.

# APOSTILA DE MATLAB

2009

## 1 - INTRODUÇÃO

### 1.1 – Conceitos, comandos e simbologia

Todos os arquivos com extensão **m** são executáveis no MATLAB, os comandos podem ser editados no editor deste software(de preferência) ou em outro editor de texto como o editor **edit** do **DOS**.

**M-file editor:** editor de documentos do MATLAB, onde você pode digitar programas salvando-os em arquivos com extensão **m**(tornando estes arquivos executáveis). Para acionar o editor de textos dentro do próprio Matlab, acione sequencialmente, as opções do menu**:**

**File**
**New**
**M-File**

Dentro do MATLAB você pode usar comandos semelhantes ao do DOS ou interagir com eles, a seguir vem alguns deles**:**

**addpath dir1:** faz com que você possa, posteriormente, executar ou carregar um arquivo gravado no diretório dir1.
**cd:** mostra o corrente diretório que você está trabalhando.
**delete( nome arquivo):** apaga o nome do arquivo especificado.
**Workspace:** janela que se abre para mostrar o nome e características das variáveis usadas no MATLAB num determinado instante. Por exemplo, quando se usa os comandos **who** ou **whos** esta janela é acionada. Também pode ser usado o ícone ( semelhante a um cubo) do Workspace browser para executar essa tarefa.
**Comand history:** mostra os últimos comandos digitados.
**Path browser:** muda o path (o caminho), onde são lidos os arquivos no MATLAB .
**Palavras em negrito:** representa termos e fatos importantes.
**Palavras em negrito itálico:** novos termos.
**Palavras em negrito na inicial dos nomes:** nomes de teclas, menus, itens de menus, nome de funções , arquivos e comandos.
**Palavras em itálico:** títulos de livros, empresas, etc.

### 1.2 - Comandos, Variáveis e Símbolos Especiais

**Comandos no Matlab:** são editados em letras minúsculas
**Nome de variáveis:** o nome de uma variável pode ser definido como uma letra ou um conjunto de caracteres, havendo o caso sensitivo, isto é, uma variável em letra minúscula é diferente daquela mesma em letra maiúscula Se for usar mais de uma palavra para representar uma variável, deve ser usado o sinal de sublinhado para ligar os nomes que representarão a variável.
Exemplo de nomes de variáveis**:** a, A, preço1, val3, custo_médio, preço_de_venda, etc.
**; :** suprime a impressão de resultados.
**% :** serve para se colocar um comentário.
**... :**indica que uma linha é continuação de uma linha anterior.
**ans:** variável usada para assumir o resultado referente ao último comando.
**inf:** infinito(exemplo**:** digite 1/0).
**NaN:** indeterminação(exemplo**:** digite 0/0).
**ctrl c:** interrompe comandos do MATLAB.
**! :** seguida de um comando do DOS, acionará este comando digitado.
**clc :** limpa a tela(semelhante ao comando CLS do DOS).

**1.3 Constantes**
**pi:** 3,1416...
**eps:** $2{,}22 \times 10^{-14}$
**i** = $\sqrt{-1}$
**j** = $\sqrt{-1}$

**2 - OPERAÇÕES ARITMÉTICAS BÁSICAS**

| **Símbolo** | **Operação** | |
|---|---|---|
| + | soma | exemplo**:** 5 + 3 |
| - | subtração | exemplo**:** 6 – 2 |
| multiplicação normal de matrizes | | exemplo**:** 6 * 5 |
| / | divisão normal de matrizes | |
| **.*** | multiplicação elemento a elemento | |
| **./** | divisão elemento a elemento de maneira semelhante a multiplicação. | |
| \ | divisão à esquerda(exemplo**:** 5\25 tem o mesmo efeito que 25/5 que resulta**:** ans = 5) | |

**3 - FUNÇÕES MATEMÁTICAS COMUNS(OBS: ALGUM EXEMPLOS)**

| **Função** | **Descrição** |
|---|---|
| **abs(x)** | módulo de x |
| **acos(x)** | arco cujo coseno é x |
| **cos(x)** | coseno de x ( x em radianos) |
| **cosh(x)** | coseno hiperbólico de x |
| **exp(x)** | exponencial **:** $e^{x}$ |
| **gcd(x, y)** | MDC dos inteiros x e y |
| **imag(x)** | parte imaginária de um complexo |
| **lcm(x, y)** | MMC dos inteiros x e y |
| **log(x)** | logaritmo natural de x |
| **$\log_{10}$(x)** | logaritmo de x na base 10 |
| **real(x)** | parte real de x |
| **round (x)** | arredonda o valor de x |
| **sin(x)** | seno de x (em radianos) |
| **sinh(x)** | seno hiperbólico de x |
| **sqrt(x)** | raiz quadrada de x |
| **tan(x)** | tangente de x |

**4 - COMANDOS CARACTERÍSTICOS DE JANELA**
**4.1 – Comandos do workspace**
**who :** exibe o nome das variáveis usadas.
**whos:** exibe na tela os nomes, dimensão, número de bytes e tipos das variáveis que estão sendo usadas no momento.
**what:** exibe arquivos de extensão **.m** e **.mat** do diretório corrente.
**disp n:** exibe o conteúdo da variável **n** sem mostrar seu nome.
**clear n:** apaga a variável n.
**clear:** apaga todas as variáveis.

**[x, y] = ginput(n):** permite que você entre com os pares ordenados (x, y) por meio do mouse, escolha com o mouse cada ponto na janela que aparecerá na tela, sendo que **n** representa o número de pontos a serem determinados.
**input ('...'):** é usado para entrada de dados
exemplo**:** n = input ('digite um número ➔')
O número que você digitar será assumido pela variável **n**.
**pause:** serve para pausar um comando de exibição do Matlab.
**pause(n):** pausa por **n** segundos.
**echo on:** na execução de um programa executável, gravado num arquivo **.m**, o uso desta opção permite a exibição, na tela, dos comandos usados no seu programa, para desativar este opção usa-se o comando **echo off**.
**break:** termina a execução de um *loop* **while** ou **for**.

### 4.2 – Comandos relacionados a arquivos e dados

**save:** salva todas as variáveis que estão sendo usadas naquele momento, no formato binário num arquivo com extensão mat no MATLAB.
**save data:** salva todas as variáveis no arquivo data.mat.
**save data a, b, c:** salva as variáveis a, b, c no arquivo data.mat.
**load filename:** carrega as variáveis salvas com o comando **save** no arquivo especificado.
**file print:** use esses comandos do menu para imprimir o conteúdo atual da janela ativa
**file new M-file:** é usado para editar um arquivo com extensão **m** no editor do Matlab.

## 5 - Formatos de exibição de números

Supondo a = 1/3, quando usamos**:**
**format short:** resultará**:** a = 0,3333,ou seja, quatro casas decimais(é o default).
**format short e:** teremos**:** a = 3.3333e-001, ou seja, quatro casas decimais e em notação exponencial(que significa 3.3333 x $10^{-1}$).
**format short g:** teremos**:** a = 0.33333, com cinco casas decimais.
**format long:** teremos a = 0.33333333333333, com catorze casas decimais.
**format long e:** teremos a = 3.333333333333333e-001, com catorze casas decimais mais o expoente .
**format hex:** teremos**:** a = 3fd5555555555555(formato hexadecimal)
**format bank:** teremos**:** a = 0.33, ou seja, dois dígitos decimais(no formato monetário).
**format:** volta ao formato normal que equivale ao **format short**.

## 6 - MATRIZES

### 6.1 - Entrada de dados

Para separar os elementos de uma dada matriz usa- se o espaço em branco ou então vírgulas e para mudar de linha usa- se "; " ou a tecla <ENTER> .

Exemplo**:**

a = [1 2 3;4 5 6;7 8 9]; <Enter> ou
a = [1,2,3;4,5,6;7,8,9]; <Enter> ou ainda,
a = [1 2 3 <Enter>
4 5 6 <Enter>
7 8 9];

**6.2 – Definição de Intervalos**

**var = [início: increm: fim];**

Define um arranjo cuja variável **var** assume os valores que variam do valor **início** até o valor **fim** com o incremento **increm**.

Exemplo**:**

c = [0**:**0.2**:**5]; <Enter>

A variável c assumirá valores que variam de 0 até 5, com o incremento 0,2 (0, 0.2, 0.4, . . . , 4.8, 5).

Quando é excluído o incremento, ele é assumido como 1**:**

d = 1**:**5 <Enter>

A variável "d" assumirá os valores**:** 1, 2, 3, 4 e 5.

**var** = **linspace (início, fim, numter )**

Define um arranjo cuja variável **var** assume os valores que variam do valor **início** até o valor **fim,** com um número determinado de termos especificado por **numter**.

Exemplo**:**

e = linspace (0, 2, 11); <Enter>

A variável e assumirá valores que variam de 0 até 2, tendo 11 termos neste intervalo(0, 0.1, 0,2, . . . ,0.9, 1).

**var = logspace (início, fim, numt Er )**

Do mesmo modo que **linspace**, só que os valores serão assumidos como potência de 10($10^{início}$, . . ., $10^{fim}$).

Exemplo**:**

f = logspace (0, 2, 11)

A variável f assumirá os valores respectivos de**:** $10^{0}$, $10^{0,1}$, . . ., $10^{2}$ , ou seja**:**

ans =

1.0 1.58 2.51 3.98 6.30 10.00 15.84

25.11 39.81 63.09 100.00

**6.3 – Operações com matrizes**

Vamos considerar como exemplo as matrizes abaixo**:**

$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. $b = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

**6.3.1 – Adição de matrizes**

c = a + b resulta $c = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$.

**6.3.2 – Multiplicação e divisão normal de matrizes**

d = a * b resulta $d = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1*5+2*7 & 1*6+2*8 \\ 3*5+4*7 & 3*6+4*8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$

Observação**:** A divisão se faz de maneira semelhante a multiplicação usando-se porém o símbolo / .

**6.3.3 – Multiplicação e divisão de matrizes elementos a elementos**

e = a. * b resulta $e = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .* \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1*5 & 2*6 \\ 3*7 & 4*8 \end{bmatrix} = \begin{bmatrix} 5 & 12 \\ 21 & 48 \end{bmatrix}$

Observação**:** A divisão se faz de maneira semelhante a multiplicação usando-se porém o símbolo **./** .

### 6.3.4 – Potenciação normal de matrizes

f = a ^ 2 resulta $f = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .* \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1*1+2*3 & 1*2+2*4 \\ 3*1+4*3 & 3*2+4*4 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 15 & 22 \end{bmatrix}$

### 6.3.5 – Potenciação de matrizes elemento a elemento

g = a .^2 resulta $g = \begin{bmatrix} 1*1 & 2*2 \\ 3*3 & 4*4 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix}$

### 6.3.6 - Orientação de matrizes

Você pode usar a operação de transposição para transformar um vetor linha num vetor coluna**:**

Exemplo**:**
a = [1 2 3];
b = a'
Cria o vetor coluna**:**
b = 1
2
3

### 6.4 - Localização de um elemento em uma matriz

Um elemento é acionado através do seu índice, se o arranjo é um vetor linha ou um vetor coluna, basta um índice; se for uma matriz M x N qualquer teremos que referenciar a linha e a coluna do elemento que queremos.

Exemplo**:** É dado o arranjo**:**
b = [7 2 3]; pede- se b(1) , resultando ans = 7
a = [1 2 3;4 5 6;7 8 9]; pede-se a(2,3) e o resultado será o elemento da linha 2 e coluna 3, ou seja ans = 6.

### 6.5 - Operações com conjuntos

Se A for [ 1 2 3 4 5 6 7] e B for dado por [ 1 3 5 8]
**union (A, B):** representa A $\cup$ B, resulta**:** ans = 1 2 3 4 5 6 7 8
**intersect (A,B):** que simboliza A $\cap$ B, resulta ans = 1 3 5
**setdiff (A,B):** representando A – B, resulta**:** ans = 2 4 6 7
**setxor(A,B):** simbolizando (A $\cup$ B) – ( A $\cap$ B ), resulta**:** ans = 2 4 6 7 8

## 7 - FUNÇÕES DE MATRIZES

**a'** **:** transposta da matriz a.
**det(a)** **:** determinante de uma matriz quadrada a.
**eig(a)** **:** autovalores(raízes da polinomial característica) de a.
**eye(n)** **:** matriz identidade de dimensão n.
**inv(a)** **:** matriz inversa da matriz a.

**length(a)** : número de linhas de a.
**magic(n)** : matriz quadrada de dimensão n.
**norm(a)** : determina a norma da matriz a.
**ones (n)** : matriz de uns de ordem n.
**poly(a)** : polinomial característica da matriz a.
**rand(m, n)** : determina uma matriz aleatória de dimensão m por n ,com valores uniformemente distribuídos entre 0 e 1.
**randn(n)** : determina uma matriz quadrada aleatória n por n com média 0 e variância 1, com valores uniformemente distribuídos entre 0 e 1.
**size(a)** : dá as dimensões da matriz a.
**trace(a)** : soma de elementos da diagonal da matriz a.
**zeros(n)** : matriz de zeros de ordem n.
**dot(a, b)** : determina o produto escalar de **a** por **b**(onde **a** e **b** contém os respectivos coeficientes das componentes i, j e k dos eixos ortogonais).
**sum(a.*b)** : também determina o produto escalar de **a** por **b**(onde **a** e **b** contém os respectivos coeficientes das componentes i, j e k dos eixos ortogonais).
**cross(a, b)** : determina o produto vetorial de **a** por **b**(onde **a** e **b** contém os respectivos coeficientes das componentes i, j e k dos eixos ortogonais).

## 8 - OPERAÇÕES RELACIONAIS E LÓGICAS

### 8.1 - Operadores relacionais

< : menor
<= : menor igual
> : maior
>= : maior igual
= = : igual
~ = : diferente

### 8.2 - Operadores lógicos

**&** : **e** lógico
**|** : **ou** lógico
**~** : **não** lógico

Ex**:** (a > b) & (b < c)

## 9 - DATA E HORA

O MATLAB oferece várias funções para manipular a hora e a data

### 9.1 - Tempo e data corrente

A função **clock** retorna a data e a hora corrente num arranjo.

Exemplo**:**
t = clock
t = 1998 10 21 09 08 39.934708,cuja ordem é ano, mês, dia, hora, minutos, segundos.
Já a função **date** especifica somente o dia /mês/ano.
Resultando por exemplo**:** 02-Aug-1999

**9.2 - Formato de conversão**

Para converter a data numérica em literais basta usar a função **datestr.**

Exemplo**:**

datestr(t)

t = 21-Out -1998 09**:**08**:**40

**9.3 - Função data**

O dia da semana pode ser encontrado usando **weekday**. O MATLAB usa a convenção onde domingo corresponde ao número 1 e sábado ao número 7.

Exemplo**:**

[d, w] = weekday('12/08/1999')

d = 4

w = wed

O MATLAB pode gerar um calendário de qualquer mês que você quiser, basta para isso usar o comando **calendar** .

**calendar:** exibe o mês atual.

**calendar(data)** ou **calendar(mês, ano):** exibe o mês e ano especificado num destes comandos.

Exemplos**:**

s = calendar (1994,12) exibe o mês de dezembro do ano de 1994.

calendar ( ' 7/17/94 ' ) exibe o mês com os respectivos dias da semana do ano de 1994.

**9.4 - Rótulo de plotagem**

Para o uso de datas em gráficos ( no eixo do x), usa-se o comando **datetick.**

Exemplo**:**

t = ( 1900**:**10**:**1990)

p = [75.995; 91.972; 105.711; 123.203; 131.669; 150.697; 179.323; 203.212; 226.505; 249.633];

plot (datenum (t, 1, 1), p)

**datetick('x ', 'yyyy '):** plota no eixo x quatro dígitos para ano.

**datetick('x ', ' mmmyy'):** plota no eixo x três dígitos para mês e dois para ano.

**10 - FLUXO DE CONTROLE**

**10.1 - For - loops**

Permite que um grupo de comandos seja repetido um número fixado de vezes.

**For var = início: fim**

**comandos ...**

**end**

Exemplo**:**

for a = 1**:** 5

y = 2* a

end

a = 1 2 3 4 5

y = 2 4 6 8 10

Para usar dois comandos for**:**

```
for m = 1: 4
for n =  1: 3
a(m, n) = 2;
end
end
disp(a)
```

O resultado será**:**

```
    2     2     2
    2     2     2
    2     2     2
    2     2     2
```

**10.2 - While - loops**

Avalia um grupo de comandos em um número fixado de vezes.

**while expressão**
**comandos ...**
**end**

Exemplo**:**

```
num = 0; a = 1
while (1+ a) < 6
a = a + 1;
num = num + 1;
end
a
```

O resultado será**:** a = 6

**10.3 - If - else – end construção**

```
If expressão
comandos ...
end
```

Exemplo**:**

```
a = input('digite o valor de a ==> ');
b = input('digite o valor de b ==> ');
if a > 5
   b = a + 1;
end
b
```

Neste caso, se o valor de **a** lido for menor que 5, o valor de **b** impresso será o mesmo lido; se o valor de **a** lido for maior que 5, o valor de **b** impresso será igual ao de **a** + 1.

ou então pode-se usar o **loop**, para o mesmo exemplo anterior, ou seja**:**

If expressão
comandos ...  ( se a expressão for verdadeira)
else
comandos ...  ( se a expressão for falsa)
end

Exemplo**:**

```
a = input('digite o valor de a ==> ');
b = input('digite o valor de b ==> ');
if a > 5
  b = a + 1;
else
b
end
```

**10.4 - Switch  case**

Quando se tem várias opções de escolha**:**

Exemplo(para transformar  polegadas ou pés em metros)**:**

```
x = 3.5;
unidade = 'm';
switch  unidade    % Converte x para centímetros
case {' inch',' in'}
y = x * 2.54;
case {'feet', 'ft' }
  y = x * 2.54 * 12;
case {'metro','m'}
  y = x/100;
otherwise
  disp('acabou')
end
disp( 'y = '); disp(y)
```

O resultado será  y = 0.0350, que representa a conversão de **x** metros para **y** centímetros.

**11 - Funções  .m - Regras e propriedades**

Essas funções são armazenadas nos arquivos **.m**. Os resistores usados em circuitos elétricos são determinados por códigos de cores.

Exemplo**:**

$R = (10 *A + B) * 10^{C}$

Usando-se  essa informação, pode-se criar um arquivo **.m** de função que retorna o valor da resistência associado com algum resistor padrão.

Exemplo**:**

```
resistor  ( ' br ' ,' bla ' ,' r ')
ans =
          1000
```

que representa a resistência de 1kilo ohm.

Segue abaixo a especificação do Matlab relacionando cores e números**:**

| Resistor | | |
|---|---|---|
| cor | MATLAB | número |
| preto | black | 0 |
| marrom | brown | 1 |
| vermelho | red | 2 |
| laranja | orange | 3 |
| amarelo | yellow | 4 |
| verde | green | 5 |
| azul | blue | 6 |
| violeta | violet | 7 |
| cinza | gray | 8 |
| branco | white | 9 |

## 12 - ANÁLISE DE DADOS

É dada uma matriz **:**

a =[2 3 5 ;4 9 8;1 2 4];

**mean (a):** para calcular a média de cada coluna.
**mean(a,1):** para calcular a média de cada coluna.
**mean(a,2):** para calcular a média de cada linha.
**median(a):** para determinar a mediana de cada coluna
**min(a):** para mostrar o menor valor.
**max(a):** para mostrar o maior valor.
**sum(a):** soma dos elementos de cada coluna.
**prod(a):** produto dos elementos de cada coluna.
**cumsum(a):** soma acumulada dos elementos de cada coluna.
**cumprod(a):** produto acumulado dos elementos de cada coluna.
**std(a):** para determinar o desvio padrão.
**diff(a):** para calcular a diferença entre um elemento e seu antecessor.
**cov(a):** serve para calcular a covariância de cada coluna de a.
**corrcoef(a):** é uma matriz de coeficientes de correlação onde cada linha é uma observação e cada coluna é uma variável.
**sort(a):** coloca em ordem crescente por coluna.
**sortrows(a,n):** coloca em ordem crescente por coluna somente em relação à coluna **n**.
**power(m, n):** calcula $\mathbf{m^n}$.
**factor(n):** determina os fatores primos de **n**.
**primes(n):** determina os números primos entre 0 e **n**.
**perms(m: n):** exibe as permutações de todos números compreendidos entre **m** e n(sendo **m < n**).
**nchoosek(m: n, p):** determina as combinações de **m** a **n**, tomados **p** a **p**(sendo **m < n**)..

## 13 - POLINÔMIOS

### 13.1 - Raízes

p = [ 1 3 5 7 9] corresponde a uma equação de grau 4**:** $x^4 + 3x^3 + 5x^2 + 7x + 9 = 0$

**roots (p):** acha as raízes da equação p.

Exemplo**:**

```
a = [1 3 2]
r = roots(a)
ans =
     -2
     -1
```

**poly(r) :** dada as raízes **r** podemos encontrar o polinômio **a:**

```
poly(r)
ans =
     1   3  2
```

que será a equação original $x^2 + 3x + 2 = 0$.

**13.2 - Multiplicação**

**conv(a,b) :** multiplica o polinômio a pelo b.

**13.3 - Adição**

Se a dimensão de a é igual à dimensão de b, a adição será dada por **:**

```
c = a + b
```

Se a dimensão de a for diferente da dimensão de b, podemos usar um dos casos**:**

i) Preencher com zeros os coeficientes das potências que faltam em um polinômio para este igualar em dimensão com outro

ii) Usar a função **polyadd(a, b)**.

**13.4 - Divisão**

**[q, r] = deconv (a,b)**

Cuja resposta consta de duas variáveis**:**

**q:** é o quociente da divisão de a por b.

**r:** é o resto da divisão de a por b.

**13.5 - Derivada**

**polyder (a):** Determina a derivada de um polinômio a.

Serão exibidos os coeficientes do polinômio que representam a derivada .

Exemplo**:**

```
a = [3  5  2  –1  5]
polyder(a)
ans =
     12  15  4 –1
```

que representa $12x^3 + 15x^2 + 4x - 1$

**13.6 - Avaliação**

**polyval (p ,x):** Avalia o polinômio p para o valor x.

Exemplo**:**

```
p = [2 4 5]
```

que representa $2x^2 + 4x + 5$

```
polyval(p, 2)
```

avalia o polinômio "p(x)" para x = 2, ou seja, p(2).

```
ans =
```

## 14 - SISTEMA DE EQUAÇÕES LINEARES

Resolve sistemas da forma**:**

**a* x = b**

Se det(a) ≠ 0 temos uma só solução, ou seja**:**

$x = a^{-1} * b$

que pode ser resolvido com qualquer um dos comandos**:**

x = inv(a) * b ou x = a\ b

Exemplo**:**

Seja resolver o sistema de equações**:**

$$\begin{cases} x_1 + 2x_2 + x_3 = 8 \\ 2x_1 - x_2 + x_3 = 3 \\ -x_1 + x_2 - 2x_3 = -5 \end{cases}$$

cuja entrada no Matlab será a = [1 2 1 ; 2 -1 1; -1 1 -2] Enter
b = [8; 3; -5] Enter

e a solução será**:**

x = inv(a)*b que resultará**:** x = 1
2
3

que representa**:**

$$\begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \end{cases}$$

## 15 - ADEQUAÇÃO DE CURVAS

p = **polyfit (x,y,n),** onde n é a ordem da função de y em relação a x.
n = 1 **:** interpolação linear.
n = 2 **:** interpolação quadrada.
n = 3 **:** interpolação usando função do terceiro grau.

Exemplo**:**

```
x = [0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
y = [-.447  1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];
n = 2;
p = polyfit(x, y, n)
p =
    -9.8108  20.1293  -0.0317
```

Esta resposta indica que a curva do segundo grau que melhor se adequa aos pontos dados é dada pela equação**:**

$y = -9.8108\ x^2 + 20.1293\ x - 0.0317$

Para traçar o gráfico dessa função junto com a curva que representa os valores de x e y anteriores deverão ser seguidos os seguintes passos**:**

a) Dar valores para traçar a curva mais adequada**:**
xi = linspace(0,1, 100)
b) Avalia a função nestes pontos**:**
z = polyval(p, xi)
c) Traça as duas curvas no mesmo gráfico **:**
plot(x, y, 'r', xi, z, '**:**') Traça duas curvas no mesmo gráfico, onde a primeira curva, traçada na cor vermelha representa uma linha ligando os pontos dados; e a segunda curva, na cor amarela e pontilhada, representa a curva de adequação encontrada.

## 16 - ANÁLISE NÚMERICA

### 16.1 – Plotagem

A plotagem de uma função deve obedecer a uma simples avaliação da função sobre algum intervalo e plotando os vetores resultantes.
**fplot:** avalia cuidadosamente a função a ser plotada, sendo dado um intervalo. Trabalha com qualquer função M- file que tenha um vetor de grandeza igual ao vetor já dado.

### 16.2 - Achando os zeros

**fzero('função', h):** determina onde a função especificada corta o eixo x mais próximo da abcissa x = **h**.

Exemplo**:**

**xzero** = **fzero**('cos',0.1)
xzero =
-1.5708

No exemplo, o valor x = -1.5708 representa onde a função y = cos (x) corta o eixo dos y, na abcissa mais próximo de x = 0.1.
**yzero = função(xzero):** determina o valor de y correspondente ao valor de x = xzero.

### 16.3 – Integração

São dadas três funções para computar a área sob a curva num intervalo finito.
**trapz:** aproxima a integral sobre a função, pelo somatório das áreas dos trapézios.
**trapz (x, y):** dá o valor da integral de y com o seu respectivo x usando para isso o método dos trapézios.
**quad** e **quad 8:** baseado no conceito matemático de quadratura.

### 16.4 - Diferenciação

**polyder(a):** determina a derivada numérica de **a**.
$y = 9x^3 + 2x^2 + 3x + 1$
y = 9 2 3 1
$\frac{dy}{dx} = 27x^2 + 4x + 3$
**polyder (y)** = 27 4 3

### 16.5 - Equações diferenciais

Nos casos em que as equações diferenciais não podem ser resolvidas algebricamente, é conveniente resolve-las numericamente. Por exemplo**:**

$\frac{d^2x}{dt^2} = \mu(1 - x^2)\frac{dx}{dt} + x = 0$ É a clássica equação diferencial de Van der Pol que descreve um oscilador.

Faz- se $y_1 = x$ e $y_2 = \frac{dx}{dt}$

então $\frac{dy_1}{dt} = y_2$ e $\frac{dy_2}{dt} = \mu(1 + y_1^2)\, y_2 - y_1$

**time span** = **tspan** = [0 30 ]

$y_0 =$ [1; 0]

ode 45 (´**vdpol´, tspan**, 40);

[t, y] = ode 45 ( ´vdpol´, tspan, 90);

Traça o gráfico de $y_1$ x $y_2$

plot ( y (**:**, 1), y ( **:** 2))

## 17 - GRÁFICO EM DUAS DIMENSÕES

### 17.1 - Usando o comando plot

**plot(x,y,'s')** onde**:** **x** representa o intervalo do eixo x usado para o traçado gráfico.

**y** representa os valores de y(ou **f(x)**).

**s** representa a cor(pode ser usada em geral a primeira letra da cor em inglês, exceto o preto que é representado pela letra **k**; representa também o tipo do caracter que vai simbolizar o traçado de cada ponto, por exemplo**: *, o, x,** etc. e também representa o estilo da linha( **-** é a sólida, **--** é a tracejada, **:** é a pontilhada, etc.)

Exemplo**:**

x = -3**:**10;

y = x.^2

**plot (x, y):** plotará o gráfico de uma função do 2º grau pré definida.

**plot(x, x.^2):** traça a mesma função anterior, definindo-a no próprio comando **plot**.

y = sin(x)

z = cos (x);

**plot (x, y, x, z):** serve para plotar duas curvas num mesmo gráfico.

**plot (x, y, 'r*', x, z, ':b'):** plota duas curvas no mesmo gráfico, sendo a primeira com linha sólida, na cor vermelha e os pontos representados por asteriscos e a segunda com linha pontilhada e na cor azul.

### 17.2 - Estilo de linhas , marcadores e cores

As configurações usadas no Matlab são**:**

| TIPOS DE CORES | MARCADORES DE PONTOS | TIPOS DE LINHAS |
|---|---|---|
| y yellow(amarelo) | . point(ponto) | - solid(sólida) |
| m magenta(magenta) | o circle(círculo) | **:** dotted(pontilhada) |
| c cyan(azul-esverdeado) | x x-mark(x) | -. dashdot(traço-ponto) |
| r red(vermelho) | + plus(+) | -- dashed(tracejada) |
| g green(verde) | * star(asterisco) | |
| b blue(azul) | s square(quadrado) | |
| w white(branco) | d diamond(diamante) | |

| | | | |
|---|---|---|---|
| k | black(preto) | v | triangle(triângulo) |
| | | ^ | triangle(triângulo) |
| | | < | triangle(triângulo) |
| | | > | triangle(triângulo) |
| | | p | pentagram(estrela de 5 pontas) |
| | | h | hexagram(estrela de 6 pontas) |

**17.3 – Janelas de gráficos**

**subplot(m, n, p):** usado para você plotar vários gráficos, um em cada janela. O valor de **m** é um inteiro que representa o número de janelas por linha, **n** representa o número de janelas por coluna e **p** representa a janela onde será plotado o gráfico contando-se em ordem crescente da direita para a esquerda

**17.4 - Configuração de grades, eixos, rótulos e legendas**

**grid on:** é um comando que coloca grade no gráfico.
**grid off:** remove a grade colocada no gráfico.
**axis on:** exibe os eixos coordenados.
**axis off:** exclui os eixos coordenados.
**axis([xinic xfim yinic yfim]):** muda os intervalos de exibição das abscissas e das ordenadas do gráfico em duas dimesnsões.

Exemplo**:**

**axis([-2 4 0 6]):** plota o gráfico da função em questão, sendo que o valores de x serão exibidos no intervalo de –2 a 4 e os valores de y serão exibidos no intervalo de 0 a 6.
**x label (' rótulo do eixo x' ):** título do eixo x.
**y label( ' rótulo do eixo y'):** título do eixo y.
**title (' título '):** coloca título no gráfico, centralizado, na parte superior da tela.
**text (x, y, ' título'):** coloca texto usando as coordenadas (x, y) como início do texto.
**gtext (' título'):** coloca título, que irá começar a ser escrito no local onde o mouse for clicado, coincidindo o início do texto com o cruzamento das duas retas que aparecem.
**legend ( 'legenda'):** cria uma caixa de legenda no canto direito superior. Quando desejar mudar de lugar, basta clicar na caixa e arrastar com o botão esquerdo do mouse.

Exemplo**:**

Se você for traçar duas curvas no mesmo gráfico, uma representando a função y = sin(x) e outra representando y = cos(x) e for fazer a legenda correspondente, deverá digitar**:**

```
x = 1:9;
plot(x, cos(x),' r ', x, sin(x), ' b ')
legend('cos(x)', 'sin(x)')
```

OBS**:** A ordem da definição das funções que aparecem na legenda, deverá seguir a mesma ordem usada no comando plot.
**legend off:** apaga a legenda.

**17.5 - Background da tela gráfica**

**colordef cor:** colore o background(cor de fundo da tela) com a cor que você selecionou, a será aplicada sempre para a figura subsequente. Há só duas opções de cores para esta função**: white** e **black**.

Exemplo**:**
**colordef black:** colore o fundo da tela de preto.
**colordef white:** colore o fundo da tela de branco.

**17.6 - Configuração de figuras**
**figure(n):** ativa a janela da figura especificada.
**clf:** apaga a corrente figura.

**17.7.- Manipulando gráficos**
**loglog:** usa escala logarítmica para ambos os eixos.
**semilogx:** usa escala logarítmica só no eixo x.
**semilogy:** usa escala logarítmica só no eixo y.
**hold on:** mantém o gráfico atual na tela para ser impressa outra curva no mesmo gráfico.
**hold off:** desativa o comando hold.
**zoom:** expande a figura a cada click do botão esquerdo do mouse e clicando-se no botão direito acontece o contrário.
**zoom off:** desativa o comando zoom .
**box:** liga a caixa ao eixo(traça eixo das ordenadas e eixo das abscissas nos dois lados).

**17.8 - Outros tipos de gráficos em duas dimensões**
**comet(x,y):** traça o gráfico, semelhante ao comando **plot** só que você observa a curva sendo traçada. Caso a curva seja traçada muito rápido, você deverá diminuir o incremento de x, para observar melhor.
**área(x, y, n):** é semelhante ao comando plot (x, y), exceto que, a área sob a curva no intervalo de x dado, é hachurada. O valor de **n** representa o valor da ordenada no qual a área sob esta ordenada não é hachurada da mesma maneira que as outras.
**pie(a,b):** gráfico de torta, onde a é um vetor dos valores e b é a especificação da fatia da torta que ficará destacada das demais.

Exemplo**:**
a = [1 2 3 4 5]
**pie(a, a == max(a)):** plota os valores de **a,** sendo destacada a fatia que representa a parte maior deste conjunto de dados.
Para referenciar o que representa cada fatia usa-se o comando **legend** colocando tanto nomes quantos forem as fatias e seguindo a ordem relacionada no vetor **a**.
Exemplo**:** **legend**('MG', 'SP', 'GO', 'PE', 'SC')
**pareto(a):** os valores do vetor a são desenhados em barras em ordem crescente.
**stairs(x, y):** só contorna o gráfico exibido na tela, gráfico de "degrau".
**bar(x,y):** plota um gráfico de barras.
**barh(x, y):** plota horizontalmente um gráfico de barras.
**hist(y):** desenha um histograma de 10 caixas.
**hist(y, n):** desenha um histograma com **n** caixas.
**hist(y, x):** desenha um histograma usando os bins especificados em x (que é um vetor).
**stem(z) :** cria um gráfico através dos valores do vetor z conectados ao eixo horizontal por uma linha.
**stem(x, z):** plotar os dados em z para os valores especificados em x.
**errorbar(x, y):** plota o gráfico do vetor x versus o vetor y com representação de uma barra(semelhante ao simbolo de "fecha colchete").

**compass(U, V):** desenha um gráfico que exibe os vetores com componentes (U, V) como setas emanando da origem.
**feather.compass(z):** desenha um gráfico que exibe o ângulo e magnitude dos elementos complexos de z como setas emanando da origem.
**feather(z):** plota alguns dados usando arranjos que têm seus pontos igualmente espaçados em uma linha horizontal, semelhantes a penas de ave.
**rose(v):** desenha um histograma polar de 20 bins para os ângulos no vetor v.
**rose(v, m):** desenha um histograma com n bins.
**ginput:** dá a média dos pontos selecionados do corrente gráfico usando o mouse.
[x, y] = ginput (n)
**fill(x, y,'c'):** preenche o polígono em duas dimensões definido por um vetor coluna x e y e com a cor especificada c.

Exemplo**:**
```
d = [1 2 3 4]
e = [2 3 4 1]
fill(d, e,' r ')
```
vai ligar os pontos dados formando um triângulo na cor vermelha.

### 17.9 – Coordenadas polares

**polar(t, r, 's')**, onde **t** = ângulo vetor em radianos
**r** = raio vetor
**s** = caracter opcional que descreve cor, marcador ou estilo de linha

Exemplo**:**
```
t = linspace(0, 2*pi)
r = sin(2*t).*cos(2*t)
polar(t, r)
```

## 18-GRÁFICOS TRIDIMENSIONAIS

Alguns comandos são iguais aos usados em gráficos de 2 dimensões, tais como os para colocar legenda, zoom, título, texto, etc.

### 18.1 - Traçado de linhas

O comando para traçar linha em duas dimensões pode também traçar essas mesmas linhas em três dimensões, usando- se para isso um outro comando**:**
**plot3:** plota linhas em três dimensões.
**plot3($x_1$,$y_1$,$z_1$,$S_1$,$x_2$,$y_2$,$z_2$,$S_2$,...)**
$x_n$, $y_n$, $z_n$ são vetores ou matrizes.
$S_n$ são caracteres opcionais especificados pela cor, marcador de estilo de linha.

Exemplo**:**
**t = linspace(0, 10*pi)**
**plot3(sin(t), cos(t), t):** desenha uma hélice.
**axis('ij'):** muda a direção da hélice.

### 18.2 - Gráficos de malha e superfície

**[x, y] = meshgrid(a,b)**

**z = f(x, y)**
**mesh (z)**
onde**:**
**a:** representa o intervalo que x varia.
**b:** representa o intervalo que y varia.
**z = f(x, y):** é a definição de z em função de x e y.
**mesh (z):** plota a malha em terceira dimensão.
**surface(z):** igual mesh só que os espaços entre as linhas são preenchidos.

### 18.3 - Manipulando gráficos

Permite que você especifique o ângulo do qual se observa um gráfico (ou figura) em terceira dimensão.
**view (azimuth, elevação):** "elevação" descreve a localização do observador como um ângulo acima do plano xy e "azimuth" refere- se ao ângulo interior o plano xy onde o observador fica .
**zlabel('rótulo'):** usa-se para colocar rótulo no eixo **z**.
**rotate3d on:** aqui pode-se girar a figura, usando-se o mouse, para visualizá-la de vários ângulos , basta para isso apertar o botão esquerdo do mouse e arrastá-lo.

### 18.4 – Outros tipos de gráficos tridimensionais

**ribbon(z):** é o mesmo que **surf(z)** , só que no lugar de malhas serão traçadas tiras em três dimensões.
**fill3(x, y, z):** os vértices do polígono são especificados pelas triplas de componentes de x, y e z.

Exemplo**:**
**fill3(rand(3,4),rand(3,4),rand(3,4),rand(3,4)):** traça 4 triângulos cujos vértices são triplas geradas aleatoriamente.
**contour(z):** plota as curvas de nível da função **z**(gráfico de em segunda dimensão).
**clabel(contour(z))** plota as curvas de nível **z** com suas respectivas cotas.
**contourf(z):** semelhante ao comando **contour(z)** só que os espaço entre as curvas de nível são preenchidos(coloridos).
**contour3(z):** plota as curvas de nível fa função **z**(gráfico em terceira dimensão).
**bar3** e **bar3h:** são versões em terceira dimensão de bar e barh.
**pie3:** é a versão em terceira dimensão de pie .
**comet3:** exibe o traçado animado de uma curva em três dimensões.

Exemplo**:**
t = 0**:**pi/50**:**10*pi;
comet3(sin(t),cos(t),t) você visualiza o desenho de uma hélice sendo traçado.

### 18.5 - Interpretando mapa de cores

**colormap([R G B]):** instala 64 entradas de cor.

| RED | GREEN | BLUE | COR |
|---|---|---|---|
| 0 | 0 | 0 | black |
| 1 | 1 | 1 | white |
| 1 | 0 | 0 | red |

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | green |
| 0 | 0 | 1 | blue |
| 1 | 1 | 0 | yellow |
| 1 | 0 | 1 | magenta |
| 0 | 1 | 1 | cyan |
| .5 | .5 | .5 | medium gray |
| .5 | 0 | 0 | dark red |
| 1 | .62 | .40 | copper |
| .49 | 1 | .83 | aquamarine |

onde**:**

**R G B** representa os números(entre 0 e 1) que simbolizam as tonalidades respectivas das cores vermelha, verde e azul.

Exemplo**:**

**colormap([0 0 1]):** fará com que a figura tridimensional seja exibida na cor azul.
**colormap([1 0 0]):** fará com que a figura seja exibida na cor vermelha.
**colormap([1 1 0]):** fará com que a figura seja exibida na cor amarela.
**colormap('default'):** plota a figura nas cores padrão do Matlab.

### 18.6 - Usando cores para adicionar informações

A cor pode ser usada para adicionar informações aos gráficos em terceira dimensão se, e somente se, isto for usado para mostrar uma quarta dimensão.
**surf(x, y, z, z):** onde o quarto argumento é usado como um índice no colormap.

### 18.7 - Criando e alterando mapa de cores

**brighten(n):** controla a intensidade de cores escuras, onde**:**
**(0 < n ≤ 1):** é usado para clarear;
**(-1 ≤ n < 0):** é usado para escurecer.

## 19 - FERRAMENTAS DE MATEMÁTICA SIMBÓLICA

É uma coleção de ferramentas(funções) do MATLAB que são usadas para manipulação e resolução de expressões simbólicas.

### 19.1 – Criação de objetos simbólicos

Os objetos simbólicos são criados por meio de valores numéricos ou literais usando a função **sym**.

Exemplo**:**

**x = sym('x'):** cria uma variável simbólica x.
**y = sym('1/3'):** cria uma variável simbólica y contendo o valor 1/3.
Pode-se usar também a fórmula **sym(a, fmt)**, onde "a" é um valor ou matriz numérica e "fmt" é uma especificação opcional de formato que pode ser "f", "r", "e" ou "d" representando respectivamente os formatos "floating point format", "rational", "estimate error format", "decimal format". Quando a variável é composta de duas ou mais variáveis, usa-se a função syms para definí-la.

**19.2 - Representação de expressões simbólicas**

O MATLAB representa expressões simbólicas internamente como expressões contendo objetos simbólicos, para diferenciá–las das variáveis, expressões ou operações numéricas.

Exemplos de expressões simbólicas**:**

| Expressões simbólicas | Representação do MATLAB |
|---|---|
| $y = \frac{1}{2x^3}$ | x = sym('x') |
| $p = \frac{1}{\sqrt{2x}}$ | x = sym('x') |
| $q = \cos(x^2) - \sin(2x)$ | x = sym('x') |
| $m = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ | syms('a', 'b', 'c', 'd '); |
| $k = \int_a^b (x^3/\sqrt{1-x})dx$ | syms x, a, b |

Entrada de dados no Matlab, relativas ao quadro acima**:**

i) **x = sym('x'):** cria uma variável simbólica x.

y =1./2.*x.^3 determina a expressão simbólica.

ii) **x = sym('x'):** cria uma variável simbólica x.

p = 1./sqrt(2.*x) determina a expressão simbólica.

iii) **x = sym('x'):** cria uma variável simbólica x.

q = cos(x.^2) – sin(2.*x) determina a expressão simbólica.

iv) **m = syms('a',' b', 'c', 'd'):** cria as variáveis simbólicas a, b, c, d.

m = [a, b; c, d] cria a matriz simbólica**:**

m = $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

det(m) = [a b; c d];

ans =

a * d – b * c acha o determinante da matriz simbólica m.

v) k= syms x a b

k = int(x.^3./sqrt(1-x), **a**, **b**) acha a integral de **a** até **b** da função especificada.

Outro exemplo**:**

f = $ax^2 + bx + c$, usando-se a sintaxe do MATLAB corresponderá a**:**

syms a b c x

f = a* x ^ 2 + b * x + c

que resulta**:**

f =

a * x ^ 2 + b * x + c

19.**3 – Derivadas simbólicas**

**diff(função, variável, ordem):** determina a derivada de uma função simbólica.

onde**:** **função :** é uma função literal de uma ou mais variáveis.
**variável:** é a variável em relação a qual você deseja derivar.
**ordem :** é a ordem da derivada a ser calculada.

Exemplos**:**

i) x = sym('x');
**diff(x^3,x, 2):** determina a derivada segunda de $x^3$ em relação a x, cujo resultado será**:** ans = 6x.

ii) syms x y;
z = x^2*log(y) + exp(-x)*sqrt(y)
ii.1) **diff(z,x,2):** determina a derivada de z em relação a x, resultando**:**
ans = 2*log(y) – exp(x)*sqrt(y)
ii.2) **diff(z,y,2):** determina a derivada segunda de z em relação a y, que resulta em**:**
ans = -x^2/y^2 + ¼*exp(-x)/y^(3/2)

19.**4 – Integrais simbólicas**

**int(função, variável, a, b):** determina a integral de uma função simbólica.

onde**:** **função :** é uma função literal de uma ou mais variáveis.
**variável:** é a variável em relação a qual você deseja integrar.
**a :** é o limite inferior da integral.
**b :** é o limite superior da integral.

Exemplos**:**

i) syms x a b;
i.1) z = x + y^2;
**int(z, x, a, b):** Calcula a integral da função **z** de **a** até **b**, em relação a x.
resulta**:** ans = ½***b**^2 + y^2***b** – ½***a**^2 - y^2***a**
i.2) Considerando a mesma função acima
**int(z, y, 1, 3):** Calcula a integral da função **z** de **1** até **3**, em relação a y.
resultando**:** ans = 2*x + 26/3

**19.5 – Inversa de funções**

**fi nverse(função, variável):** determina a inversa de uma função

onde**:** **função:** é uma função literal de uma ou mais variáveis.
**variável:** é a variável em relação a qual você deseja achar a inversa.

Exemplos**:**

i) **finverse(a*x + b):** como se fosse achar a inversa de y = ax + b, que resulta**:**
ans = -(b-x)/a

ii)**finverse(a*b + c*d – a*z, a):** como se fosse determinar a inversa de y = **a**b +cd – **a**z em relação à variável **a**, resultando em**:**
ans = -(c*d – a)/(b – z)

**19.6 – Somatório de expressões**

**symsum(função, variável, a, b):** determina a integral de uma função simbólica.

onde**:** **função : ** é uma função literal de uma ou mais variáveis.
**variável:** é a variável em relação a qual você deseja achar o somatório.
**a :** é o limite inferior da integral.
**b :** é o limite superior da integral.

Exemplos**:**

i) Para determinar $\sum_1^n (2n-1)^2$ procedemos assim**:**

```
n = sym('n')
symsum(2*n-1)^2,1,n), cujo resultado será:
ans = 11/3*n +8/3 - 4*(n + 1)^2 + 4/3*(n + 1)^3
```

ii) Para determinar $\sum_1^b (2an)^2$

symsum((2*a*n),1,b), que resultará**:**
ans = a*(b + 1)^2 - a*(b + 1)

**19.7 – Função composta**

**compose(função f, função g ):** determina a f o g, ou f(g(x)).
onde**:** **função f :** é uma função literal de uma ou mais variáveis.
**função g :** é uma função literal de uma ou mais variáveis.

Exemplo**:**
f = x^2;
g = sin(x)
**compose(f, g):** resulta**:** ans = sin(x)^2
**compose(g, f):** resulta**:** ans = sin(x^2)

**19.8 – Raízes de equações**

**solve(f, a):** determina as raízes da equação **f = 0**, em relação à variável **a**.
onde**:** **função f:** é uma função literal de uma ou mais variáveis.
**a:** é a variável livre.

Exemplo**:**
i) f = x^2 + 3*x +2
**solve(f):** acha as raízes da função **f**, resultando**:**
ans = [ -2 ]
[ -1 ]
ii) f = b – 5*a
**solve(f) ou solve(f, a):** resulta**:** ans = 5*a
solve(f, a) resulta**:** ans = 1/5*b

**19.9 – Multiplicação de polinômios**

**collect(f):** determina o resultado da multiplicação dos polinômios de **f**.
onde**:** **função f:** é uma expressão envolvendo produtos de polinômios.

Exemplo**:**
f = (x^2 + 1)*(x^3 + 2*x – 3)
collect(f) resulta**:** ans = x^5 + 3*x^3 – 3*x^2 + 2*x – 3

**19.10 – Simplificação de expressões**

Há dois comandos que podem ser usados na simplificação de expressões**:**

I) **simplify(f):** simplifica a variável **x** da função **f** pela variável(ou constante) **y**.

onde**:** **simplify:** simplifica expressões usando identidades.

**função f:** é uma função literal de uma ou mais variáveis.

II) **simple(f):** simplifica expressões e coloca na forma técnica.

Exemplo**:**

g = (2*x + 1)*(x^2 + x)/(2*x + 2)

simplify(g) resulta**:** ans = ½*x*(2*x + 1)

simple(g) resulta**:** ans = x^2 + ½*x

**19.11 – Substituição de variáveis e cálculo de expressões**

**subs(f, x, y):** substitui a variável **x** da função **f** pela variável(ou constante) **y**.

onde**:** **função f :** é uma função literal de uma ou mais variáveis.

**y :** é uma variável ou uma constante para a qual se quer avaliar uma função.

Exemplo**:**

i) f = x^2 + 2*x + 1

i) f = subs(f, x, y) resulta**:**

f = y^2 + 2*y + 1

ii) f = x^2 + 2*x + 1

f = subst(f, x, 3) substitui **x** por **3** na função **f**, resultando**:**

f = 16

**19.11.– Fatoração e expansão de polinômios**

O Matlab usa dois comandos, um para fatorar polinômios e outro que faz o inverso, ou seja faz o produto de polinômios e ordena-os.

**factor(f):** fatora o polinômio **f**, expressando-o como um produto de polinômios.

**expand(f):** distribui o produto através de uma soma.

Exemplos**:**

syms a x

i) f = x^4 – 5*x^3 + 5*x^2 + 5*x – 6

g = factor(f) fatora **f**, resultando em**:**

g = (x - 1)*(x - 2)*(x - 3)*(x + 1)

ii) h = expand(g) resultará

h = x^4 – 5*x^3 + 5*x^2 + 5*x – 6

**19.12.- Extração de numeradores e denominadores**

Se você possui um polinômio racional ou então uma expressão que pode ser transformada em polinômio racional, o MATLAB lhe dá a opção de extrair o numerador e o denominador, usando-se a função **numden.**

**[n, d] = numden(f):** determina o numerador e o denominador de um polinômio racional.

onde**:**

**n** representa o numerador da fração.

**d** representa o denominador da fração.

Exemplo**:**

```
f = a*x^2 /(b - x)
[n, d] = numden(f)
n =
        a*x^2
d =
        b - x
```

Obs**:** Se você usar o comando numden(f) , será obtido somente o numerador da fração ou seja**:** ans = a*x^2.

**19.14 – Exibição de expressões na forma mais legível**

**pretty(f):** coloca a função numa forma mais legível

onde**:** **f:** é uma função de uma ou mais variáveis.

Exemplo**:**

pretty(5/6*x^2+ sqrt(x) – 1/3*x^3) cuja resposta será;

ans = $5/6x^2 + x^{1/2} - 1/3x^3$

$$\mathbf{5/6\ x^2 + x^{1/2} - 1/3\ x^3}$$

**19.15 - Variável aritmética de precisão(vpa**

A função **vpa** avalia uma expressão simbólica para uma designada precisão sem afetar qualquer outra operação, o formato é **:**

**vpa('expressão', d)** onde **:**

expressão**:** é uma expressão simbólica.

d**:** é o número de dígitos desejado.

Exemplo**:**

**vpa(pi,16):** avalia o valor de pi com 16 dígitos

```
ans =
   3.14159265358979
```

**vpa(exp(1),12):** avalia o valor de **e**(base do logaritmo neperiano) com 12 dígitos.

ans = 2.71828182846

**19.16 – Equações diferenciais ordinárias**

**dsolve('f, cond', 'var'):** computa soluções simbólicas para EDO.

onde**:**

**f:** é uma função de uma ou mais variáveis.

**cond:** são as condições iniciais(ou de contorno).

**var** é a variável em relação a qual deseja resolver o problema.

Exemplos**:**

i) Seja resolver a EDO de 1ª ordem**:** $\frac{dy}{dx} = 1 + y^2$ , teremos**:**

**dsolve('Dy = 1 + y^2')** e cuja resposta será**:**

ans = tan(t – C1) onde C1 é uma constante de integração.

ii) Resolvendo o mesmo problema anterior, só que com a condição inicial y(0) = 1**:**

**dsolve('Dy = 1 + y^2, y(0) = 1'):** determina a solução da EDO em relação à variável x, cujo resultado será**:**

ans = tan(t + ¼*pi)

OBS**:** o comando de resolução da EDO acima poderia também ser escrito assim**:** dsolve('Dy = 1 + y^2, y(0) = 1, 'x'), indicando que está resolvendo a equação em relação à variável x.

iii) Para resolver a seguinte EDO de 2ª ordem, com duas condições iniciais**:** $\frac{d^2y}{dt^2} = \cos(2t) - y \qquad \frac{dy}{dt}(0) = 0 \qquad y(0) = 1,$ usamos o comando**:**

y = dsolve('D2y = cos(2*t) – y, Dy(0) = 0, y(0) = 1')

obtendo a resposta(em relação a **t**)**:** y = -2/3*cos(t)^2 +1/3 + 4/3*cos(t)

para simplificar usamos o comando; y = simple(y) cujo resultado será**:**

y = -1/3*cos(2*t) + 4/3*cos(t)

OBS**:** se quiséssemos esta mesma equação em relação à variável **x**, usaríamos**:**

y = dsolve('D2y = cos(2*t) – y, Dy(0) = 0, y(0) = 1', 'x') , obtendo o resultado**:**

y = cos(2*t)+(-2*cos(t)^2+2)*cos(x)

**19.17 - Conversão da forma numérica para forma simbólica**

**poly2sym(f, 'var'):** converte a função **f** da forma numérica para a simbólica.

onde**:**

**f:** é um vetor de números.

**var:** é a variável livre.

Exemplos**:**

i) f = [2 3 –1 5]

poly2sym(f, 'y') resulta**:**

ans = 2*y^3 + 3*y^2 – y + 5

ii)poly2sym([3 4 –2]) resultará**:**

ans = 3*x^2 + 4*x – 2

**19.18 - Conversão da forma numérica para forma simbólica**

**sym2poly(f,):** converte a função **f** da forma simbólica para a numérica.

onde**:**

**f:** é uma função simbólica.

Exemplo**:**

f = x^3 – 2*x +7

sym2poly(f) resultará**:**

ans = 1 –2 7

**19.19 – Determinação das variáveis livres**

As variáveis "default", principalmente as que são usadas nas transformadas de Fourier, Laplace e Z, são determinadas usando o comando **findsym**, tendo formas que permite você especificar diferentes variáveis independentes.

**findsym(f):** determina a variável livre da função **f**.

onde **f:** é uma função de uma ou mais variáveis.

Exemplos**:**

```
syms a x  t w
i)      f = x^3 – 2*x +7
findsym(f)      resultará:
ans =   x
ii) f = exp(-a*t)*cos(w*t)
findsym(f)      resultará:
ans = a, t, w
```

**19.20 - Transformada de FOURIER**

A transformada de Fourier é definida como**:**

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

A transforma inversa de Fourier é dada por**:**

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} dt$$

A transformada e a transformada inversa de Fourier são extensivamente usadas na análise de circuitos para determinar características de ambos os sistemas de domínio de tempo e frequência. O MATLAB usa um "**w**" para representar o"ω" em expressões simbólicas.

**fourier(f, t, w):** determina a transformada usando os parâmetros **t** e **w.**

**ifourier(f, t, w):** determina a transformada inversa usando os parâmetros **t** e **w.**

Exemplo**:**

```
syms t w
```

**f = t*exp(-t^2):** cria uma função.

```
f =
        t*exp(-t^2)
```

**F = fourier(f,t,w):** determina a transformada usando os parâmetros usuais t e w, que resulta em**:**

```
F = -1/2*i*pi^(1/2)*w * exp(-1/4*w^2)
```

**ifourier(F, w, t):** encontra a transformada inversa de Fourier, que resultará**:**

```
ans =
         1/2*4^(1/2)*t*exp(-t^2)
```

**19.21 – Transformada de Laplace e transformada inversa de Laplace**

A transformada de Laplace efetua a operação**:**

$$L(s) = \int_{0}^{\infty} f(t) e^{-st} dt$$

para transformar f(t) no domínio **tempo** para L(s) no domínio **s**, usamos então**:**

**laplace(f, t, s):** determina a transformada de Laplace f(t) ➔ L(s).

onde**:**

**f:** é uma função de uma ou mais variáveis.

**t:** representa o tempo.

**s:** é a variável livre.

Exemplo**:**

syms a s t w
f = exp(-a*t)*cos(w*t) cria uma expressão f(t)
L = laplace(f, t, s) acha a transformada f(t) ➔ L(s), resultando em**:**
L = (s + a)/((s + a)^2 + w^2)

Para determinar a transformada inversa de Laplace, usamos o comando**:**
**ilaplace(L, s, t)**

Exemplo**:**
**ilaplace(L, s, t):** transforma L(s) ➔ f(t), que resulta em**:**
ans = exp(-a*t)*cos(w*t)

**19.22 – Transformada Z e transformada inversa Z**

As transformadas Z são usadas para analisar sistemas discretos de tempo. A transformada Z é definida como**:**

$$F(z) = \sum_{n=0}^{\infty} f(n) z^{-n}$$ onde N é um número complexo.

**ztrans(f, n, z):** determina a transformada Z.
**iztrans(G, z, n):** determina a transformada inversa Z.

onde**:**

**f:** é uma função da variável **n.**
**n** e **z:** são os parâmetros usuais.
**G:** representa a transformada Z da função **f**.
Exemplo**:** syms n z
f = 2^n/7 – (-5)^n/7 cria uma função f(n), também representada por**:**
f = 1/7*2^n – 1/7*(-5)^n
G = ztrans(f, n, z) transforma usando os parâmetros usuais **n** e **z**, resultando**:**
G = z/(z – 2)/(z + 5)
Para determinar a transformada inversa Z usamos**:**
iztrans(G, z, n) determina a transformada inversa Z, que resultará**:**
ans = 1/7*2^n – 1/7*(-5)^n

**19.23 – Plotando expressões simbólicas**

**ezplot(f, [inic fim]):** traça o gráfico da função simbólica **f**.

onde**:**

**f:** é uma função de uma variável.
**[inic fim]:** é o intervalo de variação da abscissa x, se este for omitido será assumido o intervalo "default" [-2π 2π]

Exemplo**:** x = syms('x')
f = sin(x) + x^2
ezplot(f, [-3 5]) traça o gráfico de **f** com **x** variando de –3 até 5.

**19.24 – Caixa de ferramentas de funções(FUNTOOL)**

O Matlab tem uma caixa de ferramentas que faz diversas operações como se fosse uma "calculadora de funções". Para acioná-la basta digitar o comando **funtool**.

Surgirão na tela, três janelas com as seguintes características**:**

i) Na 1ª figura aparecerá o gráfico da função **f(x)** que for definida na janela 3.

ii) Na 2ª figura aparecerá o gráfico da função **g(x)** que for definida na janela 3.

iii) Na 3ª figura aparecerá primeiramente o espaço para você entrar(definir) com a função **f(x)**. Em seguida tem-se o espaço para você entrar(definir) com a função **g(x)**. Após aparece o local onde você definirá o intervalo da abscissa **x**(o intervalo *default* assumido é de [-2*pi, 2*pi]. Em seguida vem o espaço em que você pode entrar com uma constante **a** qualquer pela qual, sobre a função **f(x)** serão poderão ser efetuadas várias operações matemáticas.

Abaixo opções que você aciona de maneira semelhante às teclas de uma calculadora; segue a função de cada tecla**:**

**df/dx** ➔ Derivada de f(x).
**int f** ➔ Integral de f(x).
**simple** ➔ Simplifica f(x).
**num f** ➔ Traça o gráfico da função do numerador de f(x)(se for função racional).
**den f** ➔ Traça o gráfico da função do denominador de f(x)(se for função racional).
**1/f** ➔ Traça o gráfico de 1/f(x).
**finv** ➔ Plota o gráfico da inversa de f(x).
**f + a** ➔ Traça o gráfico de f(x) + a.
**f - a** ➔ Traça o gráfico de f(x) - a.
**f * a** ➔ Traça o gráfico de f(x) * a.
**f / a** ➔ Traça o gráfico de f(x) / a.
**f ^ a** ➔ Traça o gráfico de f(x) ^ a.
**f (x+ a)** ➔ Traça o gráfico de f(x + a).
**f (x* a)** ➔ Traça o gráfico de f(x* a).
**f + g** ➔ Traça o gráfico de f(x) + g(x).
**f - g** ➔ Traça o gráfico de f(x) - g(x).
**f * g** ➔ Traça o gráfico de f(x) * g(x).
**f / g** ➔ Traça o gráfico de f(x) / g(x).
**f (g)** ➔ Traça o gráfico da função composta f(g(x)).
**g = f** ➔ Faz g(x) = f(x).
**swap** ➔ Troca as funções f(x) com g(x).
**Insert** ➔ Adiciona a função f(x) atual à lista de funções existente.
**Cycle** ➔ A cada clique, vai mostrando as funções da lista de funções.
**Delete** ➔ Apaga a função ativa da lista de funções.
**Reset** ➔ Configura os valores de **f, g, x, a** e a **lista de funções** para seus valores normais(*default*).
**Help** ➔ Mostra a ajuda de caixa de ferramenta de funções(funtool).
**Demo** ➔ Mostra a geração da função f(x) = sen(x) usando 9 passos.
**Close** ➔ Fecha esta caixa de ferramentas.

**19.25 -Análise de sinal**

A caixa de ferramentas de processamento de sinal proporciona ferramentas para examinar e analisar sinais; examinando e analisando seu teor de frequências ou espectro e criando filtros.

Exemplo**:**

Vamos construir um sinal de ruídos**:**

t = linspace (0, 10, 512); eixo do tempo
x = 3*sin(5*t) – 6*cos(9*t) + 5 *randn((size(t));
plot(t, x) plota sinal com ruído Gaussiano

Maiores detalhes e outras opções relativas a análise de sinal, por se tratar de um assunto mais complexo , serão abrangidos numa próxima apostila.

## 20.- IMPRESSÃO DE FIGURAS

### 20.1.- Imprimindo figuras na impressora

Para imprimir as figuras que são feitas no MATLAB usam-se os comandos**:**

**print –dwin:** quando se quer imprimir as figuras em impressora no modo preto e branco.
**print –dwinc:** quando se quer imprimir as figuras em impressora no modo colorido.

Podem ser usados 3 comandos que especificam a orientação da figura no papel na hora da impressão**:**

**orient portrait :** especifica a impressão normal.
**orient landscape:** especifica a impressão horizontal, no sentido do maior lado da folha.
**orient tall:** funciona como se "espichasse" o desenho na horizontal e na vertical.

### 20.2.– Salvando figura em um arquivo bitmap(extensão BMP)

Para salvar uma figura num arquivo com extensão BMP, que posteriormente poderá ser inserida, como "figura", em *softwares* tais como o Word for Windows,o Excel, etc., deve ser usado o comando**:**

**print -dbitmap path filename**

onde**:**

**path:** é o caminho onde será gravado o arquivo(drive e pasta).
**filename:** é o nome do arquivo, que assumirá automaticamente a extensão **bmp**.

Exemplos**:**

i) **print -dbitmap c:\windows\parabol:** a figura atual será salva, num arquivo com o nome parabol.bmp, na pasta windows do drive C.

ii) **print -dbitmap a:\curva:** a figura atual será salva, num arquivo, com o nome curva.bmp no seu disquete.

## 21 - AJUDA(HELP)

Há diversos tipos de ajuda(help) no Matlab algumas são acionadas por meio do menu e outras são digitadas na linha de comandos.

### 21.1 - Ajuda através do menu

Acionando o menu **Help** você terá a opção dos comandos**:**

**Help window:** possibilita pesquisar os assuntos através dos tópicos do Matlab.

**Tips:** aparecerão as explicações de como usar os três tipos de Help que devem ser digitados na linha de comandos.

**Examples e demos:** há vários exemplos e demonstrações de funções do Matlab que podem ser visualizadas.

### 21.2 - Ajuda na linha de comandos

São comandos que devem ser digitados e em seguida deve ser apertada a tecla **Enter** para eles serem acionados.

**help comando:** serve para acionar a ajuda sobre o comando ou função do Matlab especificado.

Exemplo**:**

**help plot:** aparecerão as informações relativas à função **plot** na linha de comando do Matlab.

**helpwin comando:** semelhante ao comando anterior só que as mesmas informações irão aparecer numa janela de help.

**heldesk:** irá fazer uma conexão com a página do Matlab na Internet onde você poderá ter outras informações como problemas que possam surgir no Matlab, etc.

**lookfor palavra:** mostrará na linha de comandos todas as funções que tenham alguma relação com a palavra especificada.

Exemplo**:**

**lookfor inverse:** aparecerão as funções do Matlab que tenham alguma ligação com a palavra inverse, como **asec**(inversa da secante), **ifft**(transformada inversa de fourier), etc.

**demo:** semelhante ao comando do Help de menu**:** "examples e demos". **s**

### 21.3 - Outros tipos de ajudas

O Matlab tem algumas outras ajudas que estão embutidas em comandos de demonstrações específicas de funções, bastando que estes sejam digitados na linha de comandos.

Exemplos**:**

**symintro:** dá uma introdução à caixa de ferramentas de matemática simbólica.
**symcalcdemo:** demonstra a utilização de diversas funções simbólicas.
**symlindemo:** mostra algumas aplicações de álgebra linear simbólica.
**symvpademo:** demonstra o uso de variável aritmética de precisão.
**symrotdemo:** mostra características de rotações de plano.
**symeqndemo:** demonstra resolução de equações simbólicas.
**xpsound:** demonstra a capacidade de som do Matlab.
**imagedemo:** demonstra a capacidade de imagem do Matlab.
**graf2d:** mostra traçados de curvas em 2 dimensões.
**graf2d2:** mostra traçados de curvas em 3 dimensões.
**xfourier:** demonstra séries de expansão de Fourier.
**truss:** inclinações da estrutura de uma ponte.
**xpquad:** demonstra deformações horizontais e verticais num paralelepípedo.
**wrldtrv:** mostra a rota entre localidades do globo terrestre.
**xplang:** dá uma introdução à linguagem de programação do Matlab.

Existem muitos outros tipos de demonstrações deste tipo que podem ser encontrados na bibliografia anexa.

## 22.– SÓLIDO EM REVOLUÇÃO

O Matlab tem uma função chamada **makevase** que ativa uma janela denominada *Making a Vase*(fazendo um vaso), proporcionando a possibilidade de gerar sólidos em revolução. Na janela, que usa as mesmas características de confecção de um vaso de barro, tem as opções**:**

Assim que é digitado **makevase** na linha de comandos, você deve clicar na opção **New Shape**, logo aparecerá na primeira janela uma linha vermelha que representará o centro de rotação.

Você deverá, em seguida, ir clicando com o botão **esquerdo** do mouse fazendo o contorno da sua figura sendo que, o último ponto deverá ser feito com o botão **direito** do mouse, aparecendo então a figura na tela.

**New Shape:** deve ser acionado("clicado") para se fazer uma nova figura(molde).

**Comment Window:** é uma janela que fornece as instruções(passos) para confecção da superfície em revolução.

**Info:** abre uma janela de ajuda sobre como usar a função **makevase**.

**Close:** fecha a janela relativa à função **makevase**.

## 23 – BIBLIOGRAFIA


HANSELMAN, D. & LITTLEFIELD, B. *The student edition of MATLAB: version 5, user's guide*/ The Math Works, INC, Prentice – Hall**:** New Jersey, 1997.

CHAPMAN,Stephen, Programação em Matlab para Engenheiros. Thomson.2002

Universidade São João Del Rei