
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	4
1.1	Matlab	4
1.2	Octave	4
2	Operações simples entre escalares	5
2.1	Soma	5
2.2	Subtração	5
2.3	Multiplicação	5
2.4	Divisão Direta	6
2.5	Divisão indireta	6
2.6	Exponenciação	6
3	Representação de matrizes e vetores no MATLAB e Octave	7
4	Operações simples entre matrizes	10
4.1	Soma e subtração	10
4.2	Multiplicação de uma matriz por um escalar	10
4.3	Multiplicação entre matrizes	11
4.4	Divisão direta de matrizes	11
4.5	Divisão indireta de matrizes	12
4.6	Exponenciação A^b com b sendo um escalar	12
4.7	Operação elemento por elemento	12
4.8	Transposta de uma matriz	13
5	Raízes reais	13
6	Resolução de sistemas lineares	14
7	Interpolação polinomial	17
7.1	Interpolação pela resolução de um sistema linear	17
7.2	Método direto	18
8	Gráficos em MATLAB e Octave	19
8.1	Gráficos 2-D	19
8.1.1	Coordenadas Cartesianas	19
8.1.2	Coordenadas Polares	25
8.2	Gráficos 3-D	27

9	Fluxo de Controle	33
9.1	Loop FOR	33
9.2	Loop WHILE	33
9.3	Comando BREAK	34
9.4	Comando IF	34
10	Arquivos-M: Scripts e Função	35
10.1	Arquivo Script	35
10.2	Arquivo Função	39

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 terminal 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*

```
1 >> 3 + 5
2
3 ans =
4
5      8
```

2.2 Subtração

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

Exemplo 2: *Subtração entre dois escalares*

```
1 >> 4 - 8
2
3 ans =
4
5     -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*

```
1 >> 7 * 2
2
3 ans =
4
5     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*

```
1 >> 18 / 5
2
3 ans =
4
5      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*

```
1 >> 10 \ 5
2
3 ans =
4
5      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*

```
1 >> 9 ^ 3
2
3 ans =
4
5      729
```

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 7 mostra formas de representação de uma matriz 3x3.

Exemplo 7: *Formas de declaração de matrizes*

```
1 >> A = [1 2 3; 4 5 6; 7 8 9];
2 >> A
3
4 A =
5
6     1     2     3
7     4     5     6
8     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 8, tem-se um vetor-linha de dimensão 3, ou seja uma de uma matriz 1x3:

Exemplo 8: *Um vetor linha*

```
1 >> v = [1 2 3];
2 >> v
3
4 v =
5
6     1     2     3
```

O exemplo 9 é de um vetor-coluna de dimensão 3, ou matriz 3x1.

Exemplo 9: *Um vetor coluna*

```
1 >> v = [1; 2; 3];
2 >> v
3
4 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 10: *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 11, mostra o que acontece quando é acessada a posição 5 do vetor do exemplo anterior.

Exemplo 11: *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 12, a matriz z é composta de três elementos, sendo que um deles é uma matriz $1 \times n$, ou vetor linha.

Exemplo 12: *Criando uma matriz com um vetor linha como elemento*

```
1 >> z = [4 5 v];  
2 >> z  
3  
4 z =  
5  
6      4      5      1      0      3      0      8
```

A matriz resultante é uma matriz com 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 13 mostra essas operações.

Exemplo 13: *Adição e subtração de entre duas matrizes 3x3*

```
1 >> A = [3 7 2; 3 11 9; 1 0 10];
2 >> B = [8 6 4; 0 1 0; 2 5 12];
3 >> A+B
4
5 ans =
6
7      11      13      6
8       3      12      9
9       3       5     22
10
11 >> A-B
12
13 ans =
14
15      -5       1      -2
16       3      10       9
17      -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 14: *Multiplicação de uma matriz por um escalar*

```
1 >> 5*A
2
3 ans =
4
5      15      35      10
6      15      55      45
```

7

5	0	50
---	---	----

4.3 Multiplicação entre matrizes

Na multiplicação entre duas matrizes, $A*B$, o elemento $i \times j$ 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 15.

Exemplo 15: *Multiplicação entre duas matrizes*

1
2
3
4
5
6
7

```
>> 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 16, tem-se a divisão direta entre duas matrizes.

Exemplo 16: *Divisão direta entre duas matrizes*

1
2
3
4
5
6
7
8
9
10

```
>> 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 ($A \setminus B$) 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 17: *Divisão indireta entre duas matrizes*

```
1 >> B\A
2 ans =
3
4     1.92000    0.44667   -0.38667
5    -2.88000   -0.25333    1.41333
6     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 18: *Operação de exponenciação*

```
1 >> A = [1 10 3; 0 9 6; 5 5 4];
2 >> A^2
3
4 ans =
5
6     16     115     75
7     30     111     78
8     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 [19](#).

Exemplo 19: *Multiplicação entre duas matrizes, elemento por elemento*

```
1 >> [7 4 9 6 2].*[8 3 5 1 3]
2
3 ans =
4
5     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 20, é encontrada a transposta da matriz A vista no tópico anterior.

Exemplo 20: *Transposta de uma matriz*

```
1 >> A'
2
3
4 ans =
5
6      1      0      5
7     10      9      5
8      3      6      4
```

5 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, cujos coeficientes das potências em ordem decrescente são os elementos do vetor v. No exemplo 21, as raízes do polinômio $x^2 + 3x - 4$ são -4 e 1.

Exemplo 21: *Raízes de polinômio*

```
1 >> roots( [1, 3, -4] )
2
3 ans =
4
5     -4
6      1
```

6 Resolução de sistemas lineares

Considere o sistema linear abaixo:

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

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

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Figura 1: Sistema $Ax = b$

Pode-se calcular a solução do sistema diretamente, usando os comandos $x = A \setminus b$ ou $x = \text{inv}(A) \cdot b$. 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 22: *Cálculo da inversa de uma matriz*

```
1 >> inv( A )
2
3 ans =
4
5     -1.7778     0.8889    -0.1111
6     1.5556    -0.7778     0.2222
7     -0.1111     0.2222    -0.1111
```

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

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

Exemplo 23: *Cálculo de autovalores de uma matriz*

```
1 >> eig( A )
2
3 ans =
4
5     12.1229
6     -0.3884
7     -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 24: *Cálculo de auto-vetores e autovalores de uma matriz*

```
1 >> [V, D] = eig( A )
2
3 V =
4
5     -0.2998     -0.7471     -0.2763
6     -0.7075      0.6582     -0.3884
7     -0.6400     -0.0931      0.8791
8
9 D =
10
11     12.1229         0         0
12         0     -0.3884         0
13         0         0     -5.7345
```

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

Exemplo 25: *Cálculo do determinante de uma matriz*

```
1 >> det( A )
2
3 ans =
4
5     27
```

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

Exemplo 26: *Troca de colunas de uma matriz*

```
1 >> fliplr( A )
2
3 ans =
4
5      3      2      1
6      6      5      4
7      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 27: *Decomposição LU de uma matriz*

```
1 >> [L, U, P] = lu( A )
2
3 L =
4
5      1.0000      0      0
6      0.1429      1.0000      0
7      0.5714      0.5000      1.0000
8
9
10 U =
11
12      7.0000      8.0000      0
13      0      0.8571      3.0000
14      0      0      4.5000
15
16
17 P =
18
19      0      0      1
20      1      0      0
21      0      1      0
```


7 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.

7.1 Interpolação pela resolução de um sistema linear

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

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

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 28 mostra como utilizar o comando.

Exemplo 28: *Transposta da matriz de Vandermonde espelhada*

```
1 >> 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 29: *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

```

7.2 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 30 o polinômio encontrado é $x^4 - 10x^3 + 35x^2 - 50x + 24$;

Exemplo 30: 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 31: Polinômio interpolador de grau n

```

1 >> polyfit( [ 1 , 2 , 3 ] , [ 0 , 6 , 14 ] , 2 )
2
3 ans =
4
5     1.0000     3.0000    -4.0000

```

O polinômio de grau 2 obtido no exemplo 31 é $x^2 + 3x - 4$.

8 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.

8.1 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.

8.1.1 Coordenadas Cartesianas

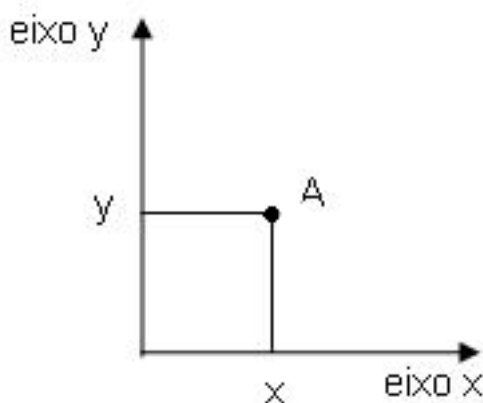


Figura 3: **Eixos cartesianos**

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 32, 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 32: *Dois gráficos em uma só figura*

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

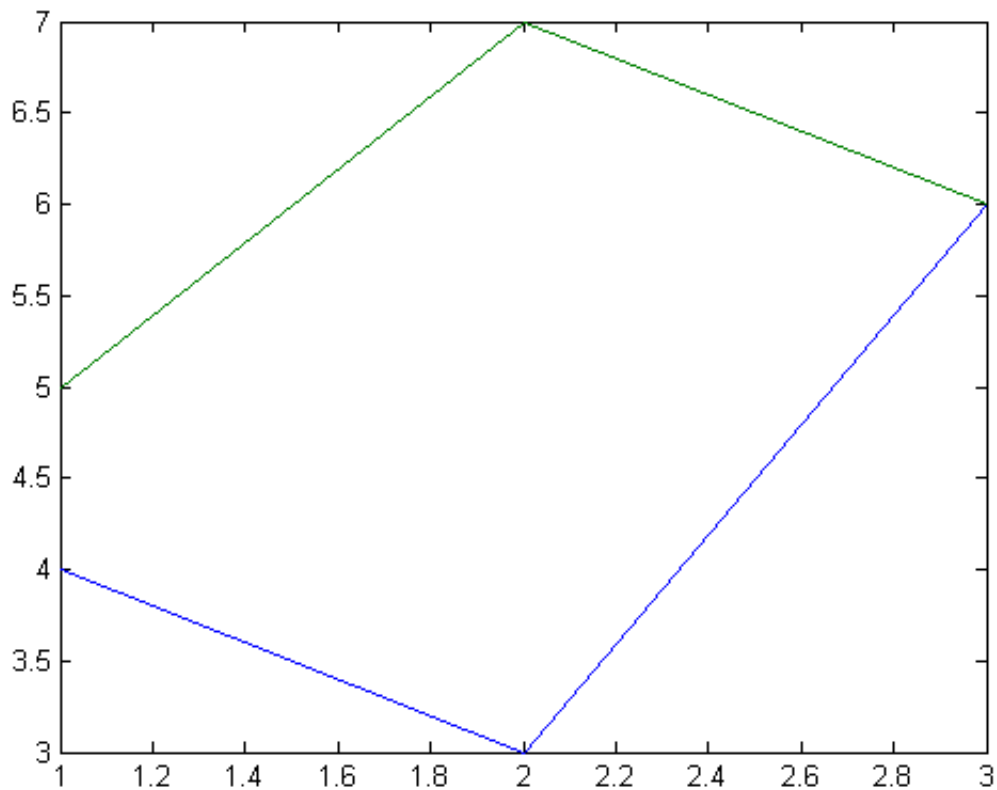


Figura 4: Gráfico gerado no exemplo 32

- **semilogx(x, y)**: Gera gráficos com y na escala linear e x na escala logarítmica. No exemplo 33 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 33: *Gráfico com x na escala logarítmica*

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

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

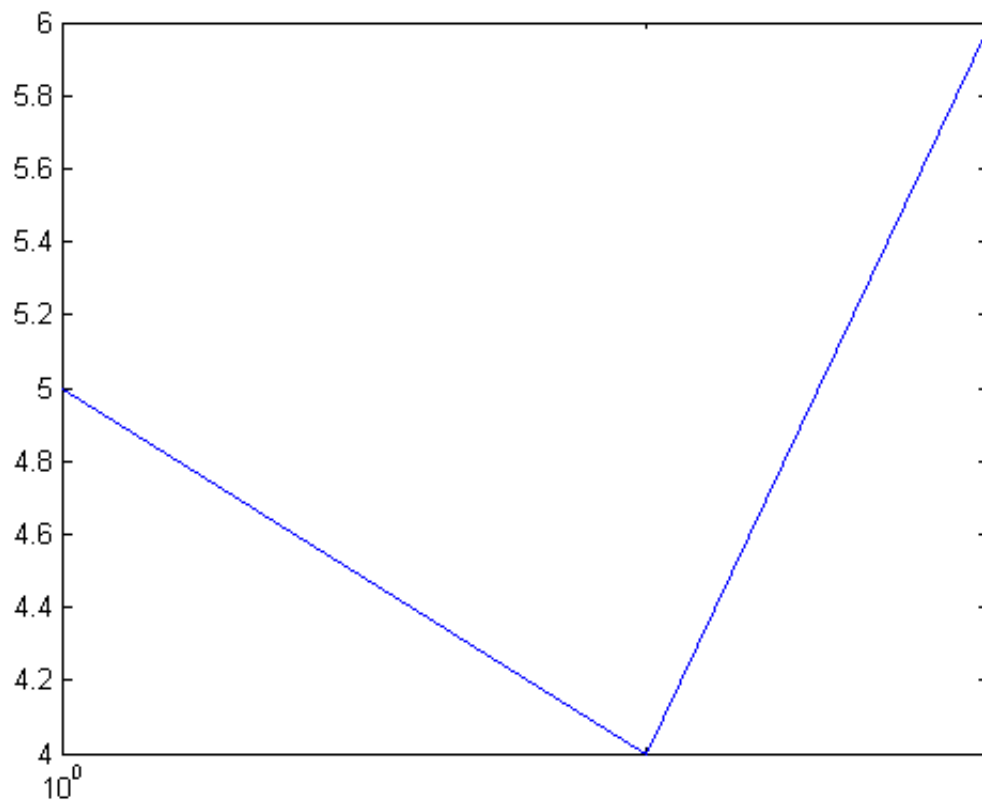


Figura 5: **Gráfico gerado no exemplo 33**

- **semilog(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 34 é o comando utilizado para gerar tal gráfico.

Exemplo 34: *Gráfico com x e y na escala logarítmica*

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

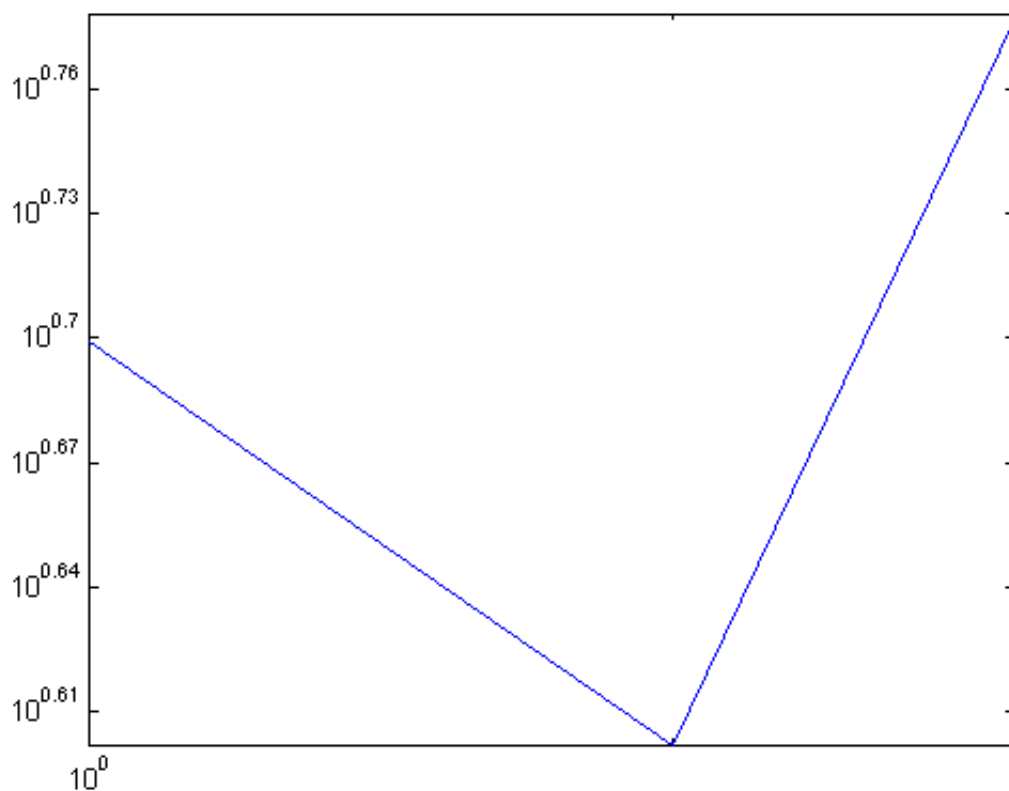


Figura 6: Gráfico gerado no exemplo 34

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 y;
- **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 exhibe 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	pontuada	:	cruz	+
vermelho	r			X	X
preto	k			estrela	*
roxo	m			triângulo	^
verde	g			triângulo invertido	V

³Para 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 pontuada 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 35: *Gráficos do seno e do cosseno em uma mesma figura*

```

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

```

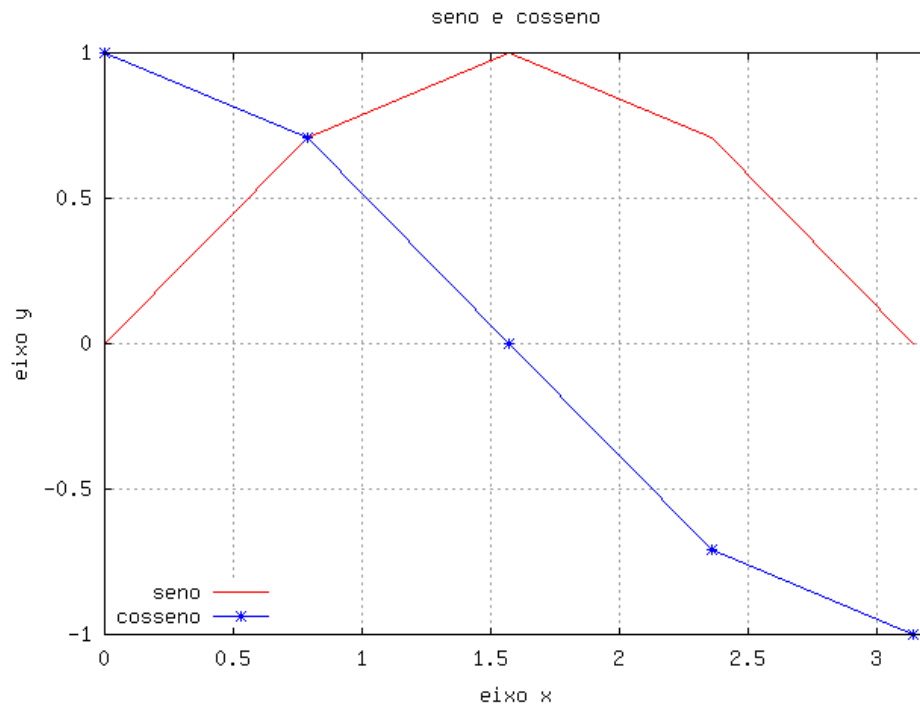



Figura 7: Gráfico gerado no exemplo 35 - Octave

8.1.2 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) .

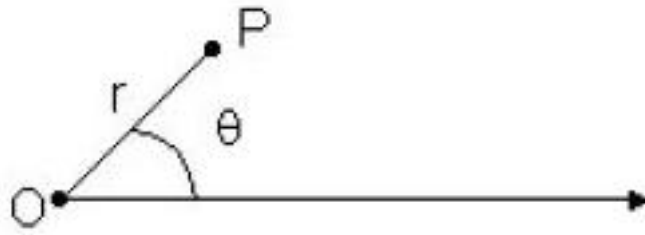


Figura 8: Coordenadas polares no Matlab

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 36, 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 36: *Gráfico em coordenadas polares*

```

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

```

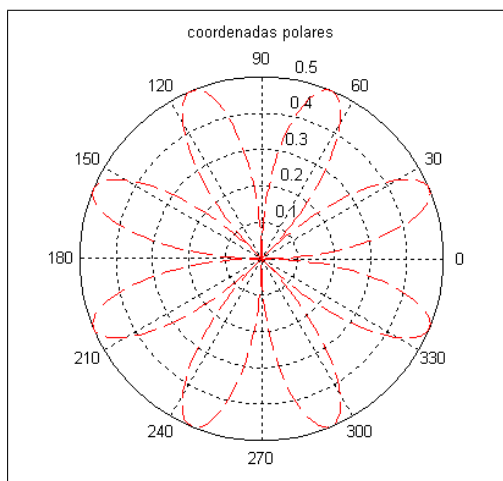


Figura 9: Gráfico gerado no exemplo 36 - Matlab

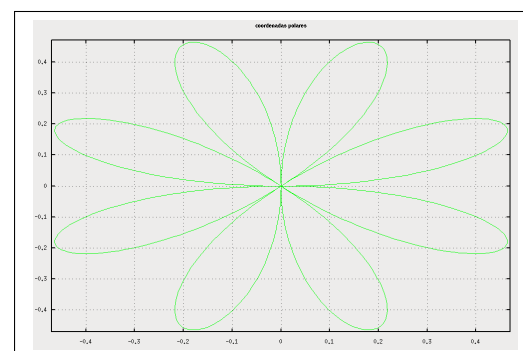


Figura 9: Gráfico gerado no exemplo 36 - Octave

8.2 Gráficos 3-D

- **plot3(x, y, z)**: Plota gráficos no espaço 3D. Observe, no exemplo 37, o comando para gerar o gráfico $(x, y, z) = (\sin(t), \cos(t), t)$. A figura 10 é o gráfico obtido.

Exemplo 37: Gráfico 3-D

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

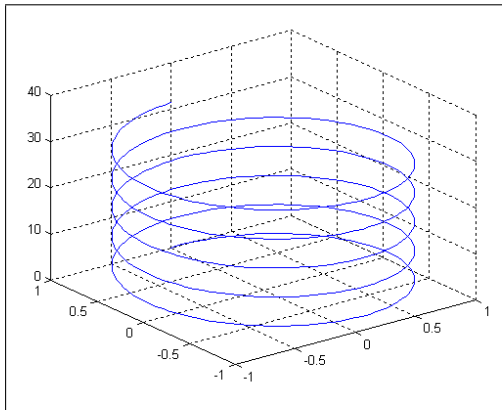


Figura 10: Gráfico gerado no exemplo 37 - Matlab

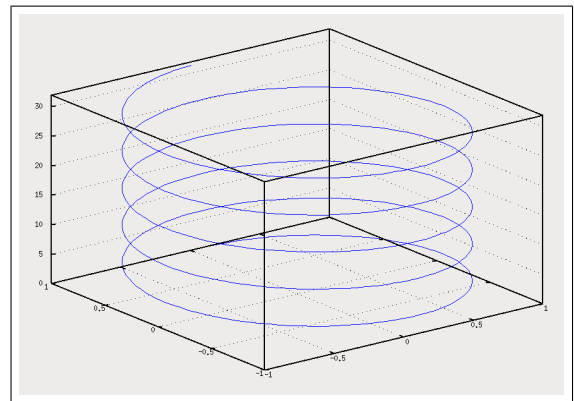


Figura 10: Gráfico gerado no exemplo 37 - Octave

- **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 38 gera o gráfico referente à função $f(x,y) = \frac{\sin(x^2+y^2)^{\frac{1}{2}}}{(x^2+y^2)^{\frac{1}{2}}}$.

O comando **eps** no exemplo 38 representa a precisão da máquina e evita uma divisão por zero na linha 3.

Exemplo 38: 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.

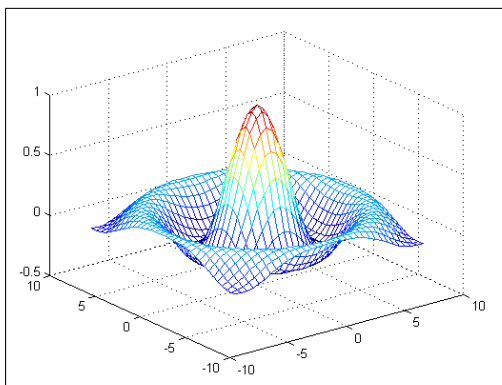


Figura 11: Gráfico gerado no exemplo 38 - Matlab

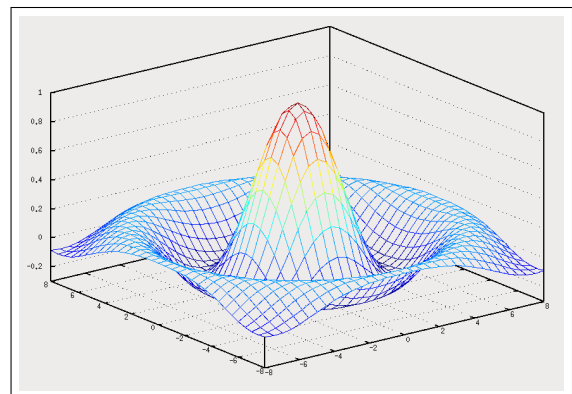


Figura 11: Gráfico gerado no exemplo 38 - Octave

- **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 39 gera a figura 12, que é o contorno em 2D da função do exemplo anterior.

Exemplo 39: *Contorno em 2D*

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

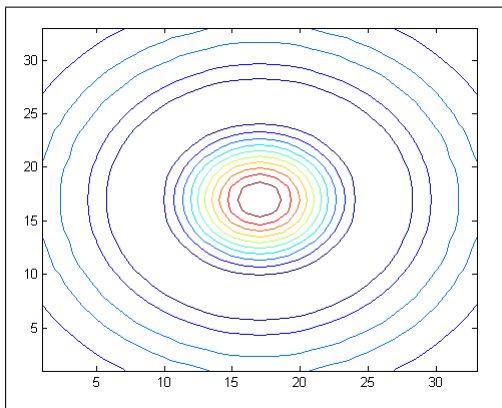


Figura 12: Gráfico gerado no exemplo 39 - Matlab

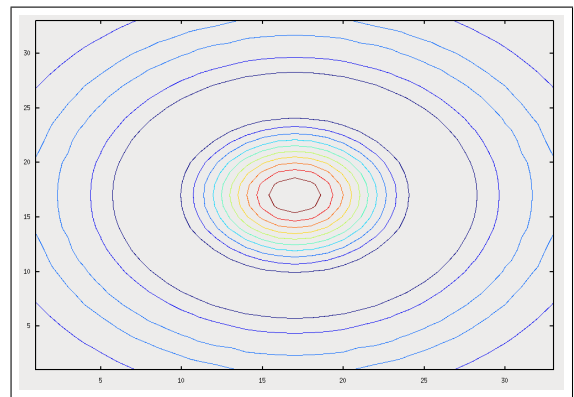


Figura 12: Gráfico gerado no exemplo 39 - Octave

- **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 38. O exemplo 40 é o comando pra gerar tal contorno.

Exemplo 40: Contorno em 3D

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

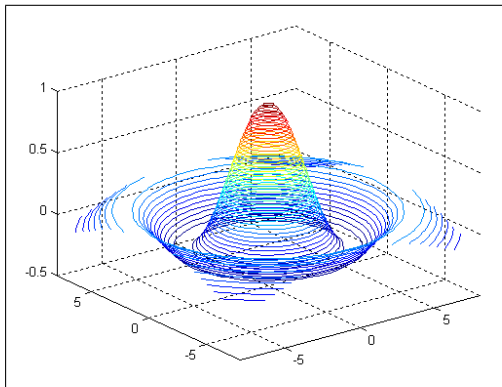


Figura 13: Gráfico gerado no exemplo 40 - Matlab

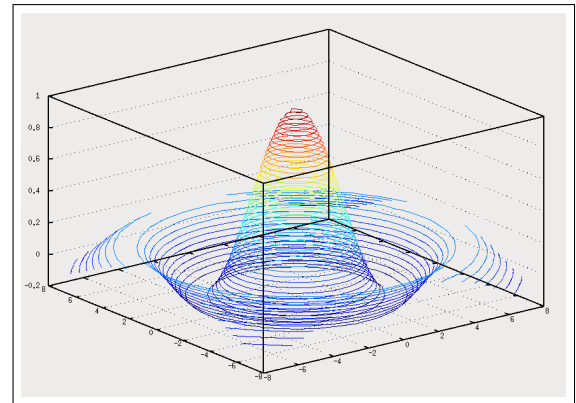


Figura 13: Gráfico gerado no exemplo 40 - Octave

- **surf(x, y, z)**: Plota superfície 3D. A superfície 3D da função do exemplo 38 é gerada pelo comando do exemplo 41. O gráfico obtido está na figura 14.

Exemplo 41: Superfície 3D

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

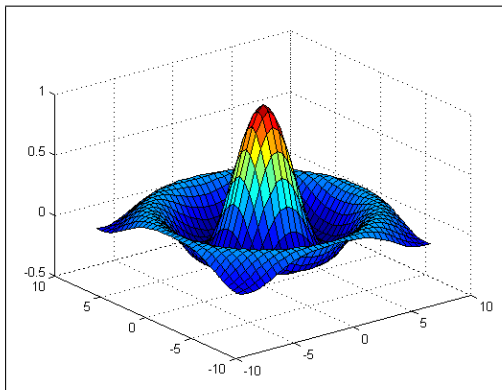


Figura 14: Gráfico gerado no exemplo 41 - Matlab

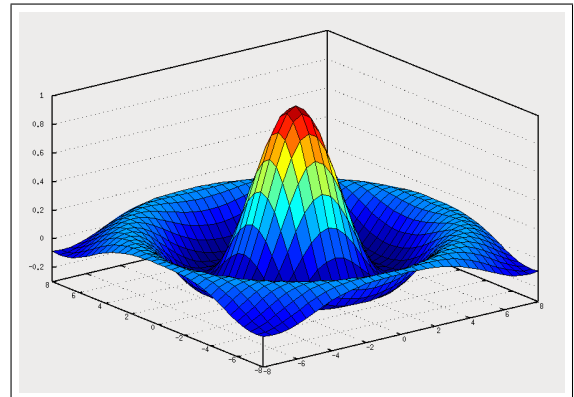


Figura 14: Gráfico gerado no exemplo 41 - Octave

- **surface(x, y, z)**: Plota a superfície de contorno 2D de um gráfico 3D. O exemplo 42 gera a figura 15, que é o contorno 2D do gráfico 3D da função do exemplo 38. 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 42: Superfície 2D

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

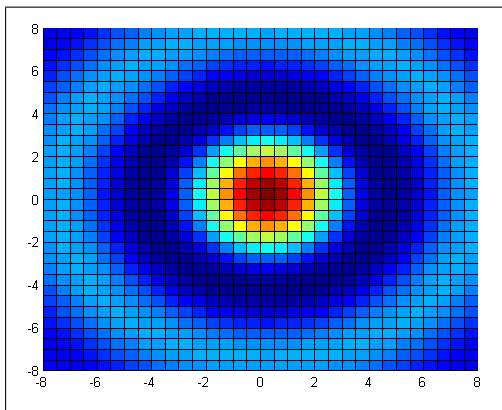


Figura 15: Gráfico gerado no exemplo 42 - Matlab

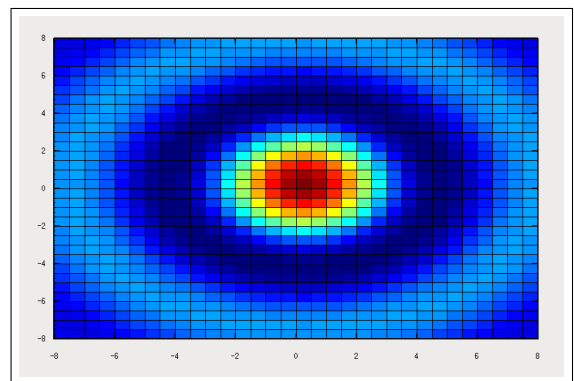


Figura 15: Gráfico gerado no exemplo 42 - Matlab

9 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.

9.1 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 43 mostra o uso do comando **for**.

Exemplo 43: *Comando **for***

```
1 >> for i=1:5
2 v( i ) = i ;
3 w( i ) = 2*v( i );
4 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 44: *Comando **for** com decremento*

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

No exemplo 44 há um decremento unitário no valor de *i*. O vetor *v* obtido é o mesmo do exemplo 43.

9.2 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 45.

Exemplo 45: *Comando **while***

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

```
5 x = x+1;  
6 end
```

No exemplo 45, as linhas de comando são executadas repetidamente até que x seja igual a 3. Assim, o valor final obtido para y é 10.

9.3 Comando BREAK

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

9.4 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 46: Comandos *if* e *break*

```
1 >> v = [1 6 3 -2 5];  
2 >> i = 1;  
3 >> while 1  
4   if v( i ) < 0 break, else v( i ), end  
5   i = i + 1;  
6 end  
7  
8 ans =  
9  
10      1  
11  
12  
13 ans =  
14  
15      6  
16  
17  
18 ans =  
19  
20      3
```

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

10 Arquivos-M: Scripts e Função

O MATLAB e o Octave também 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.

10.1 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 47: *Arquivo-M para gerar o gráfico `seno(x)`*

```
1 x = 0:pi/8:2*pi;
2 y = sin( x );
3 plot( x , y , 'r:+' )
4 title( 'seno' )
5 xlabel( 'eixo x' )
6 ylabel( 'eixo y' )
7 grid on
8 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 47, 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.

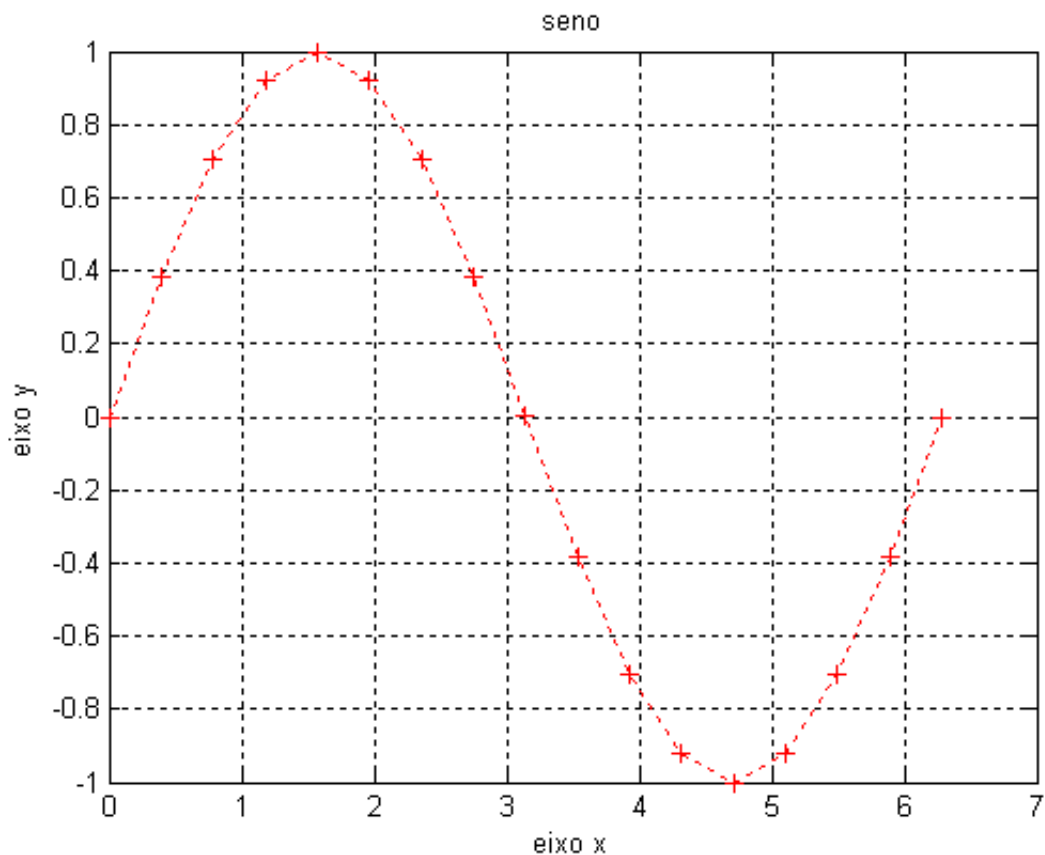


Figura 16: Gráfico gerado pelo exemplo 47

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 e 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 48, 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 48: 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:+' )

```

```

5 title( 'seno' ) % O titulo do grafico é seno
6 xlabel( 'eixo x' )
7 ylabel( 'eixo y' )
8 grid on
9 pause % usado para conseguir visualizar o gráfico

```

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 49, 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 49: Arquivo-M

```

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

```

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

Exemplo 50: Saída do Exemplo 49

```

1 + %matriz A
2 + A = [1 2 3; 4 5 6; 7 8 9];
3 + %matriz B
4 + B = [1 4 7; 2 5 8; 3 6 9];
5 + %multiplicação de A por B
6 + C = A*B
7 C =
8
9      14      32      50
10     32      77     122
11     50     122     194

```

Note que até mesmo os comentários são impressos. O comando *echo on* continua ativo até que o comando *echo off* seja executado. Assim, executando o script mostrado no Exemplo 51, a saída seria a mostrada no Exemplo 52.

Exemplo 51: Usando o echo on e o echo off

```

1 echo on
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %Função para calcular as raizes de uma equacao

```

```

4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 y = [1 2 5 8];
6 roots (y)
7
8 echo off
9
10 A = sqrt (5820);

```

Exemplo 52: Saída da execução do script do Exemplo 51

```

1 + echo on
2 + %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 + %Função para calcular as raizes de uma equacao
4 + %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 + y = [1 2 5 8];
6 + roots (y)
7 ans =
8
9     -0.12391 + 2.13317 i
10    -0.12391 - 2.13317 i
11    -1.75217 + 0.00000 i
12
13 +
14 + echo off

```

Como o comando $A = \text{sqrt}(5820)$; 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 53 é mostrado o uso do `disp` e podemos ver sua saída no Exemplo 54.

Exemplo 53: Usando o `disp`

```

1 disp('O Valor de pi é: ');
2 disp(pi)
3
4 disp('Aperte alguma tecla para continuar ...')
5 pause
6 x = [0:0.01:10];
7 y = cos(x);

```

Exemplo 54: *Saída da execução do script do Exemplo 53*

```
1 octave:1> script
2 O Valor de pi é:
3 3.1416
4 Aperte alguma tecla para continuar ...
5 octave:2>
```

10.2 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 55: *Arquivo-M que cria a função shift*

```
1 function novo_vetor = shift( velho_vetor )
2 % shift desloca os elementos de um vetor uma casa a direita.
3 % shift retorna o novo vetor.
4 % velho_vetor eh um vetor linha .
5 N = length( velho_vetor );
6 for i=1:N-1
7     novo_vetor( i+1 ) = velho_vetor ( i );
8 end
9 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 55 define uma nova função chamada shift, que é usada como qualquer outra função do MATLAB. Veja o próximo exemplo.

Exemplo 56: *Uso da função shift*

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