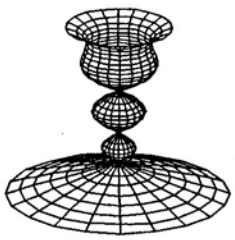
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA
CADERNOS DE MATEMÁTICA E ESTATÍSTICA
SÉRIE B: TRABALHO DE APOIO DIDÁTICO

INTRODUÇÃO AO MATLAB PARA WINDOWS

ELBA BRAVO, AUSBERTO CASTRO,
JULIO RUIZ CLAEYSSEN,
RUDNEI DIAS DA CUNHA,
MARIA PAULA GONÇALVES FACHIN

SÉRIE B, No. 31
PORTO ALEGRE, NOVEMBRO 1995

INTRODUÇÃO AO MATLAB PARA WINDOWS



E. Bravo, A. Castro,
J. Claeyssen, R. da Cunha,
M. Fachin

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PORTO ALEGRE

Índice

Prefácio

Este texto foi escrito com um objetivo essencialmente didático: proporcionar ao leitor um material básico que facilite o uso do MATLAB em microcomputador, bem como a interação deste software com processadores de texto, pacotes gráficos e com a rede Internet. A motivação deste texto teve sua origem a partir de disciplinas ministradas pelos autores na UFRGS.

O MATLAB possui como estrutura básica de dados a forma matricial. Os comandos são introduzidos gradualmente e uma atenção particular é dada aos arquivos `.m`, que permitem incorporar programas ao MATLAB como simples comandos funcionais. Os gráficos, gerados pelo MATLAB, podem ser incluídos em um documento criado num processador de texto como o Word ou o Latex. A interação do MATLAB com linguagens tais como FORTRAN e C, é estabelecida através de arquivos específicos.

Consideramos este texto como um complemento aos manuais de referência do MATLAB e não como um substituto aos mesmos. O usuário deverá consultar a documentação fornecida pela The MathWorks, Inc. sempre que for necessário.

Agradecemos o apoio recebido das nossas unidades da UFRGS: PROMEC, Instituto de Informática e Instituto de Matemática.

Os autores

Capítulo 1

O Software Matricial MATLAB

1.1 Introdução

MATLAB é um programa interativo para a computação numérica e a visualização de dados, desenvolvido por Cleve Moler e Associados. É usado como um importante laboratório para cálculos com matrizes. Além disso, proporciona um fácil acesso aos softwares matriciais avançados como LINPACK ou LAPACK e também na importação/exportação de dados com programas escritos em linguagens científicas como FORTRAN ou C. O MATLAB é um software numérico que utiliza uma variedade de notações convencionais especiais, porém, muito semelhante ao padrão matemático do papel e caneta.

Para resolver problemas de aplicações especificas, MATLAB faz uso dos conhecidos *toolboxes*: conjunto de funções MATLAB (arquivos com extensão .m) que estendem o ambiente MATLAB para resolver problemas em áreas específicas como processamento de sinais, simulação de sistemas de controle, redes neurais, otimização, etc.

O uso do MATLAB presupoõe familiaridade com a álgebra matricial ou que esteja incorporado a uma disciplina elementar de matrizes e vetores ou de métodos numéricos.

Existem duas versões do MATLAB para o ambiente operacional Win-

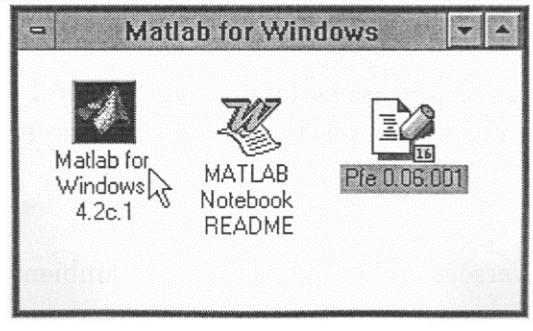
dows:

- *Profissional 4.2c*, sem restrições ou limites (exceto de memória RAM disponível e de disco rígido)
  - Mínimo de memória RAM exigido 8 Mbytes.
  - É permitido a ligação dinâmica a subrotinas FORTRAN ou C.
  - É obrigatório o uso de coprocessador matemático.
  - Permitido o uso de qualquer toolbox disponível.
- *The Student Edition 4.0* com algumas limitações:
  - O tamanho das matrizes está limitado a 8192 elementos onde o maior número de filas ou colunas deve ser 32 (máximos 32x256 ou 256x32).
  - Não é permitido a ligação dinâmica a subrotinas FORTRAN ou C.
  - Os únicos toolboxs permitidos são: *Symbolic Math* e *Signals and Systems*, que já estão embutidos.

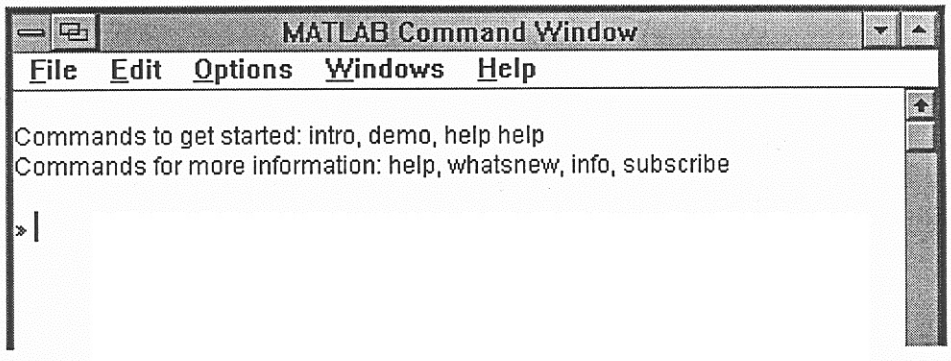
Para maiores detalhes, veja-se o MATLAB *User's Guide* publicado pela The MathWorks Inc., ou *The Student Edition of* MATLAB *User's Guide* publicado pela Prentice-Hall, Inc.

1.2 Entrando no MATLAB

Uma sessão no MATLAB no ambiente Windows pode-se inicializar selecionando com o mouse o ícone MATLAB e clicando neste duas vezes:



Deverá aparecer a janela de comandos do MATLAB:



Com esta janela a sessão esta inicializada. Na primeira vez, é recomendável executar os comandos

```
>> demo
```

ou

```
>> expo
```

e seguir as instruções.

O término de uma sessão do MATLAB é realizado com o comando `exit`.

Capítulo 2

Matrizes

2.1 Entrada de Matrizes

As matrizes podem ser introduzidas no MATLAB de diversas maneiras:

- Inseridas através de uma lista explícita de elementos;
- Geradas por comandos ou funções incorporadas ao MATLAB;
- Criadas em arquivos tipo M (*.m);
- Carregadas por meio de arquivos de dados externos.

Por enquanto, serão consideradas somente as duas primeiras formas. A declaração

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

seguida de um `<Enter>` ou `<Return>`, resulta na saída

```
>> A =
       1  2  3
       4  5  6
       7  8  9
>>
```

e a matriz A é armazenada para uso futuro. O ponto e vírgula (;) indica o final de cada linha. Para matrizes grandes, utiliza-se o <Return> em substituição ao ponto e vírgula.

Digitando-se a declaração

```
>> B = A'
```

obtém-se como resposta a matriz transposta da matriz A

```
>> B =
        1 4 7
        2 5 8
        3 6 9
```

O caracter ' denota sempre a transposta. Assim, a declaração

```
>> u = [-1 2 0 3]'
```

produz

```
>> u =
        -1
         2
         0
         3
```

Se α for o número 4/3 ou π ou raiz de 2, o MATLAB fornecerá como resultado não o valor exato, mas uma boa aproximação numérica. Quando não é utilizado um formato especial, o resultado será dado no formato curto que mostra cerca de cinco dígitos decimais significativos. Outros formatos mostram mais dígitos ou usam notação científica. Por exemplo, suponha-se que se tenha digitado

```
>> x = [4/3    1.2345E-6]
```

então, os formatos e o resultado, para este vetor, são:

- format short[1]

```
1.3333    0.0000
```

- format short *e*

```
1.3333E+000    1.2345E-006
```

- format long

```
1.333333333333338    0.000001234500000
```

e, similarmente, com o format `long e` .

O MATLAB possui recursos de sub-índices, que facilitam o trabalho com linhas, colunas, elementos ou submatrizes. Por exemplo, digitando $A(3,2)$ tem-se como resultado o elemento A_{32} da matriz `A` que no exemplo anterior é 8. *O uso de dois pontos(:) é um recurso importante no MATLAB: permite gerar vetores ou extrair submatrizes de uma matriz.* Por exemplo,

```
>> u = 1:5
```

gera um **vetor linha** cujos elementos são os números de 1 a 5, com incremento unitário. Tem-se o resultado

```
>> u =
        1 2 3 4 5
```

Outros incrementos podem ser utilizados. No MATLAB a palavra "pi" representa o número $\pi = 3.1415....$ Assim,

```
>> v = 0: pi/4: pi
```

Também

```
>> z = [1:3 4:2:8 10:0.5:11]
```

gera

```
>> z =
        [1 2 3 4 6 8 10 10.5 11]
```

[1] 1.2345E-006 = 0.0000012345

No MATLAB, arranjos individuais de elementos são acessados utilizando sub-índices, por exemplo `z(3)` é o terceiro elemento em `z`. Para acessar um bloco de elementos, o MATLAB utiliza os dois pontos (:). Assim,`z(1:4)` significa [1 2 3 4], entretanto, `z(4:-1:1) = [4 3 2 1]`, isto é, os quatro primeiros elementos de `z` em ordem inversa.

Numa matriz A, a declaração

```
>> A(:,3)
```

fornece os elementos da terceira coluna de `A`, entretanto

```
>> A(3,:)
```

resulta a terceira linha. Para extrair a submatriz formada pelas primeiras duas linhas e colunas de `A`, digita-se

```
>> A(1:2,1:2)
```

Em geral, se `v` e `w` são vetores com elementos inteiros, então

```
>> C(v,w)
```

é a matriz obtida tomando os elementos de `C` cujos índices de linhas correspondem aos elementos de `v` e os índices das colunas, aos de `w`. Por exemplo, se `C` é uma matriz de ordem 1O, então

```
>> C(1:5,7:10)
```

especifica a submatriz 5×4 formada pelos elementos das cinco primeiras linhas e das quatro últimas colunas de `C`. Por exemplo, entrar a matriz

$$\mathtt{A} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$$

como

```
>> A=[1:5;6:10;11:15;16:20]
```

e observar a saida dos seguintes comandos:

```
>> A(1,:)
>> A(2,:)
>> A(3,:)
>> A(5,:)
>> A(:,1)
>> A(:,3)
>> A(:,5)
>> A(1,1:2:5)
>> A([2 4],:)
>> A([4 2],:)
>> A(4:-1:1, 5:-1:1)
>> A([1 1],[2 2])
>> A([1 1],[4 5])
>> A([2 2],[5 4])
>> A([3 2],[5 4])
```

2.2 Operações e Gráficos Matriciais

As operações com números, da mesma forma que com calculadoras ou linguagens de programação, são definidas com os símbolos usuais: `+`, `-`, `*`, `/`, `^`. O MATLAB trabalha com números complexos z = a+ib e realiza operações sem nenhum manuseio especial. Por exemplo

```
>> z=1-2*i
>> w=3*(2-sqrt(-1)*3)
>> u=sqrt(-2)
>> (z+w)/u= -7.7782 -4.949*i
```

Pode ser conferido que

```
>>i^2
```

tem como saida

```
>>-1.0000 + 0.0000i
```

ou seja, i é a raiz quadrada de -1. A notação j para a unidade imaginária, utilizada em elétrica, é tambem incorporada pelo MATLAB. A forma polar de um nmero complexo

$$z = a + bi = Me^{i\theta}$$
$$M = \sqrt{a^2 + b^2}$$
$$\theta = tan^{-1}(b/a)$$
$$a = Mcos\theta$$
$$b = Msen\theta$$

é descrita pelo MATLAB utilizando os comandos **real** (parte real), **imag** (parte imaginária), **abs** (valor absoluto), **angle** (ángulo). Por exemplo,

```
>> z= 1-2i
>>magz=abs(z)
  magz=-1.1071
>>anglez=angle(z)
  anglez = -1.1071
>>deangz=anglez*180/pi
  anglez=-63.4349
>> reaalz=re(z)
   realz=1
>> imagz=imag(z)
   imagz=-2
```

Observe que o MATLAB calcula o angulo em radianos. O complexo conjugado de z=a+bi, definido como sendo $\overline{z} = a - bi$ é simplesmente z'.

No caso em que os elementos de uma matriz Z sejam números complexos, Z' é sua complexa conjugada: $(z'_{ij} = \overline{z}_{ji})$. Para obter a transposta simples de Z tem-se os comandos Z' ou **conj**(Z'). .

Se A e B são matrizes , então a sua soma e o seu produto são as matrizes

```
>> C = A + B
e
>> D = A*B
```

respectivamente. Certamente, as dimensões de A e B devem ser apropriadas para essas operações. De outro modo, uma mensagem de erro é emitida.

A multiplicação de um escalar b por uma matriz A é

```
>> C = b*A
```

A solução da equação $\mathbf{Ax} = \mathbf{b}$ com $det(\mathbf{A}) \neq 0$, pode ser obtida com os comandos:

```
>> x=A/b
ou
>> x=inv(A)*b
```

onde `inv(A)` é o comando para a inversa de uma matriz não-singular. Este procedimento é menos eficiente do que a eliminação Gaussiana, isto é, esses comandos requerem um número maior de operações. No MATLAB, o comando

```
>> det(A)
```

calcula de forma aproximada, o determinante de uma matriz quadrada cujos elementos são números.

Se y é um vetor linha, então o comando

```
>> plot(y)
```

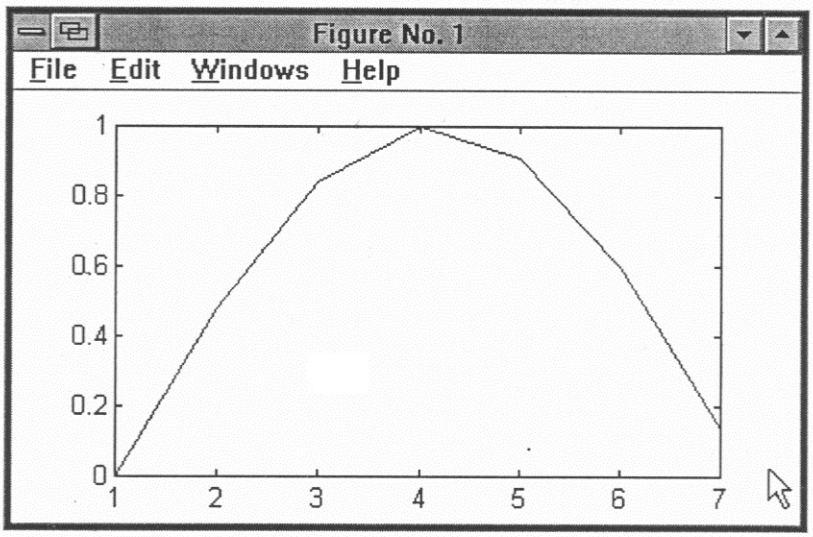
produz um gráfico linear dos elementos versus índices. Por exemplo, suponha-se que se deseja obter um gráfico com os números

```
{0.   .48   .84   1.   .91   .6   .14}.
```

Isto é realizado com os comandos

```
>> y = [.0   .48   .84   1.   .91   .6   .14];
   plot(y)
```

que resultam no gráfico na tela:



O *ponto e vírgula* ";" é utilizado para separar declarações, bem como para que estas declarações não sejam interpretadas (executadas) imediatamente pelo MATLAB. Certamente, se `y` é um vetor coluna, então o comando `plot(y')` também exibe um gráfico com os elementos do vetor `y`.

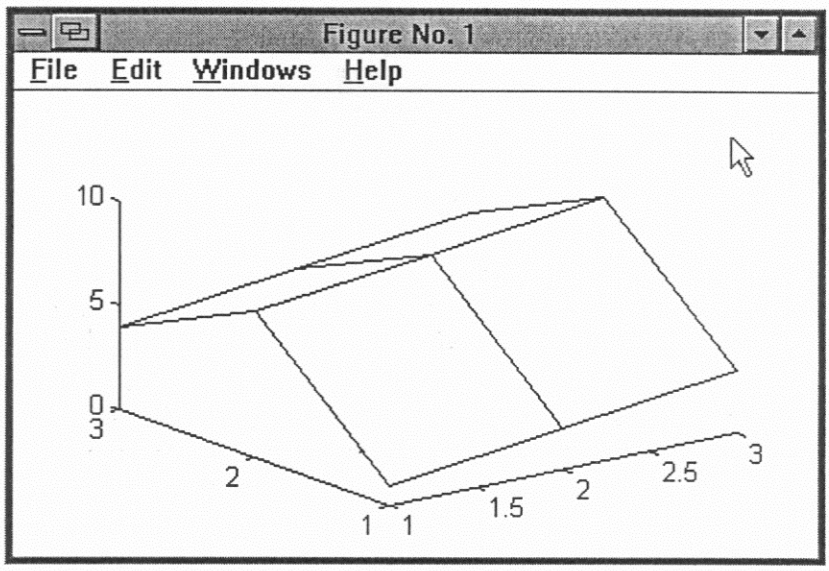
O MATLAB também fornece recursos para nomear os eixos dos gráficos ou colocar o gráfico em grade:

```
title('grafico vetorial')
xlabel('horas')
ylabel('temperatura do dia')
grid
```

O comando `mesh(A)` cria um gráfico tridimensional em perspectiva com os elementos da matriz `A`. O gráfico é formado pela conexão, através de linhas, entre os valores dos elementos que correspondem a pontos adjacentes, numa grade no plano $i - j$. Por exemplo,

```
>> A=[ 1 2 3; 7 8 9; 4 5 6];
>> mesh(A);
>> title('grafico de uma matriz')
```

produz o seguinte gráfico:



O comando `quiver(X,Y,DX,DY)` (ver Fig. 2.1) desenha vetores bidimensionais sentido geométrico ou físico (setas) em cada ponto formado por pares de elementos nas matrizes `X` e `Y`. Os pares de elementos nas matrizes `DX` e `DY` determinam a direção e grandeza relativa das setas.

Exemplo 2.2.0 Desenhar o campo gradiente da função $z = xe^{-(x^2+y^2)}$.

```
X = -1:0.2:2;
Y = X;
[x,y]=meshgrid(X,Y);
z= x.*exp(-x.^2-y.^2);
[dx,dy]= gradient(z,0.2,0.2);
contour(x,y,z);
hold on
quiver(x,y,dx,dy);
stitle('Uso dos comandos {\bold contour} e {\bold quiver}');
hold off
```

O comando `[x,y] = meshgrid(X,Y)` cria uma matriz `x` cujas linhas são cópias do vetor `X`, e uma matriz `y` cujas colunas são cópias do vetor `Y`.

O comando `[dx,dy] = gradient(z,dx,dy)` divide a diferença horizontal pelo escalar $dx = 0.2$ (para calcular $\frac{\delta z}{\delta x}$) e similarmente com a vertical. O comando `contour(x,y,z)` desenha as curvas de nivel `z = c` com os valores da função `z` no arranjo matricial que resulta de calcular `z` nos pontos x, y

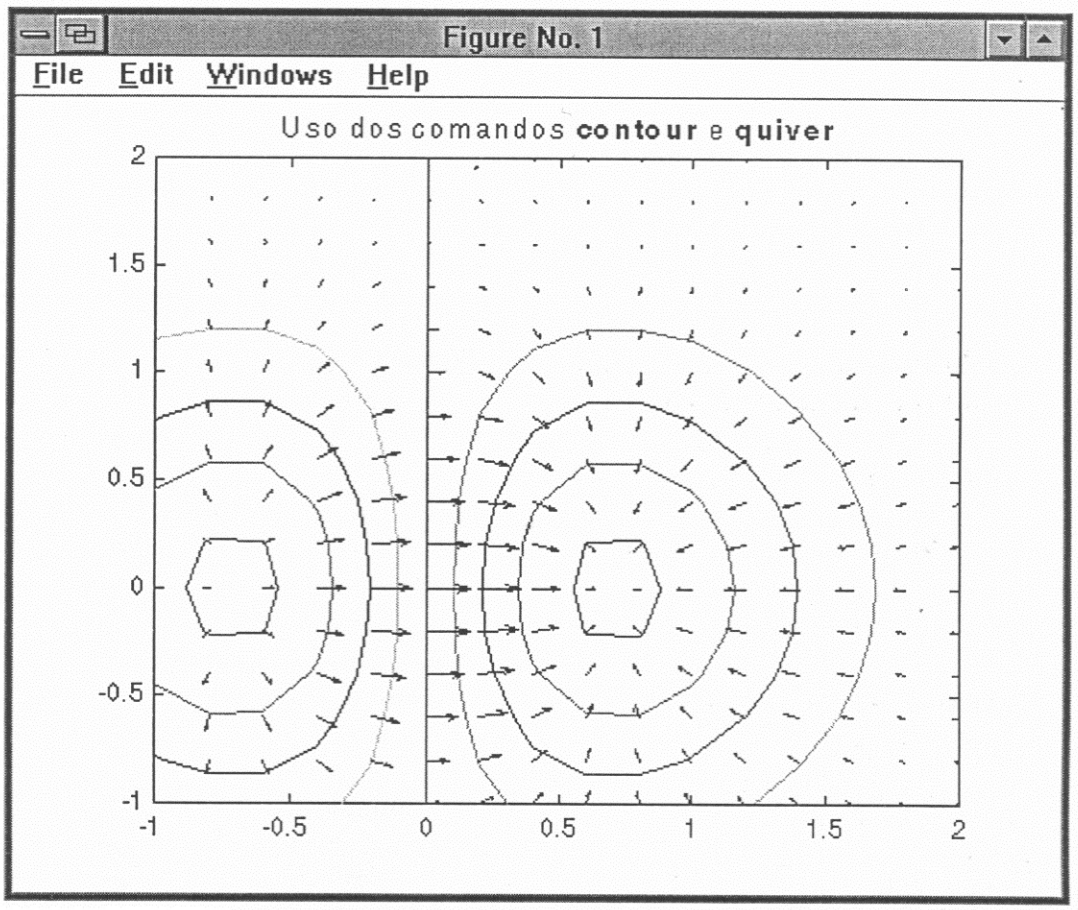


Figura 2.1: Comandos 'contour' e 'quiver'

da grade gerada. Finalmente, o comando **quiver** desenha em cada ponto da grade o vetor gradiente de `z` no ponto. O comando `hold` segura o gráfico na tela.

2.3 Fatorações Matriciais

A fatoração **L U** de uma matriz `A` é obtida através do comando

```
>> [L,U] = lu(A)
```

que fornece as matrizes triangulares da fatoração. Outras variáveis podem ser utilizadas em vez de **L** ou **U**. Deve ser observado que o programa por trás desse comando, utiliza pivotamento. Daí que a resposta contém matrizes da forma $\mathbf{P'L}$ em vez de **L** ($\mathbf{P'}$ matriz de permutação) e a matriz triangular superior **U**, de modo que $\mathbf{LU} = \mathbf{PA}$ ou $\mathbf{P'LU} = \mathbf{A}$.

Para uma matriz simétrica positiva definida **A**, o comando

```
>> chol(A)
```

fornece a fatoração de Cholesky $\mathbf{A} = \mathbf{LL'}$, isto é, a matriz `L`.

O posto de uma matriz **A** é obtido com o comando `rank(A)`. No MATLAB um vetor pode ser considerado como uma coluna ou linha com n elementos. O comando

```
dot(u,v)
```

calcula o *produto interno* de dois vetores u e v da mesma dimensão. Se u e v são vetores coluna, então os produtos matriciais $\mathbf{u}' * \mathbf{v}$ ou $\mathbf{v}' * \mathbf{u}$ são equivalentes ao produto interno de u e v. O MATLAB possui o comando

```
>> orth(V)
```

que ortonormaliza as colunas da matriz **V** através do processo de Gram-Schmidt.

A fatoração **QR** é utilizada freqüentemente no MATLAB. Escrevendo

```
>> [Q,R] = qr(A)
```

produz-se uma matriz triangular superior **R** do mesmo tamanho de **A** e uma matriz ortogonal **Q**, tal que $\mathbf{Q} * \mathbf{R} = \mathbf{A}$.

2.4 Normas Matriciais

No MATLAB, algumas normas de vetores e matrizes podem ser calculadas utilizando o comando `norm`. Assim,

```
>> norm(x)
```

fornece o valor $\sqrt{x'x}$ da norma euclidiana do vetor x. A notação $\mathbf{norm(x,1)}$ denota a norma $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$ e $\mathbf{norm(x,inf)}$ a norma $\|x\|_\infty = max|x_i|$. Similarmente,
`norm(A)` ou `norm(A,2)`, `norm(A,1)`, `norm(A,inf)`
são comandos para a norma euclidiana da matriz **A**, a norma $\|\mathbf{A}\|_1$ e $\|\mathbf{A}\|_\infty$, respectivamente.

2.5 Autovalores e Autovetores

No MATLAB pode-se determinar o polinômio característico de uma matriz quadrada **A** de ordem n utilizando o comando `poly(A)`. A saída de $poly(\mathbf{A})$ é um vetor linha de $n+1$ elementos $c(1)$, $c(2)$, $\cdots$, $c(n+1)$, os quais são os coeficientes do polinomio

$$det(\alpha\mathbf{I} - \mathbf{A}) = c(1)\alpha^n + c(2)\alpha^{n-1} + \cdots + c(n)\alpha + c(n+1) \ .$$

Para determinar todas as raízes do polinômio característico utiliza-se o comando `roots(c)` , onde **c** é o vetor formado pelos coeficientes $c(k)$.

Se $\mathbf{A}$ é uma matriz $n \times n$, o comando `eig(A)` apresenta um vetor, cujos elementos são os autovalores de $\mathbf{A}$ ordenados por sua grandeza. Para obter os correspondentes autovetores de $\mathbf{A}$ usa-se o comando

```
>> [V,D] = eig(A)
```

o qual fornece os autovetores de $\mathbf{A}$, como colunas de $\mathbf{V}$, e os correspondentes autovalores, na matriz diagonal $\mathbf{D}$.

Os valores singulares de uma matriz $\mathbf{A}$, as raízes quadradas dos autovalores positivos da matriz $\mathbf{A}^t * \mathbf{A}$, são obtidos com o comando

```
>> S = svd(A)
```

Escrevendo

```
>> [U,S,V] = svd(A)
```

obtem-se a decomposição em valores singulares de uma matriz $\mathbf{A}$ de ordem $m \times n$. Aqui $\mathbf{U}$ é uma matriz ortogonal $m \times m$, $\mathbf{D}$ uma matriz diagonal $m \times n$ contendo os valores singulares como elementos diagonais e $\mathbf{V}$ uma matriz $n \times m$ satisfazendo $\mathbf{A} = \mathbf{UDV}^t$.

O MATLAB calcula a pseudoinversa $\mathbf{A}^+$ de uma matriz $\mathbf{A}$ utilizando o comando

```
>> pinv(A)
```

Isto permite resolver, por mínimos quadrados, a equação $\mathbf{Ax} = \mathbf{b}$ com o comando

```
>> xlsq = pinv(A) * b
```

2.6 Geração de Matrizes Padrão

O MATLAB possui um variado número de funções que possibilitam gerar matrizes de uso freqüente. Por exemplo,

- $\mathbf{A}$ = eye(m,n) $\Rightarrow$ $\mathbf{A}$ $m \times n$ com uns na diagonal e zeros fora dela;

- $\mathbf{A}$ = zeros(m,n) $\Rightarrow$ $\mathbf{A}$ $m \times n$ com todos os elementos zero;

- $\mathbf{A}$ = ones(m,n) $\Rightarrow$ $\mathbf{A}$ $m \times n$ com todos os elementos iguais a um.

Para matrizes quadradas, somente um argumento é necessário

```
>> A = eye(n)
>> A = zeros(n)
>> A = ones(n)
```

A construção de matrizes, que possuem estrutura, é muito simplificada com o uso do MATLAB. Uma matriz diagonal $\mathbf{D}$ é descrita pelo comando

```
>> D = diag(d)
```

onde $\mathbf{d}$ é o vetor linha cujos elementos são os da diagonal da matriz $\mathbf{D}$. O comando

```
>> D = diag(d,k)
```

onde $\mathbf{d}$ é um vetor com n elementos e k um inteiro, gera uma matriz de ordem $n + s$, sendo s o valor absoluto de k e com os elementos de $\mathbf{d}$ na k-ésima diagonal: $k = 0$ é a diagonal principal; $k > 0$ é acima da diagonal principal e $k < 0$ é abaixo da referida diagonal.

Por exemplo, a matriz tridiagonal

$$\mathbf{B} = \begin{bmatrix} 4 & 0 & -1 & 0 & 0 \\ 0 & 4 & 0 & -1 & 0 \\ -1 & 0 & 4 & 0 & -1 \\ 0 & -1 & 0 & 4 & 0 \\ 0 & 0 & -1 & 0 & 4 \end{bmatrix}$$

pode ser escrita no MATLAB como

```
>> B = 4*eye(5) - diag(ones(1,3),2) - diag(ones(1,3),-2)
```

Exemplo 2.6.0
Exibir graficamente o vetor **u**, obtido da função $sen(2\pi x/3)$, para valores de x entre 0.1 e 2 com incremento 0.1.

Solução
Sendo que **u** é um vetor com $n = 20$ elementos, no MATLAB, procede-se da seguinte maneira:

- cria-se o vetor

```
>> u = zeros(20,1)
```

  em que todos seus elementos são zeros;

- substitui-se os elementos de **u** pelos da função dada

$$u(s) = sin(2 * pi * s * 0.1/3),$$

  fazendo s variar de 1 a 20;

- grafica-se o vetor **u**.

O programa, para tanto, é o seguinte:

```
>> n=20;
>> u = zeros(n,1);
>> for s=1:n;
   u(s)=sin(2*pi*0.1*s/3);
   end;
>> plot(u')
```

Observe-se a semelhança entre o comando do MATLAB e o comando "for" do BASIC.

2.7 Diversos

- O comando `help` é o caminho mais simples para obter auxílio sobre o tópico de interesse. Por exemplo, escrevendo-se

```
>> help sqrt
```

  obtense como resposta:

```
SQRT square root.
     SQRT(X) is the square root of the elements of X. Complex
     results are produced if X is not positive.
```

- O tempo de processamento utilizado pelo MATLAB para executar um processo é dado pelo comando `cputime`. Assim

```
>> t = cputime;
>> inv(A);
>> e = cputime - t;
```

  retorna o tempo gasto para invertir uma determinada matriz A. Outros comandos similares são **clock**, **etime**, **tic**, **toc**.

- A exatidão relativa da aritmética de ponto flutuante é descrita pelo"epsilon da máquina", isto é, uma variable permanente `eps` cujo valor é inicialmente a distância de 1.0 até o próximo maior número de ponto flutuante, isto é, o menor número que adicionado com 1.0 produz no computador um número de ponto flutuante maior do que 1. Em máquinas com aritmética IEEE, $eps = 2^{(-52)}$, o qual é a grosso modo $2.22e-16$.

- O comando

```
>> x = linspace(x1,x2,n)
```

  gera um vetor `x` de n pontos linearmente espaciados entre `x1` e `x2`. Por exemplo,

```
>> x = linspace(1,22,11)
```

  gera o vetor

```
x =
Columns 1 to 11
2 4 6 8 10 12 14 16 18 20 22
```

Capítulo 3

Operadores e Controle de Fluxo

3.1 Operadores Relacionais e Lógicos

Além das operações matemáticas tradicionais (+,-,*,/), o MATLAB suporta operações relacionais e lógicas. O objetivo destes operadores é responder a questões do tipo Verdadeiro/Falso e também controlar o fluxo ou ordem de execução de um conjunto de comandos MATLAB.

Como entradas para todas as expressões relacionais e lógicas, o MATLAB considera qualquer número diferente de zero como Verdadeiro, e qualquer zero como Falso. A saida de todas as expressões relacionais e lógicas produzem 1 para Verdadeiro e 0 para Falso.

Operador	*Significado*
<	menor que
<=	menor ou igual que
>	maior que
>=	maior ou igual que
==	igual a
~=	não igual a
&	AND lógico
|	OR lógico
~	NOT lógico

- C = A & B é a matriz com elementos 1's onde A e B tem elementos diferentes de zero, e com 0's onde A ou B tem um zero.

- C = A | B é matriz com elementos 1's onde A ou B tem um elemento diferente de zero, e com 0's onde A e B tem zeros.
- C = ~ A é a matriz com 1's onde A tem um zero, e com 0's onde A tem elementos diferentes de zero.

Exemplo 3.1.0
Se

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 5 \\ 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 0 & 8 \\ 0 & 1 & 8 \\ 3 & 3 & 0 \end{bmatrix}$$

então

$$A\&B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad A \mid B = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Além dos operadores acima, o MATLAB tem pré-definidas as seguintes funções relacionais e lógicas: `any`, `all`, `find`, `exist`, `isnan`, `isinf`, `finite`, `isempty`, `isstr`, `isglobal` e `issparse`

3.2 Controle de Fluxo

Igual às principais linguagens de programação cientifica, o MATLAB oferece mecanismos que permitem controlar o fluxo de execução de comandos baseados em estruturas de decisão. Estas são: o laço `for`, o laço `while` e o condicional `if-else-end`. Visto que, estas estruturas incorporam mais de um comando MATLAB, é recomendável que sejam escritas em arquivos de texto `.m` antes de serem escritas diretamente do teclado na janela de comandos do MATLAB, no prompt `>>` .

3.2.1 FOR

O laço `for` permite executar um grupo de comandos repetidamente um número fixo e pré-determinado de vezes. A forma geral desta estrutura é:

for k = $k_{inicial}$: inc : k_{final}
conjunto de comandos
end

Se o incremento *inc* é 1, então a forma geral é:

for k = $k_{inicial}$: k_{final}
conjunto de comandos
end

Exemplo 3.2.0
Gerar uma tabela de valores de função para $y = f(x) = 1 + sin(\pi x)$ sobre o intervalo [-2, 2] em passos de 0.25.

O programa MATLAB correspondente será:

```
for k = -2 : 0.25 : 2
    y(k) = 1 + sin(pi*k) ;
end
```

Exemplo 3.2.1
O "laço-for", pode ser utilizado para gerar matrizes. Se **d** é um vetor $n \times 1$ e **e** é um vetor $(n-1) \times 1$, então as instruções

```
B = zeros(n);
for i = 1:n-1
    B(i,i) = d(i);
    B(i,i+1)= e(i);
end;
B(n,n) = d(n)
```

geram uma matriz **bidiagonal superior** de ordem $n \times n$:

$$\begin{bmatrix} d(1) & e(1) & 0 & 0 & \cdots & 0 \\ 0 & d(2) & e(2) & 0 & \cdots & 0 \\ 0 & 0 & d(3) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & d(n-1) & e(n-1) \\ 0 & 0 & 0 & 0 & 0 & d(n) \end{bmatrix}.$$

3.2.2 WHILE

Em oposição à estrutura `for`, o comando `while` executa um grupo de comandos um número indeterminado de vezes (até uma condição lógica ser satisfeita). A forma geral é:

while expressão-lógica grupo de comandos **end**

Exemplo 3.2.2

```
dif = 1;
while   dif > 0.0005
   x1 = x2 - cos(x2) / (1 + x2) ;
   dif = abs(x2 - x1);
   x2 = x1;
end;
```

Exemplo 3.2.3
Encontrar uma matriz qualquer 2×2 gerada randomicamente com valores próprios menores que 1. Resposta:

```
A = rand(2);
while  max( abs( eig(A))) >= 1
    A = rand(2);
end;
```

3.2.3 IF

Seqüencias de comandos MATLAB condicionalmente avaliados são executados usando o comando `if`. A forma mais simples é:

if expressão-lógica conjunto de comandos **end**

A forma mais geral (com três ou mais alternativas) é:

```
if expressão-lógica-1
    conjunto de comandos 1
    elseif expressão-lógica-2
       conjunto de comandos-2
       . . . . . . . . . . . . . . . . .
            elseif expressão-lógica-(N-1)
               conjunto de comandos N-1
            else expressão-lógica-N
               conjunto de comandos N
end
```

Exemplo 3.2.4

```
j=1;
while j <= 4
   if  2^j == n
     j = j - 1;
   else
     j = j + 1;
   end;
end;
```

Exemplo 3.2.5
Para uma matriz ortogonal **P** calcular $\mathbf{y} = \mathbf{Px}$ se $\det(\mathbf{P}) = 1$, caso contrário calcular $\mathbf{y} = \mathbf{P'x}$

```
if P'*P == eye(P)
   if det(P) ==1
      y = P*x ;
   else
      y = P'*x;
   end;
end;
```

3.2.4 BREAK

O comando `break` termina a execução de laços `for` e `while`.

Exemplo 3.2.6
O seguinte segmento de programa MATLAB calcula todos os números de Fibonnaci menores que 1000:

```
Fibonnaci(1) = 1;
Fibonnaci(2) = 1;
for k = 3:1000                          % laco FOR
  x = Fibonnaci(k-1) + Fibonnaci(k-2);
  if x >= 1000
    break;                              % termina o laco FOR
  end
  Fibonnaci(k) = x;
end;
```

Programadores MATLAB com alguma experiência deverão minimizar o uso do comando **break**, pois a sua presença num programa relativamente grande, torna muito difíl a sua leitura e depuração.

Capítulo 4

Arquivos MATLAB

4.1 Arquivos de Comandos

Geralmente o MATLAB trabalha em *modo imediato,* isto é, quando escrevemos uma instrução na janela de comandos e pressionamos a tecla [ENTER], imediatamente é mostrado na tela o resultado. Porém, quando um conjunto de comandos é necessário ser usado muitas vezes, é mais eficiente escrever este conjunto em um *arquivo .m* (chamado também M–arquivo) e executá–lo cada vez que seja necessário. Os arquivos de comandos usam todas as variáveis globais que nesse momento estão definidas na memória (workspace).

Arquivos de comandos são essencialmente *programas*: sequências de comandos que realizam tarefas específicas. Usando técnicas de programação estruturada, os programas MATLAB são de tamanho relativamente pequeno. Arquivos de comandos são conhecidos também como *programas diretos.*

A estrutura de um programa direto é semelhante a dos programas nas linguagens FORTRAN, Pascal ou C. Por exemplo, deseja-se obter os valores da área da base (ab), da área lateral (al) e do volume (v) de um cilindro. Isto é feito usando os seguintes cinco comandos:

```
 r = input('Raio    r = ');
 h = input('Altura  h = ');
ab = pi*r^2*2;                    % area da base
al = 2*pi*r*h;                    % area lateral
 v = pi*r^2*h;                    % volume
```

Os programas diretos devem ser digitados em um editor de texto ASCII e gravados com a extensão **.m**. Nomeando este programa direto, como **medida**, deve ser gravado como `medida.m`. Ao digitar

`>> medida` [ENTER]

na janela de comandos, o MATLAB pedirá os valores do raio e da altura do cilindro. Após a entrada dos valores, o programa será executado e emitirá os resultados desejados.

4.2 Arquivos de Funções

Estes arquivos são construídos em uma forma diferente dos arquivos de comandos. A maioria dos comandos ou funções do MATLAB, tais como `inv`, `angle`, `sqrt`, `balance`, `plot`, etc., são arquivos–função instalados no diretório `\matlab\toolbox\matlab`. Estes arquivos (*.m) **não** devem ser modificados de maneira alguma. Arquivos de funções tem as seguintes caraterísticas:

1. A primeira linha do arquivo contém a palavra reservada `function`

2. A primeira linha do arquivo declara o *nome* da função, os *argumentos de entrada* e os *argumentos de saida.*

   `function` *arg_saida = nome_function(arg_entrada)*

   Se o número de argumentos de saida for dois ou mais, estes são escritos na forma de um vetor:

   `function` [a1, a2, ..., an] = ...

3. Linhas começando con o caracter % que aparecem a partir da segunda linha do arquivo são usadas para comentários relativos à função. Estas linhas serão mostradas quando o comando

   `>> help` *nome_function* [ENTER]

   for executado na janela de comandos do MATLAB. Estas linhas de ajuda são opcionais, porém, é recomendável fazer-lo para facilitar o entendimento por outros usuários.

4. Quando um arquivo de função é executado, os comandos deste arquivo *não* são mostrados na tela.

No exemplo acima, os dados de entrada são o raio r e a altura h do cilindro . Como saída, tem-se as variáveis ab, al e v. Assim, o arquivo de função para este propósito é dado por:

```
function [ab,al,v] = cilindro(r,h)
%CILINDRO  Area e Volume de um cilindro                <help>
%          [b,l,v] = CILINDRO(R,h) calcula:            <help>
%                Area da Base,                         <help>
%                Area Lateral  e                       <help>
%                Volume                                <help>
%           de um cilindro de raio = r    e            <help>
%                              altura = h              <help>
%

%    Esta linha nao e' parte do HELP !!
ab = pi*r^2;
al = 2*pi*r*h;
v = pi*r^2*h;
```

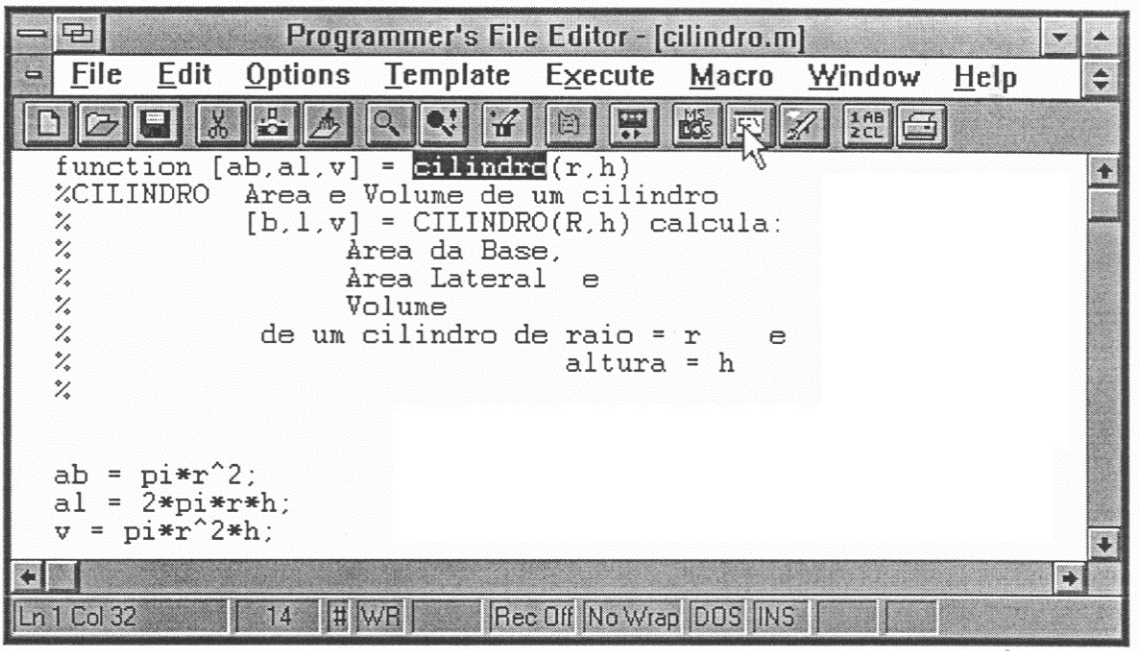
Para usar esta função, basta digitar seu nome, seguido dos dados da entrada, por exemplo,

```
>> [ab,al,v] = cilindro(8.5, 20.0)
```

Após pressionar a tecla [ENTER], o MATLAB fornece os valores da área da base, área lateral e do volume de um cilindro de raio 8.5 e altura 20.

4.3 Edição e Manipulação de Arquivos

Um arquivo de comandos bem como um arquivo de função, com suas linhas de comandos, deve ser digitado em qualquer editor de texto ASCII (sem acentos ou carateres especiais). Recomenda-se editar no PFE (Programmer's File Editor), no Notepad ou no Edit Master, e gravar no diretório de trabalho (ou onde o MATLAB possa encontrá-lo), com a extensão `*.m`.



Ao executar o comando:

```
>> help cilindro
```

aparecerá na tela a seguinte informaçãosobre o arquivo–função `cilindro.m`:

```
  CILINDRO  Area e Volume de um cilindro
            [b,l,v] = CILINDRO(R,h) calcula:
                  Area da Base,
                  Area Lateral  e
                  Volume
             de um cilindro de   raio = r    e
                               altura = h
>>
```

Os comandos de manipulação de arquivos `*.m` são:

`dir`	Lista os arquivos em diretório o subdiretório
`delete`	Apaga um arquivo do diretório
`chdir, cd`	Muda de diretório
`cd, pwd`	Sem argumentos, mostram qual é o diretório atual
`help`	Informação sobre um comando ou arquivo
`type`	Mostra na tela o conteúdo de um arquivo `.m`
`what`	Retorna uma lista de M–arquivos no diretório atual
`!`	Executa comandos do sistema operacional (copy, rename.etc.)

O comando `input`, no MATLAB, tem a seguinte sintaxe:

```
>>  input('texto')
```

Quando é acionado, solicita a entrada de um número, vetor, matriz, texto ou expressão. Para que os valores da entrada sejam armazenados, é necessário atribuir uma variável ao input. Um ponto e vírgula após o comando, elimina a exibição na tela do conteúdo da variável.

O MATLAB possui a seguinte macro que facilita a execução de texto:

```
>> eval(nome do texto)
```

Qualquer comando pode ser colocado no texto e `eval` é utilizado para executá-lo. Por exemplo,

```
>> x = 3;
>> n = 5;
>> c1 = 'cos(pi*x/n)';
>> eval(c1)
```

proporciona o valor

```
>> ans =
>>        -0.3090
```

Se um conjunto de comandos deve ser executado freqüentemente, o mesmo pode ser salvado numa cadeia de texto ('strings') e, após, ser acessado utilizando o comando `eval`. Por exemplo, suponha-se que uma coleção de funções deva ser calculada no intervalo $[a, b]$ em m passos de $(b - a)/m$. Pode-se utilizar `eval`, da maneira seguinte:

```
>>  s3 = 'f = input(''entre com a formula da funcao de x'');
    m = input('m='); a = input('a='); b = input('b= ');
    xval = [ ]; yval = [ ]; for x= a:(b-a)/m:b, xval=[xval
     x];yval=[yval eval(f)]; end, tab=[xval;yval]''
```

Observe-se que a função f e os valores m, a, b podem ser quaisquer, definidos de acordo com o interesse do usuário. Também, que a digitação da cadeia deve ser completada antes de pressionar [ENTER] ou [RETURN]. Quando a cadeia for muito comprida, como neste exemplo, pode ser concatenada da maneira seguinte:

```
>>  s3='f=input(''entre com a formula da funcao de x:'');'
>>  s3=[s3,'m=input('m= ');a=input('a= ');b=input('b= ');']
>>  s3=[s3,'for x= a:(b-a)/m:b, xval=[xval x];']
>>  s3=[s3,'yval=[yval eval(f)],end,tab=[xval;yval]']
```

Então, pressiona-se <Enter> ou <Return> e o comando eval(s3) mostrará na tela

```
>>
  entre com a formula da funcao de x:      'cos(x)' <return>
>>
>> f =
       cos(x)
>> m = 4
>> a = 0
>> b = 2*pi
>> tab =
          0.0000          1.0000
          1.5708          0.0000
          3.1416         -1.0000
          4.7124          0.0000
          6.2831          1.0000
```

A matriz vazia [ ] é utilizada para inicializar cálculos. Em cada passo do laço, xval aumenta o seu tamanho até obter um vetor linha de ordem $1 \times m$. Quando se deseja utilizar o mesmo programa com diversas funções, deve-se gravar no editor com um nome arbitrário, munido da extensão .m.

O comando eval foi utilizado para realizar operações com uma função dada. O mesmo resultado pode ser obtido através de um arquivo function.

```
 function F = funcval(f,a,b,incr)
% FUNCVAL  Calcula uma funcao escalar sobre [a,b]
%          em pontos x(j)= a + j*incr .  A funcao
%          e' especificada como um texto em f e deve
%          ser escrita como funcao de x. A saida e'
%          uma tabela dos valores x e y=f(x).

% Utilizacao de funcval(f,a,b,incr) cria-se os
% arranjos x e y
xval=[ ]; yval=[ ];
% inicia o calculo de f(x)
for x= a:incr:b
    xval=[xval x] ;   yval=[yval eval(f)];
end
F =[xval; yval]';
```

Utilizando `funcval`, os comandos

```
  f ='cos(x)'
tab = funcval(f,0,2*pi,2*pi/4)
```

fornecem a mesma tabela do exemplo `eval(f)`. Outras formas são:

```
tab = funcval('1+x',-1,1,.25)
```

ou

```
a=-1;   b=1;   passo = .25;  tab = funcval(f,a,b,passo)
```

A seguir, será construído um **arquivo de comandos** (programa direto) que exibe graficamente os valores de uma função, a partir de uma tabelade valores, que é criada com a function **funcval**. Note-seque uma **function** pode chamar outras **functions** ou arquivos de comandos ou a ela mesma.

```
% SKFUNC Esboco de uma funcao fornecida por uma formula.
%        Sega-se as instrucoes que aparecem na tela.
%        *****Este comando utiliza FUNCVAL**************
```

```
f=input('entre com a formula da funcao de x como string');
disp('entre o intervalo [a,b]')
a=input('a=  '); b=input('b= ');
disp('entre o incremento'), incr=input('incr= ');
tab=funcval(f,a,b,incr);
x=tab(:,1); y=tab(:,2); cla
disp('**escolher uma janela para o grafico **')
disp('o menor valor de x nos dados e '); xmin= min(x);
disp('o maior valor de x nos dados e '); xmax= max(x);
xp=input('menor x a ser utilizado = ');
xg=input('maior x a ser utilizado = '); cla
disp('menor y nos dados e '); yp= min(y)
disp('maior y nos dados e'); yg= max(y)
yp=input('menor valor de y a utilizar = '); cla
yg=input('maior valor de y a utilizar = '); cla
disp('pressione enter para ver o grafico');
disp('----------------------------------')
disp('para finalizar pressione outra vez enter')
pause
clg
axis([xp xg yp yg])
plot(x,y,'*')
hold; plot(x,y,':')
ft=['function' f];
title(ft)
pause
axis
hold
cla
```

O usuário digita **skfunc** e o programa é interativo, no sentido que os dados são solicitados ao usuário através do teclado.

Neste programa, foram introduzidos vários outros comandos do MATLAB. O comando

```
>> disp('texto') ou disp(A)
```

apresenta o conteúdo de um texto ou de uma variável, respectivamente. O comando

```
>> pause
```

interrompe a execução do programa e aguarda pela digitação de uma tecla para continuar. Os comandos

```
>> cla
>> clg
```

limpam, respectivamente, o texto e o gráfico da tela .

O comando

```
>> axis([v1 v2 v3 v4])
```

fornece uma escala para a janela gráfica, de tal modo que

x mínimo = v1,
x máximo = v2,
y mínimo = v3,
y máximo = v4.

Para vetores **x** e **y**, o comando

```
>> plot(x,y,'*')
```

plota os pares ordenados $(x(i), y(i))$ utilizando a marca "$*$". A letra 'w' no lugar do '+', apresenta cor branca. Normalmente, sem o uso do comando `axis`, é feita uma autoescala com o máximo e o mínimo valor dos elementos nos vetores **x** e **y**. Os comandos

```
>> min(x)
>> max(x)
```

fornecem estes valores. Se **A** é uma matriz, então `max(A)` fornece os maiores elementos nas suas colunas.

O comando

```
>> hold
```

segura a imagem gráfica que está na tela, de modo que outras imagens possam ser superpostas. Os limites dos eixos são congelados, até que o comando `hold` seja novamente chamado.

Deve-se observar que *as variáveis definidas e manipuladas dentro de uma function ou programa são locais e não existem fora dela. As únicas variáveis que permanecem, são exatamente aquelas definidas como variáveis de saída.* Para `skfunc`, as variáveis f, a, b, $incr$, tab, x, y, xp, xg, $xmin$, $xmax$, yg, yp, $ymin$ e $ymax$ serão incorporadas ao espaço de trabalho do MATLAB, uma vez que a execução do arquivo `.m` seja completada. Porém, variáveis completamente internas a `funcval` não estarão disponíveis.

4.4 Equações Diferenciais

O MATLAB possui comandos para resolver sistemas de equações diferenciais ordinárias com valores iniciais. O primeiro passo é a introdução do sistema de equações e o segundo a utilização do comando `ode23` ou `ode54` os quais são métodos de integração numérica do tipo Runge-Kutta-Fehlberg com ajuste automático do passo. Por exemplo, o sistema

$$\begin{aligned} \dot{x}_1 &= 4x_1x_2 + x_2^2 \\ \dot{x}_2 &= 3x_2sin(x_1) - (x_1^2 + x_2^2) \end{aligned}$$

entra no MATLAB como a função

```
function xdot = sistema(t,x)
%
x12 = x(1)^2 + x(2)^2 ;
xdot = [4*x(1)*x(2) + x(2)^2; 3*x(2)*sin(x(1) - x12 ]
```

Para simular o sistema de equações diferenciais, definido acima como `sistema`, sobre o intervalo $0 \leq t \leq 30$ e com a condição inicial $x_1 = 1, x_2 = 2.5$, chamamos `ode23`:

```
t0 = 0;
tf = 30;
x0 = [1 2.5];
[t,x] = ode23('sistema',to,tf,x0);
plot(t,x)
```

O comando `ode54` é utilizado da mesma maneira que `ode23`, sendo indicado quando uma maior exatidão numérica for necessária.

Capítulo 5

Gráficos e Janelas Gráficas

5.1 Gráficos Simples

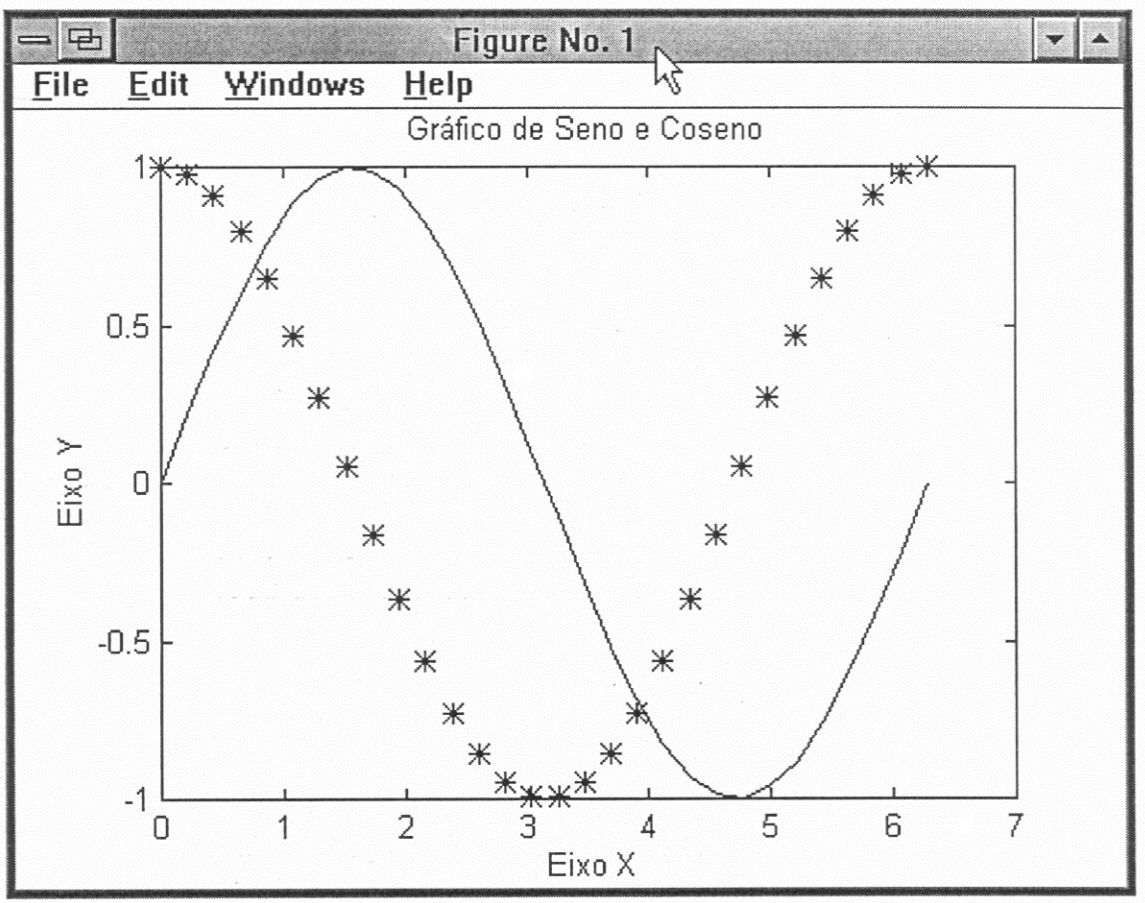
"Plot"é uma forma poderosa de visualizar um conjunto de dados numéricos no MATLAB. Um desenho é obtido usando o comando `plot`. Este comando faz a escolha automática de limites dos eixos e exibe pontos de dados individuais usando diferentes marcadores (*, +, ., -, x, o, :, –). O comando `plot` usa como argumentos pares de dados, correspondentes às coordenadas x e y.

Associados ao comando `plot`, podemos usar comandos para colocar rótulos aos eixos, título ao gráfico, desenhar uma grade, etc. Cada comando `plot` gera um gráfico na janela gráfica. Se numa janela já está sendo visualizado um gráfico, um novo comando `plot` apagará o desenho existente e mostrará outro novo. Mais de um gráfico numa única janela pode em ser visualizados usando mais de uma par de dados como argumentos do comando `plot` ou usando o comando `hold on`.

Os seguintes comandos

```
>> x=linspace(0, 2*pi,30);
>> y=sin(x);
>> z=cos(x);
>> xlabel('Eixo X');  ylabel('Eixo Y');
>> title('Grafico de Seno e Coseno');
>> plot(x,y,'-',x,z,'*')
```

produzem o gráfico



5.2 Gráficos 2D

Usando o comando `plot`, o MATLAB permite manipular os gráficos através de um conjunto de comandos e alternativas:

- **Plot**

```
>>  x = linespace(0, 2*pi,30);
                          % vetor de 30 pontos entre 0 e
                          % 2*pi espacados linearmente
>>  y = sin(x);
>>  z = cos(x);
```

```
>>  plot(x,y)          % grafica  y=sin(x)
>>  plot(y,x)          % Rota 90 graus  y=sin(x)
>>  plot(x,z)          % grafica  z=cos(x)
>>  plot(x,y,x,z)      % grafica os dois graficos numa  (*)
                       %    UNICA janela
>>  W = [ y ; z ];     % Matriz de seno e coseno
>>  plot(x,W)          % El mesmo grafico de (*)
>>  plot(W,x)          % Rotando 90 graus
```

- **Tipos de linhas, Marcadores e Cores**

  O comando `plot` tem a seguinte sintaxe:

  `plot(x1,y1, tipo1, ..., xN, yN, tipoN)`

  onde `tipo1, ..., tipoN` são strings de 1, 2 ou 3 caracteres que representam um tipo de linha ou uma determinada cor, segundo a seguinte tabela: [1]

Símbolo	Cor	Símbolo	Tipo de linha
y	amarelo	.	ponto
m	magenta	o	círculo
c	cyan	x	marca "x"
r	vermelho	+	marca "+"
g	verde	*	estrela
b	azul	-	linha *sólida*
w	branco	:	linha *pontilhada*
k	preto	-.	linha *ponto–traço*
		- -	linha *tracejada*

Tabela 5.1: Cores e tipos de linhas do comando `plot`

Exemplos:

```
>>  x = linespace(0, 2*pi,30);
>>  y = sin(x);
>>  z = cos(x);
>> plot(x,y,'g:')          %  grafico verde pontilhado
>> plot(x,y,'r--')         %  grafico vermelho tracejado
>> plot(x,y,'r',x,z,':')   %  y=sin(x) vermelho - solido
                           %  z=cos(x) vermelho - tracejado
```

[1] Se a cor não é especificada, MATLAB usa o ciclo: amarelo - magenta - cyan - vermelho - verde - azul - amarelo - ... . Se o tipo de linha não é especificado, MATLAB usa *linha sólida*.

O fundo da janela gráfica sempre é *preta* (obscuro). Para mudar para a cor branca, no inicio de uma sessão no MATLAB, dar o comando:

```
>> whitebg
```

Isto é muito importante para importar ou imprimir gráficos feitos no MATLAB.

- **Grades (malhas) e Rótulos**

O comando `grid on` desenha uma grade pontilhada sobre o atual gráfico. MATLAB também fornece comandos para rotular os eixos, colocar título ao gráfico e colocar texto em qualquer ponto (x, y):

- xlabel(' texto ')	rótulo do eixo X
- ylabel(' texto ')	rótulo do eixo Y
- title(' texto ')	titulo do gráfico
- text(x,y,' texto')	`texto` no ponto (x,y)

Como exemplo, temos:

```
>> x=linspace(-2*pi,2*pi);
>> y=10*x.*exp(-5*x.^2);
>> plot(x,y);
>> grid on;
>> xlabel('Variavel Independente X');
>> ylabel('Variavel Dependente Y');
>> title('Aqui vai o TITULO');
>> text(0,-0.7,' y = 10*x.*exp(-5*x.^2)');
```

gera o gráfico da Fig. 5.1. Como pode se observar, o tipo de texto que aparece no gráfico é de uma única fonte, de um único tamanho e não pode se incluir carateres especiais.

Outro comando que permite inserir texto dentro de uma figura é o `gtext`. A declaração:

```
>> gtext('Meu Texto');
```

permite colocar o texto `Meu texto` em qualquer ponto da janela gráfica atual (em forma interativa) usando o mouse.

Existe um toolbox chamado *Styled Text Toolbox*, escrito por Douglas M. Schwarz (e disponível no servidor de arquivos do The MathWorks), que permite:

- Usar mais de uma fonte em uma linha de texto: Times, Courier, Symbol, Zapf Dingbats, etc.
- Usar diferentes estilos de texto: itálico, negrito.

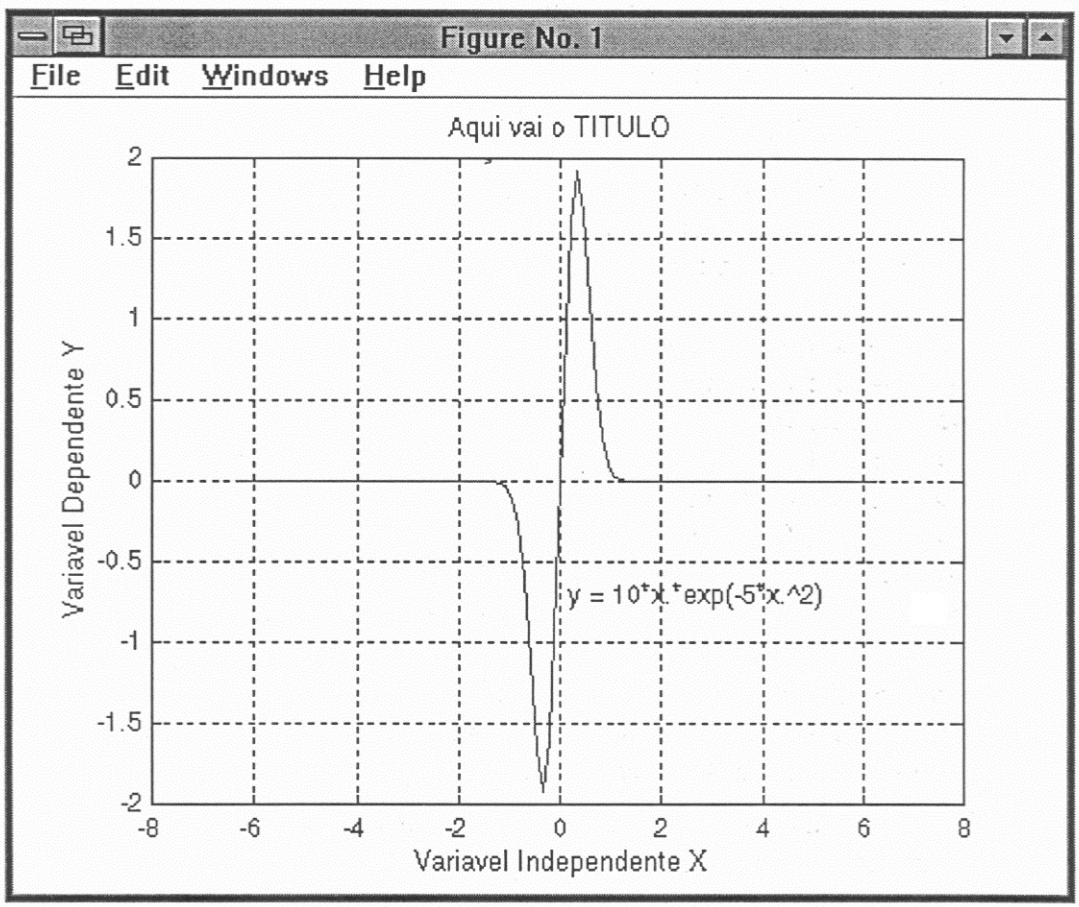


Figura 5.1: Texto normal em gráficos MATLAB

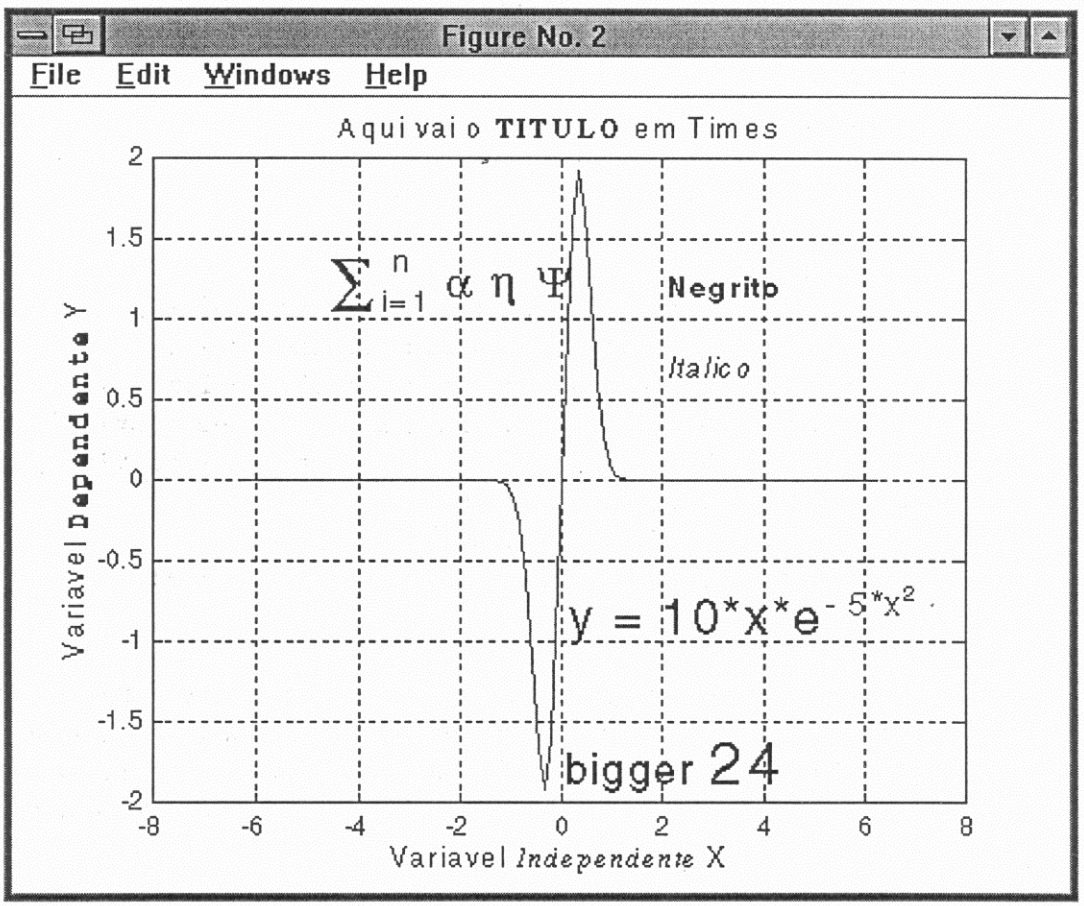


Figura 5.2: Texto usando o *Styled Text Toolbox*

- Subscrito e Superscrito: `x_i` `z^2`
- Letras gregas (maiúsculas e minúsculas)
- Caracteres LaTeX : `\leq`, `\approx`, `\cup`, `\bullet`, etc.
- Cores e tamanhos: `\red`, `\green`, `\18`, `\16`, `\20`, etc.

Obviamente, para funcionar corretamente, este toolbox debe ser instalado no diretório `\matlab\toolbox\stextfun`. O gráfico da Fig. 5.1, pode ser melhorado usando o Styled Text Toolbox, através do seguinte programa:

```
>> x=linspace(-2*pi,2*pi);
>> y=10*x.*exp(-5*x.^2);
>> plot(x,y);
>> grid on;
>> sxlabel('Variavel {\i\times Independente} X');
>> sylabel('Variavel {\b\courier Dependente} Y');
>> stitle('Aqui vai o {\b\times TITULO} em Times');
>> stext(0,-0.9,' {\14\red\larger y = 10*x*e^{-5*x^2}');
>> stext(-4.7,1.2,' \18\sum{i=1, n} \alpha \eta \Psi');
>> stext(2.1,1.2,'\b Negrito');
>> stext(2.1,0.7,'\i Italico');
>> stext(0.1,-1.8,'{\bigger bigger} {\24 24}');
```

- **Eixos**
  Os eixos X e Y de um gráfico 2D podem ser modificados através do comando `axis`. Este permite ter uma janela retangular ou quadrada, ter escalas diferentes para cada eixo, ter limites máximo e mínimo para os eixos, etc. Use o comando `help axis` para a sintaxe do comando `axis`.

- **Manipulação de gráficos**

  - `hold`: Cada comando `plot` remove o desenho existente sobre a janela gráfica. Quando queremos ter mais de um gráfico gerados por dois ou mais comandos `plot`, numa única janela, devemos usar o comando `hold on`. Os novos gráficos serão re-escalados, resultando possivelmente em deformações dos primeiros desenhos.

    ```
    >> x=linspace(-2*pi,2*pi,30);
    >> y=sin(x);
    >> z=cos(x);
    ```

```
>> w=x.^2;
>> plot(x,y)        % janela com y = sin(x)
>> hold on;
>> plot(x,z)        % janela com y=sin(x)  e z=cos(x)
>> hold on
>> plot(x,w)        % janela com tres graficos: os
                    % dois primeiros deformados !
```

- `figure`: permite criar uma nova janela gráfica para cada comando `plot`. Para cada comando `figure`, o nome da janela será modificado: Figure No 1, Figure No 2, ...
- `subplot`: permite ter mais de um desenho com seus respectivos eixos na mesma janela gráfica. O comando

  subplot(m,n,p)

  divide a janela gráfica em uma matriz de $m \times n$ partes e escolhe a parte p como área ativa. Cada parte é numerada de esquerda para direita e de acima para abaixo. Assim, `subplot(2,2,3)` significa que a janela gráfica esta dividida em 4 partes (2×2) e que a parte ativa, onde vai o desenho, é a de número 3. O programa:

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
a=2*sin(x).*cos(x);
b=sin(x)./(cos(x) + eps);
%%
subplot(2,2,1);  plot(x,y); title('sin(x)');
subplot(2,2,2);  plot(x,z); title('cos(x)');
subplot(2,2,3);  plot(x,a); title('2sin(x)cos(x)');
subplot(2,2,4);  plot(x,b); title('sin(x)/cos(x)');
```

produz a Fig. 5.3

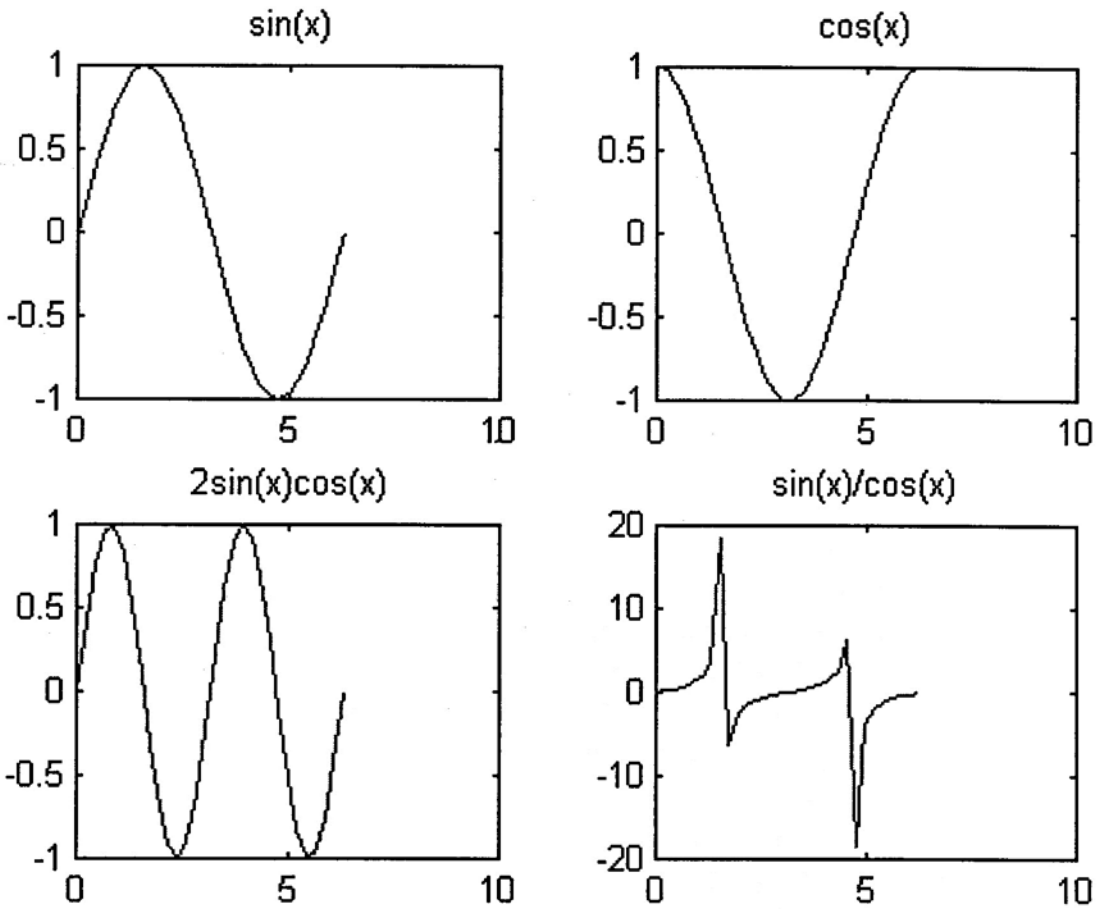


Figura 5.3: Subplots 2D

- **Outros comandos**
  - `polar(t,r,S)`: permite fazer gráficos em coordenadas polares.
  - `bar(x,y)` : gráficos de barras.
  - `stair(x,y)` : gráficos em escada.
  - `fill(x,y,'c')` : preenche o polígono 2D definido pelos vetores coluna `x`, `y` com a cor especificada por `c`. Os vértices do polígono são especificados pelos pares (x_1, y_i).

5.3 Gráficos 3D

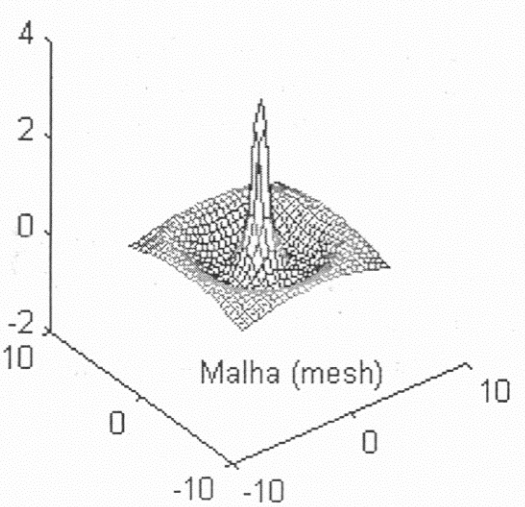
MATLAB fornece um conjunto de funções–comandos que permitem mostrar gráficos de dados no espaço tridimensional (3D).

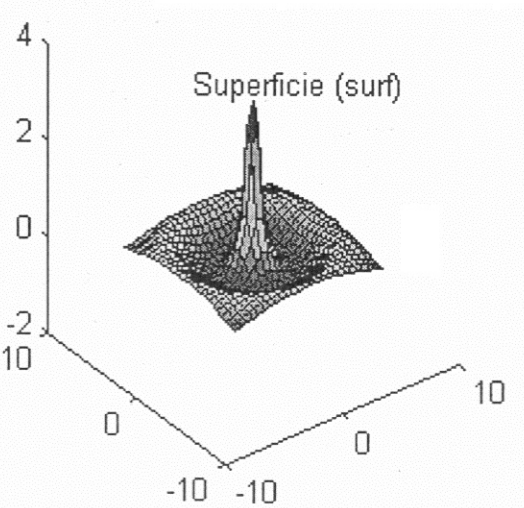
- **O comando `plot3`**
  Este é uma extensão do `plot` 2D. A sintaxe é a seguinte:
  $$\texttt{plot3}(x_1, y_1, z_1, S_1, \ldots, x_n, y_n, z_n, S_n)$$
  onde x_n, y_n e z_n são vetores ou matrizes e S_n são strings que indicam a cor, o marcador e/ou o tipo de linha.

  A função `text` também tem uma sintaxe 3D: `text(x,y,z,'string')`

- **Malhas e Superfícies 3D**
  Malhas são gráficos resultantes da união de dois pontos adjacentes através de linhas retas. Malhas são muito usadas para visualizar matrizes grandes ou para graficar funções de dois variáveis.

```
>> x = -2*pi : 0.5 : 2*pi;
>> y = x;
>> [X, Y] = meshgrid(x,y);
>> raio = sqrt(X.^2 + y.^2) + eps ;
>> Z = cos(raio) ./ raio ;
>> mesh(X,Y,Z)
```





Uma *superficie 3D* de uma matriz Z tem a forma similar a uma malha, exceto que os espaços entre as linhas são prenchidos com uma determinada cor. Superfícies são geradas usando o comando `surf` com os mesmos argumentos do comando `mesh`.

- **Contornos**
  Gráficos de *contornos* mostram linhas (curvas) de elevação constante ou profundidade. Estes gráficos podem ser 2D (`contour`) ou 3D (`contour3`).

```
>> contour(Z);            % 2D
>> contour3(Z);           % 3D
>> contour3(Z,20);        % 20 linhas de contorno
```

- **Outros comandos**
  - `view(azimuth, elevation), view([x,y,z])`
    permite especificar o angulo de visão do observador.
  - `hidden on/off`
    Controla a remoção de linhas ocultas de um gráfico 3D.
  - `meshc = mesh + contour`
  - `surfc = surf + contour`
  - `surfl = surf +` luz (iluminação)
  - `shading flat/faceted/interp` (sombreado)

- **Mapas de Cor Pré-definidos**
  MATLAB define mapas de cores como matrizes de tres colunas, onde cada fila define uma cor particular usando números entre 0 e 1. Estes números especificam os valores RGB: a intensidade do vermelho, verde e azul. O comando

```
>> colormap(M)
```

  instala a matriz M como o mapa de cores a ser utilizado pela figura em uso. MATLAB tem pré-definido 10 mapas de cores: `hsv`, `hot`, `cool`, `pink`, `gray`, `bone`, `jet`, `copper`, `prism` e `flag`.

  O fato de os mapas de color serem matrizes, permite que o usuário defina seus mapas de color particulares.

5.4 MATLAB, LaTeX e MS-Word

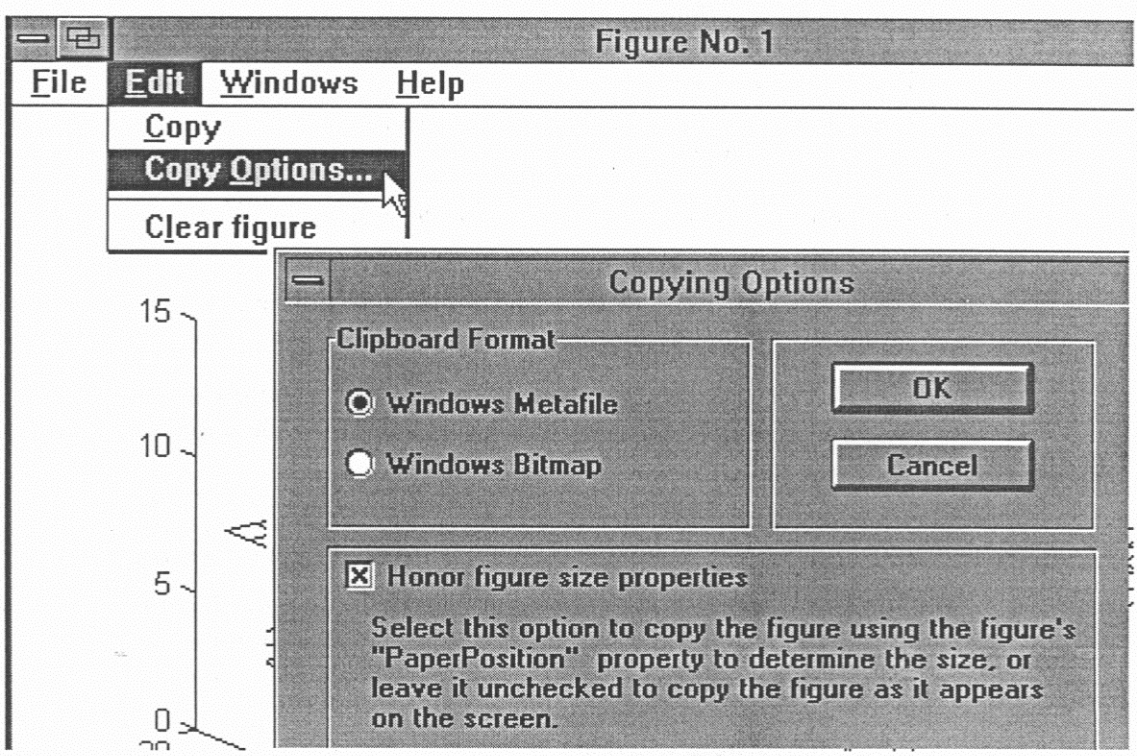
Na maioria das vezes, os cálculos e os gráficos feitos no MATLAB devem ser incorporados em um documento (relatório, tese, manual, livro, etc.). Se o documento é feito em um processador de texto *for Windows*, a incorporação é muito fácil: usar o `Copy` do menu `Edit` da janela gráfica do MATLAB e depois o `Paste` do menu `Edit` do processador de texto.

A seguir descrevemos como incorporar cálculos e gráficos para o LaTeX (versão LaTeX 2_ε, 01 Jun. 1995) e para o MS Word for Windows.

- **MATLAB - LaTeX**
  Antes de apresentar o procedimento para incorporar gráficos do MATLAB no LaTeX, devemos considerar os seguintes aspectos:
  - Cada gráfico em uma janela do MATLAB corresponde a uma página completa no papel de impressão.
  - O tamanho e tipo de página deve ser configurado pelo Print Manager do MS Windows.
  - Os gráficos (figure) MATLAB são feitos usando diferentes cores.
  - Dependendo da classe de documento LaTeX a ser usado ([a4paper, letterpaper]{article, book}) cada página tem margens e espaço fixo para texto e gráficos.
  - Cada sistema TeX usa diferentes opções gráficas e diferentes drives. O sistema **emTeX** para PCs, usa drives e arquivos somente em duas cores: branco e preto.
  - No LaTeX é possível incorporar dois tipos de gráficos: arquivos en formato gráfico (Bitmap preto e branco *.bmp, Paintbrush *.pcx, MS Paint *.msp) e PostScript (*.ps). Este último é usado, quando a impressão será feita em uma impressora laser com PostScript.
  - Os gráficos *.bmp, *.pcx e *.msp são visíveis na maioria dos 'previews' para arquivos *.DVI (dviscr, xdvi, ptiview, etc.), porém, os gráficos *.ps são visíveis somente através de programas tais como GhostScript (DOS, MS Windows) e PageView (Unix, SUN).
  - Os tamanhos máximos para arquivos *.bmp a serem incorporados em documentos processados pelo sistema **emTeX** é: width=32760 pixels, height=32766 pixels.

Com base nestas considerações, para incorporar um gráfico MATLAB no LaTeX, usando o sistema **emTeX** para PCs e na forma de **arquivos gráficos *.bmp, *.msp, *.pcx**, segue-se o seguinte algoritmo:

1. Antes de gerar qualquer gráfico, executar no prompt do MATLAB o comando `whitebg`.
2. Obter o gráfico (figure) MATLAB usando a cor preta:
   `plot(x,y,'k')`.
   Recomendável: Use o comando `colormap(zeros(8,3))`; para gráficos feitos com `mesh`.
3. No menu da janela gráfica `Edit`→`Copy Options`, escolher:
   Clipboard Format:
   ◉ Windows Metafile
   [X] Honor figure size properties



4. Executar o comando `Edit`→`Copy`
5. Imediatamente, recuperar o gráfico do Clipboard usando um programa gráfico (Sugestão: Paint Shop Pro ou Aldus PhotoStyler) e o comando `Edit`→`Paste`→`As New Image`. *Tenha paciência, demora um poquinho ...!.*

6. Transformar esta nova janela gráfica (new-*.bmp) para duas cores: preto e branco. Geralmente o arquivo obtido pelo comando `Paste` tem 16, 256 ou mais cores. Na linha de statuts, geralmente aparece
   nnn X mmm X 256
   No Paint Shop Pro, seguir o comando:
   `Colors→Decrease Color Depth→2 Colors...`
7. Executar (No Paint Shop Pro) o comando
   `View→Zoom Out (1:2)`
   e logo recortar a parte significativa do gráfico (região retangular). O gráfico original tem muitas áreas em branco que incrementam o tamanho do arquivo *.bmp .
   Depois de marcar a região, executar os seguintes comandos:
   `Edit→Copy→Paste→As New Image`

8. Dimensionar a janela gráfica resultante de acordo com o tamanho que deseja-se ter no documento impresso. Suponha-se que queremos obter um gráfico impresso com as dimensões (em centimetros) **A** × **B**. Para isto, usamos a seguinte tabela:

   largura: **A** cms $= 95.25 \times$ **A** pixels
   $= 28.45 \times$ **A** pontos (pt)

   altura: **B** cms $= 83.33 \times$ **B** pixels
   $= 28.45 \times$ **B** pontos (pt)

   Para um gráfico impresso de 10.5cms × 6cms, precisamos de uma janela gráfica de $(10.5)(95.25) \times (6)(83.33) = 1000 \times 500$ pixels.
   No Paint Shop Pro, usar o comando
   `Image→Resize`
   Salvar este arquivo no diretório de trabalho ou, onde o LaTeX possa encontrá-lo (com a extensão .BMP).
9. No preâmbulo do documento LaTeX, usar os comandos:
   `\usepackage{figura}`
   `\usepackage[emtex]{graphics}` (só para **emTeX**)
   Nota: *Ver a documentação do utilitário* `graphics`.
10. Incorpore o arquivo fráfico obtido `*.bmp` (por exemplo, teste.bmp) na parte conveniente do seu documento através dos comandos:

```
\begin{center}
  \begin{figure}[H]
      \includegraphics[300,171]{teste.bmp}
```

```
            \caption{Inclussao de um grafico BMP}
        \end{figure}
    \end{center}
```

No comando `\includegraphics[urx,ury]{file.bmp}`, os parâmetros `[urx,ury]` indicam as coordenadas da esquina superior direita da imagem. As unidades são pontos LaTeX (pt). No exemplo,

1000 pixels = 10.5 cm = 298.72 pt

500 pixels = 6 cm = 170.7 pt

Verificar os valores de `[urx,ury]` ...!

Exemplo 5.4.0

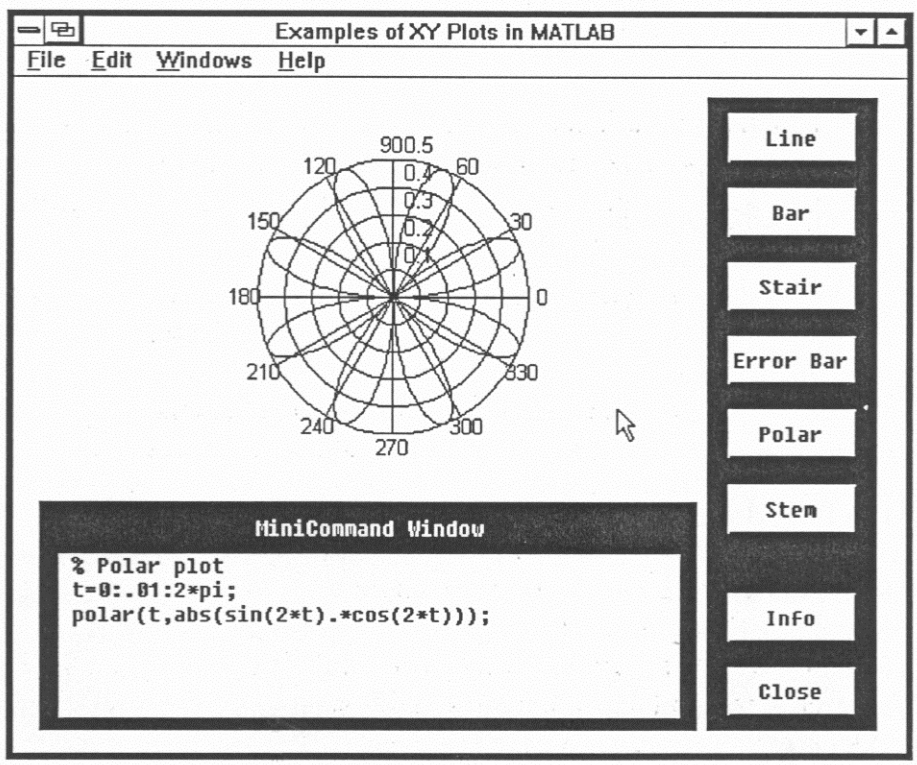


Figura 5.4: Inclussão de um gráfico BMP do Matlab (1429 x 1172 pixels) usando `includegraphics[350,280]{...}`

Para incorporar um gráfico MATLAB no LaTeX, na forma de um arquivo **PostScript** *.ps seguir o seguinte algoritmo:

1. Executar no prompt do MATLAB o comando `whitebg`.

2. Obter o gráfico MATLAB.
   Recomendável: Use o comando `colormap(zeros(8,3))`; para gráficos feitos com `mesh`.

3. Usando o Print Setup do MS Windows, configurar a impressora na qual será posteriormente impresso o documento LaTeX, usando as opções `Print Setup→Setup→Options`. Os comandos escolhidos devem ser:

   Print to:
   ◉ Encapsulated PostScript File
   Margins:
   ◉ None

   Ver Fig. 5.5

4. Imprimir o gráfico MATLAB. Usar a extensão .ps para o nome do arquivo impresso.

5. No inicio do documento LaTeX, usar o comando:
   `\usepackage{figura}`.

6. Incorpore o arquivo PostScript através dos comandos:
   `\hfigps{filename.ps}{Nome da figura}{label}`
   ou
   `\figps{filename.ps}{Nome da figura}{label}`
   Outra forma alternativa (sem nome nem número de figura) é:
   `\hfps{filename.ps}` ou `\fps{filename.ps}` ·
   A letra 'h' nos comandos acima significa *aqui* (do inglês: here)

- **MATLAB - MS Word**
  O MATLAB *Notebook* é uma interface para acessar o MATLAB dentro de um documento Microsoft Word ver. 6.0. A interface Notebook permite que comandos criados dentro de um documento MS Word sejam enviados ao MATLAB para sua avaliação. O resultado do processamento é retornado e inserido automaticamente no documento. Ambos, texto e gráficos, são permitidos.

  Supondo que o MATLAB Notebook já esta instalado no disco C, para usá-lo seguir os seguintes pasos:

  1. Ingresar no MS Word 6.0 clicando no icone correspondente.

  2. Escolha no Menu Principal a opção `File-New-M-book` como mostra o gráfico seguinte:

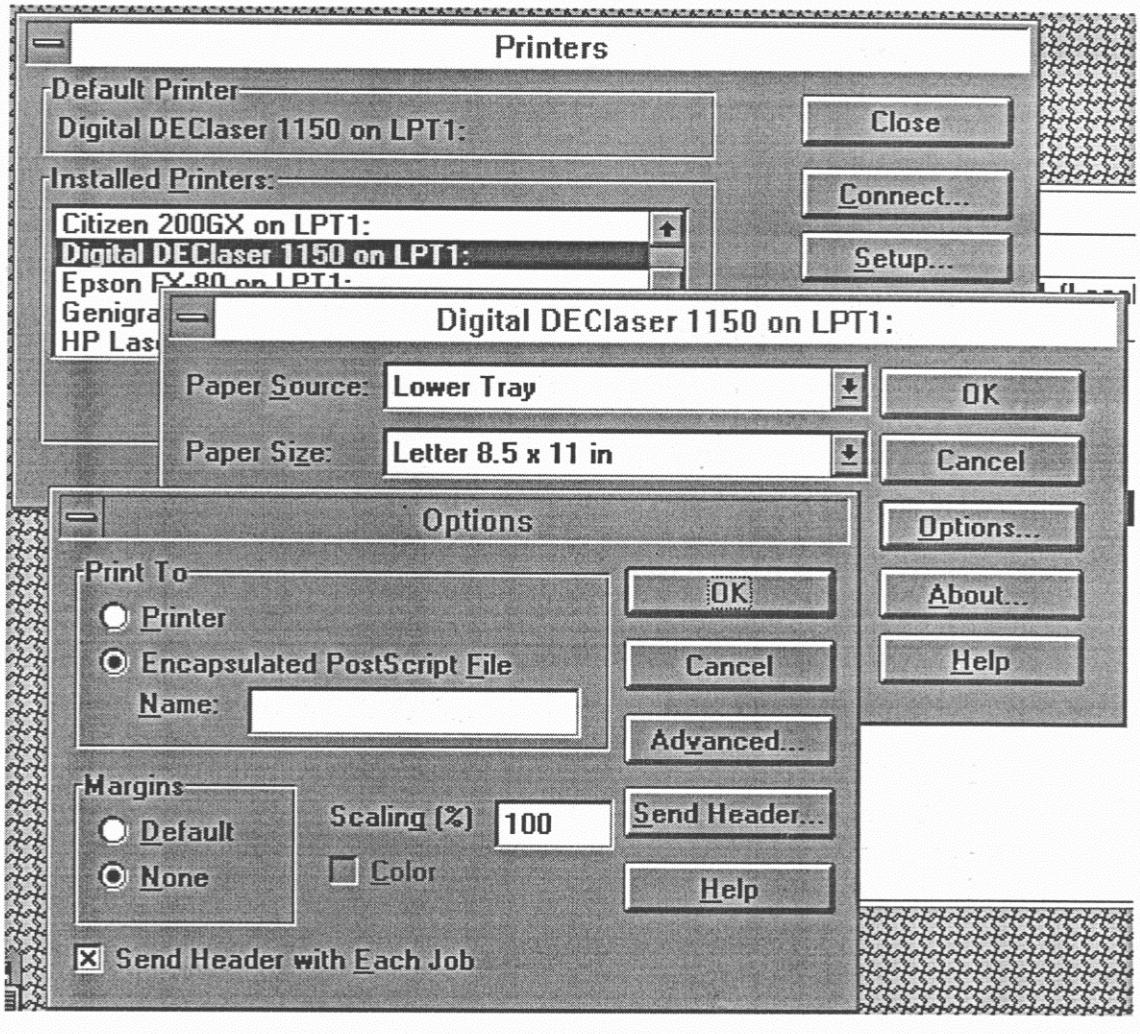
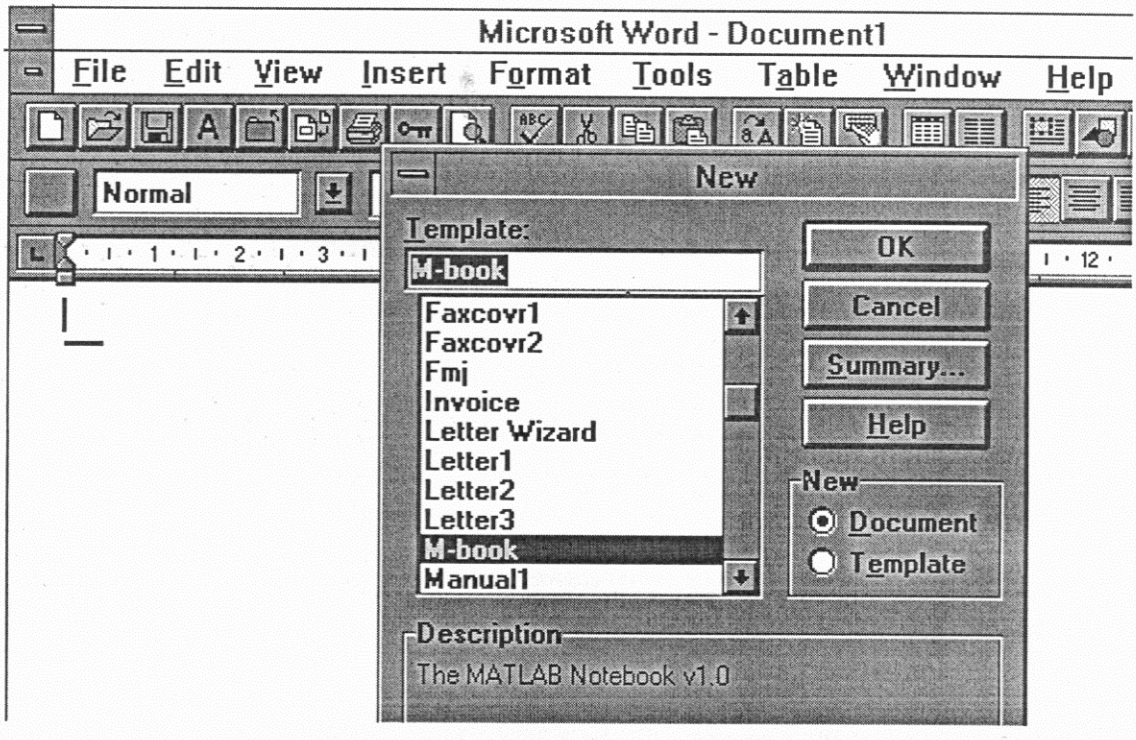
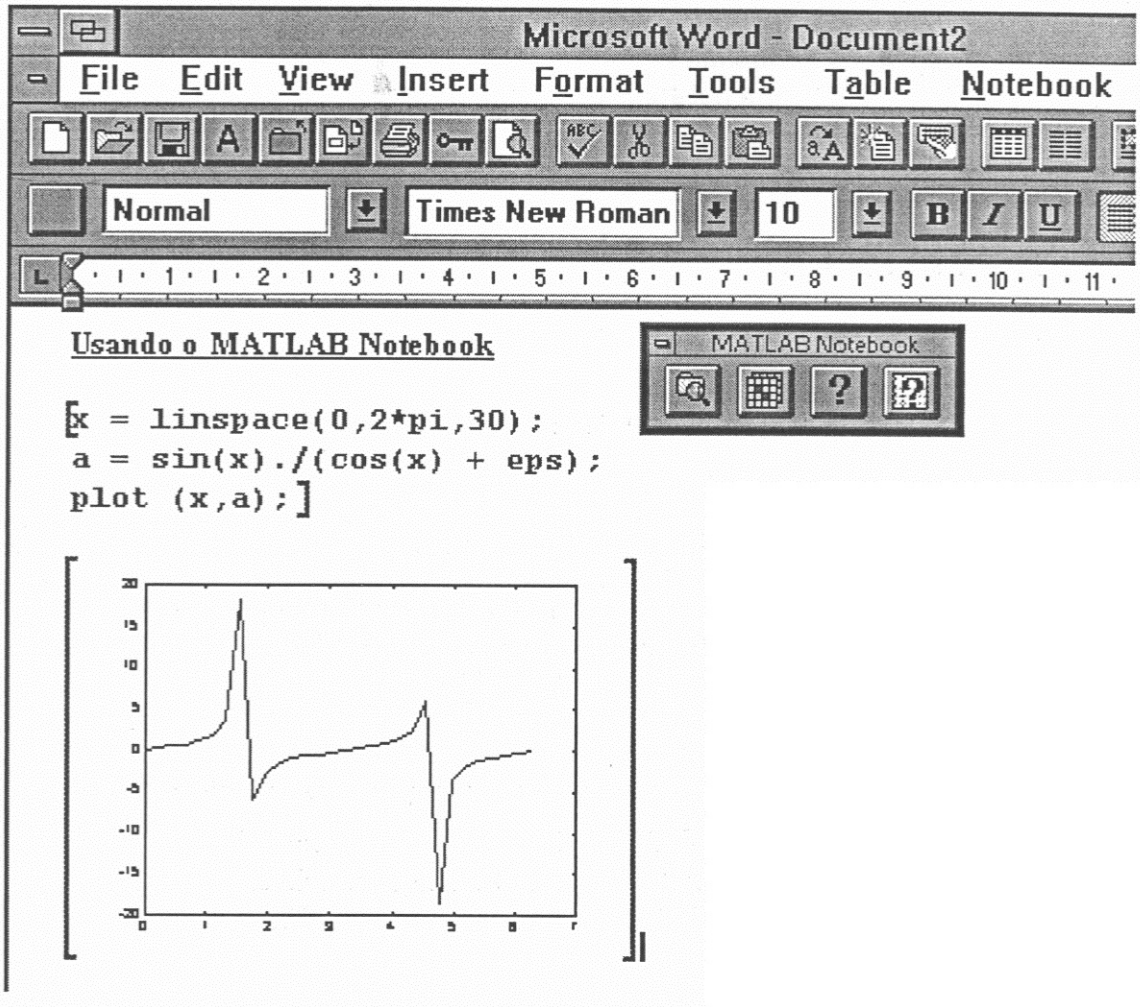


Figura 5.5: Configuração de impressora com PostScript



3. Escreva o texto e comandos que faram parte de seu documento.
4. Marcar o comando-MATLAB que deseja-se avaliar e imediatamente pressionar `<Ctrl><Enter>`. O texto-comando tomará a forma de uma expressão MATLAB como mostra o gráfico



5. Salvar e/ou imprimir.

Capítulo 6

Matrizes Esparsas

Matrizes esparsas ocorrem em inúmeras aplicações científicas, particularmente quando se utilizam discretizações (elementos finitos, diferenças finitas) de equações diferenciais parciais.

Nestes casos, as matrizes obtidas tem um grande número de elementos nulos. Se a matriz é grande (por exemplo, 10000 linhas e colunas) e tem apenas 15000 elementos diferentes de zero, estaremos disperdiçando uma enorme quantidade de memória (10000^2 elementos) para armazená-los. Além disso, provavelmente nem poderíamos trabalhar com tal matriz, pelo fato de exigirmos mais memória do que a disponível no computador que estamos utilizando.

Por isso, existem estruturas de dados que permitem armazenar apenas os elementos não-zeros de uma matriz. O MATLAB oferece funções para criar matrizes esparsas, converter de matrizes cheias ("full") para esparsas, e aplicar certas funções sobre matrizes esparsas. As operações matriciais vistas anteriormente para matrizes cheias também valem para matrizes esparsas.

Para criarmos uma matriz esparsa no MATLAB, é necessário termos no mínimo três informações: os índices (*linha* e *coluna*), e os valores não-nulos correspondentes. O comando MATLAB para se criar uma matriz esparsa seria então:

```
S = sparse(l,c,v,m,n,nzmax)
```

onde `S` é uma matriz esparsa; `l` e `c` são vetores contendo as linhas e colunas dos elementos não-nulos, e `v` contém esses elementos, de tal forma que

a posição (linha e coluna) do elemento `v(i)` está em `l(i)` e `c(i)` – note que `l`, `c` e `v` tem necessariamente o mesmo tamanho. Os parâmetros `m` e `n` indicam a dimensão $m \times n$ de `S`, e `nzmax` indica o número máximo de elementos não-zero que `S` pode armazenar. Note que nesse formato do comando, `nzmax`$\geq$`length(v)`.

Existem variações deste comando, tais como:

1. `S = sparse(A)`

   cria uma matriz esparsa `S` a partir a matriz cheia `A` (a qual deve possivelmente conter elementos nulos).

2. `S = sparse(l,c,v,m,n)`

   nesse caso, `nzmax=length(v)`.

3. `S = sparse(l,c,v)`

   nesse caso, `nzmax=length(v)`, `m=max(l)` e `n=max(c)`.

Para a manipulação de matrizes esparsas, existem várias funções, entre as quais destacamos:

1. `[l,c,v] = find(A)`

   obtém os vetores `l`, `c` e `v` conforme definidos anteriormente, inspecionando a matriz cheia `A`.

2. `[l,c,v] = find(A>100)`

   obtém os vetores `l`, `c` e `v` conforme definidos anteriormente, apenas para os elementos de `A` maiores do que `100`.

3. `A = full(S)`

   converte a matriz esparsa `S` na matriz cheia `A`.

4. `nz = nnz(A)`
   `nz = nnz(S)`

   obtém o número de elementos não-zeros de `A` ou `S`.

5. `v = nonzeros(A)`
   `v = nonzeros(S)`

   obtém os elementos não-zero de `A` ou `S`.

6. `S = speye(n)`

cria uma matriz esparsa S de dimensão n, com diagonal constante (= 1). Note que isso é equivalente a

```
A = eye(n)
S = sparse(A)
```

No exemplo que segue, criamos uma matriz cheia que contém alguns elementos nulos, obtemos a representação esparsa da mesma, e depois operamos com ambas as matrizes:

```
>> A=[4 -1 -1 0;-1 4 0 -1;-1 0 4 -1;0 -1 -1 4]

A =

     4    -1    -1     0
    -1     4     0    -1
    -1     0     4    -1
     0    -1    -1     4

>> [i,j,v]=find(A)

i =

     1
     2
     3
     1
     2
     4
     1
     3
     4
     2
     3
     4


j =
```

```
     1
     1
     1
     2
     2
     2
     3
     3
     3
     4
     4
     4


v =

     4
    -1
    -1
    -1
     4
    -1
    -1
     4
    -1
    -1
    -1
     4

>> S=sparse(i,j,v,4,4)

S =

   (1,1)        4
   (2,1)       -1
   (3,1)       -1
   (1,2)       -1
   (2,2)        4
   (4,2)       -1
   (1,3)       -1
   (3,3)        4
```

```
   (4,3)        -1
   (2,4)        -1
   (3,4)        -1
   (4,4)         4

>> Z=A-S

Z =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Maiores informações sobre estes comandos podem ser obtidos através de `help sparse` e `help sparfun`.

Capítulo 7

Algumas Aplicações

Neste capítulo são apresentados alguns programas em MATLAB relativos a álgebra matricial numérica e equações em diferenças.

7.1 Álgebra Matricial Numérica

Nesta seção são considerados três algoritmos numéricos para a resolução de sistemas lineares $\mathbf{Ax} = \mathbf{b}$. No primeiro, é utilizado o método da eliminação Gaussiana com pivotamento parcial, com $\mathbf{A}$ não-singular. Para o caso em que $\mathbf{A}$é simétrica e positiva definida, são apresentados os métodosde Cholesky e do gradiente conjugado .

7.1.1 Algoritmo de Gauss com Pivotamento Parcial

Dada uma matriz não singular $\mathbf{A}$*, de ordem* n*, e um vetor* $\mathbf{b}$*, de ordem* n*, determina-se as matrizes de permutação* $\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_{n-1}}$ *e as matrizes triangulares, com diagonal unitária* $\mathbf{L_1}, \mathbf{L_2}, ..., \mathbf{L_{n-1}}$*, tais que*

$$[\mathbf{U} \quad \mathbf{c}] = \mathbf{L_{n-1}P_{n-1}}....\mathbf{L_1P_1}[\mathbf{A} \quad \mathbf{b}].$$

O vetor $\mathbf{b}$ *é a* $(n+1)$*-coluna da matriz ampliada* $[\mathbf{A} \quad \mathbf{b}]$*. Os elementos da matriz* $\mathbf{U}$ *e do vetor* $\mathbf{c}$ *são armazenados sobre os correspondentes elementos da matriz* $\mathbf{A}$ *e do vetor* $\mathbf{b}$ *, respectivamente. A solução é obtida por*

retrosubstituição no sistema $\mathbf{Ux} = \mathbf{c}$ *e, sucessivamente, armazenada em* $\mathbf{c}$.

- *Algoritmo*

1. Para j = 1: n-1
   - 1.1. ipiv := j
   - 1.2. amax := $|a_{j,j}|$
   - 1.3. Para i =j+1:n
     - 1.3.1. Se $|a_{ij}| > amax$, então
       - 1.3.1.1. ipiv $\leftarrow$ i
       - 1.3.1.2. amax $\leftarrow |a_{ij}|$
   - 1.4. Se amax = 0, então pare.
   - 1.5. Se $ipiv > j$, então:
     - $a_{j,k} \leftrightarrow a_{ipiv,k}$ (k = 1: n+1)
   - 1.6. Para i = j+1: n
     - 1.6.1. $a_{ij} \leftarrow a_{ij} := a_{ij}/a_{j,j}$
     - 1.6.2. Para k = j+1:n+1
       - 1.6.2.1. $a_{i,k} \leftarrow a_{i,k} - a_{ij}a_{j,k}$
2. Se $a_{nn} = 0$, então: pare, senão
   - $a_{n,n+1} \leftarrow a_{n,n+1}/a_{nn}$
3. Para i = n-1:1
   - 3.1. $a_{i,n+1} \leftarrow (a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}a_{j,n+1})/aii$

- *Programa de Eliminação de Gauss com Pivotamento Parcial*

```
function x = gauss_p(A,b)
% x = gauss_p(A,b)
%     Calcula Elimina\c{c}\~ao de Gauss com Pivotamento
%
[m,n] = size(A);
a = [A b];

for j = 1:n-1
    ipiv = j;
    amax = abs( a(j,j) );
    for i = j+1 : n
        if abs( a(i,j) ) > amax
           ipiv = i;
           amax = abs( a(i,j) );
        end;
    end;
```

```
    if amax == 0
       break;
    end;
    if ipiv > j
       for k = 1: n+1
           x = a(ipiv , k);
           a(ipiv , k) = a(j,k);
           a(j,k) = x;
       end;
    end;
    for i = j+1:n
        lambda(i,j) = a(i,j) / a(j,j);
        a(i,j) = lambda(i,j);
        for k = j+1:1:n+1
            a(i,k) = a(i,k) - lambda(i,j) * a(j,k);
        end;
    end;
end;
if a(n,n) == 0
   break;
   else
   a(n,n+1) = a(n,n+1) / a(n,n);
end;
for i =n-1: -1 : 1
    s = 0;
    for j = i+1 : n
        s = s + a(i,j) * a(j,n+1);
    end;
    a(i,n+1) = ( a(i,n+1) - s )/ a(i,i);
end;
Matriz Ampliada = a
for z = 1:n
  x(z) = a(z,n+1);
end;
```

7.1.2 Algoritmo de Cholesky para uma Matriz Simétrica Positiva Definida

Dada uma matriz **A** *simétrica positiva definida, calcula-se a matriz triangular inferior* **L**, *com* $l_{ii} > 0$, *tal que* $\mathbf{A} = \mathbf{L}\mathbf{L}^t$. *O algoritmo utiliza o triângulo superior da matriz* **A**, *entretanto, no triângulo inferior são armazenados os correspondentes termos de* **L**, *com exceção dos termos diagonais* l_{ii} , *cujos recíprocos são armazenados no vetor auxiliar* **p**.

- *Algoritmo*

1. Para i = 1: n
    1.1. Para j = i: n
        1.1.1. $s := a_{ij}$
        1.1.2. $s \leftarrow s - \sum_{k=1}^{i-1} a_{ik}a_{jk}$
        1.1.3. Se $j = i$, então:
            Se $s < 0$
            então:
                pare.
            senão:
                $p_i := 1/sqrt\{s\}$
        senão:
            $a_{ij} := sp_i$

- *Programa do Algortimo de Cholesky*

```
function l = choles(a)
% l = choles(a)
%     Fatoracao de Cholesky de uma matriz simetrica
%     positiva definida (primeira versao).
%
[m,n] = size(a);
for i = 1:n
    for j = i:n
        s = a(i,j);
        r = 0;
        for k = 1:i-1
            r = r + a(i,k)*a(j,k);
        end;
        s = s - r;
        if j == i
```

```
            if s <= 0;
               break
               else
                   p(i) = 1/sqrt(s);
            end;
            else
                a(j,i) = s*p(i);
        end;
    end;
end
%               determina matriz inferior l
for i = 1:n
    p(i) = 1/ p(i);
end
l = diag(p);
for j=1:n
  for i=j+1:n
   l(i,j) = a(i,j);
  end
end
```

7.1.3 Algoritmo Gradiente Conjugado para Matriz Simétrica Definida Positiva

Dada a matriz **A**, *simétrica positiva definida de ordem n, e o vetor n-dimensional* **b** , *o algoritmo calcula a solução do sistema* $\mathbf{Ax} = \mathbf{b}$. *Além dos vetores* **b** *e* **x**, *são utilizados, os vetores auxiliares* **p** ,**r** *e* **w**.

- *Algoritmo*

  $r_0 = b - A\ x_0$
  $\rho = r_0^T r_0$
  $p_0 = r_0$
  for $k = 1 : maxit$
  $w_{k-1} = A\ p_{k-1}$
  $\alpha_{k-1} = \rho_{k-1}/p_{k-1}^T w_{k-1}$
  $x_k = x_{k-1} + \alpha_{k-1}\ p_{k-1}$
  $r_k = r_{k-1} - \alpha_{k-1}\ w_{k-1}$
  se $\| r_k \|_2 < \epsilon \| b \|_2$
  então:

pare

senão:

$$\rho_k = r_k^T r_k$$
$$\beta_k = \rho_k / \rho_{k-1}$$
$$p_k = r_k + \beta_k p_{k-1}$$

fim.

- *Código-MATLAB para o Algoritmo Gradiente Conjugado*

```
function [sol,acc,k]=cg(A,b,x,tol,maxit)
% [sol,acc,it]=cg(A,b,x,tol,maxit)
%
%  Calcula a solucao do sistema de equacoes lineares Ax=b
%  usando o metodo CG. A norma da solucao e' menor ou igual
%  a tol*||b||_{2} e "maxit" e'  o numero maximo de iteracoes.

  [n,n]=size(A);

  r=b-A*x;
  rnorm=norm(r,'fro');
  bnorm=norm(b,'fro');
  epsbnorm=tol*bnorm;

  rho=r'*r;
  p=r;

  for k=1:maxit
    w=A*p;
    alpha=rho/(p'*w);
    x=x+alpha*p;
    r=r-alpha*w;

    rnorm=norm(r,'fro');
    fprintf('%g %g\n',k,rnorm);
    if (rnorm <= epsbnorm) break; end

    rho0=rho;
    rho=r'*r;
    beta=rho/rho0;
    p=r+beta*p;
  end
```

```
sol=x;
acc=norm(b-A*x,'fro');
```

7.2 Equações em Diferenças

A seguir, apresentamos programas para equações lineares em diferenças. São ilustradas graficamente soluções de problemas com valores iniciais e da solução dinâmica para equações de segunda ordem, esta última com as condições u(0)=0 e u(1)=1. É observado, para certos valores dos parâmetros, o fenomeno da estabilidade assintótica (decaimento) e da instabilidade (crescimento ilimitado) das soluções é observado. Os programas consistem de um programa principal, uma demonstração e a função principal.

7.2.1 Equação em Diferenças de Primeira Ordem

- *O Programa Principal*

```
% Arquivo  main.m
%
resp = 'S';
while (resp=='s')|(resp=='S'),
  clc;
  disp(' ');
  disp('  EQUACOES EM DIFERENCAS DE PRIMEIRA ORDEM ');
  disp('  ========================================');
  disp(' Equacao:');
  disp('        u(k+1) = a*u(k) + b,   k = 1:n,');
  disp('           u(0) = u0');
  disp('                    ');
  disp(' Opcoes:            ');
  disp('        1. Demonstracao                    ');
  disp('        2. Equacao do usuario [a  b  n  u0]');
  disp('                                           ');
  disp('        0. Fim                             ');
  option = input(' Escolha uma opcao : ');
  if (option < 1)|(option > 2),
     break;
  end;
  if option == 1
     demodif1;
  else
```

```
    disp(' ');
    disp(' Entre com os valores  para  a, b, n, u0  ');
    u = input('(Sugestao: [-1.05   0.1  40  0] )  :  ');
    edif1(u(1), u(2), u(3), u(4));
   end;
   resp = input('  Continua? (S/N) : ','s');
end;
clc;
%------------------- end main1.m -------------------------
```

- *Uma Demonstração*

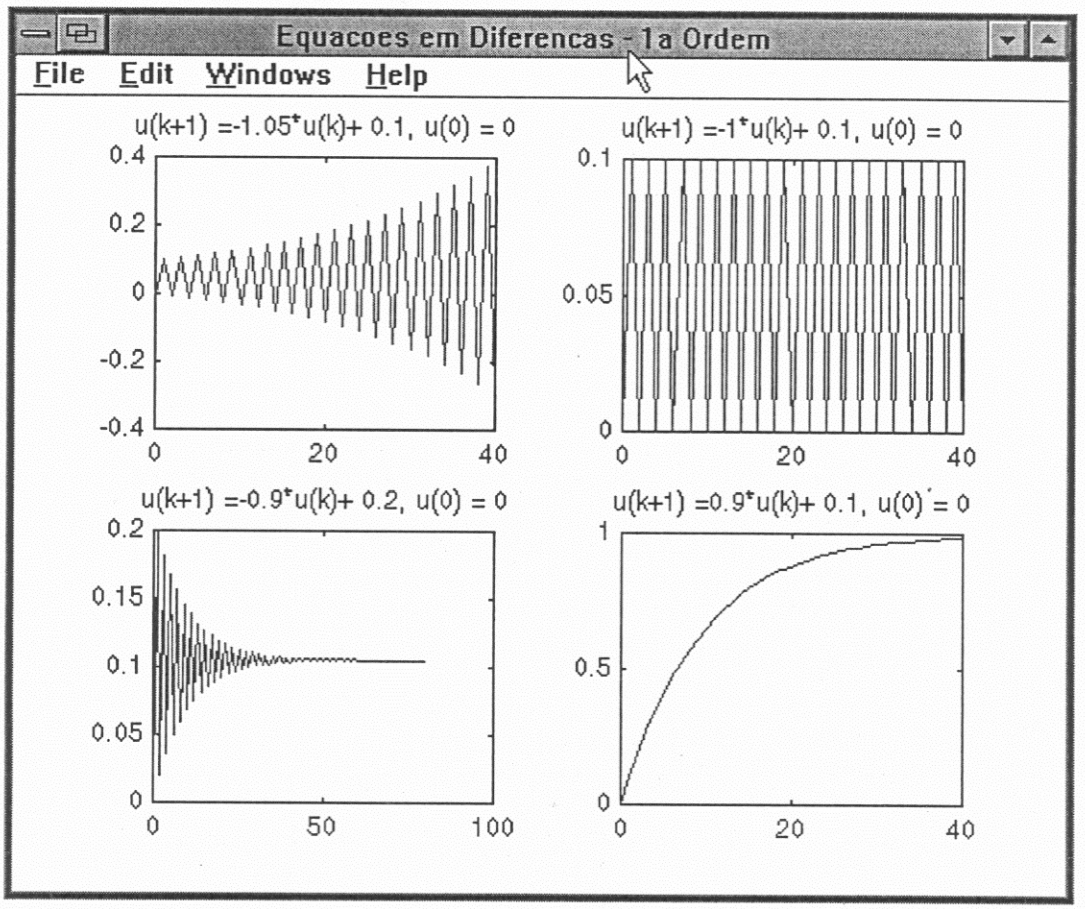


Figura 7.1: Exemplo de Equações em Diferenças - Primeira Ordem

```
% Arquivo demodif1.m
%
%     Este programa mostra os graficos da equacao em
%     diferencas:
```

```
%         u(k+1) = a*u(k) + b,     k = 1:n
%         u(0) = u0
%      para distintos valores de "a", "b", "u0" e "n":
%
clear; clc;
echo on;
   a =  [-1.05  -1      -0.9    0.9     1       1.1];
   b =  [ 0.1    0.1     0.2    0.1     1.5     0.5];
  u0 =  [ 0      0       0      0       5       0  ];
   n =  [40     40      80     40      40      40  ];
echo off;
disp('Pressione qualquer tecla ...');
disp('  ');
disp('Para cada valor de "a" geramos "n" valores u(k):');
disp('  ');
    for numeq = 1:length(a),
        u(numeq,1) = u0(numeq);
        for i=1:n(numeq),
            u(numeq,i+1) = a(numeq)*u(numeq,i) + b(numeq);
        end;
    end;

disp('  Graficos.  Pressione uma tecla');
disp('  ');
pause;

%%-------------------- 1o.  grafico ---------------------
  tit = 'u(k+1) =%g*u(k)+ %g, u(0) = %g' ;
  subplot(221),
  plot((0:n(1)),u(1,1:n(1)+1))
  title(sprintf(tit,a(1),b(1),u0(1)))
  subplot(222),
  plot((0:n(2)),u(2,1:n(2)+1))
  title(sprintf(tit,a(2),b(2),u0(2)));
  subplot(223),
  plot((0:n(3)),u(3,1:n(3)+1))
  title(sprintf(tit,a(3),b(3),u0(3)))
  subplot(224),
  plot((0:n(4)),u(4,1:n(4)+1))
  title(sprintf(tit,a(4),b(4),u0(4)));
  set(gcf, 'Name', 'Equacoes em Diferencas');

%%%-------------------- 2o. grafico --------------
  tit = 'u(k+1) =%g*u(k)+ %g, u(0) = %g' ;
  figure;
  subplot(211),
  plot((0:n(5)),u(5,1:n(5)+1))
  title(sprintf(tit,a(5),b(5),u0(5)))
  subplot(212),
```

```
  plot((0:n(6)),u(6,1:n(6)+1))
  title(sprintf(tit,a(6),b(6),u0(6)));
  set(gcf, 'Name', 'Equacoes em Diferencas');

  return
%%%%---------------- fim demodif1.m -----------------
```

- *A Função Principal*

```
%   Arquivo  edif1.m
%
function edif1(a,b,n,u0);
%

u(1) = u0;
for i = 1:n,
    u(i+1) = a*u(i)+b;
end;
clg;
plot((0:n),u)
t = 'Equacao u(k+1) = %g*u(k)+%g, k = 1:%g';
title(sprintf(t,a,b,n));
xlabel('k');
ylabel('u(k)');
set(gcf,'Name','Equacoes em Diferencas 1a Ordem',...
                  'NumberTitle','off');
return;
%---------------------- end edif.m --------------------
```

7.2.2 Equação em Diferenças de Segunda Ordem

- *O Programa Principal*

```
% Arquivo main2.m
resp = 'S';
while (resp=='s')|(resp=='S'),
  clc;
  disp(' ');
  disp('  EQUACOES EM DIFERENCAS DE SEGUNDA ORDEM');
  disp('  ========================================');
  disp('  Equacao:');
  disp('       u(k) + b*u(k-1) + c*u(k-2) = f,    k = 2:n');
  disp('             u(0) = u0;      u(1) = u1');
  disp('                                        ');
  disp('  Opcoes:');
  disp('        1. Demonstracao');
  disp('        2. Demonstracao - SOLUCAO DINAMICA');
```

```
  disp('        3. Equacao do usuario   [b c f  u0  u1  n]');
  disp('                                                   ');
  disp('        0. Fim');
  option = input('  Escolha uma opcao :  ');
  if (option < 1)|(option > 3 ),
      break;
  end;
  if option == 1
      demodif2;
  elseif option == 2
      sol_din;
  else
      disp(' ');
      disp('  Entre com os valores  para  b, c, f, u0, u1, n  ');
      disp(' ');
      u = input('(Sugestao: [1.45   0.475  2  -1  1 140] )  :  ');
      edif2(u(1),u(2),u(3),u(4),u(5),u(6));
  end;
  resp = input('  Continua ? (S/N) : ','s');
end;
%----------------  end main2.m -------------------------
```

- *Uma Demonstração*

```
% Arquivo: demodif2.m
 clc;
 disp(' ');
 disp('  EQUACOES EM DIFERENCAS DE SEGUNDA ORDEM ');
 disp('  ==========================================');
 disp(' Equacao:');
 disp('        u(k) + b*u(k-1) + c*u(k-2) = f,    k = 2:n');
 disp('              u(0) = u0;      u(1) = u1');
 disp(' para distintos valores de "b", "c", "f", "u0", "u1" e "n" ');
 disp(' ');
clear u;
echo on;
 b = [  1.45  0.05  0.11 0.1  4/sqrt(5) 0.01];
 c = [  0.475  -1  -1.089  -0.9  1 -1.01];
 f =  [ 2 2  2  -2  1   1 ];
 u0 = [ -1 1 -1  2  1  -50   ];
 u1 = [  1 -1  1   -2  -1  50   ];
 n =  [140  80 80 100 200 140  ];
echo off;
% ----- Gerando  n  valores  u(k)  ----------------
for numeq = 1:length(b),
  u(numeq,1) = u0(numeq);
  u(numeq,2) = u1(numeq);
  for k = 1: n(numeq)-1,
```

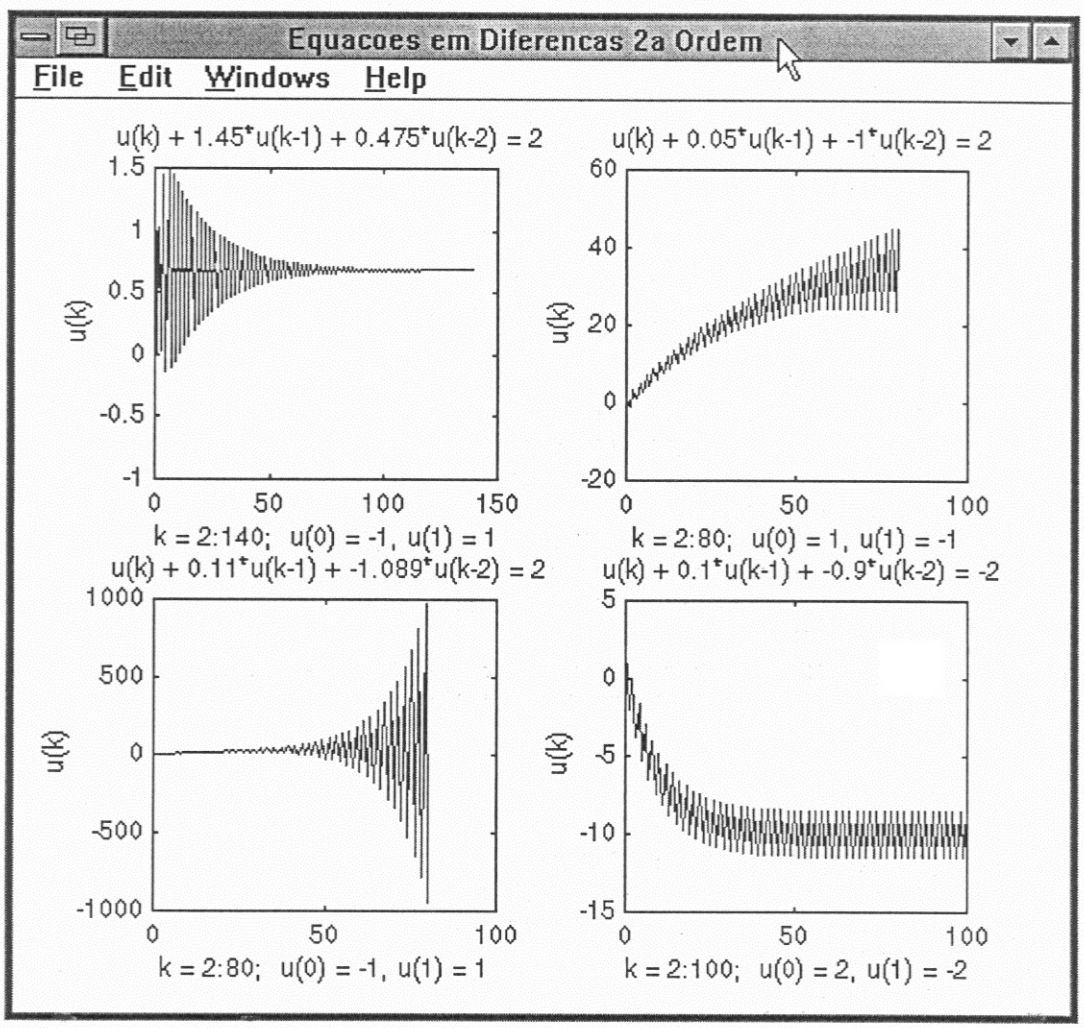


Figura 7.2: Exemplo de Equações em Diferenças - Segunda Ordem

```
      u(numeq,k+2) = f(numeq)-b(numeq)*u(numeq,k+1) ...
                     - c(numeq)*u(numeq,k);
  end;
end;
disp('Graficos. - Pressione qualquer tecla ...');
pause
  t1 = ' u(k) + %g*u(k-1) + %g*u(k-2) = %g';
  t2 = ' k = 2:%g;  u(0) = %g, u(1) = %g ';
  subplot(221),
  plot((0:n(1)),u(1,1:n(1)+1))
  title(sprintf(t1,b(1),c(1),f(1)));
  xlabel(sprintf(t2,n(1),u0(1),u1(1)));
  ylabel('u(k)');
  subplot(222),
  plot((0:n(2)),u(2,1:n(2)+1))
  title(sprintf(t1,b(2),c(2),f(2)));
  xlabel(sprintf(t2,n(2),u0(2),u1(2)));
  ylabel('u(k)');
  subplot(223),
  plot((0:n(3)),u(3,1:n(3)+1))
  title(sprintf(t1,b(3),c(3),f(3)));
  xlabel(sprintf(t2,n(3),u0(3),u1(3)));
  ylabel('u(k)');
  subplot(224), plot((0:n(4)),u(4,1:n(4)+1))
  title(sprintf(t1,b(4),c(4),f(4)));
  xlabel(sprintf(t2,n(4),u0(4),u1(4)));
  ylabel('u(k)');
  set(gcf,'Name','Equacoes em Diferencas 2a Ordem',...
                 'NumberTitle','off');
  pause;

  figure;
  subplot(211);
  plot((0:n(5)),u(5,1:n(5)+1));
  title(sprintf(t1,b(5),c(5),f(5)));
  xlabel(sprintf(t2,n(5),u0(5),u1(5)));
  ylabel('u(k)');
  subplot(212);
  plot((0:n(6)),u(6,1:n(6)+1)) ;
  title(sprintf(t1,b(6),c(6),f(6)));
  xlabel(sprintf(t2,n(6),u0(6),u1(6)));
  ylabel('u(k)');
  set(gcf,'Name','Equacoes em Diferencas 2a Ordem',...
           'NumberTitle','off');
  pause;
return;
% ----------------- end demodif2.m -------------------------
```

- *A Solução Dinâmica*

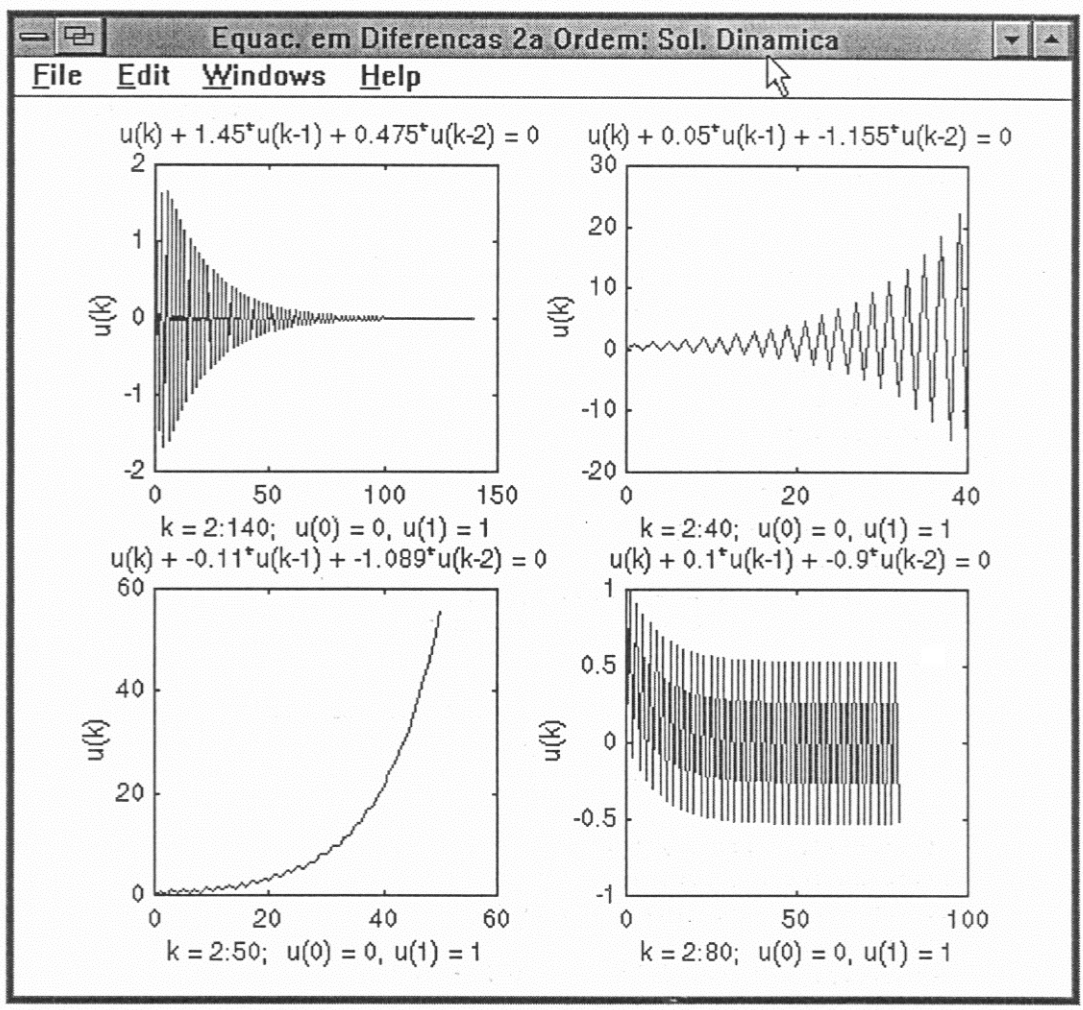


Figura 7.3: Exemplo de Solução Dinâmica

```
% Arquivo  Sol_din.m
 clc;
 disp(' ');
 disp('  EQUACOES EM DIFERENCAS DE SEGUNDA ORDEM ');
 disp('  ========================================');
 disp(' Equacao:');
 disp('       u(k) + b*u(k-1) + c*u(k-2) = 0,    k = 2:n');
 disp('             u(0) = 0;      u(1) = 1');
 disp(' para distintos valores de "b", "c"  e "n" ');
 disp(' ');
 disp(' A solucao e chamada de "S O L U C A O   D I N A M I C A');
 disp(' ');
 clear u;
 echo on;
  b =  [1.45  0.05 -0.11  0.1  2  -2.05];
  c =  [0.475 -1.155 -1.089  -0.9  1  1.05];
  n =  [140 40 50  80 80 100];
echo off;
f = zeros(1,6);
u0 = zeros(1,6);
u1 = ones(1,6);
for numeq = 1:length(b),
  u(numeq,1) = u0(numeq);
  u(numeq,2) = u1(numeq);
  for k=1:n(numeq)-1,
   u(numeq,k+2) = f(numeq)-b(numeq)*u(numeq,k+1) ...
                  - c(numeq)*u(numeq,k);
  end;
end;
%%----------------- Graficos -----------------------
t1 = ' u(k) + %g*u(k-1) + %g*u(k-2) = %g';
t2 = 'k = 2:%g;  u(0) = %g, u(1) = %g' ;
subplot(221),
plot((0:n(1)),u(1,1:n(1)+1))
title(sprintf(t1,b(1),c(1),f(1)));
xlabel(sprintf(t2,n(1),u0(1),u1(1)));
ylabel('u(k)');
subplot(222), plot((0:n(2)),u(2,1:n(2)+1))
title(sprintf(t1,b(2),c(2),f(2)));
xlabel(sprintf(t2,n(2),u0(2),u1(2)));
ylabel('u(k)');
subplot(223),
plot((0:n(3)),u(3,1:n(3)+1))
title(sprintf(t1,b(3),c(3),f(3)));
xlabel(sprintf(t2,n(3),u0(3),u1(3)));
ylabel('u(k)');
subplot(224),
plot((0:n(4)),u(4,1:n(4)+1))
title(sprintf(t1,b(4),c(4),f(4)));
```

```
xlabel(sprintf(t2,n(4),u0(4),u1(4)));
ylabel('u(k)');
set(gcf,'Name','Equac. em Diferencas 2a Ordem: Sol. Dinamica',...
                'NumberTitle','off');

figure;
subplot(211),
plot((0:n(5)),u(5,1:n(5)+1))
title(sprintf(t1,b(5),c(5),f(5)));
xlabel(sprintf(t2,n(5),u0(5),u1(5)));
ylabel('u(k)');
subplot(212),
plot((0:n(6)),u(6,1:n(6)+1))
title(sprintf(t1,b(6),c(6),f(6)));
xlabel(sprintf(t2,n(6),u0(6),u1(6)));
ylabel('u(k)');
set(gcf,'Name','Equac. em Diferencas 2a Ordem: Sol. Dinamica',...
                'NumberTitle','off');
pause;
return;
% -------------------- end  sol_din.m ---------------------------
```

- *A Função Principal*

```
% Arquivo edif2.m
function edif2(b,c,f,u0,u1,n)
%
u(1) = u0; u(2) = u1;
for k = 1:n-1,
    u(k+2) = f - b*u(k+1) - c*u(k);
end;
%                 Grafico
plot((0:n),u)
title(sprintf('u(k) + %g*u(k-1) + %g*u(k-2) = %g',b,c,f));
xlabel(sprintf('k = 2:%g;  u(0) = %g, u(1) = %g',n,u0,u1));
ylabel('u(k)');
set(gcf,'Name','Equacao em Diferencas 2a Ordem',...
                  'NumberTitle','off');
return;
%---------------------- fim edif2.m -------------------------
```

Capítulo 8

Menus e Controles

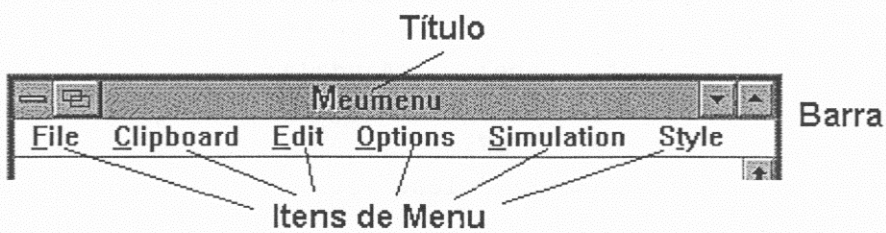
Uma das características do MATLAB é a de poder ser estendido usando suas funções internas para construir outras ferramentas e aplicações. Neste capítulo descreveremos rapidamente como construir *menus* e *controles* usando a GUI (Graphical User Interface). Os elementos da GUI são de duas classes:

- Menus e
- Controles.

8.1 Como criar menus

Menus "pull-down"permitem a usuários de uma aplicação-MATLAB fazer escolhas de um conjunto de opções. Um menu esta formado por:

- Um título.
- Uma barra de menu contendo os nomes dos itens de menu disponíveis.
- Os itens de menu.



Um menu é criado usando a função de criação de objetos-menu, **uimenu**. Visto que, o menu deve ser visualizado através de uma janela gráfica, então primeiro deve-se criar esta janela através do comando **figure**, especificando o nome da janela, a posição na tela, a forma do cursor (seta, cruz, círculo, etc.) e outras propriedades. Para listar todas as propriedades do objeto `figure`, executar o comando `get(gcf)`.

```
figure('Name','Meu Menu Principal', ... % Titulo do Menu
       'NumberTitle','off', ...
       'Visible','on', ...
       'Resize','on', ...
       'MenuBar','none', ...
       'Colormap',[], ...
       'Position',[100, 200,   300,  200],...
                %[left,bottom,width,height]
       'Pointer','arrow' );  %% cross, circle, fleur
```

A sintaxe de um menu (nível máximo, menu horizontal) é :

MenuItemX = uimenu(handle, *'NomePropriedade1','ValorPropriedade1', ...);*

Uma segunda forma é usada para criar itens de menu (menus verticais) a partir do menu anterior *MenuItemX* :

OpcaoY = uimenu(MenuItemX, *'NomePropriedade1','ValorPropriedade1', ...);*

Para controlar como os menus serão mostrados na tela e para determinar que ações serão executadas quando selecionados, os menus usam propriedades. Como pode-se observar na sintaxe, os menus são definidos especificando *propriedades* e *valores* de propriedades. Por exemplo,

```
SubMenu1 = uimenu(gcf, 'Label', 'SubMenu&1');
SubMenu2 = uimenu(gcf, 'Label', 'SubMenu&2');
SubMenu3 = uimenu(gcf, 'Label', '&Ajuda');
SubMenu4 = uimenu(gcf, ...
                 'Label', '&Fim',...
                 'Callback','close');
```

As propriedades de um menu e seus respectivos valores podem ser listados usando o comando `get`(handle_do_menu). As propriedades mais importantes são:

Label: Define o título do menu ou o nome do item do menu.

CallBack: Especifica a ação executada quando o item de menu é selecionado. Pode ser o nome de um arquivo de comandos ou um conjunto de comandos.

Parent: É geralmente o primeiro argumento do comando *uimenu*. Identifica o 'handle' do objeto pai do menu. O pai de um menu é a figura onde o menu aparece. O pai de um item de menu ou submenu é o menu onde o item aparece.

Separator: Especifica que uma linha horizontal separa o item de outro item imediatamente acima de este. Separadores são usados geralmente para agrupar itens relacionados entre si. Os valores são `on` e `off`.

8.2 Exemplo de Menu

A seguir é apresentado um exemplo completo de um menu 'Principal' (horizontal) com quatro itens:

- 'SubMenu1',
- 'Submenu2',
- 'Gravar'
- 'Fim'.

O submenu 'SubMenu1' é um menu 'secundário' (vertical) com seis itens:

- 'Soma de Matrizes',
- 'Produto de Numeros',
- 'Divide',
- 'Substracao',
- 'Escreve meu nome'
- 'Figura'.

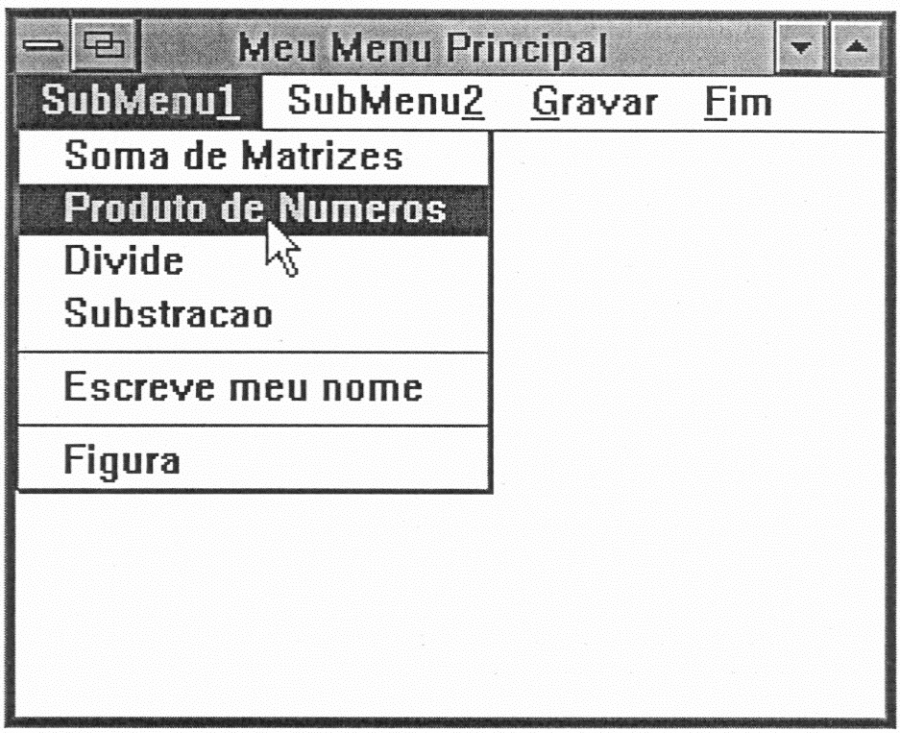
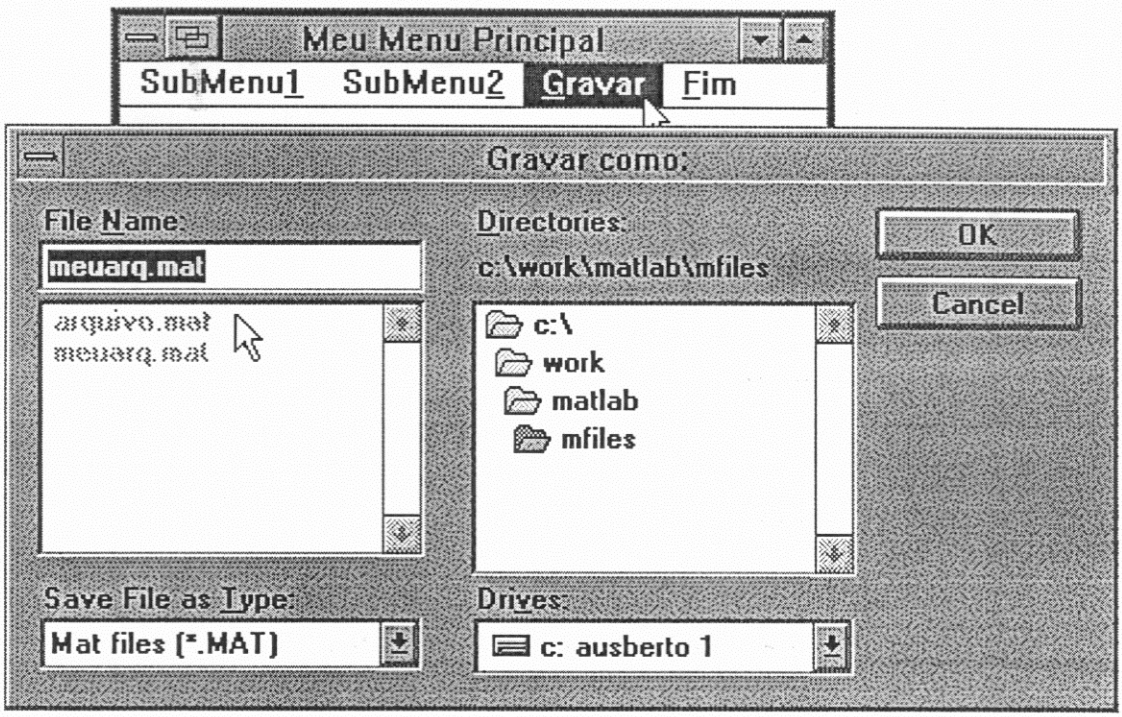


Figura 8.1: Exemplo de Menu no MATLAB

```
% Arquivo: fazmenu.m
%
%%%%   Menu Principal
%%%%   ==============
clear; clc;
%    Cria Janela para o menu, usando o comando "figure"
figNum = figure( ...
         'Name','Meu Menu Principal', ...
         'NumberTitle','off', ...
         'Visible','on', ...
         'Resize','on', ...
         'MenuBar','none', ...
         'Colormap',[], ...
         'Position',[100,200, 300, 200],...
         'Pointer','arrow' ...  %% cross, circle,fleur
                );
%%%%%  Opcoes do Menu Principal (SubMenus)
%%%%%  ==================================
% Item1: Submenu de Opcoes 1
```

```
% Item2: Submenu de Opcoes 2  (Nao implementado)
% Item3: Submenu de Ajuda     (Nao implementado)
% Item4: Submenu de Fechar Menu (Janela)
%
SubMenu1 = uimenu(gcf, 'Label', 'SubMenu&1');
SubMenu2 = uimenu(gcf, 'Label', 'SubMenu&2');
SubMenu3 = uimenu(gcf,...
         'Label', '&Gravar',...
         'CallBack',[...
         '[fn,pn]=uiputfile(''MeuArq.mat'',''Gravar como:'');',...
         'save (fn)']...
                 );
SubMenu4 = uimenu(gcf, ...
                 'Label', '&Fim',...
                 'Callback','close');


%%%%   Sub-Menu1 (Item 1 do Menu Principal)
%%%%   -----------------------------------
%%%%
M1op1 = uimenu(SubMenu1,...
              'Label', 'Soma de Matrizes',...
              'CallBack','Soma = arq1');
M1op2 = uimenu(SubMenu1, ...
              'Label', 'Produto de Numeros',...
              'CallBack','Produto = arq2');
M1op3 = uimenu(SubMenu1, ...
              'Label', 'Divide',...
              'CallBack','arq3');
M1op4 = uimenu(SubMenu1, ...
              'Label', 'Substracao',...
              'CallBack','arq4');
M1op5 = uimenu(SubMenu1, ...
              'Label', 'Escreve meu nome',...
              'Separator','on',...
              'CallBack','arq5');
M1op6 = uimenu(SubMenu1, ...
              'Label', 'Figura',...
              'Separator','on', ...
              'CallBack',[...
                        'x=linspace(-2*pi,2*pi,60);',...
                        'y=sin(x).^2./(x+eps);',...
                        'plot(x,y)'] );
```



Arquivos usados pela opção `SubMenu1`:

```
% arq1.m
 function x =arq1()

 a = [ 1 2 3; 4 5 6; 7 8 9];
 b = [10 20 30; 40 50 60; 70 80 90];
 x = a + b;
 return;
%%-----------------------
% arq2.m
 function x =arq2()
   x = 5 * 10;
   return;
%%-----------------------
% arq3.m
 function x =arq3()
   x = 5 / 10
   return;
%%-----------------------
```

```
% arq4.m
 function x =arq4()
   x = 5 - 10
   return;
%%-----------------------
% arq5.m
 function x =arq5()
   x = 'Meu nome:  Joao Fulano';
   return;
%%-----------------------
```

8.3 Controles

Controles são objetos gráficos que quando manipulados com o mouse, produzem uma ação a ser executada. Um controle é criado usando a função de criação de objetos controle, **uicontrol**:

h = uicontrol(handle, 'NomePropriedade1','ValorPropriedade1', ...);

Da mesma forma que os menus, para criar controles MATLAB, recomenda-se primeiro criar uma janela gráfica através do comando **figure**, especificando o nome da janela, a posição na tela, a forma do cursor (seta, cruz, círculo, etc.) e outras propriedades.

Para incluir um controle numa janela grafica definida, deve-se especificar o 'handle' como o primeiro argumento do comando **uicontrol**.

Um controle é definido especificando propriedades e valores e todos os controles usam as mesmas propriedades. Para determinar os valores das propriedades de um controle *h*, usa-se o comando *get(h)*.

As propriedades mais importantes de um controle efetivo (que realiza alguma ação) são: o estilo ('Style'), o nome ('String'), a posição na tela ('Position', [left, bottom, width, height]) e a ação que o MATLAB executará ('Callback').

Os controles MATLAB são de oito tipos (ver Fig. 8.2):

Push Buttons: São pequenos objetos de tela geralmente rotulados com texto. Clicando o mouse sobre um botão 'push', o MATLAB executa uma ação definida. São usados para textos como: 'OK', 'ESC', 'Cancelar',

'Salvar', etc.

As principais propriedades de botões 'push' são o estilo e o rotulo:

```
'Style', 'push',...
'String','OK',...
```

Check Boxes: Permitem selecionar uma ou mais escolhas de um conjunto de alternativas. Cada alternativa tem dois estados: `on` (selecionado) e `off` (não selecionado). O indicador gráfico é um quadrado marcado com uma X para o estado `on` e um quadrado vazio para `off`.

As propriedades mais importantes deste controle é o estilo, o nome e o valor (1 = `on`, 0 = `off`):

```
'Style', 'checkbox',...
'String','Alternativa 1',...
'Value', 1,...
```

Radio Buttons: Permite fazer escolhas de alternativas mutuamente exclusivas. Cada alternativa tem dois estados: `on` (selecionado) e `off` (não selecionado). O indicador gráfico é um círculo marcado com • para o estado `on` e um círculo vazio para `off`. 'Radio buttons' aparecem em grupos, onde somente um 'button' pode ser selecionado por vez. Grupos diferentes de 'radio buttons' devem aparecer separados fisicamente

As propriedades mais importantes deste controle é o estilo, o nome e o valor (1 = `on`, 0 = `off`):

```
'Style', 'radio',...
'String','Opcao 1',...
'Value', 1,...
```

Sliders: Permitem fazer a escolha de um valor x dentro de um intervalo de valores $[a, b]$. Esta escolha é feita movimentando um indicador entre o valor mínimo a e o valor máximo b. 'Sliders' geralmente são acompanhados de textos estáticos que indicam o nome e os valores extremos. Neste controle não é usado a propriedade 'String'.

As propriedades mais importantes dos 'sliders' são o estilo, os valores extremos e o valor default:

```
'Style', 'slider',...
'Min', -10,...
'Max', 10,...
'Value', 7,...
```

Pop-up Menus: São usados para escolher uma item de uma lista. A lista de itens é especificada usando a propriedade 'String', separando itens

individuais com uma barra vertical (|) e considerados todos como um único string: `'item1|item2| ...|itemN'`. Um menu 'pop-up' no aberto mostra unicamente a escolha default. A propriedade 'Value' contém o índice do item selecionado.

As propriedades mais importantes dos menus 'pop-up' são o estilo, os nomes dos itens e o valor (item) default:

```
'Style', 'popup',...
'String',' Item1 | Item 2 | Item 3',...
'Value', 2,...
```

Static Text: É mostrado como uma única linha de informação textual e é usado geralmente para rotular um grupo de outros controles. Texto estático não pode ser modificado pelo usuário. A propriedade 'Units = normalized', permite que o controle seja visível quando o tamanho da janela gráfica for reduzido.

```
'Style', 'text',...
'String','Principais Metodos:',...
'Units', 'normalized',...
```

Editable Text: Facilitam a entrada de uma o mais linhas de texto. Pode-se aceitar, editar, marcar, apagar ou substituir um valor de texto editável. A propriedade 'Max' deve ser maior que 1.

```
'Style','edit', ...
'String','EDITABLE|TEXT|Editar|.estas|..linhas', ...
'BackGroundColor', [0.8 0.8 0.8], ...
'Max', 2, ...
'Position', [140 40 70 80]);
```

Frames: São regiões dentro de uma janela gráfica e são usados para colocar grupos de outros controles dentro de cada região simplificando as apresentações de uma GUI através de uma cor para cada frame. Não é usado em 'frames' a propriedade 'String'.

```
'Style','frame', ...
'BackGroundColor', [0.8 0.8 0.8], ...
'Position', [140 40 70 80]);
```

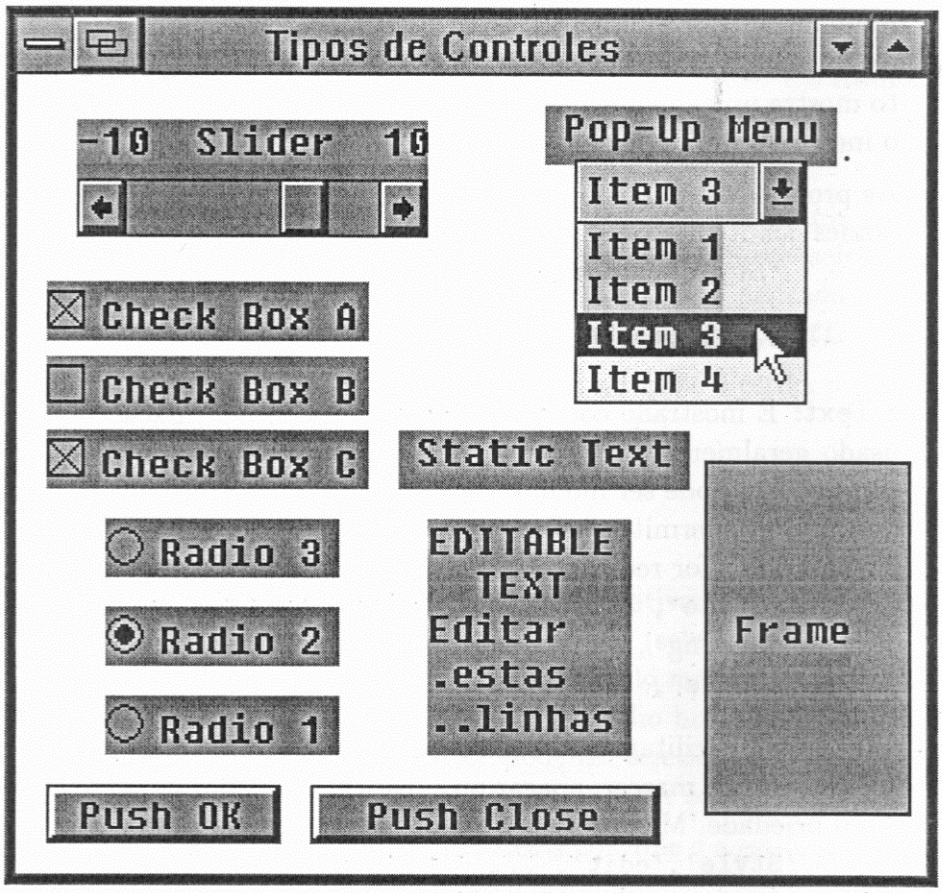


Figura 8.2: Tipos de Controles MATLAB

```
%%%
%%% Arquivo:  allctl.m
%%%
figNum=figure( ...
        'Name','Tipos de Controles', ...
        'NumberTitle','off', ...
        'Visible','on', ...
        'Resize','on', ...
        'MenuBar','none', ...
        'Color', [1 1 1], ...
        'Colormap',[0.5 0.5 0.5], ...
        'Position',[250,100, 310, 270],...
        'Pointer','arrow' ...  %% cross, circle,fleur
             );
%%%%%%%%
sttxt = uicontrol(gcf, ...
        'Style','text', ...
```

```
        'String','Static Text', ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [130 130 100 20]);
%%%%%%%%%
frtxt = uicontrol(gcf, ...
        'Style','text', ...
        'String','Frame',...
        'BackGroundColor',[0.8 0.8 0.8],...
        'Position',[245 70 40 20]...
                 );
myframe = uicontrol(gcf, ...
        'Style','frame', ...
        'BackGroundColor',[0.8 0.8 0.8],...
        'Position',[235 20 70 120]...
                 );
%%%%%%%%%
edittxt = uicontrol(gcf, ...
        'Style','edit', ...
        'String','EDITABLE|  TEXT|Editar|.estas|..linhas', ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Max', 2, ...
        'Position', [140 40 70 80]);
%%%%%%%%%
poptxt = uicontrol(gcf, ...
        'Style','text', ...
        'String','Pop-Up Menu', ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [180 240 100 20]);
popMenu = uicontrol(gcf, ...
        'Style','popup', ...
        'String','Item 1|Item 2|Item 3|Item 4', ...
        'Value', 3, ...
        'BackGroundColor', [0.9 0.9 0.9], ...
        'Position', [190 220 80 40]);
%%%%%%%%
slidtxt = uicontrol(gcf, ...
        'Style','text', ...
        'String','-10  Slider  10', ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [20 235 120 20]);
slid = uicontrol(gcf, ...
        'Style','slider', ...
        'Min', -10, 'Max',10, 'Value',5, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [20 215 120 20]);
%%%%%%%%%
chbox(3) = uicontrol(gcf, ...
        'Style','checkbox', ...
        'String','Check Box A', ...
```

```
        'Value', 1, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [10 180 110 20]);
chbox(2) = uicontrol(gcf, ...
        'Style','checkbox', ...
        'String','Check Box B', ...
        'Value', 0, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [10 155 110 20]);
chbox(1) = uicontrol(gcf, ...
        'Style','checkbox', ...
        'String','Check Box C', ...
        'Value', 1, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [10 130 110 20]);
%%%%%%%%%
fact(3)  = uicontrol(gcf, ...
        'Style','radio', ...
        'String', 'Radio 3', ...
        'Value', 0, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [30 100 80 20]);
fact(2)  = uicontrol(gcf, ...
        'Style','radio', ...
        'String', 'Radio 2', ...
        'Value', 1, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [30 70 80 20]);
fact(1)  = uicontrol(gcf, ...
        'Style','radio', ...
        'String', 'Radio 1', ...
        'Value', 0, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [30 40 80 20]);
%%%%%%%%
fecha = uicontrol(gcf, ...
        'Style','push', ...
        'String','Push Close', ...
        'Position', [100 10 120 20],...
        'CallBack', [...
                    'close(gcf),', ...
                    'clc;']);
start = uicontrol(gcf, ...
        'Style','push', ...
        'String','Push OK',...
        'Position', [10 10 80 20]...
                  );
%------------ end allctl.m ------------------------------------
```

8.4 Aplicação de Controles

A seguir apresentamos um exemplo completo de aplicação usando os controles MATLAB 'radio' e 'push'.

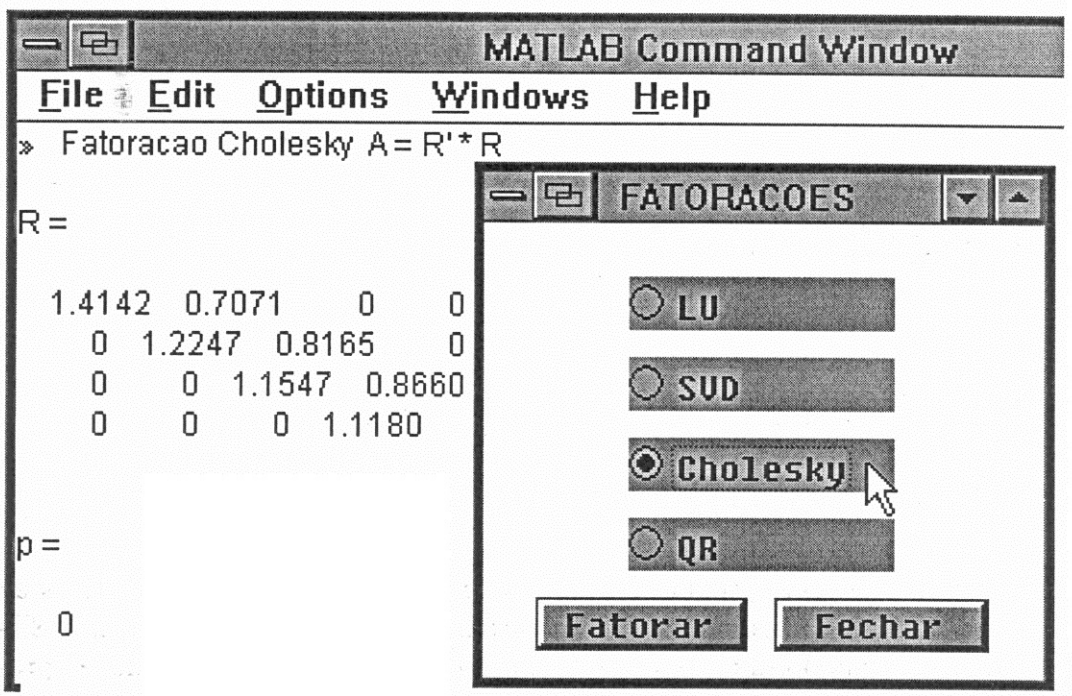
Figura 8.3: Exemplo de Controles

```
%%%-------------------------------------------------------------
%%% Arquivo:  fazctl.m
%%%
%%%          PROGRAMA PRINCIPAL
global A;
A = [2 1 0 0; 1 2 1 0; 0 1 2 1; 0 0 1 2]
control;
%------------ end fazctl -----------------------------------


%%%-------------------------------------------------------------
%%% Arquivo:  control.m
%
figNum=figure( ...
        'Name','FATORACOES', ...
        'NumberTitle','off', ...
        'Visible','on', ...
```

```
        'Resize','on', ...
        'MenuBar','none', ...
        'Color', [1 1 1], ...
        'Colormap',[0.5 0.5 0.5], ...
        'Position',[350,200, 210, 170],...
        'Pointer','arrow' ...  %% cross, circle,fleur
            );
%%%%%%%
fact(1) = uicontrol(gcf, ...
        'Style','radio', ...
        'String','LU', ...
        'Value', 1, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [55 130 100 20]);
fact(2)  = uicontrol(gcf, ...
        'Style','radio', ...
        'String', 'SVD', ...
        'Value', 0, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [55 100 100 20]);
fact(3)  = uicontrol(gcf, ...
        'Style','radio', ...
        'String', 'Cholesky', ...
        'Value', 0, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [55 70 100 20]);
fact(4)  = uicontrol(gcf, ...
        'Style','radio', ...
        'String', 'QR', ...
        'Value', 0, ...
        'BackGroundColor', [0.8 0.8 0.8], ...
        'Position', [55 40 100 20]);
%%%%
%%%%   Radio Buttons Mutuamente Exclusivos
%%%%   ----------------------------------
for i=1:4
   set(fact(i), 'UserData', fact(:,[1:(i-1),(i+1):4]))
end;
call =[...
      'me = get(gcf, ''CurrentObject'');',...
      'if(get(me,''Value'') == 1),', ...
```

```
            'set(get(me, ''UserData''), ''Value'',0),',...
        'else, ',...
            'set(me,''Value'',1),',...
        'end'];
set(fact,'CallBack', call);
%%%%%
fecha = uicontrol(gcf, ...
        'Style','push', ...
        'String','Fechar', ...
        'Position', [110 10 80 20],...
        'CallBack', [...
                    'close(gcf),', ...
                    'clc;']);
start = uicontrol(gcf, ...
        'Style','push', ...
        'String','Fatorar',...
        'Position', [20 10 80 20],...
        'CallBack', [...
            'if(get(fact(1),''Value'') == 1),', ...
                'fatora(1),', ...
            'elseif(get(fact(2),''Value'') == 1),',...
                'fatora(2),',...
            'elseif(get(fact(3),''Value'') == 1),',...
                'fatora(3),',...
            'else,',...
                'fatora(4),',...
            'end']);
 return;
%------------ end control ----------------------------------
```




```
%%%-------------------------------------------------------------
%%% Arquivo:  fatora.m
%
function fatora(x);
%%%%%%%%%%%%%%%%%%%%%%
global A;
if x == 1
   disp('  Fatoracao  A = L * U');
   [L,U] = lu(A)
elseif x == 2
   disp('  Fatoracao A = U * S * V ');
   [U,S,V] = svd(A)
elseif x == 3
   disp('  Fatoracao Cholesky  A = R'' * R ');
   [R,p] = chol(A)
else
   disp('  Fatoracao A = Q * R ');
   [Q,R] = qr(A)
end;
return;
%------------ end fatora ---------------------------------
```

Capítulo 9

MATLAB e Internet

Neste capítulo será mostrado como obter variadas informações relativas ao MATLAB (incluindo programas em arquivos .m), através da rede INTERNET, utilizando e-mail, www (World Wide Wb) e ftp (File Transfer protocol). Para estes dois últimos, é recomendado alguma familiaridade com sistemas Unix em estações de trabalho.

9.1 Contacto com The MathWorks via e-mail

`support@mathworks.com` - Suporte Tecnico
`suggest@mathworks.com` - Sugestões
`bugs@mathworks.com` - Reportando 'bugs'
`doc@mathworks.com` - Erros na Documentação
`subscribe@mathworks.com` - Registro de usuários
`service@mathworks.com` - Pedidos, renovações, senhas
`info@mathworks.com` - Vendas, preços, informação geral
`info.digest@mathworks.com` - Submission and questions for the digest

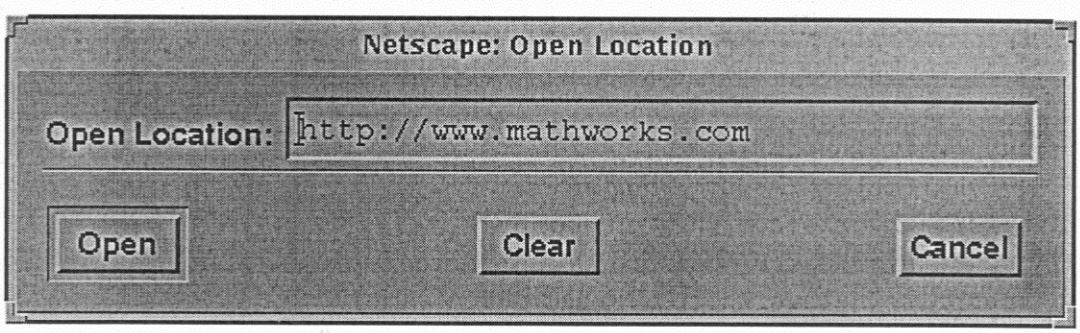
9.2 O MATLAB e a USENET

Na rede USENET existe o Newsgroup `comp.soft-sys.matlab`, onde são distribuídos diariamente dezenas de artigos e comentários relacionados com o uso do MATLAB. Neste Newsgroup são analizados aspectos gerais (arquivos-

funções, toolboxes, arquivos-contribuições, gráficos, impressão, etc.), bem como, asuntos relacionados com uma plataforma específica (MS Windows, Windows 95, Sun, Solaris, Macintosh, VMS, etc.). Para consultas sobre dúvidas relacionados com o MATLAB, este forum é relativamente rápido, pois para perguntas de 'uma linha de texto', obtem-se mais de uma resposta em um ou dois dias.

9.3 Usando o WWW

Sem dúvida alguma, uma das ferramentas mais importantes disponíveis na Internet, na atualidade, é o chamdo 'WWW' que nos permite 'surfar' com muita facilidade nas 'ondas' da rede Internet, acesando os computadores das diferentes universidades, centrosde pesquisa científica e empresas fabricantes de software e hardware. Um destes, é a empresa The MathWorks, responsável pelo desenvolvimento e comercialização do MATLAB. Os programas 'www' mais conhecidos são o Netscape e o Mosaic (para Unix e MS Windows). Para acesar a biblioteca de arquivos e utilitários do MATLAB, primeiro deve-se entrar no Netscape (ou Mosaic), clicar com o mouse no ícone open. Deve aparecer:



Escrever o endereço www:

`http://www.mathworks.com`

y clicar no butão [OPEN] e esperar que o Netscape faça a conexão com o The MathWorks. Logo clicar no menu gráfico para obter a informação desejada.

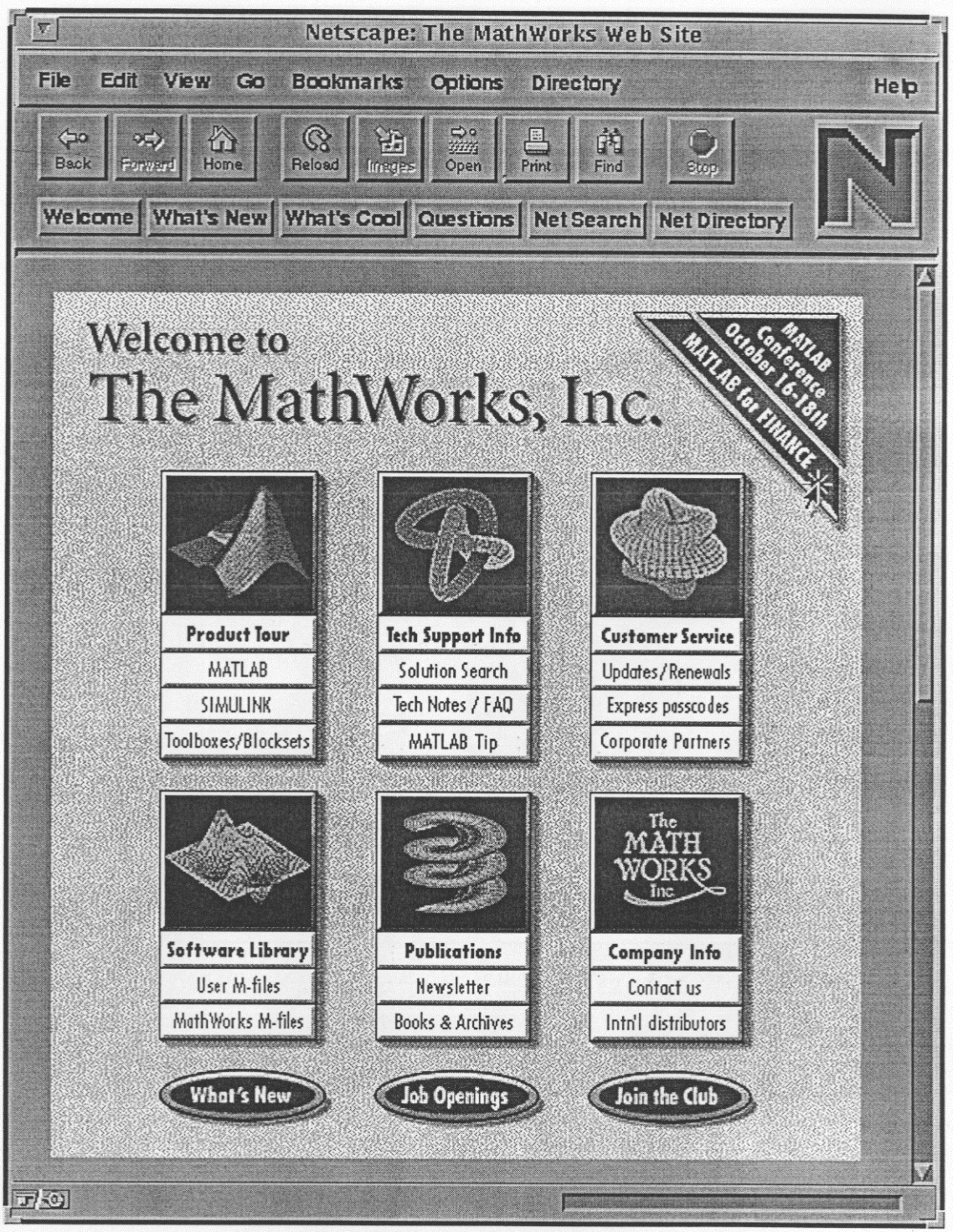


Figura 9.1: Acesando o MATLAB através do Netscape (SUN)

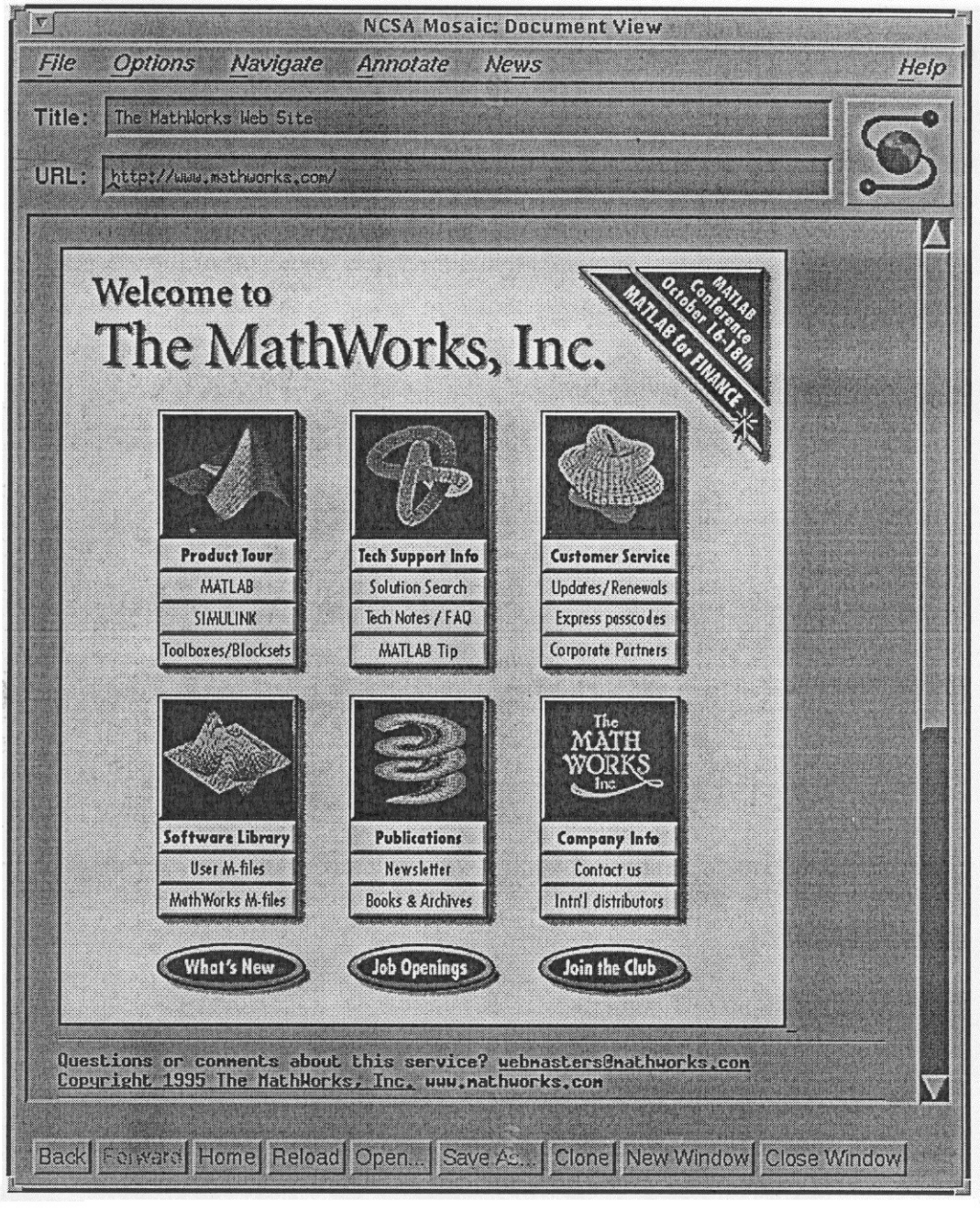


Figura 9.2: Acesando o MATLAB através do XMosaic (SUN)

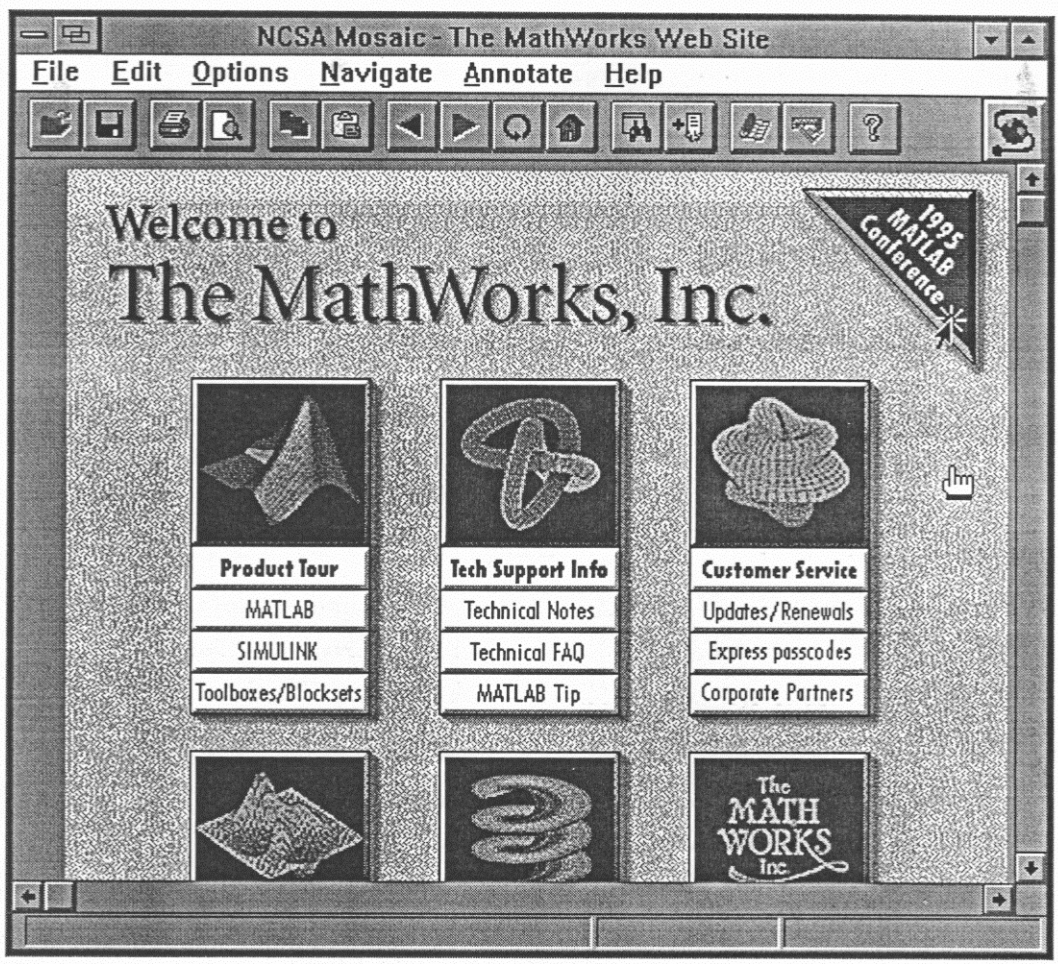


Figura 9.3: Acesando o MATLAB através do NCSA Mosaic for Windows (PC)

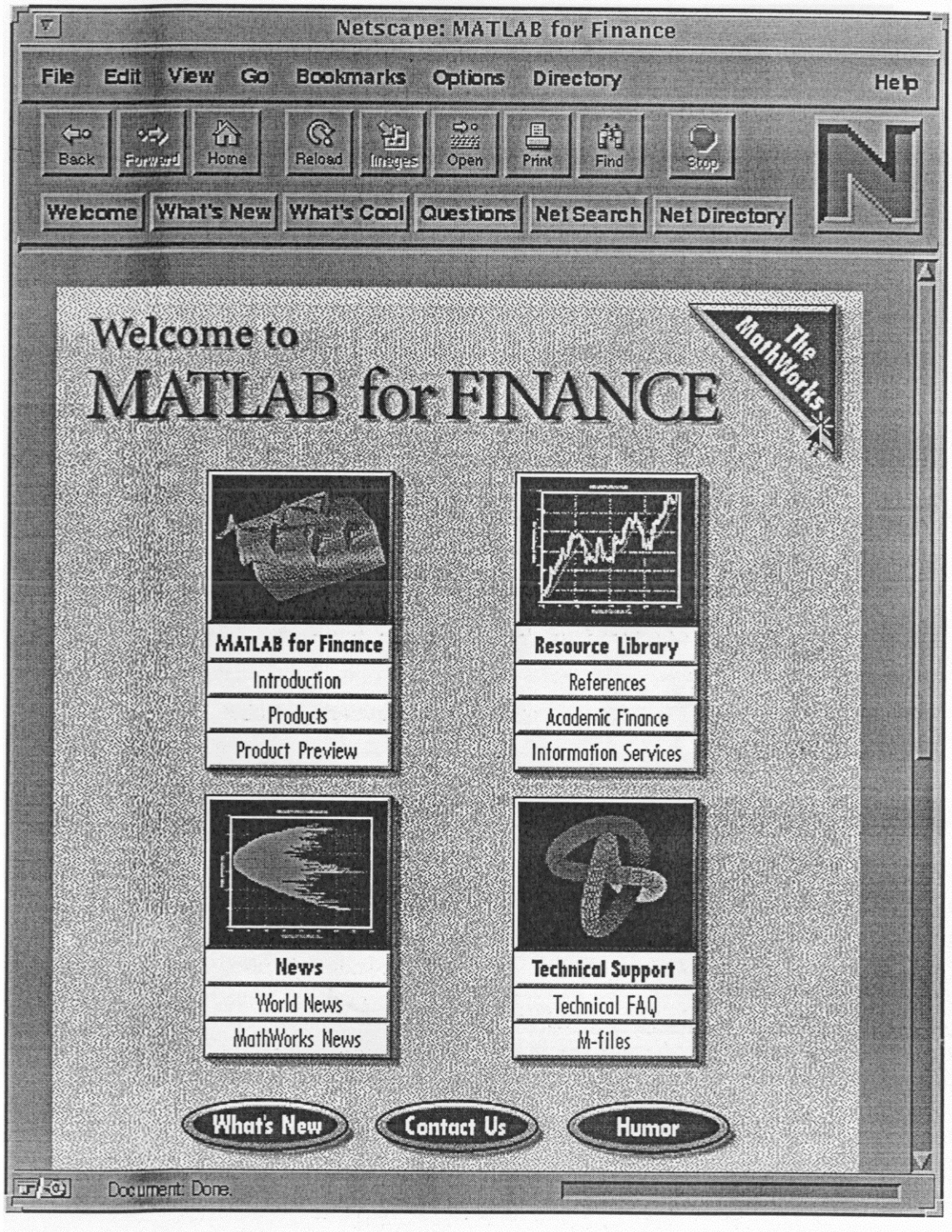


Figura 9.4: MATLAB e o Aspecto Financeiro

9.3.1 Endereços 'www' Importantes

- The MathWorks, Inc.
  `http://www.mathworks.com`
- MATLAB
  `http://www.mathworks.com/matlab.html`
- SIMULINK
  `http://www.mathworks.com/simulink.html`
- Toolboxes/Blocksets
  `http://www.mathworks.com/toolboxes.html`
- Informação sobre Suporte Técnico
  `http://www.mathworks.com/aboutts.html`
- Notas Técnicas
  `http://www.mathworks.com/technical.html`
- FAQ
  `http://www.mathworks.com/faq.html`
- Serviço ao Cliente
  `http://www.mathworks.com/custserv.html`
- Atualizações & Renovações
  `http://www.mathworks.com/custserv.html`
- Códigos-Senha
  `http://www.mathworks.com/custserv.html`
- M-arquivos Contribuições de Usuários
  `http://www.mathworks.com/ftpindex.html`
- M-arquivos do MATLAB
  `http://www.mathworks.com/contsoft.html`
- Publicações
  `http://www.mathworks.com/pubs.html`
- Newsletter
  `http://www.mathworks.com/newsletter/nn.html`
- Livros & Arquivos
  `http://www.mathworks.com/pubs.html`

- Informação sobre The MathWorks Inc
  `http://www.mathworks.com/abouttmw.html`
- Contatos
  `http://www.mathworks.com/contact1.html`
- Distribuidores Internacionais do MATLAB
  `http://www.mathworks.com/distribs.html`

9.4 Usando o FTP

Outra ferramenta (mais rápida que o WWW e sem recursos gráficos) para obtener informação relacionada com o MATLAB é o **ftp** (File Transfer Protocol).

A seguir apresentamos uma sessão completa de transferência de arquivos (ftp), aberta no Instituto de Informática (UFRGS) através do usuário **ascv** e desde a máquina **sagui**, para o servidor de arquivos do The MathWorks Library, usando os comandos **ftp, binary, cd, ls -lt, get** e **quit.**

ASCV@SAGUI:% **ftp ftp.mathworks.com**
Connected to ftp.mathworks.com.
220 ftp FTP server (Version wu-2.4(2) Tue Aug 1 10:31:36 EDT 1995) ready.
Name (ftp.mathworks.com:ascv): **anonymous**
331 Guest login ok, send your complete e-mail address as password.
Password: *(e-mail completo)*

```
230-
230-   Welcome to The MathWorks Library!
230-
230- To compress files before transfer, simply put a ".Z", for unix
230- compress, or a ".gz", for gzip, after the filename.
230-
230-   Example:  get filename.Z
230-
230- To transfer entire directories as archives, add the appropriate
230- extension to the directory name.
230-
230-        <directory>.tar       Tar archive
230-        <directory>.zip       Zip archive
230-
230- Note: If you get a "Permission denied" message, be sure you
```

```
230- have write permssion in the current local directory.
230-
230- You might want to use a mirror site near you:
230-
230- UK              ftp://unix.hensa.ac.uk/mirrors/matlab
230- Germany         ftp://ftp.ask.uni-karlsruhe.de/pub/matlab
230- Czech Republic  ftp://novell.felk.cvut.cz/pub/mirrors/mathwork
230- Japan           ftp://ftp.u-aizu.ac.jp:/pub/vendor/mathworks
230-
230- If you would like to be a mirror site for this ftp library
230- send e-mail to ftpadmin@mathworks.com.
230-
230- Send questions/comments/suggestions to ftpadmin@mathworks.com
230-
230-
230-Please read the file README
230-  it was last modified on Wed Sep 28 15:10:24 1994 - 359 days ago
230 Guest login ok, access restrictions apply.

ftp> binary
200 Type set to I.
ftp> cd pub/contrib

250-This directory contains user-contributed M/MEX-files.
250-
250-To retrieve entire directories use the syntax:
250-        get dirname.zip     %?zip archive
250-        get dirname.tar     %?tar archive
250-        get dirname.sh      %?shar archive
250-
250-If you are using a web browser, you can get an entire directory
250-by using an URL like,
250-
250- ftp://ftp.mathworks.com/pub/directoryname.tar
250-
250 CWD command successful.

ftp> ls -lt

200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 158
drwxr-xr-x  18 102      100          512 Sep 15 18:01 simulink
-rw-r--r--   1 102      100       124743 Sep 11 21:50 INDEX.ALL
drwxr-xr-x  10 102      100          512 Sep 11 21:49 optim
drwxr-xr-x  35 102      100         2048 Sep  8 18:35 graphics
drwxrwxr-x   2 102      102          512 Aug 14 22:00 symbolic
```

```
drwxr-xr-x   4 102      100          512 Aug 14 22:00 systemid
drwxr-xr-x   3 102      102          512 Aug 14 22:00 games
drwxr-xr-x   3 102      100          512 Aug 14 22:00 nnet
drwxr-xr-x   8 102      100          512 Aug 14 21:59 signal
drwxr-xr-x   3 102      100          512 Aug 14 21:59 physics
drwxr-xr-x   3 102      100          512 Aug 14 21:58 diffeq
drwxr-xr-x   8 102      100          512 Aug 14 21:57 linalg
drwxr-xr-x   6 102      100          512 Aug 14 21:57 math
drwxr-xr-x   9 102      100          512 Aug 14 21:57 stats
drwxr-xr-x  16 102      100         1024 Aug 14 21:57 tools
drwxr-xr-x   5 102      100          512 Aug 14 21:57 teaching
drwxr-xr-x  25 102      100         1024 Aug 14 21:57 misc
drwxr-xr-x   5 102      100          512 Aug 14 21:55 integration
drwxr-xr-x   9 102      100          512 Aug 14 21:54 control
drwxr-xr-x   5 102      100          512 Aug 14 21:54 approx
drwxr-xr-x   8 102      100          512 Aug 14 21:53 model
-rw-r--r--   1 102      100          864 May  1 18:41 INDEX
226 Transfer complete.
remote: -lt
1405 bytes received in 0.01 seconds (1.3e+02 Kbytes/s)

ftp> cd graphics
250 CWD command successful.
ftp> ls -lt

200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 250
drwxrwxr-x   2 102      102          512 Sep  8 18:35 stextfun
-rw-r--r--   1 102      102        28935 Sep  8 18:25 INDEX
drwxrwxr-x   2 102      102          512 Aug 28 19:22 plotseis
drwxrwxr-x   2 102      102          512 Aug 25 19:13 adhamlet
drwxr-xr-x   2 102      102          512 Aug 14 21:55 uigoodies
drwxr-xr-x   2 102      102          512 Aug 14 21:55 triplot
drwxr-xr-x   2 102      102          512 Aug 14 21:55 textfcns
drwxr-xr-x   2 102      102         1024 Aug 14 21:55 matdraw2.1
drwxr-xr-x   2 102      102          512 Aug 14 21:55 shadedrelief
drwxr-xr-x   2 102      102          512 Aug 14 21:55 rotapolh
drwxrwxr-x   2 102      102         1024 Aug 14 21:55 graphedt
drwxr-xr-x   2 102      102          512 Aug 14 21:55 matmenus
drwxr-xr-x   2 102      102          512 Aug 14 21:55 link
drwxr-xr-x   2 102      102          512 Aug 14 21:55 kwtools
drwxr-xr-x   2 102      102          512 Aug 14 21:55 jnatools
drwxr-xr-x   2 102      102          512 Aug 14 21:55 hatch
drwxr-xr-x   2 102      102          512 Aug 14 21:55 guimaker1.0
drwxr-xr-x   3 102      102         1024 Aug 14 21:54 matdraw1.0
drwxr-xr-x   2 102      102         1536 Aug 14 21:54 guimaker2.1
drwxr-xr-x   2 102      102          512 Aug 14 21:54 hull2d
```

```
drwxr-xr-x   2 102      102          512 Aug 14 21:54 gpppp
drwxr-xr-x   2 102      102          512 Aug 14 21:54 gui_plot
drwxr-xr-x   2 102      102          512 Aug 14 21:54 images
drwxr-xr-x   2 102      102          512 Aug 14 21:54 extcontour
lrwxrwxrwx   1 102      102           11 Aug 14 21:54 guimaker -> guimaker2.1
-rwxr-xr-x   2 102      102         1024 Aug 14 21:54 zoomtool
-rwxr-xr-x   2 102      102          512 Aug 14 21:54 legend
-rwxr-xr-x   2 102      102          512 Aug 14 21:54 uiprint
-rwxrwxrwx   1 102      102           10 Aug 14 21:54 matdraw -> matdraw2.1
drwxr-xr-x   2 102      102          512 Aug 14 21:54 scrledit
drwxr-xr-x   2 102      102          512 Aug 14 21:54 splotyy
drwxr-xr-x   3 102      102          512 Aug 14 21:54 mpgwrite
drwxr-xr-x   2 102      102          512 Aug 14 21:54 comets
drwxr-xr-x   3 102      102          512 Aug 14 21:54 mpgread
-rw-rw-r--   1 102      102         2065 Jun 22 21:44 suptitle.m
-rw-rw-r--   1 102      102         6664 May 30 15:14 printui.m
-rw-r--r--   1 102      102         1219 May  5 21:29 arrow2.m
-rw-r--r--   1 102      102        48921 May  5 21:29 arrow.m
-rw-r--r--   1 102      102         4008 May  5 19:09 stereo.m
-rw-rw-r--   1 102      102         6172 May  5 16:00 sirds.m
-rw-r--r--   1 102      102         1015 Mar 14  1995 smith.m
-rw-r--r--   1 102      100         3149 Dec 15  1994 curvefit.m
-rw-r--r--   1 102      102         2029 Nov 17  1994 contoura.m
-rw-r--r--   1 102      102         6784 Oct 26  1994 plotyy.m
-rw-r--r--   1 102      102         1060 Sep 12  1994 chart.m
-rw-r--r--   1 102      100         3455 Aug  9  1994 loadbmp.m
-rw-r--r--   1 102      102         4523 Jul 26  1994 zoomrb.m
drwxr-xr-x   6 102      102          512 Jun 23  1994 xplot
-rw-r--r--   1 102      102          615 May 13  1994 spanplot.m
-rw-r--r--   1 102      102         2580 May  6  1994 plot3in2.m
-rw-r--r--   1 102      102         2910 May  6  1994 signature.m
-rw-r--r--   1 102      102         2275 Apr  7  1994 waitbar2.m
-rw-r--r--   1 102      102         4862 Apr  4  1994 hist.c
-rw-r--r--   1 102      102        11780 Mar 31  1994 slegend.m
-rw-r--r--   1 102      102         6044 Dec 17  1993 lv_lscan.m
-rw-r--r--   1 102      102        26609 Dec 13  1993 modal.exe
-rw-r--r--   1 102      102        15168 Dec 13  1993 modal.dll
-rw-r--r--   1 102      102          842 Dec  3  1993 torus.m
-rw-r--r--   1 102      102         2057 Nov 16  1993 fillbar.m
-rw-r--r--   1 102      100         3559 Sep 13  1993 savebmp.m
-rw-r--r--   1 102      100         3179 Jun  3  1993 bglegend.m
-rw-r--r--   1 102      100         3170 Sep 15  1992 curvefitnl.m
226 Transfer complete.
remote: -lt
4080 bytes received in 6 seconds (0.66 Kbytes/s)
```

ftp> **get stextfun.zip**

```
200 PORT command successful.
150 Opening BINARY mode data connection for /bin/zip.
226 Transfer complete.
local: stextfun.zip remote: stextfun.zip
65795 bytes received in 3.1e+02 seconds (0.21 Kbytes/s)

ftp> get INDEX

200 PORT command successful.
150 Opening BINARY mode data connection for INDEX (28935 bytes).
226 Transfer complete.
local: INDEX remote: INDEX
28935 bytes received in 1.2e+02 seconds (0.24 Kbytes/s)

ftp> quit
ASCV@SAGUI:%
```

Apêndice A

Guia de Referência Rápida

Este Guia é baseado no *MATLAB User's Guide, version 4.0, 1993*, *MATLAB Release Notes version 4.2c, 1994* distribuído pela The MathWorks, Inc. e *The Student Edition of MATLAB User's Guide, version 4.0, 1995* publicado pela Prentice-Hall, Inc.

Principais Categorias de Funções MATLAB	
`color`	Color control and Lighting model functions
`datafun`	Análise de dados e Transformada de Fourier
`demos`	Demonstrações
`elfun`	Funções Matemáticas elementais
`elmat`	Matrizes Elementais e Manipulação matricial
`funfun`	Funções de Função
`general`	Comandos de uso geral
`graphics`	Funções Gráficas
`iofun`	Funções Entrada/Saida de baixo nivel
`lang`	Construções da linguagem
`matfun`	Funções Matriciais - Álgebra Linear Numérica
`ops`	Operadores e Carateres Especiais
`plotxy`	Gráficos 2D
`plotxyz`	Gráficos 3D
`polyfun`	Funções Polinomiais e de Interpolação
`sparfun`	Funções para Matrizes Esparsas
`specfun`	Funções matemáticas Especializadas
`specmat`	Matrizes Especializadas
`sounds`	Funções para processar som
`strfun`	Funções para strings

A.1 Comandos Gerais

Comandos e Funções de Gerenciamento	
`demo`	Demonstrações
`expo`	MATLAB EXPO (demo)
`help`	Documentação On-line (Ajuda)
`info`	Info sobre MATLAB e The Mathworks
`lasterr`	Último mensagem de erro gerado
`lookfor`	Help por palavra-chave
`path`	Mostra o path para o MATLAB
`type`	Mostra um arquivo *.m
`ver`	Versão do MATLAB e toolboxes instalados
`version`	Versão do MATLAB
`what`	Lista arquivos .M, .MAT e .MEX
`whatsnew`	Mostra arquivos README do MATLAB
`which`	Localiza funções e arquivos

Gerenciando Variáveis e Espaço de Trabalho	
`clear`	limpa variaveis e funções da memmória
`disp`	Mostra uma matriz ou um texto
`length`	Comprimento de um vetor
`load`	Carrega variáveis do disco
`pack`	Salva, limpa memória e re-carrega variáveis
`save`	Grava variáveis para o disco
`size`	Tamanho de uma matriz
`who`	Lista todas as variáveis da memória
`whos`	Idem, forma Longa

Controlando a Janela de Comandos	
`clc`	Limpa a janela de comandos
`echo`	Mostra comandos dentro de un arquivo *.m
`format`	Seleciona formato de saída
`home`	Cursor na esquina superior esquerda
`more`	Saída na janela de comandos por páginas

Inicializar-Finalizar	
`matlabrc`	Arquivo de inicialização do MATLAB
`quit`	Termina sesão do MATLAB
`startup`	Arquivo-usuário executado ao iniciar o MATLAB

A.2 Operadores e Caracteres Especiais

Operadores e Caracteres Especiais	
+	Soma
-	Substração
*	Produto matricial
.*	Produto elemento-a-elemento
^	Potência matricial
.^	Potência element-a-elemento
kron	Produto tensorial de Kronecker
\	Divisão a esquerda
/	Divisão a direita
./	Divisão elemento-a-elemento
:	Dois pontos
( )	Parêntesis
[]	Colchetes
.	Ponto decimal
. .	Diretório pai
. . .	Continuação
,	virgula
;	ponto e virgula
%	Comentário
!	Exclamação
'	Transposta ou quote
. '	Transposta não-conjugada
=	Atribuição
==	Igualdade
< >	Operadores relacionais
&	AND lógico
|	OR lógico
~	NOT lógico
xor	OR Exclusivo lógico

Funções Lógicas	
`all`	verdadeiro se todos os elementos de um vetor são verdadeiros
`any`	verdadeiro se algum elemento do vetor é verdadeiro
`exist`	verifica se existem variáveis ou funções
`find`	Achar indices de elementos não-zeros
`finite`	verdadeiro para elementos finitos
`isempty`	verdadeiro para matriz vazia
`ishold`	verdadeiro se `hold` é `on`
`isieee`	verdadeiro para aritmetica de ponto-flutuante IEEE
`isinf`	verdadeiro para elementos infinitos
`isletter`	verdadeiro para caracteres alfabéticos
`isnan`	verdadeiro para Not-A-Number
`isreal`	verdadeiro se todos os elementos da matriz são reais
`issparse`	verdadeiro para matrizes esparsas
`isstr`	verdadeiro para string de carateres

A.3 Construções da Linguagem e Depuração

MATLAB como uma linguagem de Programação	
`eval`	Avalia strings com expressões MATLAB
`feval`	Executa uma função especificada por string
`function`	Nova função
`global`	Define variável global
`nargchk`	Valida número de argumentos de entrada

Fluxo de Controle	
`break`	Termina execução de um laço
`else`	Usada com `if`
`elseif`	Usada com `if`
`end`	Termina o escopo de `for`, `while` e `if`
`error`	Mostra mensagem e termina uma função
`for`	laço `for` para repetição
`if`	Execução condicional
`return`	Retorno à invocação de uma função
`while`	Repetição indefinida

Entrada Interactiva	
`input`	Sinal para entrada de usuário
`keyboard`	Chama o teclado como se fosse um arquivo de comandos
`menu`	Gera um menu de opções para entrada do usuário
`pause`	Espera a resposta do usuário

A.4 Matrizes e Manipulação Matricial

A.4.1 Matrizes Elementais

Matrizes Elementais	
`eye`	Matriz Identidade
`gallery`	Matrizes test
`linspace`	vetor espaçado linearmente
`logspace`	vetor espaçado logaritmicamente
`meshgrid`	Matriz X-Y para gráficos 3-D
`ones`	Matriz de 1's
`rand`	Números randomicos distribuidos uniformemente
`randn`	Números randomicos distribuidos normalmente
`zeros`	Matriz de 0's
`:`	vetor espaçado regularmente

Variáveis e Constantes Especiais	
`ans`	Última resposta
`computer`	Tipo de computador (PC, UNIX, VMS, etc.)
`eps`	'epsilon da máquina'
`flops`	contador de operações em ponto-flotante
`i, j`	Unidade imaginária
`inf`	Infinito
`NaN`	Não-é-um-número
`nargin`	Número de argumentos de entrada de uma função
`nargout`	Número de argumentos de saída de uma função
`pi`	3.14159265358...
`realmax`	Máximo número decimal
`realmin`	Menor número decimal

Tempos e Datas	
`clock`	= [ano mes dia hora minuto segundo]
`cputime`	Tempo CPU usado pelo MATLAB
`date`	Calendário
`etime`	Tempo entre duas variáveis
`tic, toc`	Início-fim de um relógio

Manipulação de Matrizes	
`diag`	Cria ou extrai diagonal
`fliplr`	Inverte matriz: esquerda → direita
`flipud`	Inverte matriz: acima → abaixo
`isreal`	Verdadeiro se uma matriz contém somente números reais
`reshape`	Muda tamanho
`rot90`	Rotar uma matriz 90 graus
`tril`	Parte triangular inferior de uma matriz
`triu`	Parte triangular superior de uma matriz
`:`	Índice dentro de uma matriz

A.4.2 Matrizes Especializadas

Matrizes Especializadas	
`compan`	Matriz Companheira
`hadamard`	Matriz de Hadamard
`hankel`	Matriz de Hankel
`hilb`	Matriz de Hilbert
`invhilb`	Matriz de Hilbert inversa
`magic`	Matriz Quadrada Mágica
`pascal`	Matriz de Pascal
`rosser`	Matriz de Rosser
`toeplitz`	Matriz de Toeplitz
`vander`	Matriz de Vandermonde
`wilkinson`	Matriz teste de autovalores

A.5 Funções Matemáticas

A.5.1 Funções Matemáticas Elementais

Funções Matemáticas Elementais	
abs	Valor absoluto
acos	Coseno inverso
acosh	Coseno hiperbólico inverso
acot	Cotangente inverso
acoth	Cotangente hiperbólico inverso
acsc	Cosecante inversa
acsch	Cosecante hiperbólico inverso
angle	Angulo de fase
asec	Secante inversa
asech	Secante hiperbólico inverso
asin	Seno inverso
asinh	Seno hiperbólico inverso
atan	Tangente inversa
atan2	Tangente inversa 4to. quadrante
atanh	Tangente hiperbólico inversa
ceil	Arredondamento a $+\infty$
conj	Conjugada complexa
cos	Coseno
cosh	Coseno hiperbólico
cot	Cotangente
coth	Cotangente hiperbólico
csc	Cosecante
csch	Cosecante hiperbólica
exp	Exponencial
fix	Arredondamento para zero
floor	Arredondamento para $-\infty$
gcd	Máximo comum divisor
imag	Parte imaginária complexa
lcm	Mínimo Comum Múltiplo
log	Logaritmo natural
log10	Logaritmo decimal
real	Parte real complexa
rem	Resto após a divisão
round	Arredondamento para o inteiro mais próximo

Funções Matemáticas Elementais ...	
`sec`	Secante
`sech`	Secante hiperbólica
`sign`	Função sinal
`sin`	Seno
`sinh`	Seno hiperbólico
`sqrt`	Raiz quadrada
`tan`	Tangente
`tanh`	Tangente hiperbólica

A.5.2 Funções Matemáticas Especializadas

Funções Matemáticas Especializadas	
`bessel`	Funções de Bessel
`besseli`	Funções de Bessel Modificadas de 1a. classe
`besselj`	Funções de Bessel de 1a. classe
`besselk`	Funções de Bessel Modificadas de 2a. classe
`bessely`	Funções de Bessel de 2da.classe
`beta`	Função beta
`betainc`	Função beta incompleta
`betaln`	Lograritmo da Função beta
`ellipj`	Jacobiano das Funções Elipticas
`ellipke`	Integral Eliptica Completa
`erf`	Função de erro
`erfc`	Função de erro complementar
`erfcx`	Função de erro complementar escalada
`erfinv`	Função de erro inversa
`expint`	Integral Exponencial
`gamma`	Função gamma
`gammainc`	Função gamma incompleta
`gammaln`	Logaritmo da Função gamma
`legendre`	Funções de Legendre associadas
`log2`	Logaritmo base 2
`pow2`	`pow2(x)` $= 2^x$
`rat`	Aproximação racional
`rats`	saída racional

A.6 Funç. Matriciais - Álg. Linear Numérica

Análise Matricial	
`cond`	Número de condição de uma matriz
`det`	Determinate
`etree`	Arvore de eliminação de uma matriz
`norm`	Norma de uma matriz ou vetor
`null`	Espaço nulo
`orth`	Ortogonalização
`rcond`	Estimador de condição reciproca
`rank`	Posto
`rref`	Forma 'row echelon' reduzida
`subspace`	Ángulo entre dois subspaços
`trace`	Soma dos elementos da diagonal

Equações Lineares	
`chol`	Fatoração de Cholesky
`inv`	Matriz Inversa
`lscov`	Mínimos Quadrados na presença de covariança desconhecida
`lu`	Fatores da Eliminação Gaussiana
`nnls`	Mínimos Quadrados Não-negativos
`pinv`	Pseudoinversa
`qr`	Decomposição Ortogonal-Triangular
`\ e /`	Solução de uma Equação Linear

Valores Próprios e Singulares	
`balance`	Melhora a exatidão de valores próprios calculados
`cdf2rdf`	Forma Diag. Complexa para Forma Diagonal de bloco real
`eig`	Valores Próprios e Vetores Próprios
`hess`	Forma de Hessenberg
`poly`	Polinômio Característico
`qz`	Valores Próprios Generalizados
`rsf2csf`	Forma diagonal de Bloco real para Forma Diag. Complexa
`schur`	Decomposição de Schur
`svd`	Decomposição de valor singular

Funções Matricias	
`expm`	Exponencial matricial
`funm`	Avalia uma função matricial
`logm`	Logaritmo matricial
`sqrtm`	Raiz Quadrada matricial

Funções Matriciais de Baixo-Nível	
`qrdelete`	Elimina colunas da Fatoração QR
`qrinsert`	Insere colunas na Fatoração QR

A.7 Análise de Dados e Transformada de Fourier

Operações Básicas	
`cumprod`	Produto acumulativo de elementos
`cumsum`	Soma acumulativa de elementos
`max`	Maior componente
`mean`	Promedio ou valor medio
`median`	Valor Mediano
`min`	Menor componente
`prod`	Produto de Elementos
`sort`	Ordenação ascendente
`std`	Desvio Padrão
`sum`	Soma de elementos
`trapz`	Integração Numérica (met. trapezoidal)

Diferenças Finitas	
`del2`	Laplaciano Discreto de 5-pontos
`diff`	Função Diferença
`gradient`	Gradiente Aproximado

Correlação	
`corrcoef`	Coeficientes de Corelação
`cov`	Matriz de Covariança

Filtros e Convolução	
`conv`	Convolução e Multip. polinomial
`conv2`	Convolução Bi-dimensional
`deconv`	Deconvolução e Divisão polinomial
`filter`	Filtro digital Uni-dimensional
`filter2`	Filtro digital Bi-dimensional

Transformada de Fourier	
`abs`	Magnitude
`angle`	Ángulo de fase
`cplxpair`	Ordenação de números em pares conj. complexos
`fft`	Transformada Discreta de Fourier
`fft2`	Transformada Discreta de Fourier Bi-dimensional
`fftshift`	Shift FFT
`ifft`	Inversa da Transformada Discreta de Fourier
`ifft2`	Inversa da Transformada Discreta de Fourier Bi-dimensional
`nextpow2`	Próxima potência superior de 2
`unwrap`	Remove ángulo de fase

Funções Vetoriais	
`cross`	Produto Vetorial
`dot`	Produto Escalar Vetorial

A.8 Funções Polinomiais e de Interpolação

Funções Polinomiais	
`conv`	Convolução e Multiplicação Polinomial
`deconv`	deconvolução e Divisão Polinomial
`poly`	Polinômio Característico
`polyder`	Derivada Polinomial
`polyeig`	Valores própios polinomiais
`polyfit`	Ajustamento de curvas polinomiais
`polyval`	Avalia um polinômio
`polyvalm`	Avalia um polinômio com argumento matricial
`residue`	Residuo
`roots`	Acha raizes polinomiais

Interpolação de Dados	
`griddata`	grade de dados
`interp1`	Interpolação 1-dimensional
`interp2`	Interpolação bi-dimensional
`interpft`	Interpolação 1-dimensional usando o método FFT

A.9 Funções de Função

Funções de Função - Mét. Numéricos Não-Lineares	
`fmin`	Minimiza função de uma variável
`fmins`	Minimiza função de várias variáveis
`fplot`	Grafica uma função
`fzero`	Acha os zeros de uma função de uma variável
`ode23`	Resolve Equações diferenciais, método ordem inferior
`ode45`	Resolve Equações diferenciais, método ordem superior
`quad`	Avalia integral numericamente, método ordem inferior
`quad8`	Avalia integral numericamente, método ordem superior

A.10 Funções para Matrizes Esparsas

Matrizes Esparsas Elementais	
spdiags	Matriz esparsa formada apartir de diagonais
speye	Matriz esparsa identidade
sprandn	Matriz esparsa randómica
sprandsym	Matriz esparsa randómica simétrica

Conversion: Cheia para Esparsa	
find	Acha indices de entradas não-zero
full	Converte matriz esparsa para matriz cheia
sparse	Cria matriz esparsa apartir de zeros e indices
spconvert	Converte formato externo (load, save)

Entradas não-zeros de Matrizes Esparsas	
issparse	Verdadeiro se uma matriz é esparsa
nnz	Número de entradas diferentes de zero
nonzeros	Entradas diferentes de zero
nzmax	Quantidade de memória alocada para entradas não-zero
spalloc	Aloca memória para entradas diferentes de zero
spfun	Aplica uma função a entradas não-zero
spones	Substitui entradas não-zero com 1's

Visualizando Matrizes Esparsas	
gplot	Plotar Grafos, como em "teoria de grafos"
spy	Visualiza estrutura de Esparsidade

Algoritmos de Re-Ordenação	
colmmd	Coluna de grau minimo
colperm	Permutação de colunas
dmperm	Decomposição de Dulmage-Mendelsohn
randperm	Vetor de permutação randómica
symmmd	Grau minimo simétrico
symrcm	Ordenação Reversa de Cuthill-McKee

Norma, Número de Condição	
`condest`	Condição 1-norma estimada
`normest`	2-norma estimada
`sprank`	Posto estrutural

Miscelânea	
`spaugment`	Forma de minimos quadrados
`spparms`	Atualiza parâmetros para rotinas de matrizes esparsas
`symbfact`	Análise de Fatoração simbólica

A.11 Gráficos Bi-Dimensionais (2-D)

Gráficos Elementais X-Y	
`fill`	Exibe graficamente poligonos 2-D prenchidos
`loglog`	plotar na escala Log-log
`plot`	gráfico linear
`semilogx`	plotar na escala Semi-log, eixo-X logaritmicamente
`semilogy`	plotar na escala Semi-log, eixo-Y logaritmicamente

Gráficos Especializados X-Y	
`bar`	grafo em barras
`comet`	Exibe graficamente curvas animadas
`compass`	Exibe graficamente compass
`errorbar`	Exibe graficamente graficos com barras de erro
`feather`	Exibe graficamente vetores em forma de penas
`fplot`	Exibe graficamente uma função
`hist`	Exibe graficamente um histograma
`polar`	Exibe graficamente em coordenadas polares
`rose`	Exibe graficamente um histograma em forma angular
`stairs`	Exibe graficamente um vetor em escada
`stem`	Exibe graficamente seqüência de dados discretos

Textos em Gráficos	
`grid`	Exibe graficamente uma grid
`gtext`	insere texto usando o mouse
`legend`	Agrega legenda a um gráfico
`text`	Insere texto em um grafico no ponto (X,Y)
`title`	Titulo do gráfico
`xlabel`	Rótulo do eixo X
`ylabel`	Rótulo do eixo Y

Conversão de Sistemas de Coordenadas	
`cart2pol`	Coordenadas cartesianas para poloares
`pol2cart`	Coordenadas polares para cartesianas

Miscelânea	
`zoom`	Zoom IN e OUT de gráfico 2D usando mouse

A.12 Gráficos Tri-Dimensionais (3-D)

Linhas e Prenchimento de Areas	
`fill3`	Exibe graficamente poligonos 3-D prenchidos
`plot3`	Exibe graficamente linhas e pontos 3-D

Contornos e Gráficos 2-D de dados 3-D	
`clabel`	Agrega rótulos a curvas de contorno
`comet3`	Exibe graficamente linhas animadas em 3D
`contour`	Exibe graficamente curvas de contorno 2D
`contour3`	Exibe graficamente curvas de contorno 3D
`contourc`	Calcula a matriz de contorno, usada por `contour`
`image`	mostra uma matriz como uma imagem colorida
`imagesc`	Igual que `image`, colores diferentes
`pcolor`	Exibe graficamente pseudocolor
`quiver`	Exibe graficamente vetores velocidade
`slice`	Exibe graficamente pedaços volumétricos

Superficies e Mesh	
`mesh`	Exibe graficamente superficies 3D
`meshc`	Combina gráficos `mesh/contour`
`meshz`	mesh 3D com plano zero
`slice`	Exibe graficamente visualização volumétrica
`surf`	Superficie sombreada 3D
`surfc`	Combinação `surf/contour`
`surfl`	Superficie sombreada 3D com iluminação
`waterfall`	Gráfico tipo "cascata"

Texturas	
`axis`	Eixos de um gráfico (on, off, etc.)
`caxis`	Eixos de Pseudocolor
`colormap`	Tabela de cores
`hidden`	Mesh com remoção de linhas
`shading`	Modo com combreamento de cor
`view`	Especificação do ponto de visualização 3D
`viewmtx`	Gera matrizes de transf. do ponto de visão

Textos em Gráficos	
`grid`	Grade
`legend`	Adiciona legenda a um gráfico
`text`	Colocar texto no ponto (X,Y) em um gráfico
`title`	Título de um gráfico
`xlabel`	Rótulo no eixo X
`ylabel`	Rótulo no eixo Y
`zlabel`	Rótulo no eixo Z

Objetos 3-D	
`cylinder`	Gera um cilindro
`sphere`	Gera uma esfera

Conversão de Sistemas de Coordenadas	
`cart2sph`	Cartesianas para polares
`sph2cart`	Polares para cartesianas

A.13 Funções Gráficas

Controle e Criação de Janelas	
`capture`	Captura janela corrente
`clf`	Limpa uma figura
`close`	Fecha uma figura
`figure`	Cria uma janela gráfica
`gcf`	Obtém o handle de um gráfico
`newplot`	Preambulo para uma nova figura (arquivo .M)
`refresh`	re-exibe uma janela gráfica
`whitebg`	Tela de uma figura com fundo branco

Controle e Criação de Eixos	
`axes`	Cria eixos em qualquer posição
`axis`	Controle de eixos (apresentação)
`caxis`	Controle de pseudocolor
`cla`	Limpa uma janela gráfica
`gca`	Obtém handle dos eixos
`hold`	Segura a apresentação de uma figura
`ishold`	Verdadeiro se `hold` é `on`
`subplot`	Cria mais de um gráfico em uma única janela

Manipulação de Objetos Gráficos	
`axes`	Cria eixos
`figure`	Cria janela gráfica
`image`	Cria imagem
`line`	Cria linhas
`patch`	Cria patch
`surface`	Cria superficies
`text`	Cria text
`uicontrol`	Cria controles usando a GUI
`uimenu`	Cria menus usando a GUI

Manipulação de Operações Gráficas	
`delete`	Apaga um objeto
`drawnow`	Atualiza a tela com um gráfico
`findobj`	Acha objeto com determinadas propriedades
`gco`	Obtém handle de objeto atual
`get`	Obtém propriedades de um objeto
`reset`	Atualiza propriedades de um objeto
`rotate`	Rotar um objeto
`set`	Inicializa propriedades de um objeto

Caixas de Diálogo	
`uigetfile`	Obtém nome de arquivo (menu: abrir)
`uiputfile`	Obtém nome de arquivo (menu: escrever)

Impressão	
`orient`	Orientação do papel (landscape-portrait)
`print`	Imprime ou salva gráfico
`printopt`	Opções de impressora local

Filme e Animação	
`getframe`	Obtém vetor-frame para filme
`movie`	Roda frame de filme
`moviein`	Inicializa frame de filme

Miscelânea	
`ginput`	Entrada gráfica a partir do mouse
`ishold`	Retorna estado do `hold`
`rbbox`	Caixa Rubberband
`waitforbuttonpress`	Espera até pressionar uma tecla

A.14 Controle de Cor e Iluminação

Controle de Cor	
`caxis`	Eixos de pseudocolor
`colormap`	tabela de cor
`shading`	modo sombreado

Colormaps	
`bone`	"osso"
`contrast`	Contraste ressaltado
`cool`	Sombras de cyan e magenta
`copper`	cooper-tone linear
`flag`	Vermelho, branco, azul, e preto alternado
`gray`	grayscale linear
`hsv`	Hue-value-saturation
`hot`	Preto-vermelho-amarelo-branco
`jet`	variação de HSV
`pink`	rosa sombreado
`prism`	cores do prisma
`white`	modo monocromatico

Funções Relacionadas a Colormaps	
`brighten`	Brighten ou darken
`colorbar`	escala de cores
`hsv2rgb`	Conversion HSV para vermelho-verde-azul
`rgb2hsv`	Inverso do anterior
`rgbplot`	Exibe graficamente colormap
`spinmap`	Rota colormap

Modelos de Iluminação	
`difuse`	Reflectância difusa
`specular`	Reflectância especular
`surfl`	Superficie 3D sombreada com iluminação
`surfnorm`	Normal de superficies

A.15 Funções para Processar Sons

Funções de Som	
`saxis`	Eixos em som
`sound`	Converte vetor em som

Funções de Som .WAV	
`wavread`	Lê um arquivo .WAV em MS Windows 3.1
`wavwrite`	Gravar arquivo .WAV em MS Windows 3.1

A.16 Funções para Strings

Geral	
`abs`	Converte string para valores numéricos
`blanks`	Cria string com espaços brancos
`deblank`	Remove espaços brancos e nulos de um string
`eval`	Executa string como uma expressão MATLAB
`findstr`	Buscar um string em uma expressão MATLAB
`isstr`	Verdadeiro para strings
`setstr`	Converte valores numéricos para string
`str2mat`	Forma matriz de textos
`string`	Cadeia de carateres em MATLAB
`strrep`	Pesquisa e substitui strings
`strtok`	Primeiro caracter em um string

Comparando Strings	
`isletter`	Verdadeiro para caracter alfabético
`lower`	Converte string a letras minúsculas
`strcmp`	Compara strings
`upper`	Converte string para letras mayúsculas

Conversion String para Números	
`int2str`	Converte inteiro para string
`num2str`	Converte números para string
`sprintf`	Converte número para strng, com formato
`sscanf`	Converte string para número, com formato
`str2num`	Converte string para número

Conversion Hexadecimal para Números	
`dec2hex`	Converte decimal inteiro para string hexadecimal
`hex2dec`	Converte string hexadecimal para decimal inteiro
`hex2num`	Converte string hexadecimal para número IEEE

A.17 Funções I/O para arquivos

Abrir-Fechar Arquivos	
`fclose`	Fecha um arquivo
`fopen`	Abre um arquivo

I/O sem formato	
`fread`	Lê um dado binário de um arquivo
`fwrite`	Escreve um dado binário para um arquivo

I/O com formato	
`fgetl`	Lê uma linha de um arquivo
`fgets`	Lê uma linha de uma arquivo
`fprintf`	Escreve dado formatado para um arquivo
`fscanf`	Lê dado formatado de um arquivo

Posisionamento em Arquivos	
`feof`	Teste de fim de arquivo
`ferror`	Status de erro i/O
`frewind`	Rebobinar arquivo
`fseek`	Vai para uma posição em um arquivo
`ftell`	Obtém posição dentro de um arquivo

Conversão de Strings	
`sprintf`	Escreve dado formatado como string
`sscanf`	Lê string com formato

I/O Especializadas	
`csvread`	Lê um arquivo de valores separados por vírgulas
`csvwrite`	Escreve um arquivo de valores separados por vírgulas
`uigetfile`	Obter nome de arquivo (menu)
`uiputfile`	Escrever nome de arquivo (menu)
`wk1read`	Lê um arquivo de dados .WK1 (Lotus 1-2-3)
`wk1write`	Escreve um arquivo de dados .WK1 (Lotus 1-2-3)

Bibliografia

[1] A. Castro, *Manual do Usuário para MATLAB 3.5.* CPGCC, Instituto de Informática, UFRGS, Porto Alegre, 1989.

[2] J. R. Claeyssen, *Métodos Operacionais e Computacionais em Álgebra Matricial.* Instituto de Matemática, UFRGS, Caderno Série B No. 26, Porto Alegre, 1995.

[3] B. P. Demidovich, I.A.Maron, *Computational Mathematics.* MIR, 1987.

[4] D. Hill, C. B. Moler. *Experiments in Computational Matrix Algebra.* McGraw Hill, New York, 1988.

[5] A. Jennings, *Matrix Computation for Engineers and Scientists.* John Wiley, 1977.

[6] G. Lindfield, J. Penny. *Numerical Methods Using MATLAB.* Prentice Hall, 1995.

[7] *MATLAB Building a Graphical User Interface.* The MathWorks, June 1993.

[8] *MATLAB Release Notes version 4.2c.* The MathWorks, 1994.

[9] *MATLAB User's Guide.* The MathWorks, Feb. 1993.

[10] G. Strang, *Introduction to Linear Alegbra.* Wellesley- Cambrigde, 1993.

[11] *The Student Edition of MATLAB, Version 4, User's Guide.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[12] Ch. Van Loan, T. Coleman, *Handbook for Matrix Computations.* Frontiers in Applied Mathematics Series, SIAM, 1988.