INTRODUÇÃO AO MATLAB - PARTE 2

Murilo F. Tomé - ICMC-USP

- Operadores Relacionais/Lógicos
- Estruturas de Seleção e Repetição
- Introdução polinômios
- Gráficos

Operadores relacionais

Símbolo	**Operador**
$==$	igual
$\sim=$	diferente
$>$	maior
$>=$	maior ou igual
$<$	menor
$<=$	menor ou igual

Operadores lógicos

Símbolo	**Operador**
& &	E
||	OU
&	E (escalar)
|	OU (escalar)
$\sim$	Não
xor	OU exclusivo

Exemplos

```
>> 3+3 == 6
   ans =
        1 (true)

>> 3*6 == 15
   ans =
        0 (false)

>> v = [1 3 5]
     v =
        1 3 5
>> v >= w
   ans =        0 0 0
```

```
>> A = [1 2 3; 4 5 6]
   A =
      1 2 3
      4 5 6
>> A >= 3
   ans =
      0 0 1
      1 1 1
w = [2 4 6]
   w =
      2 4 6
>> v <= w
      ans =        1 1 1
```

Estruturas de Controle

São estruturas que definem a ordem com que um grupo de comandos são executados em um programa de computador. Existem 3 tipos básicos dessas estruturas: sequencial, seleção , repetição .

Em MATLAB, esses comandos são semelhantes aos encontrados na maioria das linguagens de programação (ex. C, C^{++}, Fortran, etc).

Comando if

Esse comando avalia uma expressão lógica áqual se for verdadeira uma sequencia de comandos é executada; caso contrário, executa-se uma outra sequencia de comandos.

```
if (cond1)
    comandos 1
elseif (cond2)
    comandos 2
else
    comandos 3
end
```

```
if (cond)
    comandos
end
```

if-Exemplo1

```
≫ A = rand(1)
A =       0.1270
≫ B = rand(1)
B =
      0.9134
≫ if (A>B)
      disp('A é maior que B')
else
      disp('B é maior que A')
end
B é maior que A
```

```
≫ A = rand(1);

≫ B = rand(1);


≫ if (A>B)
   disp('A é maior que B')
else
   disp('B é maior que A')
end;
B é maior que A'
```

if-Exemplo2

```
≫ x=rand(1)
x =
      0.6324
≫ if( (x>=0.2) && (x<=0.7))
disp('x esta entre 0.2 e 0.7')
end
x esta entre 0.25 e 0.75
```

```
≫ x=rand(1)
x =
      0.0975
≫ if( (x>=0.2) && (x<=0.7))
disp('x esta entre 0.2 e 0.7')
end
```

Estruturas de repetição

Comando for: permite que um ou um grupo de comandos sejam executados.

```
for variavel = expressão
    comandos
end
```

Exemplos:

```
>> for i=1:5
     v(i) = i;
     w(i) = 3.0*v(i);
   end
```

```
>> for i=5:-1:1
     v(i) = i;
     w(i) = 3.0*v(i);
   end
```

Comando while

Comando while: permite que um ou um grupo de comandos sejam executados enquanto uma condição for verdadeira.

```
while (condição)
    comandos
end
```

Exemplo - while

```
>> x = 0.1;
>> y = 0.1;
>> while (x+y < 1)
       x = rand(1)
       y = rand(1)
   end
        x = 0.3634
        y = 0.4637
        x = 0.0229
        y = 0.7045
        x = 0.7788
        y = 0.4558
```

Comandos auxiliares no controle de fluxo

- input - recebe dados através do teclado. Esses dados podem ou não ser armazenados em um variável.
- break - comando utilizado para interromper um loop controlado por um for ou while.
- pause - interrompe a execução do programa até que um tecla seja pressionada.

Polinômios

Os polinômios são representados por um vetor, cujos coeficientes das potências em ordem decrescente são os elementos do vetor.

Exemplos:

```
>> p2 = [1 1 1]
   p2 =
      1 1 1
```

define o polinômio $p_2(x) = x^2 + x + 1$

```
>> p3 = [3 -5 -1 1]
   p3 =
      3 -5 -1 2
```

define o polinômio $p_3(x) = 3x^3 - 5x^2 - x + 2$

Raizes dos polinômios

A funcão roots(p) retorna um vetor coluna que contém as raizes do polinômio p.

Exemplos:

```
≫ r2 = roots(p2)
  r2 =
    -0.5000 + 0.8660i
    -0.5000 - 0.8660i
```

```
≫ r3 = roots(p3)
  r3 =
    1.7483
   -0.4794
    0.3978
```

Avaliação de polinômios

Para avaliar um polinômio em um ou mais pontos, pode-se utilizar a função polyval.

Como entrada, o vetor com os coeficientes do polinômio e outro vetor que define os pontos onde o polinômio é avaliado.

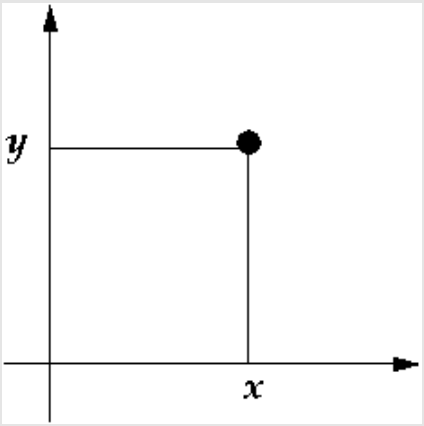
Exemplo:

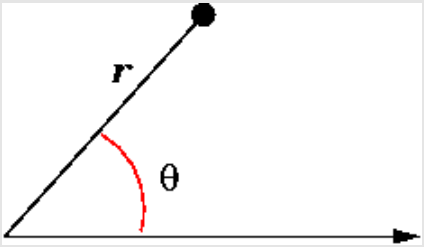
```
≫ pontos = [-1 0 1 5]
    pontos =
      -1 0 1 5
≫ polyval(p2,pontos)
    ans =
        1 1 3 31
```

```
≫ pontos = [-1 0 1 5]
    pontos =
      -1 0 1 5
≫ polyval(p3,pontos)
    ans =
     -6 1 -2 246
```

Gráficos 2D usando MATLAB

Os gráficos 2D podem ser obtidos usando coordenadas cartesianas ou polares.





Função linspace

Essa função gera vetores igualmente espaçados.

- Sintaxe:

  x = linspace(a,b) gera um vetor com 100 elementos igualmente espaçados entre **a** e **b**.

  x = linspace(a,b,n) gera um vetor com n elementos igualmente espaçados entre (e incluindo) **a** e **b**. Se $n < 2$ a função retorna somente **b**.

  $\gg$ p = linspace(1,10); cria um vetor com 100 elementos entre 1 e 10

  $\gg$ p = linspace(1,10, 20); cria um vetor com 20 elementos entre 1 e 10

Comandos para gerar gráficos

- **plot(x, y)**: Gera gráficos lineares (pontos ligados por retas) sendo **x** a variável independente e **y** a variável dependente. Normalmente são vetores contendo as coordenadas dos pontos (x_i, y_i).
- **plot(x, y, z, w)**: Plota dois gráficos (ou mais, dependendo do número de argumentos). sendo **x** a variável independente e **y** a variável dependente. Normalmente são vetores contendo as coordenadas dos pontos (x_i, y_i). O tipo da linha difere de uma curva para outra.

Comandos para gerar gráficos 2

- **semilogx(x, y)**: Gera gráficos com valores de x na escala logarítmica.
- **semilogy(x, y)**: Gera gráficos com valores de y na escala logarítmica.
- **log logy(x, y)**: Gera gráficos com valores de x e y na escala logarítmica.

Exemplos de Gráficos usando MATLAB

$\gg$ x = [0.0 0.5 1.0 1.5 2.0 2.5];

$\gg$ y = [1.0 1.5 1.8 2.0 1.9 1.5];

$\gg$ z = [0.0 0.5 1.0 1.5 2.0 2.5];

$\gg$ w = [2.0 1.7 1.3 1.4 1.6 1.9];

$\gg$ plot(x,y,z,w)

plot(x,y,z,w)

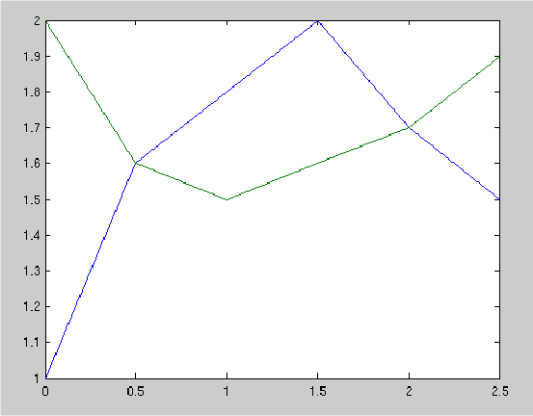


Figure 1: Curva azul: gráfico de (x,y) - Curva verde: gráfico de (z,w).

plot sen(x)

$\gg$ x1 = linspace(0,pi,10)

x1 =

0 0.3491 0.6981 1.0472 1.3963 1.7453 2.0944 2.4435 2.7925 3.1416

$\gg$ x2 = linspace(0.0,pi, 20);

$\gg$ plot(x1, sin(x1)) ver gráfico a seguir

$\gg$ plot(x2, sin(x2))

plot sen(x)

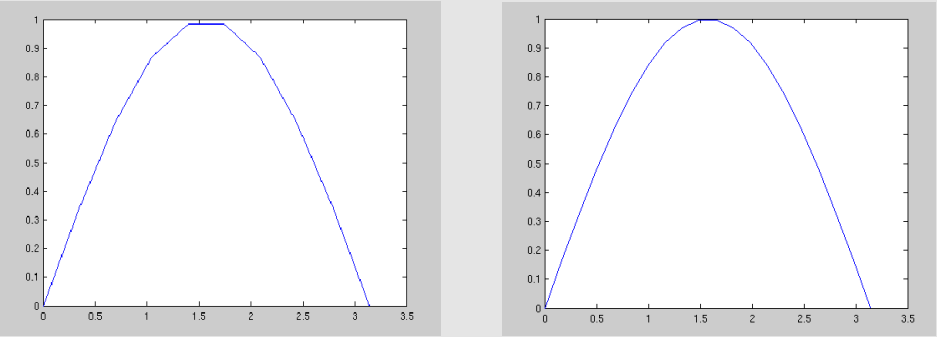
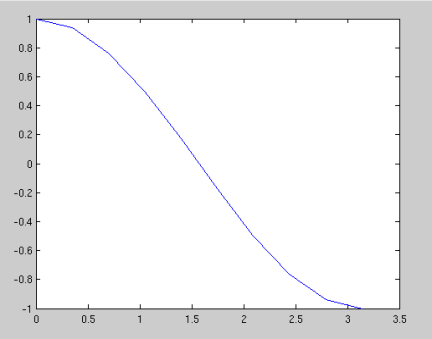


Figure 2: Gráfico de sen(x) com 10 pontos (a esquerda) e 20 pontos (a direita.

plot cos(x)



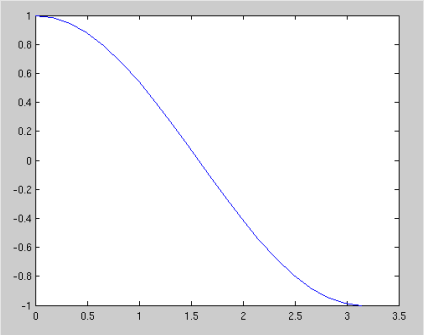


Figure 3: Gráfico de cos(x) com 10 pontos (a esquerda) e 20 pontos (a direita.

Comandos complementares

- **title('texto')** - Comando para adicionar um título (texto) no topo do gráfico.
- **xlabel('texto')** - escreva uma legenda no eixo x.
- **ylabel('texto')** - escreva uma legenda no eixo y.
- **text(x,y,'texto')** - Coloca um texto no ponto (x,y). Se x e y são vetores, o texto é colocado em cada posição (x_i, y_i).
- **legend('texto1', 'texto2', ...)** - Coloca legendas nos gráficos, na ordem em que são plotados no canto superior direito da figura.

Comandos complementares - cont.

- **legend('texto1', 'texto2', 'location', 'pos')** - Coloca legendas nos gráficos, na posição indicada por 'pos' que podem ser: north, south, east, wesst, northeast, northwest, southwest, southeast.
- **grid on** - Coloca uma grid no gráfico plotado.
- **grid off** - retira as grades do gráfico atual.
- **hold on** e **hold off** - Comando utilizado para plotar outro gráfico na mesma figura. O gráfico antigo mantém-se até que o comando **hold off** seja usado. Utilizado para sobrepor gráficos.

Comandos complementares - cont.

- **plot(x,y,'parâmetro')** - **'parâmetro'** é uma combinação de características, tipo de linha, de marcador, cor, que se deseja aplicar ao gráfico. Não há ordem para especificação desses parâmetros e não é necessário especificar os três.

Table 1: Opções do comando **plot**

Cores		**Linhas**		**Marcador**	
amarelo ●	y	sólida	-	ponto	.
azul ●	b	tracejada	**- -**	quadrado	s
ciano ●	c	traço-ponto	-.	círculo	o
branco ●	w	ponteada	:	+	+
vermelho ●	r			x	x
preto ●	k			estrela	*
roxo ●	m			triângulo	∧
verde ●	g			triângulo invertido	∨

.

Exemplo de gráfico

```
>> x = linspace(0,2.0*pi, 20);
>> y1 = sin(x);
>> y2 = cos(x);
>> hold on
>> plot(x,y1,'r-s'); % cor 'red' (r), 'solid line' (-), '□'  (s)
>> plot(x,y2,'b-.*'); % cor 'blue' (b), 'dotted line' (-), '*'  (*)
>> grid
>> xlabel('eixo x');
>> ylabel('eixo y');
>> title ('Grafico do seno e cosseno');
>> legend('sen(x)', 'cos(x)');
```

Exemplo de gráfico -2

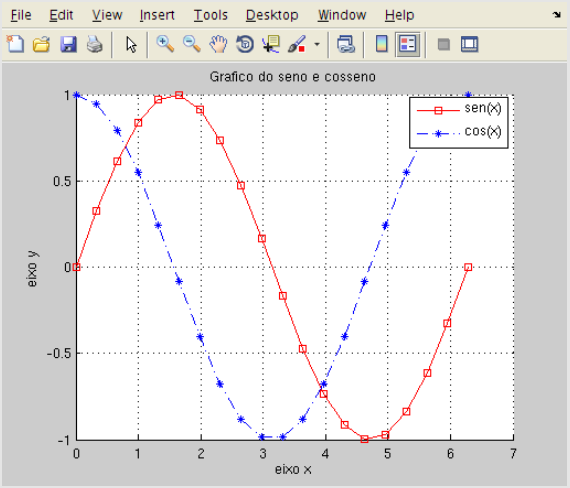


Figure 4: Gráfico de cos(x) e cos(x).