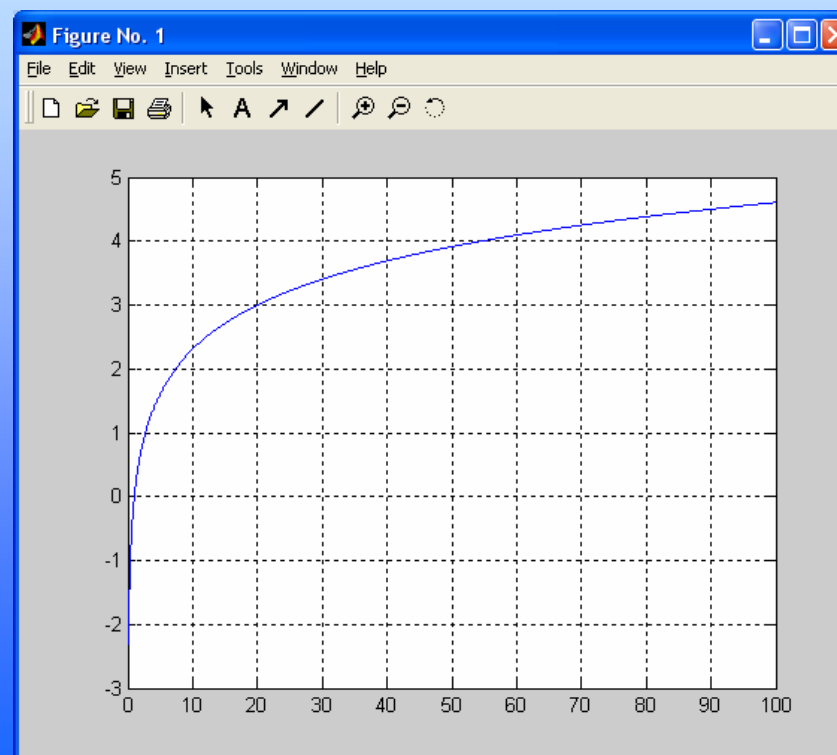


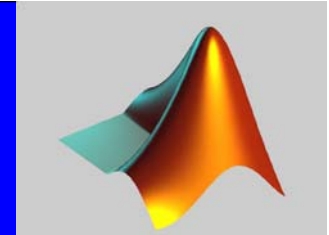
Observação em relação a aula passada:

➤ Gráfico da Função: $y = f(x) = \ln(x) / x$?

$$f : \mathbb{R}_+^* \rightarrow \mathbb{R}$$

$$x \rightarrow \ln(x)$$





Observação em relação a aula passada:

No MATLAB a função que expressa o logaritmo natural é dado pelo símbolo $\log(x)$.

```
>> help log
```

LOG Natural logarithm.

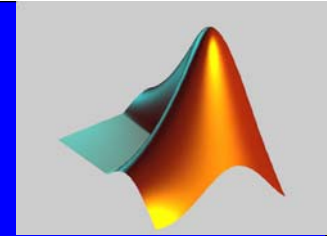
LOG(X) is the natural logarithm of the elements of X.

Complex results are produced if X is not positive.

LOG Logaritmo natural.

LOG(X) é o logaritmo natural do argumento X.

Resultados complexos são produzidos, se X não for positivo.



Observação em relação a aula passada:

```
>> log(0)
```

Warning: Log of zero.

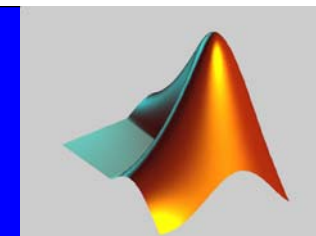
```
ans = -Inf
```

```
>> log(-2)
```

```
ans = 0.6931 + 3.1416i
```

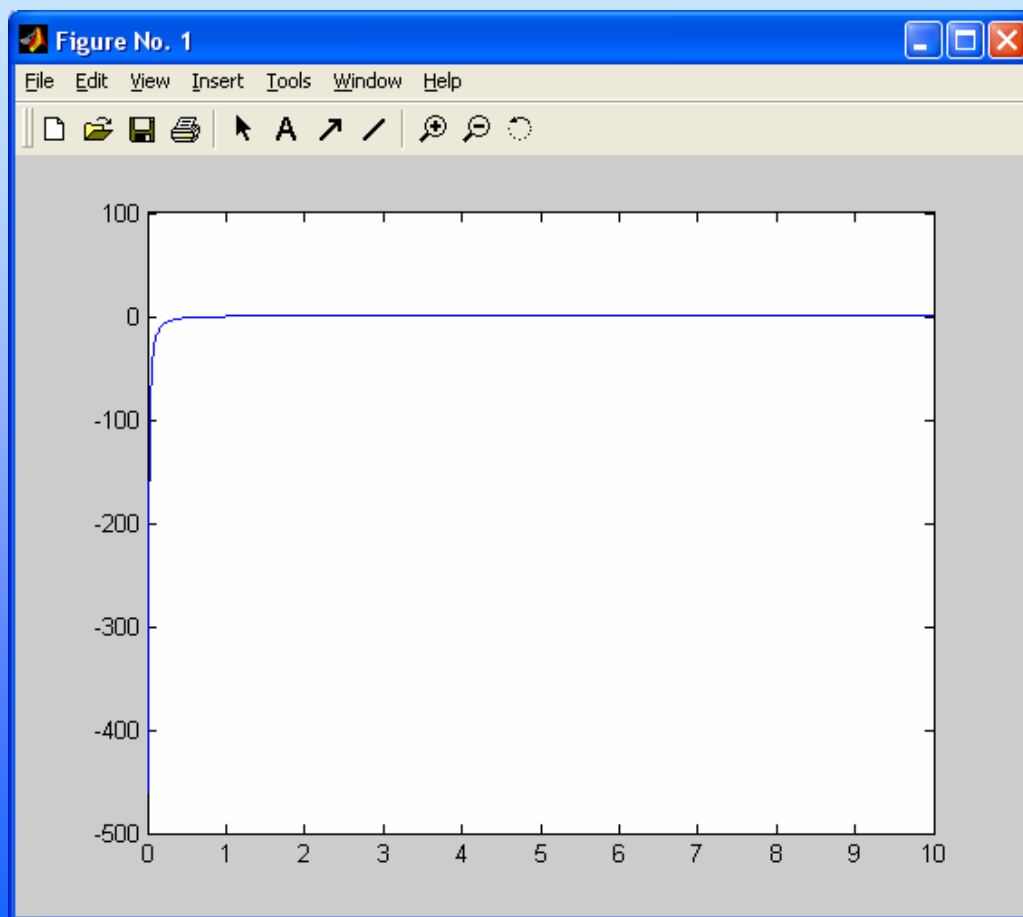
```
>> log(-pi)
```

```
ans = 1.1447 + 3.1416i
```

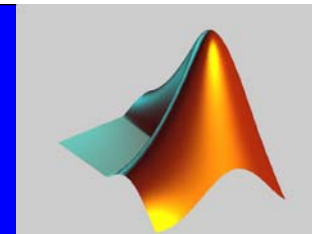


Observação em relação a aula passada:

➤ **Gráfico da Função:** $y = f(x) = \ln(x) / x$

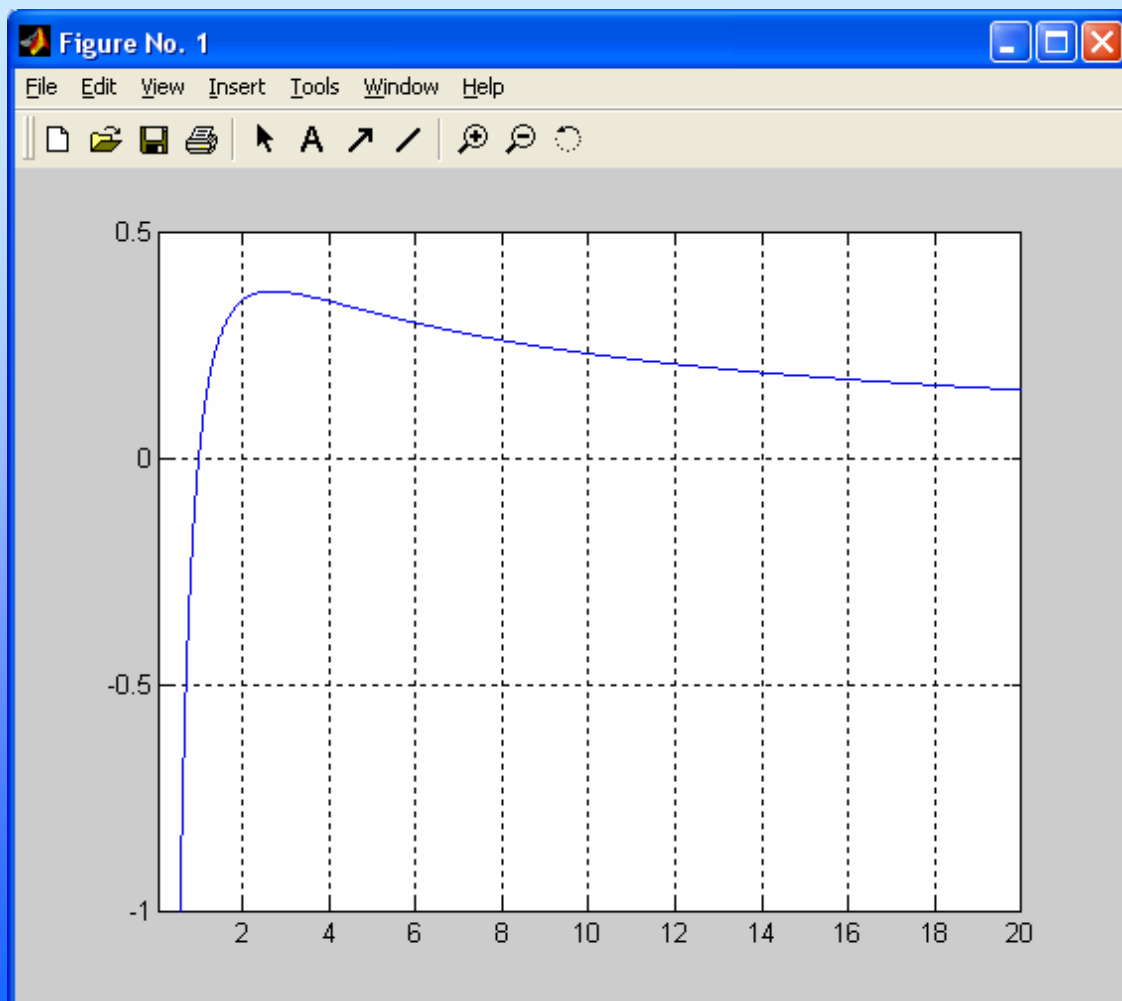


Observem que a escala no eixo-y pode nos induzir a erros de avaliação em relação ao gráfico da função.

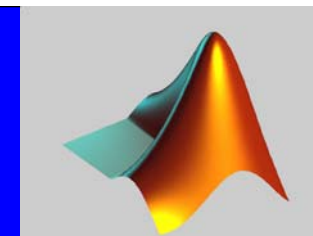


Observação em relação a aula passada:

$$y = f(x) = \ln(x) / x$$



Observem a escala
no eixo-y.



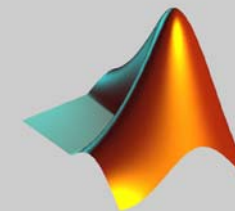
GRÁFICOS BIDIMENSIONAIS (com eixos em escalas lineares):

➤ Gráficos de Funções de uma Variável ($y = f(x)$);

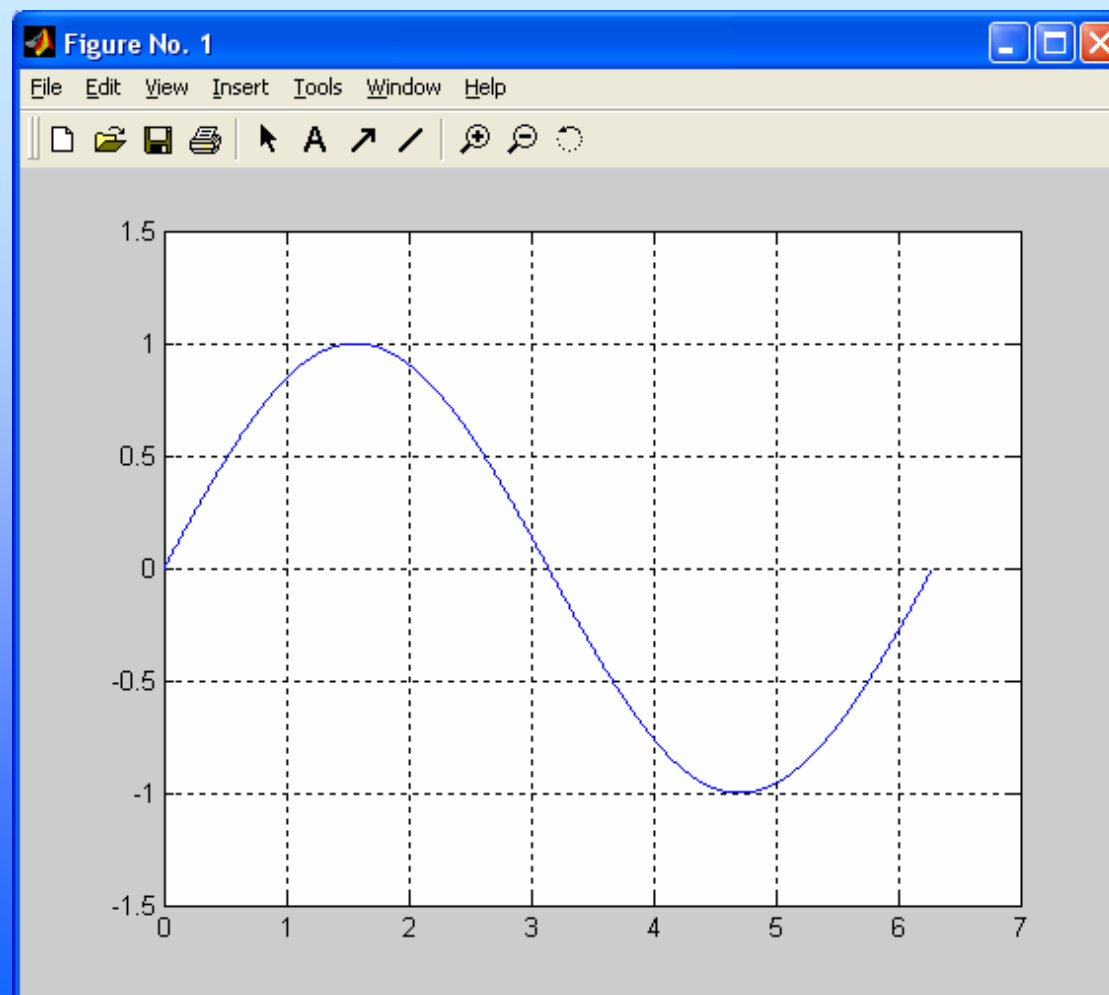
$$y = \sin(x), \quad 0 \leq x \leq 2\pi$$

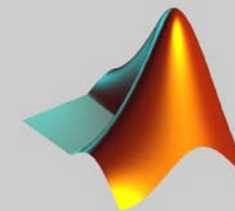
$$y = \cos(x + \sqrt{2}) + x\left(\frac{x}{2} + \sqrt{2}\right), \quad x \in [-2, -1]$$

$$y = x^3 - 3x^2(-2^{-x}) + 3x(4^{-x}) - 8^x, \quad x \in [0, 1]$$

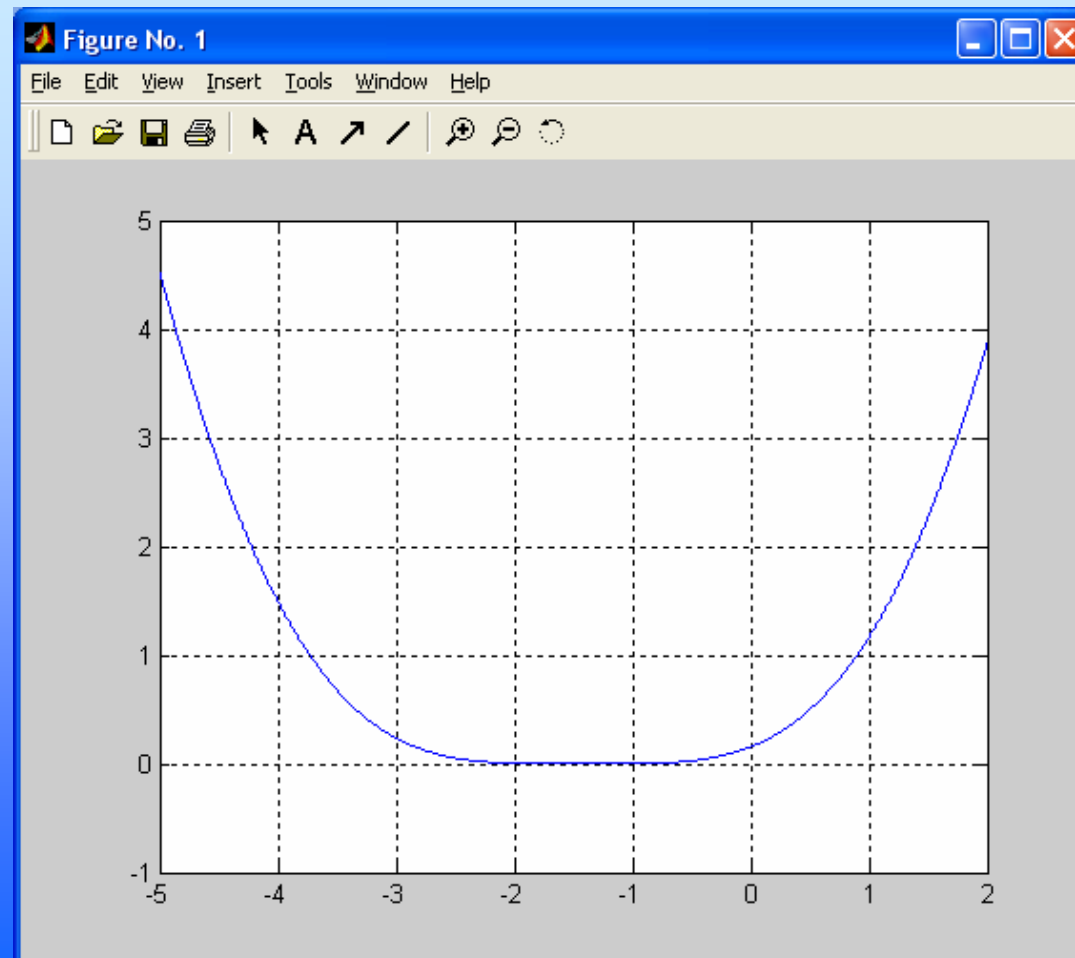


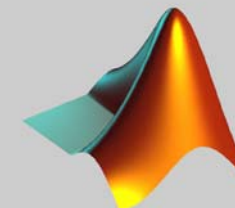
$$y = \sin(x), \quad 0 \leq x \leq 2\pi$$



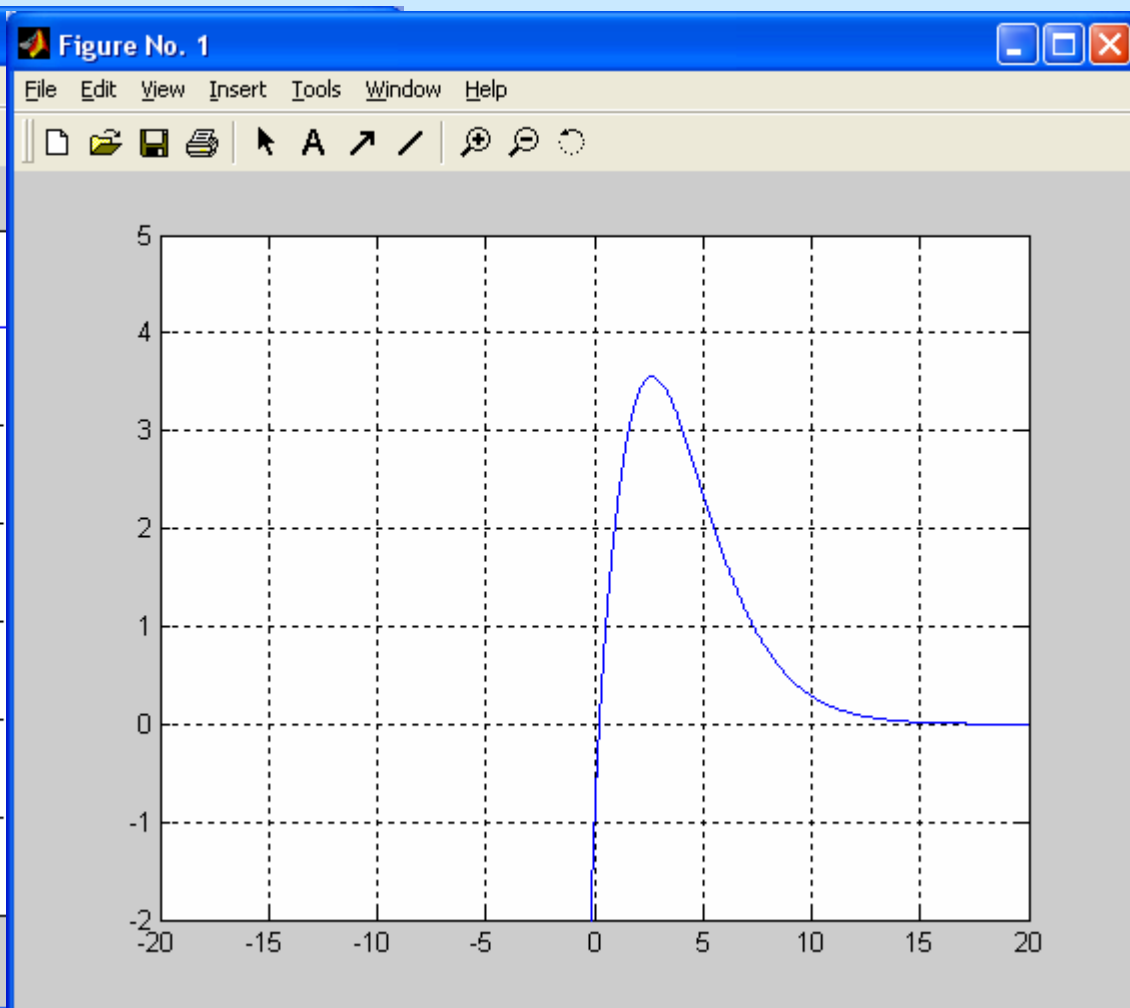
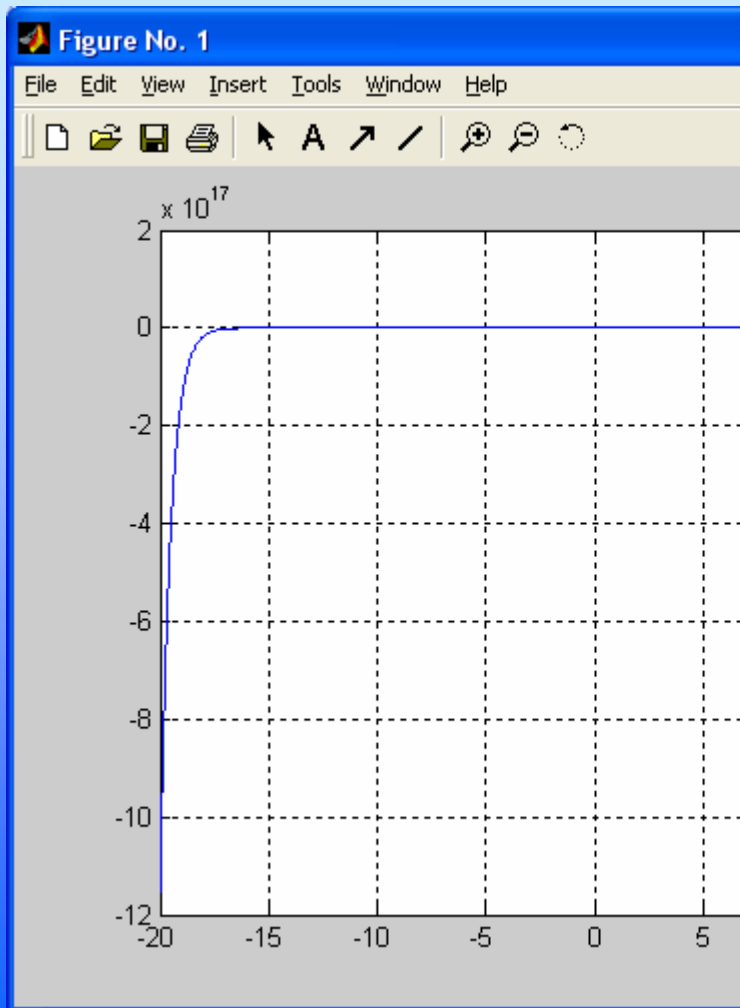


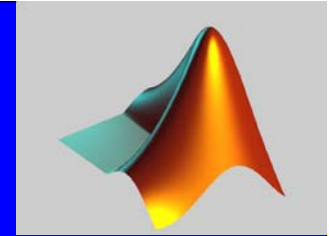
$$y = \cos(x + \sqrt{2}) + x\left(\frac{x}{2} + \sqrt{2}\right), \quad x \in [-5, -2]$$





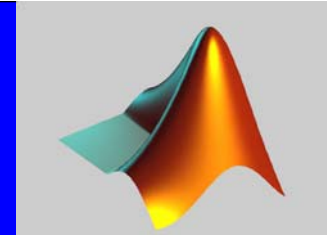
$$y = x^3 - 3x^2(-2^{-x}) + 3x(4^{-x}) - 8^x, \quad x \in [-20, 20]$$





ATENÇÃO QUANDO USAREM O MATLAB PARA VISUALIZAREM GRÁFICOS DE FUNÇÕES!!!

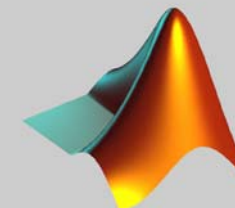
A escala no eixo-y pode nos induzir a erros de INTERPRETAÇÃO.



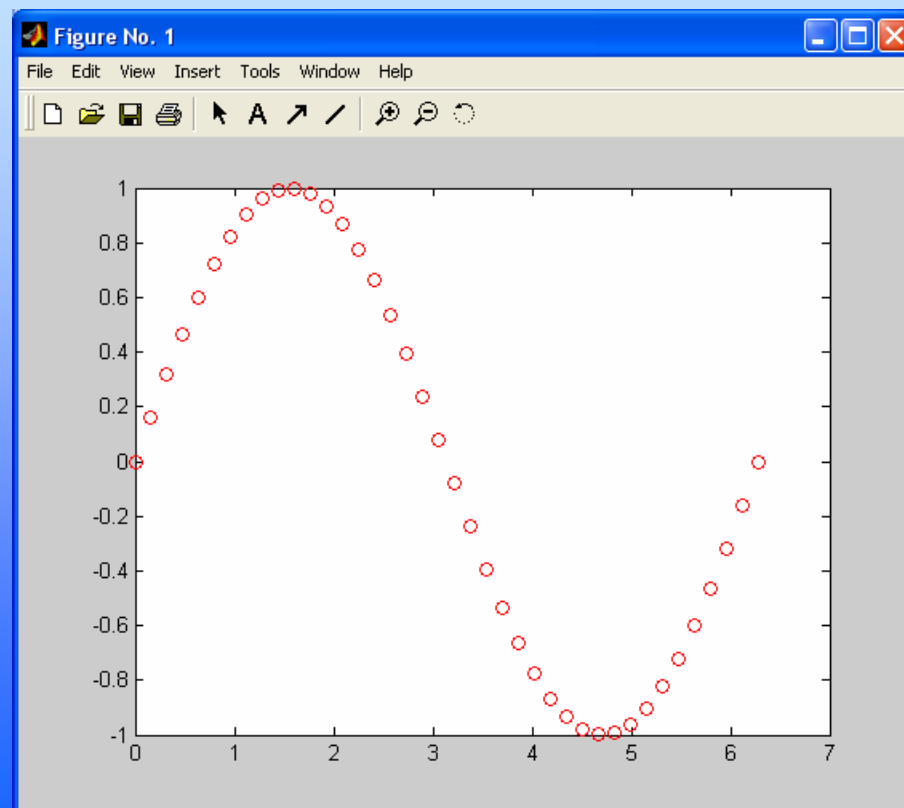
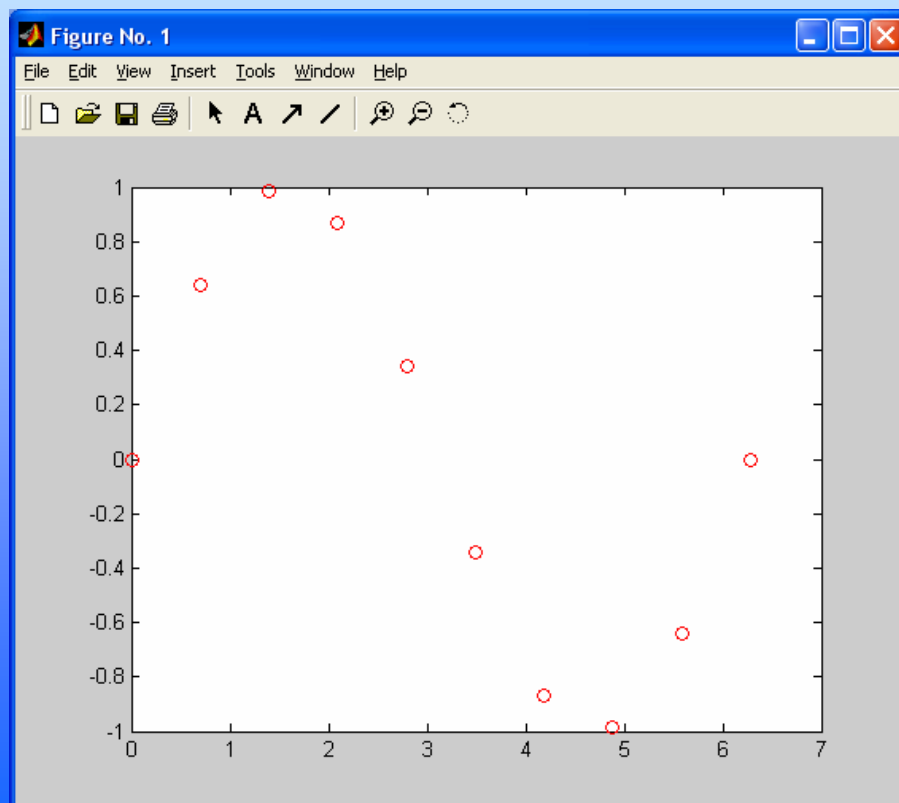
GRÁFICOS BIDIMENSIONAIS (com eixos em escalas lineares):

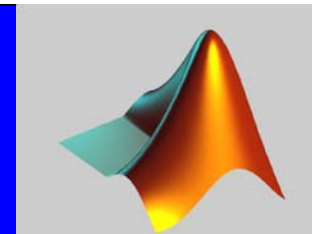
Problema: Vamos imaginar que desejamos gerar/desenhar o gráfico da função $\text{sen}(x)$ no intervalo $[0, 2\pi]$.

Solução: Sabemos que no intervalo $[0, 2\pi]$ há infinitos valores no *eixo-x*. Uma **alternativa**, seria gerar um conjunto finito de valores nesse eixo, e, em seguida, obter o valor da função $f(x)$ em cada um destes pontos. Com isso, vamos obter um conjunto de pontos no *plano-xy* e, assim, podemos conseguir uma “aproximação” da curva pretendida unindo pares de pontos consecutivos (segundo a abscissa), através de segmentos de retas.

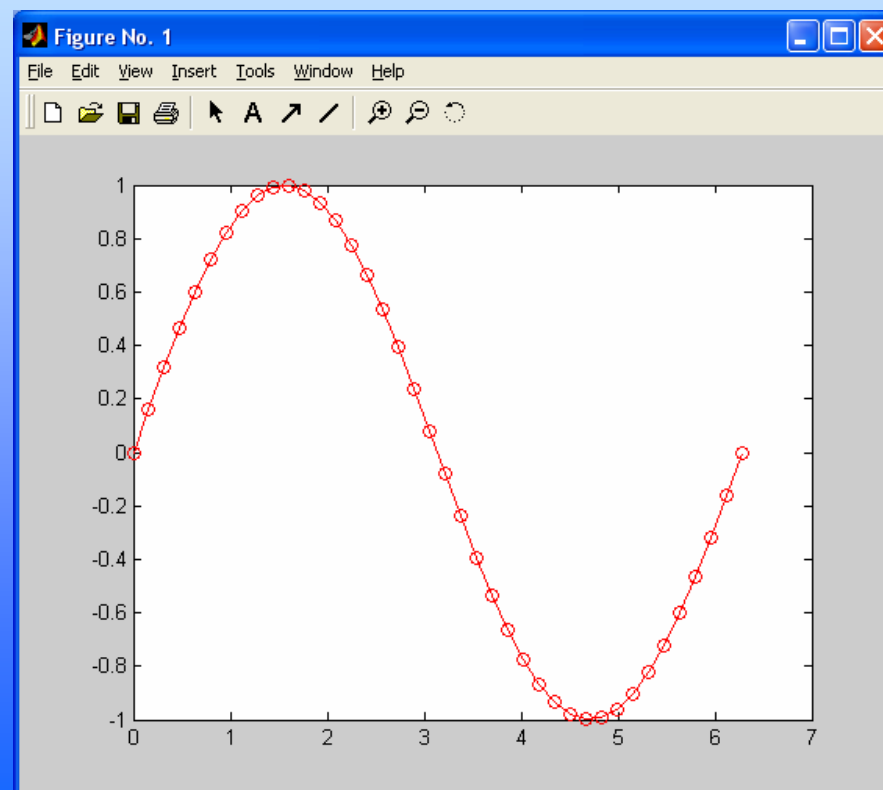
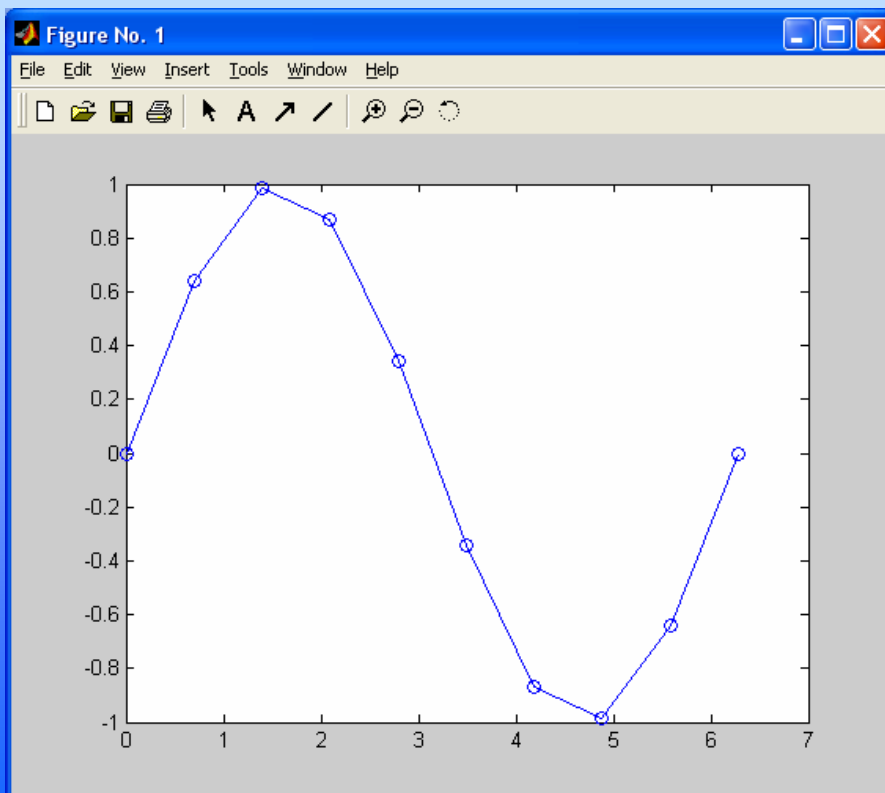


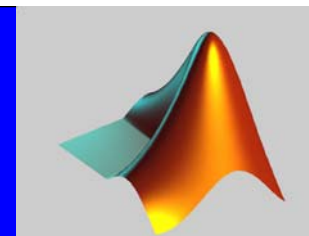
Exemplo: Na figura da esquerda foram gerados 10 pontos, enquanto na figura da direita foram gerados 40 pontos. *Qual delas aproxima melhor a curva?*



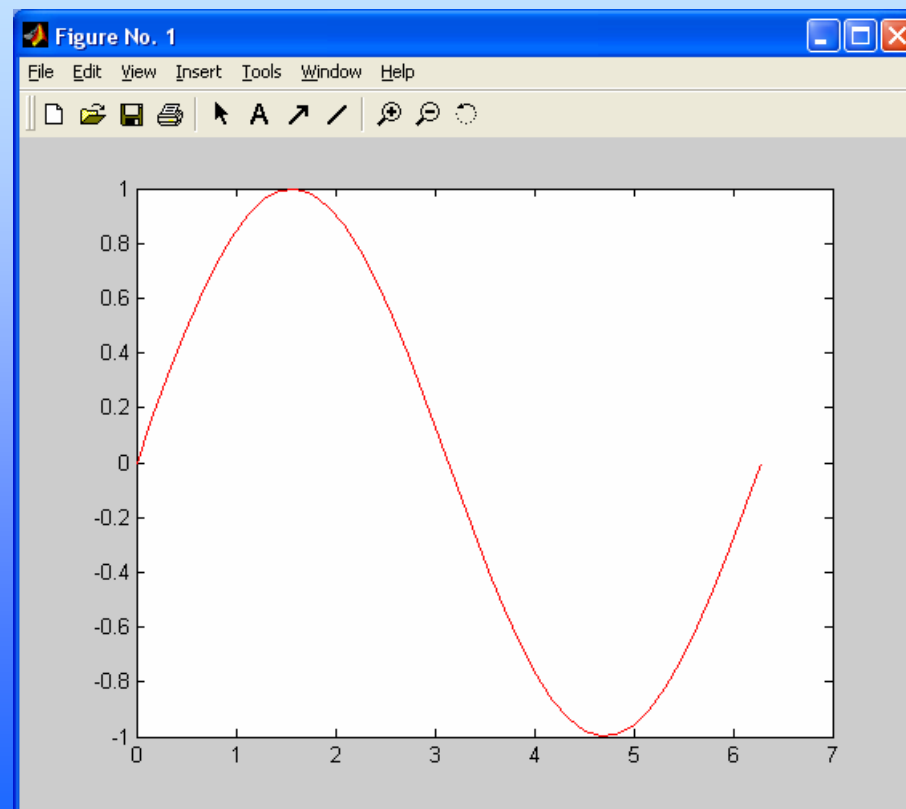
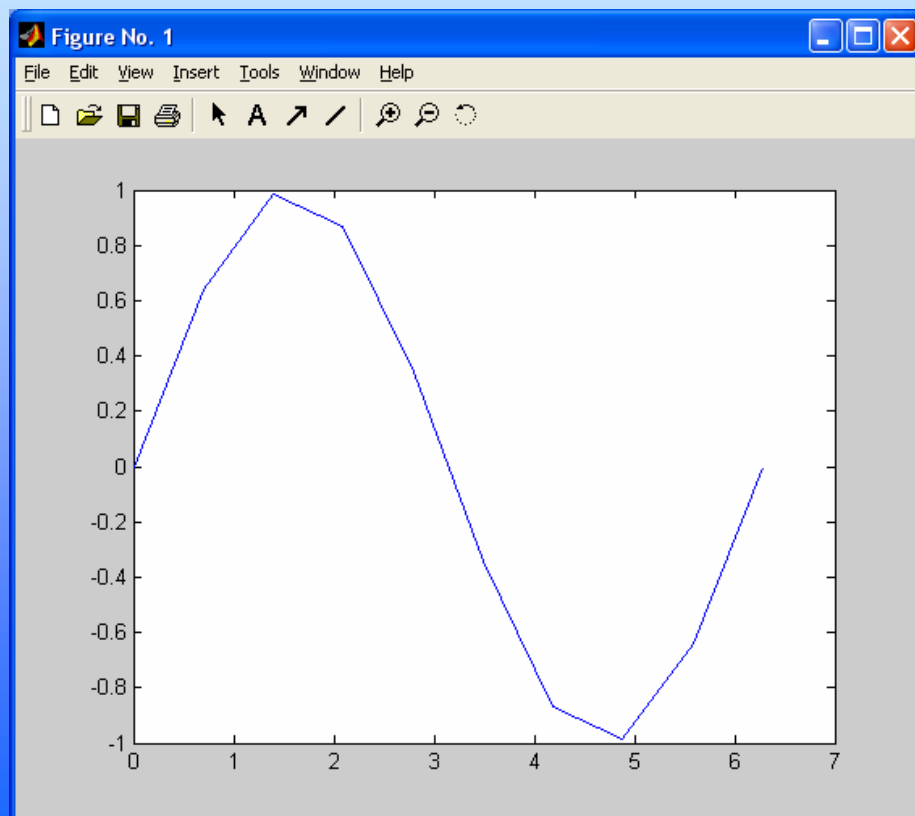


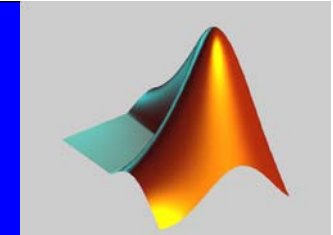
Exemplo: Unindo pares (x,y) de pontos consecutivos no plano (*segundo a abscissa*), por meio de segmentos de retas, iremos obter respectivamente, as seguintes *aproximações* da curva pretendida para os conjuntos definidos anteriormente. **Qual delas aproxima melhor a curva?**





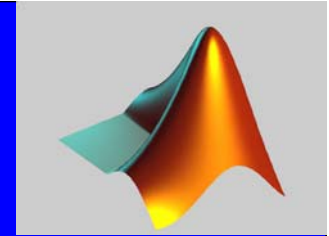
Exemplo: Curvas *aproximadas* para os respectivos conjuntos de pontos, sem os marcadores sobre os pontos. **Qual delas aproxima melhor a curva?**





Conclusões Básicas (intuitivas) :->

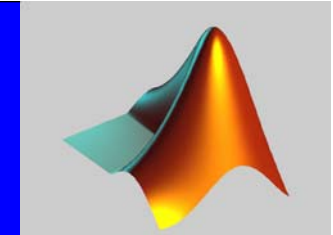
- Quanto mais próximos forem os pontos, mais precisa (fidedigna) será a representação da função que estamos visualizando;
- Isto implica que devemos fazer uma boa *amostragem* inicial, ou seja, escolher um número significativo de pontos;
- Observe que nas regiões em que a curvatura é mais acentuada, um número maior de pontos é colocado automaticamente pelo **MATLAB**;
- Na verdade, existem outras formas matemáticas de se obter a *aproximação* de uma curva ou de uma superfície. Uma abordagem mais profunda sobre este assunto será visto na disciplina de Cálculo Numérico (3º ano).!
- Por ora, o importante é saber que o **MATLAB** desenha gráficos bidimensionais através dos recursos discutidos acima.



Constatação :→ Vimos que é importante gerar uma quantidade significativa de dados, muitas vezes, algumas centenas deles, o que torna o processo extremamente moroso e monótono.

Questão :→ *Será que existe uma forma automática de gerar os dados?*

Resposta :→ *Felizmente sim. Na verdade existem duas formas básicas.*



A rigor, deveríamos introduzir o conceito de **vetores** antes, mas os recursos que precisamos são poucos e assim, podemos avançar o mais rápido possível para o traçado (desenho) de funções.

Comando `linspace(.)` :→ Cria um número especificado de pontos igualmente espaçados num dado intervalo.

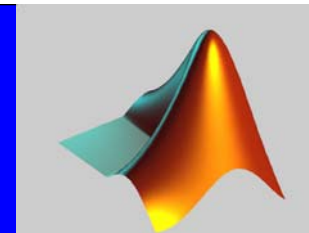
Forma Geral: `>>linspace(valor_inicial, último_valor, número_de_valores)*cte`

(Primeira Forma)

Observações:

→Através do uso deste comando, não é possível especificar o incremento entre os pontos. O MATLAB calcula automaticamente;

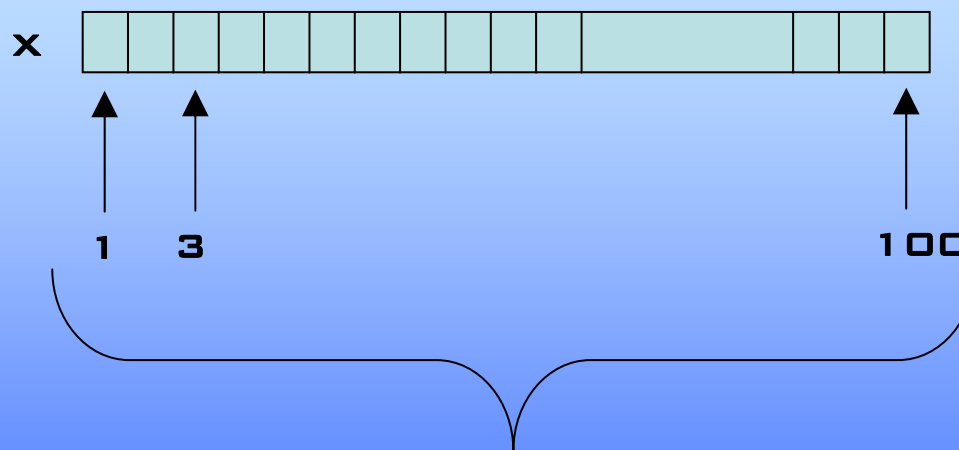
→Os valores gerados podem ser associados a uma variável estruturada do tipo vetor através do comando de atribuição. Se a constante *cte* não for especificada, o MATLAB a interpreta com o valor 1 (*pontos igualmente espaçados*).



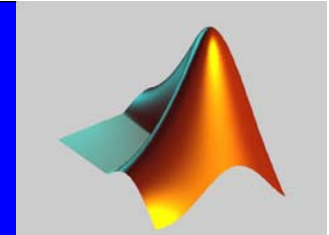
Primeira Forma :

```
>>x = linspace(1,10,100); % gera 100 pontos igualmente espaçados no intervalo [1,10] e atribui estes valores a variável x.
```

Variável estruturada de nome **x** que armazena todos os valores gerados.

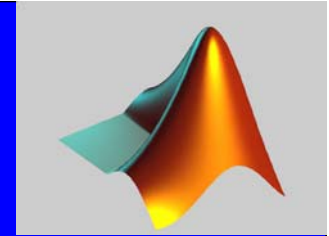


Cada componente do vetor **x** está associado a uma única posição que é determinada por um índice que varia de 1 ao número máximo



Exemplos:

```
>>a = 2*pi; % a variável simples x recebe o valor 2*pi.  
>>x = linspace(0,2*pi,10); % gera 10 pontos igualmente espaçados no  
intervalo fechado [0,2π] e associa estes valores a  
variável estruturada x.  
  
>>x = linspace(-2*pi,2*pi,60); % gera 60 pontos igualmente espaçados no  
intervalo fechado [- 2π,2π] e associa estes valores  
a variável x.
```

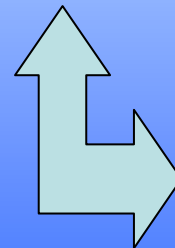


Observação:

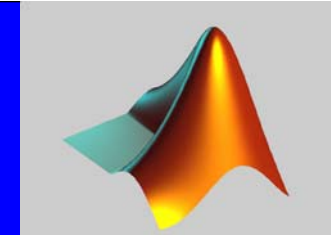
→ É possível acessar e manipular o conteúdo de cada uma das componentes da variável estruturada (vetor) \mathbf{x} , de forma independente;

Forma Geral de como acessar uma componente de um vetor:

NomeVariável (i)



i: indica o índice (posição) da componente que se deseja acessar.



Exemplos:

```
>>x = linspace(-2*pi,2*pi,60);
```

gera 60 pontos igualmente espaçados no intervalo fechado $[-2\pi, 2\pi]$ e associa estes valores a variável x .

As seguintes instruções são válidas:

```
>>x(33)
```

Visualizando o conteúdo (valor) da variável estruturada de nome x , na posição 33.

```
>>b = x(12)+x(47)*sqrt(x(21))
```

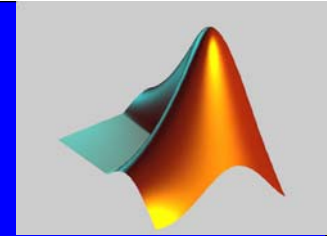
Usando os valores das componentes numa expressão aritmética.

```
>>i = 34;
```

```
>>j = 53;
```

```
>>a = x(i)*x(j)+sin(x(11))
```

Acessando os valores das componentes de forma indexada (através de um índice).



Uma outra maneira de gerar pontos de forma completamente automática é através do operador dois-pontos ‘:’

Forma Geral:

```
>> x = (valor_inicial : incremento : valor_final)*cte
```

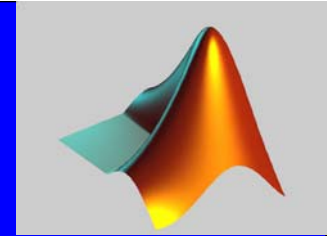
(Segunda Forma)

```
>> x = valor_inicial : incremento : valor_final*cte
```

Cria um conjunto de valores que inicia em *VALOR_INICIAL* e cuja distância entre pontos consecutivos é dada por um *incremento*. Todos os valores gerados no intervalo [*valor_inicial*, *valor_final*] são multiplicados pela constante correspondente.

Observação:

→ O último valor calculado é menor ou igual ao *valor_final*.

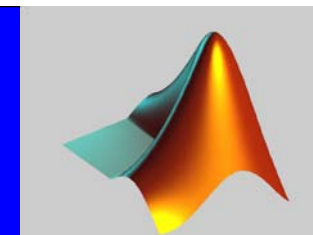


Exemplos: → Geração automática usando o operador ':' (dois-pontos)

>>x = (0:0.1:1)*pi % cada um dos valores gerados no intervalo é multiplicado por pi

>>x = (0:0.001:2*pi); % Inicia a geração no ponto igual a zero com incremento entre os pontos de um fator 0.001. O último número gerado será menor ou igual a 2π .

>>x = (-10:0.001:10); % Inicia a geração no ponto igual a -10 com incremento entre os pontos de um fator 0.001. O último número gerado será menor ou igual a 10.

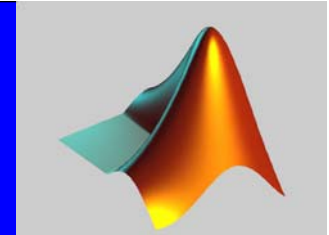


O **MATLAB** possui recursos poderosos para gerar gráficos bidimensionais e tridimensionais, embora nossos interesses aqui estarão restritos ao gráficos bidimensionais.

O comando `plot(.)` é um dos responsáveis pela criação de gráficos bidimensionais no **MATLAB** (há outros). Na verdade, a partir de um conjunto de valores no *eixo-x* (representado por um vetor), o **MATLAB** calcula os respectivos valores no *eixo-y* (um outro vetor com mesma dimensão) e liga os pontos (x,y) consecutivos no plano (segundo a abscissa) através de segmentos de retas.

Todo desenho (saída gráfica) é visualizada numa janela denominada **Janela de Figuras**, que se abre automaticamente toda vez que você executar o comando `plot()`.

O comando `plot()` ajusta os eixos (proporção) automaticamente a partir dos dados de entrada associados ao *eixo-x*. Além disso, coloca uma escala numérica em ambos os eixos.



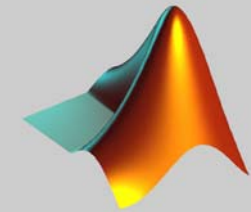
GRÁFICOS DE FUNÇÕES: Comando `plot(.)`

Forma Geral Básica:→

```
>> plot(x,y);
```

Propósito:→ Cria um gráfico no *plano-xy*, na cor azul e com linha contínua, onde: *x* e *y* são vetores de mesma dimensão.

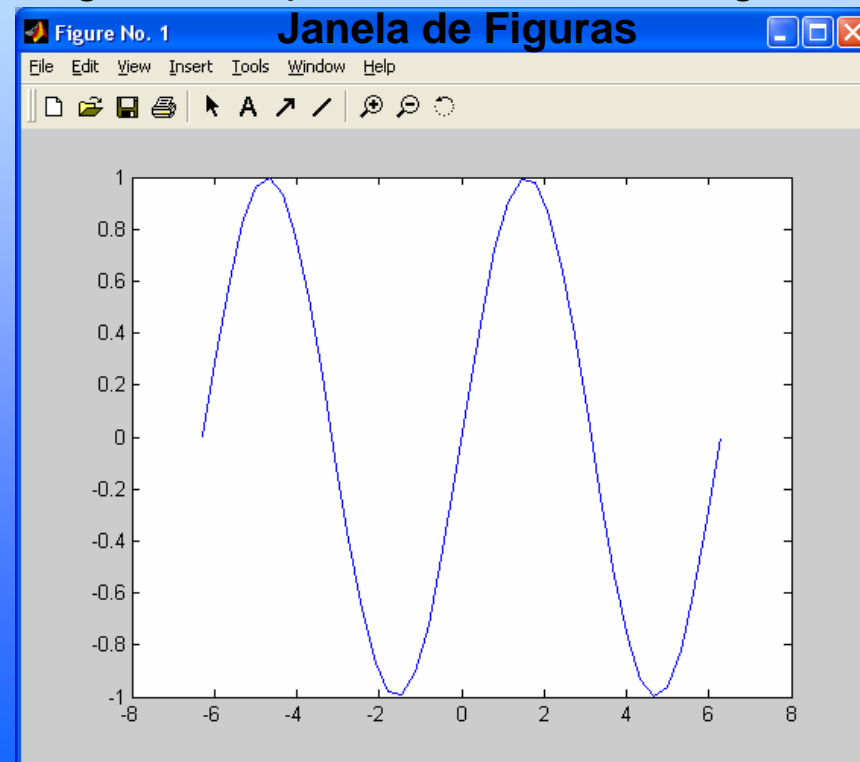
Observação: Toda vez que você executa o comando `plot(.)`, uma nova janela denominada ***Janela de Figuras*** é aberta. Se essa janela já estiver ativa (com outra figura) o **MATLAB** limpa-a e cria o novo gráfico sobre ela.

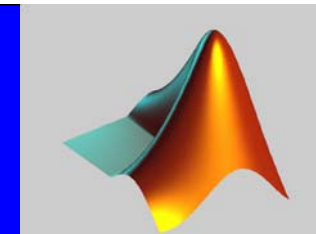


Exemplo- 01 → Gráfico da função seno no intervalo $[-2\pi, 2\pi]$

```
>> x = linspace(-2*pi, 2*pi, 30); % criando 30 pontos no eixo-x (variável x)
>> y = sin(x) % criando 30 pontos no eixo-y a partir de x
>> plot(x, y) % ligando os pontos através de segmentos, ...
```

Resultado após a execução do comando `plot(.)`.





Observações: → Em função do fato de estarmos realizando operações sobre vetores para o traçado de gráficos, muitas vezes é necessário realizar operações elementos a elementos (componente a componente).

Para fazer esta distinção, o **MATLAB** acrescenta o operador ponto (‘.’) antes da operação pretendida.

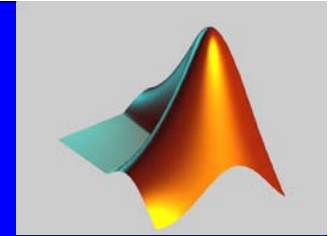
Operadores (elemento a elemento)	Símbolo
multiplicação	.*
divisão	./
potenciação	.^

Exemplo:

Seja $x = (1,2)$ e $y = (2, 3)$,
então

A operação $x.*y$ é igual a $(2\ 6)$ e
a operação $x.^y$ é igual a $(1\ 8)$.

Experimentem ??????



Exemplo: Sejam $x = (1,2,3)$ e $y = (5,6,7)$, dois vetores em \mathbb{R}^3 .

Os vetores acima podem ser gerados, respectivamente, pelos seguintes comandos:

```
>> x = linspace(1,3,3);
```

```
>> y = linspace(5,7,3);
```

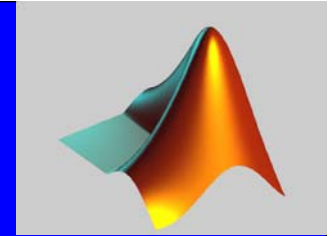
Quais os resultados das operações vetoriais $x.*y$ e $x.^y$?

Experimentem ???????

Solução:

$$x.*y = (5,12,21)$$

$$x.^y = (1,64,2187)$$



Observações: → Os operadores elemento a elemento são muito utilizados para se computar o valores de uma função ($y = f(x)$) nos respectivos pontos no *eixo-x* (eixo independente).

Exemplos: → As funções

$$y = 2 \sin(x) \cos(x)$$

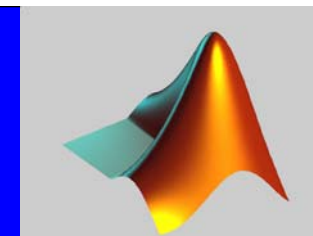
$$y = \sin(x) / (\cos(x) + \textit{eps})$$

devem ser computadas no **MATLAB** da seguinte forma:

```
>> y3 = 2*sin(x).*cos(x);
```

```
>> y4 = sin(x)./(cos(x)+eps);
```

**Multiplicação (.*) e divisão (./)
componente a componente**



Exercícios:→

01: Gerar o gráfico da função anterior com 10, 20, 30, 40, 50 e 100 pontos e verificar a diferença entre os resultados. Tire suas conclusões.

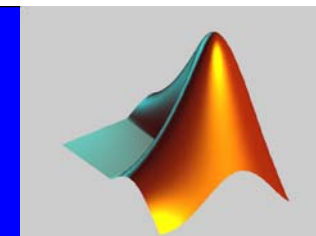
02: Coloque multiplicadores (maiores e menores que 1) no argumento da função e verifique como o período se comporta.

03: Obtenha os gráficos das funções abaixo, nos respectivos intervalos. Após uma geração inicial, sinta-se a vontade para alterar os limites.

$$y = \sin(x)^{\cos(x)}, \quad 0 \leq x \leq 2\pi$$

$$y = \cos(x + \sqrt{2}) + x\left(\frac{x}{2} + \sqrt{2}\right), \quad x \in [-2, -1]$$

$$y = x^3 - 3x^2(-2^{-x}) + 3x(4^{-x}) - 8^x, \quad x \in [0, 1]$$



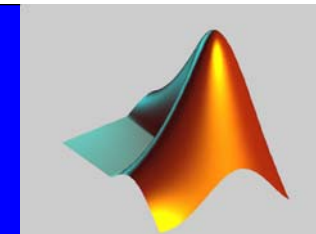
Outras Versões do Comando `plot(.)`

`>>plot(x,y,'cmt')`

cria um gráfico no *plano-xy*, na cor 'c', com marcador 'm' e com tipo de linha 't'.

Símbolo	Cor	
b	azul	
g	verde	
r	vermelho	
c	ciano	
m	magenta	
y	amarelo	
k	preto	
w	branco	

Símbolo	Tipo de Linha
-	contínua
:	pontilhada
-. .	traços e pontos
--	tracejada



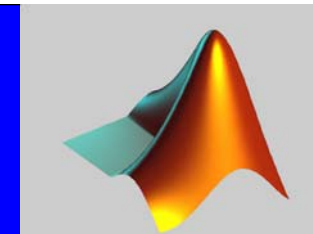
Outras Versões do Comando `plot(.)`

`>>plot(x,y,'cmt')`

cria um gráfico no *plano-xy*, na cor 'c', com marcador 'm' e com tipo de linha 't'.

Símbolo	Marcador	Símbolo	Marcador
.	ponto	p	pentagrama
o	círculo	h	hexagrama
x	x	>	triângulo a direita
+	+	<	triângulo a esquerda
*	estrela	^	triângulo para cima
s	quadrado	v	triângulo para baixo
d	losango		

Observação: → Os marcadores são aplicados somente aos pontos do *plano-xy*. Se não houver a especificação de um tipo de linha, somente os pontos são desenhados.



Outras Versões do Comando `plot(.)`

`>>plot(x,y,'cmt')` cria um gráfico no *plano-xy* com a **cor** 'c', com **marcador** 'm' e com **tipo** de linha 'l'.

Exemplo :→ Desenhando o gráfico de uma função na cor vermelha, com linha pontilhada e com marcadores “estrela” nos pontos de controle gerados do *plano-xy*.

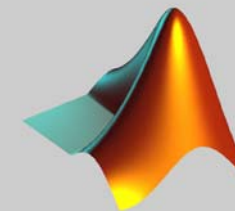
Exemplo:→

```
>> x = linspace(0,2*pi,30);
```

```
>> y = sin(x);
```

```
>> plot(x,y,'r*:') ;
```

OBSERVAÇÃO: Se você não especificar nenhuma cor, o MATLAB inicia o desenho pela cor azul e desenha o gráfico das próximas funções com as cores na ordem em que elas aparecem na tabela.



Desenhando várias funções com atributos diferentes através do comando `plot(.)`, na mesma Janela de Figuras.

```
>>plot( x1,y1,'cmt1',x2,y2,'cmt2',..., xn,yn,'cmtn' )
```

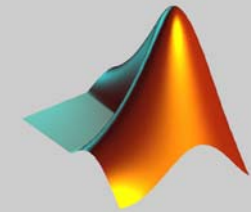
Parâmetros
da Curva 1

Parâmetros
da Curva 2

...

Parâmetros
da Curva n

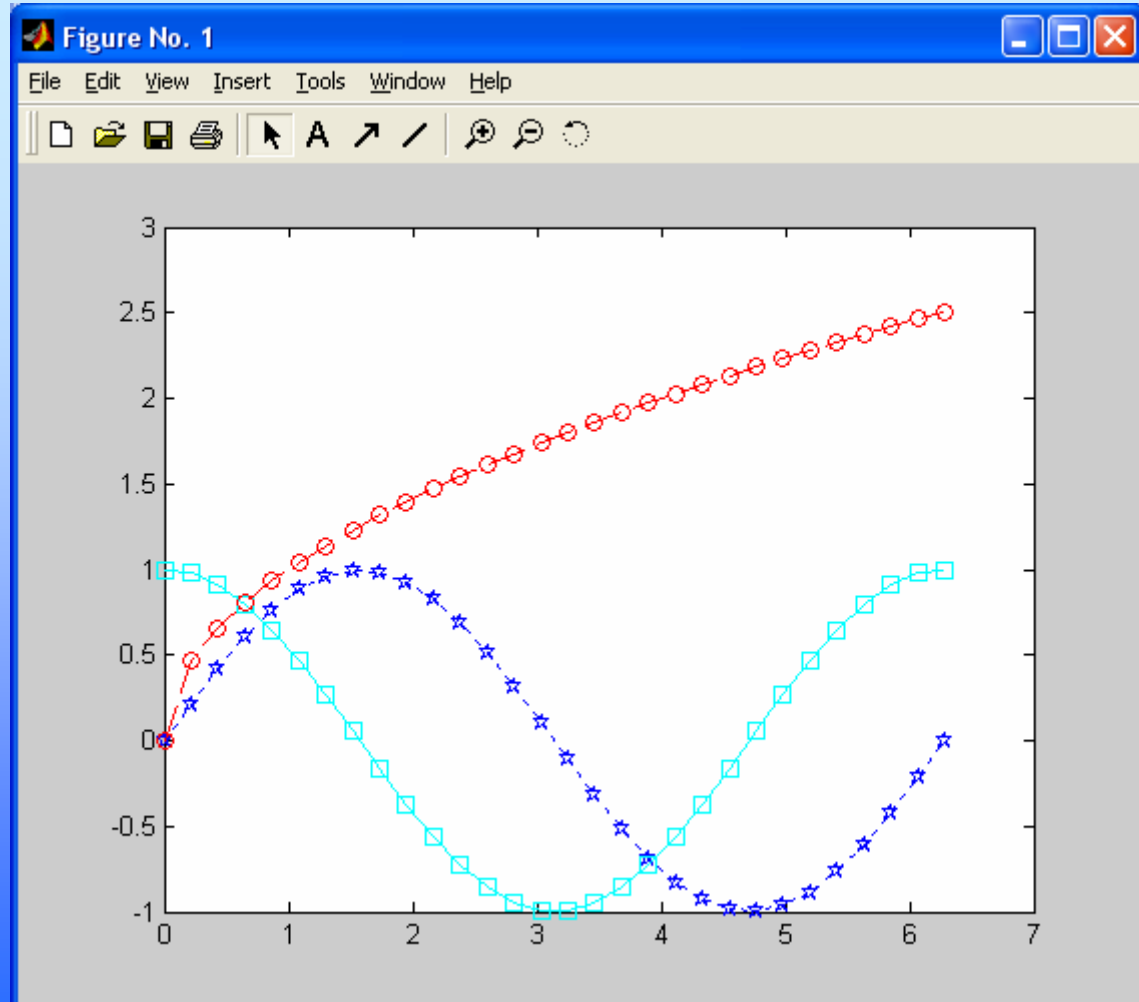
Cria um gráfico no *plano-xy* para desenhar n curvas. Em cada uma delas você pode especificar uma *cor*, um *marcador* e um *tipo* de linha. Se você não especificar nenhuma cor, o **MATLAB** inicia o desenho pela cor azul e desenha o gráfico das próximas funções seguindo as na ordem em que elas aparecem na tabela.

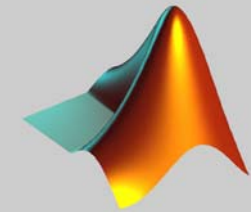


Exemplo :-> Desenhando um gráfico com três curvas, diferentes cores, tipos de linhas e marcadores.

Exemplo:

```
>> x = linspace(0,2*pi,30);
>> y = sin(x);
>> z = cos(x);
>> w = sqrt(x);
>> plot(x,y,'b:p', x,z,'cs-', x,w,'ro-');
```

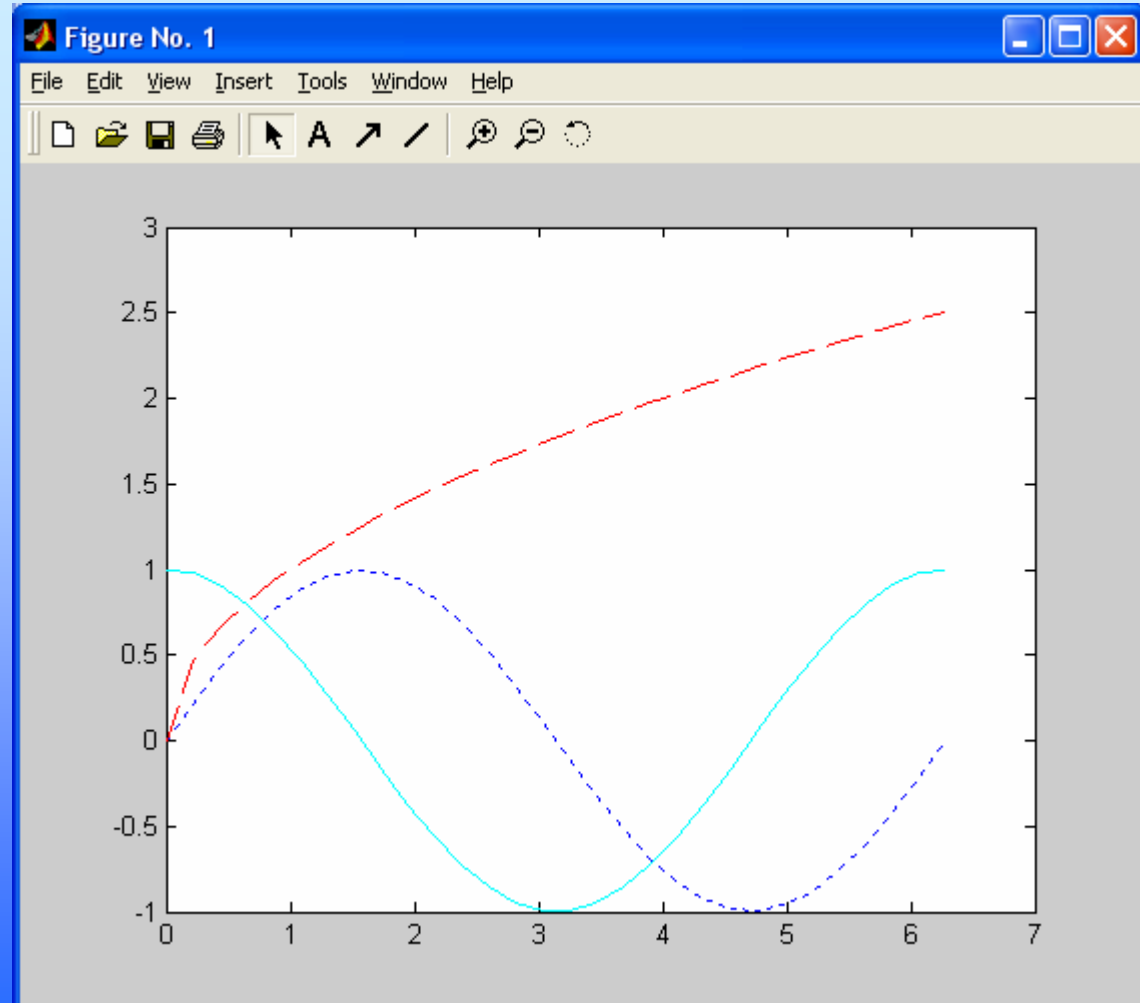


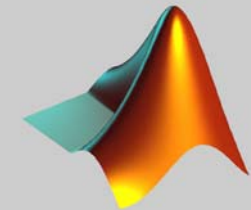


Exemplo :→ O mesmo exemplo anterior, mas sem os marcadores.

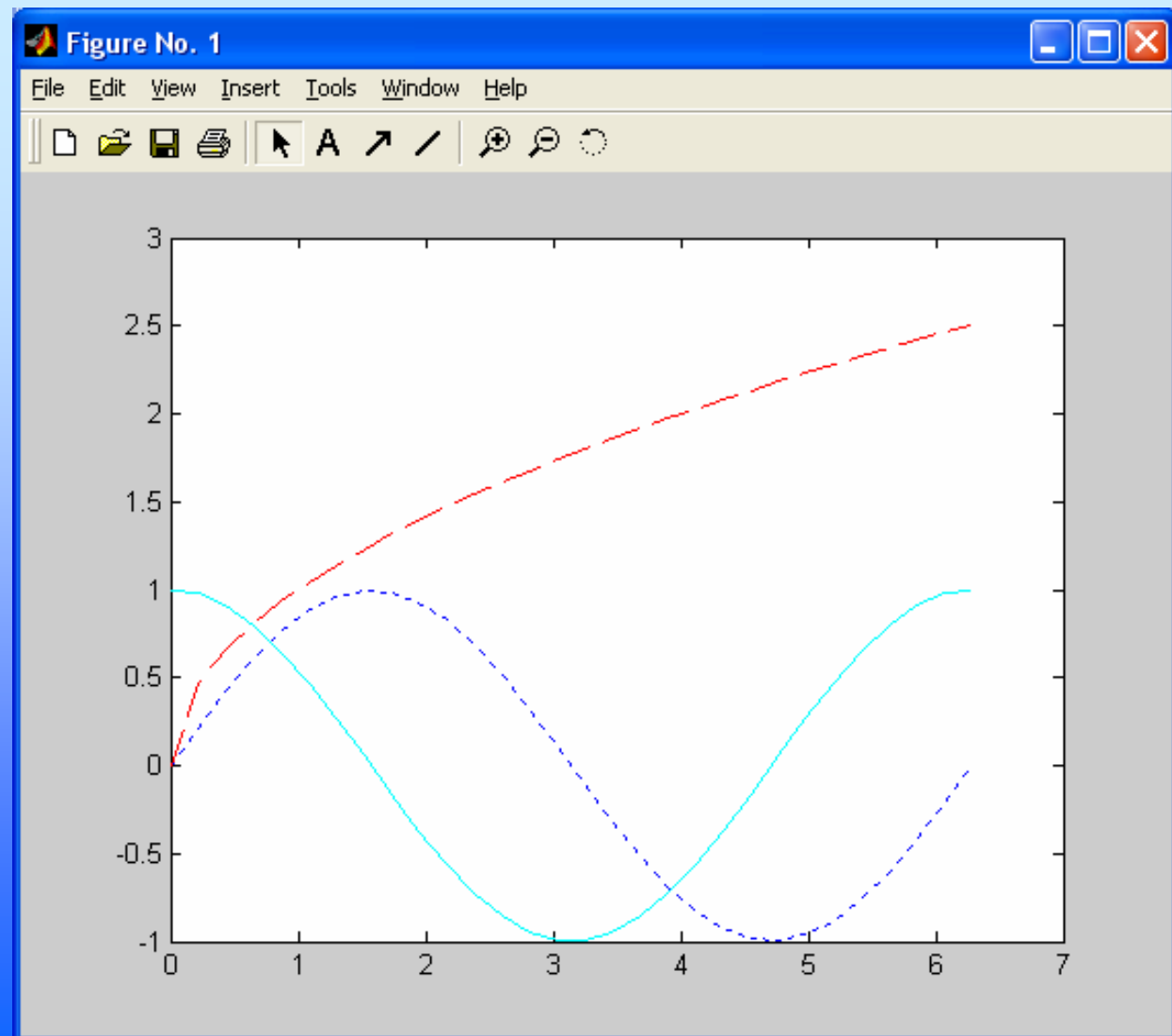
Exemplo:

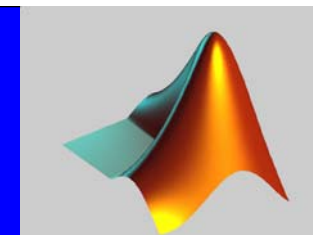
```
>> x = linspace(0,2*pi,30);
>> y = sin(x);
>> z = cos(x);
>> w = sqrt(x);
>> plot(x,y,'b-', x,z,'c-', x,w,'r-');
```





Exemplo :-> Podemos acrescentar outras informações ao *gráfico*?





Mais Funções Básicas para incrementar a Janela de Figuras.

OBJETIVOS: Acrescentar informações a **Janela de Figuras**, como *título*, nomes associados aos *eixos*, *legendas*, *nomes* relativos as funções, etc. Estas características aumentam a legibilidade da figura como um todo, tornando-a mais compreensível.

Comandos:→

>>**grid** *on/off*

Ativa/Desativa exibição de uma grade auxiliar

>>**box** *on/off*

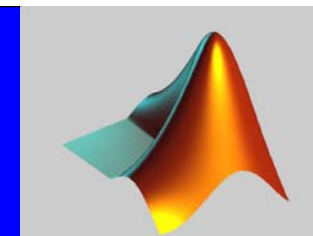
Ativa/Desativa exibição de uma caixa em linhas sólidas que contorna o gráfico

>>**title**(*'Título do Gráfico'*)

Define um título para o gráfico (parte superior)

>>**legend** *'sua_legenda'*

Cria uma legenda (canto superior direito) para o gráfico. É possível mover a legenda com o botão esquerdo do mouse. A opção *off* remove a legenda.



Outras Funções Básicas.

`>>text(x,y,'Seu texto')`

Cria um texto na posição especificada pelas coordenadas (x,y) no espaço de coordenadas do gráfico.

`>>gtext('Seu texto')`

Posiciona o texto na **Janela de Figuras** fazendo uso do mouse. Após encontrar posição de sua escolha, basta clicar com o botão direito.

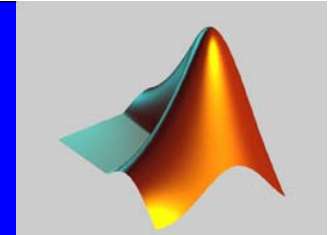
`>>xlabel('Seu Texto')`

Define um nome para o *eixo-x*.

`>>ylabel ('Seu Texto')`

Define um nome para o *eixo-y*.

Observação: → Alguns destes comandos estão disponíveis na *Barra de Menus da Janela de Figuras*.



Mais Funções Básicas.

`>>hold on/off`

Toda vez que você utiliza o comando **plot()**, se a **Janela de Figuras** já estiver ativada, o conteúdo (gráfico) é apagado um novo gráfico é desenhado. O comando **hold** permite Ativar/Desativar a exibição de gráficos numa mesma janela.

`>>hold on`

Não remove os gráficos já existentes a cada novo comando **plot()**, ou seja, novas curvas são acrescentadas.

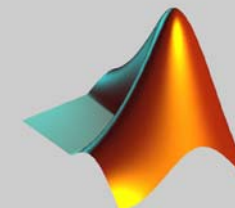
`>>hold off`

Remove os gráficos existentes e desenha um novo a partir do comando **plot()**.

`>>ishold`

Função bolena que retorna o valor 1 (true) se a função **hold** está *on*, e, 0 (false), caso contrário.

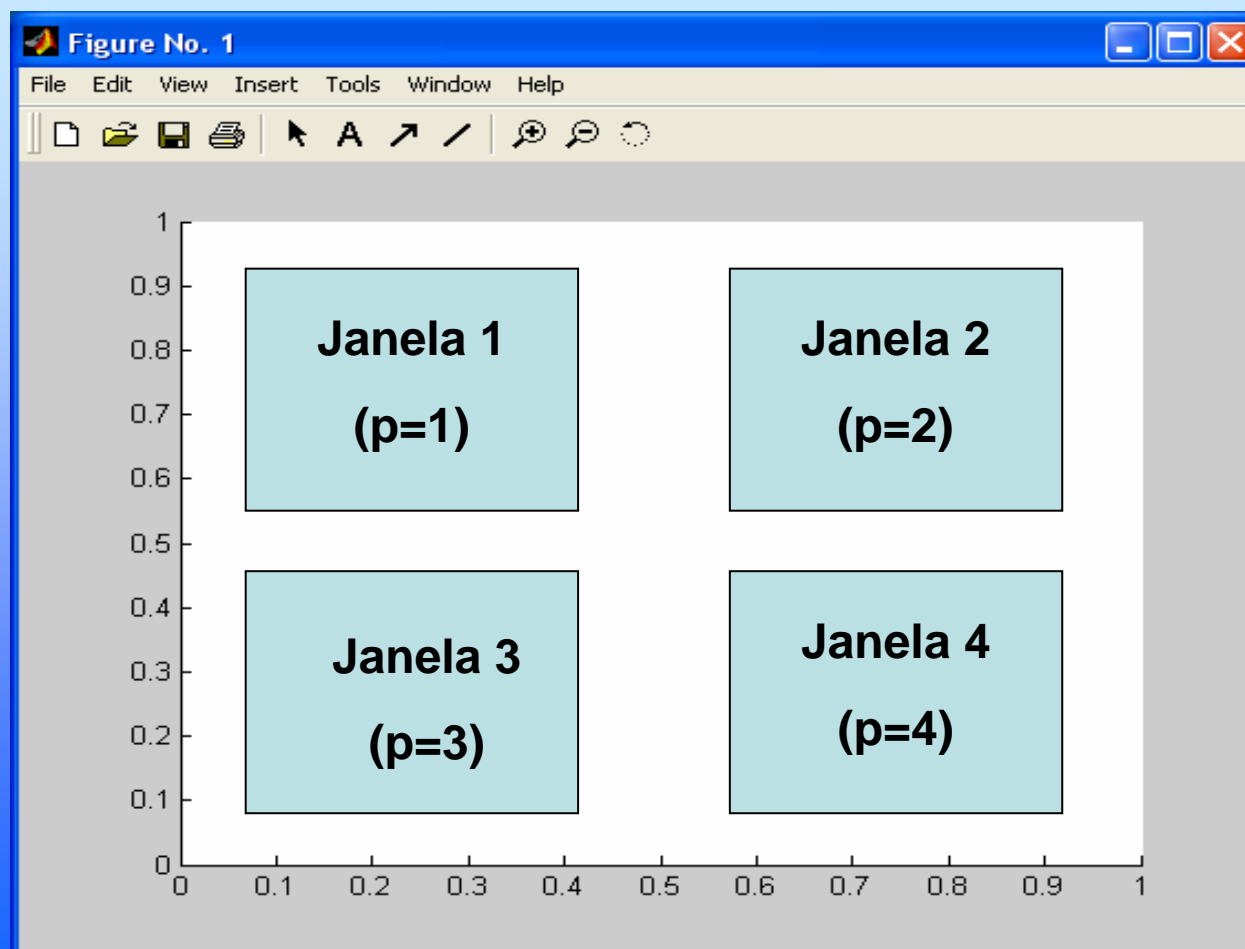
Observação: → Se os gráficos estão em escalas diferentes (limites dos eixos), então o MATLAB se encarrega de atualizar todos para a nova escala (reescalar).

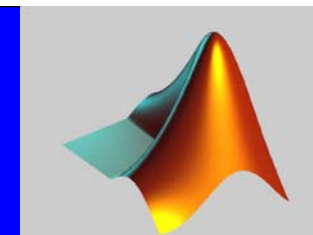


Mais Funções Básicas. `>> subplot(m,n,p)`

Divide uma **Janela de Figuras** numa matriz de $m \times n$ regiões, em que cada uma delas pode conter gráficos independentes e com diferentes eixos coordenados. As regiões são numeradas da esquerda para a direita, e, de cima para baixo.

`>>subplot(2,2,p)` 



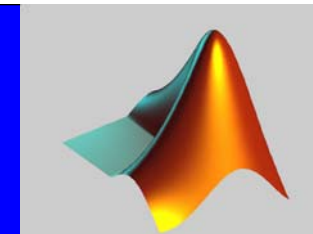


Observação: `>>subplot(m,n,p)`

O comando `subplot()` vem sempre acompanhado do comando `plot()`. Na verdade, o comando `subplot()` escolhe a região em que será desenhada a curva, enquanto o comando `plot()`, gera o gráfico correspondente.

O parâmetro p indica em qual das áreas o comando `plot()` irá atuar.

Em cada uma das áreas é possível usar comandos diferentes, como: `axis`, `title`, `gtext`, etc.



Usando o comando `subplot()`.

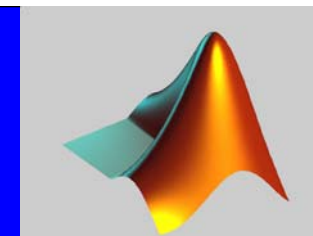
Observações:

➤ Quando você usa o comando `subplot()` ativando, por exemplo, o subgráfico 1, ele permanece ativo até que um novo subgráfico seja ativado. Logo, os comandos `axis()`, `hold`, `xlabel()`, `ylabel()`, `title()`, `grid` e `box` provocam efeitos somente no subgráfico que está ativado;

➤ Um novo comando `subplot()` alterando o número de subgráficos na **Janela de Figuras** faz com que alguns dos anteriores sejam perdidos;

➤ Para voltar ao modo padrão (*default*) e usar integralmente a área relativa a **Janela de Figuras**, é suficiente executar o comando `subplot()` com os seguintes parâmetros:

```
>> subplot(1,1,1);
```



GRÁFICOS DE FUNÇÕES: Exemplo do uso da função `subplot()`: →.

```
>>x = linspace(0,2*pi,40);
```

```
>>y1 = sin(x), y2 = cos(x);
```

```
>>y3 = 2*sin(x).*cos(x), y4 = sin(x)./(cos(x)+eps);
```

```
>>subplot(2,2,1);
```

```
>>plot(x,y1), axis([0 2*pi -1 1]), title('seno(x)');
```

```
>>subplot(2,2,2);
```

```
>>plot(x,y2), axis([0 2*pi -1 1]), title('cos(x)')
```

```
>>subplot(2,2,3);
```

```
>>plot(x,y3), axis([0 2*pi -1 1]), title('2*seno(x)*cos(x)')
```

```
>>subplot(2,2,4);
```

```
>>plot(x,y4), axis([0 2*pi -20 20]), title('seno(x)/cos(x)')
```

Definindo valores no eixo-x e 4 funções

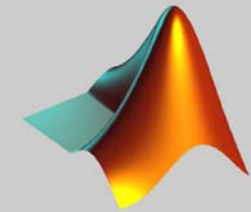
% Selecionando janela 1

% Selecionando janela 2

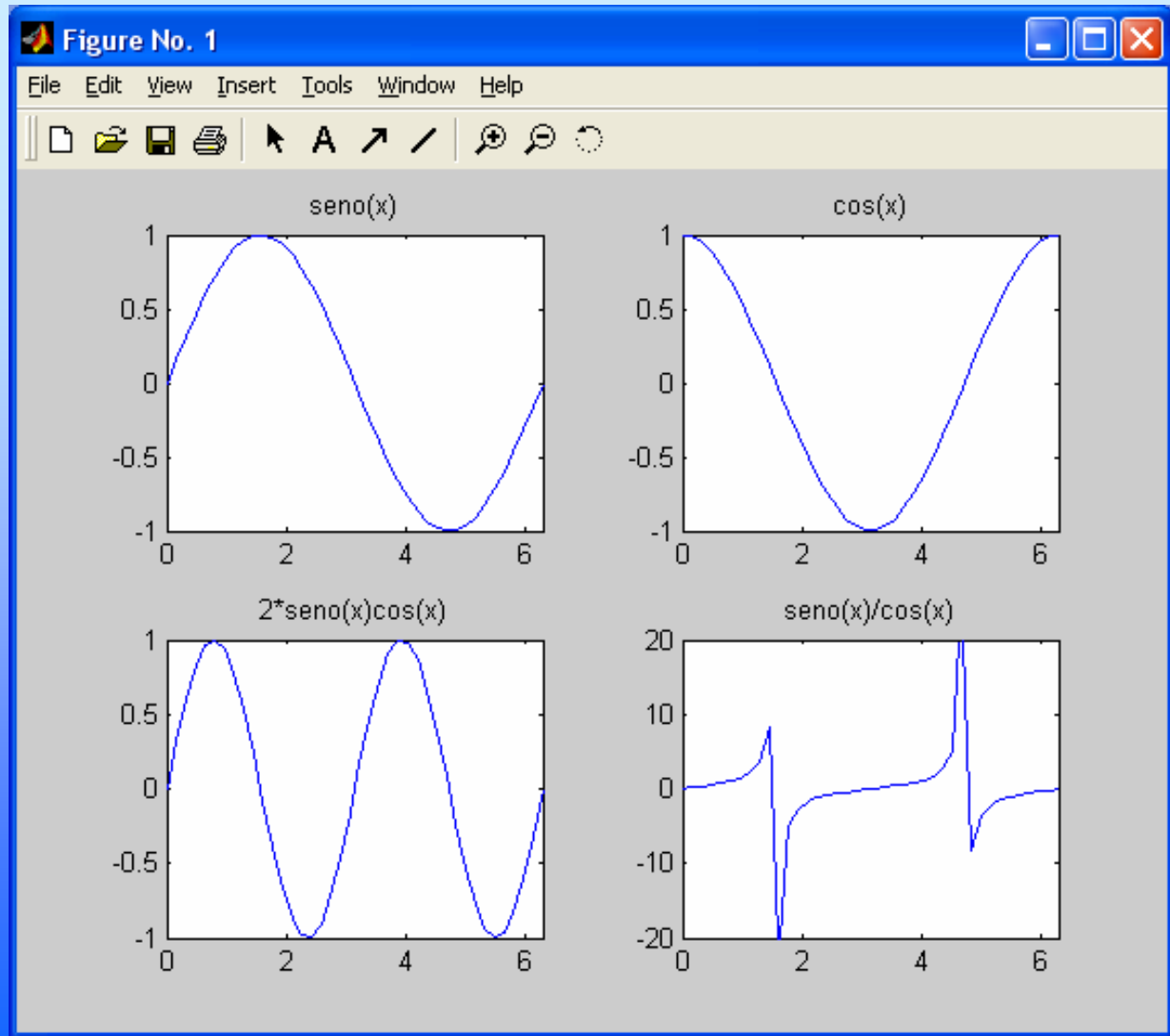
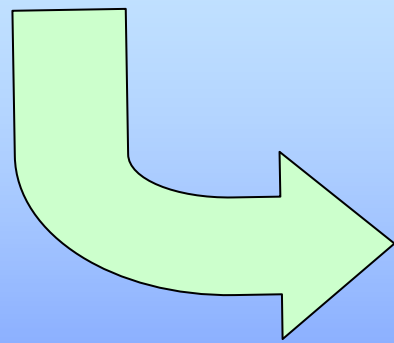
% Selecionando janela 3

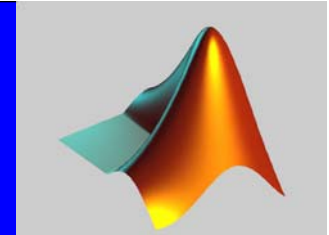
% Selecionando janela 4

Desenho no próximo slide



Resultado da execução dos comandos anteriores





Funções Básicas de Personalização de Eixos.

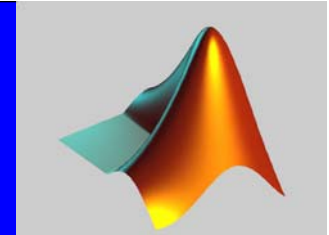
Observe que até agora, só especificamos os limites (valores) em relação ao *eixo-x*, enquanto o **MATLAB** se encarrega de encontrar os valores adequados para o *eixo-y*. Porém, é possível alterar as características de proporção entre os eixos através do comando **axis**. Há uma quantidade enorme de valores associados a este comando. Entretanto, para os nossos propósitos iniciais, vamos ver aqui somente alguns deles.

>>axis([xmin xmax ymin ymax]) Define os valores **mínimos** e **máximos** associados aos eixos.

>>axis auto Volta aos valores originais (*default*) do MATLAB (valores automáticos), onde:

$$xmin = \min(x) \leftrightarrow xmax = \max(x)$$

$$ymin = \min(y) \leftrightarrow ymax = \max(y)$$



Funções Básicas de Personalização de Eixos.

>>axis off

Retira a cor de fundo, a grade, os nomes dos eixos, a caixa (box) e os marcadores. Mantêm somente o título e os textos e gerados pelos comandos **text()** e **gtext()**.

>>axis on

Restaura os atributos retirados pela opção *off* através do comando **axis()**.

Observação: Somente no caso de terem sido definidos.

Observações:

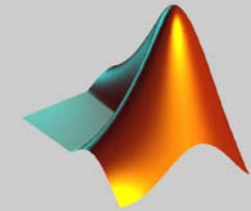
➤ Em todos os exemplos de gráficos, usamos sempre escalas lineares. Em muitas situações poderá ser útil usar escalas logarítmicas em ambos os eixos, ou, em algum deles.

➤ O **MATLAB** faz também outros tipos de gráficos bidimensionais, como por exemplo: em formatos de pizza, histogramas, etc.

FCT – UNESP – CAMPUS DE P. PRUDENTE

Licenciatura em Matemática – 2007 - Prof. Piteri

Programação Orientada a Sistemas de Processamento Simbólico



INTRODUÇÃO AO MATLAB: Conceitos Operatórios Básicos

AULA 02

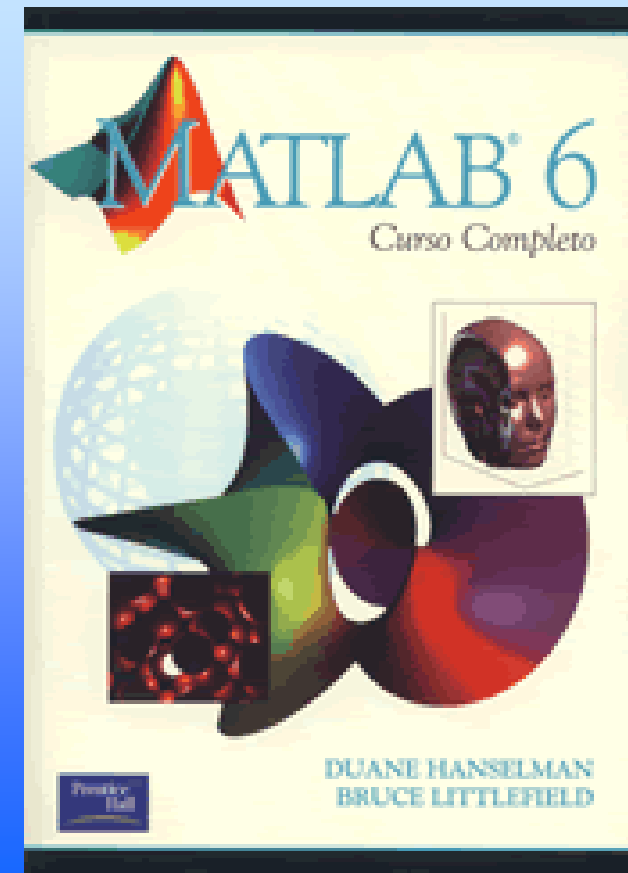
**AULA DE HOJE: → CAPÍTULOS TRABALHADOS INTEGRALMENTE
OU EM PARTE:**

1.Título: MATLAB 6 – Curso Completo

Autor(es): Duane C. Hanselman, Bruce C. Littlefield.

Editora: [Prentice Hall Brasil](http://www.prenticehall.com.br)

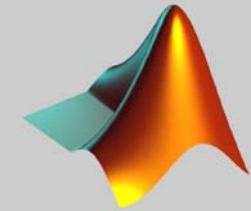
Capítulo : 25.



FCT – UNESP – CAMPUS DE P. PRUDENTE

Licenciatura em Matemática – 2007 - Prof. Piteri

Programação Orientada a Sistemas de Processamento Simbólico



INTRODUÇÃO AO MATLAB: Conceitos Operatórios Básicos

AULA 02

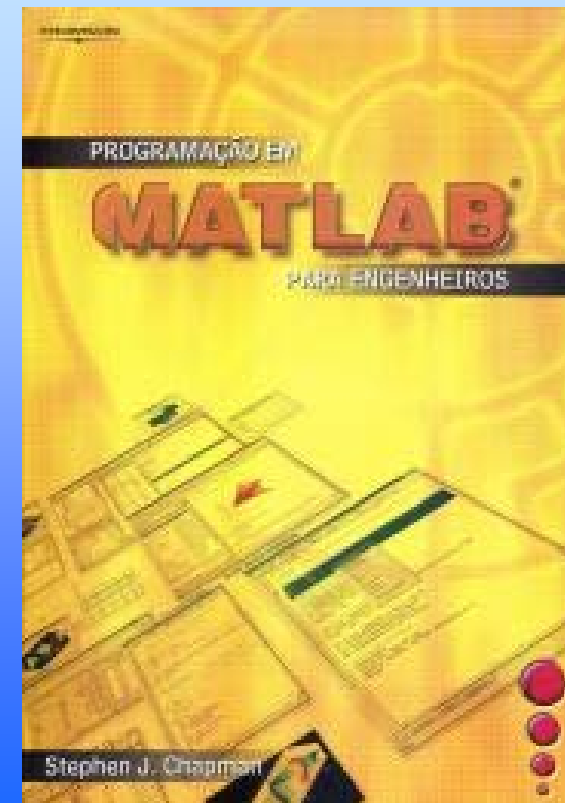
**AULA DE HOJE: → CAPÍTULOS TRABALHADOS INTEGRALMENTE
OU EM PARTE:**

2.Título: Programação em MATLAB para engenheiros

Autor(es): Stephen J. Chapman.

Editora: [Thomson Pioneira](#)

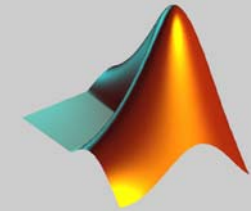
Capítulo : *Não há capítulo específico.*



FCT – UNESP – CAMPUS DE P. PRUDENTE

Licenciatura em Matemática – 2007 - Prof. Piteri

Programação Orientada a Sistemas de Processamento Simbólico



INTRODUÇÃO AO MATLAB: Conceitos Operatórios Básicos

AULA 02

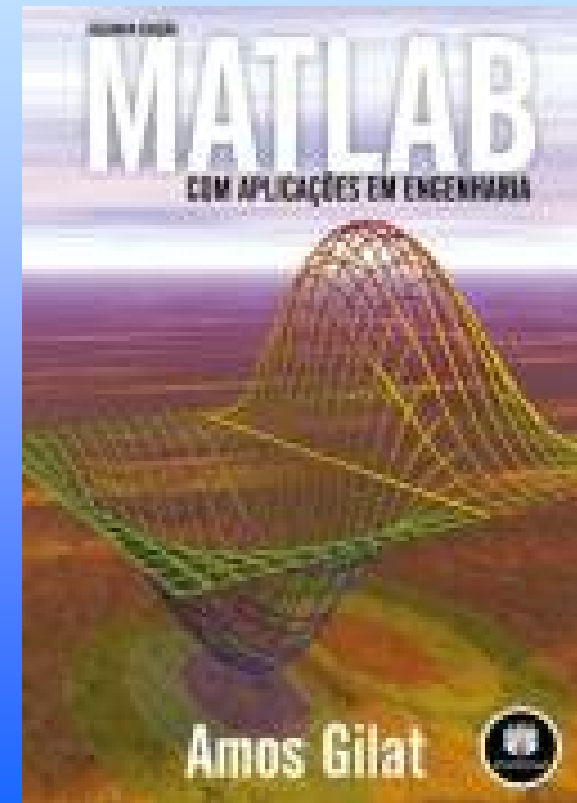
**AULA DE HOJE: → CAPÍTULOS TRABALHADOS INTEGRALMENTE
OU EM PARTE:**

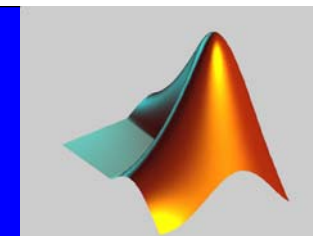
2.Título: MATLAB com aplicação em Engenharia

Autor(es): Amos Gilat.

Editora: [Bookman Companhia Ed.](#)

Capítulos : 5.





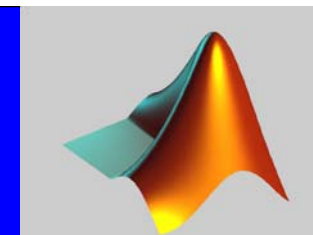
➤ **Semana que vem: → Evento Acadêmico.**

➤ **Próxima Aula: → Em duas semanas.**

➤ **Arquivos M no MATLAB;**

➤ **Segundo Trabalho; Manipulação de Gráficos com Arquivos M.**

Com o tempo e a familiaridade com o ambiente MATLAB e a continuidade do uso dessa ferramenta em outras disciplinas, vocês irão aprender outros recursos de manipulação de gráficos. Por exemplo, quando vocês estiverem fazendo *Cálculo Diferencial e Integral II*, poderão manipular gráficos de funções e superfícies no espaço 3D.



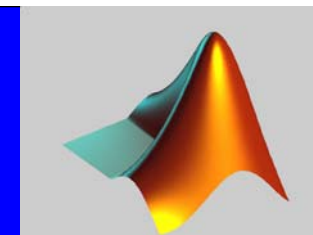
Trabalho de Pesquisa

Elaborar um texto (documento) sobre o tema de *números complexos* (*tudo o que você achar importante*):

- Aspectos históricos;
- Operações sobre o corpo dos complexos;
- Aplicações, etc.

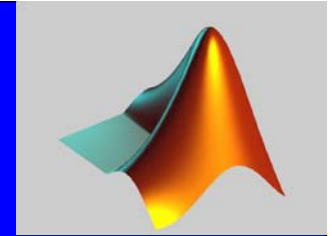
Contextualizar as operações sobre os *números complexos* no **MATLAB**, ou seja, exemplificar como as operações são realizadas no **MATLAB**.

Tome suas decisões em relação a formatação, número de páginas e organização do documento.



Trabalho de Pesquisa

- Grupos de até 4 pessoas;
- Documento a ser elaborado no Word ou Powerpoint, ou um outro software semelhante;
- Colocar uma página inicial no trabalho com os nomes dos integrantes do grupo;
- **Prazo de Entrega → 06 de setembro de 2007.**
- Enviar para piteri@fct.unesp.br



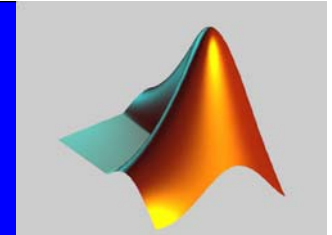
A linguagem é apenas o instrumento da ciência, e as palavras não passam de símbolos das idéias.

Samuel Johnson (1709-1794)

Um bom programador pode superar uma linguagem ruim ou um sistema operacional desajeitado, mas nem mesmo um excelente ambiente de programação poderá salvar um programador ruim.

Brian W. Kernighan e Rob Pike

The Practice of Programming



**Na primeira noite /eles se aproximam
Colhem uma flor de nosso jardim/
E não dizemos nada./Na segunda noite/
Já não se escondem;/pisam as flores/
Matam nosso cão/ e não dizemos nada/
Até um dia/ o mais frágil deles/
Entra sozinho em nossa casa/rouba-nos a lua e,
conhecendo nosso medo/
Arranca-nos a voz da garganta./
E porque não dissemos nada,/
Já não podemos dizer nada.**

Maiakowski