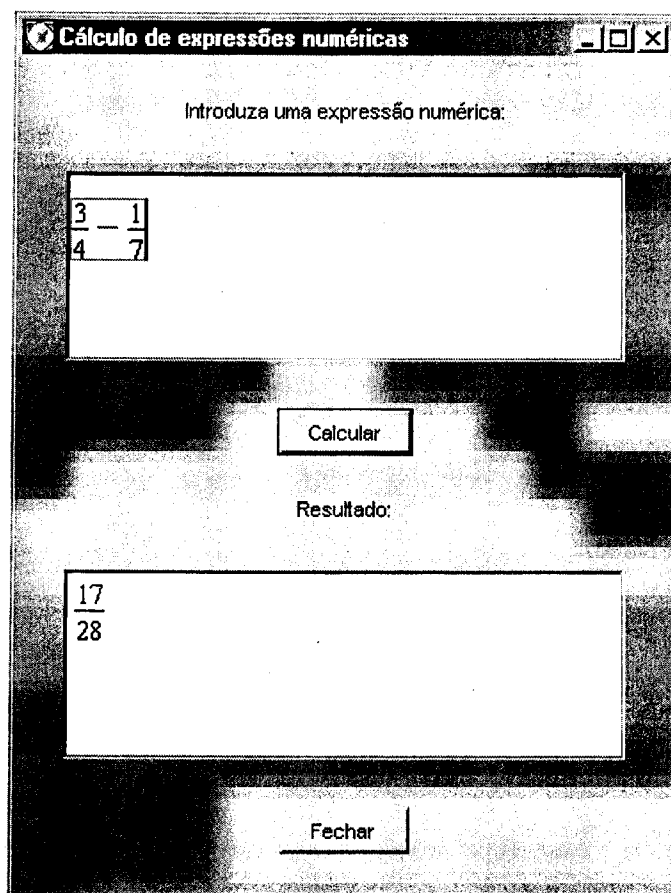


Figura 3.9: expressões numéricas 4

Estes são alguns exemplos, para calcular expressões numéricas. As Maplets podem substituir calculadoras científicas, podem fazer o mesmo que as calculadoras gráficas e muito mais.

Os alunos do ensino básico e secundário verão com bons olhos a utilização de uma Maplet que calcule as raízes reais de equações do 2.º grau. É certo que o podem fazer numa calculadora gráfica, mas certamente será visualmente mais simples trabalhar com a Maplet.

Ao construir uma Maplet que determine as raízes reais de uma equação de 2.º grau, o autor da Maplet deve ter em atenção que podem surgir raízes não reais. Para evitar que

apareçam as raízes complexas, o autor deve fazer aparecer uma janela de alerta, dizendo que não existem raízes reais.

Na próxima Maplet (fig. 3.10), o utilizador só tem de introduzir os coeficientes de uma equação do tipo $ax^2 + bx + c = 0$ e clicar no botão que fornece as raízes reais da equação ou um alerta da não existência das mesmas.

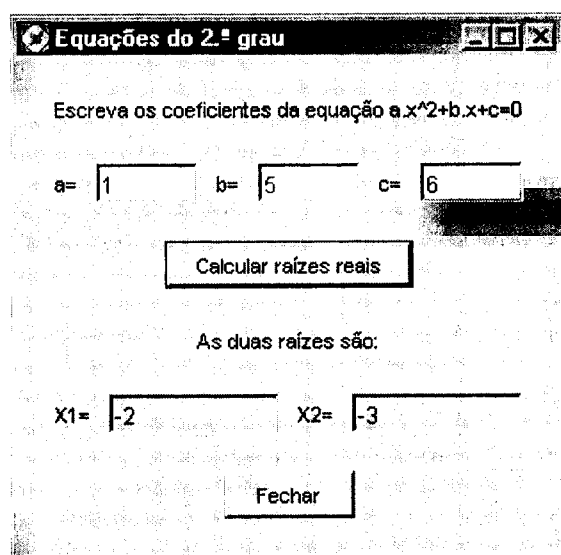
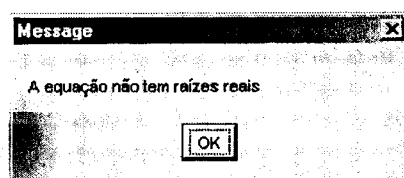


Figura 3.10: Equações do 2.º grau

Ao pressionar em 'Calcular raízes reais', o Maple efectua o comando `solve(eq,x);`, sendo 'eq' a equação introduzida. Como a equação $x^2 + 5x + 6 = 0$ tem raízes reais, o resultado foi $X1=-2$ e $X2=-3$. Se em vez dos coeficientes 1, 5 e 6, tivéssemos inserido 1, 5 e 7 respectivamente, a equação $x^2 + 5x + 7 = 0$ não teria raízes reais. Portanto, ao clicar em 'Calcular raízes reais', apareceria uma janela de alerta como a seguinte:



Outro exemplo de uma Maplet, é uma janela onde se pode testar se um número é ou não primo, quais são os divisores de um número e a sua decomposição em factores primos. Isto mostra que uma só Maplet pode ter várias funções.

No exemplo seguinte (fig. 3.11), o utilizador deve inserir um número inteiro positivo. Depois só precisa escolher a operação que pretende efectuar, bastando para tal, pressionar sobre o respectivo botão. Os resultados são apresentados na caixa de *output*.

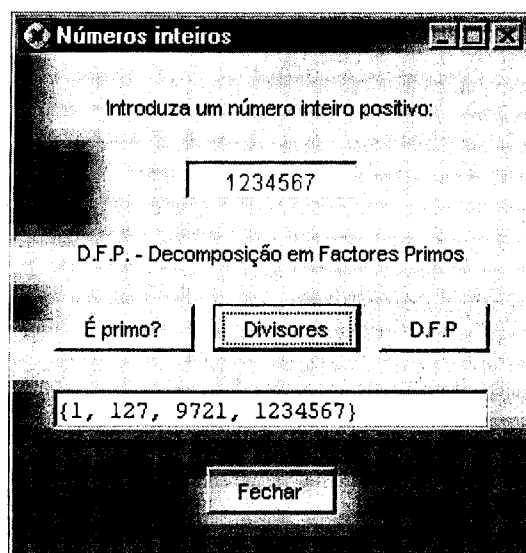


Figura 3.11: Números inteiros

Para cada execução de um botão, o Maple executa a acção correspondente. Por exemplo, ao pressionar 'É primo?', é accionado um procedimento que inclui uma declaração condicional 'if' e uma atribuição à caixa de *output*.

```
> primo:=proc(n)
if isprime(n) then
Set(Out=cat(n,"é primo"));
else
Set(Out=cat(n,"não é primo"));
```

```
end if;  
end proc;
```

Veremos mais à frente, que o elemento **Set** permite atribuir a elementos da Maplet valores obtidos em procedimentos. Assim, se **Out** for a variável que representa o que está na caixa de *output*, dependendo do valor lógico da condição, o elemento **Set** atribui 'n é primo' ou 'n não é primo' à caixa de *output*.

Uma das grandes vantagens das Maplets relativamente a calculadoras, é o cálculo algébrico. Dois bons exemplos da utilidade das Maplets no cálculo algébrico são Maplets que permitam derivação e integração, recorrendo aos algoritmos intrínsecos do Maple. Estas Maplets podem servir como um potencial formulário de derivadas e integrais.

A Maplet seguinte (fig. 3.12) é um exemplo simples de uma janela que deriva expressões com uma variável (x). Dada uma função $f(x)$, com a simples acção de carregar num botão, é dada a sua derivada $f'(x)$. A instrução que está por detrás desta acção é `diff(f(x),x);`.

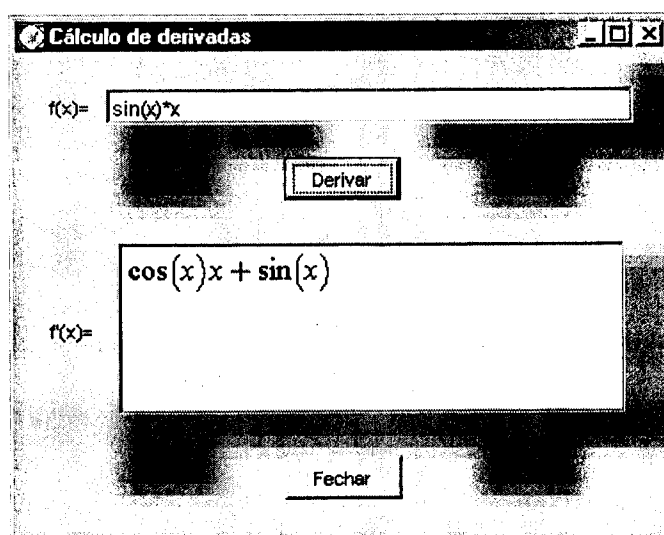


Figura 3.12: Derivação 1

Na caixa de input insere-se a função que se quer derivar na notação Maple. Pressiona-se o botão 'Derivar', e na caixa de output aparece a derivada da função inserida, na linguagem usual da matemática.

Pode-se tentar generalizar um pouco o funcionamento desta janela, possibilitando a derivação de expressões com mais do que uma variável. Neste caso, deverá aparecer uma caixa de *input* que permita escolher a variável de derivação.

A Maplet seguinte (fig. 3.13) é muito parecida com a anterior, mas aqui há a possibilidade de se inserirem expressões com mais variáveis. Por isso, há mais uma caixa de *input* logo a seguir a 'Variável de derivação:', onde o utilizador deverá introduzir a variável relativa à derivação desejada. Por opção, na caixa de *output*, aparecerá uma igualdade entre a indicação da derivada e o cálculo da mesma.

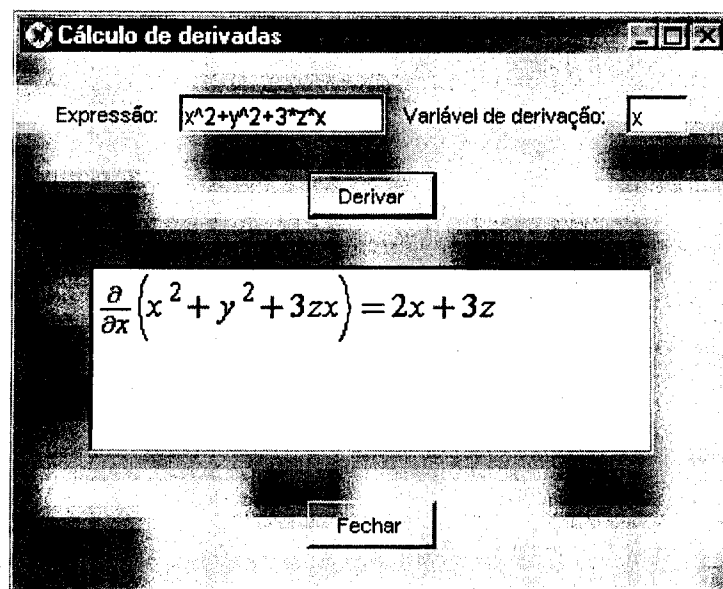


Figura 3.13: Derivação 2

Mais à frente, veremos um exemplo de uma Maplet que resolve derivadas de funções passo a passo. Esta será certamente uma Maplet bastante útil não só para os alunos, mas

também para os professores, pois são mostrados todos os passos de resolução, bastando ao utilizador escolher a regra de derivação a aplicar em cada passo.

Até aqui temos associado a uma Maplet uma janela, mas uma Maplet pode conter várias janelas. Por exemplo, na Maplet de integração é pertinente que antes de abrir a janela de integração propriamente dita, apareça uma janela (fig. 3.14) onde se pergunta qual o tipo de integração desejado: definido ou indefinido.

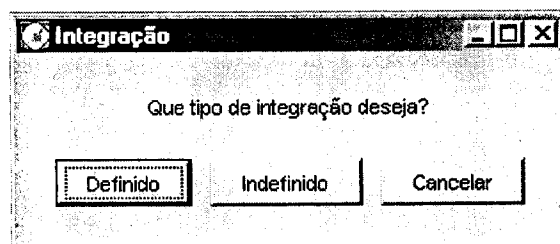


Figura 3.14: Integração 2

Ao abrir esta Maplet, a primeira janela que aparece inclui uma pergunta com duas respostas possíveis e a opção de cancelar (fechar) a Maplet. À pergunta 'Que tipo de integração deseja?', o utilizador deve responder 'Definido' ou 'Indefinido', pressionando respectivamente o primeiro ou o segundo botão. Se desejar sair da Maplet, pressiona 'Cancelar'.

Se escolher trabalhar com integrais indefinidos, ao pressionar no respectivo botão, fecha-se esta primeira janela e abre-se a janela correspondente aos integrais indefinidos (fig. 3.15).

Na janela da fig. 3.15, o utilizador insere a expressão que deseja integrar, a variável de integração e ao pressionar o botão 'Integrar', obtém na caixa de output o resultado, ou seja, o integral da expressão relativamente à variável escolhida.

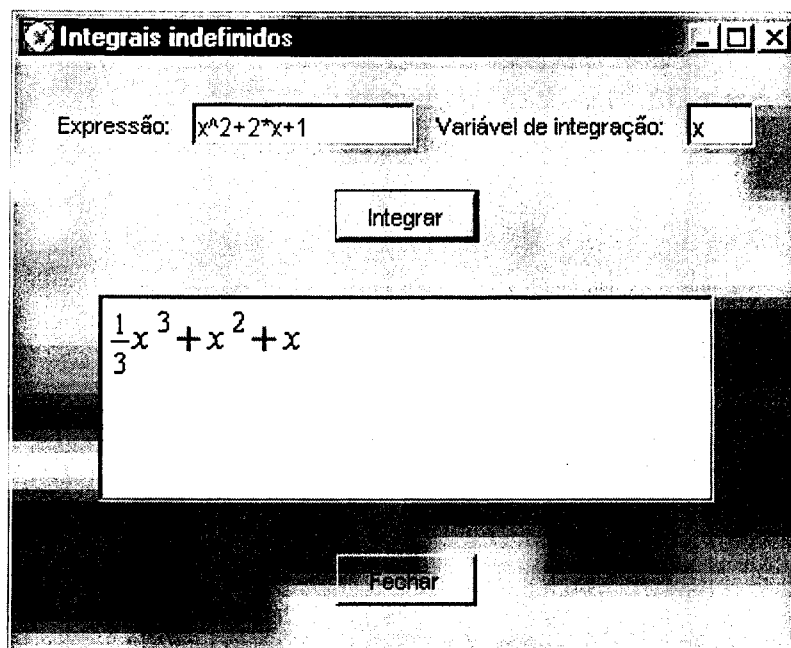


Figura 3.15: Integrais indefinidos

Se o utilizador tivesse escolhido trabalhar com integrais definidos, na primeira janela (fig. 3.14), ao pressionar no respectivo botão, esta fechava-se e abria-se a janela dos integrais definidos (fig. 3.16).

Esta janela é muito semelhante à anterior, mas aqui é necessário indicar os limites inferior e superior do integral. Quando o botão 'Integrar' é accionado, o Maple executa o comando $\text{Int}(\text{expr}, v=l1..l2) = \text{int}(\text{expr}, v=l1..l2)$, onde expr é a expressão, v é a variável de integração, $l1$ é o limite inferior do integral e $l2$ é o limite superior. Na janela anterior, o Maple executava $\text{int}(\text{expr}, v)$.

A figura 3.16 diz respeito à janela de integrais definidos, ou seja, corresponde a uma janela que permite obter o integral de uma função de uma variável (x) num determinado intervalo:

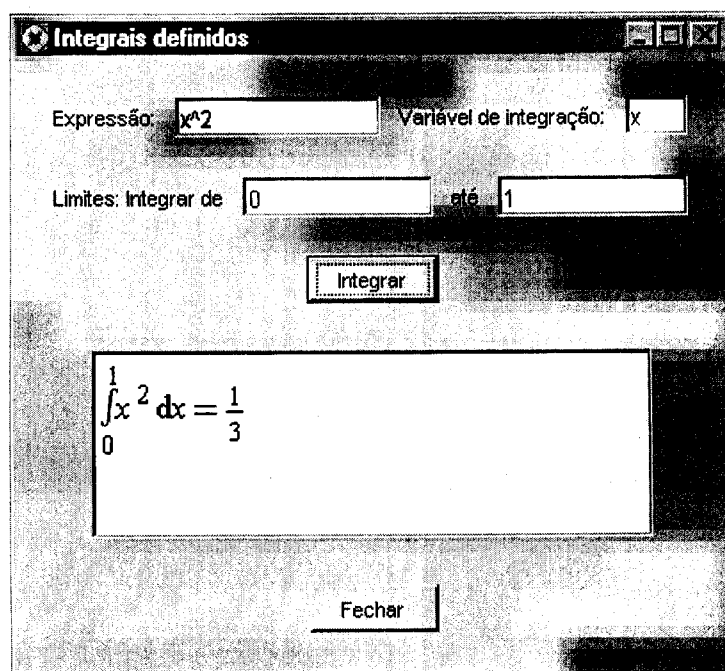


Figura 3.16: Integrais definidos

Portanto, esta Maplet acaba por ter 3 janelas, embora o utilizador cada vez que acciona a Maplet, apenas use duas janelas. A primeira janela aparece sempre e depois trabalha com a janela que escolher.

A Maplet para calcular limites de funções (fig. 3.17), na sua forma mais básica, ficaria muito parecida com a Maplet anterior.

O Maple executa o comando $\text{Limit}(f, x=l)=\text{limit}(f, x=l)$, onde f é a função ou expressão e l é o valor aproximado da variável, quando o botão 'Limite' é accionado.

Até este momento, foram vistos vários exemplos de Maplets em que o *output* é um valor ou expressão com maior ou menor formato próximo da notação tradicional. A seguir veremos que o *output* também pode ser gráfico, isto é, a caixa de *output* pode conter representações gráficas de objectos.

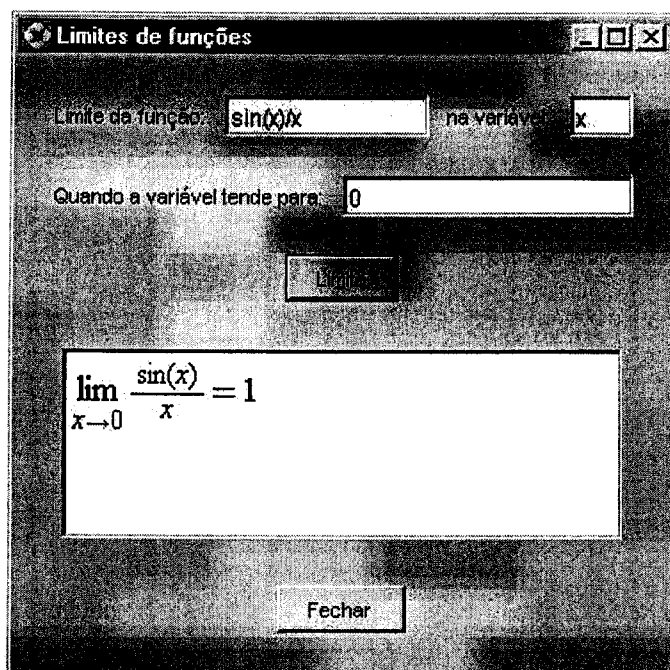


Figura 3.17: Limites 1

No primeiro exemplo sobre este tipo de Maplets (fig. 3.18), não existem opções para o gráfico da função. O utilizador limita-se a escrever a expressão da função de qualquer gráfico, mas depois não tem acesso a modificar qualquer opção. O gráfico é desenhado, por opção do autor da Maplet, no intervalo $[-10,10]$ e todas as outras opções que se poderia ter aparecem por defeito do Maple.

Assim, se inserirmos a função $f(x) = x^2$, ao pressionar o botão 'Fazer um gráfico da função', o Maple recebe a instrução para executar `plot(f(x),x)` e o gráfico é mostrado na caixa de *output* própria para gráficos (Plotter).

A janela da figura 3.18 foi alterada em termos de dimensão. Originalmente, ela era maior. De facto, o autor da Maplet pode evitar que uma janela seja alterada na sua dimensão, mas por defeito, o utilizador pode fazer as alterações que achar convenientes no que respeita

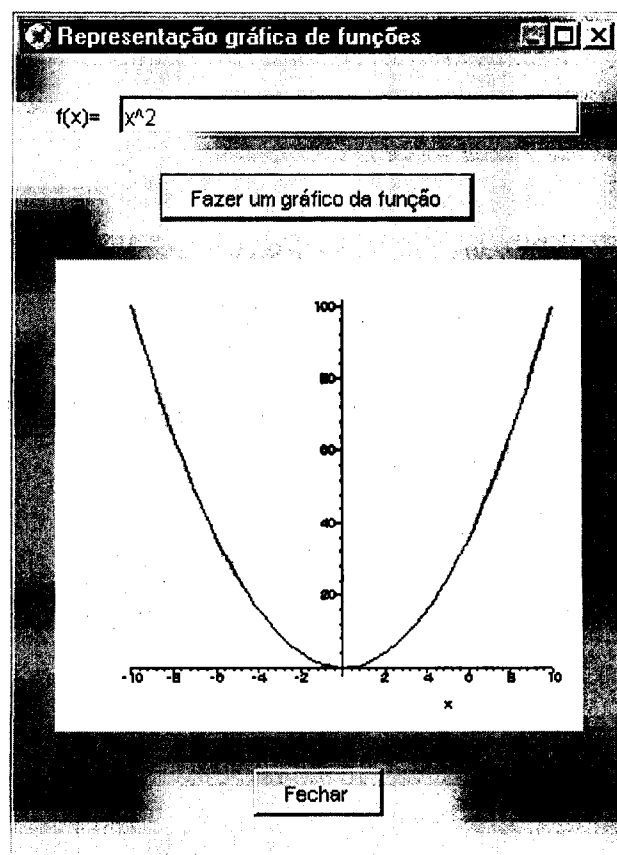


Figura 3.18: Gráfico simples 1

ao tamanho da janela.

Compete ao autor da Maplet dar as possibilidades que julga necessárias ao utilizador. Portanto, a Maplet anterior pode muito bem ser melhorada, podendo-se incluir mais opções para o utilizador escolher, mas cabe ao autor permitir tais opções. Um dos desafios deste trabalho foi construir uma Maplet de representação gráfica e estudo de funções que fosse relativamente completa ao nível das opções disponibilizadas ao utilizador. No capítulo 5 deste trabalho, essa Maplet é explicada mais pormenorizadamente. Entretanto, podemos ver um exemplo um pouco melhorado da Maplet inicial sobre representação gráfica de funções (fig. 3.19).

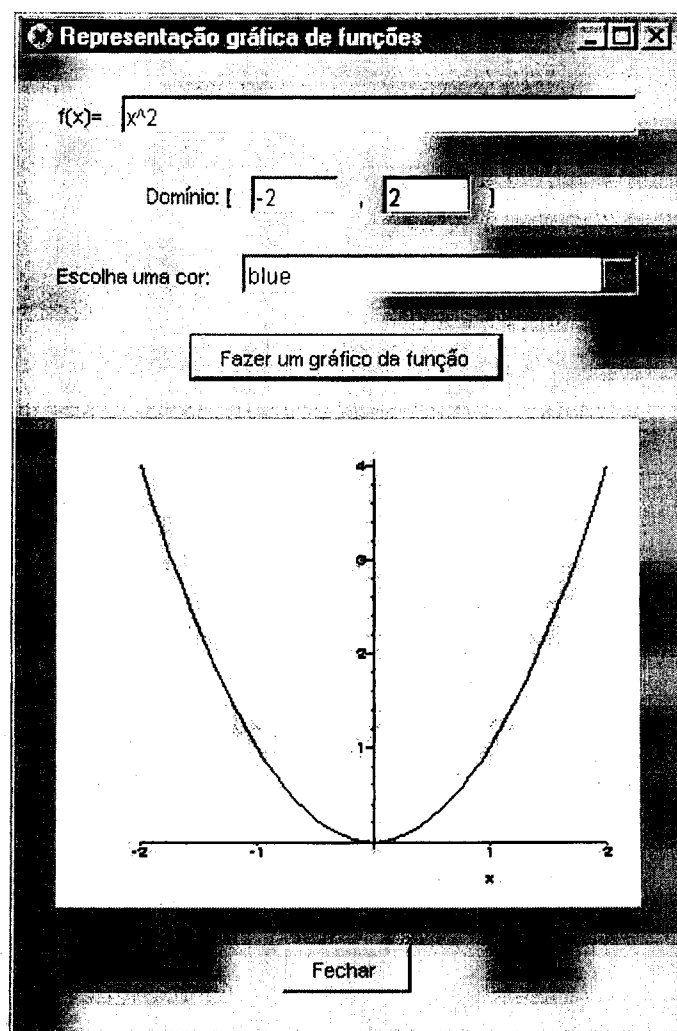


Figura 3.19: Gráfico simples 2

Nesta Maplet (fig. 3.19), o utilizador já tem mais algumas opções. Relativamente à escolha da cor, é perfeitamente possível traduzir as cores para português e tornar esta Maplet mais acessível aos nossos alunos.

Infelizmente, a única interactividade que podemos ter com o gráfico só pode ser feita através de botões ou outros elementos da Maplet. Não é possível interagir com o gráfico

utilizando o ponteiro do rato sobre a imagem. Mas esta é somente a primeira versão do Maple (Maple 8) que contém Maplets de raiz (No Maple 7, as Maplets surgiram mais tarde como um package adicional, onde era necessário fazer um *upgrade*). É natural que em futuras versões tal seja possível.

3.4 Outras Maplets

Para terminar este capítulo, vamos apresentar Maplets, cuja finalidade não é o auxílio à resolução de problemas de matemática, mas Maplets que se situam mais numa vertente lúdica para o utilizador. Pretende-se mostrar que utilizando o código Maple, se podem construir Maplets para os mais diversos fins.

Como exemplo concreto, apresentamos o conhecido jogo '*Mastermind*' (fig. 3.20). Este jogo consiste em tentar adivinhar uma sequência de quatro cores escolhida ao acaso. À medida que o jogador vai tentando colocar as cores correctas pela ordem correcta, são dadas pistas que indicam se o jogador está a colocar as cores correctas e pela ordem correcta.

Se aparecer uma marca branca, significa que o jogador colocou uma cor correcta mas na ordem errada. Se aparecer uma marca preta, significa que o jogador colocou a cor correcta na sua ordem. Se não aparecer marca, isto significa que o jogador não escolheu a cor certa.

O jogo termina quando for "achada" a sequência certa, ou seja, quando existirem quatro marcas pretas. A dificuldade do jogo é que as marcas brancas e/ou pretas não indicam a posição onde se encontra a cor para a qual dão a informação.

Para iniciar o jogo, o utilizador deve seleccionar no menu 'Jogo', a opção 'Novo jogo'. Depois é só escolher as cores que deseja para cada posição e pressionar o botão 'Tentar'.

A partir da informação dada pelas marcas brancas e/ou pretas, o jogador pode modificar a sequência até conseguir encontrar a solução final.

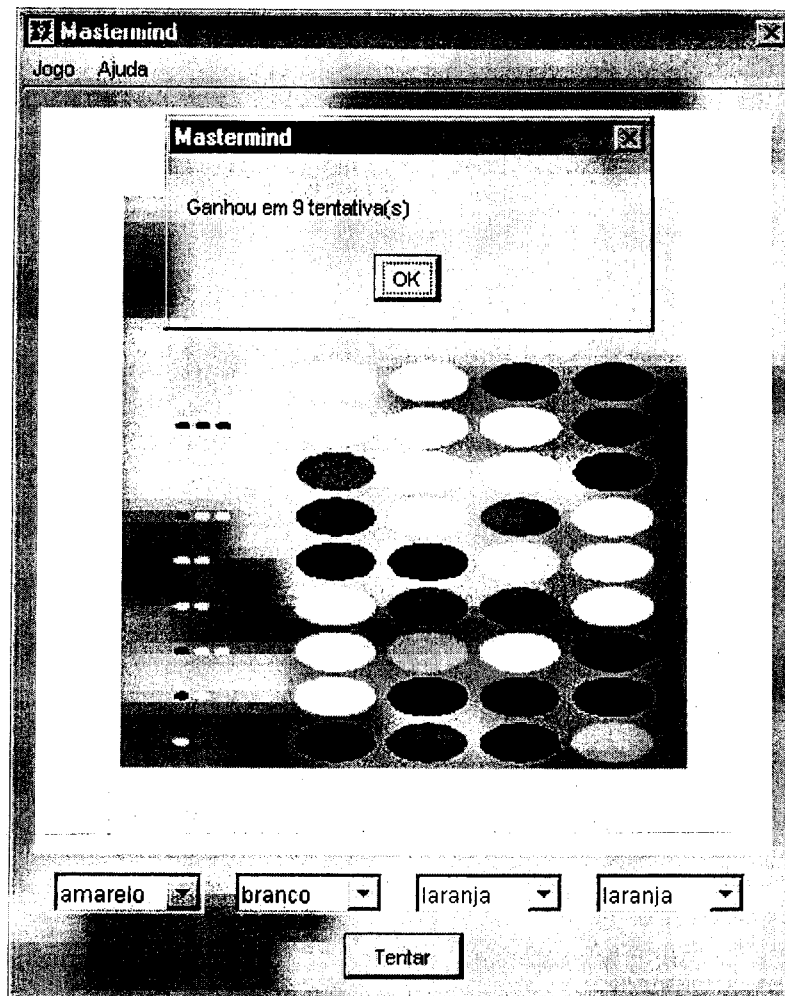


Figura 3.20: Mastermind

Quando o jogador consegue chegar à solução final, abre-se uma janela de aviso, que indica o número de tentativas do jogador para chegar à sequência final.

Esta é uma de uma lista de Maplets dedicada aos jogos, disponíveis na página da internet [23] 'www.mapleapps.com/categories/maplets/html/maplets_games.shtml'. Nesta lista, destaca-se ainda o conhecido jogo da batalha naval (fig. 3.21).

Neste jogo, depois de se colocarem os navios no tabuleiro, disparam-se os tiros nas coordenadas desejadas de forma a atingir os navios do tabuleiro do computador. Depois de serem escolhidas as coordenadas, clica-se no botão 'FOGO!'. A seguir clica-se no botão 'Vez do Computador', para dar a vez ao computador para disparar para o tabuleiro do jogador. Repetem-se estes passos até ser encontrado um vencedor. Uma quadrícula branca significa que se falhou o tiro, uma quadrícula vermelha significa que se acertou e quando as quadrículas ficam pretas, significa que se afundou o navio.

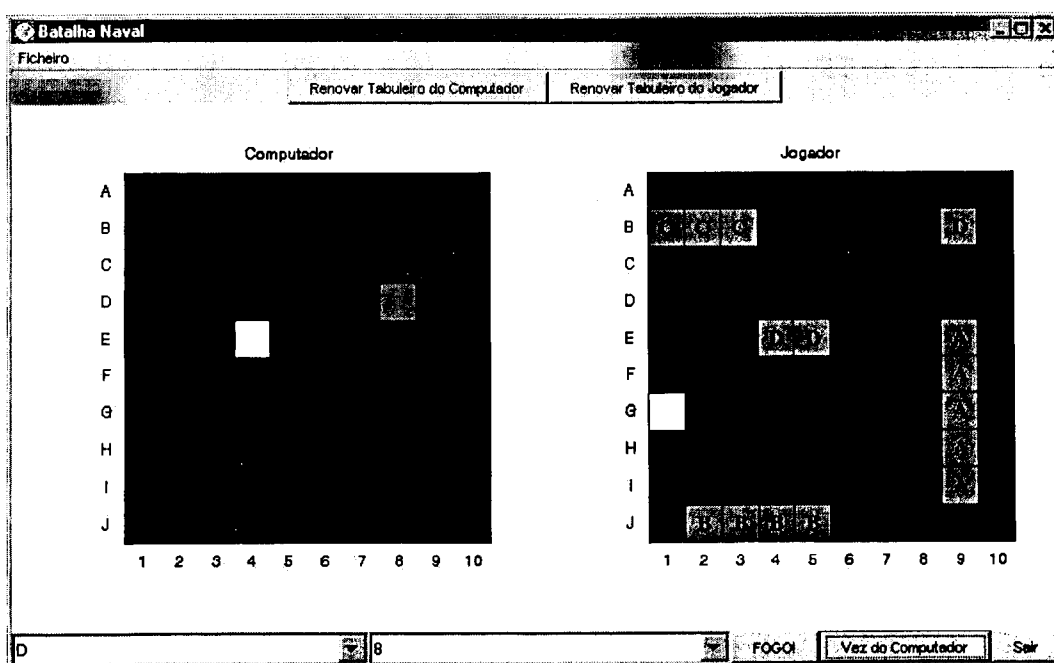


Figura 3.21: Batalha Naval

Outros exemplos de Maplets podem ser vistos ao longo deste trabalho, na página da internet: [22] www.mapleapps.com/categories/maplets/html/maplets_intro.shtml e no Help do Maple.

Capítulo 4

Construção de Maplets

Vamos agora descrever os elementos que estão presentes no módulo das Maplets e que com comandos básicos do Maple, as permitem pôr em funcionamento com toda a interactividade descrita no capítulo anterior. Far-se-á uma descrição dos elementos que podemos encontrar, para que depois no capítulo 5, se possam mostrar como os vários elementos podem interagir, de modo a produzir programas interactivos de alguma complexidade.

4.1 Maplets - Um módulo do Maple

As Maplets fazem parte de um módulo (*package*) de ferramentas do Maple. Este *package* inclui a função **Display**, e três *sub-packages*: **Elements** (Elementos), **Examples** (Exemplos) e **Tools** (Ferramentas).

A função **Display** é necessária para mostrar ou executar uma Maplet definida. O *sub-package* **Elements**, contém os elementos necessários para definir uma janela Maplet. Elementos simples como botões, barras de menus, caixas de texto, etc. Uma variedade de diferentes elementos são usados no *sub-package* **Maplets[Examples]**. Cada elemento tem uma página de ajuda, descrevendo várias opções. O *sub-package* **Examples**, contém Maplets pré-definidas que podem ser usadas separadamente ou com outras Maplets. Estes exemplos mostram como as Maplets devem ser usadas. Os exemplos mais complicados

têm associados uma página de ajuda e uma *worksheet* descrevendo a particularidade com que as Maplets foram construídas. Estas *worksheets* levam-nos das Maplets mais simples às mais complexas. Finalmente, o *sub-package* Tools, contém rotinas e procedimentos para podermos manipular e interagir com Maplets.

Uma Maplet é criada, listando-se uma série de elementos. O principal elemento dessa listagem é: `Maplet()`.

Existem várias opções para estruturar uma Maplet. Se quisermos criar uma Maplet sem qualquer formatação, podemos simplesmente especificar os elementos como listas, isto é, encaixados entre parêntesis rectos '[]'. No entanto, existem vários comandos para criar Maplets com uma formatação mais avançada e mais funcional. Quando uma Maplet está a ser executada, não é possível mexer na *worksheet* do Maple ao mesmo tempo. No entanto, depois de fechar a janela, a Maplet é encerrada, e podemos voltar a trabalhar normalmente na nossa sessão do Maple (*worksheet*).

4.2 Construção de uma Maplet simples

Como qualquer *package* do Maple, o *package* Maplets deve ser carregado com o comando **with** antes de iniciarmos a criação de uma Maplet. Como o *sub-package* Elements contém as rotinas necessárias para definir uma Maplet, é suficiente carregar este *sub-package*:

```
>with(Maplets):
```

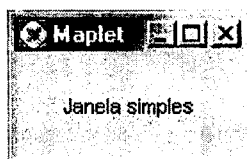
O primeiro exemplo, chamado MapletSimple usa o elemento principal `Maplet()` e a frase 'Janela simples' fica encaixada entre parêntesis rectos. Para mostrar a janela devemos usar o comando

```
>Display(Nome da Maplet);
```

De facto, ao construir uma Maplet, devemos começar por lhe dar um nome atribuindo-se o elemento **Maplet**. Todos os outros elementos que constituem a janela devem estar entre os parêntesis curvos do elemento principal **Maplet()**. Ao finalizar esta instrução com dois pontos (:), o Maple memoriza a estrutura da Maplet. Para inicializar a janela interactiva (em ambiente JAVA), devemos escrever outra linha de entrada, onde figure a função **Display**. Só com esta instrução é que o Maple inicia a janela. Portanto, a construção desta primeira Maplet deverá conter as seguintes instruções:

```
> restart:  
> with(Maplets[Elements]):  
> MapletSimples:=Maplet(["Janela simples"]):  
> Maplets[Display](MapletSimples);
```

Initializing Java runtime environment.



O '**restart:**' serve para "limpar" a memória do Maple, o '**with(Maplets[Examples]):**' serve para "chamar" o *subpackage* 'Elements' dentro do *package* 'Maplets', a instrução '**MapletSimples:=Maplet(["Janela simples"]):**' é a atribuição da nossa Maplet ao nome 'MapletSimples' e o '**Maplets[Display](MapletSimples);**' é a instrução para "exibir" a janela. Ao executar este último comando pela primeira vez numa sessão de Maple, é mostrada a mensagem '*Initializing Java runtime environment*'. Isto significa que vão ser accionados os mecanismos necessários para executar programas escritos na linguagem JAVA. De facto, estas Maplets são compiladas pelo Maple na linguagem JAVA, que actualmente pode ser considerada uma "linguagem universal", uma vez que

ela existe em sistemas operativos como o Windows, o Unix e Apple Macintosh. Esta universalidade também se traduz no facto de todas as interfaces das Maplets terem a mesma forma nos vários sistemas operativos que correm linguagem JAVA.

Nesta janela, apenas foi utilizado o elemento **Maplet**, que é o principal elemento das janelas. Todas as Maplets que se constroem devem começar com este elemento que contém todos os elementos que definem uma janela. Se houver procedimentos a serem utilizados, estes devem ser escritos "fora" do elemento **Maplet**.

Na secção 'Maplets[Elements]', veremos com maior pormenor todos os elementos que uma Maplet pode ter.

4.3 Terminologia

Ao longo do capítulo da construção de Maplets, aparecerão alguns termos para os quais é útil saber bem a sua definição geral. É de referir que a terminologia segue muitas vezes a notação original na língua inglesa.

Maplet

Uma Maplet é uma colecção de elementos, tais como janela, *layout*, janela pré-definida (*dialog*), e elementos de comando. Uma Maplet contém janelas (feitas exclusivamente pelo autor da Maplet) e *dialogs* (janelas pré-definidas).

Autor da Maplet

É um programador que usa o código do Maple para construir uma Maplet.

Utilizador da Maplet

O utilizador da Maplet é quem usa e interage com uma Maplet.

Layout

O *layout* define como os elementos de uma Maplet são visualmente organizados.

Janela

Não deve ser pensada como uma Maplet, mas sim como um elemento de uma Maplet.

Elementos da janela

Definem a janela de uma Maplet. A janela de uma Maplet consiste de uma barra para o título, de barras de menu opcionais e da parte principal da janela (*window body*).

Elementos do corpo da janela (*window body elements*)

São os elementos que são visíveis na janela além do título e barra de menus, tais como botões, caixas de texto, caixas para gráficos, etc. Podemos-nos referir a eles como os elementos da parte principal da janela.

Dialog

Um elemento do *dialog* tem *layout* pré-definido. Num *dialog*, o autor da Maplet apenas pode especificar texto.

4.4 Maplets[Elements]

Cada elemento pertence a uma das sete seguintes categorias:

- (i) Elementos do corpo da janela (*window body*).
- (ii) Elementos do *layout*.
- (iii) Elementos da barra de menus (*menu bar*).
- (iv) Elementos da barra de botões (*tool bar*).
- (v) Elementos de comando.

(vi) Elementos do *dialog*.

(vii) Outros elementos.

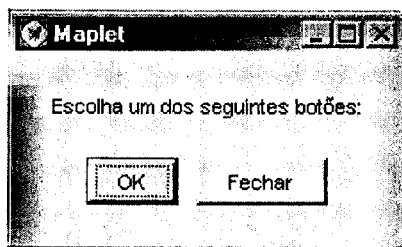
O mais significativo elemento que pode ser incluído numa Maplet é o elemento '**Window**' (Janela), pois pode conter elementos do corpo da janela, do *layout* e elementos de comando.

(i) Elementos do Corpo da Janela

Button(Botão): Define o botão que pode aparecer na janela de uma Maplet. A cada botão pode ser associado um elemento de acção, que é executada quando se clica no botão. O texto, o tipo de letra, a cor e outras propriedades do '**Button**' podem ser modificadas.



```
> with(Maplets[Elements]):
Botoes:=Maplet(["Escolha um dos seguintes botões:"],
[Button("OK",Shutdown("Executou a
janela")),Button("Fechar",Shutdown())]):
> Maplets[Display](Botoes);
```



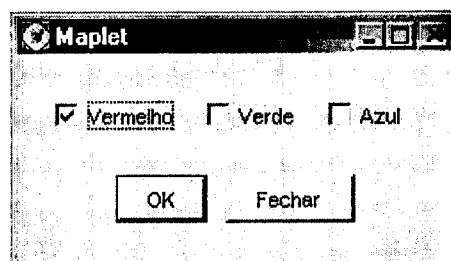
Se clicarmos no botão 'OK', a janela fecha e aparece a mensagem '"Executou a janela"' no *output* do Maple. Se clicarmos no botão 'Fechar', a janela fecha e não acontece mais nada.

CheckBox(caixa de selecção): Mostra uma caixa na janela da Maplet que ao ser seleccionada, fica com uma marca de selecção. Tais como os botões, a cada 'CheckBox' pode ser associado um elemento de acção, que é executada quando a caixa é seleccionada.



```
> Seleccao:= Maplet([
  [CheckBox[CB1]("Vermelho"), CheckBox[CB2]("Verde"),
  CheckBox[CB3]("Azul")],
  [Button("OK", Shutdown([CB1, CB2, CB3])), Button("Fechar",
  Shutdown())]
]):

> Maplets[Display](Seleccao);
```



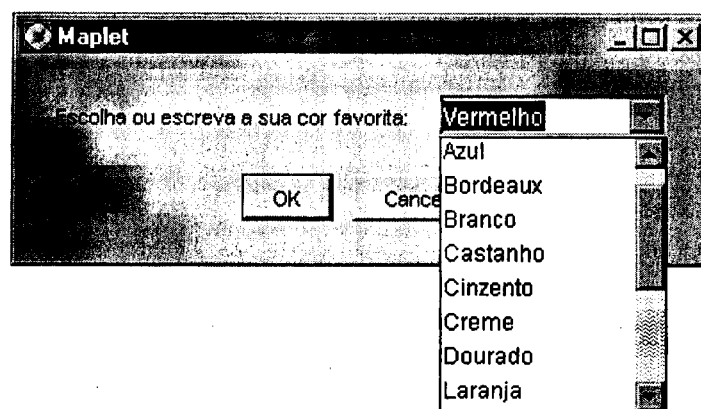
Se seleccionássemos 'Azul', este ficava com a marca e o 'Vermelho' deixava de ter marca. Ao clicar no botão 'OK', a janela fecha e no *output* do Maple aparece 'true' para a opção seleccionada e 'false' para outras opções. Neste caso, o *output* seria:

```
["true", "false", "false"]
```

ComboBox: Mostra uma caixa, onde o utilizador pode seleccionar uma opção de uma lista pré-definida, ou então, escrever uma opção que esteja ou não na lista. A opção inicial pode ser especificada, ou então, por defeito, é escolhida a primeira opção da lista pré-definida. O autor da Maplet pode definir uma lista de opções usando o elemento **Item** ou simplesmente criando uma lista com parêntesis rectos.

```
> combo:= Maplet([
  [
    "Escolha ou escreva a sua cor favorita:",
    ComboBox[C1]("Vermelho", sort(["Amarelo",
    "Azul", "Bordeaux", "Branco", "Castanho", "Cinzento",
    "Creme", "Dourado", "Laranja", "Lilás",
    "Magenta", "Prateado", "Preto", "Rosa",
    "Roxo", "Verde", "Vermelho"
  ]))
  ],
  [Button("OK", Shutdown([C1])), Button("Cancelar", Shutdown())]
]):

> Maplets[Display](combo);
```

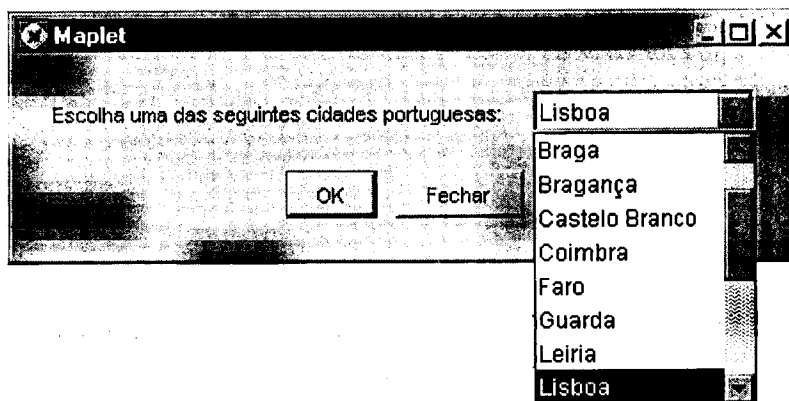


Ao clicar em 'OK' sai a opção escolhida no *output* do Maple.

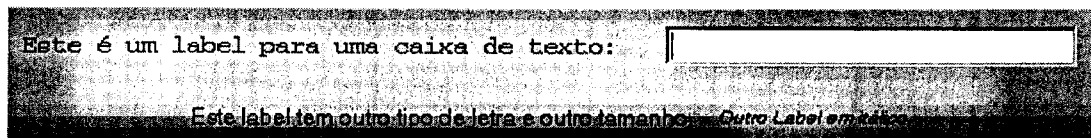
DropDownBox: Uma 'DropDownBox' é parecida com uma 'ComboBox', mas com a particularidade de não se poder escrever nenhuma opção, isto é, apenas se pode seleccionar uma opção que esteja na lista pré-definida. Este tipo de caixas são úteis quando se deseja que os utilizadores possam escolher uma opção de uma lista finita de opções. Para ver exemplos destes dois tipos de caixas, podemos comparar as Maplets 'VectorNorm' e 'MatrixNorm' disponíveis no *package* Maplets[Examples] [LinearAlgebra].

```
> dropdown:= Maplet([
  [
    "Escolha uma das seguintes cidades portuguesas:",
    DropDownBox[DD1]("Lisboa", sort(["Aveiro",
    "Beja", "Braga", "Bragança", "Castelo Branco", "Coimbra",
    "Évora", "Faro", "Guarda", "Leiria", "Lisboa",
    "Portalegre", "Porto", "Santarém", "Setúbal", "Viana do
    Castelo",
    "Vila Real", "Viseu"
  ]))
  ],
  [Button("OK", Shutdown([DD1])), Button("Fechar", Shutdown())]
]):

> Maplets[Display](dropdown);
```

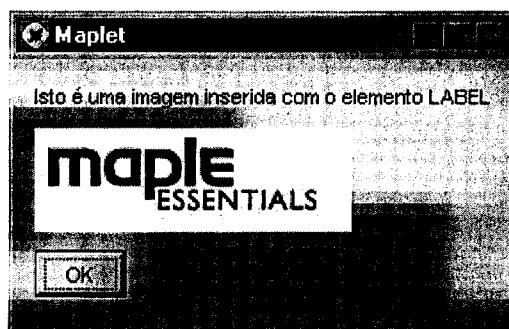


Label: Pode conter uma linha de texto ou uma imagem. O elemento "Label" (texto ou imagem) numa Maplet que está a ser executada, não pode ser seleccionado ou copiado. Para poder fazer isso deve ser usado o elemento `TextField` ou `TextBox` definidos mais à frente.



```
> labeling:= Maplet( [Label("Isto é uma
    imagem inserida com o elemento
    LABEL"),Label(Image("C:\\maplejpeg.jpg")), Button("OK",
    Shutdown())]
):

> Maplets[Display](labeling);
```

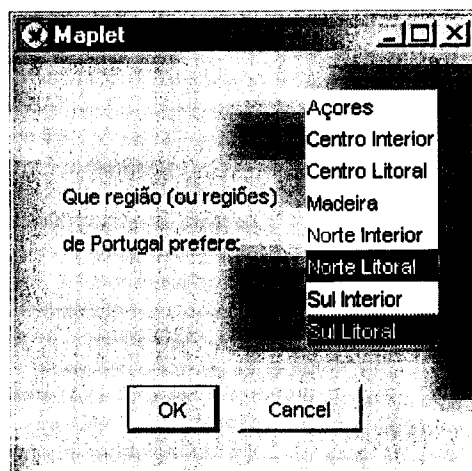


Ao inserir a imagem que queremos, devemos indicar o caminho para chegar ao ficheiro desejado. Neste caso, a imagem estava na raiz do disco C.

ListBox: Uma "ListBox" é parecida com uma "DropDownBox", em que o utilizador escolhe uma opção de uma lista pré-definida de opções, mas neste caso pode fazer mais do que uma escolha, ou seja, pode seleccionar-se mais do que uma opção. As teclas "Shift" e "Control" do teclado devem ser usadas para fazer as várias escolhas. O resultado pode ser convertido numa lista, usando a função `ListBoxSplit` do *sub-package* `Maplets[Tools]`.

```
> lista:= Maplet([
  [
    "Que região (ou regiões) de Portugal prefere:",
    ListBox[L1](sort(["Açores", "Madeira", "Norte Litoral", "Norte
    Interior",
    "Centro Litoral", "Centro Interior", "Sul
    Litoral", "Sul Interior"]
  ))
  ],
  [Button("OK", Shutdown([L1])), Button("Fechar", Shutdown())]
]):

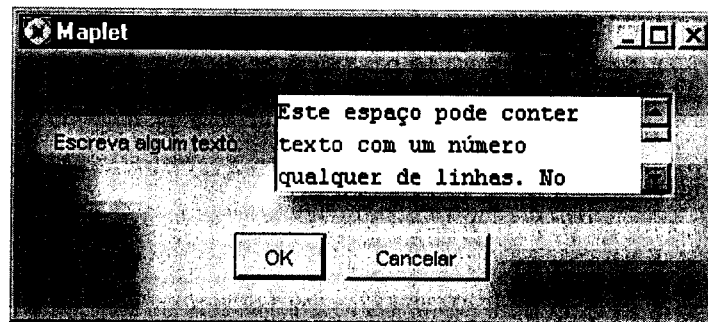
> Maplets[Display](lista);
```



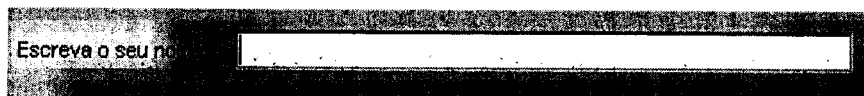
TextBox: É uma caixa para múltiplas linhas de entrada de texto ou para mostrar texto tal como se faz com "Label". Se a opção 'editable' for igual a 'true', a caixa funciona como entrada de texto, que pode ser modificada, seleccionada, copiada, apagada, etc. Se a opção 'editable' for igual a 'false', não pode ser inserido texto, ou seja, o autor usa-a apenas como um "Label" e o utilizador não poderá apagar ou modificar o texto do autor. Por defeito, a opção 'editable' é igual a 'true'. Com esta opção, é possível abrir um menu extra clicando no botão direito do rato. Este menu (*pop-up menu*), dá acesso às opções 'Copy' (Copiar), 'Paste' (Colar), 'Delete' (Apagar), 'Select All' (Seleccionar

tudo) e 'Clear' (Limpar). Se tivermos 'editable'='false', o 'pop-up menu' só dá acesso às opções 'Copy' e 'Select All'. Podem ser adicionadas outras opções no 'pop-up menu' usando a opção 'popupmenu'.

```
> caixatexto:=Maplet([
  ["Escreva algum texto:", TextBox[TB](height=3,width=30)],
  [Button("OK", Shutdown([TB])), Button("Cancelar",
    Shutdown())]
]):
> Maplets[Display](caixatexto);
```

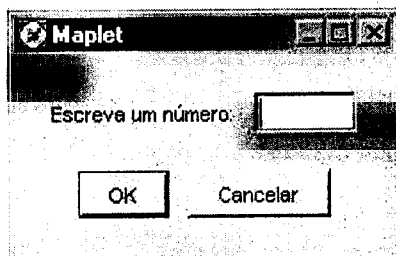


TextField: Tem a mesma função de uma "TextBox", só que utiliza apenas uma linha para o texto.



É possível escolhermos o tamanho da caixa de texto com a opção 'width'.

```
> linhatexto:=Maplet([
  ["Escreva um número:", TextField[TF](width=5)],
  [Button("OK", Shutdown([TF])), Button("Cancelar",
    Shutdown())]
]):
> Maplets[Display](linhatexto);
```



MathMLEditor: Com este elemento, podemos editar expressões no formato MathML, ou seja, expressões escritas em notação matemática em vez da sintaxe do Maple. Por exemplo, para escrever a raiz quadrada de um número, em vez de usarmos a função do Maple '`sqrt()`', clicamos no botão direito do rato e é accionada uma barra de ferramentas que contém o símbolo da raiz quadrada. Clicando no símbolo de raiz quadrada, este é inserido no editor e obtemos uma expressão com a verdadeira notação matemática. Um elemento MathMLEditor não pode conter outros elementos. Para a expressão sair para o Maple com a sintaxe do Maple, deve utilizar-se o comando `MathML[Import]()`.

```
> mat:= Maplet([
  ["Escreva uma expressão"],
  [MathMLEditor[ME]()],
  [Button("OK", Shutdown([ME]))]
]):
```

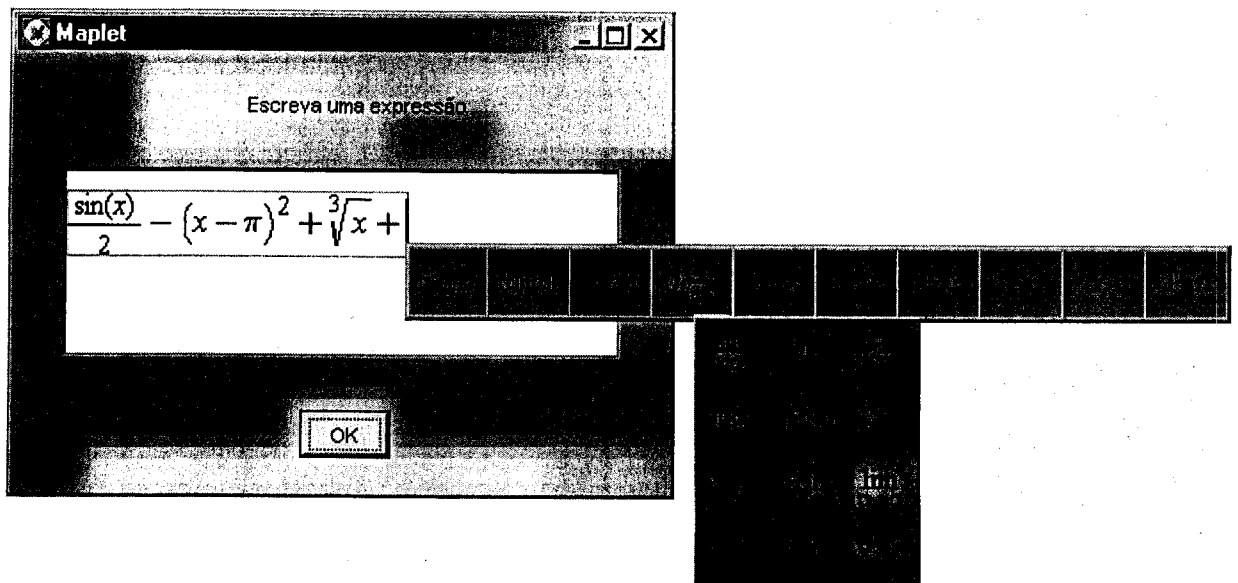
```
> resultado:=Maplets[Display](mat);
```

```
resultado := [ "<math><apply><plus/><apply><minus/><apply><divide/><apply \
><sin/><ci>x</ci></apply><cn>2</cn></apply><apply><power/><apply \
><minus/><ci>x</ci><ci>&pi;</ci></apply><cn>2</cn></apply></apply \
><apply><root/><degree><cn>3</cn></degree><ci>x</ci></apply><app \
ly><diff/><ci>x</ci></apply></apply></math>" ]
```

```
> MathML[Import](resultado[1]);
```

$$\frac{1}{2} \sin(x) - (x - \pi)^2 + \text{surd}(x, 3) + D(x)$$

Depois de executar '`resultado:=Maplets[Display](mat);`', é activada a seguinte janela:

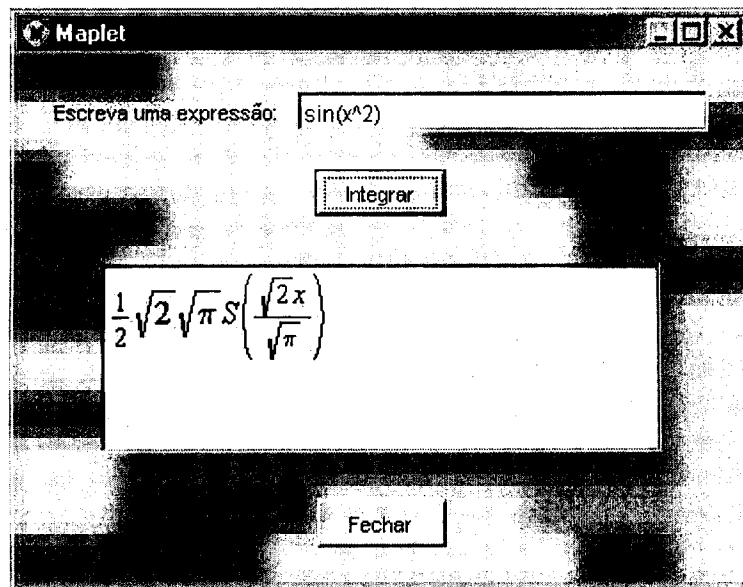


Depois de fechar a janela, fica guardado em 'resultado', o conteúdo de 'ME'. Para se ver este conteúdo na linguagem do Maple, teve de se fazer `'MathML[Import](resultado[1]);'`. O `import`, lê o que está no editor *MathML* e transforma em linguagem Maple. Também existe o `export`, que faz exactamente o inverso, ou seja, lê a linguagem Maple e transforma na linguagem usual da matemática através do *MathML*.

MathMLViewer: Com este elemento, as expressões aparecem no formato *MathML*, ou seja, as expressões são vistas num formato de notação matemática em vez da sintaxe do Maple. Por exemplo, o símbolo da raiz quadrada aparece em vez da função 'sqrt' do Maple.

```
> viewer:=Maplet([
  ["Escreva uma expressão:", TextField[Tex]()],
  [Button("Integrar", Evaluate(Mv='int(Tex,x)'))],
  [MathMLViewer[Mv]()],
  [Button("Fechar", Shutdown())]
]):

> Maplets[Display](viewer);
```



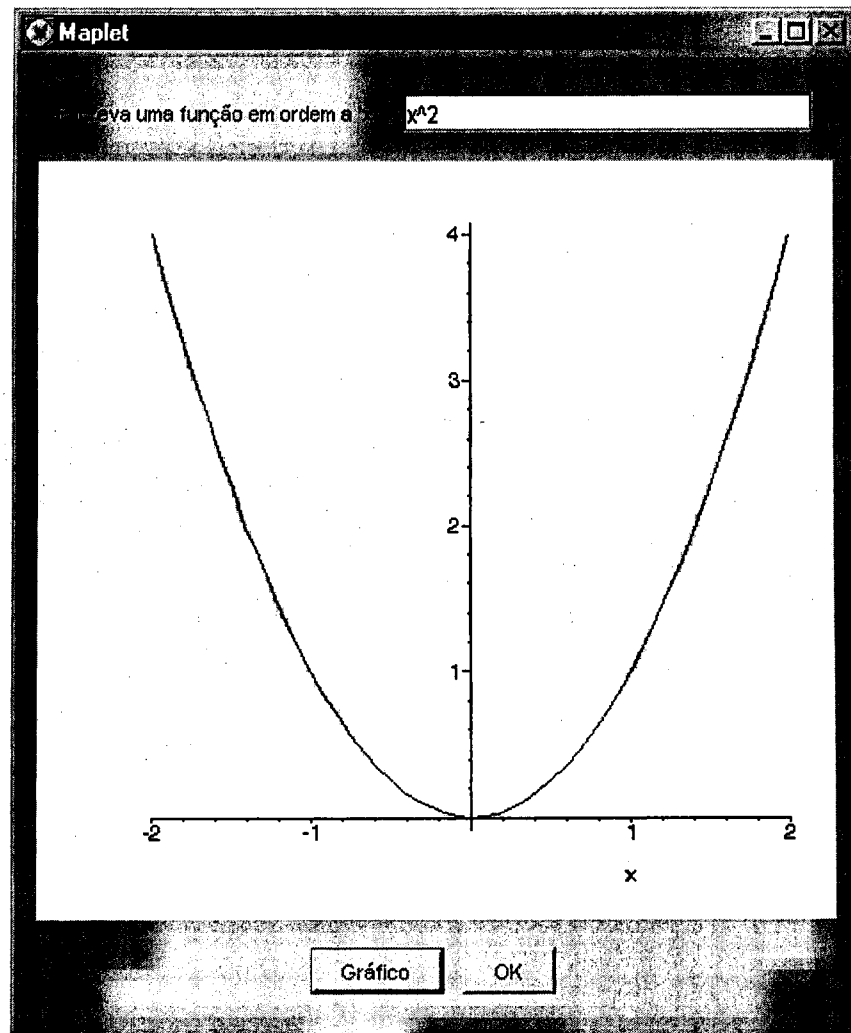
Inserimos a expressão 'sin(x²)' em linguagem Maple, e o integral desta expressão apareceu na linguagem usual da matemática numa caixa **MathMLViewer**. Foi usado o elemento **Evaluate** "dentro" do elemento **Button**, para ser executado o comando Maple incluído entre plicas (").

Plotter: Mostra gráficos 2-D ou 3-D a partir de um **plot** do Maple.

```
> grafico:=Maplet([
  ["Escreva uma função em ordem a 'x':", TextField[F1]()],
  [Plotter[P]()],
  [Button("Gráfico",
    Evaluate(P='plot(F1,x=-2..2)'), Button("OK", Shutdown())]
]):

> Maplets[Display](grafico);
```

Ao pressionar o botão 'Gráfico' na janela, o Maple executa 'plot(x²,x=-2..2)'. O elemento **Evaluate** é um dos elementos mais usados, e a sua definição será vista mais adiante.



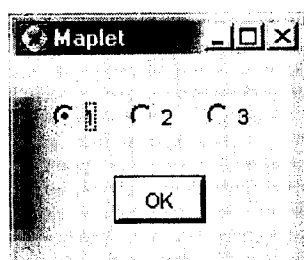
Mais à frente veremos um exemplo com gráficos animados no **Plotter**.

RadioButton: Um 'RadioButton' é parecido com uma 'CheckBox', mas tem a aparência de um círculo que é preenchido quando é seleccionado. Um 'RadioButton' não deverá aparecer isolado, mas sim agrupado com outros 'RadioButton's, pois deve funcionar como uma selecção de uma entre várias opções.



```
> radio:=Maplet([
  [RadioButton[R1]("1", true, group=G1), RadioButton[R2]("2",
    group=G1), RadioButton[R3]("3", group=G1)],
  [Button("OK", Shutdown([R1,R2,R3]))]
],
  ButtonGroup[G1]()
):

> Maplets[Display](radio);
```



A opção 'true' no **RadioButton** R1 significa que ao ser activada a janela este vai estar seleccionado e a opção 'group=G1' nos três **RadioButtons** significa que R1, R2 e R3 pertencem ao mesmo grupo de **RadioButtons**. Já "fora" do *layout* da Maplet, está o elemento **ButtonGroup**[G1]() que indica o grupo a que pertencem os **RadioButtons**.

Ao pressionar 'OK', sai no *output* do Maple:

```
["true", "false", "false"]
```

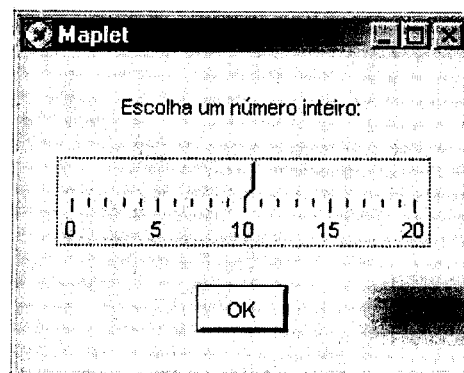
Slider: É um género de uma régua que permite ao utilizador escolher um valor inteiro num certo intervalo.

```

> slide:=Maplet([
  ["Escolha um número inteiro:"],
  [Slider[S1](10, 0..20, majorticks=5, minorticks=1,
    showticks)],
  [Button("OK", Shutdown([S1]))]
]):

> Maplets[Display](slide);

```



Nas opções do **slider** aparece o número '10'. Esta opção significa que a marca do **slider** vai aparecer no número 10 quando a janela se abrir. Outra opção é '0..20'. Isto significa que o valor mais pequeno é 0 e o maior é 20. A opção 'majorticks=5' significa que as marcas maiores no **slider** aparecem de 5 em 5 números e 'minorticks=1', significa que as restantes marcas aparecem de 1 em 1. A opção 'showticks' não era necessária, pois por defeito, esta opção é válida mesmo que não seja mencionada. Se o seu valor lógico for 'false' (falso), as marcas (pequenos traços perpendiculares à orientação do **slider**) não serão mostradas.

Table: Permite inserir dados em linhas e colunas. Os elementos **TableHeader**, **TableItem** e **TableRow** serão descritos mais à frente.

```

> L:= [sin(x), cos(x), tan(x), sec(x), csc(x), cot(x)]:

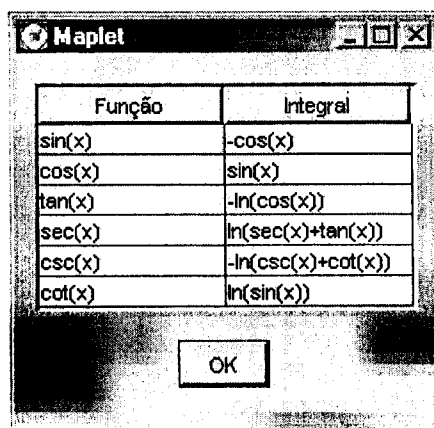
```

```

> tabela:=Maplet([
  BoxCell(Table(["Função", "Integral"], [seq( [i, int( i, x )],
    i=L)], width=200 ), as_needed),
  Button("OK", Shutdown())]
):

> Maplets[Display](tabela);

```

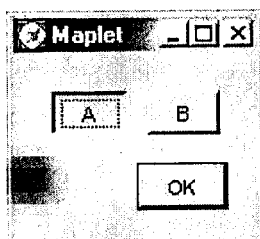


A lista L é a lista de algumas funções trigonométricas. Na Maplet 'tabela', o elemento **Table** aparece incluído no elemento **BoxCell** em vez de aparecer entre parêntesis rectos. Isto deve-se ao facto de se querer incluir na tabela uma linha de título. De facto, a linha de título só aparece na janela, se for possível aparecer um *scroll* (barra de deslocação) na parte lateral da tabela, quando esta tiver grandes dimensões. Os parêntesis rectos não permitem que se inclua a opção de *scroll*, pelo que é necessário que se use um dos seguintes elementos do *layout*: **BoxCell** ou **GridCell**, que serão definidos mais à frente. O elemento **Table** contém '["Função", "Integral"]', que é a linha de título e '[seq([i, int(i,x)], i=L)]', que são as linhas compostas pelos elementos da lista L e pelos respectivos integrais indefinidos. A opção 'width=200' do elemento **Table** serve para controlar a largura da tabela e a opção 'as_needed' do elemento **BoxCell** serve para adicionar um *scroll* sempre que for necessário.

ToggleButton: É parecido com uma 'CheckBox', excepto na aparência. É um botão que pode conter texto ou imagem.



```
> toggle:=Maplet( [
  [ToggleButton[TB1]( "A", true ), ToggleButton[TB2]( "B" )],
  Button("OK", Shutdown([TB1, TB2]))]):
> Maplets[Display](toggle);
```



A opção 'true' no elemento **ToggleButton** significa que ao activar a janela, o botão 'A' já estará pressionado. Para desactivar a selecção do botão, deve-se premir novamente o mesmo botão. Assim, podem estar seleccionados os botões que se desejar, nenhum ou todos.

Neste caso, ao pressionar em 'OK', a janela fecha-se e no output do Maple aparece:

```
[ "true", "false" ]
```

(ii) Elementos do Layout

O *layout* de uma janela descreve como os vários elementos de uma Maplet são posicionados. Existem duas formas de *layout* diferentes: o **BoxLayout** e o **GridLayout**. Pode-se ainda utilizar os parêntesis rectos para definir um *layout*, e foi o que até agora se utilizou nos exemplos de Maplets já vistos.

Os seguintes elementos são classificados como elementos de *layout*:

- **BoxLayout**
- **BoxRow**
- **BoxColumn**
- **BoxCell**
- **GridLayout**
- **GridRow**
- **GridCell**

BoxLayout: *Layout* em que se pode controlar onde os items aparecem horizontalmente ou verticalmente relativamente a outros elementos. Para o controlo horizontal, usa-se o elemento **BoxRow**, para o controlo vertical usa-se o elemento **BoxColumn**.

BoxRow: Organiza horizontalmente os elementos numa '**BoxLayout**'. O elemento **BoxRow** pode conter um número qualquer de **BoxCell**, **BoxColumn** ou **BoxRow**.

BoxColumn: Organiza verticalmente os elementos numa '**BoxLayout**'. O elemento **BoxColumn** pode conter um número qualquer de **BoxCell**, **BoxColumn** ou **BoxRow**.

BoxCell: O elemento **BoxCell** especifica a entrada numa **BoxColumn**, **BoxRow**, ou **BoxLayout**.

GridLayout: É o género de *layout* parecido com as entradas de uma tabela. Os elementos aparecem como se estivessem numa espécie de grelha.

GridRow: Organiza numa linha os elementos na '**GridLayout**'. O elemento **GridRow** pode conter um número qualquer de **GridCell**.

GridCell: O elemento **GridCell** especifica a entrada numa **GridRow**.

Listas encaixadas: São listas de listas que podem ser usadas para definir um *layout*. Por serem simples, todos os exemplos até agora vistos neste capítulo incluem um *layout* definido por listas encaixadas (parêntesis rectos).

Os elementos **BoxLayout**, **BoxRow**, **BoxColumn** e **BoxCell** podem substituir os parêntesis rectos. Isto é necessário quando se criam Maplets com *layouts* mais complicados, como por exemplo, aquelas em que as opções de margens e alinhamentos já estão definidas. Para além disso, estes elementos incluem opções que não podem ser usadas se utilizarmos apenas parêntesis rectos no *layout* da nossa Maplet. Vamos ver as diferenças nos códigos de Maplets que originam janelas idênticas nos seguintes exemplos:

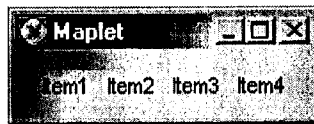
Exemplo usando parêntesis rectos:

```
> restart:
> with(Maplets[Elements]):
> MapletUm:=Maplet([
    ["Item1","Item2","Item3","Item4"]
]):
> Maplets[Display](MapletUm);
```

Exemplo usando os elementos **BoxLayout** e **BoxRow**:

```
> restart:
> with(Maplets[Elements]):
> MapletDois:=Maplet(BoxLayout(
    BoxRow("Item1","Item2","Item3","Item4")
)):
> Maplets[Display](MapletDois);
```

Tanto num como noutro a janela resultante é a mesma:

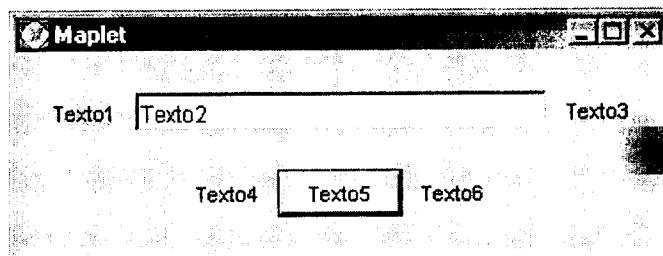


Vejamos agora a diferença entre uma Maplet com **BoxLayout** e uma Maplet com **GridLayout**:

```
> restart:

> with(Maplets[Elements]):
> layoutbox := Maplet(
  Window(layout = BL),
  BoxLayout[BL] (
    BoxColumn(
      BoxRow(BoxCell("Texto1"),
        BoxCell(TextField[Tb]("Texto2")),
        BoxCell("Texto3")),
      BoxRow(BoxCell("Texto4"),
        BoxCell(Button("Texto5", Shutdown([Tb]))),
        BoxCell("Texto6"))
    )
  )
):

> Maplets[Display](layoutbox);
```

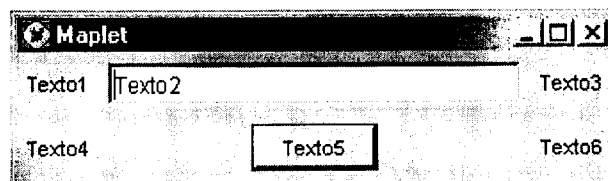


Se o *layout* for construído com **GridLayout**, a mesma janela fica com um aspecto diferente:

```
> restart:

> with(Maplets[Elements]):
> layoutgrid := Maplet(
  Window(layout = GL),
  GridLayout[GL](
    GridRow(GridCell("Texto1"),
      GridCell(TextField[Tg]("Texto2")),
      GridCell("Texto3")),
    GridRow(GridCell("Texto4"),
      GridCell(Button("Texto5", Shutdown([Tg]))),
      GridCell("Texto6"))
  )
):

> Maplets[Display](layoutgrid);
```



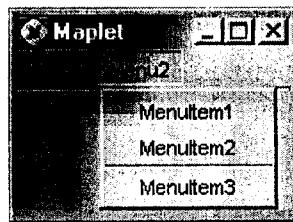
(iii) Elementos da Barra de Menus (Menu Bar)

Os seguintes elementos são classificados de elementos da barra de menus:

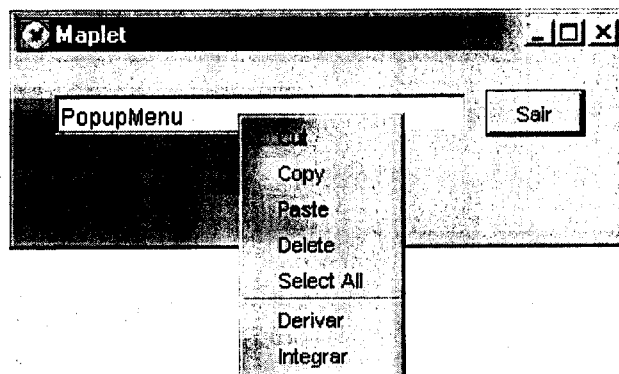
- **MenuBar** - Deve estar contido em 'Window' (janela).
- **Menu** - Deve estar contido em 'Menu', 'MenuBar' ou 'PopupMenu'.
- **MenuItem** - Deve estar contido em 'Menu' ou 'PopupMenu'.
- **MenuSeparator** - Deve estar contido em 'MenuBar' ou 'PopupMenu'.

- **PopupMenu** - Deve estar contido em 'TextField' ou 'TextBox'.

Uma barra de menus pode conter um número qualquer de menus. Cada menu pode conter items, definidos com o **MenuItem**, e submenus. Os separadores podem ser usados para distinguir diferentes grupos num menu.



O 'PopupMenu' é um menu extra que é activado quando se clica no botão direito do rato quando temos o cursor em cima de uma caixa de texto ou imagem. Por defeito, o 'PopupMenu' contém algumas opções de edição, mas podem ser-lhe adicionadas opções extra.



Neste caso foram adicionadas as opções 'Derivar' e 'Integrar'.

As opções dos menus, numa janela, funcionam como, por exemplo, os botões. A cada opção do menu está associada uma acção que será executada pelo Maple assim que esta for escolhida. As opções de um menu podem ser novos menus com outras opções. O seguinte exemplo, mostra-nos como é possível incluir menus "dentro" de menus:

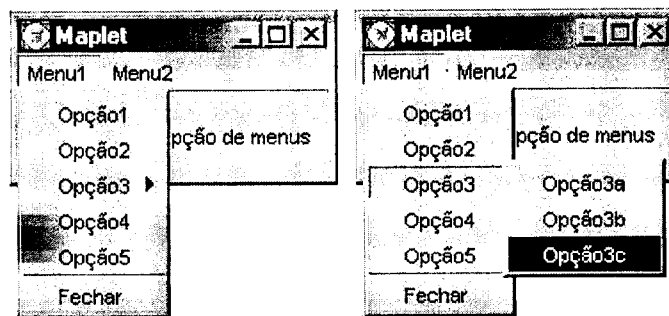
```

> restart:

> with(Maplets[Elements]):
> mapletmenus:=Maplet(Window(menubar=MB1,["Janela
com a opção de menus"]),
MenuBar[MB1](
Menu("Menu1",
MenuItem("Opção1",Shutdown("1")),
MenuItem("Opção2",Shutdown("2")),
Menu("Opção3",
MenuItem("Opção3a",Shutdown("3.1")),
MenuItem("Opção3b", Shutdown("3.2")),
MenuItem("Opção3c", Shutdown("3.3"))),
MenuItem("Opção4",Shutdown("4")),
MenuItem("Opção5",Shutdown("5")),
MenuSeparator(),
MenuItem("Fechar",Shutdown()))),
Menu("Menu2")
)):

> Maplets[Display](mapletmenus);

```



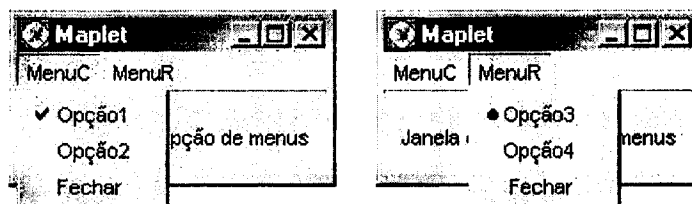
Nesta Maplet, a opção 'Opção3' é também um menu que inclui três opções. Entre a opção 'Opção5' e a opção 'Fechar', existe uma linha separadora, que é conseguida através do elemento `MenuSeparator`. Neste exemplo, todas as opções fecham a janela, diferenciando-se cada uma delas pelo resultado obtido no *output* do Maple. Assim, se por exemplo escolhermos a opção 'Opção3c', a janela fecha-se e no *output* do Maple sai "3.3".

São também elementos do 'MenuBar':

- **CheckBoxMenuItem** - É um item do menu que funciona como uma **CheckBox**. Quando este item é seleccionado, aparece uma marca que indica que a opção está activa. Quando clicamos lá de novo, a marca desaparece, o que indica que a opção está desactivada. Deve estar contido em 'Menu' ou 'PopupMenu'.
- **RadioButtonMenuItem** - Os elementos 'Menu' ou 'PopupMenu' devem conter mais do que um **RadioButtonMenuItem**. É um item do menu que funciona como um **RadioButton**. Quando um item é seleccionado, fica activado e os outros desactivados.

```
> cbrbmenus:=Maplet(Window(menubar=MB2,["Janela
com a opção de menus"]), MenuBar[MB2](Menu("MenuC",
CheckBoxMenuItem[C1]("Opção1",value=true),
CheckBoxMenuItem[C2]("Opção2"),
MenuItem("Fechar", Shutdown([C1,C2,R3,R4]))),
Menu("MenuR",
RadioButtonMenuItem[R3]("Opção3",group=BG,value=true),
RadioButtonMenuItem[R4]("Opção4",group=BG),
MenuItem("Fechar",Shutdown([C1,C2,R3,R4]))
), ButtonGroup[BG]())
):

> Maplets[Display](cbrbmenus);
```



Nesta Maplet, ao seleccionarmos 'Opção1', 'Opção2', 'Opção3' ou 'Opção4', estamos apenas a activar estas opções com os valores 'true' (verdadeiro) ou 'false' (falso), conforme

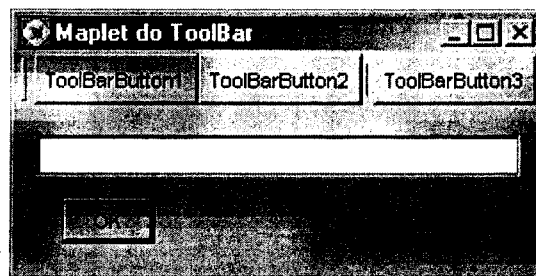
estas estejam ou não assinaladas com uma marca. Ao seleccionar 'Fechar', sai no *output* o resultado das nossas escolhas. Neste caso, estão seleccionadas as opções 'Opção1' e 'Opção3', pelo que o resultado seria ["true", "false", "true", "false"]. A diferença entre o 'MenuC' e o 'MenuR', é que no primeiro, com o elemento **CheckBoxMenuItem**, é possível escolher uma das opções, as duas opções ou nenhuma opção, enquanto que no segundo, com o elemento **RadioButtonMenuItem** só é possível escolher uma das opções.

(iv) Elementos da Barra de Botões (*Tool Bar*)

Uma barra de botões pode conter qualquer número de botões. Os botões podem ser agrupados em grupos diferentes se usarmos um separador que cria um espaço entre botões adjacentes.

Os seguintes elementos são classificados como elementos da barra de botões:

- **ToolBar** - Deve estar contido em 'Window' ou 'Maplet'.
- **ToolBarButton** - Deve estar contido em 'ToolBar'.
- **ToolBarSeparator** - Deve estar contido em 'ToolBar'.



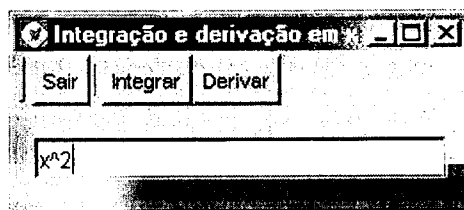
O seguinte Maplet é um exemplo da aplicação dos elementos da barra de botões:

```
> restart:
```

```

> with(Maplets[Elements]):
> maplet := Maplet(
  Window(title = "Integração e derivação em x", toolbar = ToolBar(
    ToolBarButton("Sair", Shutdown()),
    ToolBarSeparator(),
    ToolBarButton("Integrar", onclick=Evaluate(T = 'int(T, x)')),
    ToolBarButton("Derivar", onclick=Evaluate(T='diff(T,x)'))
  ),
  [
    TextField[T]()
  ])
):
> Maplets[Display](maplet);

```



O comando **Evaluate** é um dos elementos de comando que serão descritos a seguir. Tal como o **Shutdown**, o **Evaluate** é uma acção que é executada ao pressionar, por exemplo, um botão.

Neste caso, se pressionarmos em 'Integrar', o Maple através do elemento **Evaluate** executa '**int(T,x)**', em que 'T' é a expressão inserida na caixa de texto. Se pressionarmos em 'Derivar', é executado '**diff(T,x)**' e se pressionarmos 'Sair', a janela fecha através do elemento **Shutdown()**.

Mas, vamos ver como são definidos os elementos de comando:

(v) Elementos de Comando

Numa Maplet, um utilizador pode executar acções como clicar num botão, mudar um valor numa 'TextField' (caixa de texto), ou mudar uma opção numa 'ComboBox'. Por trás de cada uma destas acções está a execução de um comando. Cada comando é executado antes de o próximo se iniciar. Os elementos de comando não são visíveis numa Maplet. Os seguintes elementos são classificados como elementos de comando:

- **CloseWindow**
- **Evaluate**
- **RunDialog**
- **RunWindow**
- **SetOption**
- **Shutdown**

CloseWindow: Fecha a janela que está a ser executada.

Evaluate: O comando 'Evaluate' executa um procedimento do Maple com um dado conjunto de argumentos na sessão de Maple. No 'Evaluate' podem estar contidos elementos de 'Argument'.

RunDialog: Abre um elemento de 'Dialog'. O RunDialog abre uma janela do 'Dialog' (janela pré-definida) e aceita apenas a opção 'dialog' que é a referência à janela que vai ser executada, mas se essa referência já estiver a ser executada, nada acontece.

RunWindow: Abre uma janela (*Window*). O RunWindow abre uma janela e aceita apenas a opção 'window' que é a referência à janela que vai ser executada, mas se a janela referenciada já estiver a ser executada, nada acontece.

Nota:

Os comandos 'RunDialog' e 'RunWindow' são tratados separadamente, porque o elemento 'Maplets[Elements][Window]' é intrinsecamente diferente dos elementos de **Dialog**. Um **Dialog** tem uma estrutura pré-definida. Um autor de Maplets pode especificar as opções de um **Dialog**, mas não pode acrescentar elementos. Uma janela (**Window**) não tem uma estrutura pré-definida. O autor de Maplets pode especificar a sua estrutura usando elementos e opções. Eles também funcionam de maneira diferente. Por exemplo, uma janela (**Window**) pode ser minimizada.

SetOption: Permite que valores de certas opções possam ser modificados enquanto uma Maplet está a ser executada. Por exemplo, se o utilizador clica num botão, a opção 'onchange' do botão pode usar o elemento 'SetOption' para iniciar uma modificação, como por exemplo, o apagar de uma caixa de texto. O elemento **SetOption** é usado numa Maplet, enquanto que a rotina **Set** é usada num procedimento como será visto na secção 'Ferramentas (Tools)' mais à frente.

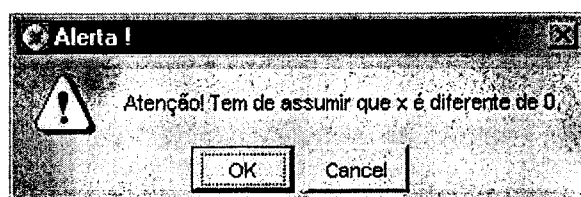
Shutdown: Fecha a Maplet que está a ser executada. Pode, opcionalmente, enviar um valor para a sessão de Maple (*worksheet*). Pode enviar valores específicos guardados numa Maplet, por exemplo, o conteúdo de uma caixa de texto ou um valor fixo.

Nos vários exemplos já vistos temos encontrado alguns destes elementos (**Shutdown**, **Evaluate**, etc.). Os restantes elementos serão melhor explicados à medida que forem aparecendo nos exemplos ao longo deste trabalho.

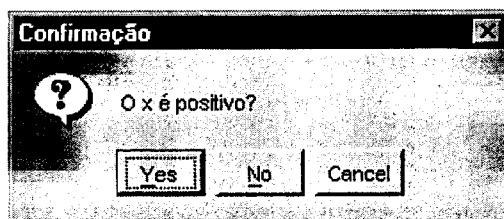
(vi) Elementos do Dialog

Os dialogs são pequenas janelas que fornecem informação, tais como mensagens de alerta ou avisos, que permitem escolher entre variadas cores ou que, por exemplo, dão a possibilidade de inserir o nome de um ficheiro. Os utilizadores respondem a um **Dialog**, clicando num botão. O autor apenas pode modificar algumas características específicas do *layout*, como por exemplo, o texto na barra de título ou o texto de um botão. Um **Dialog** é activado usando o elemento 'RunDialog'.

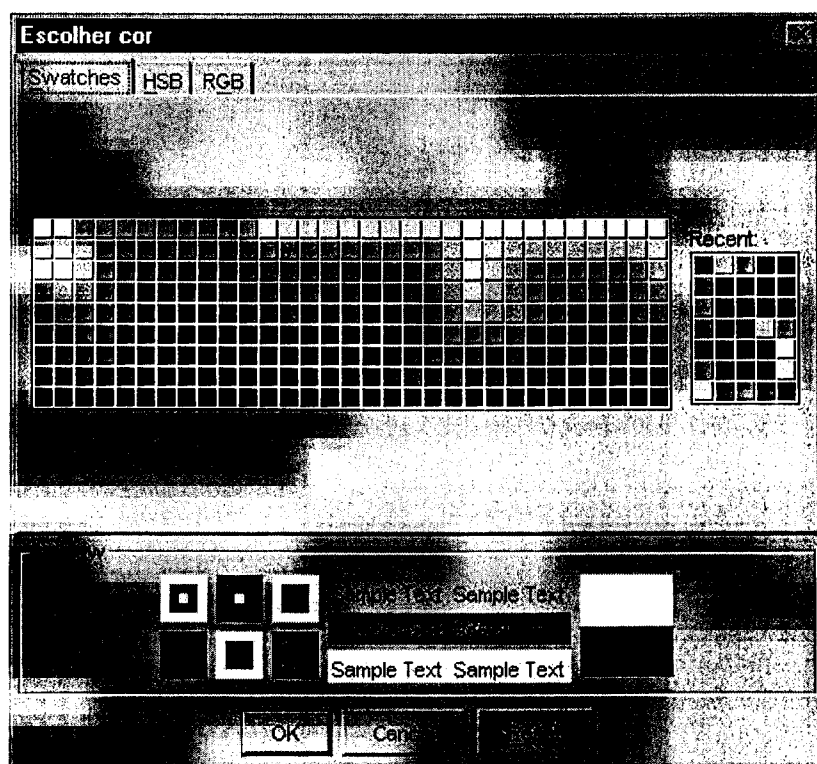
AlertDialog: É um aviso que dá atenção a um potencial problema. Permite ao utilizador ignorar o aviso e continuar (OK) ou ter em atenção o aviso (Cancel).



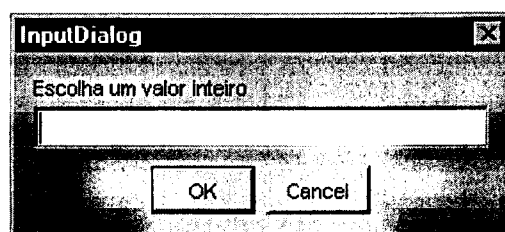
ConfirmDialog: Permite ao utilizador especificar se uma acção é ou não executada. A uma dada pergunta o utilizador responde Yes(Sim) e continua, No(Não) mas continua ou Cancel e não continua.



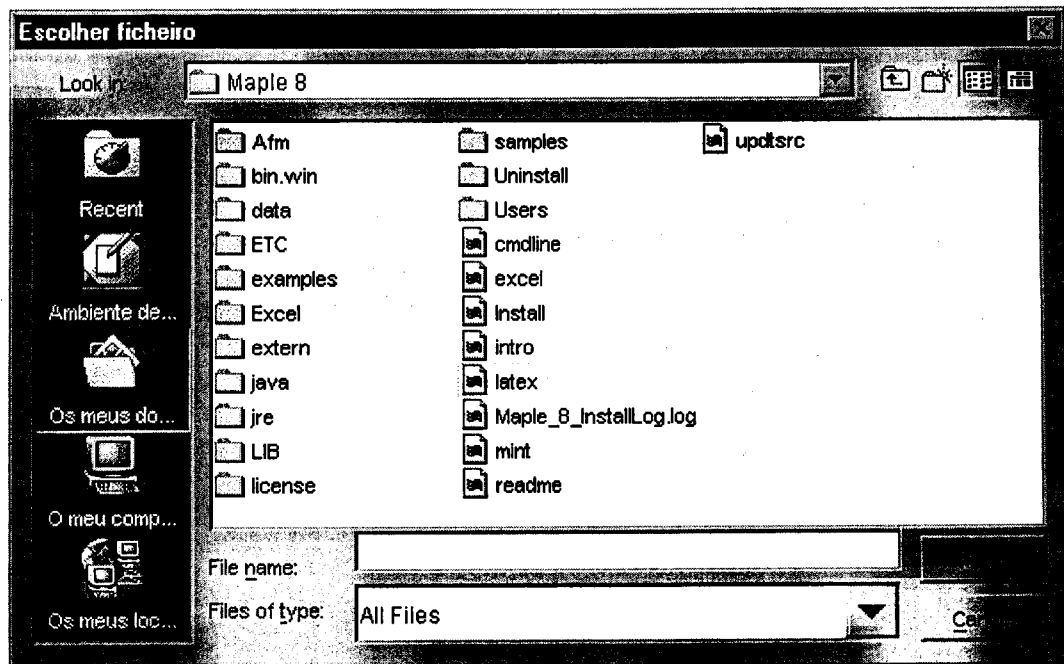
ColorDialog: Um mostruário de cores que podem ser escolhidas pelo utilizador. O utilizador pode escolher uma das 310 cores já definidas no menu 'Swatches' ou pode "fabricar" uma cor através do sistema 'HSB' ou 'RGB'.



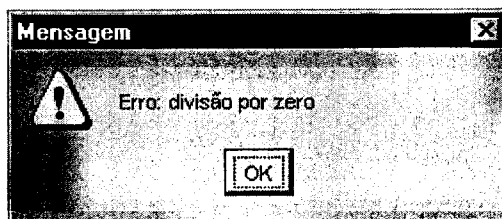
InputDialog: O **InputDialog** é parecido com o **AlertDialog**, só que este tem uma caixa de texto, onde o utilizador pode escrever ou modificar dados. Pode aparecer inicialmente um valor na caixa de texto, mas esta apareça.



FileDialog: Uma janela *standard* de abrir ficheiros.

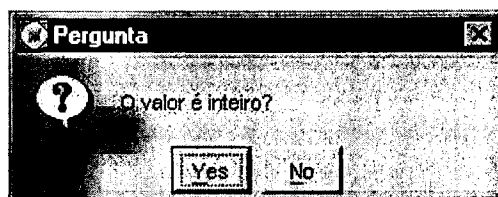


MessageDialog: Mostra ao utilizador uma mensagem de informação. A janela fecha quando o utilizador clica no botão Ok.



QuestionDialog: Coloca uma pergunta ao utilizador e permite que este responda 'Yes' (Sim) ou 'No' (Não).

Os exemplos que incluem estes elementos podem ser vistos através do *Help*, utilizando como tópico: 'Maplets,DialogElements'.



(vii) Outros Elementos

Action: Define uma acção numa Maplet. Contém qualquer número de elementos de comando ou outras 'Action' para serem executadas.

Argument: Especifica um argumento de uma função do Maple. Só pode ser usado no elemento **Evaluate**.

ButtonGroup: Muitas vezes é necessário associar botões de modo a que apenas um seja seleccionado. Se usarmos a opção 'group' combinada com o elemento **ButtonGroup**, tanto os botões **RadioButton** como os botões **ToggleButton** podem ser agrupados de modo a que só seja seleccionado um dos botões pertencentes ao grupo.

Font: Especifica um tipo de letra num elemento. Os tipos de letra disponíveis dependem do Sistema Operativo do computador.

Image: Especifica uma imagem do tipo *jpeg* ou *gif* numa Maplet.

Item: Especifica a entrada de uma **ComboBox**, **DropDownBox** ou **ListBox**. O elemento **Item** não pode conter outros elementos.

Maplet: Contém os elementos que definem uma Maplet. É o "elemento-mor" de uma Maplet.

Return: Agrupa um conjunto de valores que são enviados para a sessão de Maple (*worksheet*) quando uma Maplet é fechada. Pode conter qualquer número de elementos **ReturnItem**.

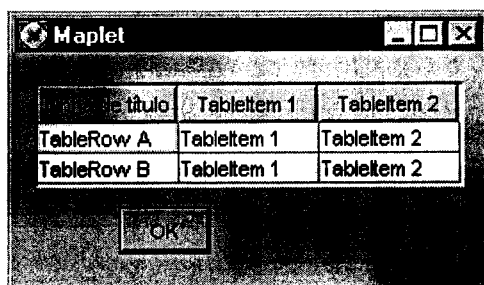
ReturnItem: Especifica que valores são enviados para a sessão de Maple quando uma Maplet é fechada. Não pode conter outros elementos.

TableHeader: O elemento **Table** deve conter no máximo uma **TableHeader** que mostra o texto de uma linha de título numa tabela e qualquer número de elementos **TableRow**.

Nota: Uma tabela com uma linha de título deve ter no elemento **BoxCell** ou **GridCell** a opção '*vscroll*'=*'as_needed'*', caso contrário, a linha de título não é mostrada.

TableItem: Especifica uma entrada numa linha ou numa linha de título de uma tabela da Maplet.

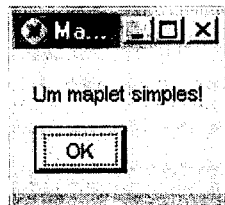
TableRow: Especifica uma linha de uma tabela da Maplet. Cada coluna de uma linha é definida usando o elemento **TableItem**. Nota: Cada **TableRow** deve ter o mesmo número de elementos **TableItem**. O número de elementos **TableItem** de uma linha de título (se especificada) deve ser igual ao número de elementos **TableItem** de uma **TableRow**.



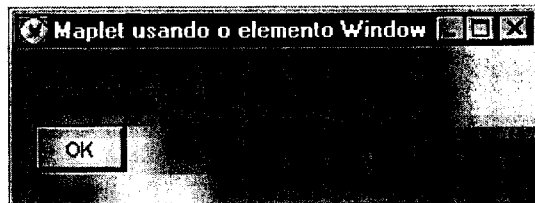
Window: Define uma janela numa Maplet. Uma Maplet pode conter várias janelas, isto é, pode conter mais do que um elemento **Window**. O elemento **Window** contém os elementos que controlam o *Layout* e a função da janela. O **MenuBar** e o **ToolBar** devem estar contidos no elemento **Window**. Ao criar uma janela, o autor da Maplet pode escolher opções que regulam o comprimento e a largura de uma janela, se esta contém ou não um título, etc. Para ver a lista completa de opções deste elemento, pode ser consultada a página de ajuda do Maple 'Maplets[Elements][Window]'.

```
> restart:

> with(Maplets[Elements]):
> mapletwin := Maplet(
    Window(title = "Maplet usando o elemento Window",
      ["Uma Maplet simples!", Button("OK", Shutdown())])
  )
> Maplets[Display](mapletwin);
```



Colocando o ponteiro do rato nos limites da janela, é possível redimensioná-la.



A Opção 'reference'

Muitos elementos são identificados, usando-se a opção 'reference' ou de uma forma mais simples, colocando a variável de identificação entre parêntesis rectos imediatamente

à frente do elemento. Por exemplo, para identificar o conteúdo de uma caixa de texto podemos fazer:

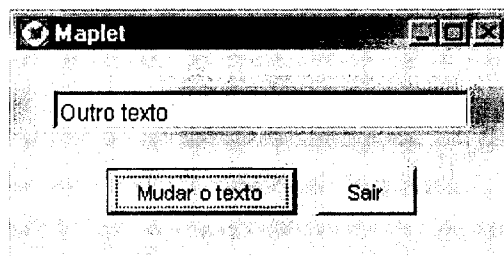
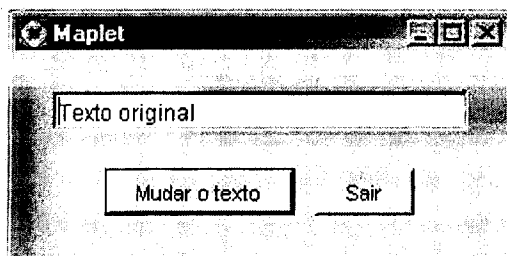
Usando a opção 'reference': `TextField(reference=T1, value= "Olá")`

De uma forma mais simples: `TextField[T1](value= "Olá")`

Estas duas formas de identificar o conteúdo de uma caixa de texto funcionam da mesma maneira, e se for feito usando os dois tipos de identificação, vale a que está entre parêntesis rectos.

No exemplo seguinte, ao clicar o botão 'Mudar o texto', o texto da `TextField` é modificado. O texto na `TextField` é identificado pela variável `T1` e o comando `SetOption` modifica o conteúdo dessa variável.

```
>maplet:=Maplet([
  [TextField[T1]( "Texto original")],
  [
    Button( "Mudar o texto", SetOption(T1= "Outro texto")),
    Button( "Sair", Shutdown())
  ]
]):
>Maplets[Display](maplet);
```



4.5 Maplets[Examples]

O *sub-package* Maplets[Examples] contém exemplos de como as Maplets são usadas para resolver problemas. Estas Maplets fornecem informação ao utilizador, questionam o utilizador ou mostram o *interface* de uma rotina ou de um *sub-package*. O *package* Examples contém as seguintes rotinas e *sub-packages*:

- Alert
- Confirm
- GetColor
- GetEquation
- GetExpression
- GetFile
- GetInput
- Message
- Question
- Selection
- SignQuery

Exemplos mais avançados:

- CurveFitting
- Integration
- KernelOpts
- ShowTable

- *Sub-package* **LinearAlgebra**

Alert: Mostra uma janela de alerta com a mensagem que introduzirmos. Na construção de uma Maplet, se escrevermos **Alert("Insira um valor real!")**, aparece uma janela **AlertDialog** com a mensagem 'Insira um valor real!'.

Exemplo da utilização do 'Alert' :

```
>restart:
>if Maplets[Examples][Alert]( "Quer mesmo apagar o ficheiro?")=true
  then remove ( "ficheiroemcausa.mpl");
end if;
```

Confirm: É a rotina de uma janela 'ConfirmDialog'.

Exemplo da utilização do 'Confirm' :

```
>restart:
>doy:= Maplets[Examples][Confirm]( "Quer a derivada em ordem a y?"):
  if doy = true then
    diff(x^2+y^2, y);
  elif doy = false then
    diff(x^2+y^2, x);
  else
    NULL;
  end if;
```

GetColor: É a rotina de uma janela 'ColorDialog'.

Exemplo da utilização do 'GetColor' :

```
>restart:
>Grafico:= proc()
  plot ( args, color = Maplets[Examples][GetColor]() );
end proc:
>Grafico( sin(x), x = 0..2*Pi );
```

GetEquation: Mostra uma janela em que é pedido ao utilizador que seja inserida uma equação numa 'TextField'. Se o utilizador escrever uma expressão, esta é igualada a zero quando clica em "Ok".

Exemplo da utilização do 'GetEquation' :

```
>restart:
>Resolve := proc()
  solve (Maplets[Examples][GetEquation]('caption' =
"escreva uma equacao em 'x':");
end proc:
>Resolve();
```

GetExpression: Igual ao 'GetEquation', mas aqui é esperada a inserção de uma expressão.

Exemplo da utilização do 'GetExpression' :

```
>restart:
>Derivada := proc()
  diff( Maplets[Examples][GetExpression]('caption' =
```

```
"escreva uma expressao para derivar em 'x':");  
end proc;  
>Derivada();
```

GetFile: É a rotina de uma janela 'FileDialog'.

Exemplo da utilização do 'GetFile', que apaga o ficheiro escolhido pelo utilizador:

```
>restart;  
>fremove(Maplets[Examples][GetFile]());
```

GetInput: É a rotina de uma janela 'InputDialog'.

Exemplo da utilização do 'GetInput':

```
>restart;  
>parse(Maplets[Examples][GetInput]("Escreva um valor inteiro:", 'type'=plain));
```

Message: É a rotina de uma janela 'MessageDialog'.

Exemplo da utilização do 'Message':

```
>restart;  
>Resolve := proc()  
  local sols;  
  sols:= isolve(args);  
  if sols = NULL then  
    Maplets[Examples][Message] ("Não há soluções inteiras!");  
  else sols;  
end proc;
```

```
end if;  
end proc;  
>Resolve(x^2=2);
```

Question: É a rotina de uma janela 'QuestionDialog'.

Exemplo da utilização do 'Question' :

```
>restart:  
>doy:= Maplets[Examples][Question]( "Quer a derivada em ordem a y?"):  
if doy = true then  
    diff(x^2+y^2, y);  
else  
    diff(x^2+y^2, x);  
end if;
```

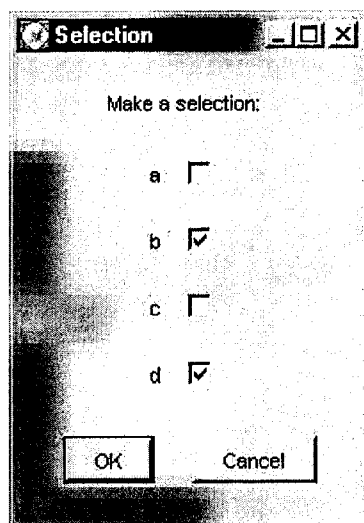
Compare-se este exemplo com o exemplo da utilização do 'Confirm'.

Selection: Mostra uma janela em que o utilizador pode fazer uma ou mais selecções usando elementos 'CheckBox'.

Exemplo da utilização do 'Selection':

```
>restart:  
>with(Maplets[Examples]):  
Selection([a, b, c, d]);
```

Neste exemplo, se o utilizador escolher b e d, sai o resultado '2, 4':



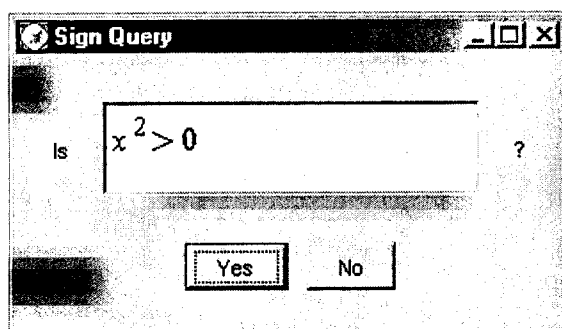
SignQuery: Mostra uma janela onde é perguntado ao utilizador qual o sinal de determinado objecto. Por opção, essa pergunta pode ser feita usando o **MathMLViewer**. A veracidade da resposta está ao critério do autor da Maplet. Se o utilizador clicar em 'Yes' (Sim), a resposta é 'true' (verdade), se clicar em 'No' (Não), a resposta é 'false' (falso).

Exemplo da utilização do 'SignQuery':

```
>restart:
```

```
>with(Maplets[Examples]):
```

```
SignQuery(x^2, mathml);
```



Os exemplos mais avançados, 'CurveFitting', 'Integration', 'KernelOpts' e 'ShowTable' são Maplets pré-definidas que executam respectivamente, aproximação de funções, integração, opções do programa e apresentação de tabelas.

Sub-package de Álgebra Linear

O sub-package 'LinearAlgebra' contém os seguintes exemplos de Maplets:

- Construção - **BezoutMatrix** e **HilbertMatrix**
- Questões - **ConditionNumber**
- Resolução - **QRDecomposition**
- Problemas de valores próprios - **SingularValues**
- Rotinas - **MatrixNorm** e **VectorNorm**

Todos estes exemplos podem ser encontrados através do *Help*, pelo tópico: 'Maplets, LinearAlgebra'.

4.6 Maplets[Tools]

Este *sub-package* das Maplets ('Maplets[Tools]') foi "feito" para os autores de Maplets e contém rotinas para manipular e interagir com Maplets e com elementos de Maplets.

As seguintes funções estão disponíveis no *sub-package* 'Tools':

AddAttribute: Adiciona atributos a um elemento previamente construído. Por exemplo, se criarmos e executarmos uma Maplet que mostra um botão, mas queremos acrescentar uma cor a esse botão, então devemos adicionar um atributo de cor ('color'="FF00FF")

ao código da Maplet. Em vez de estar sempre a modificar o código da Maplet, podemos testar a aparência dessa Maplet usando a função '**AddAttribute**'.

AddContent: Acrescenta conteúdo a um elemento previamente construído.

Get: Vai buscar o valor de uma dada "variável" na execução de uma Maplet. Deverá ser usado apenas em procedimentos. Não pode ser usado na definição de uma Maplet.

ListBoxSplit: Converte os valores de uma '**ListBox**' numa lista de *strings*.

Print: Mostra a estrutura que está por detrás de uma Maplet, isto é, mostra os dados XML que estruturam uma Maplet. São também mostrados os valores por defeito. Esta ferramenta é útil, principalmente quando se está investigar o não funcionamento de uma dada Maplet que não está como esperaríamos.

Set: Tal como o '**Get**', esta ferramenta apenas pode ser utilizada num procedimento e não na definição de uma Maplet. Esta função atribui um determinado valor a uma "variável" usada na definição de uma Maplet.

StartEngine: Inicia um ambiente de 'Maplets'.

StopEngine: Pára o ambiente de 'Maplets'. Todas as janelas são fechadas.

Variáveis Globais das Maplets

Existem duas variáveis globais no *sub-package* Maplets[Tools] que são úteis aos autores de Maplets:

lastmaplet: Sempre que um utilizador corre uma Maplet usando a função **Display**, esta é gravada numa variável global, **lastmaplet**. Pode ser usado para depurar ou mostrar a

execução da Maplet anterior.

thismaplet: Sempre que um procedimento é executado pela utilização do elemento 'Evaluate', é atribuída à variável global **thismaplet** o identificador da Maplet que chama o procedimento. Depois de executado o procedimento, a variável volta a assumir o seu valor anterior.

O seguinte exemplo inclui as ferramentas **Get** e **Set**:

```
> restart:

> with(Maplets[Elements]):with(Maplets[Tools]):
> grafico:=proc()
  plot(Get(T1::algebraic,corrections),x=Get(T2::algebraic,corrections)..
  Get(T3::algebraic,corrections));
end proc:
> texto:=proc() Set(T4="O gráfico está
feito!");
end proc:
> maplet :=
  Maplet(BoxLayout(BoxRow(Plotter[P1](),
  BoxColumn(
  BoxRow("Função:",TextField[T1]()),
  BoxRow("Domínio:  x
de",TextField[T2](width=4),"até",TextField[T3](width=4)),
  BoxRow(Button[B1]("Gráfico",onclick=A1)),
  BoxRow(TextField[T4]()),
  BoxRow(Button[B2]("Fechar",Shutdown()))
))), Action[A1]( Evaluate(P1='grafico()'), Evaluate(B1='texto()'))
):

> Maplets[Display](maplet);
```

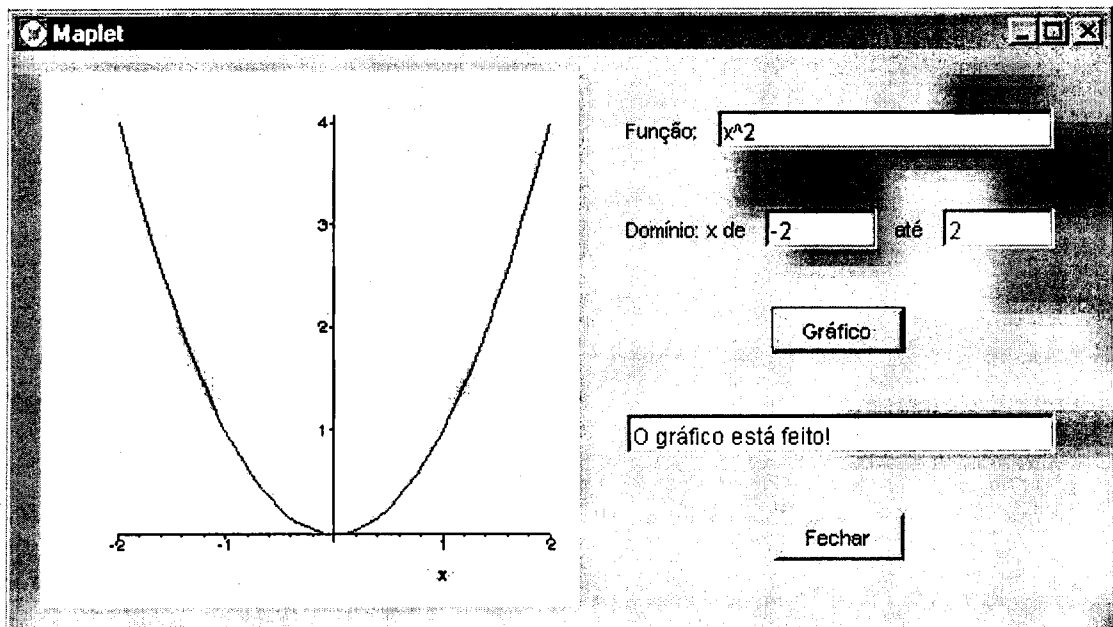
A mesma janela pode ser obtida usando o elemento **SetOption** em vez de **Set**:

```
> restart:
```

```

> with(Maplets[Elements]):with(Maplets[Tools]):
> grafico:=proc()
  plot(Get(T1::algebraic,corrections),x=Get(T2::algebraic,corrections)..
    Get(T3::algebraic,corrections));
  end proc:
> maplet :=
  Maplet(BoxLayout(BoxRow(Plotter[P1]()),
    BoxColumn(
      BoxRow("Função:",TextField[T1]()),
      BoxRow("Domínio: x
de",TextField[T2](width=4),"até",TextField[T3](width=4)),
      BoxRow(Button[B1]("Gráfico",onclick=A1)),
      BoxRow(TextField[T4]()),
      BoxRow(Button[B2]("Fechar",Shutdown()))
    )), Action[A1]( Evaluate(P1='grafico()'), SetOption(T4="O
gráfico está feito!"))
  ):
> Maplets[Display](maplet);

```



Neste capítulo, pretendeu-se dar a conhecer os elementos e as ferramentas necessárias para construir Maplets. É claro que à medida que se constroem mais Maplets, vai-se aumentando a capacidade de utilização dos elementos e das ferramentas adequadas à construção de uma Maplet.

Capítulo 5

Uma Maplet para o estudo gráfico de funções

Neste capítulo será dissecado com detalhe a construção de uma Maplet, que para além de mostrar a representação gráfica de funções, permite que o utilizador possa modificar algumas das opções do gráfico, fazer o *zoom* desejado, estudar a função quanto à existência de zeros, pontos máximos e mínimos, intervalos de crescimento e decrescimento, concavidades da função, etc. É claro que esta Maplet ainda pode ser complementada, pois existem sempre alterações que poderão melhorar a Maplet, para além de se poderem adicionar outros elementos e janelas com os mais variados objectivos.

5.1 A janela principal

No início, começou-se por idealizar apenas uma janela base, onde fosse possível inserir uma função e clicar num botão para o gráfico aparecer.

Para além do espaço para o gráfico (**Plotter**), a Maplet inclui uma caixa de texto para inserção da função (**TextField**), duas caixas de selecção (**CheckBox**) para seleccionar o domínio e/ou o contradomínio, as caixas de texto para inserir os valores extremos do domínio e do contradomínio, o botão para fazer aparecer o gráfico (**Button**) e um botão

para fechar a janela.

Para organizar estes elementos na janela usou-se como **layout** os elementos do **Box-Layout**. O título da janela escolhido foi: 'Gráficos de funções'. Nesta Maplet foi utilizado um procedimento para a construção do gráfico.

O código desta primeira Maplet começa com o comando '**Restart:**'

```
> restart:  
> with(Maplets):with(Elements):with(plots):with(Tools):
```

Warning, the name `changecoords` has been redefined

Como foram usados elementos dos *packages* `Maplets[Elements]` e `Maplets[Tools]`, foram chamados com o comando **with** as *packages* `Maplets`, `Elements`, `Tools` e também o *package* `plots` para comandos como **display**, necessário na representação gráfica.

```
> pix:=400:Digits:=5:
```

As variáveis '`pix`' e '`Digits`', representam respectivamente o tamanho em pixels da altura e comprimento do espaço destinado ao gráfico na janela (**Plotter**) e o número de algarismos significativos com que o Maple arredonda todos os cálculos feitos nesta Maplet. Neste caso, escolheu-se **`pix:=400:`** e **`Digits:=5:`**.

```
> pf:=proc(f,cbx,x_min,x_max,cby,y_min,y_max)
```

Nesta Maplet, o procedimento '`pf(a, b, c, d, e, f, g)`' constrói o gráfico da função nos intervalos definidos. Este procedimento espera 6 parâmetros de entrada. Neste caso, na posição *a*, '`pf`' espera a entrada de uma função (*f*), na posição *b*, a validação ou

não dos limites do domínio (eixo das abcissas), que na janela são validados através da activação de uma **CheckBox** (cbx), nas posições *c* e *d*, são esperados respectivamente os valores mínimo e máximo do domínio (eixo das abcissas) (*x_min* e *x_max*), na posição *e*, a validação ou não dos limites do eixo das ordenadas, que na janela também são validados através de uma **CheckBox** (cby) e nas posições *f* e *g*, são esperados respectivamente os valores máximo e mínimo do eixo das ordenadas (*y_min* e *y_max*).

```
> local xmin, xmax, ymin, ymax, xy, plt,  
    listpt, listpt0, listpt1, listpty, j, miny, maxy, eps:
```

As variáveis *xmin*, *xmax*, *ymin*, *ymax*, *xy*, *plt*, *listpt*, *listpt0*, *listpt1*, *listpty*, *j*, *miny*, *maxy* e *eps* são todas declaradas locais, pois só vão ser usadas neste procedimento.

```
> xmin:=parse(convert(x_min,string)):  
    xmax:=parse(convert(x_max,string)):  
    ymin:=parse(convert(y_min,string)):  
    ymax:=parse(convert(y_max,string)):
```

Às variáveis *xmin*, *xmax*, *ymin* e *ymax* são atribuídos respectivamente os valores que vêm de fora através de *x_min*, *x_max*, *y_min*, *y_max*. A instrução `parse(convert(x_min,string))`: garante que o valor representado por *x_min* fica no formato de *string* com uma sintaxe correcta.

```
> if cbx='true' and cby='true' then  
    Set(ChxT::anything=true):  
    Set(ChyT::anything=true):  
    xy:=x=xmin..xmax,y=ymin..ymax:
```

```

> elif cbx='true' and cby='false' then
    Set(ChxT::anything=true):
    Set(ChyF::anything=true):
    xy:=x=xmin..xmax:
elif cbx='false' and cby='true' then
    Set(ChxF::anything=true):
    Set(ChyT::anything=true):
    xy:=x,y=ymin..ymax:
else
    Set(ChxF::anything=true):
    Set(ChyF::anything=true):
    xy:=x,y:
end if:

```

Esta declaração condicional serve para definir a sintaxe que é usada no comando **plot** para indicar os limites dos eixos onde é representada a função. As possibilidades são: definir os limites dos eixos das abcissas e das ordenadas (**plot(f(x),x=xmin..xmax,y=ymin..ymax)**); definir os limites do eixo das abcissas e deixar que os limites do eixo das ordenadas seja obtido por defeito (**plot(f(x),x=xmin..xmax,y)**)¹ - o Maple ajusta estes limites ao gráfico da função ; definir os limites do eixo das ordenadas e deixar que os limites do eixo das abcissas seja obtido por defeito (**plot(f(x),x,y=ymin..ymax)**) - os limites atribuídos por defeito ao eixo das abcissas são -10 e 10; deixar que os limites dos dois eixos sejam obtidos por defeito (**plot(f(x),x,y)**). Os parâmetros 'cbx' e 'cby' representam a activação ou não das caixas de selecção 'CheckBox' que o utilizador escolhe na janela principal para activar ou desactivar os limites que pretende para o gráfico. Assim, se as duas caixas estiverem seleccionadas, cbx='true' e cby='true', a sintaxe atribuída à variável xy inclui a definição dos limites do x e do y. Os comandos **Set(ChxT::anything=true):** e **Set(ChyT::anything=true):** activam as acções 'AChxT' e 'AChyT' que validam as variáveis Xmin, Xmax, Ymin e Ymax. Senão, se estiver apenas seleccionada a caixa que activa os limites do eixo das abcissas, cbx='true' e cby='false', a sintaxe atribuída

¹Neste caso, a indicação do 'y' pode ser omitida.

à variável `xy` apenas inclui a variação do `x`. O comando `Set(ChyF::anything=true):` activa a acção 'AChyF' que invalida as variáveis `Ymin` e `Ymax`. Se tivermos `cbx='false'` e `cby='true'`, isto significa que está seleccionada apenas a caixa referente aos limites no eixo das ordenadas. O comando `Set(ChxF::anything=true):` activa a acção `AChxF` que invalida as variáveis `Xmin`, `Xmax`. Se nenhuma das caixas estiver seleccionada, os limites do gráfico são obtidos por defeito.

```
> plt:=plot(f(x),xy);
```

À variável `plt` é atribuído o comando que constrói o gráfico da função `f` restrita aos limites definidos para `xy`, ou seja, `'plt:=plot(f(x),xy);'`.

```
> listpt0:=convert(plt,list):
listpt1:=listpt0[1]:
listpt:=op(1,listpt1):
listpty:=NULL:
for j from 1 to nops(listpt) do
  listpty:=listpty,listpt[j][2]:
end do:
miny:=min(listpty):
maxy:=max(listpty):
eps:=(maxy-miny)/50:
if cby='false' then
  Set(Ymin=miny-eps):
  Set(Ymax=maxy+eps):
end if:
if cbx='false' then
  Set(Xmin=-10):
  Set(Xmax=10):
end if:
```

Na verdade, a variável `plt` é um conjunto de listas que incluem os pontos que o Maple

calcula para fazer o gráfico. Assim é possível achar os valores dos limites que o Maple calcula por defeito para o eixo das ordenadas (para o eixo das abcissas, sabemos que por defeito, o x varia entre -10 e 10). A ideia é fazer aparecer nas caixas de texto destinadas à variação do y , os valores que o Maple encontra por defeito. A lista pretendida é 'listpt', ou seja, é a lista dos pontos na forma $[x,y]$ que o Maple calcula para construir o gráfico. Esta lista é o primeiro elemento da lista 'listpt1' que por sua vez é o primeiro elemento da lista 'listpt0'. Com 'listpt0', garantimos que `plt` é visto apenas como uma lista. A lista 'listpty' é construída através de um ciclo 'for' que junta todas as ordenadas dos pontos de 'listpt'. A 'miny' é atribuída a menor ordenada e a 'maxy' é atribuída a maior ordenada da lista. Assim, sabemos que o gráfico no domínio especificado varia entre `miny` e `maxy`. Para dar um pouco mais de espaço ao gráfico, adicionou-se um *epsilon* a `maxy` e subtraiu-se o mesmo *epsilon* a `miny`. Portanto, se o y variar por defeito, nas caixas de texto onde aparecem os limites do y vão estar os valores `miny-eps` e `maxy+eps`, em que `eps` é o referido *epsilon*.

```
> display(plt);
end proc;
```

Para exibir o gráfico é necessário usar o comando **display**. Neste caso, teremos de fazer `display(plt);`. Com 'end proc' conclui-se o procedimento. Os procedimentos são criados "fora" da Maplet propriamente dita. Portanto, até este ponto ainda não está iniciada a construção da Maplet.

```
> mapletG:=Maplet(onstartup=Ainicial,
```

Atribuiu-se a esta Maplet o nome de 'mapletG'. A este nome foi atribuído o elemento **Maplet** que inclui todos os elementos e ferramentas necessárias para a apresentação da janela principal. A opção `onstartup=Ainicial`, significa que ao ser executada a Maplet, deve ser accionada a acção 'Ainicial'.

```
> Window[Principal]("Gráficos de funções",
[
  [BoxLayout(BoxColumn(
    BoxRow(
      BoxCell(value=P),
      BoxCell(
        BoxLayout(inset=0,BoxColumn(
          BoxRow(
            BoxCell(
              BoxLayout(inset=0,border=true,caption="Função",BoxColumn(
                BoxRow(
                  BoxCell("f(x)="),
                  BoxCell(value=Tf)
                ),
                BoxRow(
                  BoxCell("x:  "),
                  BoxCell(value=Chx),
                  BoxCell(value=Xmin),
                  BoxCell(value=Xmax),
                  BoxCell()
                ),
                BoxRow(
                  BoxCell("y:  "),
                  BoxCell(value=Chy),
                  BoxCell(value=Ymin),
                  BoxCell(value=Ymax),
                  BoxCell()
                ),
                BoxRow(
                  BoxCell(),
                  BoxCell(value=Bgrafico),
                  BoxCell()
                )
              )
            )
          )
        )
      )
    )
  )
])
```

```

> )),
    BoxRow(
      BoxCell(
        BoxLayout(inset=0,BoxColumn(
          BoxRow(
            BoxCell(),
            BoxCell(value=Bfechp),
            BoxCell())
          )))
      )))
    ]
  ),

```

O elemento **Window** inclui o título da janela ("Gráfico de funções") e o *layout* onde estão organizados os restantes elementos da janela. Neste caso, a organização é feita através do '**BoxLayout**'.

Depois do *layout* estar organizado e já "fora" do elemento **Window**, definem-se os restantes elementos. Estes elementos, na janela, vão situar-se exactamente no sítio em que foram indicados no *layout* através da opção `value=referência do elemento`. Por exemplo, a referência do **Plotter** é 'P', isto significa que o espaço dedicado à apresentação de gráficos vai situar-se na **BoxCell** que tem a opção `value=P`.

```

> Plotter[P](plot(0,x),height=pix,width=pix),
  TextField[Tf]("sin(x)",onchange=AF),
  CheckBox[Chx](value=true,onchange=AF),
  CheckBox[ChxT](onchange=AChxT),
  CheckBox[ChxF](onchange=AChxF),
  TextField[Xmin](value="-10",width=4,onchange=AF),
  TextField[Xmax](value="10",width=4,onchange=AF),
  CheckBox[Chy](value=false,onchange=AF),
  CheckBox[ChyT](onchange=AChyT),

```

```
> CheckBox[ChyF] (onchange=AChyF),
  TextField[Ymin] (value=-1.04,width=4,onchange=AF),
  TextField[Ymax] (value=1.04,width=4,onchange=AF),
  Button[Bgrafico] ("Gráfico",onclick=AF),
  Button[Bfechp] ("Fechar", Shutdown()),
```

A função `plot(0,x)` garante que pelo menos os eixos do gráfico serão exibidos e as opções `'height=pix'` e `'width=pix'` definem em pixels, o tamanho do **Plotter** de acordo com o valor atribuído a `'pix'`. Neste caso `pix:=400`. O elemento `TextField[Tf]` é a caixa de texto onde é inserida a função. Por defeito, foi escolhida a função $\sin(x)$ e sempre que for modificada, é activado o procedimento `pf` para construir o seu gráfico (`'onchange=AF'`). O elemento `CheckBox[Chx]` representa a caixa de selecção para validar ou não os limites do eixo das abcissas. Por defeito, a caixa aparece seleccionada (`value=true`), o que significa que o gráfico é construído com base nos limites indicados nas caixas de texto `TextField[Xmin]` e `TextField[Xmax]`. Neste caso `Xmin=-10` e `Xmax=10`. O elemento `CheckBox[Chy]` representa a caixa de selecção para validar ou não os limites do eixo das ordenadas. Esta caixa não aparece seleccionada de início (`value=false`) e portanto, os valores que aparecem nas caixas de texto `TextField[Ymin]` e `TextField[Ymax]` são os valores que o Maple obtém por defeito. Os elementos `CheckBox[ChxT]`, `CheckBox[ChxF]`, `CheckBox[ChyT]` e `CheckBox[ChyF]` são caixas virtuais que accionam respectivamente as acções `AChxT`, `AChxF`, `AChyT` e `AChyF`.

O elemento `Button[Bgrafico]` representa o botão que é accionado para mostrar o gráfico da função e o elemento `Button[Bfechp]` representa o botão que fecha a janela quando é accionado.

```
> Action[AChxT] (
  SetOption(Xmin(enabled)='true'),
  SetOption(Xmax(enabled)='true')
),
```

```

> Action[AChxF] (
    SetOption(Xmin(enabled)='false'),
    SetOption(Xmax(enabled)='false')
),
Action[ACHyT] (
    SetOption(Ymin(enabled)='true'),
    SetOption(Ymax(enabled)='true')
),
Action[ACHyF] (
    SetOption(Ymin(enabled)='false'),
    SetOption(Ymax(enabled)='false')
),

```

As acções 'AchxT' e 'AchyT' validam as variáveis 'Xmin', 'Xmax', 'Ymin' e 'Ymax', enquanto que as acções 'AchxF' e 'AchyF' desactivam as mesmas variáveis.

```

> Action[AF] (
    Evaluate(P='pf(unapply(Tf,x),Chx,Xmin,Xmax,Chy,Ymin,Ymax)')
),
Action[Ainicial] ( RunWindow(Principal),
    Evaluate(P='pf(unapply(Tf,x),Chx,Xmin,Xmax,Chy,Ymin,Ymax)')
)

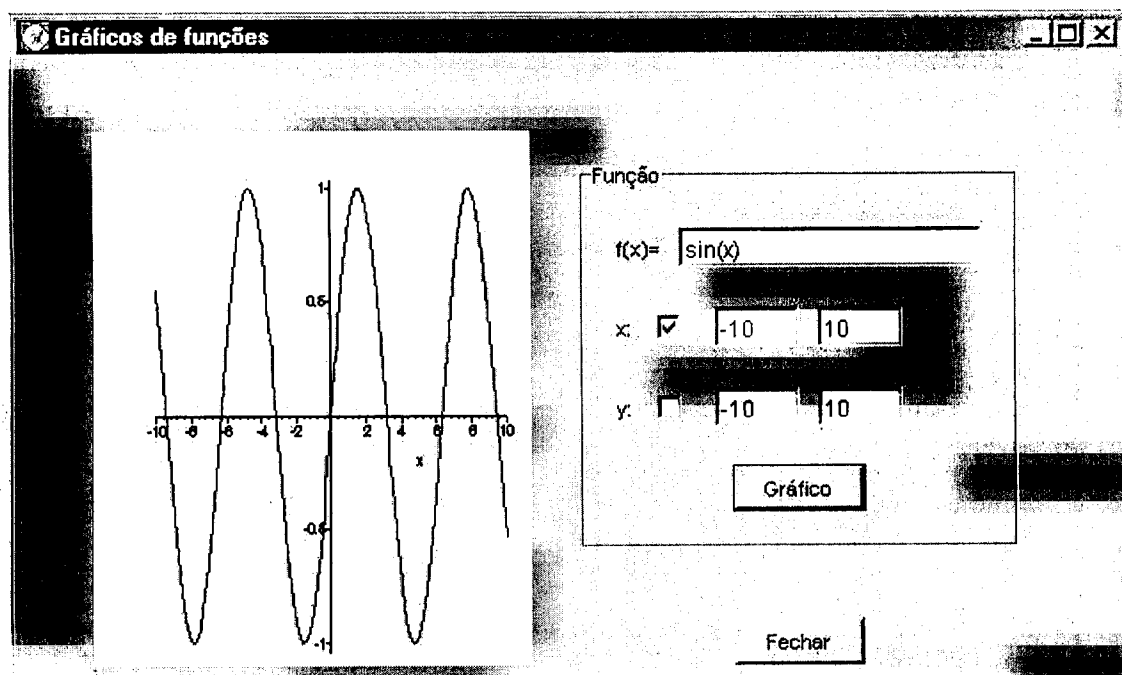
```

Na acção 'Ainicial', que é accionada mal a Maplet é executada, estão incluídas duas instruções: 'RunWindow(Principal)' e 'Evaluate(P='pf(...)')'. A primeira "abre" a janela principal e a segunda atribui ao espaço destinado ao gráfico da função (P) o procedimento que constrói o gráfico da função inserida por defeito ($\sin(x)$). Assim, quando um utilizador executa esta Maplet, é aberta uma janela na qual aparece à partida o gráfico da função $\sin(x)$ no intervalo $[-10,10]$. A acção 'AF' executa o procedimento 'pf(...)' sempre que é clicado o botão "Gráfico" ou quando são modificados os elementos que se encontram no *layout* da Maplet.

```
> ):  
> Display(mapletG);
```

No final fecha-se o elemento **Maple**. Para abrir a janela, é necessário executar a função **Display(mapletG);**.

A janela construída com este código é a seguinte:



5.2 A janela de opções

Surgiu, como é óbvio, a necessidade de se poder modificar o aspecto dos gráficos. Para isso teve de se criar um botão que abrisse uma janela de opções para o gráfico. Tomou-se como base de exemplo a janela de opções interactiva que o Maple disponibiliza através do comando **interactive**.

Criou-se, portanto, um botão para abrir uma segunda janela, onde o utilizador faz as alterações desejadas às opções possíveis num gráfico.

Relativamente à Maplet anterior, fizeram-se algumas alterações na janela principal, foi criada uma nova janela (a janela 'opcoes'), foram criados novos procedimentos relativos ao funcionamento da janela de opções e fizeram-se pequenas alterações no procedimento 'pf' de modo a que o gráfico fosse construído tendo em conta as opções utilizadas.

Na janela principal, acrescentou-se mais um botão com o objectivo de abrir a janela relativa às opções do gráfico.

```
> Button[Bopcoes] ("Opções", Action(RunWindow(Opcoes)))
```

A janela de opções ('Opcoes'), tem como título, "Opções da função" e permite escolher o tipo de eixos para o gráfico, se o gráfico tiver pontos de descontinuidade, é permitido escolher se são mostradas ou não as assíntotas verticais do gráfico, pode-se ainda escolher o número de pontos que definem o gráfico, a cor do gráfico, se for escolhido um gráfico definido por uma ou mais linhas, é permitido escolher o tipo de linha do gráfico e a sua espessura e se for escolhido um gráfico definido por pontos, é permitido escolher o símbolo que ilustra os pontos do gráfico e o seu tamanho.

```
> Janelaopcoes:=Window[Opcoes] (title="Opções
da função",
[
[GridLayout(
GridRow(
GridCell("eixos:  ",left),
GridCell(value=Deix),
GridCell(),
GridCell(),
```

```

> GridCell("assíptotas verticais:",left),
  GridCell(value=Cdc)
),
GridRow(
  GridCell("numPontos: ",left),
  GridCell(value=Tnumpts),
  GridCell(),
  GridCell(),
  GridCell("cor: ",left),
  GridCell(value=Dcor)
)
)],

```

Esta parte do *layout*, inclui uma 'DropDownBox' onde se escolhe o tipo de eixos do gráfico, uma 'CheckBox' que quando é seleccionada, mostra as assíptotas verticais do gráfico, caso existam pontos de descontinuidade, uma 'TextField' para inserir o número de pontos que definem o desenho do gráfico e outra 'DropDownBox' para escolher a cor do gráfico.

```

> ["Linha",RadioButton[Rlin](value=true,group=G1,onclick=ALin),
  DropDownBox[Dlinest](["Solida","Tracejado","Pontead","Trace./Pont."],
  value="Solida"),
  ComboBox[Clinesp](["1","2","3","4","5"],value="1")],
  ["Ponto",RadioButton[Rpt](value=false,group=G1,onclick=APt),
  DropDownBox[Dptest](["Quadrado","Cruz","Circulo","Ponto","Losango"],
  value="Ponto",enabled=false),
  ComboBox[Cptesp](["10","15","20","25","30"],value="10",enabled=false)]
,

```

Esta parte do *layout*, inclui dois 'RadioButton', duas 'DropDownBox' (não é possível editar) e duas 'ComboBox' (pode inserir-se qualquer valor), onde o utilizador pode escolher entre ter um gráfico definido por linha ou ponto. Se escolher linha, pode desde

logo escolher o tipo de linha que prefere e a espessura pretendida para a linha. Se escolher ponto, pode seleccionar o símbolo desejado para representar cada ponto e o tamanho desse símbolo.

```
> [Button("OK",onclick=FOK),
    Button("Cancelar",onclick=CloseWindow(Opcoes)),
    Button("Aplicar",onclick=AF)]
    ]),
```

Esta é a última parte do *layout* onde são incluídos os botões. O botão 'Ok', fecha a janela e actualiza as opções definidas, o botão 'Cancelar' apenas fecha a janela sem actualizar as opções definidas e o botão 'Aplicar' apenas actualiza as opções definidas sem fechar a janela. As acções 'FOK' e 'AF' são definidas na janela principal.

```
> DropDownBox[Deix]([{"Caixa","Laterais","Normais","Nenhum"},value="Normais"),
    DropDownBox[Dcor]([{"Amarelo","Azul","Azul Cyan","Azul Marinho","Azul Turquesa","Branco","Bronze","Castanho","Cinzento","Dourado","Laranja","Magenta","Preto","Rosa","Roxo","Verde","Vermelho","Violeta"},value="Vermelho"),
    CheckBox[Cdc](value=false),
    TextField[Tnumpts](width=4,"50"),
    ButtonGroup[G1](),
```

Esta é a definição dos elementos que são descritos na primeira parte do *layout*. O elemento '**ButtonGroup**' agrupa os dois '**RadioButton**' de forma a que quando um é seleccionado, o outro é automaticamente não seleccionado.

```
> Action[ALin](
    SetOption(Dlinest(enabled)='true'),
    SetOption(Clinspace(enabled)='true'),
    SetOption(Dptest(enabled)='false'),
```

```

> SetOption(Cptesp(enabled)='false')
),
Action[APt](
  SetOption(Dlinest(enabled)='false'),
  SetOption(Clinessp(enabled)='false'),
  SetOption(Dptest(enabled)='true'),
  SetOption(Cptesp(enabled)='true')
):

```

Se o utilizador escolher 'Linha' nesta janela, é accionada a acção 'ALin' que valida as variáveis 'Dlinest' e 'Clinessp' e invalida as variáveis 'Dptest' e 'Cptesp'. Se for escolhido 'Ponto', é accionada a acção 'APt' que invalida 'Dlinest' e 'Clinessp' e valida 'Dptest' e 'Cptesp'.

Em todos os elementos da janela, o utilizador pode escolher entre várias opções que o Maple não reconhece por não estarem na sua forma original. Por exemplo, quando o utilizador escolhe como cor do gráfico o 'Azul', o Maple não reconhece esta denominação. Tem de haver um procedimento que transforme esta opção do utilizador em 'blue', que é no Maple a indicação da cor azul. Assim foram criados vários procedimentos que funcionam como uma tradução das opções desta janela.

```

> estilol:=proc()
  global Dlinest1:
  if Maplets:-Tools:-Get(Dlinest)="Solida" then
    Dlinest1:"SOLID":
  elif Maplets:-Tools:-Get(Dlinest)="Tracejado" then
    Dlinest1:"DASH":
  elif Maplets:-Tools:-Get(Dlinest)="Pontado" then
    Dlinest1:"DOT":
  elif Maplets:-Tools:-Get(Dlinest)="Trace./Pont." then
    Dlinest1:"DASHDOT":
  end if:
end proc:

```

Este procedimento faz a tradução acerca do estilo da linha. Recebe a opção pela variável 'Dlinest' e guarda a versão para o Maple na variável 'Dlinest1'.

```
> estilop:=proc()
  global Dptest1:
  if Maplets:-Tools:-Get(Dptest)="Quadrado" then
    Dptest1:="BOX":
  elif Maplets:-Tools:-Get(Dptest)="Cruz" then
    Dptest1:="CROSS":
  elif Maplets:-Tools:-Get(Dptest)="Circulo" then
    Dptest1:="CIRCLE":
  elif Maplets:-Tools:-Get(Dptest)="Ponto" then
    Dptest1:="POINT":
  elif Maplets:-Tools:-Get(Dptest)="Losango" then
    Dptest1:="DIAMOND":
  end if:
end proc:
```

Este procedimento faz a tradução acerca do símbolo que o utilizador escolhe para a representação gráfica dos pontos. Recebe a opção pela variável 'Dptest' e guarda a versão para o Maple na variável 'Dptest1'.

```
> tipoeixos:=proc()
  global Deix1:
  if Maplets:-Tools:-Get(Deix)="Caixa" then
    Deix1:="BOXED":
  elif Maplets:-Tools:-Get(Deix)="Laterais" then
    Deix1:="FRAMED":
  elif Maplets:-Tools:-Get(Deix)="Normais" then
    Deix1:="NORMAL":
  elif Maplets:-Tools:-Get(Deix)="Nenhum" then
    Deix1:="NONE":
  end if:
end proc:
```

Neste procedimento são traduzidas as opções para o tipo de eixos coordenados.

```
> cores:=proc()
  global Dcor1:
  if Maplets:-Tools:-Get(Dcor)="Amarelo" then
    Dcor1:="yellow":
  elif Maplets:-Tools:-Get(Dcor)="Azul" then
    Dcor1:="blue":
  elif Maplets:-Tools:-Get(Dcor)="Azul Cyan" then
    Dcor1:="cyan":
  elif Maplets:-Tools:-Get(Dcor)="Azul Marinho" then
    Dcor1:="navy":
  elif Maplets:-Tools:-Get(Dcor)="Azul Turquesa" then
    Dcor1:="turquoise":
  elif Maplets:-Tools:-Get(Dcor)="Branco" then
    Dcor1:="white":
  elif Maplets:-Tools:-Get(Dcor)="Bronze" then
    Dcor1:="tan":
  elif Maplets:-Tools:-Get(Dcor)="Castanho" then
    Dcor1:="brown":
  elif Maplets:-Tools:-Get(Dcor)="Cinzentos" then
    Dcor1:="grey":
  elif Maplets:-Tools:-Get(Dcor)="Dourado" then
    Dcor1:="gold":
  elif Maplets:-Tools:-Get(Dcor)="Laranja" then
    Dcor1:="orange":
  elif Maplets:-Tools:-Get(Dcor)="Magenta" then
    Dcor1:="magenta":
  elif Maplets:-Tools:-Get(Dcor)="Preto" then
    Dcor1:="black":
  elif Maplets:-Tools:-Get(Dcor)="Rosa" then
    Dcor1:="pink":
  elif Maplets:-Tools:-Get(Dcor)="Roxo" then
    Dcor1:="maroon":
```

```

> elif Maplets:-Tools:-Get(Dcor)="Verde"
  then
    Dcor1:="green":
    elif Maplets:-Tools:-Get(Dcor)="Vermelho" then
      Dcor1:="red":
    elif Maplets:-Tools:-Get(Dcor)="Violeta" then
      Dcor1:="violet":
    end if:
  end proc:

```

O procedimento 'cores', faz a tradução das cores do gráfico.

Depois de escolhidas as opções nesta janela, é necessário que o Maple compreenda como as usar, ou seja, quando no procedimento 'pf' surgir o comando **plot** para construir o gráfico, é necessário que estas opções apareçam bem discriminadas neste comando. Para isso foi criado um procedimento para compor uma síntese das opções para ser inserida no comando 'plot'.

```

> sinteseop:=proc(op1)
  local o,tip,esp,eix,cor,npt:

```

O procedimento 'sinteseop' recebe o parâmetro 'op1' que é uma lista de 9 opções para o gráfico. Estas opções vêm das variáveis definidas na janela de opções (e das variáveis resultantes dos procedimentos de tradução de opções). Todas as variáveis deste procedimento são locais. Na variável 'o' vai sendo construída a síntese das opções para inserir no comando **plot**.

```

> if op1[1]='true' then
  o:="style=line":
  tip:=convert(op1[2],string):
  esp:=convert(op1[3],string):

```

```

> o:=cat(o,",linestyle=",tip,",thickness=",esp):
else
o:="style=point":
tip:=convert(op1[4],string):
esp:=convert(op1[5],string):
o:=cat(o,",symbol=",tip,",symbolsize=",esp):
end:
eix:=convert(op1[6],string):
o:=cat(o,",axes=",eix):
cor:=convert(op1[7],string):
o:=cat(o,",color=",cor):
if op1[8]='true' then
o:=cat(o,",discont=false"):
else
o:=cat(o,",discont=true"):
end:
npt:=convert(op1[9],string):
o:=cat(o,",numpoints=",npt):
end proc:

```

Na primeira posição da lista de opções 'op1' está o parâmetro 'Rlin' que informa se o **RadioButton** relativo à 'Linha' está ou não seleccionado. 'if op1[1]='true' then o:="style=line":' significa que se o utilizador seleccionou 'Linha' na janela de opções, então na síntese das opções deve constar "style=line". Na segunda posição da lista está a variável 'Dlinest1' ('op1[2]') relativa ao estilo da linha e na terceira posição está a variável 'Clinesp' ('op1[3]') relativa à espessura da linha. Se na janela de opções, 'Linha' não estiver seleccionada, é porque está seleccionado 'Ponto'. Neste caso, na síntese deverá aparecer "style=point". Na quarta posição da lista está a variável 'Dptest1' ('op1[4]') relativa ao símbolo usado pelos pontos do gráfico e na quinta posição está a variável 'Cptesp' ('op1[5]') relativa ao tamanho do símbolo. Por defeito, Rlin='true', a Dlinest1 é atribuída a sintaxe "SOLID", Clinesp=1, a Dptest1 é atribuída a sintaxe "POINT" e Cptesp=10. Na sexta posição está a variável 'Deix1' ('op1[6]') relativa ao tipo de eixos

do gráfico e na posição 7 está a variável 'Dcor1' ('op1[7]') relativa à cor da linha ou dos pontos do gráfico. Por defeito, a *Deix1* é atribuída a sintaxe "NORMAL" e a *Dcor1* é atribuída a sintaxe "red". Na oitava posição da lista, está o parâmetro 'Cdc' ('op1[8]') que informa se a **CheckBox** relativa à exibição ou não de assíntotas verticais quando o gráfico tem pontos de descontinuidade, está ou não seleccionada. Se essa caixa estiver seleccionada, na síntese do **plot**, aparecerá como opção, "**discont=false**", caso contrário, aparecerá "**discont=true**". Por defeito, *Cdc*='false'. Na última posição da lista, está a variável 'Tnumpts' ('op1[9]') que define o número de pontos que o Maple utiliza para desenhar o gráfico. Por defeito, *Tnumpts*=50.

Como a instrução **plot** no procedimento 'pf' vai ser diferente, pois vão ser acrescentadas as opções, será necessário fazer alterações em 'pf' de modo a que a partir daqui o gráfico seja construído tendo em conta as opções escolhidas.

Assim, o procedimento 'pf' vai ter mais um parâmetro de entrada, que será o parâmetro das opções.

```
> pf:=proc(f,cbx,x_min,x_max,cby,y_min,y_max,op1)
    local xmin,xmax,ymin,ymax,j,listpty,miny,maxy,eps,
    opc1,plt,opc,xy,listpt,listpt0,listpt1:
```

O parâmetro 'op1' é o mesmo do procedimento 'sinteseop', ou seja, é uma lista de 9 opções para o gráfico. Às variáveis locais que já faziam parte do procedimento, adicionaram-se 'opc1' e 'opc'.

```
>   opc1:=convert(sinteseop(op1),string):
    opc:=parse(convert(opc1,string)):
```

A 'opc1' é atribuído o resultado do procedimento 'sinteseop(op1)' em forma de *string*, ou seja, é atribuída a síntese das opções para o comando **plot**. Na variável 'opc' fica

guardada 'opc1' na forma de declaração do Maple. Vamos ver a diferença entre o que está contido em cada uma destas variáveis:

```
> opc1;
```

```
"style=line,linestyle=SOLID,thickness=1,axes=NORMAL,color=red,discont=true \
,numpoints=50"
```

```
> opc;
```

```
style = line, linestyle = SOLID, thickness = 1, axes = NORMAL, color = red,
discont = true, numpoints = 50
```

A função **parse** transforma uma *string* ("texto") numa declaração do Maple, isto é, em algo que seria igual ao que um utilizador de Maple introduz numa linha de entrada numa *worksheet* (texto).

Na função **plot** incluída neste procedimento, acrescenta-se, agora, a síntese das opções para o gráfico.

```
> plt:=plot(f(x),xy,opc);
```

Na janela principal acrescentaram-se mais algumas coisas:

```
> Janelaopcoes,
  Action[AF](
    Evaluate(Dlinest='estilol()'),
    Evaluate(Dptest='estilop()'),
    Evaluate(Deix='tipoeixos()'),
    Evaluate(Dcor='cores()'),
    Evaluate(P='pf(unapply(Tf,x),Chx,Xmin,Xmax,Chy,Ymin,Ymax,[Rlin,Dlines
t1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts])'),
  ),
```

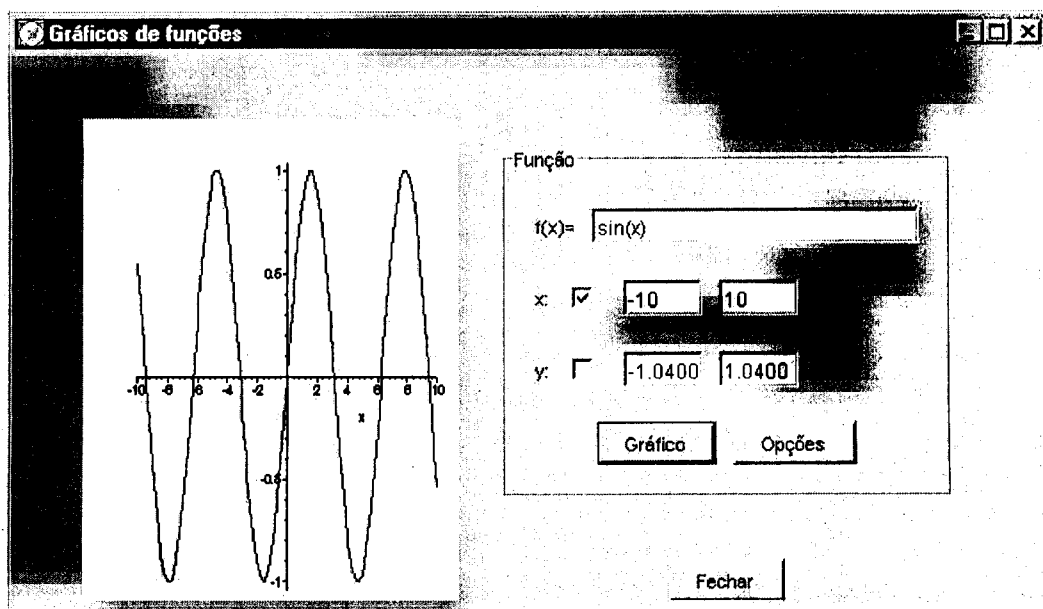
```

> Action[Ainicial](
  RunWindow(Principal),
  Evaluate(Dlinest='estilol()'),
  Evaluate(Dptest='estilop()'),
  Evaluate(Deix='tipoeixos()'),
  Evaluate(Dcor='cores()'),
  Evaluate(P='pf(unapply(Tf,x),Chx,Xmin,Xmax,Chy,Ymin,Ymax,[Rlin,Dlinest
1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts]))'),
),
Action[FOK](
  CloseWindow(Opcoes),
  Evaluate(Dlinest='estilol()'),
  Evaluate(Dptest='estilop()'),
  Evaluate(Deix='tipoeixos()'),
  Evaluate(Dcor='cores()'),
  Evaluate(P='pf(unapply(Tf,x),Chx,Xmin,Xmax,Chy,Ymin,Ymax,[Rlin,Dlines
t1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts]))')
)

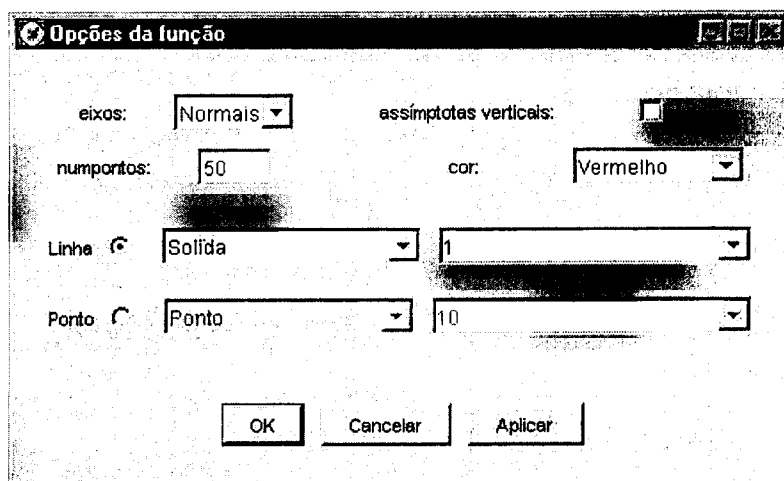
```

Em relação ao que já foi descrito para esta janela principal, acrescenta-se 'Janelaopcoes', que é a referência à janela das opções. Nas acções 'AF' e 'Ainicial', foram acrescentados 4 elementos **Evaluate** relativos às actualizações das variáveis 'Dlinest', 'Dptest', 'Deix' e 'Dcor' para 'Dlinest1', 'Dptest1', 'Deix1' e 'Dcor1' respectivamente, através dos procedimentos 'estilol()', 'estilop()', 'tipoeixos()' e 'cores()'. Para além disso, nas entradas para o procedimento 'pf' é acrescentada a lista das 9 opções para o gráfico da função. É também acrescentada mais uma acção na janela principal, embora ela seja accionada a partir da janela de opções: A acção 'FOK' fecha a janela de opções e executa o mesmo que a acção 'AF'.

A janela principal desta Maplet, em termos de aparência, fica então com o seguinte aspecto:



Se clicarmos no botão 'Opções', abre-se a seguinte janela:



5.3 As janelas do Zoom e do Trace

Depois de termos uma Maplet, onde é possível construir o gráfico de uma função com acesso à escolha de várias opções ao nível do seu visionamento, tentou-se complementar

esta Maplet com opções mais interactivas para o visionamento do gráfico, tais como o *zoom* e o *trace*, funções que se encontram facilmente em qualquer calculadora gráfica.

A situação ideal seria conseguir, através do posicionamento do cursor na área destinada à representação gráfica da função, movimentar o gráfico para onde fosse desejado, mas tal, para já não é possível nas Maplets. Portanto, foram criados botões que permitem ao utilizador deslocar o gráfico para qualquer direcção e um **Slider** (eixo onde se movimenta uma marca - ver capítulo 4) para fazer um *zoom* centrado na origem.

Relativamente à janela principal, e em termos de aspecto geral, apenas são adicionados três novos botões. O botão 'Zoom' que abre uma janela com o gráfico da função e os elementos necessários para efectuar o *zoom* desejado para o gráfico; o botão 'Trace' que também abre uma janela parecida com a do Zoom, mas com os elementos necessários para fazer o movimento de um ponto através do gráfico; e o botão 'Repor valores originais' que faz com que seja desenhado o gráfico com os valores e as opções definidas por defeito.

```
> Button[Bzoom] ("Zoom", onclick=Az),  
  Button[Btrace] ("Trace", onclick=At),  
  Button[Brvo] ("Repor valores originais", onclick=Arvo),
```

Ao serem acrescentados estes elementos na Maplet, foi também modificado o *layout* da janela principal, de modo a que estes botões apareçam organizados na janela.

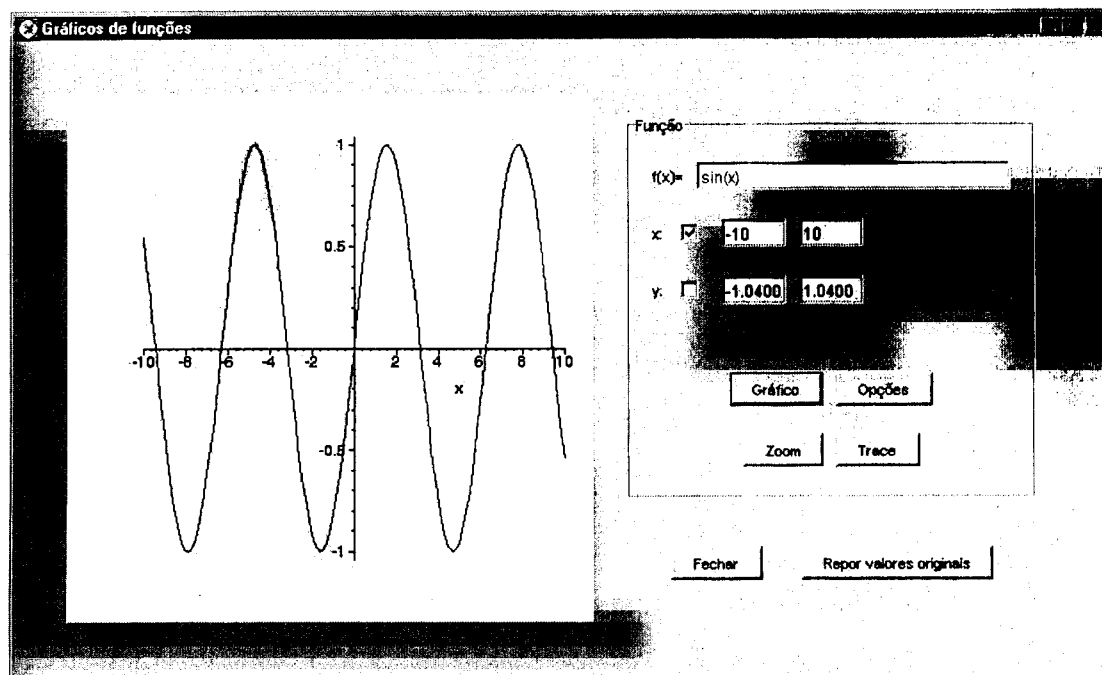
As acções 'Az' e 'At' fecham a janela principal e abrem respectivamente as janelas 'Zoom' e 'Trace', executando, desde logo, os procedimentos que fazem a representação gráfica da função.

```

> Action[Az](
  CloseWindow(Principal),
  RunWindow(Zoom),
  Evaluate(Pz='pfz(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cptes
p,Deix1,Dcor1,Cdc,Tnumpts]))')
),
Action[At](
  CloseWindow(Principal),
  RunWindow(Trace),
  Evaluate(Pt='pft(unapply(Tf,x),Txt,[Rlin,Dlinest1,Clinesp,Dptest1,Cpte
sp,Deix1,Dcor1,Cdc,Tnumpts]))')
),

```

O aspecto da janela principal com estes novos botões é o seguinte:



A janela do *zoom*, tem como título, "Zoom" e permite fazer um *zoom in* (reduzir os intervalos do eixo dos xx e do eixo dos yy) ou um *zoom out* (alargar os intervalos do eixo dos xx e do eixo dos yy) centrado na origem do referencial, ou seja, o utilizador pode

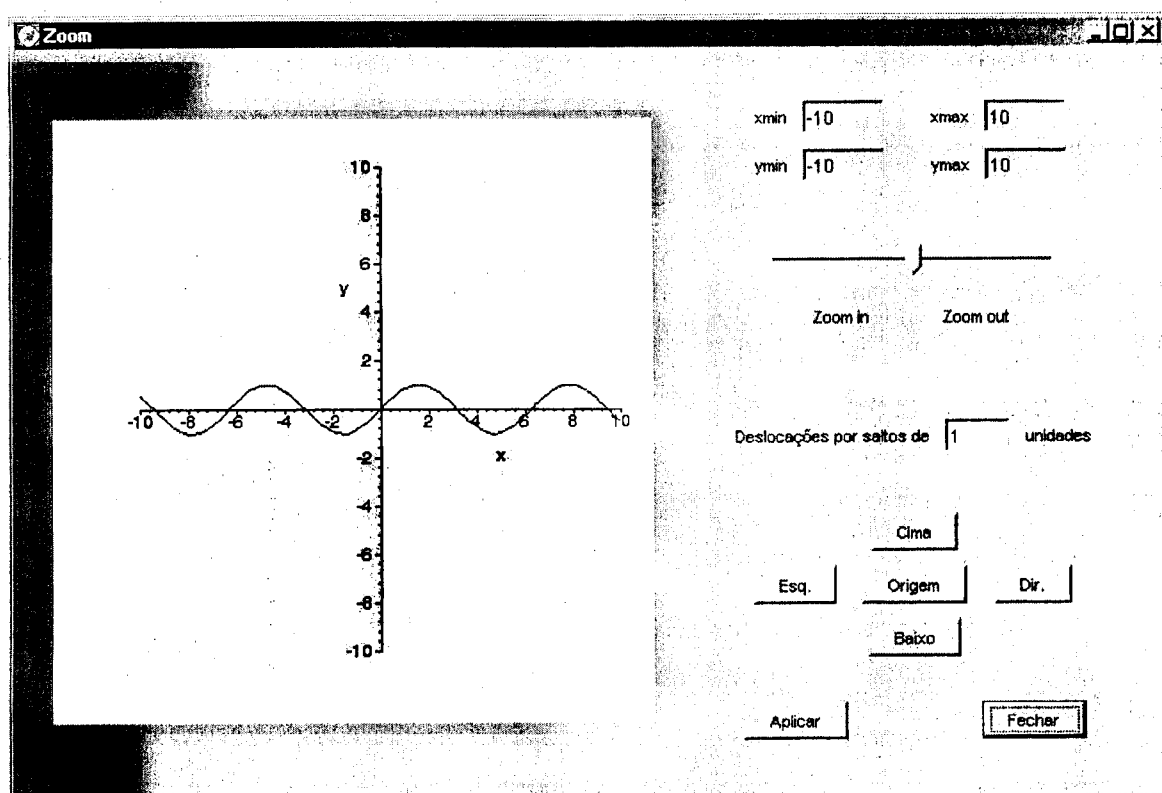
visualizar o gráfico modificando a amplitude dos intervalos do domínio e contradomínio que pode reduzir ou aumentar até 10 vezes mais. É também possível nesta janela, fazer deslocamentos verticais e horizontais, sendo assim possível visualizar as partes do gráfico que se deseja.

```
> Janelazoom:=Window[Zoom](title="Zoom",
[
[GridLayout(
(...)
)]
]),
Plotter[Pz](),
TextField[Txmin](width=5, editable=false),
TextField[Txmax](width=5, editable=false),
TextField[Tymin](width=5, editable=false),
TextField[Tymax](width=5, editable=false),
Slider[Sz](-8..10,1,filled=true,onchange=Asz),
TextField[Tslt](value="1",width=4),
Button[Bcima]("Cima",onclick=Azcima),
Button[Besq]("Esq.",onclick=Azesq),
Button[Borig]("Origem",onclick=Asz),
Button[Bdir]("Dir.",onclick=Azdir),
Button[Bbaixo]("Baixo",onclick=Azbaixo),
Button[Baplicz]("Aplicar",onclick=Aplz),
Button[Bfechz]("Fechar",onclick=Ajz):
```

O *layout* desta janela é construído com base no **GridLayout** onde são organizados os elementos contidos nesta janela.

A janela contém um **Plotter** (espaço reservado à representação gráfica da função) 'Pz' e quatro caixas de texto não editáveis ('Txmin', 'Txmax', 'Tymin' e 'Tymax') onde são exibidos os valores que definem os intervalos da variação do x e do y . À medida que se vai

"mexendo" na visualização do gráfico, estes valores vão sendo alterados e exibidos nestas caixas de texto. Um dos principais elementos desta janela é o **Slider** para fazer o *zoom in* ou o *zoom out*. A opção '-8..10' significa que a marca do **Slider** se vai mover numa barra desde a posição -8 até à posição 10. O valor da variável 'Sz' do **Slider** é igual à posição em que estiver a marca. A opção 'filled=true', serve para tornar visível o eixo onde o ponteiro do **Slider** se vai movimentar. A opção 'onchange=Asz' activa a acção 'Asz' quando o ponteiro é movimentado. Esta acção executa o procedimento 'pfz' que faz a representação gráfica da função com as alterações feitas através do **Slider** às variações do x e do y .



Esta janela também contém os botões que permitem ao utilizador fazer deslocações na visualização do gráfico. Os botões 'Cima', 'Esq.', 'Dir.' e 'Baixo' permitem deslocar o quadro de visualização do gráfico, respectivamente para cima, para a esquerda, para a direita e para baixo, o botão 'Origem' volta a mostrar o gráfico da função nos intervalos

definidos antes de se ter feito alguma deslocação. Por defeito, são feitas deslocações de uma unidade, em qualquer um dos sentidos, mas este valor pode ser alterado na caixa de texto 'Tslt', colocada no *layout* por cima destes botões.

Finalmente, para fechar a janela, o utilizador tem duas opções: ou clica no botão 'Fechar' que simplesmente fecha a janela, ou clica no botão 'Aplicar' que fecha a janela e "transporta" as alterações feitas à visualização do gráfico para a janela principal.

```
> Action[Asz](
  Evaluate(Pz='pfz(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cptes
p,Deix1,Dcor1,Cdc,Tnumpts])') ),
  Action[Ajz](
    CloseWindow(Zoom),
    RunWindow(Principal)
  ),
  Action[Aplz](
    CloseWindow(Zoom),
    RunWindow(Principal),
    Evaluate(P='pf(unapply(Tf,x),Chx,Txmin,Txmax,Chx,Tymin,Tymax,[Rlin,Dli
nest1,Clinesp,Dptest1,Cptes,Deix1,Dcor1,Cdc,Tnumpts])'),
    SetOption(Chy(value)=true),
    Evaluate(Xmin='Txmin'),
    Evaluate(Xmax='Txmax'),
    Evaluate(Ymin='Tymin'),
    Evaluate(Ymax='Tymax')
  ),
  Action[Azcima](
    Evaluate(Pz='pfc(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cptes
p,Deix1,Dcor1,Cdc,Tnumpts])') ),
  ),
  Action[Azbaixo](
    Evaluate(Pz='pfb(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cptes
p,Deix1,Dcor1,Cdc,Tnumpts])') ),
  ),
```

```

> Action[Azesq](
  Evaluate(Pz='pfe(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cptes
p,Deix1,Dcor1,Cdc,Tnumpts]))')
),
  Action[Azdir](
    Evaluate(Pz='pfd(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cptes
p,Deix1,Dcor1,Cdc,Tnumpts]))')
  ),
  Action[Aszo](
    Evaluate(Pz='pfzo(Sz,unapply(Tf,x),[Rlin,Dlinest1,Clinesp,Dptest1,Cpte
sp,Deix1,Dcor1,Cdc,Tnumpts]))')
  )

```

Estas acções fazem parte da Maplet 'mapletG', mas são accionadas a partir da janela 'Janelazoom'.

A acção 'Asz' executa o procedimento 'pfz' que mostra o gráfico da função para um determinado valor s definido no **Slider** e que diz respeito ao *zoom* do gráfico. Por de feito, esse valor s é igual a 1 e o utilizador pode escolher através do **Slider** o valor $s \in \{-8, -7, \dots, 9, 10\}$. A escolha destes valores para s , deve-se ao facto de se poder fazer *zoom* até 10 vezes mais perto ou 10 vezes mais longe.

s	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10
<i>Zoom</i>	$\frac{1}{10}$	$\frac{1}{9}$	$\frac{1}{8}$	$\frac{1}{7}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10

Assim, se no gráfico os valores do domínio variarem entre $xmin$ e $xmax$ e se os valores do eixo dos yy variarem entre $ymin$ e $ymax$, quando o utilizador, através do **Slider**, faz um *zoom in* ou um *zoom out*, alterando o valor de s , os valores do domínio passam a variar entre $xmin \times Zoom$ e $xmax \times Zoom$ e os valores do eixo dos yy passam a variar entre $ymin \times Zoom$ e $ymax \times Zoom$, onde *Zoom* é o valor correspondente pela tabela de cima ao valor de s que é indicado pelo ponteiro do **Slider**.

A acção 'Ajz' apenas fecha a janela 'Zoom' e abre a janela principal. Isto acontece quando o utilizador clica no botão 'Fechar' da janela 'Zoom'.

A acção 'Aplz' fecha a janela 'Zoom', abre a janela principal e actualiza o gráfico da janela principal com as modificações feitas na janela 'Zoom'. Isto acontece quando o utilizador clica no botão 'Aplicar' da janela 'Zoom'.

A acção 'Azcima' executa o procedimento 'pfc' que faz com que os limites verticais do gráfico aumentem, isto é, permite visualizar o gráfico um pouco mais acima. Por defeito, esse "salto" é de 1 unidade. Isto acontece quando o utilizador clica no botão 'Cima' da janela 'Zoom'. A acção 'Azbaixo' executa o procedimento 'pfb' que faz com que os limites verticais do gráfico diminuam. Esta acção é accionada quando o utilizador clica no botão 'Baixo' da janela 'Zoom'. As acções 'Azesq' e 'Azdir' executam respectivamente os procedimentos 'pfe' e 'pfd' que fazem com que os limites horizontais do gráfico diminuam ou aumentem. Isto acontece, se o utilizador clicar respectivamente nos botões 'Esq.' e 'Dir.' na janela 'Zoom'.

A acção 'Aszo' executa o procedimento 'pfzo' que faz com que o gráfico seja novamente visualizado na sua aparência original. Para que isto aconteça, o utilizador deverá clicar no botão 'Origem' na janela 'Zoom'.

Temos, para já, mais 6 novos procedimentos dos quais depende a funcionalidade da janela 'Zoom'.

```
> c:=0:d:=0:
> pfz:=proc(s,g,op1)
  local zx_min,zx_max,zy_min,zy_max,zxmin,zxmax,zymin,zymax,zxy:
  global r,zplt,zopc1,zopc,c,d:
```

Este procedimento recebe três parâmetros: 's' é o valor definido na janela do *zoom* pelo

Slider (valor inteiro entre -8 e 10), 'g' é a função definida na janela principal e 'op1' é a lista das opções do gráfico. Há variáveis usadas neste procedimento que são declaradas locais e outras que são declaradas globais, como 'c' e 'd' que inicialmente tomam o valor 0.

```
> if s<1 then
  r:=1/(2-s):
else
  r:=s:
end:
zx_min:=xmin*r+d:
zx_max:=xmax*r+d:
zy_min:=ymin*r+c:
zy_max:=ymax*r+c:
```

A declaração condicional 'if' serve para fazer a correspondência entre o valor s que vem do **Slider** e o *zoom* tal como é indicado na tabela anterior, ou seja, se $s \in \{-8, \dots, 0\}$ então $r = \frac{1}{2-s}$, se $s \in \{1, \dots, 10\}$ então $r = s$, onde r é agora o valor do *zoom* que vai ser multiplicado aos valores dos limites do gráfico, fazendo com que o campo de visualização do gráfico aumente ($r \geq 1$) ou diminua ($r < 1$). Nas variáveis 'zx_min', 'zx_max', 'zy_min' e 'zy_max' são guardados os valores dos limites do campo de visualização do gráfico com o *zoom* aplicado. Se não houver alterações, isto é, se $r = 1$, os limites são os mesmos que foram definidos na janela principal. Se não forem feitas deslocações verticais e horizontais do gráfico, os valores 'c' e 'd' valem 0.

```
> zxmin:=parse(convert(zx_min,string)):
  zxmax:=parse(convert(zx_max,string)):
  zymin:=parse(convert(zy_min,string)):
  zymax:=parse(convert(zy_max,string)):
  Maplets:-Tools:-Set(Txmin=zx_min):
  Maplets:-Tools:-Set(Txmax=zx_max):
```

```

> Maplets:-Tools:-Set(Tymin=zy_min):
Maplets:-Tools:-Set(Tymax=zy_max):
zopc1:=convert(sinteseop(op1),string):
zopc:=parse(convert(zopc1,string)):
zxy:=x=zxmin..zxmax,y=zymin..zymax:
zplt:=plot(g(x),zxy,zopc):
display(zplt);
end proc:

```

O comando 'Set', neste procedimento, atribui os valores dos limites do campo de visualização do gráfico às caixas de texto da janela *zoom*, próprias para esse efeito. O resto do procedimento é análogo ao que já foi feito no procedimento 'pf' já descrito anteriormente.

Para cada um dos botões 'Cima', 'Baixo', 'Esq.' e 'Dir.' é accionado um procedimento diferente, mas todos eles muito parecidos uns com os outros. Essa diferença reside no facto das variáveis 'c' e 'd' tomarem diferentes incrementos em cada um destes procedimentos.

```

> pfc:=proc(sc,gc,op1)
(...)
slt:=Maplets:-Tools:-Get(Tslt::algebraic):
c:=c+slt:
cx_min:=xmin*rc+d:
cx_max:=xmax*rc+d:
cy_min:=ymin*rc+c:
cy_max:=ymax*rc+c:
(...)
end proc:

```

O procedimento 'pfc' é accionado pelo botão 'Cima'. A variável 'slt' é definida pelo valor que "vem" da caixa de texto 'Tslt' da janela do *zoom*. Esse valor é escolhido pelo utilizador para definir com quantas unidades se vai deslocar o gráfico sempre que se clica num dos botões de deslocação. Por defeito, esse valor é 1, ou seja, se não for modificado,

quando se clica, por exemplo no botão 'Cima', o campo de visualização do gráfico desloca-se uma unidade para cima, ou seja, os limites verticais do gráfico aumentam em 1 unidade. Portanto, neste procedimento, tem de se fazer ' $c:=c+slt$:' para que os valores de ' $ymin$ ' e ' $ymax$ ', no *zoom* desejado, sejam aumentados segundo o valor escolhido em ' $Tslt$ '. O resto do procedimento é igual ao ' pfz '. Aliás, relativamente ao procedimento ' pfz ', podemos dizer que os procedimentos ' pfb ' executado através do botão 'Baixo', ' pfe ' executado através do botão 'Esq.' e ' pfd ' executado através do botão 'Dir.' são iguais, à excepção do incremento à variável ' c ' ou à variável ' d ', ou seja, em vez de ' $c:=c+slt$ ' temos em ' pfb ': ' $c:=c-slt$ '; em ' pfe ': ' $d:=d-slt$ '; e em ' pfd ': ' $d:=d+slt$ '. Isto garante que se na janela do *zoom*, o utilizador clicar em 'Cima', 'Baixo', 'Esq.' ou 'Dir.', o campo de visualização do gráfico desloca-se respectivamente para cima, para baixo, para a esquerda ou para a direita.

Quanto ao botão 'Origem', este faz executar o procedimento ' $pfzo$ ', que é igual ao procedimento ' pfz ', com a particularidade de voltar a atribuir às variáveis ' c ' e ' d ' o valor 0. Deste modo, o campo de visualização do gráfico volta a ser o mesmo que é mostrado quando é aberta a janela 'Zoom'.

A janela do *trace*, tem como título, "Trace" e permite visualizar um ponto pertencente ao gráfico da função, podendo fazer-se deslocar esse ponto através do gráfico para a direita, se for clicado o botão ' $-->>$ ', ou para a esquerda, se for clicado o botão ' $<<--$ '. Para além disso, é possível visualizar as coordenadas desse ponto e escolher uma abcissa para que o ponto seja localizado e encontrada a sua ordenada. Ainda nesta janela, é possível fazer uma espécie de *zoom*, pois o utilizador pode escolher a amplitude do intervalo das abcissas e a amplitude do intervalo das ordenadas. Os valores que definem estes intervalos também são exibidos nesta janela, para que o utilizador possa saber sempre em que campo de visualização se encontra o gráfico.

O código subjacente a esta janela é o seguinte:

```

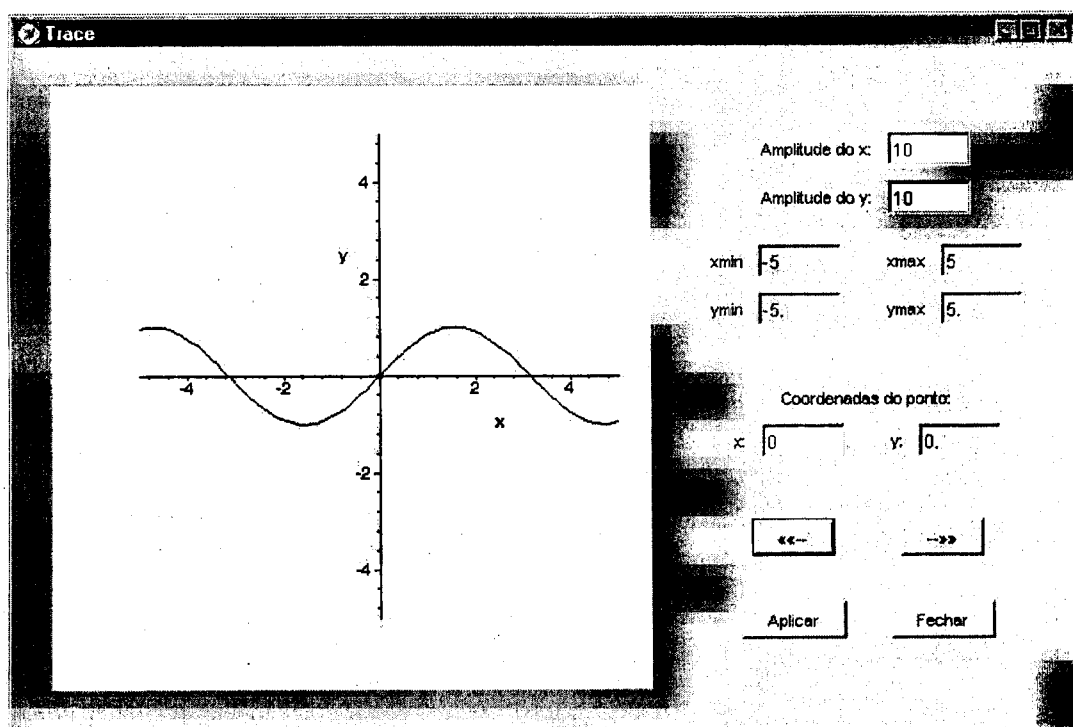
> Janelatrace:=Window[Trace] (title="Trace",
[
[GridLayout(
(...)
)]
]),
Plotter[Pt](),
TextField[Ttxamp] (value="10", width=5,onchange=Att),
TextField[Ttyamp] (value="10", width=5,onchange=Att),
TextField[Ttxmin] (width=5, editable=false),
TextField[Ttxmax] (width=5, editable=false),
TextField[Ttymin] (width=5, editable=false),
TextField[Ttymax] (width=5, editable=false),
TextField[Txt] (value="0",width=5,onchange=Att),
TextField[Tyt] (width=5, editable=false),
Button[Btesq] ("<<--",onclick=Atesq),
Button[Btdir] ("-->>",onclick=Atdir),
Button[Baplict] ("Aplicar",onclick=Aplt),
Button[Bfecht] ("Fechar",onclick=Ajt):

```

Tal como na janela do *zoom*, o *layout* desta janela é construído com base no **GridLayout** onde são organizados os elementos que vão fazer parte desta janela.

A janela 'Trace' também contém um **Plotter** denominado por 'Pt' onde vão constar o gráfico da função e ponto que pode ser deslocado ao longo do gráfico. As caixas de texto 'Ttxamp' e 'Ttyamp' servem para o utilizador escolher a amplitude dos intervalos que definem o campo de visualização do gráfico da função. Por defeito, a amplitude dos intervalos é 10. Os limites destes intervalos são exibidos nas caixas de texto 'Ttxmin', 'Ttxmax', 'Ttymin' e 'Ttymax' que são caixas de texto não editáveis. À medida que se vai movimentando o ponto, estes valores vão-se modificando. As caixas de texto 'Txt' e 'Tyt' representam as coordenadas do ponto que é exibido no gráfico. Estas caixas mudam

os seus valores à medida que as coordenadas do ponto vão sendo alteradas devido à sua movimentação ao longo do gráfico. A abcissa das coordenadas pode ser alterada pelo utilizador, deslocando-se o ponto de acordo com a abcissa escolhida. Depois de escolhida a abcissa, a ordenada é actualizada automaticamente de acordo com a função.



O botão 'Btesq' ('<<--') faz com que o ponto se movimente para a esquerda e o botão 'Btdir' ('-->>') faz com que o ponto se movimente para a direita.

O botão 'Fechar' fecha simplesmente a janela 'Trace', abrindo a janela principal e o botão 'Aplicar' fecha a janela 'Trace' e abre a janela principal exibindo o gráfico da função com as alterações feitas na janela 'Trace'.

A partir desta janela são accionadas algumas acções que são definidas na 'mapletG'.

```

> Action[Att](
  Evaluate(Pt='pft(unapply(Tf,x),Txt,[Rlin,Dlinest1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts],Ttxamp,Ttyamp)')
),
  Action[Ajt](
    CloseWindow(Trace),
    RunWindow(Principal)
  ),
  Action[Aplt](
    CloseWindow(Trace),
    RunWindow(Principal),
    Evaluate(P='pf(unapply(Tf,x),Chx,Ttxmin,Ttxmax,Chx,Ttymin,Ttymax,[Rlin,Dlinest1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts])'),
    SetOption(Chy(value)=true),
    Evaluate(Xmin='Ttxmin'),
    Evaluate(Xmax='Ttxmax'),
    Evaluate(Ymin='Ttymin'),
    Evaluate(Ymax='Ttymax')
  ),
  Action[Atesq](
    Evaluate(Pt='pfte(unapply(Tf,x),Txt,[Rlin,Dlinest1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts],Ttxamp,Ttyamp)')
  ),
  Action[Atdir](
    Evaluate(Pt='pftd(unapply(Tf,x),Txt,[Rlin,Dlinest1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts],Ttxamp,Ttyamp)'))
),

```

A acção 'Att' executa o procedimento 'pft' que mostra o gráfico da função f e um ponto com coordenadas $(0, f(0))$, podendo o utilizador modificar a abcissa desse ponto visualizando-se o gráfico sempre centrado nesse ponto com a amplitude dos intervalos desejada.

A acção 'Ajt' apenas fecha a janela 'Trace' e abre a janela principal. Isto acontece quando

o utilizador clica no botão 'Fechar' da janela 'Trace'.

A acção 'Aplt' fecha a janela 'Trace', abre a janela principal e actualiza o gráfico da janela principal com as modificações feitas na janela 'Trace'. Isto acontece quando o utilizador clica no botão 'Aplicar' da janela 'Trace'.

A acção 'Atesq' executa o procedimento 'pfte' que faz com que o ponto sob o gráfico seja deslocado para esquerda para um ponto de abcissa mais pequeno um valor que abcissa anterior. Por exemplo, se o referido ponto tiver coordenadas $(0, f(0))$, ao ser executado este procedimento apenas uma vez, as suas coordenadas passam a ser $(-1, f(-1))$. Isto acontece, quando o utilizador clica no botão '<<--' da janela 'Trace'. A acção 'Atdir' executa o procedimento 'pftd' que faz o mesmo que o procedimento anterior, mas o ponto é deslocado para a direita, isto é, se, por exemplo, o ponto tem coordenadas $(0, f(0))$, ao ser executado este procedimento apenas uma vez, as suas coordenadas passam a ser $(1, f(1))$. Isto acontece quando o utilizador clica no botão '-->>' da janela 'Trace'.

Isto significa que temos já mais 3 novos procedimentos: 'pft', 'pfte' e 'pftd'.

```
> pft:=proc(gt,xt,op1,xamp,yamp)
  local l,tx_min,tx_max,ty_min,ty_max,txmin,txmax,tymin,tymax,txy:
  global rt,tpltfunc,tpltpto,topc1,topc:
  rt:=xt:
  tx_min:=rt-xamp/2:
  tx_max:=rt+xamp/2:
  ty_min:=evalf(gt(rt)-yamp/2,4):
  ty_max:=evalf(gt(rt)+yamp/2,4):
```

Este procedimento recebe 5 parâmetros: 'gt' é a função definida na janela principal, 'xt' é a abcissa do ponto visível sob o gráfico na janela 'Trace', 'op1' é a lista das opções do gráfico, 'xamp' é a amplitude do intervalo das abcissas e 'yamp' é a amplitude do intervalo das ordenadas. Tal como nos outros procedimentos, declaram-se no início as

variáveis locais e globais. Atribui-se a 'rt' a abcissa do ponto que vai ser deslocado ao longo do gráfico. Para que esse ponto apareça centrado no espaço da janela reservado ao gráfico da função, faz-se com que o valor mínimo das abcissas seja rt diminuído de metade da amplitude escolhida para o intervalo das abcissas ($tx_min:=rt-xamp/2$), e que o valor máximo das abcissas seja rt aumentado de metade da referida amplitude ($tx_max:=rt+xamp/2$). Analogamente, para o intervalo das ordenadas, faz-se $ty_min:=gt(rt)-yamp/2$ e $ty_max:=gt(rt)+yamp/2$ onde 'gt' é a função definida na janela principal e 'yamp' é a amplitude do intervalo das ordenadas. O comando 'evalf(expr,4)' arredonda a expressão para um valor numérico com 4 algarismos significativos.

```
> txmin:=parse(convert(tx_min,string)):
   txmax:=parse(convert(tx_max,string)):
   tymin:=parse(convert(ty_min,string)):
   tymax:=parse(convert(ty_max,string)):
   Maplets:-Tools:-Set(Ttxmin=tx_min):
   Maplets:-Tools:-Set(Ttxmax=tx_max):
   Maplets:-Tools:-Set(Ttymin=ty_min):
   Maplets:-Tools:-Set(Ttymax=ty_max):
   Maplets:-Tools:-Set(Txt=rt):
   Maplets:-Tools:-Set(Tyt=evalf(gt(rt),4)):
   topc1:=convert(sinteseop(op1),string):
   topc:=parse(convert(topc1,string)):
   txy:=x=txmin..txmax,y=tymin..tymax:
   tpltfunc:=plot(gt(x),txy,topc):
   l:= [[ n, gt(n)] $n=rt..rt]:
   tpltpto:=plot(l, txy, style=point, symbol=circle, color=black,
   symbolsize=40):
   display(tpltfunc,tpltpto);
end proc:
```

O comando 'Set', neste procedimento, atribui os valores dos limites do campo de vi-

sualização do gráfico e das coordenadas do ponto às caixas de texto da janela 'Trace', próprias para esse efeito. Em 'tpltfunc' é guardada a representação gráfica da função e em 'tpltpto' é guardada a representação gráfica do ponto. O gráfico e o ponto são mostrados simultaneamente através do comando 'display'.

Os procedimentos 'pfte' e 'pftd' são executados quando o utilizador clica respectivamente nos botões '<<--' e '-->>' da janela 'Trace'. Estes procedimentos são praticamente iguais ao procedimento 'pft', mas em 'pfte' diminui-se uma unidade ao valor da abcissa do ponto que se desloca no gráfico ($rt:=xte-1$;) e em 'pftd' aumenta-se uma unidade ao valor da abcissa do referido ponto ($rt:=xtd+1$;).

```
> pfte:=proc(gte,xte,op1,xampe,yampe)
  local
  le,tex_min,tex_max,tey_min,tey_max,texmin,texmax,teymin,teymax,texy:
  global rt,tepltfunc,tepltppto,teopc1,teopc:
  rt:=xte-1:
  tex_min:=rt-xampe/2:
  tex_max:=rt+xampe/2:
  tey_min:=evalf(gte(rt)-yampe/2,4):
  tey_max:=evalf(gte(rt)+yampe/2,4):
  (...)
end proc:
> pftd:=proc(gtd,xtd,op1,xampd,yampd)
  local
  ld,tdx_min,tdx_max,tdy_min,tdy_max,tdxmin,tdxmax,tdymin,tdymax,tdxy:
  global rt,tdpltfunc,tdpltppto,tdopc1,tdopc:
  rt:=xtd+1:
  tdx_min:=rt-xampd/2:
  tdx_max:=rt+xampd/2:
  tdy_min:=evalf(gtd(rt)-yampd/2,4):
  tdy_max:=evalf(gtd(rt)+yampd/2,4):
  (...)
end proc:
```

Voltando á janela principal, tomemos atenção a um botão que aparece ao lado do botão 'Fechar', o botão 'Repor valores originais'. Ao clicar neste botão, o utilizador acciona a acção 'Arvo' que repõe os valores originais dos limites do gráfico, como quando se abre a janela.

```
> Action[Arvo](
  SetOption(Chy(value)=false),
  Evaluate(Xmin=-10),
  Evaluate(Xmax=10),
  Evaluate(Ymin=-10),
  Evaluate(Ymax=10),
  Evaluate(P='pf(unapply(Tf,x),Chx,Xmin,Xmax,Chy,Ymin,Ymax,[Rlin,Dlinest
1,Clinesp,Dptest1,Cptesp,Deix1,Dcor1,Cdc,Tnumpts]))')
),
```

Para que a Maplet inclua as janelas do *zoom* e do *trace*, é necessário incluir na 'mapletG' as referências relativas a estas janelas ('Janelazoom' e 'Janelatrace').

```
> Janelazoom,
  Janelatrace,
```

5.4 A janela sobre o estudo do gráfico

Para que o estudo gráfico de funções ficasse mais completo, adicionou-se mais uma janela que fizesse um estudo aos zeros da função, aos extremos relativos e absolutos, à monotonia e às concavidades, tanto a nível analítico como gráfico.

É de referir que esta janela está construída para serem estudadas funções diferenciáveis. Para que a janela tenha funcionalidade, as funções têm de permitir que o Maple as consiga derivar duas vezes, o número de zeros da primeira e da segunda derivada tem de ser finito

e sujeito às potencialidades do Maple, uma vez que para estudar a monotonia e as concavidades de uma função, a Maplet assume que é possível determinar a primeira e a segunda derivada de f . Se tal não for possível poderão surgir erros na utilização desta janela.

Assim, na janela principal vai aparecer um novo botão, 'Estudo', que abre uma janela onde pode ser feito o estudo do gráfico da função.

```
> Button[Bestudo]("Estudo", onclick=Ae),
```

Quando se pressiona neste botão é accionada a acção 'Ae', que fecha a janela principal e abre a janela 'Estudo', executando os procedimentos que permitem visualizar o gráfico da função, a expressão geral da função e o intervalo de variação das abcissas.

```
> Action[Ae](  
  CloseWindow(Principal),  
  RunWindow(Estudo),  
  Evaluate(Pe='GrFunc(Tf,Xmin,Xmax)'),  
  Evaluate(Tfe='Tf'),  
  Evaluate(Txmine='Xmin'),  
  Evaluate(Txmaxe='Xmax')  
)
```

O aspecto da janela principal com este novo botão é o da figura 5.1.

A janela 'Estudo', tem como título, "Estudo da função" e permite visualizar o gráfico da função num certo intervalo, e um espaço onde serão mostrados analiticamente, conjuntos e intervalos de números reais respeitantes ao estudo que se estiver a fazer do gráfico. Assim, se quisermos, por exemplo, saber quais os máximos relativos da função no intervalo definido, podemos visualizar analiticamente as coordenadas dos pontos, e graficamente

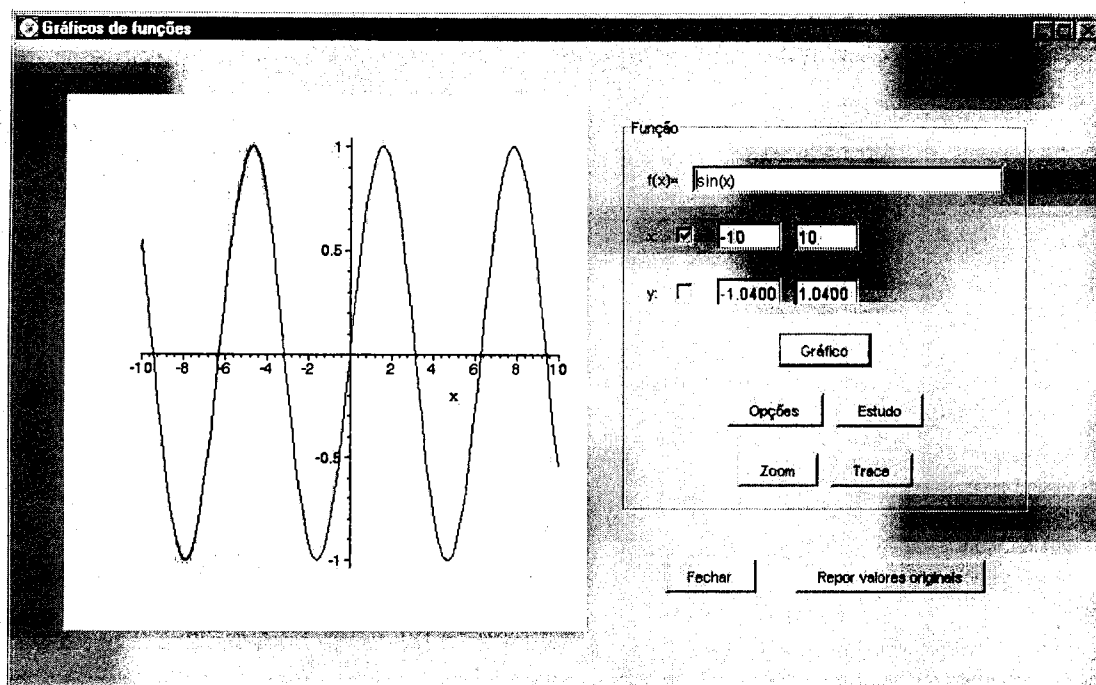


Figura 5.1: Janela principal com o botão 'Estudo'

segmentos que projectam esses pontos no eixo dos xx . Existe também, nesta janela, um botão de ajuda, que abre uma outra janela. Esta pequena janela inclui um texto com informações sobre o funcionamento da janela 'Estudo'.

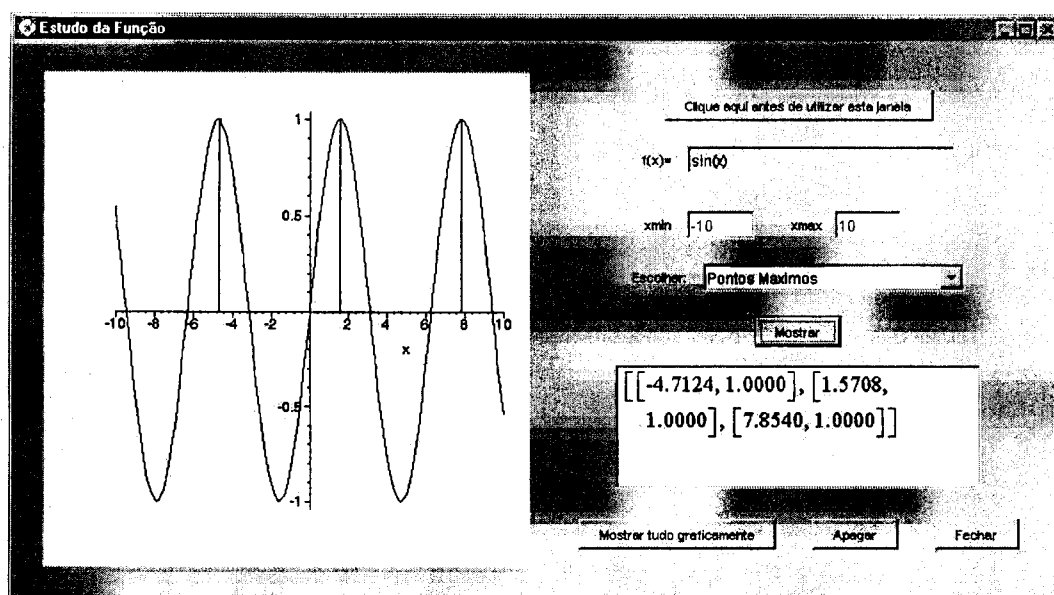
```
> Janelaestudo:=Window[Estudo](title="Estudo
da Função",
[
[GridLayout(
(...))
]),
Plotter[Pe](),
Button[Bler]("Clique aqui antes de utilizar esta janela",
Action(RunWindow(Lerestudo)))],
```

```
> TextField[Tfe](editable=false),  
  TextField[Txmine](width=5),  
  TextField[Txmaxe](width=5),  
  DropDownBox[Dde](["Zeros", "Pontos  
  Maximos", "Pontos Minimos", "Intervalos de Crescimento",  
  "Intervalos de Decrescimento", "Concavidades Voltadas para Cima",  
  "Concavidades Voltadas para Baixo"]),  
  Button[Bmos]("Mostrar", onclick=Aem),  
  MathMLViewer[MMLVe](),  
  Button[Bmtg]("Mostrar tudo graficamente", onclick=Aemt看),  
  Button[Bapa]("Apagar", onclick=Apa),  
  Button[Bfeche]("Fechar", onclick=Aje):
```

O *layout* desta janela é construído com base no **GridLayout** onde são organizados os elementos contidos nesta janela.

A janela contém um **Plotter** 'Pe', onde é exibido o gráfico da função. O primeiro botão desta janela, é o botão 'Clique aqui antes de utilizar esta janela', que abre uma janela com um texto informativo acerca do funcionamento desta janela. A expressão geral da função aparece numa caixa de texto ('Tfe') não editável, pois não é importante nesta janela que se mude de função, uma vez que isso pode ser feito na janela principal. Já o domínio da função pode ser alterado, pois o utilizador pode querer ver o comportamento da função noutros intervalos de domínio. Essas alterações podem ser feitas nas caixas de texto 'Txmine' e 'Txmaxe'. Um dos principais elementos desta janela é a **DropDownBox**, onde o utilizador pode escolher o que deseja estudar relativamente à função e ao seu gráfico. Assim, ele pode escolher entre 'Zeros', 'Pontos Máximos', 'Pontos Mínimos', 'Intervalos de Crescimento', 'Intervalos de Decrescimento', 'Concavidades Voltadas para Cima' ou 'Concavidades Voltadas para Baixo'. Depois de escolher uma destas opções, o utilizador só tem de clicar no botão 'Mostrar', que aparece logo a seguir à **DropDownBox**. Os resultados desta acção, são exibidos graficamente no **Plotter** e analiticamente no **MathMLViewer** (caixa onde as

expressões aparecem num formato de notação matemática).

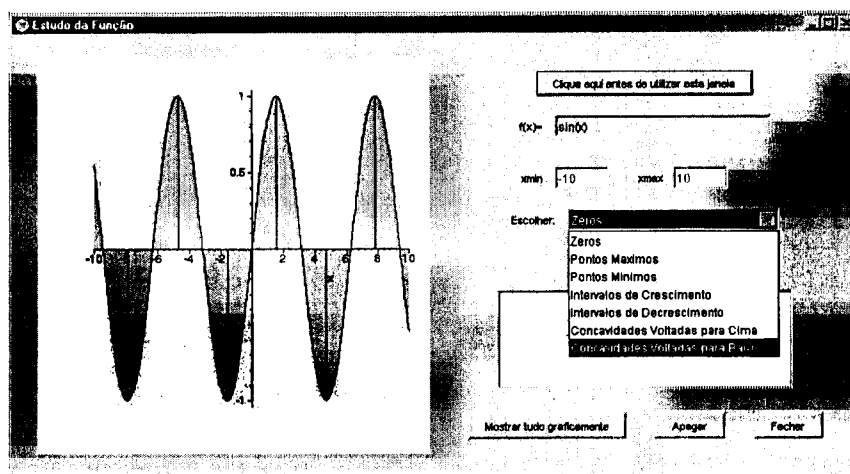


Assim, se o utilizador escolher os "Zeros" da função, ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os zeros da função, no intervalo dado. Gráficamente o utilizador visualiza os zeros, quando o gráfico da função intersecta o eixo dos xx . Se escolher os "Pontos Máximos", ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os pontos em que a função atinge um máximo relativo ou absoluto, no intervalo dado. Gráficamente, aparecerão segmentos azuis que projectam estes pontos no eixo dos xx . Se escolher os "Pontos Mínimos", ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os pontos em que a função atinge um mínimo relativo ou absoluto, no intervalo dado. Gráficamente, aparecerão segmentos vermelhos que projectam estes pontos no eixo dos xx . Se escolher os "Intervalos de Crescimento", ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os sub-intervalos do intervalo dado, em que a função é crescente. Gráficamente, aparecerá a azul a parte do gráfico em que a função é crescente. Se escolher os "Intervalos de Decrescimento", ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os sub-intervalos do intervalo dado, em que a função é decrescente. Gráficamente, aparecerá a vermelho a parte do gráfico em que a função é decrescente. Se

escolher as "Concavidades Voltadas para Cima", ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os sub-intervalos do intervalo dado, em que a função tem a concavidade voltada para cima. Gráficamente, aparecerá pintada a roxo a região entre o gráfico e o eixo dos xx em cada intervalo onde a concavidade está voltada para cima. Se escolher as "Concavidades Voltadas para Baixo", ao clicar no botão 'Mostrar', aparecerão na caixa em branco todos os sub-intervalos do intervalo dado, em que a função tem a concavidade voltada para baixo. Gráficamente, aparecerá pintada a rosa a região entre o gráfico e o eixo dos xx em cada intervalo onde a concavidade está voltada para baixo.

A janela ainda contém mais três botões. O botão 'Mostrar tudo graficamente' serve, como o nome o indica, para mostrar graficamente todas as opções da **DropDownBox** em simultâneo. Neste caso, a caixa do **MathMLViewer**, fica em branco. O botão 'Apagar', faz com que o gráfico seja exibido na sua forma original, sem referências às opções que podem ser escolhidas. Este botão, também faz com que a caixa do **MathMLViewer** fique em branco. Finalmente, o botão 'Fechar', fecha esta janela e abre a janela principal.

A seguinte figura, mostra o aspecto da janela 'Estudo', depois de se clicar no botão 'Mostrar tudo graficamente'. Aproveitou-se também para se exibir a lista de opções da **DropDownBox**:



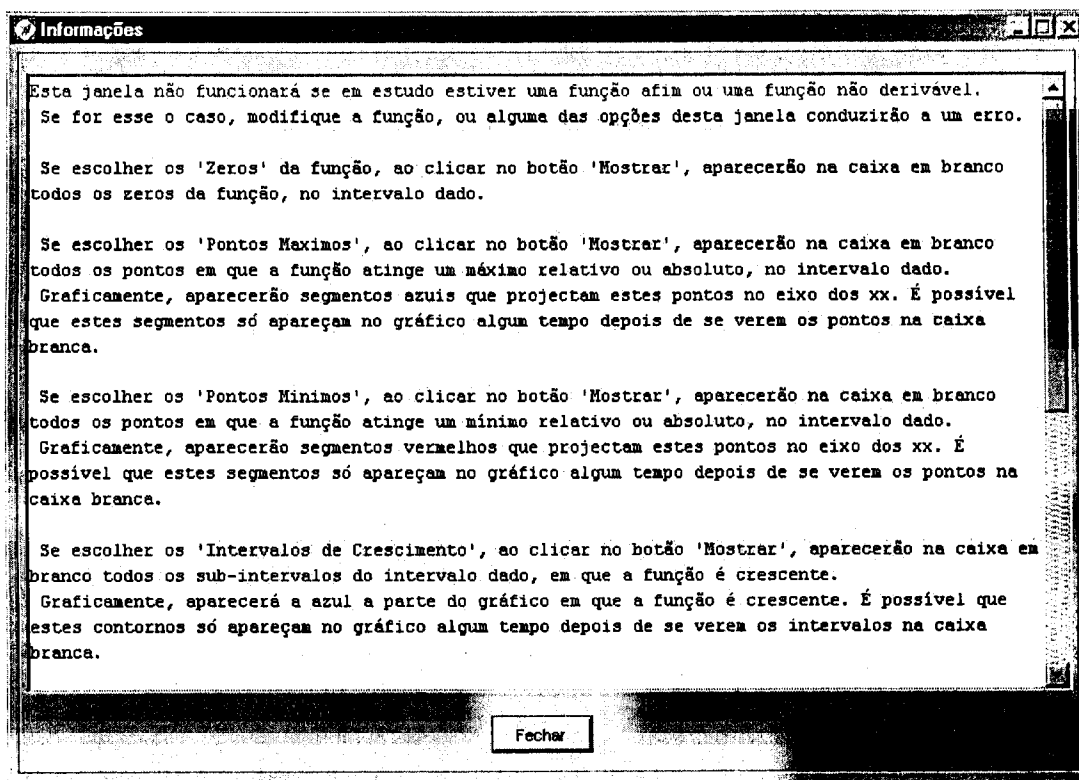
Ao clicar no botão 'Clique aqui antes de utilizar esta janela', é accionada a janela 'Lerestudo'. Esta janela é construída e guardada na variável 'Janelaler'.

```
> Janelaler:=Window[Lerestudo](
    'resizable'='false',
    'title'="Informações",
    BoxColumn('border'='true', 'inset'=0, 'spacing'=8,
    BoxCell(
    TextBox('height'=24, 'width'=100,
    'editable'='false', 'foreground'="#333399",
    'value'=infoestudo
    )
    ),
> BoxRow('inset'=0, 'spacing'=0,
    Button("Fechar",
    CloseWindow(Lerestudo))
    )
    )
):
```

Esta janela tem como título 'Informações', e contém uma **TextBox** onde é inserido um texto informativo sobre o funcionamento da janela 'Estudo'. Esse texto está guardado no parâmetro 'infoestudo'. A janela fecha-se, quando se clica no botão 'Fechar'.

É com o botão 'Mostrar' que se conseguem obter os resultados que são pretendidos na utilização desta janela. Quando se clica neste botão, é accionada a acção 'Aem' que faz com que sejam executados os procedimentos 'MMLop' e 'GraficoEst'.

```
> Action[Aem](
    Evaluate(MMLVe='MMLop()'),
    Evaluate(Pe='GraficoEst()')
    ),
```



O procedimento 'MMLOp', dependendo da escolha feita na **DropDownBox**, executa os procedimentos que mostram os resultados analíticos na caixa **MathMLViewer**. O procedimento 'GraficoEst', também dependendo da escolha feita na **DropDownBox**, executa os procedimentos relativos aos resultados gráficos que aparecem no **Plotter**.

O procedimento 'MMLZer' determina uma lista com os zeros da função no intervalo definido.

```
> MMLZer := proc(fe, negX, posX)
  local n, i, delta, li, ffe, X1, X2, sol, sol1:
  n := 100:li := []:
  ffe:=unapply(fe,x):
  delta := (evalf(posX) - evalf(negX)) / n:
  X2 := evalf(negX):
```

```

> for i from 1 to n do
  X1 := evalf(X2):  X2 := evalf(negX + i * delta):
  sol := fsolve(f(x)=0, x=X1..X2):
  if nops([sol])>1 then
    sol1:=sol[1]:
  else sol1:=sol:
  end if:
  if not member(sol1, li) then
    li := [ op(li), sol1]:
  end if:
end do:
li;
end proc:

```

Pelo que já foi visto no capítulo 2, poderíamos dizer que para achar os zeros de uma função, bastava apenas utilizar o comando 'fsolve', ou seja, se f for a nossa função no intervalo $[x_{min}, x_{max}]$, ao fazer no Maple 'fsolve(f(x)=0, x=xmin..xmax);', vamos obter os zeros da função f . Acontece, que para algumas funções, especialmente as trigonométricas, o **fsolve** não determina todas as soluções possíveis num dado intervalo. Assim, foi necessário fazer um procedimento, que dividisse o intervalo dado em intervalos muito pequenos, de forma a fazer um **fsolve** para cada um desses pequenos intervalos. Desta maneira, se houver zeros nesses intervalinhos, eles são adicionados a uma lista (a lista 'li'). É claro que há a possibilidade de nesses pequenos intervalos haver mais do que um zero da função, e o **fsolve** não conseguir detectá-los, mas, se na visualização do gráfico não for clara a existência de apenas um zero da função num determinado sub-intervalo, pode-se sempre fazer um estudo mais pormenorizado num intervalo mais pequeno. Desta maneira, o procedimento divide este novo intervalo em muitos intervalos ainda mais pequenos, e assim sucessivamente. Nesta Maplet, escolheu-se '**n:=100**:', ou seja, o intervalo é dividido em 100 pequenos intervalos.

O procedimento '**MMLMax**' determina uma lista com as coordenadas dos pontos que são

máximos relativos ou absolutos da função no intervalo definido.

```
> MMLMax:=proc(fe, negX, posX)
  local Max, ffe, PtsCrit, i, Diffunc;
  Max:=[]:
  ffe:=unapply(fe,x):
  PtsCrit:=MMLZer(diff(fe,x),negX,posX):
  Diffunc:=diff(ffe(x), x):
```

Neste procedimento, acham-se os pontos críticos da função, ou seja, os zeros da primeira derivada. A lista 'PtsCrit' é constituída por todos os pontos críticos do intervalo definido. Se esta lista estiver vazia, o procedimento acaba e mostra uma lista de pontos máximos vazia. Se a lista tiver apenas 1 ponto crítico, o procedimento verifica se esse ponto é máximo. Suponhamos que x_0 é o único ponto crítico da lista 'PtsCrit' para o intervalo $[min, max]$. Se $f'(\frac{min+x_0}{2}) > 0$ e $f'(\frac{x_0+max}{2}) < 0$ então $(x_0, f(x_0))$ é um máximo de f . É desta forma que o procedimento verifica se existe ou não um máximo.

```
> if PtsCrit <> [] then
  if nops(PtsCrit)=1 then
    if evalf(subs(x=(negX+PtsCrit[1])/2, Diffunc))>0 and
      evalf(subs(x=(PtsCrit[1]+posX)/2, Diffunc))<0 then
      Max:=[op(Max), [PtsCrit[1], value(subs(x=PtsCrit[1], fe))]];
    end if:
  end if:
  if nops(PtsCrit) <> 1 then
    for i from 1 to nops(PtsCrit) do
      if i=1 then
        if evalf(subs(x=(negX+PtsCrit[1])/2, Diffunc))>0 and
          evalf(subs(x=(PtsCrit[1]+PtsCrit[2])/2, Diffunc))<0 then
          Max:=[op(Max), [PtsCrit[1], value(subs(x=PtsCrit[1], fe))]];
        end if:
      end if:
    end for:
  end if:
end if:
```

```

> end if
  elif i <> nops(PtsCrit) then
    if evalf(subs(x=(PtsCrit[i-1]+PtsCrit[i])/2, Diffunc))>0 and
      evalf(subs(x=(PtsCrit[i+1]+PtsCrit[i])/2, Diffunc))<0 then
      Max:=[op(Max), [PtsCrit[i], value(subs(x=PtsCrit[i],
        fe))]];
    end if:
  elif i=nops(PtsCrit) then
    if evalf(subs(x=(PtsCrit[i-1]+PtsCrit[i])/2, Diffunc))>0 and
      evalf(subs(x=((posX+PtsCrit[i])/2), Diffunc))<0 then
      Max:=[op(Max), [PtsCrit[i], value(subs(x=PtsCrit[i],
        fe))]];
    end if:
  end if:
end do;
end if:
end if:
Max;
end proc:

```

Se houver mais do que um ponto crítico na lista (se $\text{PtsCrit}=[x_1, \dots, x_i, \dots, x_n]$), o procedimento começa por verificar o primeiro ponto crítico x_1 . Se $f'(\frac{\min+x_1}{2}) > 0$ e $f'(\frac{x_1+x_2}{2}) < 0$, onde x_2 é o segundo ponto crítico da lista, então $(x_1, f(x_1))$ é um máximo de f . Depois, verifica os outros pontos críticos x_i . Se $f'(\frac{x_{i-1}+x_i}{2}) > 0$ e $f'(\frac{x_i+x_{i+1}}{2}) < 0$, então $(x_i, f(x_i))$ é um máximo de f . Finalmente, analisa o último ponto crítico x_n . Se $f'(\frac{x_{n-1}+x_n}{2}) > 0$ e $f'(\frac{x_n+\max}{2}) < 0$, então $(x_n, f(x_n))$ é um máximo de f .

De uma forma geral, verificar se um determinado ponto crítico é ou não máximo, é verificar se entre o ponto crítico em estudo e o ponto crítico anterior (ou o valor mínimo do domínio) a função é crescente e se entre o ponto crítico em estudo e o ponto crítico posterior (ou o valor máximo do domínio) a função é decrescente. Se isto acontecer, é porque a imagem do ponto crítico em estudo é maior que as imagens dos outros pontos vizinhos.

O procedimento 'MMLMin', funciona de forma análoga ao procedimento anterior.

O procedimento 'MMLCres', determina os intervalos onde a função é crescente no intervalo definido.

```
> MMLCres:=proc(fe, negX, posX)
  local Cres, ffe, Pts, d1, i;
  Cres:=[]:
  ffe:=unapply(fe,x):
  d1:=diff(ffe(x), x):
  Pts:=MMLZer(diff(fe,x),negX,posX):
```

Neste procedimento, também se acham os pontos críticos da função. A lista 'Pts' é constituída pelos pontos críticos da função no intervalo definido.

```
> if not member(evalf(negX), Pts) then
  Pts:=[evalf(negX), op(sort(Pts))]:
end if:
if not member(evalf(posX), Pts) then
  Pts:=[op(sort(Pts)), evalf(posX)]:
end if:
for i from 1 to (nops(Pts)-1) do
  if evalf(subs(x=((Pts[i]+Pts[i+1])/2), d1))>0 then
    Cres:=[op(Cres), [Pts[i], Pts[i+1]]]:
  end if;
end do;
Cres;
end proc:
```

Consideremos a lista Pts de pontos críticos de f no intervalo $[min, max]$. Se incluirmos nesta lista os extremos do intervalo min e max , a lista Pts passa a ser definida como

$Pts = [x_1, \dots, x_i, \dots, x_n]$ onde $x_1 = \min$, $x_n = \max$ e $x_i, i \neq 1, n$ representa(m) o(s) ponto(s) crítico(s) de f . Se no intervalo $[\min, \max]$ não existirem pontos críticos de f , a lista Pts é apenas definida por $Pts = [x_1, x_2]$ onde x_1 e x_2 representam respectivamente o valor $\min$ e o valor $\max$. Se verificarmos que $f'(\frac{x_i + x_{i+1}}{2}) > 0$, então no intervalo $[x_i, x_{i+1}]$, a função f é crescente. Este teste é feito para todos os $n - 1$ intervalos adjacentes que se podem definir com os consecutivos elementos da lista Pts .

O procedimento 'MMLDec', que determina os intervalos em que a função é decrescente, é construído de forma análoga ao procedimento 'MMLCres'.

O procedimento 'MMLConcC', determina os intervalos onde a função tem as concavidades voltadas para cima no intervalo definido.

```
> MMLConcC:=proc(fe, negX, posX)
  local ConcC, Pts, d2, i;
  ConcC:=[]:
  d2:=diff(fe, x$2):
  Pts:=MMLZer(d2,negX,posX):
```

Neste procedimento, acham-se os pontos de inflexão da função, ou seja, os zeros da segunda derivada de f . A lista 'Pts', neste procedimento, é a lista de todos os pontos de inflexão da função, no intervalo definido.

```
> if not member(evalf(negX), Pts) then
  Pts:=[evalf(negX), op(sort(Pts))]:
end if:
if not member(evalf(posX), Pts) then
  Pts:=[op(sort(Pts)), evalf(posX)]:
end if:
```

```

> for i from 1 to (nops(Pts)-1) do
  if evalf(subs(x=((Pts[i]+Pts[i+1])/2), d2))>0 then
    ConcC:=[op(ConcC), [Pts[i], Pts[i+1]]];
  end if;
end do;
ConcC;
end proc:

```

Consideremos a lista *Pts* de pontos de inflexão de f no intervalo $[min, max]$. Se incluirmos nesta lista os extremos do intervalo min e max , a lista *Pts* passa a ser definida como $Pts=[x_1, \dots, x_i, \dots, x_n]$ onde $x_1 = min$, $x_n = max$ e $x_i, i \neq 1, n$ representa(m) o(s) ponto(s) de inflexão de f . Se no intervalo $[min, max]$ não existirem pontos de inflexão de f , a lista *Pts* é apenas definida por $Pts:=[x_1, x_2]$ onde x_1 e x_2 representam respectivamente o valor min e o valor max . Se verificarmos que $f''\left(\frac{x_i+x_{i+1}}{2}\right) > 0$, então no intervalo $[x_i, x_{i+1}]$, podemos garantir que a função f tem a concavidade voltada para cima. Este teste é feito para todos os $n - 1$ intervalos adjacentes que se podem definir com os consecutivos elementos da lista *Pts*.

O procedimento 'MMLConcB', que determina os intervalos onde a função tem as concavidades voltadas para baixo, é construído de forma análoga ao procedimento 'MMLConcC'.

Estes procedimentos são accionados a partir do 'MMLOp'. Este procedimento, a partir da escolha que é feita na **DropDownBox**, pelo utilizador na janela 'Estudo', "manda" executar o procedimento adequado à selecção feita. Se, por exemplo, na **DropDownBox** estiver seleccionado 'Zeros', este procedimento faz com que seja executado 'MMLZer', o procedimento que determina a lista dos zeros da função no intervalo definido.

```

> MMLOp:= proc()
  if Maplets:-Tools:-Get(Dde) = "Zeros" then
    MMLZer(Maplets:-Tools:-Get(Tfe::algebraic),
    Maplets:-Tools:-Get(Txmine::algebraic),
    Maplets:-Tools:-Get(Txmaxe::algebraic));
  elif Maplets:-Tools:-Get(Dde) = "Pontos Maximos" then
    MMLMax(Maplets:-Tools:-Get(Tfe::algebraic),
    Maplets:-Tools:-Get(Txmine::algebraic),
    Maplets:-Tools:-Get(Txmaxe::algebraic));
  (...)
  elif Maplets:-Tools:-Get(Dde) = "Concavidades Voltadas para Baixo"
  then
    MMLConcB(Maplets:-Tools:-Get(Tfe::algebraic),
    Maplets:-Tools:-Get(Txmine::algebraic),
    Maplets:-Tools:-Get(Txmaxe::algebraic));
  end if;
end proc:

```

Todos os procedimentos accionados a partir de 'MMLOp' recebem três parâmetros: a expressão da função (caixa de texto 'Tfe') e os limites do domínio de visualização do gráfico (caixas de texto 'Txmine' e 'Txmaxe').

O procedimento 'grMax' constrói graficamente segmentos verticais azuis, que projectam os máximos da função no eixo dos xx .

```

> grMax := proc(fe, negX, posX) option
  remember:
  local d1, d2, sols, i, p:
  d1 := diff(fe,x);
  d2 := diff(fe,x$2);
  sols := MMLMax(fe, negX, posX);

```

Neste procedimento, calcula-se a primeira e a segunda derivada da função e cria-se a lista

('sols') das coordenadas dos pontos máximos construída no procedimento 'MMLMax'.

```

> p := []:
  for i from 1 to nops(sols) do
    if evalf(subs(x=sols[i][1],d2)) <= 0 then
      p := [op(p), line([sols[i][1],0],[sols[i][1],subs(x=sols[i][1],
        fe)], color=blue, thickness=2)]:
    end if:
  end do:
  if nops(p) = 0 then
    return NULL:
  end if:
  display(seq(p[i],i=1..nops(p))):
end proc:

```

Visualmente, é fácil constatar que um ponto máximo $(x_0, f(x_0))$ verifica a condição $f''(x_0) \leq 0$. Em todos os pontos da lista de máximos onde isto acontece, é construído um segmento que une os pontos $(x_0, 0)$ e $(x_0, f(x_0))$, onde x_0 é a abcissa de um ponto máximo. Estes segmentos são construídos através do comando **line** e são guardados na lista 'p'. No final do procedimento, o comando **display** faz com que sejam desenhados todos os segmentos onde existem pontos máximos da função.

O procedimento 'grMin' que constrói segmentos verticais vermelhos que projectam os pontos mínimos no eixo dos xx , é construído de forma análoga ao procedimento 'grMax'.

O procedimento 'grCres' constrói as partes da função onde esta é crescente, ou seja, se a função for crescente, aparecerá por cima do gráfico original, nas partes onde a função é crescente, uma sobreposição de outra cor (por defeito, cor azul).

Neste procedimento, vai ser necessário ter a lista dos pontos críticos de f ($\text{sols} = [x_1, \dots, x_i, \dots, x_n]$), à qual deverão ser adicionados o valor mínimo (min) e o valor máximo

(*max*) do domínio de visualização do gráfico.

```
> grCres := proc(fe, negX, posX) option
remember:
local d1, sols, i, lin, j:
d1 := diff(fe,x);
sols := MMLZer(d1, negX, posX);
if not member(evalf(negX), sols) then
sols := [evalf(negX), op(sort(sols))]:
end if:
if not member(evalf(posX), sols) then
sols := [op(sort(sols)), evalf(posX)]:
end if:
```

Depois, consideram-se todos os elementos da lista 'sols' até ao penúltimo elemento (deixa-se o *max* de fora). Se verificarmos que $f'(\frac{x_j+x_{j+1}}{2}) > 0$, então a função é crescente no intervalo $[x_j, x_{j+1}]$, onde x_j pode ser qualquer ponto crítico da lista ou o valor mínimo (*min*) do domínio de visualização do gráfico. Depois de termos este intervalo, é só fazer o gráfico da função nesse intervalo com a cor azul (cor escolhida para representar as partes do gráfico onde esta é crescente), através do comando `plot(fe, x=sols[j]..sols[j+1], color=blue, thickness=2)`.

```
> lin := []:
for j from 1 to (nops(sols) - 1) do
if evalf(subs(x=(sols[j] + sols[j+1])/2,d1)) < 0 then
lin := [op(lin), plot(fe, x=sols[j]..sols[j+1], color=red,
thickness=2)]:
end if:
end do:
```

No final serão apresentadas todas as partes do gráfico em que a função é crescente, através do comando `display`.

```

> if nops(lin) = 0 then
  return NULL:
end if:
display(seq(lin[i], i=1..nops(lin))):
end proc:

```

O procedimento 'grDec', que constrói as partes do gráfico em que a função é decrescente (na cor vermelha), é análogo a este procedimento ('grCres').

O procedimento 'grConcC' faz com que as zonas entre as partes do gráfico em que a concavidade está voltada para cima e o eixo dos xx sejam coloridas (neste caso escolheu-se a cor roxo).

```

> grConcC := proc(fe, negX, posX) option
  remember:
  local d2, sols, i, n, polin, j, delta, X1, X2, Y1, Y2, p:
  d2 := diff(fe, x$2);
  sols := MMLZer(d2, negX, posX);
  if not member(evalf(negX), sols) then
    sols := [evalf(negX), op(sort(sols))]:
  end if:
  if not member(evalf(posX), sols) then
    sols := [op(sort(sols)), evalf(posX)]:
  end if:

```

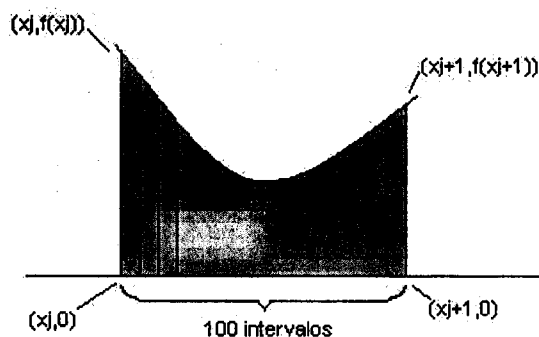
Neste procedimento, vai ser necessário ter a lista dos pontos de inflexão ('sols'), à qual se devem adicionar o valor mínimo e o máximo do domínio de visualização do gráfico (min e max).

Se for verificada a condição $f''\left(\frac{x_j+x_{j+1}}{2}\right) > 0$ em que x_j pode ser qualquer abscissa dos pontos de inflexão da lista ou o valor mínimo do domínio de visualização (min), fica

garantido que no intervalo $[x_j, x_{j+1}]$ a concavidade do gráfico fica voltada para cima.

```
> n := 100: polin := []:
  for j from 1 to (nops(sols) - 1) do
    if evalf(subs(x = (sols[j] + sols[j+1])/2, d2)) > 0 then
      delta := (sols[j+1] - sols[j]) / n:
      X2 := sols[j]:
      for i from 1 to n do
        X1 := evalf(X2): Y1 := evalf(subs(x = X1, fe)):
        X2 := evalf(sols[j] + i * delta): Y2 := evalf(subs(x=X2,fe)):
        p[i] := polygonplot([[X1, 0],[X1, Y1],[X2, Y2],[X2, 0]],
          color=plum, style=patchnogrid):
      end do:
      polin := [op(polin), seq(p[i],i=1..n)]:
    end if:
  end do:
```

Depois de se ter o intervalo com a concavidade voltada para cima definido, divide-se este intervalo em 100 sub-intervalos, pintando-se em cada um desses intervalos pequenos o quadrilátero formado pelos pontos $(X1, 0)$, $(X1, f(X1))$, $(X2, f(X2))$, $(X2, 0)$, em que $X1$ e $X2$ são, respectivamente, $(x_j + (i - 1) \times \delta)$ e $(x_j + i \times \delta)$. Aqui, i representa o i -ésimo sub-intervalo e $\delta = \frac{x_{j+1} - x_j}{100}$.



No primeiro intervalo, $X1 = x_j$ e no último intervalo $X2 = x_{j+1}$. Os quadriláteros todos juntos formam a região entre a parte do gráfico com a concavidade voltada para cima e o

eixo dos xx , e são representados através do comando **display**.

```
> if nops(polin) = 0 then
  return NULL:
end if:
display(seq(polin[i], i=1..nops(polin))):
end proc:
```

O procedimento 'grConcB', faz o mesmo que o procedimento anterior, mas para partes do gráfico onde a concavidade está voltada para baixo. A sua construção é análoga ao procedimento 'grConcC'.

O procedimento 'GrFunc' é aquele que desenha o gráfico original da função (determinou-se que seria de cor preta).

```
> GrFunc := proc(fe, negX, posX) option
  remember:
  plot(fe, x=negX..posX, color=black, thickness=1):
end proc:
```

Os resultados dos procedimentos 'grMax', 'grMin', 'grCres', 'grDec', 'grConcC' e 'grConcB', são sobrepostos ao gráfico que resulta deste procedimento, que é o gráfico definido na janela principal. Assim, foram criados novos procedimentos, que através do comando **display**, mostram os gráficos resultantes de cada procedimento e o gráfico da função construído a partir do procedimento 'GrFunc'. Os procedimentos 'GrMax1', 'GrMin1', 'GrCres1', 'GrDec1', 'GrConcC1' e 'GrConcB1', garantem que o gráfico da função é sempre desenhado.

```
> GrMax1 := proc(fe, negX, posX) option
  remember:
  display(grMax(fe, negX, posX), GrFunc(fe, negX, posX)):
end proc:
```

```

> GrMin1 := proc(fe, negX, posX) option
remember:
display(grMin(fe, negX, posX), GrFunc(fe, negX, posX)):
end proc:
> GrCres1 := proc(fe, negX, posX) option
remember:
display(grCres(fe, negX, posX), GrFunc(fe, negX, posX)):
end proc:
> GrDec1 := proc(fe, negX, posX) option
remember:
display(grDec(fe, negX, posX), GrFunc(fe, negX, posX)):
end proc:
> GrConcC1 := proc(fe, negX, posX) option
remember:
display(grConcC(fe, negX, posX), GrFunc(fe, negX, posX)):
end proc:
> GrConcB1 := proc(fe, negX, posX) option
remember:
display(grConcB(fe, negX, posX), GrFunc(fe, negX, posX)):
end proc:

```

Quando o utilizador escolhe uma opção da **DropDownBox** na janela 'Estudo', e clica no botão 'Mostrar', um dos procedimentos que é accionado é o 'GraficoEst', que consoante a escolha, faz accionar um dos procedimentos anteriores ('GrMax1', ..., 'GrConcB1').

```

> GraficoEst:= proc()
if Maplets:-Tools:-Get(Dde) = "Zeros" then
GrFunc(Maplets:-Tools:-Get(Tfe::algebraic),
Maplets:-Tools:-Get(Txmine::algebraic),
Maplets:-Tools:-Get(Txmaxe::algebraic));
elif Maplets:-Tools:-Get(Dde) = "Pontos Maximos" then
GrMax1(Maplets:-Tools:-Get(Tfe::algebraic),
Maplets:-Tools:-Get(Txmine::algebraic),
Maplets:-Tools:-Get(Txmaxe::algebraic));

```

```

> (...)
  elif Maplets:-Tools:-Get(Dde) = "Concavidades Voltadas para
  Baixo" then
    GrConcB1(Maplets:-Tools:-Get(Tfe::algebraic),
    Maplets:-Tools:-Get(Txmine::algebraic),
    Maplets:-Tools:-Get(Txmaxe::algebraic));
  end if;
end proc:

```

A janela 'Estudo' contém um botão que mostra graficamente todas as opções da **DropDownBox**. É o botão 'Mostrar tudo graficamente'. Quando o utilizador clica neste botão, é accionado a acção 'Aemtg'.

```

> Action[Aemtg] (
  Evaluate(Pe='grTodos(Tfe,Txmine,Txmaxe)'),
  SetOption(MMLVe="")
),

```

Esta acção faz com que seja executado o procedimento 'grTodos', que mostra através do comando **display**, todos os gráficos obtidos nos procedimentos 'grMax', 'grMin', 'grCres', 'grDec', 'grConcC' e 'grConcB'.

```

> grTodos := proc(fe, negX, posX) option
remember:
display(grMin(fe, negX, posX), grMax(fe, negX, posX), grConcB(fe,
negX, posX), grConcC(fe, negX, posX), grDec(fe, negX, posX),
grCres(fe, negX, posX)):
end proc:

```

Quando o utilizador clica no botão 'Apagar', na janela 'Estudo', é accionada a acção 'Apa'.

```

> Action[Apa](
    SetOption(MMLVe=""),
    Evaluate(Pe='GrFunc(Tfe,Txmine,Txmaxe)')
)

```

Esta acção, executa o procedimento 'GrFunc', ou seja, o gráfico original da função é mostrado sem os outros gráficos.

Finalmente, quando o utilizador clica no botão 'Fechar', é accionada a acção 'Aje', que fecha a janela 'Estudo' e abre a janela principal.

```

> Action[Aje](
    CloseWindow(Estudo),
    RunWindow(Principal)
),

```

Para que as janelas 'Estudo' e 'Lerestudo' sejam executadas na Maplet, temos de as adicionar na 'MapletG'.

```

> Janelaestudo,
    Janelaler,

```

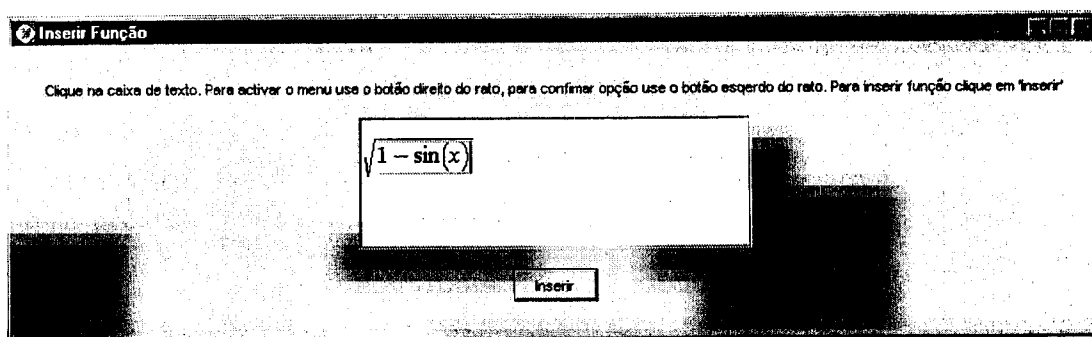
5.5 A Maplet actual

A Maplet sobre o estudo gráfico de funções pode ser considerada incompleta, uma vez que é possível adicionar mais elementos à Maplet, de forma a obter mais funcionalidades. A Maplet construída na secção anterior é constituída pela janela principal, pela janela de opções do gráfico, pelas janelas de *zoom* e de *trace* e, fundamentalmente, pela janela de estudo.

Nesta secção vamos adicionar mais uma funcionalidade, um botão que abre uma janela onde a função pode ser inserida através de um **MathMLEditor**, ou seja, onde o utilizador pode inserir a função na notação tradicional da matemática, tendo à disposição um menu de símbolos matemáticos de fácil acesso.

```
> Button[Bfunc]("Inserir  
função",onclick=Action(RunWindow(Mathfunc))),
```

Quando o utilizador clica no botão 'Inserir função', abre-se a janela 'Mathfunc'.



Para editar a expressão, o utilizador necessita de clicar primeiro com o botão esquerdo do rato, na zona de edição. Depois pode inserir a expressão e aceder à barra de símbolos matemáticos através do botão direito do rato. Depois de editada a expressão da função, o utilizador deverá clicar no botão 'Inserir' para que a função seja inserida na caixa de texto própria na janela principal. A caixa de texto destinada à expressão da função na janela principal, edita funções na notação do Maple. Portanto, para que a passagem da notação matemática para a notação Maple ocorra com sucesso, é necessário utilizar o comando **MathML[Import]**, que transforma expressões na notação matemática em expressões na notação Maple.

A janela foi construída da seguinte maneira:

```

> JanelaMath:=Window[Mathfunc](title="Inserir
  Função",
  [
    [GridLayout(
      GridRow(GridCell("Clique na caixa de texto. Para activar o menu
        use o botão direito do rato, para confirmar opção use o botão
        esquerdo do rato. Para inserir função clique em 'Inserir'")),
      GridRow(GridCell(value=Mfunc)),
      GridRow(GridCell(value=Bins)))
    ]]),
  MathMLEditor[Mfunc](),
  Button[Bins]("Inserir",onclick=Ains):

```

O título da janela é 'Inserir Função'. Esta janela contém um espaço para edição da função (`MathMLEditor`) e um botão 'Inserir'. Quando o utilizador clica neste botão, é accionada a acção 'Ains' que executa o procedimento 'mathimport' e fecha a janela 'Mathfunc'.

```

> Action[Ains](
  Evaluate(Tf='mathimport()'),
  CloseWindow(Mathfunc)
),

```

O procedimento 'mathimport' transforma a expressão em notação matemática numa expressão em notação Maple.

```

> mathimport:=proc()
  MathML[Import](Maplets:-Tools:-Get(Mfunc,corrections)):
end proc:

```

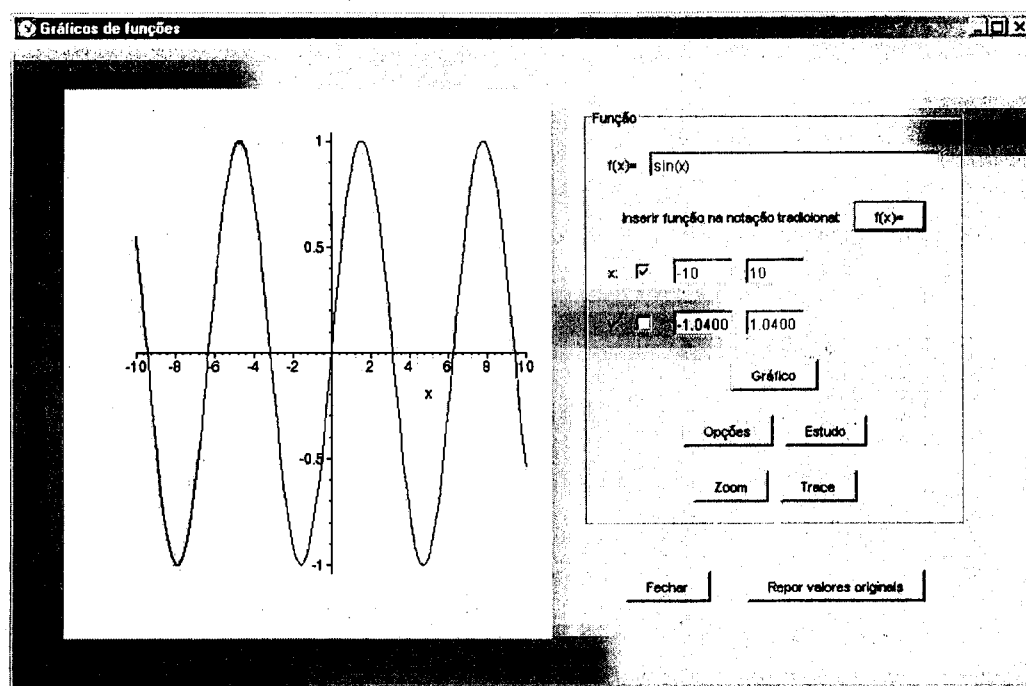
Assim, o utilizador pode escolher duas maneiras diferentes de inserir a expressão de uma função. Ou pela introdução directa na janela principal em notação Maple, ou clicando

no botão 'Inserir função', escrevendo a expressão da função na janela 'Mathfunc' em notação matemática.

Para que a janela 'Mathfunc' seja executada, é necessário que no 'MapletG' se adicione esta janela.

> JanelaMath,

O aspecto final da janela principal desta Maplet é o seguinte:



O código completo desta Maplet pode ser visto no Apêndice A (graficofuncao5).

Capítulo 6

Exemplos de Maplets e suas aplicações

Por último, serão dados alguns exemplos de Maplets que podem permitir dar a um curso elementar de Cálculo Infinitesimal uma abordagem mais interactiva, por ventura do maior agrado dos alunos. Também exibiremos Maplets que permitirão uma abordagem "dinâmica" no estudo de funções. Nessas Maplets, o aluno poderá experimentar propriedades assintóticas de diversas funções, isto é, poderão testar características que ficam invariantes após muitas iterações da aplicação f .

6.1 Exemplos úteis no Ensino

Nesta secção serão apresentadas algumas Maplets construídas para poderem ser usadas num curso introdutório de Cálculo Infinitesimal. Os três exemplos que vão ser descritos têm em comum o facto de podermos resolver problemas "passo-a-passo", isto é, estas Maplets permitem que um problema seja resolvido calculando e visualizando todos os passos da resolução até à solução final. Se o utilizador não souber o que fazer para o passo seguinte, pode sempre utilizar as sugestões que a Maplet fornece para resolver o problema. O aluno deverá tentar resolver o problema e compará-lo à resolução sugerida pela Maplet. O professor tem à sua disposição um boa ferramenta para corrigir problemas

e também para criar exercícios para os alunos resolverem.

Os três exemplos desta secção permitem resolver problemas de cálculo de limites, de derivação e de integração. Estas Maplets são uma versão adaptada de outras disponíveis no local da *Web* referenciado em [22].

Por ser das três, aquela que é mais elaborada, será feita em primeiro lugar a descrição da Maplet da integração (Secção 6.1.1). A descrição das outras duas (em 6.1.2 e 6.1.3) é menos detalhada, pois seguem o mesmo padrão da primeira.

6.1.1 Integração

Esta Maplet permite resolver problemas de integração visualizando-se todos os passos até chegar à solução final.

Com uma Maplet simples de resolução de integrais (secção 3.3), o que acontecia é que o utilizador inseria o integral que queria resolver e depois só apareceria a solução final¹. Ficava sempre a pergunta: "Mas então como é que se chegou a este resultado?". Com esta Maplet é possível ver todos os passos de resolução e assim compreender melhor o que se está a fazer. Para além disso se surgir alguma dúvida, pode-se sempre obter sugestões para resolver o problema. A janela principal desta Maplet tem o aspecto da figura 6.1, aí é possível resolver primitivação e integrais definidos.

Todas as regras de integração incluídas nesta Maplet, pertencem ao *sub-package Calculus1* do Maple, portanto, as limitações no tratamento dos problemas são as limitações da referida *package*.

O utilizador deverá começar por introduzir a função que quer integrar e a variável de integração. Se quiser resolver um integral definido, o utilizador deverá também inserir os

¹Quando o Maple tem algoritmos para solucionar o problema

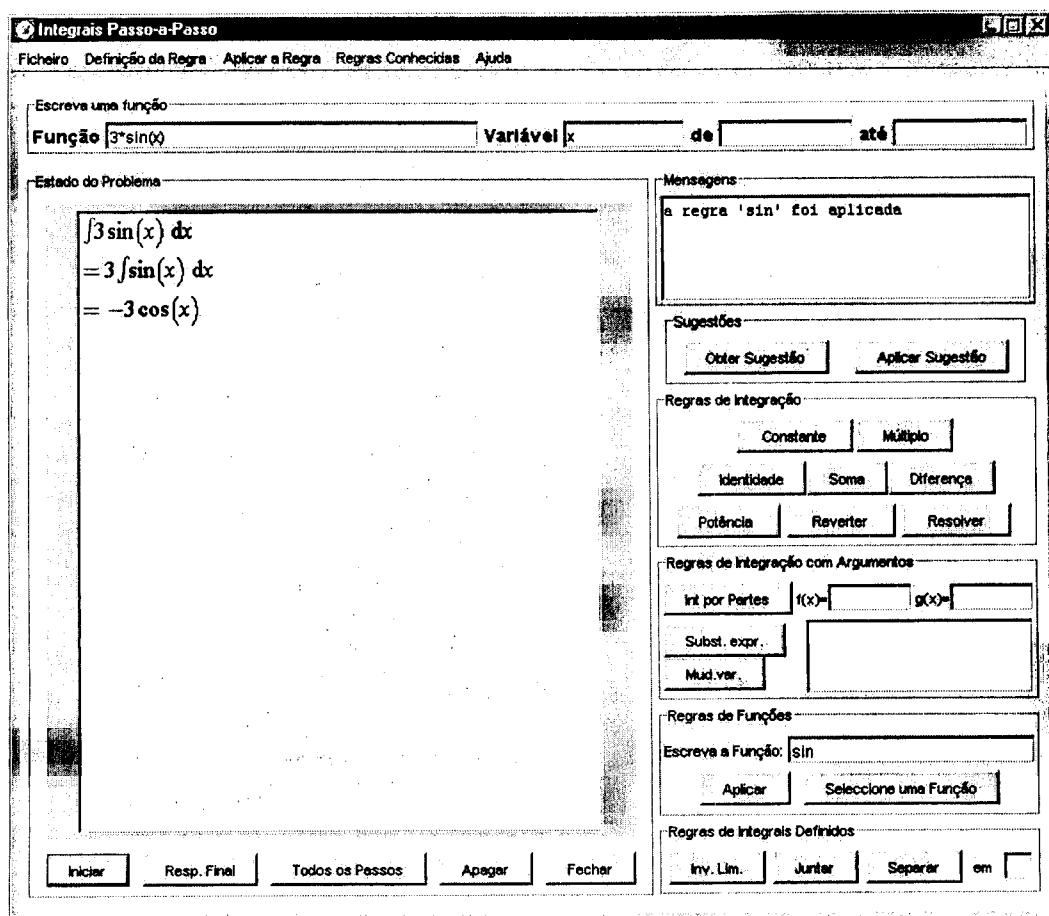


Figura 6.1: Integração passo a passo

limites de integração. Se pretender calcular um integral indefinido, deixa essas duas caixas em branco. Por defeito, a função inserida é $f(x) = 3\sin(x)$, a variável de integração é x e as caixas para os limites de integração estão em branco. Estão, portanto, criadas as condições para se resolver um integral indefinido.

Na região 'Estado do Problema' está o espaço onde vão aparecer todos os passos da resolução do problema e cinco botões: 'Iniciar' (para iniciar o problema), 'Resp. Final' (para apresentar directamente a resposta final), 'Todos os Passos' (para serem apresentados todos os passos de resolução do problema até à resposta final), 'Apagar' (para

apagar tudo o que está para trás) e 'Fechar' (para fechar a janela principal).

Em 'Mensagens' surge o espaço onde aparecem as mensagens sobre a resolução do problema. É nesta caixa que aparecem as mensagens sobre a regra que está a ser aplicada, sobre a informação de que tal regra não pode ser aplicada nesse passo, sobre as sugestões que podem obtidas para aplicar uma determinada regra no passo que se segue, etc. Nesta caixa, o nome das regras aparecem na sua forma original, ou seja, da maneira com que o Maple identifica cada regra. Por isso, não é de estranhar que apareçam nesta caixa frases como: "a regra 'constantmultiple' foi aplicada " ou "A regra 'power' não é aplicável". No menu 'Definição da Regra', é possível encontrar a correspondência entre a regra e o seu nome original.

Ao clicar no botão 'Obter Sugestão', aparecerá na caixa das mensagens uma sugestão de uma regra que pode ser utilizada naquele momento. No caso do Maple não encontrar nenhuma sugestão, aparecerá a mensagem "Nenhuma sugestão pode ser dada, ou o problema está resolvido". Para aplicar a regra sugerida, basta clicar em 'Aplicar Sugestão'. Para aplicar directamente as regras, clica-se no botão correspondente à regra que se quer aplicar ('Constante', 'Múltiplo', 'Identidade', 'Soma', 'Diferença' e 'Potência'). Se for necessário integrar por partes, o utilizador deve usar o botão 'Int por Partes', mas antes de o pressionar, deve especificar as duas funções $f(x)$ e $g(x)$, para as quais se pretende calcular $\int f(x)g'(x)dx$.

Em vários problemas, depois de algumas aplicações da integração por partes, o mesmo integral aparece mais do que uma vez em ambos os membros da equação. Nestes casos, o utilizador pode usar o botão 'Resolver' para que o Maple resolva algebricamente a equação em relação ao integral pretendido. Se for necessário escrever uma expressão mais simplificada ou de forma mais ajustada ao que se pretende para o passo seguinte, é possível trocar expressões do problema por outras expressões equivalentes. Assim, na caixa de texto ao lado do botão 'Subst. expr.' inserem-se uma ou mais equações

de substituição separadas por vírgulas e depois de clicar no botão 'Subst. expr.', as expressões são actualizadas, isto é, cada expressão do 1.º membro da equação é trocada pela expressão do 2.º membro dessa equação. Se for necessário fazer uma mudança de variável para conseguir resolver o integral, o utilizador pode usar a regra da mudança de variável. Para tal, o utilizador deverá inserir na caixa de texto indicada para a substituição de expressões, os parâmetros da mudança de variável separados por vírgulas. Os parâmetros têm como base uma equação da forma $F(x, u) = G(x, u)$ que define a relação entre a antiga variável x e a nova variável u . Isto usualmente toma a forma de $x = G(u)$ ou $u = F(x)$, mas pode ser mais geral.

O botão 'Mud. var.' concretiza as substituições e depois da integração com a nova variável estar completa, clica-se no botão 'Reverter' para substituir outra vez para a variável antiga.

Para aplicar regras de integração de funções, o utilizador deverá inserir uma função na caixa de texto 'Escreva a Função' e clicar no botão 'Aplicar'. O botão 'Selecione uma Função' abre uma janela (fig. 6.2) onde o utilizador pode escolher uma entre várias funções matemáticas reconhecidas pelo Maple.

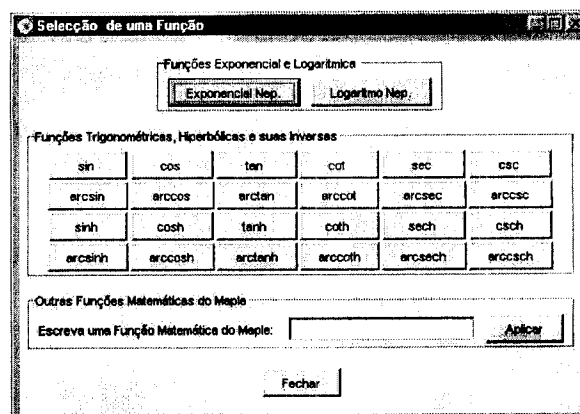


Figura 6.2: Selecção de uma Função

Na resolução de integrais definidos, o utilizador tem acesso a mais 3 regras. Se escolher o botão 'Inv. Lim.', aplica a regra $\int_a^b f(x)dx = -\int_b^a f(x)dx$. Se escolher o botão 'Juntar', aplica a regra $\int_a^b f(x)dx + \int_b^c f(x)dx = \int_a^c f(x)dx$. Se escolher o botão 'Separar' tem de especificar em que ponto deseja que o integral seja separado e este é transformado na soma de dois integrais em que o limite superior do primeiro é igual ao limite inferior do segundo.

Esta janela contém também uma barra de menus constituída pelos seguintes menus: 'Ficheiro', 'Definição da Regra', 'Aplicar a Regra', 'Regras Conhecidas' e 'Ajuda'. O menu 'Ficheiro', contém as opções 'Iniciar Problema', 'Obter Resposta Final', 'Mostrar todos os Passos', 'Obter uma Sugestão', 'Aplicar a Sugestão', 'Anular', 'Apagar' e 'Fechar'. Estas opções executam exactamente as mesmas acções que os respectivos botões na janela principal.

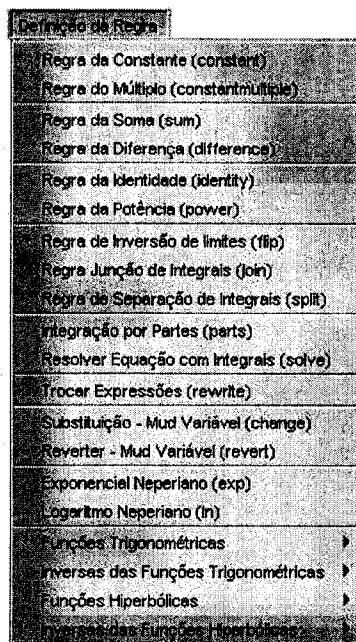


Figura 6.3: Menu Definição da Regra

O menu 'Definição da Regra' (fig. 6.3) contém uma listagem das regras que podem ser aplicadas nesta Maplet.

À frente de cada regra está entre parêntesis o nome original da regra na *package* 'Calculus1' do Maple, para que o utilizador possa identificar a regra com o seu nome original. Ao escolher uma dessas regras é accionada uma janela com a definição da regra escolhida. Por exemplo, para ver a definição da regra da identidade, deve escolher-se 'Regra da Identidade (identity)', e aparecerá a seguinte janela (fig. 6.4):

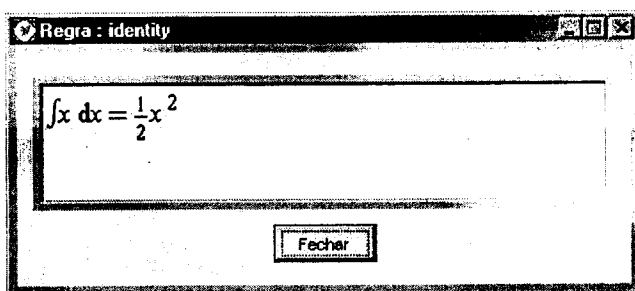


Figura 6.4: Regra Identidade

O menu 'Aplicar a Regra' contém a lista das regras que se podem aplicar nesta Maplet. As opções deste menu executam exactamente as mesmas acções que os respectivos botões na janela principal.

O menu 'Regras Conhecidas' contém também uma listagem das regras que podem ser aplicadas nesta Maplet. Neste menu, o utilizador deve seleccionar a regra ou as regras que pretende considerar como "regras conhecidas". Ao seleccionar uma regra, aparece um símbolo a indicar que a regra está seleccionada. Se uma regra é considerada como "regra conhecida", ela é automaticamente aplicada pelo Maple.

Portanto, se o utilizador considerar que determinada regra ou regras são básicas, selecciona essa(s) regra(s) neste menu, e quando o problema estiver a ser resolvido, se for possível

a(s) regra(s) ser(em) aplicada(s), não precisa de ser o utilizador a aplicá-la(s), porque o Maple fá-lo automaticamente.

O menu 'Ajuda' (fig. 6.5), contém opções para o utilizador poder escolher entre a possibilidade de mostrar ou esconder o nome das regras em todos os passos de resolução ou em cada passo e uma opção 'Usando esta Maplet' que abre uma janela com informações sobre o funcionamento da Maplet.

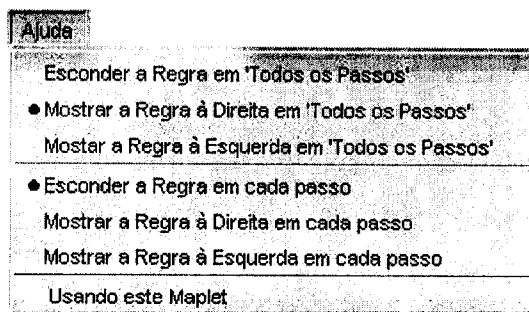


Figura 6.5: Menu Ajuda

6.1.2 Derivação

Esta Maplet (fig. 6.6) tem uma estrutura similar à da secção anterior, mas em vez de resolver problemas de integração, resolve problemas de derivação "passo-a-passo".

O utilizador pode aplicar as regras de derivação e ver a expressão resultante de cada passo. A caixa de Mensagens mostra a informação sobre a regra utilizada e a qualquer altura, o utilizador pode pedir uma sugestão sobre a próxima regra a ser utilizada, e aplicá-la. Também pode ser seleccionado um conjunto de regras consideradas básicas (regras conhecidas), que serão automaticamente aplicadas (caso seja possível).

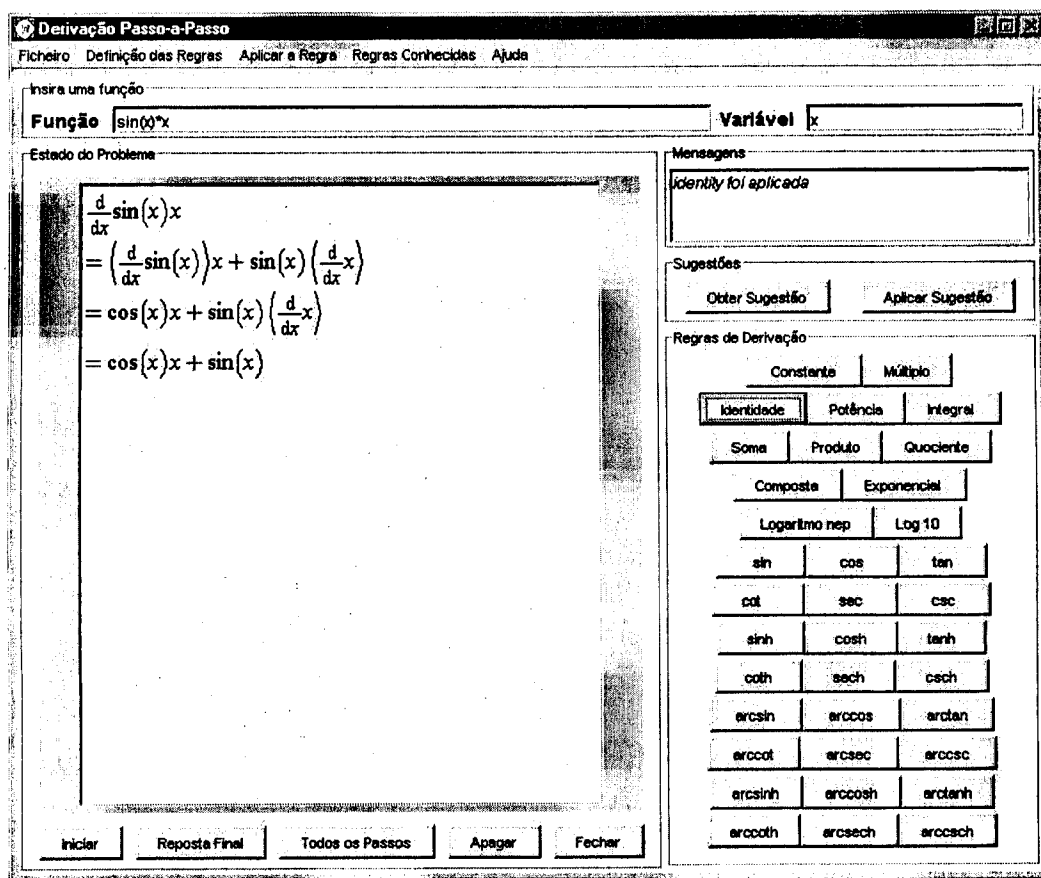


Figura 6.6: Derivação passo a passo

Primeiro escreve-se uma função com a sua variável independente. Depois clica-se no botão 'Iniciar' para começar a resolver o problema. Para aplicar as várias regras de derivação, clica-se nos botões correspondentes ou selecciona-se as regras do menu 'Aplicar a Regra'. Nesta Maplet estão incluídas regras como por exemplo, a regra da soma ($D(f + g) = D(f) + D(g)$), a regra do produto ($D(f \cdot g) = D(f) \cdot g + f \cdot D(g)$), a regra do quociente ($D(f/g) = (D(f) \cdot g - f \cdot D(g))/(g^2)$), a regra de funções compostas ($D(f(g(x))) = D(f)(g(x)) \cdot D(g)(x)$) e regras para funções especiais como $D(\sin(x)) = \cos(x)$.

Depois de iniciar o problema, o utilizador pode obter directamente a solução final, clicando no botão 'Resposta Final' ou seleccionando 'Resposta Final' do menu 'Ficheiro'. Se

o utilizador quiser ver o problema resolvido com todos os passos de resolução exibidos, só tem de clicar no botão 'Todos os Passos' ou seleccionar 'Todos os Passos' do menu 'Ficheiro'.

Para saber qual é a definição de uma determinada regra, pode consultar-se o menu 'Definição das Regras', onde consta uma lista das regras de derivação. À frente de cada regra está entre parêntesis o seu nome original. Por exemplo, se o utilizador quiser saber a definição da regra do produto, selecciona 'Produto (product)' neste menu e aparecerá a seguinte janela (fig. 6.7):

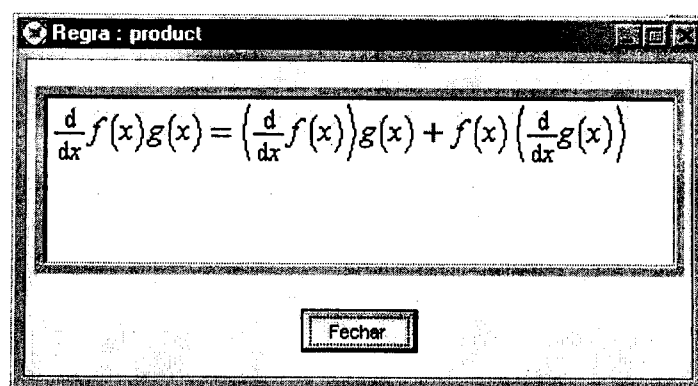


Figura 6.7: Regra do Produto

Nesta Maplet também é possível substituir uma expressão por outra que lhe seja equivalente, usando a regra 'Substituir expressões' que se encontra no menu 'Aplicar a Regra'. Também neste menu existe a opção 'Inserir uma regra de derivação' para o utilizador especificar alguma regra de derivação que não encontre nesta Maplet.

O botão 'Fechar' da janela principal ou a opção 'Fechar' do menu 'Ficheiro' fecham a janela de resolução de 'Derivadas Passo-a-Passo'.

6.1.3 Cálculo de limites

A janela de resolução de problemas de limites "passo a passo" (fig. 6.8), também tem a estrutura das janelas anteriores. O objectivo desta Maplet é calcular o $\lim_{x \rightarrow a} f(x)$ em que a é real ou $a = \pm\infty$. Também é possível calcular limites laterais para estudar a continuidade ou a derivabilidade de uma função.

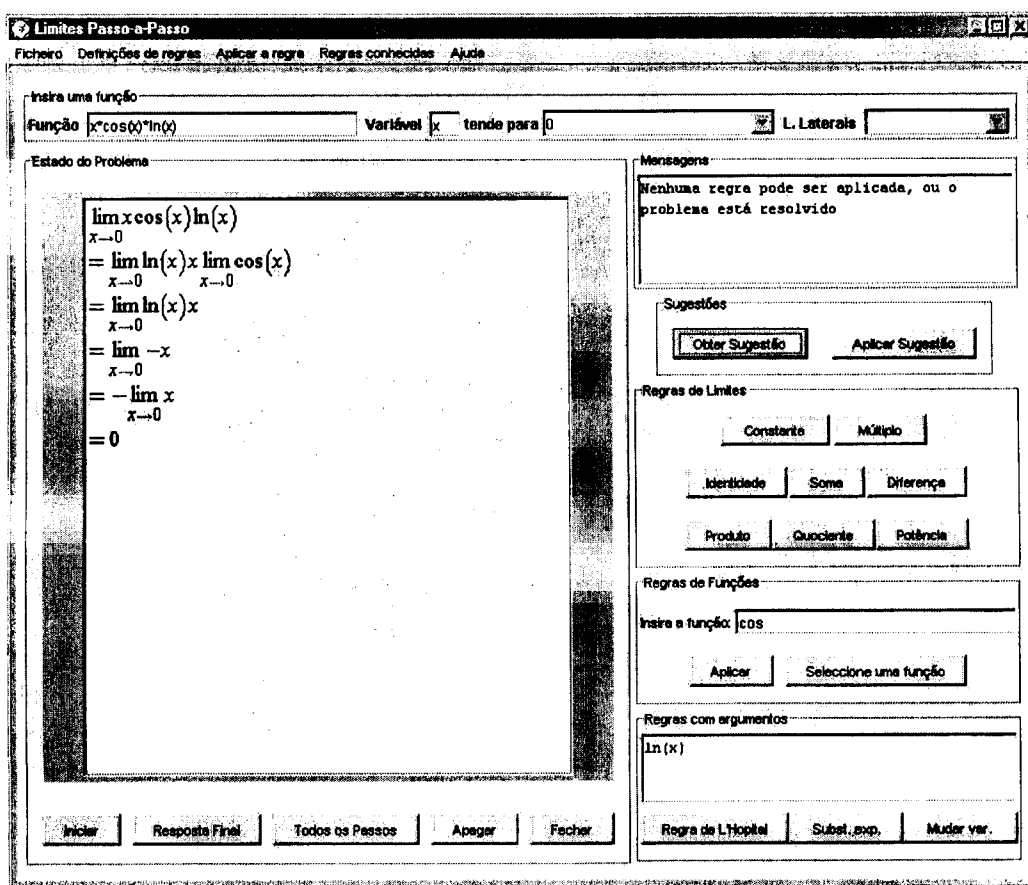


Figura 6.8: Cálculo de Limites passo a passo

Sempre que na resolução de um limite se obtenha uma indeterminação do tipo $0/0$ ², podemos começar por aplicar a regra de L'Hopital (ver em [3] o enunciado preciso).

²Isto é, se obtenha o quociente de duas funções $\frac{f(x)}{g(x)}$ em que $\lim_{x \rightarrow a} f(x) = \lim_{x \rightarrow a} g(x) = 0$

Também é possível modificar expressões através da regra 'Substituir expressões'. Assim, por exemplo, para resolver uma indeterminação do tipo ∞/∞ com um quociente de polinómios, podemos substituir os termos de cada polinómio por quocientes entre os termos e o termo de maior grau.

$$\lim_{x \rightarrow \infty} \frac{x^2 - 2x + 1}{x^2 + 3} = \lim_{x \rightarrow \infty} \frac{1 - \frac{2}{x} + \frac{1}{x^2}}{1 + \frac{3}{x^2}}$$

Nesta Maplet, o utilizador deve introduzir uma função, o nome da variável independente, o valor para o qual se pretende o limite e, se for o caso de se querer calcular um limite lateral, seleccionar se se quer calcular o limite à esquerda (*left*) ou à direita (*right*) do valor para o qual tende a variável.

Depois de inseridos estes valores, deve clicar-se no botão 'Iniciar' ou seleccionar 'Iniciar' do menu 'Ficheiro' para começar a resolver o problema. Tal como nas duas Maplets anteriores, é possível obter sugestões e aplicá-las em cada passo. Para aplicar regras de funções, existe, como na Maplet da integração, um botão que abre uma janela com as regras das funções exponencial, logarítmica, trigonométricas, hiperbólicas, etc. (fig. 6.9).

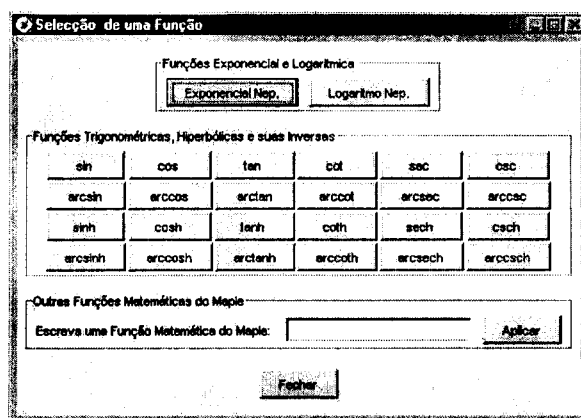


Figura 6.9: Seleção de uma Função

Nesta Maplet, podemos ainda aplicar três regras com argumentos: a regra de L'Hopital (fig. 6.10), a regra para substituir expressões e a regra da mudança de variável. Na regra de L'Hopital, devemos inserir o numerador da expressão. Na regra da substituição de expressões, devemos inserir uma equação que iguale a expressão que queremos modificar com a nova expressão e na regra da mudança de variável, devemos inserir uma equação que relacione as duas variáveis. Para ver as definições de cada regra, deve-se consultar o menu 'Definições de regras'. Como nas Maplets anteriores, também se pode usar a funcionalidade do menu 'Regras conhecidas'.

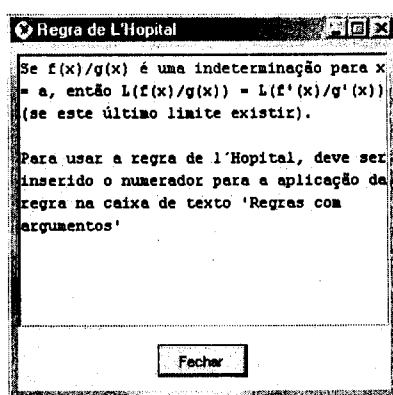


Figura 6.10: Regra de L'Hopital

No menu 'Ajuda', encontramos a opção 'Usando esta Maplet' que abre uma janela com informações sobre o funcionamento desta Maplet.

6.2 Gráficos de Funções com mais de uma variável

Com o Maple 8 é possível construir Maplets que contenham gráficos tridimensionais, mas não é possível haver aquela interacção para visioná-los de ângulos diferentes. Com o novo Maple 9 esta interactividade é possível e é por isso que se inclui esta secção no trabalho.

A Maplet da figura 6.11 é uma janela em que se pode ver e interagir (se se estiver a trabalhar com o Maple 9) com gráficos do tipo $z = f(x, y)$. É uma Maplet básica que

pode ser alterada, adicionando-se mais elementos que permitam melhorá-la de forma a que ela se torne numa Maplet mais completa e eficaz.

Como já se referiu, esta Maplet funciona melhor no Maple 9, onde o utilizador com o auxílio do rato pode interagir. Esta interactividade dos gráficos não existia no Maple 8.

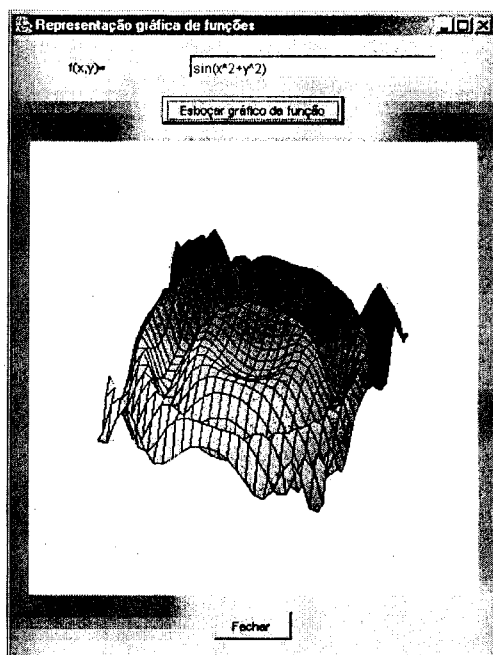


Figura 6.11: Gráfico tridimensional 1

O utilizador deverá inserir uma expressão com duas variáveis e clicar no botão 'Esboçar gráfico da função' para visualizar o gráfico. Depois de se ter o gráfico na Maplet, ele pode ser girado com o auxílio do rato: basta dar um simples clique com o botão esquerdo e, mantendo o botão esquerdo preso, girar o gráfico para uma nova posição. Para que isto seja possível, é necessário que a Maplet seja executada no Maple 9.

Tal como este tipo de gráficos tridimensionais, é possível construir outros e interagir directamente com eles na Maplet.

Por exemplo, se quisermos estudar uma superfície definida por equações paramétricas, podemos utilizar a Maplet da figura 6.12:

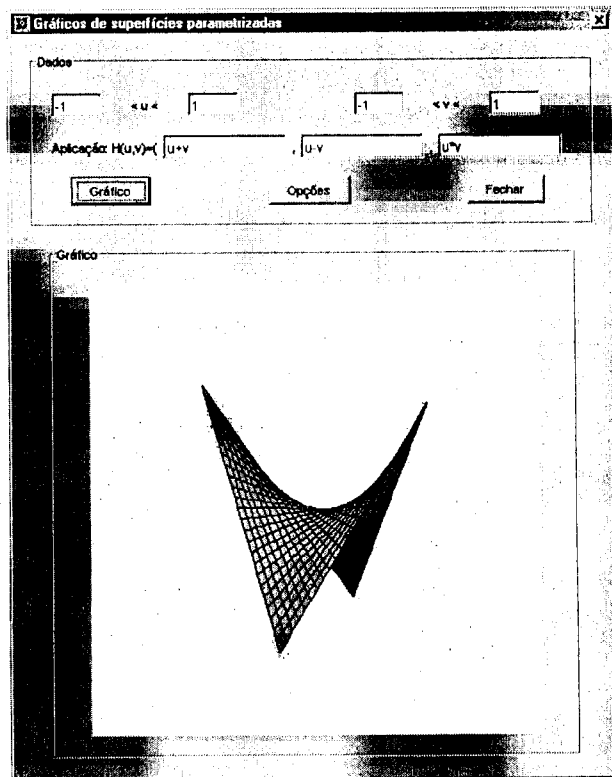


Figura 6.12: Gráfico tridimensional

Nas superfícies parametrizadas, os pontos (x, y, z) que pertencem ao gráfico da superfície, satisfazem equações do tipo:

$$\begin{cases} x = f(u, v) \\ y = g(u, v) \\ z = h(u, v) \end{cases}$$

com (u, v) pertencente a um domínio do plano predefinido.

Estes gráficos são construídos no Maple, através do comando `'plot3d'`. Por exemplo, para construir o gráfico do Maplet da figura 6.12 no Maple, deveria fazer-se, `'plot3d([u+v,u-v,u*v], u=-1..1,v=-1..1)'`.

Nesta Maplet, o utilizador só tem de inserir o domínio das variáveis u e v e escrever as expressões de $f(u,v)$, $g(u,v)$ e $h(u,v)$. Depois, só precisa de clicar em 'Gráfico', para visualizá-lo. Se pretender modificar algumas opções do gráfico, o utilizador pode clicar no botão 'Opções', que abre uma janela (fig. 6.13) onde se podem alterar algumas opções de visualização do gráfico.

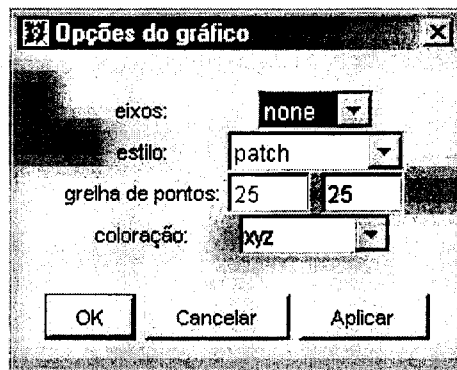


Figura 6.13: Opções do gráfico

O utilizador, pode inserir eixos cartesianos no gráfico, pode escolher o tipo de superfície que deseja e a definição da grelha de pontos que constituem a superfície e pode escolher o tipo de coloração que deseja para a superfície.

Nestas Maplets, é sempre possível modificar a posição da superfície, apenas com o auxílio do rato do computador.

Por último, apresentamos uma Maplet que pode ser usada para a visualização de funções do plano no plano. A ideia consiste em, dada uma região do plano (aqui consideramos um retângulo), através do preenchimento desse retângulo com faixas de diferentes cores,

fazer o esboço gráfico da imagem por F dessa região e observar como é que essas faixas coloridas são transformadas.

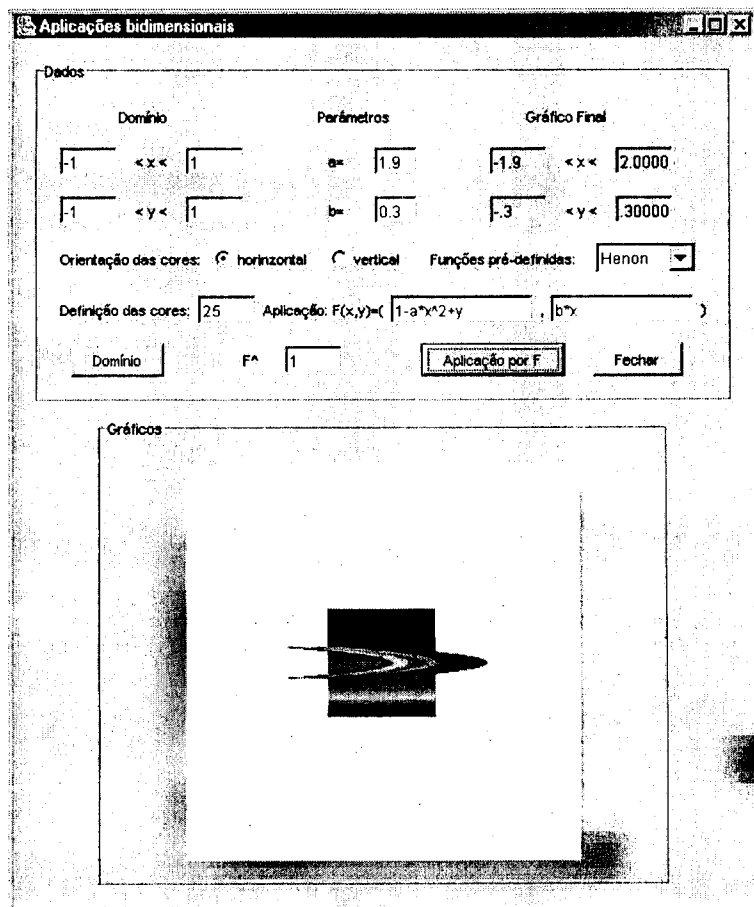


Figura 6.14: Aplicações bidimensionais

Nesta Maplet (fig. 6.14) tem-se como ponto de partida, isto é, como domínio de uma aplicação, um rectângulo R preenchido com várias cores dispostas em linhas horizontais ou em colunas verticais. O objectivo desta Maplet é visualizar como é que uma função F de duas variáveis transforma a região colorida R , apresentando no mesmo sistema de eixos R e $F(R)$.

Nesta Maplet, o utilizador deve inserir os dados relativamente ao domínio, escolher entre uma orientação das cores horizontal ou vertical, especificar quantos pontos de amostragem pretende e inserir a função de $\mathbb{R}^2$ em $\mathbb{R}^2$ que vai definir a aplicação. Depois deve clicar no botão 'Domínio de cores' para ver o plano do domínio e a seguir clicar no botão 'Aplicação por F' para ver o plano resultante da aplicação.

O botão 'Fechar' fecha esta janela. Quando as duas superfícies são visualizadas ao mesmo tempo nesta Maplet, é tido em conta o maior campo de visualização de cada uma das superfícies. Se visualizarmos apenas a superfície do domínio, o campo de visualização é, por defeito, $-1 < x < 1$ e $-1 < y < 1$. (ver figura 6.15)

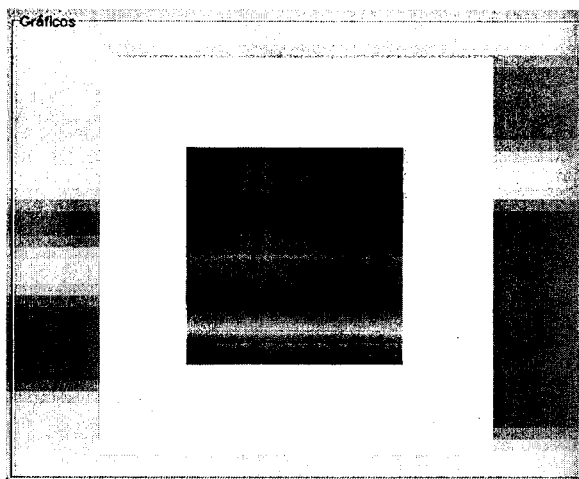


Figura 6.15: Superfície do domínio da Maplet 'Aplicações bidimensionais'

Existem variadíssimos exemplos de Maplets que podem ser utilizadas ao nível do ensino superior, secundário ou básico e até ao nível da investigação científica. Aproveitando todas as potencialidades do Maple, podemos aplicá-las a qualquer tema da Matemática, sem exigir que o utilizador tenha uma formação ao nível da linguagem Maple. Muitos exemplos desta metodologia podem ser encontrados na internet em [22].

6.3 Maplets para Sistemas Dinâmicos

Por último, neste capítulo apresentamos Maplets que se destinam à área de Sistemas Dinâmicos e serviram de suporte interactivo ao trabalho "O Maple em Sistemas Dinâmicos" [1] de Daniela Almeida.

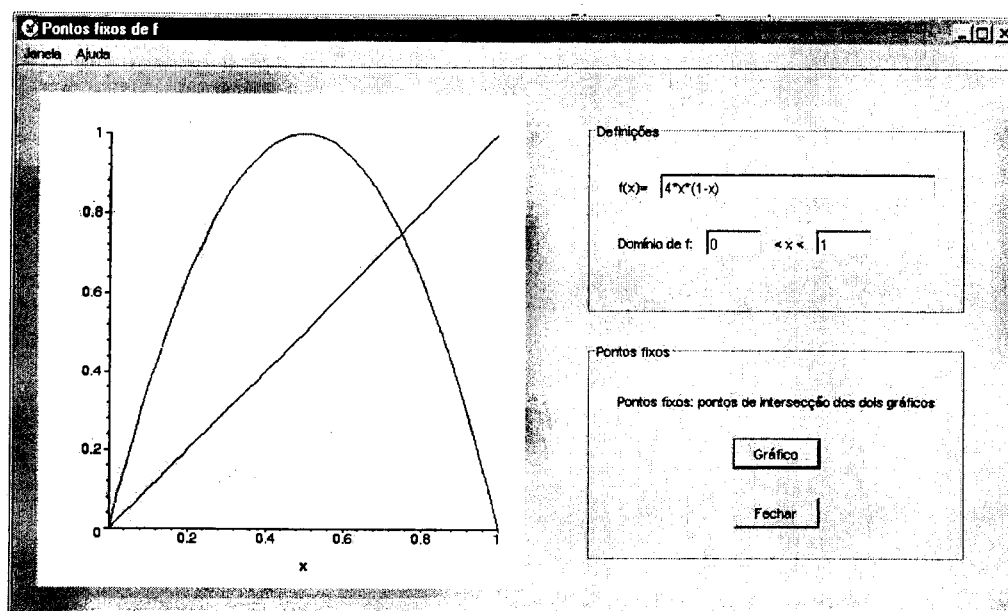
Estes exemplos mostram como a área de Sistemas Dinâmicos, pode ser usada no Ensino Secundário para motivar os conceitos aí estudados. Veremos como a vantagem da experimentação permitida pelas Maplets pode enriquecer o estudo de funções de uma variável e relacionar conteúdos como 'Estudo gráfico de funções' e 'Análise estatística', que sempre aparecem como Unidades distintas.

Começamos com uma Maplet (fig. 6.16) que pode ser usada para determinar os pontos fixos de uma função f real de variável real, isto é, determinar pontos x tais que $f(x) = x$.

Os pontos fixos de uma função podem ser determinados através da observação gráfica da função f e da função identidade. Uma função f tem um ponto fixo em a se e só se o ponto (a, a) pertence ao gráfico de f , ou seja, a função f tem um ponto fixo em a se e só se o seu gráfico intersecta a recta $y = x$ no ponto (a, a) .

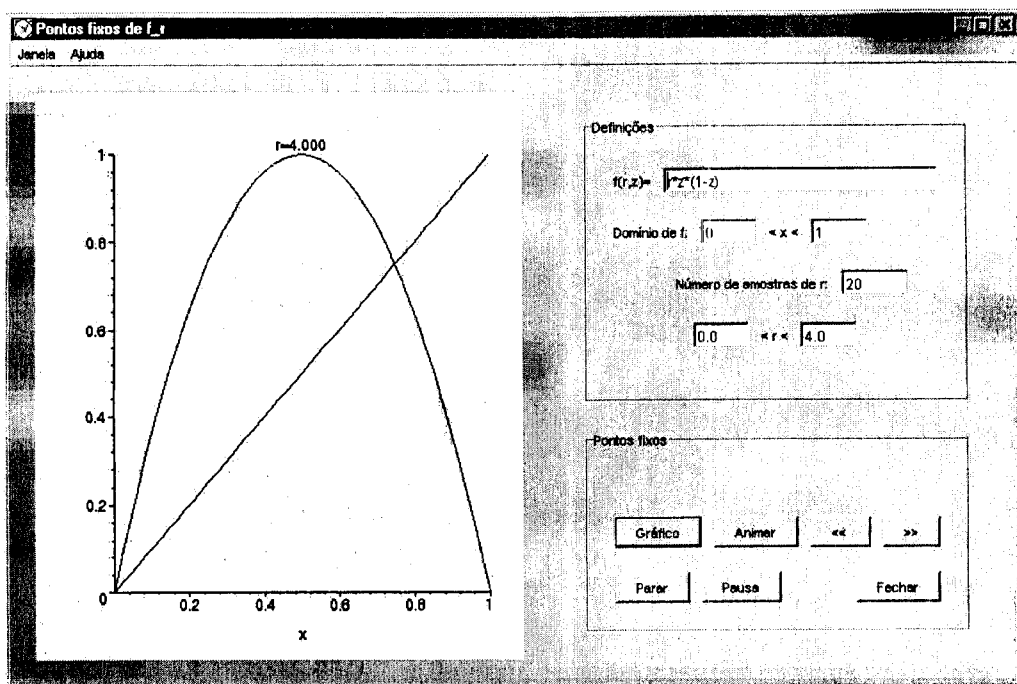
Nesta Maplet (fig. 6.16), o utilizador só tem de introduzir a expressão da função f (por defeito, $f(x) = 4x(1 - x)$) e o domínio da função (por defeito, $0 < x < 1$). Depois, para ver os gráficos da função e da recta $y = x$, clica no botão 'Gráficos'. Para fechar a janela, clica no botão 'Fechar'.

A Maplet anterior pode ser melhorada, na perspectiva do aluno poder visualizar os pontos fixos de uma função quando a expressão que a define varia com um parâmetro r . Para isso, é vantajoso para o aluno, que ele tenha acesso a uma simples animação, onde seja possível ver as sequências da função correspondentes a diferentes valores do parâmetro r . É este o objectivo da Maplet da figura 6.17, onde o aluno pode estudar a variação dos

Figura 6.16: Pontos fixos de f

pontos fixos de uma aplicação $f_r(x)$ que dependa de um parâmetro r .

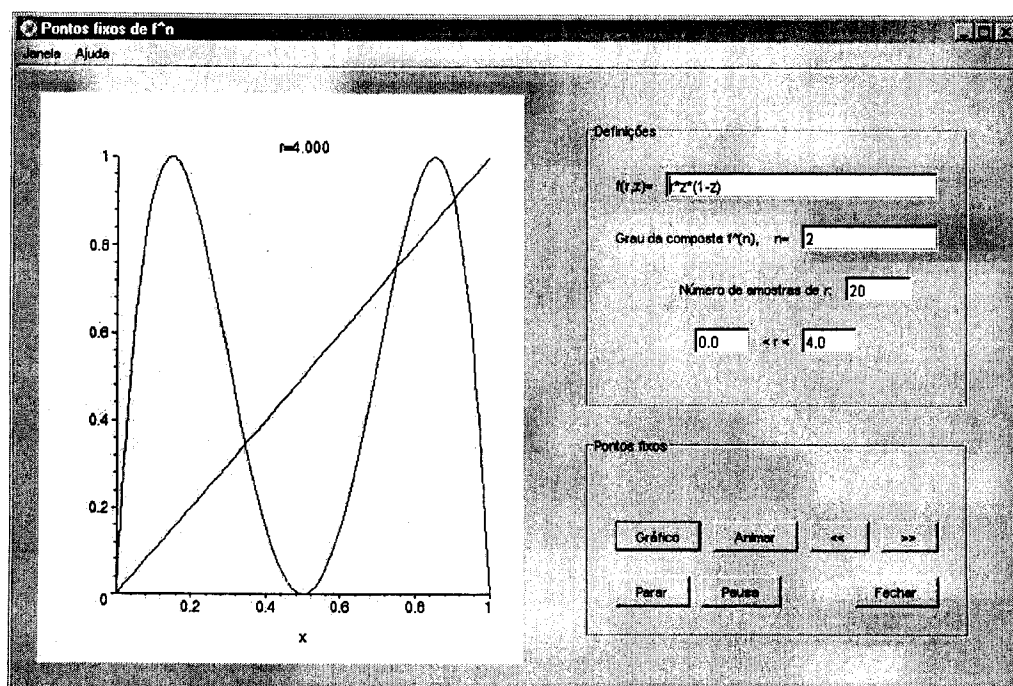
Nesta Maplet (fig. 6.17), deve inserir-se a função $f(r, z) = f_r(z)$, os valores do domínio de f_r , o número de amostras para o parâmetro r e o intervalo de variação do r . Por defeito, $f(r, z) = rz(1 - z)$, o domínio é $[0, 1]$, o número de amostras para r é 20 e o intervalo de variação do r é $[0.0, 4.0]$. O gráfico aparece para os valores dados por defeito quando se abre a janela. Se utilizador modificar alguns valores, deverá clicar no botão 'Gráfico' para visualizar o novo gráfico. Para obter a animação do gráfico só tem de clicar em 'Animar'. Para interromper a animação, deverá clicar-se em 'Parar'. Se o utilizador quiser parar a animação para um determinado valor de r e logo de seguida continuar com a animação a partir desse ponto, deverá clicar em 'Pausa' e depois em 'Animar'. Para "andar" um passo atrás ou um passo á frente, deverá clicar respectivamente em '<<' e '>>'. Para fechar a janela clica-se no botão 'Fechar' ou selecciona-se 'Fechar' do menu 'Janela'. Para abrir uma outra janela com um texto de ajuda sobre o funcionamento desta Maplet, deverá seleccionar 'Sobre a Maplet' do menu 'Ajuda'.

Figura 6.17: Pontos fixos de f_r

Esta Maplet foi melhorada (fig. 6.18) de forma a ser possível estudar os pontos fixos para f^n , $n \in \mathbb{N}$.

Nesta Maplet (fig. 6.18), o utilizador pode escolher a expressão para a função $f^{(n)}(z)$, um valor inteiro n e valores a e b para o intervalo $[a, b]$ que representa o domínio da função. A partir desta representação gráfica, é possível contar o número de pontos fixos da função $f^{(n)}(z)$, no intervalo $[a, b]$.

O utilizador deverá introduzir a função $f(z)$, o número n relativo ao grau da composta $f^{(n)}$, o número de pontos de amostragem do parâmetro r , o intervalo da variação do r e o intervalo do domínio da função. Por defeito, a função introduzida é a função $f(r, z) = rz(1 - z)$, o grau da composta é $n = 2$, o número de iterações é 20 e o intervalo de variação do r é o intervalo $\{0.0, 4.0\}$, tal como na Maplet da figura 6.17.

Figura 6.18: Pontos Fixos de f^n

A próxima Maplet (fig. 6.19), pode ser usada para análise gráfica da órbita de um ponto para funções reais de variável real.

Dada uma função f real de variável real e $x_0 \in \mathbb{R}$, a órbita de x_0 por f é a sucessão (x_n) , $n \in \mathbb{N}$, definida por $x_n = f^n(x_0)$.

Esta Maplet pode ser usada para ilustrar pedaços $(x_0, x_1, \dots, x_n)$ da órbita $(x_n)_{n \in \mathbb{N}}$ de x_0 pela aplicação f . De facto, recorrendo à identificação $a \mapsto (a, a)$, podemos analisar a dinâmica de $a \mapsto f(a)$ pela dinâmica de $(a, a) \mapsto (f(a), f(a))$. Se partirmos de um ponto (a, a) , traçarmos a recta vertical que passa por (a, a) , essa recta intersecta o gráfico de f no ponto $(a, f(a))$. De seguida, traça-se uma recta horizontal que passa por $(a, f(a))$. Esta recta intersecta a diagonal no ponto $(f(a), f(a))$. Temos assim o modo de ilustrar a passagem de (a, a) para $(f(a), f(a))$ e consequentemente, projectando no eixo do x , temos

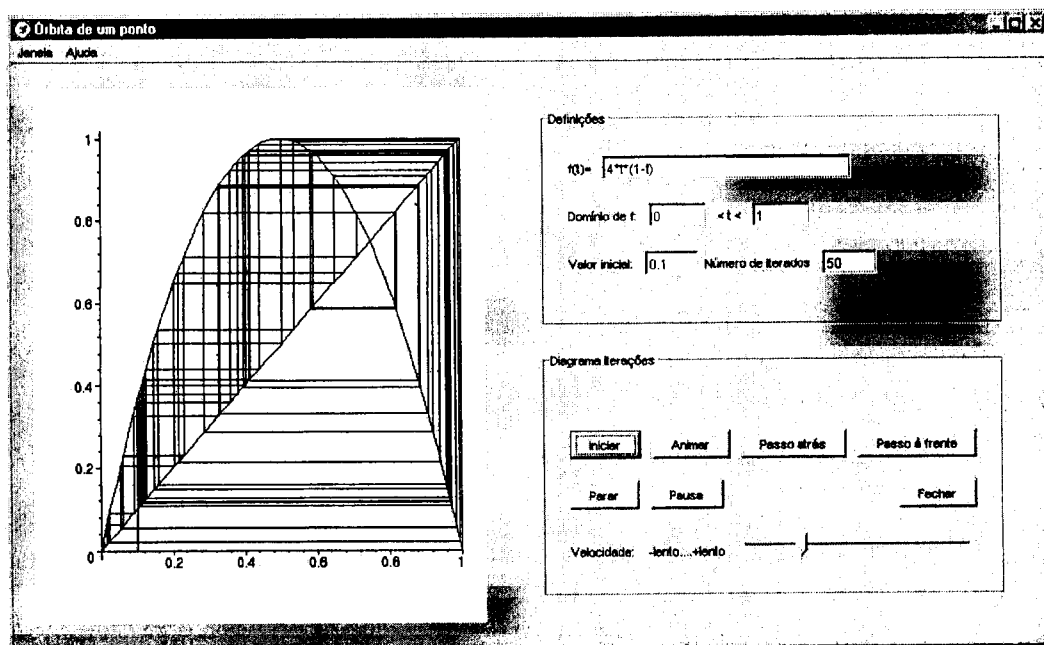


Figura 6.19: Órbita de um ponto

a passagem de a para $f(a)$. Repetindo o processo, a recta vertical que passa no ponto $(f(a), f(a))$ vai intersectar o gráfico de f no ponto $(f(a), f^2(a))$. A recta horizontal que passa por $(f(a), f(a))$ intersecta a diagonal no ponto $(f^2(a), f^2(a))$ e assim sucessivamente, obtendo o conjunto de pontos $\{a, f(a), f^2(a), \dots\}$

Quando o utilizador abre a janela, necessita de introduzir a função f que é a aplicação para a qual se quer determinar a órbita, o ponto inicial da órbita $x[0]$ e o número de iterações pretendido (por defeito, $f(t) = 4t(1-t)$, $x[0] = 0.1$ e o número de iterações é 50). Depois clica-se em 'Iniciar' para visualizar a órbita segundo a descrição do parágrafo anterior. Para o número n de iterações definidas, são esboçados os segmentos de recta $V_i = [(x_i, x_i), (x_i, f(x_i))]$ (verticais) e $H_i = [(x_i, f(x_i)), (f(x_i), f(x_i))]$ (horizontais) para todo $0 \leq i \leq n-1$.

Esta Maplet tem a vantagem de podermos ver o que acontece em cada iteração e ver a

progressão das iterações de uma forma sequencial rápida ou mais lenta. O botão 'Animar' inicia uma animação, o botão 'Passo atrás' apresenta a iteração anterior, o botão 'Passo à frente' apresenta uma iteração para a frente, o botão 'Parar' pára a animação no ponto que se desejar - se se voltar a clicar no botão 'Animar', a animação começa do início - e o botão 'Pausa' pára a animação, mas quando se clica em 'Animar', a animação recomeça do ponto onde se fez a pausa. Por baixo destes botões está um indicador que controla a velocidade da animação. Quanto mais para a direita estiver o indicador mais lenta é a animação. Finalmente, o botão 'Fechar' tem a finalidade de fechar a janela.

Esta janela também possui uma barra de menus. O menu 'Janela' tem a opção para 'Fechar' a janela e o menu 'Ajuda' tem uma opção que abre uma janela de ajuda 'Sobre esta Maplet'.

A Maplet da figura 6.20 faz um estudo estatístico da órbita de x_0 por uma aplicação $f_r : [a, b] \rightarrow \mathbb{R}$. Para $n, N \in \mathbb{N}$, considerando $I_k = [a + k\frac{b-a}{N}, a + (k+1)\frac{b-a}{N}]$, a Maplet fornece em forma de histograma o número de iterados x_i que estão em I_k , para $k = 0, \dots, N-1$. De facto, a Maplet esboça um gráfico de barras em que cada uma tem por base I_k e área igual a $\#\{x_i : x_i \in I_k\}$.

Como já referimos esta Maplet pode funcionar como uma boa oportunidade para o professor do ensino secundário relacionar conteúdos programáticos de unidades distintas (Funções e Estatística).

Para o estudo estatístico de uma órbita, podemos também recorrer ao uso de "funções teste" definidas em toda a órbita do ponto x_0 em consideração, mais especificamente tendo definido uma função real $h(x)$ com o mesmo domínio de f , podemos estudar o comportamento assintótico de $h(x_n)$ quando $n \rightarrow +\infty$, nomeadamente o "valor médio" de $h(x)$ sobre a órbita.

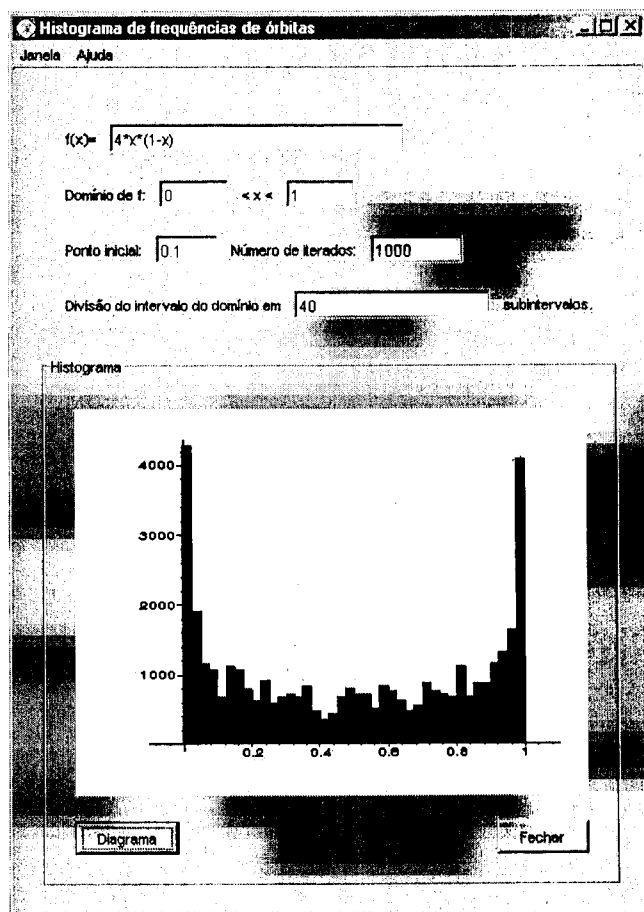


Figura 6.20: Histograma de frequências

Por exemplo, os expoentes de Lyapunov para uma aplicação f e um ponto inicial x_0 são definidos por

$$\limsup \frac{1}{n} \sum_{k=0}^{n-1} h(x_k)$$

Então podemos ver que, considerando a função teste $h(x) = \ln(|f'(x)|)$, os expoentes de Lyapunov não são mais do que $\limsup S_n$ em que

$$S_n = \frac{1}{n} \sum_{k=0}^{n-1} h(x_k)$$

Deste modo, para diferentes funções de h podemos considerar as denominadas "Som

de Birkhoff", S_n . Estudando S_n estamos, de facto, a fazer um estudo estatístico do comportamento da órbita sob a acção da função h .

A Maplet seguinte (fig. 6.21), permite então estudar a variação das Somas de Birkhoff, S_n , para aplicações f_r que dependam de um parâmetro r . Escolhendo um $x_0 \in \mathbb{R}$ inicial, definindo $f_r(x)$ e escolhendo n , a Maplet calcula para diferentes valores de r , a Soma de Birkhoff $S_n = \frac{1}{n} \sum_{k=0}^{n-1} h(x_k)$ em que $x_k = f_r^k(x_0)$.

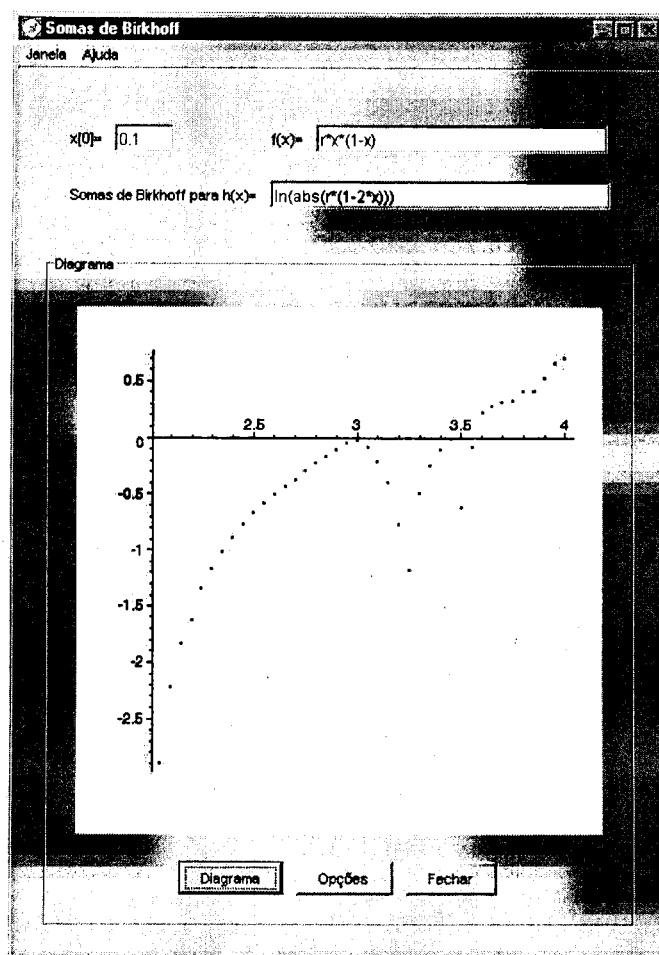


Figura 6.21: Somas de Birkhoff

Para obter o diagrama das Somas de Birkhoff, o utilizador deverá clicar no botão 'Diagrama'.

Por defeito, $f(x) = rx(1-x)$, $h(x) = \ln(|f'(x)|) = \ln(|r(1-2x)|)$ em que r é um parâmetro de variação e $x_0 = 0.1$. O utilizador pode ainda clicar no botão 'Opções' para abrir a janela 'Definições avançadas' (fig. 6.22) onde é possível alterar vários valores.

No exemplo da figura 6.21, o diagrama mostra os expoentes de Lyapunov para a família de funções $rx(1-x)$ em que o r varia entre 0 e 4. Por exemplo, para $r = 3$, o expoente de Lyapunov está bastante próximo de 0. São expoentes de Lyapunov pelo facto de $h(x) = \ln(|f'(x)|)$, caso contrário, poderíamos chamar a cada um dos pontos representados, as somas de Birkhoff para a função $h(x)$. Por defeito, são escolhidas 100 iterações, ou seja, $\text{imax}=100$, o r varia entre 2 e 4 e são escolhidos, por defeito, 40 valores igualmente espaçados para o r . Na janela 'Definições avançadas', o botão 'Ok' fecha a janela e actualiza os valores modificados, o botão 'Cancelar' fecha a janela sem aplicar os valores modificados e o botão 'Aplicar' actualiza os valores modificados sem fechar a janela.

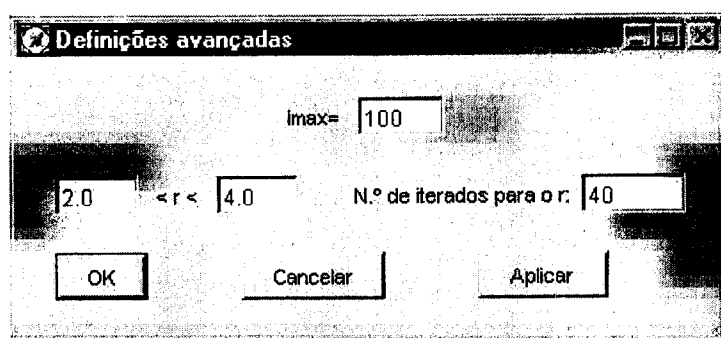


Figura 6.22: Definições avançadas

Para fechar a janela principal, clica-se no botão 'Fechar' ou selecciona-se 'Fechar' do menu 'Janela'. Para obter ajuda sobre esta Maplet, deverá seleccionar-se 'Sobre o diagrama das somas de Birkhoff' no menu 'Ajuda'.

Do mesmo modo, se poderiam construir outras Maplets, como por exemplo, fixar uma aplicação f e determinar as Somas de Birkhoff para diferentes pontos iniciais x_0 .

Agora apresentamos uma Maplet que nos permite esboçar o que classicamente em Sistemas Dinâmicos é conhecido por um diagrama da bifurcação para aplicação logística (fig. 6.23). De facto ela pode ser utilizada para determinar outros diagramas de bifurcação para outras aplicações. Dada uma aplicação real de duas variáveis $f(r, x)$ e escolhido um ponto x_0 inicial, o diagrama obtido consiste em esboçar os pontos da forma $f_r^n(x_0)$ para valores de n elevado e vários valores do parâmetro r .

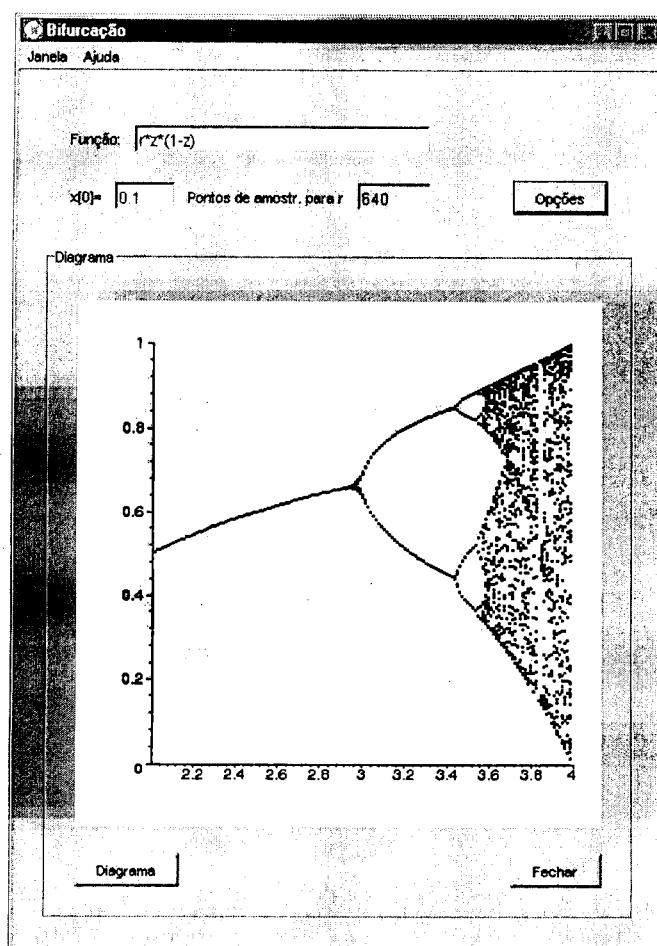


Figura 6.23: Bifurcação

Para uma função real $f_r(x) = f(r, x)$ que depende de um parâmetro r , $x_0 \in \mathbb{R}$ e $n < N$

números naturais, esta Maplet esboça os pontos $(r, f_r^k(x_0))$ para $n < k \leq N$ e para um intervalo $[r_0, r_1]$ de amostragem do parâmetro r , permitindo assim, visualizar (tomando N elevado) o comportamento assintótico de $f_r^k(x)$ com $k \rightarrow +\infty$.

Por defeito:

- $f(r, z) = rz(1 - z)$
- $x_0 = 0.1$
- Pontos de amostragem para $r = 20$
- $n = 50$ e $N = 100$
- $2 < r < 4$

O utilizador pode modificar estes valores nas caixas de texto apropriadas. Ao pressionar o botão 'Diagrama', é calculado o pedaço de órbita correspondente ao 51º iterado até ao 100º iterado de $f_r(x_0)$ com r a variar entre 2 e 4 por incrementos de 0.1. Isto significa que são representados na janela do gráfico os pontos da forma $f_r^n(x_0)$ em que $n \in \{51, 52, \dots, 100\}$ e $r \in \{\frac{i}{10}, i = 0, \dots, 9\}$. Estes valores podem ser alterados na janela 'Definições avançadas', que é aberta através do botão 'Opções' de modo a esboçar os pontos da forma $f_r^n(x_0)$ em que $n \in \{tmin, tmin + 1, \dots, tmax\}$ e $a \leq r \leq b$.

Para se obter o diagrama específico da figura 6.23, repetiram-se os seguintes passos:

- 1.º Escolhe-se um valor para o r e para $x[0]$ no intervalo $[0,1]$;
- 2.º Calcula-se a órbita de $x[0]$ por f ;
- 3.º Ignoram-se os 50 primeiros iterados e desenha-se a órbita a partir do 51.º até ao 100.º.

Na janela 'Definições avançadas' (fig. 6.24), o utilizador pode escolher entre que valores quer que o r varie, pode escolher o 'Número de pontos de amostragem para o r ', pode

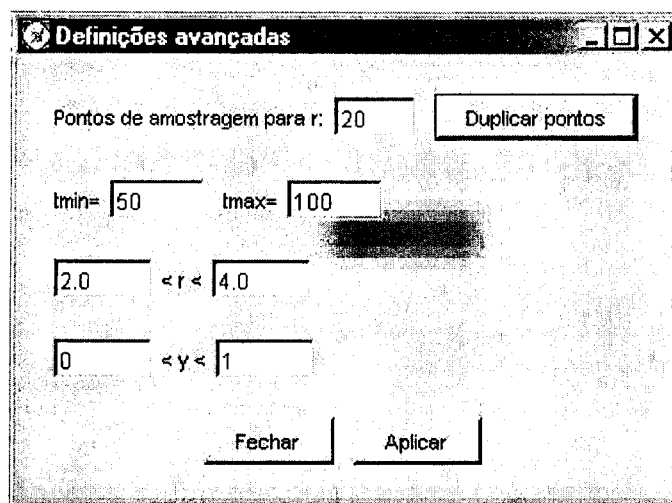


Figura 6.24: Definições avançadas

escolher entre que iterados quer que seja desenhado o diagrama e pode escolher a variação do y relativo ao eixo vertical (por defeito, $0 < y < 1$).

Para não estar sempre a escrever o número de pontos, existe um botão nesta janela que duplica o número de pontos para a amostragem do intervalo do parâmetro r . Assim, se o utilizador preferir ver o diagrama com mais detalhe, pode ir clicando no botão 'Duplicar pontos' e automaticamente o diagrama é actualizado com esse número. O botão 'Fechar' fecha a janela e o botão 'Aplicar' actualiza o diagrama com os valores definidos nesta janela.

Na janela principal, existe uma caixa de texto não editável que informa o utilizador sobre o número de pontos para o parâmetro r . No exemplo mostrado, duplicaram-se os pontos até 640.

O botão 'Fechar', fecha a janela principal. Esta Maplet contém ainda uma barra de menus. O menu 'Janela' contém apenas a opção 'Fechar' que fecha a janela e o menu 'Ajuda' contém a opção 'Sobre o diagrama de bifurcação' que abre uma janela com

uma explicação geral do funcionamento desta Maplet.

Estes são exemplos da aplicação das Maplets a uma área da Matemática tão rica como os Sistemas Dinâmicos. É suficiente visitar [22] para facilmente depreender que muitas outras áreas da matemática e não só podem ser aproveitadas para aplicações de Maplets, tornando-as mais interessantes de estudar e investigar.

Apêndice A

Códigos

Esta é uma lista dos códigos de todas as Maplets usadas neste trabalho:

- maplet simples 1 (pág. 83 - secção 3.2)
- menu 1 (pág. 84 - secção 3.2)
- botão (pág. 84 - secção 3.2)
- linha de texto 1 (pág. 85 - secção 3.2)
- expressões numéricas 1 (pág. 86 - secção 3.3)
- expressões numéricas 2 (pág. 86 - secção 3.3)
- expressões numéricas 3 (pág. 87 - secção 3.3)
- expressões numéricas 4 (pág. 89 - secção 3.3)
- equações do 2.º grau (pág. 91 - secção 3.3)
- números inteiros (pág. 92 - secção 3.3)
- derivação 1 (pág. 93 - secção 3.3)

- derivação 2 (pág. 94 - secção 3.3)
- integração 1 (pág. 95 - secção 3.3)
- limites 1 (pág. 97 - secção 3.3)
- gráfico simples 1 (pág. 98 - secção 3.3)
- gráfico simples 2 (pág. 100 - secção 3.3)
- mastermind (pág. 102 - secção 3.4)
- batalha naval (pág. 103 - secção 3.4)
- maplet simples 2 (pág. 107 - secção 4.2)
- botões (pág. 110 - secção 4.4)
- escolher cor 1 (pág. 111 - secção 4.4)
- escolher cor 2 (pág. 112 - secção 4.4)
- escolher cidade (pág. 113 - secção 4.4)
- maplet com imagem (pág. 114 - secção 4.4)
- lista (pág. 115 - secção 4.4)
- caixa de texto (pág. 116 - secção 4.4)
- linha de texto 2 (pág. 117 - secção 4.4)
- editor de matemática (pág. 117 - secção 4.4)
- integração 2 (pág. 118 - secção 4.4)
- gráfico simples 3 (pág. 119 - secção 4.4)

- escolher número 1 (pág. 121 - secção 4.4)
- escolher número com slider (pág. 122 - secção 4.4)
- tabela 1 (pág. 123 - secção 4.4)
- escolher letras 1 (pág. 124 - secção 4.4)
- layout 1 (pág. 127 - secção 4.4)
- layout 2 (pág. 127 - secção 4.4)
- layout box (pág. 127 - secção 4.4)
- layout grid (pág. 128 - secção 4.4)
- menu 2 (pág. 129 - secção 4.4)
- linha de texto 3 (pág. 129 - secção 4.4)
- menu 3 (pág. 130 - secção 4.4)
- menu 4 (pág. 131 - secção 4.4)
- barra de botões (pág. 132 - secção 4.4)
- integração e derivação (pág. 133 - secção 4.4)
- dialog alerta (pág. 136 - secção 4.4)
- dialog confirmação (pág. 136 - secção 4.4)
- dialog cor (pág. 136 - secção 4.4)
- dialog input (pág. 137 - secção 4.4)
- dialog ficheiro (pág. 137 - secção 4.4)
- dialog mensagem (pág. 138 - secção 4.4)

- dialog pergunta (pág. 138 - secção 4.4)
- tabela 2 (pág. 140 - secção 4.4)
- maplet simples 3 (pág. 141 - secção 4.4)
- mudar texto (pág. 142 - secção 4.4)
- escolher letras 2 (pág. 147 - secção 4.5)
- pergunta de sinal (pág. 148 - secção 4.5)
- gráfico simples 4 (pág. 152 - secção 4.6)
- gráfico simples 5 (pág. 152 - secção 4.6)
- gráfico função 1 (pág. 165 - secção 5.1)
- gráfico função 2 (pág. 176 - secção 5.2)
- gráfico função 3 (pág. 179 - secção 5.3)
- gráfico função 4 (pág. 195 - secção 5.4)
- gráfico função 5 (pág. 219 - secção 5.5)
- integração passo a passo (pág. 222 - secção 6.1.1)
- derivação passo a passo (pág. 228 - secção 6.1.2)
- limites passo a passo (pág. 231 - secção 6.1.3)
- gráfico 3D 1 (pág. 234 - secção 6.2)
- gráfico 3D 2 (pág. 235 - secção 6.2)
- gráfico de cores (pág. 237 - secção 6.2)

- pontos fixos (pág. 239 - secção 6.3)
- pontos fixos 0 (pág. 240 - secção 6.3)
- pontos fixos 1 (pág. 241 - secção 6.3)
- órbita de um ponto (pág. 242 - secção 6.3)
- histograma (pág. 244 - secção 6.3)
- somas de birkhoff (pág. 246 - secção 6.3)
- bifurcação (pág. 248 - secção 6.3)

Bibliografia

- [1] ALMEIDA, DANIELA : *O Maple em Sistemas Dinâmicos - Uma abordagem no ensino secundário*, Pré-publicação da dissertação de Mestrado, FCUP, 2003.
- [2] ANDRADE, L. N. : *Introdução à Computação Algébrica com o Maple*, UFP, Brasil, Maio 2003.
- [3] APOSTOL, TOM M. : *Cálculo Vol. I - Cálculo com Funções de uma Variável, com uma Introdução a Álgebra Linear*, Trad. Rio de Janeiro: Editora Reverte, 1991.
- [4] CUNHA, CLÁUDIA; VENTURA, LÚCIA; BATINAS, STELA : *Livro de Actas do ProfMat99*, APM, 1999.
- [5] DAVENPORT, J. H.; SIRET, Y.; TOURNIER, E. : *Computer Algebra. Systems and Algorithms for Algebraic Computation*, 2nd Ed., London: Academic Press, 1993.
- [6] DEVANEY, ROBERT L. : *An Introduction to Chaotic Dynamical Systems*, 2nd Ed., Addison Wesley, 1989.
- [7] ENGEL, ARTHUR : *Exploring Mathematics with your Computer*, S. L.: Mathematical Association of America, 1993.
- [8] GARVAN, FRANK : *The Maple Book*, Boca Raton: Chapman & Hall / Crc, 2002.
- [9] KEADY, GRANT : *Review of Maplets*, MSOR Connections, Vol.2, No 2, May 2002, 32-35.

- [10] KLIMA, RICHARD E.; SIGMON, NEIL; STITZINGER, ERNEST : *Applications of Abstract Algebra With Maple*, Boca Raton: Crc Press, 2000.
- [11] LYNCH, STEPHEN : *Dynamical Systems With Applications Using Maple* , Boston: Birkhauser, 2001.
- [12] PEREIRA, ANA; LOURENÇO, ANA; ALVES, VERA : *Livro de Actas do ProfMat99*, APM, 1999.
- [13] REDFERN, DARREN : *The Maple Handbook*, New York: Springer-Verlag, 1993.
- [14] SMITH, DAVID A. & OTHERS: *Computers and Mathematics - The Use of Computers in Undergraduate Instruction*, 1988.
- [15] VIVALDI, FRANCO : *Experimental Mathematics with Maple*, Boca Raton: Chapman & Hall / Crc, 2001.
- [16] WATERLOO MAPLE INC. : *Maple 8 Advanced Programming Guide*, 2002.
- [17] WATERLOO MAPLE INC. : *Maple 8 Getting Started Guide*, 2002.
- [18] WATERLOO MAPLE INC. : *Maple 8 Introductory Programming Guide*, 2002.
- [19] WATERLOO MAPLE INC. : *Maple 8 Learning Guide*, 2002.
- [20] WATERLOO MAPLE INC. : *The Maple Reporter (Winter 2002)*, 2002.
- [21] www.mapleapps.com
- [22] www.mapleapps.com/categories/maplets/html/maplets_intro.shtml
- [23] www.mapleapps.com/categories/maplets/html/maplets_games.shtml
- [24] www.maplesoft.com
- [25] www.maple4students.com
- [26] www.quarks.com.br/i-waterloo-maple.htm