Mini-curso de MATLAB e Octave para Cálculo Numérico

PET - Engenharia de Computação
Universidade Federal do Espírito Santo
http://www.inf.ufes.br/~pet

Sumário

1 Introdução

1.1 Matlab

O MATLAB (de MATrix LABoratory) é um programa produzido pela Mathworks, Inc. (maiores informações em http://www.mathworks.com), e a grosso modo serve para trabalhar com matrizes e números complexos da mesma forma como uma calculadora trabalha com números reais. Além disso, ele possui recursos de programação, agindo como uma linguagem procedural, semelhante a C, porém voltada para processamento numérico intensivo. Ele possui também programas de projeto de controle e recursos gráficos.

O MATLAB fornece também um conjunto de aproximadamente 200 subprogramas que solucionam problemas diversos tais como: álgebra matricial, aritmética com complexos, sistemas de equações lineares, determinação de autovalores e autovetores, solução de equações diferenciais, solução de equações não lineares, além de representar e de dar subsídios (sub-módulo) para a análise e para síntese de sistemas lineares e não lineares.

A interface do MATLAB é composta basicamente por uma janela de comandos, com um prompt característico (>>).

1.2 Octave

O Octave é uma linguagem de programação de alto nível, destinada ao tratamento de problemas para computação numérica. Ele é um Software Livre, produzido por uma grande equipe chefiada por John W. Eaton. Maiores informações disponíveis no site oficial do projeto: http://www.gnu.org/software/octave/.

O Octave pode efetuar cálculos aritméticos com números reais, escalares complexos e matrizes; resolver sistemas de equações algébricas; integrar funções sobre intervalos finitos e infinitos e integrar sistemas de equações diferenciais ordinárias e diferenciais algébricas.

A interface com o programador é basicamente através de uma linha de comando. Ele ainda gera gráficos 2D e 3D, utilizando o Gnuplot.

O Octave é em grande parte compatível com o MatLab. Os comandos apresentados nessa apostila servem tanto para Matlab quanto para o Octave.

Para iniciar o Octave, apenas digite *octave* no termnial do Linux.

2 Operações simples entre escalares

Conforme dito, um escalar é uma matriz 1x1 em MATLAB e Octave. As principais operações entre dois escalares serão apresentadas a seguir.

2.1 Soma

A operação "a + b"realiza a soma entre dois escalares. O exemplo 1 exibe sua execução.

Exemplo 1: *Soma entre dois escalares*

```
>> 3 + 5

ans =

     8
```

2.2 Subtração

O operador - é responsável pela subtração. O próximo exemplo mostra esta operação.

Exemplo 2: *Subtração entre dois escalares*

```
>> 4 - 8

ans =

    -4
```

2.3 Multiplicação

Para realizar a multiplicação de dois escalares utiliza-se o operador "* ", conforme o exemplo abaixo.

Exemplo 3: *Multiplicação entre dois escalares*

```
>> 7 * 2

ans =

    14
```

2.4 Divisão Direta

A divisão direta é realizada com uso da barra "/ ". Veja o exemplo a seguir.

Exemplo 4: *Divisão direta entre dois escalares*

```
>> 18 / 5

ans =

    3.6000
```

2.5 Divisão indireta

A divisão indireta é a divisão realizada da direita para esquerda, ou seja, o divisor é primeiro elemento. Nesse caso, utiliza-se o operador "\ ". O exemplo 5 mostra a divisão indireta de dois escalares.

Exemplo 5: *Divisão indireta entre dois escalares*

```
>> 10 \ 5

ans =

    0.5000
```

2.6 Exponenciação

Para efetuar a^b utiliza-se "∧ "entre a base e o expoente. O exemplo 6 ilustra como realizar a exponenciação de escalares.

Exemplo 6: *Exponenciação entre dois escalares*

```
>> 9 ^ 3

ans =

   729
```

2.7 Módulo

A função "abs"retorna o módulo de um número. O exemplo 7 ilustra o uso da função abs.

Exemplo 7: *Módulo de escalar*

```
>> abs(-5)

ans=

     5
```

3 Representação de matrizes e vetores no MATLAB e Octave

MATLAB trabalha essencialmente com um tipo de objeto, uma matriz retangular numérica (real ou complexa). Em algumas situações, denominações específicas são atribuídas a matrizes 1 por 1, que são os escalares, e a matrizes com somente uma linha ou coluna, que são os vetores.

A maneira mais fácil de se declarar matrizes é fazendo a explicitação da lista de elementos na linha de comando. Uma matriz pode ser criada, atribuindo a uma variável, valores representados entre colchetes, seguidos de ponto-e-vírgula. Os elementos das linhas são separados por espaços ou vírgulas, e as colunas, com ponto-e-vírgula. O exemplo 8 mostra formas de representação de uma matriz 3x3.

Exemplo 8: *Formas de declaração de matrizes*

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

A =

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

Um vetor nada mais é que uma matriz com uma de suas dimensões igual a 1, assim ele pode ser criado da mesma forma que uma matriz. No exemplo 9, tem-se um vetor-linha de dimensão 3, ou seja uma de uma matriz 1x3:

Exemplo 9: *Um vetor linha*

```
>> v = [1  2  3];
>> v

v =

     1     2     3
```

O exemplo 10 é de um vetor-coluna de dimensão 3, ou matriz 3x1.

Exemplo 10: *Um vetor coluna*

```
>> v = [1; 2; 3];
>> v

v =
```

```
5
6          1
7          2
8          3
```

Vale destacar que, da forma como foram apresentados a matriz ou o vetor, eles são armazenados na memória do programa, mas se quisermos que eles apareçam na tela, ou seja, visualizar o conteúdo da variável, devemos omitir o ponto-e-vírgula depois dos colchetes.

Depois que o vetor é criado, pode-se alterar um elemento acessando diretamente a sua posição. Observe o exemplo a seguir:

Exemplo 11: *Acesso a uma posição de um vetor*

```
1 >> v = [1 2 3];
2 >> v(2) = 0;
3 >> v
4
5 v =
6
7          1          0          3
```

No comando v(2) = 0, v é o nome da variável vetor e 2 é a posição cujo valor deve ser alterado, no caso para 0.

Pode-se também acessar uma posição inexistente no vetor. Nesse caso, as posições que não existiam até a posição acessada são automaticamente anuladas. O exemplo 12, mostra o que acontece quando é acessada a posição 5 do vetor do exemplo anterior.

Exemplo 12: *Acesso a uma posição inexistente de um vetor*

```
1 >> v(5) = 8;
2 >> v
3
4 v =
5
6          1          0          3          0          8
```

Repare que a nova dimensão do vetor agora é 5, exatamente a posição que não existia antes do acesso, e que a posição 4 foi preenchida com 0, pois não existia antes do acesso a uma posição inexistente do vetor.

Uma operação interessante é criar uma matriz usando uma já definida. No exemplo 13, a matriz z é composta de três elementos, sendo que um deles é uma matriz 1xn, ou vetor linha.

Exemplo 13: *Criando uma matriz com um vetor linha como elemento*

```
>> z = [4  5  v];
>> z

z =

     4     5     1     0     3     0     8
```

A matriz resultante é uma matriz coma a dimensão 7, que é devido aos dois elementos não pertencentes ao vetor v mais a dimensão deste, que é 5.

4 Operações simples entre matrizes

4.1 Soma e subtração

A soma e a subtração de duas matrizes seguem a mesma lógica para os escalares, ou seja, é efetuada elemento por elemento. O exemplo 14 mostra essas operações.

Exemplo 14: *Adição e subtração de entre duas matrizes 3x3*

```
>> A = [3  7  2; 3  11  9; 1  0  10];
>> B = [8  6  4; 0  1  0; 2  5  12];
>> A+B

ans =

    11    13     6
     3    12     9
     3     5    22

>> A-B

ans =

    -5     1    -2
     3    10     9
    -1    -5    -2
```

Vale lembrar que, caso as matrizes envolvidas sejam de dimensões diferentes, o programa acusará erro.

4.2 Multiplicação de uma matriz por um escalar

Na multiplicação de uma matriz por um escalar, cada um dos elementos da matriz é multiplicado por este. O operador utilizado no comando é mesmo que na multiplicação entre escalares. Veja o próximo exemplo, no qual A é a mesma matriz do tópico anterior.

Exemplo 15: *Multiplicação de uma matriz por um escalar*

```
>> 5*A

ans =

    15    35    10
    15    55    45
```

```
5     0     50
```

4.3 Multiplicação entre matrizes

Na multiplicação entre duas matrizes, A*B, o elemento i x j da matriz resultante é o somatório dos produtos entre os elementos das linhas i da primeira matriz pelos elementos das colunas j da segunda matriz. É necessário que o número de colunas da matriz A seja igual ao número de linhas de B, caso contrário o programa acusará erro. A multiplicação entre duas matrizes pode ser vista no exemplo 16.

Exemplo 16: *Multiplicação entre duas matrizes*

```
>> A*B

ans =

    28    35    36
    42    74   120
    28    56   124
```

As matrizes A e B são as mesmas do exemplo 4.1.

4.4 Divisão direta de matrizes

A divisão direta (A/B) entre duas matrizes é equivalente a multiplicar a matriz A pela inversa de B. No exemplo 17, tem-se a divisão direta entre duas matrizes.

Exemplo 17: *Divisão direta entre duas matrizes*

```

>> A = [9 4 0; 3 6 4; 0 9 2];
>> B = [8 3 1; 10 8 3; 4 9 8];
>> B/A

ans =

    0.7625    0.3792   -0.2583
    0.8625    0.7458    0.0083
   -0.2750    2.1583   -0.3167
```

4.5 Divisão indireta de matrizes

A divisão indireta entre duas matrizes (A\B) é equivalente a multiplicar a matriz B pela inversa de A. O exemplo abaixo mostra o resultado dessa operação entre as mesmas matrizes do exemplo anterior.

Exemplo 18: *Divisão indireta entre duas matrizes*

```
>> B\A
ans =

    1.92000    0.44667   -0.38667
   -2.88000   -0.25333    1.41333
    2.28000    1.18667   -1.14667
```

4.6 Exponenciação A^b com b sendo um escalar

A^b representa a multiplicação com b fatores iguais à matriz A. No exemplo abaixo, é efetuado o comando A^2.

Exemplo 19: *Operação de exponenciação*

```
>> A = [1  10  3; 0  9  6; 5  5  4];
>> A^2

ans =

     16    115     75
     30    111     78
     25    115     61
```

4.7 Operação elemento por elemento

O operador "."realiza uma dada operação, entre duas matrizes n x m, elemento por elemento. Se C = A.<operação>B, então $c_{ij} = a_{ij}$ <operação> b_{ij}. Observe o exemplo 20.

Exemplo 20: *Multiplicação entre duas matrizes, elemento por elemento*

```
>> [7  4  9  6  2].*[8  3  5  1  3]

ans =

     56     12     45      6      6
```

4.8 Transposta de uma matriz

O comando "'"calcula a transposta de uma matriz qualquer, ou seja, transforma as linhas em colunas, e vice-versa. No exemplo 21, é encontrada a transposta da matriz A vista no tópico anterior.

Exemplo 21: *Transposta de uma matriz*

```
>> A'

ans =

     1     0     5
    10     9     5
     3     6     4
```

4.9 Posto de uma matriz

O comando "rank"calcula o posto de uma matriz qualquer. No exemplo 22, é encontrado o posto de uma matriz.

Exemplo 22: *Posto de uma matriz*

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

ans =
     1  2  3
     4  5  6
     7  8  9
>> rank(B)
ans = 2
```

4.10 Número de condição de uma matriz

O comando "cond"calcula o número de condição de uma matriz qualquer. No exemplo 23, é encontrado o número de condição da matriz B usada no exemplo anterior.

Exemplo 23: *Número de condição de uma matriz*

```
>> cond(B)
ans = 6.0262e+16
```

4.11 Maior elemento

O comando "max"retorna o maior elemento de cada coluna. No exemplo 24, são encontrados os maiores elemetos de cada coluna de uma matriz.

Exemplo 24: *Maior valor de cada coluna*

```
>> A=[1  10  3;16  5  6;7  8  9]
A=
     1    10     3
    16     5     6
     7     8     9
>> max(A)
ans=

    16    10     9
```

Outra forma de usar o comando "max"é passando dois argumentos de entrada, como no exemplo 25. O primeiro argumento é uma matriz ou um vetor e o segundo é um escalar. O comando irá comparar os elementos da matriz ou do vetor com o escalar de entrada e trocará pelo escalar os elementos menores que ele.

Exemplo 25: *Usando a função "max"para comparar elementos com um escalar fixo*

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

    1    2    3
    4    5    6
    7    8    9

>> max(A,5)
ans =

    5    5    5
    5    5    6
    7    8    9

>> v=[1  2  3  4]
v =

    1    2    3    4

>> max(v,3)
```

```
ans =

    3     3     3     4
```

Existe mais uma forma de usar esse comando. Se for usado sobre uma matriz retorna o maior elemento da coluna e a linha que o elemeto está. Se for usado sobre um vetor retorna o maior elemento e sua posição. No exemplo 26 em 'x' é armazenado o maior valor de cada coluna da matriz e em 'xi' a linha de cada elemento. No exemplo 27 em 'x' é armazenado o maior elemento do vetor e em 'xi' a sua posição.

Exemplo 26: *Outra forma de usar a função "max"sobre matriz*

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

    7     2     3
    4     5     9
    1     8     6

>> [x,xi]=max(A)
x =

    7     8     9

xi =

    1     3     2
```

Exemplo 27: *Outra forma de usar a função "max"sobre vetor*

```
>> v=[1  2  3]
v =

    1     2     3

>> [x,xi]=max(v)
x =  3
xi =  3
```

4.12 Tamanho da matriz

O comando "size"retorna a quantidade de linha e de colunas de uma matriz. No exemplo 28, são encontrados a quantidade de linhas e colunas de uma matriz.

Exemplo 28: *Quantidade de linhas e colunas*

```
A=[1  2  3;4  5  6]
A=

    1    2    3
    4    5    6

>>  [nl,nc]=size(A)
nl=2
nc=3
```

4.13 Tamanho do vetor

O comando "length"retorna a quantidade de elementos de um vetor. No exemplo 29, é encontrado a quantidade de elementos de um vetor.

Exemplo 29: *Quandade de elementos*

```
>>A=[1  2  3]
A=

    1    2    3

>>length(A)
ans=3
```

4.14 Vetor igualmente espaçado

O comando "linspace"gera um vetor igualmente espaçado tendo uma base e um limite e a quantidade de elementos desejados. No exemplo 30, é gerado um vetor igualmente espaçado.

Exemplo 30: *Vetor igualmente espaçado*

```
>>linspace(1,10,10)
ans=

    1    2    3    4    5    6    7    8    9   10
```

4.15 Geração de matrizes

- **A=ones(n,m)**: Retorna uma matriz n por m com todos os elementos sendo '1'. No exemplo 31, é gerada a matriz A 2 por 3.

Exemplo 31: *Matriz de uns*

```
>> ones(2,3)
ans =

    1    1    1
    1    1    1
```

Essa função pode ser usada só com um parâmetro gerando uma matriz quadrada como no exemplo 32.

Exemplo 32: *Outra de forma de usar a função "ones"*

```
>> ones(3)
ans =

    1    1    1
    1    1    1
    1    1    1
```

- **A=zeros(n,m)**: Retorna uma matriz n por m com todos os elementos sendo '0'. No exemplo 33, é gerada a matriz A 3 por 2.

Exemplo 33: *Matriz de zeros*

```
>> A=zeros(3,2)
A=

     0     0
     0     0
     0     0
```

Essa função pode ser usada só com um parâmetro gerando uma matriz quadrada como no exemplo 34.

Exemplo 34: *Outra forma de usar a função "zeros"*

```
>> zeros(2)
ans =

     0     0
```

```
5      0      0
```

- **A=eye(n)**: Retorna uma a matriz identidade de ordem 'n'. No exemplo 35, é gerada a matriz A de ordem 3.

Exemplo 35: *Matriz identidade*

```
1  >> A=eye(3)
2  A =
3
4      1      0      0
5      0      1      0
6      0      0      1
```

Essa função pode ser usada com dois parâmetros, indicando o número de linhas e colunas. No exemplo 36, é gerada uma matriz A 3 por 2.

Exemplo 36: *Outra forma de usar a função "eye"*

```
1  >> A=eye(2,3)
2  A =
3
4      1      0      0
5      0      1      0
```

- **A=hilb(n)**: Retorna a matriz de Hilb de ordem n. No exemplo 37, é gerada a matriz de Hilb de ordem 3.

Exemplo 37: *Matriz de Hilb*

```
1  >> A=hilb(3)
2  A=
3
4      1.00000    0.50000    0.33333
5      0.50000    0.33333    0.25000
6      0.33333    0.25000    0.20000
```

- **A=rand(n,m)**: Retorna a matriz n por m com seus elementos gerados aleatoriamente entre 0 e 1. No exemplo 38, é gerada a matriz A 2 por 3.

Exemplo 38: *Matriz aleatória*

```
1  >> A=rand(2,3)
2  A=
```

```
0.085231    0.134837    0.307846
0.681140    0.634245    0.658524
```

Essa função pode ser usada só com um parâmetro gerando uma matriz quadrada como no exemplo 39.

Exemplo 39: *Outra forma de usar a função "rand"*

```
>> rand(2)
ans =

   0.14584    0.39775
   0.87865    0.82461
```

- **V=diag(A,n)**: Retorna um vetor coluna com os elementos da diagonal 'n' da matriz B. Se a diagonal desejada for a diagonal principal, o 'n' será indicado por '0'. Caso seja uma diagonal abaixo da diagonal principal o 'n' será um número negativo, sendo a diagonal logo abaixo da principal indicada por '-1', e se for acima, o 'n' será um número positivo, sendo a diagonal logo acima da principal indicada por '1'. No exemplo 40, é gerado o vetor V com os elementos da diagonal '-1' da matriz A.

Exemplo 40: *Vetor com a diagonal*

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

    1    2    3
    4    5    6
    7    8    9
>> V=diag(A,-1)
V=

    4
    8
```

Essa função pode ser usada só com um parâmetro, gerando um vetor coluna contendo a diagonal principal, ou sobre vetores, gerando uma matriz diagonal, como no exemplo 41.

Exemplo 41: *Outras formas de usar a função "diag"*

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

```
3
4       1     2     3
5       4     5     6
6       7     8     9
7
8   >> v=diag(A)
9   v =
10
11      1
12      5
13      9
14
15  >> diag(v)
16  ans =
17
18      1     0     0
19      0     5     0
20      0     0     9
```

4.16 Módulo dos elementos da matriz ou vetor

O comando "abs"que é usado para determinar o módulo de um escalar também pode ser usado sobre matrizes e vetores como no exemplo 42. Quando usado sobre matrizes ou vetores ele retornando uma matriz ou vetor com o módulo de todos os elementos.

Exemplo 42: *Módulo dos elementos de uma matriz ou vetor*

```
1   >> A=[-1  -2;-3  -4]
2   A =
3
4      -1   -2
5      -3   -4
6
7   >> abs(A)
8   ans =
9
10      1     2
11      3     4
12
13  >> v=[-1  -2]
14  v =
```

```
15
16     -1    -2
17
18 >> abs(v)
19 ans =
20
21     1     2
```

5 Resolução de sistemas lineares

Considere o sistema linear abaixo:

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\
&\vdots \\
&a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n
\end{aligned}
$$

Esse sistema pode ser escrito, na forma de matrizes, como A*x = b, tal que:

$$
A = \begin{bmatrix} a11 & \cdots & a1n \\ \vdots & \ddots & \vdots \\ an1 & \cdots & ann \end{bmatrix} \quad x = \begin{matrix} x1 \\ \vdots \\ xn \end{matrix} \quad b = \begin{matrix} b1 \\ \vdots \\ bn \end{matrix}
$$

Figura 1: **Sistema Ax = b**

Pode-se calcular a solução do sistema diretamente, usando os comandos x = A \ b ou x = inv(A)*b. Ambos os comandos fazem a divisão de b por A ou a multiplicação da inversa de A por b.

Na resolução de sistemas lineares por métodos iterativos, é interessante determinar se a solução converge. Nesse caso, deve-se considerar os autovalores da matriz A.

A seguir, alguns comandos úteis na resolução de sistemas lineares. Nos próximos exemplos, considere a seguinte matriz: A = [1 2 3; 4 5 6; 7 8 0];

- **inv(A)**: Calcula a inversa da matriz quadrada A; [1]

Exemplo 43: *Cálculo da inversa de uma matriz*

```
>> inv( A )

ans =

    -1.7778    0.8889   -0.1111
     1.5556   -0.7778    0.2222
    -0.1111    0.2222   -0.1111
```

- **eig(A)**: Retorna um vetor com os autovalores da matriz A;

[1] Uma matriz quadrada é aquela que possui o mesmo número de linhas e de colunas.

Exemplo 44: *Cálculo de autovalores de uma matriz*

```
>> eig( A )

ans =

    12.1229
    -0.3884
    -5.7345
```

- **[V, D] = eig(A)**: Produz matrizes de autovalores (D) e auto-vetores (V) da matriz A, de forma que A*V = V*D. A matriz D é a forma canônica de A menos uma matriz diagonal com os autovalores de A na diagonal principal. As colunas da matriz V são os auto-vetores de A;

Exemplo 45: *Cálculo de auto-vetores e autovalores de uma matriz*

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

V =

    -0.2998   -0.7471   -0.2763
    -0.7075    0.6582   -0.3884
    -0.6400   -0.0931    0.8791

D =

    12.1229         0         0
          0   -0.3884         0
          0         0   -5.7345
```

- **det(A)**: Calcula o determinante da matriz A;

Exemplo 46: *Cálculo do determinante de uma matriz*

```
>> det( A )

ans =

    27
```

- **fliplr(A)**: Troca o lado esquerdo pelo lado direito de uma matriz;

Exemplo 47: *Troca de colunas de uma matriz*

```
>> fliplr( A )

ans =

     3     2     1
     6     5     4
     0     8     7
```

- **[L, U, P] = lu(A)**: Na resolução de um sistema linear pelo método LU, pode-se usar esse comando, que decompõe A nas matrizes L (triangular inferior com os elementos da diagonal iguais a 1), U (triangular superior) e P (matriz da permutação).

Exemplo 48: *Decomposição LU de uma matriz*

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

L =

    1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000


U =

    7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000


P =

     0     0     1
     1     0     0
     0     1     0
```

6 Polinômios

Os polinômios no Octave/MatLab são representados por um vetor, cujos coeficientes das potências em ordem decrescente são os elementos do vetor. No exemplo 49 temos a representação do polinômio $x^2 + x + 1$ e do polinômio $2x^2 - 5x$.

Exemplo 49: *Representação de polinômios*

```
>> polinomio1=[1 1 1]
polinomio1 =

    1    1    1

>> polinomio2=[2 -5 0]
polinomio2 =

    2   -5    0
```

6.1 Raízes reais

Para se encontrar as raízes reais de um polinômio, basta utilizar a função **roots(v)**, que retorna em um vetor coluna, as raízes de um polinômio. No exemplo 50, as raízes do polinômio $x^2 + 3x - 4$ são -4 e 1.

Exemplo 50: *Raízes de polinômio*

```
>> roots( [1, 3, -4] )

ans =

      -4
       1
```

6.2 Avaliação de polinômio

Para avaliar um polômio em vários pontos pode-se usar a função polyval. Como entrada temos o vetor com os coeficientes e um vetor com os pontos a serem avaliados.

Exemplo 51: *Avaliação do polinômio*

```
>> pol=[1 1 1]
pol=

```

```
    1     1     1

>> pto=[1  2  3]
pto=

    1     2     3

>> polyval(pol,pto)
ans=

     3     7    13
```

6.3 Interpolação polinomial

Em muitas situações, deseja-se extrair informações de uma tabela ou derivar/integrar uma função complexa. Nesse caso, pode-se fazer uma interpolação de **n** pontos e obter-se um polinômio de grau **n-1**, para aproximar uma função em estudo ou calcular o valor da função num ponto não tabelado.

6.3.1 Interpolação pela resolução de um sistema linear

Um dos métodos de interpolação é resolver um sistema linear A*x = b, em que A é a matriz de Vandermonde dos pontos xi, em que x é o vetor de coeficientes do polinômio integrador e b é o vetor de valores yi da função em estudo nos pontos xi. Em notação de matrizes, o sistema linear fica:

$$Ax = b \Rightarrow \begin{pmatrix} 1 & \cdots & x_0^{n-1} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Figura 2: **Sistema Ax = b**

No MATLAB, **vander(v)** retorna a transposta da matriz de Vandermonde espelhada, em que os elementos são potências do vetor v, de forma que, A(i,j) = v(i)∧(n-j). O exemplo 52 mostra como utilizar o comando.

Exemplo 52: *Transposta da matriz de Vandermonde espelhada*

```
>> vander ( [2, 3, 4] )
```

```
2
3  ans =
4
5        4     2     1
6        9     3     1
7       16     4     1
```

Assim, tem-se que realizar o comando **fliplr(vander(v))'**, para que se encontre a matriz de Vandermonde no formato correto. O próximo exemplo mostra como encontrar a matriz de Vandermonde do vetor [2; 3; 4], no formato conhecido:

Exemplo 53: *Matriz de Vandermonde*

```
1  >> fliplr( vander( [2, 3, 4] ) )'
2
3  ans =
4
5        1     1     1
6        2     3     4
7        4     9    16
```

6.3.2 Método direto

Pode-se também usar uma função que retorne, de maneira direta, os coeficientes do polinômio interpolador. Por exemplo:

- **p = poly(r)**: Retorna um vetor p de coeficientes do polinômio cujas raízes são os elementos de r. O MATLAB apresenta polinômios como vetores-linha com os coeficientes em ordem decrescente de potências. Assim, no exemplo 54 o polinômio encontrado é $x^4 - 10x^3 + 35x^2 - 50x + 24$;

Exemplo 54: *Polinômio interpolador*

```
1  >> poly( [1, 2, 3, 4] )
2
3  ans =
4
5       1    -10     35    -50     24
```

- **p = polyfit(x, y, n)**: Encontra os coeficientes do polinômio p de grau n que se aproxima da função que se ajusta aos pontos (xi, yi), dado pelos vetores x e y. Veja o próximo exemplo.

Exemplo 55: *Polinômio interpolador de grau n*

```
>> polyfit( [1, 2, 3] , [0, 6, 14] , 2 )

ans =

    1.0000    3.0000   -4.0000
```

O polinômio de grau 2 obtido no exemplo 55 é $x^2 + 3x - 4$.

7 Gráficos em MATLAB e Octave

Em disciplinas como Cálculo Numérico, Cálculo II e Cálculo III, os gráficos podem ser muito úteis para comparar métodos computacionais de resolução de problemas.

7.1 Gráficos 2-D

Os gráficos em duas dimensões são os mais comuns por serem de simples entendimento e representação. Tais gráficos podem ser de coordenadas cartesianas ou polares.

7.1.1 Coordenadas Cartesianas

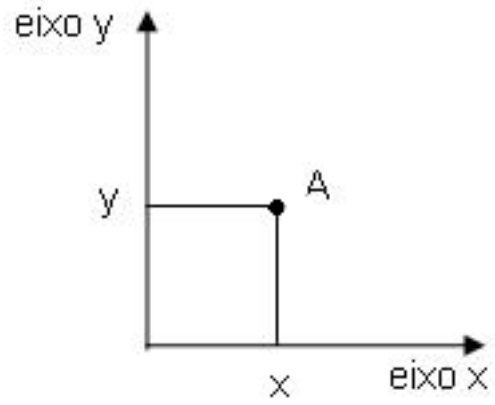

Figura 3: **Eixos cartesianos**

Comandos para gerar gráficos

- **plot(x, y)**: Gera gráficos lineares com x sendo a variável independente e y a variável dependente;
- **plot(x, y, z, w)**: Plota dois gráficos (ou mais, dependendo do número de argumentos). O MATLAB seleciona linhas diferentes para cada gráfico;

  No exemplo 56, são gerados dois gráficos. Um deles referente ao par de vetores [1, 2, 3] e [4, 3, 6] e outro, ao par [1, 2, 3] e [5, 7, 6]. Os gráficos podem ser vistos na figura 4.

Exemplo 56: *Dois gráficos em uma só figura*

```
1 >> plot( [1, 2, 3], [4, 3, 6], [1, 2, 3], [5, 7, 6] )
```

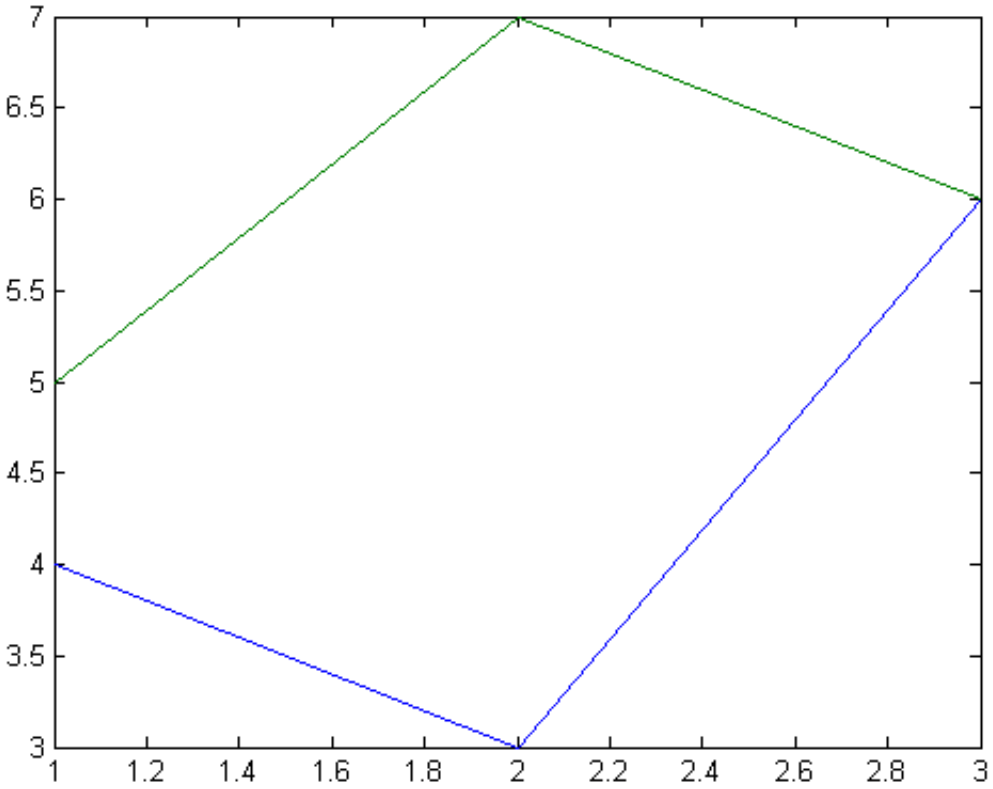

Figura 4: **Gráfico gerado no exemplo 56**

- **semilogx(x, y)**: Gera gráficos com y na escala linear e x na escala logarítmica.

  No exemplo 57 é gerado um gráfico no qual o vetor [1, 2, 3] está na escala logarítmica e [5, 4, 6] na linear. O gráfico gerado está na figura 5. [2]

Exemplo 57: *Gráfico com x na escala logarítmica*

```
1 >> semilogx( [1, 2, 3], [5, 4, 6] )
```

[2] Ter um eixo na escala logarítima significa que o gráfico apresenta o log_{10} dos valores fornecidos para aquele eixo.

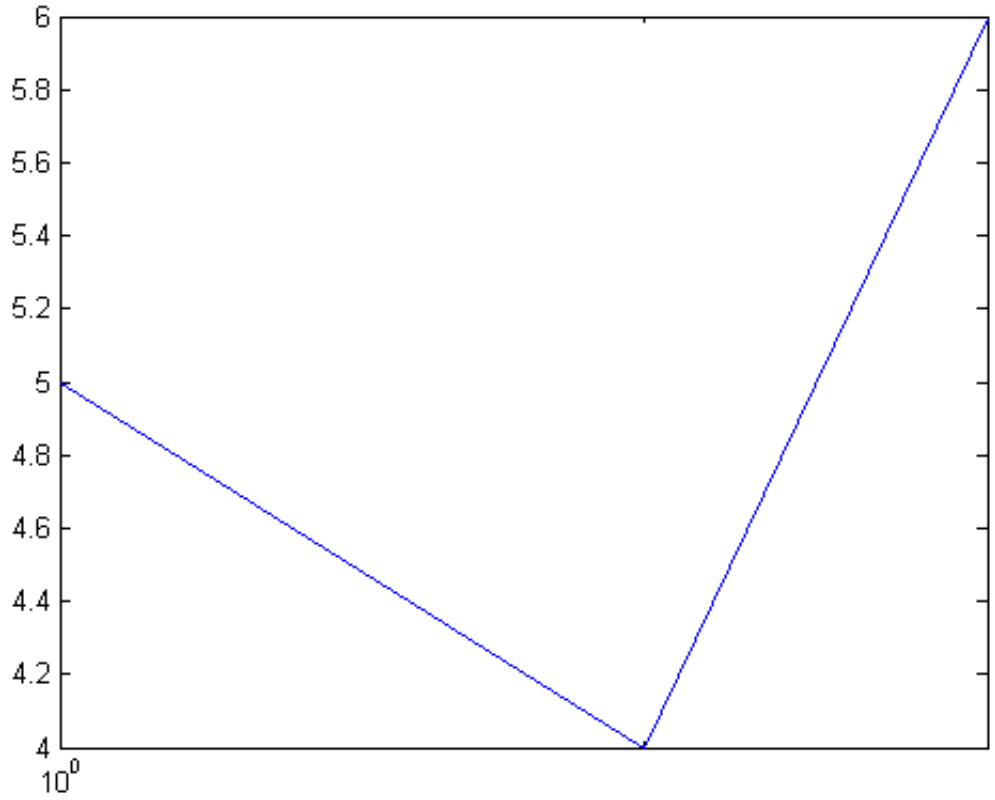

Figura 5: **Gráfico gerado no exemplo 57**

- **semilogy(x, y)**: Gera gráficos com x na escala linear e y na escala logarítmica.
- **loglog(x, y)**: Gera gráficos com x e y nas escalas logarítmicas.

  Um gráfico no qual os vetores [1, 2, 3] e [5, 4 6] estão na escala logarítmica está na figura 6. O exemplo 58 é o comando utilizado para gerar tal gráfico.

Exemplo 58: *Gráfico com x e y na escala logarítmica*

```
1 >> loglog( [1, 2, 3], [5, 4, 6] )
```

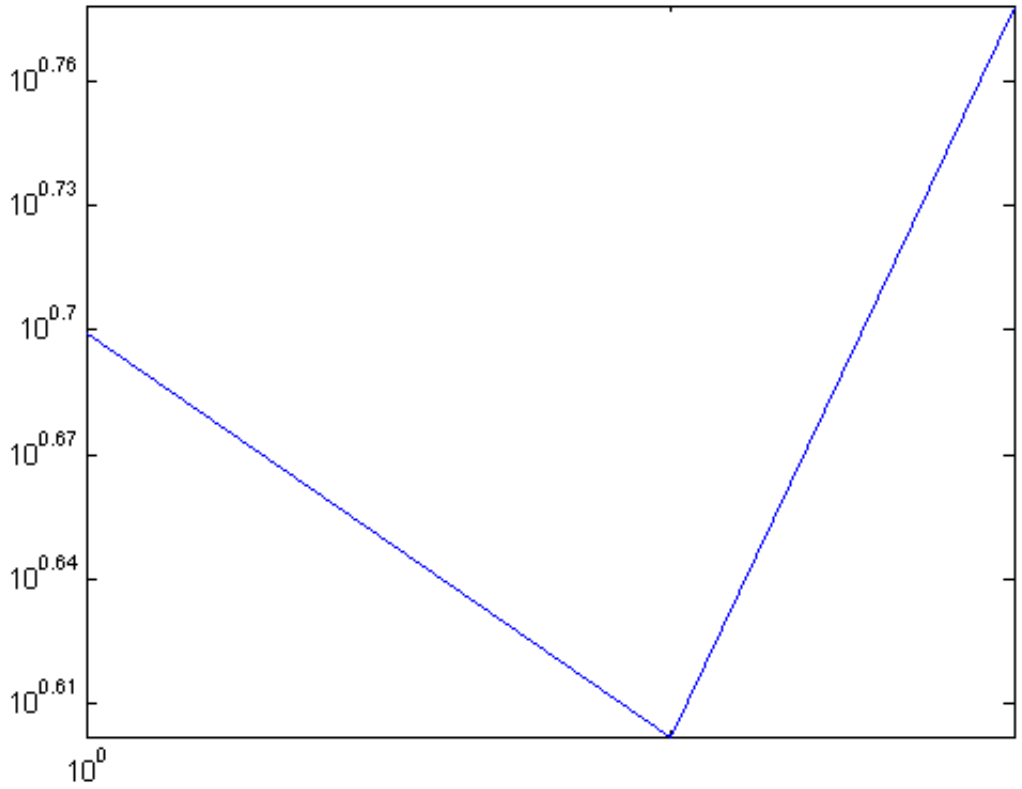

Figura 6: **Gráfico gerado no exemplo 58**

Comandos auxiliares

- **title('texto')**: Comando para adicionar um título (texto) ao topo do gráfico;
- **xlabel('texto')**: Escreve uma legenda no eixo x;
- **ylabel('texto')**: Escreve uma legenda no eixo x;
- **text(x, y, 'texto')**: Escreve um texto no ponto (x,y). Se x e y são vetores, o texto é escrito a cada ponto;
- **gtext('texto')**: Escreve um texto no ponto determinado pela posição do mouse no gráfico;
- **legend('texto1', 'texto2', ...)**: Coloca legendas nos gráficos, na ordem em que eles foram plotados, no canto superior direito da figura;

- **legend('texto1', 'texto2', 'location', 'pos')**: Coloca as legendas na posição indicada por "pos", que podem ser: north, south, east, weast, northeast, northwest, southeast, southwest.

- **grid on**: Adiciona grades no gráfico plotado;

- **grid off**: Retira as grades do gráfico corrente;

- **hold on e hold off**: Usa-se para plotar outro gráfico na mesma figura. Este comando mantém o gráfico antigo até que o hold off seja usado. Assim, consegue-se sobrepor os gráficos.

- **axis(v)**: Este comando é utilizado para representar o gráfico em uma determinada faixa de valores determinada pelo vetor v. O vetor tem como elementos os valores mínimo e máximo dos eixos x e y ([xmin xmax ymin ymax]).

- **axis 'parâmetro'**: Se o parâmetro for 'square', a função transforma a região do gráfico em quadrada ou cúbica. Se for 'tight', ela ajusta a região de acordo com os dados. E se for 'normal', ela remove qualquer ajuste definido e ajusta o gráfico da melhor forma possível.[3]

- **plot (x, y, 'parâmetro')**: 'Parâmetro' é uma combinação de características, tipo de linha, de marcador e de cor, que se deseja aplicar ao gráfico. Não há ordem específica para a combinação dos parâmetros nem é necessário determinar os três. A tabela 1 exibe os tipos de parâmetros que podem ser utilizados no comando plot para o MATLAB. Já os parâmetros para o comando plot no Octave são mostrados na tabela 2.

Tabela 1: Opções do plot para MATLAB

Cores		**Linhas**		**Marcador**	
amarelo	y	sólida	-	ponto	.
azul	b	tracejada	–	quadrado	S
azul claro	c	traço-ponto	-.	círculo	0
branco	w	ponteada	:	cruz	+
vermelho	r			X	X
preto	k			estrela	*
roxo	m			triângulo	∧
verde	g			triângulo invertido	V

[3]Para maiores informações sobre os tipos de parâmetros aceitos por axis, digite *help axis* no terminal.

Tabela 2: Opções do plot para Octave

Cores			**Linhas**		**Marcador**	
azul	b	3	sólida	-	cruz	+
branco	w	6	apenas pontos	.	quadrado	s
ciano	c	5			círculo	o
preto	k	0			triângulo invertido	V
roxo	m	4			X	X
verde	g	2			estrela	*
vermelho	r	1			triângulo	∧

No próximo exemplo, são criados dois gráficos numa mesma figura, um para o seno (linha ponteada vermelha e com pontos marcados com +) e outro para o cosseno (linha traço-ponto azul e com pontos marcados com quadrado). Os valores de x utilizados são 0, $\pi/4$, $\pi/2$ e $3\pi/4$ e π. São também criados o nome dos eixos, uma legenda para os gráficos e o título do gráfico.

Exemplo 59: *Gráficos do seno e do cosseno em uma mesma figura*

```
>> x = 0 : pi / 4 : pi ;
>> plot (x, sin(x) , 'r-' ,x, cos(x), 'b-*')
>> legend( 'seno', 'cosseno','location', 'southwest' )
>> title( 'seno e cosseno' )
>> xlabel( 'eixo x' )
>> ylabel( 'eixo y' )
>> grid on
```

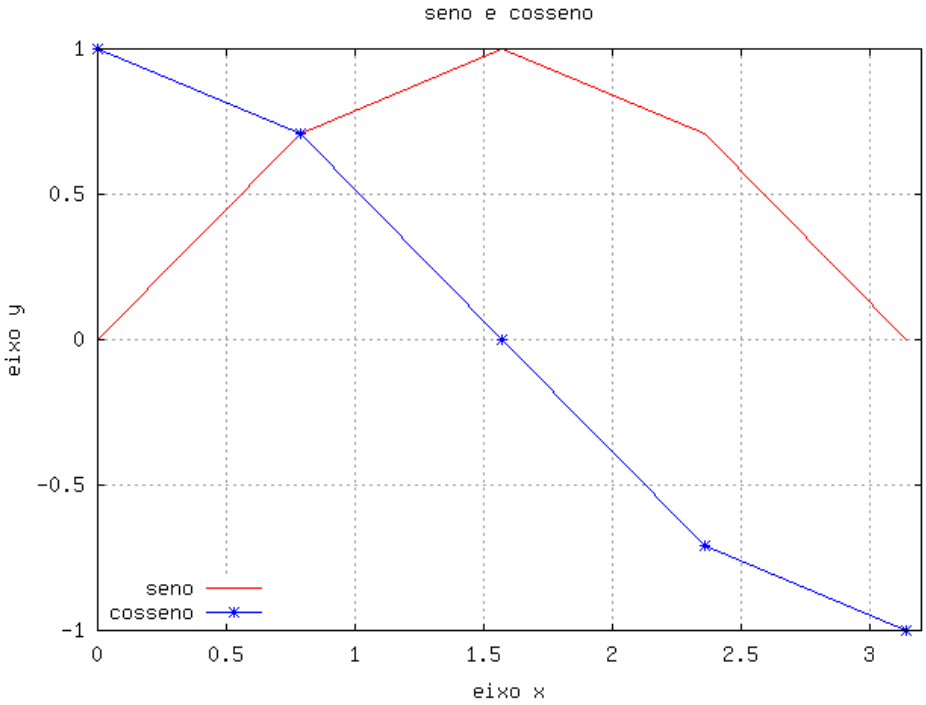

Figura 7: **Gráfico gerado no exemplo 59 - Octave**

7.1.2 Coordenadas Polares

Para representar um ponto em coordenadas polares necessitamos somente de uma semi-reta com origem em O e o ponto a ser representado. Dessa forma, o ponto é descrito pelo ângulo entre o segmento OP e a semi-reta O e a magnitude desse segmento. Assim, P pode ser escrito como (Θ,r).

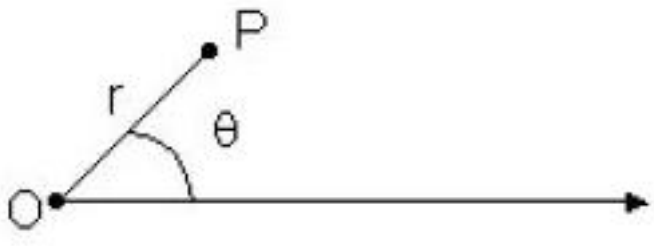

Figura 8: **Coordenadas polares no Matlab**

Para gerar gráficos em coordenadas polares:

- **polar(ângulo, r, 'parâmetros')**: Gera gráficos polares com ângulo em radianos, r uma função do ângulo e parâmetros são combinações do tipo de linha, da cor e do ponto.

  No exemplo 60, estão os comandos para gerar o gráfico, em coordenadas polares, r = sin(2*t)*cos(2*t). O gráfico obtido está na figura 9.

Exemplo 60: *Gráfico em coordenadas polares*

```
>> t = 0:.01:2*pi;
>> polar( t, sin( 2*t ).*cos( 2*t ), '--r' )
>> title( 'coordenadas polares' )
```

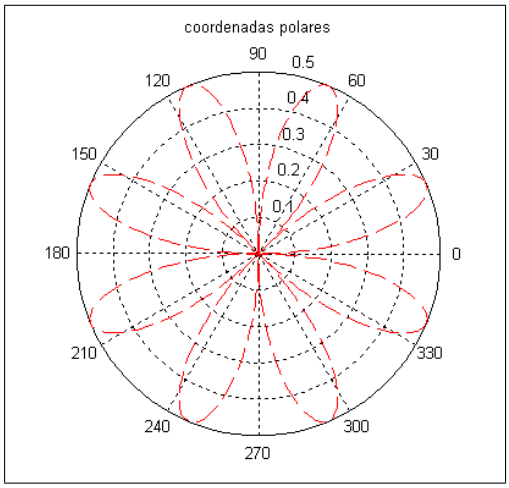

Figura 9: **Gráfico gerado no exemplo 60 - Matlab**

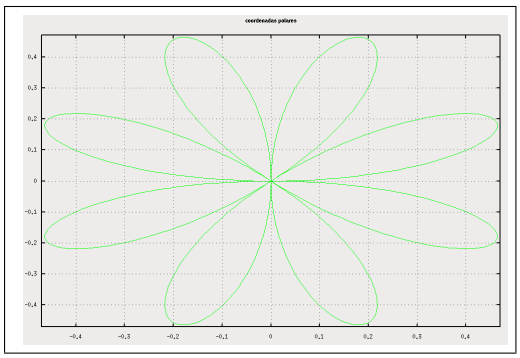

Figura 9: **Gráfico gerado no exemplo 60 - Octave**

7.2 Gráficos 3-D

- **plot3(x, y, z)**: Plota gráficos no espaço 3D. Observe, no exemplo 61, o comando para gerar o gráfico (x, y, z)=(sen(t), cos(t), t). A figura 10 é o gráfico obtido.

Exemplo 61: *Gráfico 3-D*

```
>> t = 0:pi/50:10*pi;
>> plot3( sin( t ), cos( t ), t )
>> grid on
>> axis normal
```

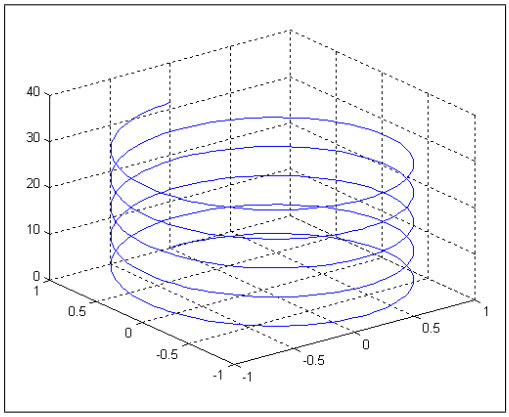

Figura 10: **Gráfico gerado no exemplo 61 - Matlab**

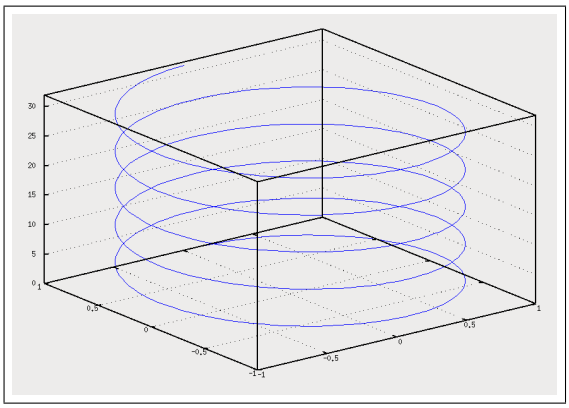

Figura 10: **Gráfico gerado no exemplo 61 - Octave**

- **mesh(x, y, z)**: Plota uma malha em 3D. Superfícies tipo do mesh são úteis para gerar gráficos de funções de duas variáveis.

  O primeiro passo para plotar uma função de 2 variáveis z=f(x,y) é gerar matrizes x e y contendo linhas e colunas repetidas, respectivamente, para funcionarem como o domínio da função. A função **meshgrid** transforma o domínio especificado por dois vetores em duas matrizes x e y. Essas matrizes então são usadas para avaliar a função de 2 variáveis. O exemplo 62 gera o gráfico referente à função f(x,y) = $\frac{sen(x^2+y^2)^{\frac{1}{2}}}{(x^2+y^2)^{\frac{1}{2}}}$.

  O comando **eps** no exemplo 62 representa a precisão da máquina e evita uma divisão por zero na linha 3.

Exemplo 62: *Malha em 3D*

```
1 >> [x,y] = meshgrid( -8:.5:8, -8:.5:8 );
2 >> r = sqrt( x.^2 + y.^2 ) + eps;
3 >> z = sin( r )./r;
4 >> mesh( x, y, z )
```

A figura 11 é o gráfico gerado no exemplo anterior. A fim de serem visualizadas as grades, a figura foi editada para um marcador do tipo ponto.

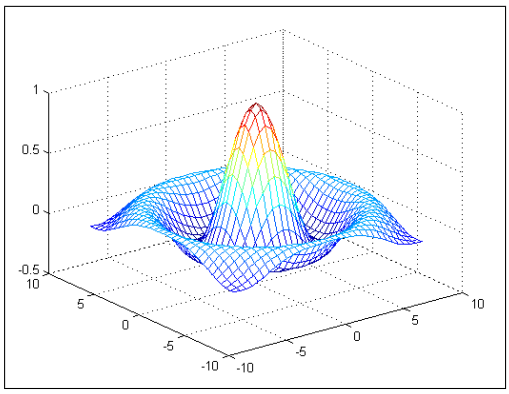

Figura 11: **Gráfico gerado no exemplo 62 - Matlab**

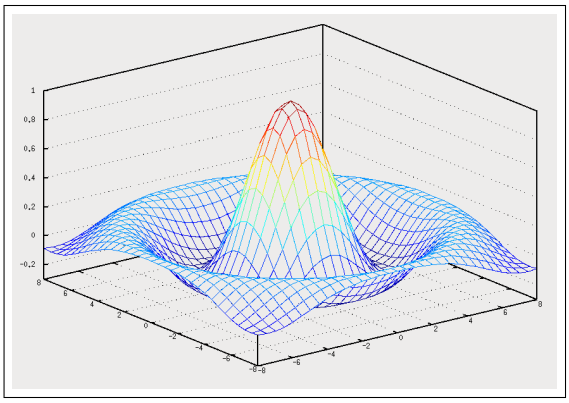

Figura 11: **Gráfico gerado no exemplo 62 - Octave**

- **contour(z, w)**: Plota linhas de contorno em duas dimensões. Se z depender de x e y, e w for uma constante, o resultado é a projeção de um gráfico 3D no plano xy com w curvas de nível. O exemplo 63 gera a figura 12, que é o contorno em 2D da função do exemplo anterior.

Exemplo 63: *Contorno em 2D*

```
>> contour( z, 10 )
```

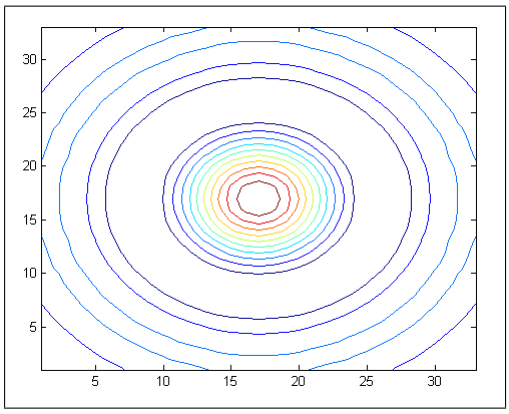

Figura 12: **Gráfico gerado no exemplo 63 - Matlab**

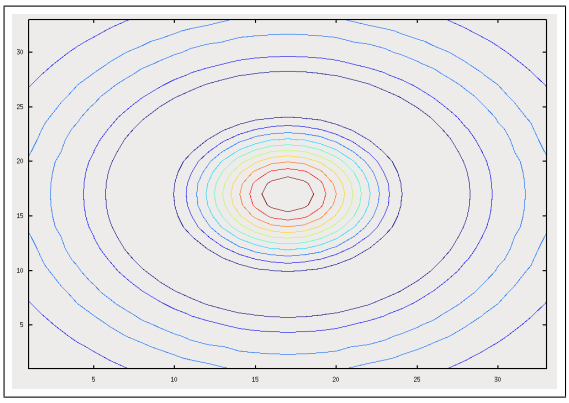

Figura 12: **Gráfico gerado no exemplo 63 - Octave**

- **contour3(x, y, z, n)**: Plota contorno em 3D com n iso-linhas. Na figura 13 tem-se o contorno 3D da função do exemplo 62. O exemplo 64 é o comando pra gerar tal contorno.

Exemplo 64: *Contorno em 3D*

```
1 >> contour3(x, y, z, 40)
```

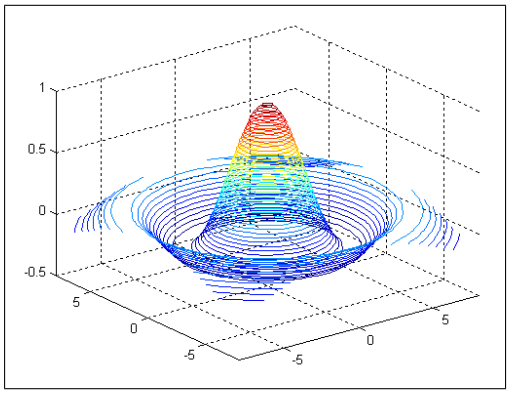

Figura 13: **Gráfico gerado no exemplo 64 - Matlab**

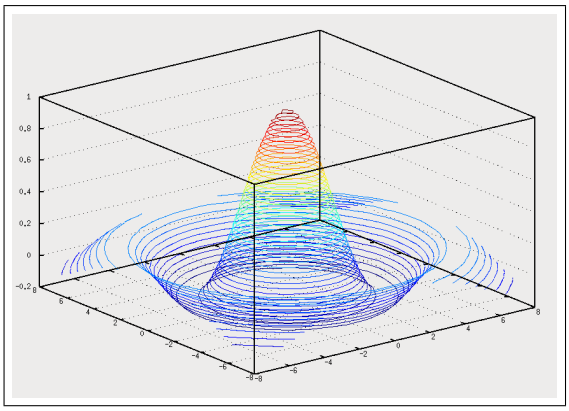

Figura 13: **Gráfico gerado no exemplo 64 - Octave**

- **surf(x, y, z)**: Plota superfície 3D. A superfície 3D da função do exemplo 62 é gerada pelo comando do exemplo 65. O gráfico obtido está na figura 14.

Exemplo 65: *Superfície 3D*

```
1 >> surf( x, y, z )
```

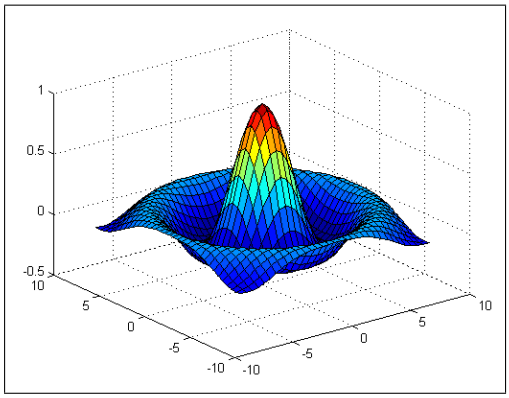

Figura 14: **Gráfico gerado no exemplo 65 - Matlab**

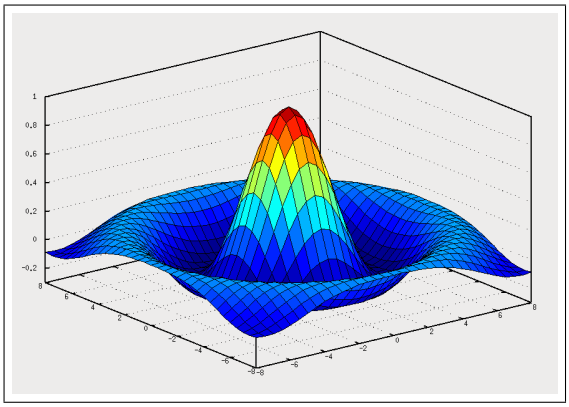

Figura 14: **Gráfico gerado no exemplo 65 - Octave**

- **surface(x, y, z)**: Plota a superfície de contorno 2D de um gráfico 3D. O exemplo 66 gera a figura 15, que é o contorno 2D do gráfico 3D da função do exemplo 62. A cor de cada ponto do gráfico é proporcional ao valor que ele possui. Isso é feito distribuindo-se cores para as faixas de valores assumidos pela função.

Exemplo 66: *Superfície 2D*

```
1 >> surface( x, y, z )
```

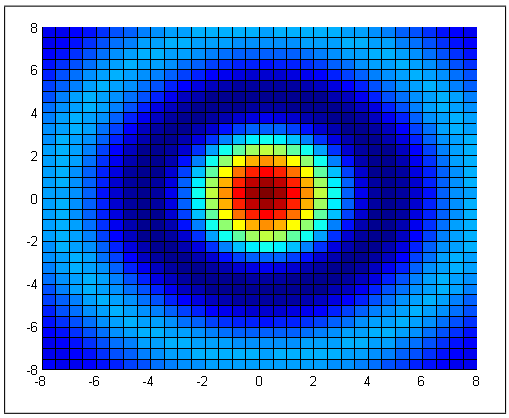

Figura 15: **Gráfico gerado no exemplo 66 - Matlab**

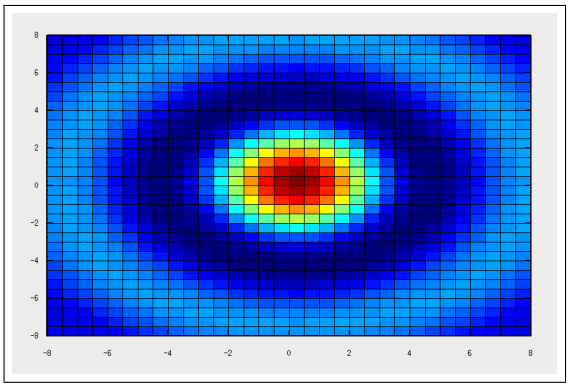

Figura 15: **Gráfico gerado no exemplo 66 - Matlab**

8 Fluxo de Controle

O MATLAB possui comandos de programação parecidos com os encontrados na maioria das linguagens de programação, tais como C, C++ e JAVA.

8.1 Loop FOR

No comando **for**, a partir de uma condição inicial, enquanto não é atingida a condição de parada, uma seqüência de instruções deve ser executada. O exemplo 67 mostra o uso do comando **for**.

Exemplo 67: *Comando* ***for***

```
>> for i=1:5
v( i ) = i;
w( i ) = 2*v( i );
end
```

No exemplo anterior, o vetor v obtido é [1, 2, 3, 4, 5] e w é igual a 2*v.

Uma importante observar que se deve sempre finalizar um loop **for** com um comando **end**. Pode-se utilizar mais de um loop dentro de um loop, mas deve-se lembrar que cada **for** deve ter seu próprio **end**.

Pode-se, também, fazer com que o incremento seja um valor diferente do unitário, conforme o próximo exemplo.

Exemplo 68: *Comando* ***for*** *com decremento*

```
>> for i=5:-1:1
v( i ) = i;
end
```

No exemplo 68 há um decremento unitário no valor de i. O vetor v obtido é o mesmo do exemplo 67.

8.2 Loop WHILE

O loop **while** permite que uma ou mais linhas de comando sejam executadas um número indefinido de vezes, através do controle de uma condição lógica. Veja o exemplo 69.

Exemplo 69: *Comando* ***while***

```
>> x = 1;
>> y = 5;
>> while x<3
y = y*x;
```

```
x = x+1;
end
```

No exemplo 69, as linhas de comando são executadas repetidamente até que x seja igual a 3. Assim, o valor final obtido para y é 10.

8.3 Comando BREAK

O comando **break** serve para interromper a execução de um loop, tal como um **for** ou um **while**.

8.4 Comando IF

No comando **if**, se uma expressão lógica é satisfeita uma seqüência de comandos é executada, caso contrário uma outra seqüência será executada.

O exemplo abaixo ilustra a utilização dos comandos **if** e **break**.

Exemplo 70: *Comandos **if** e **break***

```
>> v = [1 6 3 -2 5];
>> i = 1;
>> while 1
if v(i)<0 break, else v(i), end
i = i+1;
end

ans =

     1


ans =

     6


ans =

     3
```

No exemplo anterior, são impressos os elementos do vetor v até que seja encontrado um elemento negativo.

9 Arquivos-M: Scripts e Função

O MATLAB e o Octave tambêm podem executar uma seqüência de comandos que está armazenada em arquivos. Tais arquivos de disco são chamados arquivos-M em virtude de sua extensão ser do tipo .m. Um arquivo-M consiste de uma seqüência normal de linhas de comando do MATLAB/Octave, as quais podem conter chamadas ao próprio ou outros arquivos .m.

Existem dois tipos de arquivo .m: scripts e funções. Os scripts são arquivos contendo seqüência de comandos, enquanto que os arquivos de função permitem criar novas funções para serem utilizadas futuramente.

9.1 Arquivo Script

Quando um script é chamado, MATLAB simplesmente executa os comandos encontrados no arquivo. Os scripts são úteis quando a análise de um problema exige longas seqüências de comando, o que é cansativo para ser feito interativamente. Considere o arquivo seno.m, que contém os comandos para gerar o gráfico seno(x).

Exemplo 71: *Arquivo-M para gerar o gráfico seno(x)*

```
x = 0:pi/8:2*pi;
y = sin( x );
plot( x , y , 'r:+' )
title( 'seno' )
xlabel( 'eixo x' )
ylabel( 'eixo y' )
grid on
pause
```

Na última linha existe o comando pause. Ele é responsável por parar a execução dos comandos até que alguma tecla seja pressionada.

Existem duas maneiras de executar um arquivo .m. A primeira é, já no Octave / Matlab, escrever o nome do arquivo, sem necessidade de escrever a extensão .m, porém, é necessário que você já esteja na pasta onde se encontra o script a ser executado.

No exemplo 71, ao ser digitado *seno* na linha de comando, será gerado o gráfico seno(x), que pode ser visto na figura 16, e as variáveis x e y ficam mantidas no espaço de trabalho.

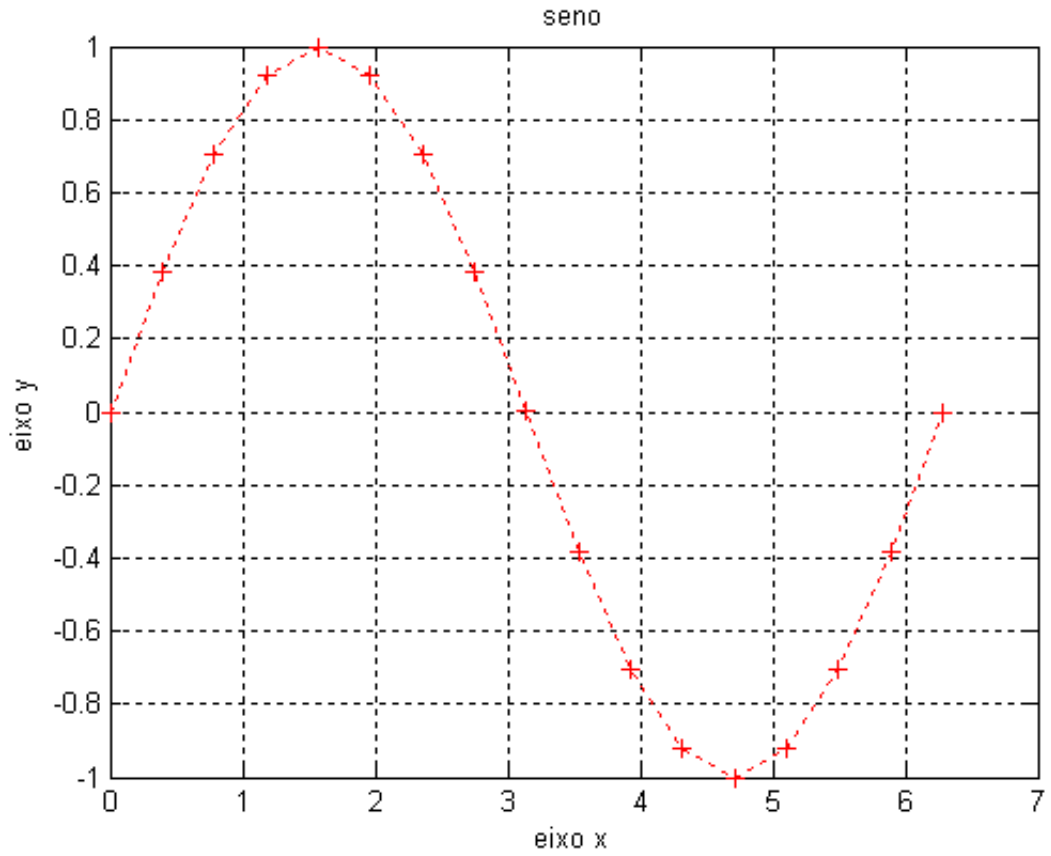

Figura 16: **Gráfico gerado pelo exemplo 71**

A segunda maneira de executar um arquivo .m é passando-o como parâmetro ao abrir o octave. Por exemplo, digitando-se *octave seno.m*, o octave será aberto é já executará o script seno.m. Note que se não houvesse o comando *pause* na última linha do script, o octave seria aberto, as instruções do script seriam executadas e logo em seguida o octave seria fechado, juntamente com o gráfico gerado, não dando tempo para ver os resultados.

Para fazer comentários em um script, coloque "%" no começo da linha. No exemplo 72, os textos "%gera o grafico de seno de x", "% O titulo do grafico é seno" e "% usado para conseguir visualizar o gráfico" são ignorados pelo interpretador.

Exemplo 72: *Arquivo-M comentado*

```
1 x = 0:pi/8:2*pi;
2 y = sin( x );
3 %gera o grafico de seno de x
4 plot( x , y , 'r:+' )
```

```
title( 'seno' ) % O titulo do grafico   seno
xlabel( 'eixo x' )
ylabel( 'eixo y' )
grid on
pause % usado para conseguir visualizar o gr fico
```

Outros dois comandos muito usados em scripts são o *echo* e o *disp*.

O echo serve para definir se as linhas digitadas no script devem ser exibidas ou não. No Exemplo 73, o echo não é usado, fazendo com que somente a matriz C seja exibida na tela, por não conter o ponto e vírgula no final do comando de multiplicação.

Exemplo 73: *Arquivo-M*

```
%matriz A
A = [1 2 3; 4 5 6; 7 8 9];
%matriz B
B = [1 4 7; 2 5 8; 3 6 9];
%multiplica   o de A por B
C = A*B
```

Mas, colocando o comando “echo on” no início do código, a saída seria:

Exemplo 74: *Saída do Exemplo 73*

```
+ %matriz A
+ A = [1 2 3; 4 5 6; 7 8 9];
+ %matriz B
+ B = [1 4 7; 2 5 8; 3 6 9];
+ %multiplica   o de A por B
+ C = A*B
C =


    14    32    50
    32    77   122
    50   122   194
```

Note que até mesmo os comentários são impressos. O comando *echo on* continua ativo até que o comando *echo off* seja seja executado. Assim, executando o script mostrado no Exemplo 75, a saída seria a mostrada no Exemplo 76.

Exemplo 75: *Usando o echo on e o echo off*

```
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fun   o para calcular as raizes de uma equacao
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y = [1 2 5 8];
roots (y)

echo off

A = sqrt (5820);
```

Exemplo 76: *Saída da execução do script do Exemplo 75*

```
+ echo on
+ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
+ %Fun  o para calcular as raizes de uma equacao
+ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
+ y = [1 2 5 8];
+ roots (y)
ans =

  -0.12391 + 2.13317i
  -0.12391 - 2.13317i
  -1.75217 + 0.00000i


+
+ echo off
```

Como o comando *A = sqrt (5820);* está após o echo off e possui um ponto e vírgula no final da linha, ele não foi exibido quando executou-se o script.

O comando disp ('texto') serve para imprimir na tela a string que é passada como parâmetro. Também podemos passar como parâmetro alguma variável ou constante usada no programa. No Exemplo 77 é mostrado o uso do disp e podemos ver sua saída no Exemplo 78.

Exemplo 77: *Usando o disp*

```
disp ('O Valor de pi  : ');
disp (pi)

disp ('Aperte alguma tecla para continuar ...')
pause
x = [0:0.01:10];
y = cos(x);
```

Exemplo 78: *Saída da execução do script do Exemplo 77*

```
octave:1> script
O Valor de pi    :
 3.1416
Aperte alguma tecla para continuar ...
octave:2>
```

9.2 Arquivo Função

Um arquivo-M que contém a palavra "function"no início da primeira linha será interpretado como um arquivo função. Uma função difere de um script pelos argumentos que devem ser passados e pelas variáveis que são definidas e manipuladas, que são locais à função e que não podem ser operadas globalmente no espaço de trabalho.

Como exemplo de uma função criada no MATLAB, considere o arquivo shift.m, que desloca os elementos de um vetor uma casa à direita.

Exemplo 79: *Arquivo-M que cria a função shift*

```
function novo_vetor = shift( velho_vetor )
% shift desloca os elementos de um vetor uma casa a direita.
% shift retorna o novo vetor.
% velho_vetor eh um vetor linha .
N = length( velho_vetor );
for i=1:N-1
    novo_vetor( i+1 ) = velho_vetor ( i );
end
novo_vetor ( 1 ) = 0 ;
```

Alguns comentários sobre arquivos do tipo função:

- A primeira linha declara o nome da função e os parâmetros de entrada e a variável de saída.
- É necessário que o arquivo .m tenha o mesmo nome da função criada.
- O símbolo % indica que o restante de uma linha é um comentário e deve ser ignorado.
- As primeiras linhas comentadas descrevem o arquivo-M e são mostradas quando você digita help <nome da função>. Criar tais comentários é opcional.
- As variáveis criadas dentro do escopo da função não aparecem no espaço de trabalho.

A existência do arquivo do exemplo 79 define uma nova função chamada shift, que é usada como qualquer outra função do MATLAB. Veja o próximo exemplo.

Exemplo 80: *Uso da função shift*

```
>> v = [1 7 4];
>> v = shift( v )

v =

     0     1     7
```