

Minicurso de MATLAB®

uma introdução

Reginaldo Cardoso

Pós-Graduação em Engenharia Mecânica
Universidade Federal do ABC

Lab. Z-202 - UFABC-SBC
September 4, 2024

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

O MATLAB[®] (uma abreviatura de MATrix LABoratory) é um sistema baseado em matrizes para cálculos matemáticos e de engenharia.

O MATLAB[®] pode ser usado de modo direto, ou seja, comandos simples são processados imediatamente e exposto na tela o resultado, mas também é capaz de executar sequências de comandos que são armazenadas em arquivos.

Uma característica conveniente é que as variáveis não precisam ser dimensionadas antes do uso, elas são geradas automaticamente. Tais variáveis permanecem na memória até que se entre com um dos comandos `>> exit`, `>> quit` ou `>> clear`.

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

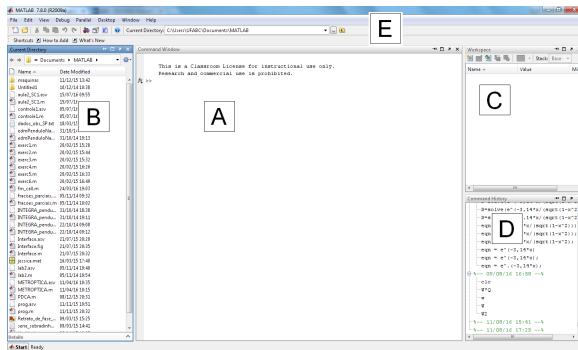
4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Introdução: Ambiente



A - **[Command Window]**: janela de comando, na qual são digitados os dados, (sinal de *prompt*);

B - **[Current folder]**: lista de arquivos contidos no diretório corrente;

C - **[Workspace]**: lista de variáveis criadas

D - **[Command History]**: armazena todas as instruções executadas

E - Menus Superiores

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Símbolos Especiais

Símbolos Especiais	
>> []	Construtor de matrizes
>> ' '	Marca os limites de uma cadeia de caracteres
>> ,	Separa índices ou elementos de matriz
>> ;	1 - Suprime o "eco" na Janela de Comando 2 - Separa as linhas da matriz 3 - Separa declarações de atribuição em uma linha
>> %	Início de Comentário
>> :	Separa índices ou elementos de matriz
>> +	Soma estrutural e matricial
>> -	Subtração estrutural e matricial
>> .*	Multiplicação estrutural
>> *	Multiplicação matricial
>> ./	Divisão estrutural à direita
>> \	Divisão estrutural à esquerda

Símbolos Especiais cont.	
>> /	Divisão matricial à direita
>> \	Divisão matricial à esquerda
>> .^	Expoente estrutural
>> '	Operador de transposição
>> ...	Continua uma declaração MATLAB [®] na linha seguinte

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Algumas Funções Matemáticas Elementares

Funções Matemáticas Elementares	
>> <code>abs(x)</code>	Valor absoluto de x
>> <code>acos(x)</code>	Arco cosseno de x
>> <code>asin(x)</code>	Arco seno de x
>> <code>atan(x)</code>	Arco tangente de x
>> <code>cos(x)</code>	Cosseno de x
>> <code>sin(x)</code>	Seno de x
>> <code>tan(x)</code>	Tangente de x
>> <code>exp(x)</code>	Exponencial (e^x)
>> <code>log(x)</code>	Logaritmo natural (base e)
>> <code>log10(x)</code>	Logaritmo na base 10
>> <code>sqrt(x)</code>	Raiz quadrada
>> <code>factorial(x)</code>	Fatorial de x ($x!$)

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Declarando Variáveis

O sinal de igualdade (=) é denominado operador de atribuição. O operador de atribuição inicializa ou modifica o valor de uma variável.

```
>> Nome_da_Variavel = valor da variável
```

por exemplo:

```
>> distancia = 100
```

Lembrando que o MATLAB[®] distingue letras maiúsculas de minúsculas e não reconhece acentos.

```
>> distância = 2
```

```
distância = 2
```

```
|
```

```
Error: The input character is not valid in MATLAB statements or  
expressions.
```

```
>> Distancia = 20
```

```
>> distancia = 100
```

portanto distancia \neq Distancia

Algumas Operações

Velocidade média:

```
>> tempo = 3
```

```
tempo =  
      3
```

```
>> velocidade_media = distancia / tempo
```

```
velocidade_media =  
33.333333333333336
```

Para suprimir a exibição da variável adicionamos um ponto-e-vírgula ao final do comando.

Formatando dados numéricos

valor a ser analisado: 12.345678901234567

Formatos de Exibição de saída		
Comando	Resultado	Exemplo
>> <code>format short</code>	4 dígitos decimais (formato padrão)	12.3457
>> <code>format long</code>	14 dígitos decimais	12.345678901234567
>> <code>format short e</code>	5 dígitos mais expoente	1.2346e+001
>> <code>format short g</code>	5 dígitos no total com ou sem expoente	12.346
>> <code>format long e</code>	15 dígitos mais expoente	1.234567890123457e+001
>> <code>format long g</code>	15 dígitos no total com ou sem expoente	12.3456789012346
>> <code>format bank</code>	Formato monetário	12.35
>> <code>format hex</code>	Exibição hexadecimal de bits	4028b0fcd32f707a
>> <code>format rat</code>	Razão aproximada entre inteiros pequenos	1000/81
>> <code>format compact</code>	Elimina espaços (+informação mostrada na tela)	
>> <code>format loose</code>	Adiciona espaços entre linhas	
>> <code>format +</code>	Exibe somente o sinal do número	+

Ordem de Precedência

Regra Associativa: sempre da esquerda para a direita.

Ordem de Precedência	
0	Parenteses ()
1	Exponenciação (^), transposição (')
2	Unary (+), negação lógica (~)
3	Multiplicação (*), divisão (/)
4	Adição (+), subtração (-)
5	Operador dois pontos (:)

Exemplo Unary: $-A$, $+2$, H'

operações com somente uma variável.

exemplos: $6/2*3 = 1$ ou 9 ?

```
>> 12/2 + 3 * (2 ^ 4)
```

```
ans =
```

```
54
```

Comandos e Variáveis

Quando criamos uma expressão e não a armazenamos em uma variável o MATLAB® a salva automaticamente na variável `ans`.

Apagar uma ou mais variáveis, comando `clear`

`>> clear tempo` → apaga somente a variável `tempo`

`>> clear` → apaga todas as variáveis

Para limpar a **[Janela de Comando]** usa-se o comando `clc`

`>> clc`

Variáveis Predefinidas	
<code>pi</code>	3.141592653589793
<code>eps</code>	Somado a 1, cria um número maior do que 1
<code>inf</code>	Infinito
<code>NaN</code>	Não numero (not a number)
<code>i</code> e <code>j</code>	$\sqrt{-1}$
<code>realmin</code> , <code>realmax</code>	menor, maior número real positivo

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Declarando Variáveis: Vetor

Vetor: matriz com somente uma dimensão, só uma linha ou só uma coluna.

(,) ou espaço : indica a separação de elementos da mesma linha, definindo colunas;

(;): finaliza a definição da linha.

Vetor linha (covetor),

```
>> [1 2 3]
```

```
ans =
```

```
1 2 3
```

```
>> [1,2,3]
```

```
ans =
```

```
1 2 3
```

Vetor coluna,

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

```
ans =
```

```
1
```

```
2
```

```
3
```

Declarando Variáveis: Vetor

Outra forma de criar um conjunto:

```
>> A=1:1:5
```

```
A =
```

```
1 2 3 4 5
```

O primeiro valor é o valor inicial, o segundo o “salto” e o terceiro o valor final.

([linspace](#)) - Gera um vetor linearmente espaçado a partir de um valor inicial, um valor final e um número de elementos,

```
>> X=linspace(0,pi,4)
```

```
X =
```

```
0 1.0472 2.0944 3.1416
```

([logspace](#)) - Gera um vetor logaritmicamente espaçado a partir de uma potência inicial, uma potência final e um número de valores,

```
>> V=logspace(0,2,5)
```

```
V =
```

```
1.0000 3.1623 10.0000 31.6228 100.0000
```

Declarando Variáveis: Vetor

Algumas Operações:

```
>> B=2*A
```

```
B =  
    2    4    6    8   10
```

```
>> Y=sin(X)
```

```
Y =  
    0    0.8660    0.8660    0.0000
```

Qual será a resposta?

```
>> Z_1 =A. ^ 2
```

```
>> Z_2 =A ^ 2
```

Possíveis respostas:

```
Z =  
    1    4    9   16   25
```

```
??? Error using ==> mpower  
Inputs must be a scalar and a square matrix.
```

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Declarando Variáveis: Matriz

A mesma coisa, mas bidimensional:

```
>> M=[1,0,-1;2,3,4;-7,1,3]
```

```
M =
```

```
    1    0   -1  
    2    3    4  
   -7    1    3
```

```
>> M=[1 0 -1; 2 3 4; -7 1 3]
```

```
M =
```

```
    1    0   -1  
    2    3    4  
   -7    1    3
```

Acessando um elemento de uma matriz.

(**M(2,3)**)→ Identifica o elemento da Segunda linha e Terceira coluna.

```
>> M(2,3)
```

```
ans =
```

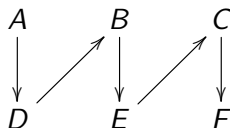
```
    4
```

```
>> M(8)
```

```
ans =
```

```
    4
```

Declarando Variáveis: Matriz



```
>> M(10)
```

```
??? Index exceeds matrix dimensions.
```

Temos um erro quando acessamos uma posição inexistente.

Declarando Variáveis: Matriz

Podemos criar matrizes a partir de vetores ou outras matrizes,

```
>> b=[2,-3,1];
```

```
>> Mx=[b',M(:,2:3)]
```

```
Mx =
```

```
    2    0   -1  
   -3    3    4  
    1    1    3
```

(`M(:, 2:3)`) parte da matriz `M` compreendida por todas as linhas (`:`) e as colunas 2 e 3 (`2:3`).

A matriz `Mx` foi gerada concatenando-se o vetor `b` transposto e as colunas 2 e 3 da matriz `M`.

Transposição - Utilizamos o operador (`'`) (aspas simples).

Algumas matrizes Predefinidas e funções

Algumas matrizes Predefinidas e funções	
<code>size(Mx)</code>	Retorna o número de linhas e de colunas de <code>Mx</code>
<code>length(Mx)</code>	A maior dimensão da matriz <code>Mx</code>
<code>inv(Mx)</code>	Calcula a matriz inversa de <code>Mx</code>
<code>zeros(n,m)</code>	Matriz de zeros com <code>n</code> linhas e <code>m</code> colunas
<code>eye(n,m)</code>	Matriz identidade com <code>n</code> linhas e <code>m</code> colunas
<code>ones(n,m)</code>	Matriz com 1, com <code>n</code> linhas e <code>m</code> colunas
<code>det(Mx)</code>	Calcula o determinante da matriz <code>Mx</code>

Qual será a resposta?

```
>> inv([2,-3,1])  
>> inv(eye(size(Mx)))  
>> length(Mx)  
>> eye(2)  
>> zeros(3)  
>> ones(3)  
>> [n,m]=size(Mx)
```

Respostas

```
>> inv([2,-3,1])  
??? Error using ==> inv  
Matrix must be square.  
>> inv(eye(size(Mx)))  
ans =  
    1    0    0  
    0    1    0  
    0    0    1  
>> length(Mx)  
ans =  
     3  
>> eye(2)  
ans =  
     1     0  
     0     1
```

```
>> zeros(3)  
ans =  
     0     0     0  
     0     0     0  
     0     0     0  
>> ones(3)  
ans =  
     1     1     1  
     1     1     1  
     1     1     1  
>> [n,m]=size(Mx)  
n =  
     3  
m =  
     3
```

Algumas funções

Problema prático! $Ax = b$, onde:

$$\begin{bmatrix} 2 & 0 & -1 \\ -3 & 3 & 4 \\ 1 & 1 & 3 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 9 \\ 8 \\ 7 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 & -1 \\ -3 & 3 & 4 \\ -3 & 3 & 4 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 9 \\ 8 \\ 8 \end{bmatrix}$$

```
>> x = inv(A)*b
```

```
>> x = A\b
```

Pseudo-inversa

```
>> x = A'*inv(A*A')*b
```

Algumas funções	
<code>who</code>	Exibe uma lista de variáveis declaradas/ativas na memória
<code>whos</code>	Exibe uma lista de variáveis declaradas na memória, com o respectivo tamanho em bytes e a classe de armazenamento

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

O MATLAB[®] possui sofisticados recursos para a visualização de dados na forma gráfica. Ele trabalha com objetos gráficos, tais como linhas e superfícies. Entretanto, o MATLAB[®] disponibiliza diversas funções que facilitam a configurações das propriedades de objetos.

Por exemplo, suponha que queiramos desenhar a função:

$$y = x^2 - 10x + 15, \text{ para valores de } x \text{ entre } 0 \text{ e } 10.$$

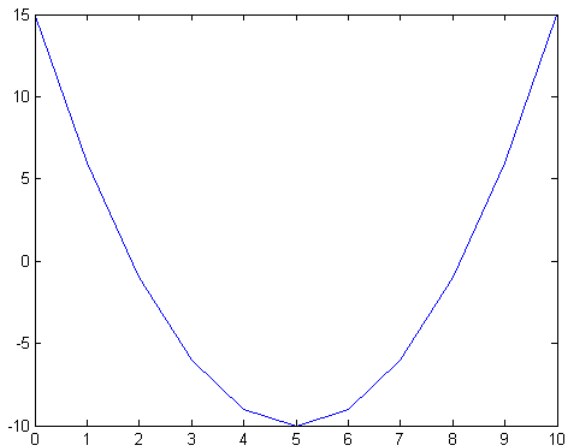
Neste caso, o gráfico será bidimensional e a sintaxe mais simples para isto:

`plot(x,y)`

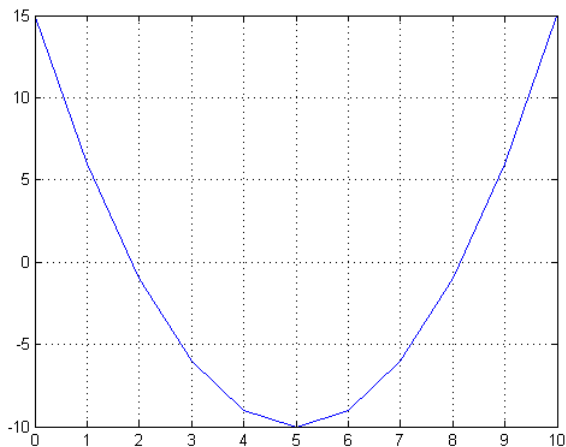
onde: **x** e **y** devem ser vetores e ambos devem possuir a mesma quantidade de elementos.

Basta três linhas de comando:

```
>> x=0:1:10;  
>> y=x.^2 -10.*x+15;  
>> plot(x,y);
```



```
>> grid
```



Gráficos: Comando "plot"

Gráficos bidimensionais no MATLAB®

Sintaxe:

```
plot(x,y,'Especificadores de linha',  
      'Propriedade',Valor propriedade)
```

<code>x e y</code>	Vetores com a mesma quantidade de elementos.
<code>'Especificações de linha'</code>	(Opcional) Especifica o tipo e a cor da linha e os tipos de marcadores.
<code>'Propriedades', Valor propriedade</code>	(Opcional) Propriedades adicionais usadas para definir a espessura da linha, o tamanho do marcador e da borda e a cor de preenchimento.

Gráficos: Possibilidades

Linhas	
Estilo	Especificador
Sólida (padrão)	" - "
Tracejada	" - - "
Pontilhada	" : "
Traço-Ponto	" - . "

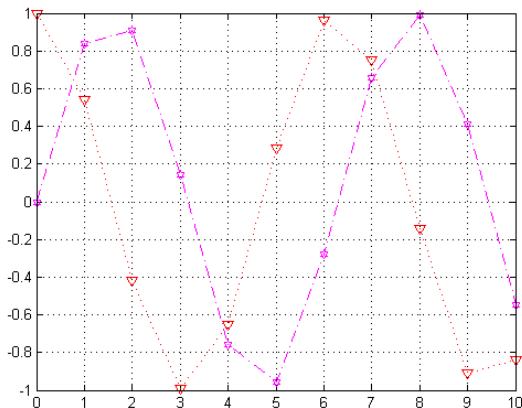
Cores	
Cor	Especificador
Azul	" b "
Verde	" g "
Vermelho	" r "
Ciano	" c "
Magenta	" m "
Amarelo	" y "
Preto	" k "
Branco	" w "

Gráficos: Possibilidades

Marcadores	
Marcador	Especificador
Sinal	" + "
Circulo	" o "
Asterisco	" * "
Ponto	" . "
Quadrado	" s "
Losango	" d "
Pentagrama	" p "
Hexagrama	" h "
Triâng. p/ baixo	" v "
Triâng. p/ cima	" ^ "
Triâng. p/ esquerda	" < "
Triâng. p/ direita	" > "

Gráficos: Exemplo

```
>> x=0:1:10;y=cos(x);z=sin(x);  
>> plot(x,y,':rv',x,z,'-.mh');grid;
```

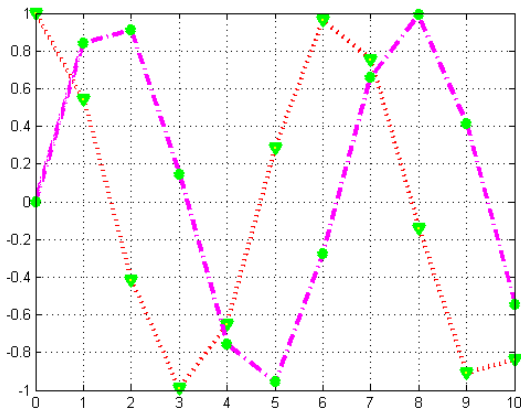


Gráficos: Mais Possibilidades

Marcadores		
Propriedade	Descrição	Valores
'LineWidth'	Largura da linha.	Número (default 0.5).
'MarkerSize'	Tamanho do marcador.	Número.
'MarkerEdge-Color'	Borda.	Cores.
'MarkerFace-Color'	Preenchimento.	Cores.

Gráficos: Exemplo

```
>> plot(x,y,':rv',x,z,'-.mh','linewidth',3,'markersize',10,  
'markeredgecolor','g','markerfacecolor','y');grid;
```



Gráficos: Formatando

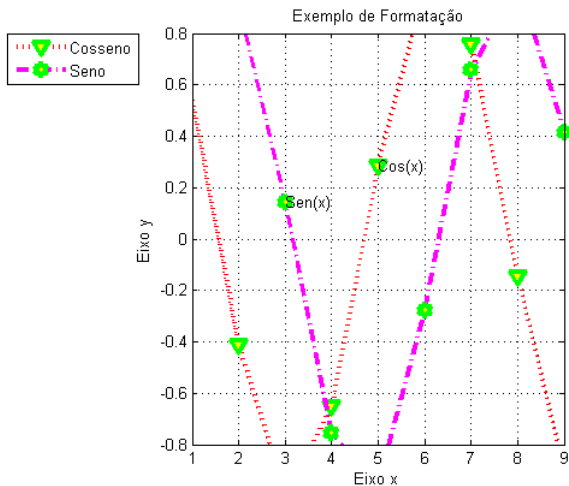
Comando	Descrição	Valores
<code>xlabel</code>	Rotulo eixo x.	<code>>> xlabel('string de texto')</code>
<code>ylabel</code>	Rotulo eixo y.	<code>>> ylabel('string de texto')</code>
<code>title</code>	Titulo ao gráfico.	<code>>> title('string de texto')</code>
<code>text</code>	Caixa de texto.	<code>>> text(x,y,'string de texto')</code>
<code>axis</code>	Limite aos eixos.	<code>>> axis([xmin,xmax,ymin,ymax])</code>
<code>legend</code>	Legenda.	<code>>> legend('string1',...,pos)</code>

pos	
<code>'location', 'North'</code>	Dentro do gráfico em cima
<code>'South'</code>	Dentro abaixo
<code>'East'</code>	Dentro direita
<code>'West'</code>	Dentro esquerda
<code>'NorthEast'</code>	Default
<code>'NorthEastOutside'</code>	Fora em cima direita

Gráficos: Exemplo

```
>> plot(x,y,':rv',x,z,'-.mh','linewidth',3,'markersize',10,  
    'markeredgecolor','g','markerfacecolor','y');grid;  
>> xlabel('Eixo x');ylabel('Eixo y')  
>> title('Exemplo de Formatação')  
>> text(x(6),y(6),'Cos(x)')  
>> text(x(4),z(4),'Sen(x)')  
>> axis([1,9,-0.8,0.8])  
>> legend('Cosseno','Seno','location','northwestoutside')
```

Gráficos: Exemplo



Gráficos: Algumas Funções

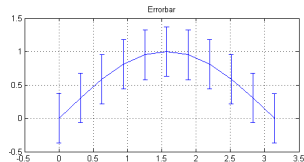
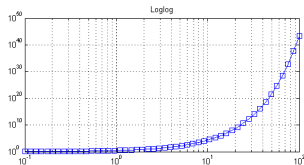
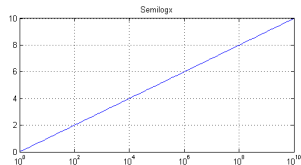
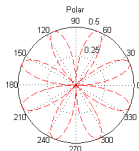
Funções	Descrição
<code>semilogx(x,y)</code>	Escala logarítmica em x e y linear
<code>semilogy(x,y)</code>	Escala logarítmica em y e x linear
<code>loglogy(x,y)</code>	Escala logarítmica em x e y
<code>plotyy(x,y)</code>	Escalas diferentes em y
<code>stem(x,y)</code>	Discreto
<code>fill(x,y)</code>	Polígono 2D
<code>polar(x,y)</code>	Coord. polar
<code>bar(x,y)</code>	Barras
<code>stairs(x,y)</code>	Plotar em degrau
<code>errorbar(x,y)</code>	Erro
<code>hist(x,y)</code>	Histograma
<code>rose(x,y)</code>	Histograma em ângulo
<code>compass(x,y)</code>	Forma de bússola
<code>comet(x,y)</code>	Tratória cometa

Gráficos: Subplot

```
>> subplot(2,2,1)
>> t = 0:0.01:2*pi;
>> prim =
sin(2*t).*cos(2*t);
>> polar(t,prim,'--r');grid
>> title('Polar')
>> subplot(2,2,2)
>> x = 0:0.1:10;
>> semilogx(10.^ x,x);grid
>> title('Semilogx')
>> subplot(2,2,3)
```

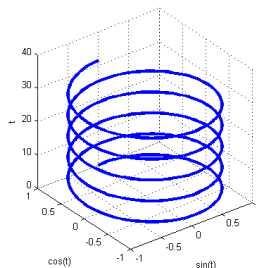
```
>> x = logspace(-1,2);
>>
loglog(x,exp(x),'-s');grid
>> title('Loglog')
>> subplot(2,2,4)
>> X = 0:pi/10:pi;
>> Y = sin(X);
>> E = std(Y)*ones(size(X));
>> errorbar(X,Y,E);grid
>> title('Errorbar')
```

Gráficos: Exemplo



Gráficos: plot3

```
>> t = 0:pi/50:10*pi;  
>> plot3(sin(t),cos(t),t)  
>> xlabel('sin(t)')  
>> ylabel('cos(t)')  
>> zlabel('t')  
>> grid  
>> axis square
```



1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Gráfico Interativo

```
>> t = 0:pi/50:10*pi;  
>> y = sin(t);
```

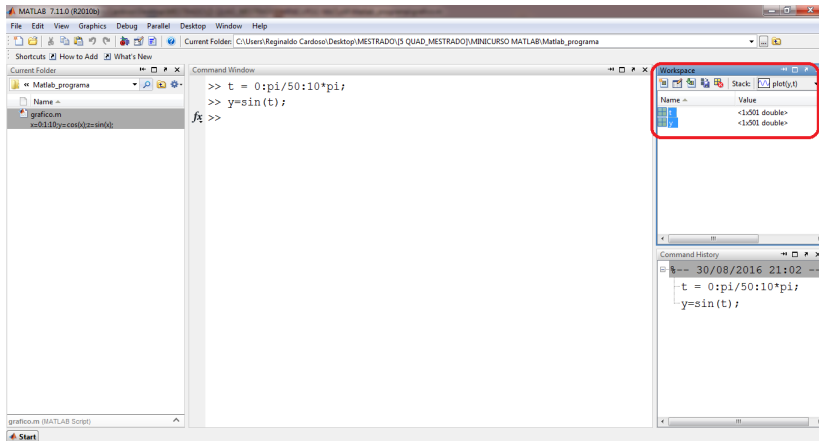
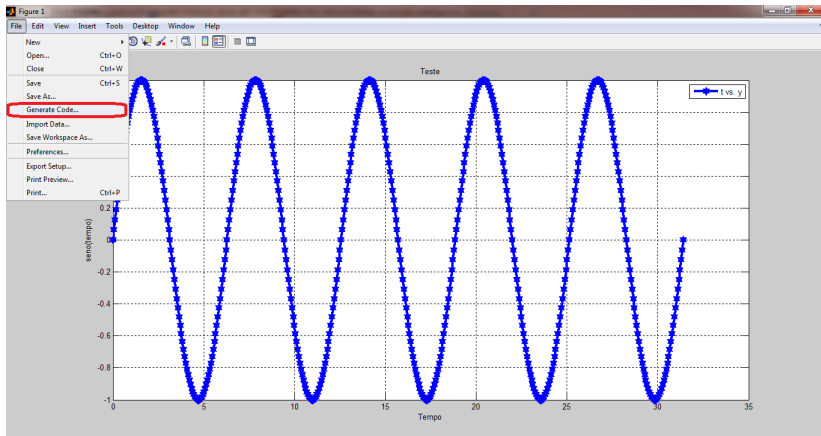


Gráfico Interativo



1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Editor de Programa

Existe um ambiente próprio do MATLAB[®] para edição de programas.

Para abri-lo, digita-se o comando:

`>> edit` ou "CTRL+N".

Tal editor apresenta algumas características interessantes:

- Linhas numeradas, o que é útil principalmente para a localização de erros de programação;
- O caractere "%" indica comentário;
- O caractere "..." indica que o comando continua na próxima linha. Tal recurso permite deixar o texto mais "organizado";
- A cor atribuída pelo editor a um texto indica a sua classe. O padrão de cores varia de acordo com a versão do MATLAB[®].

Um programa em MATLAB[®] possui a extensão ".m", chamado de m-file. Existem dois tipos de arquivos MATLAB[®]: **script** e **função**.

- Um **script** é simplesmente uma seqüência de comandos MATLAB® e utiliza variáveis do **workspace**. Isso significa que todas as variáveis de um *script* são salvas no **workspace**. Não apresenta parâmetros de entrada nem de saída.
- Uma **função** também realiza uma seqüência de comandos. Diferentemente de um *script*, uma função possui parâmetros de entrada e pode retornar parâmetros de saída

Abra o editor e digite o seguinte:

```
function [y,x] = Eq_2_grau(a,b,c,x0,xf,n)
```

```
% Explicação que irá aparecer no help
```

```
x = linspace(x0,xf,n);
```

```
y = a*x.^2 + b*x + c;
```

```
end
```

Salve!!

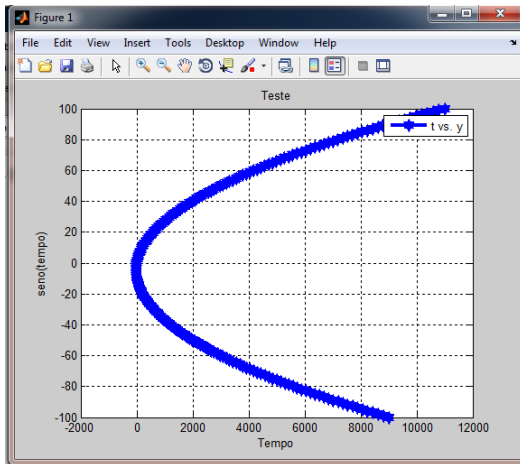
No *Command Window* faça o gráfico da seguinte Equação.

$$y = x^2 + 10x + 3$$

com x variando entre -100 até 100 .

Função

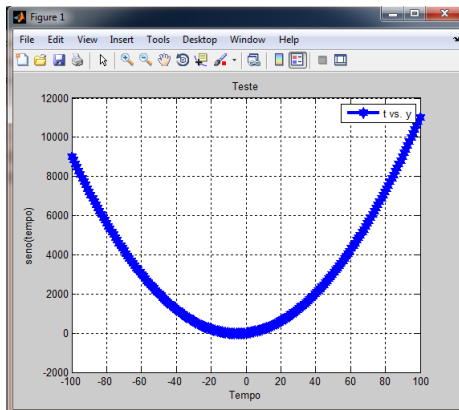
```
>> [x,y]=Eq_2_grau(1,10,3,-100,100,200);  
>> createfigure(x,y)
```



Função

Cuidado com a ordem, pois foi declarado "[y,x]" e ao usar a função, chamei "[x,y]".

```
>> [y,x]=Eq_2_grau(1,10,3,-100,100,200);  
>> createfigure(x,y)
```



Script

Vamos criar um Script que chama que chama a função `Eq_2_grau` e depois chama a função que faz o gráfico.

Novamente abra um novo editor.

```
% Este e um exemplo de script que resolve uma Equacao de 2 ordem  
%%
```

```
clc
```

```
clear all
```

```
close all
```

```
%% Declarando as variaveis
```

```
a = 1; b = 10; c = 3;
```

```
x_inicial = -100; x_final = 100; n_pontos = 200;
```

```
ylab = 'f(x)'; xlab = 'x'; tit = 'Exemplo Script'; leg = 'f(x)';
```

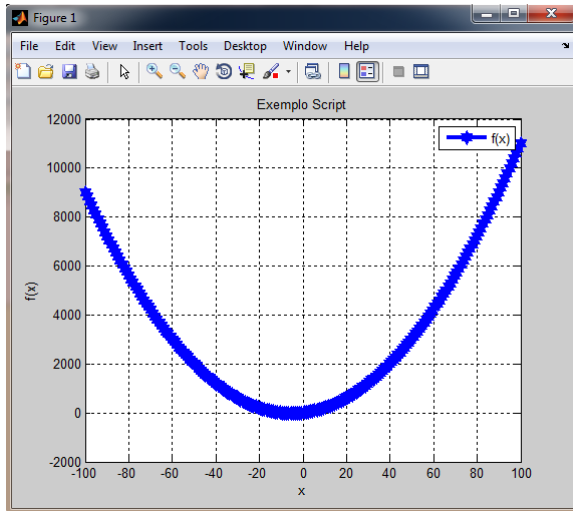
```
%% Chamando a funcao
```

```
[y,x] = Eq_2_grau(a,b,c,x_inicial,x_final,n_pontos);
```

```
figure1 = createfigure(x,y,ylab,xlab,tit,leg);
```

```
saveas(figure1,'Figure/Teste_Script.png');
```

Script



1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Operadores Lógicos

As expressões lógicas (Booleanas), são utilizadas em tomadas de decisões. Para o MATLAB[®]:

- 0 (ZERO) ou *null* → indica condição false;
- 1 (UM) ou *diferente de zero* → indica condição verdadeira.

Operador	Verdadeiro se	Exemplo
<code>==</code> ou <code>eq</code>	A igual a B.	<code>A==B</code> ou <code>>> eq(A,B)</code>
<code>~=</code> ou <code>ne</code>	A diferente de B.	<code>A~=B</code> ou <code>>> ne(A,B)</code>
<code><</code> ou <code>lt</code>	A menor que B.	<code>A<B</code> ou <code>>> lt(A,B)</code>
<code>></code> ou <code>gt</code>	A maior que B.	<code>A>B</code> ou <code>>> gt(A,B)</code>
<code><=</code> ou <code>le</code>	A menor ou igual que B.	<code>A<=B</code> ou <code>>> le(A,B)</code>
<code>>=</code> ou <code>ge</code>	A maior ou igual que B.	<code>A>=B</code> ou <code>>> ge(A,B)</code>

Operadores Lógicos

Nome	Operador Lógico	Descrição
AND	$A \& B$	Se ambos forem verdadeiros, o resultado será verdadeiro (1), se falso (0).
OR	$A B$	Se pelo menos um dos operandos for verdadeiro, o resultado será verdadeiro (1), se falso (0).
NOT	$\sim A$	Negação do operando.
XOR	<code>>> xor(7,0)</code>	Ou Exclusivo. Retorna (1) se houver desigualdade entre os operandos.
ALL	<code>>> all([6 2 3 6 7])</code>	Retorna (1) se todos os elementos de um vetor A forem diferentes de zero. Retorna (0) se um ou mais elementos forem (0).

Operadores Lógicos

Nome	Operador Lógico	Descrição
ANY	<code>>> any([6 0 3 0 0])</code>	Retorna (1) se qualquer elemento de A for diferente de zero. Retorna (0) se todos os elementos de A forem falsos.
FIND	<code>>> find([0 9 4 3 7])</code>	Se for um vetor, retorna os índices dos elementos diferentes de zero.
FIND	<code>>> find([0 9 4 3 7]>4)</code>	Retorna o endereço dos elementos que são maiores que d (qualquer operador relacional pode ser utilizado).

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

O MATLAB[®] possui estruturas para tomada de decisões, iguais às existentes em linguagens de programação estruturadas. As principais são as estruturas `for`, `if` e `while`. A tomada de decisão é baseada no resultado de uma expressão booleana. Se a expressão retornar `0` (ZERO), o MATLAB[®] interpreta condição falsa. Se uma expressão retorna um valor diferente de zero, condição verdadeira.

Laço FOR

O Laço FOR executa uma sequência de comando durante um numero especificado de vezes.

FOR - END

```
for variável = <valor inicial> : <incremento> : <valor final>  
    comandos  
end
```

Exemplo: Criação de um vetor formado por 10 múltiplos de 3.

```
>> for i = 1:10  
        v(i) = 3*i;  
>> end
```

DICA. Cuidado Laço Infinito

Comando **abort** → *Command Window* digite:

"CTRL" + "c"

Laço WHILE

O laço WHILE permite que uma sequência de comandos seja repetida enquanto uma certa condição for verdadeira.

WHILE - END

```
while <expressão condicional>  
    comandos (DEVE conter um comando que altere a condição,  
             senão entrará em laço infinito)  
end
```

Exemplo: Considere que se deseja determinar o maior valor de n tal que $n! < 10^{100}$.

```
>> n = 1;  
>> while prod(1:n) < 10e100  
    n = n + 1;  
>> end
```

O que é `prod`??

Digite: `help prod`

Condicional If

Se a expressão condicional for verdadeira (1), se executa os comandos abaixo da sentença `if`, até encontrar o `end`

IF - END

```
if <expressão condicional>  
    grupo 1 de comandos  
else  
    grupo 2 de comandos  
end
```

Exemplo:

```
>> for ii = 1:5  
    if ii == 3;  
        break;  
    end  
>> end
```


if - elseif - end

```
vetor = ['A','B','C','D','E','F','G','H','I','J','L','M','N','O','P','Q',...  
        'R','S','T','U','V','W','X','Z'];  n = length(vetor);  
for i = 1:n  
    if i == 20  
        msg(1) = vetor(i);  
    elseif i == 6  
        msg(2) = vetor(i);  
    elseif i == 1  
        msg(3) = vetor(i);  
    elseif i == 2  
        msg(4) = vetor(i);  
    elseif i == 3  
        msg(5) = vetor(i);  
    else  
        continue;  
    end  
end
```

Condicional Switch

Executa trecho de código de acordo com o valor contido em uma variável de teste.

SWITCH - CASE - END

```
switch <valor>
case <expressão caso 1>
    código 1
case <expressão caso 2>
    código 2
otherwise
    código
end
```

Exemplo Switch

```
dia = today;% today retorna o dia na forma serial
switch weekday(dia) % retorna o dia da semana (1 - 7)
    case 1
        display('Domingo');
    case 2
        display('Segunda');
    case 3
        display('Terça');
    case 4
        display('Quarta');
    case 5
        display('Quinta');
    case 6
        display('Sexta');
    case 7
        display('Sábado');
    otherwise
        display('Valor Inválido');
end
```

1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

O *Symbolic Math Toolbox* (SMT) são funções que executam operações de matemática algébrica e simbólica dentro do ambiente MATLAB[®].

Para visualizar qual versão esta instalada:

```
>> ver symbolic
```

O MATLAB[®] consegue identificar automaticamente o tipo de variável e chamar a rotina adequada: biblioteca matemática numérica para variáveis numéricas e biblioteca do SMT para variáveis simbólicas. Por exemplo:

```
>> x = sym('x'); % declarando variável x como symbolic
```

```
>> A=[sin(x),x;cos(x),x]
```

```
A =
```

```
[sin(x), x]
```

```
[cos(x), x]
```

A variável **A** contém uma matriz simbólica.

Criando uma Variável Simbólica

Comando `sym`:

```
>> a = sym('alpha');
```

- Requer parênteses e aspas. Exceto se for um número simbólico: `f=sym(5)`.
- Cria uma por vez. Melhor para a criação de números simbólicos e expressões simbólicas.

Comando `syms`:

```
>> syms x a y;
```

- Não usa parênteses e aspas.
- Pode-se criar várias variáveis.
- Serve melhor para a criação de variáveis simbólicas simples e múltiplas individuais.

Alguns exemplos:

<code>x = sym('x','real')</code>	Assume que a variável x é real.
<code>x = sym('x','positive')</code>	Assume que x é real e positivo.
<code>x = sym('x','clear')</code>	Limpa o que havia assumido.
<code>x = sym('x',[m n])</code>	Cria uma matriz m por n .
<code>x = sym('x',n)</code>	Cria uma matriz n por n .
<code>x = sym('x',flag)</code>	Cria um escalar numérico ou matriz.

Onde `flag` pode assumir: 'r' (default), 'd', 'e' ou 'f'.

'r' (racional)	<code>sym(4/3,'r')</code> .
'd' (decimal)	<code>sym(4/3,'d')</code> .
'e' (erro estimado)	<code>sym(4/3,'e')</code> .
'f' (ponto flutuante)	<code>sym(4/3,'f')</code> .

- Integral

```
x = sym('x');  
y = int(exp(x)*sin(x))
```

- Eq. 2 grau

```
eq2 =  
sym('a*x ^ 2+b*x+c');  
s = solve(eq2,'x')
```

- Visualização

```
pretty(s)
```

- Derivada parcial

```
d = diff(y,x,1)  
d2 = diff(y,x,2)
```

- Limite

```
syms h  
limit((sin(x+h)-sin(x))/h,h,0)
```

- Simplifica

```
simplify(d)
```

- Expande

```
expand(sin(h+x))
```

- Agrupar

```
f=4*x*exp(x)+3*exp(x);  
collect(f,exp(x))
```

- Substituição

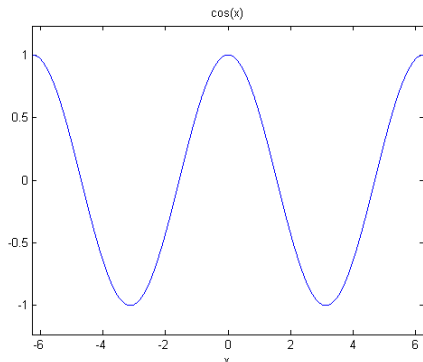
```
subs(y,x,pi)
```



```
syms h x  
li=limit((sin(x+h)-sin(x))/h,h,0);  
ezplot(li,[-2*pi,2*pi])
```

Dica

Quem usa LaTeX, pode usar o comando `latex()`.



1 Introdução

- Ambiente
- Símbolos Especiais
- Funções Matemáticas Elementares

2 Comando e Funções Básicas

- Declarando Variáveis
- Declarando Variáveis: Vetor
- Declarando Variáveis: Matriz

3 Gráficos

- Gráficos: plot Comando
- Gráficos: plot Interativo

4 Programação no Matlab

- Programação
- Operadores Lógicos
- Controladores de fluxo
- Variável Simbólica

5 Exemplo

- Exemplo: Suspensão

Sprintf

Para visualização de dados: `str = sprintf(formato,A1,...,An)`

A saída vai para uma cadeia de caracteres. Controle completo sobre a cadeia de caracteres. Útil para a criação de títulos e legendas complexos.

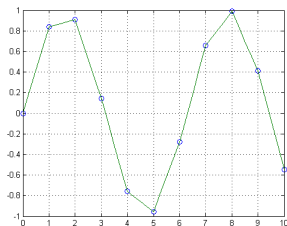
<code>%d</code> or <code>%i</code>	Base 10
<code>%o</code>	Base 8
<code>%x</code>	Base 16, minuscúlo
<code>%X</code>	Base 16 maiúsculo
<code>%f</code>	Notação ponto fixo
<code>%e</code>	Notação exponencial
<code>%c</code>	Carácter único
<code>%s</code>	Cadeia de caracteres

Sprintf

```
dh = clock;  
str_dh = sprintf('%04d%02d%02d %02d%02d%02.0f',...  
dh(1), dh(2), dh(3), dh(4), dh(5), dh(6))  
sprintf('Alfabeto \n%s',65:89)  
sprintf('Em caso de emergência:\nPolícia:%1.0f%d%1.0f\n  
Bombeiros:%c%c%1.0f\nPizza:  Anuncie  
aqui',1.3*cos(2-2),9,sin(2-2),'1','9',pi)
```

interp1

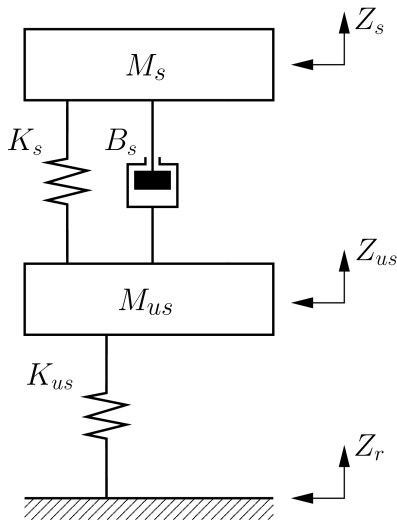
Interpolação de pontos, dados os vetores (\mathbf{X} , \mathbf{Y}) de pontos no espaço e $\mathbf{X_I}$ valores de interpolação e um método de interpolação retorna um vetor $\mathbf{Y_I}$, que formam a curva ($\mathbf{X_I}$, $\mathbf{Y_I}$) que melhor se aproxima os pontos no plano dado (\mathbf{X} , \mathbf{Y}).



interp1

```
x = 0:10;  
y = sin(x);  
xi = 0:.25:10;  
yi = interp1(x,y,xi);  
plot(x,y,'o',xi,yi);grid
```

Modelo da Suspensão Passiva



Equações:

$$\begin{aligned}M_s \ddot{Z}_s &= -K_s(Z_s - Z_{us}) - B_s(\dot{Z}_s - \dot{Z}_{us}) \\M_{us} \ddot{Z}_{us} &= K_s(Z_s - Z_{us}) + B_s(\dot{Z}_s - \dot{Z}_{us}) \\&\quad + K_{us}(Z_r - Z_{us})\end{aligned}$$

Renomeando:

$$\begin{aligned}x_1 &= Z_s - Z_{us}; & x_2 &= \dot{Z}_s; \\x_3 &= Z_{us} - Z_r; & x_4 &= \dot{Z}_{us};\end{aligned}$$

Reescrevendo:

Modelo da Suspensão

Reescrevendo:

$$\begin{aligned}\dot{x}_1 &= x_2 - x_4; \\ \dot{x}_2 &= -\frac{K_s}{M_s}x_1 - \frac{B_s}{M_s}(x_2 - x_4); \\ \dot{x}_3 &= x_4 - \dot{Z}_r; \\ \dot{x}_4 &= \frac{K_s}{M_{us}}x_1 + \frac{B_s}{M_{us}}(x_2 - x_4) - \frac{K_{us}}{M_{us}}x_3;\end{aligned}$$

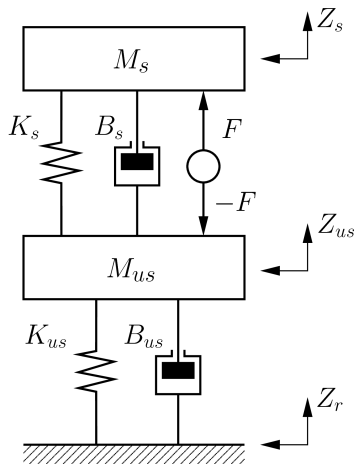
Forma matricial:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ -\frac{K_s}{M_s} & -\frac{B_s}{M_s} & 0 & \frac{B_s}{M_s} \\ 0 & 0 & 0 & 1 \\ \frac{K_s}{M_{us}} & \frac{B_s}{M_{us}} & -\frac{K_{us}}{M_{us}} & -\frac{B_s}{M_{us}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \dot{Z}_r$$

Modelo da Suspensão Passiva

Abrir o pdf

Modelo da Suspensão Ativa



Equações:

$$M_s \ddot{Z}_s = -K_s(Z_s - Z_{us}) - B_s(\dot{Z}_s - \dot{Z}_{us}) + F$$

$$M_{us} \ddot{Z}_{us} = K_s(Z_s - Z_{us}) + B_s(\dot{Z}_s - \dot{Z}_{us}) + K_{us}(Z_r - Z_{us}) - B_{us}(\dot{Z}_{us} - \dot{Z}_r) - F$$

Renomeando:

$$\begin{aligned} x_1 &= Z_s - Z_{us}; & x_2 &= \dot{Z}_s; \\ x_3 &= Z_{us} - Z_r; & x_4 &= \dot{Z}_{us}; \end{aligned}$$

Reescrevendo:

Modelo da Suspensão Ativa

Reescrevendo:

$$\begin{aligned}\dot{x}_1 &= x_2 - x_4; \\ \dot{x}_2 &= -\frac{K_s}{M_s}x_1 - \frac{B_s}{M_s}(x_2 - x_4) + \frac{F}{M_s}; \\ \dot{x}_3 &= x_4 - \dot{Z}_r; \\ \dot{x}_4 &= \frac{K_s}{M_{us}}x_1 + \frac{B_s}{M_{us}}(x_2 - x_4) - \frac{K_{us}}{M_{us}}x_3 - \frac{B_{us}}{M_{us}}(x_4 - \dot{Z}_r) - \frac{F}{M_{us}};\end{aligned}$$

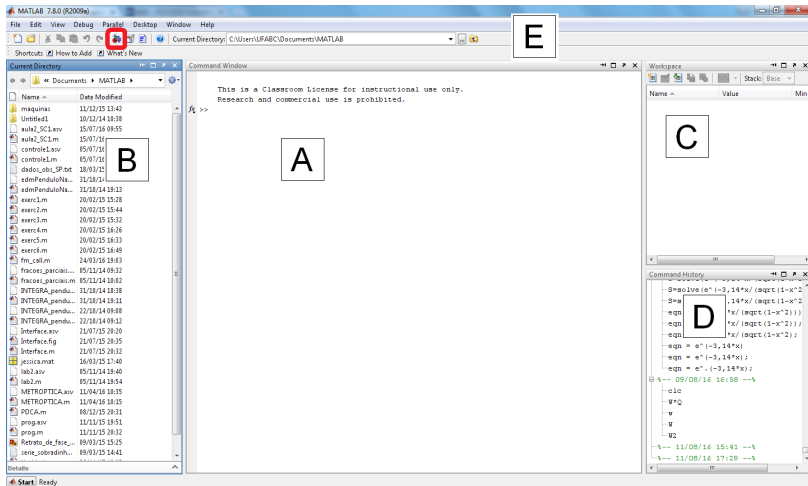
Forma matricial:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ -\frac{K_s}{M_s} & -\frac{B_s}{M_s} & 0 & \frac{B_s}{M_s} \\ 0 & 0 & 0 & 1 \\ \frac{K_s}{M_{us}} & \frac{B_s}{M_{us}} & -\frac{K_{us}}{M_{us}} & -\frac{(B_s+B_{us})}{M_{us}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{M_s} \\ -1 & 0 \\ \frac{B_{us}}{M_{us}} & -\frac{1}{M_{us}} \end{bmatrix} \begin{bmatrix} \dot{Z}_r \\ F \end{bmatrix}$$

Abrir o pdf

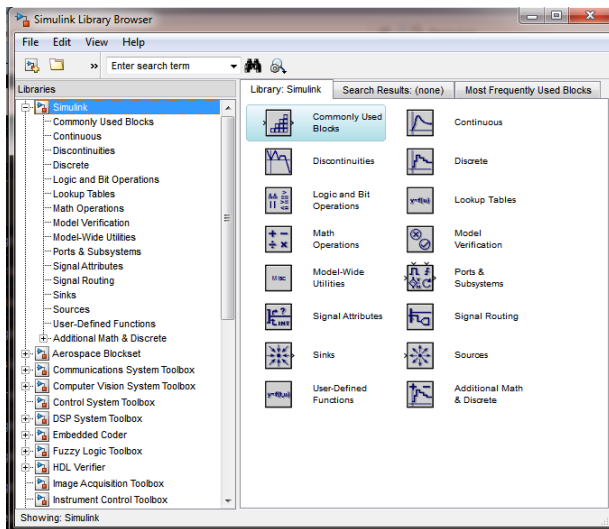
SIMULINK®

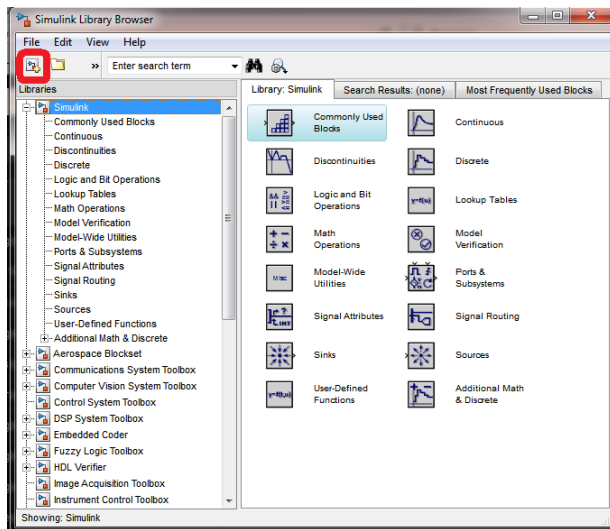
Simulink

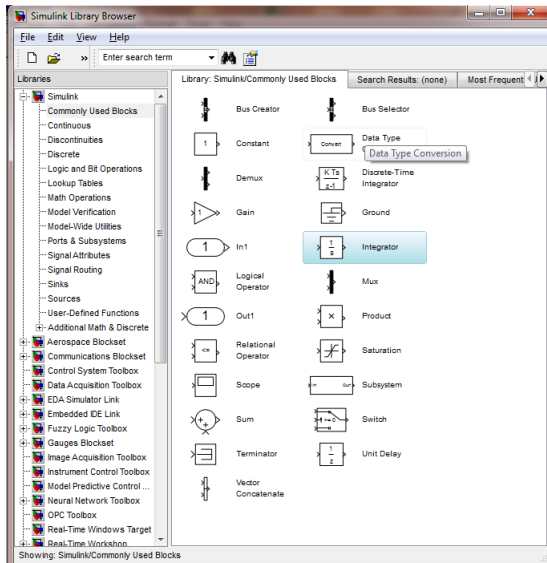


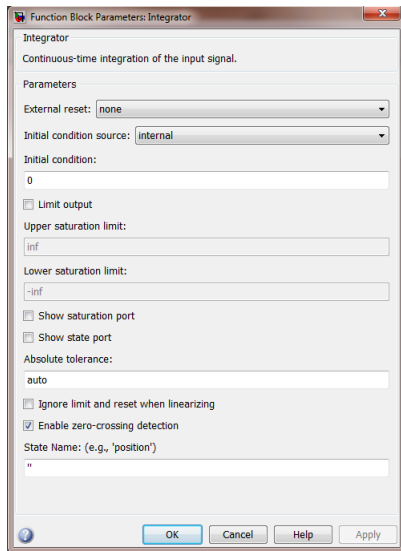
ou

>> simulink












Referências & Links Interessantes I

 Stephen J. Chapman.
Programação em MATLAB[®] para engenheiros.
Thomson Learning, 2006.

 Élia Yathie Matsumoto.
MATLAB[®] R2013a - Teoria e programação: guia prático.
Editora Érica, 2013.

 Élia Yathie Matsumoto.
MATLAB[®] 7 fundamentos.
Editora Érica, 2008.

 QUANSER INNOVATE EDUCATE.
Active Suspension Control Laboratory: Instructor Manual. Revision 2.0. Quanser Innovate Educate 48 f.. 2010. (Document Number, 845)