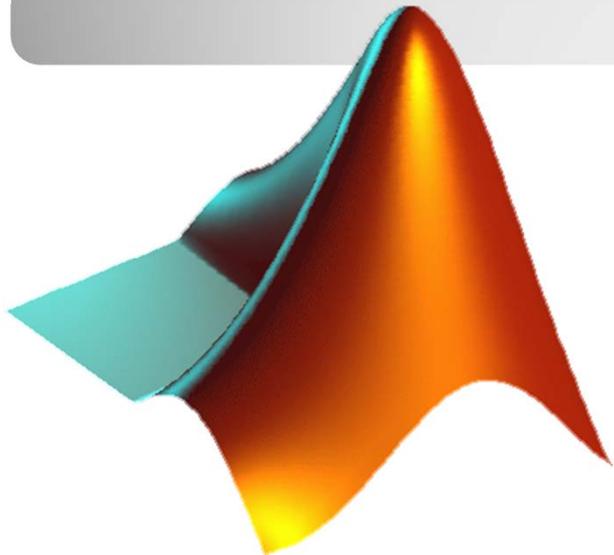


Introdução ao MATLAB



Afonso Paiva
ICMC-USP



- *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.

O que é o MATLAB?

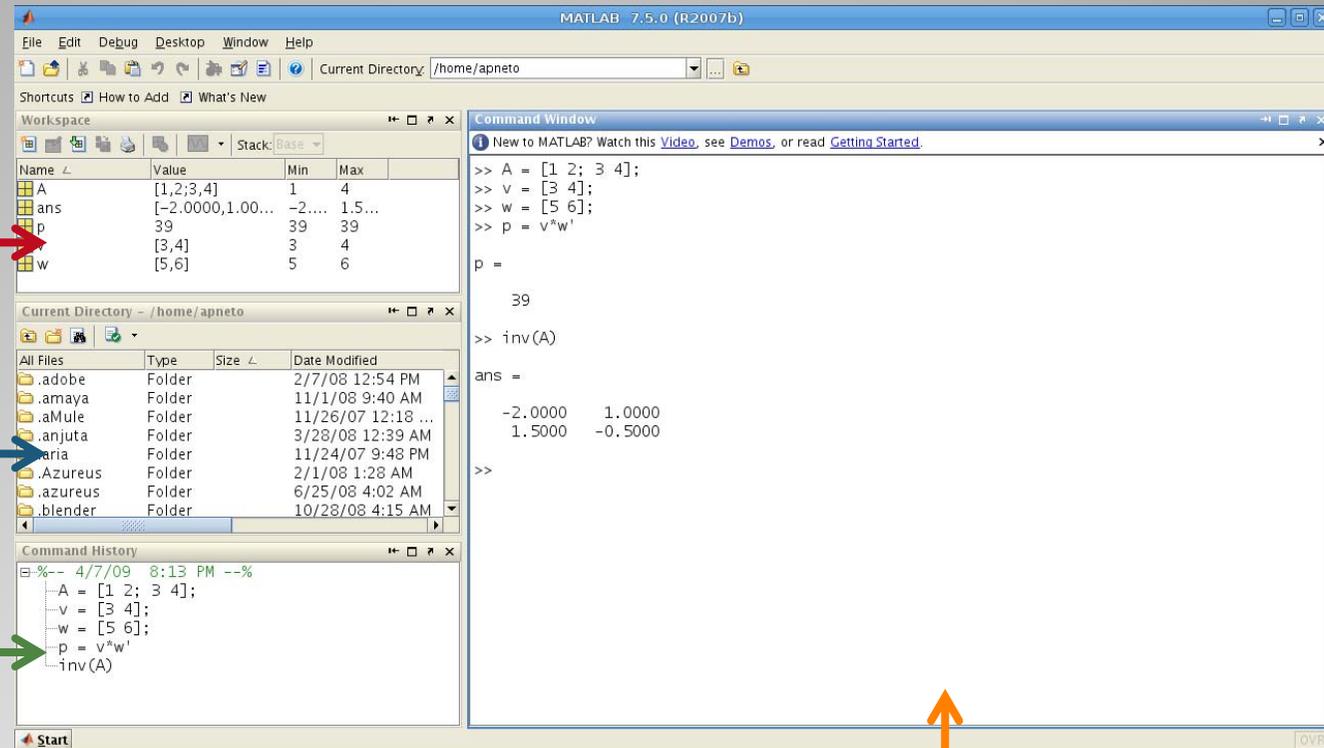
workspace



diretórios



histórico



janela de comandos

Anatomia da interface

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

Variáveis no MATLAB

- 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

Declaração de uma variável

- 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'
```

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

Exemplos

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

Operadores matemáticos

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

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

```
>> C0 = A+B
```

```
C0 =  
 6  8  
10 12
```

```
>> C2 = A*B
```

```
C2 =  
19 22  
43 50
```

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

```
C1 =  
-4 -4  
-4 -4
```

```
>> C3 = A/B % = A*inv(B)
```

```
C3 =  
3.0000 -2.0000  
2.0000 -1.0000
```

Exemplos

Símbolo	Operação
.*	multiplicação
./	divisão
.^	potenciação

Operadores ponto-a-ponto

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

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

```
>> 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

- `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
```

Declaração de uma variável

```
>> 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

- acessando um elemento de uma matriz

$$A = [1 \quad 3 \quad 5 \quad 7]$$

>> 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
```

Manipulação de matrizes

- podemos acessar diretamente elementos da diagonal

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

```
>>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)
```

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

$$L0 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 5 & 0 \\ 7 & 8 & 9 \end{bmatrix}$$

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

$$L2 = \begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix}$$

Matrices triangulares inferior

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

```
>> U0 = triu(A)
```

```
>> U1 = triu(A, 1)
```

```
>> U2 = triu(A,-1)
```

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

$$U0 = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 0 & 0 & 9 \end{bmatrix}$$

$$U1 = \begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix}$$

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

Matrices 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

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

Funções matriciais

```
>> 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

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

Matrizes especiais

Função	Descrição
$\sin(x)$	seno
$\cos(x)$	cosseno
$\tan(x)$	tangente
$\text{asin}(x)$	arco-seno
$\text{acos}(x)$	arco-cosseno
$\text{atan}(x)$	arco-tangente
$\exp(x)$	exponencial
$\log(x)$	logaritmo natural
$\log_{10}(x)$	logaritmo na base 10

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

Funções matemáticas elementares

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

Variáveis especiais

- 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 <i>comando</i>	fornece uma ajuda rápida sobre o comando

***Workspace* do MATLAB**

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*.

Formatos de números

- 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
```

- Complejo conjugado

```
>> conj(z)
```

```
ans =
```

```
3.0000 - 2.0000i
```

Números complejos

- Dado o sistema linear

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

$$A \cdot x = b$$

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 (\backslash):

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

```
x =
```

```
25.0000
```

```
22.0000
```

```
99.0000
```

Sistemas lineares

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

- 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
```

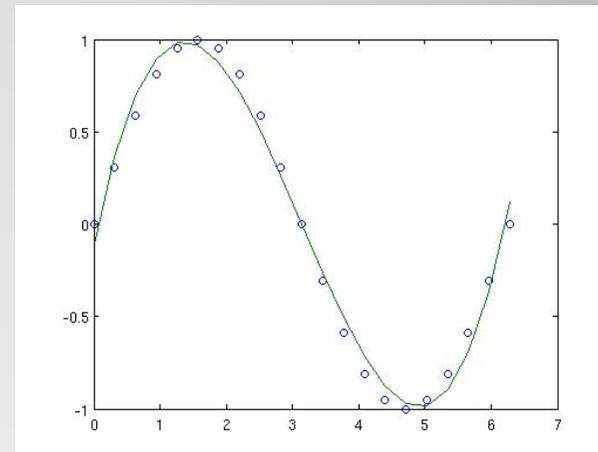
p =

-0.0000 0.0886 -0.8347 1.7861 -0.1192

- Avaliando o polinômio p:

```
>> f = polyval(p,x);
```

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



Polinômios

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

Operadores relacionais

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

Operadores lógicos

```
>> 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
```

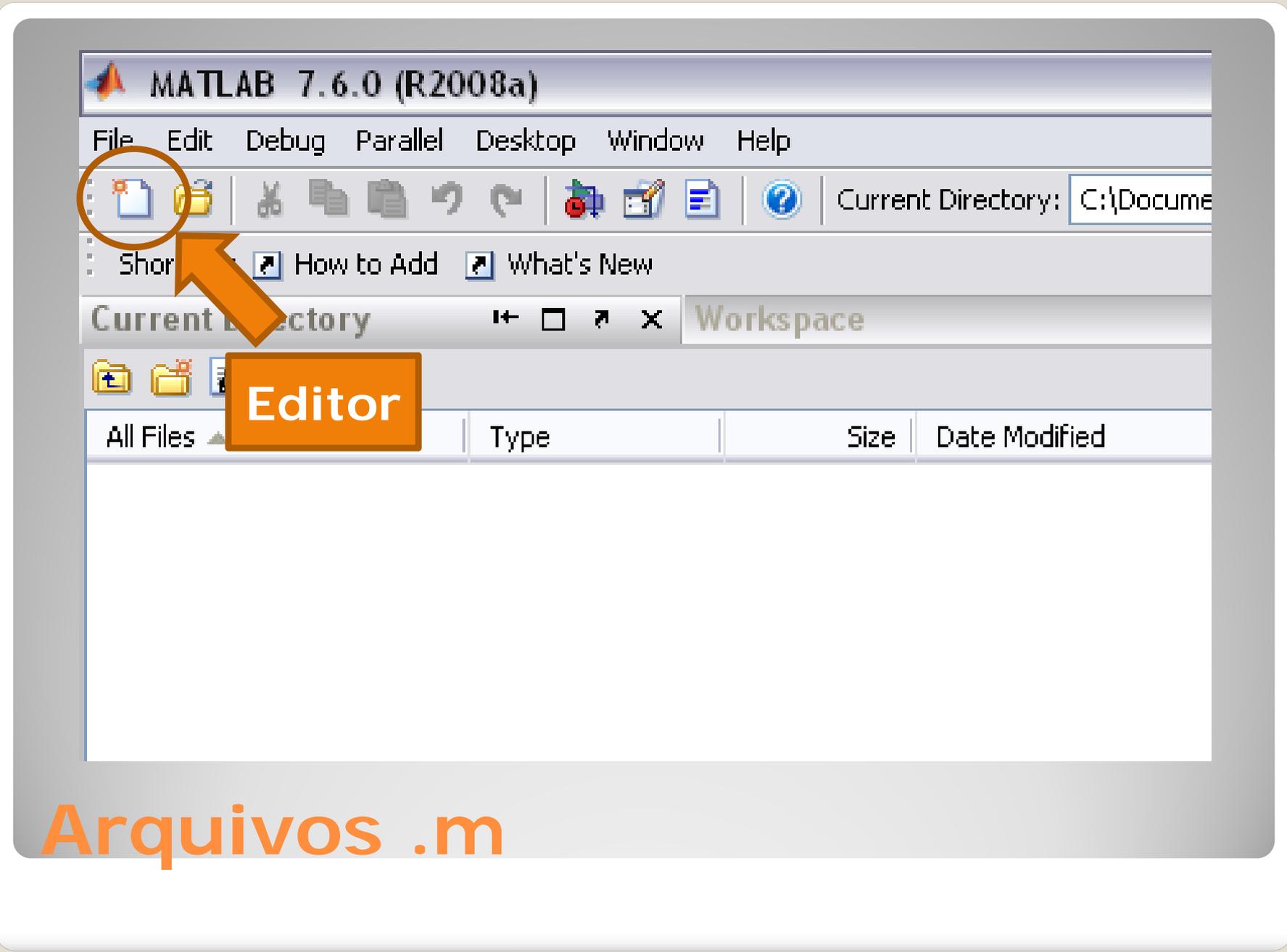
Exemplos

- 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

- 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*

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

- **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

- **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
```

Controladores 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

Funções auxiliares no controle 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
```

Controladores de fluxo

- 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)
```

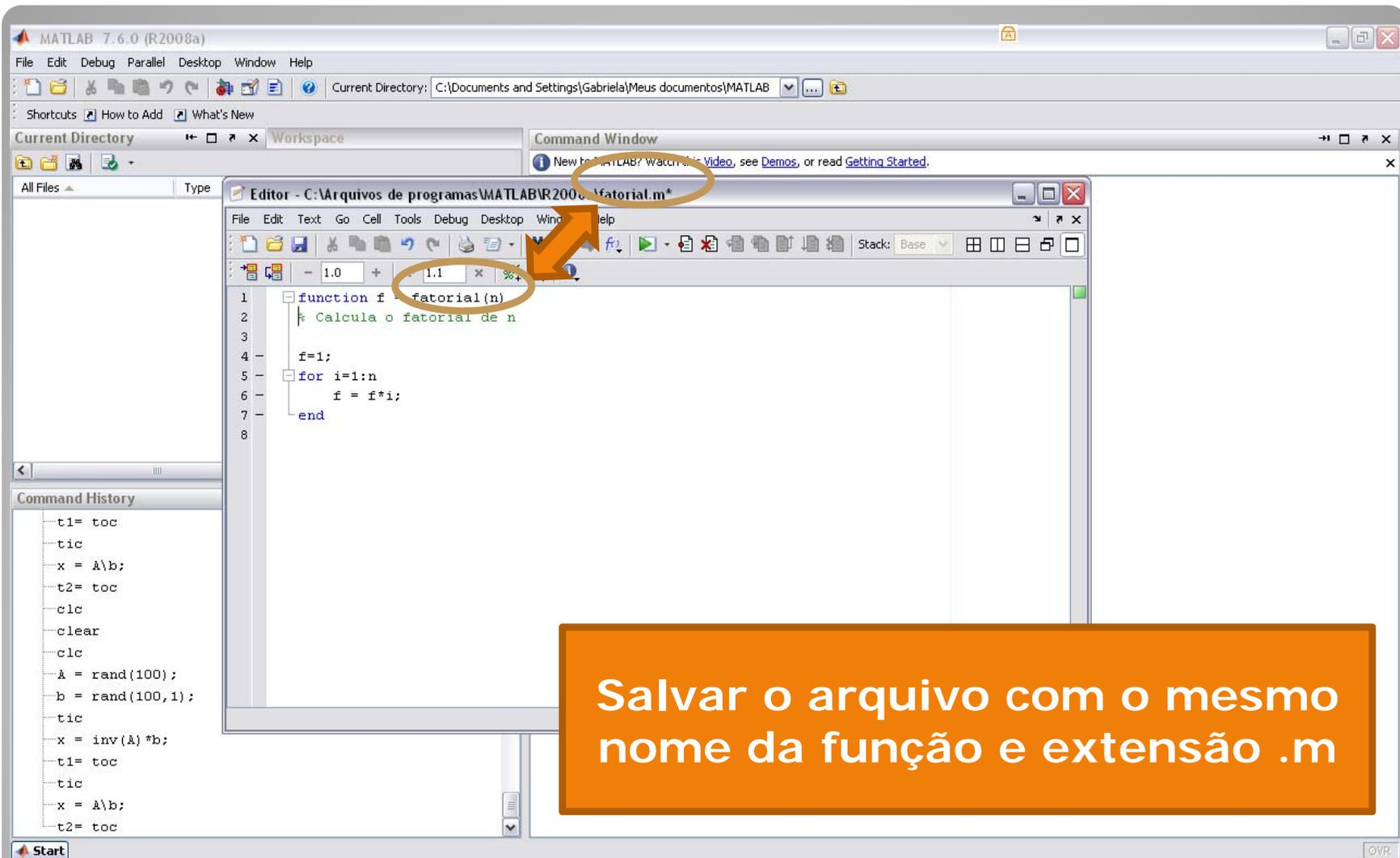
Scripts

- Exemplo de fatorial.m:

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

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

Funções



Salvar o arquivo com o mesmo nome da função e extensão .m

Funções

- MATLAB Central

<http://www.mathworks.com/matlabcentral>

The screenshot shows the MATLAB Central website interface. At the top, there is a search bar with "MATLAB Central" entered, and navigation links for "Create Account" and "Login". Below the search bar is a horizontal menu with options: "File Exchange", "Newsgroup", "Link Exchange", "Blogs", "Contest", and "MathWorks.com". The main content area is divided into three columns: "File Exchange", "Newsgroup", and "Blogs".

- File Exchange:** Described as "Files contributed by users of MATLAB, Simulink, and related products". It includes a "Submit a File" button, a "RECENT FILES" section with items like "Enigma" by James Roberts, "Matlab PCT integration with RTDA" by Winston Yu, "LINEAGE v1.0" by Kenneth Eaton, "getfilenames.m" by Joseph Burgel, "Dicom directory (of slices) to 3D volume image" by Ariel Balter, and "priority" by Ben Mitch. It also has a "POPULAR FILES" section with "Bidirectional Branch and Bound Solvers for Worst Case Loss Minimization" by Yi Cao.
- Newsgroup:** Described as "An open forum for everyone in the MATLAB and Simulink universe". It includes a "Post a Message" button, a "RECENT POSTS" section with items like "GUI : sharing variable between callback" by Bruce Crochat, "euler angles?" by Rafael Herrejon, "Finding the nth word in a given string?" by Pekka Kumpulainen, "exe files" by Dhruva satsangi, "Neural Network Weights Retrieval" by Faiyaz, and "Timer with serial port - program hangs" by Ashwini Deshpande. It also has an "ACTIVE THREADS" section with "Backpropagation Neural network Code problem" by Adeel.
- Blogs:** Described as "Weekly commentary from the people who design and build MathWorks products". It includes a "RECENT UPDATES" section with items like "LOREN ON THE ART OF MATLAB" (dated 06 APR 2009, 3:06 PM), "KEN & MIKE ON THE MATLAB DESKTOP" (dated 06 APR 2009, 08:37 AM), and "STEVE ON IMAGE PROCESSING" (dated 06 APR 2009, 03:00 AM). Each item has a "View archive" link.

On the left side of the page, there are several sections: "Now Available R2009a" with a "Learn more" link; "The Latest from MATLAB Central" with a list of updates including "New File Exchange is here!", "FAQ", "Walkthru videos", and "MATLAB Central Screensaver" with a "Download now" link; and "Upcoming Webinars" with "Model-Based Design of a SUV Anti-Rollover Control System (Upcoming)".

Repositório de arquivos .m

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

Gráficos no MATLAB

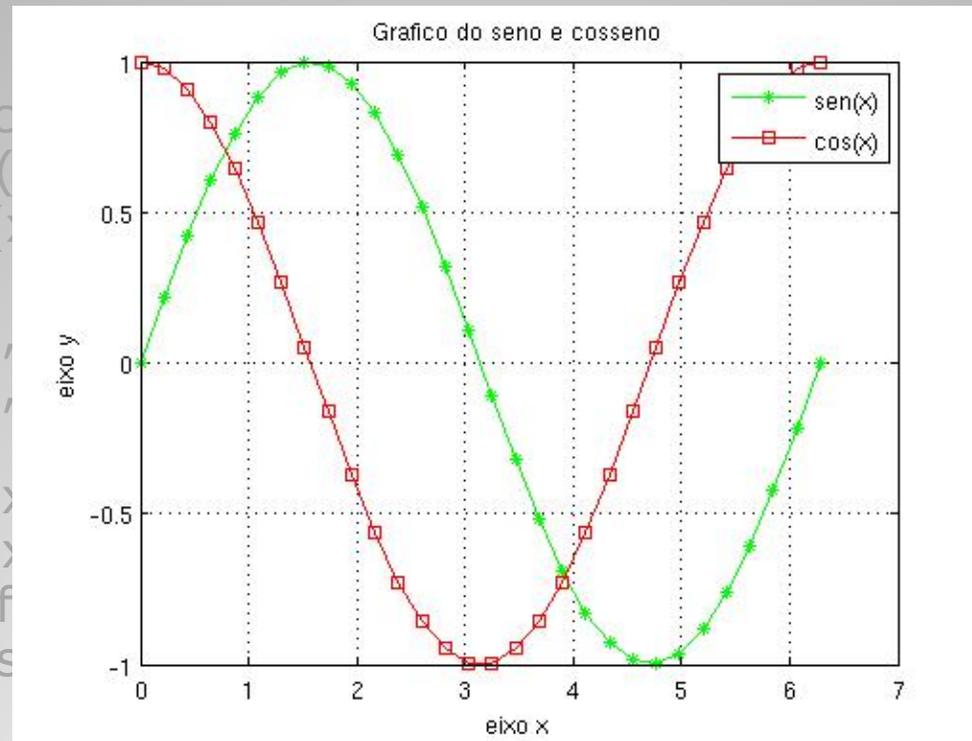
- Pode-se desenhar gráficos simples como $y = f(x)$

```
>> 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=linspace(0,6.28,100);
>> y1 = cos(x);
>> y2 = sin(x);
>> hold on
>> plot(x,y1,'r','o');
>> plot(x,y2,'g','*');
>> grid
>> xlabel('eixo x');
>> ylabel('eixo y');
>> title('Gráfico do seno e cosseno');
>> legend('s','c');
>> hold off
```



Gráfico

Gráficos 2D

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

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

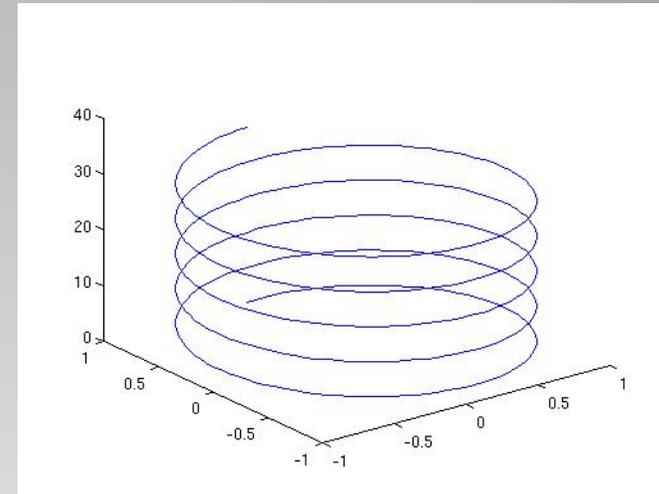
Características dos gráficos

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

Gráficos 3D

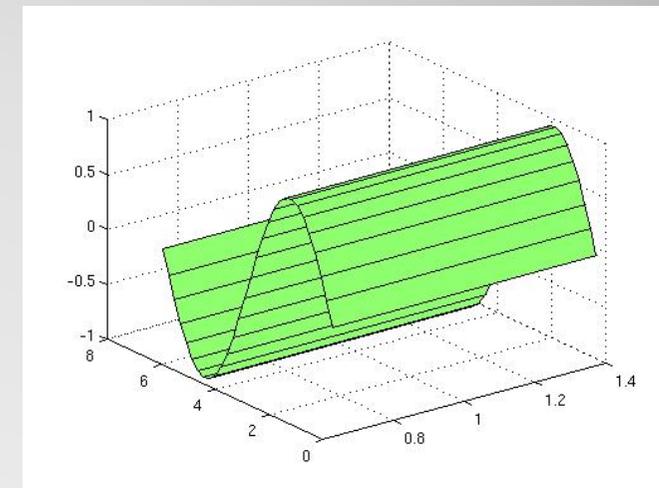
- Curvas no espaço

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



- Faixas no espaço

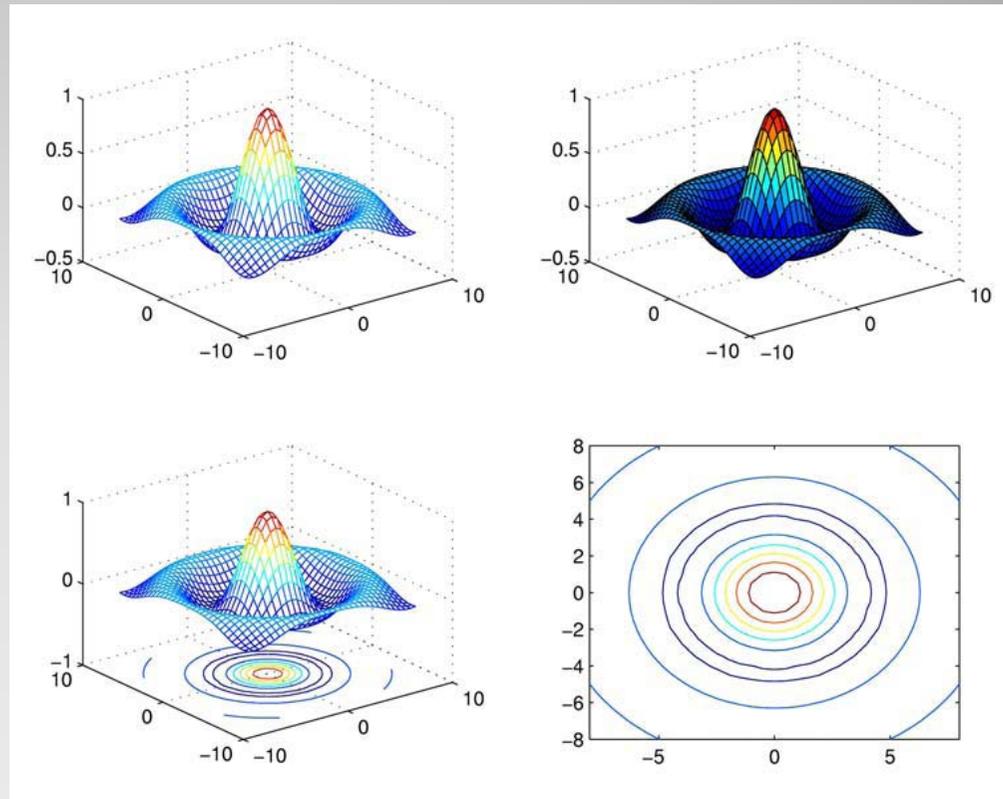
```
>> x=linspace(0,2*pi,30);  
>> ribbon(x,sin(x))
```



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)
```



Gráficos 3D

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

Help MATLAB on-line