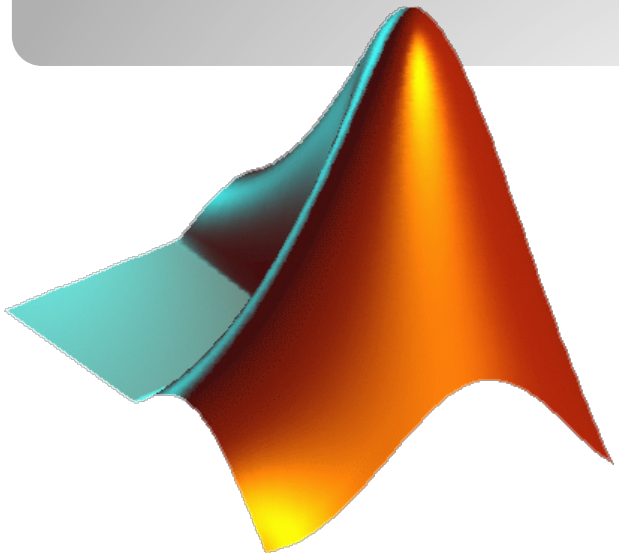
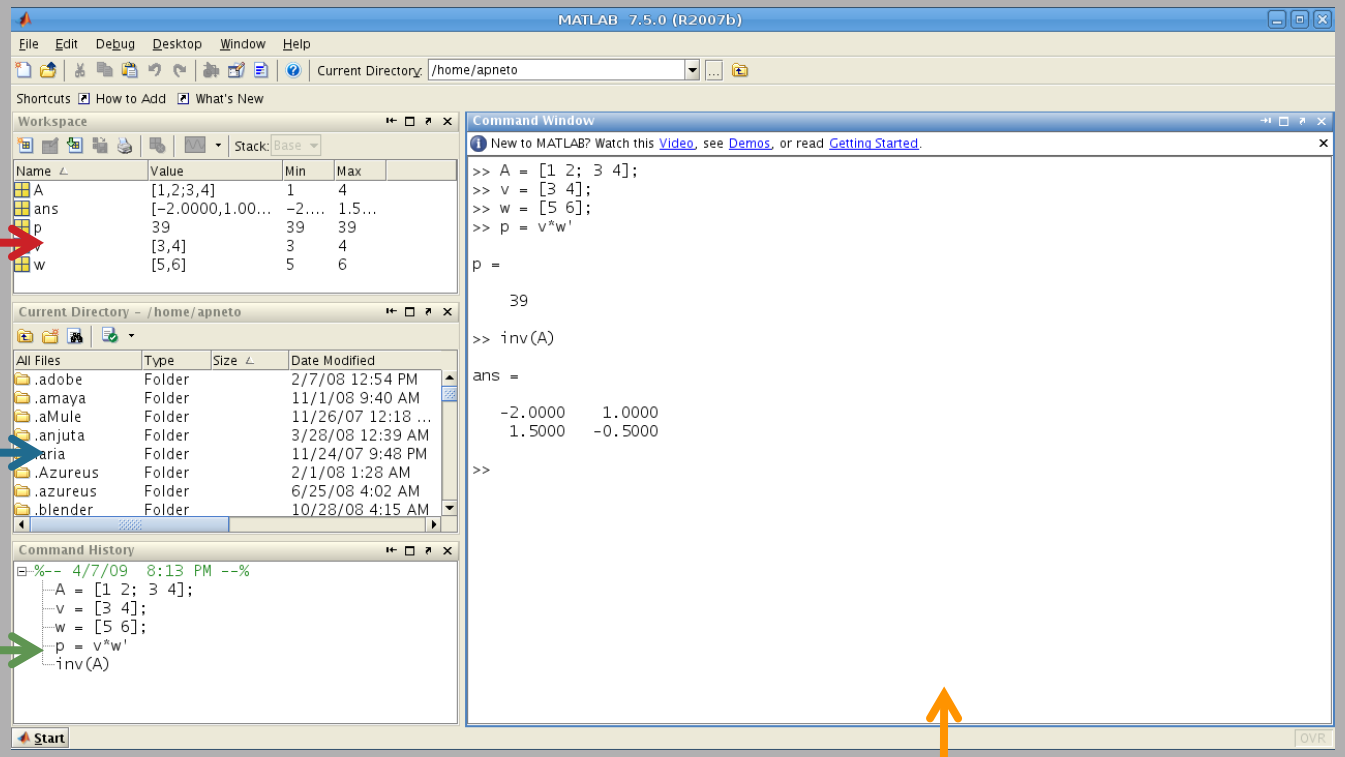
Introdução ao MATLAB



Afonso Paiva
ICMC-USP

O que é o MATLAB?

- *MATrix LABoratory* é um software para computação científica
- resolve numericamente problemas matemáticos de forma rápida e eficiente
- possui uma família de pacotes específicos (*toolboxes*):
  - otimização
  - redes neurais
  - processamento de imagens
  - simulação de sistemas, etc.



workspace

diretórios

histórico

janela de comandos

Anatomia da interface

Variáveis no MATLAB

- existe somente um tipo de variável:
  - matriz
- o tipo matriz pode ser expresso como:
  - escalar: matriz 1 x 1
  - vetor: matriz 1 x *n* ou *n* x 1
  - matriz propriamente: matriz *m* x *n*

Declaração de uma variável

- variáveis são alocadas na memória ao serem declaradas
- nomes de variáveis são sensíveis a letras maiúsculas e minúsculas
- vetores e matrizes devem ser declarados entre [ ]
- elementos de uma mesma linha numa matriz são separados por espaço(s) ou vírgula
- ponto-e-vírgula(;) indica o final de uma linha de uma matriz ou expressão

Exemplos

- Vetor linha:

```
>> A = [1 2 3 4];
```

- Vetor coluna:

```
>> B = [1; 2; 3; 4];   % ou
>> B = A';
```

Exemplos

- Matriz:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

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

- Matriz transposta:

```
>> B = A'
```

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

Operadores matemáticos

Símbolo	Operação
+	adição
-	subtração
*	multiplicação
/	divisão
^	potenciação

```
>> A=[1 2; 3 4];
>> B=[5 6; 7 8];
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

```
>> C0 = A+B
C0 =
   6   8
   10  12
```

```
>> C1 = A-B
C1=
   -4  -4
   -4  -4
```

```
>> C2 = A*B
C2=
   19  22
   43  50
```

```
>> C3 = A/B % = A*inv(B)
C3=
   3.0000  -2.0000
   2.0000  -1.0000
```

Exemplos

Operadores ponto-a-ponto

Símbolo	Operação
.*	multiplicação
./	divisão
.^	potenciação

```
>> A=[1 2; 3 4];
>> B=[5 6; 7 8];
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

```
>> C0 = A.*B
C0 =
   5   12
   21  32
```

```
>> C1 = A./B
C1=
   0.2000  0.3333
   0.4286  0.5000
```

```
>> C2 = A.^B
C2=
   1       64
   2187    65536
```

```
>> C3 = A.^3
C3=
   1   8
   27  64
```

Exemplos

Declaração de uma variável

- v=[inicio:incremento:fim]
- v=[inicio:fim] % incremento=1

ou

- v=inicio:incremento:fim
- v=inicio:fim

- Exemplo

```
>> A = 1:9

     A =
     1 2 3 4 5 6 7 8 9
```

```
>> v=[2:2:10]
      v =
      2 4 6 8 10

>> x=1:100;  % ou linspace(1,100)

>> M = [1:1:3; 4:1:6; 7:1:9]
      M =
         1 2 3
         4 5 6
         7 8 9
```

Exemplos

Manipulação de matrizes

- acessando um elemento de uma matriz

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 \end{bmatrix}$$

```
>> A (3)

ans=
     5
```

- referência deve ser sempre (linha, coluna)

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

```
>> B (2,3)

ans=
     6
```

Manipulação de matrizes

- é possível incluir matrizes em matrizes

```
>>A = [1 2 3; 4 5 6; 7 8 9];   % A é uma matriz 3X3
>> a = [10 20 30];
>> A = [A;a]    % A é uma matriz 4X3
```

```
A =
     1     2     3
     4     5     6
     7     8     9
    10    20    30
```

Manipulação de matrizes

- podemos extrair uma linha da matriz

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

```
>> linha = A(2,:)

  linha =

        4   5   6
```

Manipulação de matrizes

- e também acessar uma coluna da matriz

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

```
>> coluna = A(:,1)

   coluna =
            1
            4
            7
```

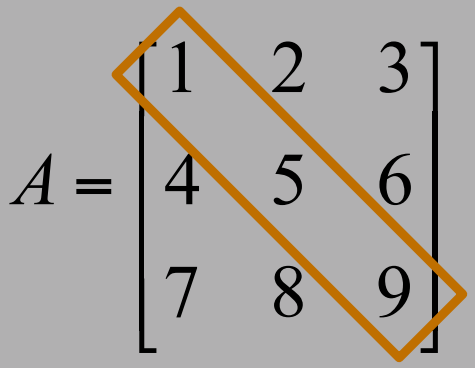
Manipulação de matrizes

- podemos extrair submatrizes de uma matriz

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

```
>> B = A(1:2,2:3) % ou B = A([1 2],[2 3])

  B =

      2   3
      5   6
```

- podemos acessar diretamente elementos da diagonal




```
>>d =diag(A)

d =
     1
     5
     9
```

Manipulação de matrizes

```
>> A = [1 2 3;4 5 6;7 8 9]
>> L0 = tril(A)
>> L1 = tril(A, 1)
>> L2 = tril(A,-1)
```

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

$$L0 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 5 & 0 \\ 7 & 8 & 9 \end{bmatrix} \qquad L1 = \begin{bmatrix} 1 & 2 & 0 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad L2 = \begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix}$$

Matrizes triangulares inferior

```
>> A = [1 2 3;4 5 6;7 8 9]
>> U0 = triu(A)
>> U1 = triu(A, 1)
>> U2 = triu(A,-1)
```

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

$$U0 = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix} \qquad U1 = \begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix} \qquad U2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 8 & 9 \end{bmatrix}$$

Matrizes triangulares superior

```
>> A = [1 2 3 4; 5 6 7 8]
   A =
        1  2  3  4
        5  6  7  8
>> s = size(A)
   s =
        2  4

>> l = size(A,1)        % numero de linhas
   l =
        2
>> c = size(A,2)        % numero de colunas
   c =
        4
```

Dimensão de vetores e matrizes

```
>> length(A) % retorna o numero de linhas ou de colunas
  ans =
        4

>> v= 1:7;
>> size(v)     % número de colunas
  ans =
        1  7

>> length(v) % retorna o comprimento do vetor
  ans =
        7
```

Dimensão de vetores e matrizes

Funções matriciais

Comando	Descrição
det(A)	calcula o determinante da matriz
[V,D] = eig(A)	determina os autovetores e autovalores de A
inv(A)	calcula a inversa da matriz
rank(A)	determina o posto linha ou coluna de A
max(A)	retorna um vetor com o máximo de cada coluna A
min(A)	retorna um vetor com o mínimo de cada coluna A
norm(A,1)	calcula a norma coluna
norm(A, 'fro')	calcula a norma de Frobenius
norm(A,inf)	calcula a norma linha

```
>> A = [ 1 7 3; -6 2 1; 9 2 -2];

>> det(A)
   ans =
        -117

>> I = inv(A)
   I =
     0.0513  -0.1709  -0.0085
     0.0256   0.2479   0.1624
     0.2564  -0.5214  -0.3761

>> max(A)
   ans =
        9  7  3

>> norm(A,1)
   ans =
        16

>> norm(A,inf)
   ans =
        13

>> norm(A,' fro ')
   ans =
        13.7477
```

Exemplos

Matrizes especiais

Comando	Descrição
A = rand(m,n)	gera matriz com elementos aleatórios
A = eye(n)	gera matriz identidade
A= ones(m,n)	gera matriz com todos elementos iguais a 1
A= zeros(m,n)	gera matriz com todos elementos iguais a 0

Função	Descrição
sin(x)	seno
cos(x)	cosseno
tan(x)	tangente
asin(x)	arco-seno
acos(x)	arco-cosseno
atan(x)	arco-tangente
exp(x)	exponencial
log(x)	logaritmo natural
log10(x)	logaritmo na base 10

Funções matemáticas elementares

Funções matemáticas elementares

Função	Descrição
abs(x)	valor absoluto
ceil(x)	arredondamento na direção de mais infinito
floor(x)	arredondamento na direção de menos infinito
round(x)	arredondamento para o inteiro mais próximo
sign(x)	função sinal
sqrt(x)	raiz quadrada
gcd(x,y)	máximo divisor comum dos inteiros *x* e *y*
lcm(x,y)	mínimo múltiplo comum dos inteiros *x* e *y*
rem(x,y)	resto da divisão de *x* por *y*

Variáveis especiais

Variável	Valor
ans	variável padrão usada para resultados
pi	3.14159 26535...
eps	precisão de máquina
inf	infinito
NaN ou nan	*not a number*
realmin	menor número de ponto flutuante
realmax	maior número de ponto flutuante
i, j	unidade imaginária ($i = j = \sqrt{-1}$)

Workspace do MATLAB

- Os dados e variáveis criados na janela de comandos são armazenados no que é chamado de *workspace.*

Comandos	Descrição
who ou whos	mostra os nomes das variáveis que estão no workspace
clear	apaga as variáveis do workspace
clc	limpa a tela de comando
help *comando*	fornece uma ajuda rápida sobre o comando

Formatos de números

Comando	Exemplo	Observações
format short	50.833	5 dígitos
format long	50.83333333333334	16 dígitos
format short e	5.0833e+01	5 dígitos+expoente
format long e	5.083333333333334e+01	16 dígitos+expoente
format short g	50.833	melhor entre short - short e
format long g	50.83333333333333	melhor entre long - long e
format hex	40496aaaaaaaaaab	hexadecimal
format bank	50.83	2 dígitos decimais
format rat	305/6	aproximação racional

Para mudar o padrão de formato de números no MATLAB, basta ir em *File > Preferences*.

Números complexos

- Criando um número complexo

```
>> z = 3+2*i
   z =
        3.0000 + 2.0000i
```

- Parte real de *z*

```
>> real(z)
   ans =
          3
```

- Parte imaginária de *z*

```
>> imag(z)
   ans =
          2
```

Números complexos

- Módulo de *z*

```
>> abs(z)
   ans =
         3.6056
```

- Argumento de *z*

```
>> angle(z)
   ans =
         0.5880
```

- Complexo conjugado

```
>> conj(z)
   ans =
         3.0000 - 2.0000i
```

Sistemas lineares

- Dado o sistema linear

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 366 \\ 804 \\ 351 \end{bmatrix}$$

$$A \cdot x = b$$

Como encontrar a solução *x* no MATLAB?

Sistemas lineares

- Através do cálculo explícito da inversa de A:

$$x = A^{-1} \cdot b$$

```
>> A = [ 1 2 3; 4 5 6; 7 8 0];
>> det(A)  %primeiro vamos ver se o sistema tem solução única
   ans =
         27
>> b = [366; 804;351];
>> x = inv(A)*b
   x =
         25.0000
         22.0000
         99.0000
```

Sistemas lineares

- Outra maneira é utilizar a *decomposição LU*, representada no MATLAB pelo operador de divisão à esquerda (\):

```
>> x = A\b
   x =
         25.0000
         22.0000
         99.0000
```

Calculando o tempo: tic e toc

```
>> A = rand(100);
>> b = rand(100,1);
>> tic, x = inv(A)*b; t1 = toc;
>> tic, x = A\b; t2 = toc;
```

Rotinas de tempo computacional

Polinômios

- No MATLAB , um polinômio é representado por um vetor linha contendo seus coeficientes em ordem decrescente.

- Exemplo: $x^4 - 12x^3 + 25x + 116$

```
>> p = [1 -12 0 25 116];
```

Polinômios

- Cálculo das raízes de p:

```
>> r = roots(p)
   r =
      11.7473
       2.7028
      -1.2251 + 1.4672i
      -1.2251 -  1.4672i
```

- Dadas as raízes, podemos construir o polinômio associado:

```
>> r=[-2;2];
>> pp = poly(r)
   pp =
         1  0  -4
```

Polinômios

- Podemos derivar polinômio:

```
>> p=[1 -7 3 1];
>> pd = polyder(p)
   pd =
        3  -14   3
```

- Multiplicando p e pd:

```
>> conv(p,pd)   % na divisão usa-se deconv(p,pd)
   ans =
        3  -35  110  -60  -5  -3
```

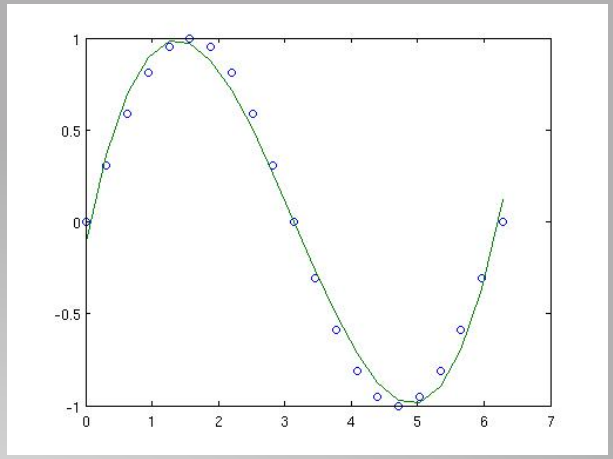
Polinômios

- Ajuste de curvas:

```
>> x=0:.1*pi:2*pi;
>> x = x';
>> y = sin(x);
>> p = polyfit(x,y,4)   % aproxima o seno por um polinômio de grau 4

   p =
       -0.0000  0.0886  -0.8347  1.7861  -0.1192
```

- Avaliando o polinômio p:

```
>> f = polyval(p,x);
>> plot(x, y, 'o', x, f, '-');
```



Operadores relacionais

Símbolo	Operador
==	igual
~=	diferente
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

Operadores lógicos

Símbolo	Operador
&&	E
||	OU
&	E (escalar)
|	OU (escalar)
~	Não
xor	OU exclusivo

```
>> 2 + 2 == 4
   ans =
          1 % verdadeiro

>> 10 > 100
   ans =
          0  % falso

>> A = [1 2; 3 4];
>> B = 2*ones(2);
>> A == B
   ans =
          0  1
          0  0
```

```
>> C = [1 2 3; 4 5 6]
   C =
       1  2  3
       4  5  6

>> C >= 4
   ans =
          0  0  0
          1  1  1
```

Exemplos

```
>>x = eye(2)
     x =
          1  0
          0  1

>>y = [1 1; 0 0]
     y =
          1  1
          0  0

>> x & y
   ans =
          1  0
          0  0
```

```
>> x | y
   ans =
          1  1
          0  1

>> xor(x,y)
    ans =
          0  1
          0  1
```

Exemplos

Importando e Exportando Dados

- save: salva dados em arquivo (.mat)

```
>> a=1;
>> A= ones(5);
>> save meus_dados A a
```

- load: carrega dados de um arquivo

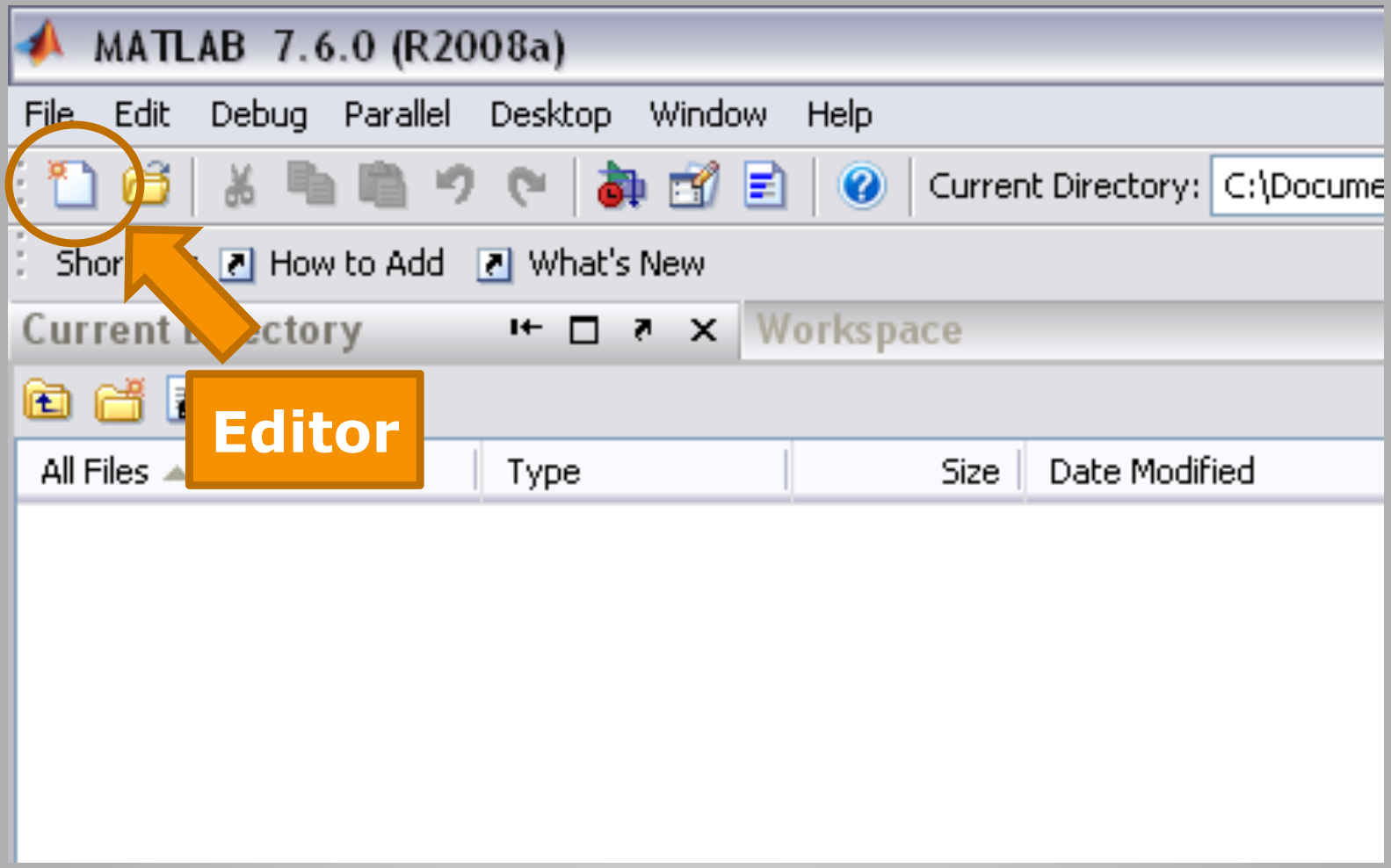
```
>> load meus_dados
```

- programas muito mais simples: escrita mais rápida e com menos erros
- versatilidade, mais fácil de adaptar a diferentes tipos de dados
- agiliza os comandos mais digitados
- é uma linguagem interpretada
- pode ser criada utilizando qualquer editor de texto
- possui interface com C/C++

Programação em MATLAB

Arquivos .m

- podemos criar novas funções ou scripts
- MATLAB possui um editor próprio e um *debugger*
- comentários começam por %
- ao se criar uma função ou script ela dever ser definida no *path*


MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Docume
How to Add
What's New
Workspace
Editor
All Files
Type
Size
Date Modified

Arquivos .m

Arquivos .m

- scripts : executam os argumentos diretamente, automatizando uma série de comandos
- função : argumentos podem ser passados para a função, havendo uma manipulação de variáveis

function [*res1*,*res2*,...] = nome_da_função (*arg1*,*arg2*,...)
% *comentário para help*
lista de procedimentos da função

Arquivos .m

ATENÇÃO!!!

- Um script pode chamar uma função.
- Uma função pode chamar outra função.
- Para isso é necessário que os arquivos .m estejam no **mesmo diretório**.
- **Não** use “espaços” no nome da função, “_” é uma boa opção.
- A função será salva em um arquivo e o nome do arquivo **deve ter o mesmo nome** dado à função 'nome_func.m’.
- O script pode ter qualquer nome.
- **Evite usar** nomes de funções/scripts já existentes no MATLAB.

Controladores de fluxo

- if : cria caminhos alternativo no programa

```
if ( condição1 )
    instruções1
elseif ( condição2 )
    instruções2
else
    instruções3
end
```

Controladores de fluxo

- Exemplo com if:

```
x=rand(1);
y=rand(1);
if (x < y)
    disp('y eh maior do que x')
else
    disp('x eh maior do que y')
end
```

Controladores de fluxo

- Mais um exemplo com if:

```
x=rand(1);
if ( ( x>=1 ) && ( x<=3 ) )
    disp('x estah entre 1 e 3')
end
```

Controladores de fluxo

- for : permite que um comando ou um grupo de comandos se repitam

```
for variável = expressão
    instruções
end
```

Controladores de fluxo

- Exemplo com for:

```
n=3;
A = zeros(n);
for i = 1:n
    A(i,i) = 2*i;
end
```

Controladores de fluxo

- Exemplo usando if e for

```
nrows = 10; % Alocando a matriz
ncols = 10;
myData = ones(nrows, ncols);

% Preencher a matriz
for r = 1:nrows
   for c = 1:ncols
      if r == c
          myData(r,c) = 2;
      elseif abs(r - c) == 1
          myData(r,c) = -1;
      else
           myData(r,c) = 0;
      end
    end
end

myData % Ver a matriz
```

Controladores de fluxo

- while : permite que um ou mais comandos sejam repetidos enquanto a expressão de controle for verdadeira

```
while ( condição )
    instruções
end
```

Controladores de fluxo

- Exemplo com while:

```
i=0;
while ( sqrt(i) < 5 )
    i = i+1
end
```

Controladores de fluxo

- switch : Permite ramificar alguns casos especiais de modo mais claro do que o *if*

```
switch ( expressão do switch )
    case expressão caso_1
          instruções
    case expressão caso_2
          instruções
    case expressão caso_n
          instruções
    otherwise % opcional
          instruções
end
```

Funções auxiliares no controle de fluxo

- input –recebe dados através do teclado, que podem ser ou não armazenados em uma variável
- break – encerra um laço mais interno controlado pelo comando for
- pause – pára a execução do programa até que uma nova tecla seja pressionada

Controladores de fluxo

- Exemplo com switch

```
meu_numero = input('Enter a number:');

switch meu_numero
  case -1
      disp('negative one');
  case 0
      disp('zero');
  case 1
      disp('positive one');
  otherwise
      disp('other value');
end
```

Scripts

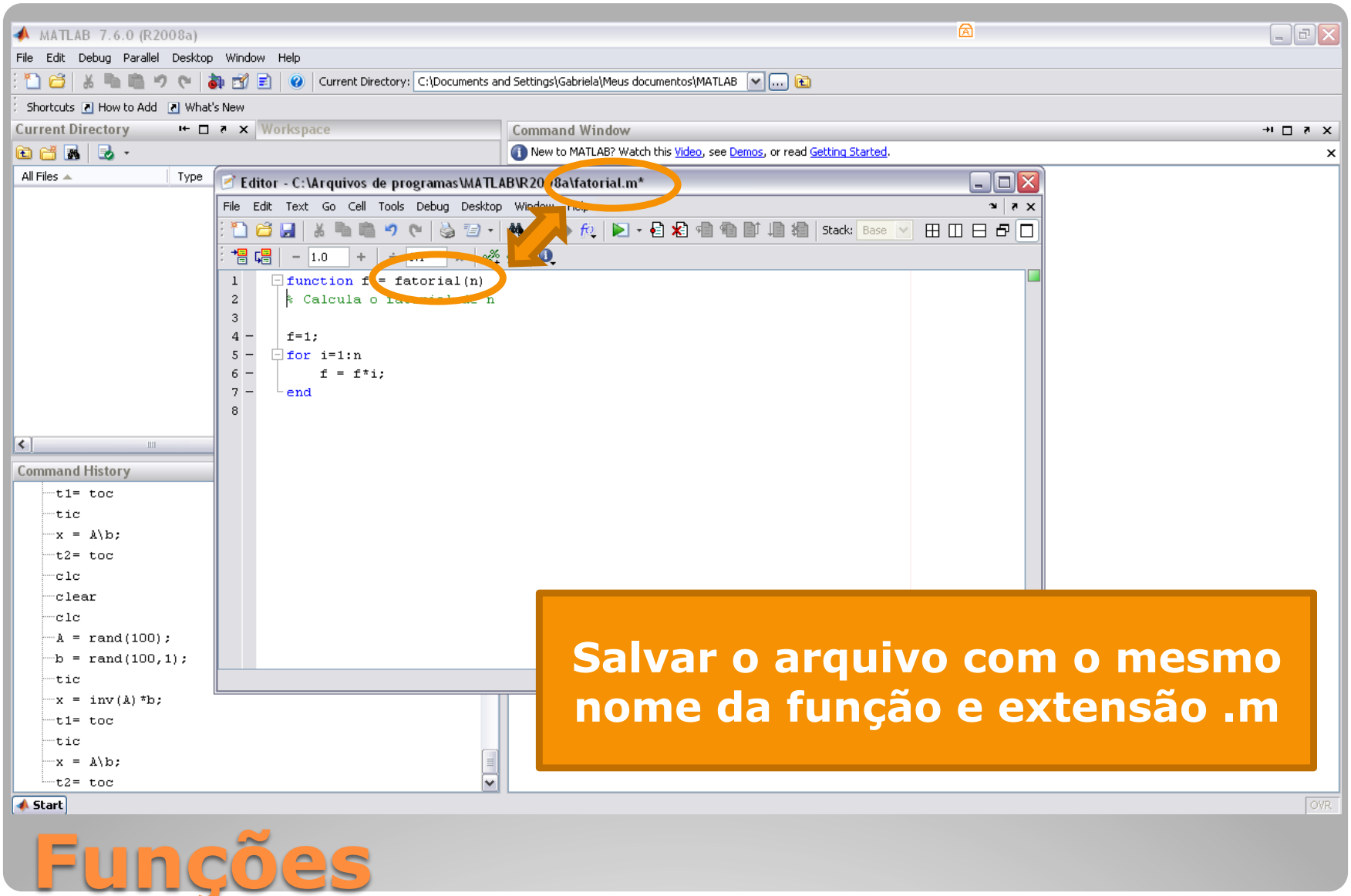
- Exemplo de hipotenusa.m:

```
% Calcula a hipotenusa de um triangulo retângulo
  clc
  c1=input('Cateto 1 = ');
  c2=input('Cateto 2 = ');
  hipotenusa = sqrt( c1^2 + c2^2)
```

- Exemplo de fatorial.m:

```
function f = fatorial(n)
% Calcula o fatorial de n

f=1;
 for i=1:n
    f = f*i;
 end
```

Funções


MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Documents and Settings\Gabriela\Meus documentos\MATLAB
Shortcuts How to Add What's New
Current Directory
Workspace
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
All Files
Type
Editor - C:\Arquivos de programas\MATLAB\R2008a\fatorial.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1 function f = fatorial(n)
2 % Calcula o fatorial de n
3
4 f=1;
5 for i=1:n
6 f = f*i;
7 end
8
Command History
t1= toc
tic
x = A\b;
t2= toc
clc
clear
clc
A = rand(100);
b = rand(100,1);
tic
x = inv(A)*b;
t1= toc
tic
x = A\b;
t2= toc
Start
OVR
Salvar o arquivo com o mesmo nome da função e extensão .m
Funções

Funções

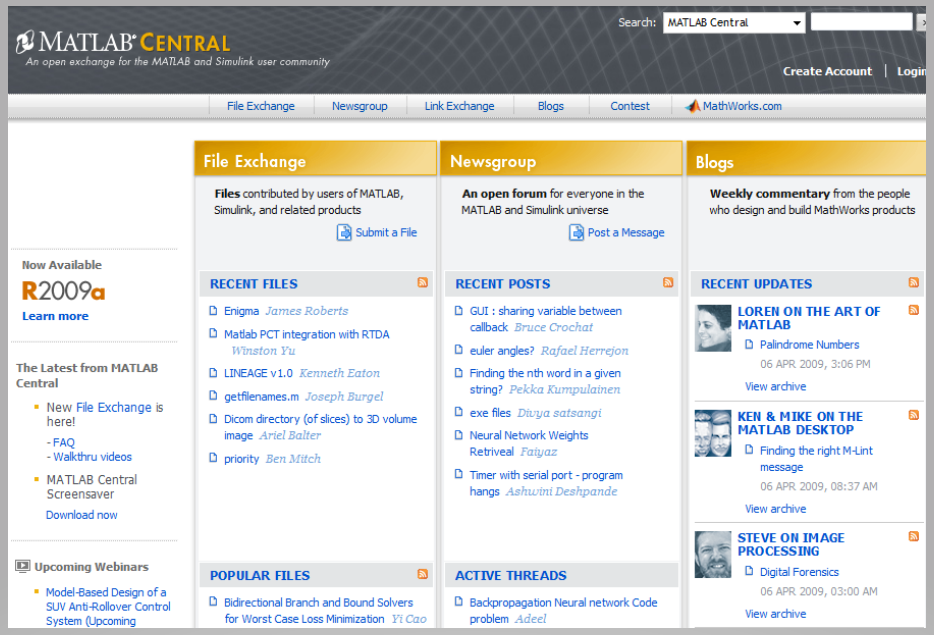
- Defina duas funções em um arquivo chamado stat2.m, onde a primeria função chama a segunda.

```
function [m,s] = stat2(x)
n = length(x);
m = media(x,n);
s = sqrt(sum((x-m).^2/n));
end

function m = media(x,n)
m = sum(x)/n;
end
```

- Função media é uma função local.

```
>> values = [12.7, 45.4, 98.9, 26.6, 53.1];
>> [media,desvio_padrao] = stat2(values)

media = 47.3400
desvio_padrao = 29.4124
```

- MATLAB Central
  http://www.mathworks.com/matlabcentral



Repositório de arquivos .m

Gráficos no MATLAB

- existem muitas funções para gerar gráficos 2D e 3D
- os gráficos podem ser armazenados em arquivos, coloridos ou em preto e branco

Gráficos 2D

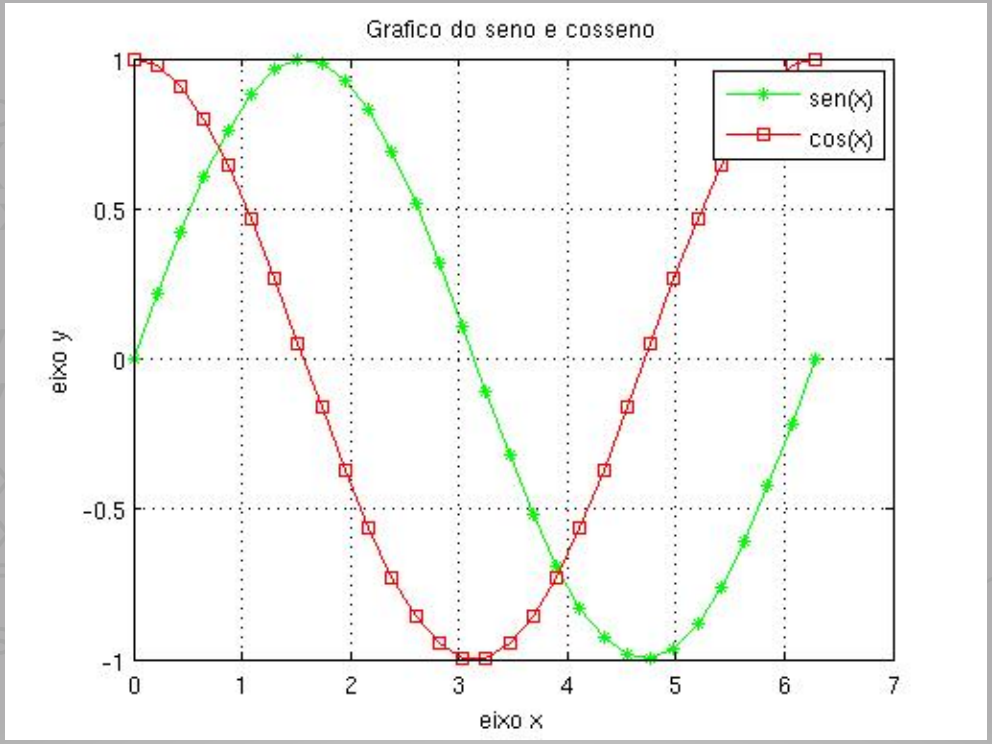
- Pode-se desenhar gráficos simples como *y = f(x)*

```
>> x=linspace(0,2*pi,30);
>> y1 = cos(x);
>> y2 = sin(x);
>> hold on
>> plot(x,y1, 'r-s');
>> plot(x,y2, 'g-*');
>> grid
>> xlabel('eixo x');  % legenda no eixo horizontal
>> ylabel('eixo y');  % legenda no eixo vertical
>> title('Grafico do seno e do cosseno');  % título do gráfico
>> legend ('sen(x) ', 'cos(x) '); % legenda
>> hold off
```

Gráficos 2D

- Pode-se desenhar gráficos simples como *y = f(x)*

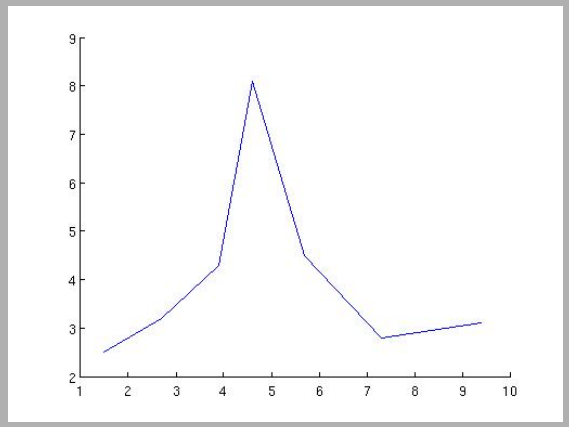
```
>> x=linspac
>> y1 = cos(
>> y2 = sin(
>> hold on
>> plot(x,y1,
>> plot(x,y2,
>> grid
>> xlabel('ei
>> ylabel('ei
>> title('Graf                                    ráfico
>> legend ('s
>> hold off
```



Gráficos 2D

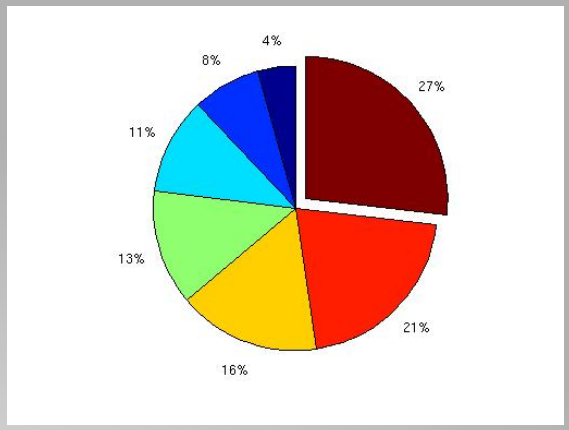
- Dados os vetores x e y, o gráfico pode ser construído ligando-se os pontos (x(i),y(i)):

```
>> x = [1.5 2.7 3.9 4.6 5.7 7.3 9.4];
>> y = [2.5 3.2 4.3 8.1 4.5 2.8 3.1];
>> plot(x,y);
>> box off  % retira a caixa do gráfico
```



- Gráfico de torta

```
>> pie(x,x==max(x));
```



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

Características dos gráficos

Símbolo	Marcador
.	ponto
o	círculo
x	x
+	+
*	estrela
s	quadrado
d	losango
^	triângulo
p	pentagrama
h	hexagrama

Características dos gráficos

Características dos gráficos

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

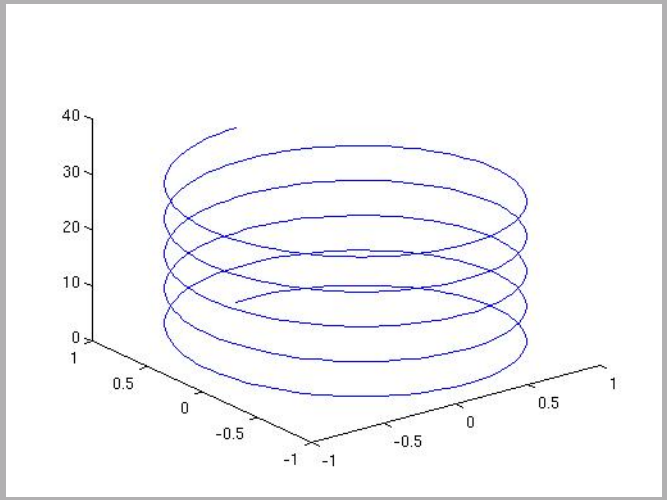
Gráficos 3D

Comando	Descrição
plot3	curvas 3d
surf, surfc, surfl	superfícies 3d
mesh, meshc, meshz	linhas em perspectiva 3d
contour	curvas de níveis

Gráficos 3D

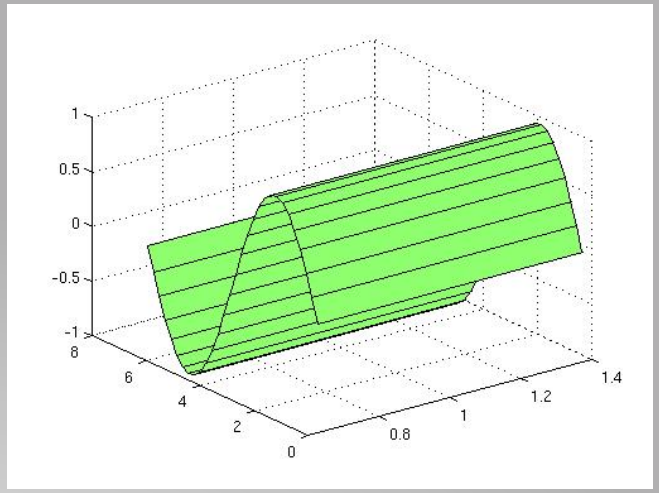
- Curvas no espaço

```
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t)
```



- Faixas no espaço

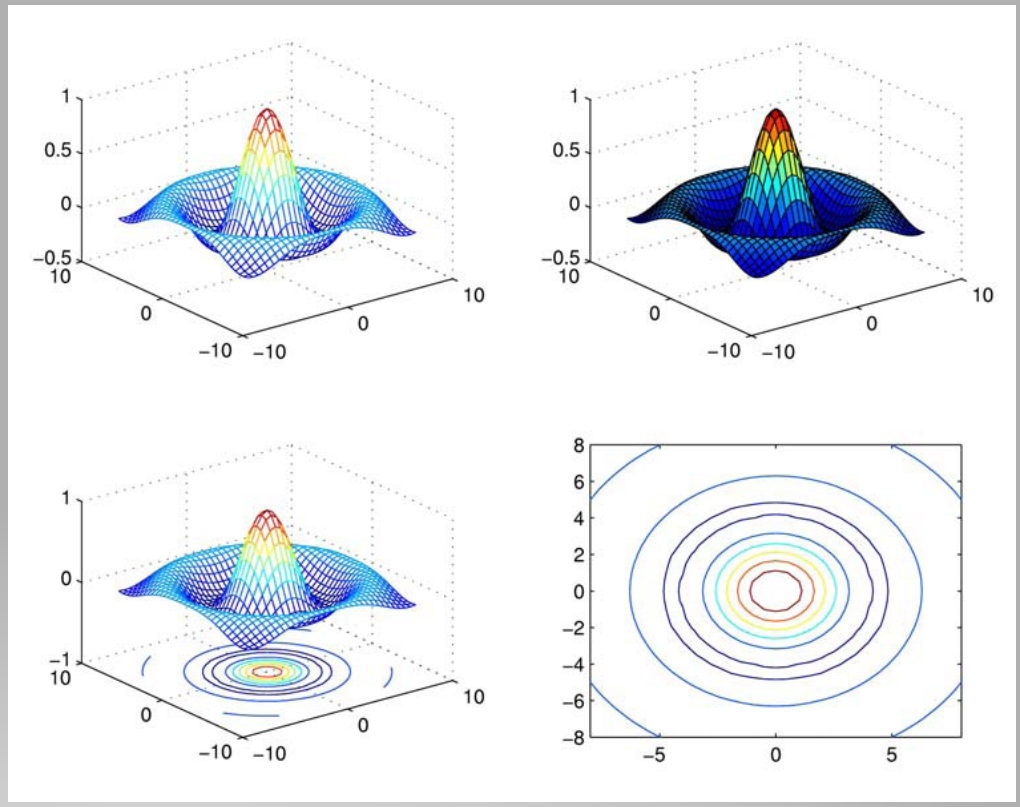
```
>> x=linspace(0,2*pi,30);
>> ribbon(x,sin(x))
```



Gráficos 3D

- Superfícies

```
>> [X,Y] =meshgrid(-8:0.5:8,-8:0.5:8);
>> r =sqrt(X.^2+Y.^2)+eps;
>> Z = sin(r)./r;
>> subplot(221)
>> mesh(X,Y,Z);
>> subplot(222)
>> surf(X,Y,Z);
>> subplot(223)
>> hold on
>> mesh(X,Y,Z);
>> meshc(X,Y,Z)
>> hold off
>> subplot(224)
>> contour(X,Y,Z)
```



Erros de arredondamento

- Considere a função $f(x) = (x-1)^7$

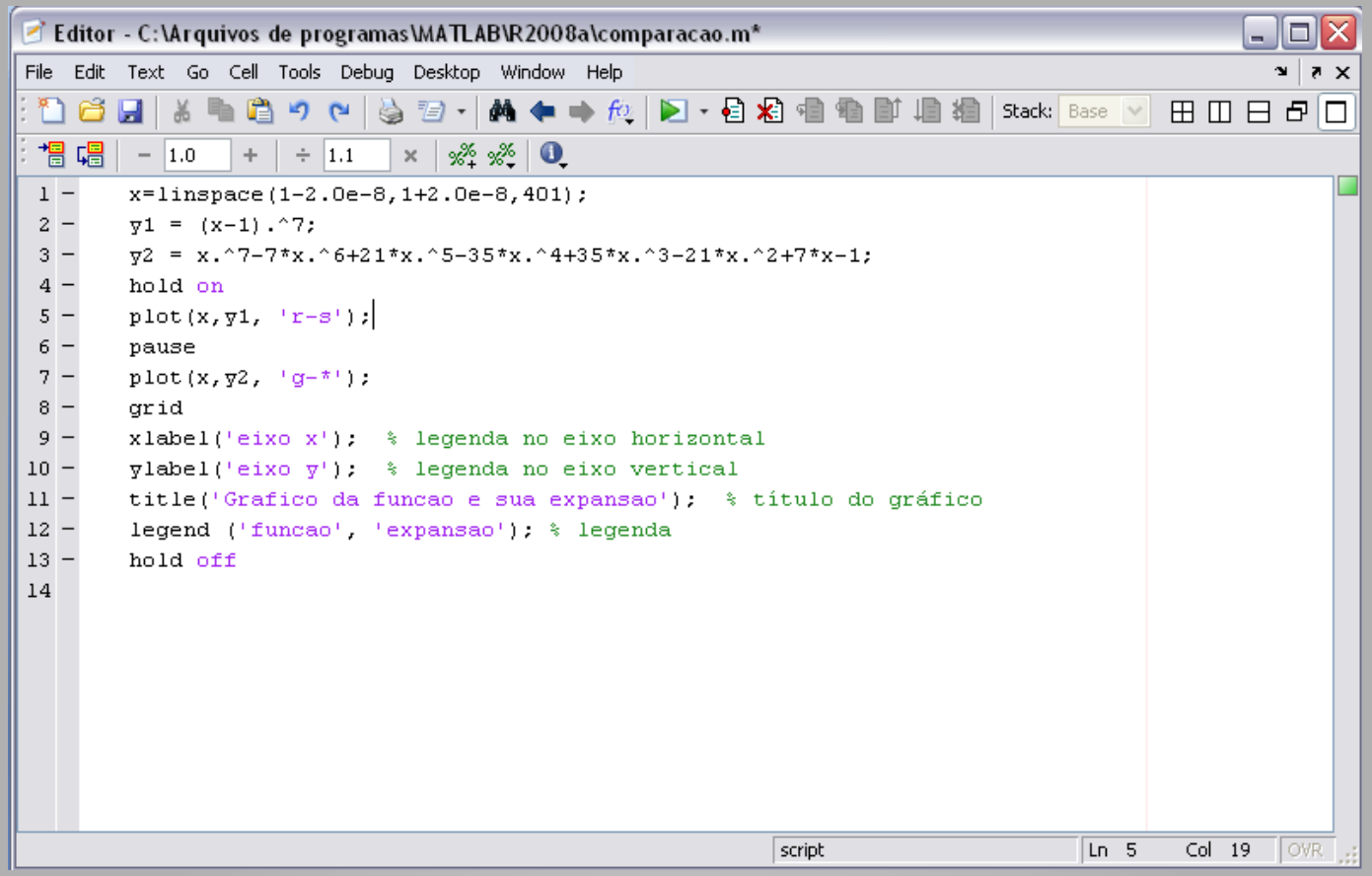
  e sua expansão:

$$f(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$

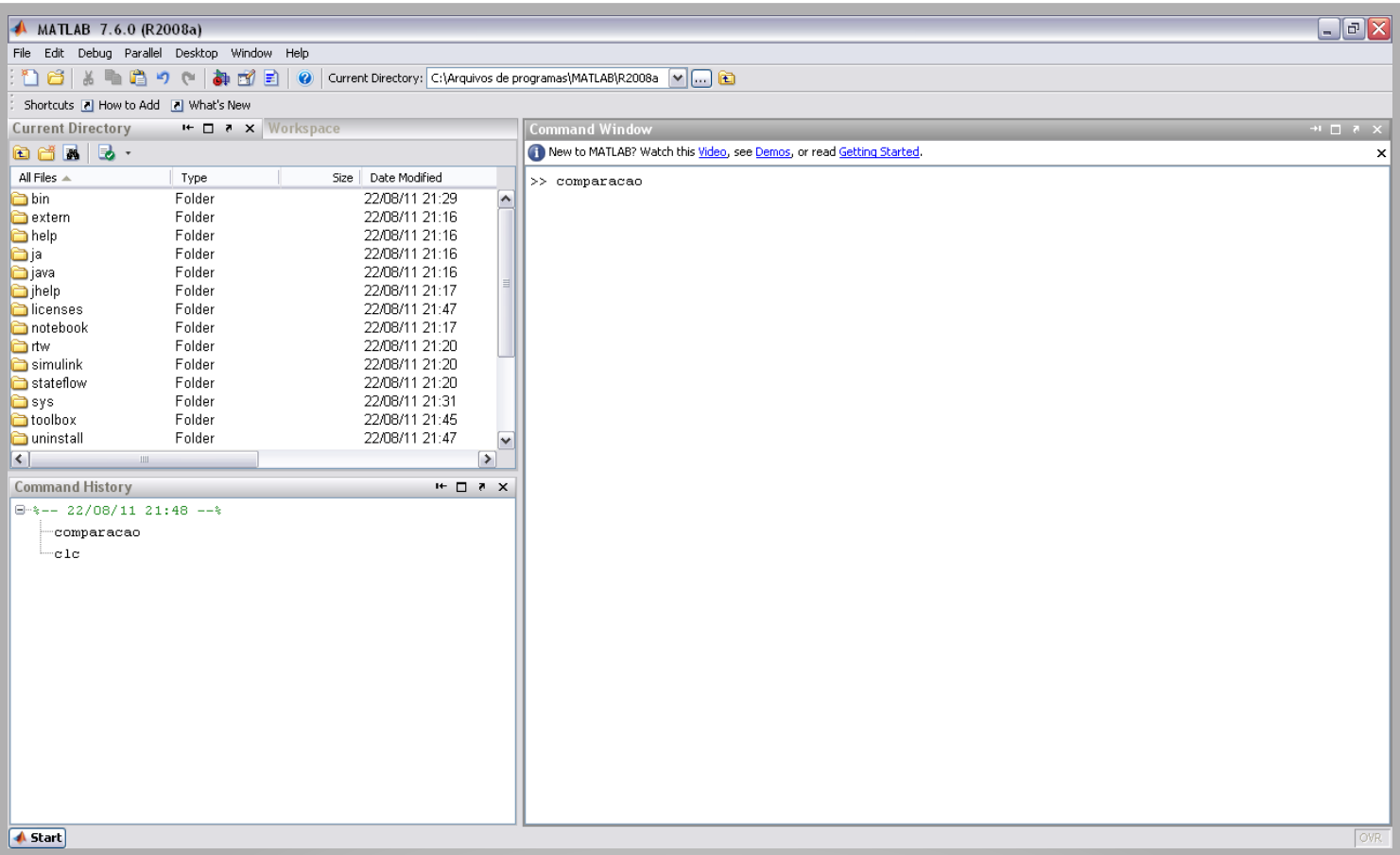
- Avalie estas funções em *401* pontos equidistantes no intervalo:

$$I = [1 - 2.0 \times 10^{-8}, 1 - 2.0 \times 10^{-8}]$$

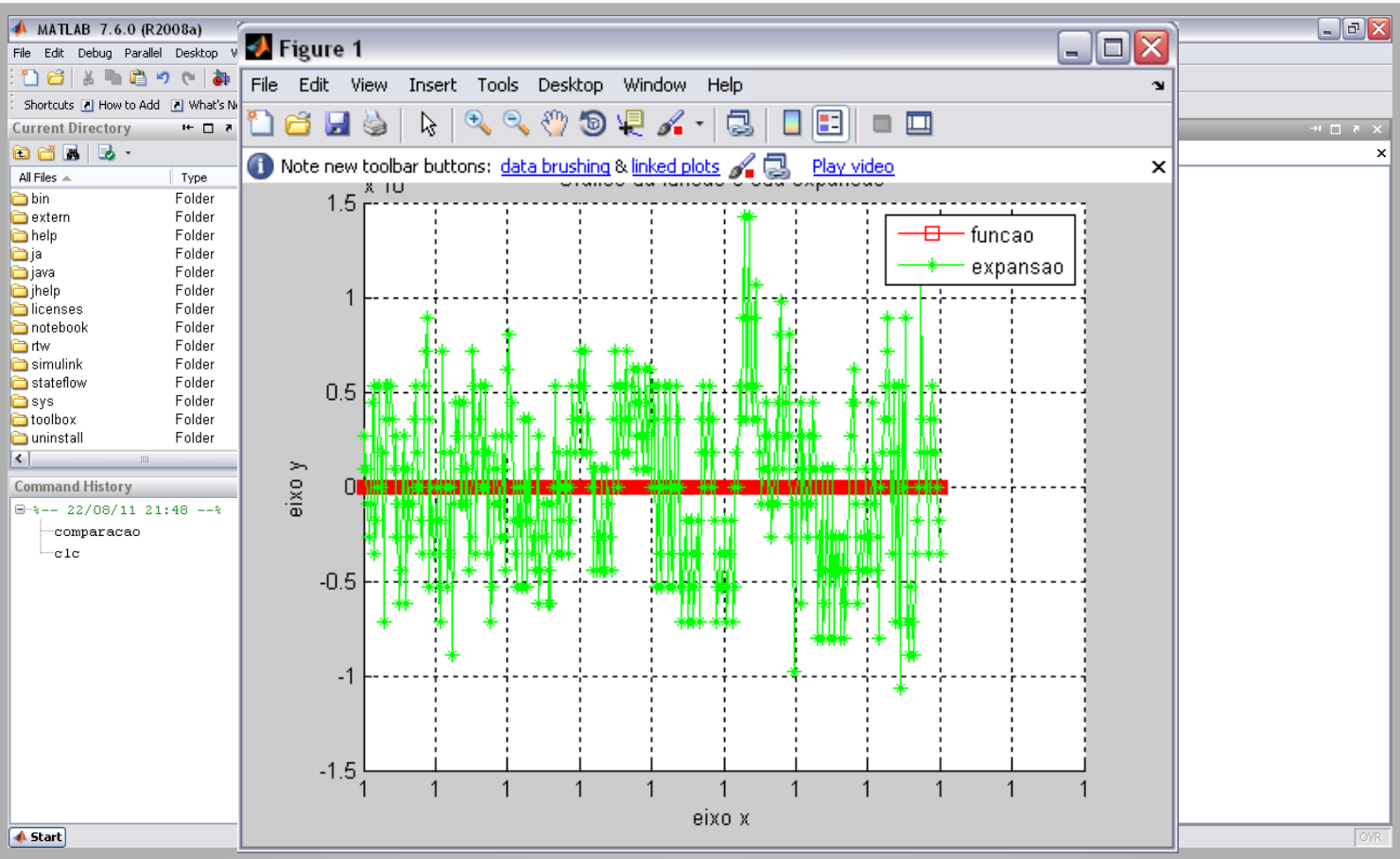
- O que ocorre ao plotar os grácos?



Erros de arredondamento



Erros de arredondamento



Erros de arredondamento

Help MATLAB on-line

- http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ref.shtml
- http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml