MATLAB
O que se deve aprender ...

Prof. José Flávio Feiteira
Prof. Diomar Cesar Lobão
UFF - Universidade Federal Fluminense
Volta Redonda, 2007

SEMANA de Engenharia: Nov 2007

Conteúdo:

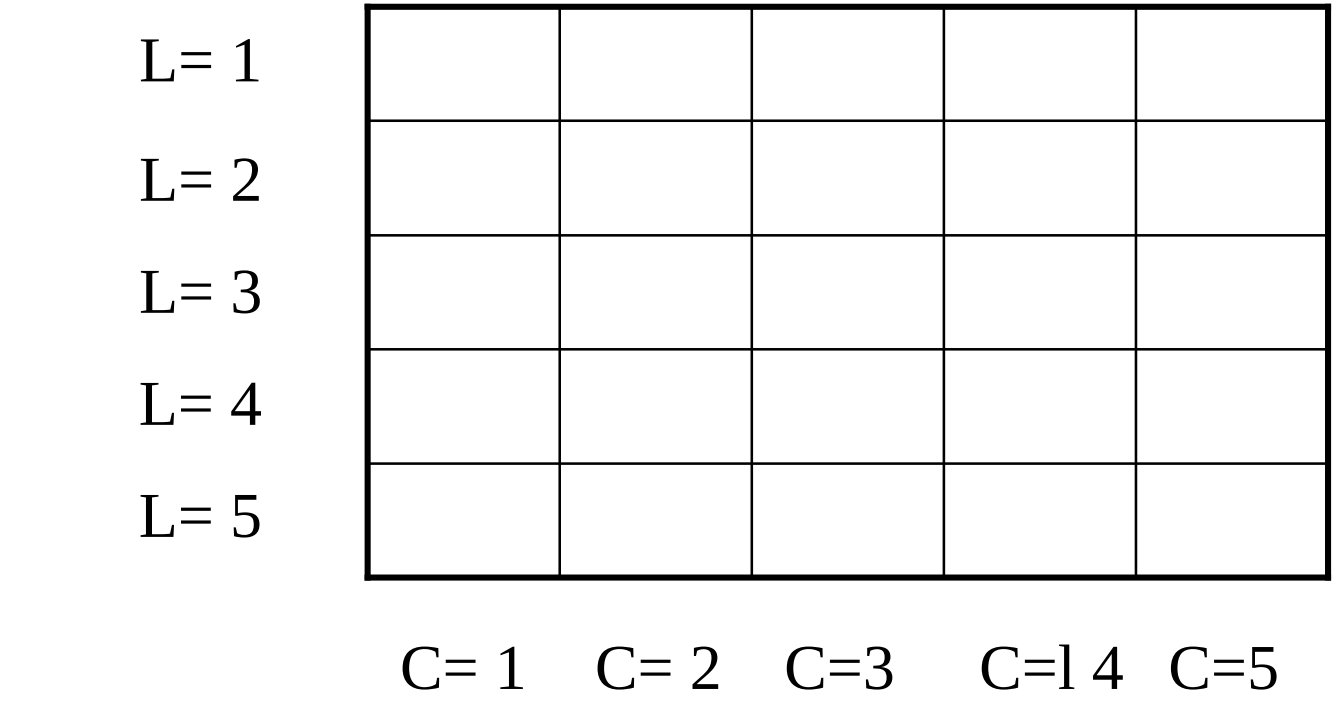
Introdução:

1. Calcular: atividade inerente ao ser humano; sistema decimal
2. Dados numéricos; caracteres; lógicos;
3. Computador: máquina para resolver problemas
4. Computador: *hardware* e *software*;
5. Algoritmo, Programas, Linguagens (sintaxe e semântica)
6. O conceito de variável
7. Estruturas Básicas de Controle
8. Matlab: laboratório de matemática ?
9. Atribuição
10. Entrada / Saída / Condicionais e Repetições

Arrays, Matrizes e Vetores

- **Array** – coleção de dados organizados em *linhas* e *colunas* e conhecido por meio de uma variável simples

Nome(L,C) =



Vetores e Matrizes

- **Vetor –** é um array uni-dimensional escrito em uma linha ou coluna

$$b = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \qquad c = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

- **Matriz–** um array multi-dimensional

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Índices ARRAY

- Se o índce é uni-dimensional temos um vetor
  - x(1)= Primeiro elemento
  - x(2)= Segundo elemento, etc.
- Se o índice é bi-dimensional temos uma matriz
  - x(R,C) onde R=número de Linha & C=número de coluna
  - x(R,:) todos os elementos na linha R
  - x(:,C) todos os elementos na coluna C
  - R e C pode ser um conjunto
  - R=1:3 os primeiros 3 ou R=2:6

Índices de ARRAY

- Arrays Multidimensionais são acessados via índices

A =

1	2	3
4	5	6
7	8	9
10	11	12

1	A(1,1)	A(1)
4	A(2,1)	A(2)
7	A(3,1)	A(3)
10	A(4,1)	A(4)
2	A(1,2)	A(5)
5	A(2,2)	A(6)
8	A(3,2)	A(7)
11	A(4,2)	A(8)
3	A(1,3)	A(9)
6	A(2,3)	A(10)
9	A(3,3)	A(11)
12	A(4,3)	A(12)

CRIANDO VETORES em LINHA

- x=***início :incremento:final***
  - o incremento é “um” por default quando não definido
- x=***linspace***(***início, final, número de valores***)
  - o número de valores pode ser requerido como entrada
- x=***logspace(início potência, potência final, número de valores)***
  - 10^X1 e 10^X2
- Vetores são formados termo a termo =>[1 2 3]

VETOR em COLUNA?

- Entre termo por termo

  x=[1;2;3;4;5;6;7;8;9]

- Ou entre como linha e aplica a transposta it (‘)

  x=1:9, x=x’

- Para números complexos, a transposta fornece o conjugado – ponto transposto (.’) não

  f=3+2i; f’=>3-2i

  x=[1+j;1+j;9+9j] => x=x.’ => não!

  x=x’ => sim!, obtém conjugado

ENTRANDO com Matriz

- x=[1 4 6 ; 2 9 4 ; 3 6 1]

Cada linha é entrada e uma nova linha é definida após “ ; ”

  – Todas as colunas devem ter o mesmo tamanho

- x=[1 4 6
  2 9 4
  3 6 1]

  – Tecla ‘enter’ no fim de cada linha

Operador ‘:’ (Colon Operator)

- Usado como referência de toda linha ou coluna

$$C = [-1,-0,0;\ 1,1,0;\ 1,-1,0;\ 0,0,2]$$

4 Linhas, 3 colunas

```
x = C(:,1);
y = C(:,2);
z = C(:,3);
```

```
C =
    -1     0     0
     1     1     0
     1    -1     0
     0     0     2
```

```
x =     y =     z =
    -1      0       0
     1      1       0
     1     -1       0
     0      0       2
```

Operações com Submatriz

- Um subconjunto de uma dada matriz
- Experimente:

$C\_partial1 = C(:,2:3);$ Todas linhas, 2 e 3 colunas

$C\_partial2 = C(3:4,1:2);$ 3 e 4 linhas e 1 e 2 colunas

C_partial1 =

0	0
1	0
-1	0
0	2

C_partial2 =

-1	-1
0	0

Operações com Matriz

- Operações Matriz: *, /, + , -,^, inv(A), \
- Elemento de Array com elemento precede *, /, ^ ➔ por um **“ . ”**
  - **A*A não é o mesmo que A.*A !**
- **Ax = b,**
- **$X=A^{-1}b$ – Matematicamente**
- **x= inv(A)*b** ou **A \ b <= Matlab**

Solução de sistema de equações
x é incógnita

$$\begin{aligned} 2y + 3z &= 8 \\ 2y + 4z &= 6 \end{aligned} \longrightarrow Ax = b \longrightarrow x = A^{-1}b \longrightarrow \texttt{x=A\b}$$

$$A = \begin{bmatrix} 2 & 3 \\ 2 & 4 \end{bmatrix} \qquad x = \begin{bmatrix} y \\ z \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ 6 \end{bmatrix}$$

"\" => Matlab usa Eliminação de Gauss

Input do Usuário

- Algumas vezes é necessário perguntar ao usuário para entrar com dados e armazená-los **`input`**…

```
R = input('Entre valores p/ 3 resistores em paralelo entre[ ]');
```

- Agora, calcule a resistência equivalente:

```
Req = 1/(1/R(1) + 1/R(2) + 1/R(3))
```

Opções de Output

```
FORMAT          Default. Same as SHORT.
FORMAT SHORT    Scaled fixed point format with 5 digits.
FORMAT LONG     Scaled fixed point format with 15 digits.
FORMAT SHORT E  Floating point format with 5 digits.
FORMAT LONG E   Floating point format with 15 digits.
FORMAT BANK     Fixed format for dollars and cents.
```

Escrevendo Texto e Valores

`disp(Req)`← Escreve o valor armazenado na variável “Reg”

`EXEMPLO:`

`disp(‘Resistencia Ohms’)` ← Escreve o texto entre apóstrofe

Output Formatado:

`fprintf(format,variáveis)`

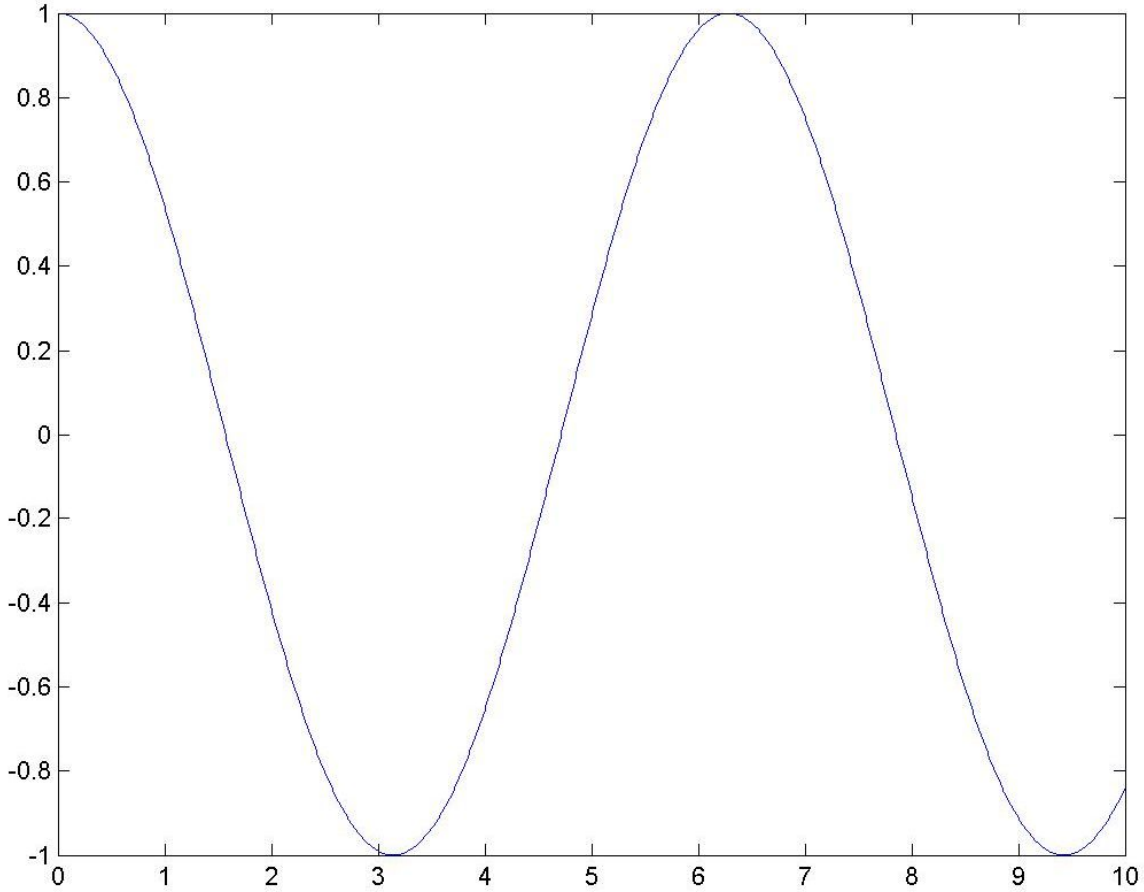
```
fprintf('A resistencia equivalente e %f ohms \n', Req)
```

`%f, %e, %g -` formato p/ as variáveis, elas são repostas aos valores armazenados nas variáveis

`\n -` controle, significa: ***alimenta um linha***.

Plotagem Básica

- Seja: x=0:0.01:10; y=cos(x); plot(x,y)



Algumas dicas

- Subplotagem

`subplot(m,n,p)`

`m -> linhas de plots`

`n -> colunas de plots`

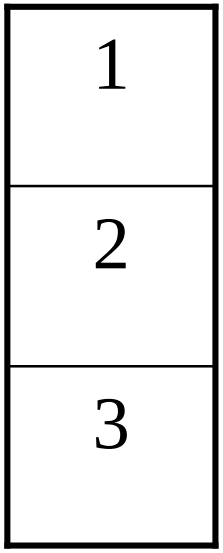
`p -> atual plot`

Plots são numerados da esquerda p/ direita de cima p/ baixo

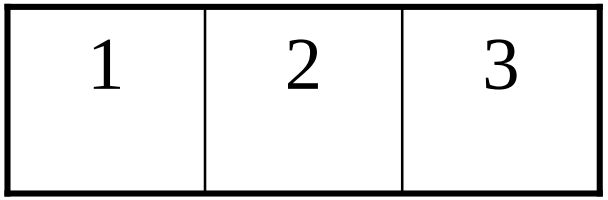
Numeração de Subplot

EXEMPLOS:

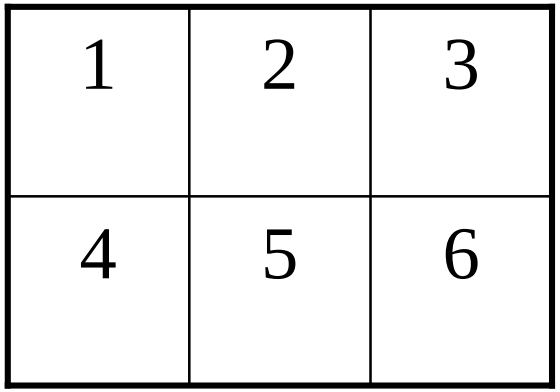
`subplot(3,1,p)`



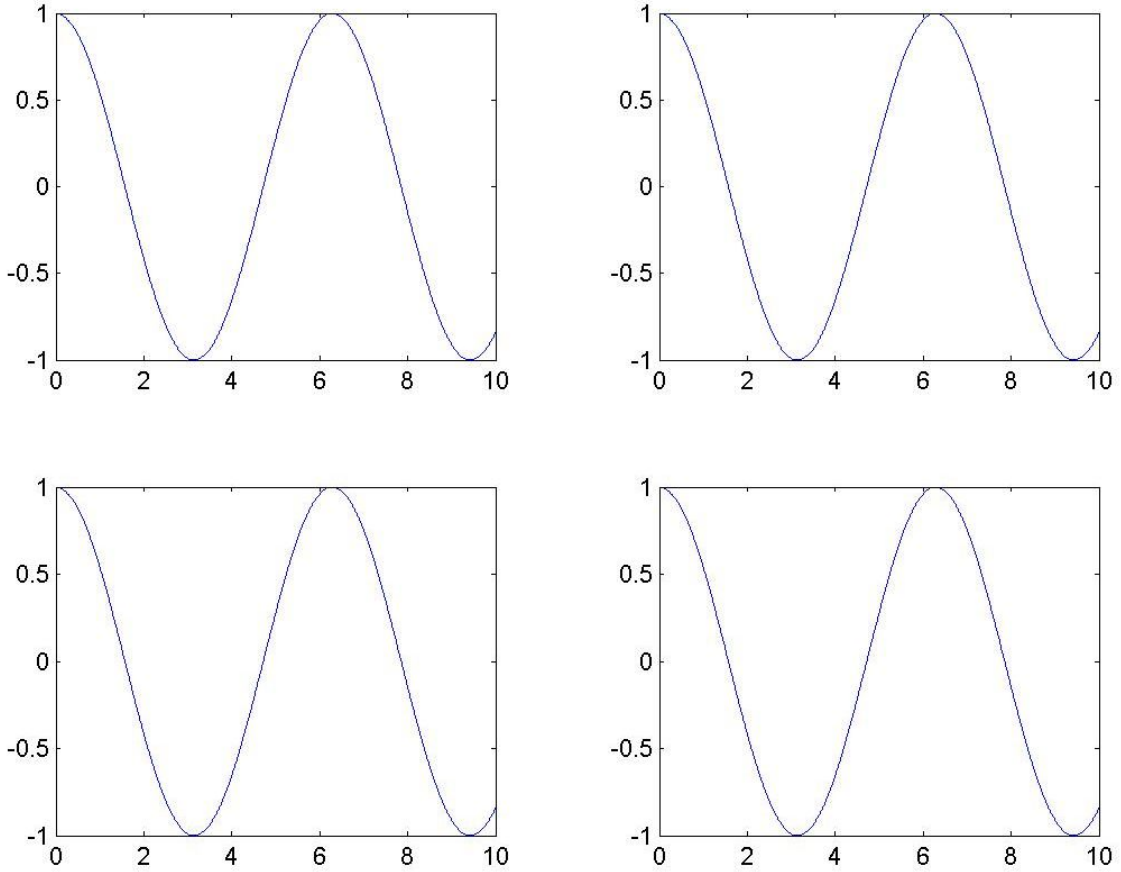
`subplot(1,3,p)`



`subplot(2,3,p)`



```
x=0:0.01:10; y=cos(x);
subplot(2,2,1),plot(x,y)
subplot(2,2,2),plot(x,y)
subplot(2,2,3),plot(x,y)
subplot(2,2,4),plot(x,y)
```



Algumas dicas mais

- Controlando eixos x- e y- Limites

```
axis ([xmin xmax ymin ymax]);
```

- Plotando Multiplos Plots sobre mesmo eixo

```
plot(x,y1)
hold on;
plot(x,y2)
hold off
```

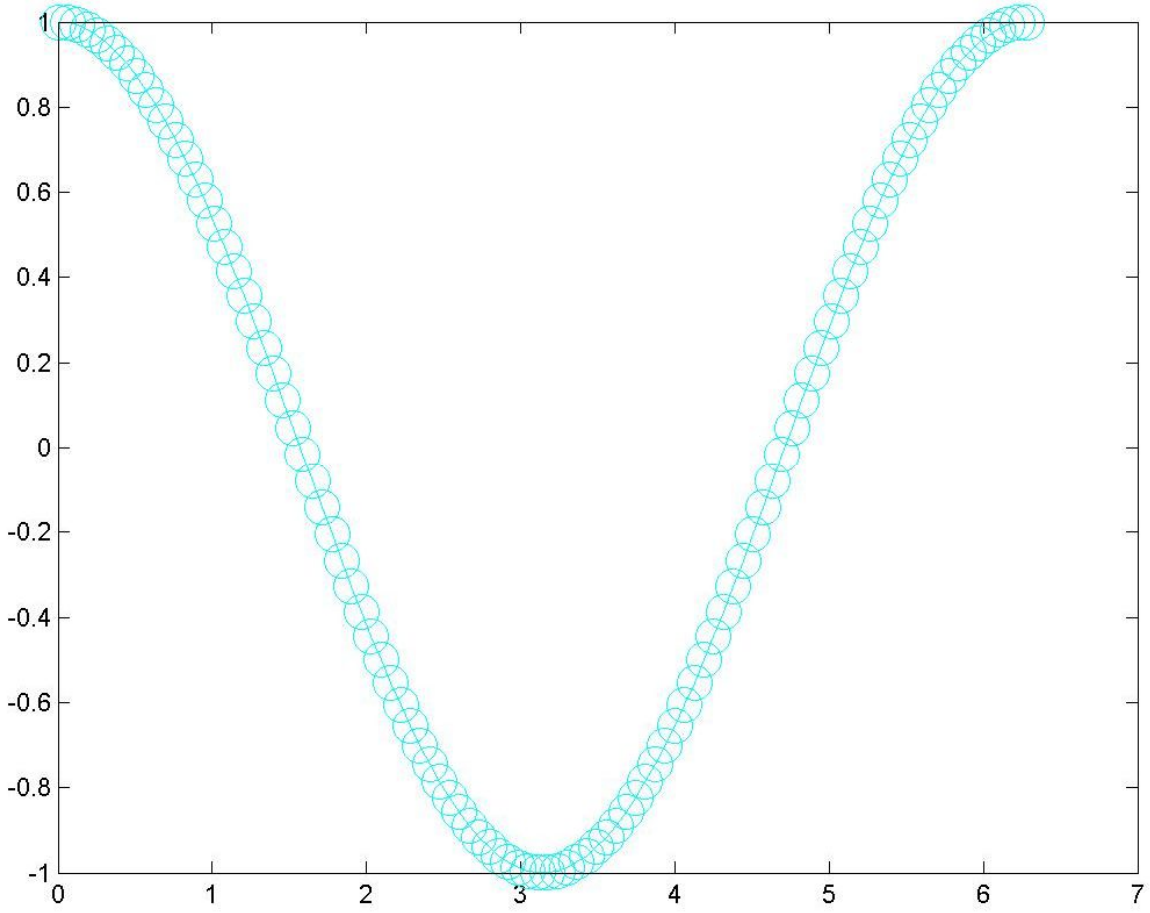
Função *Handle:* Gráfico

- Handle – uma variável que identifica um objeto particular de um gráfico (figura, lineplot, axes, x e y label, title, text, legend, etc.)
- Com handle pode-se então customizar as propriedades de um objeto particular (font, font size e weight, line widths e colors)
- Quando se cria um objeto (uma figure, plot line, axes, um title or text), um handle é criado e salvado em uma variável.

Gráfico :Handle

Vejamos o exemplo de: y=cos(x)

```
x = linspace ( 0, 2*pi, 100 );
y = cos(x);
id = plot ( x, y )
set ( id, 'Marker', 'o' )
set ( id, 'MarkerSize', 15 )
set ( id, 'Color', 'Cyan' )
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Comando Get

- Pode-se ver quais são as propriedades de qualquer handle digitando: **`get(`*`handlename`*`)`**
- **`get(id)`** retorna dois grupos de propriedades que são separadas por uma linha em branco
- O primeiro grupo é uma lista de propriedades única de um objeto ‘line’
- O segundo grupo são as propriedades em comum

get(id): Comando no arquivo.m

Color = [0 1 1]
EraseMode = normal
LineStyle = -
LineWidth = [0.5]
Marker = o
MarkerSize = [15]
MarkerEdgeColor = auto
MarkerFaceColor = none
XData = [(1 by 100) double array]
YData = [(1 by 100) double array]
ZData = []

BeingDeleted = off
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [101.009]
Selected = off
SelectionHighlight = on
Tag =
Type = line
UIContextMenu = []
UserData = []
Visible = on

Comando Set

- Usado para alterar qualquer propriedade de *um objeto gráfico*
- **`set(handlename)`** fornece uma lista de todas as propriedades que *se pode mudar*
- O primeiro grupo lista as propriedades das *linhas*
- O segundo grupo lista as propriedades em *comum*

set(id)

Color

EraseMode: [{normal} | background | xor | none]
LineStyle: [{-} | -- | : | -. | none]
LineWidth
Marker: [+ | o | * | . | x | square | diamond | v | ^ | > | < | pentagram | hexagram | {none}]
MarkerSize
MarkerEdgeColor: [none | {auto}] -or- a ColorSpec.
MarkerFaceColor: [{none} | auto] -or- a ColorSpec.
XData
YData
ZData

ButtonDownFcn: string -or- function handle -or- cell array
Children
Clipping: [{on} | off]
CreateFcn: string -or- function handle -or- cell array
DeleteFcn: string -or- function handle -or- cell array
BusyAction: [{queue} | cancel]
HandleVisibility: [{on} | callback | off]
HitTest: [{on} | off]
Interruptible: [{on} | off]
Parent
Selected: [on | off]
SelectionHighlight: [{on} | off]
Tag
UIContextMenu
UserData
Visible: [{on} | off]

Mudando as Propriedades

- Use o comando ***set*** para mudar uma propriedade

set (id, 'Marker', 'o')

set (id, 'MarkerSize', 15)

set (id, 'Color', 'Cyan')

Mudando o Título

`> title_handle=title('Coseno Exemplo');`

`> set(title_handle)` mostra uma única propriedade

- BackgroundColor
- Color
- EdgeColor
- EraseMode: [{normal} | background | xor | none]
- Editing: [on | off]
- FontAngle: [{normal} | italic | oblique]
- FontName
- FontSize
- FontUnits: [inches | centimeters | normalized | {points} | pixels]
- FontWeight: [light | {normal} | demi | bold]
- HorizontalAlignment: [{left} | center | right]
- LineStyle: [{-} | -- | : | -. | none]
- LineWidth
- Margin
- Position
- Rotation
- String
- Units: [inches | centimeters | normalized | points | pixels | characters | {data}]
- Interpreter: [{tex} | none]
- VerticalAlignment: [top | cap | {middle} | baseline | bottom]

Mudando o Título

`set(title_handle,'Color',[.2 .7 .4])` ou usando cores padrão (‘b’, ‘w’, ‘k’ etc.)

`set(titlehandle,'FontSize',20,'FontName',`
`'Algerian' ,'FontWeight','bold')`
fixa o tamanho da fonte e tipo para *Algerian* e peso ‘*bold*’

Mudando a Figura

`set(id)`

Lista as propriedades que podem ser mudadas

set(id,'Color',[.8 1 .8]) => muda a cor do background fora do gráfico

Chamando o Handle do Axes

`axes_handle=gca`

Fornece o handle do eixo da figura atual

`set('axes_handle','Xtick',[0:.01:.1])`

Permite especificar "tick marks" em cada eixo

Funções Internas do Matlab

- Comandos e funções do Matlab são divididas em muitas categorias
- Digite: ***help elfun***
  - Funções trigonométricas
  - Funções exponencial
  - Funções de número complexo
  - Funções de arredondamento/resto

Funções Matemáticas comuns do Matlab

- *Trigonométrica* – **`cos, sin, tan, atan`**
  - Assegure de usar ***radianos*** no argumento
- *Raíz quadrada* – **`sqrt(2)`** produz: 1.414
- *Log Natural* – **`log(3)`** produz: 1.0986
- *Log Base 10* – **`log10(4)`** produz: 0.6021
- *Exponencial* – **`exp(2)`** produz: 7.3891

Funções Matlab

- Digite: ***help ops***
  - Funções aritmética
  - Funções relacional
  - Funções lógicas
  - Funções de caracteres especiais
  - Operadores: “bitwise operators”
  - Operadores de conjuntos

Funções Matlab

- Digite: ***help datafun***
  - Funções básicas estatísticas
  - Diferenças finitas
  - Correlação
  - Filtragem digital
  - Transformadas de Fourier
  - Som e áudio

Funções Matlab

- Digite: ***help general***
  - Informação sobre funções
  - Funções de administração de “workspace”
  - Funções do sistema operacional
  - Funções de “debug”
- Digite: ***help timefun*** (tempo e data)
- Digite: ***help datatypes*** (classes/objetos-orientados)

Funções Matlab

- Digite: ***help elmat***
  - Matrizes elementares
  - Informação sobre array's
  - Manipulação de matrizes
  - Variáveis especiais
  - Matrizes especiais
- Digite: ***help matfun*** (Matrizes avançadas)
- Digite: ***help sparfun*** (Matrizes esparsas)

Funções Matlab

- Digite: ***help lang*** (programação)
  - controle fluxo
  - Avaliação e execução
  - Função script
  - Manuseio de argumentos
  - Escrevendo mensagens
  - “input” interativo

Funções Matlab

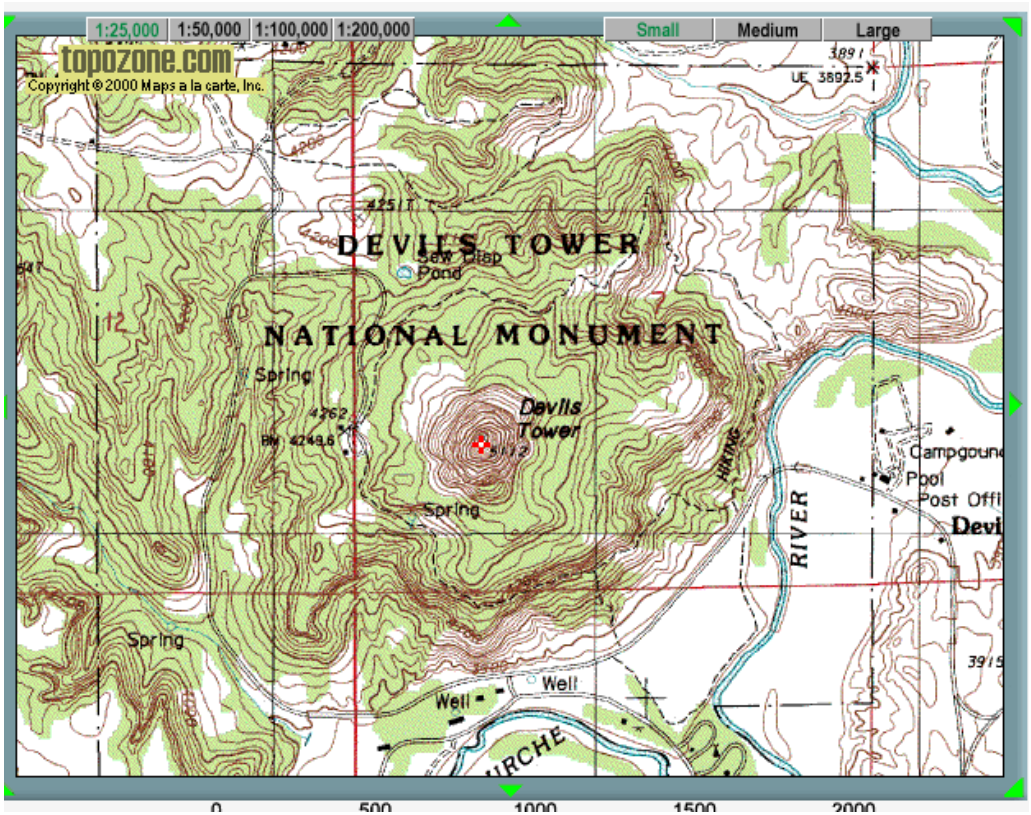
- Digite: ***help graph2d***
  - Tipo de gráfico
  - Controle dos eixos
  - ‘label’ dos gráficos
- Digite: ***help graphics***
- Digite: ***help graph3d***

Funções Matlab

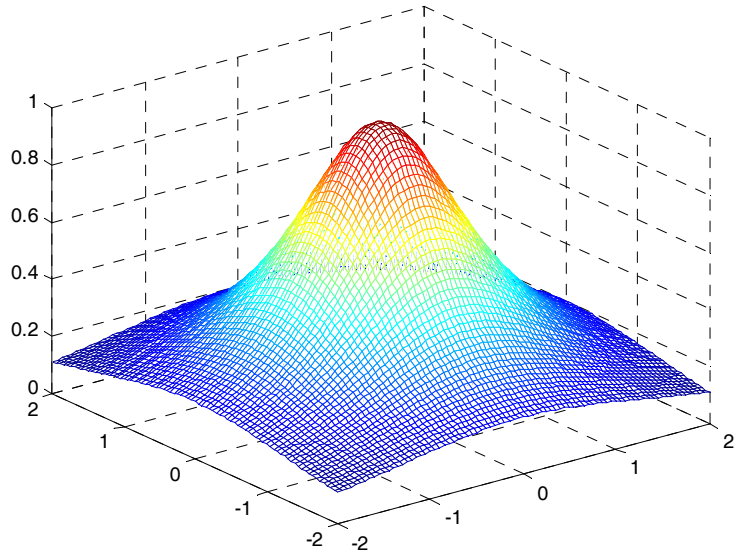
- Digite: ***help uitools*** (GUI interfaces)
- Digite: ***help strfun*** (character strings)
- Digite: ***help iofun*** (input/output)
- Digite: ***help funfun*** (eq. Diferencial ordinárias, *)
- Digite: ***help polyfun*** (polinômios)
- * Integração: quad, quadl, dblquad, triplequad

Funções Multidimensionais

- Exemplos de mapas com relevo



Gráficos com Multi-Variáveis



- *Crie os limites de domínio 2-d*

```
X=-2:0.05:2;
Y=-2:0.05:2;
```

- *Crie uma malha 2-d*

```
[x_grid,y_grid] = meshgrid(x,y)
```

- *Crie a função*

```
Z=1./(1+x_grid.^2+y_grid.^2);
```

- *Plote no domíno 2-d*
- **`mesh(x_grid,y_grid,z)`**

Operadores Relacional

- Operadores Relacional, com dois argumentos numéricos ou caracteres que produz resultado ***true(1)*** ou ***false(0)***

  A_1 ***operador*** A_2

Operador	Operação
= =	Equal to
~ =	Not equal
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Operadores Lógicos

- Operadores com um ou dois argumentos que produz um resultado lógico
- Binário - **AND, OR** e exclusivo **XOR**
  - A ***operador*** B
- Unário
  - ***Operador*** A

Inputs		**and**	**or**	**xor**	**not**
A	B	A & B	A | B	xor(A,B)	~A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Operador Lógico

- Aplica à escalares, vetores e matrizes ‘arrays’
- Para ‘arrays’ aplica à cada elemento
- Os ‘arrays’ para comparação devem ter o mesmo número de elementos

$$a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ e } \quad b = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \text{ então } a \mid b = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

or

IF ELSEIF ELSE END

- Usado para controlar o fluxo de execução

```
if expressão lógica
   executa alguns comandos
end
```

- Execução ocorre se a expressão lógica é ‘true’ (igual 1)
- **break** pode ser usado dentro de um ‘***if***’, ‘***while***’ para sair fora do ‘loop’

```
if expressão lógica
    break
end
```

IF ELSEIF ELSE END

- Forma

  if ***expressão lógica***

  1º conj. de comandos

  elseif ***expressão lógica***

  2º conj. de comandos

  else % Opcional

  3º conj. de comandos

  end

- ***Somente um conjunto de comandos será executado***

EXEMPLO

- Achar as raízes equação quadrática $ax^2+bx+c=0$
- Há vários casos a ser considerado:
  - a=0, b=0 e c=0 – - sem sentido!!!
  - a=0 & b~=0 - a eq. Não é quadrática
  - a=0 & b=0 - a eq. Não é verdadeira
  - outra - duas raízes que podem ser reais, iguais ou complexas

Exemplo: Equação Quadrática

```
% 1º caso onde a=b=c=0
if  a==0&b==0&c==0   % note duplo '=='
    disp('Equacao sem sentido!!!')
% 2º caso onde a=0
elseif a==0&b~=0
    root=-c/b
    disp('Não e Quadratica')
    fprintf('Raiz simples = %g \n',root)
```

Continuação

```
% 3º Caso onde somente a=b=0
elseif a==0&b==0
    disp('Equacao não e verdadeira')
else   % 4º Caso onde existe 2 raizes
   root1=(-b+sqrt(b^2-4*a*c))/(2*a);
   root2=(-b-sqrt(b^2-4*a*c))/(2*a);
   disp(root1), disp(root2)
    if (b^2-4*a*c)<0    %duas raízes são complexas
       disp('Raizes Complexas')
```

Exemplo: Equação Quadrática

```
elseif (b^2-4*a*c)>0   %Raiz real dupla
    disp('Raiz real')
    x=linspace(root1,root2,100)
    plot(x,a*x.^2+b*x+c), title('Raiz real')   %plota a funcao
    xlabel('valor x'), ylabel('valor da Funcao')
else                          % Raiz unica  real
     root1=-b/(2*a);
    disp('Raiz única = %12.6 \n', root1)
end
end
```

Projeto de um Programa

- Projeto de um Programa – "*Procedural*"
  - Programação usando estrutura "***Top down***" – Divide grandes tarefas em ***subtarefas***.
  - Passos "***steps***"
    - Definir claramente o problema
    - Definir "inputs" e "outputs"
    - Projetar o algoritmo usando "***pseudocódigo"***
    - ***Traduz*** o algoritmo na ***linguagem MATLAB***
    - Testar com a execução do código

Programação “Top Down”

- **Definir claramente o problema**
  - Avaliar a funçao f(x,y) especificado x,y
- **Definir inputs e outputs**
  - Input – x and y
  - Output – f(x,y)
- **Projete o algoritmo usando pseudocódigo**
  - **Tarefas principais**

Leia os valores de input: x e y
Calcular f(x,y)
Escreva f(x,y)

  - **Sub tarefas**

Mostre pedido de entrada p/ x e y
Leia: x e y
If x =>0 and y => 0
    fun x +y
Elseif x <=0 and y< 0
    fun x +y^2
Elseif x <=0 and y => 0
    fun x^2 +y
Else
    fun x^2 +y^2
End
Escreva resultado de: f(x,y) ==> Fim!

$$f(x,y)=\begin{cases} x+y & x\geq 0 \text{ e } y\geq 0 \\ x+y^2 & x\geq 0 \text{ e } y<0 \\ x^2+y & x<0 \text{ e } y\geq 0 \\ x^2+y^2 & x<0 \text{ e } y<0 \end{cases}$$

- **Codifique o algoritmo em comandos de MATLAB**
- **Submeta um teste**

Estrutura de Repetição: FOR “LOOPS”

- Usada p/ repetir uma série de comandos um número de vezes determinado (“***For controlado***”)
- ***Forma de definição:***

```
for n=array,
    Conj. de comandos
end
```

- ‘array’ pode ser um vetor ou uma matriz

FOR “LOOPS”

- O índice “n” no comando **`for n=array`**, é usualmente um índice em algum ‘*array*’ sendo definido termo a termo no “*loop*”
- *Exemplo*

```
for n=1:1001,      % loop executa 1001 vêzes
   x(n)=sin(2*pi*(n-1)/1000);  % x é construído termo a
     termo
end
```

- É freqüente em Matlab usar vetores p/ substituir “loops”-***Vetorização***

```
n=1:1001;
x=sin(2*pi*.(n-1)/1000); DEVE Sempre ser USADO!
```

FOR “LOOPS”

- O “***loop***” *não pode ser terminado* prematuramente redefinindo ***n***
- Depois de executado, ***n*** será um valor simples = 1001, não um vetor, não se pode usar: *plot(n,x).* *Solução: plot(1:n,x)*
- Note que o índice começa com *n*=1 porém se a função começa com *tempo = 0*, então *(n-1)* é usado na função no lugar de *n*

FOR “LOOPS” Aninhados

- Pode-se criar qualquer ‘*array*’ usando múltiplos *for “loops”* aninhados um dentro dos outros
- ‘Arrays’ pode ser: 1, 2, 3 ou de dimensão maior
  - A(linhas, colunas, paginas, etc., etc., etc.)

```
for n=1:4,
   for m=1:3,
      A(n,m)=2*n-3/m;
   end
end
```

Estrutura de Repetição: While “Loops”

- Executa um grupo de comandos **`while(enquanto)`**uma expressão lógica é verdadeira “true”
- ***Forma de definição:***

```
while expresão lógica
    conjunto de comandos
end
```

- **Função:*`break`*** pode ser usada para terminar(exit)

a qualquer momento.

Exemplo: While

```
EPS=1;%EPS é Tolerância. Matlab:2.2204e-016
num=0;%num é o número de bits
while (1+EPS)>1
    EPS=EPS/2;
   num=num+1;
end
disp(2*EPS)
disp(num-1)
```

```
2.22044604925031e-016

   52
```

While com “Break”

```
EPS=1; num=0;
while num<100
      num=num+1;
      EPS=EPS/2;
   if (1+EPS)>1
      trueEPS=EPS; bits=num;
   else
      break
   end
end; disp(trueEPS), disp(bits)
```

```
2.220446049250313e-016

  52
```

Operações com Matrizes

- Porque ?
- *As matrizes são usadas largamente em engenharia para ajudar a simplificar a descrição de fenômenos físicos*:

Eq. Momentum da Mecânica dos Fluidos

$$\frac{\partial \rho \vec{V}}{\partial t} + \nabla \bullet (\rho \vec{V} \vec{V}) = -\vec{\nabla} p + \vec{\nabla} \bullet |T|$$

Eq. de Maxwell

$$\vec{\nabla} \bullet \vec{E} = \frac{\rho}{\varepsilon_o}$$

$$\vec{\nabla} \bullet \vec{B} = 0$$

$$\vec{\nabla} \times \vec{E} = \frac{-\partial \vec{B}}{\partial t}$$

$$\vec{\nabla} \times \vec{B} = \mu \vec{J} + \mu \varepsilon \frac{\partial \vec{E}}{\partial t}$$

Algumas Terminologias

- Matriz quadrada: # de linhas, # de colunas

$$A = \begin{bmatrix} 1 & 10 & 32 \\ 99 & 21 & 34 \\ 3 & 2 & 1 \end{bmatrix}$$

- Diagonal

  `diag(A)`

  Retorna um vetor linha

$$A = \begin{bmatrix} 1 & 10 & 32 \\ 99 & 21 & 34 \\ 3 & 2 & 1 \end{bmatrix}$$

- Sub-diagonais

  `diag(A,1)`

$$A = \begin{bmatrix} 1 & 10 & 32 \\ 99 & 21 & 34 \\ 3 & 2 & 1 \end{bmatrix}$$

- Matriz triangular superior

$$U = \begin{bmatrix} 1 & 10 & 32 \\ 0 & 21 & 34 \\ 0 & 0 & 1 \end{bmatrix}$$

```
U=triu(A,0)
```

- Matriz triangular inferior

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 99 & 21 & 0 \\ 3 & 2 & 1 \end{bmatrix}$$

```
L=tril(A,0)
```

- Matriz Identidade

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
I=eye(3)
```

Matriz Transposta

- Produz a troca dos elementos de um linha por uma coluna em uma matriz dada

$$A = \begin{bmatrix} 2 & 5 & 1 \\ 7 & 3 & 8 \\ 4 & 5 & 2 \\ 16 & 13 & 0 \end{bmatrix} \quad A^T = \begin{bmatrix} 2 & 7 & 4 & 16 \\ 5 & 3 & 5 & 13 \\ 1 & 8 & 21 & 0 \end{bmatrix}$$

- Comando MATLAB
  - Transposta de **A** é escrita por: **A'**

Produto Interno (Escalar- “dot”)

- Resulta em um escalar definido por:

$$A \bullet B = \sum_{i=1}^{N} a_i b_i$$

- Comando MATLAB: **`dot(A,B)`**

Multiplicação de Matrizes

- Cada elemento do produto é o resultado de um “dot produto’’ das linhas de uma matriz e as colunas da outra
- ***Condição***
- # colunas da matriz A = # linha da matriz B

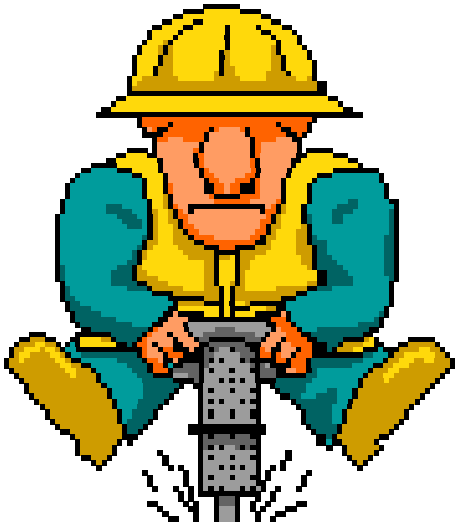
$$C = AB$$

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$

Matriz Inversa e “Rank”

- A Inversa de uma matriz quadrada A é A^{-1}
  `inv(A)`
- Onde: $A^{-1}A = I$
- A Inversa não existe p/ matrizes mal-condicionadas ou matrizes singulares
- Rank: # de eq. independentes representadas pelas linhas e colunas de uma matriz
- ***Se rank = # linhas, a matriz é não singular e a inversa existe***
  `rank(A)`

Agora é só Trabalhar...



Referências:

1. Etter, D.M. and D.C. Kuncicky, 1999, "*Introduction to MATLAB®*", E-Source, Prentice Hall, Upper Saddle River, New Jersey
2. Palm III, W.J., "*Introduction to MATLAB® for Engineers,*" B.E.S.T Series, McGraw-Hill, Boston.
3. Hahn, B.D., 1997, "*Essential MATLAB® for Scientists and Engineers,*" Arnold, London.
4. Harman, T.L., J. Dabney, and N. Richert, 2000, "*Advanced Engineering Mathematics with MATLAB® - Second Edition,*" Brooks/Cole – Thomson Learning, Australia
5. Mathews, J.H. and K.D. Fink, 1999, "*Numerical Methods Using MATLAB – Third Edition,*" Prentice Hall, Upper Saddle River, New Jersey
6. Middleton, G.W., , "*Data Analysis in the Earth Sciences using MATLAB®,*" Prentice Hall, Upper Saddle River, New Jersey
7. Nakamura, S., 2002, "*Numerical Analysis and Graphic Visualization with MATLAB – Second Edition,*" Prentice Hall PTR, Upper Saddle River, New Jersey
8. Nuruzzaman, M., 2003, "*Tutorials on Mathematics to Matlab*", www.1stbooks.com (do a search on *Author: Nuruzzaman* to link to the book information) -- Note: for sale online only. Relatively inexpensive collection of Matlab applications.
9. Van Loan, C.F., 2000, "*Introduction to Scientific Computing – A Matrix-Vector Approach Using MATLAB®,*" Prentice Hall, Upper Saddle River, New Jersey