UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
CENTRO DE CIÊNCIAS EXATAS E NATURAIS/CCEN
Departamento de Tecnologia da Informação - TCI
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro dos Martins
Maceió/AL - Brasil CEP: 57.072-970 CGC: 24.464.109/0001-48
Telefone:(082) 214-1401 / FAX:(082) 214-1615
E-mail: tec-inf@dcc.ufal.br

UMA INTRODUÇÃO AO MATLAB

Eliana Silva de Almeida
e-mail: ea@fapeal.br

AGOSTO-2001

I. INTRODUÇÃO

O MATLAB ("MATrix LABoratory) é um software de computação científica que tem como característica ser um ambiente de programação interativa. Como um ambiente de programação, o MATLAB possui seu próprio editor de programas, simplificando bastante o seu uso e seus programas são escritos e gravados em arquivos, chamados "script files", cujos nomes possuem extensão ".m".

Este texto tem por objetivo apresentar uma introdução ao uso do MATLAB para a disciplina de Introdução a Computação do curso de graduação em Engenharia Civil. Ele conterá um resumo de como utilizar este ambiente no seu modo interativo, ao tratar da manipulação de matrizes e inteiros, e também apresentará os comandos básicos de programação, permitindo a compreensão de como um programa ou uma função pode ser construída em MATLAB.

Após a sua execução (entrada no ambiente), o MATLAB apresenta em sua janela inicial a primeira linha de comando iniciada através do prompt:

```
>>
```

O prompt indica que o MATLAB está esperando que seu usuário digite algum comando para ser interpretado. Por exemplo:

- Para sair do ambiente MATLAB basta digitar "exit" na linha de comando:

```
>> exit
```

- Para entender o significado de um comando, utiliza-se o comando "help":

```
>> help nome-do-comando
```

Neste texto deixaremos implícito para o leitor a indicação de que a tecla "ENTER" deverá ser digitada após cada comando para que o mesmo seja interpretado.

II. CONSTRUÇÃO DE MATRIZES:

O MATLAB possui vários tipos de matrizes que são construídas no sistema. Por exemplo, uma matriz 7x7 com entradas aleatórias (random) é produzida digitando:

```
>> rand(7)
```

Uma matriz com entrada aleatória, de outro tamanho, por exemplo, uma matriz 2x5, basta digitar:

>> rand(2,5)

Para conhecer mais sobre o comando “rand” digite:

>> help rand

Algumas matrizes padrões da álgebra linear são facilmente produzidas, por exemplo:

>> eye(6)

Produz uma matriz 6x6 cujos elementos da diagonal principal são iguais a 1 e todos os outros iguais a zero.

>> zeros(4,7)

Produz uma matriz 4x7 cujos elementos são iguais a 0.

>> ones(5)

Produz uma matriz 5x5 cujos elementos são iguais a 1.

É possível também construir matrizes com qualquer tamanho e com qualquer entrada desejada da seguinte forma:

>> [1 2 3 5 7 9]

Produz uma matriz com 1 linha e 6 colunas.

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

Produz uma matriz 3x3. Observe que o “;” separa as linhas da matriz. Logo, no exemplo, a primeira linha é composta pelos elementos “1 2 3”, a Segunda por “4 5 6” e a terceira por “7 8 9”. Estes elementos podem ser separados por “,” ou por um espaço.

>> [1 2 “ENTER” 3 4 “ENTER” 5 6]

Neste exemplo, temos uma matriz 3x2, onde a separação das linhas está sendo realizada através do “ENTER”.

A sintaxe do MATLAB também facilita a construção de matrizes a partir de matrizes existentes. Por exemplo:

>> [eye(2);zeros(2)]

>> [eye(2),ones(2,3)]

III. VARIÁVEIS:

O MATLAB possui variáveis pré-definidas tais como pi e ans. Podemos saber seus respectivos valores, a partir do interpretador:

```
>> pi
```

Aparecerá como resposta o valor de pi (3,1416)

```
ans =

   3.1416
```

A qualquer momento que se deseje saber quais variáveis estão ativas, utiliza-se o comando "who".

```
>> who
```

Para saber mais sobre este comando, basta utilizar o help:

```
>> help who
```

A variável "ans" irá informar a última saída que não foi atribuída a nenhuma outra variável. Por exemplo:

```
>> eye (4)

ans =

    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1

>> x = ans

x =

    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1


>> x = [x, eye(4)]
```

x =

1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1

Desde que uma nova variável, x, foi criada, ela aparecerá como uma variável ativa:

```
>> who
```

Your variables are:

ans x

Para remover uma variável, digita-se o comando “clear”:

```
>> clear x
```

IV. FUNÇÕES:

Seja a uma matriz definida por:

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

a =

1	2	3
4	5	6
7	8	9

a) A matriz transposta de a é definida por:

```
>> a'
```

b) A multiplicação de todos os elementos de a por um número:

```
>> 3*a
```

c) Multiplicação por –1:

```
>> -a
```

d) A operação soma:

```
>> a+(-a)
```

e) A função abaixo retorna a linha onde o valor máximo da matriz ocorre:

```
>> b = max(a)

b =
    7     8     9

>> max(b)

ans =
     9
```

f) Esta função max também pode retornar mais de um valor, por exemplo:

```
>> [m, i] = max(b)

m =
    9

i =
    3
```

g) Outros exemplos:

```
>> b = 2*a

b =
    2     4     6
    8    10    12
   14    16    18

>> a*b

ans =
   60    72    84
  132   162   192
  204   252   300
```

O MATLAB tem a convenção de que o ponto na frente de uma operação altera o significado da operação. Por exemplo, no caso da multiplicação, a.*b executará a multiplicação entrada por entrada no lugar da multiplicação usual entre matrizes:

```
>> a.*b

ans =
    2     8    18
   32    50    72
   98   128   162
```

```
>> x = 5

>> x^2  (fornece o quadrado de x)

ans =
    25

>> a*a

>> a^2

ans =
   30   36   42
   66   81   96
  102  126  150

>> a.^2   ((outro ponto) fornece o quadrado de cada elemento de a)

ans =
    1    4    9
   16   25   36
   49   64   81
```

>> triu(a) (gera uma matriz com os elementos acima da diagonal principal de a e zera os elementos que estão abaixo)

>> tril(a) (gera uma matriz com os elementos abaixo da diagonal principal de a e zera os elementos que estão acima)

>> diag(a) (fornece os elementos da diagonal principal)

>> diag(diag(a)) (gera uma matriz com os elementos da diagonal principal de a e com zeros nas outras posições)

```
>> c=rand(4,5)

>> size(c) (fornece o tamanho da matriz c)

>> [m,n] = size(c)
```

h) Algumas funções aplicadas aos escalares podem também ser aplicadas as matrizes:

```
>> sin(d)

>> exp(d)
```

```
>> log(d)

>> abs(d)
```

i) funções que permitem “arredondar”. São elas: round, fix, ceil e floor.

```
>> f=[-.5 .1 .5]

f =
   -0.5000    0.1000    0.5000

>> round(f)

ans =
    -1     0     1

>> fix(f)

ans =
     0     0     0

>> ceil(f)

ans =
     0     1     1

>> floor(f)

ans =
    -1     0     0

>> sum(f)

ans =
    0.1000

>> prod(f)

ans =
   -0.0250
```

Apenas para relembrar, observe que, quando atribuímos o resultado da execução de uma função ou operação a uma variável, a variável “ans” não aparece no resultado da interpretação. Ela só aparece quando a atribuição não é utilizada.

V. RELAÇÕES E OPERAÇÕES LÓGICAS:

Nesta seção devemos imaginar que 1 corresponderá ao valor verdade “verdadeiro” (“true”) e 0 ao valor verdade “falso”(“false”) – é desta forma que o MATLAB interpreta os valores verdade. As notações &, |, ~ correspondem a "and", "or" e "not", respectivamente, e a notação == corresponde a igualdade (lembre-se que “=” corresponde ao comando de atribuição).

```
>> a=[1 0 1 0]

>> b=[1  1  0  0]

>> a==b

>> a<=b

>> ~a

>> a&b

>> a & ~a

>> a | b

>> a | ~a
```

VI. NOTAÇÃO UTILIZANDO O DOIS PONTOS (“ : “)

Através desta notação o MATLAB permite criar arrays e tirar proveito deles, como por exemplo:

```
>> x=-2:1

x =
   -2    -1     0     1

>> length(x)

ans =
       4

>> 3:.5:5
```

```
ans =
   3.0000    3.5000    4.0000    4.5000    5.0000

>> a = [1:3;4:6;7:9]

a =
    1    2    3
    4    5    6
    7    8    9

>> a(2,:)

ans =
    4    5    6

>> a(:,2)

ans =
    2
    5
    8

>> a(2:3,:)

>> a(:,2:3)

>> a(2:3,1:2)

>> a(1:2:3,:)
```

Para fazer atribuições:

```
>> b=rand(3)

b =
   0.0579    0.0099    0.1987
   0.3529    0.1389    0.6038
   0.8132    0.2028    0.2722

>> b([1 2],:)=a([1 2],:)

b =
   1.0000    2.0000    3.0000
   4.0000    5.0000    6.0000
   0.8132    0.2028    0.2722
```

a(:,[1 2])=b(:,[2 3])

VII. OBSERVAÇÕES GERAIS:

1. MATLAB é "case sensitive", isto é, "a" não é o mesmo que "A". A função "casesen" faz esta propriedade ser ativada ou desativada.

2. display do MATLAB mostra apenas 5 dígitos no seu modo padrão. O MATLAB sempre armazena e computa em precisão dupla com 16 dígitos decimais e arredonda o display para 4 dígitos. Logo, quando um número é inteiro, o MATLAB o mostra como um inteiro mas, no caso de um número real, o MATLAB mostra uma aproximação com até 4 casas decimais e caso os dígitos significativos estejam além desta faixa, o MATLAB os apresentará em notação científica. O comando:

>> format long

Apresenta o display com todos os 16 dígitos e o comando:

>> format short

retorna ao display reduzido. É possível também colocar o display para notação científica com os comandos

>> format short e
e
>> format long e

3. Nem, sempre é necessário para o MATLAB apresentar os resultados de um comando na tela. Para isto, basta colocar um ponto e vírgula, " ; ", após o comando. Desta forma, após o "ENTER", o prompt do MATLAB irá aparecer sem que o resultado da execução do comando seja mostrado na tela. Desta forma, podemos escrever vários comandos em uma mesma linha, separados por ponto e vírgula, e após suas respectivas interpretações, nada ser mostrado na tela. Por exemplo:

```
>> x=1; y=2; z=5;
>> soma = x + y + z

soma =
    8
```

Observe que o mesmo não acontece se as atribuições fossem separadas por vírgula:

>> x = 1, y = 2, z = 5

```
x =
    1

y =
    2

z =
    5

>> x = 1, y = 2; z = 5

x =
    1

z =
    5
```

Neste último exemplo, o valor de y não foi mostrado pois está seguido por ponto e vírgula.

4. Algumas vezes, em uma seção do MATLAB são criadas várias variáveis que serão necessárias em uma outra vez em que o MATLAB for utilizado. É possível salvar os valores destas variáveis em um arquivo digitando:

```
>> save nome-do-arquivo
```

Este comando irá criar um arquivo do tipo nome-do-arquivo.mat que, para ser utilizado posteriormente, basta utilizar o comando

```
>> load nome-do-arquivo
```

VIII. PROGRAMANDO NO MATLAB:

O MATLAB é também uma linguagem de programação. Criando um arquivo com a extensão .m é possível facilmente escrever e executar programas. Uma vez que um arquivo de programa "nome.m" no MATLAB é criado, este arquivo será considerado um comando do MATLAB e será executado como qualquer outro comando ou função do MATLAB. Não é necessário compilar o programa uma vez que o MATLAB é uma linguagem interpretada. Este arquivo é chamado de m-file.

Algumas construções básicas de programação:

a) ATRIBUIÇÃO

```
x = a
```

Exemplo: DEFINIÇÃO DE FUNÇÃO:

```
function r=mod(a,d)

% r=mod(a,d): Se a e d são inteiros, então
% r é o resto da divisão de a por d.
% Se a and d são matrizes inteiras então,
% r  é a matriz dos restos após a divisão correspondente.
% rem(a,d) é a função que retorna o resto da divisão de a por d.

r=a-d.*floor(a./d);
```

Este programa deverá ser armazenado em um arquivo com o mesmo nome da função, mod.m. Logo, está função mod pode ser executada a partir da linha de comando do MATLAB:

```
>> mod(5,3)
```

Ao digitar “help mod”, será apresentada as linhas de comentários da função “%”, que é ignorada quando o programa está sendo executado.

b) COMANDO CONDICIONAL:

```
if < condition>, <program> end
```

ou

```
if <condition1>, <program1> else <program2> end
```

Outra variação é:

```
if <condition1>, <program1>
elseif <condition2>, <program2>
end
```

Neste caso, se condition1 não for 0, o program1 é executado e, se condition1 for 0 e se condition2 não for 0, então program2 é executado e, caso contrário, o controle é passado para a próxima construção.

Exemplo:

```
function b=even(n)

% b=even(n). If n for um número par, então b=1
% senão, b=0.
```

```
if mod(n,2)==0,
   b=1;
   else b=0;
end
```

c) REPETIÇÃO DEFINIDA (For Loops): comando for:

O comando for é uma construção da forma:

for i=1:n, <program>, end

Este comando indica que <program> será repetido um número definido de vezes, número este indicado por cada valor assumido pelo índice i, que neste caso irá variar de 1 (menor valor) até n (maior valor) . Alguns exemplos são:

Uma função para adicionar duas matrizes:

```
function c=add(a,b)
% esta função soma duas matrizes, a e b.
% tem a mesma função da função do MATLAB a+b.
[m,n]=size(a);
[k,l]=size(b);
if m~=k | n~=l,
   r='ERRO no uso do add: as matrizes não possuem a mesma dimensão';
   return,
end
c=zeros(m,n);
for i=1:m,
   for j=1:n,
      c(i,j)=a(i,j)+b(i,j);
   end
end
```

função que fornece a multiplicação de duas matrizes:

```
function c=mult(a,b)
% esta função multiplica duas matrizes, a e b.
% tem a mesma função da função do MATLAB a*b.
[m,n]=size(a);
[k,l]=size(b);
if n~=k,
   c='ERRO no uso do mult: as matrizes não são compatíveis para realizar uma
multiplicação';
   return,
```

```
end,
c=zeros(m,l);
for i=1:m,
  for j=1:l,
    for p=1:n,
      c(i,j)=c(i,j)+a(i,p)*b(p,j);
    end
  end
end
```

Vale observar que as mensagens de erro são apresentadas entre ` `. tudo que se encontra entre ` ` deve ser tratado como texto. Nos exemplos, as mensagens correspondem ao conteúdo da variável c.

O comando RETURN indica o retorno do controle do programa para a função que o chamou ou o retorno para o prompt. Recomenda-se que o return seja usado unicamente no contexto de uma mensagem de erro.

Na construção "for i=1:n, <program>, end ", o índice i pode ocorrer dentro do programa. O MATLAB permite que seja colocado qualquer vetor no lugar do vetor 1:n.

Logo, a construção:

for i=[2,4,5,6,10], <program>, end

é perfeitamente legítima e, nesse caso, "program" é executado 5 vezes e os valores assumidos por i durante a execução são 2, 4, 5, 6 e 10, sucessivamente.

d) REPETIÇÃO INDEFINIDA (While Loops):

O comando while é uma construção da forma:

while <condition>, <program>, end

Onde "condition" é uma função do MATLAB como uma construção condicional que deve retornar o valor 1 (verdadeiro) ou 0 (falso). O programa "program" será executado sucessivamente enquanto o valor da condição "condition" seja diferente de 0.

SCRIPTS

Um script é um m-file sem a declaração de função no topo. Um script se comporta deferente da função pois não é necessário retornar um valor como na função. Para se construir um script, da mesma forma que uma função, utiliza-se o editor do matlab, escreve a sequencia de comandos e salva com um nome desejado. Para executar um script basta escrever o nome do arquivo que o contém no prompt do MATLAB.

Ao utilizar um script, muitas vezes tem-se a necessidade de fazer uso de comandos de entrada e saída. Apresentaremos a seguir duas funções que podem ser utilizados para este fim.

e) FUNÇÃO DE ENTRADA:

R = input (‘mensagem de texto’)
Esta função mostra na tela a mensagem de texto e aguarda que o usuário digite um valor ou uma expressão que será avaliada e armazenada na variável R.

R = input (‘mensagem de texto’ , ‘s’)
Esta função mostra na tela a mensagem de texto e aguarda que o usuário digite um texto (string) que será armazenado (mas não alvaliado) na variável R.

f) FUNÇÃO DE SAÍDA:

disp (X) apresenta o conteúdo da matriz X na tela, sem mostrar o seu nome. Se X é um texto (string) este também é mostrado.

Exemplo:

```
>>disp ([12 13 15])

    12   13   15

>> a = 14

 a =
    14

>>disp (a)

    14

>>disp (‘maria dd’)

    maria dd
```

f) FORMATAÇÃO NO COMANDO DISP

Os dados que são apresentados no comando de saída disp podem ser formatados utilizando a função “sprintf” cujo formato é:

Sprintf (<especificações de formato para cada variável da lista>, <lista de variáveis>)

Onde a especificação do formato tem a seguinte forma:

% I_1. I_2 f , onde I_1 é um número inteiro indicando o quanto de espaço existe para escrever o número e I_2 é um número inteiro indicando o quanto deste espaço definido por I_1 será considerado como a parte decimal.

Ex:

```
>> disp(sprintf (‘ %2.0f  %3.0f   %6.3f ‘ , K, D, y(K)))

>> disp (sprintf (‘um grau = %5.3e radianos’, pi/180))
```

f) SUGESTÕES:

- Utilize sempre o estilo de programação indentado. Isto permite uma maior legibilidade do programa, facilitando sua leitura e também a descoberta de erros de sintaxe.
- Coloque o máximo de comentários no programa para facilitar a sua compreensão.
- Procure sempre estruturar o programa de forma tal que o seu resultado seja entrada para alguma outra função.
- Utilize o máximo possível funções pré-definidas do MATLAB. Lembre-se que o MATLAB é otimizado para executar estas funções.

g) PROGRAMAS DEMONSTRATIVOS NO MATLAB:

O MATLAB possui um grande número de programas demonstrativos. Para acessar estes programas digite:

```
>>demo
```

IX. LISTA DE ALGUMAS FUNÇÕES DEFINIDAS NO MATLAB:

A seguir será apresentada uma lista com algumas funções que estão disponíveis no MATLAB e que pode ser consultada como um simples lembrete. Para saber os detalhes de cada função basta utilizar o comando “help”.

```
intro   <       chol    end     function lu      quit    sprintf
help    >       clc     eps     global   macro   qz      sqrt
demo    =       clear   error   grid     magic   rand    startup
[       &       clg     eval    hess     max     rcond   string
]       |       clock   exist   hold     memory  real    subplot
(       ~       conj    exit    home     mesh    relop   sum
)       abs     contour exp     ident    meta    rem     svd
.       all     cos     expm    if       min     return  tan
,       ans     cumprod eye     imag     nan     round   text
;       any     cumsum  feval   inf      nargin  save    title
%       acos    delete  fft     input    norm    schur   type
!       asin    det     filter  inv      ones    script  what
:       atan    diag    find    isnan    pack    semilogx while
'       atan2   diary   finite  keyboard pause   semilogy who
```

```
+       axis    dir     fix     load    pi      setstr  xlabel
-       balance disp    floor   log     plot    shg     ylabel
*       break   echo    flops   loglog  polar   sign    zeros
\       casesen eig     for     logop   prod    sin
/       ceil    else    format  ltifr   prtsc   size
^       chdir   elseif  fprintf ltitr   qr      sort

acosh   demo    hankel  membrane print   table1
angle   demolist hds    menu    quad    table2
asinh   dft     hilb    meshdemo quaddemo tanh
atanh   diff    hist    meshdom quadstep tek
bar     eigmovie histogram mkpp  rank    tek4100
bench   ergo    hp2647  movies  rat     terminal
bessel  etime   humps   nademo  ratmovie toeplitz
bessela expm1   idft    nelder  readme  trace
besselh expm2   ieee    neldstep residue translate
besseln expm3   ifft    nnls    retro   tril
blanks  feval   ifft2   null    roots   triu
cdf2rdf fft2    info    num2str rot90   unmkpp
census  fftshift inquire ode23  rratref  vdpol
citoh   fitdemo int2str ode45   rratrefmovie versa
cla     fitfun  invhilb odedemo rref    vt100
compan  flipx   isempty orth    rsf2csf vt240
computer flipy  kron    pinv    sc2dc   why
cond    funm    length  plotdemo sg100   wow
conv    gallery log10   poly    sg200   xterm
conv2   gamma   logm    polyfit sinh    zerodemo
corr    getenv  logspace polyline spline  zeroin
cosh    ginput  matdemo polymark sqrtm
ctheorem gpp    matlab  polyval square
dc2sc   graphon mean    polyvalm std
deconv  hadamard median  ppval   sun

addtwopi buttap  cov     fftdemo  freqz   kaiser  specplot
bartlett butter  decimate filtdemo fstab  numf    spectrum
bilinear chebap  denf    fir1     hamming readme2 triang
blackman chebwin detrend fir2     hanning remez   xcorr
boxcar  cheby   eqnerr2 freqs   interp  remezdd xcorr2
yulewalk

>> help
tópicos do HELP:

matlab/general      - General purpose commands.
matlab/ops          - Operators and special characters.
matlab/lang         - Language constructs and debugging.
```

matlab/elmat - Elementary matrices and matrix manipulation.
matlab/specmat - Specialized matrices.
matlab/elfun - Elementary math functions.
matlab/specfun - Specialized math functions.
matlab/matfun - Matrix functions - numerical linear algebra.
matlab/datafun - Data analysis and Fourier transform functions.
matlab/polyfun - Polynomial and interpolation functions.
matlab/funfun - Function functions - nonlinear numerical methods.
matlab/sparfun - Sparse matrix functions.
matlab/plotxy - Two dimensional graphics.
matlab/plotxyz - Three dimensional graphics.
matlab/graphics - General purpose graphics functions.
matlab/color - Color control and lighting model functions.
matlab/sounds - Sound processing functions.
matlab/strfun - Character string functions.
matlab/iofun - Low-level file I/O functions.
matlab/demos - The MATLAB Expo and other demonstrations.
toolbox/chem - Chemometrics Toolbox
toolbox/control - Control System Toolbox.
fdident/fdident - Frequency Domain System Identification Toolbox
fdident/fddemos - Demonstrations for the FDIDENT Toolbox
toolbox/hispec - Hi-Spec Toolbox
toolbox/ident - System Identification Toolbox.
toolbox/images - Image Processing Toolbox.
toolbox/local - Local function library.
toolbox/mmle3 - MMLE3 Identification Toolbox.
mpc/mpccmds - Model Predictive Control Toolbox
mpc/mpcdemos - Model Predictive Control Toolbox
mutools/commands - Mu-Analysis and Synthesis Toolbox.: Commands directory
mutools/subs - Mu-Analysis and Synthesis Toolbox -- Supplement
toolbox/ncd - Nonlinear Control Design Toolbox.
nnet/nnet - Neural Network Toolbox.
nnet/nndemos - Neural Network Demonstrations and Applications.
toolbox/optim - Optimization Toolbox.
toolbox/robust - Robust Control Toolbox.
toolbox/signal - Signal Processing Toolbox.
toolbox/splines - Spline Toolbox.
toolbox/stats - Statistics Toolbox.
toolbox/symbolic - Symbolic Math Toolbox.
toolbox/wavbox - (No table of contents file)
simulink/simulink - SIMULINK model analysis and construction functions.
simulink/blocks - SIMULINK block library.
simulink/simdemos - SIMULINK demonstrations and samples.
toolbox/codegen - Real-Time Workshop

Para cada um destes tópicos é bastante digitar o help seguido do tópico desejado e obter todos os detalhes a cerca de cada um dos ítens acima.

X. REFERÊNCIAS:

R. Smith, "MATLAB Tutorial". http://www.math.ufl.edu/help/matlab-tutorial/index.html#SEC5, novembro, 1988.

D. Hanselman & B. Littlefield, "MATLAB – Versão Estudante – Guia do Usuário – Versão 4", MAKRON Books do Brasil. São Paulo, 1997.

C.F. Van Loan – "Introduction to Scientific Computing"