MINISTÉRIO DA CIÊNCIA E TECNOLOGIA

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-9381-PRE/5041

Introdução a Programas Científicos de Distribuição Gratuita: GNU/Octave, GNU/Maxima, LaTeX e GNU/RCS

Dra. Margarete Oliveira Domingues
Dr. Odim Mendes Junior

Trabalho apresentado no Encontro Regional de Matemática Aplicada e Computacional-(Mini-curso), realizado em Natal -29-31, agosto/2002.

INPE
São José dos Campos
2002

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Introdução a Programas Científicos de Distribuição Gratuita:

GNU/Octave, GNU/Maxima, LaTeX e GNU/RCS

Encontro Regional de Matemática Aplicada e Computacional

(Mini-curso)

Elaboração:

Margarete Oliveira Domingues
CPTEC/INPE

Odim Mendes Júnior
CEA/INPE

Apoio:

SBMAC
INPE

São José dos Campos, 12 de Setembro de 2002

AGRADECIMENTOS

Os autores agradecem:

- a todos os que contribuem pelo desenvolvimento de ferramentas de livre distribuição.

- a Dra. Cláudia Dezotti, Dr. Regivan Santiago e Profa. Márcia Cruz, da Universidade do Rio Grande do Norte, o incentivo para a concretização deste material.

- ao Dr. Rubens Sampaio, presidente da Sociedade Brasileira de Matemática Aplicada e Computacional, o apoio a este tipo de empreendimento.

- ao INPE pelas condições de inovar e, como resultado do desenvolvimento das próprias pesquisas, poder contribuir com a sociedade.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

CAPÍTULO 1

INTRODUÇÃO

Atualmente existem excelentes programas de livre distribuição e/ou de distribuição gratuita disponíveis para a realização das mais diversas atividades de pesquisa e ensino. Esses programas são desenvolvidos por milhares de pessoas no mundo inteiro e tornam–se cada vez mais de interesse da comunidade científica e do público em geral. A divulgação de algumas dessas ferramentas é o objetivo desse Mini–curso, atento ao lema: investir em conhecimento, aumentar o desempenho e reduzir custos.

Muitas das atividades científicas e de ensino nos cursos de Cálculo e Física são desenvolvidas utilizando programas proprietários. Muitos desses programas possuem similares gratuitos que podem auxiliar a reduzir os custos de tais ferramentas computacionais, ao mesmo tempo que dá acesso irrestrito a ferramentas robustas. Dois desses programas são o GNU/Octave, para cálculos númericos, e o GNU/Maxima, para a computação simbólica. Além dessas ferramentas, serão abordados sucintamente o pacote de editoração digital LaTeX, para a redação de textos científicos e a preparação de diapositivos (slides) de alta qualidade, e o programa RCS, que é um sistema de controle de versão dos textos.

A utilização das poderosas ferramentas computacionais de livre distribuição e de distribuição gratuita ainda é muito modesta em vários segmentos da comunidade científica brasileira (ver Apêndice A). Este Mini–curso incentiva a utilização de algumas dessas ferramentas no âmbito da Matemática Aplicada, da Física e de áreas afins, objetivando o ensino e a pesquisa. Optou–se, então, neste mini–curso, por dar–se uma visão geral de um pequeno conjunto de ferramentas de computação numérica, de geração de gráficos 2D, de edição de textos científicos e de gerenciamento de versões de texto e programas. Com este texto, divulga–se também materiais didáticos já disponíveis para aqueles interessados em se aprofundarem nessas ferramentas.

O Mini–curso constitui–se dos seguintes tópicos:

a) Introdução ao cenário de programas de livre distribuição e de código aberto

b) Introdução ao GNU/Octave

- Introdução ao ambiente computacional
- Noções dos principais comandos
- Operações com matrizes
- Operações com funções

- Gráficos de dados e funções
- Elementos de programação

c) Introdução ao GNU/Maxima

- Introdução ao ambiente computacional
- Noções dos principais comandos
- Operações com matrizes
- Operações com funções
- Gráficos de funções

d) Conceitos básicos do LaTeX

- Classes do documento
- Pacotes de auxílio
- Corpo do documento

e) Gerenciamento de código-fonte utilizando o RCS

- Estrutura de Organização
- Operações básicas

Comentários Adicionais

Os programas GNU/Octave (que precisa do gnuplot), RCS e o pacote teTeX (LaTeX) já vem disponíveis nos CDROMs das principais distribuições GNU/Linux. Na Tabela 1.1 estão apresentados os endereços de todos esses programas e de alguns outros programas de interesse neste contexto. Em geral, versões atualizadas são oferecidas com muita freqüência. Dessa forma, é sempre interessante consultar esses endereços eletrônicos.

TABELA 1.1 – Endereços na Web dos programas gratuitos a serem discutidos neste Mini–curso.

Programa	Endereço na Web
GNU/Maxima	http://www.ma.utexas.edu/users/wfs/maxima.html
GNU/Octave	http://www.gnu.org
Prosper	http://prosper.sourceforge.net/
RCS	http://www.gnu.org
gnuplot	http://www.gnu.org
LaTeX	http://www.ctan.org
TeX–br	http://biquinho.furg.br
Kile	http://xm1.net.free.fr/kile
xfig	http://www.xfig.org
gimp	http://www.gimp.org/
psutils	http://www.go.dlr.de:8081/pdinfo_dv/psutils.html
abntex	http://abntex.codigolivre.org.br/
BiBTeX	http://www.ecst.csuchico.edu/ jacobsd/bib/formats/bibtex.html

CAPÍTULO 2

PROGRAMAS DE LIVRE DISTRIBUIÇÃO E DE CÓDIGO ABERTO

O código aberto e o software livre são formas de solidariedade e fomentam o desenvolvimento pelo trabalho, reunindo mentes em um esforço recíproco e simbiótico. Sinalizando uma perspectiva de mudança nas condutas pessoal e profissional, o que essas novidades da informática (sistemas operacionais e programas livres) trazem de benefícios ou de desafios para uma instituição científica? Software comercial ou software livre, qual a melhor ferramenta de trabalho? A resposta pessoal, mas com efeitos coletivos, só pode ser dada caso sejam melhor conhecidas as alternativas existentes. Esta secção pretende contribuir para um melhor entendimento do cenário de ferramentas de trabalho atualmente disponíveis.

2.1 A importância social da quebra de paradigmas

Paradigmas são padrões nas formas de pensar resistentes a mudanças. Isso dá a impressão que eles são imutáveis e até naturais.

No entanto, resultado de contradições intrínsecas e de abordagens inesperadas, esses padrões ruem e dão lugar a formas renovadas de idéias e procedimentos. Neste instante, tem-se as mudanças de atitudes sob efeito avalanche.

Há uma nova *moeda* circulando no espaço de convivência: a qualidade de vida. *O que você pode me oferecer de qualidade de vida? E o que eu posso te oferecer de qualidade de vida ?* Um dos atributos essenciais desta moeda é a consistência interna e externa.

Tão importante quanto a quebra de paradigmas é o momento em torno da quebra. A quebra de paradigmas representa o abandono de "alicerces" de pensar e agir e a tomada de novos; que implica mudanças práticas nas formas de serviços ou na disponibilidade de novos serviços. Essa situação estabelece a oportunidade para a criatividade emergente e o redirecionamento de necessidades existentes.

O Brasil é um dos países que pode beneficiar-se da quebra de paradigmas em termos do modelo filosófico e de uso prático de sistemas operacionais e programas proprietários e da adoção de programas livres *sob licença GPL*. O reconhecimento público de que o modelo de sistema operacional dominante não atendia todas as necessidades corporativas ou pessoais e a opção de experimentar outras soluções *SO/GPL* podem reiniciar um ciclo de criação e desenvolvimento intensivos de soluções tecno-científicas, que vem suprir necessidades desassistidas ou mesmo gerar novas necessidades. Daí resultam produtos

ou serviços renovados e inclusive novos. Redimensionam-se mercados e abrem-se novos mercados. O resultado prático é o aproveitamento de competências locais e o fomento a novas competências. Isso envolve direta e indiretamente:

1) aumento de ferramentas de produtividade científica e genéricas;

2) reciclagem de profissionais e fomento a capacitação de estudantes;

3) economia de recursos públicos e do potencial de reinvestimentos;

4) aumento de autonomia e de segurança tanto na iniciativa pública quanto privada;

5) participação da sociedade civil no fomento ao desenvolvimento local.

Que impacto essas mudanças podem produzir no ambiente de trabalho? Nas secretarias e serviços administrativos não se perceberá nem que se mudou de ambiente operacional e de programas. Na pesquisa científica haverá um ganho de facilidades e de estabilidade; embora situações específicas devam ser consideradas. Nos serviços técnicos e de engenharia há a perspectiva de ganho de produtividade. Nos serviços de rede de informática, a prestação de serviços, manutenção e segurança serão significativamente facilitados. Na interação com outras instituições externas, um instituto terá facilidades de reintegrar-se a um ambiente já em mudança para o sistema operacional GNU/Linux. Junto a União, haverá ganho significativo de economia ou de investimento.

2.2 Existe diferença entre GNU/Linux e Linux?

Em 1985, decorrente de ser impedido de melhorar um recurso de impressora, o estadunidense Richard Stallman criou a Fundação Pró Software Livre (*Free Software Foundation*), que fomenta a filosofia de que programas de computador são patrimônio da humanidade. A designação **GNU** identifica e caracteriza um projeto de criar um ambiente computacional completo e livre. Isso seria o equivalente a dizer que todos tem o direito a conhecer a fórmula de uma vacina que, fabricada segundo a necessidade de cada um ou grupo, pudesse salvar vidas. Por isso, falar de GNU/Linux significa entender e dar valor a um sistema de solidariedade humana.

O GNU/Linux diz respeito a todo o ambiente computacional criado para servir a humanidade, ilustrativamente pode ser entendido como um automóvel. Já o Linux é o núcleo, aquilo que faz o computador funcionar, podendo ser entendido como o motorista.

Com a divulgação em 1991 pelo estudante finlandês Linus Torvald do sistema operacional Linux, o **projeto GNU**, servindo-se do Linux, encontrou condições imediatas de integrar

os vários serviços em desenvolvimento, constituindo um ambiente computacional completo. Esse projeto visa fomentar uma filosofia de programas de códigos abertos, de uso, de implementação e de divulgação sempre livres (em que *livre* diz respeito a liberdade e não necessariamente a gratuidade).

2.3 Existe diferença entre programas de código de livre distribuição (Free software) e código aberto(Open Source)?

O código de livre distribuição caracteriza-se por quatro níveis de liberdade:

a) Liberdade de usar o programa;

b) Liberdade de alterar o programa conforme necessidades pessoais;

c) Liberdade de aperfeiçoar o programa e distribuir cópias para a comunidade; e

d) Liberdade de melhorar o programa e publicá-lo com essas melhorias.

A Licença Geral Pública (*General Public License*) regulamenta o exercício dessa liberdade. Um produto GPL, se for distribuído, terá de continuar GPL, obrigatoriamente com o código aberto e sob livre distribuição.

Já o programa de código aberto pode ser proprietário e com várias restrições a sua livre utilização. Tanto filosófica como praticamente, o código aberto não-GPL implica sujeição de uma pessoa ou grupo aos caprichos de outra pessoa ou grupo.

Convém mencionar que existem ainda os programas de códigos escondidos ou fechados, que estabelecem um nível de sujeição total de uma pessoa ou grupo ao arbítrio de outras pessoas ou grupos.

2.4 Entendendo o cenário de ferramentas

Código aberto Programa de código aberto é um programa que tem o seu código fonte exposto ao conhecimento de qualquer pessoa.

Programas livres (free software) Em geral os programas livres recaem em uma dessas 3 categorias:

Programa de domínio público programa sem nenhuma restrição.

Programa livremente distribuível programa que após as modificações deve ter autorização do seu proprietário para redistribuição

Programa de Licença Pública Geral (GPL) qualquer pessoa pode pegar o programa, alterá-lo se desejar, e redistribuir; porém não pode restringir esse mesmo procedimento a qualquer outra pessoa.

GNU/GPL é uma licença especial de programas, desenvolvida pela *Free Software Foundation*. GPL significa *General Public License*, Licença Pública Geral, isto é, que um programa tem desenvolvimento aberto e distribuição livre. GNU, palavra com um caráter propositalmente intrigante, significa **GNU is Not Unix**. O Projeto GNU tem por objetivo a criação de um completo sistema operacional.

2.4.1 Sistemas operacionais e distribuições GNU/Linux

Existem vários sistemas operacionais, isto é, um núcleo lógico no interior do computador que dá a ele a capacidade de agir. É interagindo com esse núcleo vital que outros códigos executáveis tornam o computador o que ele é: uma máquina poderosa nos procedimentos e aparentemente inteligente a serviço das necessidades humanas.

O GNU/Linux é um sistema operacional de código aberto e distribuição livre, caracterizando-se por ser estável, robusto no funcionamento, multitarefa, multi-usuário, nativo em rede, e seguro.

O GNU/Linux está disponível para uso na forma de distribuições, que poderíamos entender como sabores de Linux. Há o núcleo (kernel), o verdadeiro Linux, e vários códigos que podem ser integrados, constituindo uma distribuição, para atender necessidades gerais ou mais específicas de grupos de pessoas.

As atividades gerais atendidas pelo ambiente GNU/Linux e Programas Livres são: atividades de pesquisa, de ensino, de engenharia, produção técnica, edição gráfica, admininstrativas, de entretenimento (em desenvolvimento), e doméstica (mais recente). Atualmente existem um vasto suporte técnico, inúmeros treinamentos, uma farta documentação e muitos grupos de discussão.

2.4.2 Outras informações

Em 1969, a companhia estadunidense AT&T Bell lançou um sistema operacional robusto e elegante, o UNIX.

Em 1984, Richard Stallman, pesquisador do MIT, estabelece as bases de uma colaboração mundial. Sua maior contribuição não foi o código aberto; porém a filosofia GPL.

Em 1991, o finlandês Linus Torvald estabelece um clone do sistema Minix, que deveria constituir–se em um Unix para uso doméstico, ou seja, um sistema operacional suportado em microcomputadores pessoais.

No ambiente de desenvolvimento GPL busca–se recursos computacionais robustos, estáveis, eficientes e com portabilidade, que para muitos traduz–se em "bom, bonito e barato".

Pode–se executar aplicativos GNU/Linux em uma estação de trabalho Sun/Solaris (ver http://www.sun.com/linux/lxrun).

Existem outros sistemas operacionais do tipo **UNIX**: GNU/FreeBSD, GNU/Hurd e BeOS.

CAPÍTULO 3

INTRODUÇÃO AO GNU/Octave

O GNU/Octave é uma linguagem de alto nível basicamente voltada para computação numérica. Esse programa provê uma interface por linha de comandos para solução numérica de problemas lineares e não–lineares e para implementar outros experimentos numéricos usando uma linguagem que é compatível com o **Matlab**. O programa pode ser utilizado também em modo *script* (textos de programação) e permite incorporar módulos escritos nas liguagens C^{++}, C, Fortran e outras. O GNU/Octave foi escrito por John W. Eaton e muitos outros, estando disponível na forma GPL (Eaton, 2001).

O GNU/Octave tem ferramentas amplas para soluções numéricas de problemas comuns de álgebra linear, para a determinação de raízes de equações não–lineares, manipulações polinomiais e integração de equações diferenciais ordinárias e equações diferenciais algébricas.

Quando executado pelo comando **octave** em um terminal, o GNU/Octave resulta em algumas informações e inicia um ambiente de trabalho por linha de comando. Um exemplo é mostrado seguir:

$ octave

```
GNU Octave, version 2.1.35 (i386-redhat-linux-gnu).
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.
*** This is a development version of Octave.  Development releases
*** are provided for people who want to help test, debug, and improve
*** Octave.
***
*** If you want a stable, well-tested version of Octave, you should be
*** using one of the stable releases (when this development release
*** was made, the latest stable version was 2.0.16).
octave:1> t=linspace(0,2*pi,200);
octave:2>plot(sin(t))
```

Para se obter saídas gráficas na tela é necessário estar em um ambiente XWindow, o

GNU/OCTAVE criará automaticamente uma janela separada para a apresentação gráfica (Figura 3.1). Por outro lado, mesmo em ambiente texto pode-se capturar as saídas em um arquivo que será posteriormente visto em outro aplicativo. Para ver a lista de terminais suportados, utiliza-se o comando **gset term**. Como um exemplo de redirecionamento de saída, tem–se:

```
octave:1>t=linspace(0,2*pi,200);

octave:2>gset term postscript
octave:3>gset output ¨grafico.ps¨

octave:4>plot(sin(t))
```

Toda a parte gráfica do GNU/OCTAVE é feita pelo GNUPLOT (Williams et al., 2001).

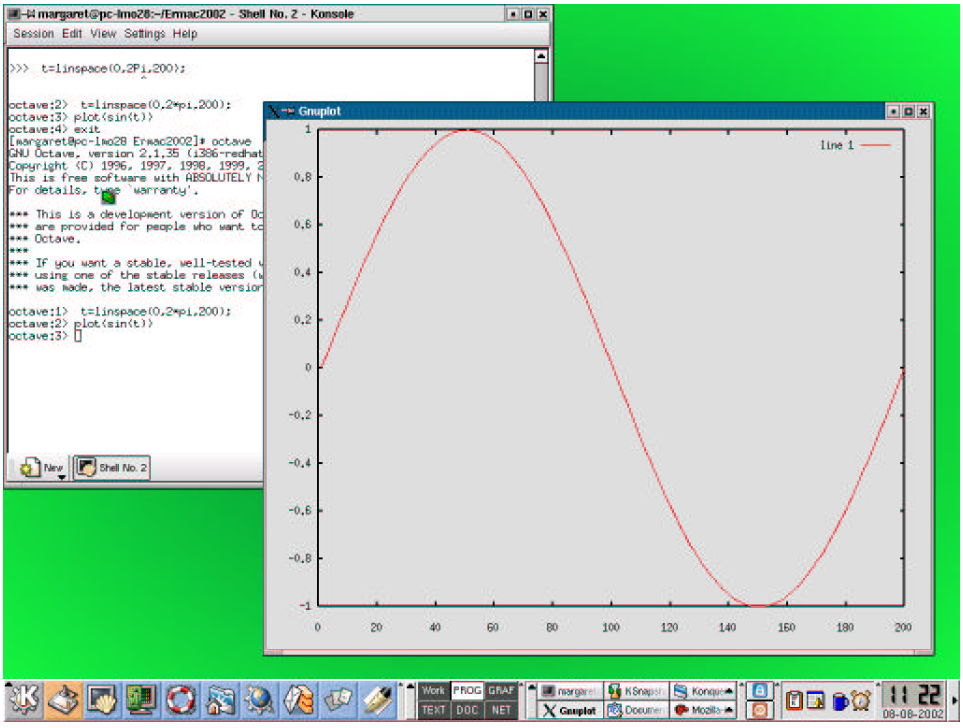

FIGURA 3.1 – Exemplo da interface GNU/OCTAVE e de uma saída gráfica.

3.1 Noções dos principais comandos

Cada comando no GNU/OCTAVE termina com um **Enter** (a tecla Enter deve ser acionada). Os comentários, em uma linha de comando, são obtidos iniciando a linha com os símbolos cerquilha (#) ou percentagem (%), sendo esse último o símbolo usado no

Matlab. Por exemplo:

```
#Este é um comentário no Octave
% Este é um comentário no Octave e no Matlab
```

Caso não se deseje que o resultado seja apresentado no monitor, pode–se terminar o comando com ponto–e–vírgula (;) seguido do **Enter**.

Para sair do GNU/OCTAVE, digite **quit** ou **exit** no prompt do **octave**. Caso haja problemas ao executar o **octave**, pode–se interrompê–lo com a tecla **CTRL-c**. Caso o sistema suporte o controle por tarefa (*job control*), pode–se suspender o **octave** pelo envio de um sinal **SIGTSTP**, usualmente obtido com a tecla **CTRL-z**.

Para obter ajuda, utiliza-se o comando **help**. Esse comando pode ser utilizado sozinho, fornecendo todas as informações que podem ser selecionadas, ou acompanhado do comando específico de que se queira obter informações.

Exemplo 1:

```
octave:1 > help
Help is available for the topics listed below.

 Additional help for built-in functions, operators, and variables
is available in the on-line version of the manual.

Use the command ‘help -i <topic>’ to search the manual index.
Help and information about Octave is also available

on  the WWW at http://www.che.wisc.edu/octave/octave.html and
via the help-octave@\-bevo.che.wisc.edu mailing list.

*** operators:
!    #    &&   )    +    -    .*   .^   ;    <>   >    \    |    ~=
!=   %    ’    *    ++   --   .**  /    <    =    >=   ]    ||
"    &    (    **   ,    .’   ./   :    <=   ==   [    ^    ~
*** reserved words:
all_va_args            end_unwind_protect      gplot
break                  endfor                  gsplot
catch                  endfunction             if
continue               endif                   return
```

```
else                    endwhile                try
elseif                  for                     unwind_protect
end                     function                unwind_protect_cleanup
end_try_catch           global                  while
```

⋮

Exemplo 2:

```
octave:1> help plot

plot is the user-defined function from the file
/usr/share/octave/2.1.35/m/plot/plot.m
 - Function File:  plot (ARGS)
     This function produces two-dimensional plots.  Many different
     combinations of arguments are possible.  The simplest form is
          plot (Y)
     where the argument is taken as the set of Y coordinates and the X
     coordinates are taken to be the indices of the elements, starting
     with 1.
     If more than one argument is given, they are interpreted as
          plot (X, Y, FMT ...)
```

No GNU/Octave, expressões são blocos básicos de construção de assertivas. Uma expressão calcula um valor, que pode ser impresso, testado, armazenado, passado a uma função ou designar um novo valor para uma variável por meio de um operador de atribuição. As operações algébricas fundamentais são apresentadas a seguir.

```
octave:1>7+9
ans=16

octave:2> 7-9
 ans=- 2

octave:3>3*6
ans=18

octave:4>2/3
ans = 0.66667
```

```
octave:5>2**3
ans=8

octave:6>2^3
ans= 8
```

Para utilizar o resultado de um cálculo anterior, pode–se atribuir o valor a uma variável ou referir-se a esse resultado por meio do indicador **ans** suprido automaticamente. O **ans** refere-se ao resultado calculado mais recente. Por exemplo:

```
octave:1>b=6^2
ans= 36
octave:2>sqrt(ans)
ans=6
octave:3>b
ans=36
```

O GNU/Octave conhece operações com números complexos, em que $\imath = \sqrt{-1}$. Por exemplo, no cálculo da raiz quadrada:

```
octave:1> sqrt(-6.0);
ans = 0.00000 + 2.44949i
```

O GNU/Octave mantém um histórico das linhas de comando executadas, até um número especificado pela variável **history_size**, que permanecem salvas em um arquivo. Esses comandos podem ser reapresentados por meio das teclas de ↑ e ↓, como também pelos comandos **CTRL-p** e **CTRL-n**, respectivamente. Existem ainda outros comandos disponíveis.

Outras funções básicas $\cos, \sin, \tan, \log, \exp, \cdots$ são expressas da forma usual. Por exemplo:

```
octave:1>cos(pi)
ans=    - 1

octave:1>log(2)
ans= 0.69314718055995
```

Em geral, é mais conveniente a criação de arquivos de programação. Os comandos do GNU/OCTAVE podem ser salvos em um arquivo texto de extensão **.m**, como por exemplo **prog.m**, para posterior processamento. Para executar esse arquivo dentro do ambiente, utiliza–se o nome do arquivo sem a extensão. Por exemplo

```
octave:1>prog
```

3.2 Operações com Matrizes

Existem operações definidas para matrizes que utilizam uma notação de comandos muito similar à apresentada nos comandos de aritmética básica. Alguns exemplos dessas operações são vistas a seguir.

- Definição de matrizes:

```
octave:2> A= [2,3,4;5,7,6;1,2,4]
A =

  2  3  4
  5  7  6
  1  2  4
```

Há várias matrizes muito utilizadas que podem ser construídas diretamente por funções pré–definidas. Por exemplo:

eye(N) para construir uma matriz identidade $N \times N$.

```
octave:22> eye(3)
ans =
  1  0  0
  0  1  0
  0  0  1
```

eye(N,M) para construir uma matriz com elementos de valor 1 na diagonal principal.

```
octave:23> eye(3,4)
ans =
  1  0  0  0
  0  1  0  0
```

```
  0  0  1  0
```

ones(N,M) para construir uma matriz $N \times M$ com elementos de valor 1;

```
octave:24> ones(2,3)
ans =
  1  1  1
  1  1  1
```

zeros(N, M) para construir uma matriz $N \times M$ com elementos de valor 0;

```
octave:25> zeros(3,4)
ans =
  0  0  0  0
  0  0  0  0
  0  0  0  0
```

diag(V,K) para construir uma matriz com elementos do vetor V em uma diagonal K;

```
octave:26> diag([2,3,4],2)
ans =
  0  0  2  0  0
  0  0  0  3  0
  0  0  0  0  4
  0  0  0  0  0
  0  0  0  0  0
```

rand(N,M) para construir uma matriz $N \times M$ com elementos de valor aleatório;

```
octave:39> rand(2,4)
ans =

  0.47778  0.95810  0.22112  0.44798
  0.81703  0.21598  0.42929  0.69537
```

A seguir apresentam–se algumas operações com matrizes.

- Determinante de uma matriz. Seja

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 7 & 6 \\ 1 & 2 & 4 \end{bmatrix}$$

```
octave:28> det(A)
ans = 2.0000
```

- Matriz transposta da matriz $A = A^t$:

```
octave:29> A'
ans =
  2  5  1
  3  7  2
  4  6  4
```

- Matriz inversa da matriz $A = A^{-1}$:

```
octave:30> inv(A)
ans =
   8.00000  -2.00000  -5.00000
  -7.00000   2.00000   4.00000
   1.50000  -0.50000  -0.50000
```

- Multiplicação de matrizes. Seja

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

```
octave:32> A*B
ans =
   2  12  24
   5  28  36
   1   8  24
```

A expressão $A^{-1}B$ é obtida pelos comandos:

```
octave:37> A\B
ans =
    8.0000   -8.0000  -30.0000
   -7.0000    8.0000   24.0000
    1.5000   -2.0000   -3.0000

octave:38> inv(A)*B
ans =
    8.0000   -8.0000  -30.0000
   -7.0000    8.0000   24.0000
    1.5000   -2.0000   -3.0000
```

- Mutiplicação elemento a elemento de matrizes

```
octave:33> A.*B
ans =
   2   0   0
   0  14   0
   0   0  12
```

- Inversão dos elementos de uma matriz

```
octave:35> 1./A
ans =
  0.50000  0.33333  0.25000
  0.20000  0.14286  0.16667
  1.00000  0.50000  0.25000
```

- Multiplicação de uma matriz por um escalar:

```
octave:34> 3*A
ans =
   6   9  12
  15  21  18
   3   6  12
```

3.3 Operações com funções

Alguns recursos no manuseio de funções são apresentados nesta seção.

- Definição de uma função a ser calculada para um certo valor da variável independente.

  function

  corpo da função

  endfunction

  Exemplo, definição da função:

```
octave:41> function y=f(x)
> b=0.01;
> a0=10;
> c=1000;
> y=a0 * exp(b*x) * sin(2*pi/c *x)
> endfunction
```

  uso da função definida:

```
octave:42> f(2)
y = 0.12820
ans = 0.12820
```

- Solução de sistemas lineares $Ax = b$. Por exemplo:

```
octave:43> A=rand(3,3)
A =

  0.454912  0.718749  0.923162
  0.048882  0.485173  0.068764
  0.841294  0.962446  0.644441

octave:44> b=rand(3,1)
b =
```

```
  0.32278
  0.36149
  0.12898

octave:45>x= A\b
ans =

  -0.89913
   0.81302
   0.15972
```

Isso é conceitualmente equivalente a usar $A^{-1}b$, mas evita calcular essa inversa explicitamente.

- Solução de um conjunto de equações não–lineares. Neste cálculo é utilizada a sub–rotina *hybrd* do MINIPACK. Exemplo de uso:

  Sejam

$$\begin{aligned} y_1 &= -2x_1^2 + 3\,x_1\,x_2 + 4\,sen(x_2) - 6, \\ y_2 &= 3\,x_1^2 - 2x_1\,x_2^2 + 3\,cos(x_1) + 4, \end{aligned}$$

  com as condições iniciais $y_1 = 1, y_2 = 2$. Então, no GNU/Octave, a solução é implementada como:

```
octave:48> function y=f(x)
> y(1)=-2*x(1)**2+3*x(1)*x(2)+4*sin(x(2))-6;
> y(2)=3*x(1)**2 - 2*x(1)*x(2)**2+3*cos(x(1))+4;
> endfunction
octave:49> [x,info]=fsolve('f',[1;2])
x =

  0.57983
  2.54621

info = 1
```

  O valor **info=1** indica que a solução converge.

- Cálculo da integral definida em um intervalo para uma variável. Para isso utiliza–se a função **quad**, em uma operação estabelecida da seguinte forma:

  ```
  [v,ier,nfun,err]= quad(‘‘f’’,a,b,tol,sing)
  ```

  em que, quanto ao lado direito da expressão, f é o nome da função a ser chamada para calcular o valor do integrando (deve ter a forma **y=f(x)**, sendo y e x escalares); os argumentos a e b são os limites da integração (que podem ser $\pm\infty$); o argumento *tol* é um vetor que especifica a precisão desejada do resultado; e o argumento *sing* é um vetor de valores em que o integrando é conhecido ser singular. Quanto ao lado esquerdo da expressão, têm–se que v é o resultado da integração e *ier* é um código de erro (0 indica que a operação foi bem sucedida); o valor *nfun* indica quantas iterações foram necessárias; e o valor de *err* é uma estimativa do erro na solução. Exemplo:

  Seja

  $$\int_0^3 x\ sen\left(\frac{1}{x}\right)\ \sqrt{|1-x|}\,dx,$$

  então, no GNU/Octave, a solução é implementada como:

  ```
  octave:62> function y=f(x)
  >  y=x.*sin(1./x).*sqrt(abs(1-x));
  > endfunction

  octave:63> [v, ier, nfun, err] = quad ("f", 0, 3)

  v = 1.9819
  ier = 1
  nfun = 5061
  err =  1.1522e-07
  ```

  OBS: Embora prescindível para o cálculo de **quad**, a forma *ponto* dos operadores usados no GNU/Octave torna mais fácil gerar um conjunto de pontos para gerar gráficos (pois isso possibilita utilizar argumentos vetoriais para produzir resultados vetoriais).

3.4 Gráficos

- Fazer um gráfico bidimensional. Exemplo:

```
octave:67> function y=f(x)
>  b=-0.01;
> a0=10;
> c=100;
> y=a0 .* exp(b.*x) .* sin(2.*pi/c .*x)
> endfunction


octave:68> x=linspace(0,1000);
octave:69> plot(f(x))

octave:70> gset term postscript
octave:71> gset output ¨harm.ps¨
octave:72> replot
```

Com esse exemplo gerou–se um gráfico no monitor e uma saída gráfica no formato *postscript*, apresentado na Figura 3.2.

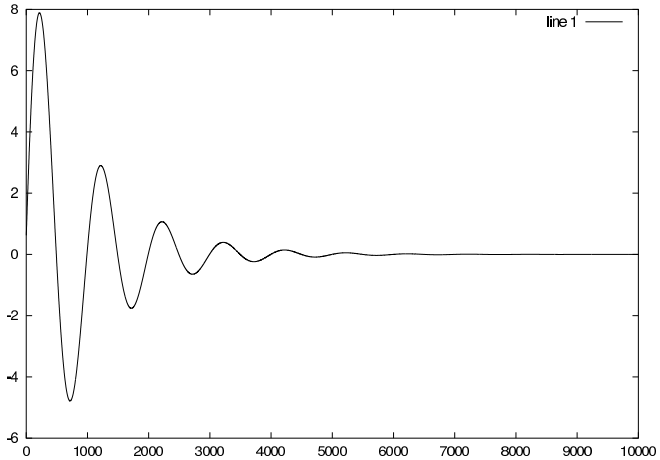

FIGURA 3.2 – Saída *postscript* de gráfico 2D no GNU/Octave.

Pode–se também escrever uma seqüência de comandos em um arquivo texto e executá-lo copiando esse texto diretamente no **prompt**, ou apenas chamando seu nome sem a extensão. Por exemplo o arquivo ExFit.m ajusta uma curva a

um conjunto de dados pelo método de quadrados mínimos e gera um gráfico com esse resultado (Figura 3.3).

```
% Arquivo: ExFit.m
%data: 10/06/2002
% Finalidade:  Exemplo de ajuste de dados pelo metodo dos quadrados
               minimos

clear all
printf('\n Ajuste de dados pelo metodo dos quadrados minimos \n');
        %os pontos x
x =[1.1 2.5 3.9 4.5 5.4 6.0 6.5 7.0 8.9];
        %f(x)
fx=[1.2 1.5 1.7 2.5 2.7 2.8 3.5 4.1 5.1];

gset title "Dados a serem ajustados por mínimos quadrados "
gshow title
xlabel('x')
ylabel('fx')
plot(x,fx);
printf('\n\t\t\t\tPara continuar pressione a tecla enter ...\n\n');
pause -1 % pausa
     %n é o grau do polinômio a ser ajustado
n=1;
     %p é um vetor que contém os coeficientes para um polinômio
        de grau n

p=polyfit(x,fx,n);
     %d é um vetor que contém os valores do polinômio p nos pontos x
d=polyval(p,x);

                      %Visualizando os resultados
gset title   'Curva ajustada'
gshow title
xlabel(' x')
ylabel('fx')
plot(x,d,";curva;",x,fx,"ob;dados;")
```

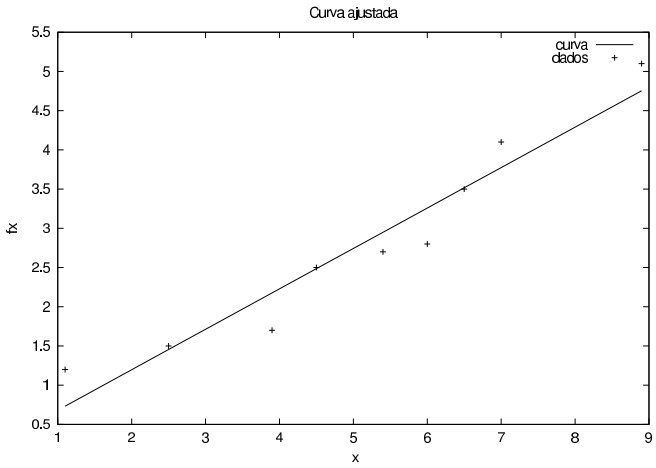

FIGURA 3.3 – Resultado gráfico do programa ExFit.m

- Fazer um gráfico tridimensional. Há muitas formas de se criar um gráfico 3D. Primeiro é necessário gerar a malha de dados a serem visualizados (e.g. usando o recurso **meshgrid**). Para executar a visualização 3D pode–se utilizar o comando **mesh** e para geração de contornos o comando **contour** . Exemplo:

```
octave:1>b=-0.01;
octave:2>a0=10;
octave:3>c=100;

octave:4>  xx = yy = linspace (0, 200, 100)';
octave:5> [x, y] = meshgrid (xx, yy);
octave:6> z=a0 .*exp(b.*x) .*sin(2*pi/c.*x) .*exp(b.*y) .*sin(2*pi/c.*y);
octave:7> mesh(x,y,z);

octave:8>pause -1
octave:9>printf('\n\t\t\tPara continuar pressione a tecla enter...\n\n');

octave:10>contour(xx,yy,z,10);
```

Isso resulta no gráfico 3D apresentado na Figura 3.4.

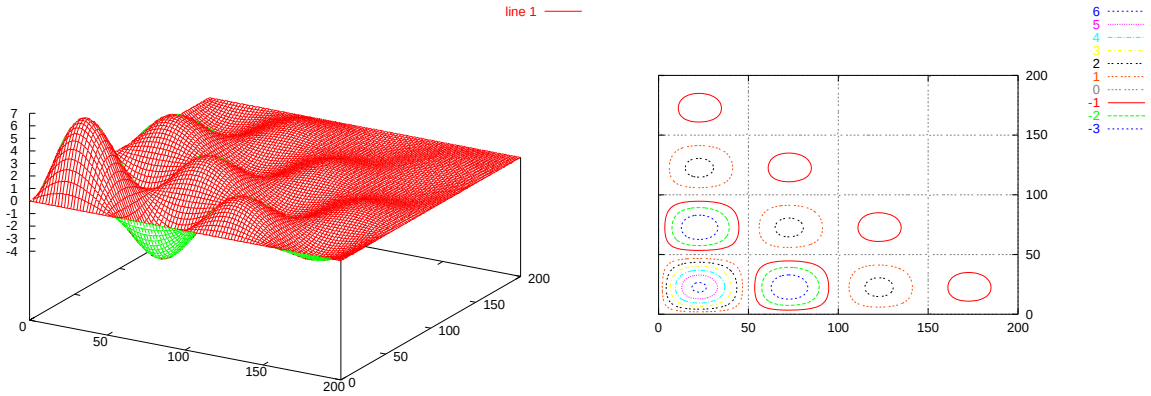

FIGURA 3.4 – Gráficos 3D e em curvas de nível no GNU/Octave.

Muitos outros recursos gráficos podem ser obtidos no GNU/Octave, incluindo animações.

3.5 Elementos de programação

A utilização do GNU/Octave apresenta um potencial maior quando se faz uso de recursos de programação. Com este propósito, é importante conhecer as formas de controlar os fluxos de cálculos. A seguir são relacionados de forma breve alguns exemplos de uso desses elementos.

Controle **if**

```
if (rem (x, 2) == 0)
  printf ("x is even\n");
else
  printf ("x is odd\n");
endif
```

Controle **for**

```
fib = ones (1, 10);
for i = 3:10
  fib (i) = fib (i-1) + fib (i-2);
endfor
```

Controle **while**

```
fib = ones (1, 10);
i = 3;
while (i <= 10)
  fib (i) = fib (i-1) + fib (i-2);
  i++;
endwhile
```

Controle **switch**

```
nome='Maria';

switch (nome)
case ('Ana')
        printf('Ana\n');
case ('Liam')
printf('Liam\n');
otherwise
          printf('Nome desconhecido\n');
endswitch
```

O comando **break** pode ser utilizado para sair de qualquer um desses controles. O comando **continue** pode ser utilizado para os laços **for** e **while** quando se deseja retornar a condição desse laço.

No Apêndice C são apresentados alguns exemplos da utilização desses controles em uma programação mais completa.

REFERÊNCIAS BIBLIOGRÁFICAS

Eaton, J. **Octave manual**, 2001. <http://www.octave.org/doc/>. Acesso em: 01/09/2002.

Williams, T.; Kelley, C.; Lang, R.; Kotz, D.; Campbell, J.; Elber, G.; Woo, A.; et al. **Gnuplot FAQ**, 2001. <http://www.ucc.ie/gnuplot/>. Acesso em: 01/09/2002.

CAPÍTULO 4

INTRODUÇÃO AO GNU/Maxima

O GNU/Maxima é um programa de computador, sob a licença GPL, para lidar com sistemas algébricos. William F. Shelter desenvolveu esse programa em LISP baseado na implementação original do *Macsyma* no MIT. O GNU/Maxima pode ser utilizado para cálculos matemáticos, manipulação simbólica, computação numérica e criação de gráficos (Schelter, 2001b,a; ICM Institute for Computational Mathematics, 2002).

Quando executado pelo comando **maxima** em um terminal, o GNU/Maxima resulta em informações iniciais e inicia um ambiente de trabalho por linha de comando. Um exemplo é mostrado seguir:

```
$ maxima
GCL (GNU Common Lisp)  Version(2.4.0) Wed May  9 12:02:00 CDT 2001
Licensed under GNU Library General Public License
Contains Enhancements by W. Schelter
Maxima 5.6 Wed May 9 12:01:49 CDT 2001 (with enhancements by W. Schelter).
Licensed under the GNU Public License (see file COPYING)
(C1) 2*3;
(D1)                                  6
(C2)
```

Nesta forma de execução, para se obter saídas gráficas é necessário estar em um ambiente XWindow. Pode–se trabalhar também por meio de uma interface gráfica conhecida como XMaxima, apresentada na Figura 4.1, que é acionada com o comando **xmaxima**.

4.1 Noções dos principais comandos

Cada comando no GNU/Maxima termina com um ponto–e–vírgula (;) e os comentários são comandos constituídos de textos contidos entre aspas, por exemplo:

```
¨Este é um comentário no Maxima¨;
```

Caso não se deseje que o resultado seja apresentado no monitor, pode–se terminar o comando com $ ao invés de ponto–e–vírgula; A finalização do GNU/Maxima é feita com o comando

```
quit();
```

ou no modo gráfico selecionando-se no menu File e Exit. Para obter ajuda, utiliza-se o comando **describe** ou o comando constituído do símbolo interrogação, um espaço em branco e o comando de que se deseje obter informações. Exemplo:

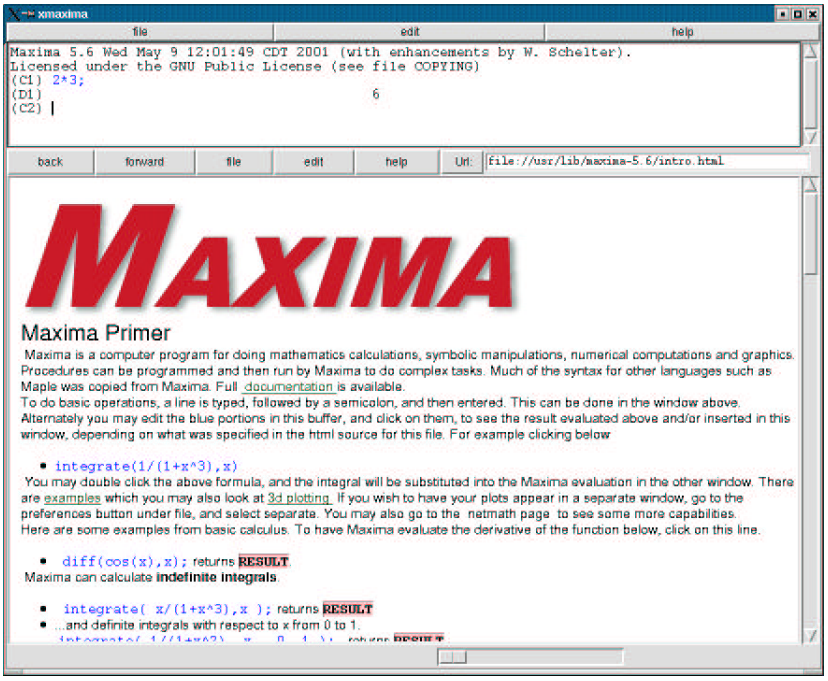

FIGURA 4.1 – Exemplo da interface gráfica XMaxima.

```
(C1) describe(plot);

 0: (maxima.info)Plotting.
 1: Definitions for Plotting.
 2: OPENPLOT_CURVES :Definitions for Plotting.
 3: PLOT2D :Definitions for Plotting.
 4: PLOT2D_PS :Definitions for Plotting.
 5: PLOT3D :Definitions for Plotting.
 6: PLOT_OPTIONS :Definitions for Plotting.
 7: SET_PLOT_OPTION :Definitions for Plotting.
Enter n, all, none, or multiple choices eg 1 3 :
```

ou

```
(C1) ? plot

 0: (maxima.info)Plotting.
 1: Definitions for Plotting.
 2: OPENPLOT_CURVES :Definitions for Plotting.
 3: PLOT2D :Definitions for Plotting.
 4: PLOT2D_PS :Definitions for Plotting.
 5: PLOT3D :Definitions for Plotting.
```

```
 6: PLOT_OPTIONS :Definitions for Plotting.
 7: SET_PLOT_OPTION :Definitions for Plotting.
Enter n, all, none, or multiple choices eg 1 3 :
```

As operações algébricas são feitas da forma usual e os exemplos são apresentados a seguir.

```
(C1) 7+9;
(D1)     16

(C2) 7-9;
(D2)    - 2

(C3) 3*6;
(D3)     18

(C4) 2/3.0;
(D4)      0.66666666666667
```

Outras operações de interesse são:

- Representar a fração expressa por $\frac{2}{3}$:

  ```
  (C5) 2/3;
  ```

  $$(D5) \quad \frac{2}{3}$$

  que é diferente de efetuar a divisão por meio de 2/3.0.

- Calcular a raiz quadrada:

  ```
  (C6) sqrt(-6.0);
  (D6)    2.449489742783178 %I
  ```

  O GNU/Maxima conhece operações com números complexos, em que $\imath = \sqrt{-1}$ é representado por %I.

- Utilizar o resultado de um cálculo anterior. Para isso, pode–se atribuir o valor a uma variável ou referir-se a esse resultado por meio de um indicador suprido automaticamente (um rótulo). O símbolo de % refere-se ao resultado calculado mais recente. Por exemplo:

  ```
  (C7) 6^5.0;
  (D7)      7776.0
  (C8) a:D7;
  ```

```
(D8)      7776.0
(C9) %^(1/5.0);
(D9)        6.0
(C10) 2*a;
(D10)      15552.0
(C11) a^(1/5.0);
(D12)        6.0
```

- Para reapresentar as últimas n linhas de dados e/ou comandos, utiliza–se o comando **playback**. Exemplo:

```
(C18) 1+2;
(D18)     3
(C19) 2*6;
(D19)     12
(C20) exp(-20);
(D20)   %E ^ -20
(C21) playback(3);


(D19)     12
(C20) EXP(-20);
(D20)   %E ^ -20
(D21)    DONE
(C22)
```

- Criação de um arquivo de programação. Os comandos do GNU/Maxima podem ser salvos em um arquivo texto extensão **.mac**, como por exemplo **prog.mac**, para posterior processamento. Para executar esse arquivo dentro do ambiente, utiliza–se o comando **batch**. Por exemplo

```
(C1)batch(¨prog¨);
```

- As funções cos, sin, tan, log, exp, etc ... são expressas da forma usual. Por exemplo:

```
(C13) cos(%PI);
(D13)        - 1


(C14) log(2.0);
(D14)         0.69314718055995
(C15) exp(D14);
(D15)        2.0
```

- Calcular fatorial:

```
(C19) factorial(5);
(D19)        120
```

- Fatorar em números primos:

```
(C20) factor(120);
```

(D20) $2^3\ 3\ 5$

- Expandir e fatorar polinômios:

```
(C21) expand((x+7)*(x-7));
```

(D21) $x^2 - 49$

```
(C22) factor(D21);
(D22)          (x - 7) (x + 7)
```

- Simplificar uma expressão:

```
(C23)display((25*x^5*y^7*z^9)^(1/5));
```

(D23) $(25x^5\,y^7\,z^9)^{\frac{1}{5}} = 25^{\frac{1}{5}}\ x\,y^{\frac{7}{5}}\,z^{\frac{9}{5}}$

- Calcular uma decomposição parcial fracionária para expressões fracionárias:

```
(C32) expand(1/(x-5)*1/(x-7));
(D32)
```

$$\frac{1}{x^2 - 12\,x + 35}$$

```
(C33) partfrac(%,x);
(D33)
```

$$\frac{1}{2(x-7)} - \frac{1}{2(x-5)}$$

4.2 Operações com Matrizes

Algumas das operações básicas com matrizes são vistas a seguir.

- Exibição de matrizes:

```
(C9) A:matrix ([2,3],[5,7]);
```

$$A = \begin{bmatrix} 2 & 3 \\ 5 & 7 \end{bmatrix}$$

- Determinante da matriz A:

```
(C10)determinant(A);
(D10) -1
```

- Matriz transposta da matriz A:

```
(C10)transpose(A);
```

$$\begin{bmatrix} 2 & 5 \\ 3 & 7 \end{bmatrix}$$

- Matriz inversa da matriz A:

```
(C10)Ainv:invert(A);
```

$$\begin{bmatrix} -7 & 3 \\ 5 & -2 \end{bmatrix}$$

- Multiplicação de matrizes:

```
(C10)A.Ainv;
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Observar que um ponto (.) é usado para multiplicação de matrizes e não um asterisco (*).

- Multiplicação de uma matriz por um escalar:

```
(C10)3*A;
```

$$\begin{bmatrix} 6 & 9 \\ 15 & 21 \end{bmatrix}$$

- Operação com manuseio simbólico:

```
(C3) E:matrix([a,b,c],[a-b,b-c,c*c],[1,2,3]);
```

$$E = \begin{bmatrix} a & b & c \\ a-b & b-c & c^2 \\ 1 & 2 & 3 \end{bmatrix}$$

```
(C4) determinant(E);
```

(D4) $-b(3(a-b)-c^2)+a(3(b-c)-2c^2)+c(c-b+2(a-b))$

4.3 Operações com funções

Alguns recursos no manuseio de funções são apresentados nesta seção.

- Definição de uma função a ser calculada para um certo valor da variável independente. Exemplo:

```
(C1) f(x):=x^3-x^2+3*x; f(-1);
(D1) -5
```

- Solução de uma equação. Exemplo:

```
(C1)solve(8*x+ 7=11);
(D1) x=1/2
```

- Solução de uma equação para uma variável designada Exemplo:

```
(C1)solve(8*x+ 7+5* y=9,y);
(D1) y= -(8 x-2)/5
```

- Cálculo de $\sum_{x=a}^{b} f(x)$, em que $f(x) = x^5$. Exemplo:

```
(C1) sum(x^5,x,1,8);
(D1) 61776
```

- Cálculo da derivada com respeito a uma variável. Exemplo:

```
(C1) diff(x^5,x);
(D1)  5  x^4
```

- Diferenciação implícita em dois passos.
  a) Informa–se ao GNU/Maxima que uma variável é dependente de outra:
  ```
  (C1) depends(y,x);
  (D1) [y(x)]
  ```
  b) Diferencia-se a função de forma implícita:
  ```
  (C2)diff(x^2*y=9,x);
  (D2)
  ```
  $$x^2\frac{\mathrm{d}y}{\mathrm{d}x} + 2xy = 0$$

- Integração

```
(C1)integrate(5 * x^4,x);
(D1) x ^5
```

- Cálculo da integral definida em um intervalo

```
(C2) integrate(5 * x^4,x,1,3);
(D2)  242
```

- Solução de sistemas lineares. Por exemplo:

```
(C1) linsolve([3*x+4*y-z=7,2*x+a*y+b*z=13,x+y+z=10],[x,y,z]);
(D1)
```

$$x = \frac{33b + 17a - 65}{b + 4a - 10} \qquad y = \frac{23b - 18}{b + 4a - 10} \qquad z = \frac{23a - 53}{b + 4a - 10}$$

- Solução de um conjunto de equações não–lineares. Por exemplo:

```
(C29)eq1: x^3 +5 * x* y + y^2 = 0$
(C30)eq2: 3*x +2* y =1$
(C31)solve([eq1,eq2]);
(D31) [[y = - 0.02483745477298, x = 0.34989161849711],
           [y = 0.71257391845627, x =-0.14171594563751],
           [y = - 7.062736205593349, x = 5.041824295922656]]
```

4.4 Gráficos

- Fazer um gráfico bidimensional. Exemplo:

```
(C9) plot2d([cos(2*x),x^3],[x,-%pi,%pi]);
```

Isso resulta no gráfico apresentado na Figura 4.2, que possui na lateral esquerda (executado no modo interface gráfica) um menu que possibilita fazer várias tarefas. Entre essas tarefas está salvar o arquivo no formato postscript. Um exemplo dessas saídas é apresentado na Figura 4.3.

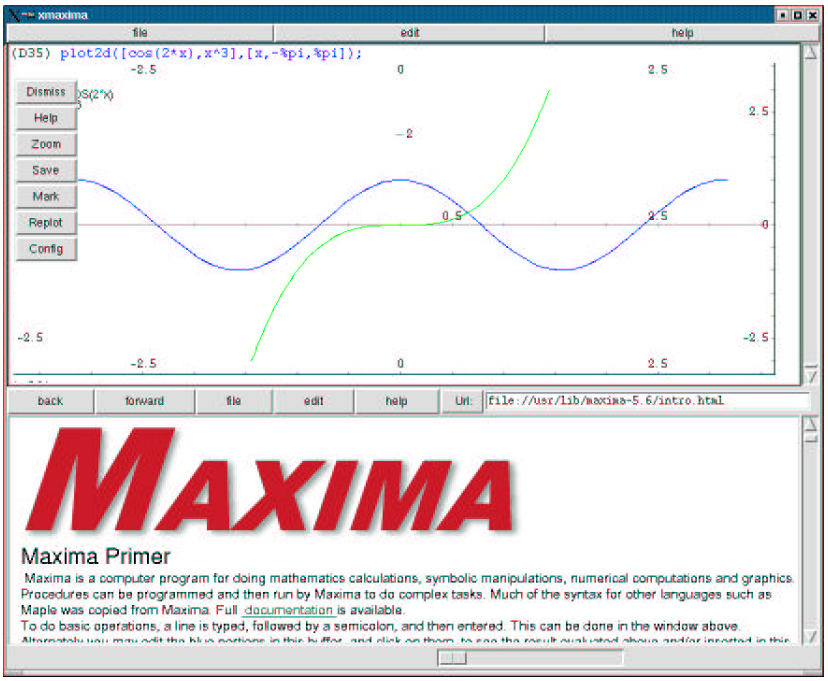

FIGURA 4.2 – Ambiente de tratamento de gráfico bidimensional.

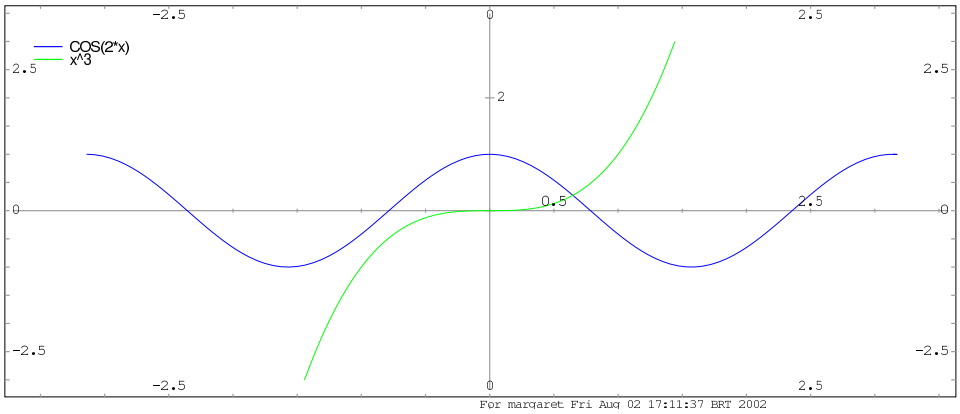

FIGURA 4.3 – Saída *postscript* de gráficos 2D no GNU/Maxima.

- Fazer um gráfico tridimensional. Exemplo:

  ```
  (C10)plot3d(sin(x^2+y^2)/(x^2+y^2),[x,-5,5],[y,-5,5],[grid,45,45]);
  ```

  Isso resulta no gráfico 3D. É possivel girar a figura no espaço. Um exemplo dessas saídas é apresentado na Figura 4.4.

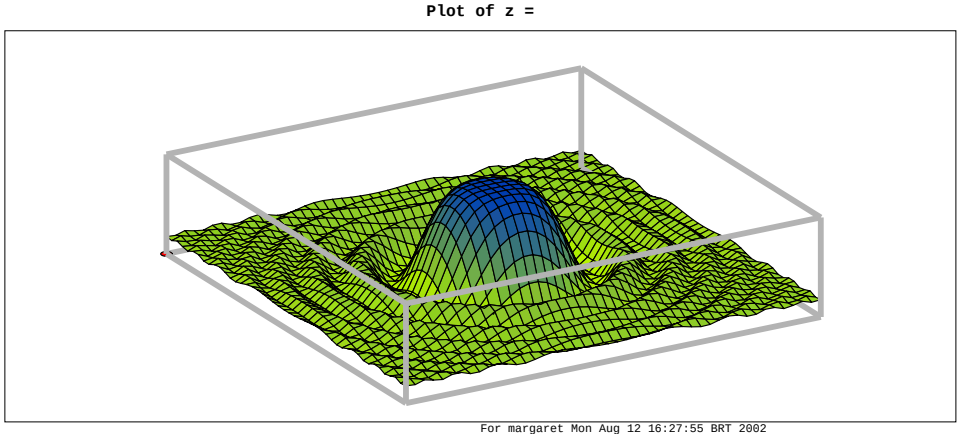

FIGURA 4.4 – Saída *postscript* de gráficos 3D no GNU/Maxima.

Muitos outros recursos gráficos podem ser obtidos no GNU/Maxima, incluindo animações.

REFERÊNCIAS BIBLIOGRÁFICAS

Institute for Computational Mathematics (ICM). **Interactive demos of mathematical computations**, 2002.
<http://icm.mcs.kent.edu/research/demo.html>. Acesso em 01/09/2002.

Schelter, W. **Basic Maxima commands**, 2001a.
<http://math.cochise.cc.az.us/Doc/maxima/ basic_maxima.html>. Acesso em 01/09/2002.

——. **Maxima manual**, 2001b.
<http://www.ma.utexas.edu/users/wfs/maxima.html>. Acesso em 01/09/2002.

CAPÍTULO 5

CONCEITOS BÁSICOS DE LaTeX

O LaTeX é um sistema de tipografia digital muito utilizado para a produção de textos científicos e matemáticos devido a sua performace e alta qualidade (Lamport, 1994). Ele também pode ser utilizado para outros tipos de documentos, de cartas simples até livros. O LaTeX utiliza o TeX como seu mecanismo de formatação (Knuth, 1984). Esse sistema é multiplataforma e de fácil uso. A idéia deste capítulo é apresentar alguns dos principais recursos de formatação de texto disponívels e onde é possivel buscar outras formas mais avançadas de formatação.

Textos básicos e avançados para formatação em LaTeX são apresentados em (Lamport, 1994; Goossens et al., 1995; Kopka e Daly, 1999; Oetiker et al., 2001; Steding-Jessen, 2001) e (Steding-Jessen, 1998)

Os textos em LaTeXsão escritos em arquivos texto com a extensão **.tex**. A seguir são processados por um programa chamado **latex**, que gera um arquivo de mesmo nome com extensão **.dvi**. Esses arquivos podem ser visualizados no programa **xdvi** ou convertidos em arquivos postscript, extensão **.ps**, com o programa **dvips**, ou ainda convertidos em arquivos PDF, com o programa **ps2pdf**. Pode–se processar os arquivos **.tex** diretamente para PDF com o programa **pdflatex**. Em (Goossens et al., 1995) e (Steding-Jessen, 2001) encontram–se detalhes de como gerar os códigos **.tex** para isso.

Em um terminal, operando por linha de comando, apresenta-se a forma explícita de lidar com um arquivo LaTeX. Suponha–se ter um arquivo *exemplo.tex* e um arquivo de referências convenientemente estabelecido (e.g. *referencia.bib*). Os comandos seriam:

```
latex exemplo
latex exemplo
bibtex exemplo
latex exemplo
latex exemplo
dvips exemplo -o exemplo.ps
ps2pdf exemplo.ps exemplo.pdf
```

O comando **latex** foi repetido duas vezes para que a operação ajustasse, quando necessário, alterações de referências.

Esses programas e uma série de outras facilidades (BiBTeX, makeindex, etc.) pertencem

ao pacote TeTEX, que está disponível na maioria das distribuições GNU/Linux. No entanto, o que em geral pode ocorrer é esse pacote não estar instalado. Há uma série de ferrametas úteis neste contexto, entre elas o recurso (pacote) **psutils**, que manipula arquivos **.ps**; o **xfig**, que permite fazer desenhos e diagramas de bloco; o **gimp** que permite a edição e a conversão de formatos de imagens; e o **Kile**, apresentado Figura 5.1, que é uma interface gráfica, integrando editor–de–texto, chamadas ao processador de LATEX e chamadas a outros programas correlatos. O Kile que em geral não está disponível nas distribuições e precisa ser instalado. Os endereços desses programas encontram–se na Tabela 1.1, página 13.

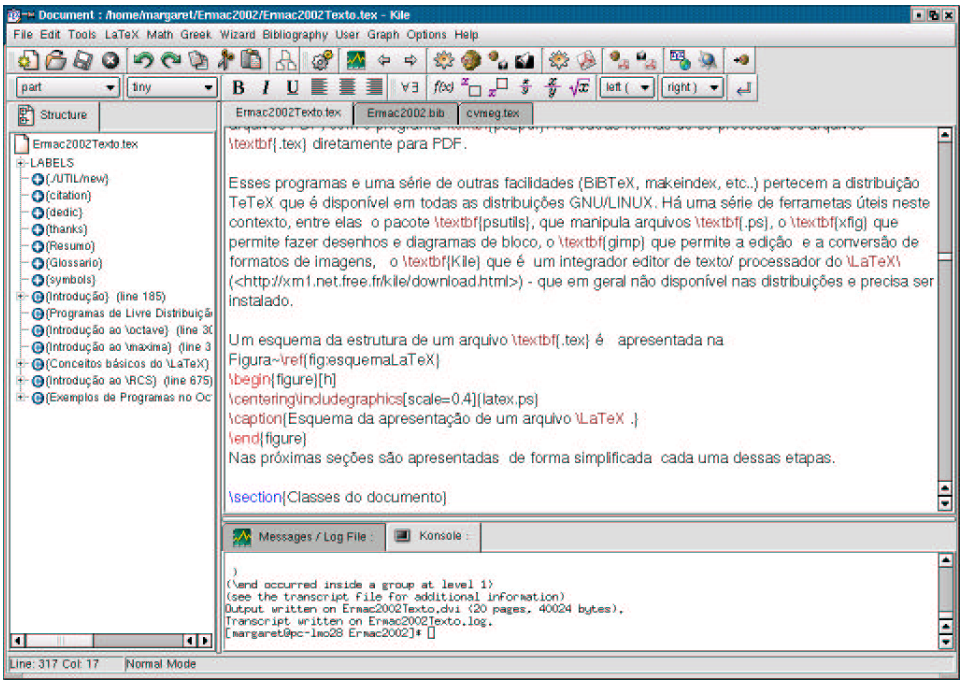

FIGURA 5.1 – Kile: interface gráfica, integrando editor–de–texto, chamadas ao processador de LATEX e chamadas a outros programas correlatos.

Um esquema da estrutura de um arquivo **.tex** é apresentada na Figura 5.2.

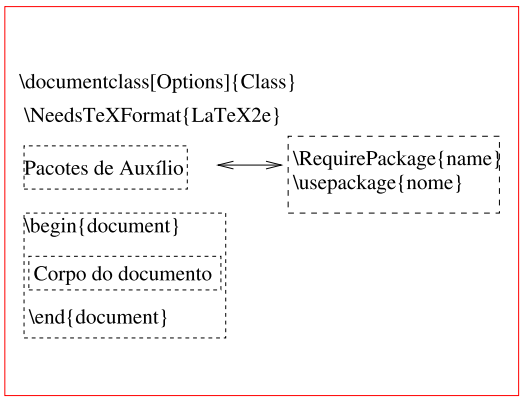

FIGURA 5.2 – Esquema da apresentação de um arquivo LaTeX.

Nas próximas seções são apresentadas de forma simplificada cada uma dessas etapas.

5.1 Classes do documento

O LaTeX possui formatos básicos de processamento de textos. Esses formatos são definidos pelas seguintes classes:

article para documentos pequenos, como artigos de revistas;

book para livros, ou textos desse porte, com a inclusão de vários capítulos;

letter para cartas pessoais ou comerciais;

report para relatórios técnicos, manuais e apostilas, com alguns capítulos;

slides para apresentação de material para transparências.

Existem também muitas outras classes criadas ou derivadas dessas, em geral desenvolvidas por colaboradores, para formatar textos específicos como cartas personalizadas, teses ou relatórios internos e artigos para revistas. Muitas dessas novas classes podem ser encontradas na internet. Nesse contexto, duas classe em especial são muito utilizadas pelos organizadores desse mini–curso. Uma delas é a classe Prosper, apresentada no Apêndice B (Goualard, 2001). Essa classe é muito simples de ser utilizada e permite a criação de diapositivos (*slides*) para canhões de projeção com efeitos de movimento, cores e delimitação de área . A outra é a classe **A0poster**, que permite a criação de cartazes no formato A0 ($1,23\text{m} \times 0,88\text{m}$). Maiores informações sobre essas classes ou pacotes podem ser encontradas com a ferramenta de busca (*search*) no endereço <http://www.ctan.org>.

5.1.1 Macros padrões

As macros das classes do documento são:

\documentclass define a classe do documento para o arquivo em questão. Na versão anterior do LaTeX, a 2.09, utilizava-se o comando *\documentstyle*;

\begin{document} inicia o corpo do documento;

\end {document} termina o corpo do documento;

As opções básicas das classes do documento são:

a4paper ou **letterpaper** que define o tamanho do papel como A4 ou carta;

11pt ou **12pt** que controla o tamanho da fonte em 11 ou 12 pontos;

draft para apresentar o texto em forma de rascunho, por exemplo: sem figuras;

fleqn para manter as fórmula matemáticas na margem esquerda (o padrão é que elas fiquem centralizadas);

leqno para numerar as fórmulas à esquerda (o padrão é que os números fiquem à direita);

openright ou **openany** para começar os capítulos sempre nas páginas de número par ou em qualquer das páginas;

5.2 Pacotes de Auxílio (usepackage)

Os ambientes e comandos do LaTeX não estão necessariamente pré–definidos. Para utilizar esses recursos é preciso fazer uma chamada ao pacote que os definem. Pode-se determinar que um pacote é essencial a compreensão do documento

\RequirePackage[opções do pacote]nome do pacote

ou que ele é necessário à formatação do documento

\usepackage[opções do pacote]nome do pacote

Existem inúmeros pacotes disponíveis na distribuição teTeX. Em geral, é possível ler suas documentações no *index.html* do diretório */usr/share/texmf/doc*.

5.3 Corpo do Documento

5.3.1 Ambientes

Tabelas, figuras, listas, fórmulas, desenhos e outros casos específicos são definidos em ambientes especiais dentro do corpo do documento. De uma forma geral tem-se:

```
\begin{nome do ambiente}
\caption{título}
\label{referência}
\end{nome do ambiente}
```

A seguir são apresentados alguns dos ambientes mais utilizados.

a) **Tabelas, Figuras e Gráficos**

figure numera e posiciona uma figura ou um desenho e gera uma legenda com o comando \caption, por exemplo:

```
\begin{figure}[H]
\includegraphics[scale=0.1]{barco.eps}
\caption{Um barco.}
\end{figure}
```

picture cria um desenho com comandos próprios do LaTeX;

table numera e posiciona uma tabela e gera uma legenda com o comando \caption;

tabular cria uma tabela de valores ou figuras;

b) **Listas**

enumerate cria uma lista enumerada, cujos ítens são indicados com o comando \item. Necessita do pacote **enumerate**. Exemplo:

```
\begin{enumerate}[1)]
\item primeiro elemento;
\item segundo elemento;
\end{enumerate}
```

1) primeiro elemento;
2) segundo elemento;

itemize cria uma lista não enumerada, cujos ítens são indicados com o comando \item;

```
\begin{itemize}
\item primeiro elemento;
\item segundo elemento;
\end{itemize}
```

- primeiro elemento;
- segundo elemento;

description cria uma lista de descrições, cujos ítens são indicados com o comando \item, seguido da expressão a ser descrita entre colchetes e da descrição própriamente dita;

```
\begin{description}
\item [red] cor vermelha;
\item [blue]cor azul;
\end{description}
```

red cor vermelha;

blue cor azul;

c) **Fórmulas e textos matemáticos**

equation numera e posiciona uma equação. Permite o uso de um identificador de referência, o \label. Exemplo:

```
A Equação~\ref{eq:fx} é
        \begin{equation}
         f(x) = \alpha x + \beta.
        \label{eq:fx}
        \end{equation}
```

A Equação 5.1 é

$$f(x) = \alpha x + \beta. \tag{5.1}$$

O símbolo $ permite escrever uma fórmula inserida em um texto. Exemplo:

```
$\alpha$ é uma letra grega.
```

α é uma letra grega.

array organiza fórmulas matemáticas em linhas e colunas;

eqnarray organiza fórmulas matemáticas em linhas e colunas, numera e pode referenciar cada uma das linhas;

theorems referencia e numera teoremas.

d) **Principais ambientes especiais**

verbatim exibe o texto exatamente como apresentado no arquivo, i.e., não reconhece os comandos e palavras–chave usadas no LaTeX;

theBibliograph cria a bibliografia. Inclui os ítens da bibliografia com o comando
\bibitem[referenciação]{chave de citação}. As referências são citadas no texto pelo comando \cite{chave de citação}. Exemplo: O artigo de \cite{Mats:1997}.

5.3.2 Comandos

A seguir são apresentados alguns dos comandos mais utilizados.

a) **Estilo da página**

\pagestyle{style} define o posicionamento e o estilo da numeração das páginas para todo o documento. O estilo pode ser

empty sem numeração;

plain escreve apenas a numeração inferior;

headings escreve a numeração superior tanto nas páginas pares como nas ímpares;

myheadings define seu próprio estilo de numeração superior;

\thispagestyle{style} define o posicionamento e o estilo da numeração das páginas para a página em questão;

b) **Paragráfos e linhas**

\indent indenta o parágrafo;

\noindent não indenta o parágrafo;

\centering centraliza o parágrafo;

\raggeright mantém o texto do parágrafo à direita;

\raggeleft mantém o texto do parágrafo à esquerda;

\linebreak força a quebra a linha de texto;

\newline força uma nova linha;

\nolinebreak força que não seja formatada uma nova linha;

\newpage força uma nova página;

\nopagebreak força que não seja formatada uma nova página;

\pagebreak força uma nova página se necessário ;

c) **Estilo de fontes**

\textit{ } altera para *itálico* o texto entre os parênteses;

\textbf{ } altera para **negrito** o texto entre os parênteses;

\textsf{ } altera para a fonte sans serif o texto entre os parênteses;

\textsc{ } altera para LETRAS CAIXA ALTA PEQUENAS o texto entre os parênteses;

\textsl{ } altera a *inclinação* do texto entre os parênteses;

\texttt{ } altera para a fonte `Curier` o texto entre os parênteses;

\textrm{ } altera para a fonte `Roman` o texto entre os parênteses;

\underline{ } apresenta o texto `sublinhado`;

d) **Tamanho das fontes**

{\tiny } define um tamanho de fonte ^{muito pequeno};

{\scriptsize } define o tamanho de fonte _{dos índices};

{\footnotesize } define o tamanho de fonte da nota de pé de página;

{\small } define um tamanho de fonte pequeno;

{\normalsize } define um tamanho de fonte normal, que é o padrão;

{\large } define um tamanho de fonte maior que a normal;

{\Large } define um tamanho de fonte grande;

{\LARGE } define um tamanho de fonte maior que a grande;

{\huge } define um tamanho de fonte enorme;

{\Huge } define um tamanho de fonte maior que a enorme;

Esses tamanhos dependem da fonte definida no \documentclass. Por exemplo, \large para 11pt é menor do que o \large para 12pt. Também pode-se utilizar essas palavras chaves como ambientes \begin{}; \end{}, como \begin{large} texto \end{large}.

e) **Caracteres**

_{*texto*} faz com que o texto seja apresentado como subscrito;

$\hat{}${texto} faz com que o texto seja apresentado como superscrito;

\symbol{number}] apresenta o símbolo do número indicado no conjunto de caracteres, por exemplo \symbol{17} é igual a " ;

\ldots apresenta o sinal de reticências . . . ;

% indica que a linha em questão é um comentário e não pertence ao texto a ser apresentado;

f) **Divisões do texto**

\part{título} inicia um nova parte;

\chapter{título} inicia um novo capítulo;

\section{título} inicia uma nova seção;

\subsection{título} inicia uma nova subseção;

\subsubsection{título} inicia uma nova sub subseção;

\appendix inicia um apêndice;

\tableofcontents este comando prepara um sumário automaticamente

\listoffigures este comando prepara uma lista com todos os títulos (\caption) apresentados no ambiente **figure**.

\listoftables este comando prepara uma lista com todos os títulos apresentados no ambiente **table**.

g) **Referenciação**

\label{label-name} nomeia uma certa referência

\ref{label-name} faz a referenciação cruzada com a referência requisitada no label-name;

\pageref{label-name} faz a referenciação cruzada indicando a página da referência requisitada no label-name;

\cite{label-name} faz a referenciação cruzada com a referência bibliográfica requisitada no label-name;

Para referenciação, os autores sugerem a criação de um banco de dados no formato do BibTEX em um arquivo com extensão **.bib**. Esse formato é discutido em muitos textos disponíveis na rede. No **Kile**, o formato do BibTEX encontra-se disponível na opção Bibliography. Por exemplo, caso queira-se referenciar um livro, preencher–se–ia os campos relacionados a seguir:

```
@Book{,
ALTauthor = {},
ALTeditor = {},
title = {},
publisher = {},
year = {},
OPTkey = {},
OPTvolume = {},
OPTnumber = {},
OPTseries = {},
OPTaddress = {},
OPTedition = {},
OPTmonth = {},
OPTnote = {},
OPTannote = {}
}
```

Após o nome do campo, neste caso **Book**, acrescenta-se a palavra–chave que será referenciada no comando **\cite**. O termo **ALT** refere–se a que apenas um dos termos deve ser mantido na referenciação. O termo **OPT** refere-se a campos optativos. Ambos os casos devem ser retirados dos campos preenchidos. Por exemplo, o arquivo referencia.bib conteria o seguinte texto:

```
@Book{Lamport:1994,
author = {Lamport, L.},
title = {\LaTeX: A Document preparation system},
publisher = {{Addison-Wesley}},
year = {1994},
OPTkey = {},
OPTvolume = {},
OPTnumber = {},
OPTseries = {},
OPTaddress = {},
edition = {2},
```

```
OPTmonth = {},
note = {ISBN 0-201-52983-1},
OPTannote = {}
}
```

Depois esse arquivo **referencia.bib** deve ser adicionado ao texto principal (e.g. arquivo **texto.tex**), onde deseja–se que a bibliografia seja introduzida, utilizando o comando **bibliography**. Por exemplo:

```
\documentclass  ...

\begin{document}

.... aqui se escreve  o texto de interesse

\bibliography{./referencia}

\end{document}
```

Para que as citações apareçam na visualização é necessário fazer o seguinte conjunto de procedimentos:

```
latex texto
bibtex texto
latex texto
latex texto
```

REFERÊNCIAS BIBLIOGRÁFICAS

Goossens, M.; Mittelbach, F.; Samarin, A. **The LaTeX companion**. Reading: Addison-Wesley, 1995. 528 p. 11th Printing.

Goualard, F. Manual for the Prosper class, 2001. <http://prosper.sourceforge.net/>. Acesso em 01/09/2002.

Knuth, D. E. **The TeXbook, volume A of computers as typesetting**. Addison-Wesley, 1984. ISBN 0-201-13448-9.

Kopka, H.; Daly, P. **A guide to LaTeX**. 3. ed. Harlow: Addison-Wesley, 1999. 600 p.

Lamport, L. **LaTeX: A document preparation system**. 2. ed. Addison-Wesley, 1994. ISBN 0-201-52983-1.

Oetiker, T.; Partl, H.; Hyna, I.; Schlegl, E. **Introdução ao LaTeX 2$_\varepsilon$**, Agosto 2001. <http://www.ctan.org>, lshortBR.pdf. Acesso em 01/09/2002.

Steding-Jessen, K. **Latex: Uma alternativa mais eficiente comparada aos sistemas WYSIWYG-parte 1: Introdução, vantagens e instalação LaTeX**, 1998. <http://biquinho.furg.br/tex-br/doc/>. Acesso em 01/09/2002.

——. **Latex-demo**, versão 1.13, 2001. <http://biquinho.furg.br/tex-br/doc/>. Acesso em 01/09/2002.

CAPÍTULO 6

GERENCIAMENTO DE CÓDIGO-FONTE UTILIZANDO O RCS

O Sistema de controle de revisão (RCS) é uma ferramenta muito útil no auxílio ao desenvolvimento de programas ou textos LaTeX. Com essa ferramenta é possível manter um histórico da sequência de etapas de construção de um código, de forma organizada e sem repetições desnecessárias. Desta forma, a qualidade do trabalho em desenvolvimento é assegurada e os contratempos decorrentes de dificuldades de administração do desenvolvimento minimizados (GNU, 2002; Petersen, 1998; Loukides e Oram, 1998).

6.1 Estrutura de Organização

O RCS é baseado em uma estrutura de árvore. Dessa forma, a primeira revisão é a raiz da árvore, que receberá a designação de revisão 1.1. As revisões posteriores descendem linearmente na árvore e passam a ter as designações $1.2, 1.3, \cdots$, conhecidas como ramos. É possível também iniciar uma nova seqüência de revisões para cada uma desses ramos estabelecidos, conhecida como galhos (Figura 6.1). Utiliza-se então a ferramenta **rcsmerge** para auxiliar na consolidação de versões a partir de diferentes ramos de desenvolvimentos.

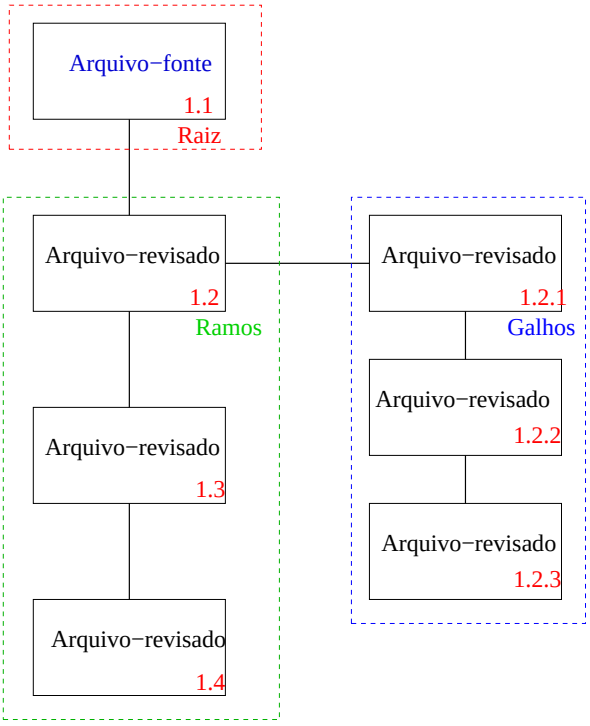

FIGURA 6.1 – Árvore de revisão do RCS.

6.2 Operações básicas

A primeira etapa a ser feita é criar um sub-diretório com o nome **RCS** no diretório onde está o código–fonte. Introduz–se no código fonte desejado uma seqüência de identificação da revisão, sempre entre comentários em relação ao compilador em que este código estará sendo processado.

Os principais caracteres de identificação são: **$Header$, $Author$, $Date$, $Locker$, $Revision$, $Source$** e **$State$**. É possível ainda colocar um identificador no programa executável resultante. Para isso, deve-se inserir no código-fonte

```
char rcsid[]=¨@(#)$Header: /home/margaret/Ermac2002/RCS/RCS.tex,v 1.1 2002/08/13
             11:44:31 margaret Exp margaret $¨;
```

Exemplo de inclusão em um arquivo LaTeX:

```
%$Header: ./RCS/RCS.tex,v 1.1 2002/08/13 11:44:31 margaret Exp margaret $,
%$Author: margaret $
%$Date: 2002/08/13 11:44:31 $
%$Locker: margaret $
%$Revision: 1.1 $
%$Source: ./RCS/RCS.tex,v $
%$State: Exp $


\documentclass[11pt]{report}
....
```

Exemplo de inclusão em um arquivo GNU/Octave:

```
%$Header: ./RCS.tex,v 1.1 2002/08/13 11:44:31 margaret Exp margaret $,
%$Author: margaret $
%$Date: 2002/08/13 11:44:31 $
%$Locker: margaret $
%$Revision: 1.1 $
%$Source: ./RCS/RCS.tex,v $
%$State: Exp $

% Conversao binario - decimal e decimal - binario -> Menu
controle=0;
for ii=1:300,
```

```
controle = menu('Conversao de dados','Decimal-Binario','Binario-Decimal','Exit');
...
```

Exemplo de inclusão em um arquivo GNU/Maxima:

```
¨$Header: ./RCS/RCS.tex,v 1.1 2002/08/13 11:44:31 margaret Exp margaret $¨;
¨$Author: margaret $¨;
¨$Date: 2002/08/13 11:44:31 $¨;
¨$Locker: margaret $¨;
¨$Revision: 1.1 $¨;
¨$Source: ./RCS/RCS.tex,v $¨;
¨$State: Exp $¨;

taylor((x+y)/sqrt(x^2+y^2),[x,0,3],[y,0,3]);
...
```

Os dois comandos fundamentais para a utilização do RCS são **ci** e **co**. O comando **ci** introduz uma nova versão no sub-diretório RCS e o comando **co** retorna a versão mais recente guardada no diretório RCS. Nesse sub-diretório há um arquivo com o nome do programa-fonte original acrescido de uma nova extensão **,v** (vírgula letra v minúscula). Exemplo:

```
$ ci -l nome-do-arquivo
```

que gerará uma revisão mais atualizada do arquivo. Com esse comando, um prompt surgirá a espera uma linha de comentário sobre a revisão que estará sendo guardada. Exemplo:

```
[margaret@pc-lmo28 Ermac2002]$ ci Ermac2002Texto.tex
RCS/Ermac2002Texto.tex,v  <--  Ermac2002Texto.tex
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Material do Ermac/Natal 2002
>> Texto inicial com a inclusão de todos os modos
>> .
initial revision: 1.1
done
```

A fim de utilizar um arquivo que foi colocado no RCS, pode-se extrair, para uso, o arquivo-fonte com o comando:

```
$ co -l  nome-do-arquivo
```

É possível também recuperar uma certa versão de revisão do arquivo-fonte estabelecida pela árvore de revisão, como por exemplo com o comando

```
$ co -r1.2 nome-do-arquivo
```

Os indicadores, após a chamada do primeiro **co**, geram uma série de informações sobre a revisão no código fonte. Por exemplo:

```
%$Header: ./RCS/RCS.tex,v 1.1 2002/08/13 11:44:31 margaret Exp margaret $
%$Author: margaret $
%$Date: 2002/08/13 11:44:31 $
%$Locker: margaret $
%$Revision: 1.1 $
%$Source: ./RCS/RCS.tex,v $
%$State: Exp $
```

Existem várias outras opções no pacote RCS. Opções como, por exemplo, não permitir que uma determinada revisão seja alterada ou definir quem está autorizado a alterar a revisão (**rcs -anome1,nome2,nome3,nome4 nome–de–arquivo**); definir se o estado da revisão é uma revisão experimental (**rcs -s nome–do–arquivo**), por definição uma revisão é sempre experimental, ou verificar o histórico da revisão (**rlog nome–do–arquivo**).

Por exemplo:

```
$ rlog  Ermac2002Texto.tex

RCS file: RCS/Ermac2002Texto.tex,v
Working file: Ermac2002Texto.tex
head: 1.1
branch:
locks: strict
        margaret: 1.1
access list:
symbolic names:
keyword substitution: kv
total revisions: 1;     selected revisions: 1
description:
Material do Ermac/Natal 2002
Texto inicial com a inclusão de todos os modos
----------------------------
revision 1.1    locked by: margaret;
```

```
date: 2002/08/09 16:14:54;  author: margaret;  state: Exp;
Initial revision
=============================================================================
```

REFERÊNCIAS BIBLIOGRÁFICAS

GNU. **RCS manual**, 2002. <http://www.gnu.org>. Acesso em 01/09/2002.

Loukides, M.; Oram, A. **Ferramentas GNU**. O'Reilly, 1998. 147–217 p.

Petersen, R. **Linux - programer's reference**. Berkeley: Mc-Graw-Hill, 1998.

APÊNDICE A

USO DE RECURSOS DE INFORMÁTICA EM INSTITUIÇÕES PÚBLICAS

As novidades tecnológicas têm sido marcantes no estilo de vida da sociedade atual. Um dos aspectos principais dessa evolução acelerada de recursos materiais a ser analisado, apontado por algumas pessoas proeminentes, é a qualidade de vida. Dizem os especialistas que ela deveria ser a questão prioritária nos próximos anos.

Essa preocupação motiva-nos a uma análise de duplo caráter das ferramentas computacionais hoje disponíveis: do ponto de vista institucional e do ponto de vista individual.

No sentido institucional, há a possibilidade, que já pode ser pressentida, de uma dependência forte de uma empresa com respeito a outras e uma fragilização de sua capacidade de atuação, e isso, indiscutivelmente, afeta o seu poder de decisão. Por isso, uma instituição necessita ter mais consciência das ferramentas de trabalho que precisa. Isso exige antecipar (antever) as suas necessidades institucionais no tempo.

Dentro do quesito qualidade de vida para uma instituição, o momento presente requer análise da compra, instalação e uso de produtos e serviços básicos de informática. Essas aquisições devem atender a:

a) aumento ou estabilidade de produtividade da empresa;
b) aumento ou garantia de segurança intrínseca dos serviços da empresa;
c) garantia de comunicação integrada e harmoniosa entre serviços;
d) inserção eficiente da instituição na sociedade já reaparelhada;
e) eliminação de gastos desnecessários para permitir economia pública ou, o que é preferível, reinvestimentos com benefício público.

Assim, algumas perguntas devem ser feitas inteligentemente e respondidas habilmente por todos os membros ativos da comunidade. Hoje a época é outra, iniciativas individuais têm consequências globalizadas. Para as decisões é necessário haver negociação, transparência, responsabilidade, retreinamentos e avaliação de serventia de produtos e serviços. Quanto a questões gerais, pergunta-se:

Que posição deve-se ter perante essa inevitável revolução tecnológica e de mudança de paradigmas? Há de se ter um sistema operacional de preferência (sistema oficial)? Qual o sistema operacional que melhor permite a integração e desempenhos com outros sistemas? Qual a posição a se ter perante programas proprietários comprados e programas

de livre distribuição com licença pública geral? Que critérios de custo/benefício devem ser atendidos na aquisição de produtos com financiamento público?

Quanto a questões específicas, pergunta-se:

Dentre os existentes, qual sistema operacional e programas de escritório permitem aproveitar melhor os treinamentos e reduzir retreinamentos? Dentre os sistemas operacionais existentes, qual sistema é de uso mais duradouro? Algum dos sistemas operacionais força a exclusão de outros sistemas? Que atitude adotar institucionalmente com respeito a suporte?

Do ponto de vista prático, escolhas adequadas de ferramentas de informática representam diferenças de milhões de reais ao longo do tempo. Ressalta-se ainda que a escolha adequada de plataformas de trabalho traduz-se em perspectivas de real produtividade, segurança, eficiência e de colaboração para a formação de competência local.

APÊNDICE B

INTRODUÇÃO AO Prosper

PROSPER é uma classe do LaTeX que permite criar diapositivos (*slide)* com efeitos especiais, exibidos na tela do computador ou impressos em transparências. A criação do PROSPER surgiu da necessidade de se ter diapositivos com a mesma qualidade gráfica do texto produzido em LaTeX e com a mesma facilidade de apresentação do PowerPoint ou StarPresentation. As principais vantagens dessa classe são:

- qualidade gráfica superior;
- resultado final compatível em qualquer plataforma que possua um visualizador de PDF;
- distribuição gratuita;
- opção de remoção de plano de fundo, o que é ideal para imprimir o trabalho de forma econômica em transparência ou papel;
- desenvolvimento em código aberto, o que possibilita o aprimoramento dessa classe.

Neste caso o código fonte do documento **.tex** é escrito como:

```
%\documentclass[opções]{prosper}
\documentclass[pdf,colorBG,slideColor,azure]{prosper}
\usepackage[brazil]{babel}
\usepackage[latin1]{inputenc}
\usepackage{pstricks,pst-node,pst-text,pst-3d}
\usepackage{multicol}
\usepackage{verbatim}
\usepackage{enumerate}
\usepackage{amsmath}
\usepackage{color}          .
          .... outros pacotes de interesse

\begin{document}

 \begin{slide}[Transição]{Título do slide}
        .... texto do slide  .
          .
 \end{slide}
          .
\end{document}
```

São opções definidas no cabeçalho do documento:

draft o conteúdo do diapositivo colocado dentro de molduras;

final o conteúdo colocado numa posição especificada. A nota de roda–pé(*footnote*) pode ser usado como uma macro;

slidecolor o conteúdo do diapositivo pode conter cores;

slideBW determina o estilo de plano de fundo do diapositivo;

total mostra uma legenda contendo o número da página atual seguida do número total de slides (rodapé);

nototal mostra apenas o número da página atual;

nocolorBG deixa o plano de fundo do diapositivo na cor branca;

ps o formato final do arquivo compilado no LaTeX é Postscript;

pdf o formato final do arquivo compilado no LaTeX é PDF;

accumulate define o uso de outras macros;

noaccumulate não permite que outras macros sejam interpretadas no modo PS;

distiller esta opção pode ser usada se o arquivo em Postscript pode ser interpretado dentro de um PDF usando o Adobe Distiller.

Os tipos de transição entre diapositivos disponíveis atualmente no Prosper são:

split o texto varre a tela de forma transversal dando origem a um novo diapositivo;

blinds o texto parece cair de cima para baixo formando um novo diapositivo;

box o texto aparece em forma de caixa de texto no centro do diapositivo;

wipe o texto surgir de forma transversal dando a impressão de movimentar-se de um lado para o outro;

dissolve o texto desfaz-se em pequenos fragmentos dando origem a outro diapositivo;

glitter o texto surge em pequenos pedaços formando um novo diapositivo;

replace o texto aparece como se tomasse o lugar do antigo diapositivo;

Exemplo de um código fonte de um diapositivo:

```
\overlays{2}{%
\begin{slide}[Blinds]{Código Fonte}
\begin{center}
\bf {Este é o Código fonte usando o Prosper no \LaTeX}
\end{center}
\begin{itemstep}
\item{Texto do documento de apresentação}
\end{itemstep}
\end{slide}
}
```

As principais macros do PROSPER são:

\title define o autor da apresentação;

\subtitle define o subtítulo da apresentação;

\author define o nome ou os nomes dos autores;

\email define os emails;

\slidecaption define informações adicionais que aparecem no roda–pé;

\logo(x,y){mylogo} ou \Logo{mylogo} define a posição de uma figura.

\itemstep{ } define as sessões do texto que aparecerá com efeito animado a cada toque do teclado. Essa macro é usada justamente com a macro *overlays* {N} em que {N} é o número de sessões animadas.

\fonttitle{c} {BW } define mudança de cor no título do slide;

\onttext{c} {BW } define a cor da fonte do texto do slide;

\myitem{1v1} {def } define para um determinado ítem o tipo de marcador, que pode ser do tipo 3D;

\myitem{1} {includegraphics[width=.4cm] {red-bullet-on-blue.ps}}, em que *width* define o tamanho do marcador;

\onlyslide{p} {mat} define a amostragem de vários gráficos no mesmo slide sem que para isso seja definido um novo slide.

Para criar novos estilos é necessário ter conhecimento do pacote **Pstricks**, sendo que o nome do novo estilo deve ser antecedido pelo prefixo PPR, por exemplo PPRMeuEstilo.sty.

Para que a apresentação possua animações, é necessário que o arquivo esteja em PDF. Assim, após criar o arquivo postscript, deve-se usar o conversor **ps2pdf**. Exemplo de uso:

```
ps2pdf arq.ps arq.pdf
```

APÊNDICE C

EXEMPLOS DE PROGRAMAS NO GNU/Octave

C.1 Utilização de menu de opções

Os programas a seguir representam uma conversão de números na base binária para a decimal e vice-versa.

C.1.1 Programa Principal: DecBinMain.m

```
% Conversao binario - decimal e decimal - binario -> Menu
controle=0;

for ii=1:300,

  controle = menu('Conversao de dados','Decimal-Binario','Binario-Decimal','Exit');
  if controle == 1
     DecBin;
    elseif controle == 2  BinDec;
        else break;
  if ii==300
     ii=1;
    end
 end
disp(' ');disp(' ');
disp('                                  Tecle <enter> para continuar...');
pause
clc

end
```

C.1.2 Função de conversão Decimal-Binario: DecBin.m

```
clc
clear all
disp('');
disp('Conversao decimal->binario de numeros ponto flutuante');

% Entrada dos dados
n = input('Entre com o numero a ser convertido -> ');
disp('');disp('Aguarde...');

% 1a. Parte : Separacao da parte inteira da parte fracionaria
in =fix(n);
```

```
fn = n - in;

% 2a. Parte : Tratamento da conversao da parte inteira
i=1;
for i=1:1000 % 1000 Valor max do contador
    j= n-i;
    resto= ((in/2) - fix(in/2)) *2;
    if resto == 0
       bin_aux(i)='0';
      else bin_aux(i)='1';
      end
    in=fix(in/2);
    if in==0
       break;
       end
end
imax = i;
% Inversao da decomposicao do resto para binario
for j = 1:i
    bin(j)=bin_aux(i);
    i=i-1;
   end

% 3a. Parte : Tratamento da conversao da parte fracionaria
i=imax;
i=i+1;
bin(i)='.';
for j = 0:30 % 30 numero max de digitos decimais. Evita-se overflow
    i=i+1;
    resto=fix(fn*2);
    if resto==0
       bin(i) = '0';
else bin(i) = '1';
      end
    fn = (fn*2) - resto;
    if fn==0
    break;
    end
end
disp('');resp = sprintf('Conversao de %g em decimal para:',n);
disp(resp);disp(bin);disp('');
disp('em binario.');
disp('                                Tudo  okay.')
```

C.2 Gráficos em diversas janelas diferentes

O progroma a seguir faz a Transformada de Fourier de uma função, apresenta seu espectro de potência e procede a inversa da Transformada de Fourier utilizando apenas parte das freqüências originais (iterativamente).

C.2.1 Programa: fourier.m

```
clc
clear  all

disp(' '); disp('');disp(' ');
disp('Exemplos:');
disp(' Função  e Spectro de Potência');
disp('');disp('');

% (A) Definicao: Numero de pontos e Intervalo de Tempo

dt=1;
N=input('Entre com numero de pontos da funcao ->') %numero total de pontos
disp('Veja o gráfico da função exemplo ...');
disp(' '); disp(' ');   disp(' ');

% (B) Calculo da fx

for i=1:(N)
    x(i) =  ((i-1) * 2 * pi) /N ;
    fx(i)  = sin(x(i));
    end

for i=N/2:(2*N/3)
    x(i) =  ((i-1) * 2 * pi) /N ;
    fx(i)  = sin(x(i))+sin(10000*x(i));
    end

% (C) Calculo da transformada de Fourier

cc = fft(fx);


% (D) Calculo da amplitude media quadrada para spectro
%             e do interv. de Nyquist

c = abs(cc);
c= c*2;
```

```
c= c/N;
for i=1:(N/2+1)
    sp(i)=c(i);
    nyq(i)=(i-1)/(N);
    end

% (E) Visualizacao Grafica

% (1)Definicao do eixos,
%    grafico da funcao gerada

v=[0,pi,-1,1];
axis=v;

figure(1) ;
xlabel('tempo')
ylabel('Variação da função')
title('Função escolhida para teste')
plot(x,fx);

controle=2;
while(controle ~= 1)
controle=1;

% (2)Definicao do eixos,
%    grafico do spectro

v=[0,0.05,0,100];
axis=v;

figure(2) ;
xlabel('frequência')
ylabel('log - espectro ')
title('Espectro de Potencia')
plot(nyq,sp,";Espectro de Potência;");

% (F) Calculo da filtragem e da Transformada inversa de Fourier

fk = input('Entre com o valor da frequência de corte [0,0.5] ->');
fk = (fk/2)*N;
N1    = fk;

fcc=cc;
for i=N1:N
    fcc(i)=0;
```

```
    end

ffx=ifft(fcc);

% (G) Visualizacao Grafica

% (1)Definicao do eixos,
%    grafico da funcao reconstruida

v=[0,pi,-1,1];
axis=v;

figure(3) ;
xlabel('tempo')
ylabel('Variacao da funcao')
title('Funcao escolhida e a reconstruida')
plot(x,fx,";Função teste;",x,ffx,";Função reconstruida;");

pause

controle= input('Se nao deseja continuar digite 1 ->');

disp('Continuando ...')
end % while

disp('')
disp('Fim do Programa')
```

C.3 Curva de Koch

O programa a seguir é utilizado para criar o gráfico da curva fractal de Helg von Koch conhecida como flocos de neve.

Observações:

1) A letra "i "indica a parte imaginária de um número complexo.

2) As funções: a) gset noxtics, b) gset noytics, e c) gset noborder são utilizadas no octave para eliminar os eixos x e y e retirar as bordas da figura. No Matlab utilizar–se–ia a função "axis off".

C.3.1 Programa: koch.m

```
clear all
dp= (sqrt(3)/2)*i +0.5;
dn= (sqrt(3)/2)*i -0.5;
```

```
SnowFlakes=[dp,dn,0,dp];

plot(SnowFlakes);

pause(0.5);

for k=1:8;
        SnowFlakesOld=SnowFlakes;
        [m,n]= size(SnowFlakesOld);
        n=n-1; %Existe n-1 lados da Figura do floco de neve
        for j=0:n-1
                New= (SnowFlakesOld(j+2)- SnowFlakesOld(j+1))/3;
                SnowFlakes(4*j+1)= SnowFlakesOld(j+1);
                SnowFlakes(4*j+2)= SnowFlakesOld(j+1)+New;
                SnowFlakes(4*j+3)= SnowFlakes(4*j+2)+ New *((1-sqrt(3)*i)/2);
                SnowFlakes(4*j+4)= SnowFlakesOld(j+1) + 2* New;
                end;
        SnowFlakes(4*n+1)= SnowFlakesOld(n+1); %últimos elementos são idênticos

        gset noxtics;
        gset noytics;
        gset noborder;
        plot(SnowFlakes);
        pause(0.5);
        end;
```