# Freemat: Uma introdução

Ulysses Sodré
Andrielber da Silva Oliveira
Talita Paiva Corrêa
Leandro Meneses da Costa

Londrina-PR, 20 de Maio de 2009

# Conteúdo

# 1 Abertura do Freemat

Freemat é um programa de cálculo numérico para Matemática e ciências afins, que funciona interativamente, baseado em matrizes, permitindo a visualização gráfica de vários dos seus objetos. O Freemat é rápido e simples, resolvendo problemas numéricos em tempo menor que o exigido em uma linguagem de programação como C ou Fortran.

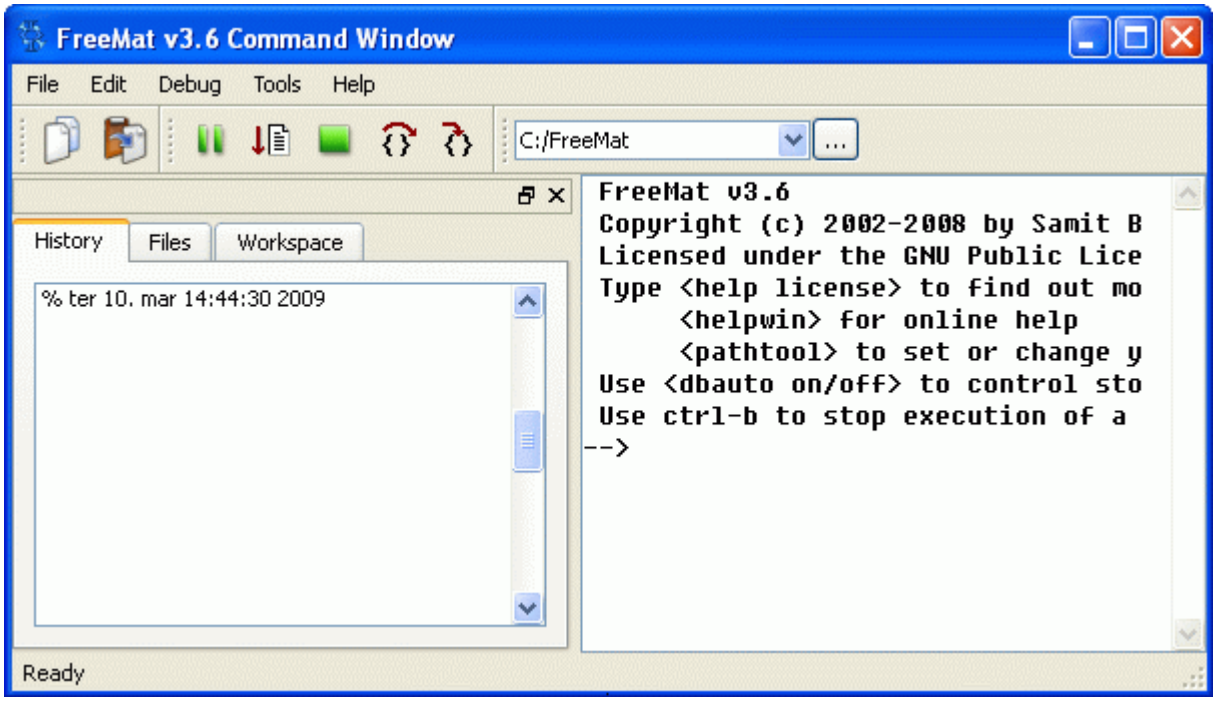
# 2 Iniciando o Freemat

a. Em muitos sistemas como Linux e Windows, podemos iniciar o programa Freemat, digitando `freemat` na linha de comando ou acionando o ícone relacionado ao Freemat. Para encerrar o Freemat, basta digitar na linha de comando a palavra `quit` ou `exit`.

b. O Freemat possui um editor interno de texto, que pode ser acionado pela digitação da palavra `editor` na linha de comando ou pelo pressionar das teclas `Ctrl+E`. Tal editor serve para editar arquivos `.m` que são muito usados no Freemat.

c. O Freemat contém um *notebook* onde realizamos operações matemáticas. A todo momento podemos escrever, apagar, editar qualquer parte e depois salvar o arquivo como um documento de texto.

d. Ao abrir o Freemat, aparece uma tela semelhante a:




e. No Freemat, o usuário pode definir novas variáveis, scripts e funções e usar tudo isto de modo natural.

f. O Freemat mantém uma lista visível (à esquerda na tela) onde as funções, variáveis e estruturas ficam armazenadas junto com os seus valores. Pode-se plotar gráficos, analisar e salvar os mesmos.

g. O Freemat é leve e não exige muito da sua máquina, funcionando em quase todos os sistemas computacionais.

h. Este tutorial é uma introdução ao Freemat, que possui muitas outras qualidades e feitos que não apresentamos. Você deve digitar o código sugerido neste material para obter os resultados indicados.

i. Usando `help assunto` na linha de comando do Freemat, obtemos uma ajuda on-line com muita informação. Digitando na linha de comando `help eig` obtemos informações sobre a função `eig`. O próprio comando `help` mostra uma lista de tópicos disponíveis.

# 3 Cálculos Básicos

a. Para realizar cálculos básicos no Freemat, basta inserir as operações nas formas usuais, lembrando que parênteses alteram a precedência nas operações realizando primeiro as operações envolvidas por eles.

b. Para ganhar espaço no tutorial, optamos por indicar várias linhas de comando com as suas respectivas respostas, sem colocar os espaços que o Freemat mostra na tela.

c. Alguns cálculos com números reais.

Freemat: Linhas de comando

```
2 * 3 + 4 / 4 - 3 ^ 2
                ans = -2
2*(3+4)
                ans = 14
(2+3)*4
                ans = 20
2.1+3.99+1.25
                ans = 7.3400
1.5 * 2^2
                ans = 6
(1.5 * 2)^2
                ans = 9
5*(2+3)+2^(2+2)
                ans = 41
(5+2/3)^2 / (1+(4/2+1))
                ans = 8.0278
- 3^2 + (-3)^2
                ans = 0
```

d. Alguns cálculos com números complexos, usando a unidade imaginária `i`:

Freemat: Linhas de comando

```
i^2
                ans = -1.0000 +  0.0000i
sqrt(-1)
                ans =  0.0000 +  1.0000i
sqrt(-25) + 3.5
                ans =  3.5000 +  5.0000i
```

e. *Nota*: Quando não está indicada a variável, o Freemat armazena cada resposta nova na variável `ans` (answer).

f. Algumas variáveis e operações complexas com elas.

Freemat: Linhas de comando

```
z1 = 2+6i
                z1 =   2.0000 +  6.0000i
z2 = 4+3i
                z2 =   4.0000 +  3.0000i
z1 + z2
                ans =  6.0000 +  9.0000i
z1 / (z2-3)
                ans =  2.0000 +  0.0000i
3 * z1
                ans =  6.0000 + 18.0000i
z1^2
                ans =-32.0000 + 24.0000i
sqrt(z1)
                ans =  2.0402 +  1.4705i
z1^3
                ans = -2.0800e+002 -1.4400e+002i
z1^(1/3)
                ans = 1.6913 +  0.7479i
real(z1)
                ans = 2
imag(z1)
                ans = 6
real(5.5)
                ans = 5.5000
imag(5.5)
                ans = 0
abs(z1)
                ans = 6.3246
conj(z1)
                ans = 2.0000 -  6.0000i
angle(z1)
                ans = 1.2490
```

g. Inserimos comentários no Freemat, com o caracter % (porcentagem) antes do texto.

Freemat: Linhas de comando

```
% Este é um comentário: Soma de 4 com 2:
4+2
        ans = 6
```

h. Para limpar a tela do ambiente de trabalho, digite

Freemat: Linhas de comando

```
clc
```

# 4 Trabalhando com constantes e com variáveis

a. O Freemat possui algumas constantes embutidas, como por exemplo: `inf` infinito, `nan` (Not-a-Number), `i` ou `j` raiz quadrada de `-1`, `pi` $= 3,1416$, `e` $= 2,7183$ constante de Euler (base do logaritmo natural), `eps` constante da máquina de dupla precisão em ponto flutuante, `true` valor lógico verdadeiro, `false` valor lógico falso.

Freemat: Linhas de comando

```
pi
     ans = 3.1416
e
     ans = 2.7183
eps
     ans = 2.2204e-016
```

b. A constante `eps` (epsilon) indica à máquina uma unidade de arredondamento da ordem de `10^(-16)`, que é útil em operações de processos de convergência iterativa.

c. Vamos fazer alguns cálculos com tais constantes:

Freemat: Linhas de comando

```
% Comentário: Area do circulo de raio 3.5
pi*3.5^2
          ans = 38.484510006
```

d. Além das constantes usuais, podemos definir novas constantes ou variáveis.

e. Vamos definir algumas variáveis e realizar operações com elas. Observe como a variável deve ser definida:

Freemat: Linhas de comando

```
Base = 4.5; Altura = 10;
Base * Altura
                    ans = 45
(Base * Altura)/2
                    ans = 22.5
```

f. *Nota*: O sinal de ; (ponto e vírgula) no final da expressão, faz com que o Freemat não mostre a resposta do cálculo.

g. Podemos eliminar (=deletar=limpar) a variável `Base` criada por nós, com:

Freemat: Linhas de comando
```
clear Base
```

h. O comando `clear` sozinho elimina todas as variáveis que não são permanentes.

i. O Freemat não informa que a variável `Base` foi eliminada da memória do computador, mas executando uma operação com a variável `Base`, como:

Freemat: Linhas de comando
```
Base * 2
              Error: unable to resolve Base to a function call
```

o Freemat informa que a variável `Base` não mais está definida.

j. *Dica*: Para criar uma nova constante, basta definir a mesma como uma variável comum e tomar o cuidado de não redefinir a mesma durante uma operação com ela.

Freemat: Linhas de comando
```
A = 3.5; B2 = 2.25; B3 = 2;
A*(B2+B3)
              ans = 14.8750
```

# 5 Funcões pré-definidas no Freemat

a. O Freemat possui muitas funções pré-definidas embutidas. Podemos acessar uma lista destas funções usando:

Freemat: Linhas de comando
```
=[Tecla TAB]
```

b. Três funções trigonométricas (variável em radianos): `sin()`, `cos()`, `tan()`, etc.

Freemat: Linhas de comando
```
[ sin(pi), sin(pi/2), sin(pi/3), sin(pi/4) ]
              ans = 0.0000    1.0000    0.8660    0.7071
[ cos(pi), cos(pi/2), cos(pi/3), cos(pi/4) ]
              ans = -1.0000    0.0000    0.5000    0.7071
[ tan(pi), tan(pi/2), tan(pi/3), tan(pi/4) ]
              ans = 1.0e+016 *
                    -0.0000    1.6332    0.0000    0.0000
```

c. *Nota*: A notação `1.0e+016 *` na linha anterior indica que todas as respostas devem ser multiplicadas por `1.0e+016`, isto é, `1` seguido de `16` zeros.

d. Três funções trigonométricas (variáveis em graus): `sind()`, `cosd()`, `tand()`, etc.

Freemat: Linhas de comando

```
[ sind(180), sind(90), sind(60), sind(45) ]
              ans = 0.0000   1.0000   0.8660   0.7071
[ cosd(180), cosd(90), cosd(60), cosd(45) ]
              ans = 1.0000   0.0000   0.5000   0.7071
[ tand(180), tand(90), tand(60), tand(45) ]
              ans = 1.0e+016 *
                              -0.0000   1.6332   0.0000   0.0000
```

e. Funções usadas para converter `radianos` em `graus` e `graus` em `radianos`.

Freemat: Linhas de comando

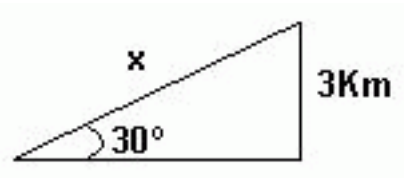
```
[ deg2rad(30), deg2rad(60), deg2rad(90), deg2rad(180) ]
              ans = 0.5236    1.0472    1.5708    3.1416
[ rad2deg(1), rad2deg(pi/2), rad2deg(pi), rad2deg(2*pi) ]
              ans = 1.0e+002 *
                 0.5730    0.9000    1.8000    3.6000
```

f. Podemos misturar todas estas funções nos cálculos;

Freemat: Linhas de comando

```
atand(sqrt(3))
              ans = 60.0000
sin(rad2deg(pi/2))
              ans = 1
sin(deg2rad(30))
              ans = 0.50000
sin(deg2rad(45))
              ans = 0.707106781
cos(deg2rad(90))
              ans = 0
```

g. *Exemplo*: Algo é lançado formando um ângulo de `30` graus com o chão. Se o ângulo é mantido fixo em toda a trajetória, qual é a distância percorrida pelo objeto quando a sua altura atinge 3km?




Como `seno(x) = CO/Hip`, ou seja, `sen(30) = 3000/x`, assim o valor de `x` é obtido por uma regra de três simples e realizando uma divisão.

Freemat: Linhas de comando

```
sind(30)
              ans = 0.5
x = 3000 / 0.5
              ans = 6000
```

h. *Exemplo*: Se `x` é um ângulo agudo tal que `sin(x)=1/3`, usamos a inversa da função seno para calcular `x` e depois obter `cos(x)` e `tan(x)`.

Freemat: Linhas de comando

```
a = asin(1/3)
              a = 0.3398
cos(a)
              ans = 0.9428
tan(a)
              ans = 0.3536
```

i. Estão disponíveis outras funções como: logaritmo, exponencial e raiz quadrada.

j. Podemos alterar o número de dígitos após a vírgula, com os comandos `format long` ou `format short`. Por exemplo:

Freemat: Linhas de comando

```
format short
[ pi, 3/4 ]
              ans = 3.1416    0.7500
format long
[ pi, 3/4 ]
              ans = 3.14159265358979    0.75000000000000
```

k. Para que o Freemat volte a operar com o formato padrão, digite:

Freemat: Linhas de comando

```
format short
```

l. O Freemat também pode utilizar a notação científica.

Freemat: Linhas de comando

```
format short e
[ 2000 + 300, 3/1000 ]
              ans = 2.3000e+003 3.0000e-003
format long e
[ 2000 + 300, 3/1000 ]
              ans = 2.30000000000000e+003 3.00000000000000e-003
```

m. Para que o Fremat volte a operar com o formato padrão, digite:

Freemat: Linhas de comando

```
format short
```

n. Caso não obtenha ajuda para alguma função, você pode definir a sua própria função inserindo comentários nas linhas iniciais do arquivo para obter ajuda on-line desta nova função no Freemat.

# 6 Definindo funções

a. Existem várias modos de definir funções no Freemat, sendo os mais comuns: *inline* e *anonymous*.

b. Agora usamos o formato *inline* para definir a função $f1(x) = 2x^2 - 3x + 7$:

Freemat: Linhas de comando

```
f1 = inline('2.*x.^2-3.*x+7','x')
f1 = inline function object
     f(x) = 2.*x.^2-3.*x+7
```

c. O Freemat informa que a função `f1` está definida na memória do computador, indicando este fato na janela `history` (da sessão), junto com as variáveis.

d. Agora, podemos realizar algumas operações com esta função:

Freemat: Linhas de comando

```
f1(3)
               ans = 16
f1(2/5)
               ans = 0.32
0.5 .* f1(3+4)
               ans = 49
```

e. Podemos *eliminar* a função `f1` da memória do computador com a função `clear`.

Freemat: Linhas de comando

```
clear f1
```

f. O Freemat não informa que a função `f1` foi eliminada da memória do computador, mas ao testar a função `f1` em um valor, obtemos a seguinte resposta:

Freemat: Linhas de comando

```
f1(5)
    Error: Undefined function or variable f1
```

g. Agora usamos a forma *anonymous* para definir a mesma função com o nome `f2` para trabalharmos com variáveis reais `x`:

Freemat: Linhas de comando

```
f2 = @(x) 2*x^2-3*x+7
f2 =
     @(x) 2*x^2-3*x+7
```

h. Para que esta função aceite uma *matriz* como argumento, devemos tomar mais cuidado na definição, inserindo alguns `pontos` em duas operações excepcionais:

Freemat: Linhas de comando

```
f2 = @(x) 2.*x.^2-3.*x+7
f2 =
     @(x) 2.*x.^2-3.*x+7
```

i. Agora podemos operar com esta função, como por exemplo:

Freemat: Linhas de comando

```
f2(3)
               ans = 16
f2(2/5)
               ans = 6.12
0.5 .* f2(3+4)
               ans = 42
x=[1,2;3,4];
f2(x)
                6  9
               16 27
```

j. O Freemat também suporta funções *inline* com duas variáveis, como:

Freemat: Linhas de comando

```
Area = inline('b.*h', 'b', 'h')
               f(b,h) = b.*h
[ Area(5, 3), Area(5, 3+2) ]
               ans = 15  25
```

k. Funções anônimas com duas variáveis possuem a forma mais simples:

Freemat: Linhas de comando

```
Area = @(b,h) b*h
               @(b,h) b*h
[ Area(5, 3), Area(5, 3+2) ]
               ans = 15  25
```

l. Funções anônimas com três variáveis possuem a forma:

Freemat: Linhas de comando

```
% Norma de um vetor 3D indicada por N3(a,b,c)
N3 = @(a,b,c) sqrt(a^2 + b^2 + c^2)
               @(a,b,c) sqrt(a^2 + b^2 + c^2)
[ N3(3,4,12), N3(5,7,24) ]
               ans =  13.0000   25.4951
```

m. Forma geral de função anônima: `func = @(v,a,r,i,e) expressão da função`.

n. Pesquise no `help` do Freemat sobre a palavra `hilb`, digitando na linha de comando:

Freemat: Linhas de comando

```
help hilb
```

Provavelmente você obterá a resposta

Freemat: Linhas de comando

```
Error: no help available on hilb
       nenhuma ajuda disponível sobre hilb
```

o. Como não encontrei ajuda no Freemat sobre a função `hilb` que gera uma matriz de Hilbert de ordem `mxn`, eu editei e salvei o arquivo `hilb.m` com o seguinte conteúdo:

Função: hilb.m

```
% HILB Matriz de Hilbert de ordem mxn
%
% Retorna uma matriz de Hilbert. A sintaxe é:
%
%   y = hilb(m,n)
%
% onde m e n são inteiros positivos.
% Função construída por Ulysses Sodré.

function z = hilb(m,n)
    for i = 1:m
        for j = 1:n
            z(i,j) = 1/(i+j-1);
        end
    end
```

p. Agora, pesquise de novo no `help` do Freemat para ver se encontra algo sobre a palavra `hilb`, com a mesma linha de comando usada antes:

Freemat: Linhas de comando

```
help hilb
```

Você deverá ver algo semelhante às palavras que estavam comentadas, de forma contígua, no início da função criada no arquivo `hilb.m`.

Função: hilb.m

```
 HILB Matriz de Hilbert de ordem mxn

 Retorna uma matriz de Hilbert. A sintaxe é:

   y = hilb(m,n)

 onde m e n são inteiros positivos.
 Função construída por Ulysses Sodré.
```

# 7 Dois pontos, Somas, Regras Trapezoidal e de Simpson

a. Usamos a notação com `dois pontos` para gerar um vetor com algum formato.

b. Um vetor com elementos de 1 até 10 com passo 1:

Freemat: Linhas de comando

```
x = [1 : 10];
    ans =  1  2  3  4  5  6  7  8  9 10
```

c. Um vetor com elementos de 1 até 10 com passo 1:

Freemat: Linhas de comando

```
x = [1 : 1 : 10]
    ans =  1  2  3  4  5  6  7  8  9 10
```

d. Um vetor com elementos de 1 até 10 com passo 2:

Freemat: Linhas de comando

```
x = [1 : 2 : 10]
    ans =  1 3 5 7 9
```

e. Um vetor com elementos de 1 até 8 com passo 0.5:

Freemat: Linhas de comando

```
x = [1 : 0.5 : 8]
x =
    Columns 1 to 5
      1.0000    1.5000    2.0000    2.5000    3.0000
    Columns 6 to 10
      3.5000    4.0000    4.5000    5.0000    5.5000
    Columns 11 to 15
      6.0000    6.5000    7.0000    7.5000    8.0000
```

f. Para não mostrar todos estes valores na tela, acrescente um ponto e vírgula no final da linha de comando:

Freemat: Linhas de comando

```
x = [1 : 0.5 : 10];
```

g. Um vetor com os quadrados dos elementos de 10 até 1 com passo -1:

Freemat: Linhas de comando

```
x = [10 : -1 : 1];
y = x.^2
    ans =  100  81  64  49  36  25  16   9   4   1
```

h. Um vetor com os senos dos elementos de 0 até pi com passo pi/4:

Freemat: Linhas de comando

```
x = [0 : pi/4 : pi];
y = sin(x)
y =
    0    0.7071    1.0000    0.7071    0.0000
```

i. Obtemos a soma dos 100 primeiros números naturais com o código:

Freemat: Linhas de comando

```
x = 1:100;
Soma = sum(x)
    ans = 5050
```

j. Obtemos a soma dos 100 primeiros números ímpares positivos com o código:

Freemat: Linhas de comando

```
N = 100;
x = 1:2:2*N-1;
Simp = sum(x)
    ans = 10000
```

k. Obtemos a soma dos 100 primeiros números pares positivos com o código:

Freemat: Linhas de comando

```
N = 100;
x = 2:2:2*N;
Spar = sum(x)
    ans = 10100
```

l. Obtemos a soma dos quadrados dos 100 primeiros números naturais com:

Freemat: Linhas de comando

```
x = 1:100;
Squad = sum(x.^2)
    ans = 338350
```

m. Obtemos a soma dos cubos dos 100 primeiros números naturais com:

Freemat: Linhas de comando

```
x = 1:100;
Scub = sum(x.^3)
    ans = 25502500
```

n. Obtemos a soma das raízes cúbicas dos 100 primeiros números naturais com:

Freemat: Linhas de comando

```
x = 1:100;
Scub = sum(x.^(1/3))
    ans = 3.5016e+002
```

o. Sabe-se que a integral definida da função `f(x)=x^2` no intervalo `[0,1]` é igual a `1/3`.

p. Construímos a Regra de Trapezoidal para calcular um valor aproximado da integral da função `f(x)=x^2` no intervalo `[0,1]` usando `41` subintervalos.

Freemat: Linhas de comando

```
a = 0; b = 1; N = 41;  % N pode ser um número natural PAR ou ÍMPAR
h = (b-a)/N;
y = inline('x.^2');
x = a:h:b;
f = y(x);
Si = sum(f(2:N));
Soma = (h/2)*(y(a) + 2*Si + y(b))
    Soma = 0.3334
```

q. Construímos a Regra de Simpson para calcular um valor aproximado da integral da função `f(x)=x^2` no intervalo `[0,1]` usando `N=20` subintervalos. (Este número N deve ser PAR)

Freemat: Linhas de comando

```
a = 0; b = 1; N = 20;  % N deve ser PAR
h = (b-a)/N;
y = inline('x.^2');
x = a:h:b;
f = y(x);
S1 = sum(f(2:2:N));
S2 = sum(f(3:2:N-1));
Soma = (h/3)*(y(a) + 4*S1 + 2*S2 + y(b))
    Soma = 0.3333
```

# 8 Inserindo Matrizes

a. O Freemat trabalha basicamente com matrizes reais ou complexas retangulares, sendo que todas as variáveis no Freemat são *matrizes*.

b. Algumas vezes, matrizes `1x1` operam como escalares. Matrizes contendo apenas uma linha ou apenas uma coluna são interpretados como vetores.

c. Matrizes são inseridas no Freemat: explicitamente, por declarações ou por arquivos.

d. Podemos criar explicitamente uma matriz de ordem `3x3`, associando a variável `A` a esta matriz, através do código:

Freemat: Linhas de comando

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

e. Os elementos em cada linha são separados por , vírgulas e as linhas são separadas por ; ponto e vírgula.

f. Evite espaços em branco ao digitar números como `2.34e-9` na forma exponencial. Não escreva na forma `2.34 e-9`.

g. Podemos obter os autovalores da matriz `A` com:

Freemat: Linhas de comando

```
eig(A)
ans =
      16.1168
      -1.1168
      -0.0000
```

h. Podemos calcular o determinante da matriz `A` através de:

Freemat: Linhas de comando
```
det(A)
ans = 6.6613e-016 = 0.0000000000066613
```

i. Matrizes aceitam números complexos nas entradas nas suas operações e funções.

Freemat: Linhas de comando
```
A = [1,2;3,4] + i*[5,6;7,8]
A =
    1.0000 +  5.0000i    2.0000 +  6.0000i
    3.0000 +  7.0000i    4.0000 +  8.0000i
```

j. Para matrizes, não escreva `2 + 6 * i` mas digite `2+6*i`, *evitando* os espaços vazios.

k. Se as letras `i` ou `j`, normalmente utilizadas para a unidade imaginária, já foram usadas como índices ou variáveis em algumas expressões, podemos criar uma nova unidade imaginária, como `I=sqrt(-1)`.

l. Matriz de Hilbert `3x3` gerada pela função `hilb()` (página 13 deste tutorial). Podemos chamar esta função com:

Freemat: Linhas de comando
```
hilb(3,3)
ans =
      1.0000    0.5000    0.3333
      0.5000    0.3333    0.2500
      0.3333    0.2500    0.2000
```

m. Matriz `us` `2x4` criada por declarações no Freemat em uma única linha:

Freemat: Linhas de comando
```
for i=1:2, for j=1:4, us(i,j)=i+j, end, end
us =
     2 3 4 5
     3 4 5 6
```

n. O Freemat possui um editor interno de textos que pode ser acionado com as teclas de atalho `Ctrl+E` ou mais simplesmente, digitando na linha de comando a palavra:

Freemat: Linhas de comando
```
editor
```

o. Matriz criada em um arquivo e gerada em um editor de texto puro. Abra o editor interno do Freemat e digite o código na sua área à direita:

Arquivo: mat9.m
```
mat9 = [1 2 3
        4 5 6
        7 8 9]
```

p. Salve o arquivo com o nome `mat9.m` e feche o editor interno do Freemat.

q. Você pode carregar a matriz `mat9.m` para a memória da máquina se digitar

Freemat: Linhas de comando

```
mat9
```

r. Para listar uma matriz com muitas entradas, é melhor usar um arquivo de texto criado no editor local, onde corrigimos erros facilmente. O arquivo deve ser um arranjo retangular (similar ao da matriz mat9) com as entradas numéricas da matriz.

s. Existem algumas funções embutidas: `eye`, `rand`, `ones` e `zeros`, facilitam a criação de outras matrizes.

t. Matrizes ou vetores somente aceitam índices que sejam *números naturais*.

u. Matrizes também aceitam mais do que dois índices, como um índice triplo `(i,j,k)`.

v. Podemos criar matrizes de ordem `mxnxp`, como por exemplo:

Freemat: Linhas de comando

```
for i=1:2, for j=1:3, for k=1:3, andr(i,j,k)=i+j+k, end, end, end
andr =
      (:,:,1) =
                3 4 5
                4 5 6
      (:,:,2) =
                4 5 6
                5 6 7
      (:,:,3) =
                5 6 7
                6 7 8
```

w. Podemos alterar o valor da posição `(1,2,1)` da matriz `andr` para que valha `pi`, com o código

Freemat: Linhas de comando

```
andr(1,2,1)=pi
andr =
       (:,:,1) =
                3.0000    3.1416    5.0000
                4.0000    5.0000    6.0000
       (:,:,2) =
                4.0000    5.0000    6.0000
                5.0000    6.0000    7.0000
       (:,:,3) =
                5.0000    6.0000    7.0000
                6.0000    7.0000    8.0000
```

x. Uma entrada de uma matriz ou vetor pode ser indicada com índices dentro de parênteses. O comando `A(2,3)` indica a entrada da linha `2` e coluna `3` da matriz `A` e `x(3)` indica a terceira coordenada do vetor `x`.

# 9 Operações com Matrizes e Arranjos

a. As operações usuais com matrizes são as seguintes: + (soma), - (subtração), * (multiplicação), ^ (potência), \ (divisão pela esquerda), / (divisão pela direita) e ' (transposta conjugada).

b. Na Linguagem de programação C, a divisão entre dois números inteiros, como por exemplo, 7/4=1. No Freemat, a operação funciona normalmente, isto é,

Freemat: Linhas de comando

```
7/4
ans =
      1.7500
```

c. Divisão de argumentos complexos:

Freemat: Linhas de comando

```
a = 3 + 4*i; b = 5 + 8*i;
c = a/b
c =
    0.5281 - 0.0449i
```

d. Número complexo dividido por um tipo duplo e o resultado promovido a dcomplex.

Freemat: Linhas de comando

```
b = a/2.0
b =
    1.5000 + 2.0000i
```

e. Divisão pela direita (sem ponto) de uma matriz a por uma matriz b significa o produto da matriz a pela inversa da matriz b, isto é,

Freemat: Linhas de comando

```
a = [1,2;3,4]; b =  [2,3;6,7];
c = a/b
c =
    1.2500   -0.2500
    0.7500    0.2500
```

f. Divisão pela direita (com ponto) de uma matriz a por uma matriz b significa a divisão da matriz a pela matriz b, elemento-a-elemento:

Freemat: Linhas de comando

```
a = [1,2;3,4]; b =  [2,3;6,7];
c = a./b
c =
    0.5000  0.6667
    0.5000  0.5714
```

g. Divisão pela direita elemento-a-elemento de matriz por escalar:

Freemat: Linhas de comando

```
c = a/3
c =
    0.3333  0.6667
    1.0000  1.3333
```

h. Divisão pela direita elemento-a-elemento de escalar por matriz:

Freemat: Linhas de comando

```
c = 3/a
c =
    3.0000  1.5000
    1.0000  0.7500
```

i. Operador produto elemento-a-elemento de duas *matrizes*:

Freemat: Linhas de comando

```
a = [1,2;3,4]; b = [2,3;6,7];
c = a .* b
c =
     2  6
    18 28
```

j. Operador produto elemento-a-elemento de escalar por matriz:

Freemat: Linhas de comando

```
c = 3*a
c =
    3  6
    9 12
```

k. Operador produto elemento-a-elemento de matriz por escalar:

Freemat: Linhas de comando

```
c = a*3
c =
    3  6
    9 12
```

l. Cálculo da transposta conjugada de uma matriz real

Freemat: Linhas de comando

```
A = [1,2; 0,4; 1,-1];
A'
ans =
      1  0  1
      2  4 -1
```

m. Cálculo da transposta conjugada de uma matriz complexa

Freemat: Linhas de comando

```
A = [1+i,2-i]; A'
ans =
      1.0000 - 1.0000 i
      2.0000 + 1.0000 i
```

n. Cálculo da transposta (sem a conjugação) de uma matriz complexa

Freemat: Linhas de comando

```
A = [1+i,2-i]; A.'
ans =
      1.0000 + 1.0000 i
      2.0000 - 1.0000 i
```

o. Uso do operador de divisão para resolver o sistema `AY=B`.

Freemat: Linhas de comando

```
A = [1,1;0,1];  B = [3;2];
Y = A \ B
Y =
    1
    2
```

p. Estas operações matriciais também funcionam para matrizes escalares `1x1`.

q. Se a operação matricial *não é compatível*, o Freemat envia uma mensagem de erro para o console, exceto no caso de operações com matrizes escalares (soma, subtração, divisão e multiplicação) pois cada entrada da matriz opera como se fosse um escalar.

r. A *divisão de matrizes* deve ser tratada de modo especial. Se `A` é uma matriz quadrada inversível e `b` é uma matriz-coluna ou matriz-linha compatível, então, `x=A\b` é a solução de `A*x=b` e `x=b/A` é a solução de `x*A=b`.

s. Na divisão pela esquerda, se a matriz `A` é quadrada, ela é fatorada pelo Método de Eliminação de Gauss e estes fatores são usados para resolver `A*x=b`.

t. Se a matriz `A` não é quadrada, ela é fatorada pelo Método de ortogonalização de Householder com *pivot* em colunas e os fatores são usados para resolver o sistema *subdeterminado* e *sobredeterminado* no sentido dos mínimos quadrados.

u. A divisão pela direita é obtida com a divisão pela esquerda por `b/A=(A'\ b')'`.

v. As operações: `*`, `^`, `\` e `/`, também podem operar elemento-a-elemento se colocarmos um ponto `.` antes delas, ficando da forma `.*`, `.^`, `.\` e `./`. O operador `.` ponto é útil na construção de gráficos.

w. O mesmo código `ponto-a-ponto`, agora com exponencial:

Freemat: Linhas de comando

```
[1,2,3,4] .^ 2
              ans = [1,4,9,16]
```

# 10 Construindo gráficos de funções

a. Primeiramente, vamos definir uma função `tal=tal(x)`

Freemat: Linhas de comando
```
tal = @(x) x.*sin(x)
```

b. Agora plotamos esta função *anonymous* `tal(x)=x.sin(x)` sobre o intervalo `[1,16]` em cor *vermelha*, (r=red) com o comando `plot`

Freemat: Linhas de comando
```
x=1:0.01:16;
plot(tal(x),'r-')
```

c. Deverá aparecer uma janela gráfica com o gráfico da função `tal(x)=x*sin(x)`:

d. Podemos anexar uma `grade` e o título `Função Tal` ao gráfico:

Freemat: Linhas de comando
```
x=1:0.01:16;
plot(tal(x),'r-'), grid, title('Função Tal')
```

e. *Nota*: Se conhecemos o gráfico de uma função `f=f(x)`, podemos obter uma outra função `g(x)=A.f(B(x-C))+D`, sendo que as variáveis `A`, `B`, `C` e `D` representam:

| | | | |
|---|---|---|---|
| A | Ampliação/Redução vertical | B | Ampliação/Redução horizontal |
| C | Deslocamento horizontal | D | Deslocamento vertical |

f. Exercício: Comparar os gráficos de `f1(x)=sin(x)` e `f2(x)=5.sin(3(x-pi/2))+2`.

# 11 Estatística com o Freemat

a. O Freemat possui algumas funções estatísticas pré-definidas: média, máximo, mínimo, soma, produto, variância, desvio padrão e soma acumulada (cumsum).

b. As funções estatísticas que não existem no Freemat, podem ser criadas através de arquivos `.m`.

c. *Problema*: Nos meses de um certo ano, um setor registrou certos fatos de acordo com a tabela numérica de ocorrências:

| J | F | M | A | M | J | J | A | S | O | N | D |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 3 | 5 | 5 | 10 | 8 | 9 | 6 | 3 | 4 | 8 | 7 |

d. Vamos inserir os dados do problema em uma matriz:

Freemat: Linhas de comando
```
x=[4,3,5,5,10,8,9,6,3,4,8,7]
          x = 4  3  5  5 10  8  9  6  3  4  8  7
```

e. Agora, vamos realizar alguns cálculos com os dados do problema:

Freemat: Linhas de comando

```
mean(x)
           ans = 6
sum(x)
           ans = 72
std(x)
           ans = 2.3741
var(x)
           ans = 5.6364
cumsum(x)
           ans = 4  7 12 17 27 35 44 50 53 57 65 72
```

f. A *soma acumulada* de um arranjo numérico `n`-dimensional `x` na dimensão `d` é gerada com a função `y=cumsum(x,d)` e a saída `y` é do mesmo tipo que o arranjo `x`. Neste caso, tipos inteiros são promovidos a tipos `int32`. Se a dimensão `d` não está indicada, a soma é aplicada apenas à primeira dimensão não-singular.

g. *Exercício*: Testando um sistema de freio, uma indústria constatou que 21 motoristas, com velocidade de 120 Km por hora, conseguiram parar dentro das seguintes distâncias de frenagem (em metros) 58, 70, 80, 46, 61, 65, 75, 55, 67, 56, 70, 72, 75, 61, 66, 58, 68, 70. Obter a média, soma, mínimo, máximo, desvio padrão e variância.

h. O Freemat também opera com algumas estatísticas relacionadas a pares ordenados.

i. Primeiro, vamos inserir os pares `(x,y)` sendo que `x` e `y` são vetores-linha, com:

Freemat: Linhas de comando

```
x = [2, 4, 7, 9,12,13,14];
y = [7,12,17,23,30,35,40];
```

j. Obtemos as médias aritméticas de `x` e de `y`, com:

Freemat: Linhas de comando

```
xm = mean(x)                % Média aritmética dos x
     xm = 8.7143
ym = mean(y)                % Média aritmética dos y
     ym = 23.4286
n  = numel(x)               % Número de elementos de x
     ans = 7
Sx = sum(x)                 % Soma dos elementoss de x
     ans = 61
Sy = sum(y)                 % Soma dos elementos de y
     ans = 164
Sxx = sum(x.*x) - n*xm*xm   % Soma ajustada de produtos xx
      ans = 1.2743e+002
Sxy = sum(x.*y) - n*xm*ym   % Soma ajustada de produtos xy
      ans = 3.3386e+002
```

Freemat: Linhas de comando

```
Syy = sum(y.*y) - n*ym*ym  % Soma ajustada de produtos yy
      ans = 8.9371e+002
C = Sxy/n                  % A covariância do conjunto
    ans = 47.6939
r = Sxy/sqrt(Sxx*Syy)      % Ceficiente de correlação
    r = 0.9893
a = Sxy/Sxx                % coeficiente angular da reta de ajuste
    ans = 2.6200
b = ym-a*xm                % coeficiente linear da reta de ajuste
    ans = 0.5975
```

k. Agora vamos plotar apenas os pares ordenados `(x,y)`, com:

Freemat: Linhas de comando

```
plot(x,y,'ro')
```

l. Construímos os coeficientes da reta:

Freemat: Linhas de comando

```
p = [a, b]
    p = 2.6200    0.5975
```

m. Agora criamos as ordenadas da função `f` que representa a reta de melhor ajuste, correspondentes aos valores de `x` dados inicialmente:

Freemat: Linhas de comando

```
f = polyval(p,x)
ans = 5.8374  11.0774  18.9372  24.1771  32.0370  34.6570  37.276
```

n. Plotamos os pontos `(x,y)` e a reta de melhor ajuste, com:

Freemat: Linhas de comando

```
plot(x,f,'r-', x,y,'bs','linewidth',2)
```

`r-` indica os pontos da reta `(x,f)` plotados em `red` com ligação dos pontos por segmentos, `bs` indica os pontos `(x,y)` plotados em `blue` com `square` e `'linewidth',2` indica as linhas traçadas com espessura `2` (O normal é `1`).

o. Os coeficientes da reta de melhor ajuste, podem ser obtidos no Freemat com:

Freemat: Linhas de comando

```
p = polyfit(x,y,1)
    p =  2.6200    0.5975
```

p. As ordenadas da reta `f` de melhor ajuste, correspondentes aos valores de `x` dados inicialmente são obtidos por

Freemat: Linhas de comando

```
f = polyval(p,x);
```

q. A plotagem dos pontos `(x,y)` e da reta de melhor ajuste, é obtida com:

Freemat: Linhas de comando

```
plot(x,f,'r-', x,y,'bs','linewidth',2)
```

# 12 Usando condições lógicas

a. Criando algumas variáveis para testes lógicos.

Freemat: Linhas de comando

```
a = 3.5; b = 8.0; c = 5.7;
```

b. Testando condições lógicas sobre as variáveis.

Freemat: Linhas de comando

```
(a < b)
                        ans = 1
(a == b)
                        ans = 0
(b >= a) & (b >= c)
                        ans = 1
(c >= b) & (c >= a)
                        ans = 0
(c >= b) | (c >= a)
                        ans = 1
```

# 13 Declarações, Expressões e Variáveis

a. Freemat é um programa que opera com `expressões` e elas são interpretadas e calculadas. Uma declaração é usualmente escrita na forma

Freemat: Linhas de comando

```
variavel = expressao
```

b. Outra forma de definir uma expressão é simplesmente escrever:

Freemat: Linhas de comando

```
expressao
```

c. Em geral, expressões são compostas de operadores, funções e nomes de variáveis.

d. O cálculo de uma expressão gera uma matriz, que é mostrada na tela e associada a uma variável para uso posterior.

e. Se o nome da variável e o sinal de igualdade = são omitidos, será criada automaticamente uma variável `ans` (answer=resposta) para o correspondente resultado.

f. Normalmente, uma declaração é encerrada por um `[ENTER]`.

g. Uma declaração pode ser continuada na próxima linha com o acréscimo de três pontos ... ou mais pontos seguidos por um `[ENTER]`.

h. Várias declarações podem ser postas em uma única linha separada por vírgulas ou ponto-e-vírgulas.

i. Se o último caracter de uma declaração é um ; ponto-e-vírgula, a saída não mostra o resultado, mas a associação é realizada. Este fato é essencial para não mostrar saídas que você não deseja ou resultados intermediários.

j. Freemat é sensível ao contexto com nomes de comandos, funções e variáveis. Por exemplo, a palavra `Comando` é diferente da palavra `comando`.

k. O comando `who` (ou `whos`) lista as variáveis disponíveis no workspace.

l. Um processo ou cálculo pode ser interrompido em muitas máquinas sem fechar o Freemat, usando as teclas de atalho `CTRL+C` (ou `CTRL+BREAK` em um PC).

# 14 Salvando uma sessão

Ao encerrar o Freemat, todas as variáveis são perdidas, mas usando o comando `save` antes de sair, você salvará todas as variáveis em um arquivo. Ao reentrar no Freemat, o comando `load` irá reestabelecer o workspace ao estado anterior.

# 15 Funções para construir Matrizes

a. Existem algumas funções embutidas que servem para construir matrizes.

b. O comando `eye` gera uma matriz identidade. Por exemplo:

Freemat: Linhas de comando

```
Id = eye(3)
>>> Id = 1 0 0
         0 1 0
         0 0 1
```

c. O comando `zeros` gera uma matriz de zeros. Por exemplo, `zeros(m,n)` produz uma matriz `mxn` de zeros e `zeros(n)` produz uma matriz `nxn`.

Freemat: Linhas de comando

```
Z23 = zeros(2,3)
>>> 0 0 0
    0 0 0
Z22 = zeros(2,2)
>>> 0 0
    0 0
Z3 = zeros(3)
>>> 0 0 0
    0 0 0
    0 0 0
```

d. Se `A` é uma matriz, então `zeros(size(A))` produz uma matriz de zeros com a mesma dimensão de `A`.

Freemat: Linhas de comando

```
A=[1,2,3; 3,4,5; 7,8,9]
>>> A = 1 2 3
        4 5 6
        7 8 9
Z = zeros(size(A))
>>> Z = 0 0 0
        0 0 0
        0 0 0
```

e. O comando `ones` gera uma matriz em que as entradas são todas iguais a `1`.

Freemat: Linhas de comando

```
O23 = ones(2,3)
>>> 1 1 1
    1 1 1
O22 = ones(2,2)
>>> 1 1
    1 1
O3 = ones(3)
>>> 1 1 1
    1 1 1
    1 1 1
```

f. Se `A` é uma matriz quadrada, `diag(A)` é um vetor com a diagonal principal de `A`.

Freemat: Linhas de comando

```
D = diag(A)
>>> D = 1
        5
        9
```

g. Se `x` é um vetor, `diag(x)` é a matriz diagonal com `x` na diagonal principal.

Freemat: Linhas de comando

```
x=[1,5,9]
>>> x = 1 5 9
DD=diag(x)
>>> DD = 1 0 0
         0 5 0
         0 0 9
```

h. O que é `diag(diag(A))`?

i. O comando `rand` gera uma matriz aleatória. Por exemplo:

Freemat: Linhas de comando

```
R23 = rand(2,3)
>>> 0.3759    0.9134    0.7604
    0.0183    0.3580    0.8077
R22 = rand(2,2)
>>> 0.0990    0.3478
    0.4972    0.0276
R3 = rand(3)
>>> 0.5313    0.4921    0.8670
    0.9958    0.7597    0.2714
    0.2079    0.3365    0.2174
```

j. Matrizes podem ser construídas com blocos (como uma casa). Por exemplo, se `A` é uma matriz `3x3`, então

Freemat: Linhas de comando

```
B = [A, zeros(3,2); zeros(2,3), eye(2)]
>>> B = 1 2 3 0 0
        4 5 6 0 0
        7 8 9 0 0
        0 0 0 1 0
        0 0 0 0 1
```

é uma matriz `5x5`, sendo que `B(1,1)=A`, `B(1,2)=zeros(3,2)`, `B(2,1)=zeros(2,3)` e `B(2,2)=eye(2)`.

# 16 Laços For, while, if e Operações relacionais

a. Em geral, laços controlam o fluxo de operações através de declarações, da mesma forma que em outras linguagens de programação. Uma matriz vazia é escrita como `x=[]`.

b. **Laço For**: Podemos gerar um vetor `3`-dimensional com a declaração (em uma linha)

Freemat: Linhas de comando

```
x = []; for i = 1:3, x=[x,i^2], end
>>> x = 1
>>> x = 1 4
>>> x = 1 4 9
```

c. O mesmo vetor `3`-dimensional pode ser gerado, em várias linhas, com:

Freemat: Linhas de comando

```
x = [];
for i = 1:3
    x = [x,i^2]
end
```

d. O mesmo vetor obtido antes, agora na ordem invertida, é gerado pela declaração:

Freemat: Linhas de comando
```
x = []; for i = 3:-1:1, x=[x,i^2], end
```

e. Podemos construir uma matriz de Hilbert 3x2 com as declarações:

Freemat: Linhas de comando
```
for i = 1:3
    for j = 1:2
        H(i,j) = 1/(i+j-1);
    end
end
H   % Solicita que a matriz seja posta na tela
>>> ans = 1.0000    0.5000
          0.5000    0.3333
          0.3333    0.2500
```

f. O ponto e vírgula informa ao Freemat para não mostrar os resultados intermediários.

g. A declaração `for` permite que `qualquer` matriz seja usada no lugar de `1:n`. A variável assume exatamente, de modo consecutivo o valor de cada coluna da matriz.

h. Para calcular a soma de todas as entradas da matriz `A`, digitamos:

Freemat: Linhas de comando
```
sum(sum(A))
>>> 45
```

i. **Laço While**: A forma geral de um laço `while` é

Freemat: Linhas de comando
```
while relacao
    declaracoes
end
```

Neste caso, a linha `declaracoes` é executada repetidamente até que `relacao` seja verdadeira.

j. Por exemplo, para um número dado `1000`, o laço abaixo irá calcular e mostrar o menor inteiro não-negativo `n` tal que $2^n \geq 1000$:

Freemat: Linhas de comando
```
n = 0;
while 2^n < 1000
    n = n + 1;
end
n
>>> ans = 10
```

k. **Condicional If**: Se `relacao` é verdadeira, a linha `declaracoes` é executada. A forma mais simples de uma declaração `if` é

Freemat: Linhas de comando

```
if relacao
    declaracoes
end
```

l. Vários ramos também são possíveis, como está ilustrado abaixo:

Freemat: Linhas de comando

```
if n < 0
    paridade = 0;
elseif rem(n,2) == 0      % rem resto da divisão de n por 2.
    paridade = 2;
else
    paridade = 1;
end
```

m. Em declarações com apenas dois ramos, a parte `elseif` deve ser omitida.

n. **Operadores relacionais**: Os operadores relacionais do Freemat são: < (menor que), > (maior que), <= (menor ou igual), >= (maior ou igual), == (igual) e ~= (não igual).

o. *Nota*: O sinal simples de igualdade = serve para associar um valor, mas o sinal duplo de igualdade == é usado como um operador relacional.

p. Relações lógicas podem ser conectadas ou quantificadas com os seguintes operadores lógicos: `&` (e), `|` (ou) e `~` (não).

q. Quando usadas em escalares, uma relação é realmente o escalar `1` ou `0` se, respectivamente, a relação é verdadeira ou falsa. Alguns exemplos de saídas verdadeiras (1) ou falsas (0):

Freemat: Linhas de comando

```
3 < 5
>>> ans = 1
3 > 5
>>> ans = 0
3 == 5
>>> ans = 0
3 ~= 3
>>> ans = 0
```

r. Quando usadas em matrizes de mesma dimensão, uma relação é uma matriz de `zerosd` e `uns` dando o valor da relação entre as entradas correspondentes.

Freemat: Linhas de comando

```
a = rand(3)
a  =  0.6565    0.1273    0.0840
      0.3236    0.9954    0.4618
      0.5196    0.5407    0.9132
```

Freemat: Linhas de comando

```
b = eye(3)
b  =  1 0 0
      0 1 0
      0 0 1
```

Freemat: Linhas de comando

```
a == b
ans = 0 0 0
      0 0 0
      0 0 0
```

s. Os laços `while` e `if` interpretam como `verdadeira` se cada entrada da relação matricial é `não-nula`. Para executar uma `declaracao` em que `A` e `B` são iguais, digite:

Freemat: Linhas de comando

```
if  A == B
    declaracao
end
```

t. Para executar uma `declaracao` quando `A` e `B` não são iguais. Digite com dois `any` pois este operador é vetorial.

Freemat: Linhas de comando

```
if  any(any(A ~= B))
    declaracao
end
```

u. As funções `any` e `all` reduzem as relações matriciais entre vetores ou escalares.

v. Outra forma mais simples para o código anterior:

Freemat: Linhas de comando

```
if  A == B  else
    declaracao
end
```

w. O que parece óbvio

Freemat: Linhas de comando

```
if  A ~= B, declaracao, end
```

não fornece o desejado pois `declaracao` deve ser executada apenas se `cada` uma das entradas de `A` e `B` são diferentes.

# 17 Funções escalares, vetoriais e matriciais

a. Existem três tipos comuns de funções: escalares, vetoriais e matriciais.

b. **Funções escalares**: Funções que operam sobre escalares, mas operam elemento-a-elemento se aplicadas a matrizes. As mais comuns são: `abs`, `acos`, `asin`, `atan`, `cos`, `ceil`, `exp`, `floor`, `log` (natural), `rem` (resto), `round`, `sign`, `sin`, `sqrt` e `tan`.

c. **Funções Vetoriais**: Funções que operam essencialmente sobre vetores (linha ou coluna), mas atuam sobre uma matriz `mxn` ($m \geq 2$) coluna-a-coluna para produzir um vetor-linha contendo os resultados da sua aplicação a cada coluna. A ação linha-a-linha pode ser obtida pelo uso da transposta, como por exemplo `mean(A')'`. As funções vetoriais mais comuns: `all`, `any`, `max`, `mean`, `min`, `prod`, `sort`, `std` e `sum`.

d. A entrada de maior valor da matriz `A` é dada por `max(max(A))` ou por `max(A)`.

e. **Funções Matriciais**: Coisas interessantes acontecem no Freemat com o uso de suas funções matriciais. As funções matriciais mais úteis são:

(a) `chol` Fatoração de Cholesky
(b) `cond` Número condicional (norma2)
(c) `det` Determinante
(d) `eig` Autovalores e autovetores
(e) `expm` Matriz exponencial
(f) `inv` Inversa
(g) `lu` Fatoração LU
(h) `hess` Forma de Hessenberg
(i) `norm` normas: 1, 2, F, infinito
(j) `poly` Polinômio característico
(k) `qr` Fatoração QR
(l) `rank` Posto
(m) `rref` Forma escada linhas-reduzida
(n) `schur` Decomposição de Schur
(o) `size` Dimensão
(p) `sqrtm` Matriz raiz quadrada
(q) `svd` Decomposição valor singular

f. Tais funções podem ter argumentos simples ou múltiplos na saída. Por exemplo, para obter um vetor coluna com os autovalores da matriz `A`, digite:

Freemat: Linhas de comando

```
A=[1,2,3; ... [ENTER]
4,5,6; ... [ENTER]
7,8,1]; [ENTER]
y=eig(A)                    % Podemos usar apenas eig(A)
y = 12.4542
    -0.3798
    -5.0744
```

g. Também podemos usar o código `eig(A)` para obter o mesmo resultado mostrado antes.

h. Para gerar uma matriz `U` com os autovetores da matriz `A` nas colunas e uma matriz diagonal `D` com os autovalores da matriz `A` na diagonal principal.

i. Usamos o código para obter a matriz de autovetores e a matriz diagonal com os autovalores.

Freemat: Linhas de comando

```
[U,D] = eig(A)
>>> U = -0.2937   -0.7397   -0.2972
        -0.6901    0.6650   -0.3987
        -0.6615   -0.1031    0.8676
    D = 12.4542         0         0
              0   -0.3798         0
              0         0   -5.0744
```

# 18 Editando e Rechamando uma linha de comando

a. A linha de comando pode ser editada facilmente. O cursor pode ser posto com as setas para a esquerda e para a direita e as teclas `Backspace` (ou `Delete`) usadas para apagar o caracter à esquerda do cursor. Outros feitos de edição também estão disponíveis. No PC, tente usar as teclas `Home`, `End` e `Delete`.

b. Um feito conveniente é usar as setas `up/down` para rolar a pilha dos comandos executados anteriormente. Ao obter o comando, podemos chamar o mesmo, editar e executar a linha de comando revisada.

c. Para pequenas rotinas, é sempre mais conveniente *rolar a pilha* do que usar um arquivo `.m` que exige alguma ação entre o programa e o editor.

d. Para comparar gráficos das funções `y=sin(mx)` e `y=sin(nx)` sobre o intervalo `[0,2pi]` para vários parâmetros `m` e `n`, podemos digitar a seguinte linha de comando:

Freemat: Linhas de comando

```
m=2; n=3; x=0:.01:2*pi;
y=sin(m*x);
z=cos(n*x);
plot(x,y,x,z)
```

# 19 Submatrizes e a notação dois pontos

a. Vetores e submatrizes são muito usados para manipular dados complexos.

b. A notação `:` (também usada para gerar vetores e referências a submatrizes) indexa vetores de inteiros como chaves realizando bem a manipulação de tais objetos.

c. O uso criativo deste tipo de indexação serve para vetorizar operações, permite reduzir o uso de `laços` (operação lenta), simplificar o código e ficar mais fácil de ler.

d. *Nota*: É bom se esforçar para ficar familiarizado com este tipo de indexação!

e. A expressão `1:5` (já usada antes) é na verdade um vetor-linha. Realmente

Freemat: Linhas de comando
```
1:5
>>> 1 2 3 4 5
```

f. Os números não precisam ser inteiros nem o incremento precisa ser 1.

Freemat: Linhas de comando
```
0.2 : 0.2 : 1.2
>>> 0.2 0.4 0.6 0.8 1.0 1.2
5 : -1 : 1
>>> 5 4 3 2 1
```

g. Por exemplo, com as seguintes declarações, iremos gerar uma tabela de senos:

Freemat: Linhas de comando
```
x = [0.0 : 0.1 : 2.0]';
y = sin(x);
[x y]'
```

h. *Nota*: A função `sin` opera ponto-a-ponto, usando o vetor `x` para gerar um vetor `y`.

i. A notação `:` pode ser usada para acessar sub-matrizes de uma matriz.

j. Geramos um vetor-coluna com as `4` primeiras entradas da coluna `3` da matriz `A`, com:

Freemat: Linhas de comando
```
A=[1,2,3; 4,5,6; 7,8,9; 10,11,12];
A(1:4,3)
>>>  3
     6
     9
    12
```

k. Um sinal `:` sozinho indica uma linha inteira ou coluna inteira. O código

Freemat: Linhas de comando
```
A(:,3)
```

gera a coluna 3 da matriz $A$, isto é, o mesmo resultado que o item anterior.

l. Para gerar as linhas `1,2,3,4` da matriz `A`, digite o código:

Freemat: Linhas de comando
```
A(1:4,:)
>>> A =  1  2  3
          4  5  6
          7  8  9
         10 11 12
```

m. Em geral, vetores de inteiros ficam como índices. Geramos as colunas `2` e `4` de `A`, com:

Freemat: Linhas de comando

```
A(:,[2 3])
>>>   2  3
      5  6
      8  9
     11 12
```

n. Tais índices podem ser usados de ambos os lados de uma declaração de associação.

o. Para trocar trocar as colunas `2,4,5` da matriz `A` pelas colunas `1,2,3` da matriz `B`, digite:

Freemat: Linhas de comando

```
A=[1,2,3,4,5; 6,7,8,9,10; 11,12,13,14,15; 16,17,18,19,20; 1,2,3,4,5]'
B=[1,1,1,1,1; 2,2,2,2,2; 3,3,3,3,3; 4,4,4,4,4; 5,5,5,5,5]'
A(:,[2 4 5]) = B(:,1:3)
A =  1  1 11  2  3
     2  1 12  2  3
     3  1 13  2  3
     4  1 14  2  3
     5  1 15  2  3
```

Neste caso, `toda` a matriz `A` que foi alterada é mostrada e associada com a letra `A`.

p. As colunas 2 e 4 da matriz `A` são multiplicadas pela direita pela matriz `[1,2;3,4]`:

Freemat: Linhas de comando

```
A=[1,2,3,4,5; 6,7,8,9,10; 11,12,13,14,15; 16,17,18,19,20; 1,2,3,4,5]
A(:,[2,4]) = A(:,[2,4])*[1 2;3 4]
>>> A =  1  14   3  20   5
         6  34   8  50  10
        11  54  13  80  15
        16  74  18 110  20
         1  14   3  20   5
```

De novo, toda a matriz alterada é mostrada e associada à própria letra `A`.

q. Se `x=[1,2,3,4,5,6,7]`, o que significa a declaração `x=x(7:-1:1)`?

# 20 Arquivos no Freemat

a. Os tipos de arquivos funcionais do Freemat, são arquivos: com a extensão `.m`, de script e de funções.

b. **Arquivo com a extensão m**: Permite executar uma sequência de declarações armazenadas em arquivos com a extensão `.m`. No Freemat usamos muito os arquivos `.m`, que são de dois tipos: *script* e *funções*. Tais arquivos são criados no editor local de textos.

c. **Arquivo de Script**: Contém sequências de declarações normais. Se o arquivo possui o nome `rotate.m`, então o comando `rotate` irá chamar as declarações no arquivo a ser executado. As variáveis no arquivo de script são *globais* e mudam os valores das variáveis homônimas no ambiente da sessão atual do Freemat.

d. Um arquivo de Script serve para inserir dados em uma matriz, para que os erros sejam corrigidos facilmente. Por exemplo, se o arquivo `data.m` possui a entrada:

Freemat: Linhas de comando
```
A = [1 2 3 4
     5 6 7 8];
```

então a declaração `data` irá associar esta matriz a `data.m` para realizar a operação desejada. É mais fácil usar a função `load` para carregar este arquivo.

e. Um arquivo `.m` pode fazer referência a outros arquivos `.m`, inclusive se referir ao próprio arquivo de modo recursivo.

f. **Arquivo de Função**: Permite expandir o Freemat. Podemos criar novas funções para os nossos problemas que terão o mesmo status que as outras funções já embutidas no Freemat. Variáveis em um arquivo de função são *locais* por default, mas uma variável pode ser declarada como *global* (ver `help global`).

g. Vamos ilustrar com um exemplo simples de um arquivo de função.

Freemat: Linhas de comando
```
function  a = randint(m,n)
% RANDINT Matriz de inteiros gerada randomicamente.
%         randint(m,n) retorna matriz mxn com entradas em [0,9].
a = floor(10*rand(m,n));
```

h. Uma versão mais geral desta função está abaixo:

Freemat: Linhas de comando
```
function  a = randint(m,n,a,b)
% RANDINT Matriz de inteiros gerada randomicamente.
%         randint(m,n) retorna matriz mxn com entradas em [0,9].
%         rand(m,n,a,b) retorna entradas inteiras em [a,b].
if nargin < 3, a = 0; b = 9; end
a = floor((b-a+1)*rand(m,n)) + a;
```

i. Insira o conteúdo acima em um arquivo com o nome `randint.m` (o mesmo nome da função). A primeira linha declara o nome da função, argumentos de entrada e de saída. Sem esta linha, o arquivo deve ser um arquivo de script.

j. A declaração `z=randint(4,5)`, faz com que os números 4 e 5 sejam passados às variáveis `m` e `n` no arquivo da função e o resultado de saída passado à variável `z`.

k. Como são *locais* as variáveis em um arquivo de função, seus nomes independem dos nomes do ambiente atual do Freemat.

l. O uso do comando `nargin` (*número de argumentos de entrada*) permite configurar o valor padrão omitido de uma variável de entrada, como `a` e `b` no exemplo acima.

m. Uma função também pode ter vários argumentos de saída. Por exemplo,

Freemat: Linhas de comando

```
function [mean, stdev] = stat(x)
% STAT   Média e desvio padrão
%        Para um vetor x, stat(x) retorna a m´dia de x;
%        [mean, stdev] = stat(x) retorna ambos, média e desvio padrão.
%        Para uma matriz x, stat(x) atua coluna-a-coluna.
[m  n] = size(x);
if m == 1
    m = n;  % trata o caso de um vetor-linha
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```

n. Se o conteúdo acima for posto no arquivo `stat.m`, o comando `[xm,xd]=stat(x)`, associará a média `xm` e o desvio padrão `xd` das entradas de `x`, respectivamente.

o. Associações simples também podem ser realizadas com uma função tendo vários argumentos de saída. Por exemplo, `xm=stat(x)` (não precisa colocar colchetes em volta de `xm`) irá associa a média de `x` a `xm`.

p. O símbolo % indica que o resto da linha é um comentário. As primeiras linhas comentadas contíguas, documentam o arquivo `.m`, ficando disponíveis no help on-line e serão mostradas se, por exemplo, digitarmos `help stat`. Tal documentação deve *sempre* ser incluída em um arquivo de função. Esta função ilustra alguns dos feitos que podem ser usados para produzir bons códigos.

q. Observe que `x.^2` é a matriz dos quadrados das entradas de `x`, que `sum` é uma função vetorial, que `sqrt` é uma função escalar e que a divisão em `sum(x)/m` é uma operação de divisão de uma matriz por um escalar, assim, todas as operações são vetorizadas e os laços devem ser evitados.

r. Se não podemos vetorizar alguns cálculos, podemos fazer com que os laços `for` sejam mais rápido pré-alocando vetores ou matrizes para que a saída seja guardada.

s. Ao incluir a segunda declaração abaixo com a função `zeros`, reservamos espaço para guardar `E` na memória. Sem isto, o Freemat deve irá redimensionar `E` com mais uma coluna em cada iteração, tornando mais lenta a execução.

Freemat: Linhas de comando

```
M = ones(6);
E = zeros(6,50);
for j = 1:50
    E(:,j) = eig(M^j);
end
```

t. Alguns feitos mais avançados estão ilustrados pelas funções seguintes. Como observamos antes, alguns dos argumentos de entrada de uma função, tal como `tol` neste exemplo, pode ser opcionais com o uso de `nargin` (*número de argumentos de entrada*). A variável `nargout` pode ser usada da mesma forma.

u. O fato que uma relação é um número (1 se verdadeiro; 0 se é falso) é usado e quando `while` ou `if` avalia uma relação, `nonzero` significa `true` e $0$ significa `false`.

v. Algumas destas funções são embutidas e outras são distribuídas em arquivos `.m`.

# 21 Strings de texto, Mensagens de erro, Entrada

a. Strings de texto são inseridas em Fremat envolvidas por aspas simples. Por exemplo,

Freemat: Linhas de comando
```
s = 'Este é um teste.'
```

associa o texto dado à variável `s`.

b. Strings de texto podem ser mostradas com a função `disp`. Por exemplo:

Freemat: Linhas de comando
```
disp('esta mensagem está sendo exibida')
```

c. Mensagens de erro são mostradas de melhor maneira com a função `error`.

Freemat: Linhas de comando
```
error('Desculpe, a matriz deve ser simétrica.')
```

pois quando postas em um arquivo `.m`, elas abortam a execução do arquivo `.m`.

d. Em um arquivo `.m` o usuário pode ser solicitado a inserir dados interativamente com a função `input`. Por exemplo, quando a declaração

Freemat: Linhas de comando
```
iter = input('Entre com o número de iterações:  ')
```

é encontrada, a mensagem de prompt é mostrada e a execução interrompe enquanto o usuário realiza a entrada de dados. Após pressionar a tecla `[ENTER]`, o dado é associado à variável `iter` e a execução continua, até ...

# 22 Gerenciando arquivos `.m`

a. Usando o Freemat, comumente desejamos criar ou editar um arquivo `.m` com o editor local e então retornar ao Freemat, mas queremos manter o programa ativo ao mesmo tempo que editamos o arquivo pois encerrando o Freemat, todas as variáveis serão perdidas na saída.

b. Isto pode ser feito facilmente com as teclas de atalho `Ctrl+E` que abre um editor interno, ou pelo simples digitar da palavra `editor` na linha de comando. Basta editar o seu arquivo, salvar e continuar a trabalhar com o programa.

Freemat: Linhas de comando

```
Colocar um gráfico aqui.
```

c. Podemos editar um arquivo e salvar com o nome `rotate.m` e voltar ao programa e usar este arquivo recentemente digitado.

d. No Windows, é preferível manter ativos, tanto o Freemat como o editor local, com um processo ativo e o outro suspenso. Se tais processos rodam em várias janelas, você deve manter o Freemat ativo em uma janela e o editor em outra janela.

e. Várias ferramentas para depuração estão disponíveis. Ver `help dbtype` ou a lista de funções na última sessão.

f. O comando `pwd` (present working directory) retorna o nome do atual diretório de trabalho e `cd` pode ser usado para mudar o diretório de trabalho.

g. Os comandos `dir` ou `ls` listam os conteúdos do diretório de trabalho enquanto que o comando `what` lista somente os arquivos `.m` no diretório.

h. Arquivos `.m` devem estar em uma pasta acessível ao Freemat. Arquivos `.m` na atual pasta de trabalho também são acessíveis.

i. A lista atual de pastas no Fremat pesquisa o caminho com o comando `path`. Este comando também pode ser usado para acrescentar ou deletar pastas ao caminho de pesquisa. Ver `help path`.

# 23 Formato de saída

a. Os no Freemat são realizados em dupla precisão, mas o formato da saída pode ser controlado por:

| | |
|---|---|
| `format short` | ponto fixado com 4 dígitos após a vírgula (o padrão) |
| `format long` | ponto fixado com 14 dígitos após a vírgula |
| `format short e` | notação científica com 4 dígitos após a vírgula |
| `format long e` | notação científica com 15 dígitos após a vírgula |
| `format rat` | aproximação por razão de inteiros pequenos |
| `format hex` | formato hexadecimal |
| `format bank` | dolares e centavos fixados |
| `format +` | +, -, espaço em branco |

b. Uma vez chamado um formato, este permanece tendo efeito até que seja alterado.

c. O comando `format compact` elimina linhas vazias permitindo que mais dados sejam postas na tela ou na página. O comando `format loose` faz voltar ao formato não-compacto. Estes comandos são independentes dos outros comandos de formato.

# 24 Cópia física

a. Uma cópia física pode ser obtida com o comando `diary`. O comando mostra sequencialmente na tela (exceto os gráficos) expressões que são escritas em um arquivo.

Freemat: Linhas de comando
```
diary NomeArquivo
```

b. Se o nome do arquivo não for informado ele será escrito para um arquivo padrão denominado `diary` até que seja digitado o comando `diary off`. O comando `diary on` escreve para o arquivo para encerrar, etc.

c. Quando terminar, você pode editar o arquivo e imprimir o mesmo em seu sistema.

# 25 Gráficos

Freemat gera gráficos de: curvas no plano e no espaço, superfícies gradeadas e facetadas. Os principais comandos para tais gráficos são: `plot, plot3, meshgrid, surf`.

## 25.1 Gráficos de curvas no plano

a. O comando `plot` cria gráficos lineares no plano XY. Se `x` e `y` são vetores de mesma dimensão, o comando `plot(x,y)` abre uma janela gráfica e desenha um gráfico `XY` com os elementos de `x` versus os elementos de `y`.

b. Um gráfico simples da função `sin()` sobre o intervalo `[-4,4]` com o código:

Freemat: Linhas de comando
```
x = -4 : 0.01 : 4;
y = sin(x);
plot(x,y)
```

c. Os gráficos podem ter títulos, grades, eixos com etiquetas e textos colocados dentro do gráfico com os seguintes comandos que aceitam uma string como argumento.

| | |
|---|---|
| `title` | Título do gráfico |
| `xlabel` | Etiqueta do eixo OX |
| `ylabel` | Etiqueta do eixo OY |
| `text` | Texto posicionado em coordenadas específicas |

d. Gráfico da função `sin()` sobre `[-4,4]` com uma grade:

Freemat: Linhas de comando
```
x = -4 : 0.01 : 4;
y = sin(x);
plot(x,y), grid
```

e. No código acima, `x` é uma partição do intervalo `[-4,4]` com passo `0.01` e `y` é um vetor associando ponto-a-ponto, os valores da função `sin` aos nós desta partição.

f. Podemos anexar um título no gráfico anterior, tomando o código:

Freemat: Linhas de comando
```
x = -4 : 0.01 : 4;
y = sin(x);
plot(x,y), grid, title('Função seno')
```

g. Colocamos a palavra `Texto` no ponto `(2,0.5)` do gráfico da função `sin()` com:

Freemat: Linhas de comando
```
plot(x,y), grid, title('Função seno'), text(2,0.5,'Texto')
```

h. Colocamos várias palavras no gráfico com o comando

Freemat: Linhas de comando
```
plot(x,y), grid, title('Função seno'), ...
text([0,1],[-0.1,0.2],{'Texto1','Texto2'}, 'fontsize',15)
```

em que `(0,-0.1)` é o ponto onde foi posto `Texto1` e `(1,0.2)` é o ponto onde foi posto `Texto2`. Anexamos uma opção para o tamanho da fonte ser igual a `15`.

i. Plotamos o gráfico da função `sin()` sobre o intervalo `[-4,4]`, anexando palavras aos eixos `x` e `y`, com o código:

Freemat: Linhas de comando
```
x = -4 : 0.01 : 4;
y = sin(x);
plot(x,y), grid, xlabel('Eixo x'), ylabel('Eixo y')
```

j. Podemos plotar vários gráficos, um para cada tempo da figura *atual* onde os gráficos dos comandos de plotagem seguintes são mostrados. Se `figure 1` é a figura atual, então o comando `figure(2)` (ou `figure`) abre uma segunda figura (se necessário) a constroi a figura atual. O comando `figure(1)` irá mostrar figure 1 e irá fazer isto com a figura atual. O comando `gcf` retorna o número atual de figuras.

k. Podemos desenhar o gráfico da curva de Gauss `y=exp(-x^2)` sobre `[-1.5,1.5]` anexando uma grade ao gráfico. Neste caso, usamos o sinal `.^` de potência para garantir que a operação seja ponto-a-ponto.

Freemat: Linhas de comando
```
x = -1.5 : .01 : 1.5;
y=exp(-x.^2);
plot(x,y), grid
```

l. A função `plot` pode gerar o gráfico de uma função definida em um arquivo. Criamos o arquivo `expnormal.m` com o código:

Função: expnormal()
```
function y = expnormal(x)
    y=exp(-x.^2);
```

e depois geramos o gráfico, digitando:

Freemat: Linhas de comando
```
x = linspace(-pi,pi);
plot(x,expnormal(x)), grid, title('Curva normal de Gauss')
```

m. Curvas parametrizadas podem ser plotadas no plano. Por exemplo:

Freemat: Linhas de comando
```
t = 0 : 0.001 : 2*pi; x = cos(3*t); y = sin(2*t);
plot(x,y), grid, title('Curva de Lissajous')
```

n. Normalmente, os eixos possuem uma auto-escala, o que pode ser sobreposto pelo comando `axis`. Alguns feitos do comando `axis` são:

| | |
|---|---|
| `axis([x_mi,x_ma,y_mi,y_ma])` | coloca escala de eixos nos limites indicados |
| `axis(axis)` | Limpa a escala para os gráficos seguintes |
| `axis auto` | Retorna à auto-escala |
| `v = axis` | Retorna o vetor `v` mostrando a escala atual |
| `axis square` | Mesma escala em ambos os eixos |
| `axis equal` | Mesma escala e marcas tic em ambos os eixos |
| `axis off` | Desliga escala dos eixos e marcas tic |
| `axis on` | Liga escala dos eixos e marcas tic |

o. O comando `axis` deve ser colocado *depois* do comando `plot`.

p. Podemos usar duas formas para vários gráficos em um simples gráfico:

Freemat: Linhas de comando
```
x=0:.01:2*pi; y1=sin(x); y2=sin(2*x); y3=sin(4*x);
plot(x,y1, x,y2, x,y3), grid
```

e formar uma matriz `Y` contendo os valores funcionais como colunas:

Freemat: Linhas de comando
```
x=0:.01:2*pi;
Y=[sin(x)', sin(2*x)', sin(4*x)'];
plot(x,Y)
```

q. Um outro modo é com `hold`. O comando `hold on` limpa a tela gráfica atual de modo que os gráficos seguinte são colocados sobre a tela gráfica. Mas, os eixos podem se tornar re-escalonados. Entrando `hold off` ocorre o *hold*.

r. Podemos sobrepor os tipos de linha padrão, tipos de pontos e cores. Por exemplo,

Freemat: Linhas de comando
```
x=0:.01:2*pi; y1=sin(x); y2=sin(2*x); y3=sin(4*x);
plot(x,y1,'--', x,y2,':', x,y3,'+')
```

gera uma linha tracejada e uma linha com pontos para os dois primeiros gráficos enquanto que para o terceiro, o símbolo + é colocado em cada nó.

s. Os tipos de linhas são: - (sólida), : (pontos), -. (traço e ponto) e -- (tracejada).

t. Os tipos de marcas são: . (ponto), o (círculo), x (vezes), + (mais), * (asterisco), s (quadrado), d (diamante), v (v), ^ (acento circunflexo), < (menor) e > (maior).

u. As Cores são: r (red), g (green), b (blue), k (black), c (cyan), m (magenta) e y (yellow).

v. Para plotar uma linha tracejada -- em vermelho r, digitamos:

Freemat: Linhas de comando
```
plot(x,y,'r--')
```

w. O comando subplot pode ser usado para particionar a tela em vários pequenos gráficos que podem ser postos em uma figura. Ver help subplot.

Freemat: Linhas de comando
```
t = linspace(-pi,pi);
subplot(2,2,1)
  plot(t,cos(t*1).*exp(-2*t));
subplot(2,2,2);
  plot(t,cos(t*2).*exp(-2*t));
subplot(2,2,3);
  plot(t,cos(t*3).*exp(-2*t));
subplot(2,2,4);
  plot(t,cos(t*4).*exp(-2*t));
```

x. **Cópia física de gráficos**

1. Uma saída gráfica das figuras dos gráficos atuais pode ser facilmente obtida com o comando print. Ao entrar este comando, é enviada uma cópia de alta-resolução com a figura contendo os gráficos atuais para a impressora default.
2. O arquivo printopt.m serve para a configuração padrão usada pelo comando print. Você pode mudar os padrões editando este arquivo (Ver help printopt).
3. O comando print NomeArquivo salva a figura gráfica atual para o arquivo indicado no arquivo de formato padrão. Se *NomeArquivo* não tem extensão, então uma extensão apropriada tal como .ps, .eps, ou .jet é anexada.
4. Por exemplo, se PostScript é o formato de arquivo padrão, então

   Freemat: Linhas de comando
   ```
   print lissajous
   ```

   cria um arquivo lissajous.ps com o gráfico atual que pode ser impressa na sequência usando o sistema de impressão. Se NomeArquivo já existe, ele será sobreposto a menos que você use a opção -append.
5. O comando

   Freemat: Linhas de comando
   ```
   print -append lissajous
   ```

   anexa a figura gráfica atual ao arquivo existente lissajous.ps. Deste modo, podemos salvar várias figuras gráficas em um mesmo arquivo.

6. A configuração padrão pode, de modo natural, ser sobreposta. Por exemplo,

Freemat: Linhas de comando
```
print -deps -f3 saddle
```

salva o arquivo Encapsulated PostScript `saddle.eps` com os gráficos da figura 3, mesmo que ela não seja a figura atual.

## 25.2 Gráficos de curvas no espaço

a. Análogo a `plot` em 2D, o comando `plot3` gera curvas no espaço 3D. Se `x`, `y` e `z` são vetores de mesma dimensão, então o comando `plot3(x,y,z)` produz um gráfico em perspectiva de uma curva linear por pedaços em 3D passando pelos pontos com coordenadas nos respectivos pontos `(x,y,z)`. Tais vetores são usualmente definidos na forma paramétrica.

b. Por exemplo,

Freemat: Linhas de comando
```
t = .01 : .01 : 20*pi;
x = cos(t);
y = sin(t);
z = t.^3;
plot3(x,y,z)
```

produz uma hélice cilíndrica que é comprimida próximo do plano `z=0`.

c. Da mesma forma que para gráficos planares, podemos anexar: um título e etiquetas nos eixos (incluindo `zlabel`). Os feitos do comando `axis` descritos para 2D, também valem para gráficos 3D; configurando a escala dos eixos para os limites prescritos, mas agora exigem um vetor 6-dimensional.

## 25.3 Gráficos de superfícies e de grade

a. Gráficos de superfícies gradeadas em 3D são plotados com o comando `mesh`. Aqui `mesh(z)` cria um gráfico em perspectiva 3D dos elementos da matriz `z`. Os pontos da superfície são definidos pelas coordenadas `z` relativos à grade retangular em `z=0`.

b. Por exemplo, digite

Freemat: Linhas de comando
```
[X,Y] = meshgrid(-1.6:.4:1.6);
```

c. Do mesmo modo, gráficos de superfícies 3D são plotados com o comando `surf`.

Freemat: Linhas de comando
```
surf(meshgrid(-1.6:.4:1.6))
```

d. Para plotar o gráfico de `z=f(x,y)` sobre um retângulo, devemos definir vetores `xx` e `yy` que particionam os lados do retângulo. Usando a função `meshgrid`, criamos uma matriz $x$, sendo cada linha igual a `xx` e cuja coluna tenha dimensão de `yy`, e do mesmo modo, criamos uma matriz `y`, em que cada coluna é igual a `yy`, como segue:

Freemat: Linhas de comando

```
[x,y] = meshgrid(xx,yy);
```

e. Assim calculamos uma matriz `z`, obtida pelo cálculo de `f` atuando ponto-a-ponto sobre as matrizes `x` e `y`, para as quais a função `surf` pode ser aplicados.

f. Por exemplo, você pode desenhar o gráfico de $z = e^{-x^2-y^2}$ sobre a região quadrada `[-2,2]x[-2,2]` como segue:

Freemat: Linhas de comando

```
xx = -2:.2:2;
yy = xx;
[x,y] = meshgrid(xx,yy);
z = exp(-x.^2 - y.^2);
surf(x,y,z)
```

g. Ao invés de gerar `meshgrid`, podemos trocar as três primeiras linhas do código acima por: `[x,y]=meshgrid(-2:.2:2,-2:.2:2);` e obter algo mais simples, como:

Freemat: Linhas de comando

```
[x,y] = meshgrid(-2:.2:2, -2:.2:2);
z = exp(-x.^2 - y.^2);
surf(x,y,z)
```

h. A função `repmat()` realiza *cópias* de uma matriz de acordo com o tipo de chamada.

Copia 1 linha com a matriz `x` repetida 1 vez:

Freemat: Linhas de comando

```
x = [1 2 3 4]
y = repmat(x,1,1)
y = 1 2 3 4
```

Copia 1 linha com a matriz `x` repetida 2 vezes:

Freemat: Linhas de comando

```
x = [1 2 3 4]
y = repmat(x,1,2)
y =
    1 2 3 4 1 2 3 4
```

Copia 2 linhas com a matriz `x` repetida 3 vezes:

Freemat: Linhas de comando

```
x = [1 2 3 4]
y = repmat(x,2,3)
y =
    1 2 3 4 1 2 3 4 1 2 3 4
    1 2 3 4 1 2 3 4 1 2 3 4
```

Copia 5 linhas com a matriz x repetida 1 vez:

Freemat: Linhas de comando

```
x = [1 2 3 4]
y = repmat(x,[5,1])
```

Copia 2 linhas com a matriz x repetida 3 vezes:

Freemat: Linhas de comando

```
x = [1 2 3 4]
y = repmat(x,[2,3])
```

i. Podemos construir uma outra função com o código:

Freemat: Linhas de comando

```
x = repmat(linspace(-1,1),[100,1]);
y = x';
r = x.^2 + y.^2;
z = exp(-r*3).*cos(5*r);
c = r;
surf(x,y,z,c)
axis equal
view(3)
```

j. Como observamos, os feitos do comando `axis` descritos na seção sobre gráficos no plano também valem para gráficos 3D, bem como os comandos para: título, etiquetas de eixos e o comando `hold`.

k. O sombreamento de cores de superfícies é obtido pelo comando `shading`. Três formas configuram o sombreamento: `faceted` que é o default, `interpolated` e `flat`, que podem ser chamados por qualquer um dos comandos indicados abaixo:

Freemat: Linhas de comando

```
shading faceted, shading interp, shading flat
```

l. As superfícies produzidas com `surf`, configuradas por `interpolated` e `flat` removem a sobreposição das linhas da rede. Experimente os diferentes tipos de sombreamento citados acima. O comando `shading` (bem como `colormap` e `view` abaixo) devem ser colocados *depois* do comando `surf`.

m. O perfil de cores de uma superfície é controlado pelo comando `colormap` e podem ser o seguintes:

Freemat: Linhas de comando

```
hsv (default), hot, cool, jet, pink, copper, flag, gray, bone
```

n. Por exemplo, o comando `colormap(cool)` irá configurar uma determinada cor para a figura atual. Faça testes com várias *colormaps* na superfície produzida acima.

o. O comando `view` mostra o ponto de vista de coordenadas esféricas ou cartesianas o objeto 3D. Ver `help view`.

p. A função `peaks` gera uma superfície bonita sobre a qual podemos experimentar o sombreamento com `shading`, `colormap` e `view`.

q. Podemos construir gráficos de superfícies parametrizadas com as funções `sphere` e `cylinder`. A superfície conhecida como Toro:

Freemat: Linhas de comando

```
function [x,y,z] = toro(r,n,a)
%TORO Gera um Toro com toro(r,n,a), raio central a, raio lateral r.
%     n indica o número de faces na superfície.
%     Variáveis de entrada opcionais com padrão r=0.5, n=50, a=1.
%     [x,y,z] = toro(r,n,a) gera três matrizes (n+1)x(n+1)
%     para construir surf(x,y,z) cuja saída gráfica é o toro.
if nargin < 3, a = 1;   end
if nargin < 2, n = 50;  end
if nargin < 1, r = 0.5; end
u = pi*(0:2:2*n)/n;
v = 2*pi*(0:2:n)'/n;
xx = (a + r*cos(v))*cos(u);
yy = (a + r*cos(v))*sin(u);
zz = r*sin(v)*ones(size(u));
if nargout == 0
    surf(xx,yy,zz)
    ar = (a + r)/sqrt(2);
    axis([-ar,ar,-ar,ar,-ar,ar])
else
    x = xx; y = yy; z = zz;
end
```

## 25.4 Manipulando Gráficos

O sistema gráfico do Freemat também possui funções de baixo nível que controlar quase todos os aspectos do ambiente gráfico ao produzir gráficos sofisticados. Estude o comando `set(1)` e `gca,set(ans)` para ver mais propriedades que podemos controlar.

# 26 Cálculos com Matrizes Esparsas

a. Em geral, as matrizes são cheias, isto é, a maioria das entradas é não-nula, mas, se uma matriz possui muitos *zeros*, o cálculo pode ser mais rápido, evitando muitas operações aritméticas sobre as entradas nulas e gastando menos memória que o exigido normalmente, guardando apenas as entradas não nulas da matriz. Desse modo, ocorre redução no tempo e no armazenamento quando o problema envolve matrizes com grande quantidade de entradas. Matrizes com muitos zeros são denominadas *esparsas*.

b. Existem duas formas para guardar matrizes: `full` e `sparse`, sendo que a forma cheia é o normal. A funções `full` e `sparse` realizam a conversão entre os dois modos.

c. Para a matriz `A`, o código gera uma informação sobre o número de zeros de `A`:

Freemat: Linhas de comando

```
a = [1,0,4,2,0; 0,0,0,0,0; 0,1,0,0,2]
a =
    1 0 4 2 0
    0 0 0 0 0
    0 1 0 0 2
A = sparse(a)
    A = Matrix is sparse with 5 nonzeros
```

d. Podemos obter a matriz cheia `A`, cigitando

Freemat: Linhas de comando

```
full(A)
ans =
      1 0 4 2 0
      0 0 0 0 0
      0 1 0 0 2
```

e. Se `A` é uma matriz, a função `nnz(A)` retorna o número de elementos não nulos de `A`.

Freemat: Linhas de comando

```
nnz(A)
    ans = 5
```

f. Uma matriz esparsa é guardada como um arranjo linear dos seus elementos não nulos usando os seus índices de linhas e colunas. A função `sparse` permite criar uma matriz esparsa pela listagem das entradas não nulas. Se o vetor `s` contém as entradas não nulas de `S` e os vetores de inteiros `i` e `j` listam os seus correspondentes índices, então `sparse(i,j,s,m,n)` cria a matriz esparsa `S` de ordem `mxn`.

g. Vamos construir uma matriz esparsa tal que `S(1,1)=5`, `S(2,2)=6`, `S(3,3)=7`, `S(4,1)=8`, `S(4,2)=9` e `S(4,3)=10`. Assim, definimos os índices de linhas e de colunas, bem como os seus valores:

Freemat: Linhas de comando

```
Linha  = [1 2 3 4 4 4];  % Índice da linha
Coluna = [1 2 3 1 2 3];  % Índice da coluna
Valor  = [5 6 7 8 9 10];
S = sparse(Linha,Coluna,Valor,4,3), full(S)
S = Matrix is sparse with 6 nonzeros
    ans =
          5  0  0
          0  6  0
          0  0  7
          8  9 10
```

h. A lista de saída com entradas não nulas na coluna de maior ordem com os índices de linha e coluna para cada entrada. A declaração `F=full(S)` reestabelece `S` para o modo cheio. Vamos verificar o que foi dito com uma matriz $A$ usando o comando

Freemat: Linhas de comando

```
issparse(A)
```

i. As matrizes análogas na forma esparsa para `eye`, `ones` e `randn` que valem para matrizes cheias, são, respectivamente, `speye`, `spones` e `sprandn`

j. Os comandos `spones` e `sprandn` aceitam argumentos matriciais e trocam somente as entradas nulas por valores iguais a `1` e números aleatórios normalmente distribuídos, respectivamente. `randn` também aceita que a estrutura de matriz esparsa seja randomizada. O comando `sparse(m,n)` cria uma matriz esparsa nula.

k. Um outro exemplo.

Freemat: Linhas de comando

```
n = 6;
e = floor(10*rand(n-1,1));
E = sparse(2:n,1:n-1,e,n,n)
    E = Matrix is sparse with 4 nonzeros
full(E)
    ans =
ans =
      0 0 0 0 0 0
      0 7 0 0 0 0
      0 0 4 0 0 0
      0 0 0 6 0 0
      0 0 0 0 7 0
```

l. As operações aritméticas de muitas funções podem ser aplicadas, independente do modo de armazenamento.

m. Sobre o armazenamento do resultado, operações sobre matrizes cheias produzem matrizes cheias. Na sequência, a letra `E` representa uma matriz esparsa e a letra `C` representa uma matriz cheia. Para as operações indicadas, obtemos uma matriz:

| Algumas Operações com matrizes | Resultado |
|---|---|
| `E+E, E*E, E.*E, E.*C, E^n, E.^n, E\E` | Matriz esparsa |
| `E+C, E*C, E\C, C\E` | Matriz cheia |
| `inv(E), chol(E), lu(E)` | Matriz esparsa |

n. Para uma matriz esparsa simétrica `S`, `eig(S)` é uma matriz cheia mas se `S` é anti-simétrica o resultado pode ser uma matriz cheia ou esparsa. A função `svd` exige como argumento uma matriz cheia.

o. Podemos comparar entre os dois modos de armazenamento, observando a eficiência na resolução de um sistema `Ax=b` de equações.

. . . . . Arquivo: FreematIntro.tex