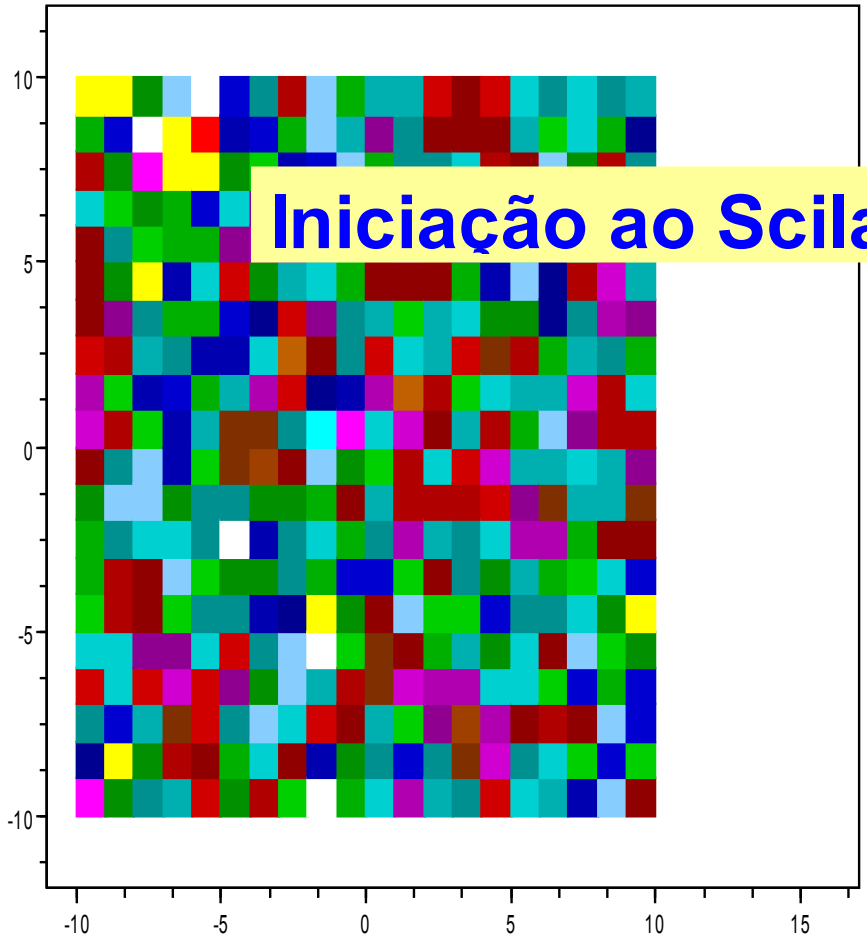
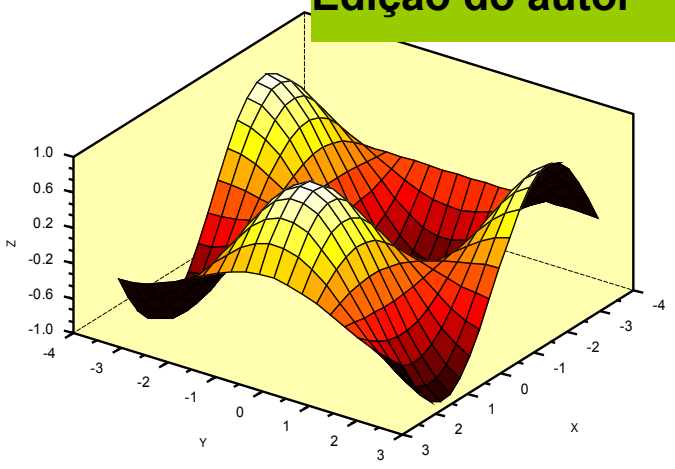
Luís Soares Barreto



Iniciação ao Scilab

$f(x)=2\ sin(x)+5$

Edição do autor



Costa de Caparica
2008

Gráficos da capa obtidos de exemplos

da Ajuda do Scilab, editados,

e depois manipulados no Word

Iniciação ao Scilab

Sem texto

Iniciação ao Scilab

Luís Soares Barreto

Professor Catedrático Jubilado
do Instituto Superior de Agronomia

INICIAÇÃO AO SCILAB

Edição do autor

Prof. Doutor Luís Soares Barreto
Av. do Movimento das Forças Armadas, 41 – 3D
2825-372 Costa de Caparica
Portugal

Com os melhores cumprimentos

Para a Sandra Isabel,
que trouxe à minha atenção
vário software de fonte aberta e livre,
com terna gratidão

Sem texto

"Grande, todo-poderosa, todo aperfeiçoadora, e divina é a força do número, começo e governante da vida divina e humana, participante de tudo. Sem número, tudo é confuso e obscuro"

Filolau de Cretona, filósofo pitagórico, séc. V a. C.

"Não acreditamos que no mundo físico exista alguma coisa não-matematizável."

P. J. Davis e E. R. Hersh. *Descarte's Dream: The World According to Mathematics.*

Traduzido em Davis e Hersh (1997:30)

Luís Soares Barreto, engenheiro silvicultor, M.F., Ph. D. (Duke University, E.U.A.), é professor catedrático jubilado do Instituto Superior de Agronomia, da Universidade Técnica de Lisboa, tendo anteriormente exercido actividade na investigação e ensino superior, em Moçambique. É o único português que estabeleceu uma teoria científica. Começou por criar uma teoria, de carácter sintáctico, dedutivo-matemático, abrangendo de forma unificada povoamentos florestais puros e mistos, tanto regulares como irregulares, sendo a única disponível, com esta abrangência, neste domínio. Em 2005, apresentou uma construção do mesmo tipo no âmbito da ecologia teórica, de que a primeira passou a ser um caso particular. É sócio honorário da Associação Portuguesa de Engenheiros do Ambiente por ter concebido, instalado, e consolidado a primeira licenciatura em engenharia do ambiente, em Portugal, na Faculdade de Ciências e Tecnologia, da Universidade Nova de Lisboa, em 1977, onde prestou colaboração graciosa durante cerca de oito anos. A sua lista de textos didácticos, científicos, técnicos e de comunicações a várias reuniões científicas ultrapassa as duas centenas. Exerceu actividade de consultoria no âmbito das engenharias do ambiente e florestal.

Trabalhos recentes do autor

Conceitos e Modelos da Dinâmica de uma Coorte de Árvores. Aplicação ao Pinhal. "e-book". 2ª edição. Instituto Superior de Agronomia, Lisboa, 2004.
Pinhais Bravos. Ecologia e Gestão. "e-book". Instituto Superior de Agronomia, Lisboa, 2004.
Theoretical Ecology. A Unified Approach. "e-book". Edição do autor, Costa de Caparica, 2005.
The Stochastic Dynamics of Self-Thinned-Pure Stands. A Simulative Quest. *Silva Lusitana,* 14(2):227-238, 2006.
The Changing Geometry of Self-Thinned Mixed Stands. A Simulative Quest. *Silva Lusitana,* 15(1):119-132, 2007.

Trabalhos submetidos para publicação

The Reconciliation of r-K, and C-S-R Models for Life-History Strategies. Submetido à revista *Silva Lusitana,* em Fevereiro de 2004.
Growth, Regeneration, and Survival Indices for Tree Species. . Submetido à revista *Silva Lusitana,* em Maio de 2004.
O Algoritmo Barcor: Classificação de Cortiça para Rolhas Recorrendo a Quatro Atributos de Qualidade. Submetido à revista *Silva Lusitana,* em Fevereiro de 2007.
Simulação do Carbono Retido no Pinhal Bravo e da sua Acreção. Submetido à revista *Silva Lusitana,* em Setembro de 2007.
Gause's Competition Experiments with Paramecium *sps. Revisited.*. Submetido à revista *Silva Lusitana,* em Setembrro de 2007.
The Blended Geometry of Self-Thinned Uneven-Aged Mixed Stands. Submetido à revista *Silva Lusitana,* em Outubro de 2007.
*Caracterização da Estrutura e Dinâmica das Populações de Lince Ibérico (*Lynx pardinus). *Uma Digressão Exploratória.* Submetido à revista *Silva Lusitana,* em Outubro de 2007.

Índice

Prefácio

É um truísmo afirmar-se que a informática, como poderoso instrumento de cálculo, armazenamento e manuseamento de dados, potencia a matematização de praticamente quase todos os domínios do conhecimento.

Não são por isso triviais as vantagens que advêm de se poder utilizar uma linguagem de cálculo numérico. Pelas razões que a leitora encontrará adiante expostas, entendo ser o Scilab uma das linguagens mais adequadas para qualquer pessoa se iniciar na programação para o cálculo numérico.

Embora tenha outros programas comerciais de cálculo simbólico e numérico, iniciei-me no Scilab, no verão de 2005, em alguns dias, quando pretendi propor, de maneira facilmente controlável por terceiros, a minha elaboração unificada no domínio da ecologia matemática ou teórica (Barreto, 2005). Isto foi possível porque o Scilab é, de facto, fácil de aprender, poderoso e eficiente, como constatará.

É pois, o propósito deste livro introduzir o leitor no uso do Scilab, software para o cálculo numérico, de fonte aberta e livre, de origem francesa.

A escrita deste texto iniciático é informada pelas seguintes preocupações:

- Não afogar a leitora com uma avalanche de informação, que a possa confundir e desencorajar, nesta fase de aprendizagem, dominado por uma preocupação de completude.
- Permitir uma abordagem aberta, que possa posteriormente ser aprofundada, na direcção que for tida por mais conveniente.
- Tirar todo o partido dos instrumentos que o software disponha, capazes de facilitar a sua utilização. O surgimento do rato e das interfaces gráficas não são alheias à rápida difusão da informática.
- Construir uma introdução de forma progressiva, inicialmente centrada no cálculo das soluções de problemas do conhecimento da leitora, desde o ensino pré universitário, e depois passar a situações eventualmente menos .familiares. Não é pois uma iniciação para 'cientistas e engenheiros', mas pretende abranger um público mais vasto.

Procurei tornar mais agradável e frutuosa a leitura do texto, através dos seguintes procedimentos:

- Na medida do possível, padronizar a exposição dos temas, e permitir uma antevisão clara dos tópicos que irá encontrar em cada capítulo.
- Fornecer, no início do livro, uma sinopse facilmente consultável, de todos os capítulos, no sentido de, em qualquer ocasião, ser possível encontrar-se sem esforço, o que se pretende..
- Encerrar os capítulos com o resumo dos comandos em linguagem Scilab, neles introduzidos.
- Inserir exercícios de forma informal, ao longo do texto, para estimular a leitora a controlar o seu progresso na aprendizagem do Scilab.

Quando iniciei este texto, pretendia que fosse particularmente dirigido a biólogos e ecologistas, mas como não conheço outro similar em língua portuguesa e não existe

tradução para português da ajuda do Scilab, procurei escrevê-lo na perspectiva de poder ser útil a qualquer pessoa interessada. No entanto, reconheço que é impossível expurgar completamente a influência do domínio da minha especialização, no produto final que disponibilizo aqui.

O livro, para além da Sinopse Geral, estende-se por oito capítulos. O primeiro tem por finalidade proporcionar, de uma maneira simples e suave, uma iniciação ao uso do Scilab.

O capítulo dois pretende ser uma transição para aplicações mais avançadas. Os conhecimentos de Scilab aqui introduzidos são-no através de exercícios simples de aritmética, matemática e física elementares, recorrendo a conhecimentos que adquiriu nos ensinos básico e secundário.

A criação, manipulação e análise de vectores e matrizes preenchem o capítulo terceiro.

No quarto capítulo inicio o leitor na utilização dos vastos recursos do Scilab para criar gráficos, tirando o máximo benefício da interface gráfica das janelas dos gráficos.

No quinto capítulo introduzo outros comandos do Scilab, tidos como entre os mais comumente utilizados.

Para a leitora não familiarizada com a programação, exponho os conceitos básicos sobre o tema, no sexto capítulo.

O capítulo sétimo ocupa-se da programação em Scilab e, finalmente, o oitavo apresenta vários programas de aplicação.

Ao longo do texto, uso a primeira pessoa tanto no plural como no singular. Não é inconsistência. No primeiro caso por entender ser a leitura deste livro uma conversa a dois, uma aventura intelectual compartilhada, de certo modo um acto de companheirismo. No segundo, porque ocorrem situações ao longo da caminhada, que têm de ser assumidas só pelo autor.

Desejo-lhe uma agradável e proveitosa leitura e .... mãos à obra.

L. S. B.

Costa de Caparica, Março de 2008

Inicie-se,

ficando já a conhecer…

Sem texto

…uma dúzia de boas razões para usar o Scilab

No cálculo numérico, a utilização do Scilab pode ser defendida, genericamente, com os seguintes argumentos:
1- É fácil a aprendizagem do Scilab, embora seja um ambiente poderoso e rápido de cálculo numérico.
2 - É simples a obtenção de gráficos de alta resolução.
3 - Os dados, ao fim e ao cabo, são fundamentalmente vectores ou matrizes de números (lembre-se das folhas de cálculo). O Scilab é particularmente dotado para lidar com estas estruturas, no que resulta em elevada rapidez de cálculo.
4 - Os resultados e gráficos que proporciona podem ser copiados e inseridos facilmente noutros documentos.
5 - Os resultados numéricos que proporciona são de elevada precisão.
6 - Depois do MATLAB, tido com a língua franca do cálculo numérico em ciência, o Scilab deve ser o software mais usado por matemáticos, cientistas e engenheiros, para o mesmo fim.
7 - O Scilab tem acesso aos ficheiros de MATLAB e do Excel.
8 - É um software aberto em contínuo aperfeiçoamento e evolução.
9 - É o melhor substituto para as calculadoras, susceptíveis a erros de propagação de arredondamentos, e com visor pequeno.
10 - Dada a afinidade entre as duas linguagens, quem aprende Scilab é-lhe fácil transitar para MATLAB, e vice-versa.
11 - É gratuito, enquanto o MATLAB é considerado caro mesmo por autores americanos (Morris e Doak, 2002, página 14, nota 4), embora a sua versão para estudantes seja relativamente acessível.
12 - Os alunos do ensino pré-universitário têm a oportunidade para simultaneamente intensificarem a sua familiarização com a informática e a língua inglesa.

Sem texto

0 Sinopse Geral

0.1 Capítulo 1 – Mãos à obra

0.1.1 Tópicos

Este capítulo tem por finalidade proporcionar-lhe, de uma maneira simples e suave, uma iniciação ao uso do Scilab.

- Vai ficar a ter uma ideia geral do que é o Scilab e como obtê-lo
- Como introduzir comandos no **prompt** do Scilab e obter resposta
- Conhecer as suas três principais janelas de interface com o utilizador
- A estrutura mais comum dos comandos do Scilab
- Como obter ajuda
- Como guardar o seu trabalho
- Saber como formatar números
- Familiarizar-se com algumas constantes, operadores e comandos básicos do Scilab
- Ver o Scilab a funcionar nalguns procedimentos simples, que sugerimos que repita no seu computador

0.1.2 Índice de comandos

Comandos	Descrição
%pi	3.1415927 35
-	subtracção
%e	2.7182818 35
%eps	Precisão "máquina" do Scilab 35
%i	Raiz quadrada de -1. Por exemplo, para entrar o complexo 2+4i escreve-se 35 2+4*%i 35
%inf	Infinito 35
%nan	"Not a number" 35
*****	Multiplicação 35
/	Divisão 35
^	Potência. 3^2=9 35
+	Soma 35
<	Menor que 35
<=	Menor ou igual que 35
<> ou ~=	Diferente de 35
>	Maior que 35
>=	Maior ou igual que 35
acos	Arcocosseno de 36
apropos	Pede ajuda a partir de uma palavra-chave 34, 38
asin	Arcosseno de 36
atan	Arcotangente de 36
clc()	Limpa a janela dos comandos 37
clc(n)	Limpa n linhas acima da actual linha de comandos e move o cursor para lá. 37
clear x y	Remove as variáveis x e y da memória. 38
clear()	Remove todas as variáveis da memória. 37
cos	Cosseno de 36
exp(x)	Exponecial de x, e^x. 36, 37

Comando	Descrição
format	Formata um número 35
help	Pede ajuda sobre um comando 30, 34
log	Logaritmo natural ou na base e 32, 33, 34, 36
log10	Logaritmo na base 10 36
log2	Logaritmo na base 2 36
matrix	Cria uma matriz 33
rand	Número casual entre zero e um 36
save	Guarda um ficheiro 39
SciPad()	Abre o editor de texto do Scilab 38
sin	Seno de 30, 34, 36
sort	Ordena um conjunto de números, vector ou matriz, numa sequência decrescente 36, 37
sqrt	Raiz quadrada 36
tan	Tangente de 36
who	Lista as variáveis 38
whos	Lista as variáveis longas. 38

0.2 Capítulo 2 – Exercícios de aquecimento

0.2.1 Tópicos

- Sistematizar alguns procedimentos já anteriormente ilustrados.
- Fazer operações aritméticas com escalares
- Utilizar funções trigonométricas.
- Criar um vector simples e utilizá-lo como objecto de algumas funções.
- Criar gráficos com procedimentos simples
- Calcular permutações, arranjos e combinações
- Criar uma função que permite obter permutações, arranjos e combinações
- Criar funções "on-line"
- Resolver um sistema de equações lineares
- Iniciar-se na utilização das cadeias de caracteres ("strings")
- Criar e analisar uma função polinomial
- Realizar operações aritméticas com polinómios
- Limites
- Estatística elementar
- Probabilidades elementares

0.2.2 Índice de comandos

0.3 Capítulo 3 – Vectores e matrizes

0.3.1 Tópicos

- Criar vectores e matrizes
- Conhecer matrizes especiais que o Scilab cria facilmente
- Identificar elementos de vectores e matrizes
- Manipular matrizes, como acrescentar e eliminar elementos
- Realizar operações com vectores e matrizes
- Analisar matrizes
- Resolver sistemas de equações lineares no contexto da álgebra matricial

0.3.2 Índice de comandos

Comando	Descrição
\	Divisão direita de uma matriz
.*	Multiplicação elemento a elemento
./	Divisão elemento a elemento
.\	Divisão direita elemento a elemento
bdiag	Valores e vectores próprios de uma matriz 84
det	Determinante de uma matriz 83, 84
diag	Diagonal principal de uma matriz 75, 84
expm	Exponencial de uma matriz 82, 83
eye	Matriz de zeros com 1's na diagonal principal 74
inv	Inversa de uma matriz 84, 86
linsolve	Solução de um sistema de equações lineares 85, 86
logm	Logaritmo de uma matriz 82, 83
logspace	Vector de números correspondentes aos logaritmos na base 10 do intervalo especificado, igualmente espaçados 73
ones	Matriz de 1's. Sintaxe igual à do comando **zeros**
spec	Valores próprios de uma matriz 85
sqrm	Raiz quadrada de uma matriz 82, 83
trace	Traço de uma matriz 84
zeros	Matriz só de zeros 74

0.4 Capítulo 4 – Gráficos

0.4.1 Tópicos

- Expandir a criação de gráficos de duas dimensões
- Inserir texto na área do gráfico
- Introduzir os gráficos a três dimensões
- Obter as coordenadas de pontos duma curva num gráfico
- Utilizar os menus e botões da janela dos gráficos para facilmente os formatar
- Fazer gráficos com marcas ou símbolos gráficos no traçado das curvas

0.4.2 Índice de comandos

0.5 Capítulo 5 – Outros comandos com interesse

0.5.1 Tópicos

- Ajustamento de curvas não lineares
- Interpolação
- Diferenciação
- Integração
- Sistemas de equações não lineares
- Solução numérica de equações diferenciais ordinárias
- Equações discretas
- Optimização

0.5.2 Índice de comandos

Comando	Descrição
cumprod	Calcula o produto acumulado 132-134
cumsum	Calcula a soma acumulada 132, 133
datafit	Ajusta uma equação não linear 113, 114, 116
derivative	Aproxima numericamente a derivada de uma função num dado ponto 120
fsolve	Acha a solução de um sistema de equações não lineares 122
interp1	Interpola valores de $y_j=f(x_j)$ 117, 118
interp2d	Interpola valores de $z_j=f(x_j,y_j)$ 118
interp3d	Interpola valores de $v_j=f(x_j,y_j,z_j)$ 118
intsplin	Integração numérica de dados usando interpolação pelo método spline 121
inttrap	Integração numérica de dados usando interpolação pelo método trapezoidal 121
linpro	Resolve problemas de programação linear 128-131
lsqrsolve	Ajusta uma equação não linear pelo método dos mínimos quadrados 114-116
ode	Resolve equações diferenciais ordinárias 123-127
optim	Rotina de optimização não linear 128
portr3d	Traça o diagrama de fase de um sistema de 3 equações diferenciais ordinárias 126
quapro	Resolve problemas de programação quadrática 128, 131, 132
smooth	Interpolação recorrendo ao método spline 118, 119

0.6 Capítulo 6 – Introdução elementar à programação

0.6.1 Tópicos

- Linguagem de programação
- Abordagem heurística dos problemas
- Articular algoritmos
- Estruturar fluxogramas
- Utilizar pseudocódigos
- Estruturas de controlo

- Modular programas
- Recursividade

0.7 Capítulo 7 – Programação em Scilab

0.7.1 Tópicos

Neste capítulo, é especialmente dedicado à programação numa perspectiva mais abrangente e não sequencial Abordaremos:

- Tipos de dados que o Scilab reconhece
- Matrizes multidimensionais
- Estruturas de dados
- Comandos de inquirição sobre tipos de funções
- Tipos de variáveis
- Entrada de dados
- Saída de resultados e mensagens
- Estruturas de controlo do fluxo do programa:
  - -Declarações if
  - - Declarações select e case
  - - Ciclos for
  - - Ciclos while
  - - Declarações break
- Despistagem e correcção de erros nos programas

0.7.1 Índice de comandos

Comando	Descrição
abort	Interrompe a execução do programa 181
break	Interrompe a execução de um ciclo 180
clearglobal	Apaga as variáveis globais 157
excel2sci	Lê folhas de cálculo em formato de texto (*.txt) 163
for	Comando para executar ciclos 177-179
full	Converte matrizes esparsas em matrizes completas 149
global	Define variáveis globais 156, 157
hypermat	Cria matriz multidimensional 154
if	Cria um Comando de execução condicional 167-173, 180
input	Pede uma entrada de informação através do teclado 165, 164

Comando	Descrição
isglobal	Verifica se uma variável é global 157
list	Cria um objecto list (lista) 150, 156
mgetl	Lê linhas de texto de um ficheiro *.txt 157, 158
pause	Interrompe o program e aguarda uma entrada através do teclado 181
printf	Comando para imprimir no monitor 166, 179
printsetupbox	Comando para ter acesso à jenela de controlo das impressoras 167
resume	Retoma a execução interrompida de um programa 181
return	Retoma a execução interrompida de um programa 181
select	Comando de escolha condicional 174-176
sparse	Cria uma matriz esparsa 148, 149
tk_getdir()	Obtém o caminho para aceder a um ficheiro 164
tlist	Cria um objecto do tipo tlist 150, 151, 155, 156
type	Verifica o tipo da variável 156
typeof	Verifica o tipo de um objecto Scilab 156
xls_open	Abre um ficheiro em formato do Excel 161
xls_read	Lê um ficheiro Excel previamente aberto 161, 162
x_message	cria uma janela para mensagem 164, 165
while	Comando para ciclo condicional 179, 180

0.8 Capítulo 8 – Aplicações

0.8.1 Tópicos

Numa escolha arbitrária, apresentam-se os seguinte doze programas:

- reglinmul – ajusta modelos da regressão linear multivariada
- RK4 – aplica o método de Runge-Kuta de 4ª ordem a um sistema de EDO
- sbpred11 – acha o equilíbrio do modelo SBPRED(1,1) para a interacção predador-presa e analisa a sua estabilidade

- paramcaos – aplica o modelo BACO2 a populações de *Paramecium* sp. E ilustra a ocorrência do efeito de borboleta determinístico
- sematdial – dada uma matriz de Leslie, estima a sensibilidade e a elasticidade do seu valor próprio dominante, os valores reprodutivos das classes e a distribuição etária estável
- SOBANPK – simula os ciclos biogeoquímicos do N, K e P no pinhal bravo regular
- Cnobravo – simula o carbono retido no pinhal bravo e a sua acreção
- plantafu – simula o crescimento de uma planta anual no hemisfério norte, com selecção de data de sementeira e latitude
- funcroc – simula o crescimento de uma população de crocodilos australianos
- ProbExt – calcula a probabilidade acumulada de extinção de uma espécie
- plamarg – simula o planeta das margaridas de James Lovelock
- fuMBE2 – um modelo de balanço da energia da Terra
- Todos os programas, em formato *.sci, encontram-se na pasta "Aplicações", que acompanha este livro. Para os correr basta **File→Exec...**. e dois cliques no ficheiro escolhido.

Capítulo 1

Mãos à obra

Este capítulo tem por finalidade proporcionar-lhe, de uma maneira simples e suave, uma iniciação ao uso do Scilab.

- Vai ficar a ter uma ideia geral do que é o Scilab e como obtê-lo
- Como introduzir comandos no **prompt** do Scilab e obter resposta
- Conhecer as suas três principais janelas de interface com o utilizador
- A estrutura mais comum dos comandos do Scilab
- Como obter ajuda
- Como guardar o seu trabalho
- Saber como formatar números
- Familiarizar-se com algumas constantes, operadores e comandos básicos do Scilab
- Ver o Scilab a funcionar nalguns procedimentos simples, que sugerimos que repita no seu computador

1.1 Apresentando o Scilab

O Scilab é um potente software livre

- De cálculo numérico;
- Visualização gráfica;
- E simultaneamente uma linguagem de programação de nível elevado.

Não faz operações de cálculo simbólico, mas permite interfaces com software que o faça.

O Scilab aceita vários tipos de dados e é baseado fundamentalmente na manipulação de vectores e matrizes. Contém uma rica colecção de algoritmos que podem ser utilizados para executar complexas operações de cálculo e traçar gráficos, com comandos simples e curtos. O facto de ser uma linguagem de programação permite ao utilizador desenvolver comandos para seu uso próprio, ou programá-los em linguagem C ou FORTRAN.

O Scilab tem inúmeros comandos que são pequenos programas chamados "functions" (funções), agrupados, de acordo com a afinidade dos algoritmos que executam, em "toolboxes" (caixas de ferramentas) que designaremos por **bibliotecas**. Este software dispõe de um manual em ficheiro PDF (Scilab Reference Manual) onde são identificadas as bibliotecas e as suas funções descritas.

As bibliotecas do Scilab são as seguintes: Programming ; Graphics Library ; Elementary Functions; Input/Output Functions; Handling of functions and libraries; Character string manipulations; GUI and Dialogs; Utilities; Linear Algebra; Polynomial calculations; General System and Control; Robust control toolbox; Optimization and simulation; Signal Processing toolbox; Arma modelisation and simulation toolbox; Metanet: graph and network toolbox; Sound file handling; Language or data translations; TdCs; Statistic basics; Cumulative Distribution Functions; Inverses, grand; Identification; Matlab to Scilab conversion tips; Sparse solvers; PVM parallel toolbox; TCL/Tk interface; Java Interface.

Todos os comandos ou funções do Scilab, e as suas descrições sumárias (uma linha) em Times New Roman, tipo 12, estendem-se por 31 páginas A4.

Não surpreende que neste livro introdutório não abordemos todas as bibliotecas, nem utilizaremos todos os comandos daquelas a que recorreremos. Além dos procedimentos indispensáveis e básicos de uso do software (e.g., input/output), a nossa escolha recairá naqueles de maior relevância para os cálculos associados aos instrumentos matemáticos e de visualização tidos como os mais utilizados.

No entanto, convém esclarecer que com o enriquecimento da abordagem quantificada destas ciências, este quadro de referência não é estático e outros comandos e bibliotecas podem vir a ganhar interesse. A abordagem faz-se também na perspectiva de que uma vez iniciado, o leitor poderá por si, com os instrumentos de apoio que acompanham o Scilab e outros que venha a dispor, alargar o horizonte da sua utilização. Mas não se esqueça que o software de cálculo, simbólico e numérico, ser-lhe-á de pouca utilidade se não tiver o conhecimento adequado dos sistemas que aborda, e dos instrumentos matemáticos a que recorre.

O Scilab pode ser instalado em ambientes Windows e Linux.

Se a leitora já utilizou o software MATLAB não vai ter nenhuma dificuldade em utilizar o Scilab pois há grande afinidade entre eles.

Não existe versão portuguesa do Scilab, só está disponível em inglês.

Na elaboração deste texto usei o Scilab 4.0.

1.2 Como obter o Scilab

O Scilab foi desenvolvido, em França, pelo Institut National de Recherche en Informatique et en Automatique (I.N.R.I.A.). Pode ser descarregado gratuitamente do site seguinte:

http://www-rocq.inria.fr/scilab

O software vem acompanhado de instruções de instalação, e esta em ambiente Windows não oferece qualquer problema. Instalei-o em computadores com a versão ME, depois a XP sem qualquer dificuldade.

Para obter documentação aceder ao site:

http://www-rocq.inria.fr/scilab/doc.html

1.3 Convenções tipográficas

Neste texto, para destacar o significado operacional dos diversos tipos de texto usarei as seguintes convenções:

- O tipo de letra usado é garamond (o presente), criado na Renascença, por um tipógrafo francês chamado Claude Garamond.
- Para comandos e outras entradas na janela de comandos do Scilab, será empregue o tipo century.
- Os resultados proporcionados pelo Scilab serão impressos em arial.
- Texto em locais onde devem ser inseridas palavras associadas à execução de um comando será apresentado em itálico. Por exemplo, para obter ajuda sobre um assunto, tópico ou comando escreve-se help *tópico*. No caso de se querer pedir ajuda sobe o comando relativo ao seno de um ângulo, escreve-se help sin.
- Na primeira vez que usamos um comando, para mais fácil localização, o seu nome vem a cor vermelha, no texto.

1.4 As janelas de interface com o utilizador

A principal interface do Scilab com o utilizador é feita através das janelas

- De comandos (figura 1.1)
- Gráficos (figura 1.2)
- Edição de comandos e programação (pode ser substituída por um processador de texto simples) (figura 1.3)

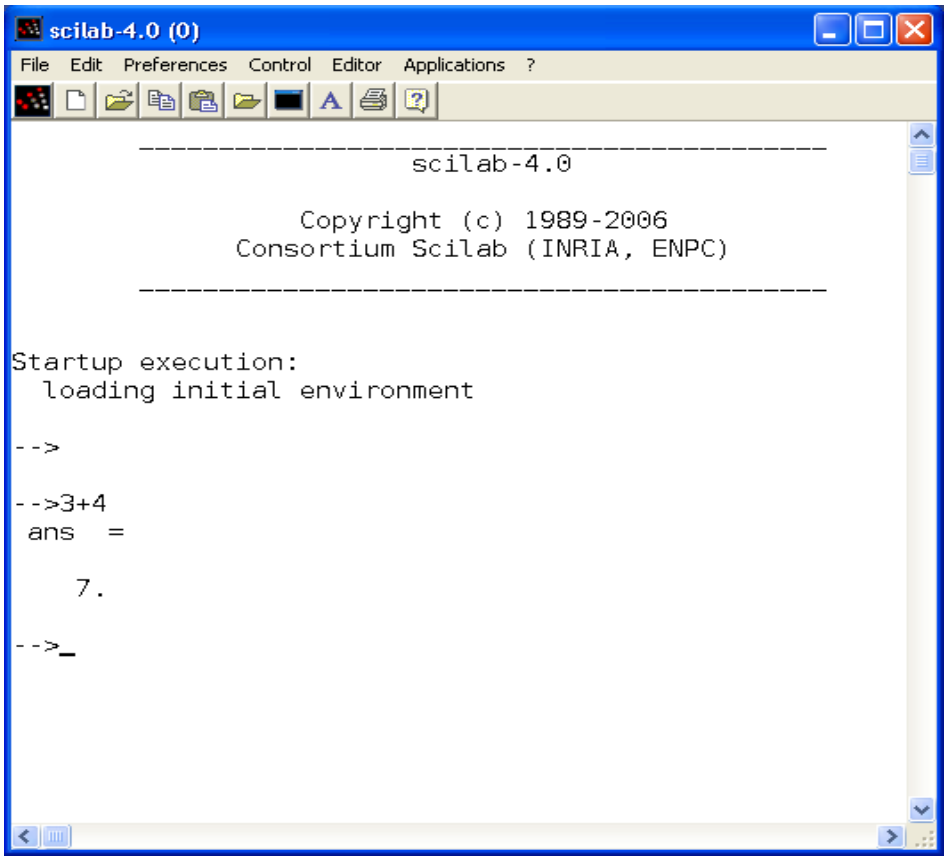


Figura 1.1. Janela de comandos do Scilab.

Para evitar uma leitura passiva deste texto, sugiro que o leitor instale o Scilab, se ainda não o fez, e abra a aplicação. Como pressuponho que está familiarizado com o uso do computador e outras aplicações, sugiro que explore os menus e botões das janelas, que usam ícones conhecidos, e basta apontar para eles com o rato para se obter informação a seu respeito.

A janela de comandos do Scilab exibe um **prompt** (-->) que assinala o local de introdução de instruções, dados, etc., pelo utilizador. A figura 1.1exibe a soma de 3 com 4, e o seu resultado (`ans` de "answer" – resposta) que é 7. A leitora já concluiu que as operações aritméticas básicas introduzem-se como se escrevem e, como todos os comandos, concluem-se pressionando a tecla **enter/return**.

O primeiro botão à esquerda permite nova(s) janela(s) de trabalho.

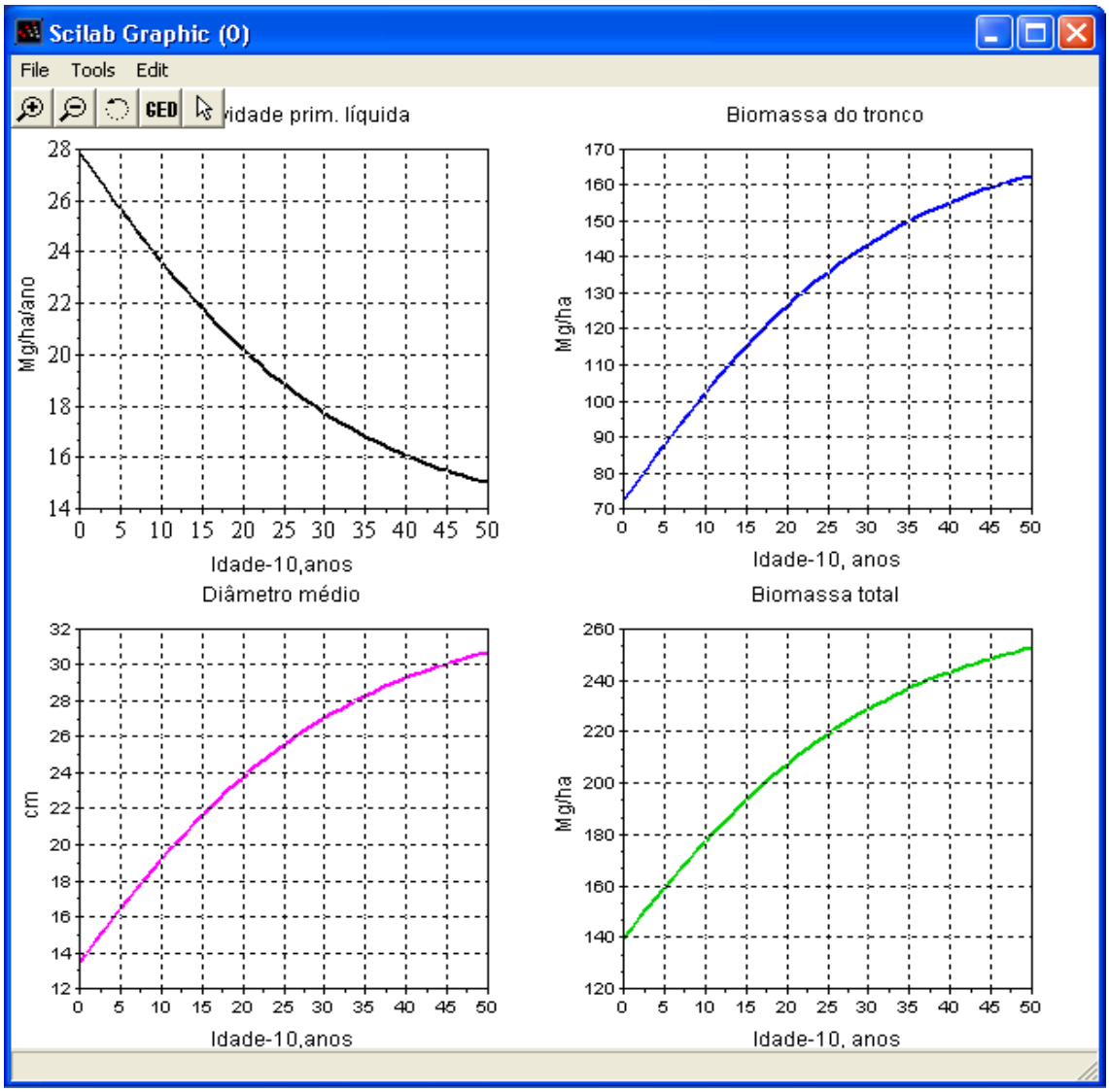


Figura 1.2. Janela de gráfico com quatro sub janelas contendo gráficos relativos à dinâmica de um pinhal bravo regular, nas idades dos 10 aos 60 anos.

Se fecharmos a janela com o rato encerra-se o Scilab. O mesmo pode ser feito escrevendo **Exit**, **Enter/Return** ou clicando esta palavra no menu **File**.

1.5 Algum conhecimento prévio necessário

Antes de iniciar uma série de sessões tutoriais convém dispor de algum conhecimento sobre certos tópicos da linguagem do Scilab, que passo a abordar.

A forma geral dos comandos do Scilab é a seguinte:

Entidade que recebe o resultado=Função do Scilab(objecto a que a função se aplica, atributos)

Um exemplo. Obter o valor do logaritmo natural de 30 e atribui-lo à variável x, com o comando log.

```
-->x=log(30)
 x  =

    3.4011974
```

Outro exemplo. Usando o comando matrix, transformar os números inteiros de 1 a 12, numa matriz de 4 linhas e 3 colunas e atribui-la a M.

```
-->M=matrix(1:12, 4,3)
 M  =

    1.    5.    9.
    2.    6.    10.
    3.    7.    11.
    4.    8.    12.
```

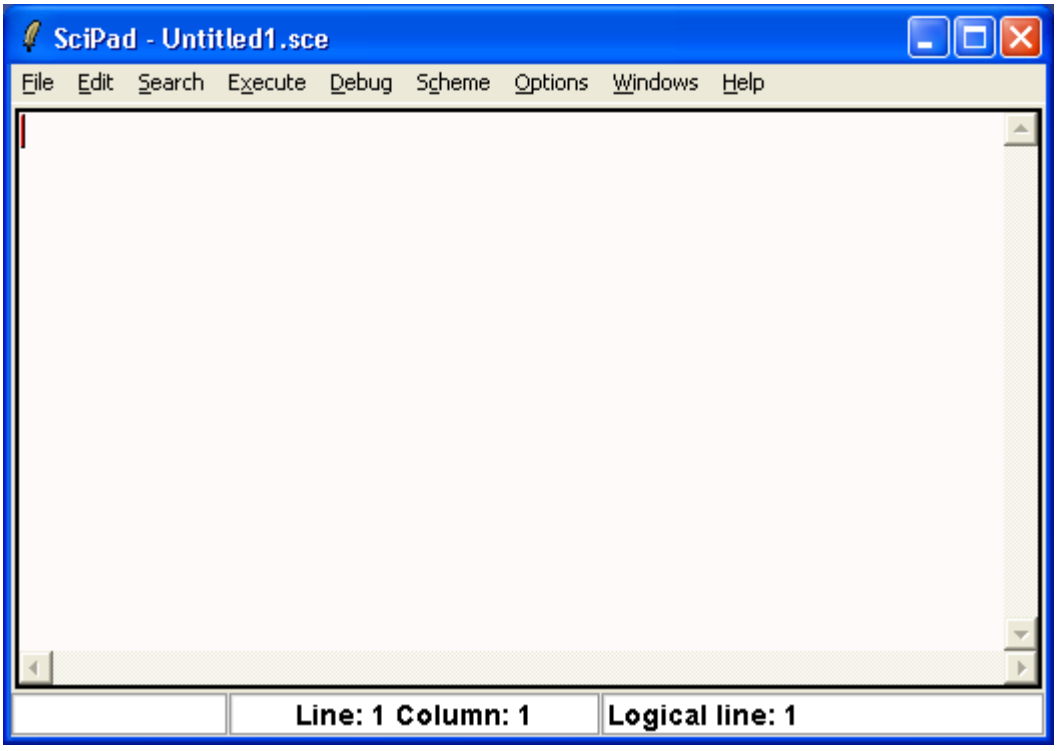


Figura 1.3. Janela do editor de comandos e programas SciPad. É um editor de texto.

O Scilab ordena os números por colunas, mas pode-se obter uma ordenação por linhas, como veremos adiante.

O leitor anotou que uma sequência de números inteiros seguidos tem a forma

Primeiro valor /dois pontos/último valor

Se os números não forem inteiros seguidos, o intervalo entre eles é introduzido do seguinte modo:

Primeiro valor /dois pontos/intervalo/dois pontos/último valor

Para criar o vector linha z com os elementos 1, 1.1, 1.2, …..2 procede-se assim:

```
-->z=1:0.1:2
 z  =

   1.    1.1    1.2    1.3    1.4    1.5    1.6    1.7    1.8    1.9    2.
```

Também podemos aplicar a função log ao vector z e obter outro vector x dos logaritmos:

```
-->x=log(z)
 x  =


        column 1 to 7

    0.      0.0953102      0.1823216      0.2623643      0.3364722      0.4054651
0.4700036

        column  8 to 11

   0.5306283    0.5877867    0.6418539    0.6931472
```

Se terminarmos uma linha com ponto e virgula, o Scilab não escreve a saída do comando no monitor.

Semelhantemente, o vector z pode ser objecto de outras funções que nos permitem obter informação sobre os seus elementos, como a média e o desvio padrão, por exemplo. O mesmo se passa com a matriz M.

O Scilab trata todas as variáveis como uma matriz. Um escalar, real ou complexo, é visto como uma matriz 1x1. Um vector linha com n elementos é visto como uma matriz 1xn. Um vector coluna como uma matriz nx1. O Scilab lida com matrizes de números, polinómios, booleanas e cadeias de caracteres ('strings').

1.5.1 Sensibilidade ao tipo de letra

O Scilab distingue entre maiúsculas e minúsculas. Se escrever sin(2) ele devolve-lhe o valor do seno de um ângulo; se escrever Sin(2) ele diz que a variável Sin não foi definida.

1.5.2 Ajuda

Os dois comandos mais utilizados para aceder a ajuda sobre o Scilab são:

apropos *palavra-chave* faz uma pesquisa a partir de uma palavra-chave.

help *tópico* obtém informação específica sobre um só comando ou função.

A ajuda do Scilab pode abrir-se escrevendo help help, usando o menu ou o ícone ?.

demoplay dá acesso a uma janela que permite correr várias demonstrações de funções do Scilab.

Os exemplos na ajuda referente a um comando podem ser seleccionados, copiados, colados e executados na janela do Scilab, se nisso houver interesse.

1.5.3 Constantes e variáveis especiais

%pi 3.1415927
%e 2.7182818
%i raiz quadrada de -1. Por exemplo, para entrar o complexo 2+4i escreve-se 2+4*%i
%inf infinito
%eps precisão "máquina" do Scilab
%nan "Not a number"

Estas variáveis não podem ser apagadas, nem alteradas.
Um exmplo:

```
-->%eps
 %eps  =

   2.220D-16
```

1.6.4 Formato dos números

O Scilab não requer o prévio dimensionamento das variáveis e separa a parte inteira da decimal com um ponto. Por defeito os números têm dez posições (incluindo o ponto). Para aumentar este valor e passar a ter uma variável longa usa-se format(*n*);variável. Por exemplo:

```
-->format(15);%pi
 %pi  =

   3.14159265359
```

1.5.5 Operadores aritméticos e relacionais

+ soma
- subtracção
* multiplicação
/ divisão
^ potência. 3^2=9
< menor que
<= menor ou igual que
> maior que
>= maior ou igual que
<> ou ~= diferente de

1.5.6 Prioridade de execução dos operadores

Além dos operadores atrás introduzidos, o Scilab tem outros que serão apresentados adiante. Os operadores agrupam-se em sete grupos de igual prioridade. Se um operador tem maior prioridade que outro é executado primeiro. Os sete grupos de prioridade dos operadores do Scilab ordenados por ordem decrescente são:

(' , .')
(^)

(* , / , \ , .* , ./ , .\ , .*. , ./. , .\.)
(+ , -)
(== , > , < , <= , >= ,~=)
(&)
(|)

A prioridade inata aos operadores pode ser alterada com recurso a parênteses. Por exemplo, 1+2^2=5 e (1+2)^2 =9. Do mesmo modo, 18/2+4=13 e 16/(2+4)=3.

1.5.7 Funções básicas

log logaritmo natural ou na base e
log10 logaritmo na base 10
log2 logaritmo na base 2
sin seno de
cos cosseno de
tan tangente de
asin arcosseno de
acos arcocosseno de
atan arcotangente de
exp(x) exponecial de x, e^x.
sqrt raiz quadrada
rand número casual entre zero e um
sort ordena um conjunto de números , vector ou matriz, numa sequência decrescente

Alguns exemplos.

```
-->exp(1)
 ans  =

   2.7182818

-->rand
 ans  =

   0.0683740
```

Criemos uma matriz 3x3, de números aleatórios com o comando rand:

```
-->A=rand(3,3)
 A  =

   0.5608486   0.1985144   0.2312237
   0.6623569   0.5442573   0.2164633
   0.7263507   0.2320748   0.8833888
```

Agora achemos a raiz quadrada de cada elemento da matriz com o comando sqrt:

```
-->C=sqrt(A)
 C  =
```

```
   0.7488983   0.4455495   0.4808573
   0.8138531   0.7377380   0.4652561
   0.8522621   0.4817414   0.9398876

-->D=sort(A)
 D  =

   0.8833888   0.5608486   0.2312237
   0.7263507   0.5442573   0.2164633
   0.6623569   0.2320748   0.1985144
```

1.5.8 Na linha de comandos

Para se mover o cursor na linha de comandos usar as teclas com setas como num processador de texto. As teclas **home (**ou a combinação Ctrl-a) e **end** (ou a combinação Ctrl-e) levam ao princípio e fim da linha respectivamente. As teclas de apagar texto usam-se com no processador de texto.

Ctrl-p chama a última linha de comando introduzida

Se colocarmos o cursor no **prompt** e pressionarmos sucessivamente a tecla ↑ vamos chamando sucessivas linhas de comando, a começar na última introduzida.

Uma linha ou qualquer região da janela de comandos pode ser seleccionada usando o rato, copiada e colada no **prompt** activo ou noutro documento.

Uma linha de comandos que termine com um ponto e vírgula (;) não mostra ou ecoa o resultado na janela de comandos. Sem ponto e vírgula ecoa a operação, como se ilustrou atrás.

Uma alternativa a estas instruções é usar o menu **Edit** e clicar em **History**.

Podem-se inserir comentários precedidos de //.

Exemplo de inserção de um comentário, sem saída do resultado (; no fim do comando):

```
-->exp(1);  //exponencial de 1

-->
```

Num comando com várias linhas, estas terminam com três pontos, até ao **Enter** final. Numa linha, pode-se introduzir mais de um comando, separados por ponto e vírgula.

```
-->a=2;b=5;z=a+b
 z  =

   7.
```

Nos comandos anteriores criámos três variáveis, a, b, z. Os nomes de variáveis podem começar por uma letra ou %, seguido de outras letras, algarismos, ou caracteres especiais, como #. Não devem ter mais de 24 caracteres.

1.5.9 Manipulação e informação sobre o espaço de trabalho

clc()	limpa a janela dos comandos
clc(n)	limpa n linhas acima da actual linha de comandos e move o cursor para lá.
clear()	remove todas as variáveis da memória.

clear x y remove as variáveis x e y da memória.
whos lista as variáveis longas.
who lista as variáveis

1.5.10 Lidar com ficheiros

Para lidar com ficheiros (abrir, guardar), mudar de directório e imprimir, a forma mais expedita de o fazer é usar o menu **file,** como num processador de texto, e activar o comando desejado. Para este efeito também existem alguns comandos específicos que podem ser seleccionados escrevendo, por exemplo, apropos print. Obtém-se uma janela. de ajuda como se exibe na figura 1.4. Clicando num dos comandos da lista apresentada à esquerda, o respectivo texto de ajuda aparece à direita, como é habitual

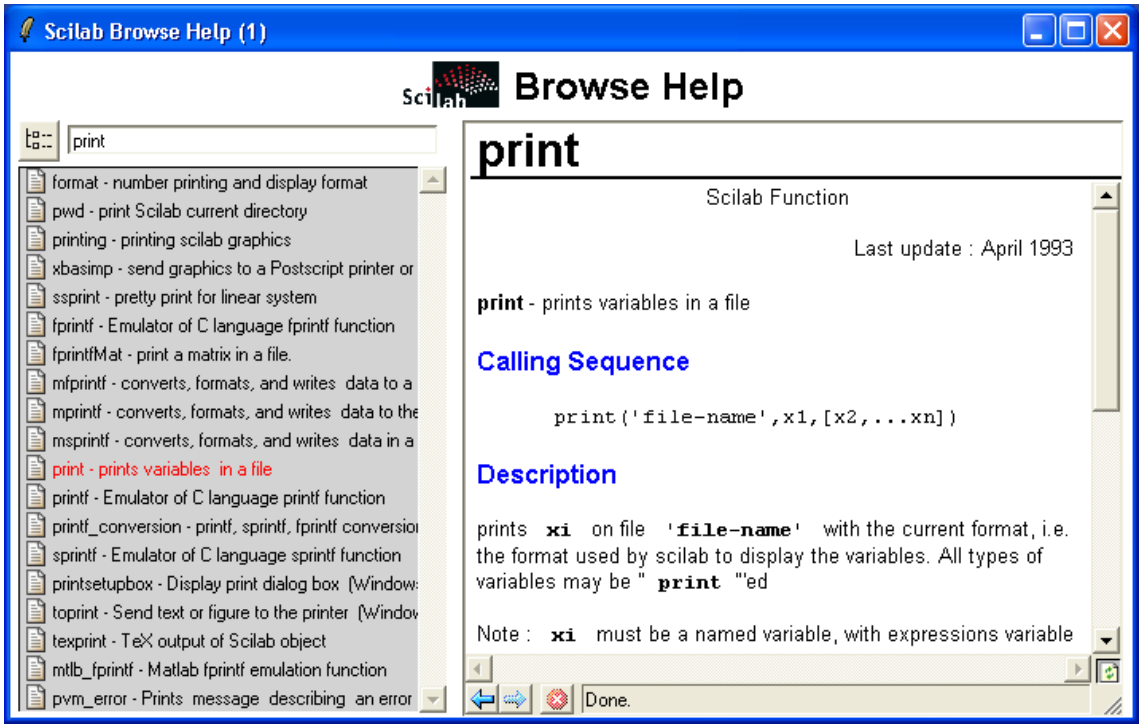


Figura 1.4. Janela da ajuda que surge depois de se ter introduzido apropos print.

1.5.11 Outra forma de utilizar o Scilab

Além da forma interactiva directa de submeter instruções ao Scilab na linha de comandos, a seguir ao **prompt**, na janela do software, há outro procedimento mais controlado de o fazer. É particularmente adequado para uma sequência de comandos interligados num algoritmo com um determinado propósito, programas e funções definidas pelo utilizador. Aqui as instruções são previamente escritas no SciPad ou num processador de texto.

O editor de texto SciPad pode ser aberto de quatro maneiras diferentes:

- Inserindo o comando SciPad().
- Inserindo o comando SciPad(*localização e nome de um ficheiro*)
- Clicando o ícone que representa uma página em branco.
- Clicando a palavra Editor.

O procedimento mais seguro é o seguinte:
- Escrever os seus comandos no SciPad, e guardar o texto como um ficheiro ***.sci.**

- Voltar à janela de comandos, clicar **File → Exec**, procurar o ficheiro na janela que se abre e fazer um duplo clique no ficheiro para ser executado

É possível seleccionar texto no SciPad, copiá-lo e inseri-lo na janela de comandos, sendo imediatamente executados, se estiverem correctos. Caso contrário, os erros serão assinalados.

Se estiver a utilizar outro processador de texto, as operações de cópia e colagem podem ser igualmente feitas, mas o texto deve ser guardado como um ficheiro ***.txt**. Agora, na janela do **Exec** tem de escolher a opção de ver todos os ficheiros, para ver os ficheiros ***.txt** e poder executá-los.

Para além destes aspectos gerais e básicos, a leitora pode obter mais informação no **Help** do SciPad.

1.5.12 Guardar e imprimir todo o trabalho na janela de comandos

O ambiente de trabalho pode ser guardado, por exemplo numa disquete, no ficheiro "meuficheiro" com o comando save, ou clicando no menu **File→Save**.

```
-->save('A:\\meuficheiro.sci')
```

Para carregar o ficheiro, clicar no menu **File→Load,** na janela que se abre, escolher a opção ver todos os ficheiros e fazer um duplo clique no ficheiro pretendido. Na janela do Scilab aparece:

```
-->load('A:\meuficheiro.sci');disp('file loaded');

 file loaded
```

A partir de agora o utilizador pode evocar as variáveis que definiu na sessão correspondente ao ficheiro.

Também se pode seleccionar todo o ambiente de trabalho (menu **Edit→Select All**), copiar e colar num processador de texto e depois guardar o ficheiro no formato ***txt**, que o SciPad abre, ou outro. No processador de texto, ou no SciPad, pode eliminar os erros e outra informação não executável e passar a dispor de um ficheiro que pode activar com a sequência, atrás descrita. **File → Exec.**

No fim do trabalho, em vez do comando save, pode usar o comando diary. O comando diary(0) interrompe o diário.

Para imprimir toda a sessão de trabalho presente na janela de comandos do Scilab, basta clicar no ícone que representa uma impressora.

Sugerimos, mais uma vez, que o leitor não faça uma leitura passiva deste livro e introduza os comandos, nele apresentados, na janela do Scilab, e procure fazer cálculos que já anteriormente realizou num calculadora, o que permite controlar os seus resultados.

Leitora, nas sociedades actuais, o conhecimento é sobretudo performativo. Sabe, quem sabe fazer.

1.6 Quadro sinóptico dos comandos introduzidos neste capítulo

Comando	Descrição
%pi	3.1415927
-	subtracção
%e	2.7182818
%eps	Precisão "máquina" do Scilab
%i	Raiz quadrada de -1. Por exemplo, para entrar o complexo 2+4i escreve-se 2+4*%i
%inf	Infinito
%nan	"Not a number"
*	Multiplicação
/	Divisão
^	Potência. 3^2=9
+	Soma
<	Menor que
<=	Menor ou igual que
<> ou ~=	Diferente de
>	Maior que
>=	Maior ou igual que
acos	Arcocosseno de
apropos	Pede ajuda a partir de uma palavra-chave
asin	Arcosseno de
5atan	Arcotangente de
clc()	Limpa a janela dos comandos
clc(n)	Limpa n linhas acima da actual linha de comandos e move o cursor para lá.
clear x y	Remove as variáveis x e y da memória.
clear()	Remove todas as variáveis da memória.
cos	Cosseno de
exp(x)	Exponecial de x, e^x.
format	Formata um número
help	Pede ajuda sobre um comando
log	Logaritmo natural ou na base e
log10	Logaritmo na base 10
log2	Logaritmo na base 2
matrix	Cria uma matriz
rand	Número casual entre zero e um
save	Guarda um ficheiro
scipad()	Abre o editor de texto do Scilab
sin	Seno de
sort	Ordena um conjunto de números, vector ou matriz, numa sequência decrescente
sqrt	Raiz quadrada
tan	Tangente de
who	Lista as variáveis
whos	Lista as variáveis longas.

Capítulo 2

Exercícios de aquecimento

Este capítulo pretende ser uma transição para aplicações mais avançadas. Para não apresentarmos os conceitos no vazio, vamos servir-nos de exercícios simples de aritmética, matemática e física elementares, recorrendo a conhecimentos que adquiriu nos ensinos primário e secundário. As finalidades deste capítulo são:

- Sistematizar alguns procedimentos já anteriormente ilustrados.
- Fazer operações aritméticas com escalares
- Utilizar funções trigonométricas.
- Criar um vector simples e utilizá-lo como objecto de algumas funções.
- Criar gráficos com procedimentos simples
- Calcular permutações, arranjos e combinações
- Criar uma função que permite obter permutações, arranjos e combinações
- Criar funções "on-line"
- Resolver um sistema de equações lineares
- Iniciar-se na utilização das cadeias de caracteres ("strings")
- Criar e analisar uma função polinomial
- Realizar operações aritméticas com polinómios
- Limites
- Estatística elementar
- Probabilidades elementares

2.1 Cálculo aritmético com escalares

Vamos começar por reintroduzir alguns exemplos já apresentados, recorrendo ao material das secções 1.6.3 e 1.6.5.

Passemos a ilustrar operações com números reais.

Introduzimos 3 varáveis (a,b,c) e depois calculamos uma expressão numérica com elas.

A expressão numérica será:

$$d = \frac{a - b}{(2c - a)^2}$$

```
-->a=3.5;b=2/3;c=0.5;

-->d=(a-b)/(2*c-a)^2
 d  =

   0.4533333

-->exp(log(d))
 ans  =

   0.4533333
```

Criemos um vector linha cujos elementos são a,b,c, e depois calculamos a soma e o produto dos seus elementos (comandos sum, prod, max).

```
-->v=[a b c]
 v  =

   3.5    0.6666667    0.5

-->sum(v)
 ans  =

   4.6666667

-->prod(v)
 ans  =

   1.1666667

-->max(v)
 ans  =

   3.5
```

A soma dos quadrados dos números 3.5, 2/3, 0.5 obtém-se facilmente:

```
-->sum(v^2)
 ans  =

   12.944444
```

Criemos o vector k e vamos distinguir os valores positivos dos negativos, com o comando sign.

```
-->k=[-2 3 4 -5 -8];

-->sign(k)
 ans  =

  - 1.    1.    1.  - 1.  - 1.
```

Os valores absolutos do elementos do vector k obtém-se com abs.

```
-->k#=abs(k)
 k#  =

   2.    3.    4.    5.    8.
```

O comando int proporciona a parte inteira de um número:

```
-->int(3.8)
 ans  =

   3.
```

Por sua vez, o comando modulo proporciona o resto da divisão de um número por outro:

modulo(*dividendo, divisor*)

```
-->modulo(44,2)
 ans  =

   0.
```

Podemos pedir a dimensão do vector k, recorrendo ao comando size:

```
-->size(k)
 ans  =

   1.    5.
```

Ilustremos o uso do comando round.

```
 -->round(3.1)
```

```
 ans  =

    3.

-->round(3.7)
 ans  =

    4.
```

Criemos o vector z utilizando os comandos ceil e floor:

```
-->z=[floor(3.7) 3.7 ceil(3.7)]
 z  =

    3.    3.7    4.
```

Calcular o volume V de um cilindro com o diâmetro do 5 cm e a altura de 13 cm.

```
-->V=%pi*(5/2)^2*13
 V  =

    255.25440310417070577387
```

Lidemos agora com números complexos, seguindo um procedimento semelhante. Sabidos os valores de a, b e c queremos calcular a expressão:

$$A = \frac{(a + 2b)^3}{cb - a}$$

```
-->a=-5+3*%i; b=-2+3*5*%i;c=6-2*%i;

-->A=(a+2*b)^3/(c*b-a)
 A  =

  - 213.51146 - 369.06334i
```

Alternativamente podemos usar o comando imult:

```
-->2+imult(5)
 ans  =

    2. + 5.i
```

Separemos a parte real da imaginária, de A, com os comandos real e imag:

```
-->real(A)
 ans  =

  - 213.51146
```

```
-->imag(A)
 ans  =

 - 369.06334
```

Obtenhamos agora o módulo de A com o comando abs:

```
-->abs(A)
 ans  =

   426.37412
```

Com o auxílio do comando conj o conjugado de A virá

```
-->conj(A)
 ans  =

 - 213.51146 + 369.06334i
```

A altura de uma planta na idade t, alt(t), é dada pela equação (de Gompertz) seguinte:

$alt(t) = h_f \, R^k$

em que h_f=14 cm; R=0.2148; $k=e^{-0.0284\,t}$. Desejamos obter o vector das alturas da planta, em centímetros, de duas em duas semanas, desde o nascimento até à idade de 140 dias. No Scilab isto obtém-se facilmente da forma seguinte:

```
-->t=0:14:140;

-->alt=14*0.2148^exp(-0.0284*t)
 alt  =


       column 1 to 5

   3.0072    4.9808255    6.9911331    8.779845    10.232157

       column  6 to 10

   11.340635    12.152137    12.729778    13.133257    13.411526

       column 11

   13.601808
```

Podemos obter os acréscimos periódicos recorrendo ao comando diff:

```
-->acres=diff(alt)
 acres  =
```

```
    column 1 to 5

 1.9736255   2.0103076   1.7887119   1.4523119   1.1084782

    column  6 to 10

 0.8115017   0.5776413   0.4034793   0.2782690   0.1902817
```

A planta tem o maior acréscimo no período entre os 14 e 28 dias.

2.2 Fazer um gráfico simples de vectores

Vamos visualizar o crescimento em altura num gráfico simples. Para obtermos mais pontos e termos uma curva mais suave, reescrevemos t como um vector de maior dimensão. E vamos usar o comando plot2d

plot2d(*valores do eixo horizontal, [valores do vector]*)

Para escrever o título e as legendas dos eixos temos o comando xtitle

xtitle(*"Título do gráfico","legenda do eixo x","legenda do eixo y"*)

```
-->t=0:0.1:140;

-->alt=14*0.2148^exp(-0.0284*t);

-->plot2d(t,[alt])
```

Podemos agora dar um título ao gráfico, inserir as legendas dos eixos e inscrever nele uma grelha (comando xgrid()), como se ilustra, obtendo o gráfico da figura 2.1.

```
-->xtitle("Altura de uma planta","Idade, dias","Centímetros");

-->xgrid();
```

Podemos também fazer um gráfico dos dez acréscimos periódicos:

```
-->x=1:10;

-->plot2d(x,[acres])

-->xtitle("Acréscimos periódicos em altura","Períodos","Acréscimos, cm");

-->xgrid();
```

O gráfico dos acréscimos exibe-se na figura 2.2.

A capacidade do Scilab para fazer gráficos é muito rica, por isso lhe dedicaremos um capítulo específico adiante.

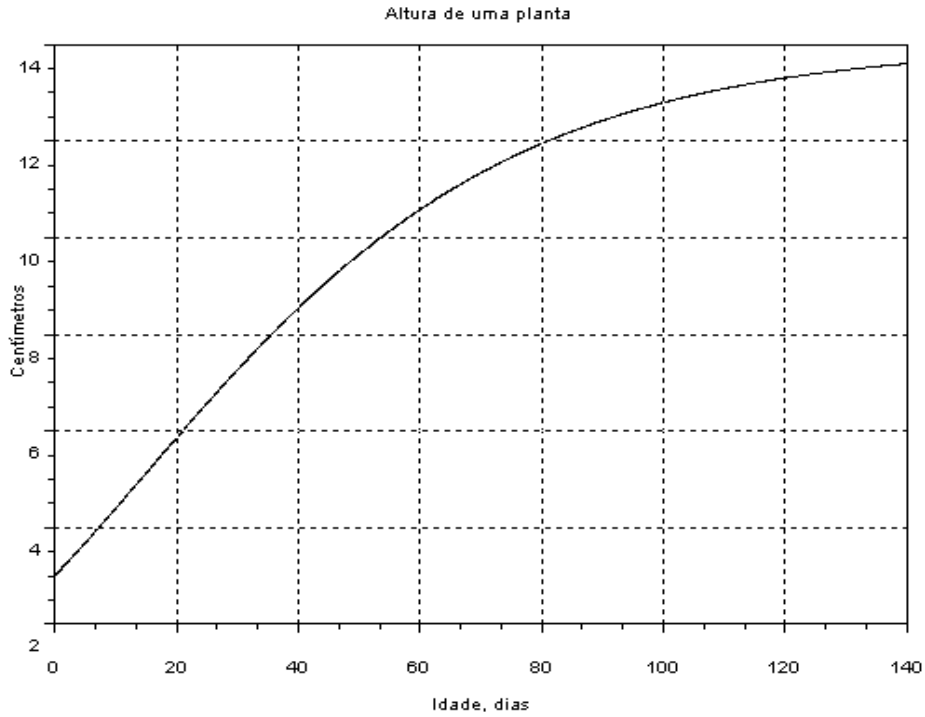


Figura 2.1. Gráfico do crescimento em altura de uma planta.

Podemos agora fazer um gráfico de uma equação exponencial e da curva de Gompertz, no domínio de 0 a três com 300 valores igualmente espaçados. Comecemos por criar o domínio com o comando linspace:

```
-->x=linspace(0,3,300);
```

x é um vector linha de 300 colunas. A matriz a grafar tem de ter 300 linhas e duas colunas. Para isto criamos a matriz com duas linhas e 300 colunas (; separa as linhas) [2*exp(0.5*x);2*0.2^(exp(-0.8*x)-1)], e depois obtemos a sua transposta usando o apostrofo (‘). Com os comandos seguintes obtém-se a figura 2.3.

```
-->plot2d(x,[2*exp(0.5*x);2*0.2^(exp(-0.8*x)-1)]')
```

```
-->xtitle("Curvas exponencial e de Gompertz","x","y");
```

Vamos verificar as dimensões da matriz [2*exp(0.5*x);2*0.2^(exp(-0.8*x)-1)]':

```
-->size([2*exp(0.5*x);2*0.2^(exp(-0.8*x)-1)]')
 ans  =

   300.    2.
```

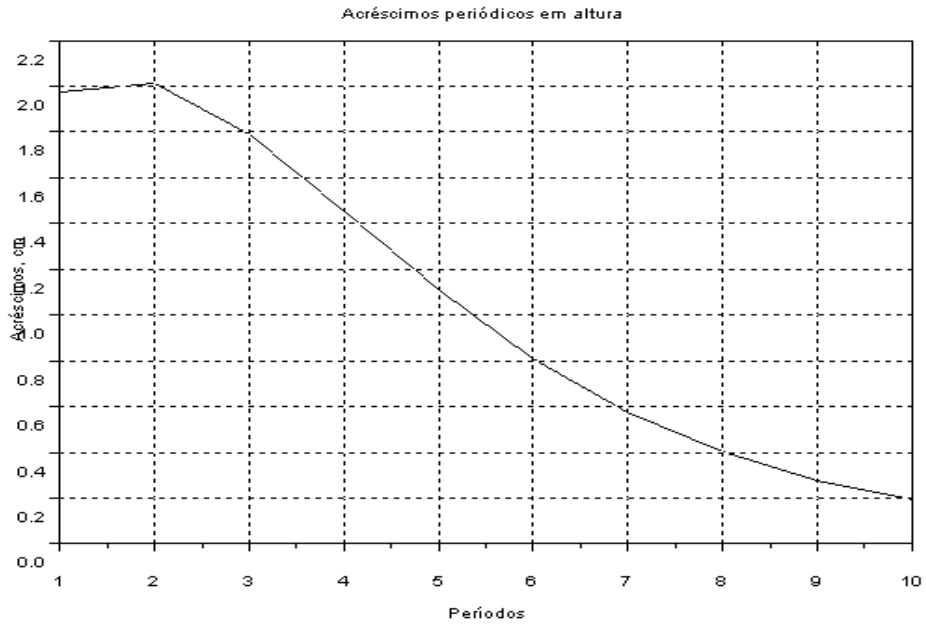


Figura 2.2. Gráfico dos acréscimos bissemanais da altura de uma planta

Vamos salvar o gráfico da figura 2.1, usando o menu **File→Save** da janela do gráfico, numa disquete na drive A. Chamamos-lhe planta.scg.

Alternativamente, podemos usar o comando xsave('A://planta.scg').

Para abrir o gráfico, clicar no menu **File→Load,** na janela que se abre, escolher a opção ver todos os ficheiros e fazer um duplo clique no ficheiro pretendido. Na janela do Scilab aparece:

```
-->load('A:\planta.scg');disp('file loaded');

 file loaded
```

Para fechar a janela de um gráfico clicar no botão X do canto superior direito, ou no menu **File→Close.**

Para limpar a janela de um gráfico, no menu **Edit→Erase figure.** Ou na janela de comandos entrar com xbasc(*número da janela do gráfico*). No caso de só existir uma janela de gráfico basta xbasc().

2.3 Análise combinatória

O comando prod permite-nos calcular o factorial de um inteiro. Suponhamos que queremos 6!. Para isso criamos o vector dos números inteiros de 1 a 6, e depois utilizamos o comando prod.

```
-->prod([1:6])
 ans  =

   720
```

O factorial do número n também se pode obter com o comando gamma(*n+1*), referindo-se gamma à função gama. 4!=24, vem

```
-->z=gamma(5)
 z  =

   24.
```

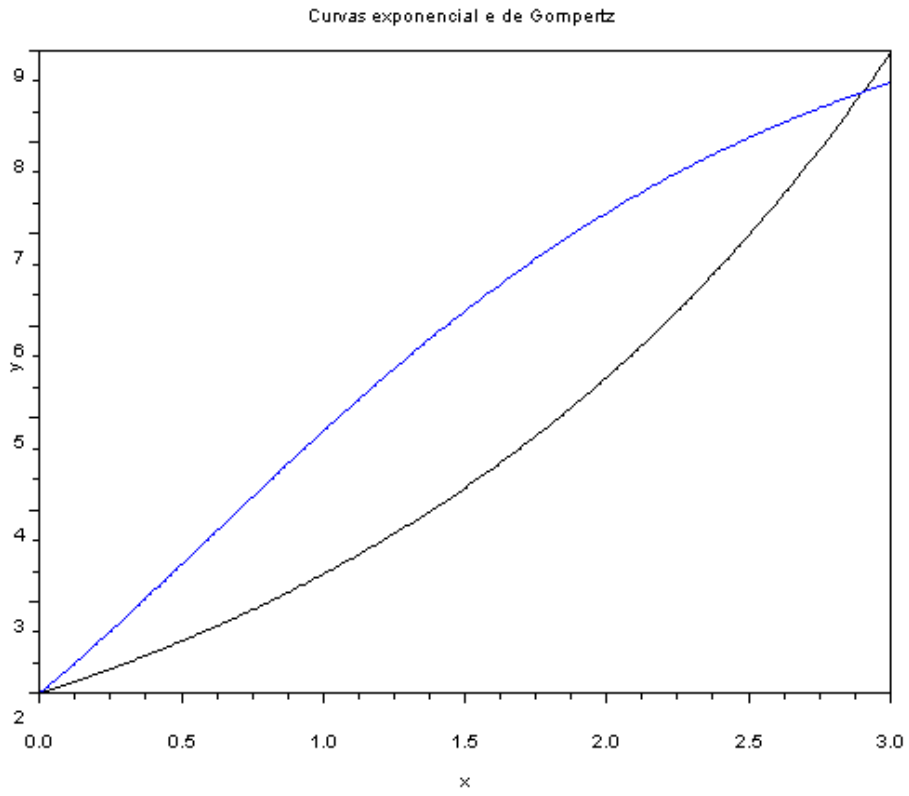


Figura 2.3. Gráfico de uma curva exponencial e outra de Gompertz

Agora, o número de arranjos de 6 objectos 2 a 2 é dado por:

```
-->n=6;r=2;

-->n2=n-r;

-->ar62=prod(1:n)/prod(1:n2)
 ar62  =

   30.
```

Para calcularmos o número de combinações de 6 objectos 2 a 2 procedemos deste modo:

```
-->co62=ar62/prod(1:r)
 co62  =

   15.
```

2.4 Cálculo trigonométrico

O Scilab expressa os ângulos em radianos.

Verificar a igualdade $sen^2 x+cos^2x=1$ (comandos sin e cos), sem saída do resultado. Atribuímos o calculo da expressão a a e depois comparamo-la com 1, usando ==. Se for verdadeira o Scilab devolve T (de "true"- verdadeiro), se for falsa a saída é F.

```
-->x=2;

-->a=sin(x)^2+cos(x)^2;

-->a==1
 ans  =

  T
```

A igualdade foi comprovada.

Para valores de x no intervalo .[-2π, 2 π}, criemos um gráfico das funções cos^2x e cos x^2., na figura 2.4.

```
-->x=[-2*%pi:0.01:2*%pi];

-->M=[cos(x)^2;cos(x^2)]';

-->plot2d(x,[M])
```

Para obter informação sobre as funções trigonométricas hiperbólicas, entre com

```
-->apropos hyperbolic
```

Dada a função f(x)=3 cos x +x , fazer o seu gráfico no domínio D_f=[0, 8π], e achar o valor mínimo da função. Vamos criar cem mil pontos no domínio.

```
-->x=linspace(0,8*%pi,100000);    //Domínio

-->f=3*cos(x)+x;                  //Valores da função

-->min(f)                         //Obtenção do mínimo de f
 ans  =

  - 0.0266714
```

Introduzimos agora o comando plot.

```
-->plot(f)

-->xtitle("Gráfico de f(x)=3*cos(x)+x","Número do ponto","f")

-->xgrid()
```

Na figura 2.5, verifica-se que comando plot(f) não revela os valores de x, mas a ordem dos pontos de zero a cem mil.

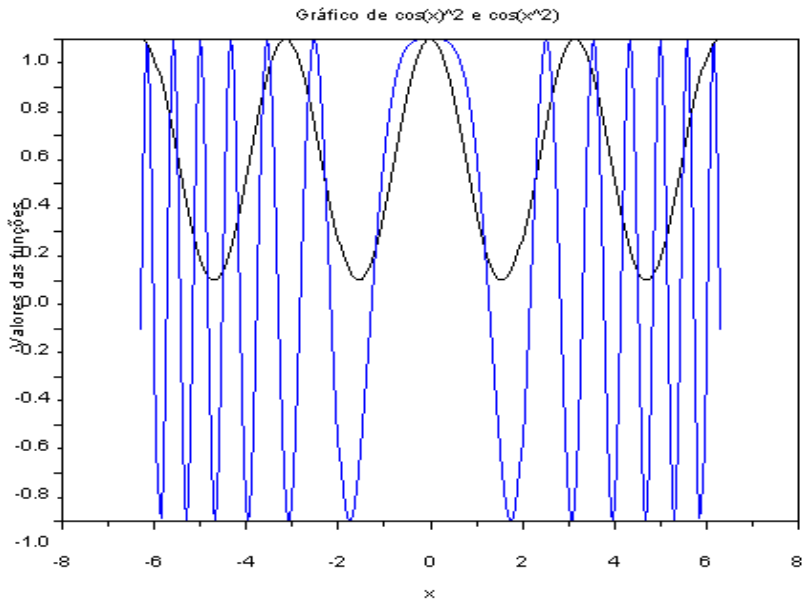
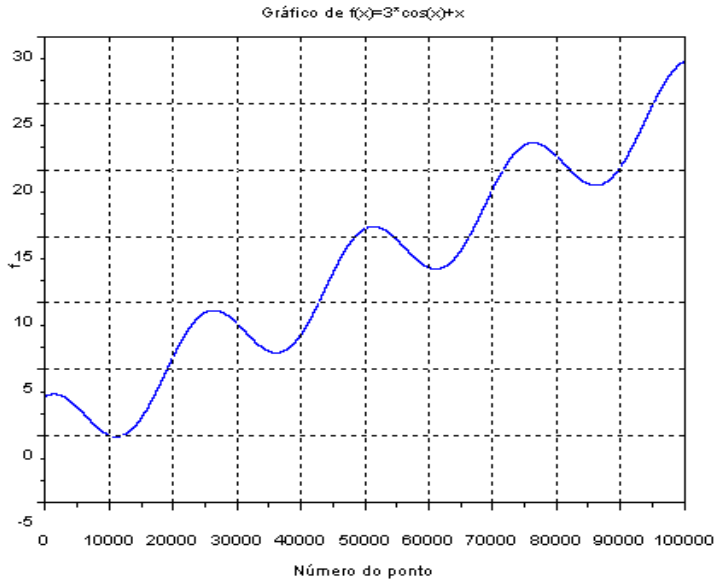


Figura 2.4. Gráficos das funções trigonométricas $\cos^2 x$ (menor amplitude) e $\cos x^2$.



.

Figura 2.5. Gráfico da função trigonométrica f(x)=3 cos x +x

2.5 Resolvendo alguns problemas de física

Agora que se querem construir mais 10 barragens em Portugal, comecemos por um problema de pressão de fluidos.

Uma barragem com o comprimento de 100 m (L), de forma rectangular, sustém uma albufeira com 30 m de altura de água (H). Pretendemos calcular a pressão hidrostática na parede da barragem.

A expressão que nos dá a pressão pretendida (F, N) é a seguinte

$$F = g\rho_a L\frac{H}{2}$$

sendo L=100 m, H=30 m, g=9,81 m s^{-2}, ϱ_a=1000 kg m^{-3}. Podemos proceder agora ao cálculo de F.

```
-->L=100; H=30; g=9.81; roa=1000;

-->F=g*roa*L*H/2
 F  =

   14715000.
```

Passemos à termodinâmica. Um aeróstato cheio com hélio (gás monoatómico) tem o volume de 400 m^3 (V), sob uma pressão de 10^5 Pa (p). Devido ao calor do Sol, a sua temperatura subiu de 18 (t_1) para 30 (t_2) graus Celsius. Deseja-se saber em quanto aumentou a energia interna do hélio.

A diferença entre as energias internas aos 18 e 30 graus Celsius (ΔU, J), é dada pela equação seguinte:

$$\Delta U = \frac{3}{2}pV(\frac{T_2}{T_1} - 1)$$

Calculemos ΔU.

```
-->3/2*10^5*400*((273.15+30)/(273.15+18)-1)
 ans  =

   2472952.1
```

Por fim, o clássico problema da cinemática, do projéctil de um canhão, ignorada a resistência do ar.

Um canhão disparou um projéctil, sob um ângulo com a horizontal de 45 graus (α), com a velocidade inicial de 380 m s^{-1} (v_0). Desejamos saber o tempo de voo do projéctil, a altura máxima que atinge e o seu alcance.

O tempo de voo (t_v , s) do projéctil é dado pela relação:

$$t_v = \frac{2v_0 sen\alpha}{g} \quad \text{segundos}$$

O tempo de ascensão é igual a metade de t_v, da a simetria da trajectória parabólica do projéctil.

A altura máxima (h_m, m) vem dada por:

$$h_m = \frac{v_0^2 sen^2\alpha}{2g} \quad \text{metros}$$

O alcance do tiro (l, m) obtém com a equação seguinte:

$$l = \frac{v_0^2 sen(2\alpha)}{g} \quad \text{metros}$$

Os cálculos no Scilab produzem um vector sol com a solução completa do problema:

```
-->vo=380;a=%pi/4;g=9.81;

-->tv=2*vo*sin(a)/g;hm=vo^2*sin(a)^2/(2*g);l=vo^2*sin(2*a)/g;

-->sol=[tv hm l]
 sol  =

   54.780953    3679.9185    14719.674
```

2.6 Uma anotação preliminar sobre cadeias de caracteres ("strings")

As cadeias de caracteres (ou simplesmente **strings**) são texto entre aspas que não é calculado nem avaliado, a não ser que se use um comando específico para o efeito. Associadas ao comando disp são úteis para tornar mais inteligível a saída dos resultados.

- Uma variável x pose ser transformada numa **string** com o comando string(x).
- Uma expressão contendo variáveis numéricas sob a forma de **strings** é avaliada pelo comando evstr.
- O comando eval avalia uma matriz de "strings"

Exemplo da aplicação de evstr associado ao comando plot2d. O gráfico inserto na figura 2.6 simula o crescimento do volume em pé de um pinhal bravo

```
-->t=10:10:80;V='380*0.4076^exp(-0.05*(t-10))';

-->plot2d(t,[evstr(V)])

-->xtitle("Crescimento do volume de um pinhal bravo","Idade, anos","m.c./ha")
```

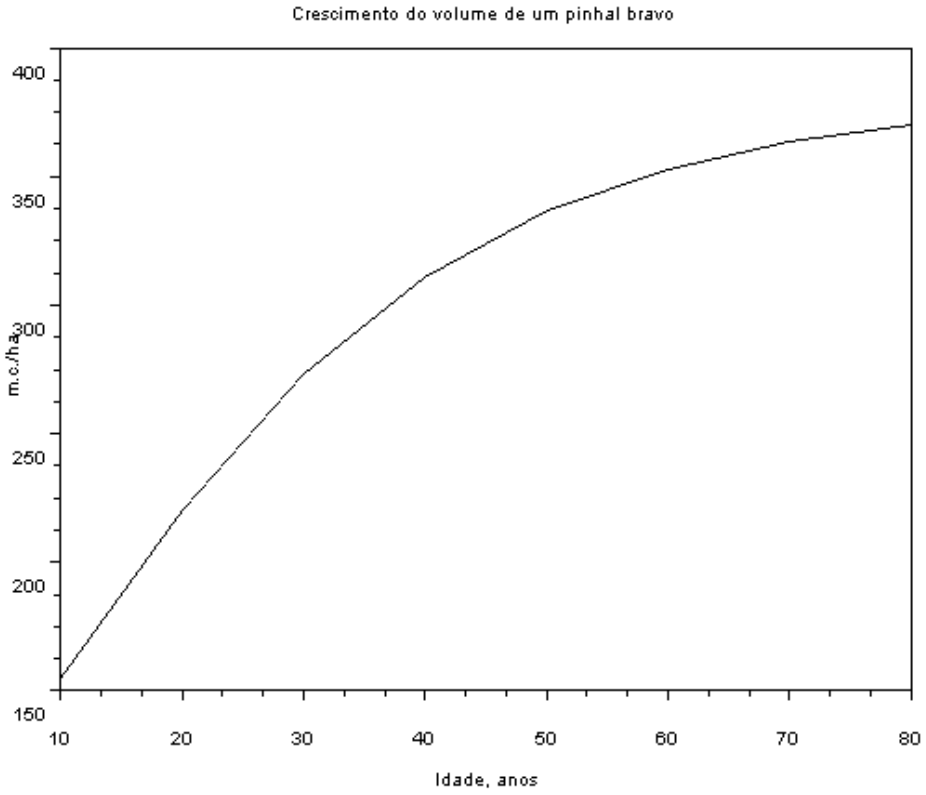


Figura 2.6. Simulação do crescimento do volume em pé de um pinhal bravo

Exemplo da aplicação de eval.

```
-->Z=['2*x^2+5*y^3+x*y+30','2*x^2+5*y^3+x*y+40'];

-->x=2;y=5;

-->[h]=eval(Z)
 h  =

   673.    683.
```

2.7 Criar uma função simples

Vamos aproveitar o material da secção 2.3 (análise combinatória) combinatória que nos vai permitir obter os arranjos e combinações de n r a r.

Para isto temos que usar o comando function que tem a seguinte estrutura:

```
function [argumentos de saída]= nome da função(argumentos de entrada)
comandos para execução
endfunction
```

Abra o SciPad escreva as seguintes instruções (ver secção 2.3)

```
function [c]=combi(n,r)
f1=prod(1:n);n2=n-r;
f2=prod(1:n2);f3=prod(1:r);
a=f1/f2;
c=a/f3;
disp([“Permutaçõe:” string(f1)])
disp(["Arranjos:” string(a)] )
disp([ “Combinações:” string(c)])
endfunction
```

Salve esta função, com o nome combi.sci, usando o menu **File** do SciPad, numa disquete.

Volte à janela de comandos, clicando nela como é usual no Windows. Use o menu **File→Exec**, abra a disquete e clique duas vezes no ícon do ficheiro. Na janela de comandos surgirá o seguinte:

```
exec('A:\combi.sci');disp('exec done');
```

Agora entre o comando:

```
-->combi(5,2)
```

A saída do Scilab será:

```
!Permutaçõe:  120  !

!Arranjos:  20  !

!Combinações:  10  !
 ans  =

   10.
```

A função combi criada pode ter utilidade, por exemplo, para resolver certos problemas do cálculo das probabilidades.

2.8 Achar a solução de um sistema de equações lineares

Seja o sistema de equações lineares

$0{,}5\ x_1 + 3x_2 + 1{,}5\ x_3 = 2$
$0{,}6\ x_1 + 2x_2 - 2\ x_3 = 5$
$2\ x_1 - 4x_2 + 12\ x_3 = 3$

Deste sistema pode-se extrair uma matriz 3x3, com os coeficientes das variáveis de acordo com a tabela

0,5	3	1,5
0,6	2	-2
2	-4	12

e o vector coluna

2
5
3

Vamos introduzir a matriz no Scilab, atribuindo-a à variável A, separando as linhas por ;. O vector coluna será a variável b. Os elementos dos vectores tanto podem ser separados por espaços como por vírgulas.

O comando a utilizar será X=lsq(A,b), que utiliza o método dos mínimos quadrados.

```
-->A=[0.5,3,1.5;0.6,2,-2;2,-4,12];

-->b=[2,5,3]';

-->X=lsq(A,b)
 X  =

   5.8238636
   0.0482955
 - 0.7045455
```

Vamos verificar o erro associado à solução achando a seguinte diferença:

```
-->b-(A*X)
 ans  =

  1.0D-14 *

 - 0.5773160
 - 0.1776357
 - 0.3552714
```

O erro é virtualmente zero, sendo x_1=5.8238636, x_2=0.0482955 e x_3=- 0.7045455. O vector coluna b também podia ter sido entrado com os elementos separados por ponto e vírgula, [2;3;5], em vez de transpor um vector linha.

2.9 Análise de uma função polinomial

O Scilab tem uma biblioteca denominada "Polynomial calculations", recheada de ferramentas para lidar com polinómios. Vamos tirar benefício de algumas delas, nesta secção.

Usamos o comando poly.

poly([*coeficientes do polinómio a começar na constante*],"nome da variável", "coeff")

Criemos o polinómio do quarto grau $y=3 - 2x - 12x^2 - 2x^3 + 2x^4$ com o comando

```
-->y=poly([3 -2 -12 -2 2],"x","coeff")
 y  =

                  2   3      4
   3 - 2x - 12x - 2x + 2x
```

Criemos um gráfico do função y(x), na figura 2.7, no domínio de -2 a 3,2. Para isso temos de primeiro avaliar o polinómio no domínio usando o comando horner. A sequência de comandos é a seguinte:

```
-->x=[-2:0.1:3.2};

-->k=[horner(y,x)];

-->plot2d(x,k')

-->xtitle("Função polinomial","x","y");

-->xgrid()
```

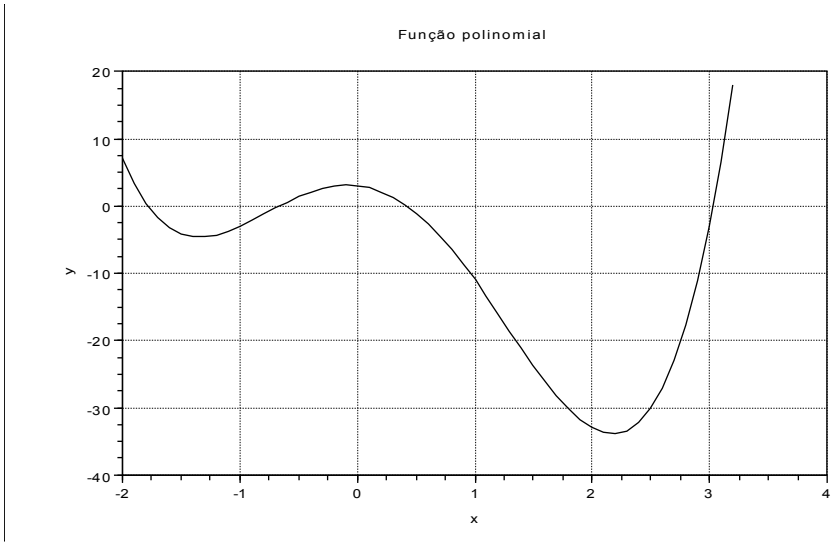


Figura 2.7. Gráfico da função polinomial definida.

Para obter as suas raízes usamos o comando roots(*y*)

```
-->zery=roots(y)
 zery  =

   0.4165815
 - 0.6652887
 - 1.7844029
   3.0331101
```

Para derivar numericamente o polinómio usamos o comando derivat.

```
-->d=derivat(y)
 d  =

                2     3
 - 2 - 24x - 6x + 8x
```

que tem as raízes

```
-->zerd=roots(d)
 zerd  =

 - 0.0853624
 - 1.3438946
   2.1792569
```

Calculamos agora o polinómio nos pontos extremos

```
-->extr=horner(y,zerd)
 extr  =

   3.0846341
 - 4.6069015
 - 33.93867
```

Vamos, de modo semelhante, na figura 2.8, traçar o gráfico da primeira derivada.

```
-->m=[horner(d,x)];

-->plot2d(x,m');

-->xtitle("Primeira derivada","x","Derivada");

-->xgrid()
```

Passemos ao cálculo da segunda derivada.

```
-->dd=derivat(d)
 dd  =
```

```
                    2
-    24 - 12x + 24x
```

Os zeros da segunda derivada dão-nos os pontos de inflexão:

```
-->inflex=roots(dd)
 inflex  =

 - 0.7807764
   1.2807764
```

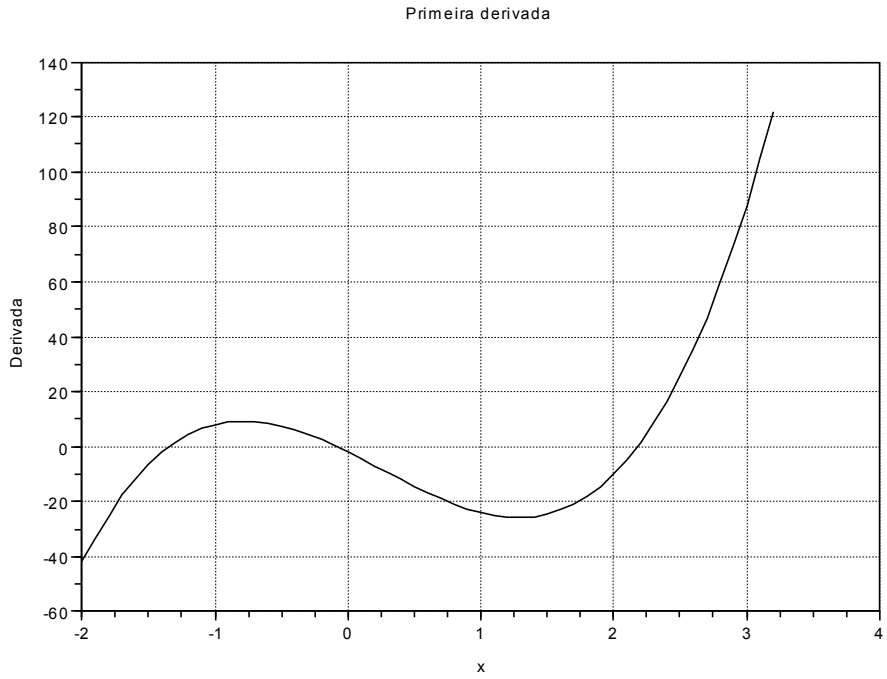


Figura 2.8. Gráfico da primeira derivada do polinómio y

Tracemos o gráfico da segunda derivada (figura 2.9).

```
-->x=[-2:0.1:3.2};

-->n=[horner(dd,x)];

-->plot2d(x,n');

-->xtitle("Segunda derivada","x","Derivada");

-->xgrid()
```

Avaliemos o polinómio nos pontos de inflexão:

```
-->x=[inflex];

-->infl=horner(y,x)
 infl  =

  - 1.0585946
  - 18.066405
```

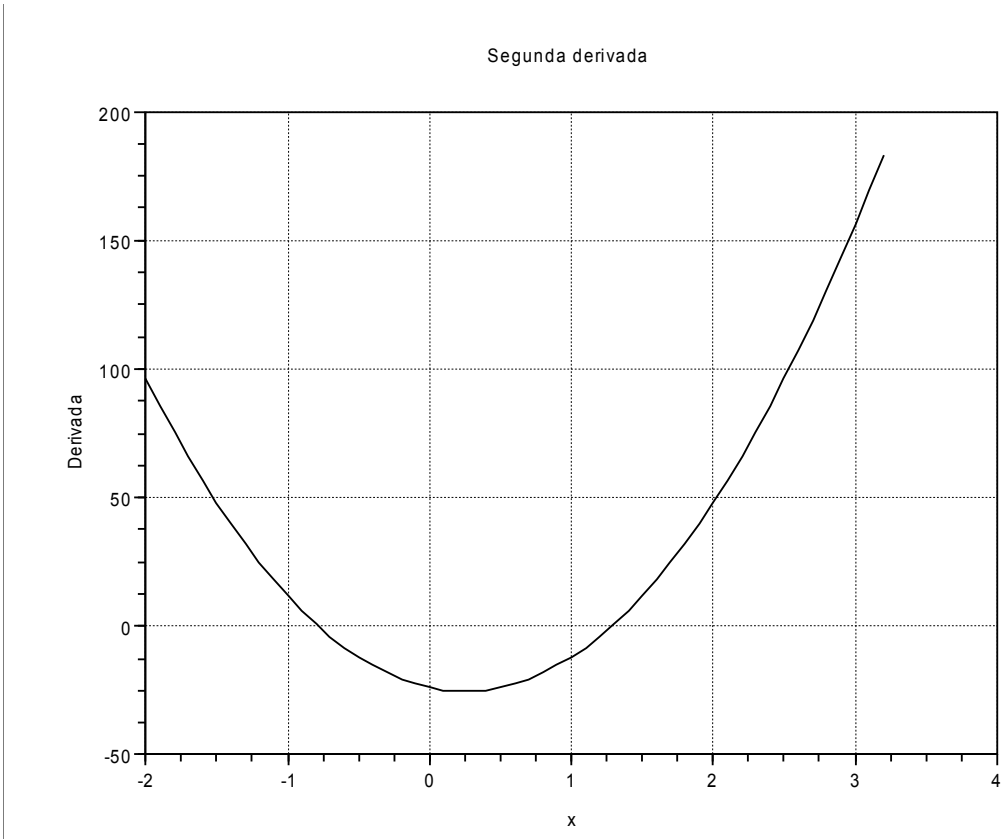


Figura 2.9. Gráfico da segunda derivada do polinómio.

2.10 Operações com polinómios

Vamos criar dois polinómios e realizar com eles operações aritméticas. A leitora já está em condições de interpretar, por si, os comandos seguintes:

```
-->a=poly([5 2 9 8 1],"x","coeff")  //criar o polinómio a
 a  =

                2    3    4
   5 + 2x + 9x + 8x + x

-->b=poly([5 -3 8],"x","coeff")  //criar o polinómio b
 b  =
```

```
                2
   5 - 3x + 8x
```

```
-->soma=sum(a,b) //somar os dois polinómios
 soma  =

                2      3    4
   5 + 2x + 9x + 8x + x
```

```
-->sub=a-b //subtrair b de a
 sub  =

          2    3   4
   5x + x + 8x + x
```

```
-->mult=a*b  //multiplicar os polinómios
 mult  =

                2       3      4       5     6
   25 - 5x + 79x + 29x + 53x + 61x + 8x
```

Para dividir os polinómios usamos o comando pdiv.

```
-->div=pdiv(a,b)  //dividir os dois polinómios
 div  =

                                        2
   1.4394531 + 1.046875x + 0.125x
```

```
-->poten=a^3  //potência de a
 poten  =

                        2          3          4          5          6          7          8
   125 + 150x + 735x + 1148x + 1878x + 2802x + 2835x + 2676x + 2082x
                 9        10      11  12
         + 950x + 219x + 24x + x
```

Por último criemos a fracção z=a/b, e façamos o seu gráfico no domínio -5 a 5, na figura 2.9.

```
-->z=a/b
 z  =
```

```
                2     3    4
   5 + 2x + 9x + 8x + x
   -------------------
                 2
      5 - 3x + 8x
```

Antes de grafarmos z, ele tem de ser calculado com o comando horner.

```
-->x=[-5:0.01:5];

-->plot2d(x,[horner(z,x)])

-->xtitle("Gráfico duma fracção polinomial","x","z")
```

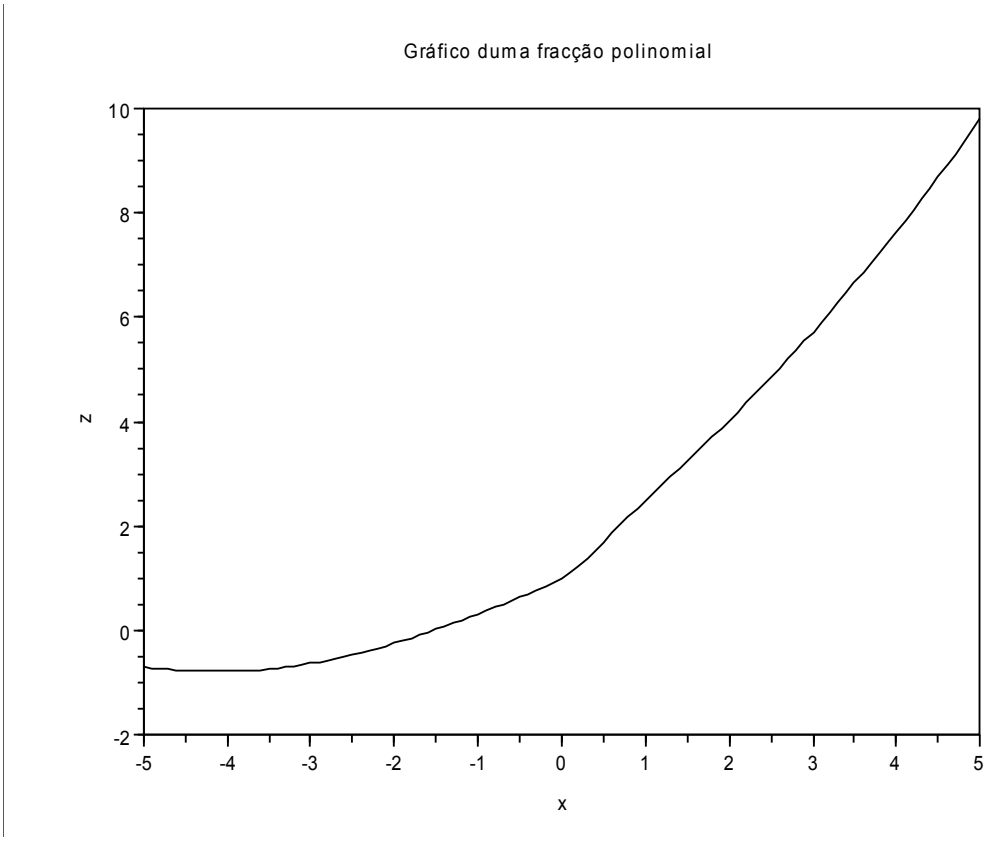


Figura 2.10. Gráfico da fracção polinomial z.

Também podemos obter um polinómio com raízes pré definidas:

```
-->poly([1 2], 'x')
 ans  =

          2
   2 - 3x + x


-->roots(ans)
```

```
 ans  =

    1.
    2.
```

2.11 Limites

Nesta secção, apresentarei uma breve incursão sobre o tópico limites de funções.

A função

$$f(x) = \frac{seno(x)}{x}$$

é uma indeterminação no ponto x=0. Recorrendo à regra de l'Hopital, mostra-se que o seu valor quando x tende para zero é 1.

Numericamente, podemos aproximar o valor deste limite, com os comandos seguintes:

```
-->x=[-.001 -.0001 -.00001 -.000001 -.0000001 0.0000001 0.000001 0.00001
0.0001 0.001];

-->limite=[sin(x)/x]
 limite  =

    0.9999998
```

Por sua vez, a função racional

$$f(x) = \frac{x^2 - 4}{x - 2}$$

redunda numa indeterminação para x=2. Apliquemos o mesmo procedimento:

```
-->x=[1.99 1.999 1.9999 1.99999 1.999999 2.000001 2.00001 2.0001 2.001
2.01];

-->[limite]=[(x^2-4)/(x-2)]
 limite  =

    4.
```

Usando um procedimento que introduziremos adiante, podemos escrever os valores da função para o domínio considrado (vector x) como um vector coluna, e fazer os seu gráfico, na figura 2.11.

```
-->disp([lim])

    3.99
    3.999
```

3.9999
3.99999
3.999999
4.000001
4.00001
4.0001
4.001
4.01

2.12 Estatística elementar

Criemos um vector linha com 5000 elementos aleatórios, extraídos de uma população de valores com distribuição normal de média 2 e desvio padrão 3. O comando que o permite isto é grand:

Y=grand(*dimensoões da matriz de valores pretendida, 'nor', média, desvio padrão*)

-->dados=grand(1,5000,"nor",2,3);

O gráfico da nossa amostra pode ser obtido como se ilustra na figura 2.12. Este gráfico não é muito informativo, adiante inseriremos outro.

-->plot(dados)

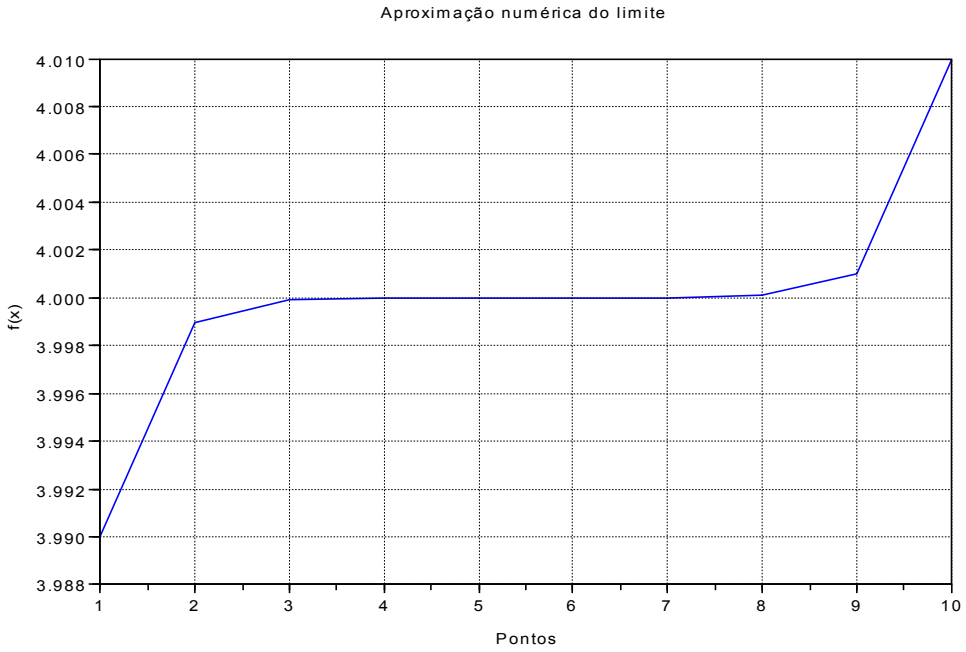


Figura 2.11. Aproximação numérica do limite da função $f(x)=(x^2-4)/(x-2)$

Vamos ordenar a nossa amostra casual por ordem crescente, com o comando gsort.

[x]=gsort(*dados,'g','i'*);

Obtenhamos um vector com os elementos mínimo, mediana, média e desvio padrão, com os comandos min(*x*), median(*x*), mean(*x*), max(*x*),st_deviation(*x*).

```
-->anal=[min(x), median(x), mean(x), max(x),st_deviation(x)]
 anal  =

 - 10.220008    1.9318621    1.957948    12.118696    2.9556666
```

A média (1,957948) e o desvio padrão (2,9556666) são virtualmente os teóricos (2 e 3).

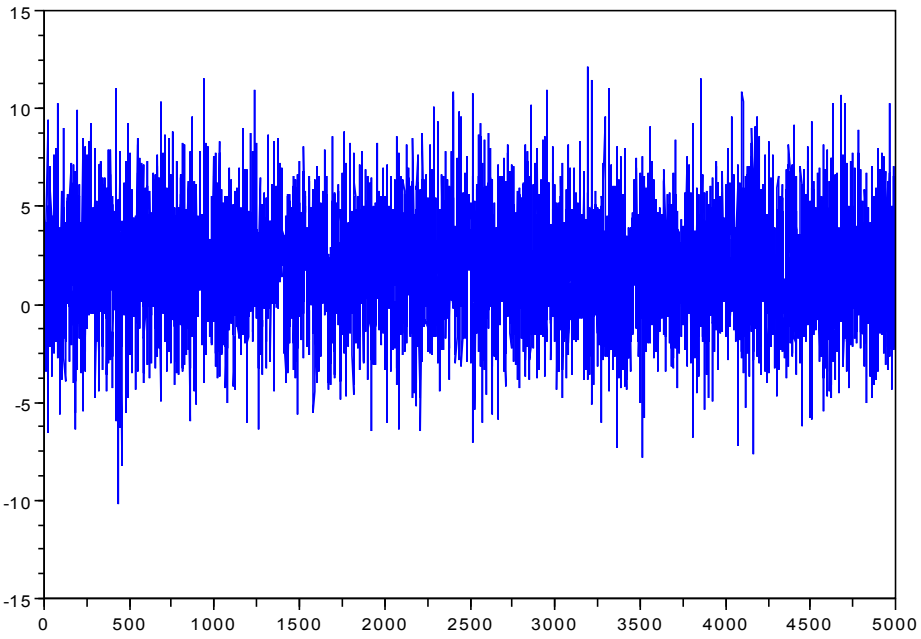


Figura 2.12. Gráfico da amostra dados.

A sequência de comandos seguinte gera uma amostra casual semelhante à anterior, inscreve um histograma (comando histplot) da amostra e sobrepõe-lhe a curva normal correspondente, como se verifica na figura 2.12, mais informativa que a figura anterior.

histplot(*valor mínimo: intervalo : valor máximo*)

```
c=grand(1,5000,"nor",2,3); //amostra
miu=mean(c);st=(st_deviation(c)); //média e desvio padrão
var=st^2; //variância
x=-20:1:20; //domínio
nor=1/(sqrt(var*2*%pi))*exp(-(x-miu)^2/(2*var));//curva normal
plot2d(x,[nor])
xtitle("Comparação normal-amostra casual","Valores","Frequência")
```

```
histplot([-20:1:20],c)  //histograma
```

Copie os comandos anteriores e cole-os na janela de comandos do Scilab e obterá o gráfico da figura 2.13. Alternativamente, cole-o no SciPad, e salve o ficheiro com o nome norcomp.sci, numa disquete (ou outro suporte), e pode passar a ccorre-lo com o comando **Exec,** como já se viu anteriormente (secção 2.7).

.

2.13 Probabilidades elementares

Comecemos pela distribuição binomial.

O comando binomial permite obter as probabilidades desta distribuição, conhecidos n e p. Um exemplo, com n=12 e p=0,2. O resultado é apresentado sob a forma de um gráfico de linhas verticais criado pelo comando plot2d3. Vamos inserir o comando

```
Binomial(p,n)
```

no comando plot2d3, para obter o gráfico da figura 2.14.

```
-->plot2d3(0:12,binomial(0.2,12))
```

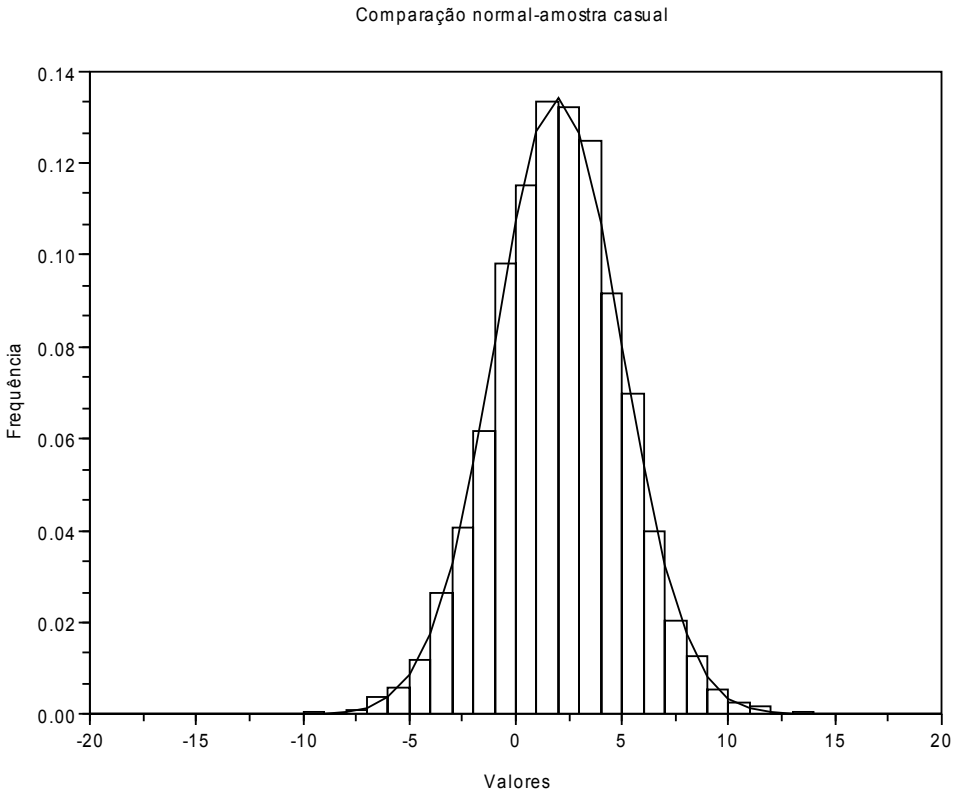


Figura 2.13. Comparação entre o histograma de uma amostra de distribuição normal e a curva normal associada

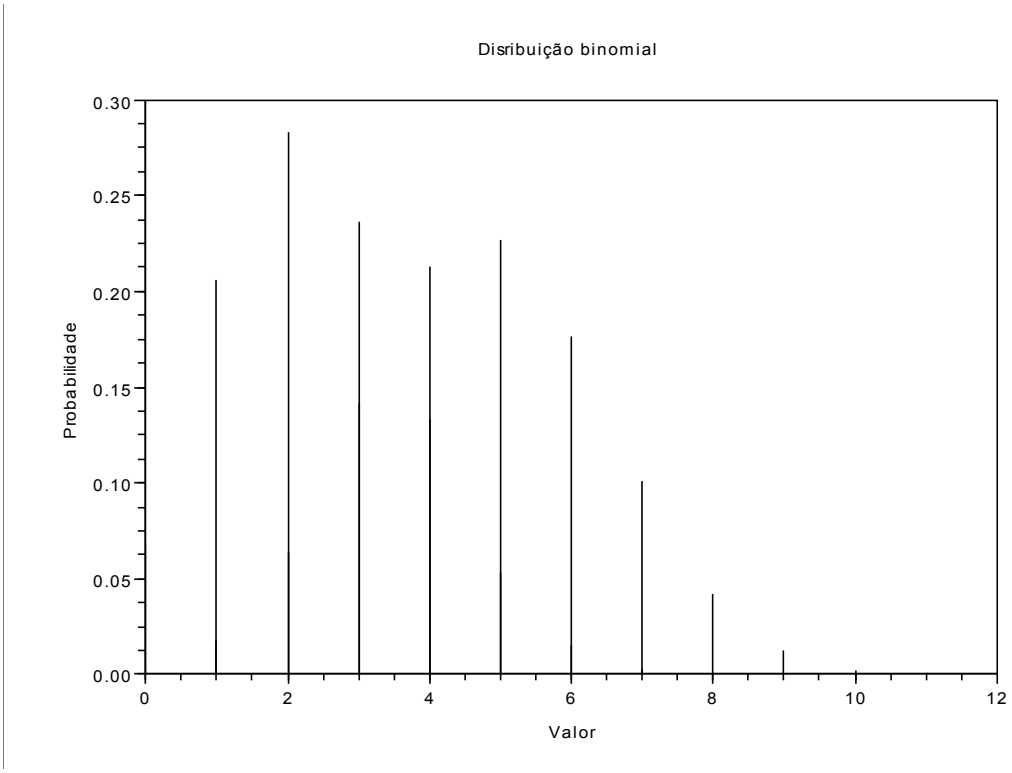


Figura 2.14. Gráfico da distribuição discreta binomial, n=12, p=0,2.

Na inspecção de um produto, são tiradas amostras de 10, que são classificados em aceites ou rejeitados. Verificou-se que 10% dos objectos são rejeitados. Qual é a probabilidade de não mais do que um seja rejeitado?

Temos n=10, p=0.1, S=1..

Vamos recorrer ao comando cdfbin .

```
[P,Q]=cdfbin("PQ", S, n ,p, 1-p)
```

```
-->[P,Q]=cdfbin("PQ",1,10,0.1,0.9)
 Q  =

   0.2639011
 P  =

   0.7360989
```

A probabilidade pedida é P=0.7360989.

O comando cdfbin tem outras alternativas de utilização que pode verificar inserindo help cdfbin

Agora a distribuição normal.

Vamos aqui apresentar uma função que pede que se insira no **prompt** f(*média, desvio padrão, valor da variável*), e fornece a probabilidade do valor da variável e probabilidade de ocorrência de valores iguais ou inferiores à variável (probabilidade acumulada). Recorremos ao comando cdfnor.

```
//distribuição normal
disp("f(média, desvio padrão, valor da variável)")
function [Pr]=f(m,dp,x)
var=string(x);// transformar a variável numa "string"
Pr=(1/(sqrt(2*%pi)*dp))*exp(-((x-m)^2)/(2*dp));//calcular a probabilidade
pr=string(Pr); //transforma a probabilidade numa "string"
disp(["Probabilidade de", var, "ocorrer:", pr]) // output do resultado
[P,Q]=cdfnor("PQ",x,m,dp);ac=string(P); //calcular a prob. acumulada
disp(["probabilidade acumulada até", var, ":", ac]) //saída do resultado
endfunction
```

Esta função foi salva para uma disquete com o nome Prcdfnor.sci e depois executada (**File→Exec**), conforme se ilustra abaixo. O valor da variável é zero, na distribuição normal com média zero e variância um (N(0,1).)

```
-->exec('A:\Prcdfnor.sci');disp('exec done');

 f(média, desvio padrão, valor da variável)

 exec done

-->f(0,1,0)

!Probabilidade de  0  ocorrer:  0.3989423  !

!probabilidade acumulada até  0  :  0.5  !
 ans  =

   0.3989423
```

Semelhantemente ao comando cdfbin, o comando cdfnor tem outras alternativas de utilização que pode escrutinar inserindo help cdfnor

Como dissemos no capítulo 1, o Scilab trata os escalares, reais ou complexos, como matrizes 1x1. Dizendo de forma enfática, para o Scilab tudo são matrizes. Por isso, não admira que muitas das funções aplicáveis a escalares também o sejam para matrizes, incidindo sobre os seus elementos, um a um, fornecendo assim matrizes de resultados das mesmas dimensões que as iniciais. Isto já foi ilustrado com os comandos log, sign e abs. Podem ainda aplicar-se a matrizes os comandos exp, sqrt, int, real, imag, conj, floor, ceil, round, modulo e as funções trigonométricas.

O leitor deve criar matrizes de reais e complexos, e ensaiar a aplicação dos comandos citados sobre elas.

2.14 Criar uma função "on-line"

É possível criar funções no programa ou sequência de comandos que se esteja a criar, ditas "on-line" que podem ser facilmente chamadas. Para isto, usa-se o comando deff:

deff('[*variáveis de saída*]=nome da função(*variáveis de ntrada.*)',['expressões a calcular'])

Um exemplo para calcular funções trigonométricas:

```
-->deff('[x1,x2,x3]=trig(a,b,c)',['x1=sin(a)';'x2=cos(b);x3=tan(c)'])

-->[x1,x2,x3]=trig(1,1.5,2)
 x3  =

  - 2.1850399
 x2  =

    0.0707372
 x1  =

    0.8414710
-->x1,x2
 x1  =

    0.1411200
 x2  =

    0.2836622

-->x3
 x3  =

  - 1.5574077

-->a=3;b=5;c=-1;

-->[x1,x2,x3]=trig(a,b,c)
 x3  =

  - 1.5574077
 x2  =

    0.2836622
 x1  =

    0.1411200
```

Estas funções podem ser guardadas com ficheiros *.sci, e chamadas com **Load**, ou **Exec,** do menu **File**. As funções "on-line" são importantes por abrirem a possibilidade de utilização de outros comandos, como veremos adiante.

Depois de apresentarmos o quadro sinóptico dos comandos introduzidos neste capítulo, vamos aprofundar o nosso conhecimento sobre a construção e manipulação de vectores e matrizes.

2.15 Quadro sinóptico dos comandos introduzidos neste capítulo

Comando	Descrição
[ ]	Enquadra os elementos de um vector ou matriz. Z=[] cria uma matriz vazia onde mais tarde se pode inserir os elementos
‘	A seguir a uma matriz fornece a sua transposta
==	Igualdade de comparação
abs	Valor absoluto ou o módulo de um número complexo
cdfbin	Função da probabilidade acumulada de uma distribuição binomial
cdfnor	Função da probabilidade acumulada de uma distribuição normal
ceil	Arredonda para o inteiro superior
conj	Fornece o conjugado de um número complexo
deff	Cria funções “on-line”
derivat	Avaliação numérica da derivada
diff	Dá a diferença entre elementos seguidos de um vector e a derivada numérica de uma função
disp	Exibe variáveis
eval	Avalia uma matriz de “strings”
evstr	Avalia uma variável que esteja sob a forma de texto
floor	Arredonda para o inteiro inferior
grand	Gera uma amostra de números casuais com uma distribuição definida
gsort	Ordenar por ordem crescente
histplot	Cria um histograma
horner	Avalia um polinómio para um valor da sua variável
imag	Permite obter a parte imaginária de um número complexo
imult	Multiplica um número por i
int	Extrai a parte inteira de um número
int	Retorna aparte inteira de um número
linspace	Cria um vector de valores igualmente espaçados
lsq	Aplica o método dos mínimos quadrados
max	Máximo
mean	Média de uma amostra
median	Mediana de uma amostra
modulo	Dá o resto da divisão de um número por outro
pdiv	Divisão de polinómios
plot	Obtenção de um gráfico simples
plot2d	Permite traçar um gráfico no espaço a duas dimensões
poly	Define um polinómio
prod	Produto
real	Permite obter a parte real de um número complexo
roots	Calcula as raízes de um polinómio
round	Arredonda um número para o inteiro mais próximo
sign	Sinal. Os números positivos são assinalados com 1, e os negativos com -1.
size	Fornece as dimensões de um vector ou matriz
st_deviation	Desvio padrão
string	Transforma uma variável numa cadeia de texto ou “string”
sum	Soma dos elementos de um vector ou duma matriz
xbasc	Apaga o conteúdo numa janela de gráfico
xgrid()	Insere uma grelha num gráfico
xtitle	Insere o título e as legendas dos eixos num gráfico

Capítulo 3

Vectores e matrizes

Grande parte da capacidade e poder do Scilab vem da sua elevada aptidão para lidar com matrizes. O Scilab é um software, em grande extensão, baseado em matrizes, por isso justifica-se que nos ocupemos deste tópico num capítulo específico. Nele, vamos abordar os seguintes temas:

- Criar vectores e matrizes
- Conhecer matrizes especiais que o Scilab cria facilmente
- Identificar elementos de vectores e matrizes
- Manipular matrizes, como acrescentar e eliminar elementos
- Realizar operações com vectores e matrizes
- Analisar matrizes
- Resolver sistemas de equações lineares no contexto da álgebra matricial

O Scilab têm um elenco rico de comandos para lidar e criar matrizes. Recomendamos à leitora que os veja com o comando `apropos matrix`.

3.1 Recapitular a criação de vectores e matrizes

3.1.1 Vectores

Como já vimos um vector pode ser criado de diversas maneiras. Aqui se apresentam.

Todos os seguintes comandos produzem o mesmo vector linha:

```
-->a=[1 2 3 4];

-->a=[1, 2, 3, 4];

-->a=[1:4];

-->a=matrix(1:4,1,4);

-->a=matrix([1 2 3 4],1,4)
 a  =

   1.    2.    3.    4.
```

Todos os seguintes comandos criam o mesmo vector coluna:

```
-->acol=a';

-->acol=[1:4]';

-->acol=[1; 2; 3; 4];

-->acol=matrix(1:4,4,1);
.
-->acol=matrix([1 2 3 4],4,1)
 acol  =

   1.
   2.
   3.
   4.
```

Introduzimos na secção 2.4 o comando linspace para criar um vector de números igualmente espaçados num dado intervalo. O comando logspace permite obter um vector de números correspondentes aos logaritmos na base 10 do intervalo especificado, igualmente espaçados. Um exemplo:

```
-->logspace(2,3,5)
 ans  =

   100.    177.82794    316.22777    562.34133    1000.
```

3.1.2 Matrizes

Eis várias maneiras de criar uma matriz:

```
-->amat=[1 2 3; 4 5 6; 7 8 9];

-->amat=[1, 2, 3; 4, 5, 6; 7, 8, 9];

-->amat=matrix([1, 2, 3, 4, 5, 6, 7, 8, 9],3,3);

-->amat=matrix([1:9],3,3)
 amat  =

   1.   4.   7.
   2.   5.   8.
   3.   6.   9.
```

O comando matrix ordena os valores coluna a coluna, se os quisermos ordenados por linhas pedimos a transposta:

```
-->mat=matrix([1:9],3,3)'
 mat  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   9.
```

Se a matriz for muito longa, passa-se para a linha seguinte como habitualmente acabando a linha com três pontos e continuando na seguinte.

3.2 Criar matrizes especiais

3.2.1 Com 1's na diagonal principal

Seja uma matriz 3x3, com 1's na diagonal principal:

```
-->diag1=eye(3,3)
 diag1  =

   1.   0.   0.
   0.   1.   0.
   0.   0.   1.
```

3.2.1 De zeros

Par obter uma matriz 3x3 de elementos todos iguais a zero:

```
-->sozeros=zeros(3,3)
 sozeros  =

   0.   0.   0.
   0.   0.   0.
   0.   0.   0.
```

3.2.1 De números casuais
Este tipo de matriz já foi ilustrado na sub-secção 1.5.5.

```
-->casu=rand(3,3)
 casu  =

   0.2113249   0.3303271   0.8497452
   0.7560439   0.6653811   0.6857310
   0.0002211   0.6283918   0.8782165
```

3.2.1 Com um vector na diagonal
Desejamos criar uma matriz com o vector a, atrás definido, na diagonal:

```
-->diag(a)
 ans  =

   1.   0.   0.   0.
   0.   2.   0.   0.
   0.   0.   3.   0.
   0.   0.   0.   4.
```

3.2.1 Um vector com a diagonal de uma matriz
Vamos utilizar a matriz amat, atrás criada:

```
-->diag(amat)
 ans  =

   1.
   5.
   9.
```

3.2.1 Um vector com a primeira diagonal de uma matriz acima da principal

```
-->diag(amat,1)
 ans  =

   4.
   8.
```

3.3 Identificar os elementos de um vector ou matriz

O procedimento para identificar elementos de um vector ou matriz é o seguinte:

nome do vector ou matriz(*número da linha*, *número da coluna*)

Por exemplo o elemento da linha 2 e coluna 3 da matriz amat:

```
-->amat(2,3)
```

```
 ans  =

    8.
```

Para extrair os elementos de uma linha de uma matriz usa-se o comando:

Nome da matriz(*número da linha*, *primeira coluna:última coluna*)

Seja o caso da segunda linha da matriz amat:

```
-->linha2=amat(2,1:3)
 linha2  =

    2.    5.    8.
```

Para extrair os elementos de uma coluna de uma matriz usa-se o comando

Nome da matriz(*primeira linha:última linha*, *número da coluna*)

Para extrair a primeira coluna de amat escrevemos:

```
-->coluna1=amat(1:3,1)
 coluna1  =

    1.
    2.
    3.
```

No caso do vector linha a, para obter o terceiro elemento, tanto podemos escrever

```
-->a(1,3);
```

como

```
-->a(3)
 ans  =

    3.
```

Analogamente para o vector coluna acol:

```
-->acol(3,1);

-->acol(3)
 ans  =

    3.
```

3.4 Manipular vectores e matrizes

Ocupemo-nos da alteração de vectores e matrizes existentes.

3.4.1 Vectores

Desejamos acrescentar um elemento a um vector linha existente:

```
-->c4=[2 4 6 8];


-->ce=[5];

-->c5=[c4 ce]
 c5  =

   2.   4.   6.   8.   5.
```

Eliminar um elemento a um vector linha

```
->c4(:,2)=[]
 c4  =

   2.   6.   8.
```

Repetir o mesmo procedimento com um vector coluna:

```
-->v1=[5 6 7]';

-->v2=[8];

-->v4=[v1;v2]
 v4  =

   5.
   6.
   7.
   8.


v4(2,:)=[]
 v4  =

   5.
   7.
   8.


v4(2:3,:)=[]
```

```
 v4  =

    5.
```

3.4.2 Matrizes

Vejamos como adicionar uma linha a uma matriz.

```
-->A=matrix([1:12],4,3)
 A  =

    1.    5.    9.
    2.    6.    10.
    3.    7.    11.
    4.    8.    12.

-->linha=[13:15];

-->A=[A;linha]
 A  =

    1.     5.     9.
    2.     6.     10.
    3.     7.     11.
    4.     8.     12.
    13.    14.    15.
```

Adicinar uma coluna:

```
-->colun=[16:20]';

-->A=[A,colun]
 A  =

    1.     5.     9.     16.
    2.     6.     10.    17.
    3.     7.     11.    18.
    4.     8.     12.    19.
    13.    14.    15.    20.
```

Eliminar a 3 linha da matriz A.

```
-->A(3,:)=[]
 A  =

    1.     5.     9.     16.
    2.     6.     10.    17.
    4.     8.     12.    19.
    13.    14.    15.    20.
```

Eliminar as colunas de 1 e 2 da matriz a:

```
-->A(:,1:2)=[]
 A  =

    9.    16.
    10.   17.
    12.   19.
    15.   20.
```

Eliminar a segunda e quarta linhas da matriz A:

```
-->A([2 4],:)=[]
 A  =

    9.    16.
    12.   19.
```

3.5 Operações com vectores e matrizes

3.5.1 Vectores

Sejam dois vectores linha com as mesmas dimensões, então é possível realizar as seguintes operações com eles:

Adição

```
-->a=[13.   14.   15.   20.];

-->b=[2.    6.    10.   17.];


-->a+b
 ans  =

    15.   20.   25.   37.
```

Subtração

```
-->a-b
 ans  =

    11.   8.   5.   3.
```

Multiplicação elemento a elemento:

```
-->a.*b
 ans  =

    26.   84.   150.   340.
```

Divisão elemento a elemento:

```
-->a./b
 ans  =

   6.5    2.3333333    1.5    1.1764706
```

Repare no ponto antes do sinal de multiplicar e dividir.
Do mesmo modo posso calcular uma potência de um vector:

```
-->b.^a
 ans  =

  1.0D+13 *

   8.192D-10    0.0078364    100.    4.064D+11
```

Por fim o produto escalar de dois vectores:

```
-->a*b'
 ans  =

   600.
```

O Scilab executa a multiplicação de um vector por um escalar. Ilustremos

```
-->a=[1:5];

-->b=3*a
 b  =

   3.    6.    9.    12.    15.

-->c=a';

-->d=2*c
 d  =

   2.
   4.
   6.
   8.
   10.
```

3.5.2 Matrizes

Se duas matrizes forem do mesmo tamanho podem ser adicionadas e uma subtraída da outra.

```
-->M=matrix([1:9],3,3);

-->N=matrix([10:18],3,3);

-->P=M+N
 P  =

   11.   17.   23.
   13.   19.   25.
   15.   21.   27.

-->Q=M-N
 Q  =

 - 9.  - 9.  - 9.
 - 9.  - 9.  - 9.
 - 9.  - 9.  - 9.
```

Se a matriz A tiver um número de linhas igual ao número de colunas de B, então a operação A*B é possível.

```
-->A=matrix([2:2:24],3,4);

-->B=matrix([1:2:23],4,3);

-->C=A*B
 C  =

   236.   588.   940.
   268.   684.   1100.
   300.   780.   1260.
```

Se a matriz E e F forem quadradas e tiverem a mesma dimensão a operação A/B é possível e é igual ao produto de E pela inversa de F.

Aproveitamos para introduzir um comando para gerar números aleatórios com distribuição uniforme entre limites por nós escolhidos, e não entre zero e um como o comando rand.

Y=grand(*dimensoões da matriz de valores pretendida, "unf", valor mais baixo, valor máximo*)

```
-->E=grand(3,3,"unf",0,2)
 E  =

   1.6294474   1.6700172   1.8267517
   0.2709540   0.2539736   0.4420681
   1.8115839   1.9377355   1.2647185


-->F=grand(3,3,"unf",0,2)
```

```
 F  =

   0.6163341    0.5569964    1.9857626
   0.1950808    0.3767640    1.9150137
   1.0944412    1.093763     1.9929227

-->G=E/F
 G  =

  - 1.0933805  - 0.1258587    2.1270103
    0.0678591  - 0.0704704    0.2219193
  - 2.28301    - 0.0402763    2.9481145
```

Como a matriz E é quadrada podemos calcular E^2 que é igual a E*E.

```
-->E^2
 ans  =

   6.4169106    6.6851072    6.0251838
   1.3111639    1.3736115    1.166331
   5.7680614    5.9681999    5.7654379
```

Os símbolos relacionais podem ser usados para comparar os elementos de duas matrizes. Um exemplo:

```
-->E<F
 ans  =

  F F T
  F T T
  F F T
```

Como destacámos no final do capítulo 2, muitos dos comandos que se aplicam a escalares pode ser aplicados a matrizes. Um exemplo:

```
-->sin(E)
 ans  =

   0.9982805    0.9950816    0.9674219
   0.2676508    0.2512521    0.4278097
   0.9711505    0.9334298    0.9535227
```

Convém introduzir uma anotação sobre os comandos exp, log e sqrt. Como dissemos incidem individualmente sobe cada elemento da matriz, com sobre um escalar isolado e os comandos expm, logm, sqrm.

O comando expm(E) calcula e^E.

```
-->exp(E)
 ans  =
```

```
   5.101055    5.3122591   6.21367
   1.3112148   1.2891378   1.5559217
   6.1201332   6.943011    3.5420955

-->expm(E)
 ans  =

   18.790997   18.475466   17.271767
   3.5145105   4.6300492   3.5139609
   16.655774   17.355739   16.601059
```

O comando logm(E) calcula log(E) tal que $E=e^{\log(E)}$.

```
-->log(E)
 ans  =

   0.4882409   0.5128339   0.6025394
 - 1.3058062 - 1.3705249 - 0.8162914
   0.5942015   0.6615200   0.2348496

-->logm(E)
 ans  =

 - 1.0360764 + 1.6402894i   4.4605244 - 1.5605189i   1.5593685 - 1.439862i
   1.3078812 - 0.3001921i - 3.51656 + 2.8295601i   - 0.3749895 - 0.2879067i
   1.0123651 - 1.3849357i   0.5631634 - 1.4395615i   0.0748286 + 1.8133358i
```

sqrm(E) calcula $\sqrt{E}$.

```
-->sqrt(E)
 ans  =

   1.2764981   1.2922914   1.3515738
   0.5205324   0.5039580   0.6648820
   1.3459509   1.3920257   1.124597

-->sqrtm(E)
 ans  =

   0.9128043 + 0.1831078i   0.9488079 + 0.1464902i   0.8754476 - 0.2302457i
   0.1825192 + 0.0924804i   0.1897183 + 0.2028180i   0.1750496 - 0.1442129i
   0.8420519 - 0.2911514i   0.8752648 - 0.3725548i   0.8075907 + 0.3963682i
```

3.6 Analisar matrizes

Vamos calcular o determinante, a inversa, o traço, os valores e vectores próprios da matriz E.

Para o cálculo do determinante usamos o comando det.

```
-->det(E)
 ans  =

   0.0113583
```

A inversa de E obtém-se com o comando inv.

```
-->inv(E)
 ans  =



 - 47.137912    125.69327    24.151051
   40.337323  - 109.92166  - 19.841108
   5.7176488  - 11.626924  - 3.4037874
```

Para o traço de E usa-se trace.

```
-->trace(E)
 ans  =

   3.1481395
```

O comando para obter os vectores e valores próprios de E escreve-se [Ab,X,bs]=bdiag(E, 1/%eps).A diagonal de Ab dá-nos os valores próprios. A matriz X é a dos vectores próprios.

```
 -->[Ab,X,bs]=bdiag(E,1/%eps);

-->disp("Valores próprios da matriz=")

 Valores próprios da matriz=

-->disp(diag(Ab))

   3.6485329
 - 0.4940928
 - 0.0063007

-->disp("Vectores próprios da matriz=")

 Vectores próprios da matriz=

-->X
 X  =

   0.7051855  - 0.3996759  - 0.5164079
   0.1410049  - 0.2823142    0.4483045
```

```
   0.6505258    0.7227022    0.0525726
```

O comando spec também permite obter os valores próprios de uma matriz:

```
-->vpropo=spec(E)
 vpropo  =

   3.6485329
 - 0.0063007
 - 0.4940928
```

Podemos também obter o polinómio característico da matriz com o comando poly;

```
-->car=poly(E,"x")
 car  =

                                          2    3
 - 0.0113583 - 1.8225889x - 3.1481395x + x
```

Aplicando o comando roots, introduzido no capítulo 2, obtemos novamente os valores próprios:

```
-->roots(car)
 ans  =

 - 0.0063007
 - 0.4940928
   3.6485329
```

3.7 Novamente os sistemas de equações lineares

Vamos abordar a solução de sistemas de equações lineares no contexto da álgebra linear. Sugiro que recapitule a secção 2.8. Suponhamos um sistema com a matriz A e o vector b abaixo introduzidos no Scilab.

```
-->A=[0.5,3,1.5;0.6,2,-2;2,-4,12];

-->b=[2,5,3]';
```

Vamos recorrer ao comando linsolve que resolve o sistema A*x+b=0.

```
-->[x,kerA]=linsolve(A,b)
 kerA  =

    []
 x  =

 - 5.8238636
```

```
 - 0.0482955
   0.7045455
```

Verifiquemos a precisão da solução.

```
-->b+A*x
 ans  =

   1.0D-14 *

    0.0222045
  - 0.6217249
  - 0.1776357
```

O erro é negligenciável. Também é possível usar o sinal de divisão à direita (\) para resolver o sistema. A equação resolvida é Ax=b, por isso os sinais da solução não são os mesmos.

```
-->xd=A\b
 xd  =

    5.8238636
    0.0482955
  - 0.7045455
```

Se efectuarmos o produto A*xd obtemos b:

```
-->A*xd
 ans  =

    2.
    5.
    3.
```

A matriz inversa de A também pode ser aqui utilizada:

```
-->inv(A)*b
 ans  =

    5.8238636
    0.0482955
  - 0.7045455
```

O comando \ é preferível a inv(A)*b pois é mais rápido e numericamente mais estável, o que no caso de grandes sistemas de equações pode ser importante.

Consulte a ajuda do comando linsolve (help linsolve).

3.8 Outra aplicação da álgebra linear

A utilização de vectores e matrizes permite instruções compactas e evitar o uso de ciclos ("loops") o que redunda em programas mais simples e de execução mais rápida. Vou ilustrar esta asserção com o ajustamento da equação de uma recta.

Por outras palavras, dispondo de um vector dos valores da variável y, linearmente dependente dos elementos do vector x, deseja-se estimar os parâmetros a e b na sua relação y=a+ b x, e avaliar a qualidade do ajustamento. Esta tarefa é realizada pelos seguintes comandos:

```
// Ajusta a regressão linear y=a+b*x
//Para a geométrica usar x= log(x) e y=log(y);
// Para a exponecial fazer y=log(y)
disp("y=a+b*x");
//dados
x=[1 2 3 4 5 6];
y=[3.2 5.4 8 9.1 11.5 13.8];
n=6;
//Cálculos intermédios
j=sum(x);k=sum(y);l=sum(x^2);
m=sum(y^2);rx=sum(x.*y);
//cálculos de a e b
b = (n * rx - k * j) / (n * l - j ^ 2);
a = (k - b * j) / n;
//Preparar as saídas de a e b
a1=string(a);b1=string(b);
//exibir os valores de a e b
disp(["a=" a1 "b=" b1])
//Estatísticos de avaliação do ajustamento
j = b * (rx - j * k / n); m = m - k ^ 2 / n; k = m - j; r2 = j / m;
rr=string(r2);co=string(sqrt(r2));err=string(sqrt(k/(n-2)));
disp(["R quadrado= ", rr])
RSS=m-j;R2aj=1-((RSS/(n-2))/(m/(n-1)));
disp(["R quadrado ajustado:", string(R2aj)])
disp(["coef. de correlação.= " co ])
disp(["erro padrão de estimativa=" err])
F=j/(RSS/(n-2));
disp(["F:", string(F)])
dfe=n-2;dfm=1;
[P,Q]=cdff("PQ",F,dfm,dfe);
disp(["p:",string(Q)])
den=sum((x-mean(x))^2);
s2=RSS/(n-2);varb=s2/den;
vara=s2*(1/n+(mean(x)^2)/den);
disp(["Variância de a:", string(vara)])
disp(["Variância de b:", string(varb)])
tb=b/sqrt(varb);ta=a/sqrt(vara);
disp(["t de b:",string(tb)]);
```

```
disp(["t de a:",string(ta)])
//fazer um gráfico dos dados e da regressão
z=a+b*x;
d=x';
[M]=[y;z]';
plot2d(d,[M]);
xtitle("Dados e ajustamento","x","y")
```

Seleccione este texto, copie-o e cole-o no SciPad. Guarde-o com um nome a sua escolha como um ficheiro *.sci. Na janela de comandos execute-o. Obterá a saída seguinte e o gráfico da figura 3.1.

```
y=a+b*x

!a=  1.26  b=  2.0685714  !

!R quadrado=   0.9931338  !

!R quadrado ajustado:  0.9914172  !

!coef. de correlação.=   0.9965610  !

!erro padrão de estimativa=  0.3597618  !

!F:  578.56071  !

!p:  0.0000177  !

!Variância de a:  0.1121714  !

!Variância de b:  0.0073959  !

!t de b:  24.053289  !

!t de a:  3.7620921  !

 exec done
```

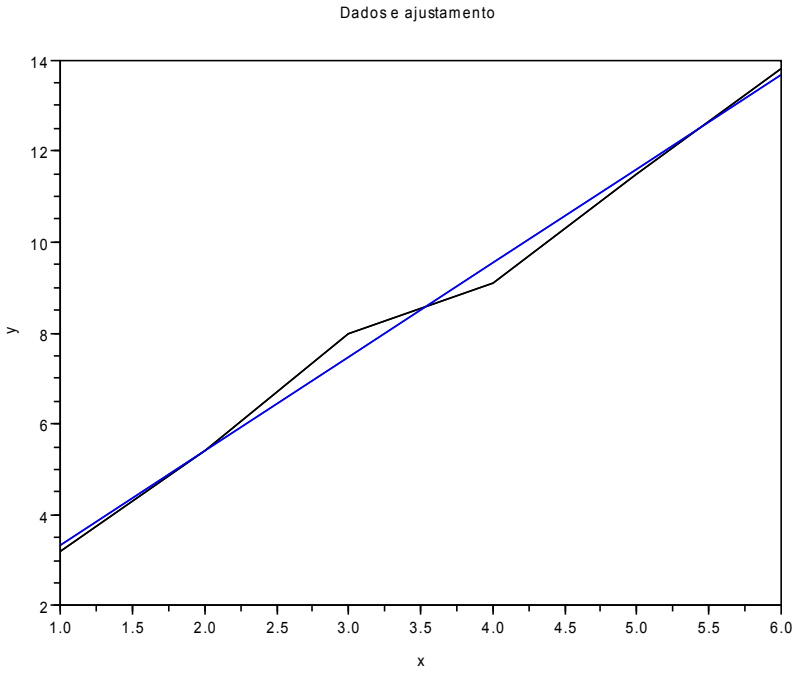


Figura 3.1. Dados e recta ajustada

3.9 Quadro sinóptico dos comandos introduzidos neste capítulo

Comando	Descrição
\	Divisão direita de uma matriz
.*	Multiplicação elemento a elemento
./	Divisão elemento a elemento
.\	Divisão direita elemento a elemento
bdiag	Valores e vectores próprios de uma matriz
det	Determinante de uma matriz
diag	Diagonal principal de uma matriz
expm	Exponencial de uma matriz
eye	Matriz de zeros com 1's na diagonal principal
inv	Inversa de uma matriz
linsolve	Solução de um sistema de equações lineares
logm	Logaritmo de uma matriz
logspace	Vector de números correspondentes aos logaritmos na base 10 do intervalo especificado, igualmente espaçados
ones	Matriz de 1's. Sintaxe igual à do comando **zeros**
poly	Polinómio característico de uma matriz
spec	Valores próprios de uma matriz
sqrm	Raiz quadrada de uma matriz
trace	Traço de uma matriz
zeros	Matriz só de zeros

Capítulo 4

Gráficos

A panóplia de recursos do Scilab para lidar com gráficos é muito variada e potente e é impossível cobri-la toda num texto introdutório. Nalguma extensão, ampliaremos agora a abordagem iniciada no capítulo 2, sob a perspectiva de tirar o máximo benefício da interface gráfica das janelas dos gráficos. Iremos assim:

- Expandir a criação de gráficos de duas dimensões
- Inserir texto na área do gráfico
- Introduzir os gráficos a três dimensões
- Obter as coordenadas de pontos duma curva num gráfico
- Utilizar os menus e botões da janela dos gráficos para facilmente os formatar
- Fazer gráficos com marcas ou símbolos gráficos no traçado das curvas

4.1 Continuando a criar gráficos de duas dimensões (2D)

Na secção 2.13 introduzimos o comando plot2d3 (saída na figura 2.14) o que faz supor que existem pelo menos os comandos plot2d1 e plot2d2. De facto, assim é, e não só.

O comando plot2d pode ser chamado de duas maneiras:

- plot2d(*x,[y]*)
- plot2d(*x,[y], atributos opcionais*)

Vamos usar os menus e botões da janela dos gráficos para introduzir os atributos opcionais. Vejamos agora as variantes de plot2d existentes:

- plot2d1 é uma forma obsoleta em desuso
- plot2d2 faz o gráfico e uma função de escada. Se entrarmos com plot2d2(), para obtermos uma demonstração, a saída é o gráfico da figura 4.1.
- plot2d3 desenha o gráfico com barras verticais, como vimos na figura 2.14.
- plot2d4 insere setas na linha do gráfico. Se entrarmos com plot2d4(), para obtermos uma demonstração, a saída é o gráfico da figura 4.2.

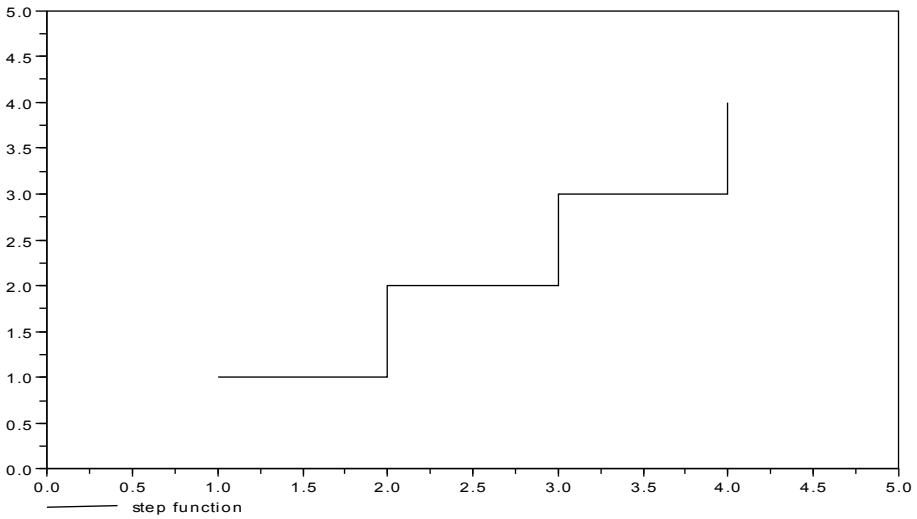


Figura 4.1. Gráfico criado pelo comando plot2d2

4.1.1 Identificação de pontos numa curva de um gráfico

Suponha o leitor que antes de fazer a análise da função polinomial da secção 2.9, inscrita na figura 2.7., queria identificar alguns dos pontos de interesse no seu gráfico.

O comando locate permite fazer isto. Recriemos o gráfico com os comandos

```
-->y=poly([3 -2 -12 -2 2],"x","coeff");

-->x=[-2:0.1:3.2};

-->plot2d(x,[horner(y,x)'])

-->xgrid():
```

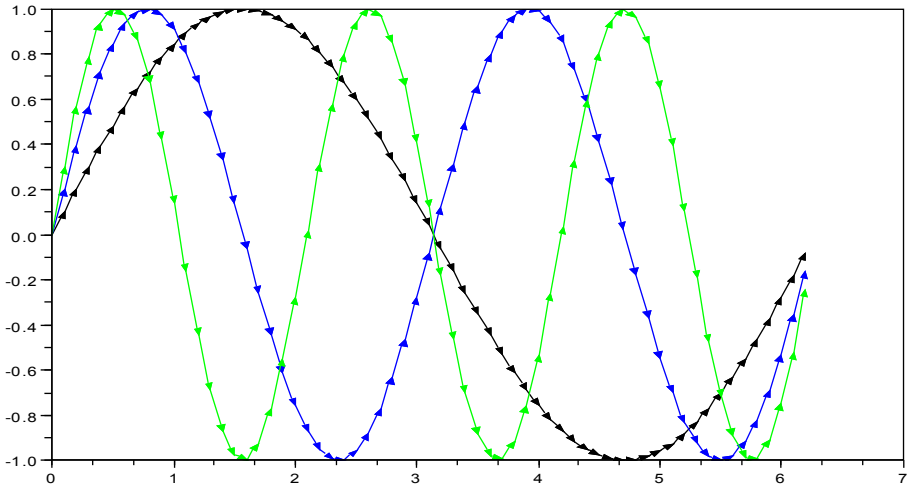


Figura 4.2. Gráfico criado pelo comando plot2d4

Agora só nos falta usar o comando locate, para podermos localizar os pontos com o rato:

x=locate([número de pontos, opção (dígito)])

Queremos identificar 9 pontos e marcá-los com X (opção=1), até clicarmos o último. O comando a usar é:

```
-->x=locate(9,1)
 x  =


       column 1 to 5

 - 1.7734375  - 1.3125        - 0.7578125  - 0.6875       - 0.078125
 - 0.1796407  - 4.6107784    - 0.8982036  - 0.1796407    3.1736527

       column 6 to 9

   0.421875     1.1796875    2.1875         3.03125
 - 0.0598802  - 15.628743  - 34.071856  - 0.0598802
```

Compare a pontaria dos meus cliques, com os pontos obtidos analiticamente, na secção 2.9. Antes de clicar o nono ponto, o aspecto do gráfico é o da figura 4.3.

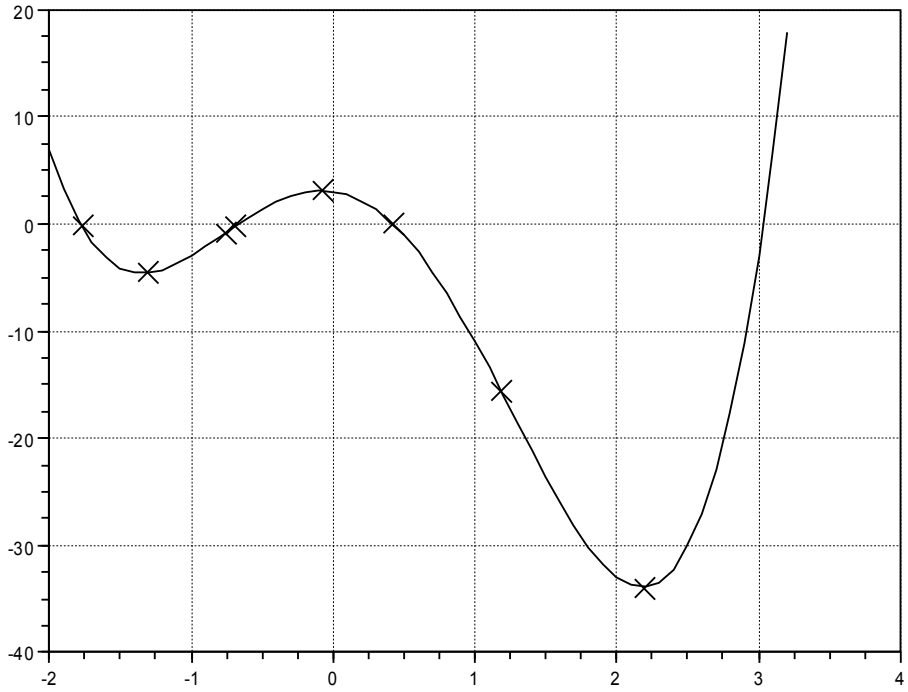


Figura 4.3. Gráfico da função polinomial da secção 2.9, submetida ao comando locate, ao fim da selecção do oitavo ponto.

4.2 Escrever no espaço do gráfico e identificar curvas

Na figura 2.3, apresentamos duas curvas, exponencial e de Gompertz, sem qualquer identificação. Nesta secção e na seguinte vamos sanear esta lacuna.

Recriemos as curvas.

```
-->x=linspace(0,3,300);

-->plot2d(x,[2*exp(0.5*x);2*0.2^(exp(-0.8*x)-1)]');

-->xgrid()
```

O comando a que vamos recorrer usa como localização, as escalas dos eixos do gráfico, x e y, por esta ordem. É o xstring:

xstring(*valor de x, valor de y, 'texto'*)

```
-->xstring(0.1,7,'Curvas exponencial e de Gompertz');

-->xstring(0.75,5,'Gompertz')

-->xstring(2,5,'Exponencial')
```

O aspecto final do gráfico é o da figura 4.4.

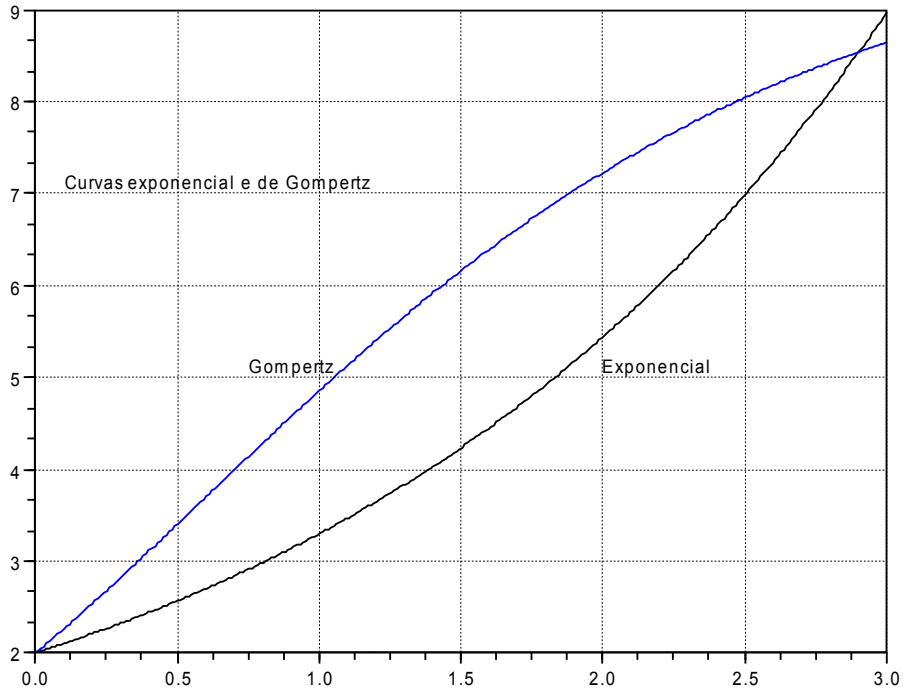


Figura 4.4. Gráfico com legendas, de uma curva de Gompertz e outra exponecial

4.3 Escrever no espaço do gráfico as legendas das curvas

Agora queremos inscrever a legendas das curvas no espaço do gráfico, com uma moldura, num ponto que escolhemos depois de arrasta a moldura e fazer um clique com o rato, no sítio desejado. Vou usar o comando legend

```
hl=legend(['nome da 1ª curva';'nome da 2ª curva'],a=5);
```

```
-->hl=legend(['Exponecial';'Gompertz'],a=5);
```

hl corresponde à expressão handle da biblioteca de gráficos.

Depois deste comando e de ter clicado no canto inferior direito da área do gráfico, ele fica com o aspecto da figura 4.5.

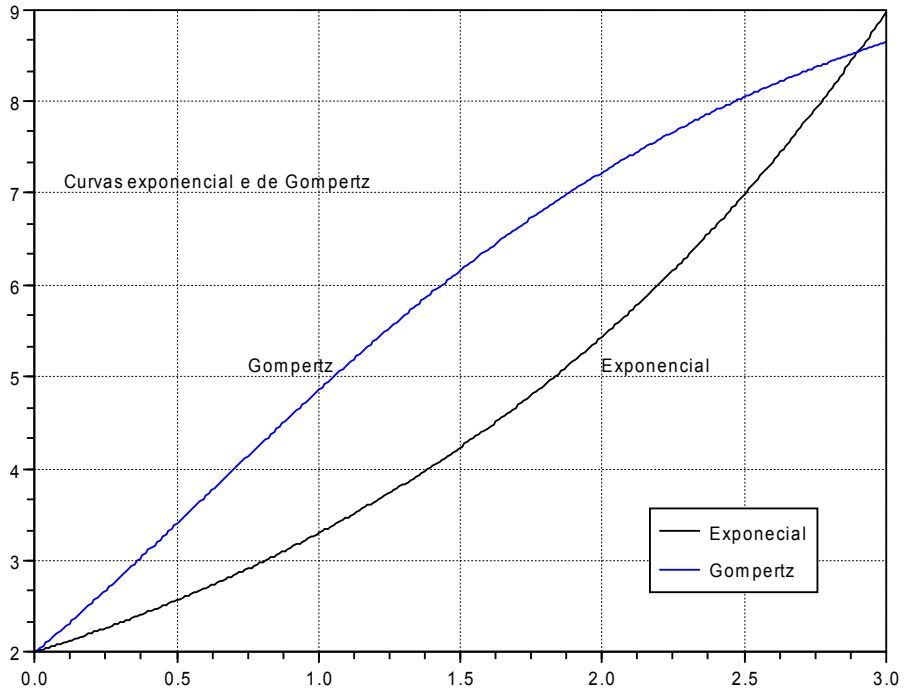


Figura 4.5. O gráfico com as legendas das curvas, no canto inferior direito.

4.4 Formatar gráficos 2D com os menus e botões da sua janela

Os programadores do Scilab criaram a interface de edição dos gráficos com o propósito de a tornar mais fácil e nós vamos tirar partido dela.

Usemos os comandos já introduzidos na secção 2.7, para inserir a num só gráfico, obtendo a figura 4.6, que queremos formatar.

Comecemos com um gráfico que só tem os eixos e as curvas, sem mais nenhum texto, como o da figura 4.6, obtido inserido no mesmo gráfico a função polinomial e as suas derivadas, abordadas na secção 2.7. Detenhamo-nos no que a janela do gráfico exibe, para além da figura a que serve de suporre.

4.4.1 Os menus da barra superior

Comecemos pelos menus da barra superior.

O menu **File** permite o habitual e pouco mais:

New – abrir uma nova janela de gráfico em branco.

Load – abrir um ficheiro de um gráfico. Por exemplo, se quisermos comparar dois gráficos estando na janela de um deles, abrimos outra com **New** e depois nesta nova janela usamos o **Load** para abrir o outro gráfico. Podemos redimensionar as janelas e arrumá-las lado a lado.

Save – guardar o gráfico**.**

Export – guardar em formatos tais como postscript e Latex**.**

Copy to clipboard – copiar para a memória para eventualmente ser colado, por exemplo, numa página de processador de texto**.**

Print setup – Configurações para imprimir
Print – imprimir.
Close – fechar a janela do gráfico

O menu **"Tools"** exibe:

Toolbar – fazer aparecer ou desaparecer a barra de ferramentas.
Zoom – ampliar uma região do gráfico.
UnZoom – anular a ampliação.
2D/3D Rotation – Proceder à rotação dos gráficos

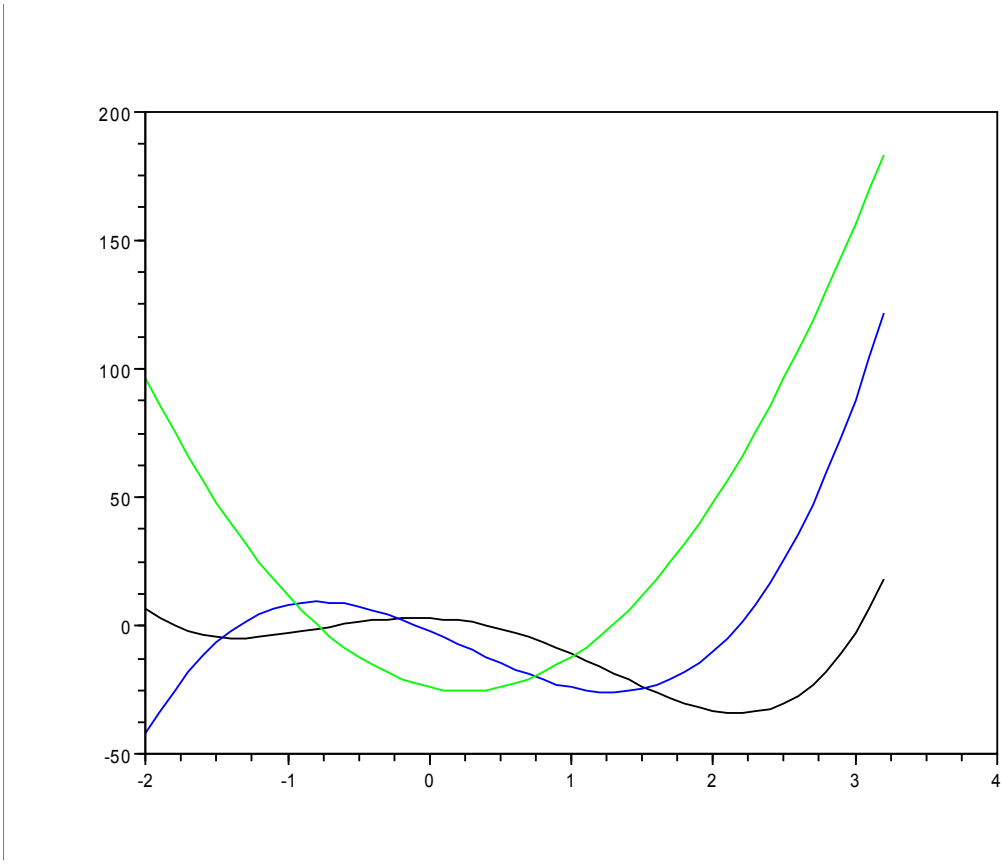


Figura 4.6. O gráfico antes de ser formatado.

O menu **Edit** tem o seguinte:

Select figure as current – quando se tem mais de uma janela de gráfico selecciona a que vai ser objecto de comandos.
Redraw figure – Redesenhar a figura (em desuso)
Erase figure – limpa a janela do gráfico
Figure properties – abre uma janela de alteração das propriedades da figura
Current axes properties – abre uma janela que permite formatar os eixos
Start entity picker – activa a possibilidade de escolher uma área
Stop entity picker – encerra a possibilidade de escolher uma área

4.4.2 Os botões da barra de ferramentas

Da esquerda para a direita:
1º Activa o zoom; anula a ampliação; 2º permite rodar o gráfico a três dimensões; 3º abre a janela que permite formatar todo o gráfico; 4º activa/desactiva a possibilidade de escolher

uma área (**clipping)**. São alternativas a alguns comandos dos menus. Nesta iniciação vamos ignorar tudo o que tenha a ver com a selecção de áreas no gráfico.

4.4.3 Formatar os eixos e título e atributos gerais

Numa iniciação ao software, não aconselho que se mexa nas propriedades da figura. Por isso comecemos pelos eixos do gráfico. Façamos um clique no botão **GED.** Surge a janela da figura 4.7, onde fizemos um clique em Áxis(1).

À direita do seleccionador dos elementos do gráfico ("Objects Browser"), surge um conjunto de separadores, sob o título "Objects Properties" (propriedades dos objectos).

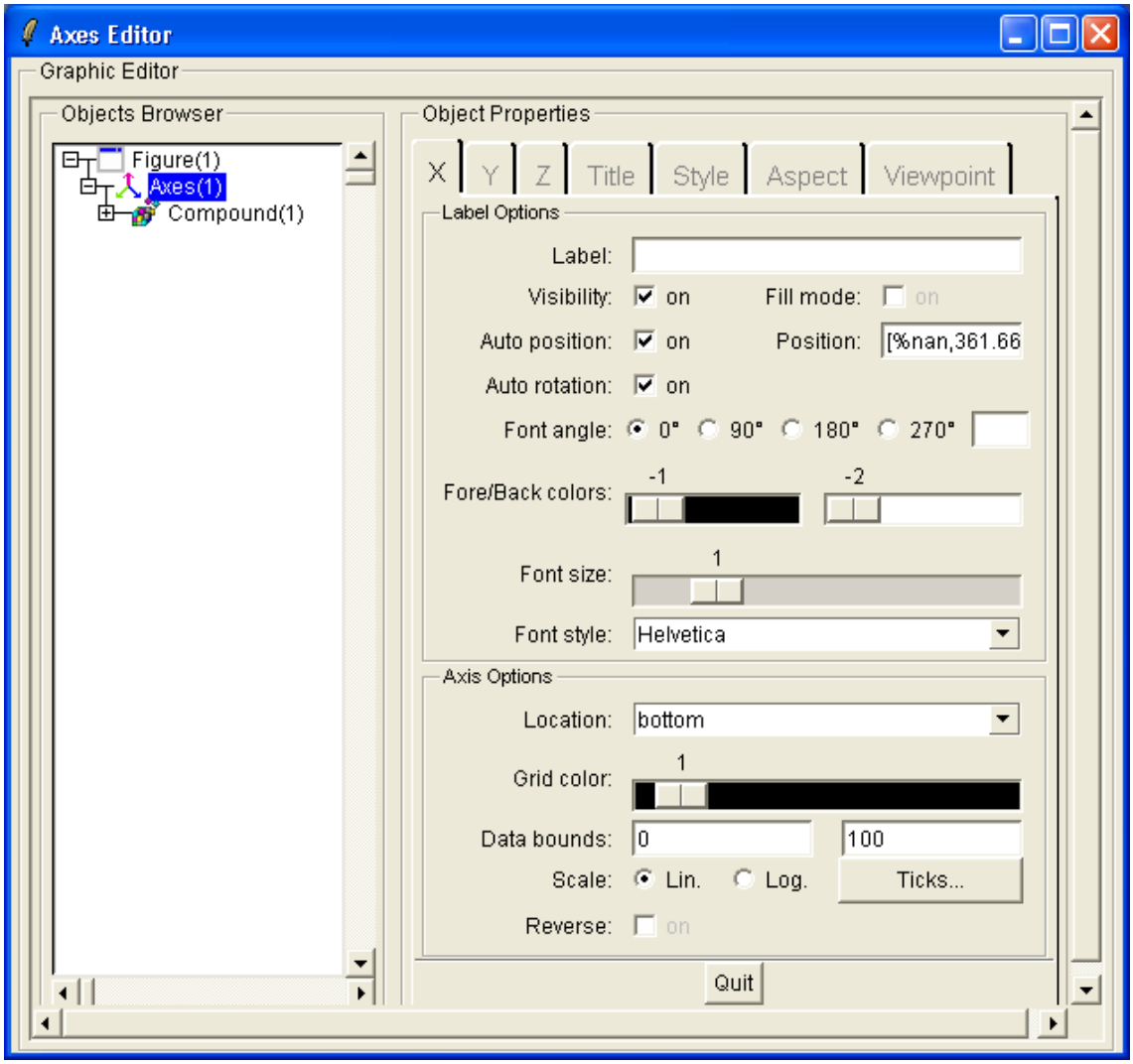


Figura 4.8. Figura do editor de gráfico.

Os separadores dos três eixos (X, Y, Z) são iguais em possibilidades de intervenção por parte do utilizador. Estão reunidas em dois grupos:

1- **Label Options** que permitem formatar a legenda quanto ao tipo de letra, posição e sua orientação.

2- **Axis Options** que incidem sobre o eixo (posição, espessura do traço, cor da grelha correspondente, escala linear ou logarítmica, limites da escala, marcações da escala).

Sejamos menos sintéticos. Comecemos pelas opções para a legenda (**Label Options**):

- No espaço para entrada de texto (caixa de texto), em frente de "**Label**", clica-se, escreve-se a legenda do eixo, e depois pressiona-se **Enter/Return**. O botão de escolha ou opção **Visibility** deve ser activado.
- Se escolhermos **Fill mode on,** é traçado um rectângulo à volta da legenda, cujo fundo pode ser colorido, como veremos abaixo.
- Sugiro que se mantenha o botão **Auto-position on**, e a legenda é inserida a meio do comprimento do eixo.
- No entanto, na caixa de texto em frente a **Position**, permite escrever a posição da legenda, referenciada à escala do eixo. Não se esqueça de pressionar **Enter/Return**.
- Os botões de opção **Font angle** permitem orientar a legenda. Sem qualquer intervenção, a legenda é escrita ao longo do eixo.
- Na linha seguinte, os botões deslizante denominados **Fore/Back colors** permitem escolher a cor das letras (o da esquerda) e da caixa da legenda, se activámos **Fill mode**.
- O deslizante **Font size** permite escolher o tamanho da letra.
- A entrada **Font style**, como em qualquer processador de texto, permite escolher o tipo de fonte. Por defeito é o Helvética.

  Vejamos as opções de formatação para o eixo (**Axis Options**).
- **Location** permite três posições para o eixo: no topo, em baixo, no meio.
- **Data bounds** permite alterar os limites da escala do eixo.
- **Grid color,** se o gráfico tem grelha, permite escolher a cor das linhas correspondentes á escala do eixo.
- **Scale** oferece a possibilidade de escolher uma escala linear ou logarítmica.
- **Ticks** abre uma janela que permite alterar as marcações no eixo.
- **Reverse** põe a escala com o valor máximo onde por defeito fica o mínimo.

O separador para inserir o título do gráfico tem só as **Label Options** idênticas às dos eixos, e que se utilizam da mesma maneira.

O separador **Style**:

- Deve ter o botão de opção **Visibility** activado para se ver o gráfico na sua janela.
- **Font style** permite escolher o tipo de letra das escalas dos eixos.
- **Font color** a cor dos algarismos das escalas.
- **Font size** o tamanho do tipo de letra escolhido atrás.
- **Fore color** refere-se a cor do traço dos eixos.
- **Back color** a cor de fundo ou do espaço definido pelos eixos.
- **Thickness** à expessura do traço dos eixos.

  Nesta fase de aprendizagem ignoramos **Hidden color** e **Line style**.

No separador **Aspect,** sugerimos que a leitora, por si, verifique o efeito de activar os botões de opção **Auto clear, Auto scale** e **Box. Isoview, Tight limits. Cube scaling** só se aplica aos gráficos 3D. Ignore o resto do separador.

O separador **Viewpoint,** nos gráficos 3D, permite definir a elevação re o azimute do gráfico. Estes dois conceitos esclarecem-se na figura 4.9. Se num gráfico 2D clicarmos no botão 3D , ele aparece no plano definido pelos eixos x e y, do gráfico a 3D.

Tente agora, no seu computador: a) criar o gráfico da figura 4.6; b) inscrever um título, legendas dos eixos, alterar os seus tipos de letra, tamanhos, pelo menos. Mas sugiro que se aventure a mais.

4.4.4 Formatar o traçado das curvas e as entidades no espaço dos eixos

Na janela do editor do gráfico, clique em **compound(1)** e depois em **Polyline(3).** Surge-lhe o editor das linhas das curvas do gráfico, denominado **Polyline Editor**, como se exibe na figura 4.6, que permite formatar a curva da função, a primeira grafada.

.

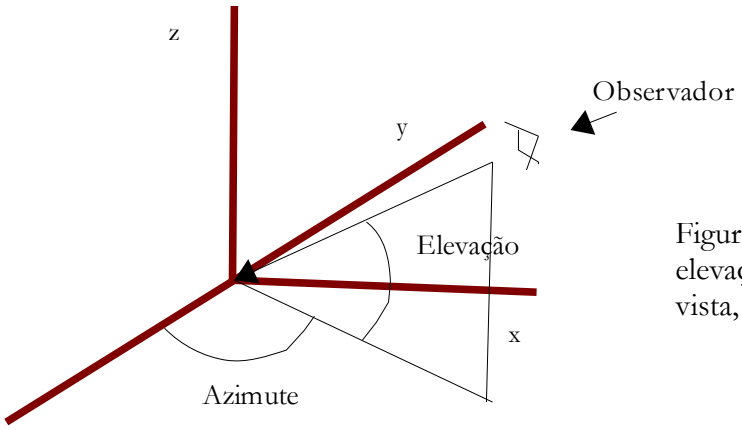


Figura 4.9. Azimute e elevação do ponto de vista, nos gráficos 3D

O editor das curvas (**Polyline Editor**) tem três separadores: **Style**, **Data**, **Clipping**. Vamos ignorar o último.

Comecemos pelo mais curto, no número de opções.

4.4.4.1 Editar os dados de uma curva

O **Data** tem uma caixa de texto, que é simultaneamente um menu. Por defeito dá informação sobre a estrutura dos dados, no nosso caso uma matriz ([53x2 double array]). Se clicarmos no botão à direita desta informação, aparece uma caixa onde surge **Edit data**. Se clicarmos em **Edit data,** aparece uma espécie de folha de cálculo com os valores das variáveis (ver figura 4.11), que permite alterá-los. Clique na célula cujos valores quiser alterar, ela fica com o fundo a amarelo, e introduza os novos valores e pressione **Enter/Return**. Este procedimento é repetido em cada célula a alterar. No fim, clique no botão, acima e à esquerda, **Refresh**, para sair e fechar.

4.4.4.2 Editar uma curva

Ocupemo-nos agora do separador **Style**. Vejamos, então, as opções de controlo, ignorando **Interp. mode**.

- **Visibility** é um um botão que permite fazer desaparecer e aparecer o a linha da área dos eixos.
- **Closed** fecha a curva traçando uma recta entre os seus extremos.
- **Fill mode** com a opção **Close** activada permite preencher o espaço entre a curva e recta com uma cor que se escolhe abaixo, no deslizante
- **Background.**

Existem dois grupos de editor. Ou escolhemos uma linha continua, activando **Line mode,** ou de uma série de marcas (por exemplo, triângulos, estrelas, cruzes, etc.), seleccionando o botão **Mark mode**, que fica mais abaixo. Concetremos na curva desenhada por linha (**Line mode**).

- **Polyline style** tem uma caixa de texto que permite escolher vários tipos de reprentação dos valores (por exemplo, em escada, com setas), clicando no seu botão à direita.

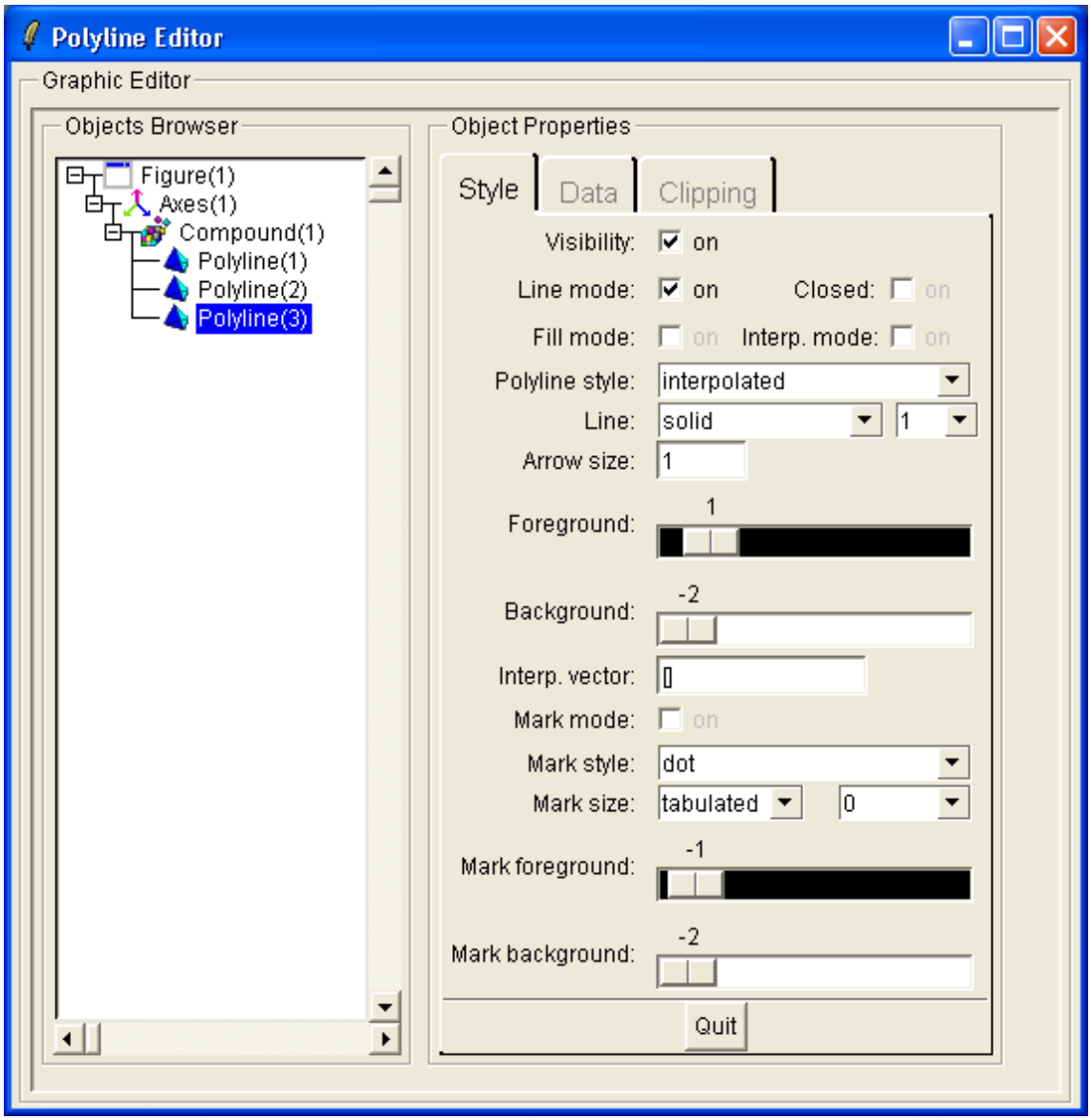
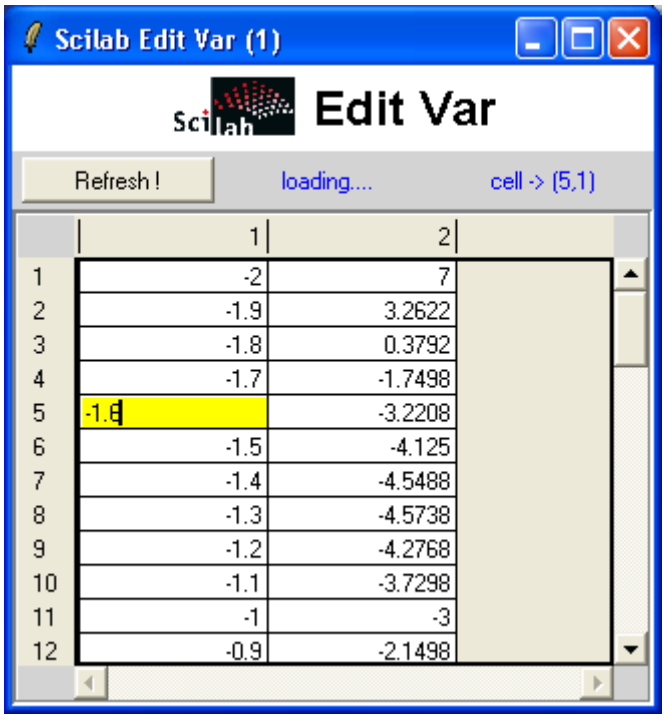


Figura 4.10. Editor do gráfico pronto a iniciar a formatação da curva da função polinomial.

- Se tivermos escolhido o estilo correspondente a uma linha simples (**interpolated**), como na figura 4.6, na caixa de texto **Line** é possível escolher entre uma linha continua e vários tipos de ponteado e tracejado.
- **Arrowed size** permite escolher o tamanho das setas, se tivermos escolhido em **Polyline style,** uma linha com setas (**Arrowed**).
- **Foreground** tem um botão deslizante que permite escolher a cor da linha.
  Igoramos também **Interp. vector**.
  Em **Mark mode**, podemos:
- Escolher o símbolo da marca em **Mark style**.
- A caixa de texto pode altera a marca seleccionada para pontos (**point**) ou mantê-la (**tabulated**).
- Ao lado pode-se escolher o tamanho das marcas.
- **Mark foreground** escolhe as cores das linhas da marca.

- **Mark background** escolhe a cor de enchimento da marca. Por exemplo, uma estrela pode ser desenhada com uma linha verde, e preenchida com uma cor azul.



	1	2
1	-2	7
2	-1.9	3.2622
3	-1.8	0.3792
4	-1.7	-1.7498
5	-1.6	-3.2208
6	-1.5	-4.125
7	-1.4	-4.5488
8	-1.3	-4.5738
9	-1.2	-4.2768
10	-1.1	-3.7298
11	-1	-3
12	-0.9	-2.1498

Figura 4.11. Editor dos dados (**Edit Var**) da curva da função polinomial, com uma célula seleccionada para ser alterada

---

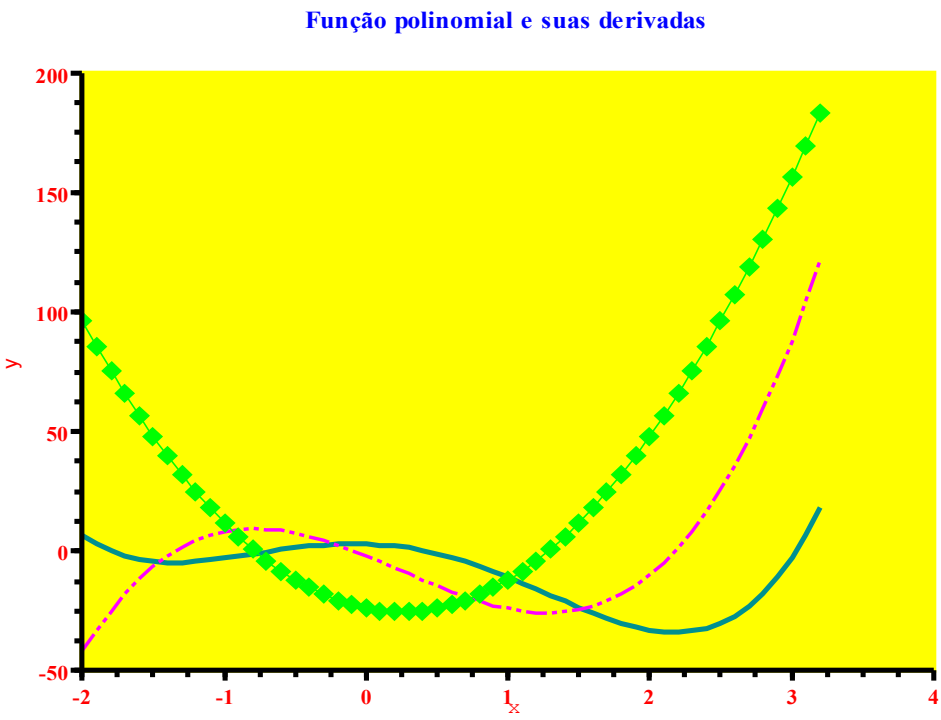


Figura 4.12. O aspecto do gráfico da figura 4.6, depois de modificado com o auxílio do editor de gráfico (**Graphic Editor**).

Chegados aqui, sugiro ao leitor que, como se diz, "ponha as mãos na massa". Recrie o gráfico da figura 4.6, e use o editor do gráfico, que merece ser bem explorado. Para acicatar a sua iniciativa, deixo-lhe a figura 4.12. Compare-a com a figura 4.6.

Se a leitora activar o editor de gráfico para a figura 4.5, obtém o que se exibe na figura 4.13, depois de abrir toda a árvore da figura (clique em todos os sinais "+").

A árvore apresenta dois "Áxis". O "Áxis(1)" corresponde à moldura, texto e linhas da legenda que está inserida em baixo, à direita, no espaço dos eixos, que também pode ser formatada.

O "Áxis(2)" além da formatação mencionada na descrição do editor de gráfico, acabada de fazer, permite editar o texto inserido no espaço dos eixos (Text(3), Text(4), Text(5)). Experimente também este caso de formatação de gráfico.

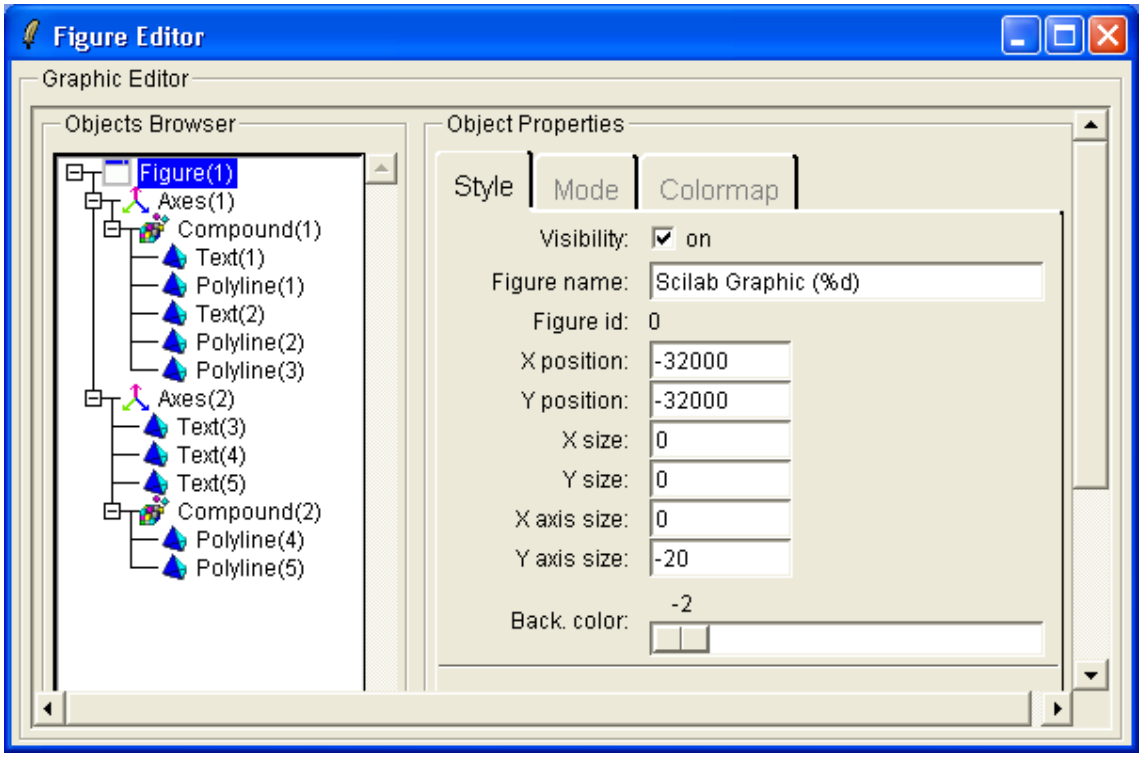


Figura 4.13. Editor de gráfico associado à figura 4.5.

4.5 Outras possibilidades de criar facilmente gráficos a duas dimensões

O Scilab ainda apresenta os seguintes comandos para gráficos a duas dimensões, além do plot2d e histplot:

fplot2d: faz o gréfico de uma curva definida por uma função.
champ: cria um campo de vectores de duas dimensões
champ1: permite colorir os vectores do campo
fchamp: campo de vectores associado a uma equação diferencial ordinária (EDO)
contour2d: traça as curvas de nível de uma superfície num gráfico 2D
fcontour2d: traça as curvas de nível de uma superfície definida por uma função num gráfico 2D
grayplot: cria uma superfície num gráfico a duas dimensões usando cores

fgrayplot: cria uma superfície definida por uma funçã num gráfico a duas dimensões usando cores
pie: cria um gáfico circular, como se ilustra na figura 4.13
Sgrayplot: suaviza uma superfície num gráfico a duas dimensões usando cores, como se ilutra na figura 4.14
Sfgrayplot: suaviza uma superfície definida por uma função num gráfico a duas dimensões usando cores.
errbar : acrescenta baras verticais de erros a um gráfico 2D
Matplot : cria uma quadricula preenchida a cores

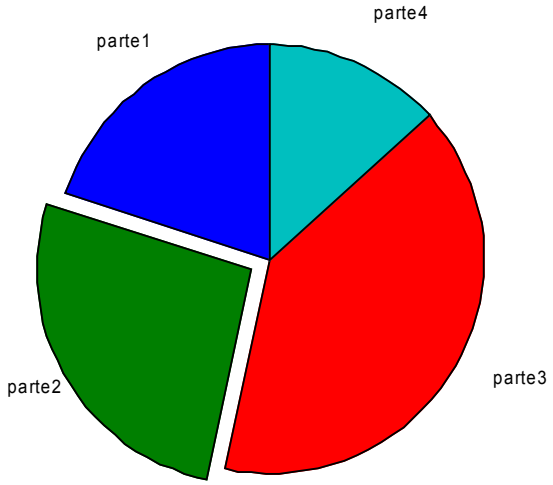


Figura 4.13. Exemplo de um gráfico circular

O Scilab tem ainda comandos para criar tipos particulares de gráficos a duas dimensões, tais como:

xpoly: desenha uma linha ou um polígono
xpolys: desenha um conjunto de linhas ou polígonos
xrpoly: desenha um polígono regular
xsegs: desenha segmentos de recta não ligados
xfpoly: preenche um polígono
xfpolys: preenche um conjunto de polígonos
xrect: desenha um rectângulo
xfrect: preenche um rectângulo
xrects: desenha ou preenche um conjunto de rectângulos
xarc: desenha parte de uma elipse

xarcs: desenha partes de um conjunto de elipses
xfarc: preenche parte de uma elipse
xfarcs: preenche partes de um conjunto de elipses

Para obter mais informação e exemplos de uso dos comandos, como já sabe, entre com help *nome do comando*. Copie os exemplos da Ajuda, e cole-os na janela de comandos para ver o que produzem e poder interpretar melhor o significado das instruções que têm.

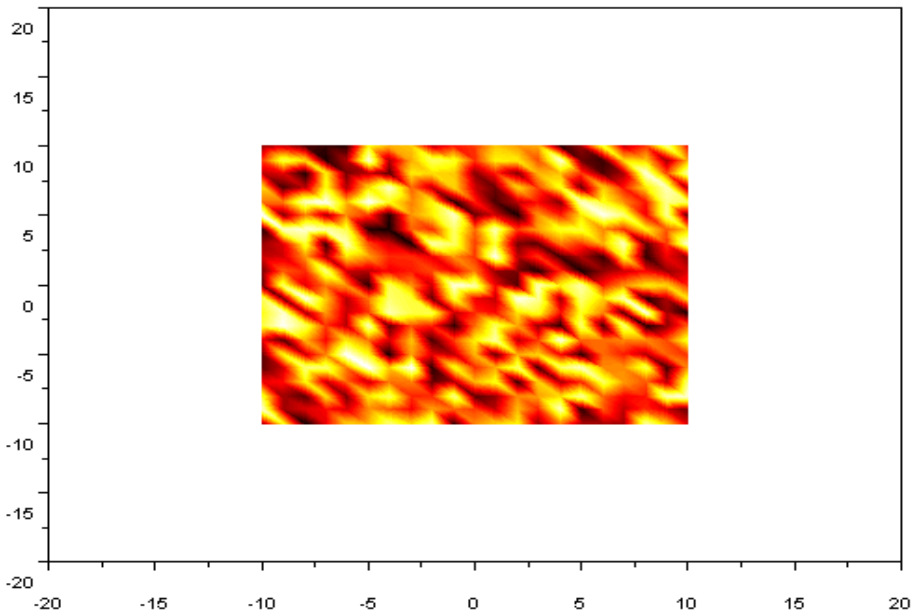


Figura 4.14. Gráfico obtido com o comando Sgrayplot

4.6 Inserir vários gráficos na mesma janela

Desde o capítulo 1, dada a figura 1.2, que o leitor sabe que se pode inscrever mais de um gráfico numa janela. O comando que permite criar um vector ou uma matriz de gráficos (caso da figura 1.2) é o seguinte subplot:

subplot(número de linhas com gráficos, número de colunas com gráficos, número de ordem do gráfico)

Abaixo deste comando insere-se o comando que cria o gráfico na célula da matriz de gráficos, como por exemplo, plot2d(x,[y]).

Se quisermos dois gráficos lado a lado teremos:

subplot(1,2,1)
Comando para criar o gráfico 1
subplot(1,2,2)
Comando para criar o gráfico 2

Gráfico 1	Gráfico2

Se quisermos um gráfico a cima e outro abaixo:

```
subplot(2,1,1)
Comando para criar o gráfico 1
subplot(2,1,2)
Comando para criar o gráfico 2
```

O último exemplo: uma matriz de quatro gráfico:

```
subplot(2,2,1)
Comando para criar o gráfico 1
subplot(2,2,2)
Comando para criar o gráfico 2
subplot(2,2,3)
Comando do gráfico 3
subplot(2,2,4)
Comando do gráfico 4
```

Vejamos um exemplo de aplicação, grafando a variação da densidade das árvores, os crescimentos em diâmetro, altura e volume do tronco, de um pinhal bravo. Repare na instrução boxed=1, no comando xtitle, que cria uma caixa nos títulos e legendas. Os gráficos foram depois de criados, foram formatados com o editor de gráficos.

```
// tabela de produção do pinheiro bravo, dos 10 aos 80 anos
//den=densidade, árvores/hectare, gráfico 1
//dap=diâmetro à altura do peito médio, cm, gráfico 2
//alt=altura média, metros, gráfico 3
//vol=volume do tronco, metros cúbicos/ha, gráfico 4
//t=Idade em anos
//Modelo KHABA de L. S. Barreto
//quatro equações de Gompertz
xbasc()
t=10:1:80;
den=314*6.18^exp(-0.05*(t-10));
dap=34*0.4^exp(-0.05*(t-10));
alt=24*0.4^exp(-0.05*(t-10));
vol=400*0.4^exp(-0.05*(t-10));
subplot(2,2,1)
plot2d(t,[den])
xtitle('Densidade','Idade, anos','àrv./ha',boxed=1)
subplot(2,2,2)
plot2d(t,[dap])
xtitle('Diâmetro médio','Idade, anos','cm',boxed=1)
subplot(2,2,3)
plot2d(t,[alt])
xtitle('Altura média','Idade, anos','m',boxed=1)
```

```
subplot(2,2,4)
plot2d(t,[vol])
xtitle('Volume em pé','Idade, anos','m.c./ha',boxed=1)
```

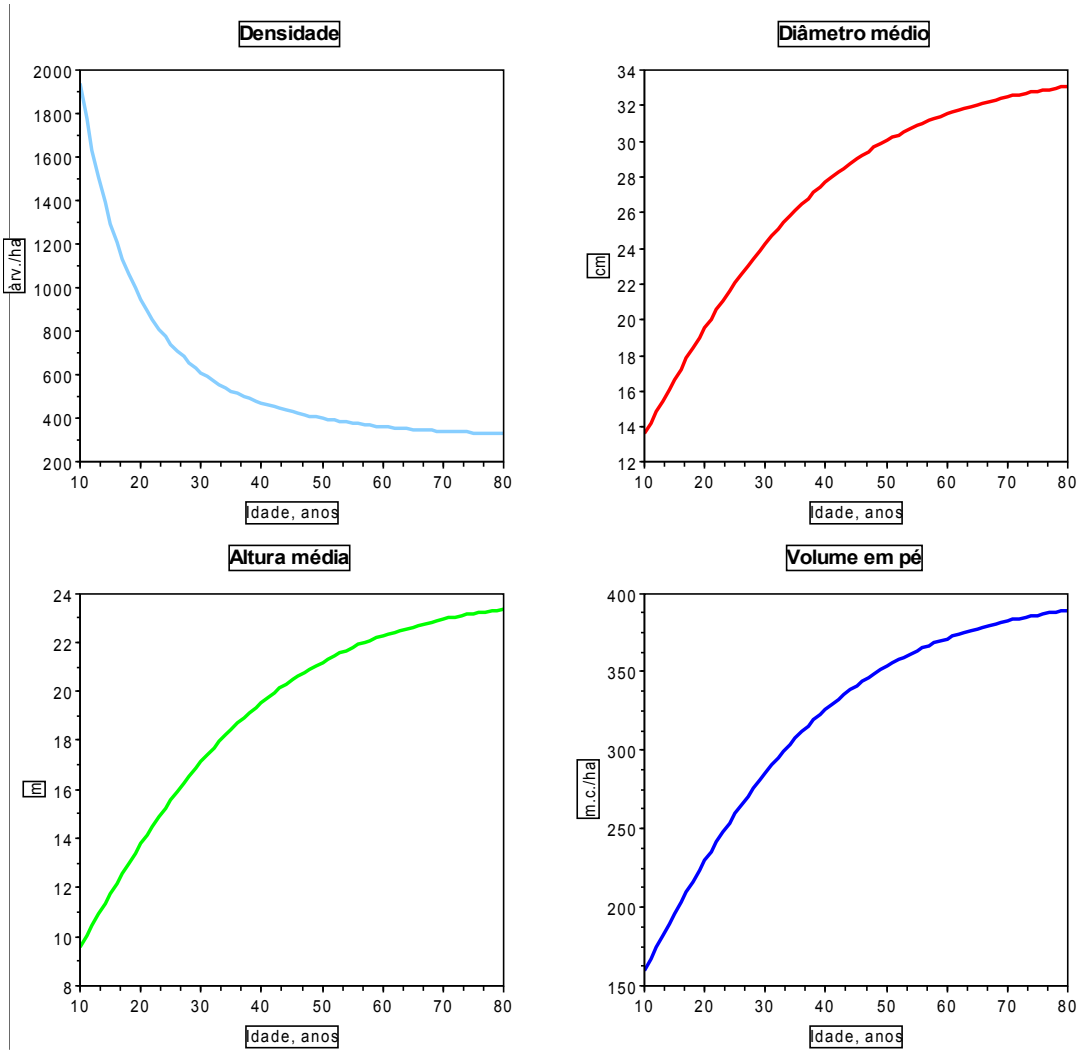


Figura 4.15. Tabela de produção do pinhal bravo, criada usando o comando subplot

4.7 Gráficos a três dimensões (3D)

Um dos comandos tidos como mais empregues pelos utilizadores do Scilab, é o plot3d, que apresenta variantes tal como o plot2d. Enumeremos esta família de comandos:

plot3d – faz um gráfico de uma superfície 3D
plot3d1 - faz um gráfico 3D de uma superfície cinzento ou com níveis a cores
plot3d2 - faz um gráfico 3D de uma superfície definida por rectângulos
plot3d3 - faz um gráfico 3D de uma superfície em rede ("mesh") definida por rectângulos
mesh – faz um gráfico 3D definido por uma rede

Para usar o comando plot3d, criamos primeiro os vectores de valores de x, y e z.

Agora com a existência do editor do gráfico, basta escrever:
plot3d(x,y,z)

e depois fazer a formatação desejada com o editor de gráficos. A listagem de comandos seguinte cria uma janela com uma matriz de quatro gráficos que ilustram as variantes de plot3d, exibida na figura 4.16, com os títulos editados. Os exemplos foram tirados da ajuda do Scilab.

```
xbasc() //limpa qualquer janela de gráfico aberta
//dados para a função z=f(x,y)
//usada pelos comandos plot3d e plot3d1
t=[0:0.3:2*%pi]';
z=sin(t)*cos(t');
//dados para os comandos plot3d2 e plot3d3
u = linspace(-%pi/2,%pi/2,40);
v = linspace(0,2*%pi,20);
X = cos(u)'*cos(v);
Y = cos(u)'*sin(v);
Z = sin(u)'*ones(v);
//inserir 4 gráficos na mesma janela
subplot(2,2,1)
plot3d(t,t,z)  //comando plot3d
xtitle("plot3d","x","y","z")
subplot(2,2,2)
plot3d1(t,t,z)  //comando plot3d1
xtitle("plot3d1","x","y","z")
subplot(2,2,3)
plot3d2(X,Y,Z);  // comando plot3d2
xtitle("plot3d2","X","Y","Z")
subplot(2,2,4)
plot3d3(X,Y,Z)  // comando plot3d3
xtitle("plot3d3","X","Y","Z")
```

4.8 Formatar gráficos 3D com os menus e botões da sua janela

Se chamarmos o editor de gráfico da janela da figura 4.16, e abrirmos as árvores dos quatro conjuntos de eixos obtemos o que se exibe na figura 4.17. Como de costume, o "Axis1" corresponde ao gráfico 4 (plot3d3), o "Axis2" ao gráfico 3 (plot3d2), o "Axis3" ao gráfico 2 (plot3d1), e o "Axis4" ao gráfico 1 (plot3d).

A formatação dos eixos, chamada clicando em "Áxis" é idêntica à descrita para os gráficos 2D. O aspecto distintivo relevante surge no separador "Viewpoint" que permite alterar a elevação e azimute do observador virtual, e reduzir o gráfico a duas dimensões.

Os gráficos 1 e 2, não criam rectângulos por isso são identificados por Plot3d. O comando plot3d2, do gráfico 3 cria uma superfície de rectângulos ("facets"), por isso surge Fac3d. O gráfico 4 cria dois conjuntos de linhas ("polylines") "Compound(1)" e "Compound(2)". O primeiro com 40 linhas e o segundo com 20 linhas, que se editam como nos gráficos 2D.

Cliquemos, no "Axis4" em Plot3d. Surge-nos a janela da figura 4.18, que têm aspectos comuns ao da figura 4.10, referida a gráficos 2D e que se usam da mesma maneira. O aspecto mais inovador desta janela diz respeito aos botões de opção de "Color flag" que permitem passar do aspecto do gráfico um para o dois e vice-versa.

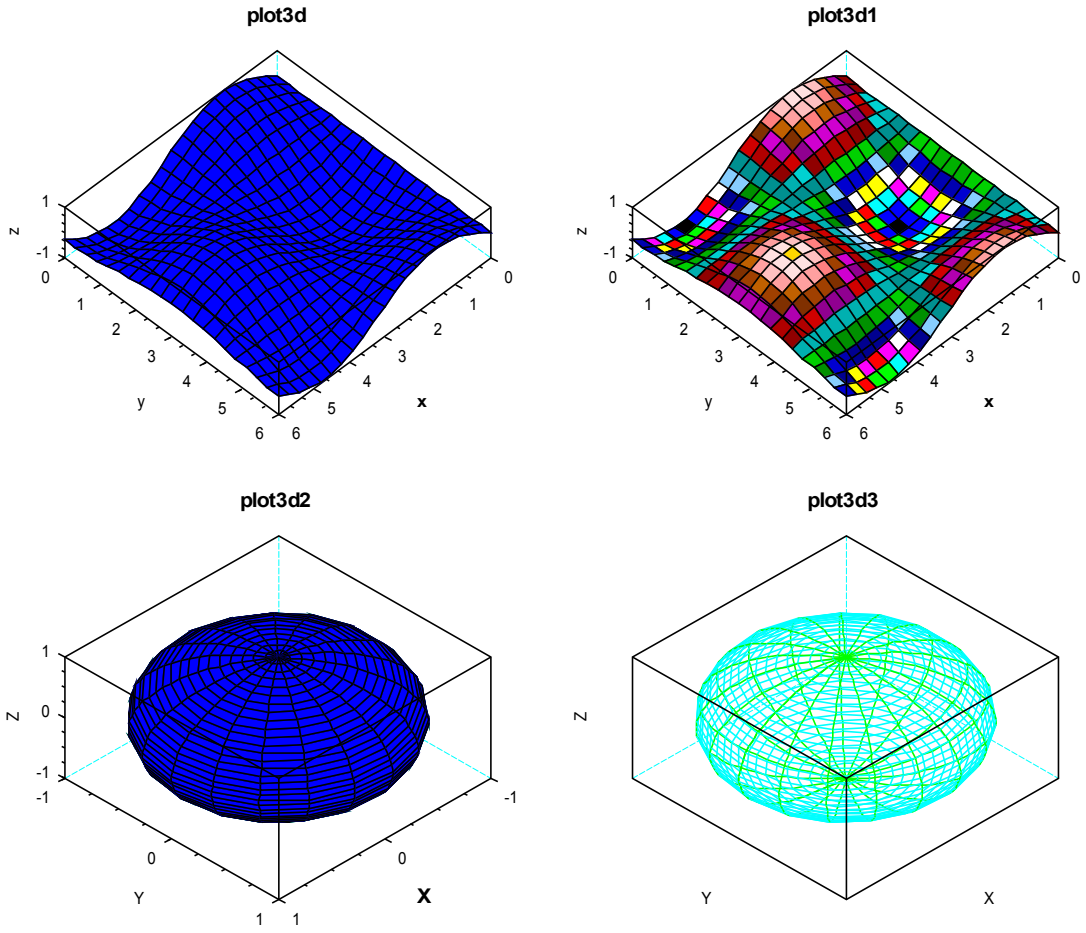


Figura 4.16. Gráficos a 3D básicos.

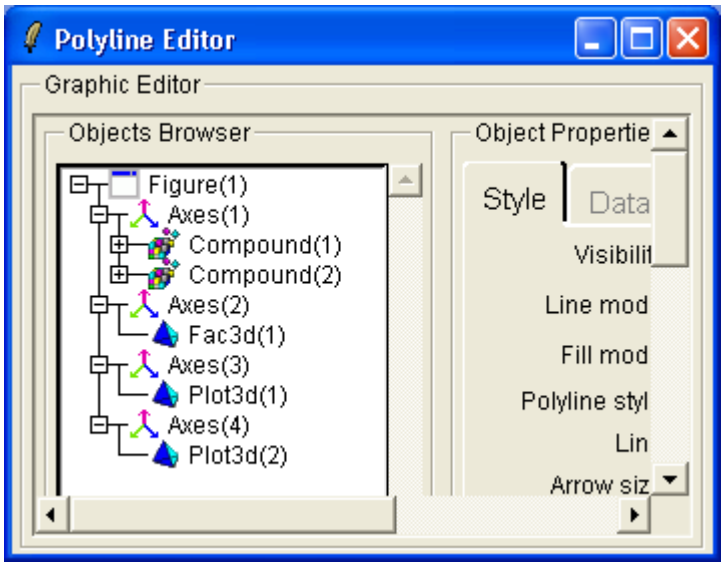


Figura 4.17. O navegador no gráfico com os "ramos" "Áxis" abertos.

Como fizemos para os gráficos 2D, sugerimos que a leitora explore as possibilidades do editor de gráficos com o gráfico da figura 16. Sem preocupações de estética nem de design gráfico, mais uma vez para lhe estimular a iniciativa, deixamos-lhe a figura 4.19, resultado de um exercício de formatação do gráfico em questão.

A "Hidden color" corresponde à face inferior da superfície criada, se esta tiver porções visíveis.

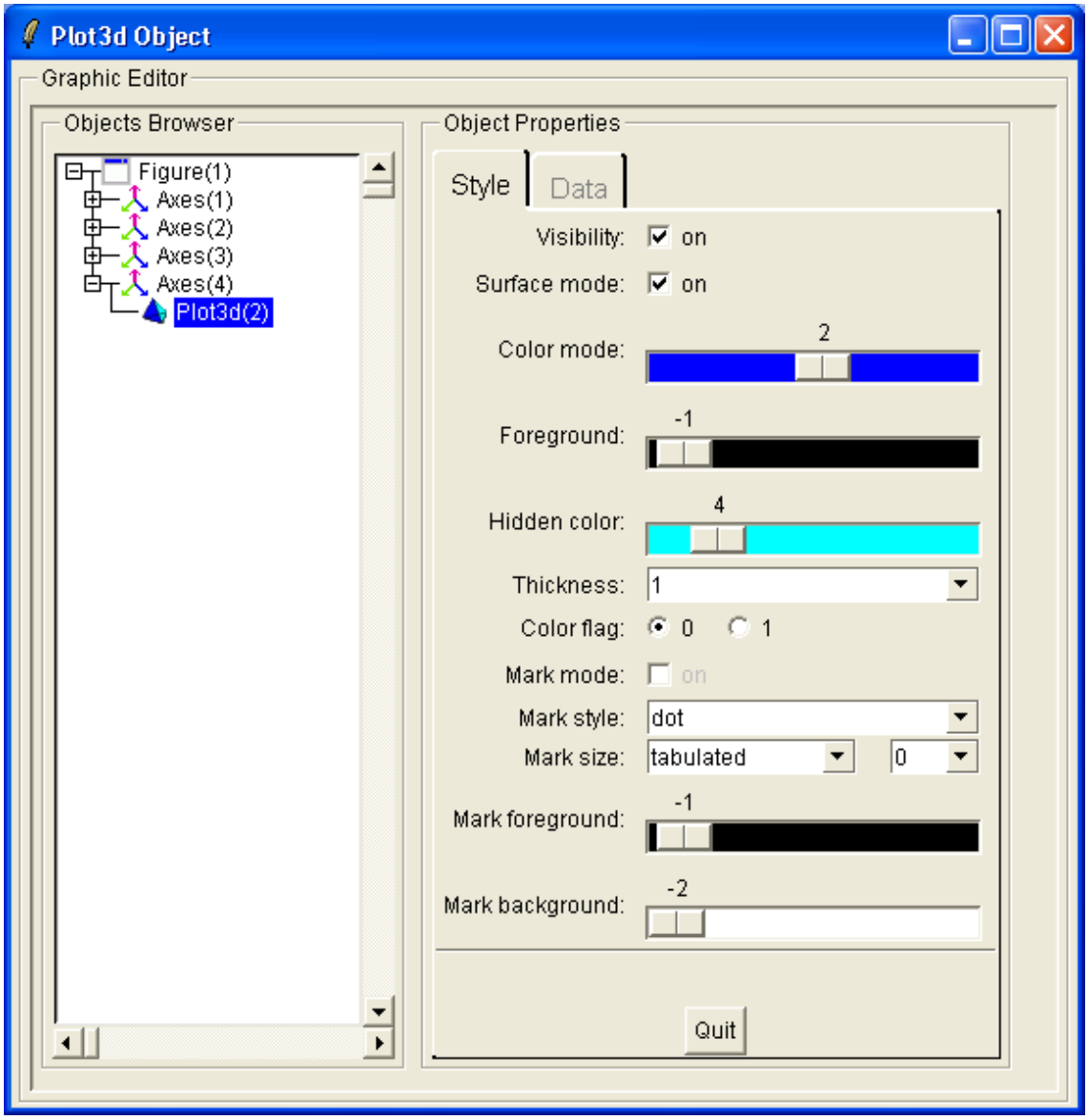


Figura 4.18. Editor do Plot3d do gráfico 4, da figura 4.16.

.

Sugiro ao leitor que copie da ajuda do Scilab para o comando mesh, o exemplo que lá vem, e execute-o, na janela de comandos. Depois abra ao editor do gráfico e utilize uma cor diferente da branca para a página inferior da superfície (deslizante da "Hidden color").

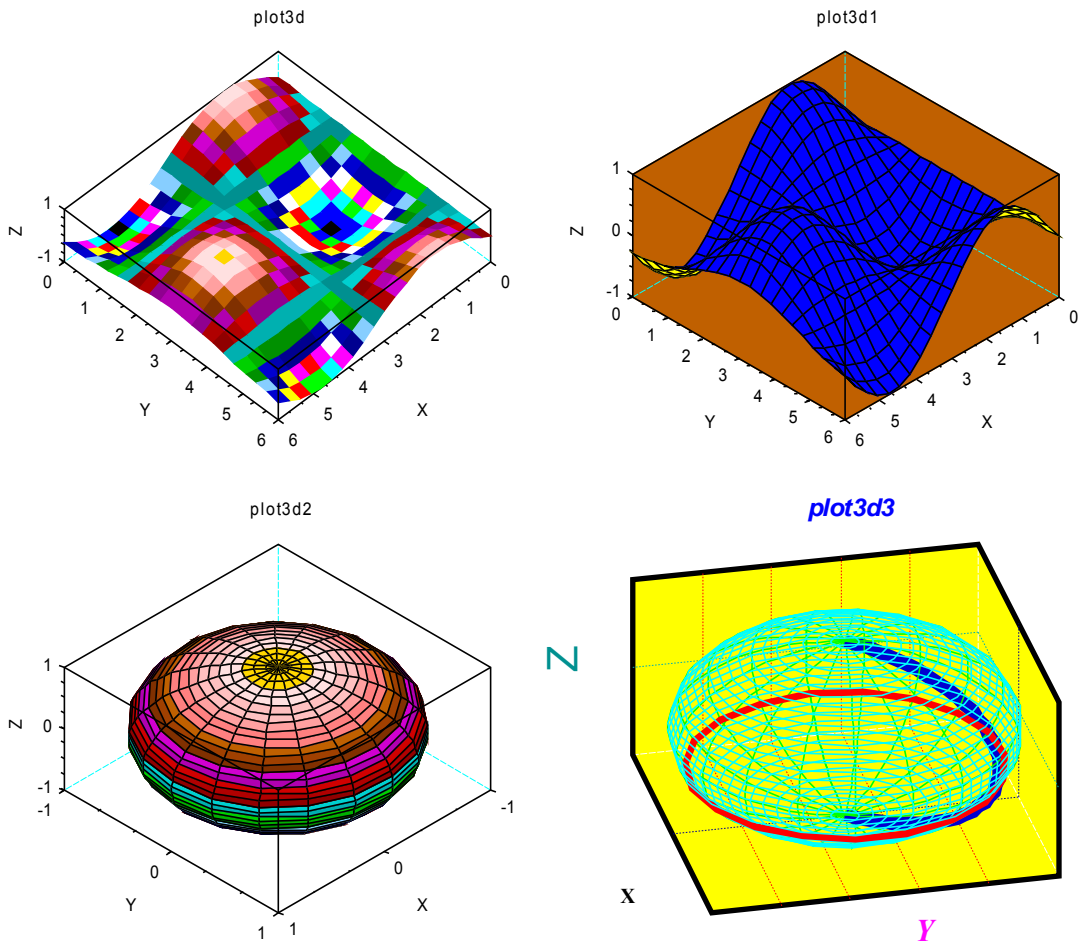


Figura 4.19. Os gráficos da figura 4.16 depois de submetidos a alguns actos de formatação usando o editor de gráficos

4.9 Outras possibilidades de criar facilmente gráficos 3D

À semelhança do que se passa com os gráficos 2D, o Scilab ainda apresenta outros comandos para se obterem gráficos 3D. Eis alguns:

fplot3d : faz um gráfico 3D de uma superfície definida por uma função
fplot3d1 : faz um gráfico 3D de uma superfície definida por uma função com níveis de cizento ou cor
param3d : faz o gráfico de uma curva
param3d1 faz o gráfico de curvas
contour : insere curves de nível num gráfico 3D
fcontour : insere curvas de nível no gráfico de uma superfície definida por uma função
hist3d : histogram em 3D

4.10 Quadro sinóptico dos comandos introduzidos neste capítulo

Comando	Descrição
champ	Cria um campo de vectores de duas dimensões
champ1	Permite colorir os vectores do campo
contour2d	Traça as curvas de nível de uma superfície num gráfico 2D
errbar	Acrescenta baras verticais de erros a um gráfico 2D
fchamp	Campo de vectores associado a uma equação diferencial ordinária (EDO)
fcontour2d	Traça as curvas de nível de uma superfície definida por uma função num gráfico 2D
fgrayplot	Cria uma superfície definida por uma funçã num gráfico a duas dimensões usando cores
fplot2d	Faz o gréfico de uma curva definida por uma função.
grayplot	Cria uma superfície num gráfico a duas dimensões usando cores
legend	Insere uma legenda com o nome das curvas do gráfico, no espaço dos eixos
locate	Permite obter as coordenadas de pontos numa curva
matplot	Cria uma quadricula preenchida a cores
mesh	Faz um gráfico 3D definido por uma rede
pie	Cria um gáfico circular, como se ilustra na figura 4.13
plot2d	Cria um gráfico a duas dimensões
plot3d	Faz um gráfico de uma superfície 3D
plot3d1	Faz um gráfico 3D de uma superfície cinzento ou com níveis a cores
plot3d2	Faz um gráfico 3D de uma superfície definida por rectângulos
plot3d3	Faz um gráfico 3D de uma superfície em rede ("mesh") definida por rectângulos
portr3d	Um gráfico de fase a 3D
sfgrayplot	Suaviza uma superfície definida por uma função num gráfico a duas dimensões usando cores.
sgrayplot	Suaviza uma superfície num gráfico a duas dimensões usando cores, como se ilutra na figura 4.14
subplot	Cria mais de um gráfico nua janela
xarc	Desenha parte de uma elipse
xarcs	Desenha partes de um conjunto de elipses
xfarc	Preenche parte de uma elipse
xfpoly	Preenche um polígono
xfpolys	Preenche um conjunto de polígonos
xfrect	Preenche um rectângulo
xpoly	Desenha uma linha ou um polígono
xpolys	Desenha um conjunto de linhas ou polígonos
xrect	Desenha um rectângulo
xrects	Desenha ou preenche um conjunto de rectângulos
xrpoly	Desenha um polígono regular
xsegs	Desenha segmentos de recta não ligados
xstrig	Insere texto no espaço dos eixos, num gráfico

Capítulo 5

Outros comandos com interesse

Do rico acervo de recursos disponibilizados pelo Scilab, vamos ainda abordar mais alguns, tidos como entre os mais frequentemente utilizados. Assim, introduziremos comandos concernentes aos seguintes tópicos:

- Ajustamento de curvas não lineares
- Interpolação
- Diferenciação
- Integração
- Sistemas de equações não lineares
- Solução numérica de equações diferenciais ordinárias
- Equações discretas
- Optimização

5.1 Ajustar curvas

O Scilab apresenta várias possibilidades para se ajustar curvas. Para além do comando lsq (ver secção 2.8) tem outros comandos ligados ao método dos mínimos quadrados, cuja exploração deixamos à leitora, salvo o caso ilustrado abaixo. Abordemos o comando datafit, que é uma versão melhorada do fit_dat.

[vector p dos parâmetros desejados, erro mínimo quadrado, err] = datafit(*função a minimizar G que tem de ser criada*, *conjunto dos valores medidos Z, tentativa de uma solução inicial,p0, opções*)

Antes de utilizar o comando deve:

- Definir a curva que quer ajustar, usando o comando function (secção 2.7) ou deff (secção 2.14).
- Dispor do vector das variáveis independente e da variável dependente (dados observados ou medidos).
- Definir a função G(p,z) que é utilizada para obter o melhor vector dos parâmetros pretendidos, por aproximações de G(p,z_i)=0 , para um conjunto de valores medidos z_i. O vector p é encontrado, minimizando a expressão
G(p,z_1)'WG(p,z_1)+G(p,z_2)'WG(p,z_2)+...+G(p,z_n)'WG(p,z_n)
Em que W é um vector opcional de ponderação dos parâmetros.
- Por fim, resolver o problema com o comando [p,err]=datafit(G,Z,p0).

As experiências com *Paramecium* sp. realizadas por Gause , nos anos trinta do século passado são referidas em todos os textos de ecologia que abordam a competição. Gause tanto fez crescer misturas de dois ciliados, como as suas culturas puras. Determinou diariamente, o número de organismos em 0,5 cm^3. Vamos ajustar aos valores contados para uma das suas culturas puras de *P. aurélia*, uma equação de Gompertz da forma:

$$y(t) = y_f\left(\frac{y_0}{y_f}\right)^{\exp(-ct)}$$

pretendemos estimar y_f que fazemos igual a a e c que fazemos igual a b, ao definir a função no Scilab.

O programa para ajustar a curva aos dados de Gompertz é o seguinte:

```
//Ajustar uma curva de Gompertz
 //Ao crescimento de uma cultura de Paramecium aurelia
 //Dados de Gause
 deff('y=FF(x)','y=a*(Y(1)/a)^exp(-b*x)') //definir a função de Gompertz
 X=[];Y=[]; //criar dois vectores vazios
 x=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19];//dias da contagem
 Y=[2 10 17 29 39 63 185 258 267 392...
 510 570 650 560 575 650 550 480 520 500];//número de paramecias em 0,5
c.c.
 X=[X,x];//Recriar o vector X
 Z=[Y;X];//Criar o vector Z
 //Definir a função critério
 deff('e=G(p,z)','a=p(1),b=p(2),y=z(1),x=z(2),e=y-FF(x)')
```

```
//Obter o resultado com os valores propostos 500 e 0.2
[p,err]=datafit(G,Z,[500;0.2])
//Calcular a razão entre o valor inicial e final
R=Y(1)/p(1)
//Simular dados com a curva ajustada
t=0:1:19;
z=p(1)*R^exp(-p(2)*t);
//Calcular o vector dos erros absolutos percentuais
e=(abs(z-Y)./Y)*100;
//Calcular o erro absoluto percentual médio
eapm=mean(e)
//Criar a matriz dos valores medidos e simulados
M=[z;Y];
//Fazer o gráfico dos valores medidos e simulados
plot2d(x,[M]')
xtitle("Ajustamento da curva de Gompertz","t, dias","z,Y. Nº em meio c.c.")
xgrid()
```

A saída do programa é a seguinte:

```
err  =

   88069.143
p  =

   606.03349
   0.2888947
R  =

   0.0033001
eapm  =

   32.406001

exec done
```

Tendo-se pois y_f=606,03349 e c=0,2888947. O gráfico obtido consta da figura 5.1, tendo sofrido formatação com a interface gráfica.

O leitor deve consultar o texto de ajuda (help datafit) para obter informação sobre as opções.

Para efeitos comparativos vamos introduzir o comando lsqrsolve que usa o algoritmo modeficado de Levenberg-Marquardt para ajustar uma curva não linear.

[vector dos parametros pretendidos, valores da função usada no algoritmo]=lsqrsolve([*tentativa de solução inicial*],*função a criar, size(X,1)*);

Inserimos este comando aos mesmos dados de Gause, nos termos da aplicação anterior. Ver também a figura 5.2. Para mais pormenores consultar a ajuda do comando.

```
//Ajustar uma curva de Gompertz
//Ao crescimento de uma cultura de Paramecium aurelia
//usando o commando lsqrsolve. Dados de Gause
deff('y=FF(X)','y=a*(Y(1)/a)^eXp(-b*X)');
 X=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]';//dias da contagem
 Y=[2 10 17 29 39 63 185 258 267 392...
 510 570 650 560 575 650 550 480 520 500];//nº de ciliados/0,5 c.c.
//solução
function e=f1(ab,m)
  a=ab(1);b=ab(2),
  e=Y-a*(Y(1)/a)^exp(-b*X)';
endfunction
[ab,v]=lsqrsolve([600;0.2],f1,size(X,1));
ab
 //Calcular a razão entre o valor inicial e final
 R=Y(1)/ab(1)
 //Simular dados com a curva ajustada
 t=0:1:19;
 z=ab(1)*R^exp(-ab(2)*t);
 //Calcular o vector dos erros absolutos percentuais
 e=(abs(z-Y)./Y)*100;
 //Calcular o erro absoluto percentual médio
 eapm=mean(e)
 //Criar a matriz dos valores medidos e simulados
 M=[z;Y];
 //Fazer o gráfico dos valores medidos e simulados
 plot2d(X,[M]')
 xtitle("Ajustamento da curva de Gompertz usando lsqrsolve","t, dias","z,Y")
 xgrid()

ab  =

   606.03612
   0.2888923
 R  =

   0.0033001
 eapm  =

   32.40536

 exec done
```

Os dois ajustamentos são virtualmente idênticos.

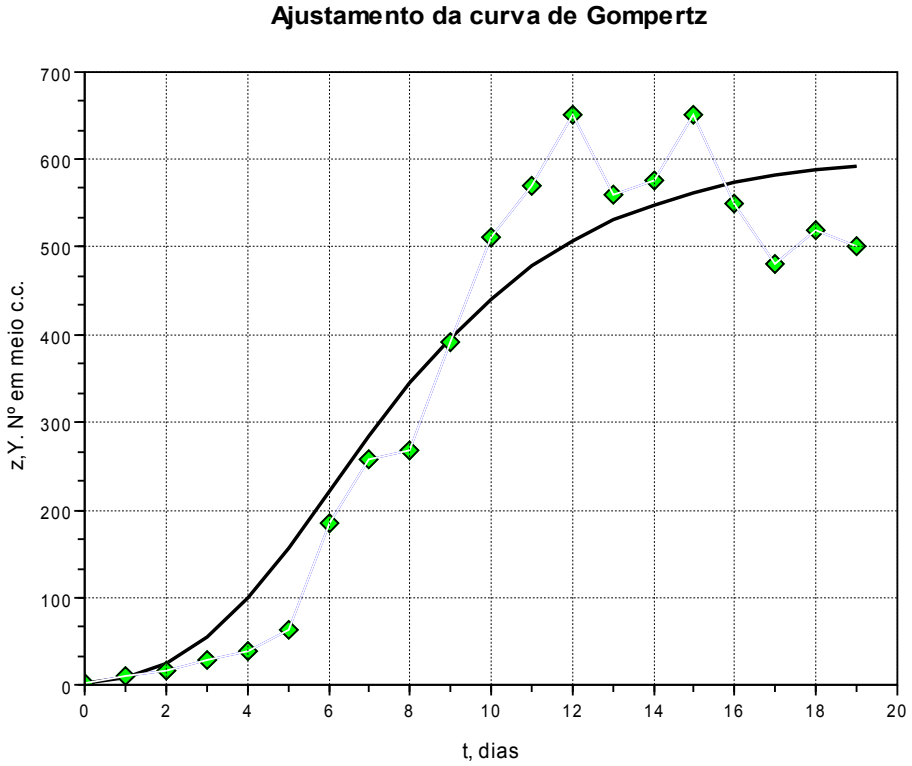


Figura 5.1. Aplicação do comando datafit. Valores medidos e simulados pelo ajustamento de uma curva de Gompertz a dados de Gause, de uma cultura pura de *Paramecium aurélia*

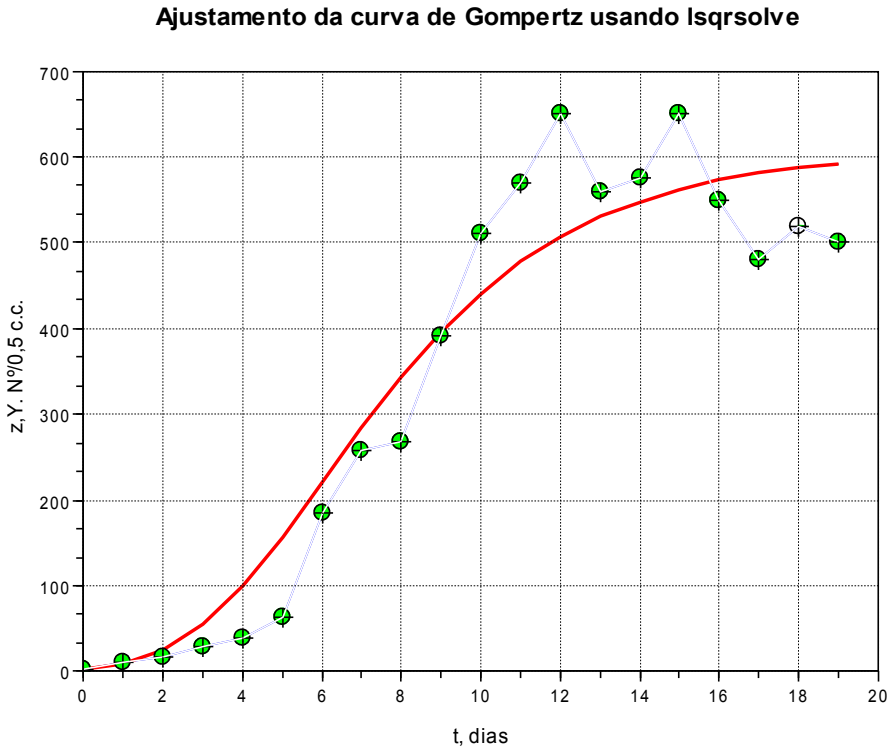


Figura 5.2. Aplicação do comando lsqrsolve. Valores medidos e simulados pelo ajustamento de uma curva de Gompertz a dados de Gause, de uma cultura pura de *Paramecium aurelia*

5.2 Interpolação

Todos estamos familiarizados com problemas de interpolação, quando temos valores de duas variáveis tabelados, e queremos achar estimativas de valores intermédios. Sejam as variáveis x e y. Geralmente temos um valor de x que não vem na tabela e cai entre dois tabelados, e queremos o valor de y associado. Os valores de x têm de ser todos crescentes ou decrescentes (variar monotonicamente). É um problema diferente de ajustar uma curva porque queremos que a linha estimada passe forçosamente pelos pontos disponíveis.

Existem vários métodos de interpolação e o Scilab é generosos na sua apresentação. Se a leitora os quiser cotejar todos, entre com apropos interpolation e obterá uma lista de 12 comandos, no navegador da janela da Ajuda (Browse Help). Nós vamos abordar o comando interp1.

Valores pretendidos interpolados = interp1(*vector dos valores de x disponíveis, vector dos valores de y disponíveis, vector dos valores de x para os quais se quer interpolar y,* '*método seleccionado*')

Vamos ilustrar uma aplicação usando os três métodos do comando, e usando a equação de Gompertz ajustada na secção anterior. A lista de comandos seguinte encarrega-se dessa tarefa.

```
//Ensaiar os métodos de interpolação do comando interp1
//Criar cinco valores de x
x=linspace(0,19,5);
//Criar os cinco valores corresponedntes de y
y= 606.03612*0.0033001^exp(-0.2888923*x);xx=linspace(0,20,21);
//Criar o vector de x para os quais se querem as interpolações
xx=linspace(0,19,20);
//Aplicar o método de interpolação linear
yy1=interp1(x,y,xx,'linear');
//Aplicar o método spline
yy2=interp1(x,y,xx,'spline');
//Aplicar o método nearest
yy3=interp1(x,y,xx,'nearest');
//Criar uma matriz de três gráficos
subplot(2,2,1)
plot(xx,[yy3],x,y,'*')
xtitle("Interpolação com o método nearest","x","y")
subplot(2,2,2)
plot(xx,[yy1],x,y,'*')
xtitle("Interpolação com o método linear","x","y")
subplot(2,2,3)
plot(xx,[yy2],x,y,'*')
xtitle("Interpolação com o método spline","x","y")
```

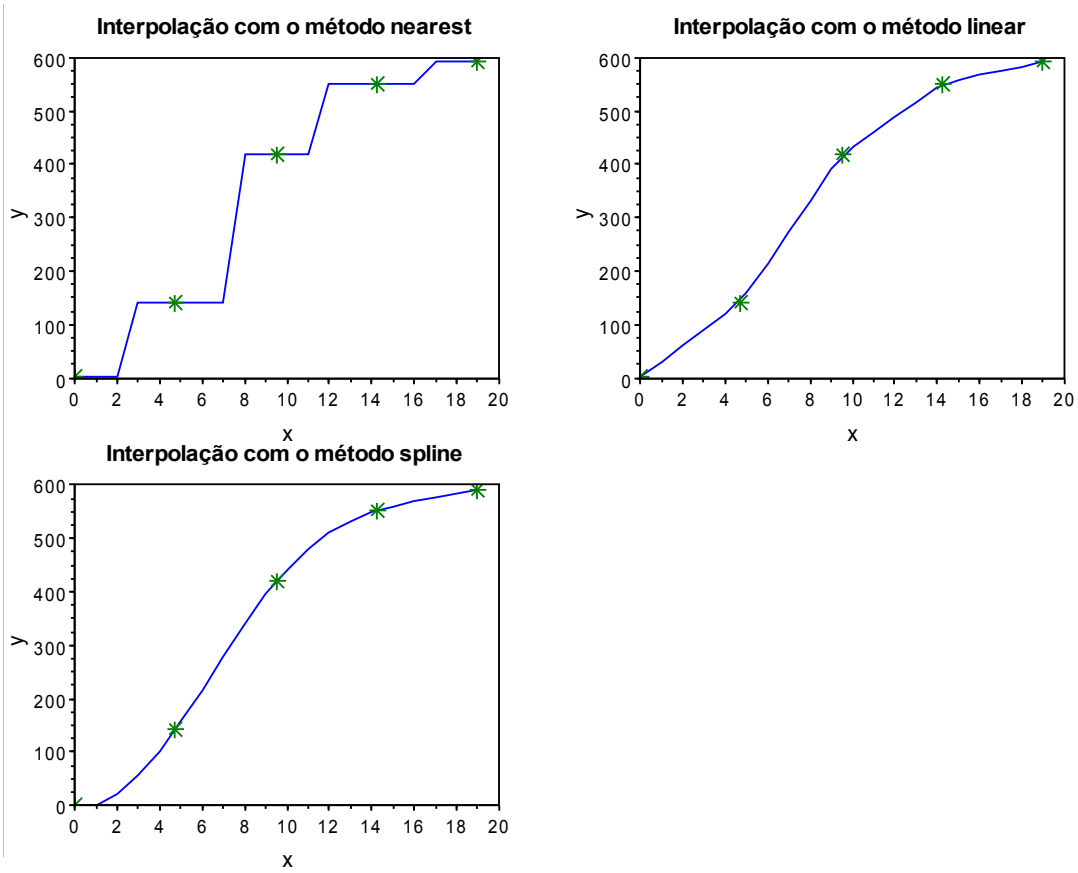


Figura 5.3. Ilustração do uso do comando interp1, com os seus três métodos. Gráfico com as legendas formatadas com o editor gráfico

O método spline surge como o de melhores resultados. Fica como exercício para o leitor, calcular as diferenças entre os valores dados pela equação ajustada e os proporcionados pela interpolação.

A interpolação que realizámos é dita linear poque y só depende de uma variável x. Mencionemos ainda mais dois comandos para interpolação não linear:

O comando interp2d permite interpolar os valores de uma variável z que depende de outras duas ($z_j=f(x_j,y_j)$) usando o método spline.

O comando interp3d permite interpolar os valores de uma variável v que depende de outras três ($v_j=f(x_j,y_j,z_j)$) usando o método spline.

Outro comando que usa o método spline para interpolar é smooth.

Vector interpolado=smooth([pontos x e y], salto discreto)

Vamos interpolar valores do seno de oito ângulos e depois interpolar a curva cm smooth.

-->//Interpolação smooth.

-->//Imprime um gráfico de rectas com os dados iniciais,

-->// da curva sinusoidal

-->//depois a curva da interpolação smooth

```
--> xbasc()

--> x=linspace(0,2*%pi,8);//8 valores de x

--> y=sin(x);//8 valores de y

--> plot2d(x,[y]')//fazer um gráfico dos 8 pontos

--> yi=smooth([x;y],0.1);//interpolação

--> plot2d(yi(1,:)',yi(2,:)')//fazer um gráfico da interpolação

--> z=locate(5,1)//identificar 5 pontos
 z  =

   0.3480874    1.6180328    3.2398907    4.7087432    6.2693989
   0.3731020    1.0036153  - 0.1012292  - 1.0036153  - 0.0086768
```

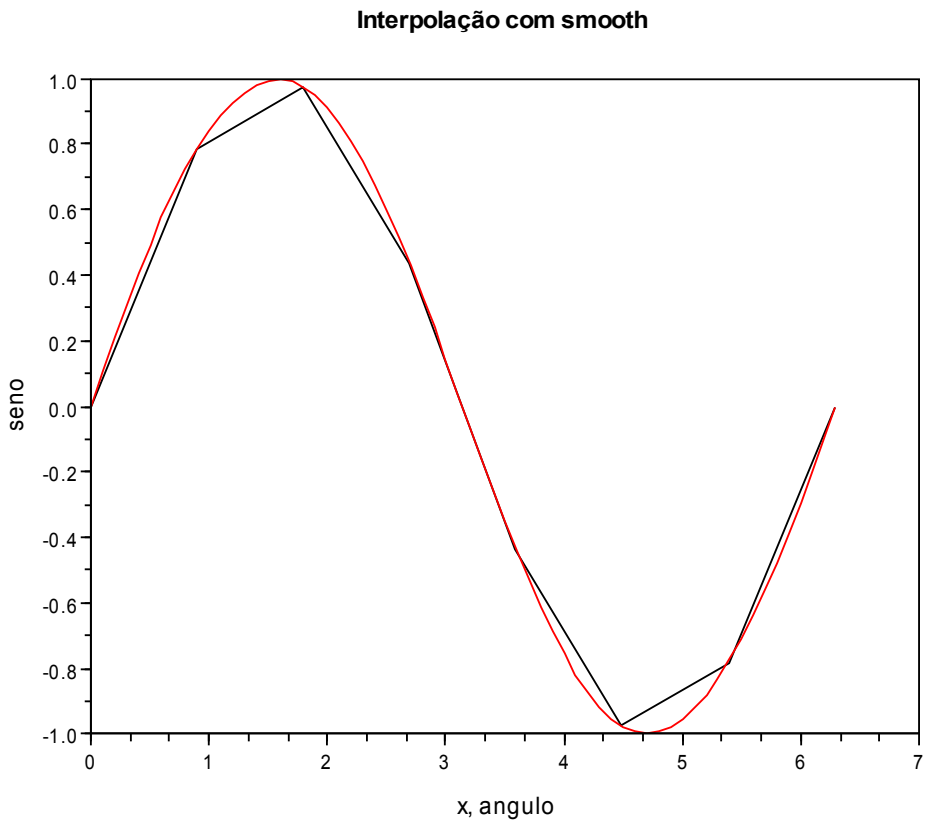


Figura 5.4. Interpolação da curva do seno com o comando smooth. Formatado com editor gráfico

5.3 Diferenciação

O comando derivative dá o valor da derivada em ordem a uma dada variável, de uma função num dado ponto.

Para usar este comando temos primeiro que definir a função (ver secção 2.14), com o comendo deff. Depois estabelecer o ponto onde queremos a derivada (valores das variáveis independentes). Depois chamar o comando que na sua forma mais simples tem a forma;

[Valor da derivada, J]=derivative(*função definida, ponto escolhido*)

Ilustremos com o caso de funções de uma e duas variáveis:

```
//Derivada de uma função de 1 variável
deff('y=f(x)','y=[0.05*x(1)*cos(x(1))^2+cos(x(1))]')
x(1)=3*%pi/4;
x0=[x(1)];
disp("Derivada")
[J]=derivative(f,x0)
//derivada de uma função de duas variáveis
deff('y=f(x)','y=[0.05*x(1)*cos(x(2))^2+cos(x(1))]')
x(1)=3*%pi/4;x(2)=1.5*%pi/4;
x0=[x(1),x(2)]';
disp("Derivada")
[J]=derivative(f,x0)
```

A saída é a seguinte:

```
Derivada
 J  =

  - 0.5642971
Warning :redefining function: f


 Derivada
 J  =

  - 0.6997845  - 0.0833041

 exec done
```

Use a ajuda do Scilab se pretender mais informação sobre o comando.

.

5.4 Integração

Existem dois comandos que calculam o integral definido de uma função. Vejamos o de utilização mais simples.

integrate(*'função','variável',limite inferior,limite superior*)

Vejamos um exemplo.

```
-->integrate('-0.05*log(0.4076)*exp(-0.05*x)','x',0,50)
 ans  =

   0.8238002
```

O comando intg requer a definição prévia da função. A sintaxe é a seguinte:

intg(*limite inferior,limite superior, nome da função*)

Vejamos a sua utilização:

```
-->deff('[y]=f(x)','y=-0.05*log(0.4076)*exp(-0.05*x)')

-->intg(0,50,f)
 ans  =

   0.8238002
```

Proporcionam o mesmo resultado.

É também possível calcular o integral de um vector de dados procedendo primeiro à interpolação da sua curva. É o que fazem os comandos intsplin que usa a interpolação pelo método spline, e o inttrap que emprega a interpolação trapezoidal.

Comecemos pela última:

inttrap(*vector dos pontos disponíveis, função*)

Calculemos o mesmo intergral a partir de um vector de 10 pontos. Teremos:

```
-->x=linspace(0,50,10);

-->inttrap(x,-0.05*log(0.4076)*exp(-0.05*x))
 ans  =

   0.8290905
```

A sintaxe do comando intsplin é semelhante, e fornece melhor resultado

```
-->x=linspace(0,50,10);

-->intsplin(x,-0.05*log(0.4076)*exp(-0.05*x))
```

```
 ans  =

    0.8238128
```

5.5 Solução de sistemas de equações não lineares

O comando fsolve permite achar a solução de um sistema de n equações não lineares.

[solução]=fsolve([*tentative de valores iniciais*],*nome da função*)

Eis um exemplo, recorrendo às equações diferenciais de um sistema presa-predador. Quando as equações são iguais a zero não há variação e os tamanhos das populações da presa e do predador mantêm-se constantes, como se ilustra na figura 5.5, que confirma a solução.

```
-->function x=f(y)
-->x(1)=y(1)*(1-y(1)/1000)-0.04*y(1)*y(2)
-->x(2)=0.03*0.02*y(1)*y(2)-0.25*y(2)
-->endfunction
-->[y]=fsolve([450,20],f)
 y  =

    416.66667    14.583333
```

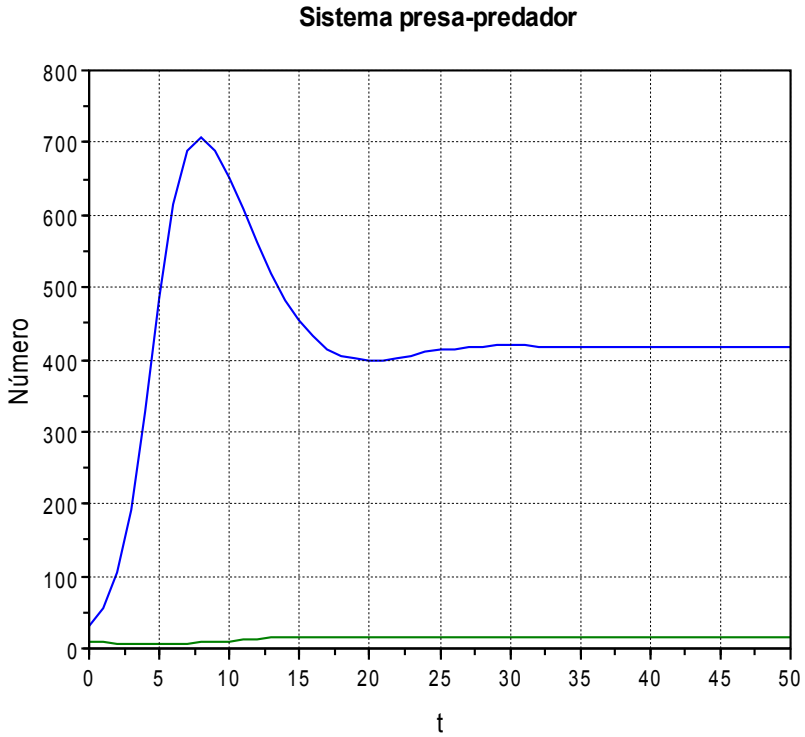


Figura 5.5. Dinâmica de um sistema presa-predador. A população da presa no equilíbrio é de 417 indivíduos e a do predador 14.

5.6 Solução numérica de equações diferenciais ordinárias

O comando do Scilab para resolver equações diferenciais ordinárias (EDO; é o ode (.ode=ordinary differential equations):

Antes de usar o comando temos de:

1 – Definir a equação diferencial ordinária ou um sistema desta equações.
2 – Estabelecer o valor ou valores iniciais.
3 – Definir o início do tempo, t0
4 – Definir o período da solução, vector t
5 – Usar o comando ode para obter a solução

O comando ode na sua forma mais simples, a única que abordaremos nesta iniciação ao software, é:

Solução=ode(*vector dos valores iniciais, início do tempo, período de integração, nome da função*)

Damos um exemplo que é resolver a equação diferencial da curva de Gompertz:

$$\frac{dy}{dt} = 0{,}5y \ln 250\left(1 - \frac{\ln y}{\ln 250}\right)$$

```
-->// Solução da equação diferencial da curva de Gompertz

-->xbasc()

-->//Definir a equação diferencial

-->deff("yprim=f(t,y)",["yprim=0.5*y*log(250)*(1-log(y)/log(250))"])

-->y0=210;//Valor inicial

-->t0=0;//Início do tempo

-->t=0:15;//Período da solução

-->y=ode(y0,t0,t,f)' //solução
 y  =

    210.
    224.9122
    234.46814
    240.46087
    244.17003
    246.44753
    247.83925
    248.6872
    249.20293
```

```
 249.51625
 249.70648
 249.82193
 249.89198
 249.93447
 249.96025
 249.97589
```

```
-->plot(ode(y0,t0,t,f)) //Gráfico da solução
-->xtitle("Equação de Gompertz","t","y")
```

O gráfico da solução insere-se na figura 5.6.

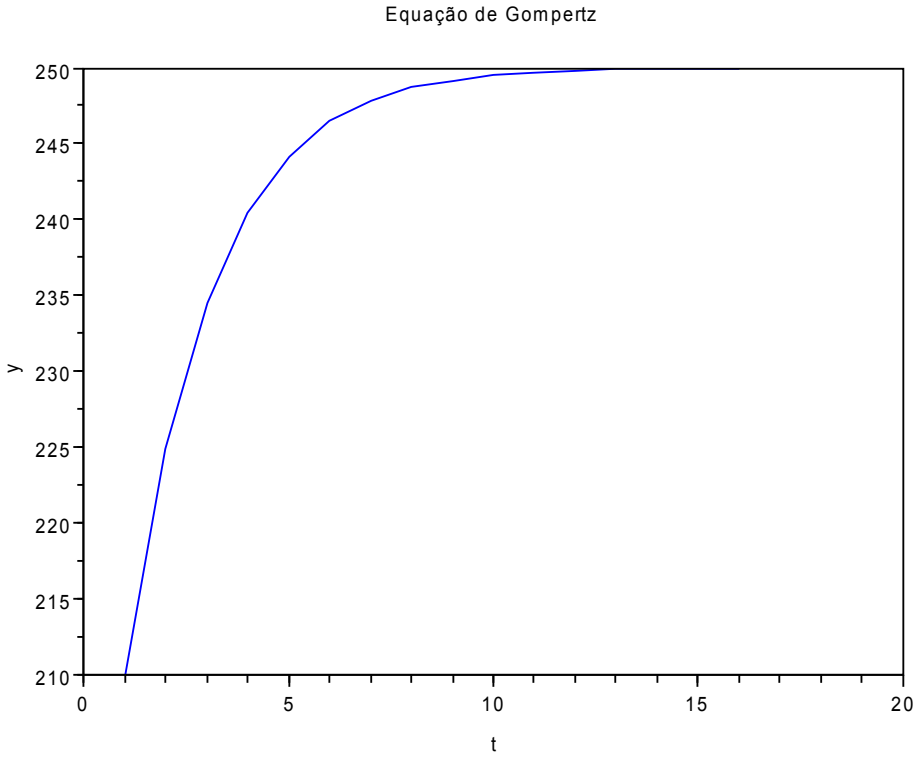


Figura 5.6. Solução da equação diferencial da curva de Gompertz, com o comando ode.

Desejamos agora a solução do sistema de duas equações diferenciais seguinte.

$$\frac{dy_1}{dt} = -0{,}1y_1 + 0{,}003y_1y_2$$

$$\frac{dy_2}{dt} = -0{,}0003y_1y_2 + 0{,}3y_2$$

A solução é dada pela seguinte listagem:

```
//Modelo de duas EDO com ciclo
xbasc()
//Definir o sistema
deff("ydot=f(t,y)",["ydot1=-0.1*y(1)+0.003*y(1)*y(2)";...
"ydot2=-0.0003*y(1)*y(2)+0.3*y(2)";...
"ydot=[ydot1;ydot2]"])
//valores iniciais de y(1) e y(2)
y0=[10,10];
//instante inicial
t0=0;
//Perídos da simulação
t=0:500;
//Resolver
[M]=(matrix(ode(y0,t0,t,f),2,501))';//Obter 2 vectores coluna da solução
//Gráfico da solução
subplot(2,1,1)
plot(t,[M]');
xtitle("Solução de um sistema de duas EDO","t","y(1),y(2)")
//Diagrama de fase
subplot(2,1,2)
plot2d(M(:,1),[M(:,2)])
xtitle("Diagrama de fase","y(1)","y(2)")
```

Transpor a matriz

O gráfico da solução e o diagrama de fase exibe-se na figura 5.7.
Também podíamos ter entrado com:

```
plot(ode(y0,t0,t,f))
```

e obtínhamos o gráfico da figura 5.8.

O comando ode permite resolver equações de diferenças. Vamos usar a equação de diferenças da curva logística. A sintaxe do comando é a seguinte:

Solução=ode("discrete",*vector dos valores iniciais, início do tempo, período de integração, nome da função*)

```
-->// solução da equação logística discreta

-->deff("yprim=f(t,y)",["yprim=1.6487*y/(1+0.00342*y)"])

-->y0=31;

-->t0=0;

-->t=0:15;

-->plot(ode("discrete",y0,t0,t,f))

-->xtitle("Curva logística","t","y")
```

Ver a figura 5.9.

Em Barreto (2005, eq. (11.2)) insiro o meu modelo BACO3, que tem a expressão seguinte:

$$\frac{dy_i}{dt} = c_i y_i \ln(y_{fi})(1 - \frac{\ln(y_i)}{\ln(y_{fi})} - \sum_{j=1}^{n} \alpha_{ij} \frac{\ln(y_j)}{\ln(y_{fi})})$$

Proponho à leitora o seguinte exercício de programação e simulação. Usar a equação geral apresentada para estabelecer um modelo de três competidores, i=1,2,3, com os seguintes parâmetros: c_1=0,5; c_2=0,6; c_3=0,3; y_{f1}=2,718282; y_{f2}=1,6059; y_{f3}=1.234327;α_{12}=2; α_{13}=4; α_{21}=1/3; α_{23}=2; α_{31}=1/3; α_{32}=1/3. Faça y0=[1 1 1]. Deverá obter um gráfico semelhante ao da figura 5.10.

Tente traçar o diagrama de fase recorrendo ao comando portr3d.

O leitor deve explorar as opções do comando ode e outros que estão relacionados com ele, se este tópico é do seu particular interesse (apropos ode).

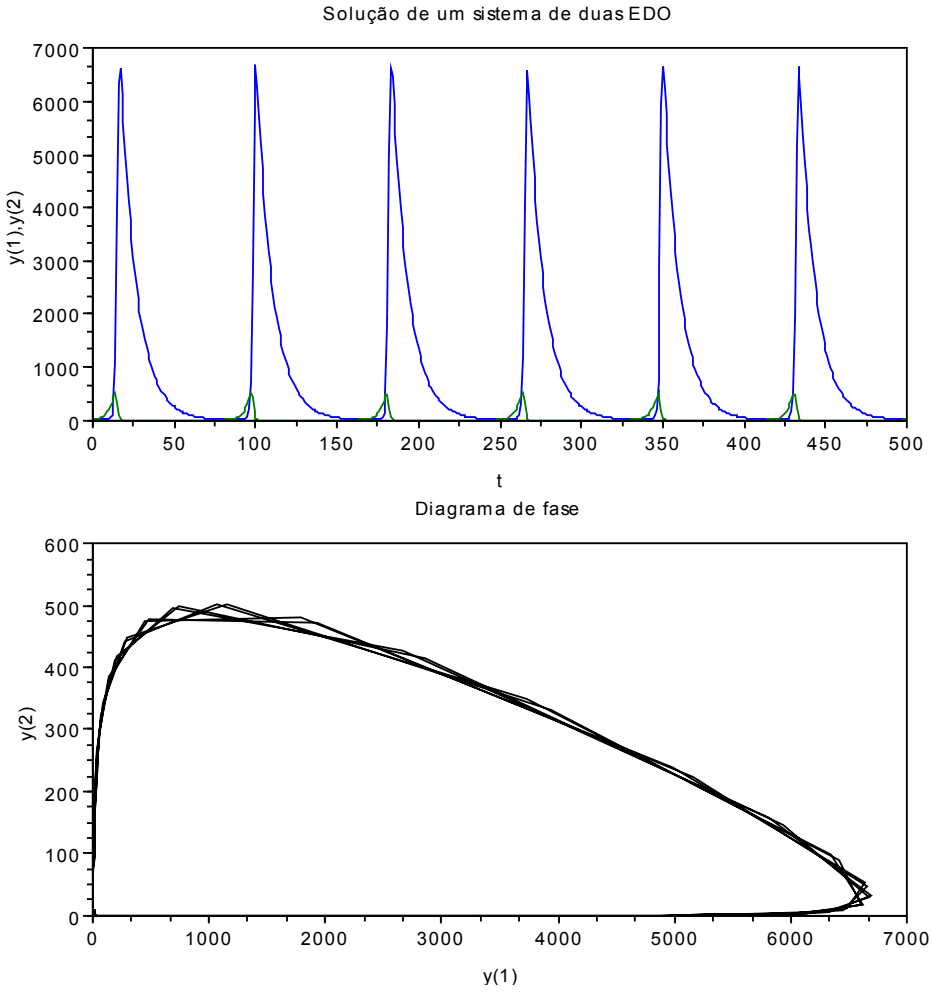


Figura 5.7. Gráficos relativos à solução do sistema de duas equações diferenciais.

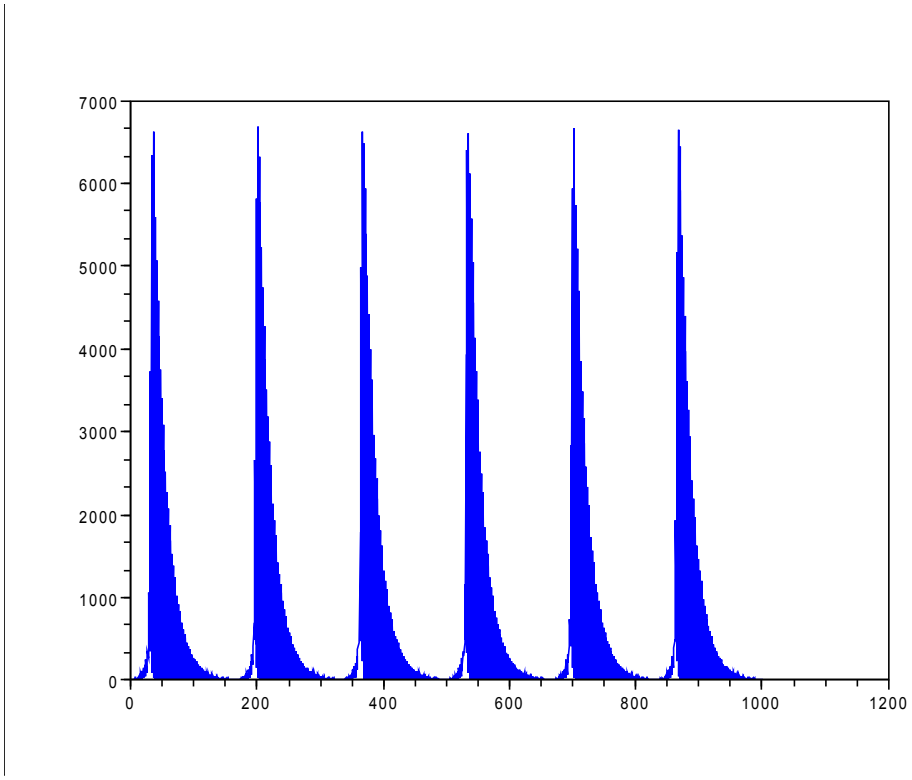


Figura 5.8. Gráfico da solução do sistema de duas EDO, criado pelo comando plot(ode(y0,t0,t,f))

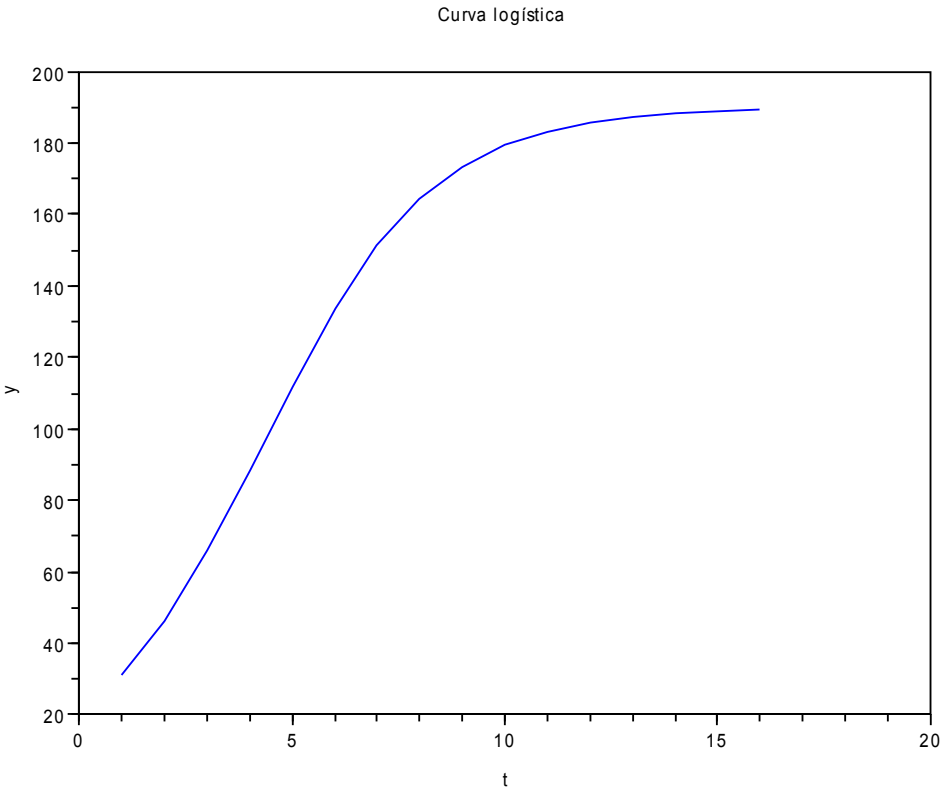


Figura 5.9. Solução da equação de diferenças da curva logística

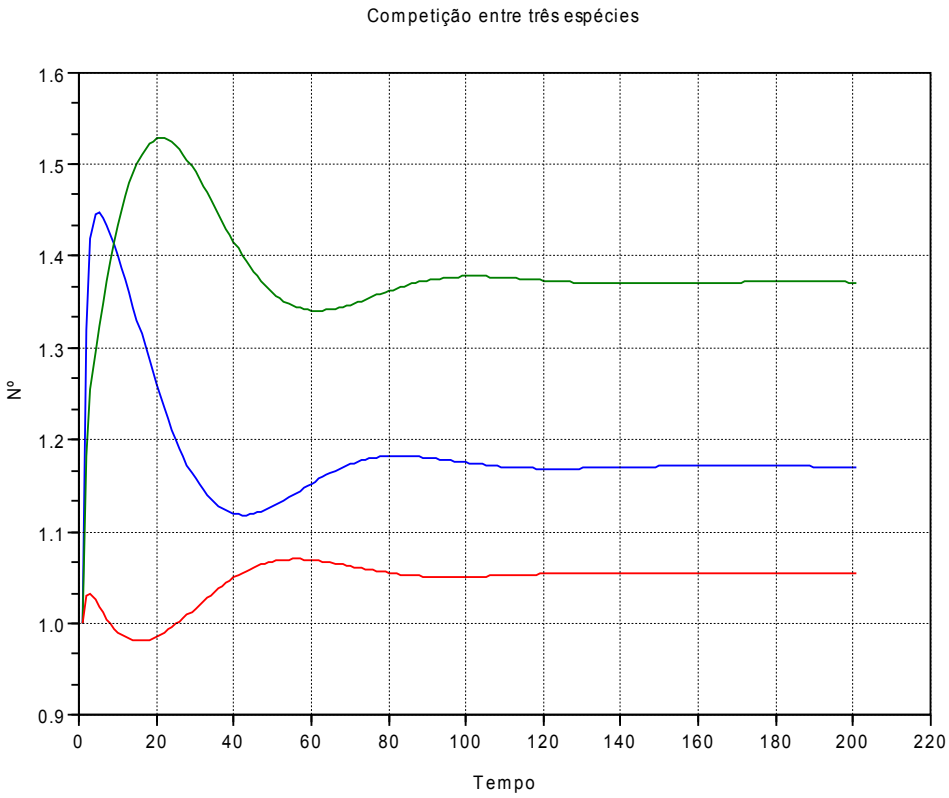


Figura 5.10. Dinâmica da competição entre três espécies, obtida com o modelo BACO3. Para os valores dos parâmetros ver o texto.

5.7 Optimização

Vamos aqui abordar os comandos: optim, linpro, quapro.

optim acha o mínimo de uma função, que antes tem de ser definida, sem sujeição a restrições. Definamos a função:

deff('[*f,g,ind*]=cost(*x,ind*)','f=expressão da função,g=gradiente');

Activar o commando:

[*f,xopt*]=optim(*cost, tentativa de solução proposta proposta*)

f =valor da função no mínimo, xopt=valor de x no mínimo
Vejamos um exemplo:

```
-->deff('[f,g,ind]=cost(x,ind)','f=3-2*x+2*x^2,g=4*x-2');//definir a função

-->x0=0.7; //valor inicial proposto

-->[f,xopt]=optim(cost,x0)
```

```
 xopt  =

   0.5
 f  =

   2.5

-->x=-5:0.1:5; //domínio de x para gráfico

-->plot2d(x,[3-2*x+2*x^2])  //fazer o gráfico da função
```

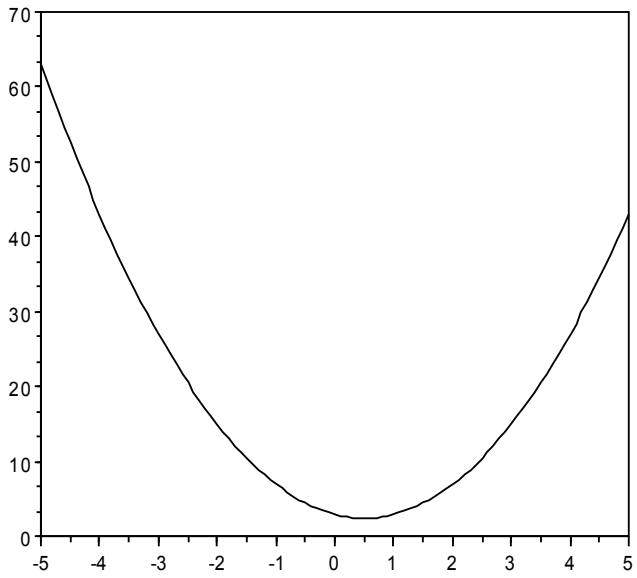


Figura 5.11. Gráfico da função minimizada

O comando linpro resolve problemas de programação linear. Assumo que a leitora está familiarizado com o tópico.

O comando aceita restrições que sejam igualdades e do tipo menor ou igual (<=). Tem três possibilidades de solução:

[x,lagr,f]=linpro(p,C,b [,x0])
Minimiza p'*x sujeito às restrições C*x <= b

[x,lagr,f]=linpro(p,C,b,ci,cs [,x0])
Minimiza p'*x sujeito às restrições C*x <= b , tendo x limites inferiores e superiores, ci <= x <= cs

[x,lagr,f]=linpro(p,C,b,ci,cs,me [,x0])
Minimize p'*x sujeito às restrições de igualdade, em número de me
C(j,:) x = b(j), j=1,...,me
e do tipo <=
C(j,:) x <= b(j), j=me+1,...,me+md
tendo x limites inferiores e superiores

ci <= x <= cs

lagr é o vector dos multiplicadores de Lagrange. Ver mais pormenores na ajuda do comando.

x0 é uma solução inicial fisível. Se for desconhecida, em seu lugar escrever 'v'.

Seja o problema de programação linear seguinte:

Minimizar

$z=-1{,}8\ x_1 - 2{,}4\ x_2 - 6{,}3\ x_3 - x_4$

Sujeito a

$2{,}4\ x_1 + 3{,}2\ x_2 + 4\ x_3 + 7{,}2\ x_4 = 21$

$3x_1 + 17\ x_2 + 80\ x_3 + 2\ x_4 \leq 48$

$x_1, x_2 \geq 0 \quad , \quad x_1, x_2 \leq 10000$

```
-->//C1*x = b1  (uma restrição de igualdade, logo me=1)

-->C1= [2.4, 3.2, 4, 7.2];

-->b1=[21];

-->//C2*x <= b2  (1 restrição de inequação)

-->C2=[3, 17, 80, 2];

-->b2=[48];

-->//com x entre ci e cs:

-->ci=[0;0;0;0];cs=[10000;10000;10000;10000];

-->//e minimizar p'*x com

-->p=[-1.8;-2.4;-6.3;-1]
 p  =

  - 1.8
  - 2.4
  - 6.3
  - 1.

-->//sem solução inicial: x0='v';

-->C=[C1;C2]; b=[b1;b2] ; me=1; x0='v';

-->[x,lagr,f]=linpro(p,C,b,ci,cs,me,x0)
 f  =

  - 16.707
 lagr  =
```

```
   0.
 - 0.572
   0.
 - 4.092
   0.695
   0.044
x  =

   8.2666667
   2.591D-15
   0.29
 - 2.168D-17
```

A solução é pois x_1=8.2666667, x_3=0.29, x_2=x_4=0. O valor óptimo da função a minimizar é z=- 16.707.

Os primeiros 4 elementos do vector lagr correspondem às variáveis. Como x_2 e x_4 são iguais a zero, pois só temos duas restrições, os seus multiplicadores são diferentes de zero. Como

```
-->C1*x
 ans  =

   21.

-->C2*x
 ans  =

   48.
```

os dois últimos multiplicadores são diferentes de zero. É a solução do problema dual.

Se o leitor se interessa por questões de optimização, deixo como exercício familiarizar-se com as alternativas do comando quapro, cuja sintaxe tem muito de comum com a do comando linpro. Para além do exemplo da ajuda do Scilab, deixo-lhe mais outro.

Minimizar

$z= 0.5*x'*Q*x + p'*x$

em que Q=[1,0,0,0; 0,0,0,0; 0,0,0,0; 0,0,0,0]

p=[-2;-1;0;0]

sujeito a

$2x_1 + 3x_2 + x_3 = 6$

$2x_1 + x_2 + x_4 = 4$

$x_1,x_2 \geq 0$, $x_1,x_2 \leq 10000$

Eis a solução no Scilab:

```
-->//Achar x em R^4 tal que:

-->//C1*x = b1  (duas restrições de igualdade i.e me=2)

-->C1= [2, 3, 1, 0; 2,1,0,1];
```

```
-->b1=[6;4];

-->//com x entre ci e cs

-->ci=[0;0;0;0];cs=[10000;10000;10000;10000];

-->//e minimizar 0.5*x'*Q*x + p'*x

-->p=[-2;-1;0;0];Q=[1,0,0,0;0,0,0,0;0,0,0,0;0,0,0,0];

-->C=C1;b=b1 ; me=2;

-->[x,lagr,f]=quapro(Q,p,C,b,ci,cs,me)
 f  =

  - 2.8888889
 lagr  =

    0.
    0.
  - 0.3333333
    0.
    0.3333333
  - 6.144D-17
 x  =

    1.3333333
    1.1111111
  - 1.894D-17
    0.2222222
```

5.8 Comandos supervenientes

Existem ainda dois úteis comandos que convém não ignorar e são cumsum e cumprod. O uso dos comandos é simples e têm três alternativas.

y=cumsum atribui a y soma acumulada pela ordem das colunas.

y=cumsum(x,'c') (ou y=cumsum(x,2)) atribui a y a soma acumulada das colunas de x: y(i,:)=cumsum(x(i,:))

y=cumsum(x,'r') (or y=cumsum(x,1)) atribui a y a soma acumulada das linhas de x: y(:,i)=cumsum(x(:,i))

cumsum(*vector ou matriz*)

cumsum*(vector ou matriz, 'c ou r')*

Vejamos alguns exemplos:

```
x=[1 2 3 4];

-->cumsum(x)
 ans  =

   1.    3.    6.    10.

-->A=[1,2;3,4];

-->cumsum(A)
 ans  =

   1.    6.
   4.    10.

-->cumsum(A,'r')
 ans  =

   1.    2.
   4.    6.

-->cumsum(A,'c')
 ans  =

   1.    3.
   3.    7.
```

y=cumprod atribui a y o produto acumulado pela ordem das colunas.

y=cumprod(x,'c') (ou y=cumprod(x,2)) atribui a y o produto acumulado das colunas de x: y(i,:)=cumprod(x(i,:))

y=cumprod(x,'r') (or y=cumprod(x,1)) a y o produto acumulado das linhas de x: y(:,i)=cumprod(x(:,i))

cumprod *(vector ou matriz)*

cumprod*(vector ou matriz, 'c ou r')*

Exemplifiqumos:

```
-->cumprod(A)
 ans  =

   1.    6.
```

```
   3.    24.

-->cumprod(A,'r')
 ans  =

   1.    2.
   3.    8.

-->cumprod(A,'c')
 ans  =

   1.    2.
   3.    12.
```

O comando calendar() cria um calendário do mês em que se está. calendar(1955, 5) dá o calendário do mês de Maio de 1955.

O comando date() retorna o dia, mês e ano actuais.

O comando getdate() fornece um vector com vária informação que pode ser obtida noa ajuda.

```
-->dt=getdate();

-->printf("Ano:%i, Mês:%i, Dia:%i, Hora:%i, Minutos:%i", dt(1), dt(2), dt(6), dt(7), dt(8));
Ano:2008, Mês:3, Dia:11, Hora:18, Minutos:58
```

O comando printf será abordado no capítulo 7.

5.9 Quadro sinóptico dos comandos introduzidos neste capítulo

Comando	Descrição
calendar	Fornece calendário do mês
cumprod	Calcula o produto acumulado
cumsum	Calcula a soma acumulada
datafit	Ajusta uma equação não linear
date	Dá o dia, mês e ano correntes
derivative	Aproxima numericamente a derivada de uma função num dado ponto
fsolve	Acha a solução de um sistema de equações não lineares
getdate	Dá um vector linha com vária informação sobre a data
interp1	Interpola valores de $y_j=f(x_j)$
interp2d	Interpola valores de $z_j=f(x_j,y_j)$
interp3d	Interpola valores de $v_j=f(x_j,y_j,z_j)$
intsplin	Integração numérica de dados usando interpolação pelo método spline
inttrap	Integração numérica de dados usando interpolação pelo método trapezoidal
linpro	Resolve problemas de programação linear
lsqrsolve	Ajusta uma equação não linear pelo método dos mínimos quadrados
ode	Resolve equações diferenciais ordinárias
optim	Rotina de optimização não linear
portr3d	Traça o diagrama de fase de um sistema de 3 equações diferenciais ordinárias
quapro	Resolve problemas de programação quadrática
smooth	Interpolação recorrendo ao método spline

Capítulo 6

Introdução elementar à programação

Se a leitora já está familiarizado com a programação, noutra ou noutras linguagens disponíveis, pode seguir para o capítulo seguinte, sem perda de continuidade. De modo sucinto, e na perspectiva da propedêutica ao capítulo seguinte, abordamos aqui alguns conceitos básicos da programação e da sua lógica. Os temas a tratar são os seguintes:

- Linguagem de programação
- Abordagem heurística dos problemas
- Articular algoritmos
- Estruturar fluxogramas
- Utilizar pseudocódigos
- Estruturas de controlo
- Modularização
- Recursividade

6.1 Problemas, algoritmos e programação

O Scilab permite dar ordens a um computador, para executar um conjunto de operações numéricas e lógicas, porque dispõe de um acervo de comandos e padrões de instruções predeterminados que actuam sobre a memória e o processador do computador. Isto é, o Scilab é uma **linguagem de programação** científica (particularmente eficiente no cálculo de expressões matemáticas) e de alto nível (por ser independente do computador utilizado).

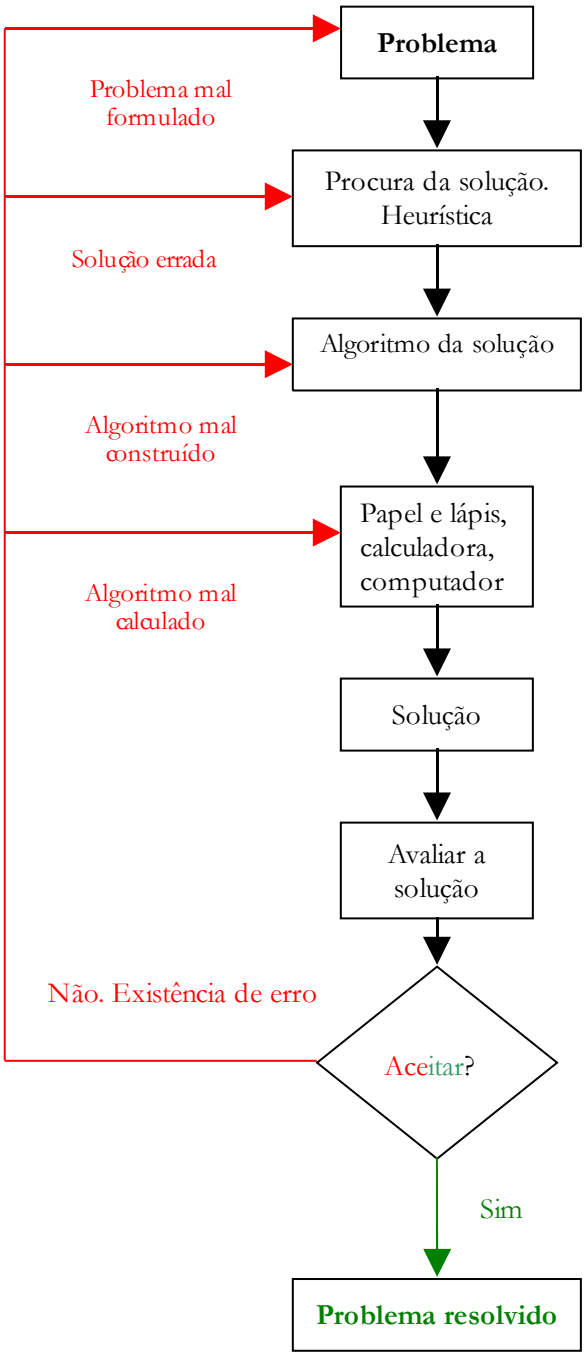


Figura 6.1. Sequência operacional da solução de um problema

O leitor recorre à sua calculadora (esperamos que passe a utilizar o Scilab) quando pretende realizar uma sequência de contas para um determinado fim. Isto acontece porque antes de iniciar os cálculos, tem um **problema** para resolver e dispõe de um conjunto finito de operações logicamente ordenadas (um **algoritno)** que lhe permite alcançar a **solução** pretendida. Esquematizemos a situação vivida na figura 6.1.

Em computação, um algoritmo implementado num programa é tido como eficiente se é de cálculo rápido e usa pouca memória, além de não ter erros, claro.

O gráfico da figura 6.1 diz-nos que, ao procurarmos a solução para um problema, podemos incorrer em quatro tipos de erros, que qualquer aluno conhece quando, numa prova:

- Percebe mal o enunciado dum problema e tenta resolver outro problema.
- Entende correctamente o problema mas não o resolve correctamente.
- A lógica da sua solução está correcta, mas é mal transposta para as operações a executar.
- Incorre em erros ao utilizar a calculadora.

Da figura 6.1 infere-se:

- A procura de solução é um processo recursivo, de tentativa e erro.
- Os problemas abordáveis numa dada época dependem dos meios que temos para os resolver (cálculo dos algoritmos da sua solução). A grande capacidade dos computadores em armazenar e manipular eficientemente grandes volumes de dados e executar rapidamente numerosas operações tem vindo a mudar o nosso modo de vida.

Quando na figura 6.1, a leitora utiliza o computador, primeiro tem de traduzir o algoritmo numa sequência de comandos de uma linguagem de programação, que o processador aceite, entenda e execute – um **programa**. Fica pois claro que algoritmos e programas são entidades distintas. Os primeiros existem há séculos. os segundos surgiram de forma difundida, na segunda metade do século passado. No começo da escolaridade, ao nos iniciarmos na aritmética, aprendemos algoritmos que permitem obter o produto de dois números ou dividir um pelo outro. Como se costuma dizer, aprendemos a fazer contas de multiplicar e dividir, e para isso precisamos de saber a tabuada.

6.2 Solucionar um problema

Hoje há consenso sobre o entendimento de que o conhecimento não é "constatante", mas "performativo". Nós não sabemos de uma vez por todas, mas vamos sabendo, resolvendo sucessivamente vários problemas aglutinados em problemáticas, que surgem de dados de facto, e o conhecimento disponível não consegue solucionar. Vivemos mergulhados num constante processo de pesquisa, invenção e inovação, de maior ou menor criatividade e rotina.

A procura da solução não começa, obviamente, do nada e o conhecimento existente da solução de problemas idênticos e parecidos é o ponto de partida mais comum.

Vários investigadores têm-se ocupado do processo de criação de ideias que conduzem à solução de problemas. Embora a criatividade seja uma característica pessoal ingénita, é possível com aprendizagem e prática melhorar a nossa capacidade de lidar com a resolução de problemas. Um brilhante matemático, George Polya (1888-1985), entre vários notáveis textos sobre o assunto, escreveu um livro intitulado "*How to Solve It*", em que propõe, no seu fecho uma estratégia para resolver problemas. O seu trabalho foi ponto de partida para contribuições de ouros autores entre os quais o professor do ensino superior norte-americano A. H. Schoenfeld que propôs uma estruturação dos **princípios**

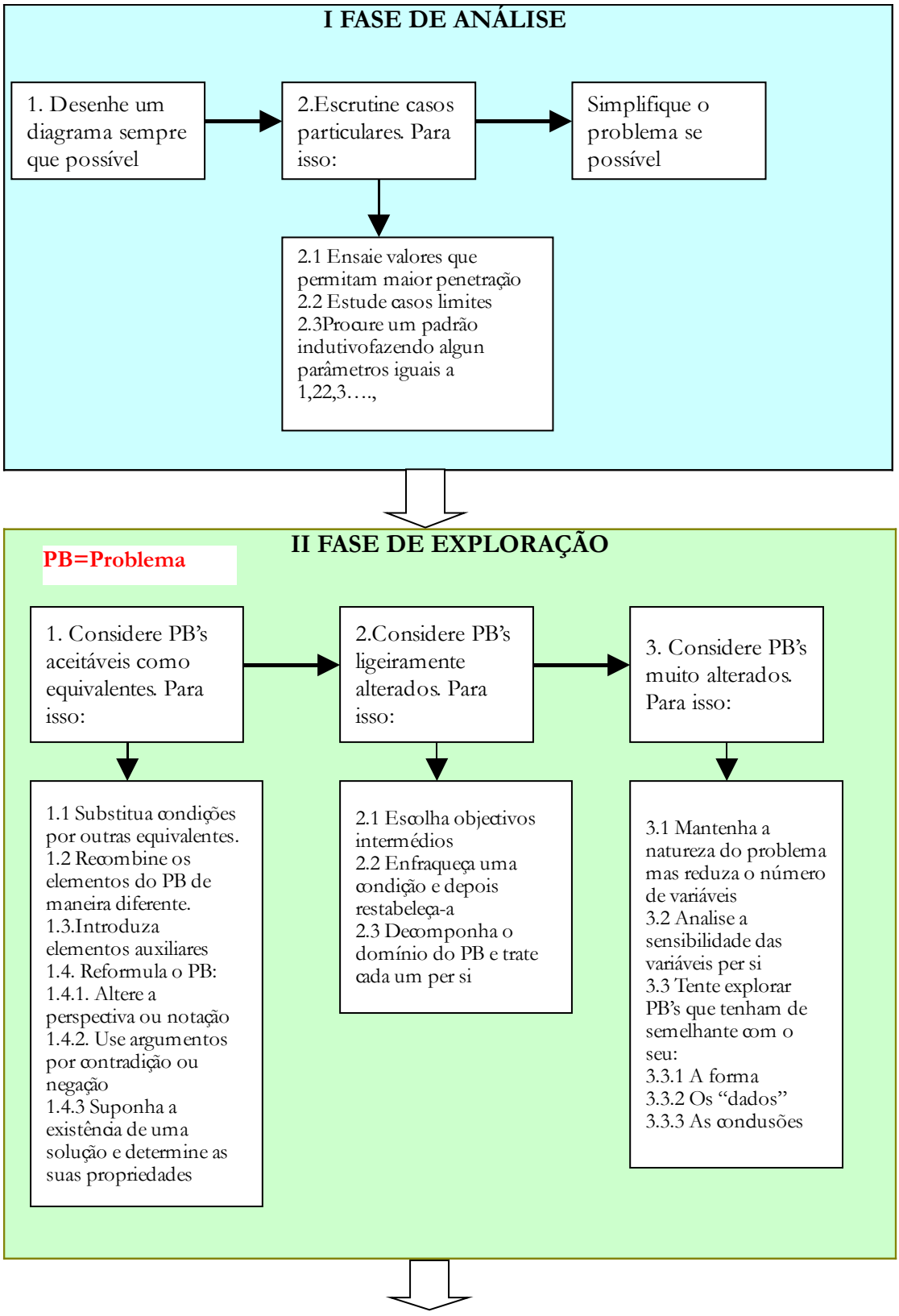


Figura 6.2. Diagrama da heurística de A. H. Schoenfeld.

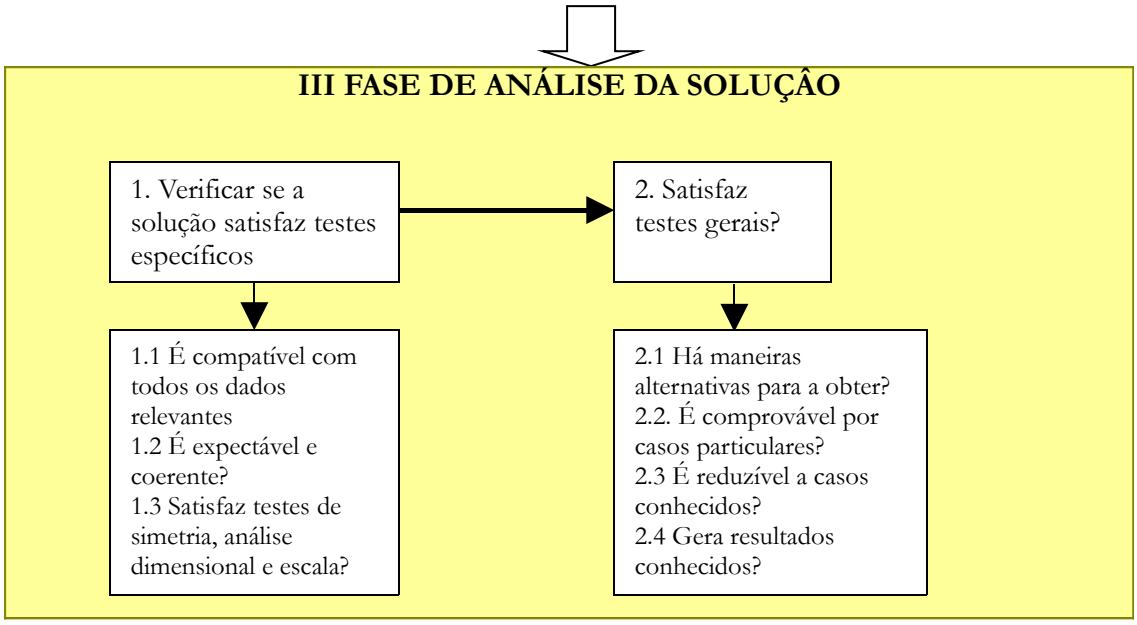


Figura 6.2. Continuação.

heurísticos usados frequentemente. As traduções das propostas heurísticas destes dois investigadores podem ser encontrada em Davis e Hersh (1995:269-271). Na figura 6.2, apresentamos uma descrição diagramática da heurística de A. H. Schoenfeld, que privilegia problemas matematizáveis.

Na perspectiva desta abordagem sumária, encontrada a lógica da solução e traduzida esta num algoritmo correcto, passamos à fase de programação da solução. Em problemas muito complexos, esta pode por si constituir outra área problemática, a que a estratégia proposta por A. H. Schoenfeld pode ser autonomamente aplicada. Pode acontecer que por limitações de capacidade de computação obrigue à procura de algoritmo mais tratável do ponto de vista da sua computação. No processo recursivo de tentativa e erro, da figura 6.1, o estabelecimento do algoritmo e a programação estão intimamente ligados.

6.3 Antes de começar a programar

Quando se faz um programa pretende-se que ele satisfaça quatro quesitos de qualidade:

1. Não tenha erros de lógica relativamente ao que se quere que ele execute (ver figura 6.1)
2. Não tenha erros de linguagem, que o façam ter a sua execução interrompida.
3. Seja inteligível em qualquer altura, para quem o programou ou qualquer outra pessoa.
4. Seja eficiente, isto é, tenha uma execução rápida, com o mínimo recurso a memória.

Para alcançar estes desideratos, a experiência consagrou a adopção de alguns procedimentos que vamos abordaremos adiante, em maior pormenor, a saber:

- Planear previamente o programa.

- Inserir comentários adequados e se se justificar prover documentação.
- Procurar clareza e simplicidade.
- Estruturar ou modularizar o programa.
- Não poupar esforços em aperfeiçoar o programa, após ter concluído a primeira versão executável e correcta.

6.4 Planear o programa

Quando no Outono de 1967, nos E.U.A., aprendi FORTRAN, tudo começava num fluxograma. Esta situação manteve-se ainda por duas décadas, e caiu praticamente em desuso, por duas razões principais: o incómodo de desenhar o fluxograma, a necessidade de memorizar o significado das caixas gráficas. O primeiro óbice é facilmente ultrapassável com os actuais processadores de texto, que permitem desenhar expeditamente as formas gráficas utilizadas nos fluxogramas, o segundo mantém-se.

Abandonado o fluxograma, surgiu a escrita em pseudocódigo que é uma forma abreviada de dizer o que o programa faz, em português normal. Passa a ser uma espécie de memória descritiva, ou guião, do futuro programa.

No entanto, há autores brasileiros que mencionam o "Portugol" (português com o Algol) , com o propósito de criar um conjunto básico de instruções primitivas que permitam planear um programa que seja facilmente quer traduzido numa linguagem quer corrido no computador.

Como os programas devem ser descritos pelo pseudocódigo com rigor, clareza e sem ambiguidade, não sendo prático, por outro lado, utilizar toda a riqueza expressiva da nossa língua, há quem use o português aproximando o seu emprego de linguagens de programação, nomeadamente do C e Pascal. Isto requer o prévio conhecimento desta duas linguagens, o que nem sempre ocorre.

Sem pretender inventar o "Portugrama", dada a natureza iniciática deste texto, em que o recurso a todos os instrumentos que facilitem a apreensão, pelo leitor, do exposto é sempre justificado, não esquecendo a heurística de A. H. Schoenfeld ("sempre que possível faça um diagrama") vamos usar um número restrito de símbolos de fluxograma, se necessário acompanhados de mais texto, para além do pseudocódigo.

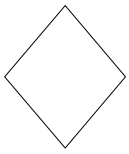
Para além do fluxograma e pseudocódigo, é possível também descrever o programa recorrendo ao Diagrama de Chapin ou NS (de Nassi e Schneiderman), que não será utilizado..

6.5 O fluxograma clássico

Os fluxogramas são particularmente aptos para representar programas onde ocorram acções diferentes para muitas decisões. Introduzimos algumas das formas gráficas codificadas ou pré estabelecidas, utilizadas nos fluxogramas. O Writer do OpenOffice esclarece o significado dos símbolos, se apontados com o rato, na barra de ferramentas inferior (Desenho).

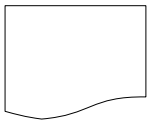
Início ou *fim* de um fluxograma

Processamento. Atribuição de uma valor a uma variável ou mudança de valor de uma variável. Também poe indicar a abertura ou fecho de um ficheiro

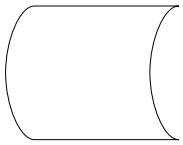


Decisão. Indica a mudança do fluxo de computação

Monitor. Indica saída de informação para o monitor

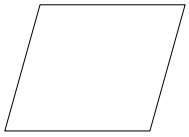


Dicumento. A informação é impressa em papel

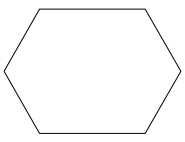


Input/output. Entrada ou saída de dados armazenados

Introdução de dados a partir do teclado



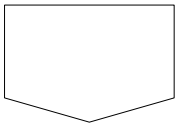
Entrada de dados durante a compilação do programa



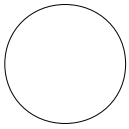
Preparação. Indica o início de um ciclo

Figura 6.2. Símbolos de fluxograma

Processamento predefinido. Transferência para um sub-programa ou sub-rotina



Conector de folhas. Mudança do fluxo de uma página para outra



Conexão. Transferência do fluxo de lógica de um ponto da página para outro ponto da página

Figura 6.2. Continuação

Suponhamos o pequeno problema de uma programa para calcular as raízes da equação do segundo grau a x^2+ b x +c =0, sendo a, b e c valores que se introduzem usando o teclado, com saída do resultado para a impressora, se as soluções forem reais, e para o monitor se forem números complexos. O fluxograma será o da figura 6.3.

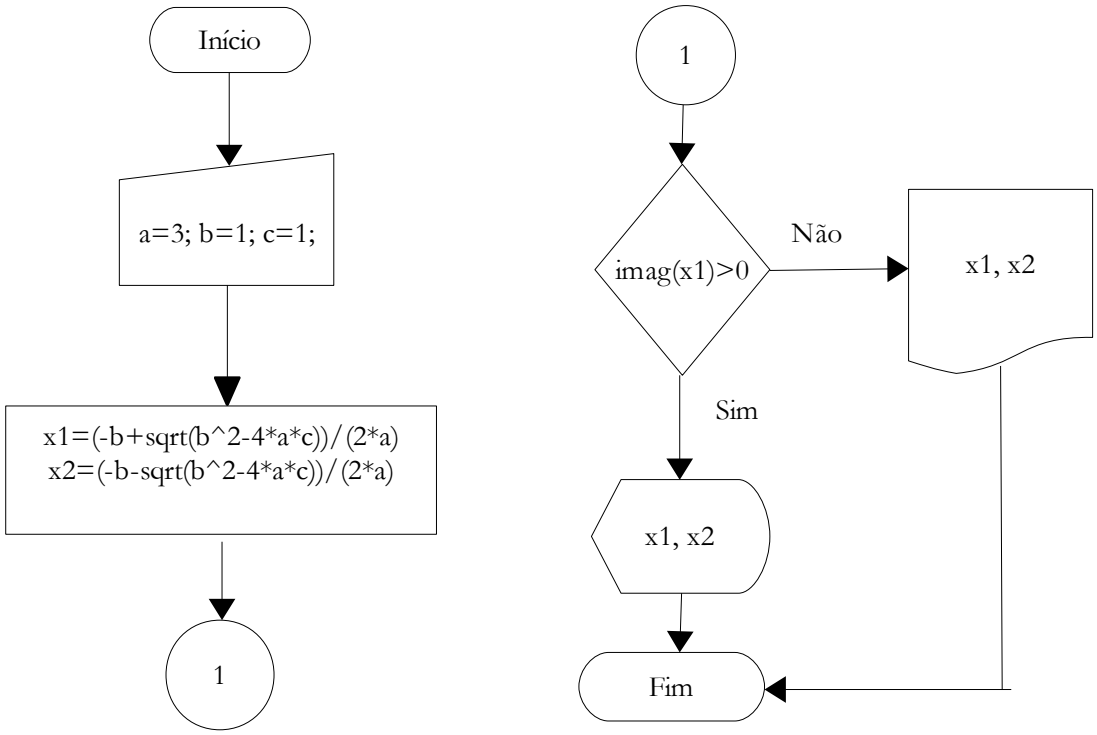


Figura 6.3. Fluxograma do cálculo da solução de uma equação do segundo grau

Ignorando a simbologia normalizada dos fluxogramas, podemos ter, entre outras formulações possíveis ,um diagrama do nosso programa de acordo com a figura 6.4.

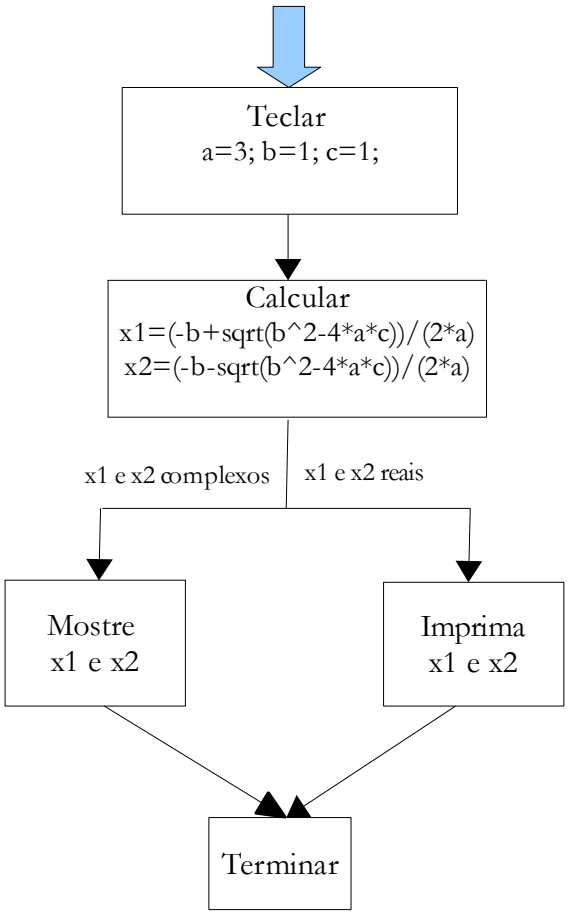


Figura 6.4. Diagrama do cálculo da solução de uma equação do segundo grau. Mostre = saída no monitor; Imprima = saída na impressora

6.6 O pseudocódigo

Como se disse, o pseudocódigo é a descrição do programa em português corrente. Só é vantajoso que a sua estrutura não seja totalmente arbitrária, mas obedeça a um mínimo de padronização que automatize as suas escrita, leitura e facilite a sua passagem para uma linguagem de programação. Vamos adoptar a estrutura descrita na Caixa 6.1.

A leitora pode refinar alguns aspectos do pseudocódigo.

Por exemplo, usar expressões codificadas para indicar a origem dos dados, como se apresentam (uma ou mais variáveis por linha).

Fazer o mesmo para a saída, como fizemos atrás, na figura 6.4, com as palavras mostre e imprima.

Se tiver muitas variáveis agrupáveis por afinidades relevantes pode usar prefixos no nome das variáveis, que permitam a sua atribuição a um dado grupo. Ilustrando: se estiver a simular um modelo ecológico com várias plantas, herbívoros e carnívoros, pode usar o prefixo he para as variáveis que representam herbívoros.

Caixa 6.1. Estrutura do pseudocódigo

Comentários
PROGRAMA: Nome do programa
Variáveis : nomes e tipo
Constantes: nomes e tipo
Início
Entrada de dados (input)
Bloco de comandos
Saída de dados (output)
Fim
FIM DO PROGRAMA

Sobre os comentários justificam-se as seguintes anotações:

- Os comentários que acompanham o programa devem permitir que qualquer pessoa que o leia entenda o que ele faz, de modo suficientemente inteligível para o poder corrigir ou modificar, se for caso disso.
- Do mesmo modo devem-se inserir instruções sobre a sua utilização..
- Descrever as variáveis mais importantes.
- Estruturas de dados utilizadas.
- Nome do autor
- Data da programação
- Menção sobre quaisquer métodos especiais utilizados, nomeadamente da análise numérica

Mais alguns procedimentos que conferem qualidade a um programa:

- Use uma estrutura modular, isto é, o programa é formados por curtos grupos de instruções, com propósito bem definido, e o mais independente possível em relação ao resto do programa. Isto tem as vantagens de os módulos poderem ser mantidos, feitos, refeitos e testados em qualquer altura. Podem igualmente ser utilizados noutros programas.
- Se possível use um comando por linha.
- Separe os módulos com linhas em branco.
- Separa os comentários e o corpo do programa com uma linha em branco.
- Comentários no corpo do programa devem estar entre linhas em branco.
- Escolha nomes sugestivos para as suas variáveis.
- Use indentação dos comandos, quando ocorrerem estruturas embutidas noutras.

- Quando fizer revisões do programa actualize os comentários. É preferível não ter comentários do que ter comentários fora de contexto.

Estes procedimentos facilitam a tarefa de aperfeiçoamento continuo do programa.

Sugiro agora, ao leitor, que escreva um pseudocódigo para o programa descrito pela figura 6.3.

6.7 Estruturas de controlo de um programa

Um programa traduz em operações realizadas pelo computador a lógica da solução encontrada para o problema de que nos ocupamos.

Da mesma maneira que o raciocínio que resolve um problema pode ser expresso em palavras portuguesas, inglesas, etc., também pode ser descrito por uma linguagem de programação adequada, como o Scilab para o cálculo numérico.

À semelhança da nossa língua, que tem estruturas básicas (substantivos, adjectivos, etc.) que usamos de acordo com regras gramaticais, as linguagens de computador também têm estruturas básicas e uma sintaxe, mais ou menos ricas conforme a complexidade da linguagem e o principal fim a que se destinam.

Para mimetizar certos procedimentos lógicos no computador, existem várias estruturas que controlam os actos que executam. Por exemplo, no fluxo do programa da figura 6.3, o programa tem que decidir se manda a solução da equação do segundo grau para o monitor ou para a impressora.

A mais simples estrutura é a **sequencial** em que os comandos são executados sucessivamente, pela ordem em que surgem. É a que tem sido usada até agora.

No entanto existem outras disponíveis:

- **Condicional**, que pode ser simples, composta e selectiva.
- **Repetitiva,** com controlo inicial, final, com variável de controlo, embutida ou não..

O Scilab revela recursividade no sentido em que uma função pode ser chamada a actuar de dentro dela própria.

No capítulo seguinte, particularizamos estas estruturas, para além de outros aspectos da programação em Scilab.

Em situações de programas simples, e este conceito é relativo à experiência de quem programa, pode-se omitir a fase de planeamento prévio (fluxograma ou pseudocódigo).

Capítulo 7

Programação em Scilab

Este capítulo, é especialmente dedicado à programação numa perspectiva mais abrangente e não sequencial Abordaremos:

- Tipos de dados que o Scilab reconhece
- Matrizes multidimensionais
- Estruturas de dados
- Comandos de inquirição sobre tipos de funções
- Tipos de variáveis
- Entrada de dados
- Saída de resultados e mensagens
- Estruturas de controlo do fluxo do programa:
  - -Declarações if
  - - Declarações select e case
  - - Ciclos for
  - - Ciclos while
  - - Declarações break
- Despistagem e correcção de erros nos programas

Como se espera, estes novos comandos e capacidades de programação serão utilizados conjuntamente com operadores aritméticos, relacionais e comandos anteriormente apresentados.

7.1 Tipos de dados que o Scilab reconhece

Em benefício da completude e por se tratar de uma informação que pode condicionar a utilização que a leitora faça do Scilab, nomeamos todos os tipos de dados que esta linguagem reconhece, a saber:

7.1.1 Matrizes de números

Foram objecto do capítulo 3.

7.1.2 Matrizes esparsas de números

Estas matrizes contêm um número restrito de elementos diferentes de zero, que podem ser números reais, complexos ou símbolos de verdadeiro (T) ou falso (F).Estas matrizes são criadas pelo comando sparse

sparse([*posições dos elementos diferentes de zero separadas por ;*], [*valores do elementos correspondentes separadas por :*], [*número de linhas, número de colunas*])

Vejamos um exemplo:

```
-->sparse([1 5;5 5;5 1],[1, 3, 2+3*%i],[5,5])
 ans  =

(    5,    5) sparse matrix

(    1,    5)      1.
(    5,    1)      2. + 3.i
(    5,    5)      3.
```

A saída do comando dá indicação do tamanho da matriz, que é esparsa, a localização e valores dos elementos diferentes de zero.

Outro exemplo da aplicação do comando sparse.

```
-->M=[1 2 0;0 0 0;0 0 0]
 M  =

    1.    2.    0.
    0.    0.    0.
    0.    0.    0.
```

```
-->sparse(M)
 ans  =

(   3,    3) sparse matrix

(   1,    1)        1.
(   1,    2)        2.
```

O comando full restitui a matriz completa:

full(*nome da matriz esparsa*)

```
-->full(M)
 ans  =

   1.    2.    0.
   0.    0.    0.
   0.    0.    0.
```

O Scilab tem vários comandos dedicados a manipulação e operações com este tipo de matrizes (apropos sparse).

7.1.3 Matrizes booleanas

São matrizes de duas dimensões de símbolos lógicos: Um exemplo:

nome da matriz=[*matriz de %T e %F*]

```
-->z=[%T %F; %T %F]
 z  =

 T F
 T F
```

É também possível obter matrizes booleanas esparsas, mas não cobriremos este tópico.

7.1.4 Matrizes de "strings"

As cadeias de caracteres ou "strings" também podem ser elementos de matrizes de duas dimensões:

```
-->['Autor:' 'L. S. Barreto'; 'Título:' 'Iniciação ao Scilab';'Ano:' '2008']
 ans  =

!Autor:   L. S. Barreto       !
!                             !
!Título:  Iniciação ao Scilab  !
!                             !
!Ano:     2008                !
```

7.1.5 Listas

O Scilab cria objectos com vários tipos de dados, denominado "list", com o comando list .

Um exemplo:

```
-->L=list('Ano',[2008 3 6],['Inverno', 'bissexto'])
 L  =


      L(1)

 Ano

      L(2)

    2008.    3.    6.

      L(3)

!Inverno  bissexto  !
```

O comando tlist cria tipos de listas, o que permite criar novos objectos abstractos de dados. Vamos criar o objecto planetas, com o nome , a distância ao Sol em milhões de quilómetros, e a duração da órbita em anos (1=duração da órbita da Terra).

O comando Chars é obrigatório. Cria linhas separadas de "strings", das variáveis entradas.

```
->planetas=tlist(['Chars';'Nome';'Distancia';'Órbita'],['Marte';'Saturno';'Urano'],...
-->[228;1427;2875],[1.88;29.50;84.00])
 planetas  =


      planetas(1)

!Chars      !
!          !
!Nome       !
!          !
!Distancia  !
!          !
!Órbita     !

      planetas(2)

!Marte    !
!         !
!Saturno  !
!         !
!Urano    !

      planetas(3)

    228.
    1427.
    2875.

      planetas(4)

    1.88
```

```
   29.5
   84.
```

Alguns exemplos de utilização da lista planetas:

```
-->planetas('Nome')(2)
 ans  =

 Saturno

-->planetas('Distancia')(find(planetas('Nome')=='Marte'))
 ans  =

   228.


-->sum(planetas('Distancia'))
 ans  =

   4530.
```

7.1.6 Matrizes multidimensionais

Nesta subsecção e na seguinte, vou introduzir dois aspectos mais avançados de estruturas de dados.

O Scilab pode criar matrizes multidimensionais. Por exemplo, a seguinte matriz de três dimensões:

```
-->A=ones(2,2,3)
 A  =

(:,:,1)

   1.   1.
   1.   1.
(:,:,2)

   1.   1.
```

1. 1.

(:,:,3)

1. 1.
1. 1.

:

Do mesmo modo, podemos ter M=ones(2,2,3,4). Veja a matriz A como um livro de um capítulo, com 3 páginas, cada uma com uma matriz de 1's, 2x2. A matriz M como um livro de 4 capítulos, todos iguais ao capítulo único do "livro A". Esta metáfora pode albergar, vários livros, estantes, bibliotecas num bairro, etc.. Vejamos uma matriz de sete dimensões: N=ones(2,2,4,5,6,7,3). Pode ser vista como:

> [**1** matriz de 1´s **2x2** numa página, **4** páginas por capítulo, **5** capítulos por livro, **6** livros iguais, **7** estantes de 6 livros iguais, **3** bibliotecas só com 7 estantes destas]

O leitor tente estabelecer uma metáfora homóloga recorrendo ao conceito de ficheiro, hierarquia de pastas, CD, caixa com Cd's, armário com caixas com CD´s...

Todas as operações, **elemento a elemento**, são válidas para as matrizes de todas as dimensões, tais como 5*N, cos(M). Se A e B forem duas matrizes multidimensionais do mesmo tamanho, então as operações A+B e A-B são executáveis. Exemplifiquemos:

-->B=5*A;

-->C=A+B
C =

(:,:,1)

6. 6.
6. 6.

(:,:,2)

6. 6.
6. 6.

(:,:,3)

6. 6.
6. 6.

-->D=log(C)

```
 D  =


(:,:,1)

   1.7917595   1.7917595
   1.7917595   1.7917595
(:,:,2)

   1.7917595   1.7917595
   1.7917595   1.7917595
(:,:,3)

   1.7917595   1.7917595
   1.7917595   1.7917595
```

Um comando alternativo para criar matrizes multidimensionais é hypermat. Vejamos uma ilustração:

```
-->M=hypermat([2 3 2 2],1:24)
 M  =


(:,:,1,1)

   1.   3.   5.
   2.   4.   6.
(:,:,2,1)

   7.   9.   11.
   8.   10.  12.
(:,:,1,2)

   13.  15.  17.
   14.  16.  18.
(:,:,2,2)
```

```
   19.   21.   23.
   20.   22.   24.
```

7.1.7 Estruturas

É possível criar estruturas de registos no Scilab, denominadas "struct array" com "fields" (campos). No lado esquerdo dos comandos para criar a estrutura só pode entrar com letras do alfabeto inglês. Ç e á não são aceites. Criemos um ficheiro de três países com três campos: nome, estado, população, expressa em milhões de habitantes.

```
-->paises(1).nome='Portugal';

-->paises(2).nome='Espanha';

-->paises(3).nome='França';

-->paises(1).estado='República';

-->paises(2).estado='Monarquia';

-->paises(3).estado='República';

-->paises(1).populacao=10;

-->paises(2).populacao=39;

-->paises(3).populacao=62;

-->paises
 paises  =

3x1 struct array with fields:
   nome
   estado
   populacao
```

As estruturas não não aceitam certos comandos como as tlist. Por exemplo, o comando sum não é aplicável, por isso o procedimento será:

```
-->paises(1).populacao+paises(2).populacao+paises(3).populacao
 ans  =

   111.
```

7.2 Identificar tipos de funções

Para identificar os tipos de funções o Scilab tem dois comandos.

O comando type proporciona um inteiro que corresponde a um código de tipos. Por exemplo::

```
-->type(planetas)
 ans  =

   16.
```

Os códigos existentes são os constantes do quadro 7.1.

O comando typeof retorna o nome do tipo associado ao objecto inquirido:

```
-->typeof(planetas)
 ans  =

  Chars
```

Quadro 7.1. Códigos de identificação de funções

Código	Descrição
1	matriz de constantes reais ou complexas
2	matriz de polinómios.
4	matriz booleana
5	matriz esparsa
6	matriz booleana esparsa
8	matrix of integers stored on 1 2 or 4 bytes
9	matriz de "handles" de gráficos
10	matriz de cadeias de caracteres
11	função não compilada
13	função compilada
14	biblioteca de funções
15	list. (lista)
16	lista tipificada (tlist)
17	mlist
128	pointer

7.3 Tipos de variáveis

Cada função do Scilab tem as sua próprias variáveis locais e pode ler as variáveis criadas no seu espaço de computação. O comando global permite que as variáveis possam ser acedidas por várias funções.

Queremos tornar globais as variáveis x, y, z:

```
global x y z
```

O comando cleargloba apaga todas as variáveis globais.

O comando isglobal verifica se uma variável é global.

7.4 Entrada de dados

Os dados utilizados num programa podem ser introduzidos de várias maneiras:

- Fazendo parte do próprio programa (o que tem sido usado)
- Lendo ficheiros em formato *.txt
- Lendo ficheiros de folhas de cálculo
- Usando uma janela de diálogo.

7.4.1 Ler ficheiros *.txt (ASCII)

A minha experiência leva-me a propor uma maneira de ler os ficheiros que seja bastante flexível, isto é, sirva para números e "strings". Isto alcança-se usando simultaneamente evstr e mgetl.

Suponhamos que temos um ficheiro chamado z2 (z2.txt) gravado numa disquete, na drive A. escrito da seguinte maneira:

```
["aq" "rr" "yt";
"dd" "3" "4";
"qwe" "d" "yui"];
```

Vamos lê-lo com o comando:

```
-->z=evstr(mgetl("A:\z2.txt",[,-1]));
```

Verifiquemos se a leitura foi correcta:

```
-->z
 z  =

!aq   rr  yt   !
!          !
!dd   3   4    !
!          !
```

```
!qwe  d   yui  !
```

Voltemos ao sistema de equações lineares. Suponhamos o ficheiro A.txt, com o seguinte texto:

```
[0.5 3 1.5;
0.6 2 -2;
2  -4 12 ]
```

e o ficheiro b.txt com o texto:

```
[2;5;3];
```

Vamos ler a matriz A e o vector coluna b, da disquete e resolver o sistema.

```
-->A=evstr(mgetl("A:\A.txt",[,-1]));

-->b=evstr(mgetl("A:\b.txt",[,-1]));

-->[x,kerA]=linsolve(A,b)
 kerA  =

    []
 x  =

  - 5.8238636
  - 0.0482955
    0.7045455
```

Obtivemos a solução já conhecida. Podíamos escreve a matriz e o vector na matriz A2:

```
-->A2=evstr(mgetl("A:\A2.txt",[,-1]))
 A2  =

    0.5    3.    1.5    2.
    0.6    2.  - 2.     5.
    2.   - 4.    12.    3.
```

e depois dela extrair a matriz A e o vector b (ver capítulo 3):

```
-->b=A2(:,4)
 b  =


    2.
    5.
```

```
   3.
```

```
-->A=A2(:,1:3)
 A  =

   0.5    3.    1.5
   0.6    2.  - 2.
    2.  - 4.    12.
```

e agora resolver o sistema:

```
-->[x,kerA]=linsolve(A,b)
 kerA  =

   []
 x  =

 - 5.8238636
 - 0.0482955
   0.7045455
```

7.4.2 Usar caixa de diálogo ("Scilab Dialog")

Os dados também podem ser inseridos, usando caixas de diálogo. Introduzamos o comando x_matrix que abre uma caixa para introduzirmos uma matriz 3x3. A caixa criada exibe uma matriz de zeros, que nós substituímos com os elementos da nossa matriz.

```
-->x=evstr(x_matrix('Valores de x',zeros(3,3)))
```

Surge a caixa de diálogo da figura 7.1.

Figura 7.1. Caixa de diálogo para se introduzir uma matriz 3x3

Depois de substituirmos os zeros temos a janela da figura 7.2. Clicamos em "OK" e surge-nos a saída:

```
 x  =

    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Figura 7.2. Matriz introduzida antes de clicarmos "OK"

Podemos confirmar a existência da matriz pedindo a sua transposta:

```
-->d=x'
 d  =
```

1. 4. 7.
2. 5. 8.
3. 6. 9.

7.4.3 Ler um ficheiro Excel

Seja uma folha do Excel com a seguinte matriz:

x1	x2	x3	b
2	-2	2	2
3	5	7	5
6	1	4	3

guardada no ficheiro A2exc.xls na disquete inserida na drive A.

Primeiro temos que abrir o ficheiro, com o comando xls_open, da seguinte maneira:

[Um número que tem a ver com o fluxo de dados do ficheiro Excel, o texto que surge na folha Excel, os nomes lidos das folhas de cálculo, um vector de números que referencia a posição das folhas na informação do ficheiro]=xls_open('*caminho e nome do ficheiro Excel*')

```
-->[fd,SST,Sheetnames,Sheetpos] = xls_open('A:\A2exc.xls')
 Sheetpos  =


   1825.   2346.   2609.
 Sheetnames  =

!Sheet1  Sheet2  Sheet3  !
 SST  =

!x1  x2  x3  b  !
 fd  =


   1.
```

Agora vamos ler o ficheiro aberto com o comando xls_read, e usando da saída anterior fd e Sheetpos:

[Os números da matriz, os índices dos cabeçalhos das colunas]=xls_read(*fd, Sheetpos*)

```
-->[Value, TextInd]=xls_read(fd, Sheetpos)
 TextInd  =

    1.    2.    3.    4.
    0.    0.    0.    0.
    0.    0.    0.    0.
    0.    0.    0.    0.
 Value  =

   Nan   Nan   Nan   Nan
    2.  - 2.    2.    2.
    3.    5.    7.    5.
    6.    1.    4.    3.
```

Extraímos a matriz A:

```
-->A=Value(2:4,1:3)
 A  =

    2.  - 2.    2.
    3.    5.    7.
    6.    1.    4.
```

e o vector b:

```
-->b=Value(2:4,4)
 b  =

    2.
    5.
    3.
```

Podemos agora resolver o sistema de equações lineares associado:

```
-->[x,kerA]=linsolve(A,b)
 kerA  =

   []
 x  =

   0.0454545
   0.1818182
  -0.8636364
```

Os ficheiros do Excel a abrir e ler pelo Scilab devem ter os números com a parte inteira separada da decimal por um ponto.

Um ficheiro guardado em formato *.csv, só com números respeitando o formato atrás mencionado, em qualquer folha de cálculo, pode ser lido com o comando excel2sci.

Um exemplo:

Matriz a que se atribuem os dados lidos= excel2sci('*caminho e nome do ficheiro*')

```
-->M=excel2sci('A:\A3cs.csv');
-->M
 M  =

!1 3 7 6   !
!          !
!2 1 11 4  !
!          !
!5 4 5 7   !
```

7.4.4 O comando input

O comando input permite introduzir em qualquer ponto de um programa dados, através do teclado, e atribuí-los a uma variável ou objecto.

Caso de matriz numérica:

```
-->A=input("Meta a matriz A")
Meta a matriz A-->[4 3;2 1]
 A  =
```

4. 3.
2. 1.

Caso de matriz de "strings":

:

```
-->C=input("Meta a matriz C","string")
Meta a matriz C-->[Lisboa Portugal;Madride Espanha;Paris França]
 C  =

 [Lisboa Portugal;Madride Espanha;Paris França]
```

Se o utilizador tiver dificuldade em se lembrar do caminho para um ficheiro use o comando tk_getdir(), que abre o ficheiro geral do utilizador do computador, em árvore, para busca.

7.5 Mensagens e saída de resultados

7.5.1 O comando x_message

O comando x_message permite fazer surgir uma pequena janela com um mensagem que pode ser passiva ou decisória.

Vejamos o primeiro caso.

```
x_message(['Texto da mensagem'])
```

Se teclar o seguinte comando,

```
x_message(['A matriz é singular';'use o método dos mínimos quadrados'])
```

surge-lhe a janela da figura 7.3. Depois de tomar conhecimento do seu texto deve clicar "OK" para o programa prosseguir.

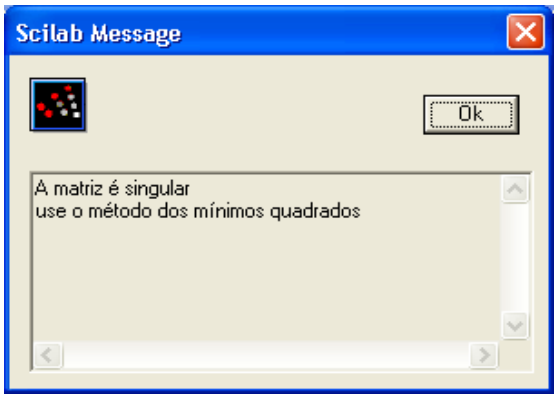


Figura 7.3. Caixa de mensagem passiva associada ao comando x_message

A outra alternativa :

resposta=x_message('*Texto da mensagem*', [*legenda do botão 1, legenda do botão 2*])

```
resposta=x_message('Interromper o programa?',['Sim','Não'])
```

Este comando faz surgir a janela da figura 7.4. Se clicar no botão "Sim", o programa imprime:

```
resposta  =

1.
```

Se tivesse clicado em "Não" teria aparecido no monitor:

```
resposta  =

2.
```

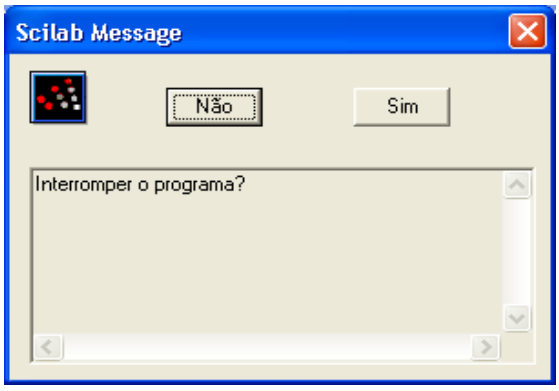


Figura 7.4. Janela do comando x_message para permitir escolha pelo utilizador

A leitora deve explorar a biblioteca GUI and Dialogs da ajuda do Scilab.

7.5.2 Saída de resultados

O comando de utilização mais comum é o já conhecido disp. O número de linhas de texto é automaticamente ajustado á janela do Scilab, mas pode ser modificado pelo comando lines, o que não trataremos nesta iniciação.

```
-->disp(["A raiz quadrada de dois é:",string(sqrt(2))])
```

```
!A raiz quadrada de dois é:  1.4142136  !
```

É possível dispor o texto em linhas de acordo com:

```
-->disp(["Isto é";"um vector";"de texto"])

!Isto é     !
!          !
!um vector  !
!          !
!de texto   !

disp(["Isto é       uma matriz";"de          texto"])

!Isto é       uma matriz  !
!                        !
!de           texto       !
```

O comando printf permite também dispor de informação no monitor. É mais flexível que disp, pois permite misturar texto e valores de variáveis sem recorrer ao comando string. Os números também podem ser formatados. Se escrever:

- %i o número é formatado como inteiro
- %e com notação científica
- %f ponto fixo, com o número de casa decimais que se queira

Ilustremos a sua utilização:

```
-->a=2.3;b=2345;c=7.6789325;g=sqrt(2);
-->printf("As variáveis são: %i, %e, %f, %1.3f %1.7f",a,b,c,c,c)
As variáveis são: 2, 2.345000e+003, 7.678933, 7.679 7.6789325
-->printf("A raiz quadrada de %i é %f",a,g)
A raiz quadrada de 2 é 1.414214
```

O comando file permite gerir ficheiros no Scilab, mas esse tema ultrapassa o âmbito desta iniciação. O leitor entre com apropos file, para explorar este assunto, se tiver necessidade disso.

Como se disse no capítulo 1, é possível imprimir toda a sessão de trabalho na janela de comandos do Scilab, clicando o ícone que representa uma impressora, na barra da sua janela. Se antes se quiser ter acesso à janela de definição da impressora pode-se utilizar o comando printsetupbox, que a faz surgir.

7.6 O controlo if

Suponha a leitora que está em casa num dia muito nebulado e decide sair de casa. Pensa: se *(if)* estiver a chover então (*then*) levo o guarda chuva. Levanta-se da cadeira vai à janela: acontece que está a chover, vai buscar o guarda-chuva e sai; foi à janela e verifica que não chove, limita-se a sair de casa.

Para tomar esta decisão numa situação condicional, o Scilab, à semelhança de várias linguagens de programação tem o seguinte comando:

```
if condição then acção ;
end
```

A acção só é executada se a condição for verificada. No seu caso a condição era ser verdade (V) estar a chover e a acção consequente levar o guarda-chuva. Se a condição fosse falsa (F) não chover limitava-se a continuar o seu acto de sair de casa.

Podemos escrever o fluxograma associado ao comando if...then....end, como se exibe na figura 7.5.

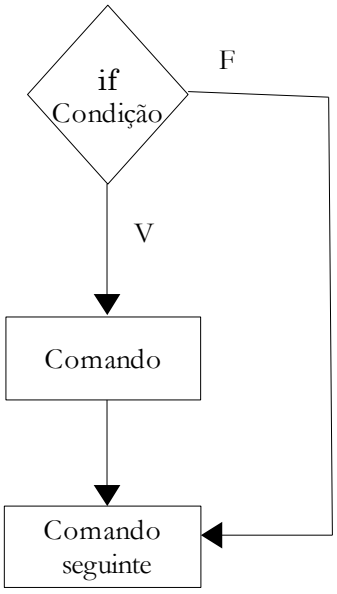


Figura 7.5. Fluxograma associado ao comando if...then....end.

Sugiro que o leitor, daqui em diante, escreva os pseudocódigos correspondentes aos fluxogramas que forem surgindo no texto.

Vejamos um exemplo.

```
-->a=1;
-->if a>0 then disp("Levo o guarda-chuva");

 Levo o guarda-chuva
-->end

-->disp("Saio de casa")

 Saio de casa
```

Agora seja a=0:

```
-->a=0;

-->if a>0 then disp("Levo o guarda-chuva");
-->end

-->disp("Saio de casa")

 Saio de casa
```

Suponha que a leitora é uma pessoa prudente e pensava: se estiver a chover levo o guarda-chuva, senão estiver a chover levo a gabardine. Nesta situação o leitor se não estivesse a chover não se limitava a sair simplesmente de casa, antes ia vestir a gabardine. Uma situação homóloga desta pode ser transmitida ao computador com o comando:

```
if condição then acção 1;
else
acção 2;
end
```

A este comando associa-se o fluxograma da figura 7.6.

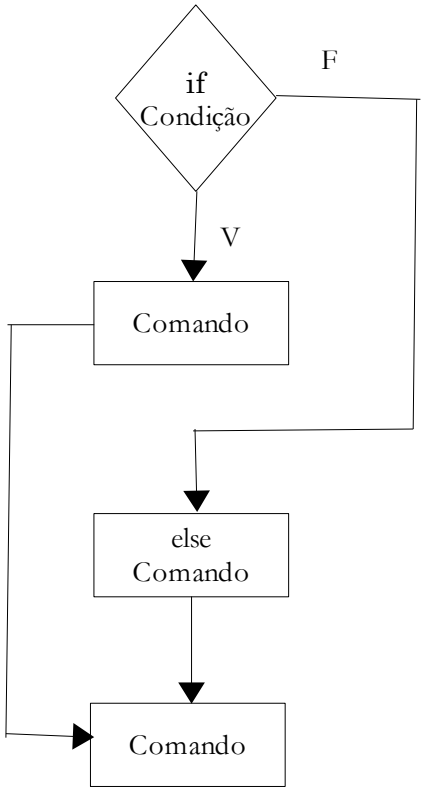


Figura 7.6. Fluxograma associado ao comando if...then....else...end.

Um exemplo:

```
-->a=1;

-->if a>0 then disp("Chove");

 Chove
-->else
-->disp("Não chove")
-->end

-->disp("Termina aqui")

 Termina aqui
```

O complementar:

```
-->a=0;

-->if a>0 then disp("Chove");
-->else
-->disp("Não chove")

 Não chove
-->end

-->disp("Termina aqui")

 Termina aqui
```

Admitamos que a leitora tem um bom impermeável de oleado (de que não gosta muito), com capucho, de boa qualidade, além do guarda-chuva e da gabardine. Decide agora: se chove muito intensamente levo o impermeável de oleado, se chover menos o guarda-chuva, se não chover levo a gabardine.

O Scilab lida com a lógica desta situação com o comando:

if *condição* then *acção* 1;
elseif
accção 2;
else
acção 3;
end

Como espera, agora tem três exemplos:

Exemplo 1.

```
-->a=2;

-->if a>1 then disp("Chove muito");

 Chove muito
-->elseif a>0
-->disp("Chove");
```

```
-->else
-->disp("Não chove")
-->end

-->disp("Termina aqui")

 Termina aqui
```

Exemplo 2.

```
-->a=1;

-->if a>1 then disp("Chove muito");
-->elseif a>0
-->disp("Chove");

 Chove
-->else
-->disp("Não chove")
-->end

-->disp("Termina aqui")

 Termina aqui
```

Exemplo 3.

```
-->a=0;

-->if a>1 then disp("Chove muito");
-->elseif a>0
-->disp("Chove");
-->else
-->disp("Não chove")
```

```
 Não chove
-->end

-->disp("Termina aqui")

 Termina aqui
```

Podemos inserir mais do que uma declaração elseif, antes do else final.

Vejamos o fluxograma deste comando, na figura 7.7.

É possível incluirmos mais de uma condição para ser simultaneamente satisfeita, num comando if. O símbolo que liga as condições é &. Ilustremos.

```
-->a=2;b=4;

-->if a>2 & b<3 then disp('Não satifaz');
-->else
-->disp('Satisfaz')

 Satisfaz
-->end
```

Do mesmo modo, o comando pode conter alternativas. O símbolo equivalente a 'ou' é |. Modifico o exemplo anterior para:

```
-->if a>2 | b<5 then disp('Não satifaz');

 Não satifaz
-->else
-->disp('Satisfaz')
-->end
```

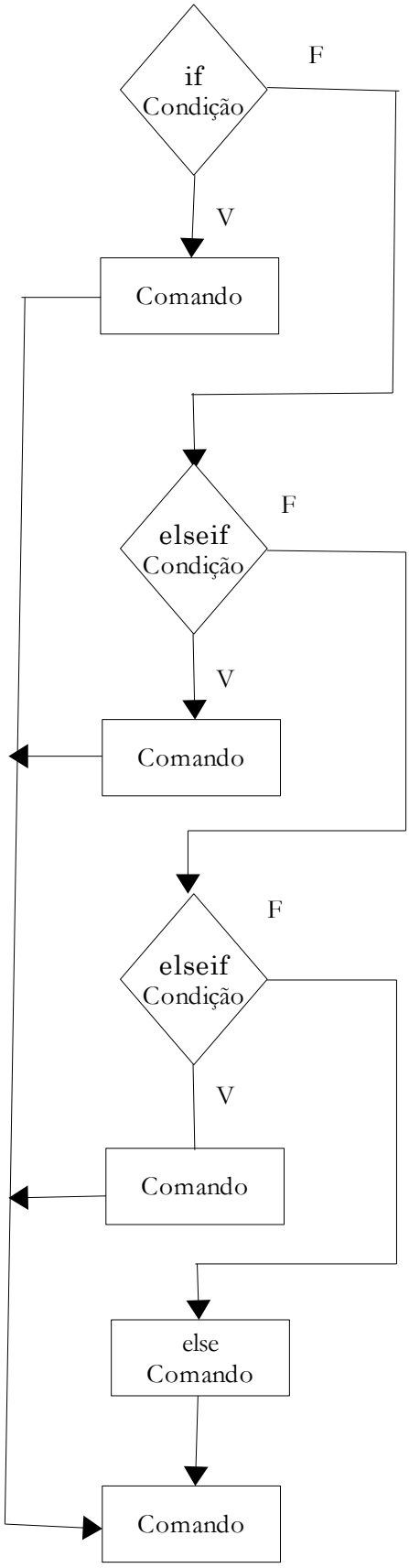


Figura 7.7. Fluxograma do comando if.. then...elseif...elseif else end

7.7 O controlo select

O comando select ....case tem a possibilidade de executar uma série de outros, e escolhe um deles conforme o valor de uma variável escolhida.

O fluxograma insere-se na figura 7.8, correspondente à selecção de dois casos.

.

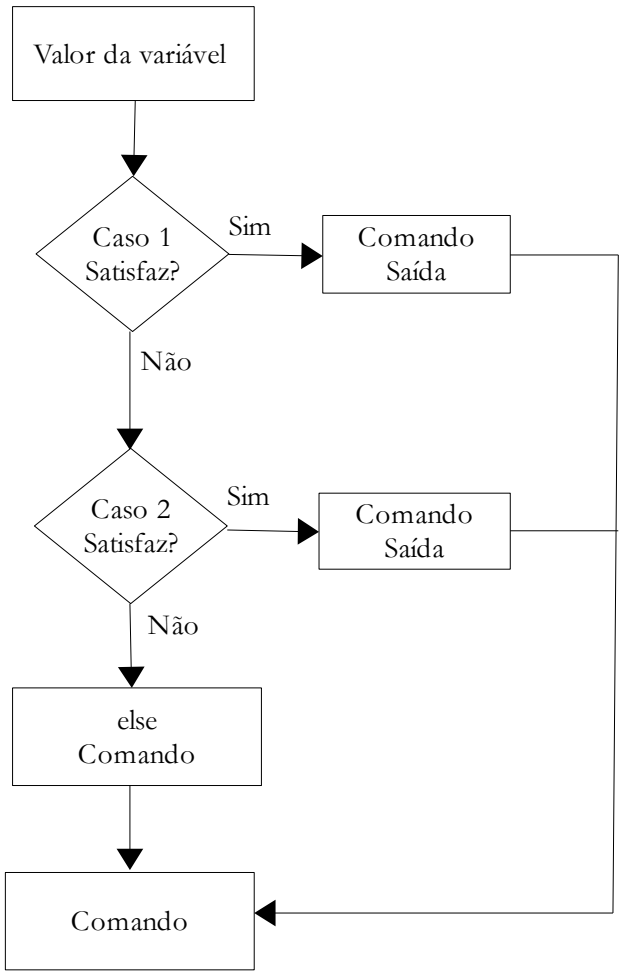


Figura 7.8. Fluxograma do comando select ...case

A estrutura do comando para dois casos é a seguinte:

```
select nome da variável
case primeiro valor da variável then
comando
case segundo valor da variável then
comando
```

```
else
comando
end
```

Pode ou não ocorrer

Imaginemos a seguinte situação. Um programa calcula uma variável a, que só pode assumir os valores 1 (baixo), 2 (médio) e 3 (alto), caso contrário ocorreu um erro. Deseja-se que o programa dê logo informação sobre o nível da variável. Vamos considerar as três situações possíveis.

Exemplo 1

```
-->a=1;

-->select a
-->case 1 then
--> disp('Baixo')
```

Sempre na mesma linha

```
 Baixo
-->case 2 then
--> disp('Médio')
-->case 3 then
--> disp('Alto')
-->end

-->disp('Termina aqui')

 Termina aqui
```

Exemplo 2

```
-->a=2;
->select a
-->case 1 then
--> disp('Baixo')
-->case 2 then
--> disp('Médio')
```

```
 Médio
-->case 3 then
--> disp('Alto')
-->end

-->disp('Termina aqui')

 Termina aqui
```

Exemplo 3.

```
-->a=3;
-->select a
-->case 1 then
--> disp('Baixo')
-->case 2 then
--> disp('Médio')
-->case 3 then
--> disp('Alto')

 Alto
-->end

-->disp('Termina aqui')

 Termina aqui
```

Exemplo 4, onde também se pretende assinalar a ocorrência de erro.

```
-->a=5;

-->select a
-->case 1 then
```

```
--> disp('Baixo')
-->case 2 then
--> disp('Médio')
-->case 3 then
--> disp('Alto')
--> else
--> disp('Ocorreu erro')

 Ocorreu erro
-->end

-->disp('Termina aqui')
 Termina aqui
```

7.8 O comando for...end

O comando for...end cria ciclos de comandos que no seu conjunto são repetidos, tantas vezes quantas queiramos, actualizando os valores das variáveis em cada ciclo, naturalmente.

A estrutura do comando é a seguinte:

Se omisso é igual a 1

```
for contador=valor inicial:intervalo:valor máximo
comandos
end
```

Os comandos for ... end podem estar incluídos uns nos outros. O fluxograma do comando é o da figura 7.8. Um curto exemplo:

```
for n=1:50
m=(m+1)/m;
end
```

Outro exemplo que soma os elementos de uma matriz, embora o comando sum seja mais eficiente. Criamos dois ciclos, um embutido no outro, porque temos de somar os elementos de cada linha, coluna a coluna. Veja a secção 3.3, sobre a identificação dos elementos de uma matriz.

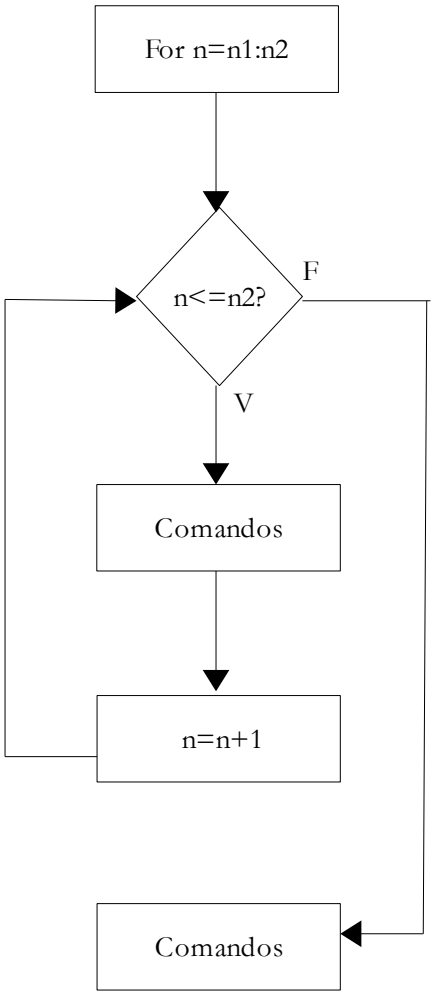


Figura 7.8. Fluxograma do comando for ... end

```
//soma dos elementos de uma matriz
//definir a matriz
mat=matrix([1:2:29],3,5);
//criar uma variável inicialmente igual a zero
//a que são somados os elementos da matriz
tot=0;
//seleccionar os elementos de uma linha
for i=1:3
//somar os elementos da linha, coluna a coluna
  for j=1:5
    tot=tot+mat(i,j);
```

```
  end
end
printf(O total é %i",tot)
```

A saída deste programa é:

```
O total é 225
```

7.9 O comando while ... end

O comando while repete um ciclo enquanto um condição for satisfeita.

```
while condição
comandos
end
```

O fluxograma de while exibe-se na fgura 7.9.

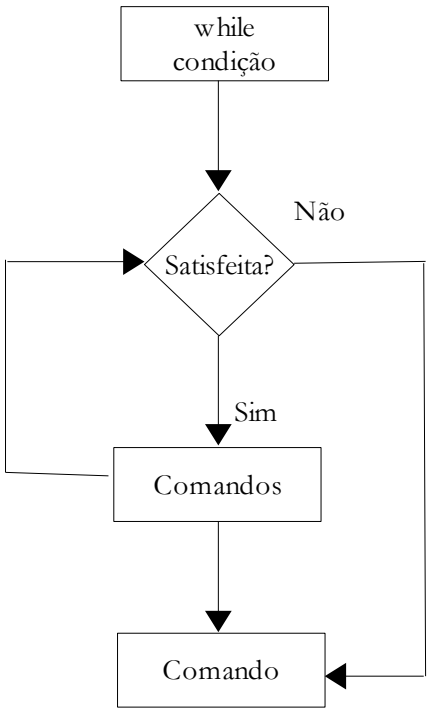


Figura 7.9. Fluxograma do comando while...end.

Um matemático alemão chamado Lothar Collatz, em 1937, apresentou a conjectura que passo a descrever. Nunca foi numericamente contraditada, nem provada. Se tomarmos um número qualquer e, sucessivamente, o dividirmos por dois se for par e multiplicarmos

por três e adicionarmos um, se for impar, acabamos por obter o número 1. Vamos fazer um pequeno programa para verificar esta conjectura.

```
//Conjectura de Lothar Collatz
//limpar a janela de gráfico
xbasc()
//gerar um número aleatório entre 1 e 10 mil milhões
x=grand(1,"unf",1,10^10);
xo=x;
//criar um vector para guardar a sequência de valores gerada
v=[];
//criar um contador para avaliar o número de valores gerado
n=0;
//aplicar a conjectura
while x>1
  if modulo(x,2)==0 then x=x/2;
  else
  x=3*x+1;
  end
  //aumentar o contador
  n=n+1;
  //guardar o novo valor no vector para o gráfico
  v=[v x];
end
t=1:1:n;
//fazer um gráfico do logaritmo natural dos valores
plot2d(t,[log(v)]')
xtitle("Conjectura de Collatz","Número de ordem dos números gerados","Logaritmo natural dos números gerados");
disp("Número de valores da sequência, número inicial e último valor")
disp([n xo x])
```

Um exemplo da saída desta listagem:

```
Número de valores da sequência, número inicial e último valor

  563.    8.147D+09    1.

exec done
```

O gráfico do vector dos números gerados insere-se na figura 7.10, depois de editado.

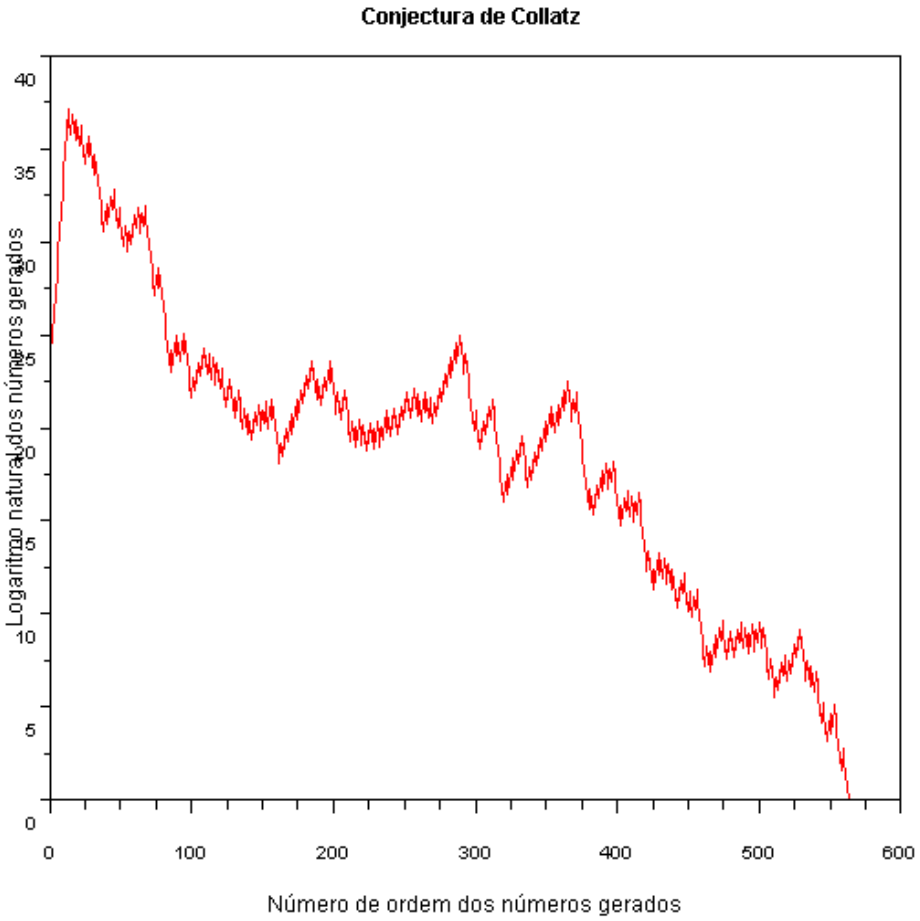


Figura 7.10. Gráfico do logaritmo natural dos valores da sequência da conjectura de Collatz.

7.10 O comando break

O comando break dentro dum ciclo for ou while interrompe a execução do ciclo mesmo que as suas condições sejam verdadeiras. Introduzamos o comando break no programa anterior do seguinte modo:

```
//Calcular as potências de três inferiroes a 5000
//Criar o vector vazio para as potências de 3
pot3=[];
//primeiro expoente
n=0;
//valor inicial para num3
num3=0;
while num3<2000
  num3=3^n;
```

```
  //introduzir o comando break
  if num3>500 then break;
  end
  pot3=[pot3 num3];
  //aumentar o expoente
  n=n+1;
end
pot3
```

```
 pot3  =

    1.    3.    9.    27.    81.    243.
```

Sugiro ao leitor que tente estabelecer o fluxograma e o pseudocódigo para o programa anterior.

7.11 Erros nos programas

Quando existe um erro num programa, o Scilab suspende a execução do programa e escreve uma mensagem de erro, na sua janela. Um exemplo:

```
-->a=matrix([1:12],3,4);

-->D=det(a);
        !--error 20
first argument must be square matrix
```

O Scilab tem vários tipos de erros identificados por um número. É possível obter a tabela de identificação dos erros com o comando help error_table.

O comando pause suspende a execução do programa e permite entrar comandos, num **prompt** que numera as interrupções e abre um novo "ambiente" local.

Para voltar ao ambiente de computação anterior entrar com return ou resume.

A execução de uma programa pode ser interrompida com o comando abort.

A leitora deve explorar o menu "control" da janela do Scilab, se ainda não o fez.

7.12 Quadro sinóptico dos comandos introduzidos neste capítulo

Comando	Descrição
abort	Interrompe a execução do programa

Comando	Descrição
clearglobal	Apaga as variáveis globais
excel2sci	Lê folhas de cálculo em formato de texto (*.txt)
for	Comando para executar ciclos
full	Converte matrizes esparsas em matrizes completas
global	Define variáveis globais
hypermat	Cria matriz multidimensional
if	Cria um Comando de execução condicional
input	Pede uma entrada de informação através do teclado
isglobal	Verifica se uma variável é global
list	Cria um objecto list (lista)
mgetl	Lê linhas de texto de um ficheiro *.txt
pause	Interrompe o program e aguarda uma entrada através do teclado
printf	Comando para imprimir no monitor
printsetupbox	Comando para ter acesso à jenela de controlo das impressoras
resume	Retoma a execução interrompida de um programa
return	Retoma a execução interrompida de um programa
select	Comando de escolha condicional
sparse	*Cria uma matriz esparsa*
tk_getdir()	Obtém o caminho para aceder a um ficheiro
tlist	Cria um objecto do tipo tlist
type	Verifica o tipo da variável
typeof	Verifica o tipo de um objecto Scilab
xls_open	Abre um ficheiro em formato do Excel
xls_read	Lê um ficheiro Excel previamente aberto
x_message	cria uma janela para mensagem
while	Comando para ciclo condicional

Capítulo 8

Aplicações

Vou agora apresentar aplicações do Scilab a vários problemas, alguns deles requerendo conhecimentos de um nível superior aos implícitos naqueles resolvidos no capítulo 2. Este livro é de iniciação ao Scilab, e mais nada pretendo ensinar nele, por isso assumo que o leitor tem a formação necessária para abordar os modelos que lhe interessarem, que serão no entanto acompanhados de um enquadramento mínimo e de pelo menos uma referência bibliográfica atinente e relevante.

Numa escolha certamente arbitrária, apresentarei os seguinte doze programas:

- reglinmul – ajusta modelos da regressão linear multivariada
- RK4 – aplica o método de Runge-Kuta de 4ª ordem a um sistema de EDO
- sbpred11 – acha o equilíbrio do modelo SBPRED(1,1) para a interacção predador-presa e analisa a sua estabilidade
- paramcaos – aplica o modelo BACO2 a populações de *Paramecium* sp. E ilustra a ocorrência do efeito de borboleta determinístico
- sematdial – dada uma matriz de Leslie, estima a sensibilidade e a elasticidade do seu valor próprio dominante, os valores reprodutivos das classes e a distribuição etária estável
- SOBANPK – simula os ciclos biogeoquímicos do N, K e P no pinhal bravo regular
- Cnobravo – simula o carbono retido no pinhal bravo e a sua acreção
- plantafu – simula o crescimento de uma planta anual no hemisfério norte, com selecção de data de sementeira e latitud
- funcroc – simula o crescimento de uma população de crocodilos australianos
- ProbExt – calcula a probabilidade acumulada de extinção de uma espécie
- plamarg – simula o planeta das margaridas de James Lovelock
- fuMBE2 – um modelo de balanço da energia da Terra

Todos os programas, em formato *.sci, encontram-se na pasta "Aplicações", que acompanha este livro. Para os correr basta **File→Exec...**. e dois cliques no ficheiro escolhido.

8.1 Programa reglinmul

A regressão linear múltipla é um dos instrumentos estatísticos mais utilizados, por isso vou apresentar um programa que ajusta este modelo. São inúmeros os livros onde se explana a abordagem matricial deste tipo de regressão. Recorri a Myers (1990).

Genericamente, o modelo escreve-se:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... \beta_n x_n + \varepsilon$$

A listagem do programa insere-se abaixo. Os dados que usa, com o propósito de controlo, são respigados de Myers (1990), página 174 e seguintes. Os resultados deste autor são reproduzidos pelo reglinmul.

Além os estatísticos referentes aos coeficientes de regressão, são também estimados R^2. R_{aj}^2, análise da variância, e os estatísticos PRESS, AIC e BIC.

Da saída do programa constam duas janelas de gráficos. A primeira apresenta quatro gráficos identificados e comumente exibidos ao se analisarem os resíduos (figura 8.1). Esta janela pode-se manter durante o tempo que se quiser, mesmo copiar, com a execução do programa suspensa. Para retomar os cálculos entre com return e enter. Surge outro gráfico (figura 8.2) que compara um histograma dos resíduos com a distribuição normal associada, e na janela dos comandos a análise descritiva dos resíduos.

Deixo como desafio à leitora, seleccionar a combinação das variáveis independentes que minimiza o PRESS.

O leitor pode modificar o início do programa de odo a que ele possa aceitar outros dados (secção 7.4). O programa não recorre a ciclos o que acelera a sua execução-

Eis a listagem do programa:

```
//© L. S. Barreto, 2007.
//Programa reglinmul
//Este programa estabelece uma regressão linear múltipla
//Abordagem matricial
//Validado com dados de Myers (1990), tab. 4.1, p.174
xbasc()
//dados
x=[1 5.5 31 10 8; 1 2.5 55 8 6; 1 8.0 67 12 9;...
  1 3.0 50 7 16;1 3.0 38 8 15;1 2.9 71 12 17;...
  1 8.0 30 12 8;1 9.0 56 5 10;1 4.0 42 8 4;...
  1 6.5 73 5 16;1 5.5 60 11 7;1 5.0 44 12 12;..
  1 6.0 50 6 6;1 5.0 39 10 4;1 3.5 55 10 4];
x(:,[])=[];//tirar coluna(s)
y=[79.3 200.1 163.2 200.1 146.0 177.7 30.9 291.9 160.0...
  339.4 159.6 86.3  237.5 107.2 155.0];
```

```
s1=size(x);n=s1(1);k=s1(2);p=k-1;
//Cálculo dos coeficeintes de regressão
b=inv((x'*x))*x'*y';
yhat=x*b;
//Cálculo dos resíduos
disp("Resíduos");e=(y'-yhat)
//Cálculo dos estatísticos da qualidade da regressão
rss=e'*e;
rnu=sum((yhat-mean(y))^2);rden=sum((y-mean(y))^2);
r2=rnu/rden;
rss=rden-rnu;s2=rss/(n-p-1);
ssreg=rnu/p;F=ssreg/s2;
r2aj=1-(s2*(n-1)/rden);df1=p;df2=n-p-1;
c=inv(x'*x);
cii=diag(c);seb=sqrt(s2*cii);
t=b./seb;tt=abs(t);a=ones(k,1)*(n-k);
[P,Q]=cdft("PQ",tt,a);
pt=2*Q;
//Imprimir resultados
disp([" Betas erro t p"])
disp([b seb t pt])
disp([" R2 R2aj s2"]);
disp([r2,r2aj,s2])
disp([" SQmodelo SQerro SQtotal F p"])
[P,Q]=cdff("PQ",F,df1,df2);
disp([rnu, rss, rden, F,Q])
//Cálculo do PRESs, AIC e BIC
g=[];
h=x*inv(x'*x)*x';
hii=diag(h);su=(ones(n,1)-hii);
g=(e./su)^2;
d=g.*hii/(s2*p);
sr=[];
su2=sqrt(su);
```

```
sr=sqrt(abs(e./(sqrt(s2)*su2)));
PR=sum(g);AIC=log(rss/n)+2*k/n;nob=1:1:15;
bi=log(rss/n)+k/n*log(n);
disp([" PRESS AIC BIC"]);
disp([PR,AIC,bi])
//Preparar e fazer gráficos
z=zeros(15,1);
p1=[z e];p2=[z d];p3=[z sr];
subplot(221);
plot2d3(yhat,[p1]);
xtitle("Resíduos vs. predições","Predições","Resíduos");
subplot(224);
plot2d3(nob,[p2]);
xtitle("Distâncias de Cook, Dc","Nº da observação","Dc");
subplot(223);
plot2d3(yhat,[p3]);
xtitle("Localização de escala","Predição","(Res. estand.)^1/2");
subplot(222);
histplot([-10:5:10],e);
xtitle("Histograma dos resíduos","Classes","Frequência");
xx=x;
//paragem para ver gráficos,
//para continuar teclar return + enter
pause
clf
c=e;
a=size(c);n=a(1,1);
miu=mean(c);st=(st_deviation(c));var=st^2;
meanc=ones(n,1)*miu;
m3=sum((c-meanc)^3)/n;
m4=sum((c-meanc)^4)/n;
assim=m3/st^3;
excd=m4/st^4-3;
disp("Resíduos")
```

```
disp([" med var assim excd"])
disp([miu;var;assim;excd]')
x=-20:1:20;
nor=1/(sqrt(var*2*%pi))*exp(-(x-miu)^2/(2*var));
plot2d(x,[nor])
xtitle("Comparação distribuição dos resíduos-
normal","Valores","Frequência")
histplot([-20:4:20],c)
x=xx;
resume
```

```
 Resíduos
 e  =

   0.4119052
 - 1.0993919
 - 4.3689851
 - 8.688148
   2.0298928
   5.4187247
 - 5.5589343
   5.7670654
   1.9465496
 - 2.6748748
   0.4713782
   6.005349
   2.0782691
   1.9070532
 - 3.645853

    Betas          erro              t              p

   177.22856    8.7873825    20.168527    1.977D-09
   2.1702103    0.6736981    3.2213393    0.0091543
   3.5380138    0.1091521    32.413626    1.841D-11
```

```
 - 22.158336    0.5453732  - 40.629674    1.953D-12
   0.2035384    0.3188579    0.6383358    0.5375986

         R2           R2aj           s2

    0.9970734    0.9959027    26.207283

  SQmodelo       SQerro       SQtotal          F                p

    89285.045    262.07283    89547.117    851.71976    1.285D-12

      PRESS          AIC           BIC

    741.75573    3.5272389    3.7632557

-1->return

 Resíduos

        med           var          assim          excd

    1.037D-13    18.719488  - 0.3132837  - 0.9874243

 exec done
```

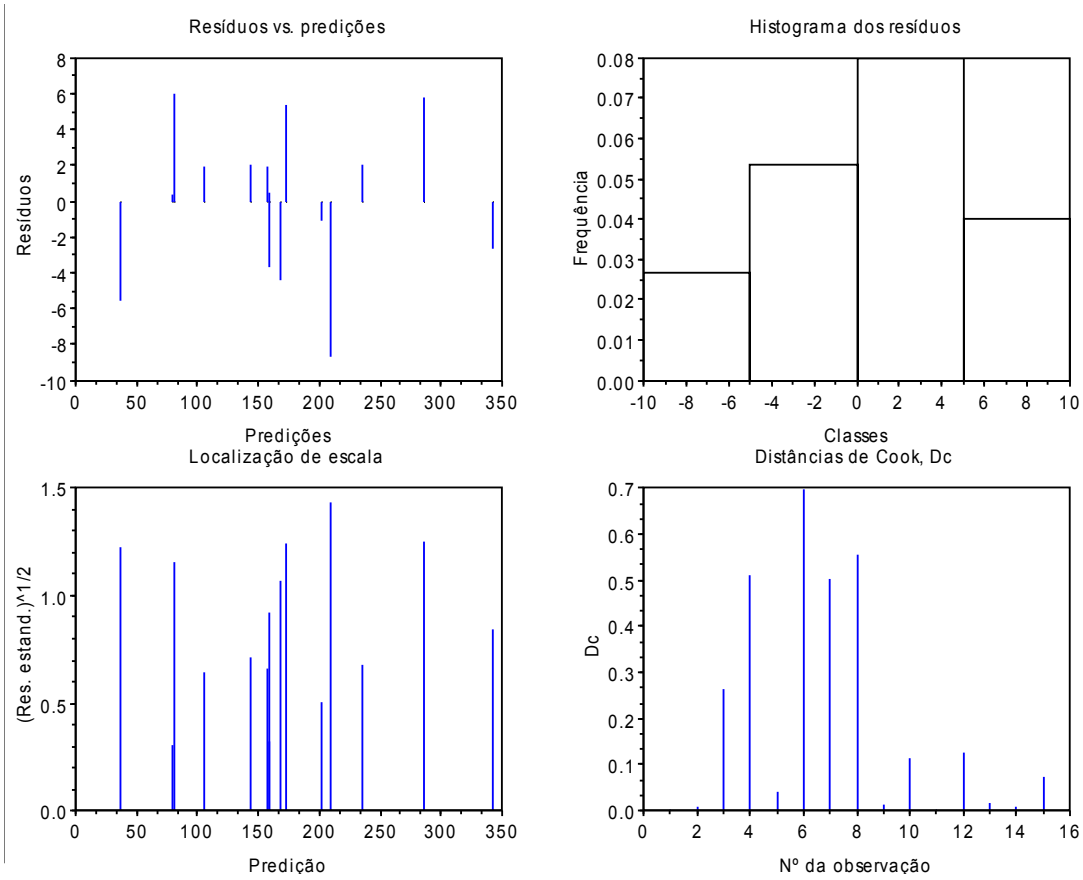


Figura 8.1. Primeira saída de gráficos do programa reglinmul

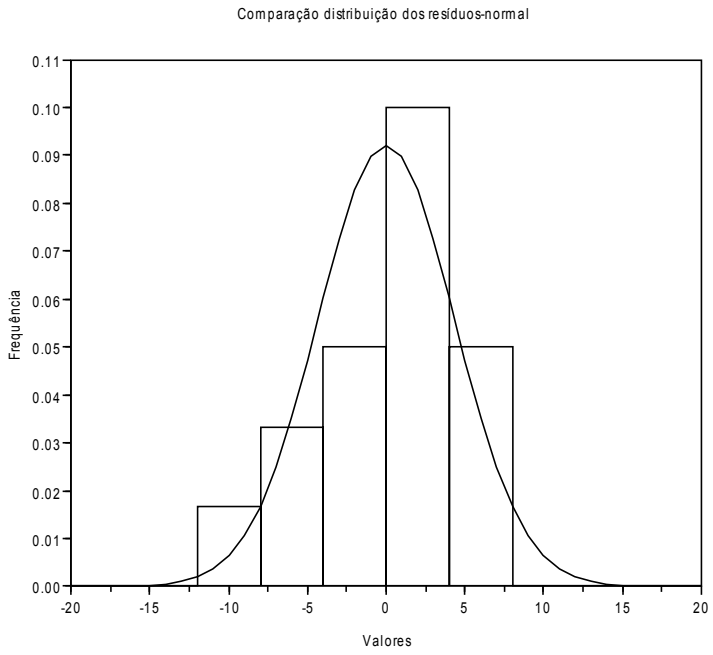


Figura 8.2. Segunda saída de gráficos do programa reglinmul. Comparação da distribuição dos resíduos com a distribuição normal

8.2 Programa RK4

Se a leitora não está familiarizado com o método de Runge-Kuta de 4ª ordem, para resolver numericamente equações diferenciais, ele vem descrito tanto em obras que tratam de equações diferenciais, como de análise numérica (por exemplo, Antia, 2000). O programa RK4 aplica este método.

O programa RK4 ilustra a utilização de um vector de "strings" cujos elementos são duas equações diferenciais (já nossas conhecidas, com constantes diferentes), que depois o comando eval se encarrega de calcular. A saída do programa exibe-se na figura 8.3.

```
//© L. S. Barreto, 2007.Programa RK4
//Aplica o método de Runge-Kuta 4ª ordem
//a um sistema de 2 EDO
dx=[];x=[];x0=[];k1=[];k2=[];k3=[];k4=[];v=[];
//equações diferenciais
dx(1)='x(1)*(-0.1+0.002*x(2))';
dx(2)='x(2)*(-0.0025*x(1)+0.2)';
//valores iniciais
x0(1)=40;x0(2)=40;
//peíodos da solução
n=150;
//número de equações
m=2;
//intervalo de discretização
h=1;
a=1;
for i=1:h:n
   x=x0;
   k1=eval(dx);
   x=x0+h/2*k1;
   k2=eval(dx);
   x=x0+h/2*k2;
   k3=eval(dx);
   x=x0+h*k3;
   k4=eval(dx);
   for i=1:m
     x(i)=x0(i)+h*1/6*(k1(i)+2*(k2(i)+k3(i))+k4(i));
```

```
    end
    x0=x;
    for i=1:m
     v(a,i)=x0(i)+h*1/6*(k1(i)+2*(k2(i)+k3(i))+k4(i));
    end
    a=a+1;
end
for i=1:m
plot(v(:,i))
end
```

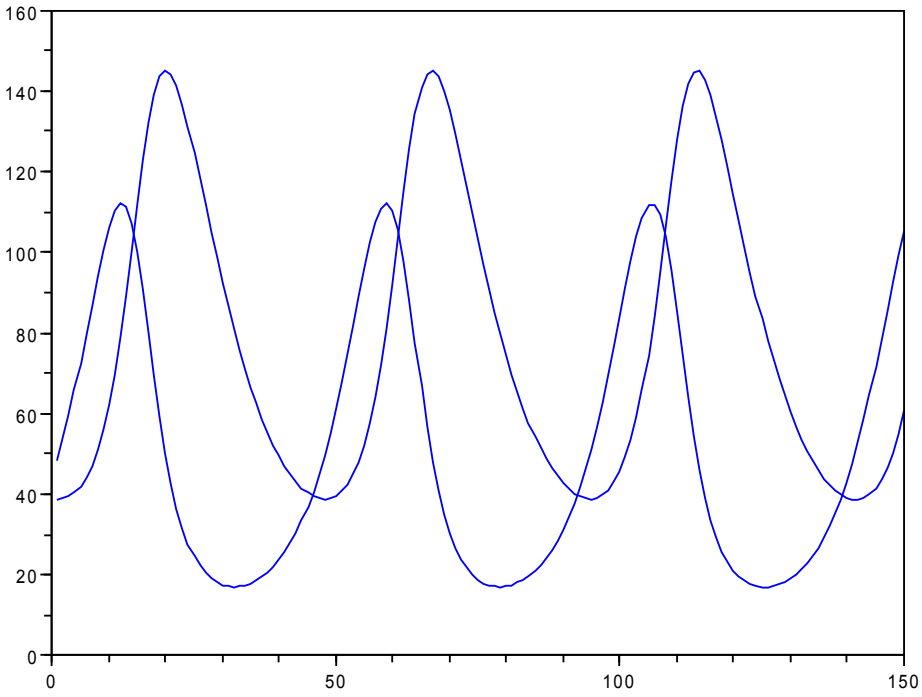


Figura 8.3. Saída do programa RK4.

Leitor: para que serve a variável a?

Pode eliminar as equações solucionadas e inserir um procedimento para aceitar outras quaisquer. Formate a figura 8.3.

Se se interessa por equações diferenciais, veja na ajuda do Scilab os comandos champ e fchamp.

8.3 Programa sbpred11

Em Barreto(2005b), propus o seguinte modelo para a interacção presa-predador, que designei sbpred(1,1):

$$\frac{dy_1}{dt} = c_1 y_1(\ln(k_1) - \ln(y_1)) - \frac{a_{11} y_1 x_1}{1 + a_{11} h_{11} y_1}$$

$$\frac{dx_1}{dt} = c_1 m_{11} x_1(\ln(b_{11} y_1) - \ln(x_j))$$

Obviamente, a variável y representa a presa e x o predador. Basicamente envolve duas formas diferencias da equação de Gompertz e uma resposta funcional de Holling do tipo II.

O programa sbpred(1,1) estima os pontos de equilíbrio e analisa a estabilidade da solução. Tem a seguinte listagem:

```
//© L. S. Barreto, 2008.Programa sbpred11
//Programa sbpred11
//Modelo sbpred de Barreto para a interacção presa-predador
//Acha o ponto de equilíbrio e analisa a sua estabilidade
//c1=0.5;m=5.12;h=0.1;a=5;b=0.5;k1=19
disp("[]=g(c1,m,h,a,b,k1)")
m=1e-10;
function []=g(c1,m,h,a,b,k1)
printf("c1=%1.3f, m=%1.3f, h=%1.3f, a=%1.3f, b=%1.3f, k=%i", c1,m,h,a,b,k1)
deff('x=f(y)','x=[c1*y(1)*(log(k1)-log(y(1)))-a*y(1)*y(2)/(1+a*h*y(1));...
c1*m*y(1)*(log(b*y(1))-log(y(2)))]')
disp("Equilibrio")
[y]=fsolve([7,0.3],f);
disp(y)
y0=[y(1);y(2)];
disp("Jacobiano")
[J]=derivative(f,y0);
disp(J)
z=poly(J,"x");
disp("Valores próprios");
```

```
v=roots(z);
disp(v)
if real(v(1))<0 & real(v(2))<0 then disp("Equilíbrio estável")
elseif disp("Equilíbrio insttável")
end
endfunction
```

Eis um exemplo da saída do programa:

```
c1=0.500, m=5.000, h=1.000, a=1.000, b=0.400, k=20
 Equilibrio

   9.6852294    3.8740918

 Jacobiano

 - 4.0110722  - 9.6846614
   2.5        - 6.25

 Valores próprios

 - 5.1305361 + 4.7914981i
 - 5.1305361 – 4.7914981i

 Equilíbrio estável
```

8.4 Programa paramcaos

Em Barreto (2004b) apresentei o meu modelo BACO2 para a competição entre plantas. Assenta na seguinte concepção da competição:

A1. As competições intra e interespecíficas são fenómenos

A2. O maior efeitoa da competição é ao nível da população.

A3. A competição afecta a alometria da população e o seu padrão de crescimento, proporcionalmente à diferença da capacidade competitiva das espécies. Quanto mais próximas forem estas menores são as perturbações no crescimento das populações.

A4. A varição relativa da variável seleccionada, z, para descrever a população numa mistura (número ou biomassa), no caso da população a, escrita como $RVR_a=1/y_a\ dy_a/dt$, é afectada

pela proporção das outras espécies na mistura e o logaritmo natural da razão da variação da outra /numerador) espécie e da espécie a (denominador). Generalizando para uma mistura de n espécies, na idade t escrevo:

$$^{z}RVR_{at} = RVR_{azt}(1+\sum_{j=1}^{n}(\frac{y_j}{\sum_{j=1}^{n} y_j}\ln(r_{ajt})))$$

onde

r_{ajt}= (variação relativa da espécie j na idade t) / variação relativa da espécie a na idade t).

Em Barreto (2005a, b) a leitora pode encontrar informação complementar sobre o modelo BACO2, e a sua aplicação à análise de experiências de competição com *Paramecium* sp., que em culturas puras crescem em número de acordo com a curva de Gompertz.

O programa paramcaos simula a dinâmica de misturas de *P. aurelia* e *P. caudatum,* com combinações de vários tamanhos iniciais das populações e revela graficamente que pequenas diferenças nos tamanhos iniiais são ampliadas durante o crescimento das populações, como se exibe na figura 8.4, com o gráfico que produz.

A listagem do programa é a seguinte:

```
//(© L. S. Barreto,  2008  Programa paramcaos
//Aplica o modelo BACO2 a duas populações de ciliados
// com vários valores iniciais
//e ilustra o efeito de borboleta determinístico
//gerar combinações de valores iniciais recorrendo a dois comandos for
i=1;
for p01=22:80:302
  for p02=22:80:302
  //valores característicos das espécies de paramécia utilizadas
  c1=0.297;c2=0.253;r1=0.0033;r2=0.0090;k1=602;k2=223;
  a=[];
  w1=p01;w2=p02;
  cop1=-c1*log(r1);
  cop2=-c2*log(r2);
  p1=p01;p2=p02;yy=1;
  for q=0:0.01:34
     g=p1+p2;
     x=yy/100;
```

```
    if (yy/100+1)==int(yy/100+1) then a(x,1)=p1;
    end
    if (yy/100+1)==int(yy/100+1) then a(x,2)=p2;
    end
    //cálculo das variações relativas de cada espécie
    ep1=exp(-c1*q);ep2=exp(-c2*q);
    rm1=cop1*ep1;rm2=cop2*ep2;
    //logaritmos das razões das variações relatvas das espécies
    f1=log(rm2/rm1);f2=log(rm1/rm2);
    g1=p2/g;g2=p1/g ;// proporções de cada espécie
    //variações das espécies
    p1=p1*(1+0.01*rm1*(1+g1*f1));
    p2=p2*(1+0.01*rm2*(1+g2*f2));
    //coeficientes de competição entre as espécies
    //a(ij) i=esp. afectada, j=esp. que afecta
    //cuja saída pode ser pedida
    a12=(-c1*w1*(log(k1)-log(w1))+(p1-w1))/(c1*w1*log(w2));
    a21=(-c2*w2*(log(k2)-log(w2))+(p2-w2))/(c2*w2*log(w1));
    w1=p1;w2=p2;
    yy=yy+1;
  end
  // criar os vectores das populações da combinação das densidades iniciais
  [p1]=a(:,1)';[p2]=a(:,2)';
  plot2d4(p1,[p2],[i]);
  xtitle("Dinâmica das populações de P. aurelia e P caudatum, para vários
  valores iniciais","P.aurelia","P. caudatum");
  xgrid();
  i=i+1;
  end
end
```

Muda a cor das linhas

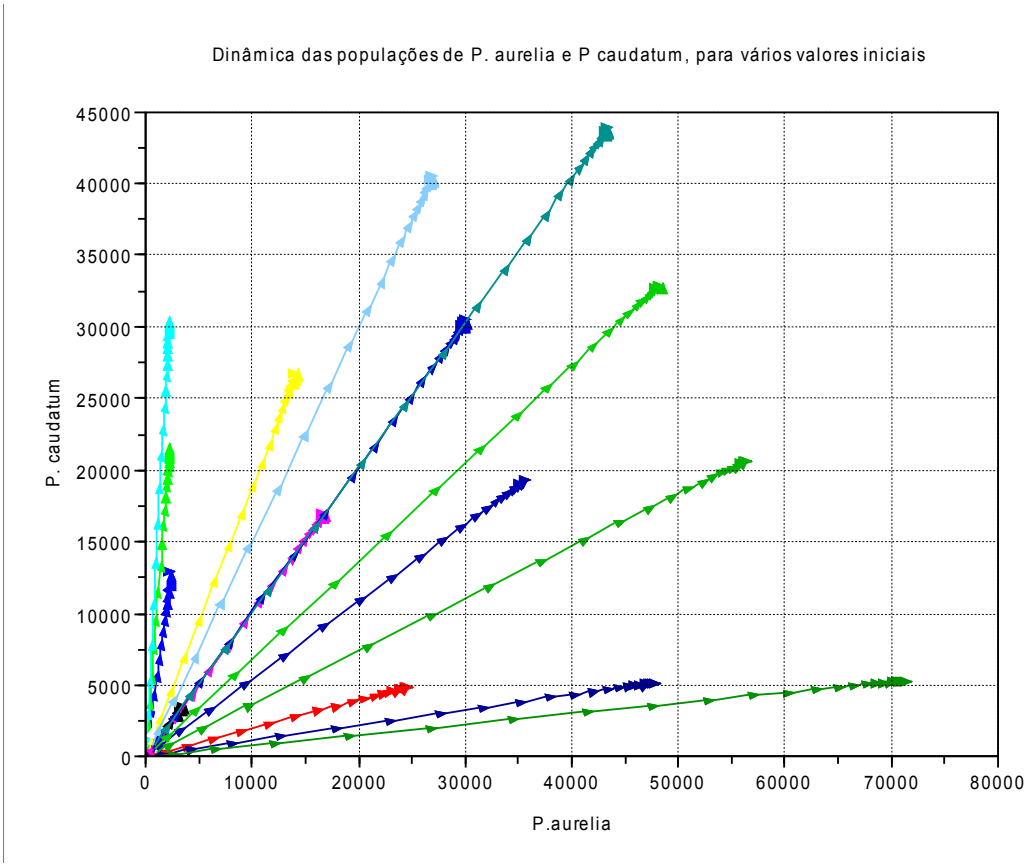


Figura 8.4. Gráfico criado pelo programa paramcaos. Ilustra o efeito de borboleta determinístico em situações de competição interespecífica.

8.5 Programa sematdial

Este programa está relacionado com a matriz de Leslie para populações estruturadas. Caswell (2001) é a mais completa obra que conheço sobre modelos matriciais de populações.

O programa sematdial calcula os valores próprios da matriz, as matrizes da sensibilidade e elasticidade do valor próprio dominante, os valores reprodutivos das classes e a distribuição etária estável.

```
//© L. S. Barreto, 2007. Programa sematdial
//Dada uma matriz de Leslie acha os valores próprios
//As suas matrizes da sensibilidade e elasticidade,
//os valores reprodutivos das classes e
//a distribuição etária estável
// a matriz da minha autoria
```

```
//diz respeito ao veado-mula americano
//classes de idade de 3 anos
a=[0.546      0.656  0.7015       0.743  0.743;
0.133642      0.534924      0       0       0;
0       0.213302      0.604583      0       0;
0       0       0.266801      0.636378      0;
0       0       0       0.297919      0.651575];
valpr=spec(a);
[ab,x,bs]=bdiag(a,1/%eps);
[ic,ir]=find(ab'==max(ab'));
w=conj(inv(x));
v1=x(:,ic);
w1=real(w(ic,:))';
s=w1*v1';
VR=v1*1/v1(1,1);
tot=sum(w1);
eee=1/tot*w1;
h=sort(spec(a));
el=s.*a/h(1,1);
disp("Sensibilidade")
disp(s)
disp("Elasticidade")
disp(el)
disp("Valores próprios")
disp(h)
disp("Valor reprodutivo")
disp(VR)
disp("Distribuição etária estável")
disp(eee)
```

O leitor pode modificar o programa de modo a aceitar qualquer matriz de Leslie que se queira analisar.

A saída do programa é a seguinte:

```
 Sensibilidade
```

```
 0.3043126   0.0874455   0.0471710   0.0346107   0.0295935
 1.0337951   0.2970653   0.1602469   0.1175777   0.1005337
 1.3181592   0.3787785   0.2043257   0.1499195   0.1281873
 1.1534797   0.3314571   0.178799    0.1311899   0.1121727
 0.6489292   0.1864725   0.1005894   0.0738053   0.0631065
```

Elasticidade

```
 0.1661544   0.0573641   0.0330904   0.0257157   0.0219880
 0.1381582   0.1589071   0           0           0
 0           0.0807941   0.1235316   0           0
 0           0           0.0477037   0.0834862   0
 0           0           0           0.0219880   0.0411186
```

Valores próprios

```
 1.0000017
 0.6429011 + 0.2182143i
 0.6429011 - 0.2182143i
 0.3438280 + 0.0812467i
 0.3438280 - 0.0812467i
```

Valor reprodutivo

```
 1.
 0.2873541
 0.1550084
 0.1137340
 0.0972472
```

Distribuição etária estável

```
 0.0682518
 0.2318615
```

```
 0.2956392
 0.2587046
 0.1455430

exec done
```

8.6 Programa SOBANPK

Este programa é uma versão em Scilab de um simulador anteriormente apresentado, em Visual Basic 6. Sobre o fundamento do simulador veja-se Barreto (2004a).

O programa SOBANKP para um pinhal bravo regular, auto-desbastado, com compasso quadrado, calcula a biomassa total, os pesos de azoto, potássios e fósforo retidos na biomassa, absorvidos e restituídos em cada ano. Pede que entre com a classe de qualidade do local onde cresce o pinhal e o seu factor de Wilson.

A altura dominante é usada como indicador da bondade (genericamente, fertilidade) do local onde a floresta existe, para o seu crescimento. Uma forma de a medir é achando a média da altura das cem árvores de maior diâmetro à altura do peito, num hectare.

O factor de Wilson é uma medida de espaçamento relativo das árvores e é um indicador da ocupação do espaço pelas árvores.

Na idade t, este factor é dado pela relação seguinte:

$$f_t = 100/(h_t N_t^{1/2})$$

N_t representa o número de árvores num hectare, isto é, a densidade do povoamento. Manifestamente, quanto maior for a densidade menor é f. O factor de Wilson é constante, num povoamento florestal auto-desbastado, em que as árvores são todas da mesma espécie e praticamente da mesma idade.

A relação entre os pesos do nutrientes envolvidos é N>K>P.

A listagem do programa é a seguinte:

```
//(c) L. S. Barreto, 2008 Programa SOBANPK
//Estabelece os ciclos biogeoquímicos de
//azoto, potássio e fósforo
//h=altura dominante aos 40 anos, f=factor de Wilson,
// admite espaçamento quadrado;
disp("f(altura dominante aos 40 ano (16-26), Factor de Wilson (0.17-0.30)")
function []=f(h,f)
xbasc()
//calcular índice de performance
s = h * f; pf = 6700.12 / s ^ 2;
if h >= 22 then k = (-1625 * f + 939.75) / 100;
end
if 18 <= h & h < 22 then k = (-1450 * f + 876.5) / 100;
end
if h < 18 then k = (-1225 * f + 790.75) / 100;
```

```
end
df = k * s * 1.221883;
vf = 32.388 + .405 * (df / 200) ^ 2 * 3.14159 * pf * 24 * 1.221883;
// biomassas da copa a=agulhas rv=ramos vivos rm=ramso mortos t/ha
a = .07851 * vf ^ .87848; rv = .14369 * vf ^ .8789; rm = .04001 * vf ^ .85503;
cp = a + rv + rm;
//projecção do povoamento
//nutrientes acumulados na biomassa
sn = '276.367 * exp(.00192 * (vf * 0.4076^exp(-0.05*t)))';
sp = '29.391 * exp(.00179 * (vf * 0.4076^exp(-0.05*t)))';
sk = '163.222 * exp(.00216 * (vf * 0.4076^exp(-0.05*t)))';
//nutrientes absorvidos no ano
un = '120.508 * exp(-.00102 * (vf * 0.4076^exp(-0.05*t)))';
up = '13.166 * exp(-.00118 * (vf * 0.4076^exp(-0.05*t)))';
uk = '71.344 * exp(-.00132 * (vf * 0.4076^exp(-0.05*t)))';
//nutrientes restituídos no ano
rn = '109.508 * exp(-.00082 * (vf * 0.4076^exp(-0.05*t)))';
rp = '12.119 * exp(-.00101 * (vf * 0.4076^exp(-0.05*t)))';
rk = '63.377 * exp(-.00107 * (vf * 0.4076^exp(-0.05*t)))';
//produtividade primária líquida
//biomassa total
bt ='0.0769 * (vf * 0.4076^exp(-0.05*t)) ^ .9731+...
0.42029 * (vf * 0.4076^exp(-0.05*t)) ^ .98692...
 + 0.176 * (vf * 0.4076^exp(-0.05*t)) ^ .93056...
 + .07851 * vf ^ .87848+.14369 * vf ^ .8789+.04001 * vf ^ .85503';
t=0:1:50;z=10:1:60;
S=[eval(sn);eval(sp);eval(sk)]';
U=[eval(un);eval(up);eval(uk)]';
R=[eval(rn);eval(rp);eval(rk)]';
subplot(221);
xset ("thickness",2)
plot2d(z,[eval(bt)]);
xtitle("Biomassa total","Idade,anos","Mg/ha");
xgrid();
subplot(222)
plot2d(z,[S]);
xtitle("Acumulaçao de N, P, K","Idade, anos","kg/ha");
xgrid();
subplot(223);
plot2d(z,[U]);
xtitle("Absorção de N, P, K","Idade,anos","kg/ha/ano");
xgrid();
subplot(224)
plot2d(z,[R]);
xtitle("Restituição de N, P, K","Idade, anos","kg/ha/ano");
xgrid();
endfunction
```

Para a entrada f(20,0.19) o programa proporciona os gráficos da figura 8.5.

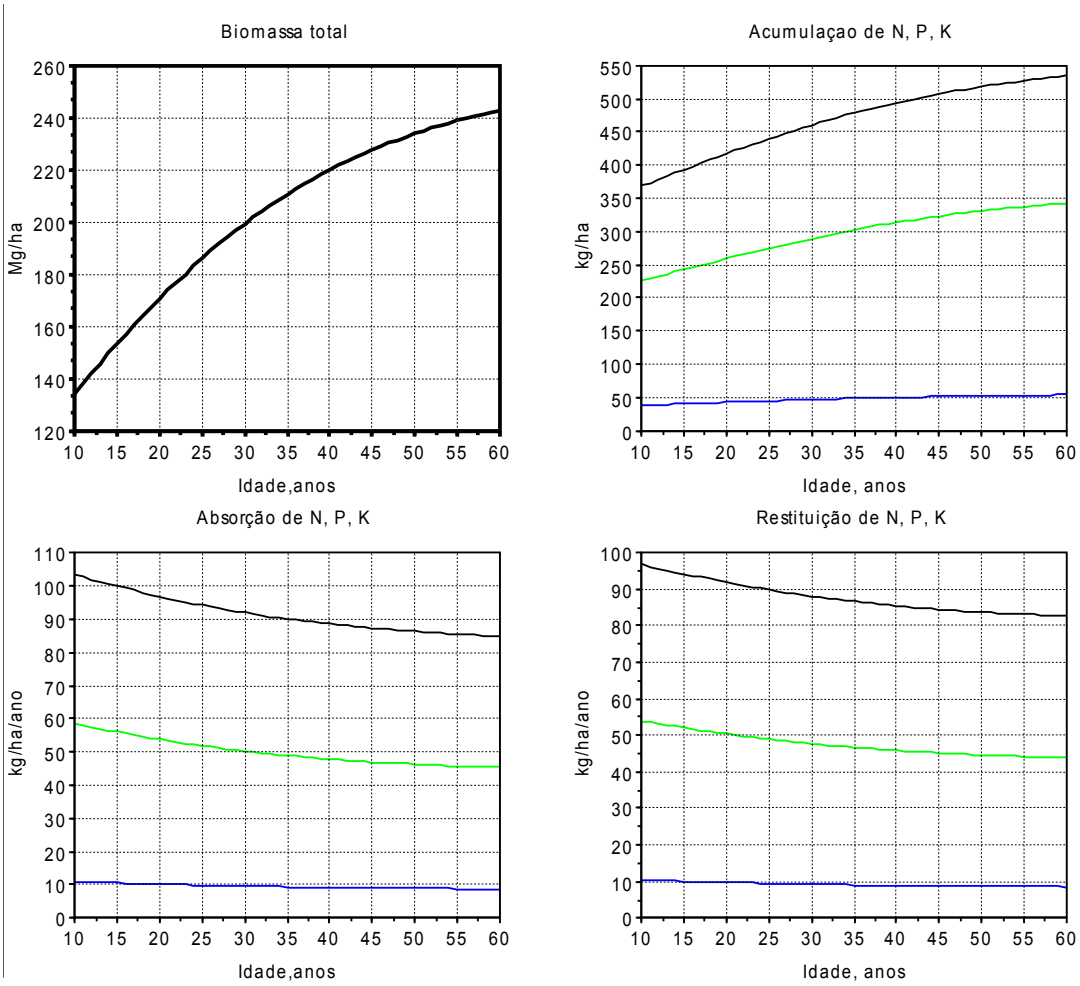


Figura 8.5. Saída do programa SOBANPK, para a entrada f(20,0.19).

8.7 Programa Cnobravo

O efeito estufa veio conferir um interesse difícil de suspeitar, décadas atrás, ao sequestro do carbono da atmosfera pela vegetação, nomeadamente pelas florestas.

Este programa calcula a produtividade primária líquida anual, biomassa total, e os seus teores em carbono em pinhais bravos regulares auto-desbastados, com compasso quadrado. É, de certo modo, uma sequela do anterior, e beneficia das aferições do carbono contido na biomassa de pinheiro bravo realizadas por Lopes e Aranha (2006).

A listagem do programa é a seguinte:

```
//© L. S. Barreto, 2007.Programa Cnobravo
//Estima o carbono no pinhal bravo regular
//h=altura dominante aos 40 anos, f=factor de Wilson,
// admite espaçamento quadrado;
```

```
disp("f(altura dominante aos 40 ano (16-26), Factor de Wilson (0.17-0.30)")
function []=f(h,f)
xbasc(0)
//Achar índice de performance
s = h * f; pf = 6700.12 / s ^ 2;
if h >= 22 then k = (-1625 * f + 939.75) / 100;
end
if 18 <= h & h < 22 then k = (-1450 * f + 876.5) / 100;
end
if h < 18 then k = (-1225 * f + 790.75) / 100;
end
//achar diâmetro final
df = k * s * 1.221883;
//Achar volume em é final
vf = 32.388 + .405 * (df / 200) ^ 2 * 3.14159 * pf * 24 * 1.221883;
//projecção do povoamento
//ppl=produtividade primária líquida
ppl ='51.13159*exp(-0.00251*(vf * 0.4076^exp(-0.05*t)))';
tr = '0.0769 * (vf * 0.4076^exp(-0.05*t)) ^ .9731+...
0.42029 * (vf * 0.4076^exp(-0.05*t)) ^ .98692';
//biomassa total
bt ='0.0769 * (vf * 0.4076^exp(-0.05*t)) ^ .9731+...
0.42029 * (vf * 0.4076^exp(-0.05*t)) ^ .98692...
 + 0.176 * (vf * 0.4076^exp(-0.05*t)) ^ .93056...
 + .07851 * vf ^ .87848+.14369 * vf ^ .8789+.04001 * vf ^ .85503';
t=0:1:50;z=10:1:60;
subplot(221);
plot2d(z,[eval(ppl)]);
xtitle("PPL","Idade,anos","Mg/ha/ano");
xgrid();
subplot(222)
plot2d(z,[0.457*eval(ppl)]);
xtitle("C na PPL","Idade, anos","Mg/ha/ano");
xgrid();
```

```
subplot(224);
plot2d(z,[0.457*eval(bt)]);
xtitle("C na biomassa total","Idade,anos","Mg/ha");
xgrid();
subplot(223)
plot2d(z,[eval(bt)]);
xtitle("Biomassa total","Idade-10, anos","Mg/ha");
xgrid();
endfunction
```

Novamente para f(20,0.19) o programa proporciona o gráfico da figura 8,6.

8.8 Programa plantafu

Há cerca de vinte anos atrás, na disciplina de Ecologia Florestal, no Instituto Superior de Agronomia, distribui pelos seus alunos um programa, em BASIC, chamado PLANTASB, que simulava o crescimento de uma planta anual, sendo as latitudes do hemisfério norte e datas de sementeira à escolha do utilizador, incluindo mesmo a antecipação do futuro efeito de estufa, de acordo com o conhecimento disponível então, sobre este assunto. A finalidade do programa era ilustrar o efeito do meio físico nos organismos.

O programa plantafu é uma versão modificada desse programa, agora em Scilab.

Na criação do programa PLANTASB, beneficiei da consulta de Brock (1981), Charles-Edwards *et al.* (1986), Hederson-Selers e McGuffie (1987), Jørgensen (1986).

O programa simula o crescimento de uma planta anual de dias longos, isto é, a floração só se inicia depois da duração do dia (quando o sol está acima da linha do horizonte) tiver ultrapassado um número de horas mínimo e crítico. A planta apresenta crescimento vegetativo até ao início da floração. Começada esta, a planta vai perdendo folhas até que morre. Tanto a transição do crescimento vegetativo para o reprodutivo como a germinação da semente são controladas pela temperatura.

O simulador é uma função da latitude, dia e mês da sementeira. As latitudes passíveis de escolha são 20, 30, 40, 50 N.

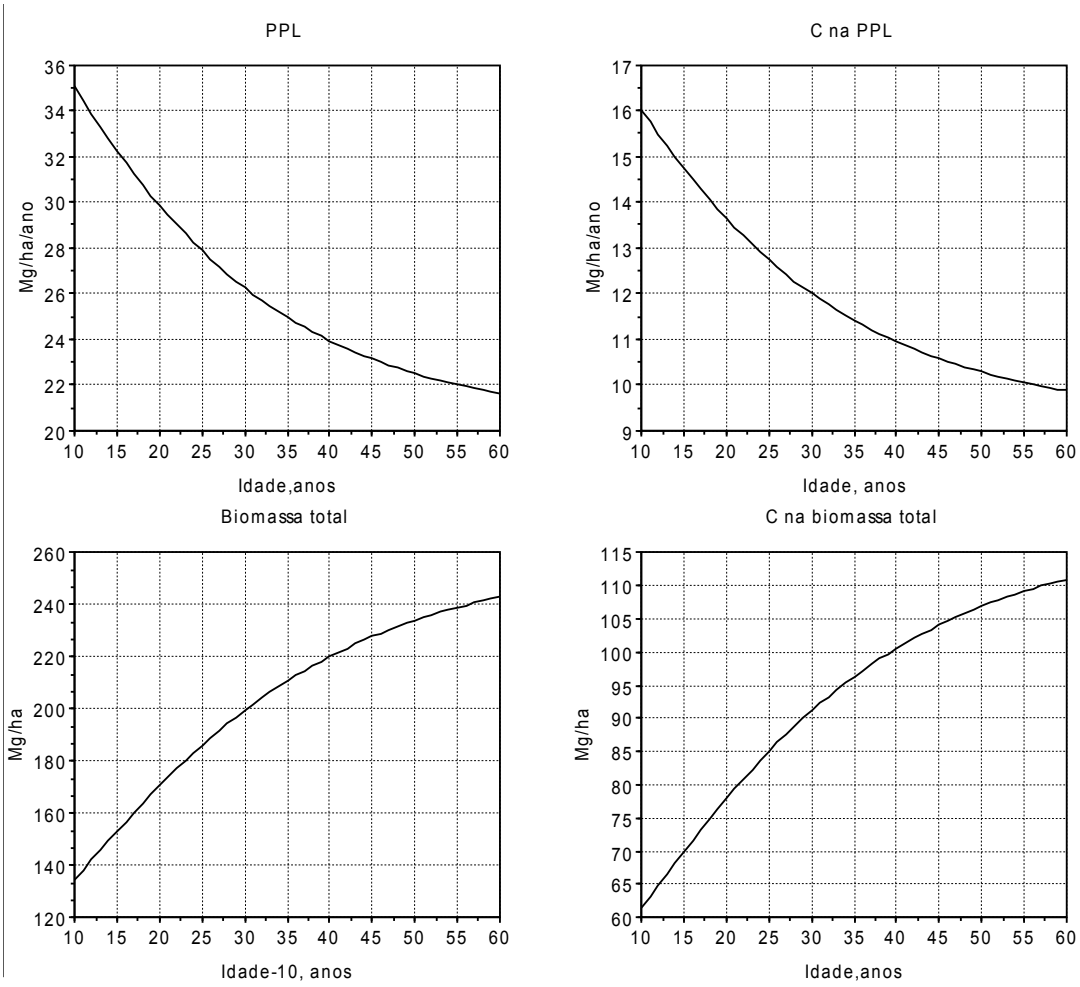


Figura 8.6. Saída do programa Cnobravo, para a entrada f(20,0.19)

A listagem desta função é a seguinte:

```
//(c) L.S. Barreto, 2008 Programa plantafu
//Variáveis
//lat=latitude norte; dj=dia juliano;
//pf=peso das folhas;pc=peso do caule;pr=peso das raízes;
//pfr=pso das sementes;
disp("lat=latitude:20,30,40,50 N;d=dia do mês;m=mês (1,2..12)")
disp("f(lat,d,m)")
function []=f(lat,d,m)
xbasc(0)
w=[];zz=1;M=[];
select lat
```

```
  case 20
  pf=2; pc = 2.73; pr = 2.72; pfr = .375;
  case 30
  pf = 1.6; pc = 2.2; pr = 2.2; pfr = .3;
  case 40
  pf = 2.8; pc = 3.85; pr = 3.85; pfr = .53;
  else
  pf = 3.27; pc = 4.36; pr = 4.36; pfr = .6;
end
select m
  case 1
  dj=d;
  case 2
  dj=d+31;
  else
  dj= int(30.6 * m + d - 32.4);
end
hsum = 0; tlag = lat / 2;
// germinar
tm = 32.5 - .45 * lat; td = tm * (.015 * lat - .1);
for i = dj:730
  if i>365 then i=i-365;
end
t=tm+td*sin(2*%pi*(i+283-tlag)/365)+0.75;
if t>15 then break;
end
end
dg=i+1;
if dg > 365 then dg=dg-365;
end
z = dg;
for j=dg:730
  t = tm + td * sin(2 * %pi * (j + 283 - tlag) / 365);
  hsum = hsum + (t - 10);
```

```
 if hsum > 350 then break
 end
 end
 df = j + 1; z = df;
 dff = df
 //vegetar
 f1=df-dg;
 for i = 1:f1
   trc = 1 + .214 * exp(-.051 * i);
   pf = pf * trc; pc = pc * trc; pr = pr * trc;
   w(zz,1)=pf;w(zz,2)=pc;w(zz,3)=pr;
   zz=zz+1;
 end
 [k]=w(:,1)';[m]=w(:,2)';[n]=w(:,3)';x=1:f1;
 [M]=[k;m;n]';
 //reproduzir:
 f2=df - dg + 1;G=[];ww=1;
   for i = f2:365
   trc = 1 + .642 * exp(-.051 * i);
   pf = .9 * pf; pc = pc; pr = pr; pfr = pfr * trc;
   if pf < .1 then break;
 end
 G(ww,1)=pfr;
 ww=ww+1;
end
[K]=G(:,1)';
xset("font",2,3)
subplot(211)
plot2d(x,[M]);
xtitle("Fase vegetativa","dias","gramas de raiz, caule e folhas");
xgrid();
subplot(212)
plot(K);
xtitle("Fase reprodutiva subsquente","dias","Gramas de semente");
```

```
xgrid();
a='Duração da fase vegetativa: ';aa=string(f1);a2=" dias";
b='Duração da fase reprodutiva: ';bb=string(ww-1);
c='Vida da planta: ';cc=string(f1+ww-1);
se1="Peso da semente: ";se2=string(pfr);se3=" gramas";
disp([a,aa, a2;b,bb, a2;c,cc a2;se1,se2,se3])
endfunction
```

Um exemplo da saída do programa é a seguinte:

```
 lat=latitude:20,30,40,50 N;d=dia do mês;m=mês (1,2..12)

 f(lat,d,m)

 exec done

-->f(40,17,3)

!Duração da fase vegetativa:    49         dias    !
!                                !
!Duração da fase reprodutiva:   65         dias    !
!                                !
!Vida da planta:                114        dias    !
!                                !
!Peso da semente:            1.3852013   gramas  !
```

A esta saída está associado o gráfico da figura 8.7.

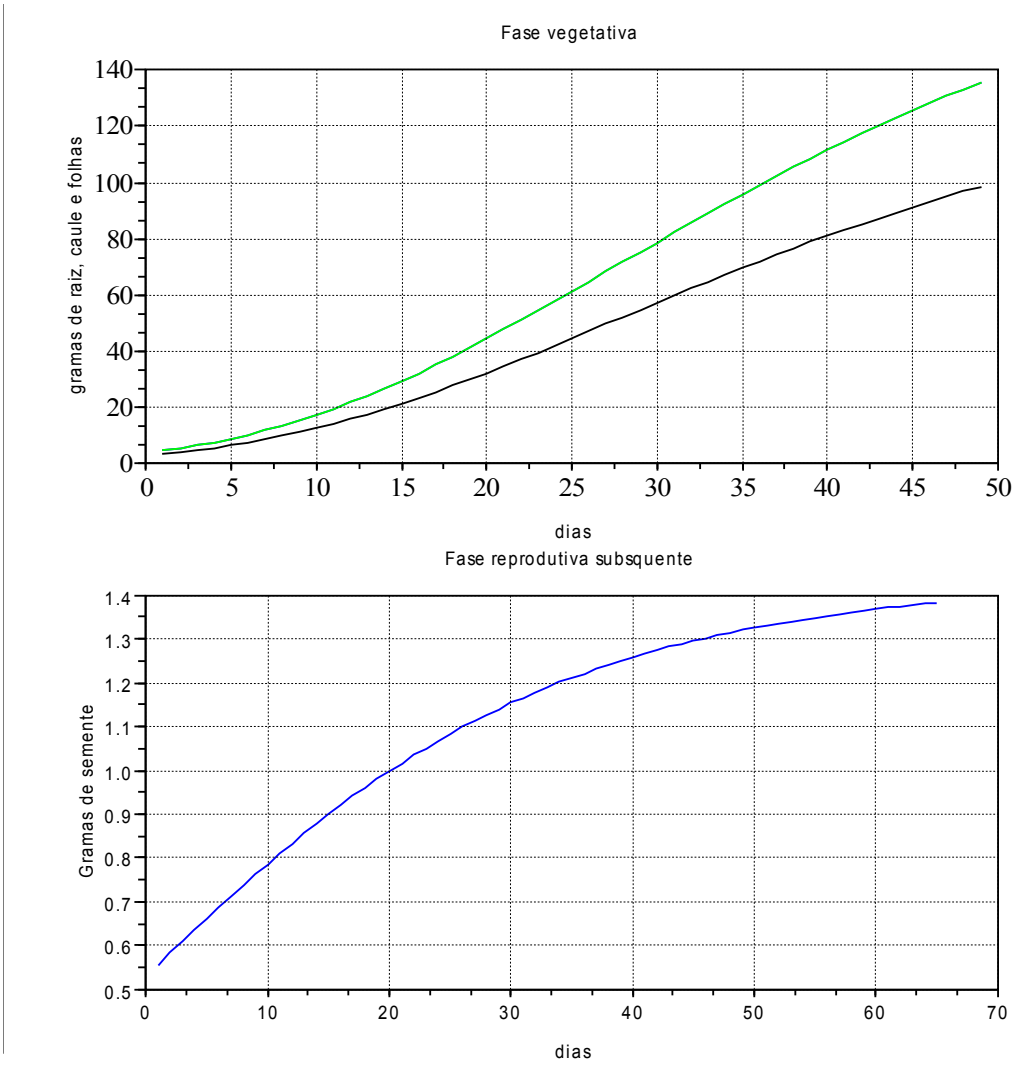


Figura 8.7. Saída do programa plantafu. Os pesos do caule e raiz são iguais, e o das folhas é menor.

8.9 Programa funcroc

Os crocodilos, no sentido genérico, incluindo aligátores e gaviais, são das espécies mais persistentes que habitam o nosso planeta. Estima-se que existam há cem milhões de anos, tendo por isso sobrevivido às causas que extinguiram os dinossauros, 65 milhões de anos atrás.

Ao contrário do que acontece na generalidade das aves e mamíferos, incluindo a nossa espécie (o célebre cromossoma Y), o sexo dos indivíduos não é determinado geneticamente, mas sim pela temperatura do local de postura e incubação dos ovos, de acordo com o esquema representado na figura 8.8. Este mecanismo não é exclusivo dos crocodilos. Por exemplo, o mesmo se passa nalgumas espécies de tartarugas.

O modelo funcroc simula uma população de crocodilos em que a procriação observa o esquema implícito na figura. As fêmeas primeiro ocupam os mais abundantes lugares de postura da zona I (só nascem fêmeas), se sobrarem fêmeas sem ninho, digamos assim, transitam para a zona II (nascimentos dos dois sexos em proporções iguais), e se ainda existem fêmeas sem local de postura transitam para a zona III, onde só nascem machos nos poucos locais de postura disponíveis.

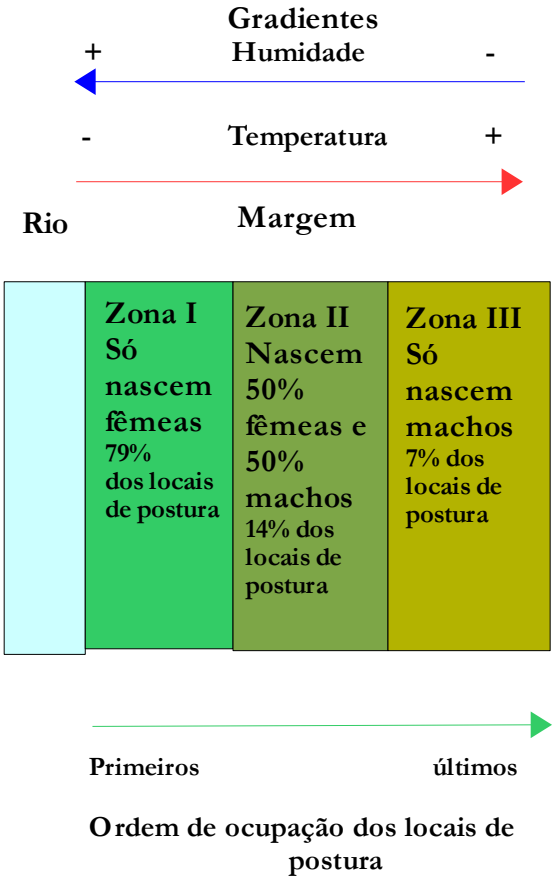


Figura 8.8. Planta do rio e das margens, com os gradientes de humidade e temperatura, e as zonas dos locais de postura.

A conclusão que se tira desta descrição é que quando uma população de crocodilos é pequena, só nascem fêmeas, o que permitem evitar a sua extinção e crescer rapidamente. Isto tem como consequência que a razão dos sexos não é 1:1, nem próximo disto, mas desequilibrada em benefício das fêmeas. Nas populações de crocodilos, razões de 10:1 são aceites como ocorrendo. Isto faz com que os machos controlem haréns numerosos. Por outro lado, os indivíduos adultos tem praticamente uma mortalidade natural igual a zero, pois estão bem adaptados ao meio e não existem inimigos capazes de os matar.

O programa funcroc não simula nenhuma população em particular, mas utiliza parâmetros inspirados em valores reais duma população estudada de aligátores australianos, e referidos em Murray (2002: capítulo 4). Em cada sexo são consideradas quatro classes de idade. Indivíduos de idade até um ano; jovens com idades de 1 a 11 anos; adultos com idades entre os 11 e 60 anos; senis com idades superiores.

O programa é uma função que se inicia com a entrada de um número de adultos fêmeas e adultos machos (f(af, am)). Dada a simplicidade das equações do modelo, deixo como exercício, à leitora, fazer uma descrição verbal delas.

A listagem do programa funcroc é a seguinte:

```
//© L. S. Barreto, 2007. Programa funcroc
//a partir de dados de aligátor australiano
//NM=nascidos machos (idade<1 anos); NF=nascidos fêmeas"
//JM=jovens machos (idade de 1 a 11 anos); JF=jovens fêmeas;"
//AM=adultos machos (60>idade=>11 anos); AF=adultos fêmeas;
//SM=senis machos (idade>60);SF=senis fêmeas;
//p=nº de ovos na postura, 15;
//s0=fracção de ovos que dá origem a crocodilos0(.3);
//s1=fracção de nascidos que chega a jovens (0.12);
//s2=sobrevivência dos jovens (0.8);
//s4==1; sobrevivência dos adultos (0.97)
//s5 sobrevivência dos senis (0.15;
//nº de locais de postura de só filhos fêmeas=k1;
//nº de locais de postura de metade de cada sexo=k2;
//nº de locais de postura de só filhos machos=k3;
xbasc(0)
global nm jm am sm nf jf af sf k1 k2 k3 f1 f2 f3
global nm1 jm1 am1 sm1 nf1 jf1 af1 sf1
disp("f(adultos fêmeas(af), adultos machos (am)")
function []=f(af,am)
if am==0 then abort;
end
if af==0 then abort;
end
k1=20;k2=30;k3=15;p=15;
x = 1:1:100;z=1;w=[];
for i = 1:1:100
  if af>k1 then f1=k1;
  else f1=af;
  end
  if af-f1>k2 then f2=k2;
  else f2=af-f1;
  end
  if af-f1-f2>k3 then f3=k3;
```

```
  else f3=af-f1-f2;
  end
  nf1=p*0.3*(f1+0.5*f2);nm1=p*0.3*(0.5*f2+f3);
  jm1=0.12*nm+0.8*(jm-jm/10);jf1=0.12*nf+0.8*(jf-jf/10);
  am1=0.08*jm+0.97*am-0.97*am/49;af1=0.08*jf+0.97*af-0.97*af/49;
  sm1=0.97*am/49+0.15*sm;sf1=0.97*af/49+0.15*sf;
  w(z,2)=jf1;w(z,3)=jm1;w(z,1)=nm1+nf1;w(z,4)=af1;w(z,5)=am1;
  jm=jm1;nm=nm1;am=am1;sm=sm1;jf=jf1;nf=nf1;af=af1;sf=sf1;
  z=z+1;
end
[k]=w(:,1)';[l]=w(:,2)';[m]=w(:,3)';[n]=w(:,4)';[j]=w(:,5)';
Q=[k;l;m;n;j]';
xset("font",2,3)
plot2d(x,[Q])
xtitle("Dinâmica de uma população de crocodilos ","Anos","Número")
xset("font",2,3)
hl=legend(['Nascidos';'Jovens fêmeas';'Jovens machos';'Adultos
fêmeas';'Adultos machos'],a=1);
disp("Nascidos, Jovens, Adultos, Senis")
fem=string(int([nf,jf,af,sf]));
mac=string(int([nm,jm,am,sm]));
disp([fem],"Fêmeas")
disp([mac],"Machos")
endfunction
```

Um exemplo da saída do programa é:

```
 f(adultos fêmeas(af), adultos machos (am))


 exec done

-->f(2,1)
```

```
 Nascidos, Jovens, Adultos, Senis

 Fêmeas

!387  165  264  6  !

 Machos

!63  26  42  0  !
```

Esta saída numérica é acompanhada pelo gráfico da figura 8.9.

O capítulo mencionado de Murray (2002) pode ser indicado como documentação complementar do programa, para o leitor mais interessado.

8.10 Programa ProbExt

Quando o excesso de caça, para venda das peles, já tem posto em perigo a persistência de algumas populações de crocodilos, não é descabido introduzir agora um modelo atinente à análise da viabilidade de populações.

O programa é curto mas potente. O mais difícil nestas situações é obter experimentalmente (dados de campo) a variância da estocacidade ambiental e da demográfica, mas este assunto ultrapassa o âmbito deste livro. Uma boa referência neste tópico (análise da viabilidade das populações) é Morris e Doak (2002).

As probabilidades acumuladas de extinção são um instrumento muito valioso na análise da viabilidade de populações.

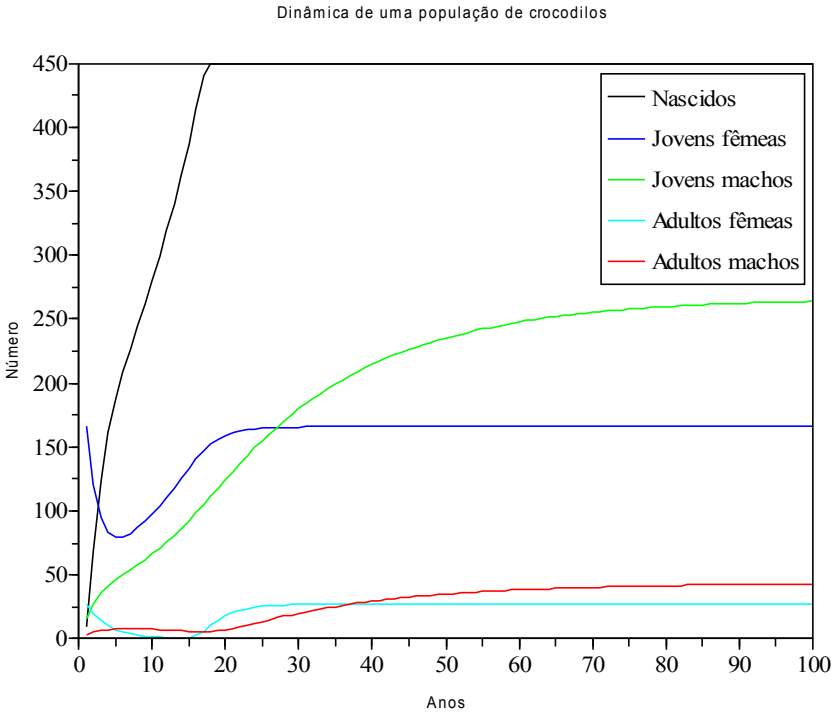


Figura 8.9. Saída do programa funcroc, para uma população inicial de duas fêmeas adultas e um macho adulto. Os indivíduos da primeira classe de idade (0-1 ano) apresentam-se agregados num só valor, e as classes dos indivíduos senis não são representadas

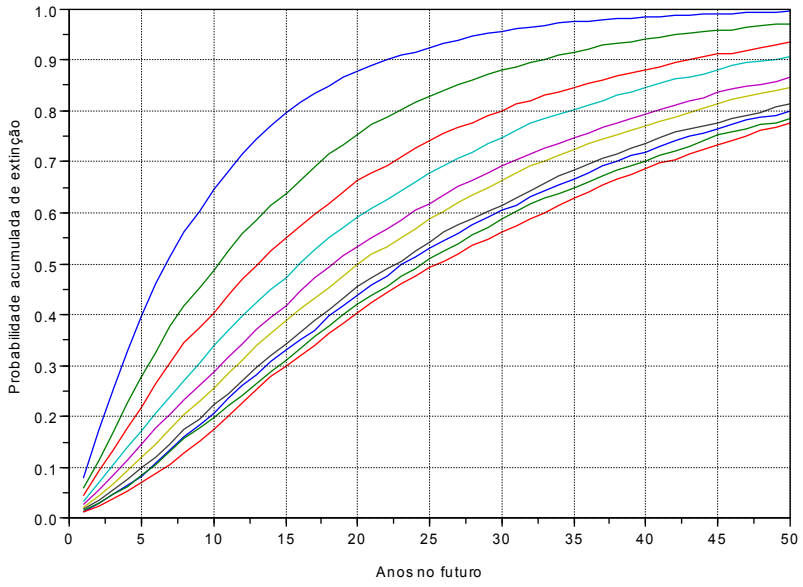


Figura 8.10. Probabilidades acumuladas de extinção, pelo período de cinquenta anos, de uma população inicial de uma fêmea de veado-mula, para valores de k de 2, 4,...20, sendo variância da estocacidade ambiental =0,3 e da demográfica=0,45. Para cada valor de k foram realizadas 5000 simulações

O programa ProbExt assume a situação hipotética de uma população de veado-mula existindo numa paisagem onde já só ocorrem pequenas manchas de habitat que lhe é favorável, capazes de sustentar populações máximas (capacidade de sustentação, k) não superiores a 20 indivíduos. Consideramos a estocacidade ambiental e a demográfica. Pretende-se obter informação sobre a probabilidade acumulada de extinção, nos próximos 50 anos, para locais com as várias capacidades de sustentação.

O programa ProbExt só tem saída gráfica, exibida na figura 8.10. Vou deixar a análise da solução encontrada para o problema, implementada pelo programa, como exercício para a leitora interessada.

A listagem do programa é a seguinte:

```
////(© L. S. Barreto, 2008 Programa ProbExt
//CDF de extinção do veado-mula
//usando estocacidades demográfica e ambiental
//capacidade de sustentação, k=2,4,6,...20
results=[];
xbasc()
for k=2:2:20
PrExt=zeros(50,1);
//variâncias das estocacidades
va=0.3;vd=0.45;
for j=1:5000
n=1;
for t=1:1:50
//variância e desvio padrão globais
v=sqrt(va+vd/n);
//variável aleatória
z=(grand(1,"nor",0,v));
//modelo discreto estocástico de Gompertz
n=n*(exp(0.2239*log(k)-0.2239*log(n))+z);
if n<=0
  PrExt(t)=PrExt(t)+1;
  break;
  end
end
end
 PrExt=cumsum(PrExt/5000);
```

```
 results=[results PrExt];
end
plot([results])
xtitle("","Anos no futuro","Probabilidade acumulada de extinção");
xgrid();
```

8.11 Programa plamarg

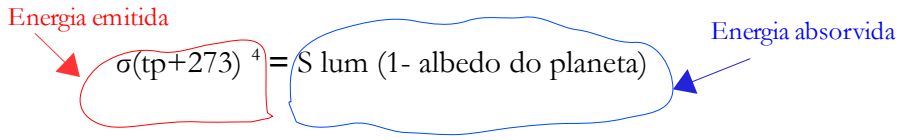
O meu primeiro contacto com balanços de energia ocorreu no Outono de 1967, quando preparava o mestrado em ecologia florestal, nos E.U.A., e um dos livros de uma disciplina era o clássico de W. D. Sellers (1965): *Physical Climatology*. Numa escala diferente, voltei aos balanços de energia quando estimei o da região da então cidade de Lourenço Marques, hoje Maputo, em Barreto (1974), O facto de os programas desta secção e da seguinte assentarem em balanços de energia, suscitou esta digressão memorialista.

As pessoas que se interessam pelo tema da mudança global e afins (e.g., efeito de estufa), estão familiarizados com o conceito de Gaia, de James Lovelock, e o seu planeta das margaridas ("Daisyworld").

Informação sobre este modelo, incluindo a sua génese, e a sua exploração no sentido de que o nosso planeta é um sistema em que as suas macro componentes (atmosfera, biosfera, hidrosfera e litosfera) actuam como um todo, e a presença dos seres vivos tende a tornar o meio ambiente mais favorável à vida, pode ser encontrada em Lovelock (1989: capítulo 3; 2007: 44-45). Uma análise recente da evolução deste tipo de modelos, com contribuições de outros autores, está disponível em Wilkinson (2006: secção 6.2).

O programa plamarg reproduz o modelo do planeta das margaridas, no caso da existência de uma variedade clara (maior albedo, menor absorção de calor) e outra escura (menor albedo, maior aquecimento desta vegetação). Os dois tipos de vegetação têm uma taxa de crescimento (resc, rcla) que é uma função parabólica das suas temperaturas, entre 5 e 45 º C. O programa não simula a evolução do planeta condicionada pela variação da luminosidade da estrela que o aquece, mas sim pontualmente para valores da luminosidade que o leitor selecciona. Ensaiando vários valores verificará que quanto a luminosidade cresce, aumenta a fracção do planeta coberta pelas margaridas claras (aquecem menos) e diminui a das margaridas escuras (aquecem mais).

A listagem do programa é auto explanatória. A equação do balanço de energia do planeta, que permite estimar a temperatura média do planeta, tp, é a seguinte:



σ é a constante de Stefan-Boltzman, S é fluxo constante de energia tal que $S/\sigma=1{,}68\times10^{10}$ K^4 e lum é uma constante adimensional que mede a luminosidade da estrela do planeta das margaridas. Com mais precisão tp+273 é a temperatura efectiva de radiação.

A listagem do programa é a seguinte:

```
//(© L. S. Barreto, 2008 Programa plamarg
//O célebre universo de James Lovelock
global mortcla mortesc albesc albcla
global albsolo m n frc fre
//taxas de mortalidade da vegetação
mortcla=0.3;mortesc=0.3;
//albedos da vegetação e do solo
albesc=0.25;albcla=0.65;albsolo=0.4;
m=0.00003265;
n=20;
//alpla=albedo médio do planeta
//tp=temperatura média do planeta
//lum=luminosidade
//valores iniciais de coberto
frc=0.1;fre=0.1;
//criar vectores para os dados
te=[];tc=[];tpl=[];
are=[];arc=[];
t=[1:0.1:2500];
disp("[]=f(Luminosidade do Sol. Actual=1)")
function []=f(lum)
xbasc(0)
for a=1:0.1:2500
  cob=frc+fre;//coberto total
  //albedo do planeta como uma média ponderada
  alpla=(1-cob)*albsolo+albesc*fre+albcla*frc;
  tp=((1.68*10^10)*lum*(1-alpla))^0.25-273;//da equação do balanço
energético
  //temperatura das margaridas escuras
  tesc=n*(alpla-albesc)+tp;
  //temperatura das margaridas claras
  tcla=n*(alpla-albcla)+tp;
  //taxa de crescimento das margaridas escuras
```

```
  resc=1-m*(22.5-tesc)^2;
  //taxa de crescimento das margaridas claras
  rcla=1-m*(22.5-tcla)^2;
  //variação da área da margaridas escuras
  fredot=fre*((1-cob)*resc-mortesc);
  //variação da área da margaridas claras
  frcdot=frc*((1-cob)*rcla-mortcla);
  //valores actualizados dos cobertos
  fre=fre+0.1*fredot;
  frc=frc+0.1*frcdot;
  are=[are fre];
  arc=[arc frc];
  te=[te tesc];
  tc=[tc tcla];
  tpl=[tpl tp];
end
M=[are;arc]';
subplot(211);
plot2d(t,[M]);
xtitle("Fracções do coberto das margaridas","Tempo","Fraccção da área
ocupada");
xgrid();
N=[te;tc;tpl]';
subplot(212)
plot2d(t,[N]);
xtitle("Temperaturas das margaridas e do planeta","Tempo","Temperatura,
graus C");
xgrid();
a='Área das margaridas escuras=';b='Área das margaridas
claras=';c='Temperatura do
planeta=';aa=string(fre);bb=string(frc);cc=string(tp);
disp([a,aa;b,bb;c,cc])
endfunction
```

Entrando com o valor 1 para a luminosidade, o programa cria o gráfico da figura 8.11 e a saída que se lhe segue.

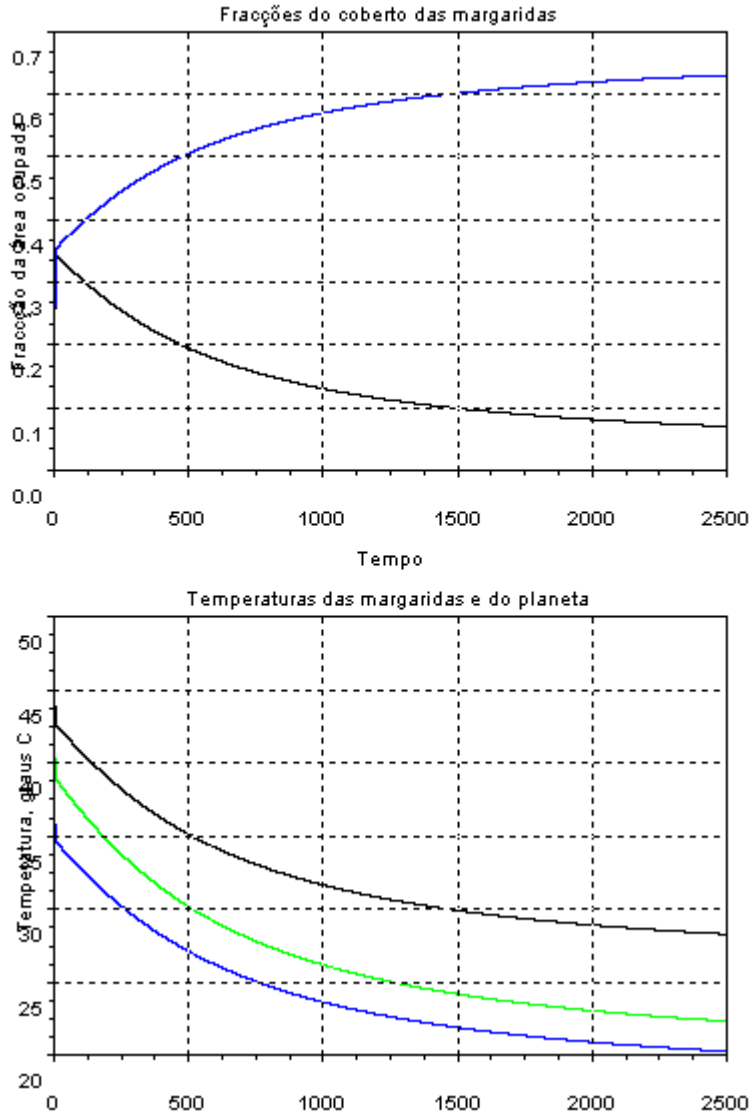


Figura 8.11. Gráfico gerado pelo programa plamarg, quando a luminosidade é igual a 1 (f(1)). A cor preta corresponde às margaridas escuras.. A temperatura do planeta é dada pela linha verde

```
-->f(1)

!Área das margaridas escuras=  0.0699783  !
```

```
!                                  !
!Área das margaridas claras=   0.6299462  !
!                                  !
!Temperatura do planeta=       22.361908  !
```

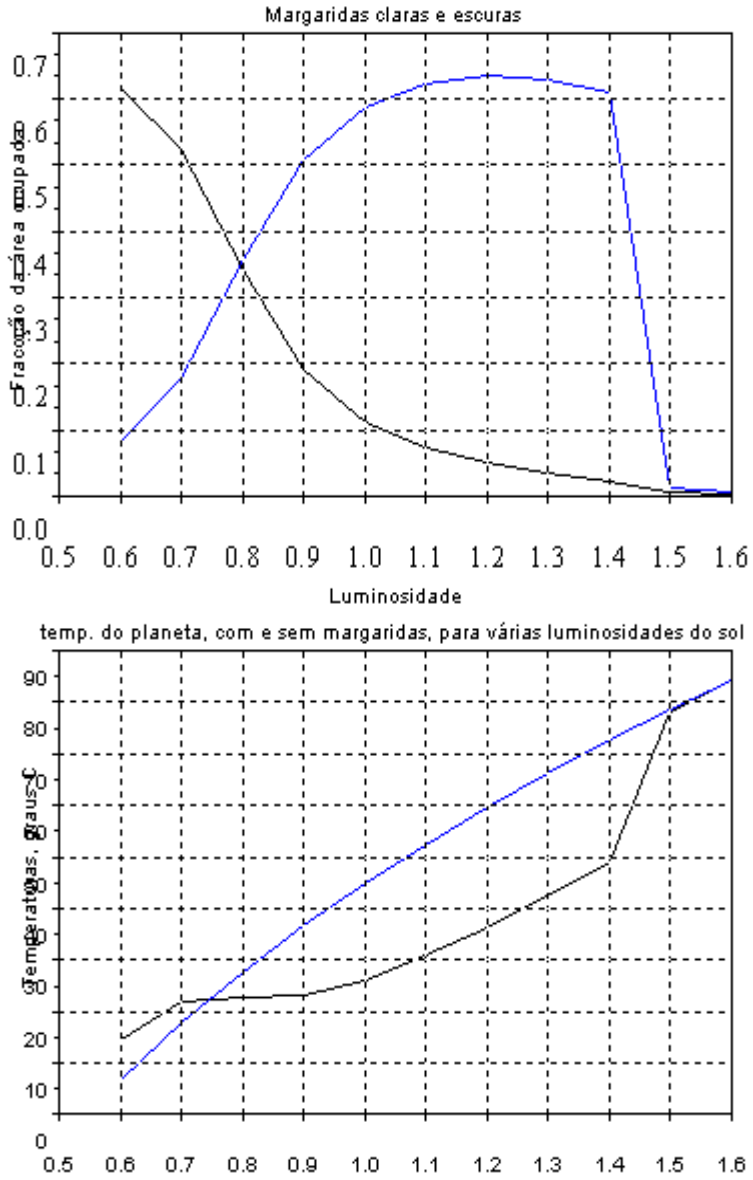


Figura 8.12. Dinâmica do planeta das margaridas associada á variação da luminosidade da sua estrela.. Gráfico superior: variação das fracções das duas áreas (cor preta associada à margarida escura). No gráfico inferior, a estrela do planeta é designada abreviadamente por sol, e a linha a azul corresponde ao planeta sem vegetação

A tese de Lovelock é de facto sustentada pela simulação do planeta das margaridas, como se ilustra na figura 8.12. No segundo gráfico da figura, compara-se a temperatura média do planeta com e sem margaridas. No intervalo de temperatura em que crescem, a tp é claramente inferior. Sugiro à leitora que modifique o program plamarg de modo a produzir os gráficos da figura 8.12.

Como documentação para este modelo sugiro Hannon e Ruth (1997: capítulo 42) e Roughgarden (1998: secção 7.2.3)

8.12 Programa fuMBE2

Vamos agora abordar a última aplicação. A actual preocupação, com o efeito estufa e a mudança global das condições físicas do meio, justificam esta simplíssima incursão pelo assunto.

Nesta área temática há duas referências que não podem ser ignoradas. Uma básica sobre a "fisiologia" da Terra deve-se a Kump, Kasting e Crane (2004). A outra sobre modelação climática é a de McGuffie e Henderson-Sellers (2005) que trás um CD muito útil com ligação directa aos sítios com modelos utilizáveis de simulação climática, entre outra informação, incluindo fontes bibliográficas..

O modelo que programei, como já referi na secção anterior, é do balanço de energia, e deve-se a M. I. Budyko e W. D. Sellers, que o estabeleceram em 1969, independentemente um do outro. Cheguei a ele através da mediação do livro de McGuffie e Henderson-Sellers, que dedicam um capítulo, o terceiro, aos vários tipos de modelos de balanço da energia, enfatizando a utilidade que revelam, apesar da sua simplicidade relativa. Kump, Kasting e Crane (2004) também dedicam o capítulo 6 aos modelos climáticos e apresentam uma clara resenha histórica do assunto.

No modelo fuMBE2, a superfície do planeta é dividida em faixas de 10 graus de latitude, cujas equações do balanço de energia são do tipo:

Energia da radiação absorvida = Energia transferida para as faixas vizinhas + radiação infra-vermelha emitida

As equações atinentes ao model estão descritas em McGuffie e Henderson-Sellers (2005: 85-96) e não são aqui reproduzidas.

Relativamente ao modelo de Budyko e Sellers, apresentado no livro de McGuffie e Henderson-Sellers, o programa fuMBE2 usa uma regressão calibrada para ter em conta a nebulosidade, no cálculo do albedo das faixas a partir das suas temperaturas, e também estabelece a temperatura média do planeta de modo diferente.

À semelhança do programa anterior, o utilizador deve entrar com f (luminosidade do Sol), sendo a actual igual a 1. Os outros parâmetros, como os ligados à transferência de energia, albedo do gelo, por exemplo, podem ser igualmente alterados pelo leitor.

. O programa tem a seguinte listagem:

```
//(© L. S. Barreto,  2008  Programa fuMBE2
//Modelumo de balanço da energia de Budyko e Sellers
//Parâmetros da 'table 3.1' de McGuffie e Henderson-Sellers (2005)
//e da listagem
//modificado no cálculo do albedo usa regressão
//e no cálculo da temp média tx
```

```
disp("ENTRAR f(luminosidade)")
function [res]=f(lum)
algelo=0.62;a=204;b=2.17;c=3.81;tcrit=-10;
lat=["80-90" "70-80" "60-70" "50-60" "40-50" "30-40" "20-30" "10-20" "0-10"
"0-10"  "10-20" "20-30" "30-40" "40-50" "50-60" "60-70" "70-80" "80-90"]';
calb=0.5;in=%pi/36;p2=%pi/2;
solat=[0.5 0.531 0.624 0.77 0.892 1.021 1.12 1.189 1.219 1.219 1.189 1.12
1.021 0.892 0.77 0.624  0.531  0.5];
albi=[ 0.5 0.3 0.1 0.08 0.08 0.2 0.2 0.05 0.05 0.08 0.05 0.1 0.08 0.04 0.04 0.6
0.7 0.7] ;
q=342.5;
//nuvem
nuvemi=[0.52 0.58 0.62 0.63 0.57 0.46 0.40 0.42 0.50 0.50 0.42 0.40 0.46 0.57
0.63 0.62 0.58 0.52];
tinic=[-16.9 -12.3 -5.1 2.2 8.8 16.2 22.9 26.1 26.4  26.4  26.1 22.9 16.2 8.8 2.2
-5.1 -12.3 -16.9];
templat=tinic;
//constante solar
consol=lum*1370/4;
 latice=0;
 nl=0;
//calcular albedo das zonas
alblat=0.4049261*ones(1:18)-0.0062709*templat;
for h=1:250
 for i=1:18
  if templat(i)==26.4 then nl=0;
  end
 end
  for i=11:17
  if templat(i)<tcrit then alblat(i)=algelo;
  else
  dp(i)=(templat(i+1)-tcrit)*in/(templat(i+1)-templat(i));
  g=dp(i);
  a2=p2-(i+0.5)*in;latice=a2+dp(i);
  select g
```

```
 case g>0.0872564 then
 a3=p2-i*in;
 a4=p2-(i-1)*in;
 a5=(sin(latice)-sin(a3))/(sin(a4)-sin(a3));
 nc=algelo-(algelo-alblat(i))*a5;
 nl=i;
 case g<=0.0872564 then
 a3=p2-(i+1)*in;
 a4=a3-in;
 a5=(sin(a4)-sin(latice))/(sin(a4)-sin(a3));
 nc=alblat(i+1)*(1-a5)+algelo*a5;
 nl=i;
 end
end
end
//
if templat(18)<tcrit then alblat(18)=algelo;
end
//
if alblat(1)==albi(1) then ni=90/57.296;
end
sm=0;
d=708.66667;u=d;area=[];l=0;
 for i=1:9
 are=%pi*integrate('(6378^2-x^2)^1/2*(1+(-x/(6378^2-x^2))^2)^0.5','x',l,u);
 area=[area are];
 l=u;
 u=u+d;
 end
 g=area/(2*sum(area));
 peso=[abs(sort(-g)) g];
 tx=sum(peso.*templat);
for i=1:18
 ol(i)=(1-nuvemi(i))*(a+b*templat(i));
```

```
 ol(i)=ol(i)+nuvemi(i)*(a+b*(templat(i)-5));
 asol(i)=consol*solat(i)*(1-alblat(i));
 tm(i)=templat(i);
 templat(i)=consol*solat(i)*(1-alblat(i))-a+c*tx;
 templat(i)=templat(i)/(c+b);
end
am=0;ic=0;
for i=1:18
 ma=abs(templat(i)-tm(i));
 if templat(i)>800 then break;
 end
 if ma>am then am=ma;
 end
end
if am<0.01 then ic=1;
end
if ic==1 then break;
end
med1=(templat(1)+templat(18))/2;
templat(1)=med1;templat(18)=med1;
med2=(templat(2)+templat(17))/2;
templat(2)=med2;templat(17)=med2;
end
for i=1:18
 if templat(i)<=tcrit then alblat(i)=0.62;
 end
end
xbasc(0)
t1=[85 75 65 55 45 35 25 15 5];t2=[5:10:85];
tpn=templat(:,1:9)';tps=ol(1:9,1);
aln=alblat(:,1:9)';als=asol(1:9,1);
subplot(2,2,1)
plot2d(t1,[tpn], [-3])
xtitle("Norte, Temperatura","Latitude","Temperatura, graus C");
```

```
subplot(2,2,2)
plot2d(t1,[tps],[-2])
xtitle("Norte, Radiação para o espaço","Latitude","W por m. q.");
subplot(2,2,3)
plot2d(t1,[aln],[-1])
xtitle("Norte, Albedo","Latitude","Albedo");
subplot(2,2,4)
plot2d(t1,[als],[-4])
xtitle("Norte, Radiação absorvida","Latitude","W por m.q.");
format('v',5);
latid=[85 75 65 55 45 35 25 15 5 5 15 25 35 45 55 65 75 85]';
disp([ 'Latitude N' 'Temperatura' 'Albedo' 'Rad. saída' 'Rad absov.'])
res=[t1' tpn aln tps als]
endfunction
```

A informação dos dois hemisférios é igual, por isso a saída só se reporta ao hemisfério Norte. Para as nove faixas latitudinais inscreve a temperatura, albedo, energia absorvida e a irradiada (W m^{-2}) e apresenta os respectivos gráficos.

Para a luminosidade igual a 1, a saída é:

```
-->f(1)

!Latitude N  Temperatura  Albedo  Rad. saída  Rad absov.  !
 ans  =

   85.  - 11.9    0.62    173.    83.8
   75.  - 10.7    0.62    175.    94.2
   65.  - 4.22    0.44    188.    120.
   55.    2.5     0.39    203.    161.
   45.    8.87    0.35    217.    199.
   35.    16.4    0.30    235.    244.
   25.    23.     0.26    250.    283.
   15.    27.3    0.24    259.    309.
   5.     28.8    0.24    261.    318.
```

sendo o gráfico correspondente exibido na figura 8.13.

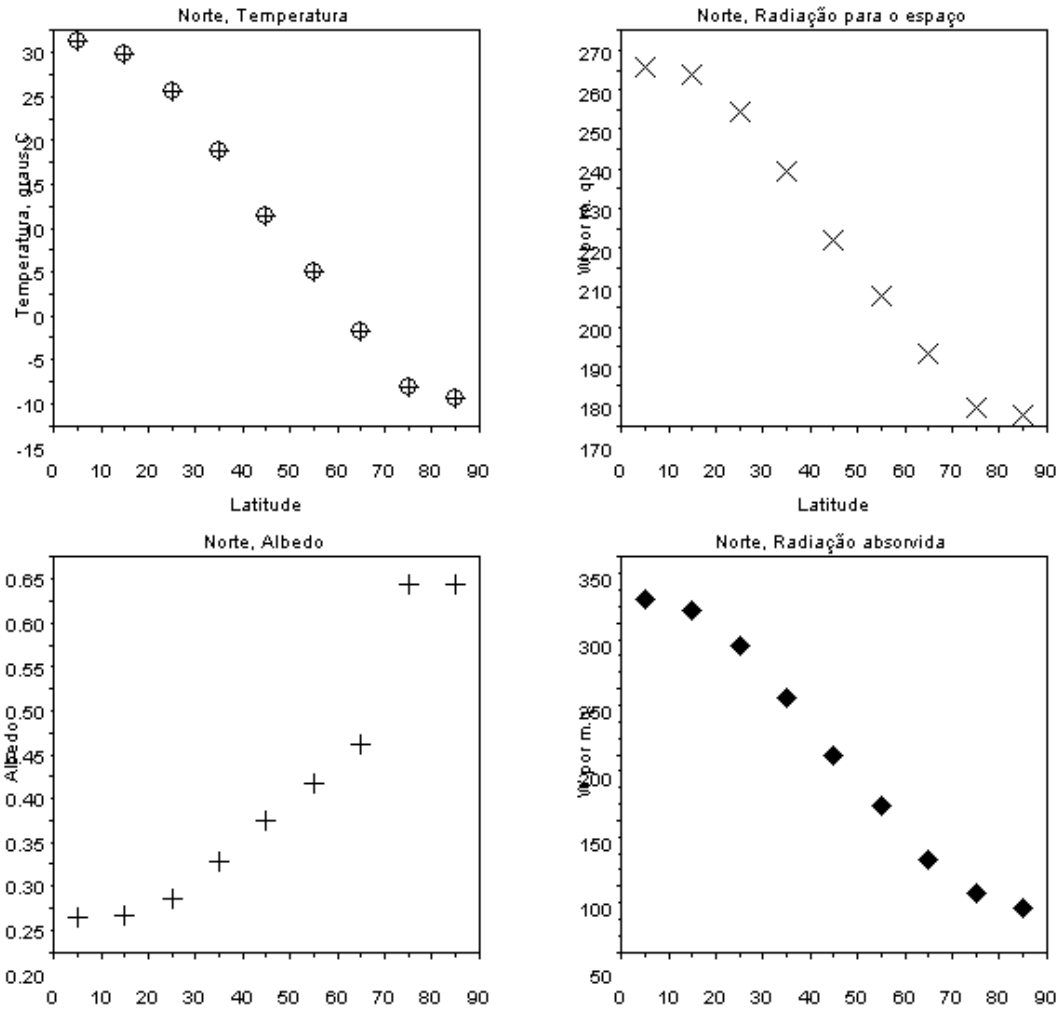


Figura 8.13. Gráficos criados pelo programa fuMBE2, para a luminosidade igual a 1 (f(1)).

Duas sugestões: a) no programa, fazer a tabela da saída, com o seu cabeçalho mais apresentável; b) se consultou o livro McGuffie e Henderson-Sellers, acrescente os comentários que acha que faltam no programa.

Admite-se que a menor luminosidade do Sol foi 70% da actual. Veja o que acontece com este valor.

Chegados aqui, espero que este texto o tenha, **de facto**, iniciado na utilização do Scilab. Foi com muito gosto que o escrevi para si, e agora aqui o disponibilizo.

Bibliografia

Allaire, G. S. e M. Kaber, 2002. *Introduction à Scilab. Exercices pratiques corrigés d'algèbre linéaire.* Ellipses Éditions Marketing S. A., Paris.

Anjo, A. J. B., R. Fernades e A. S. Carvalho, 2003. *Curso de MatLab.* Principia. Publicações Universitárias e Científicas, S. João do Estoril.

Antia, H. M., 2000. *Numerical Methods for Scientists and Engineers.* 2nd edition. Birkhäuser Verlag, Basel.

Barreto, L. S., 1974. Estimativas provisórias de balanços de energia em Moçambique. 1. Lourenço Marques. *Revista de Ciências Agronómicas* (Lourenço Marques), 7:63-70.

Barreto, L. S., 2004a. *Pinhais Bravos. Ecologia e Gestão.* "e-book". Instituto Superior de Agronomia, Lisboa.

Barreto, L. S., 2004b. *A Unified Theory for Self-Thinned Mixed Stands. A Synoptic Presentation.* Research Paper SB-02/04. Departamento de Engenharia Florestal, Instituto Superior de Agronomia.

Barreto, L. S., 2005a. *Gause´s Competition Experiments with Paramecium sps. Revisited.* Research Paper SB-01/05.. Departamento de Engenharia Florestal, Instituto Superior de Agronomia. Verão revista submetida à Silva Lusitana em Setembro de 2007.

Barreto, L. S., 2005b. *Theoretical Ecology. A Unified Approach.* "e-book". Edição do autor, Costa de Caparica.

Ben-Israel, A. E R. Gilbert, 2002. *Computer-Supported Calculus.* Springer Verlag, Wien.

Brock, T. D., 1981. Calculating solar radiation for ecological studies. *Ecological* Modelling, 14:1-19.

Caswell, H., 2001. *Matrix Population Models.* Second edition. Sinauer, Sunderland.

Charles-Edwards, D. A., D. Doley, e G. M. Rimmington, 1986. *Modelling Plant Growth and Development.* Academic Press, Sidney.

Davis, J. H., 2000.*Differential Equations with Maple. An Interactive Approach.* Birkhäuser, Boston.

Davis, P. J. e E. R. Hersh, 1995. *A Experiência Matemática.* Gradiva, -Publicações, Lda., Lisboa.

Davis, P. J. e E. R. Hersh, 1997. *O Sonho de Descartes. O Mundo Segundo A Matemática.* Difusão Cultural, Lisboa.

Edelstein-Keshet, L., 2005. *Mathematical Models in Biology.* Society for Industrial and Applied Mathematics, Philadelphia.

Gomez, C., Editor, 1999. *Engineerind and Scientific Computing with Scilab.* Birkhäuser, Boston.

Guerre-Delabrière, S. e M. Pastel, 2004. *Méthodes d'approximation. Équations différentielles. Applications Scilab.* Ellipses Éditions Marketing S. A., Paris.

Lopes, D. e J. Aranha, 2006. Avaliação do Conteúdo de Carbono na Matéria Seca de Diferentes Componentes de Árvores de *Eucalyptus globulus* e de *Pinus pinaster. Silva Lusitana* 14(2):149-154.

Hannon, B. e M. Ruth, 1997. *Modeling Dynamic Biological Systems.* Springer, New York.

Hederson-Sellers, A. e K. McGuffie, 1987. *A Climate Modelling Primer.* Wiley, Chichester.

Jørgensen, S. E., 1986. *Fundamentals of Ecological Modelling.* Elsevier, Amsterdam.

Kump, L. R., J. F. Kasting e R. G. Crane, 2004. *The Earth System.* Second Edition. Pearson Prentice Hall, Upper Saddle River, N. J., E. U. aA.

Lovelock, J., 1989. *As Eras de Gaia. Uma Biografia da nossa Terra Viva.* Publicações Europa-América, Mem Martins.

Lovelock, J., 2007. *A Vingança de Gaia.* Gradiva, Lisboa.
Lynch, S., 2000. *Dynamical Systems with Applications Using Maple.* Birkhäuser, Boston.
McGuffie, K. e A- Henderson-Sellers, 2005. *A Climate Modelling Primer.* Third Edition. Wiley, Chichester.
Morris, W. F. e D. F. Doak, 2002. *Quantitative Conservation Biology. Theory and Practice of Population Viability Analysis.* Sinauer Associates, Inc. Publishers, Massachusetts, U.S.A.
Murray, J. D., 2002. *Mathematical Biology. I. An Introduction.* Third edition. Springer, New York.
Pratap, R., 1999. *Getting Started with MATLAB 5. A Quick Introduction for Scientists and Engineers.* Oxford University Press, Oxford.
Myers, R. H., 1990. *Classical and Modern Regression with Applications.*Second Edition. Duxbury, Pacific Grove, California.
Roughgarden, J., 1998. *A Primer of Ecological Theory.* Prentice Hall, Upper Saddle River, New Jersey.
Said, R., 2007. *Curso de Lógica de Programação.* Digerati Books, Universo dos Livros Editora Lda, São Paulo, Brasil.
Scilab Group *Introduction to SCILAB. User´s Guide*. INRIA Meta2 Project/ENPC Cergrene Domaine de Voluceau , Rocquencourt, France. Acompanha o Scilab em formato PDF:
Scilab Group. *Scilab Reference Manual. On-line Documentation*. INRIA Meta2 Project/ENPC Cergrene Domaine de Voluceau , Rocquencourt, France. Acompanha o Scilab em formato PDF:
Sellers, W. D., 1965. *Physical Climatology.* Chicago University Press, Chicago.
Stanoyevitch, A., 2005. *Introduction to MATLAB with Numerical Preliminaries.* John Wiley & Sons, Inc., Hoboken, New Jersey.
Wilkinson, D. M., 2006. *Fundamental Processes in Ecology an Earth Systems Approach.* Oxford University Press, Oxford, U. K.
Yeargers, E. K., R. W. Shonkwiler e J. V. Herod, 1996. *An Introduction to the Mathematics of Biology with Computer Algebra Models.* Birkhäuser, Boston..