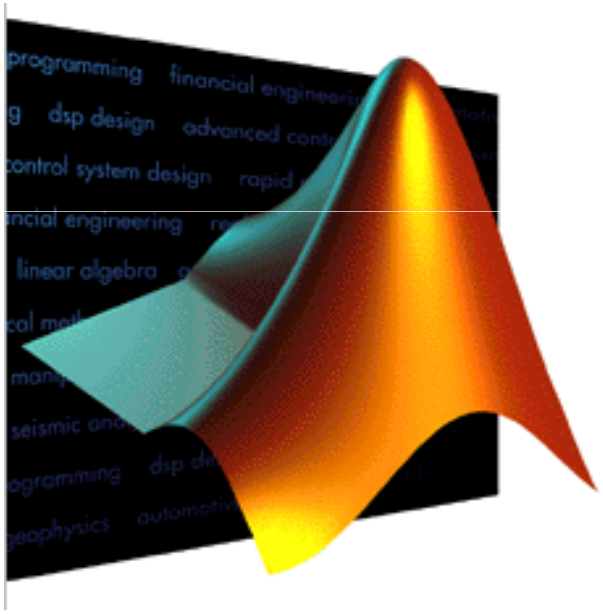
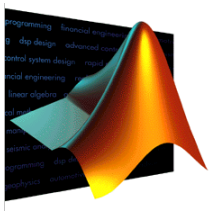
MATLAB

The Language of Technical Computing

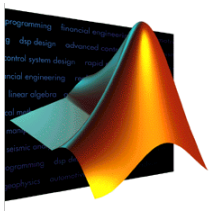


Carlos Alexandre Mello



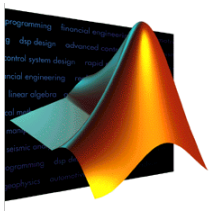
MATLAB
O Curso

- Computação
- Programação
- Visualização
- Simulink
- Toolbox de Sistemas de Controle



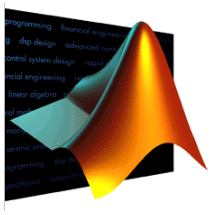
MATLAB

- O que é ?
  - Ferramenta de apoio à Engenharia
    - Cálculos Matemáticos
    - Desenvolvimento de Algoritmos
    - Análise, Exploração e Visualização de dados
    - Gráficos de engenharia
- MATLAB = *Matrix Laboratory*
  - Toolboxes (M-files)
  - Simulink : simulação de sistemas dinâmicos não-lineares
- Desenvolvido pela *Math Works*



MATLAB

- Tela de Entrada
- Comandos:
  - helpwin
  - demo
- Prompt: >>
  - Sistema pronto para executar tarefas
    - **Exemplo: >> v = [1 2 3; 4 5 6; 7 8 9]**



MATLAB
Funções de Gerenciamento de Memória

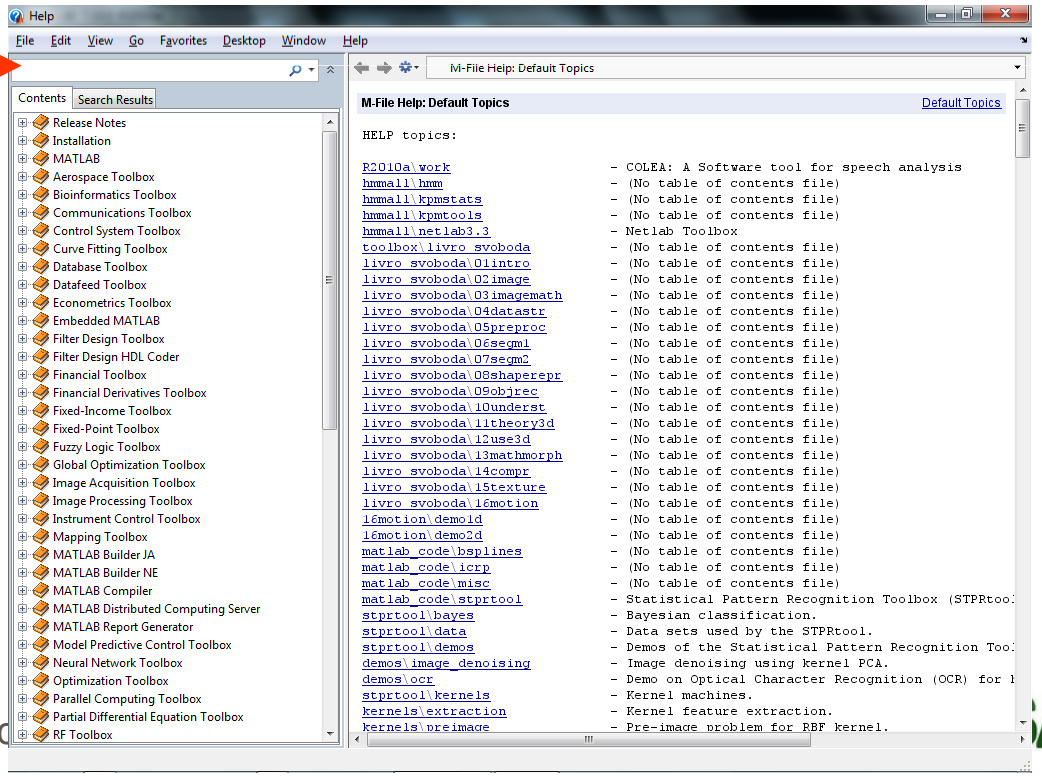
- whos
  - mostra a memória alocada

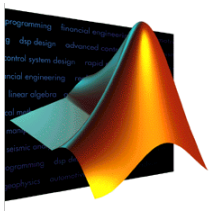
```
» y = zeros(1,10)

y =

     0     0     0     0     0     0     0     0     0     0

» whos
  Name        Size              Bytes  Class

  y           1x10                 80  double array

Grand total is 10 elements using 80 bytes

» clear
» whos
»
```

MATLAB
Help

- O Help (Ajuda) do MatLab pode ser acessado de duas formas diferentes:
  - Comando *helpwin*

O termo a ser pesquisado deve ser inserido aqui

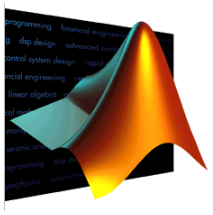




MATLAB
Help

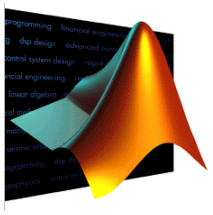
- Help do MatLab:
  - Comando *help* (usado no prompt):

```
» help sqrt

 SQRT   Square root.
    SQRT(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.

    See also SQRTM.
```



MATLAB
Workspace - Área de Trabalho

- Exemplo:
  - >> a = 2;
  - >> b = 3;
  - >> c = a + b
  - >> c = 5
- Caso nenhuma variável seja definida, o MatLab assume uma variável padrão chamada *ans* (de *answer* = resposta, em inglês)
- No exemplo anterior, teríamos:
  - >> a + b
  - >> ans = 5



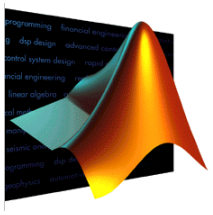
MATLAB
Workspace - Área de Trabalho

- Comentários
  - Qualquer texto precedido de %
- Pontuação
  - ; - Ponto e Vírgula
    - Suprime a visualização de um resultado

```
» a = 2

a =

     2

» a = 2;
»
```

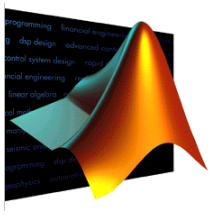
Sem ponto e vírgula o resultado é repetido na tela

Com ponto e vírgula o resultado é apenas armazenado na memória



MATLAB
Variáveis Especiais no MatLab

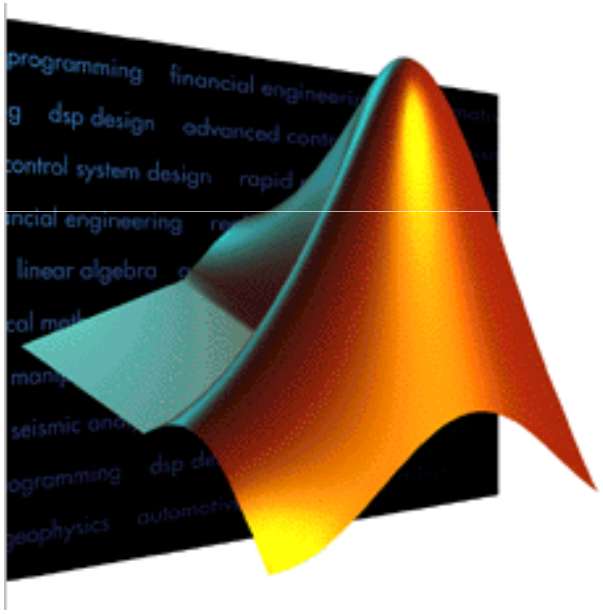
- Valores Especiais:
  - pi — 3.1415926535897....
  - i,j — Unidade Imaginária
  - inf — Infinito (n/0 = inf)
  - NaN — Not-a-Number (inf/inf)
  - version — Versão do MatLab
  - computer — Tipo de computador
  - flops — Contagem de operações de ponto flutuante



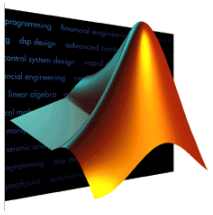
MatLab
Números Complexos

- Representados pelas letras *i* ou *j*
  - >> a=1 - 2i;
  - >> a = 1 - 2j;
- Uma das poucas variáveis do MatLab que permite a notação com o uso da variável i ou j justaposta sem o sinal da multiplicação
  - 2i = 2*i
  - 2j = 2*j

Matemática Elementar

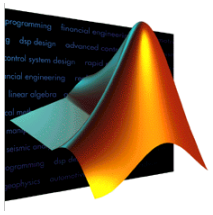


Carlos Alexandre Mello



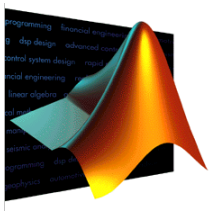
MATLAB
Matemática Elementar

- Operações aritméticas:
  - Adição:
    - a+b
  - Subtração
    - a-b
  - Multiplicação
    - a*b
  - Divisão
    - a/b ou b\a (b divide a)
  - Potenciação
    - a^b
  - Raiz Quadrada
    - sqrt(a)



MATLAB
Funções Matemáticas Elementares

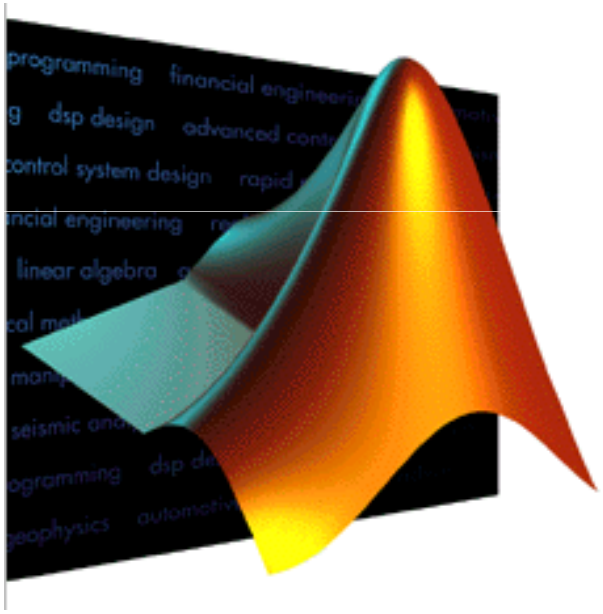
- Funções trigonométricas básicas em Radianos
  - sin, cos, tan, asin, acos, atan
- Arredondamento
  - ceil(x)
    - Arredondamento para cima
  - floor(x)
    - Arredondamento para baixo
  - fix(x)
    - Arredondamento na direção do zero
  - round(x)
    - Arredondamento para o número inteiro mais próximo



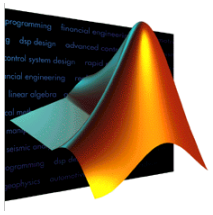
MATLAB
Funções Matemáticas Elementares

- Números Inteiros
  - rem(x,y)
    - Resto da divisão de x por y
  - sign(x)
    - Função sinal: retorna o sinal de x; igual a 1, se x > 0, -1, se x < 0 e 0, se x = 0
  - gcd(x,y)
    - Máximo Divisor Comum dos inteiros x e y
  - lcm(x,y)
    - Mínimo Múltiplo Comum dos inteiros x e y

Vetores

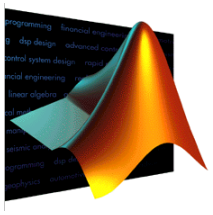


Carlos Alexandre Mello



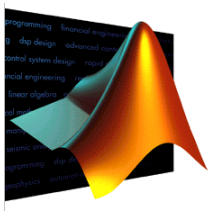
MATLAB
Vetores

- Vetores
  - Matrizes com apenas uma linha
  - Conjunto de elementos que podem ser números inteiros, reais, complexos ou caracteres
    - >> x = [1 2 3 4 5];
  - Elementos separados por espaços em branco e delimitados por colchetes [ ]
    - >> y = [‘matlab’ ‘versão 5’];
  - Caracteres sempre entre apóstrofos ‘’ !!



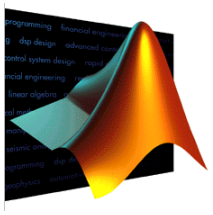
MATLAB
Vetores

- Endereçamento Vetorial
  - Todos os elementos de um vetor podem ser acessados por um índice
    - >> x = [1 4 7];
    - x(1) = 1 x(2) = 4 x(3) = 7
  - Cuidados:
    - O primeiro elemento tem índice 1
    - Não é possível acessar um elemento que não exista no vetor:
      - x(4) = ERRO !!!!



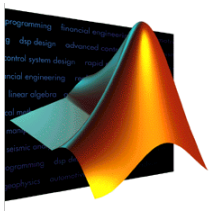
MATLAB
Vetores

- Endereçamento Vetorial
  - Observação:
    - x = [‘matlab’ ‘versão 5’]
      - x(1) = ‘m’ x(12) = ‘o’
    - Cada caractere é um elemento do vetor !
  - É possível acessar um bloco de elementos com o operador *dois pontos* (:)
    - >> x= [1 4 6 2 7 8 3];
    - >> x(1:4) = [1 4 6 2]
      - Todos os elementos de x do 1o. ao 4o.



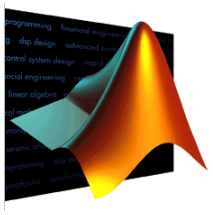
MATLAB
Vetores

- Endereçamento Vetorial
  - Pode-se acessar os elementos em ordem inversa
    - >> x= [1 4 6 2 7 8 3];
    - >> x(4:-1:1) = [2 6 4 1]
      - Todos os elementos de x do 4o. ao 1o. de -1 em -1
      - -1 = Passo
    - O passo pode ser qualquer número inteiro positivo ou negativo



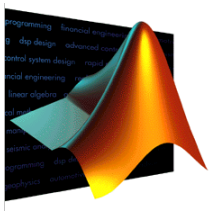
MATLAB
Vetores

- Construção automática de vetores
  - >> a = 1:5
  - >> a = (1:5)
    - Cria um vetor a com elementos 1, 2, 3, 4, 5
  - >> a=1:2:5
  - >> a=(1:2:5)
    - Cria um vetor a com elementos 1, 3, 5
    - Passo = 2
  - Como antes, o passo pode ser negativo
    - >> a =(5:-1:1)*pi



MATLAB
Vetores

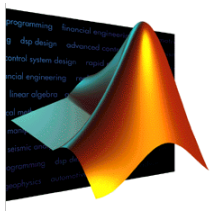
- Construção automática de vetores
- Dados 2 vetores quaisquer a e b:
  - a = [1 2 5]
  - b = [3 4 0]
- pode-se construir um terceiro vetor c através da junção de a e b
  - c = [a b]
  - c = [1 2 5 3 4 0]
  - Se c = [b a], então c = [3 4 0 1 2 5]



MATLAB
Vetores

- Orientação de Vetores
  - Transposição de Vetores
    - Vetores Reais

```
» a=1:5

a =

     1     2     3     4     5

» b=a'

b =

     1
     2
     3
     4
     5

»
```



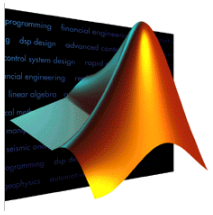
MATLAB
Vetores

- Orientação de Vetores
  - Transposição de Vetores
    - Vetores complexos

```
» a=[1+i   2+3i]

a =

   1.0000+ 1.0000i   2.0000+ 3.0000i

» b=a'

b =

   1.0000- 1.0000i
   2.0000- 3.0000i

» b=a.'

b =

   1.0000+ 1.0000i
   2.0000+ 3.0000i
```

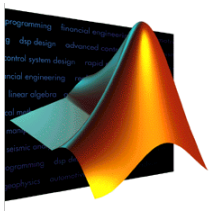
Operador ponto...

...gera a matriz transposta conjugada



MATLAB
Vetores
Funções Matemáticas Elementares

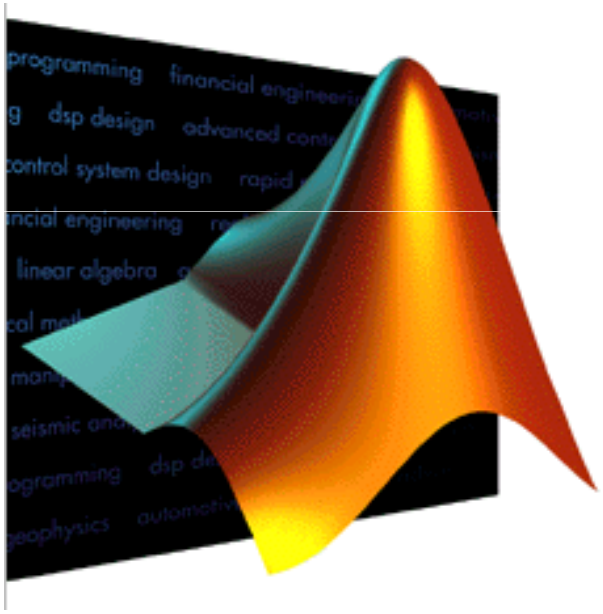
- **Para um vetor:**
  - **max(x)**
    - Elemento máximo
  - **min(x)**
    - Elemento mínimo
  - **mean(x)**
    - Média
  - **median(x)**
    - Mediana
  - **prod(x)**
    - Produto dos elementos
  - **sum(x)**
    - Soma dos elementos
  - **sort(x)**
    - Ordenação dos elementos de forma decrescente



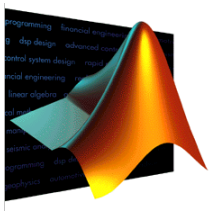
MATLAB
Vetores e Matrizes

- Vetores Especiais
  - rand(1,n)
    - Cria um vetor de n elementos onde cada elemento é um número *aleatório* entre 0 e 1
      - Exemplo: rand(1,5)
    - Dica: Para gerar um número (UM número apenas, não um vetor!!) aleatório entre 0 e X usa-se o comando rand da seguinte forma:
      - rand(1)*X
      - Exemplo: rand(1)*10
        - Cria um número REAL entre 0 e 10
      - Exemplo: round(rand(1)*10)
        - Cria um número INTEIRO entre 0 e 10

Matrizes

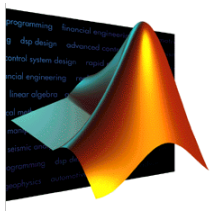


Carlos Alexandre Mello



MATLAB
Matrizes

- Definição de Matrizes
- Manipulação de Matrizes
- Sub-matrizes
- Comparação
- Dimensão
- Operações

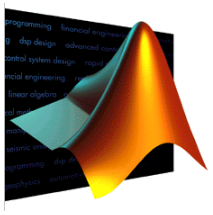


MATLAB
Matrizes

- Criação de Matrizes

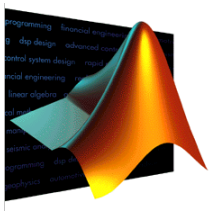
```
» a=[1 2 3; 4 5 6; 7 8 9]

a =

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

- Ponto-e-vírgula é usado para separar as linhas da matriz
- CUIDADO!!!
  - Todas as linhas da matriz devem ter o mesmo número de elementos!!!



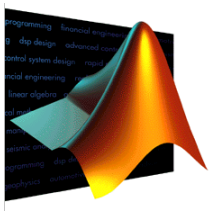
MATLAB
Matrizes

- Operações Escalares
  - Seja B um vetor ou uma matriz, são válidas quaisquer operações com escalares:
    - B - 2
    - 3*B - 4
    - B/2
    - B + 5



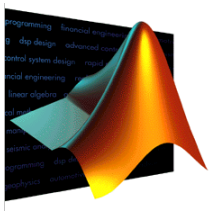
MATLAB
Matrizes

- Operações entre Matrizes
  - Adição e Subtração são válidas sempre que as matrizes envolvidas tiverem as mesmas dimensões
  - Multiplicação e Divisão devem obedecer às normas relacionadas as dimensões das matrizes:
    - Amxn*Bnxp = Cmxp
    - Amxp/Bnxp=Cmxn
    - A/B = inv(B)*A, se B for quadrada



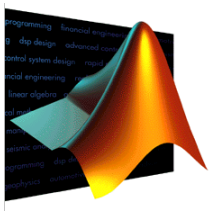
MATLAB
Matrizes

- Operações entre Matrizes
  - Operações Pontuadas
    - Sejam A e B duas matrizes
      - A.*B -> Multiplica cada elemento de A pelo correspondente em B
      - O mesmo acontece na divisão em A./B
  - Exponenciação
    - A.^2 -> Eleva cada elemento de A ao quadrado (ou qualquer outra potência)
    - A^2 = A*A



MATLAB
Matrizes

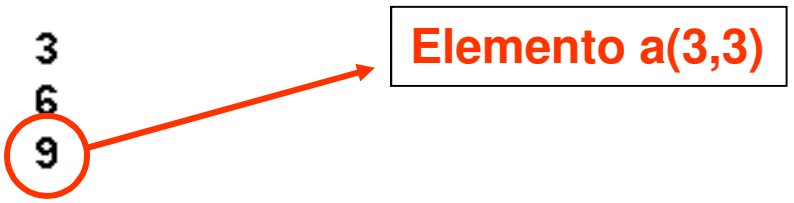
- Matrizes Especiais
  - ones(m,n)
    - Cria uma matriz mxn onde todos os seus elementos são iguais à 1
  - zeros(m,n)
    - Cria uma matriz mxn onde todos os seus elementos são iguais à 0



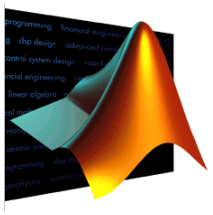
MATLAB
Matrizes

- Manipulação de Matrizes
  - Seja:

```
» a=[1 2 3; 4 5 6; 7 8 9]

a =

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



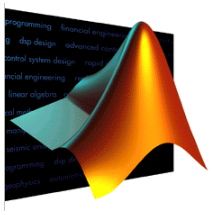
  - a(3,3)=0 faz com que o elemento da 3a. linha e 3a. coluna seja igual a zero



MATLAB
Matrizes

- Manipulação de Matrizes
  - a(:,3) lê todos (:) os elementos da terceira coluna de a
  - a(3,:) lê todos os elementos da terceira linha de a

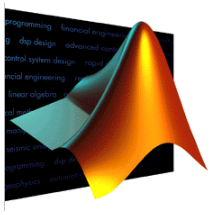
```
» a(:,3)

ans =

     3
     6
     9

» a(3,:)

ans =

     7     8     9
```



MATLAB
Vetores e Matrizes

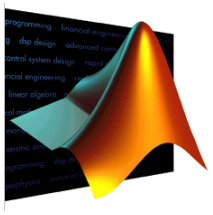
- Vetores Lógicos
  - Construção de Vetores a partir de vetores lógicos

```
» y=a(abs(a) > 1)

y =

    -3    -2     2     3
```



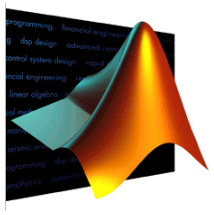
MATLAB
Vetores e Matrizes

- Dimensão de Vetores e Matrizes
  - size(A)
    - Retorna um vetor [n_linhas n_colunas]
    - Qualquer elemento do vetor pode ser acessado como um vetor qualquer
      - >> x = size(A)
      - >> x = [2 3]
      - x(1) = 2 x(2) = 3
  - length(A)
    - Retorna apenas o número de linhas OU o número de colunas (o que for maior)
      - Corresponde a max(size(A))



MATLAB
Vetores e Matrizes

- Matrizes Especiais
  - rand(m,n)
    - Cria uma matriz mxn onde cada elemento é um número *aleatório* entre 0 e 1



MATLAB
Resolvendo Equações Lineares
Vetores e Matrizes

- AX = B

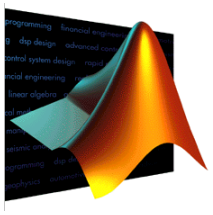
$$\begin{cases} w + 2y + z = 3 \\ 2w - y + 2z = 1 \\ y + 3z = 2 \end{cases}$$

Ax = B, logo: x = A\B

```
» A = [1 2 1; 2 -1 2; 0 1 3]

A =

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

```
» B = [3; 1; 2]

B =

     3
     1
     2
```

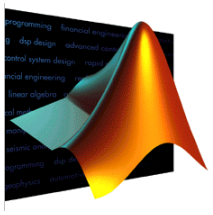
```
» x = A\B

x =

    0.6667
    1.0000
    0.3333
```



MATLAB
Funções Matriciais

- A Matriz Identidade I é tal que $A*A^{-1} = I$ e $A^{-1}*A = I$
- É calculada no MatLab com a função *eye*
  - eye(m)
    - Matriz identidade quadrada mxm

```
» eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1
```

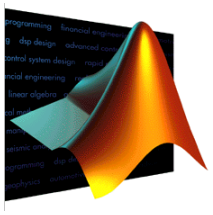


MATLAB
Funções Matriciais

- O inverso de uma matriz A é calculado com a função *inv(A)*
- Também é possível calcular o determinante de uma matriz A mxn com a função *det(A)*

```
» A = [1 2 1; 2 -1 2; 0 1 3]

A =

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

```
» inv(A)

ans =

    0.3333    0.3333   -0.3333
    0.4000   -0.2000         0
   -0.1333    0.0667    0.3333

» det(A)

ans =

   -15
```



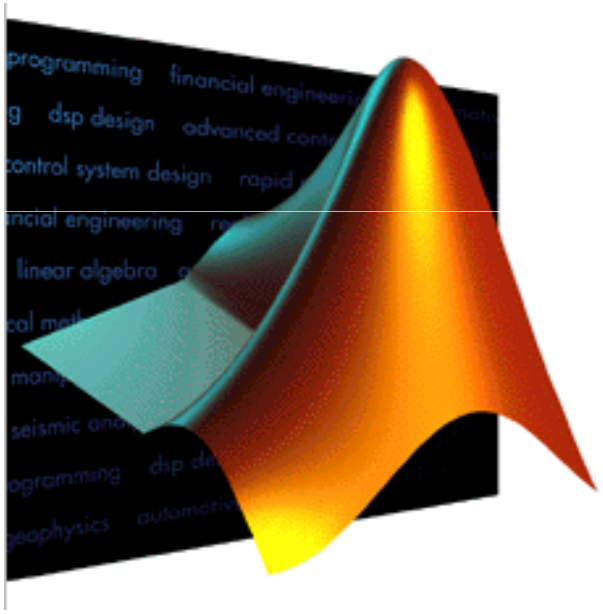
MATLAB
Potências e Exponenciais

- Exemplo:

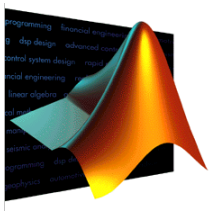
```
A = [1 2 3; 4 5 6; 7 8 9];
```

X = A^2	⇒	X = A*A
Z = A.^2	⇒	Z = [a i j ^2]
Y = sqrtm(X)	⇒	X^(1/2)
Y = sqrt(X)	⇒	X.^(1/2)

Polinômios



Carlos Alexandre Mello



MATLAB
Polinômios

- MATLAB representa polinômios como matrizes linha contendo os coeficientes ordenados de forma decrescente pela potência. Por exemplo, o polinômio

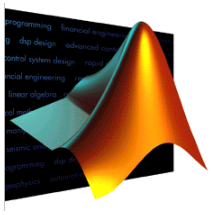
$$p(x) = x^3 - 2x - 5$$

é representado no MATLAB como:

```
p = [1 0 -2 -5]
```

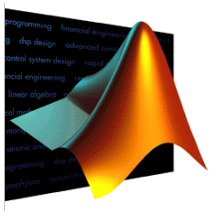
Observe a presença do coeficiente de valor zero !!!

- >> r = roots(p)
  - Calcula as raízes do polinômio
- >> p2 = poly(r)
  - Retorna os coeficientes do polinômio dadas as suas raízes



MATLAB
Polinômios

- polyval(p, n)
  - Resolve o polinômio para o valor *n*
- polyvalm(p,X)
  - Resolve o polinômio p para uma matriz X
  - No caso, por exemplo:
    - **p(X) = X^2 + 3*X + X*I**
    - **onde I é a matriz identidade**

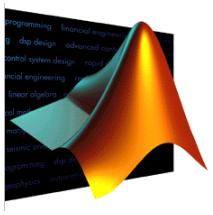


MATLAB
Polinômios

- Multiplicação e divisão polinomial corresponde às operações de convolução e deconvolução, implementadas pelas funções *conv* e *deconv*
- Exemplo: a(x)*b(x)

```
>> a = [1 2 3]; b = [4 5 6];
>> c = conv(a,b)
>> c = [4   13   28   27   18]
```

  - **comprimento de c = comprimento de a + comprimento de b - 1**



MATLAB
Convolução e Deconvolução
Polinômios

- Exemplo: c(x)/a(x)

```
» [q,r] = deconv(c,a)
```

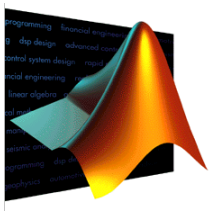
Quociente

```
q =

     4     5     6
```

= b(x)

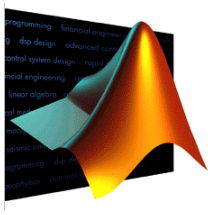
Resto

```
r =

     0     0     0     0     0
```



MATLAB
Adição de Polinômios
Polinômios

- Não há função direta para o cálculo
- Alguns cuidados devem ser tomados:
  - $A(x)= x^3 + 3x +4$
  - $B(x)= 3x^2 + 2x + 1$
  - $C(x)=A(x)+B(x) = x^3 + 3x^2 + 5x + 5$



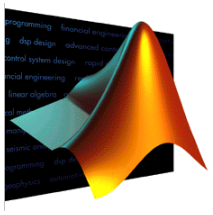
MATLAB
Adição de Polinômios
Polinômios

- No MatLab:
  - A=[1 0 3 4]
  - B=[3 2 1]
  - C=A+B

ERRADO

  - B=[0 3 2 1]
  - C=A+B

CERTO



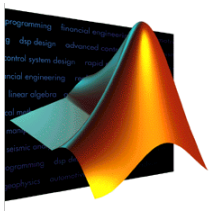
MATLAB
Derivativa Polinomial
Polinômios

- A função *polyder* calcula a derivada de qualquer polinômio
  - Exemplo:

```
>> p = [1 0 -2 -5];
>> q = polyder(p)
>> q = 3 0 -2
```

- A função também calcula a derivada do produto de funções:

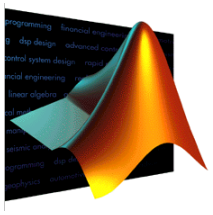
```
>> c = polyder(a, b)
```



MATLAB
Derivativa Polinomial
Polinômios

- A derivada da divisão de dois polinômios é calculada chamando polyder com **2** argumentos de saída

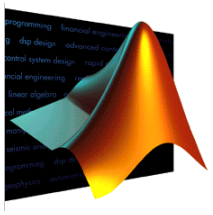
```
» [q,d] = polyder(a,b)

q =

    -2    -8    -2


d =

     4    16    40    48    36

» Cálculo de derivada de a/b
  é dado por q/d
```



MATLAB
Expansão em Frações Parciais
Polinômios

- A função *residue* encontra a expansão em frações parciais da razão de dois polinômios
- Exemplo:

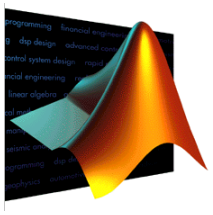
$$\frac{-4 + 8s^{-1}}{1 + 6s^{-1} + 8s^{-2}} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \ldots + \frac{r_n}{s - p_n} + ks$$



MATLAB
Expansão em Frações Parciais
Polinômios

- Exemplo (cont):

```
» b = [-4 8];
» a = [1 6 8];
» [r,p,k] = residue(b,a)

r =

   -12
     8

p =

    -4
    -2

k =

     []
```

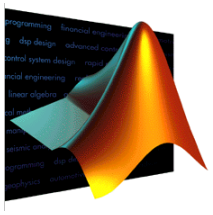


MATLAB
Expansão em Frações Parciais
Polinômios

- Exemplo (cont): Volta ao polinômio original

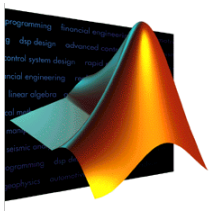
```
» [b2,a2] = residue(r,p,k)

b2 =

    -4     8


a2 =

     1     6     8
```

$$\frac{-4 + 8s^{-1}}{1 + 6s^{-1} + 8s^{-2}}$$



MATLAB
Representando Polinômios Graficamente

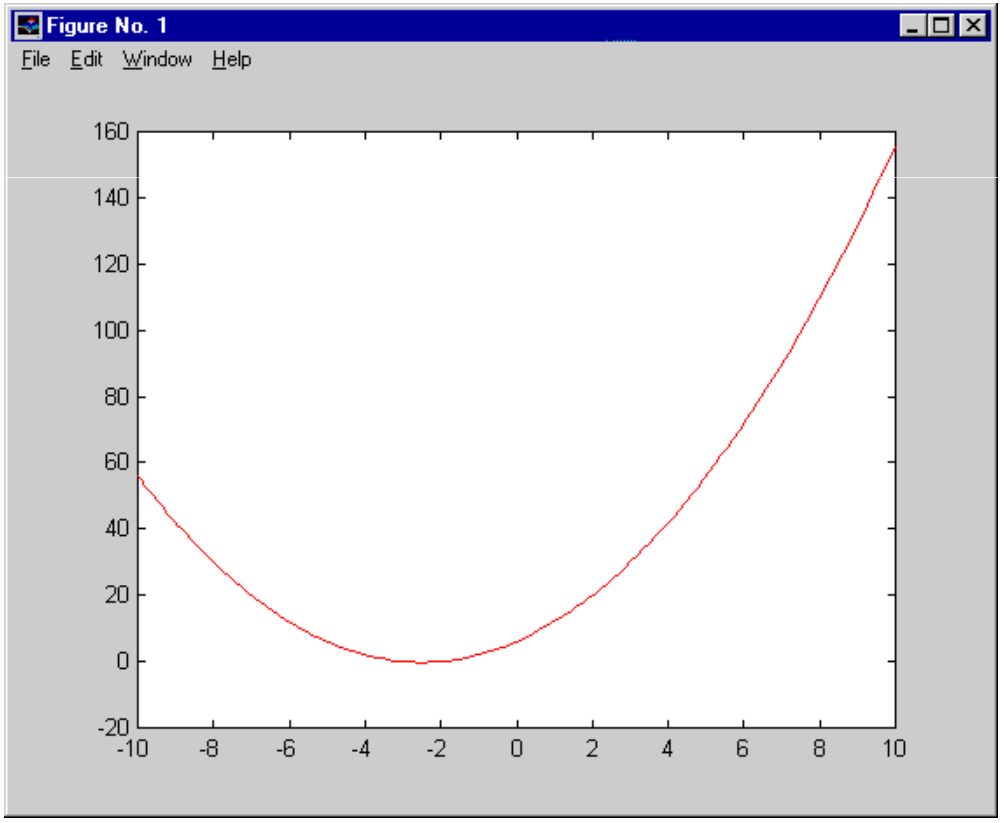
- Para traçar o gráfico de um polinômio, pode -se usar a função fplot:
  - fplot ('função ou nome', [início fim da plotagem])

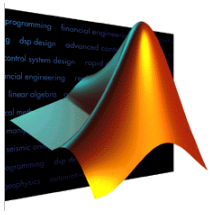


MATLAB
Representando Polinômios Graficamente

- Exemplo:

```
» fplot ('x^2 + 5*x + 6', [-10 10])
»
```





MATLAB
Função de Funções

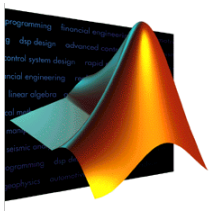
- MATLAB representa funções matemáticas expressando-as em arquivos-M
- Por exemplo, considere a função:

$$f(x) = 1/(x^2 + 2)$$

Essa função pode ser salva em um arquivo-M (nome.M) como:

```
function y = nome(x)
...
y=1./(x.^2 + 2);
...
```

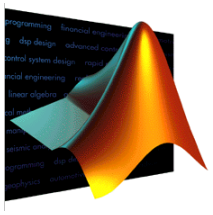
Observe esses pontos !!



MATLAB

Função de Funções

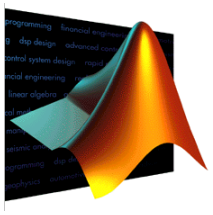
- Quando definimos uma função dentro de um arquivo-M no MatLab, precisamos utilizar o operador . (ponto) sempre que nos depararmos com operações de *, /,\ e ^ que envolvam a variável de entrada
  - Como não há uma definição de tipos no MatLab, isso diz a ele que trabalhe com qualquer tipo que seja a variável de entrada (número, matriz, vetor, etc).
- Isso não mudará em nada sua função !! É só uma forma de trabalho do MatLab



MATLAB
Encontrando Zeros de Funções
Funções de Função

```
» Por exemplo, para a função y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
» a = fzero('y', -0.2)

a =

    -0.1316
```

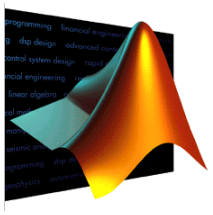
Acha o zero da função próximo ao ponto -0.2



MATLAB
Integração Numérica
Funções de Função

- A área abaixo de uma seção de uma função *F(x)* pode ser determinada pela integração numérica de F(x), processo chamado de *quadratura*

função	descrição
quad	**regra de Simpson Adaptativa**
quad8	**regra de Newton Cotes Adaptativa**



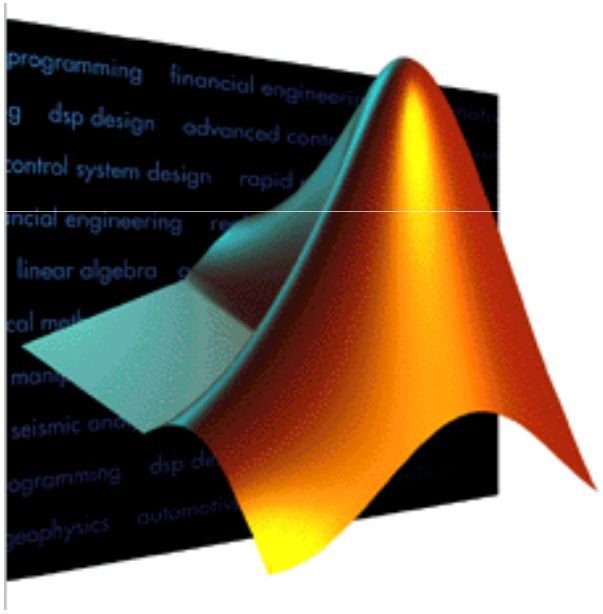
MATLAB
Integração Numérica
Funções de Função

- Por exemplo, para integrar a função definida no arquivo de teste *humps* entre 0 e 1, usamos:

```
» q = quad ('humps', 0,1)

q =

   29.8583
```

- Integração dupla: Função *dblquad*
  - **>> dblquad ('*filename*', xmin, xmax, ymin,ymax)**
    - **Problemas com a função na atual versão do MatLab !!!**

Gráficos

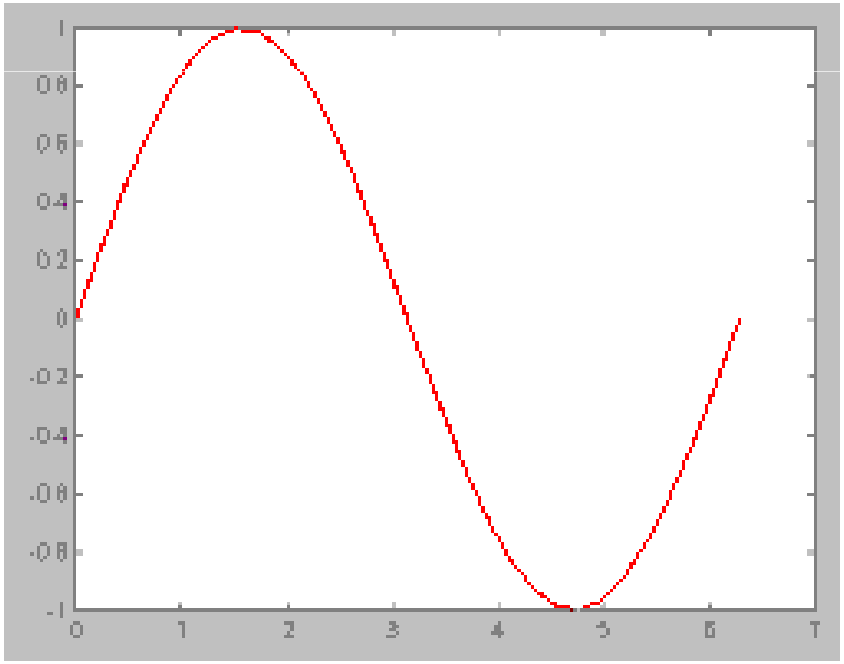


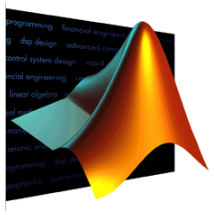
Carlos Alexandre Mello

Gráficos Bidimensionais

- Comando *plot*

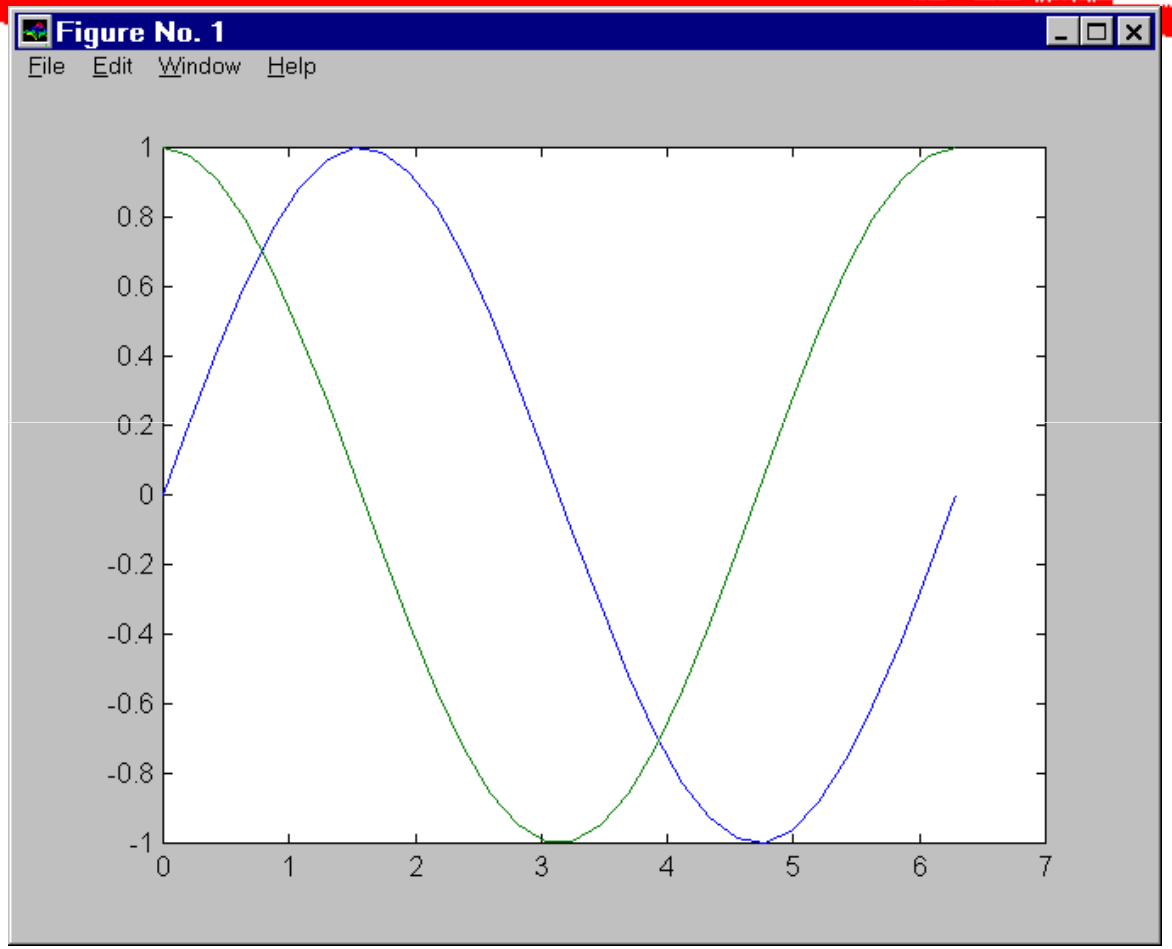
```
>> x=linspace (0,2*pi,30);  %Cria um vetor
>> y=sin(x);
>> plot(x,y)                %Plota o gráfico x versus y
```

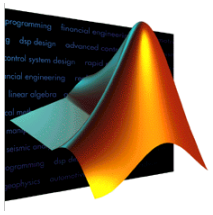




Gráficos Bidimensionais

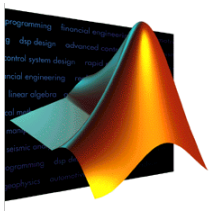
```
>> z=cos(x);
>> plot (x,y,x,z);
```





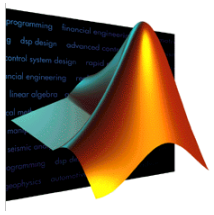
Gráficos Bidimensionais

- Comando *plot*:
  - Estilos de linha, marcadores e cores:
  - plot(X,Y,s)
    - **s = y,m,c,r,g,b,w,k (cores)**
    - **s = ., o, x, +, *, s, d, v, ^, <, >, p, h**
    - **s = -, :, -., --**

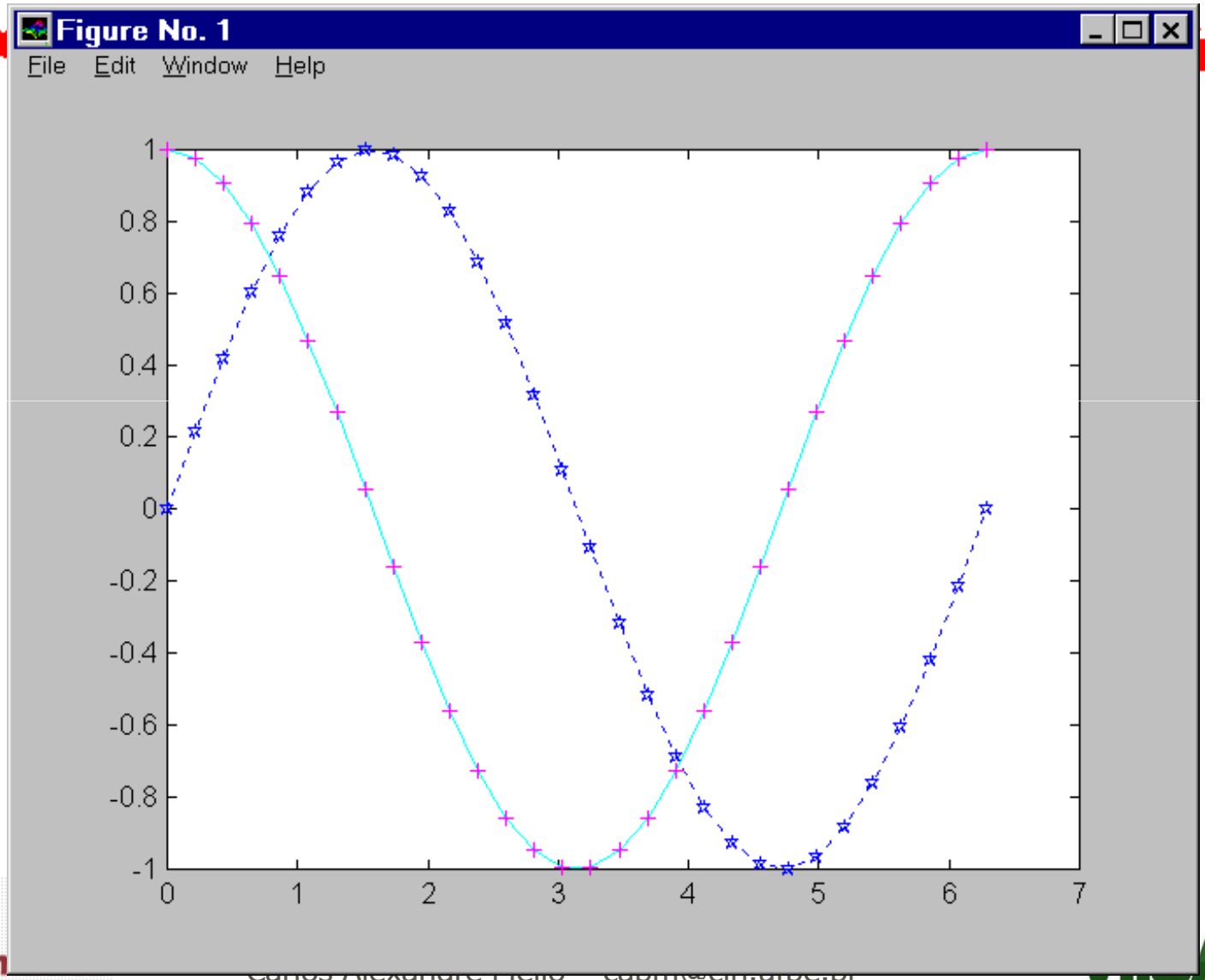


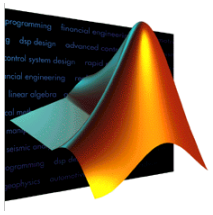
Gráficos Bidimensionais

- Comando *plot*:
  - Estilos de linha, marcadores e cores: Exemplo:
    - **>> plot(x,y,'b:p',x,z,'c-', x,z,'m+')**



Gráficos Bidimensionais



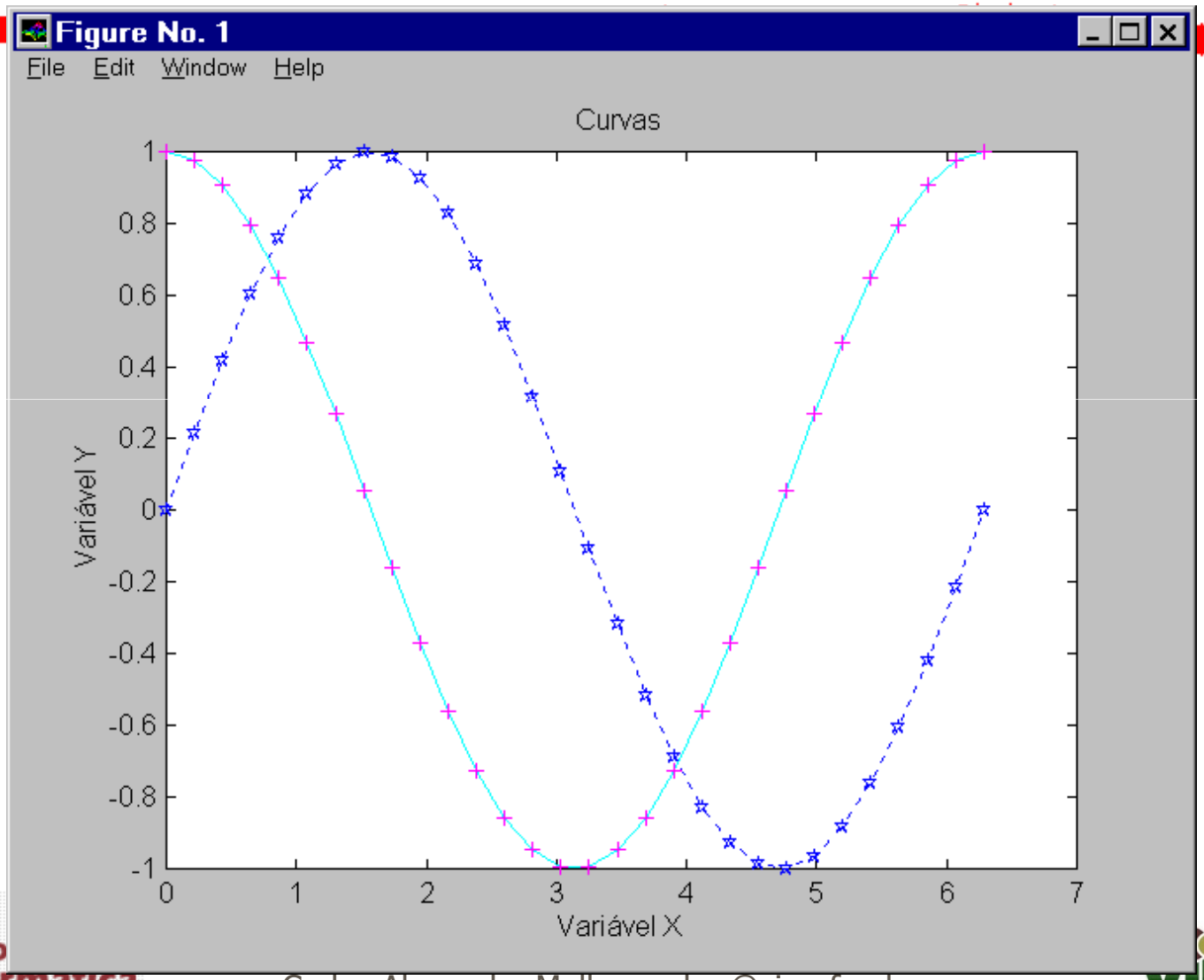


Gráficos Bidimensionais

- Legendas

```
>> xlabel('Variável X');  % Legenda do eixo horizontal
>> ylabel ('Variável Y'); % Legenda do eixo vertical
>> title ('Curvas'); % Título do Gráfico
```

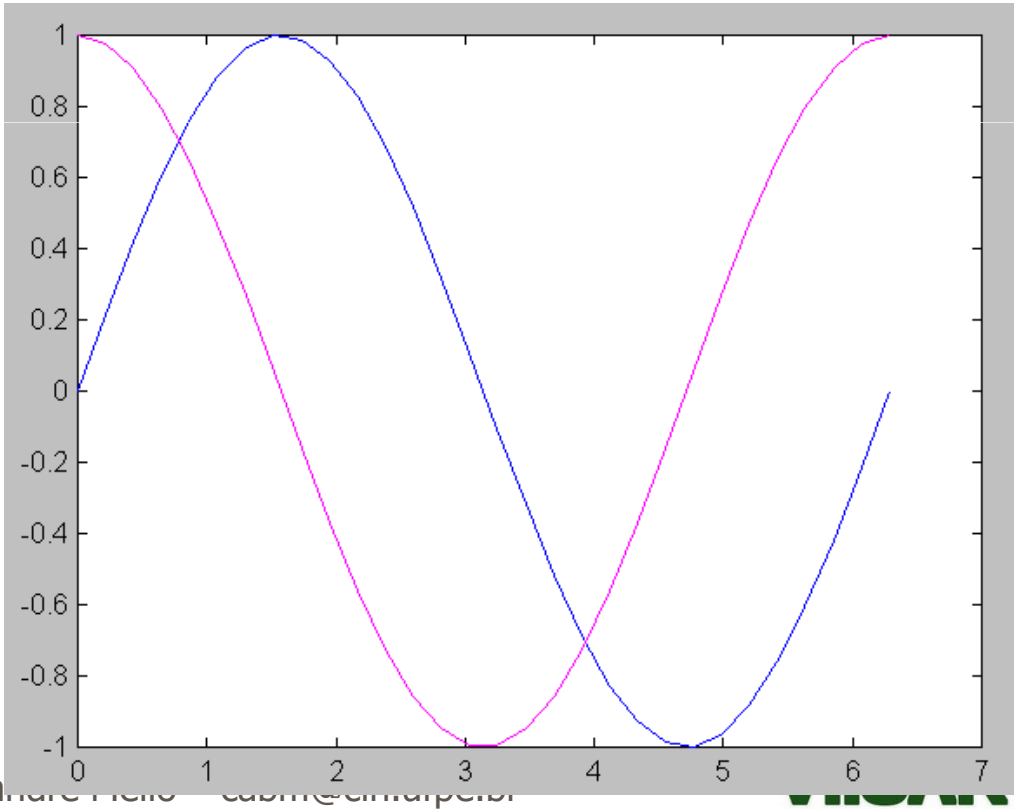
Gráficos Bidimensionais

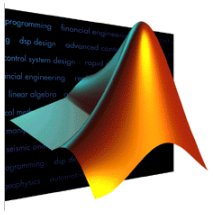


Gráficos Bidimensionais

- Manipulação de Gráficos

```
>> x=linspace(0,2*pi,30);
>> y=sin(x);
>> z=cos(x);
>> plot(x,y);
>> hold on
>> plot (x,z,'m ')
>> hold off
```

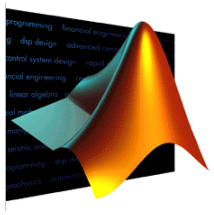




Gráficos Bidimensionais

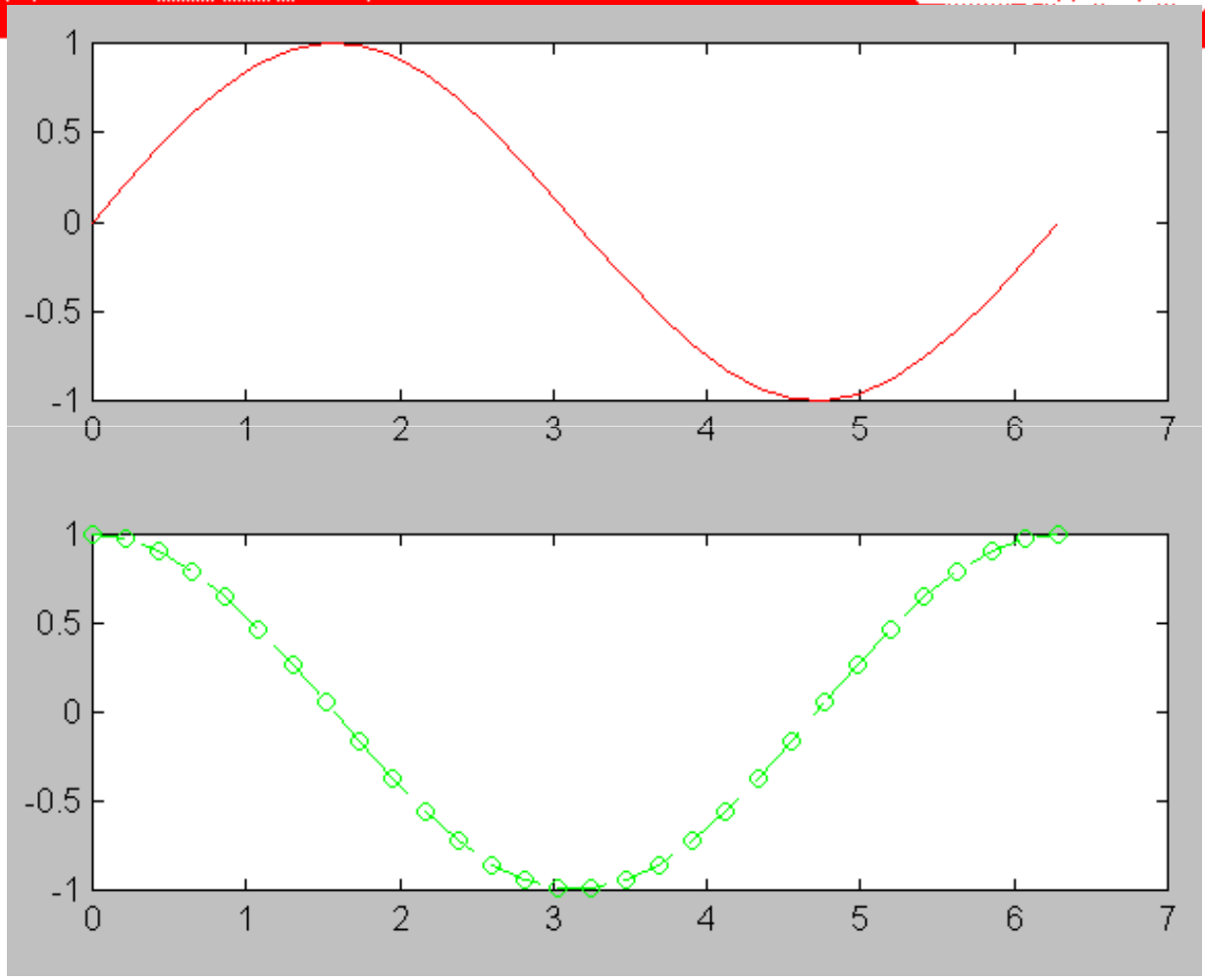
- subplot
  - subplot(m,n,p) divide a janela de figura em uma matriz mxn de pequenos eixos e seleciona o p-ésimo eixo para a atual plotagem
- Exemplo:

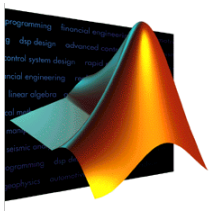
```
» subplot(2,1,1)
» plot(x,y,'r-');
» subplot(2,1,2);
» plot (x,z,'g--o');
»
```



Gráficos Bidimensionais

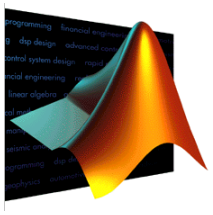
- subplot





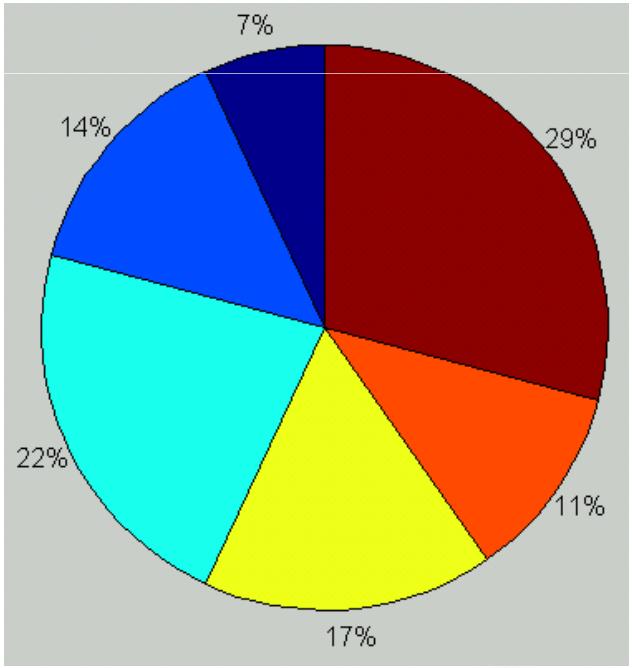
Gráficos Bidimensionais

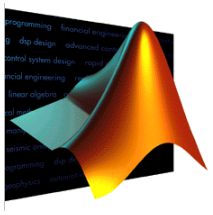
- loglog
  - Equivalente ao comando *plot*, exceto pelo fato de escalas logarítmicas serem usadas em ambos os eixos
- semilogx
  - Equivalente ao comando *plot*, exceto pelo fato de uma escala logarítmica ser usada no eixo x
- semilogy
  - O mesmo para o eixo y



Gráficos Bidimensionais

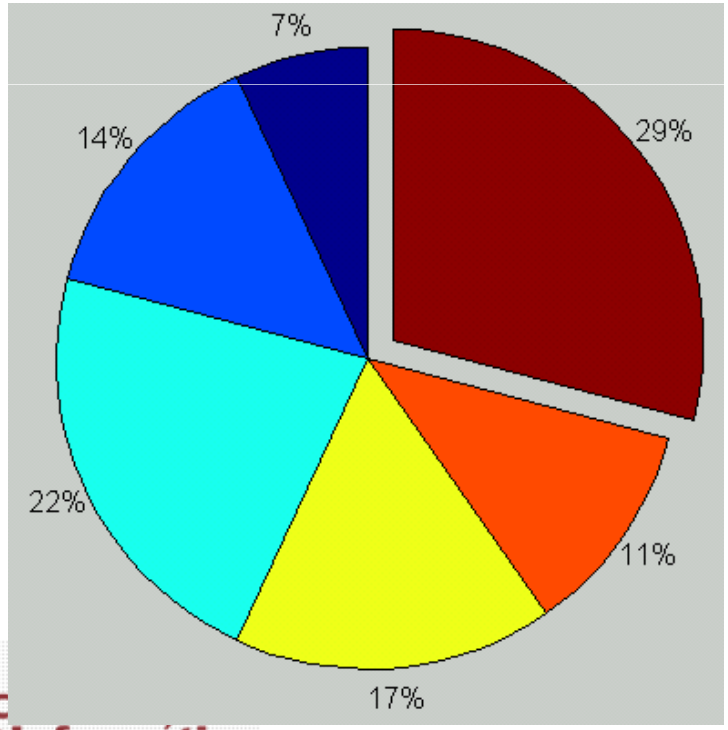
- area(x,y)
  - Equivalente a *plot(x,y)*, exceto pelo fato de que a área entre os valores 0 e y é preenchida
- pie(a,b)
  - Cria gráficos de pizza
    - **>> a=[.5 1 1.6 1.2 .8 2.1]**
    - **>> pie(a);**

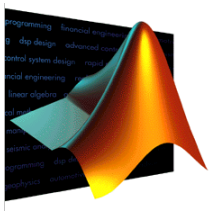




Gráficos Bidimensionais

- pie(a,b)
  - **>> a=[.5 1 1.6 1.2 .8 2.1]**
  - **>> pie(a,a==max(a)); % Traça o gráfico e separa a % maior fatia**

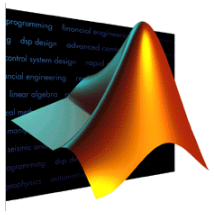




Gráficos Tridimensionais

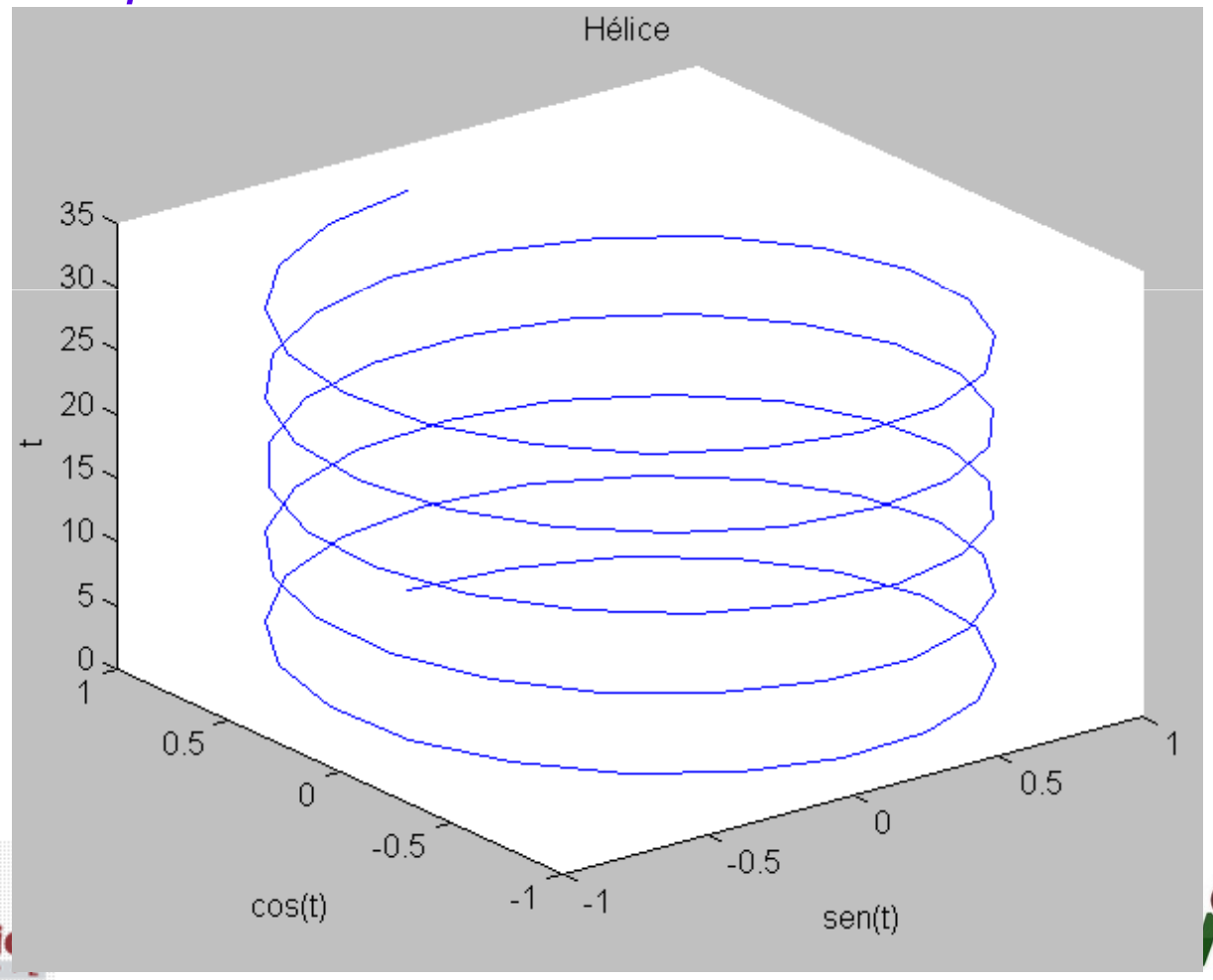
- Comando *plot3*

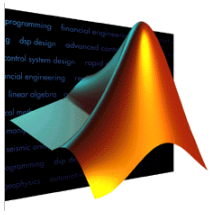
```
>> t=linspace(0,10*pi);
>> plot3(sin(t), cos(t),t);
>> title('Hélice'), xlabel('sen(t)'),ylabel('cos(t)'),zlabel('t')
```



Gráficos Tridimensionais

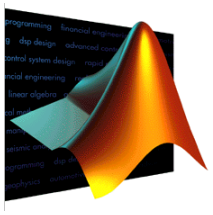
- Comando *plot3*: Gráficos de linha





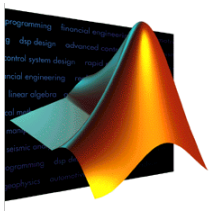
Gráficos Tridimensionais

- Gráficos de Rede e de Superfície
  - O MatLab define uma superfície de rede por meio de coordenadas *z* dos pontos correspondentes a uma grade retangular no plano *xy*
  - Ele forma o gráfico unindo os pontos adjacentes com linhas e retas
  - O resultado parece-se com uma rede de pesca com os nós nos pontos correspondentes aos dados
  - Gráficos em rede são úteis para visualização de matrizes grandes ou para a representação gráfica de funções de duas variáveis



Gráficos Tridimensionais

- Função *meshgrid*
  - [X,Y]=meshgrid(x,y)
  - Cria uma matriz X cujas linhas são cópias do vetor x e uma matriz Y cujas colunas são cópias do vetor y
  - Esse par de matrizes pode ser usado para calcular funções de duas variáveis usando os recursos de matemática vetorial do MatLab

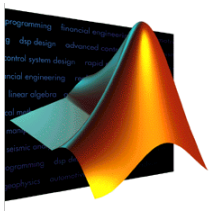


Gráficos Tridimensionais

- Função *meshgrid*
  - Exemplo: Gerar pontos de dados uniformemente espaçados no plano xy entre -7,5 e 7,5

```
>> x=-7.5:.5:7.5;
>> y=x;
>> [X,Y]=meshgrid(x,y);
```

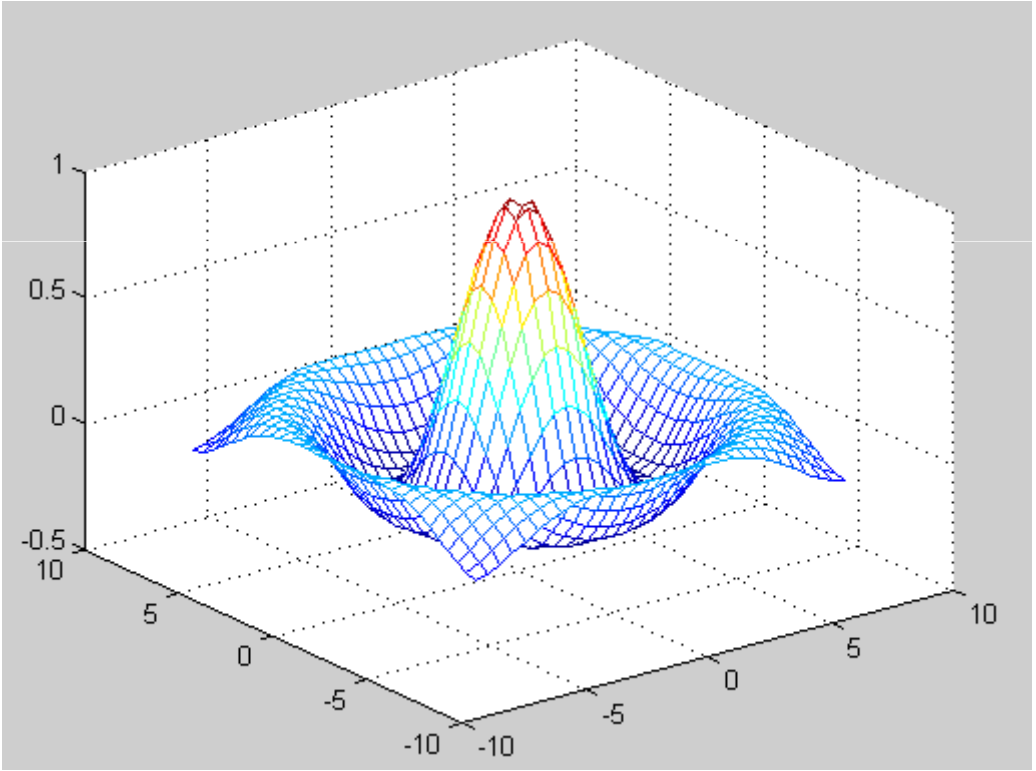
  - X e Y são um par de matrizes representando uma grade retangular de pontos no plano *xy*
  - Qualquer função *z=f(X,Y)* pode ser gerada usando-se esses pontos

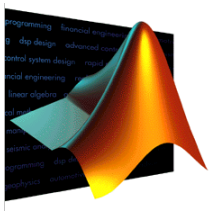


Gráficos Tridimensionais

- Função *meshgrid*
  - Exemplo:

```
>> R=sqrt(X.^2+Y.^2);
>> Z=sin(R)./R;
>> mesh(X,Y,Z)
```

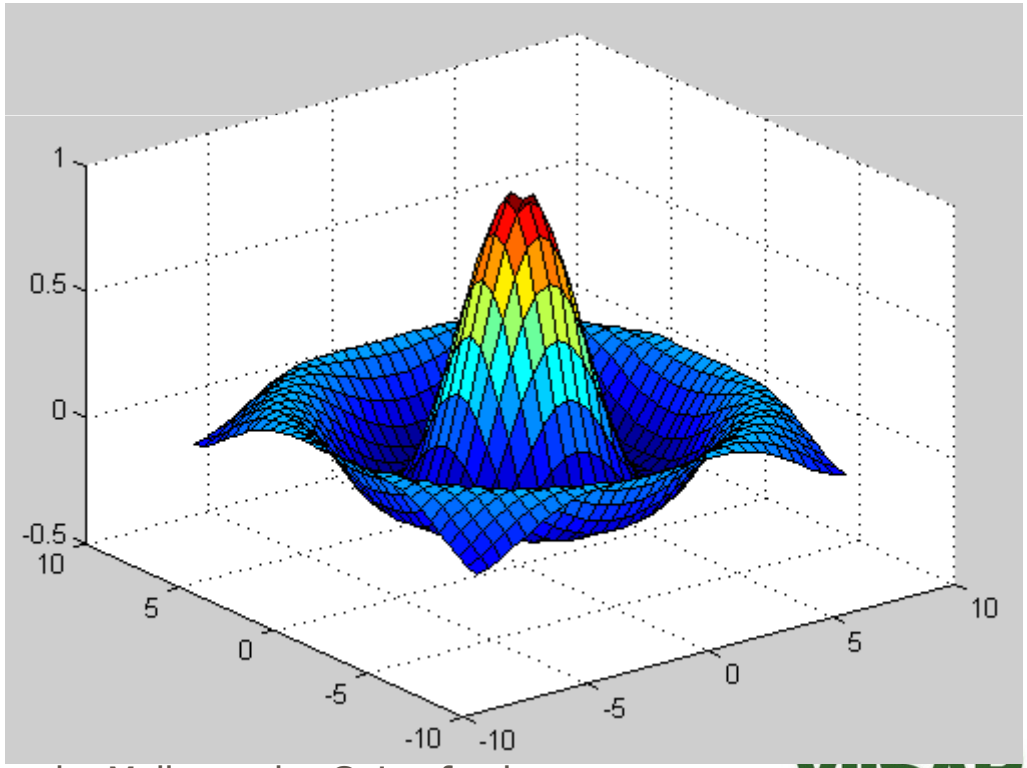


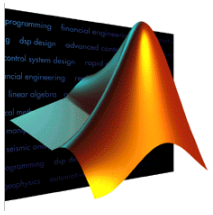


Gráficos Tridimensionais

- **Função *surf***
  - **Gera um gráfico de superfície da matriz Z, onde os espaços entre as linhas (chamados de retalhos) são preenchidos.**

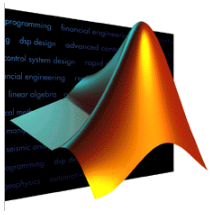
```
>> surf(X,Y,Z)
```





Gráficos Tridimensionais

- Função *peaks*
  - Gera uma matriz com valores baseados em uma distribuição Gaussiana
  - Útil para demonstrar algumas características das funções mesh, surf, pcolor, contour

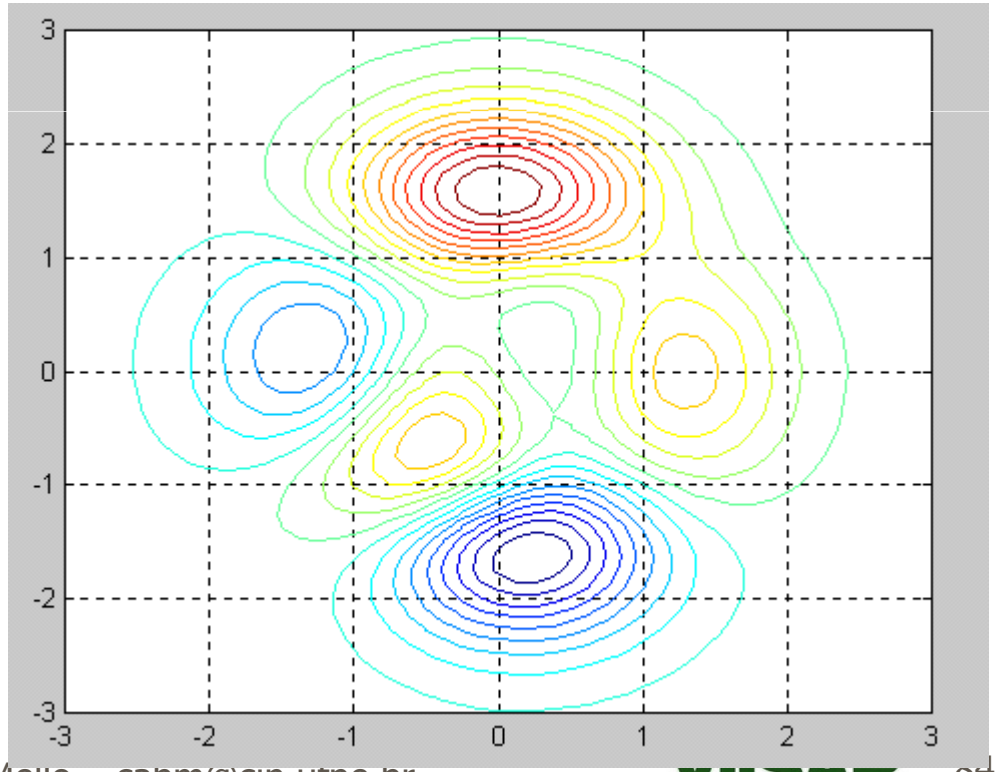


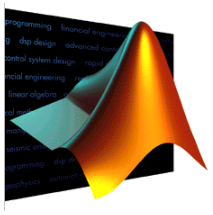
Gráficos Tridimensionais

Função *contour*

- Plota o contorno de uma matriz Z tratando os valores de Z como alturas sobre um plano

```
>> [x,y,z]=peaks;
>> contour(x,y,z,20);
```



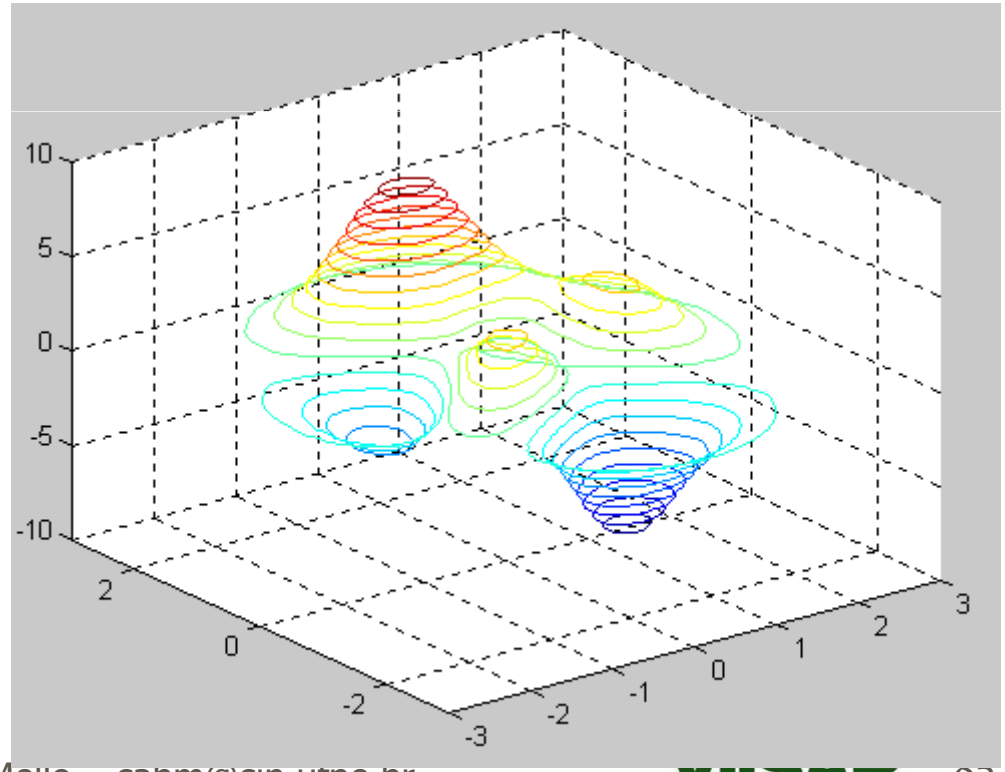


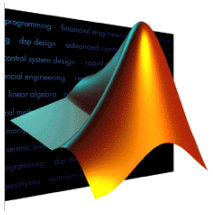
Gráficos Tridimensionais

Função *contour3*

- Plota o contorno de uma matriz Z tratando os valores de Z como alturas sobre um plano em 3 dimensões

```
>> [x,y,z]=peaks;
>> contour3(x,y,z,20);
```

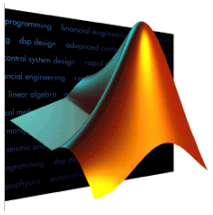




Gráficos Tridimensionais

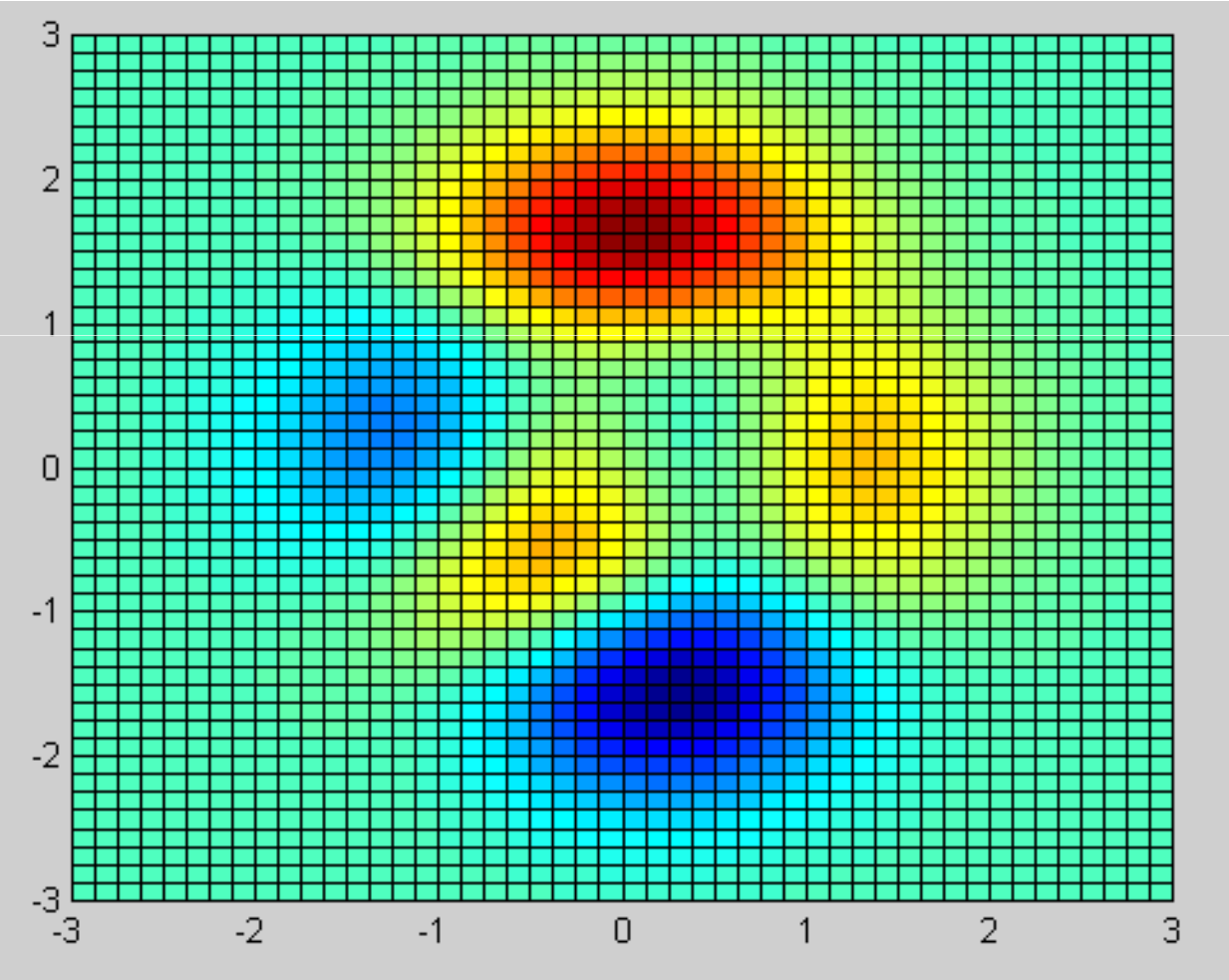
- **Função *pcolor***
  - **Mapeia a altura em um conjunto de cores e apresenta a mesma informação do gráfico de curvas de nível *contour* na mesma escala**

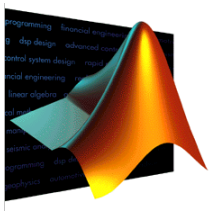
```
>> [x,y,z]=peaks;
>>pcolor(x,y,z)
```



Gráficos Tridimensionais

- Função *pcolor*

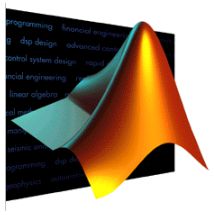




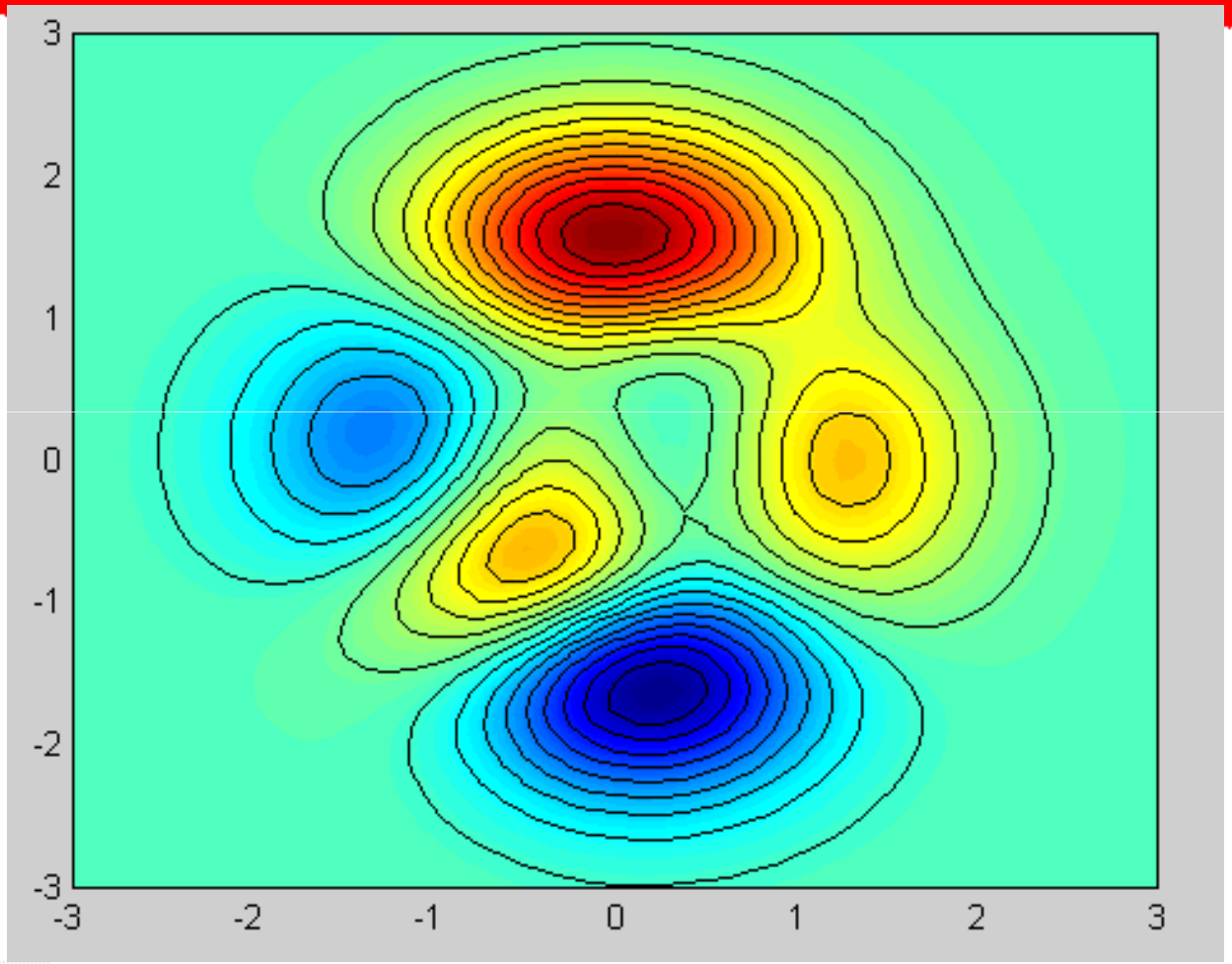
Gráficos Tridimensionais

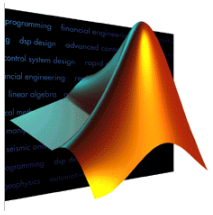
- Uma vez que *pcolor* e *contour* mostram a mesma informação na mesma escala, pode ser útil sobrepor os dois:

```
>> [x,y,z]=peaks;
>> pcolor(x,y,z)
>> shading interp   % Remove a grade de linhas
>> hold on
>> contour(x,y,z,20,'k')
>> hold off
```



Gráficos Tridimensionais

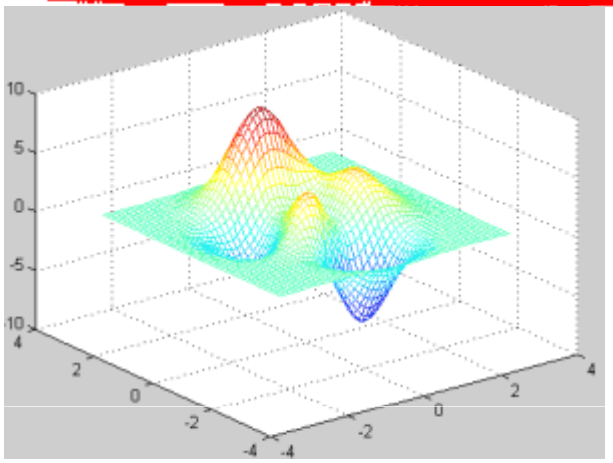




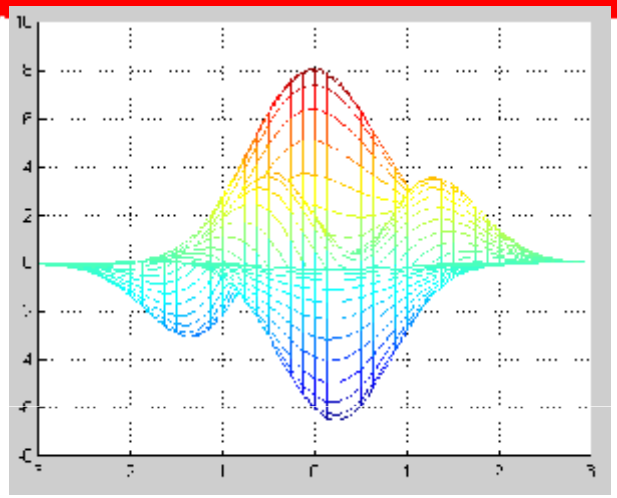
Gráficos Tridimensionais

- Manipulação de Gráficos
  - Função *view*
    - **Permite mudar o ângulo de observação de um gráfico**
    - **Uso: view(azimute, elev)**
    - **padrão: azimute=-37.5 e elev=30**

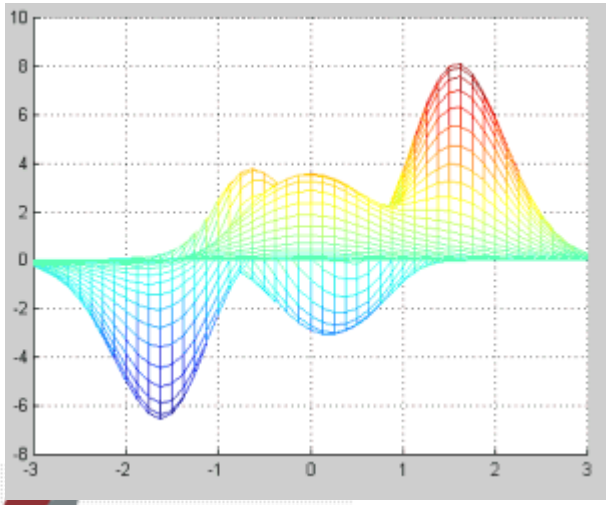
Gráficos Tridimensionais



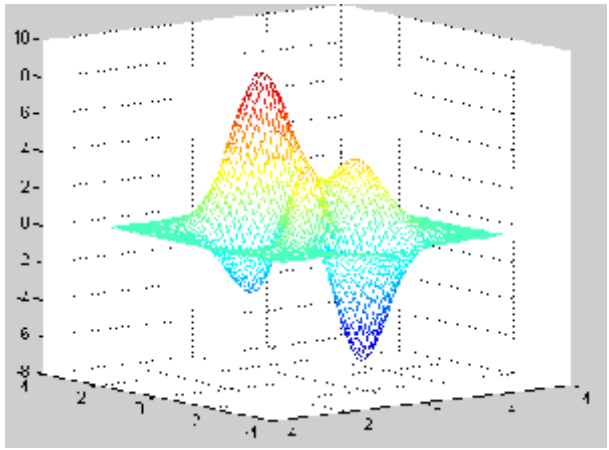
- 37.5
30



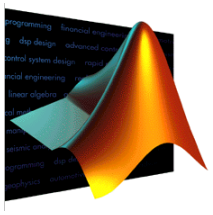
0
0



90
0



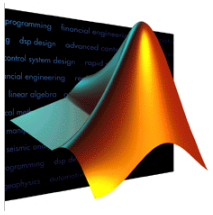
- 37.5
10



Gráficos Tridimensionais

- Exemplo de Animação:
  - Uso do comando For para gráficos

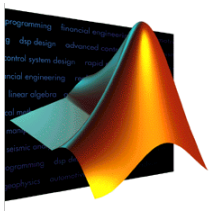
```
>> [x,y,z]=peaks;
>> mesh(x,y,z)
>> for i=0:360
        view(i,30)
        pause(0.1);
    end
```



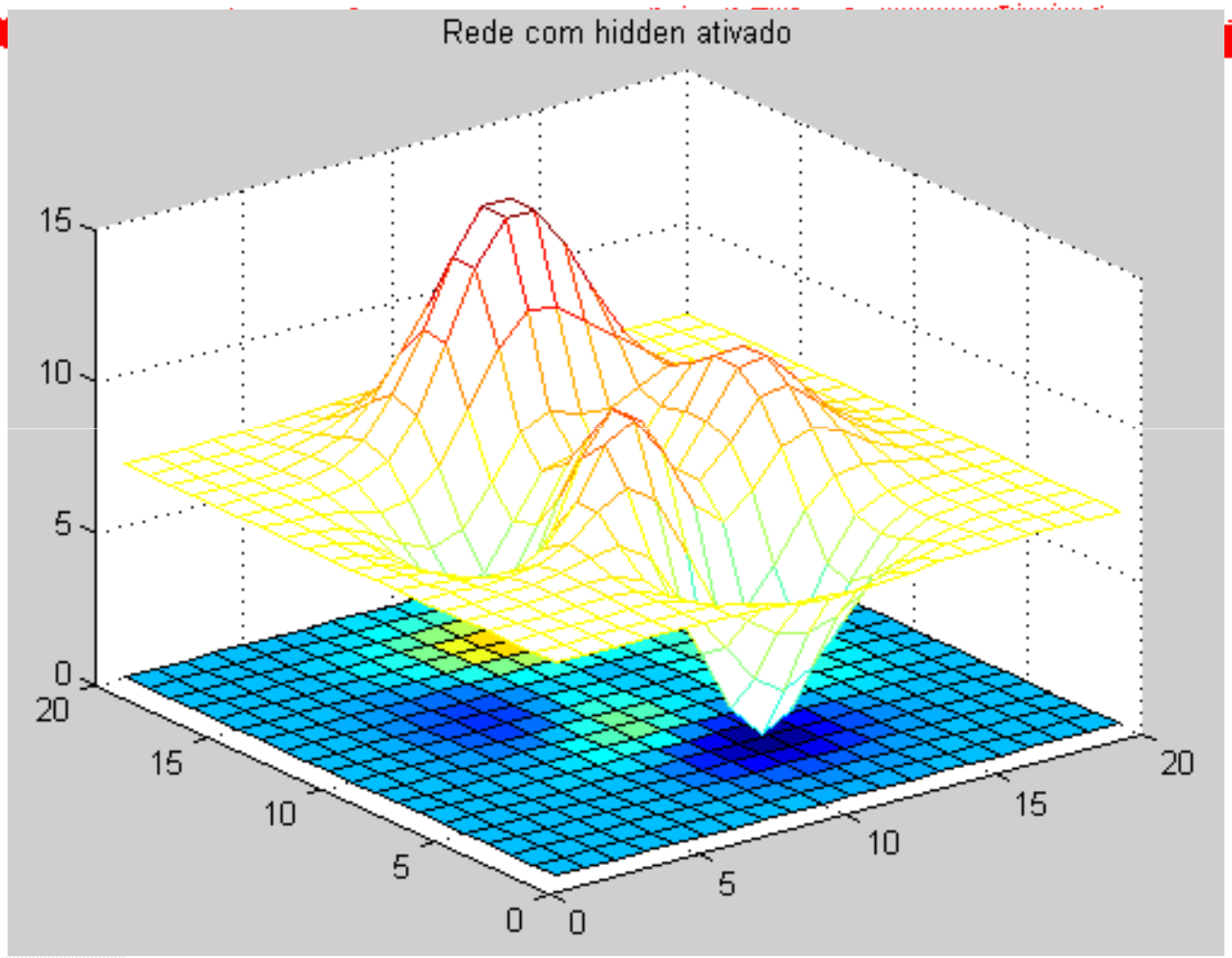
Gráficos Tridimensionais

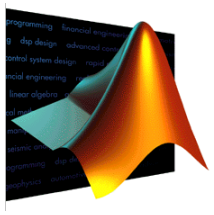
- Comando *hidden*
  - Controla a remoção de linhas escondidas
  - Desativando-se o comando *hidden* você pode olhar através da rede

```
>> mesh(peaks(20)+7)
>> hold on
>> pcolor(peaks(20))
>> hold off
>> title('Rede com hidden ativado')
```



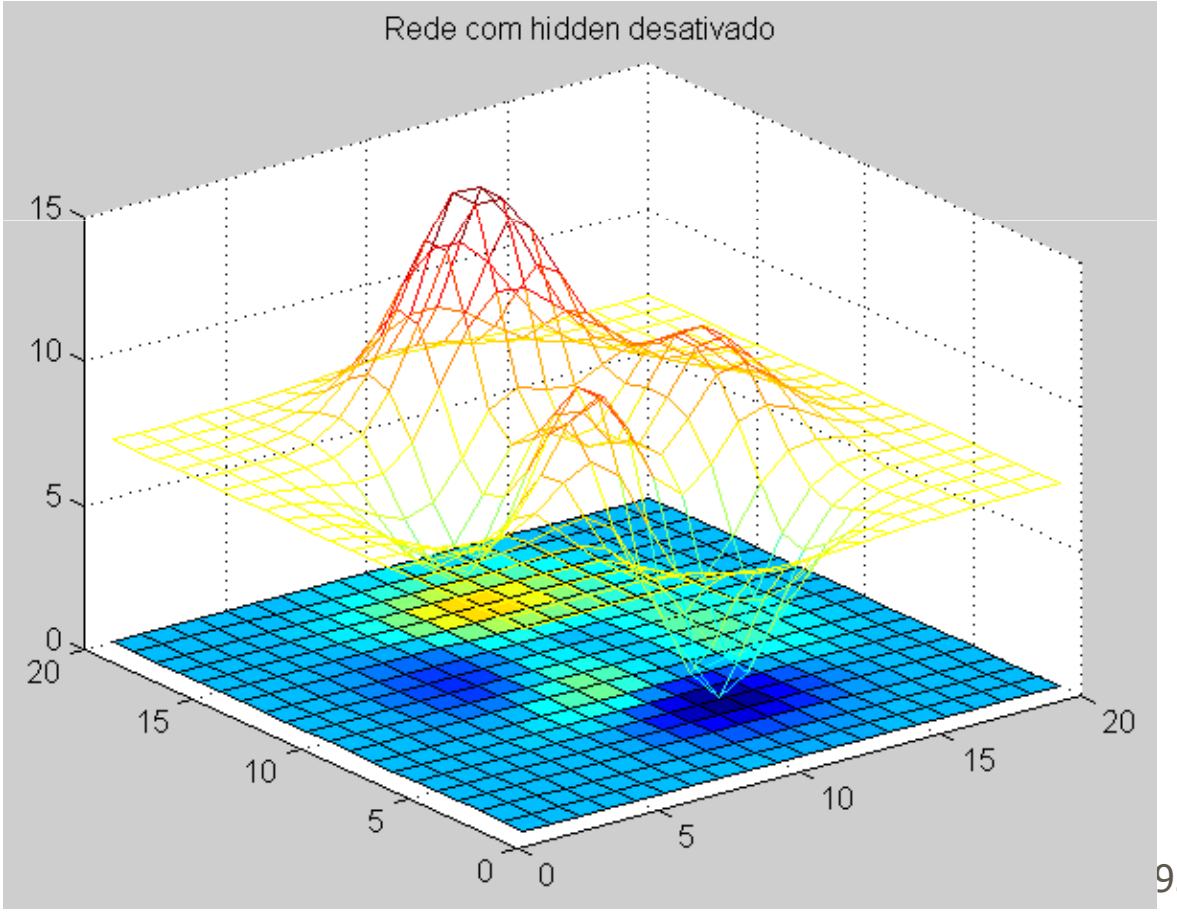
Gráficos Tridimensionais

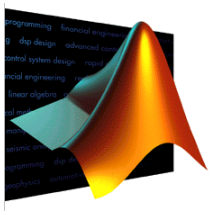




Gráficos Tridimensionais

```
>> hidden off
>> title('Rede com hidden desativado');
```

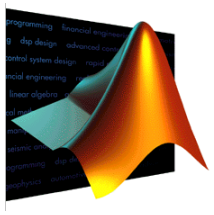




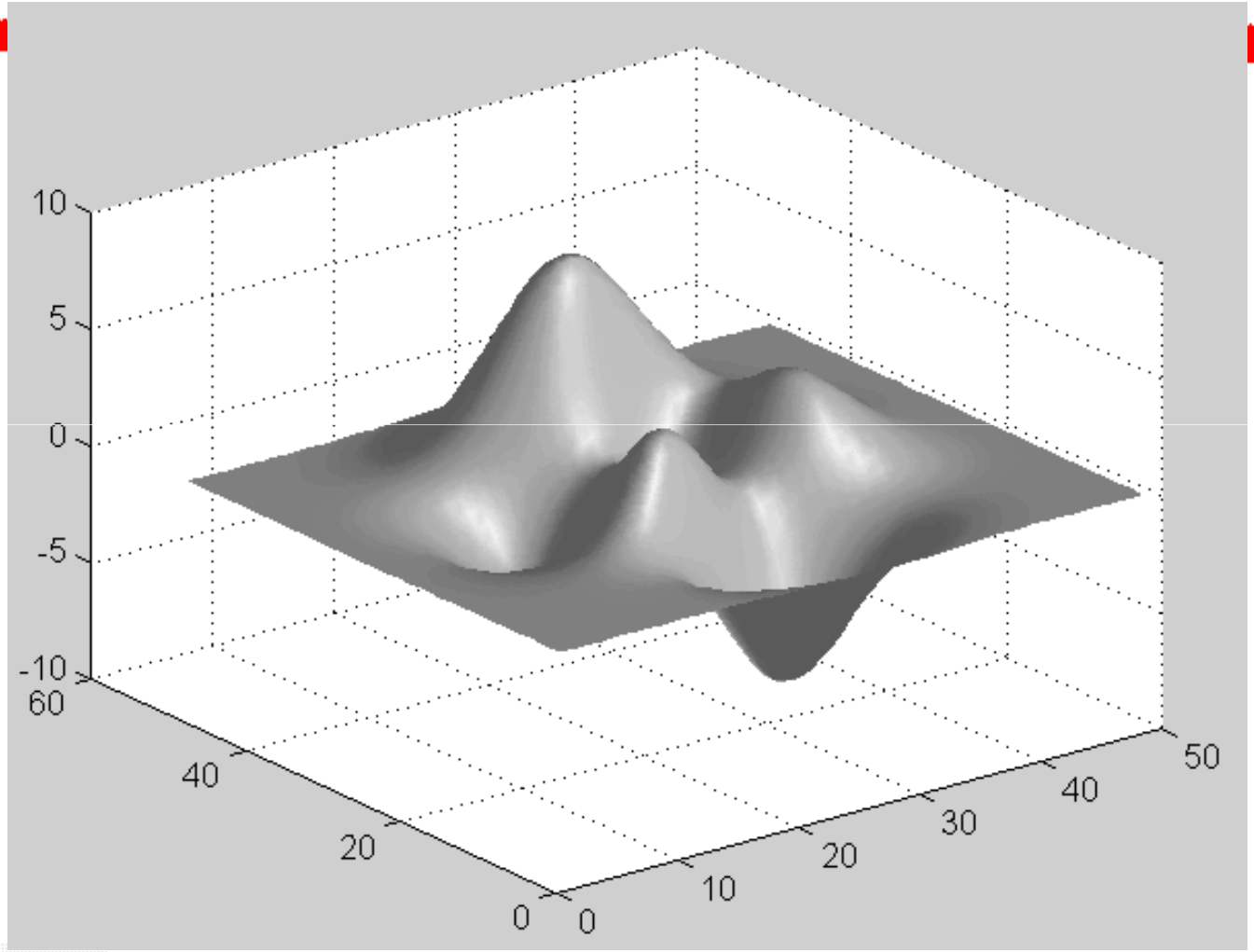
Gráficos Tridimensionais

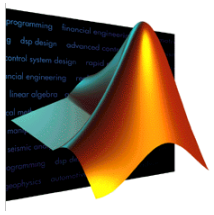
- **Função *surfl***
  - **Desenha um gráfico e acrescenta contrastes luminosos a partir de uma fonte de luz**

```
>> colormap(gray);  % Define as cores usadas
>> surfl(peaks), shading interp;
```



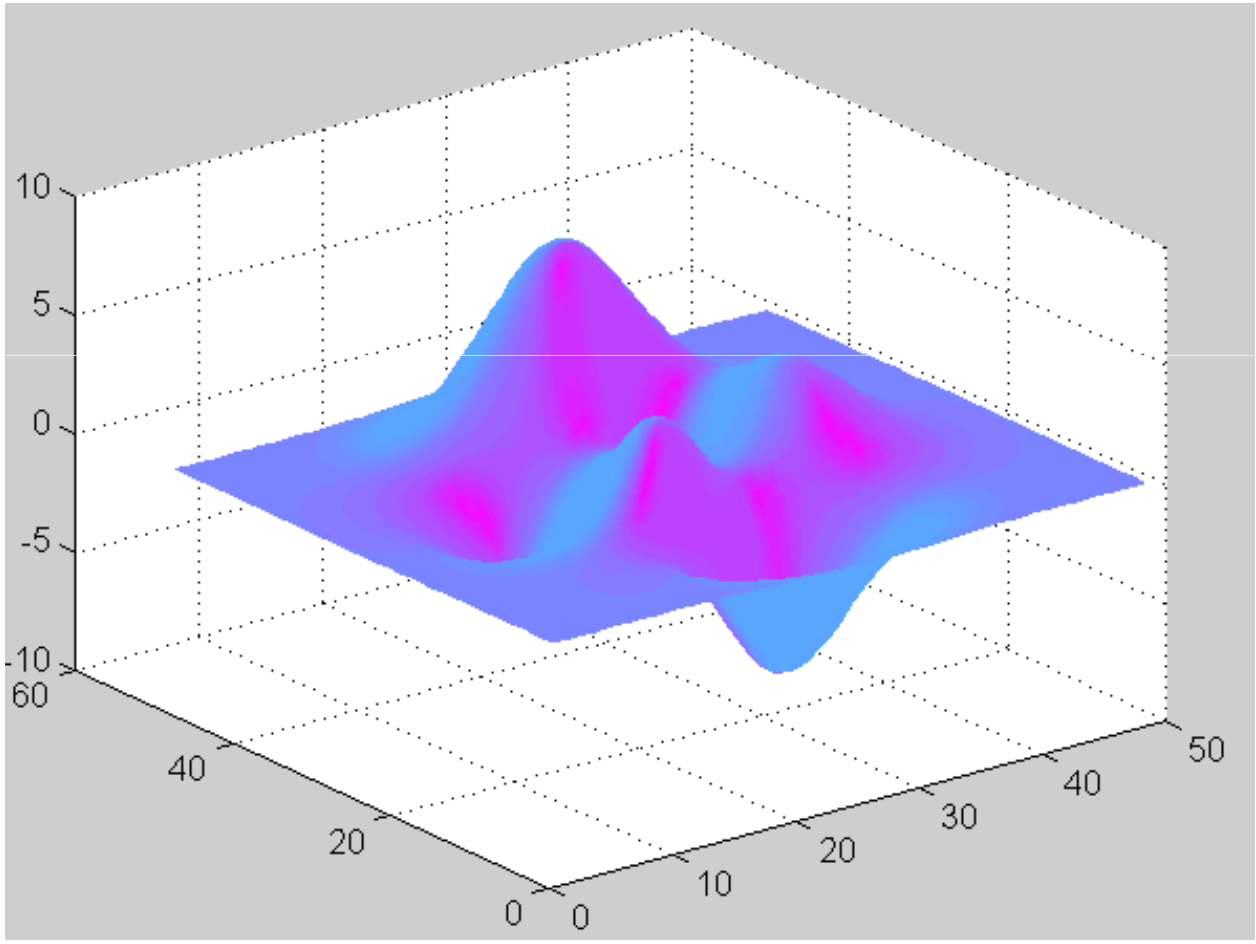
Gráficos Tridimensionais

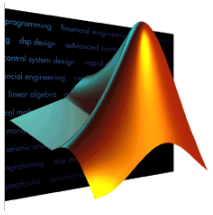




Gráficos Tridimensionais

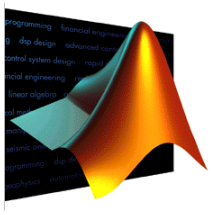
>>colormap(cool)





Gráficos Tridimensionais

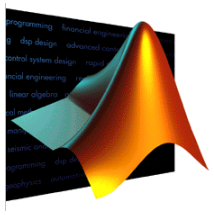
- colormap:
  - **hsv, hot, gray, bone, copper, pink, white, flag, jet, prism, cool, lines, colorcube, summer, autumn, winter, spring**



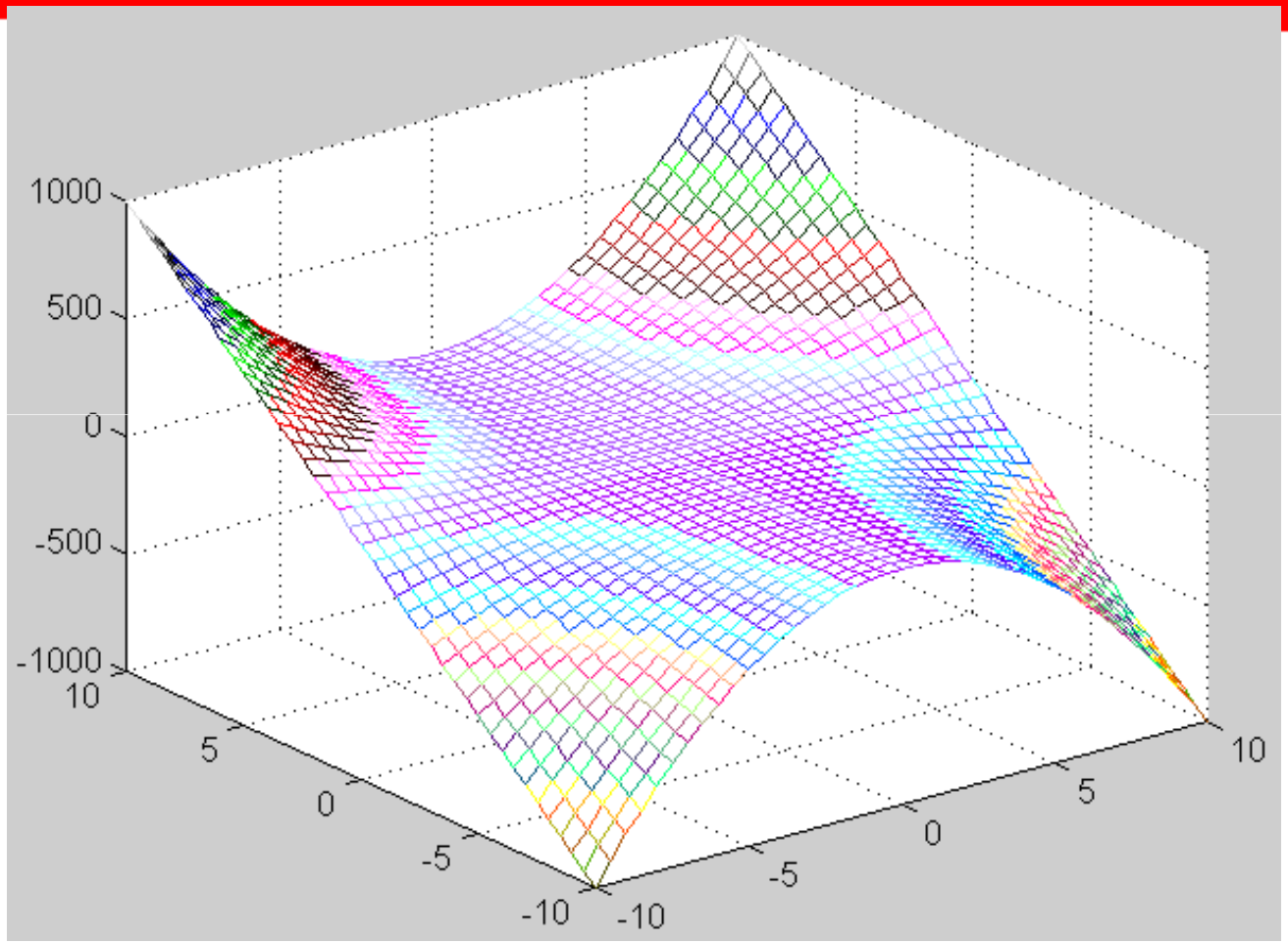
Gráficos Tridimensionais

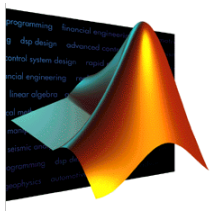
- Gráficos Tridimensionais de Funções

```
>> a=-10:.5:10;
>> b=a;
>> [X,Y]=meshgrid(a,b);
>> Z= (X.^2).*Y;
>> mesh(X,Y,Z)
```



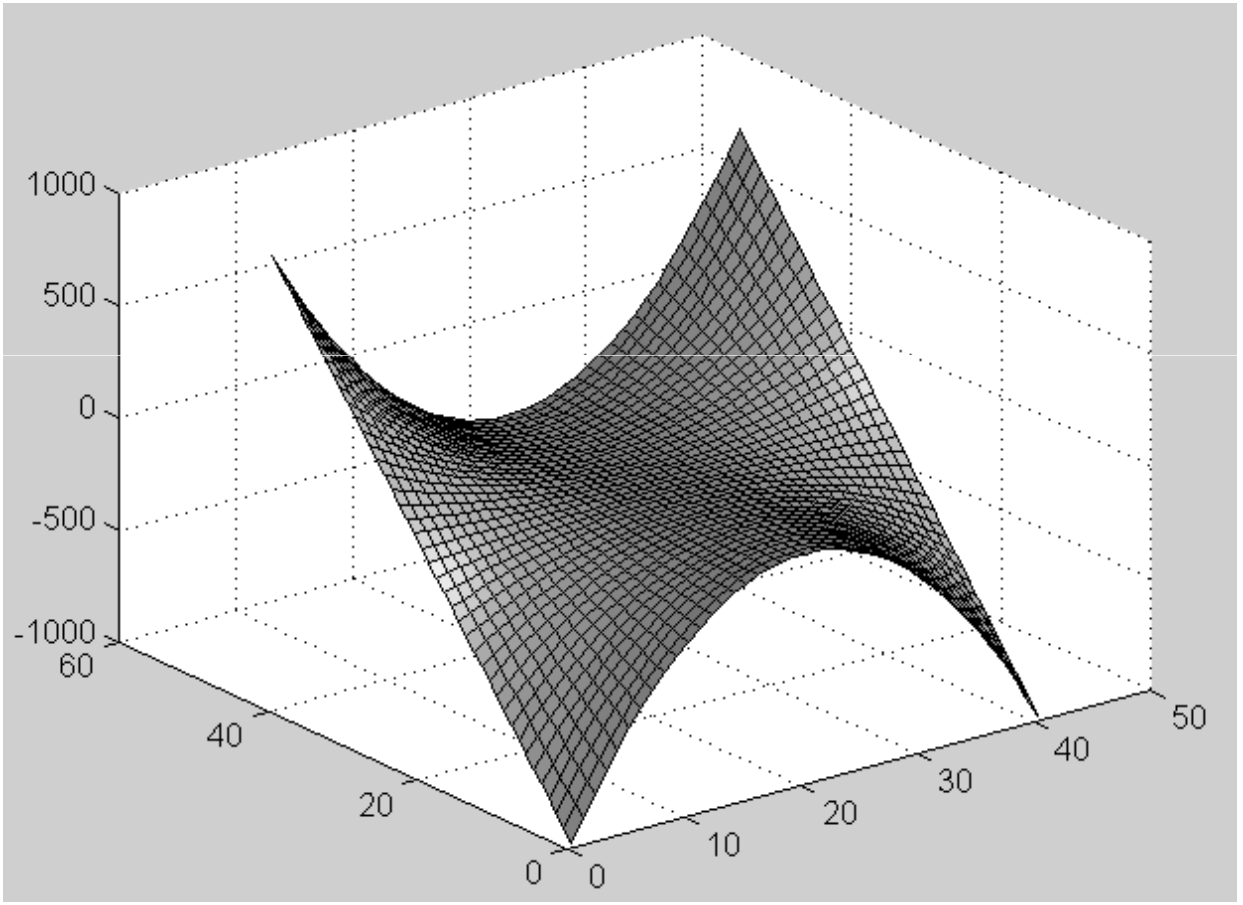
Gráficos Tridimensionais

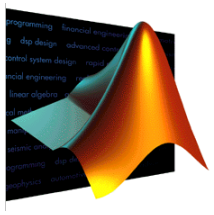




Gráficos Tridimensionais

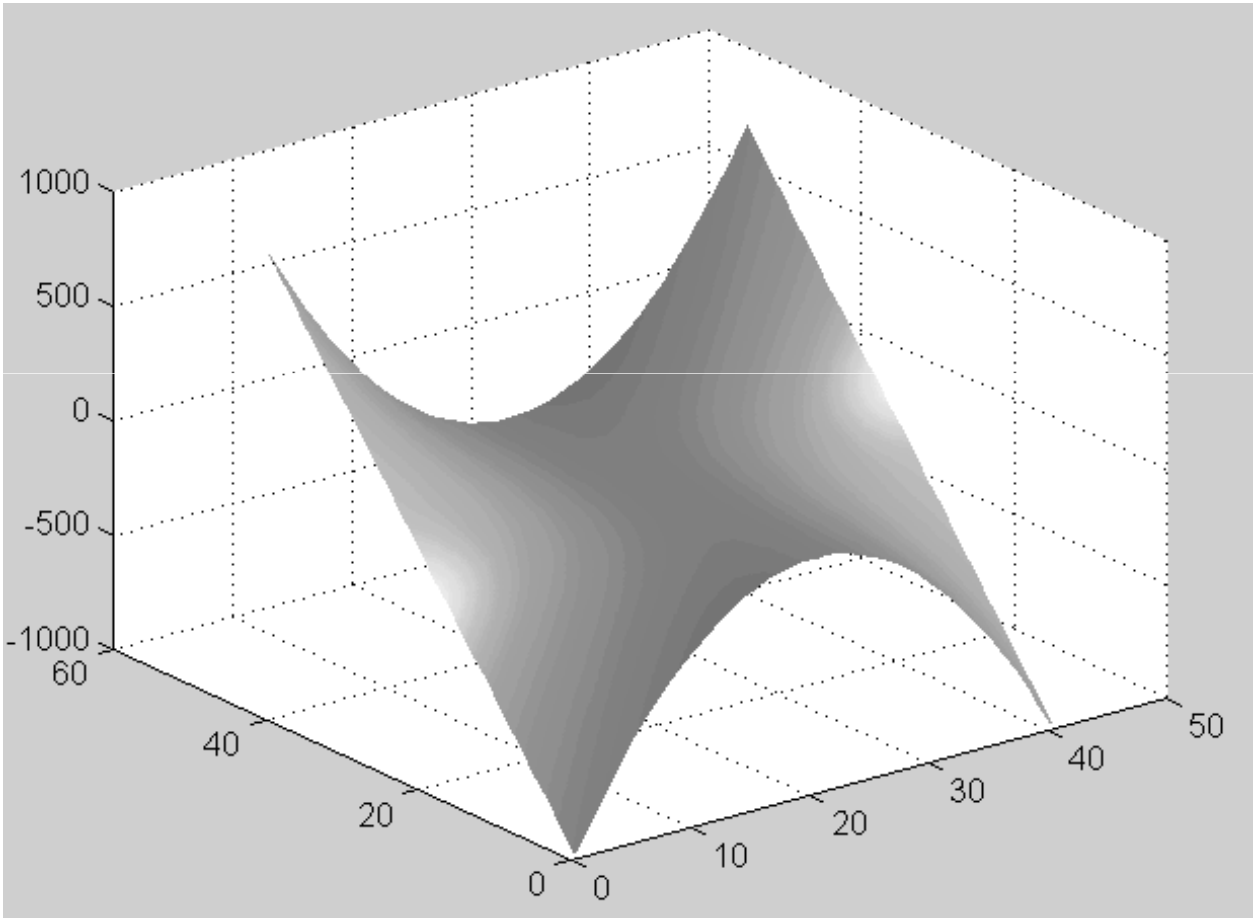
```
>> colormap(gray);
>> surfl(Z)
```



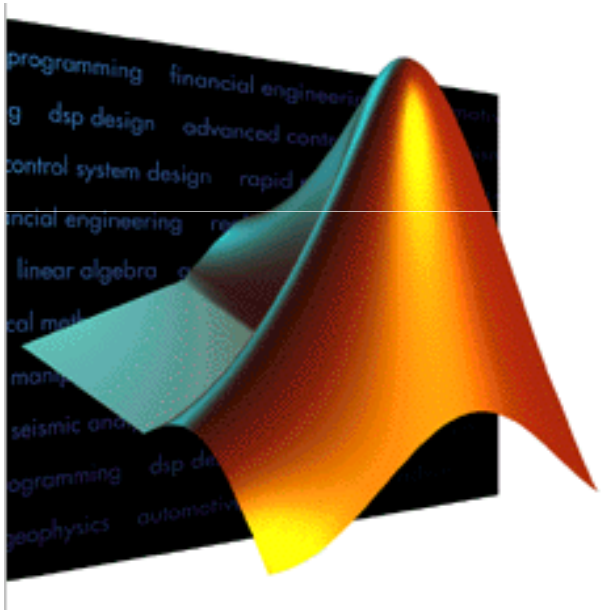


Gráficos Tridimensionais

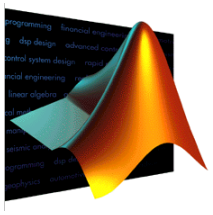
```
>> colormap(gray);
>> surfl(Z);
>> shading interp;
```



Programação no MatLab
Arquivos-M de Comandos

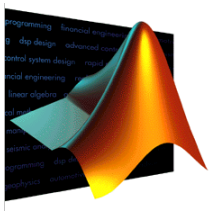


Carlos Alexandre Mello



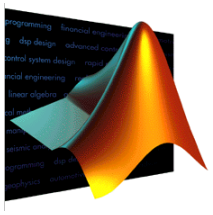
MATLAB
Programação

- No MatLab, os programas são escritos em arquivos de texto denominados *Arquivos-M*
  - Arquivos-M: arquivos que contêm código do MatLab. Podem ser:
    - Funções
    - Scripts (Arquivos-M de comandos)
- Como são arquivos de texto podem ser criados usando editores de texto comum



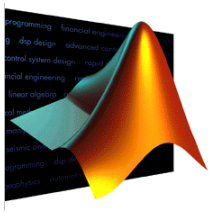
MATLAB
Programação

- Os Arquivos-M são *interpretados* pelo MatLab para a execução de seus comandos
- Assim, para serem executados, os arquivos-M necessitam do MatLab não podendo operar isoladamente
- Diferente de outras linguagens como, C ou Pascal, onde seus programas são *compilados*



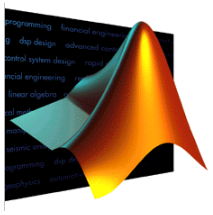
MATLAB
Programação

- Interpretação
  - Vantagem
    - Código aberto
    - Fácil de ser estudado
  - Desvantagem
    - Necessidade do *"Programa-Pai"*



MATLAB
Programação

- Arquivos-M
  - Para problemas simples, é mais fácil introduzir os comandos no prompt do MatLab
  - No entanto, se o número de comandos é grande, ou se você deseja mudar o valor de uma ou mais variáveis e re-executar alguns comandos, pode tornar-se tedioso introduzir os comandos no prompt
  - Solução:
    - Introduzir os comandos em um arquivo texto chamado Arquivo-M



MATLAB
Programação

Arquivos-M de Comando:

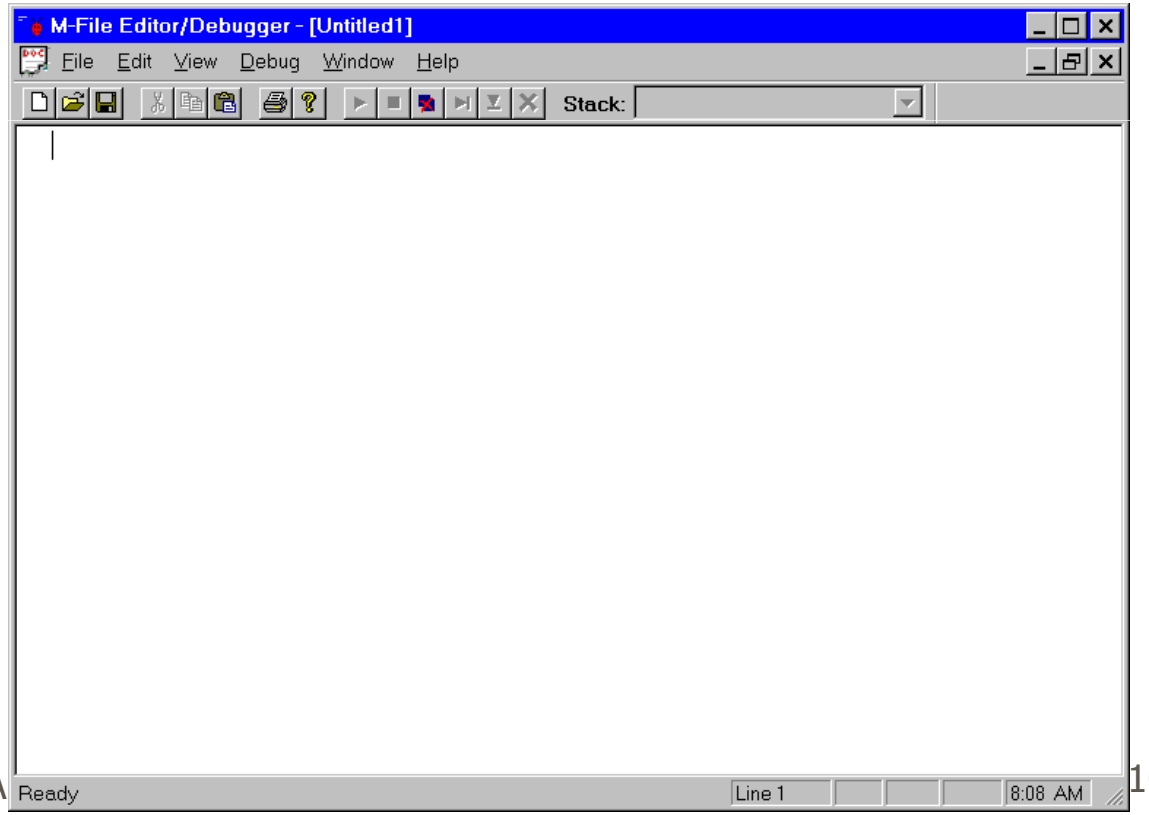
- Não aceitam argumentos de entrada
- Útil para automatizar passos que sejam executados muitas vezes

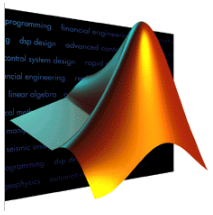
Funções:

- Aceitam argumentos de entrada
- Útil para expandir o MATLAB para suas aplicações

MATLAB
Criação de um Arquivo-M
Programação

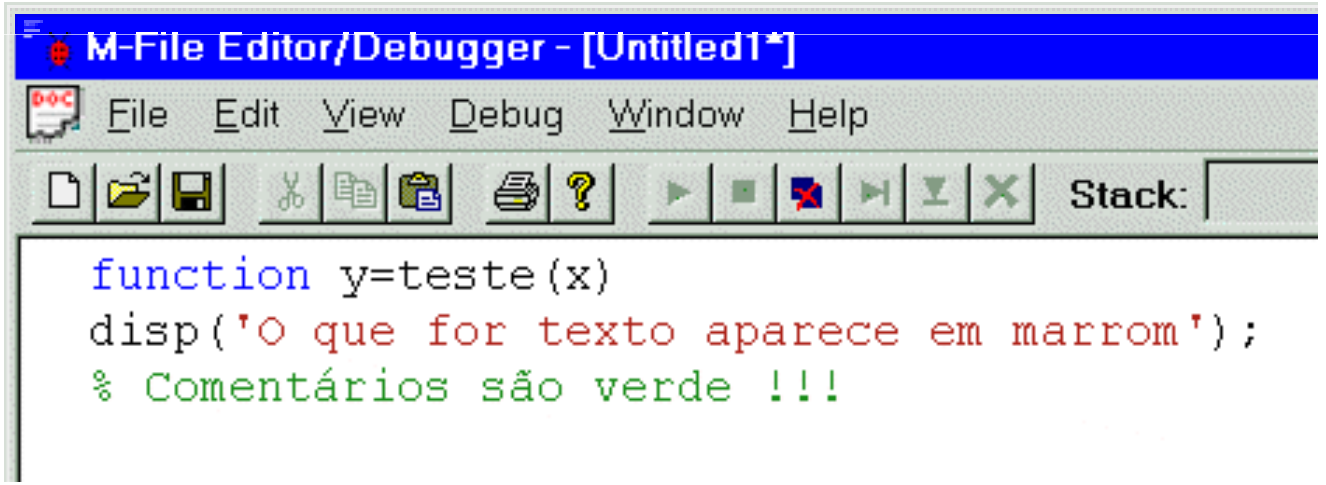
- Qualquer editor de textos
- Através do próprio MatLab:
  - comando *edit*
    - **edit *filename***
    - ***edit***

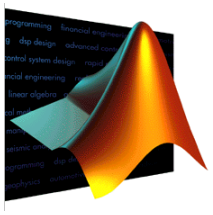




MATLAB
Criação de um Arquivo-M
Programação

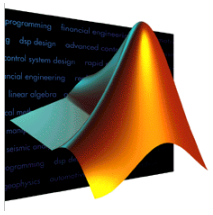
- Vantagens do uso do editor do MatLab
  - **Variação nas cores**
  - **Possibilidade de uso do *Debbuger***





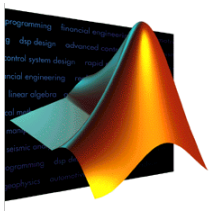
MATLAB
Arquivos-M de Comandos
Programação

- Mais simples forma de arquivo-M
- Não aceitam argumentos de entrada ou saída
- Operam com dados do Workspace
- Quaisquer variáveis criadas pelo script permanecem no workspace e podem ser usadas quando o script termina



MATLAB
Criação de Arquivos-M de Comandos
Programação

- Edita um arquivo tipo M (editor de texto qualquer)
- Insere comandos e variáveis
- Executa a partir do MatLab, bastando, para isso, digitar o nome do arquivo na linha de comando

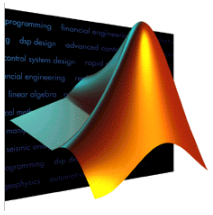


MATLAB
Criação de Arquivos-M de Comandos
Programação

- Exemplo de Script:

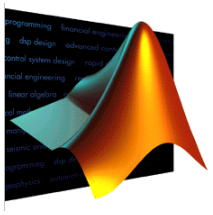
```
% Exemplo de um script
% Soma de dois números
a = input ('Entre com o primeiro número:  ');
b = input ('Entre com o segundo número:  ');
soma = a + b
```

Saída do Script



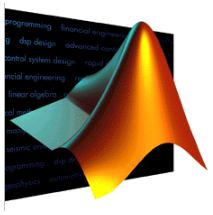
MATLAB
Programação

- Em virtude da grande utilidade dos arquivos de comandos e funções, o MatLab possui diversas funções que são particularmente apropriadas para o uso em arquivos-M
- Essas funções também podem ser utilizadas no prompt do MatLab
- Veremos a seguir algumas dessas



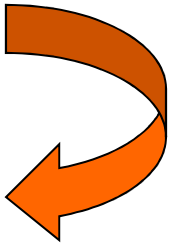
MATLAB
Programação

- disp(x)
  - Apresenta algum resultado na tela do MatLab
  - Exemplo 1:
    - >> x= 2;
    - >> disp(x);
    - 2
  - Exemplo 2:
    - >> disp('x');
    - x

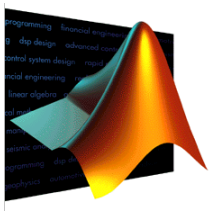


MATLAB
Programação

- disp(x)
  - Exemplo 3:
    - >> disp('Este é o MatLab');
    - Este é o MatLab
  - Exemplo 4:
    - >> x = 2; y = 4;
    - >> disp('x+y');
    - x+y
    - >> disp(x+y)
    - 6

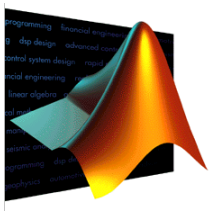


Observe a diferença entre as duas formas no uso do apóstrofo !!!!!!



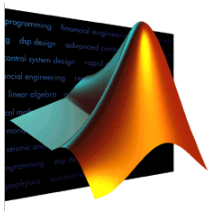
MATLAB
Programação

- input
  - Solicita ao usuário algum dado de entrada
  - Exemplo 1:
    - >> x = input('Qual o valor de X: ')
    - Qual o valor de X: 2
    - x = 2
      - O valor 2 será atribuído à variável x



MATLAB
Programação

- pause
  - Suspende a execução de um programa até que alguma tecla seja pressionada
- pause(n)
  - Suspende a execução por n segundos



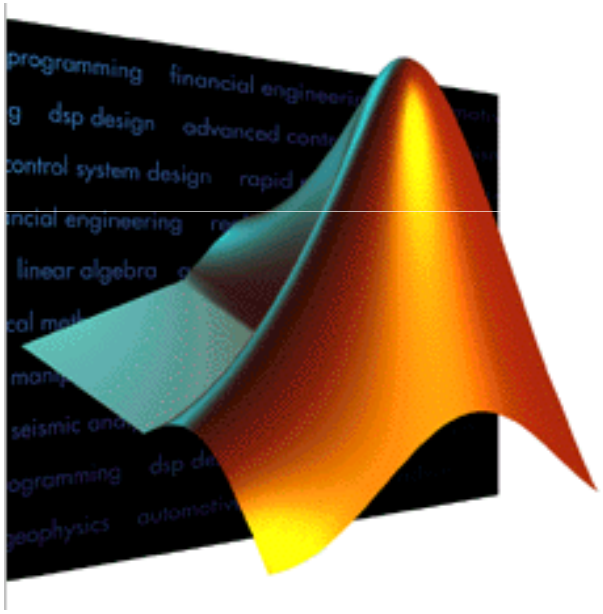
MATLAB
Programação

- Exemplo de Script:
  - Faça um Script que calcule as raízes de um polinômio do segundo grau

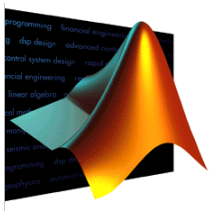
```
a=input('Entre com o primeiro coeficiente :');
b=input('Entre com o segundo coeficiente :');
c=input('Entre com o terceiro coeficiente :');
delta = b^2 - 4*a*c;
x1=(-b + sqrt(delta))/(2*a);
x2=(-b - sqrt(delta))/(2*a);
```

  - e salve como baskhara.m no diretorio atual
  - Para executá-lo, basta digitar baskhara na workspace do MatLab

Operações Relacionais e Lógicas

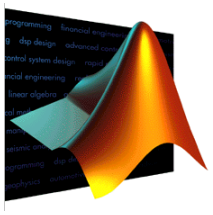


Carlos Alexandre Mello



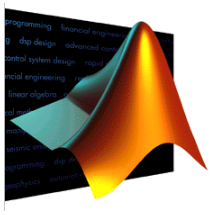
MATLAB
Operações Relacionais e Lógicas

- Operadores Relacionais
  - o < Menor que
  - o > Maior que
  - o <= Menor ou igual a
  - o >= Maior ou igual a
  - o = Igual a
  - o == Igual a?
  - o ~= Diferente de



MATLAB
Operações Relacionais e Lógicas

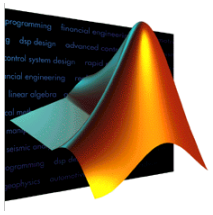
- Diferença entre = e ==
  - = : Afirmação (atribuição de um valor)
    - A = 2
      - A é igual a 2
      - É feita uma afirmação
      - Não há resposta do sistema
  - == : Questionamento (dúvida quanto a um valor)
    - A==2
      - A é igual a 2 ???
      - É feita uma pergunta
      - O sistema responde com Verdadeiro ou Falso (1 ou 0)



MATLAB
Operações Relacionais e Lógicas

- Exemplo:

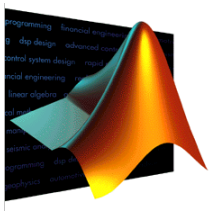
```
» a=10;
» a > 2

ans =

     1

» a < 3

ans =

     0

» a==10

ans =

     1

» a~=20

ans =

     1

» a=2

a =

     2

» a

a =

     2
```



MATLAB
Operadores Lógicos

- Combinam ou negam operações relacionais
  - & E
  - | OU
  - ~ Não

```
» a=1:9

a =

     1     2     3     4     5     6     7     8     9

» b=(a>2)&(a<6)

b =

     0     0     1     1     1     0     0     0     0
```



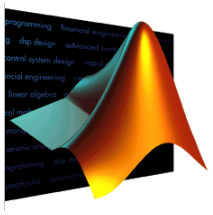
MATLAB
Funções de Verificação

- ischar(S)
  - V se a entrada é um vetor de caracteres

```
» a=['carlos' 1];
» ischar(a)

ans =

     1
```

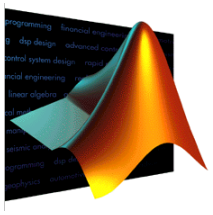
  - Cuidado!!!

```
» a=['carlos' 1];
» ischar(a)

ans =

     1
```



MATLAB
Controle de Fluxo
Programação

- for
- if...elseif...else
- while
- switch....case

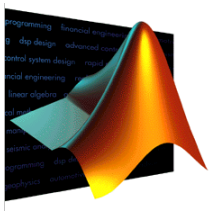


MATLAB
Controle de Fluxo
Programação

- Laços for
  - Possibilitam que uma série de comandos seja repetida por um número de vezes fixo e pré-definido
- Uso:

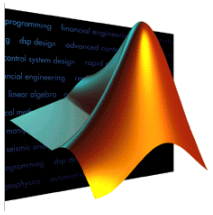
```
for índice = começo:incremento:fim
   comandos
end
```

- Se o incremento não for escrito, o MatLab considera 1



MATLAB
Controle de Fluxo
Programação

- Laços for
  - Para criar um laço decremental deve-se colocar o incremento igual a -1
    - for i = 5:-1:1
  - Sem esse incremento, o laço não funcionará !!
    - for i=5:1

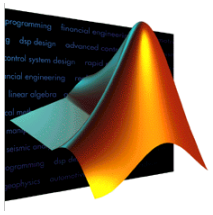


MATLAB
Controle de Fluxo
Programação

- Laços for
  - Exemplo:

```
for i = 1:5
   disp(i)
end
```

  - Resultado:
    - 1 2 3 4 5

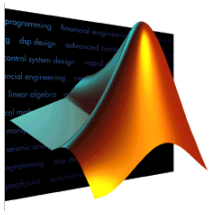


MATLAB
Controle de Fluxo
Programação

- Laços for
  - Exemplo: Criação de um vetor

i varia de 1 a 5 no índice do vetor a e nos seus valores

```
» for i=1:5
a(i)=i + 3;
end
» a

a =

     4     5     6     7     8
```

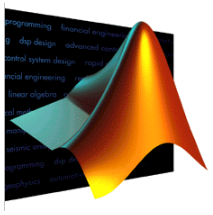


MATLAB
Controle de Fluxo
Programação

- Laços for
  - Exemplo: Laço em ordem decrescente

i varia de 5 a 1 →

```
» for i=5:-1:1
disp(i)
end
     5

     4

     3

     2

     1
```

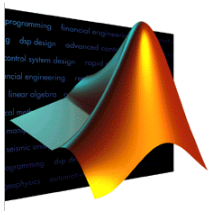


MATLAB
Controle de Fluxo
Programação

- Laços for
  - Exemplo: Variação no passo

i varia de 1 a 10 de 2 em 2

```
» for i=1:2:10
disp(i);
end
     1

     3

     5

     7

     9
```

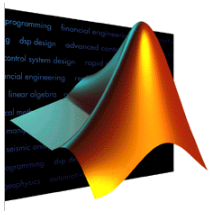


MATLAB
Controle de Fluxo
Programação

- Laços for
  - Exemplo: Erro

Não foi definido o passo !!!

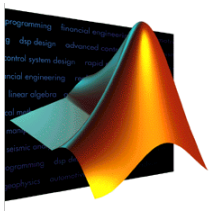
```
» for i=5:1
disp(i)
end
»
```



MATLAB
Controle de Fluxo
Programação

- Dentro de um laço for podem vir quaisquer comandos do MatLab, inclusive outros laços

```
for i=1:10
   for j=1:5
            .....
   end
end
```



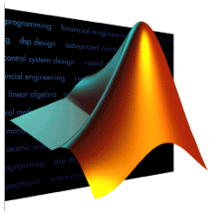
MATLAB
Controle de Fluxo
Programação

- Cuidados com os laços encadeados !!!
  - Observe a diferença no 2 programas a seguir:

```
%Programa1
for externo=1:4
  for interno=1:5
    disp(interno*externo);
  end
end
```

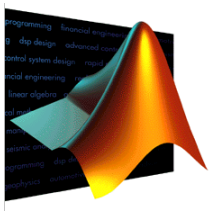
```
%Programa2
for externo=1:4
end
  for interno=1:5
    disp(interno*externo);
  end
```

  - Qual o resultado da execução dos dois programas no MatLab ??



MATLAB
Controle de Fluxo
Programação

- Cálculo da soma de um conjunto de N elementos
  - Seja A um conjunto com N valores positivos ou negativos
  - 1) soma = 0
  - 2) Para todos os elementos de A, calcule soma = soma + elemento_de_A
- Exemplo: Seja A = [1 3 2 4]
  - 1) soma = 0;
  - 2) soma = 0 + 1 = 1;
  - 3) soma = 1 + 3 = 4;
  - 4) soma = 4 + 2 = 6;
  - 5) soma = 6 + 4 = 10;
  - Como implementar esse algoritmo no MatLab ??

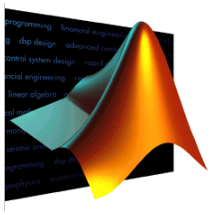


MATLAB
Controle de Fluxo
Programação

```
%Soma dos elementos de um vetor
%.....
.......
valor = 0;
for i=1:length(a)
  valor = valor + a(i);
end
disp(valor);
```

Pseudo-código não-implementável !!! Ainda não definimos como entrar com a variável a

Variável inicializada com 0. Poderia ser qualquer valor.



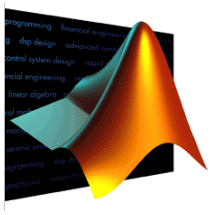
MATLAB
Controle de Fluxo
Programação

- Vetorização de Laços
  - **Exemplo:**

```
i = 0;
for t = 0:0.01:10
        i = i+1;
        y(i) = sin(t);
end
```

Versão Vetorizada do código acima

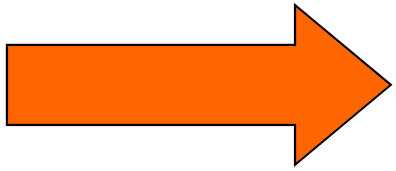
```
t = 0:0.01:10;
y = sin(t);
```



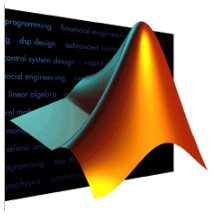
MATLAB
Controle de Fluxo
Programação

- if-else
  - Execução de uma seqüência de comandos condicionalmente com base em um teste relacional
- Uso:

```
if condição
   comandos
else
   comandos
end
```

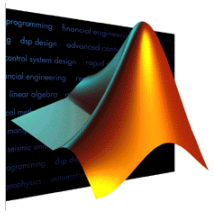


```
Se condição
        comandos
Senão
        comandos
fim
```



MATLAB
Controle de Fluxo
Programação

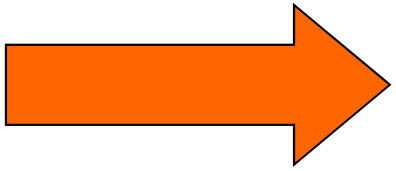
- if-else
  - Observações:
    - Não há obrigação do uso do *else*
    - Os comandos após o *if* serão executados se a *condição* for verdadeira
    - Se a *condição* envolve muitas subexpressões lógicas, todas são executadas mesmo que uma das primeiras subexpressões já seja suficiente para determinar o valor lógico da *condição*
      - **Uso de operadores lógicos**



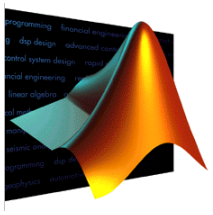
MATLAB
Controle de Fluxo
Programação

- if-else-elseif

```
if condição1
   comandos
elseif condição2
   comandos
else
   comandos
end
```



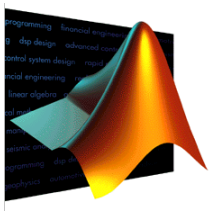
```
Se condição1
        comandos
Senão Se condição2
        comandos
Senão
        comandos
fim
```



MATLAB
Controle de Fluxo
Programação

- Exemplo: Resolução de Equações do segundo grau

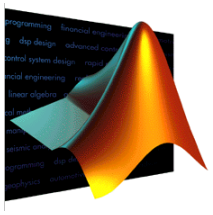
```
% Resolucao de Equacoes do Segundo Grau
a = input('Entre com o primeiro coeficiente: ');
b = input('Entre com o segundo coeficiente: ');
c = input('Entre com o terceiro coeficiente: ');
if (a == 0)
  disp('Polinomio do Primeiro Grau')
  disp('Solucao: ');
  x = -b/c
else
  delta = b^2 - 4*a*c;
  disp('Polinomio do Segundo Grau');
  disp('Solucao: ');
  x1 = (-b + sqrt(delta))/2*a
  x2 = (-b - sqrt(delta))/2*a
end
```



MATLAB
Controle de Fluxo
Programação

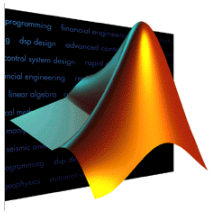
- Exemplo: Jogo da Adivinhação

```
%Jogo de adivinhacao
disp('Adivinhe qual o numero (0-10)');
x=round(rand(1)*10);
num=input('Qual o numero: ');
if (ischar(num))
  disp('Eh preciso digitar um numero !!');
else
  if (num==x)
    disp('Voce acertou!!');
  else
    disp('Voce errou !!');
    disp ('O numero eh:');
    disp(x);
  end
end
```



MATLAB
Controle de Fluxo
Programação

- Laço while
  - Ao contrário do laço for que executa um grupo de comandos um número fixo de vezes, o *while* executa um grupo de instruções um número indefinido de vezes



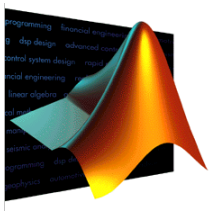
MATLAB
Controle de Fluxo
Programação

- While
  - Uso:

```
while expressão
        comandos
end
```

Enquanto *expressão*
comandos
fim

- Enquanto a expressão for verdadeira, os comandos serão executados
- Pode-se usar o comando *break* para sair do laço



MATLAB
Controle de Fluxo
Programação

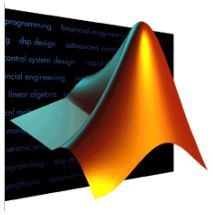
- Exemplo: Considere os 2 programas:

```
%While1
total = 0;
while (total <= 100)
  total = total + 10;
end
disp(total);
```

Aqui, total sai do while com valor de 110 já que, quando total for igual a 100 ele ainda entrará no laço e será somado com 10

```
%While2
total = 0;
flag = 1;
while (flag)
  if (total < 100)
    total = total + 10;
  else
    flag = 0;
  end
end
disp(total);
```

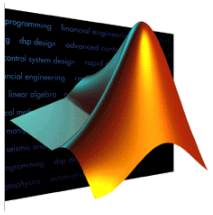
Nesse caso, total sai com valor 100 do laço



MATLAB
Controle de Fluxo
Programação

- Exemplo: Checagem de cartão de crédito
  - O programa a seguir verifica se uma pessoa entrou com um número ao ser requisitado o número de seu cartão de crédito

```
x = input('Entre com o numero de seu cartao de credito: ');
while (ischar(x))
  disp('Voce deve entrar com um numero !!');
  x = input('Entre com o numero de seu cartao de credito: ');
end
disp('OK');
```



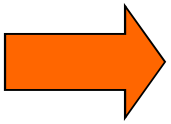
MATLAB
Controle de Fluxo
Programação

- Exemplo: Laço interminável

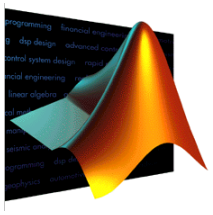
```
i=100;
while (i > 0)
    i=i/2;
    disp(i);
end
```

- Provocando um fim no laço....

```
i=100; count = 0;
while (i > 0)
    i=i/2;
    disp(i);
    count = count + 1;
    if (count > 20)
       break;
    end
end
```

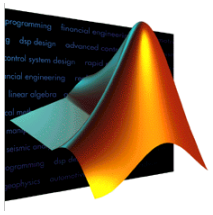


Uma contagem é feita e, caso o contador atinja o valor de 20, o laço é quebrado



MATLAB
Controle de Fluxo
Programação

- Estrutura switch-case
  - Usada quando seqüências de comandos devem ser condicionalmente executadas baseadas no uso repetido de um teste de igualdade com um argumento comum

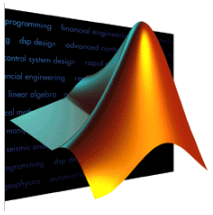


MATLAB
Controle de Fluxo
Programação

- switch

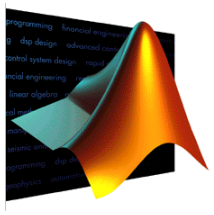
```
switch expressão
        case valor1
                comandos
        case valor2
                comandos
        .....
        otherwise
                comandos
end
```

```
Verifique expressão
        caso valor1
                comandos
        caso valor2
                comandos
        ....
        caso contrário
                comandos
fim
```



MATLAB
Controle de Fluxo
Programação

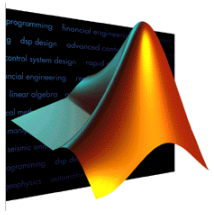
- switch - Observação
  - **Para programadores de C:**
    - **NÃO USA *BREAK !!!!***
    - **Não tem ":"**



MATLAB
Controle de Fluxo
Programação

- switch
  - **Exemplo:**

```
switch input_num
   case -1
         disp('negativo');
   case 0
         disp('zero');
   case 1
         disp('positivo');
   otherwise
         disp('outro valor');
   end
```



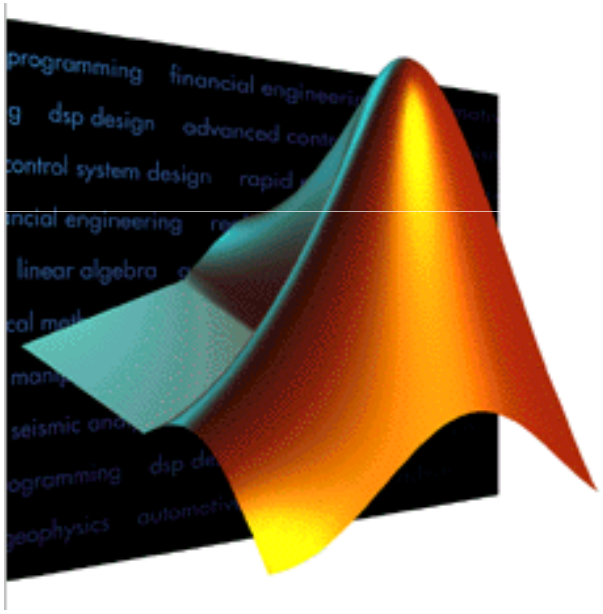
MATLAB
Controle de Fluxo
Programação

- Exemplo: Calculadora de 4 operações

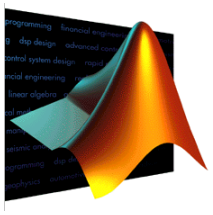
```
x = input ('Entre com o valor1: ');
y = input ('Entre com o valor2: ');
op = input ('Qual a operacao: ');
switch op
case {'+'}
  z = x+y;
case {'-'}
  z=x-y;
case {'*'}
  z = x*y;
case {'/','\'}
  z = x/y;
otherwise
  disp('Operacao desconhecida')
  z = nan;
end
disp(z);
```

Por que a definição de z=nan ? O que acontece se retirarmos essa linha ?

Programação no MatLab
Funções



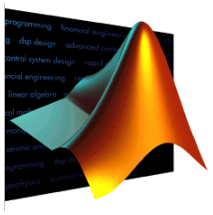
Carlos Alexandre Mello



MATLAB
Funções

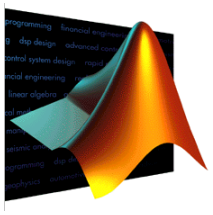
- Aceitam argumentos de entrada e retornam argumentos de saída
- Operam com variáveis próprias fora do workspace do MatLab
- Exemplo de Função:

```
function y = average(x)
% Average Mean of Vector elements
y = sum(x)/length(x);
```



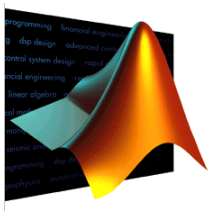
MATLAB
Funções

- **Regras e Propriedades**
  - O nome da função deve ser idêntico ao nome do arquivo
  - As linhas de comentário de um arquivo que antecedem a primeira linha que não contém comentários constituem o texto que é apresentado como help da função



MATLAB
Funções

- **Regras e Propriedades**
  - **As variáveis usadas nas funções independem das variáveis da workspace do MatLab**
    - **Se, na workspace do MatLab, há uma variável A com valor 10 e este mesmo nome de variável é utilizado dentro de uma função, assumindo o valor 20 (por exemplo), ao executarmos a função, A será igual a 20 apenas dentro da função. Na workspace, A continua sendo igual a 10**

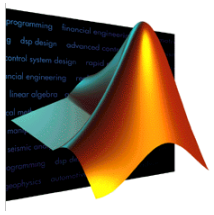


MATLAB
Checando o Número de Argumentos
Funções

- Funções *nargin* e *nargout* determinam o número de argumentos de entrada e saída de uma função

```
function c = testarg1(a,b)
if (nargin == 1)
        c = a.^2;
elseif (nargin == 2)
        c = a + b;
end
```

- nargout pode ser usado em funções que apresentam respostas diferentes dependendo do número de saídas (como polyder(a,b))



MATLAB
Estrutura de uma Função
Funções

Nome que deve ser usado no arquivo-M

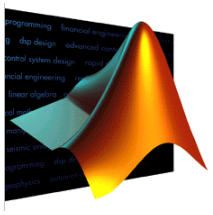
Linha de definição da função →

Comentário →

Corpo da função →

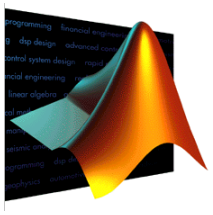
```
function f = fat(n)
% FAT Fatorial
f = prod(1:n);
```

Arquivo-M



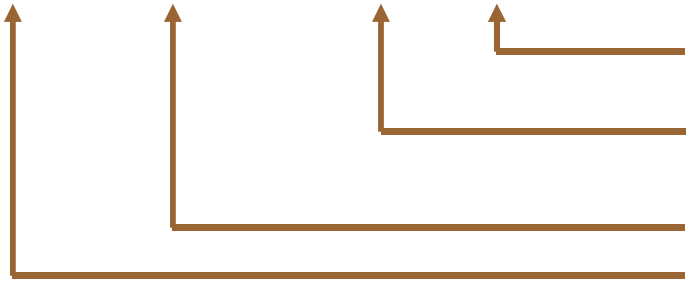
MATLAB
Anatomia de uma Função
Funções

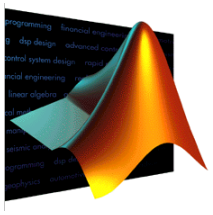
- Uma linha de definição da função
- Texto de ajuda da função
  - Servirá como Help da função
- Corpo da função
- Comentários



MATLAB
Anatomia de uma Função
Funções

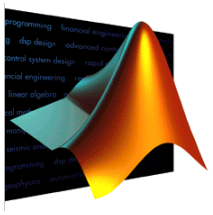
- Uma linha de definição da função
  - Essa linha informa ao MatLab que o arquivo contém uma função e especifica os argumentos da função
    - **function y = average(x)**





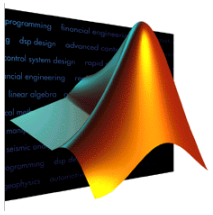
MATLAB
Anatomia de uma Função
Funções

- Uma linha H1
  - Primeira linha de ajuda do texto é uma linha comentada seguindo imediatamente a linha de definição da função
    - **% AVERAGE Média de um vetor**



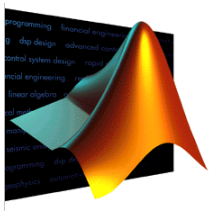
MATLAB
Anatomia de uma Função
Funções

- Texto de ajuda da função
  - Texto de ajuda com uma ou mais linhas comentadas para facilitar o uso da função
  - Mesma estrutura da linha H1
    - **Linhas precedidas de %**



MATLAB
Anatomia de uma Função
Funções

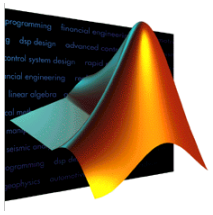
- Corpo da função
  - Códigos do MatLab para desempenhar toda a computação necessária e atribuir valores aos argumentos de saída
    - **Chamadas a funções**
    - **Construções de Programação (if...else, etc)**
    - **Cálculos**
    - **Comentários**
    - **....**
- Comentários
  - Podem aparecer em qualquer parte do arquivo sempre precedidos pelo símbolo %



MATLAB
Nomes das Funções
Funções

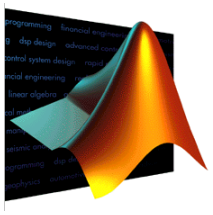
- O MatLab só usa os 31 primeiros caracteres
- O nome deve começar com uma letra
- Pode ser composto por letras, números, etc
- Nome do arquivo terminado por ".m"

filename.m



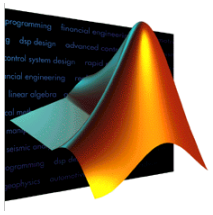
MATLAB
Funções de Gerenciamento de Memória

- Removendo uma Função da Memória
  - O código do arquivo-M é removido da memória quando:
    - **A função é chamada novamente com um novo código nela**
    - **A função é apagada explicitamente com o comando *clear***
    - **Todas as funções são explicitamente apagadas com o comando *clear functions***
    - **MATLAB sai da memória**



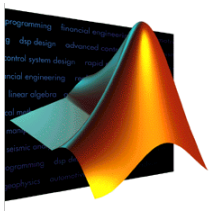
MATLAB
Funções de Gerenciamento de Memória

- Para Conservar Memória
  - Evite usar as mesmas variáveis como entrada e saída de uma função:
    - **y = fun(x, y)**
  - Use a função *clear* ou *clear nome_da_variável* sempre que possível
  - Re-use variáveis o máximo possível



MATLAB
Subfunções

- Arquivos-M de funções podem conter mais de uma função (chamadas *subfunções*)
- Observe que apenas a função primária (a qual gerou o nome do arquivo-M) pode ser executada do MatLab
- As outras funções são chamadas de dentro da função primária apenas



MATLAB
Subfunções

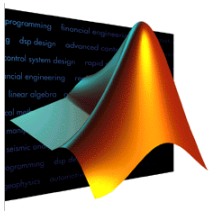
Função Primária →

Subfunção →

Subfunção →

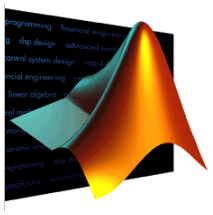
```
function y = teste(a)
.....

function x = teste2(b)
.....

function z = teste3(c)
.....
```

Arquivo teste.m



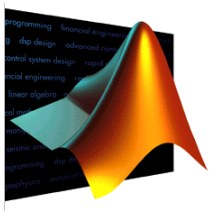
MATLAB
Subfunções

- A função teste é executada de dentro do MatLab e chama as subfunções teste2 e teste3
- Se tentarmos executar externamente teste2 ou teste3, um erro será gerado, pois apenas o arquivo teste.m existe



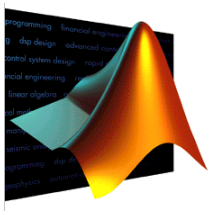
MATLAB
Funções

- Ao ser chamada, a função permanece na memória até que o comando *clear* seja dado:
  - ***clear function_name***
    - **remove uma função específica do workspace**
  - ***clear functions***
    - **remove todas as funções**
  - ***clear all***
    - **remove funções e variáveis**



MATLAB
Funções

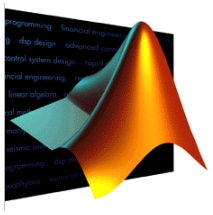
- Exemplos de Funções:
  - Função que calcula as raízes de um polinômio de grau menor ou igual a dois
  - Caso I:
    - Os coeficientes do polinômio são inseridos dentro da função
    - Ou seja, a função não possui argumentos de entrada, mas tem uma saída (Y) que pode ser um número apenas ou um vetor
    - Função baskhara1.m
    - Observação: Não foi considerada a possibilidade do usuário atribuir ZERO aos três coeficientes



MATLAB
Funções

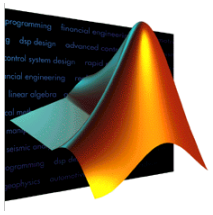
- baskhara1.m

```
function y=baskhara1
a=input('Entre com o primeiro coeficiente: ');
b=input('Entre com o segundo coeficiente: ');
c=input('Entre com o terceiro coeficiente: ');
if (a == 0)
   disp('Polinomio do Primeiro Grau');
   y=-b/c;
else
   disp('Polinomio do Segundo Grau');
   delta = b^2 - 4*a*c;
   y(1) = (-b + sqrt(delta))/(2*a);
   y(2) = (-b - sqrt(delta))/(2*a);
end
```



MATLAB
Funções

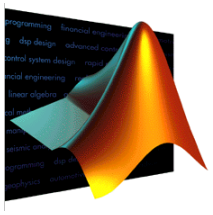
- Função que calcula as raízes de um polinômio de grau menor ou igual a dois
  - Caso II:
    - Os coeficientes do polinômio são passados como argumentos para a função
    - Ou seja, a função tem três entradas (A, B, C) e uma saída (Y) que pode ser um número apenas ou um vetor
    - Função baskhara2.m



MATLAB
Funções

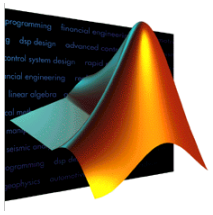
- baskhara2.m

```
function y=baskhara2(a,b,c)
if (nargin<2)
   disp('Erro !! Deve entrar com pelo menos 2 coeficientes!');
   return;
end
if (nargin==2)
   disp('Polinomio do Primeiro Grau');
   y=-b/a;
else
   disp('Polinomio do Segundo Grau');
   delta = b^2 - 4*a*c;
   y(1) = (-b + sqrt(delta))/(2*a);
   y(2) = (-b - sqrt(delta))/(2*a);
end
```



MATLAB
Funções

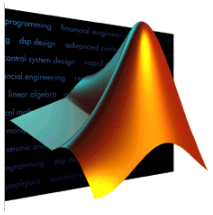
- Função que calcula as raízes de um polinômio de grau menor ou igual a dois
  - Caso III:
    - Os coeficientes do polinômio são passados como argumentos para a função dentro de um vetor
    - Ou seja, a função tem uma entrada (X) na forma de um vetor e uma saída (Y) que pode ser um número apenas ou um vetor
    - Função baskhara3.m



MATLAB
Funções

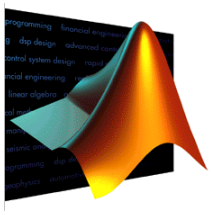
- baskhara3.m

```
function y=baskhara3(x)
tam=size(x);
if (tam(2) == 2)
   disp('Polinomio do Primeiro Grau');
   y=-x(2)/x(1);
elseif (tam(2) == 3)
   disp('Polinomio do Segundo Grau');
   a=x(1); b=x(2); c=x(3);
   delta = b^2 - 4*a*c;
   y(1) = (-b + sqrt(delta))/(2*a);
   y(2) = (-b - sqrt(delta))/(2*a);
else
   disp ('Erro ! Polinomio desconhecido !!');
end
```



MATLAB
Funções

- Função que calcula as raízes de um polinômio de grau menor ou igual a dois
  - Caso IV:
    - Os coeficientes do polinômio são passados como argumentos para a função dentro de um vetor
    - A função tem duas saídas
    - Ou seja, a função tem uma entrada (X) na forma de um vetor e duas saídas (Y, Z)
    - Função baskhara4.m

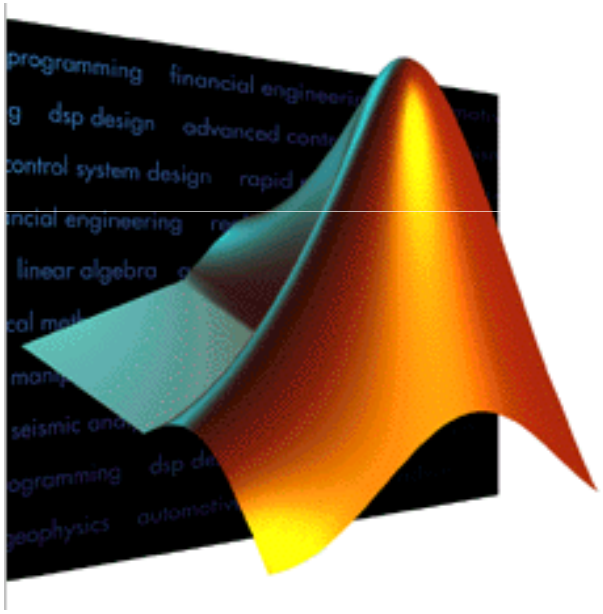


MATLAB
Funções

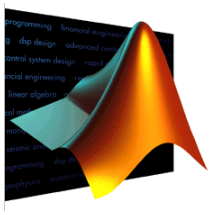
- baskhara4.m

```
function [y,z]=baskhara4(x)
tam=size(x);
if (tam(2) == 2)
   disp('Polinomio do Primeiro Grau');
   y=-x(2)/x(1);
   z=nan;
elseif (tam(2) == 3)
   disp('Polinomio do Segundo Grau');
   a=x(1); b=x(2); c=x(3);
   delta = b^2 - 4*a*c;
   y = (-b + sqrt(delta))/(2*a);
   z = (-b - sqrt(delta))/(2*a);
else
   disp ('Erro ! Polinomio desconhecido !!');
end
```

Simulink

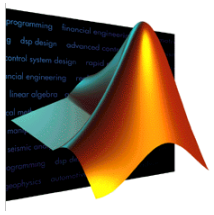


Carlos Alexandre Mello



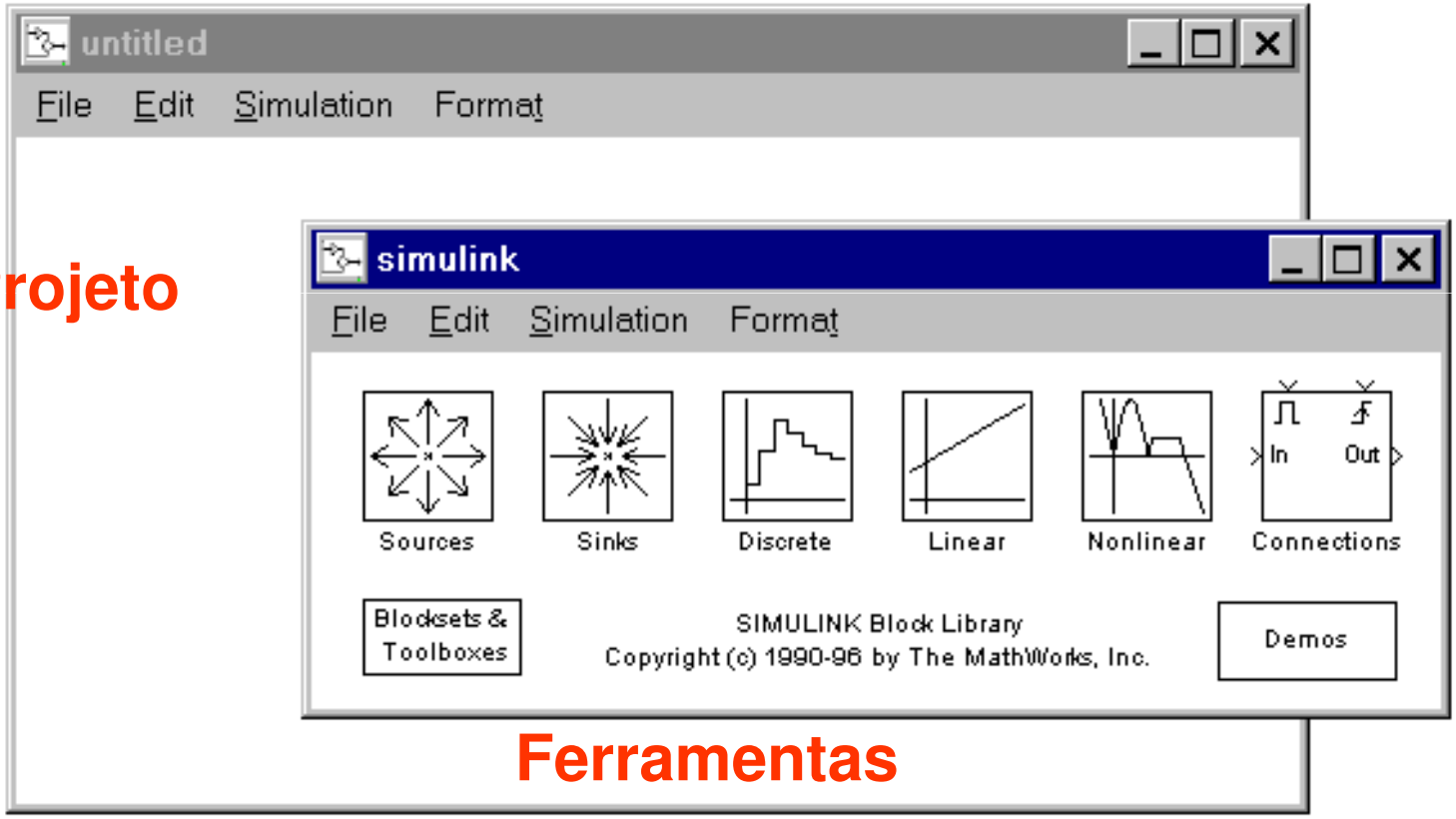
Simulink

- Módulo para análise de modelos e construção de funções
- Para executá-lo, digite
  - >> simulink
- dentro do MatLab



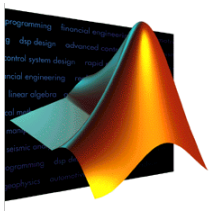
Simulink

Tela de Entrada



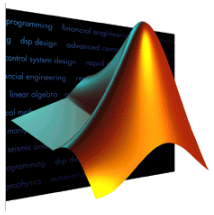
Área de Projeto

Ferramentas



Simulink

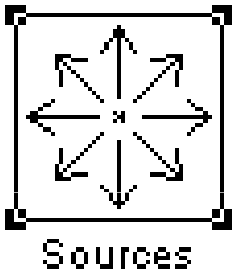
- Para criar um sistema, basta arrastar um bloco das ferramentas para a área de projeto
- As conexões entre os blocos são feitas através das entradas (>) e saídas (>) de cada bloco
- As conexões podem ser marcadas (clicando com o mouse) e deletadas (com del)

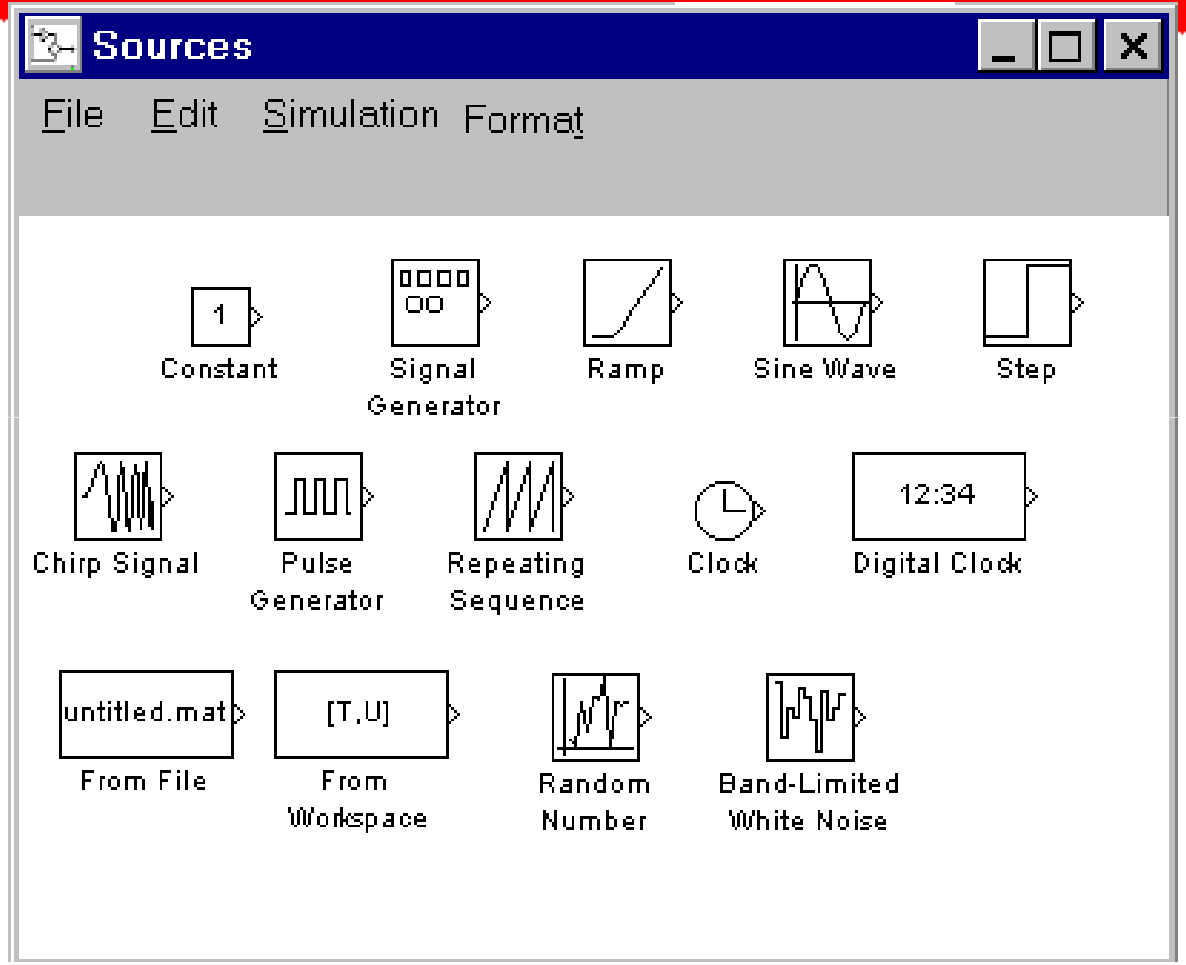


Simulink

Módulos

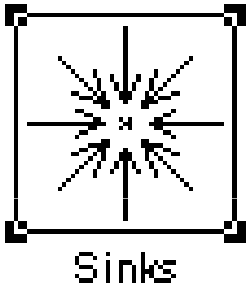
- Fontes

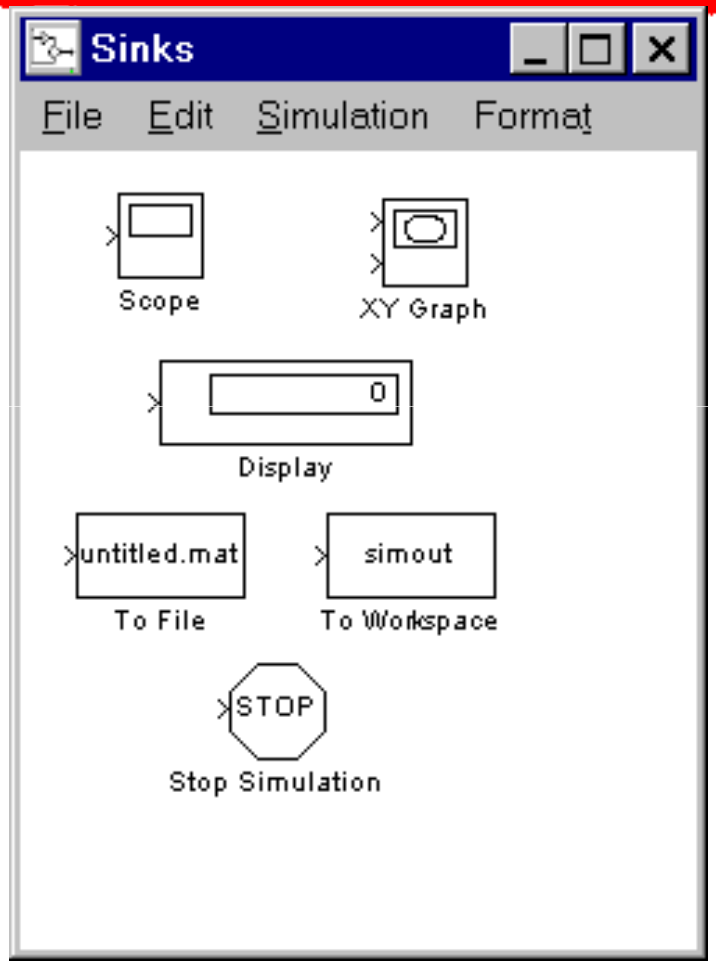




Simulink
Módulos

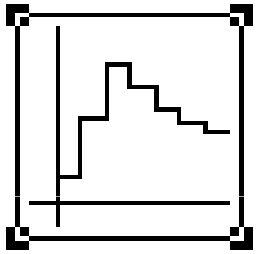
- Saídas

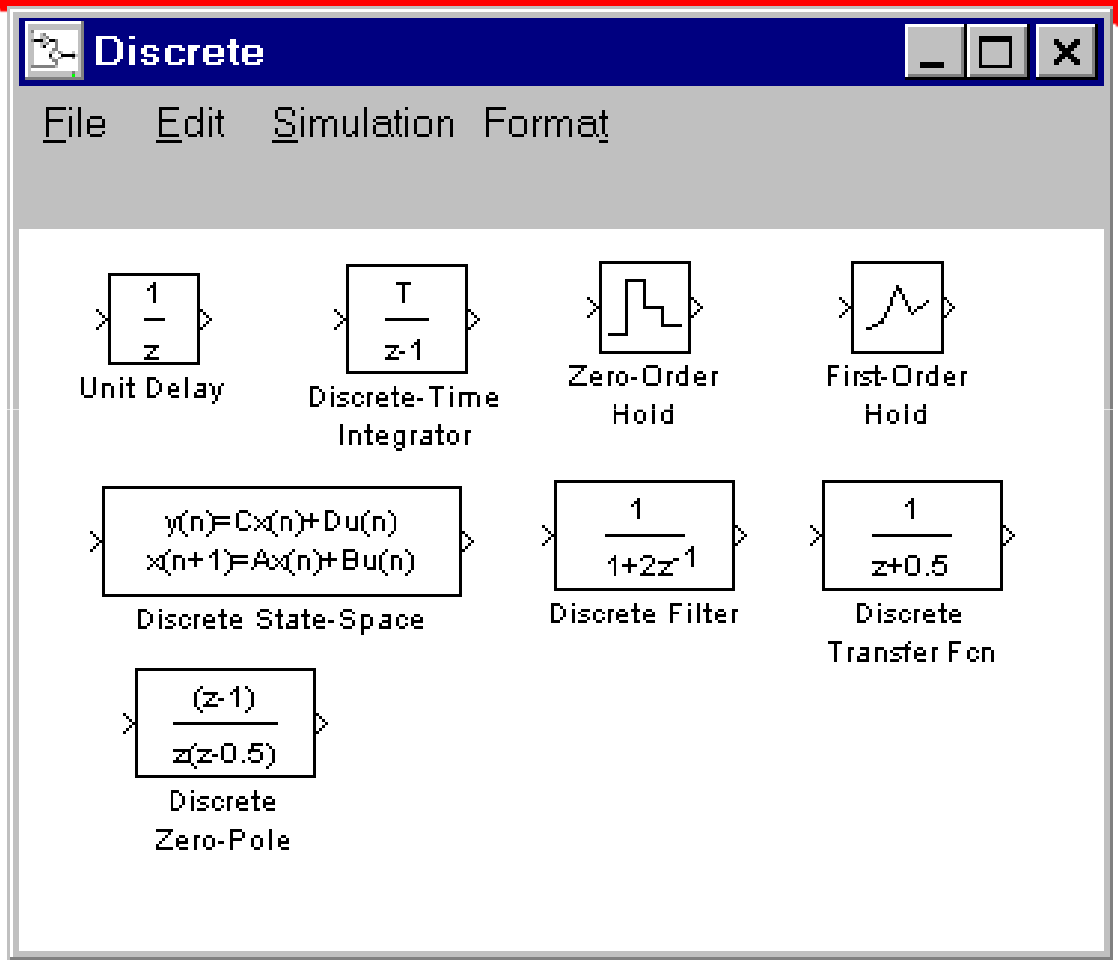




Simulink
Módulos

- Discreto

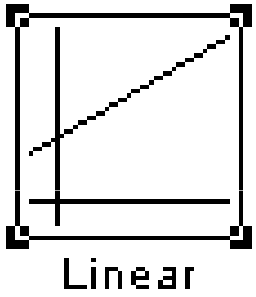


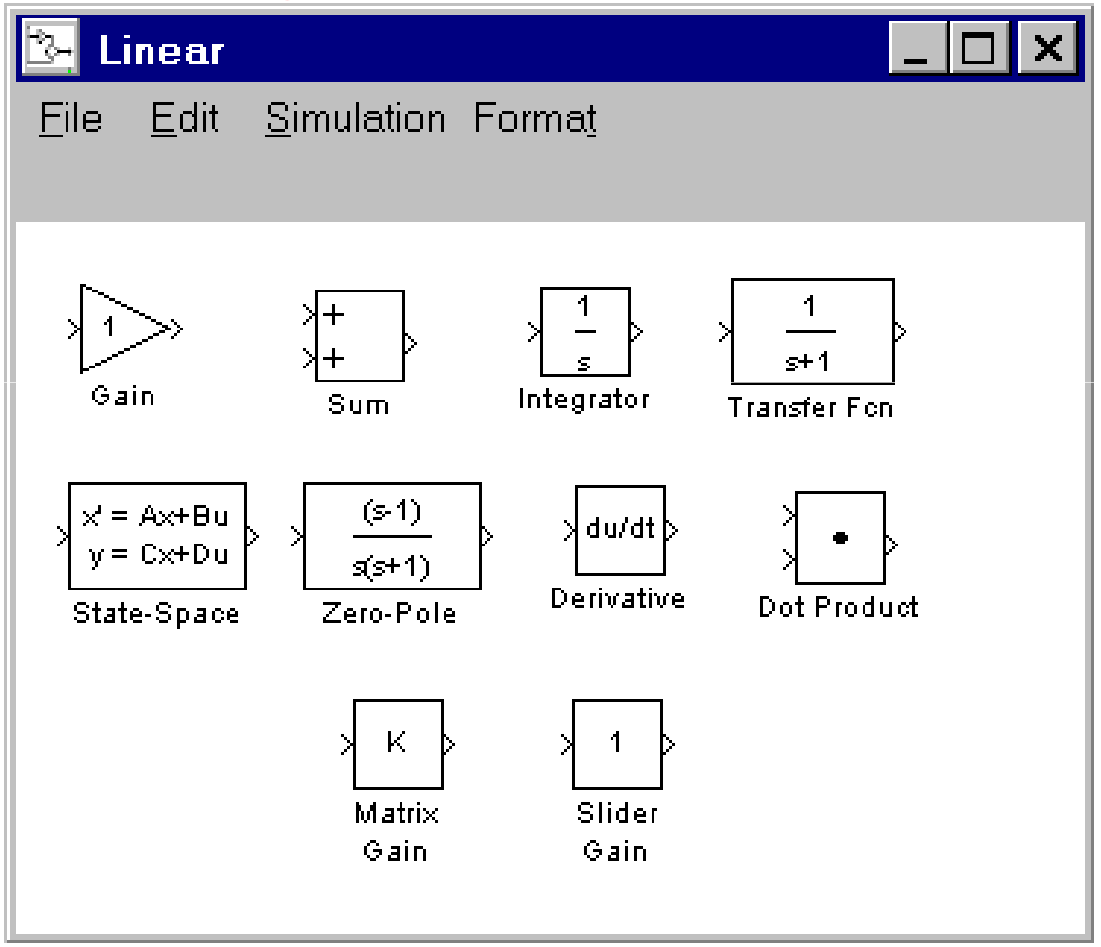


Simulink

Módulos

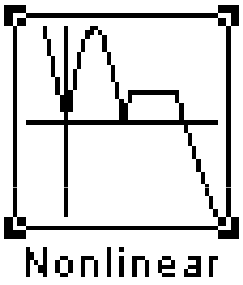
- Linear

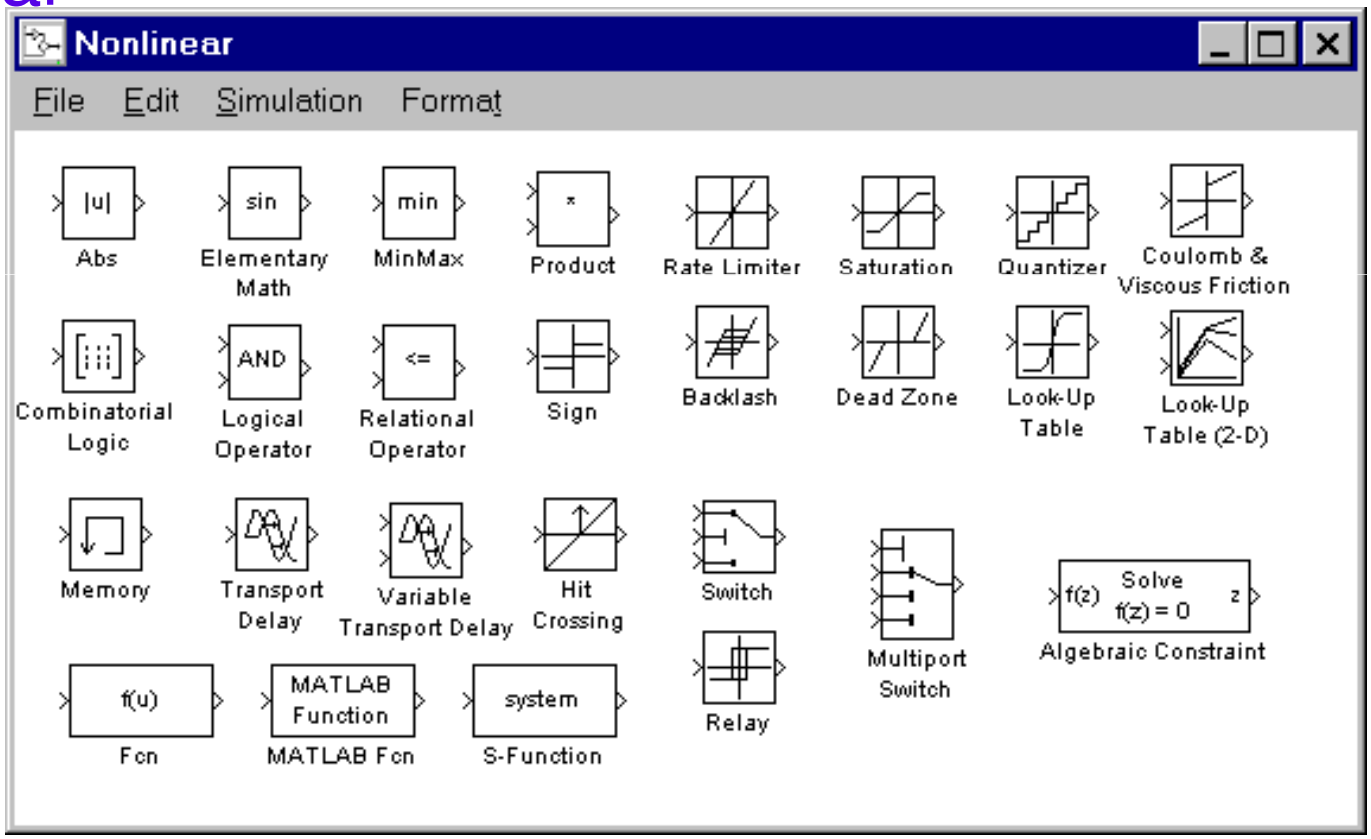




Simulink
Módulos

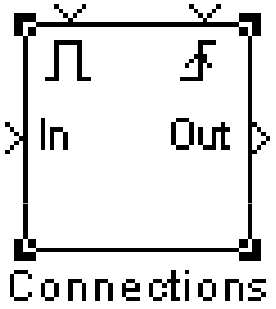
- Não-Linear

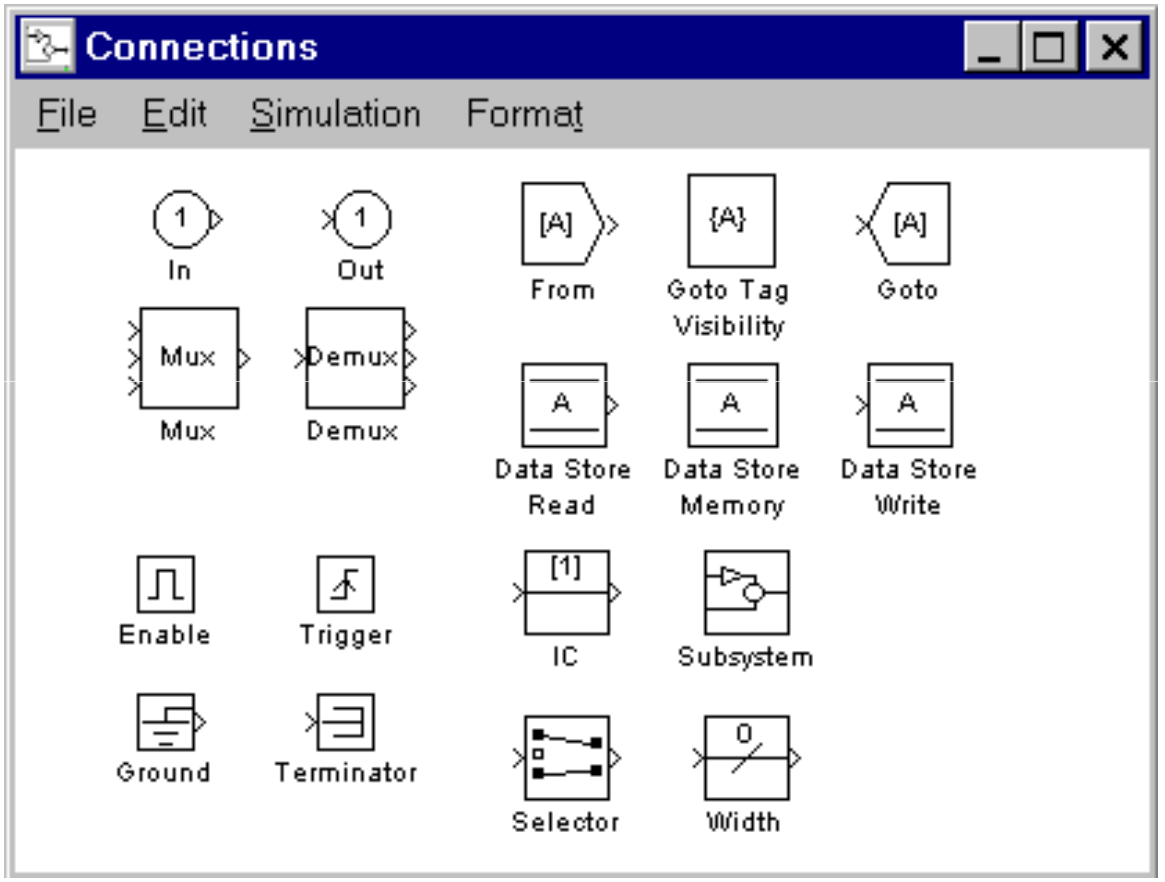




Simulink
Módulos

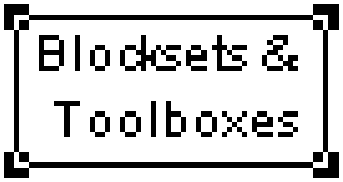
- Conexões

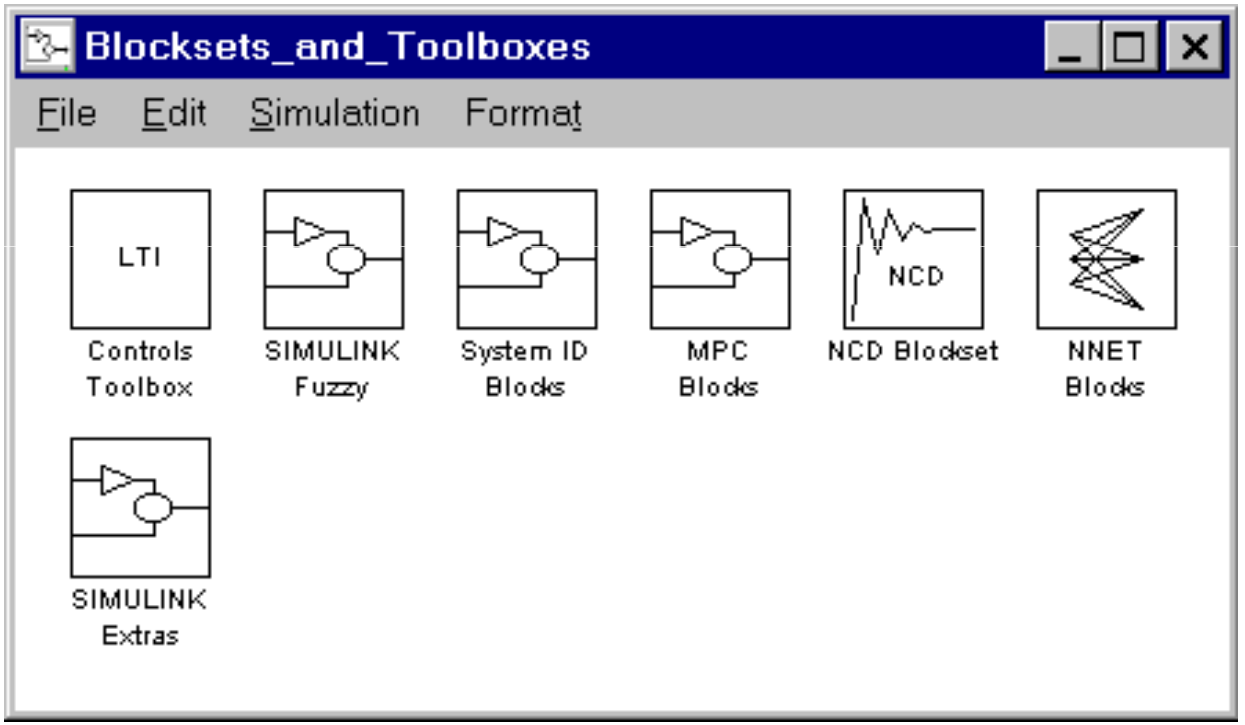


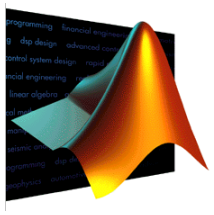


Simulink
Módulos

- Blocos e Ferramentas

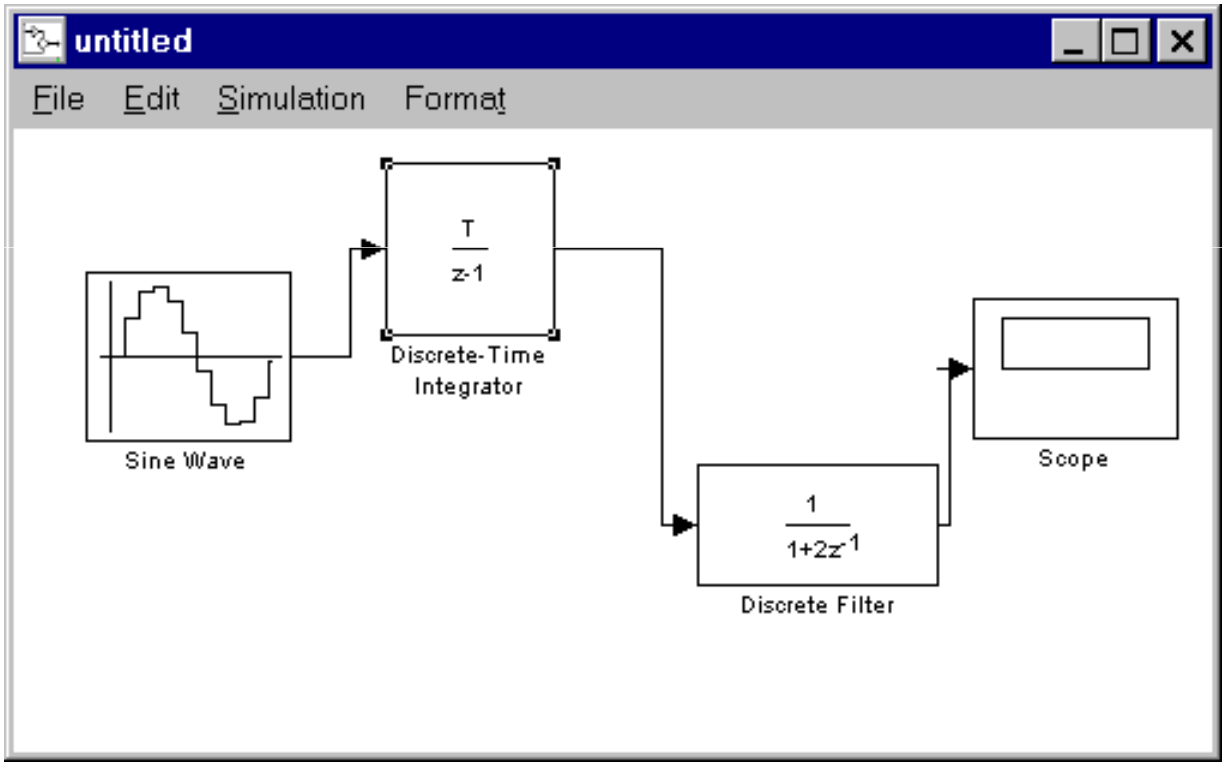


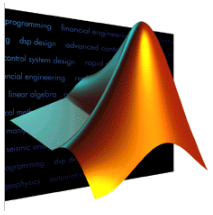




Simulink

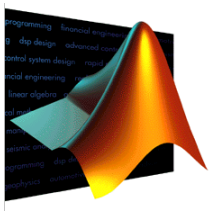
- Exemplo 1:





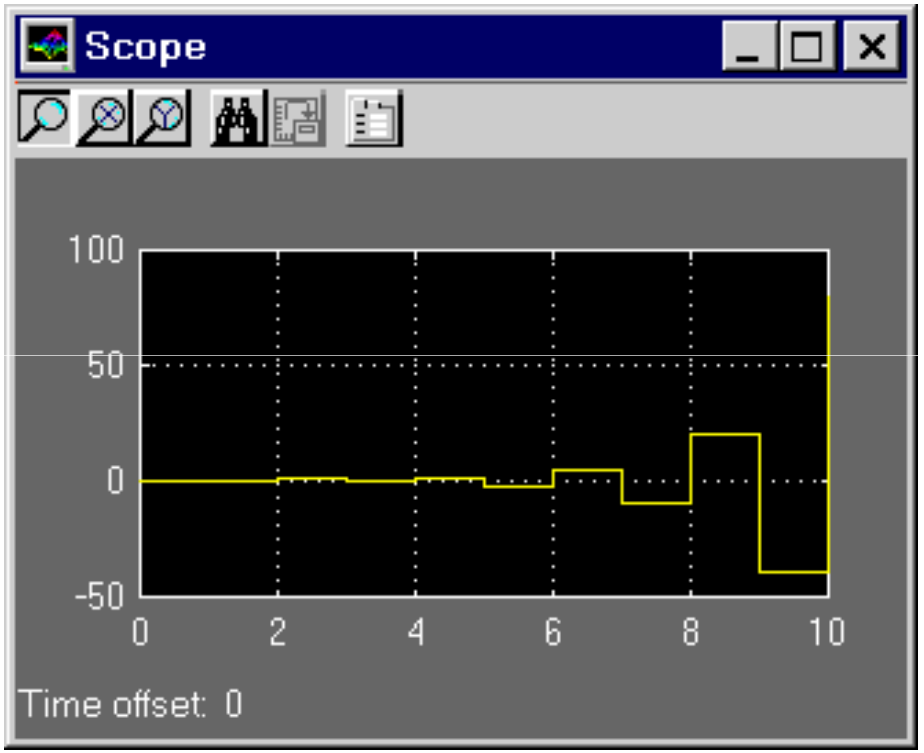
Simulink

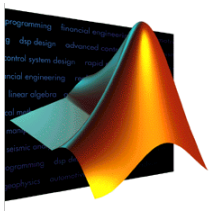
- Exemplo 1: A simulação anterior deve ser executada
  - Simulation -> Start
- Ao terminar, o resultado pode ser visto no osciloscópio, clicando duas vezes nele



Simulink

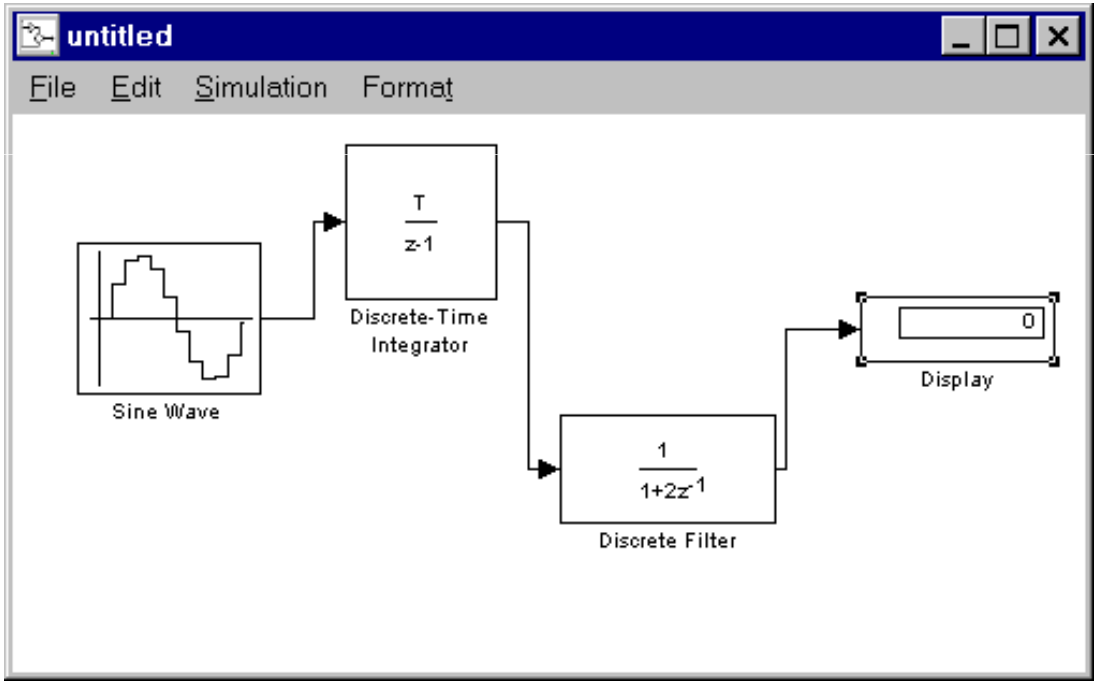
- Exemplo 1:





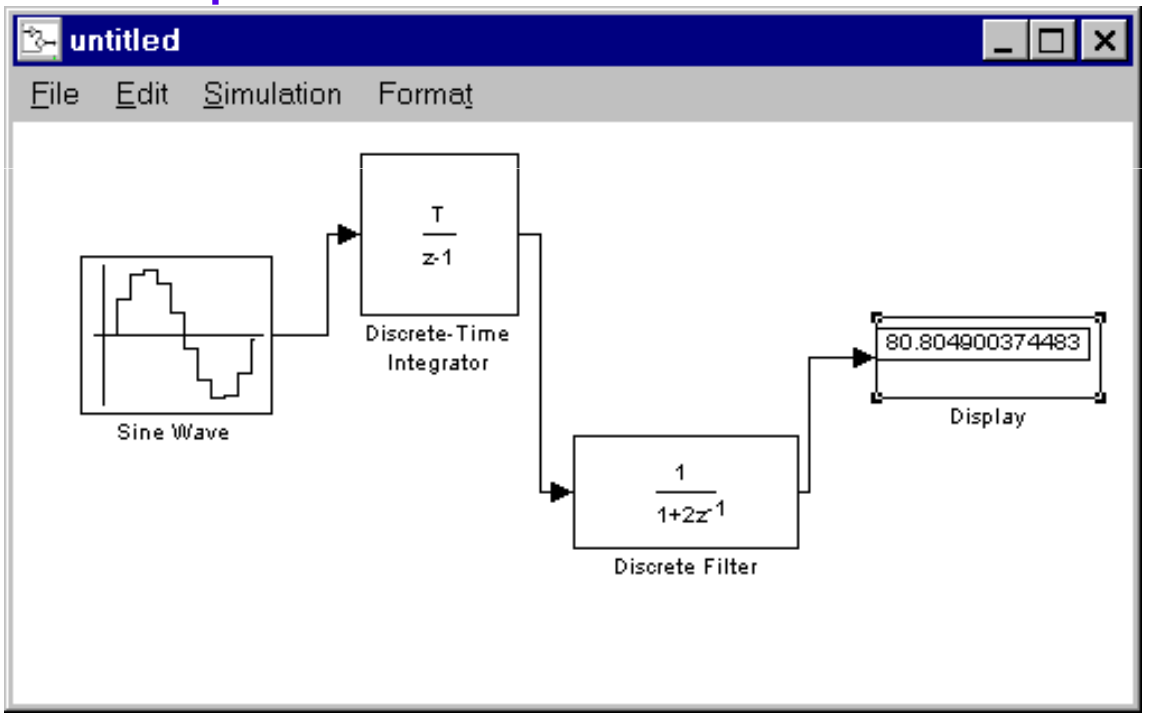
Simulink

- Exemplo 2: A saída do sistema pode ser alterada para um novo formato:



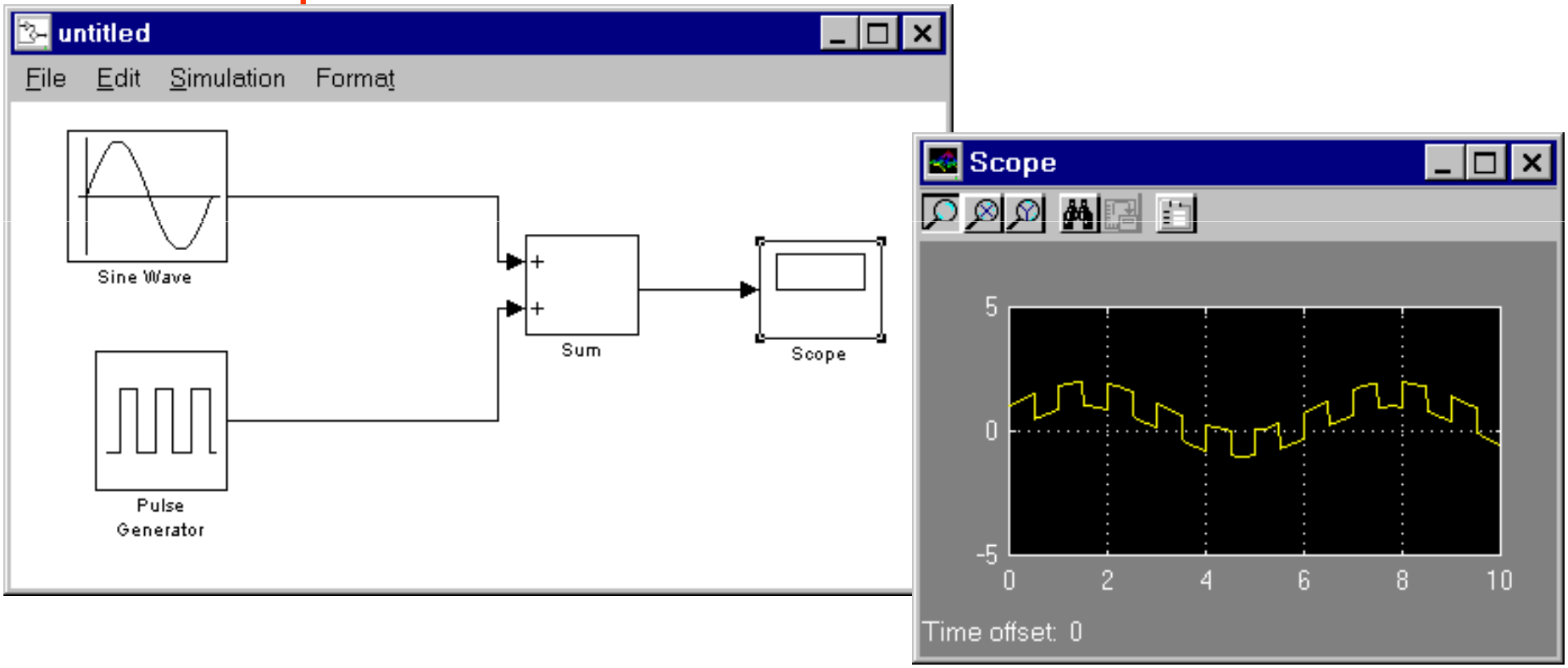
Simulink

- Exemplo 2: Após a execução do sistema, o resultado é apresentado



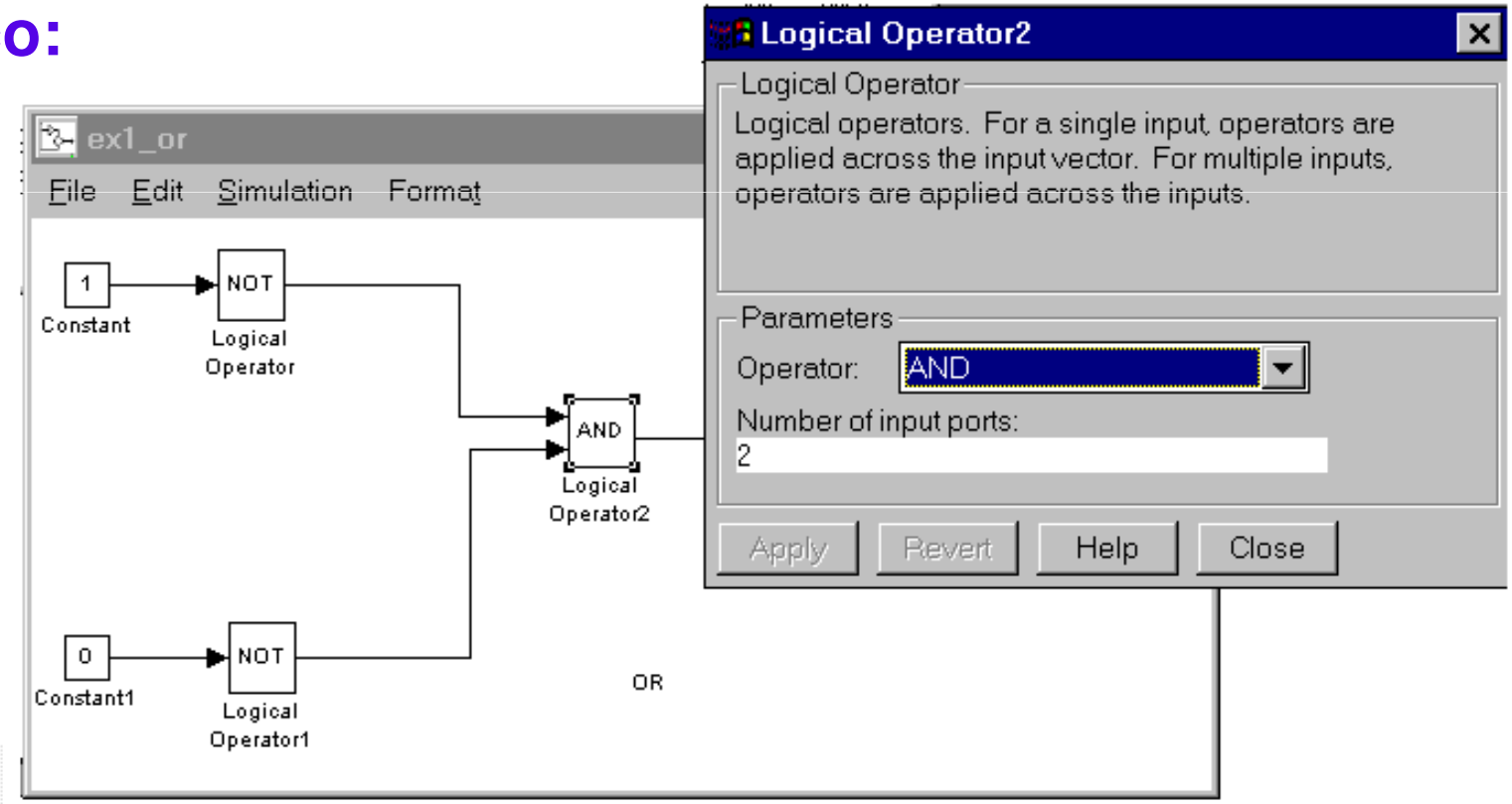
Simulink

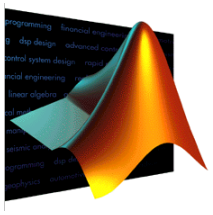
- Exemplo 3: Soma de duas entradas:



Simulink

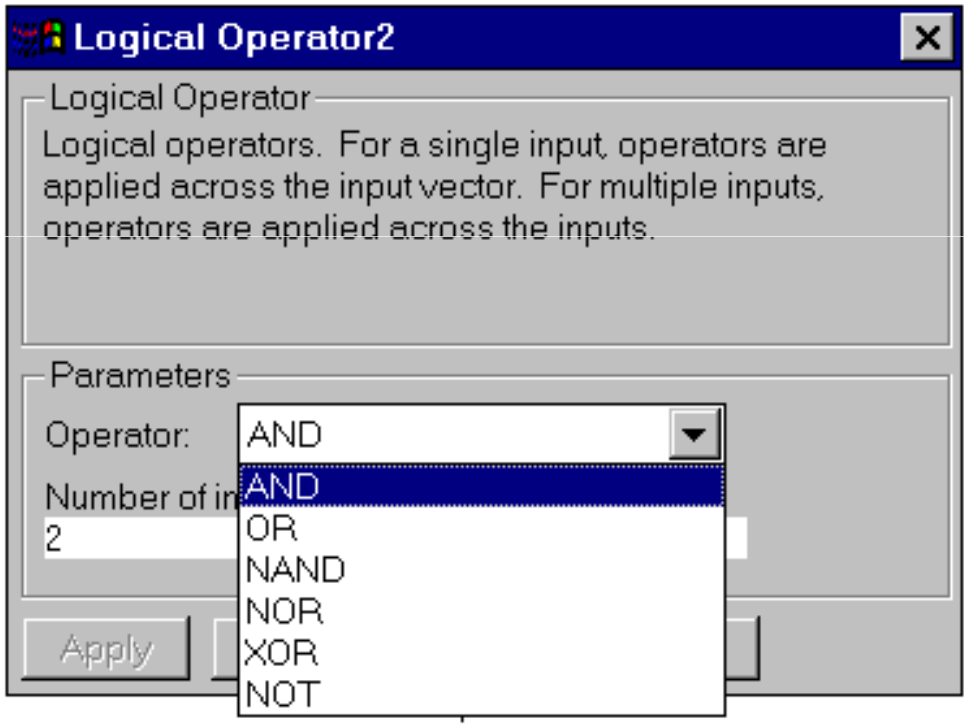
- Clicando duas vezes em um bloco, abre-se uma janela com as opções que podem ser alteradas do bloco:





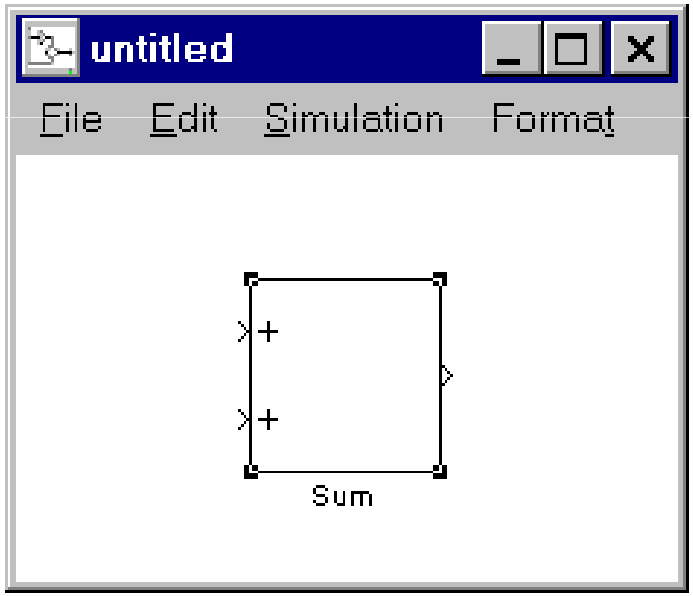
Simulink

- Um mesmo bloco pode assumir diferentes funções:



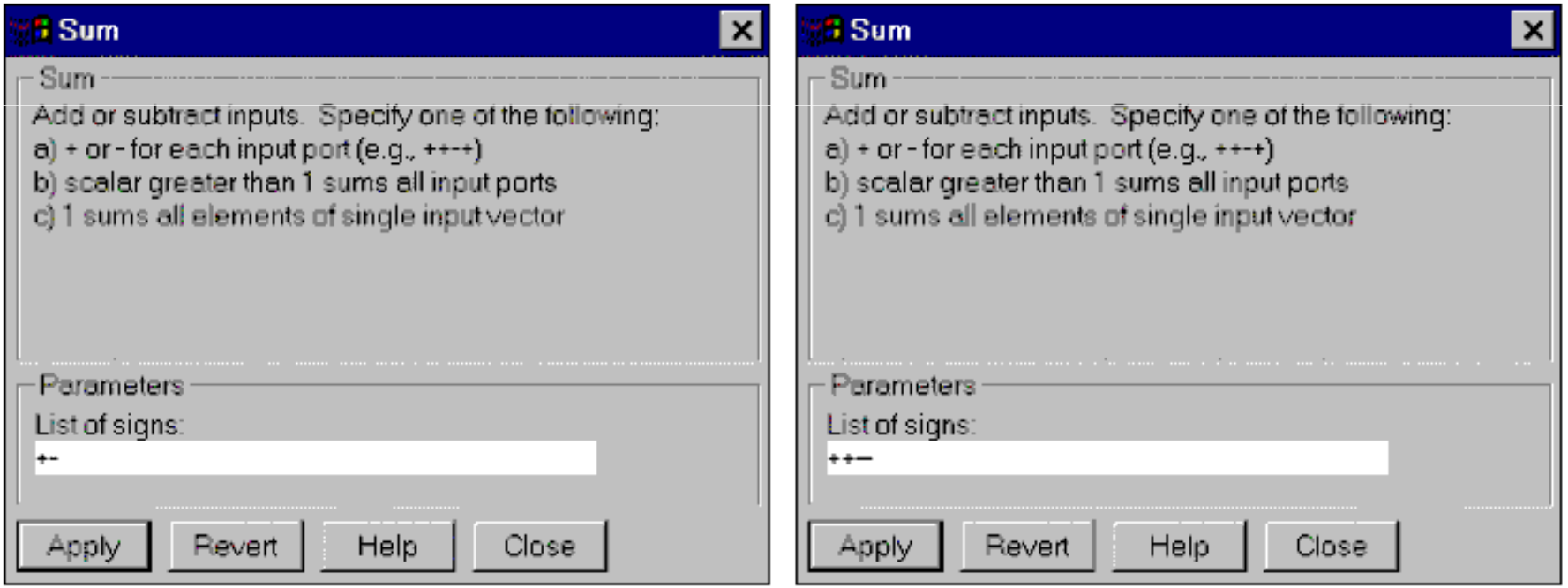
Simulink

- As propriedades de um bloco podem alterar suas próprias funções, como um somador, por exemplo



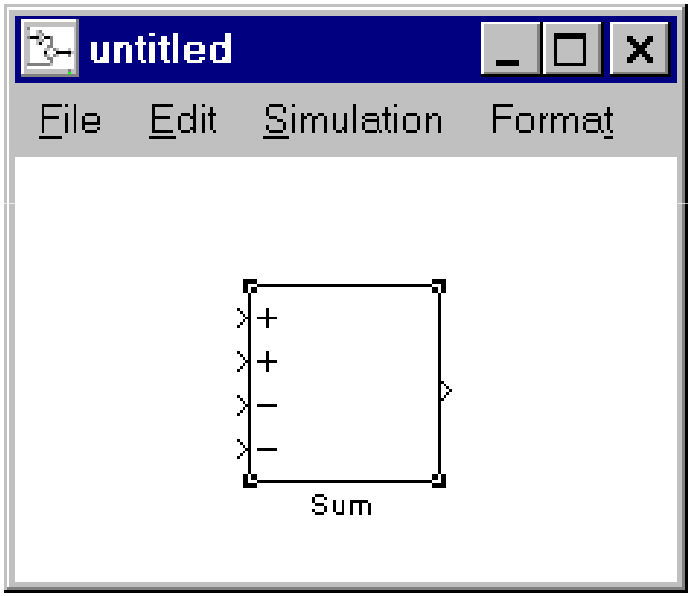
Simulink

- Ao ser clicado duas vezes, suas propriedades permitem que o bloco tenha suas funções alteradas , incluindo o número de entradas



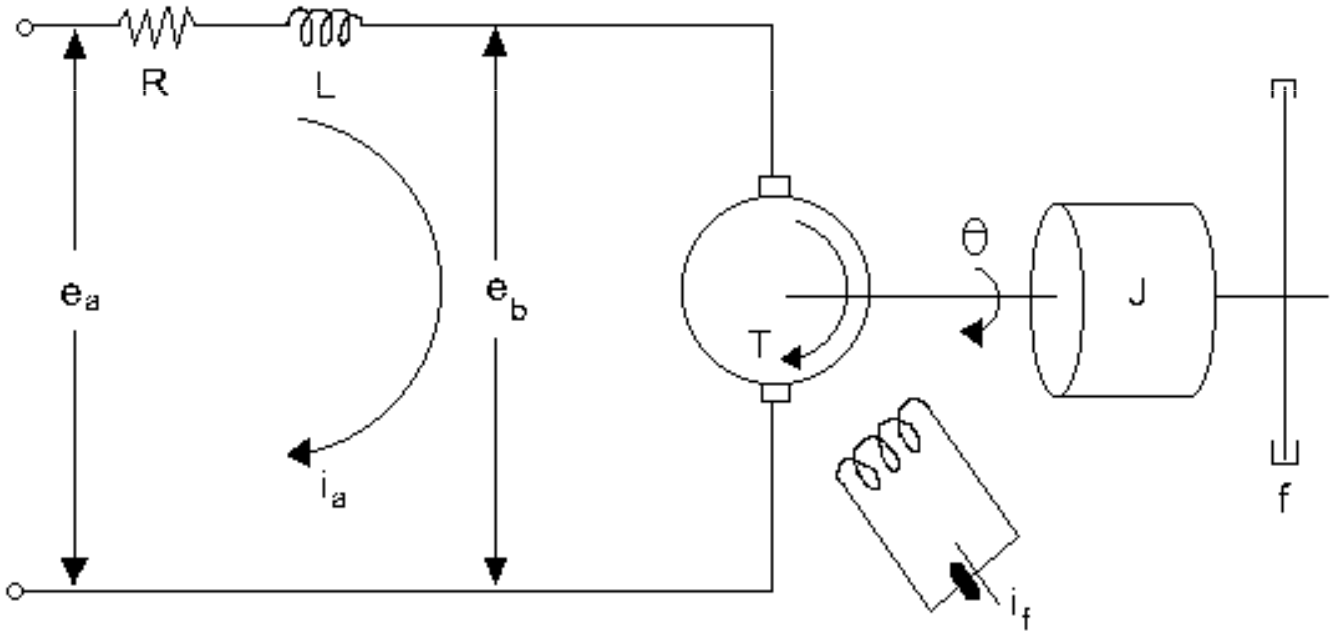
Simulink

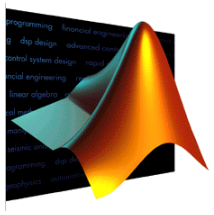
- Resultando num novo bloco



Simulink

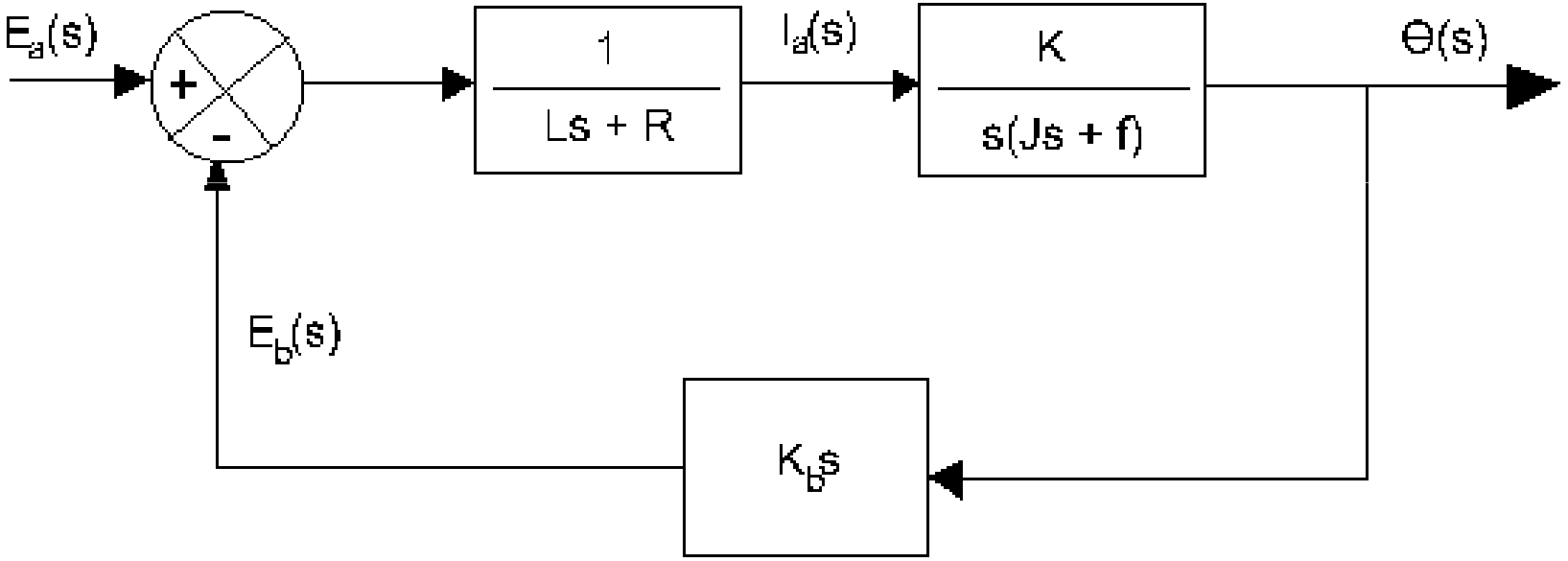
- **Exemplo 4: Motor c.c. controlado por armadura (retirado do livro Engenharia de Controle Moderno, Ogata, pg 110)**





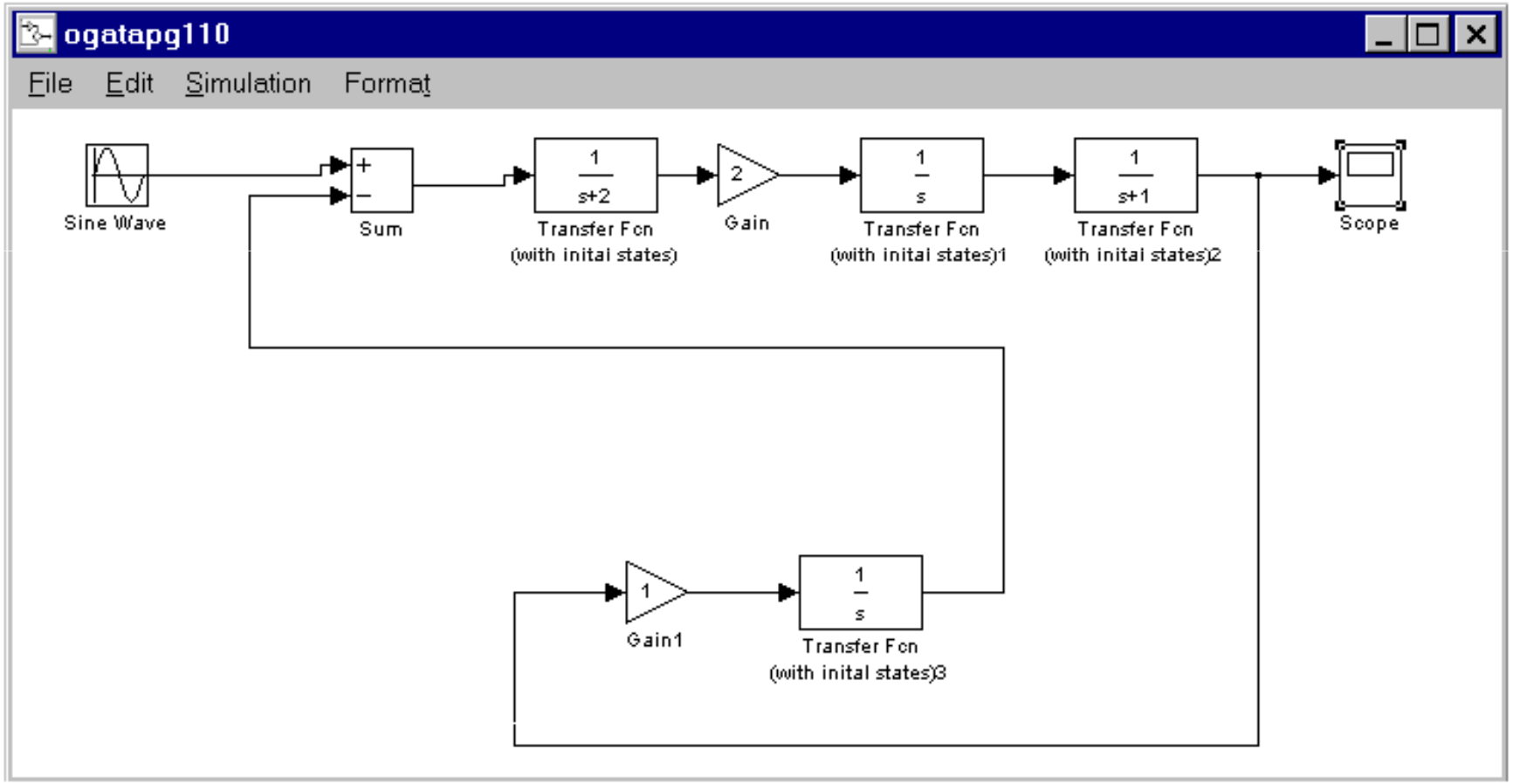
Simulink

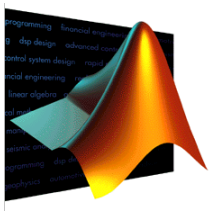
- **Exemplo 4: Em Diagrama de Blocos:**



Simulink

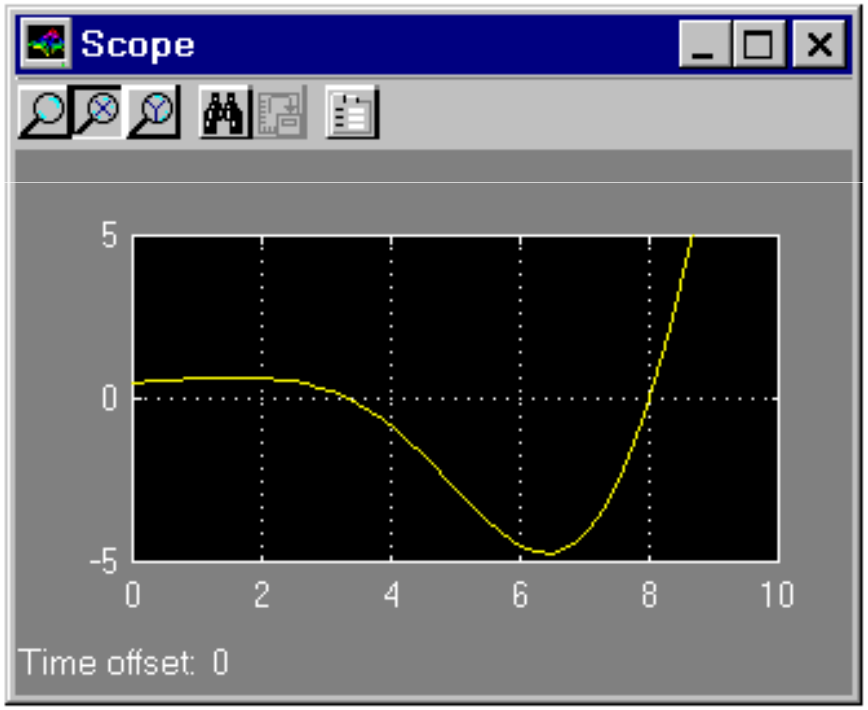
- **Exemplo 4: No Simulink, para R=L=J=f=K=Kb=1, temos :**

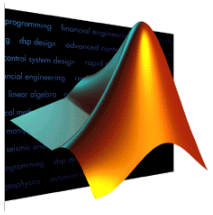




Simulink

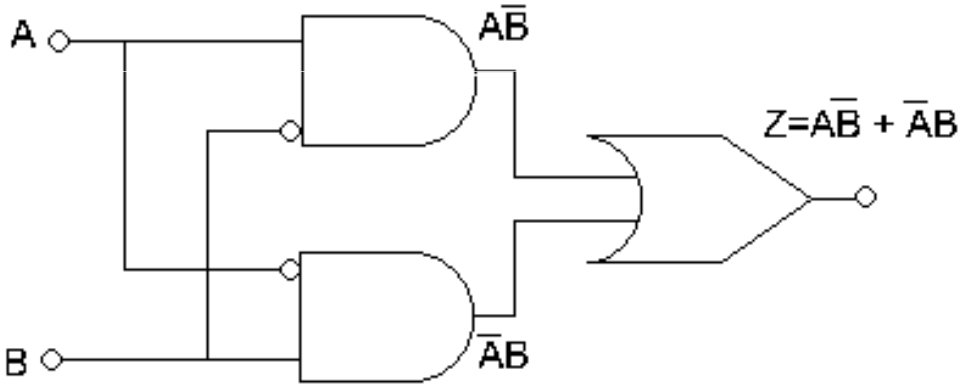
- **Exemplo 4: Ao ser inicializado, a resposta do osciloscópio será mostrada como abaixo:**





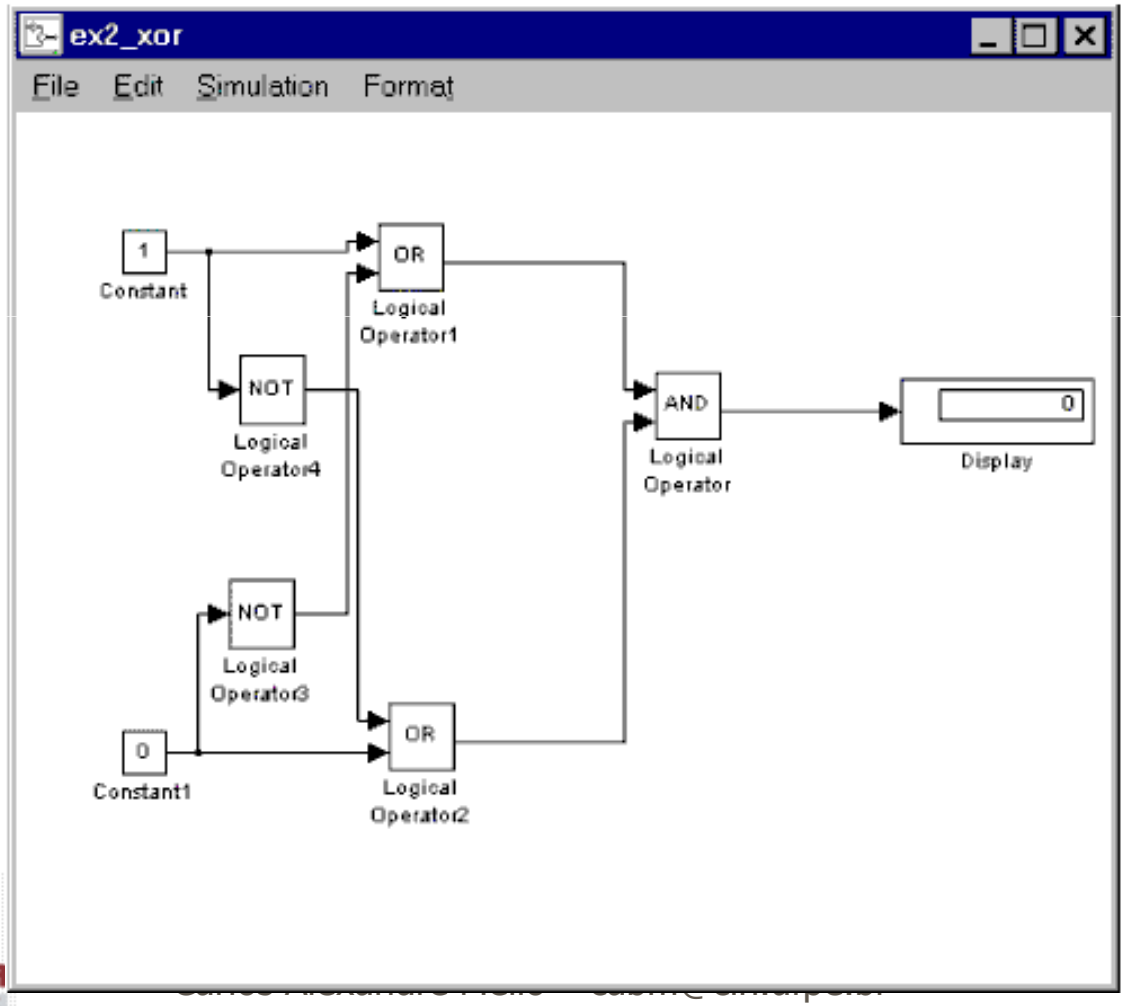
Simulink

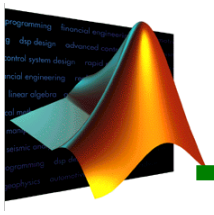
- **Exemplo 5: Eletrônica Digital: Ou-Exclusivo realizado com portas E, Não e Ou**
  - **Diagrama:**



Simulink

- **Exemplo 5: No Simulink**

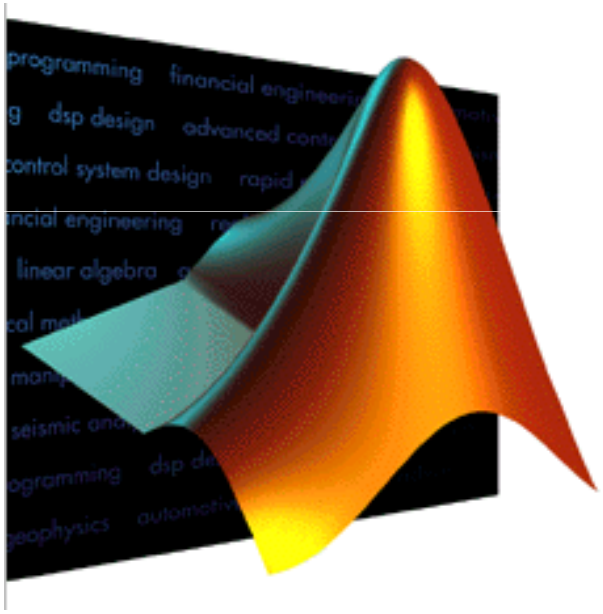




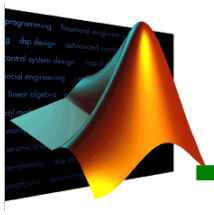
ToolBox de Sistemas de Controle

- Os sistemas contínuos e discretos podem ser descritos na forma de uma função de transferência de zeros e pólos
- As funções de transferência são descritas por razões de polinômios o que é facilmente tratado no MatLab através de vetores com os coeficientes dos polinômios

MatLab
Sistemas de Controle

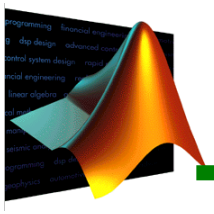


Carlos Alexandre Mello



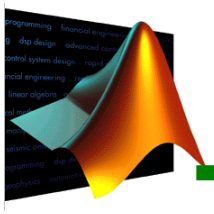
ToolBox de Sistemas de Controle

- Os sistemas LTI (*linear time-invariant*) podem ser especificados através de modelos de funções de transferência, modelos de zero/pólo/ganho e modelos de estado-espaço
- Para criar um objeto LTI basta usar uma das funções construtoras correspondentes:
  - tf, zpk, ss



ToolBox de Sistemas de Controle

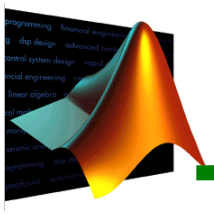
- No MatLab:
  - sys = tf(num, den) % função de transferência
  - sys = zpk(z,p,k) % zero/pólo/ganho
  - sys = ss(a,b,c,d) % espaço-estado
- Para criar modelos discretos no tempo, acrescenta-se o período de amostragem Ts às funções acima
  - sys = tf(num, den, Ts)



ToolBox de Sistemas de Controle

- Exemplos:
  - h = tf(1,[1 1])
- cria a função de transferência 1/(s + 1)

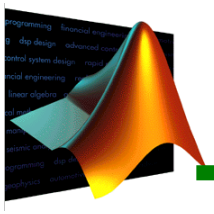
```
» h=tf(1,[1 1])

Transfer function:
  1
-----
s + 1

» 1+h

Transfer function:
s + 2
-----
s + 1
```



ToolBox de Sistemas de Controle

- Exemplos:
  - h = tf([1 2],[1 1 10]);
- cria a função de transferência

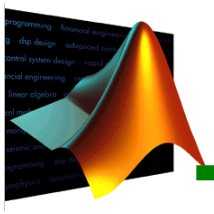
```
..
» h = tf([1 2], [1 1 10])

Transfer function:
    s + 2
------------
s^2 + s + 10
```



ToolBox de Sistemas de Controle

- Exemplos:

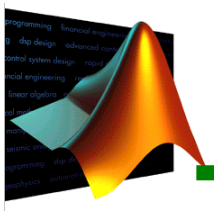
```
» h = tf([1 2], [1 1 10])

Transfer function:
    s + 2
------------
s^2 + s + 10

» q = 2 + h

Transfer function:
2 s^2 + 3 s + 22
----------------
  s^2 + s + 10
```



ToolBox de Sistemas de Controle

- Exemplos: Outra variação....

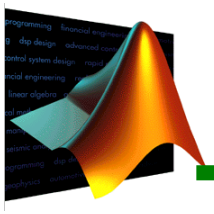
```
» h = tf(1, [1 -1], 'var', 'z')

Transfer function:
  1
-----
z - 1

Sampling time: unspecified
```



ToolBox de Sistemas de Controle

- Você pode usar o comando tf com apenas um argumento para especificar ganhos simples ou matrizes de ganho

```
» g=tf([1 0; 2 1])

Transfer function from input 1 to output...
 #1:  1

 #2:  2

Transfer function from input 2 to output...
 #1:  0

 #2:  1

Static gain.
```

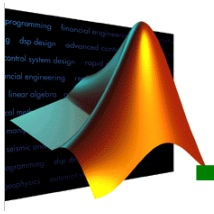


ToolBox de Sistemas de Controle

- Sistemas com múltiplas entradas e múltiplas saídas (função de transf):

```
» num = {0.5,[1 1]};
» den = {[1 0], [1 2]};
» H = tf(num,den)

Transfer function from input 1 to output:
0.5
---
 s

Transfer function from input 2 to output:
s + 1
-----
s + 2
```

$$H(s) = \begin{bmatrix} \frac{0.5}{s} & \frac{s+1}{s+2} \end{bmatrix}$$

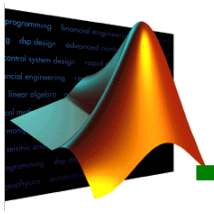


ToolBox de Sistemas de Controle

- Sistemas com múltiplas entradas e múltiplas saídas (outra forma):

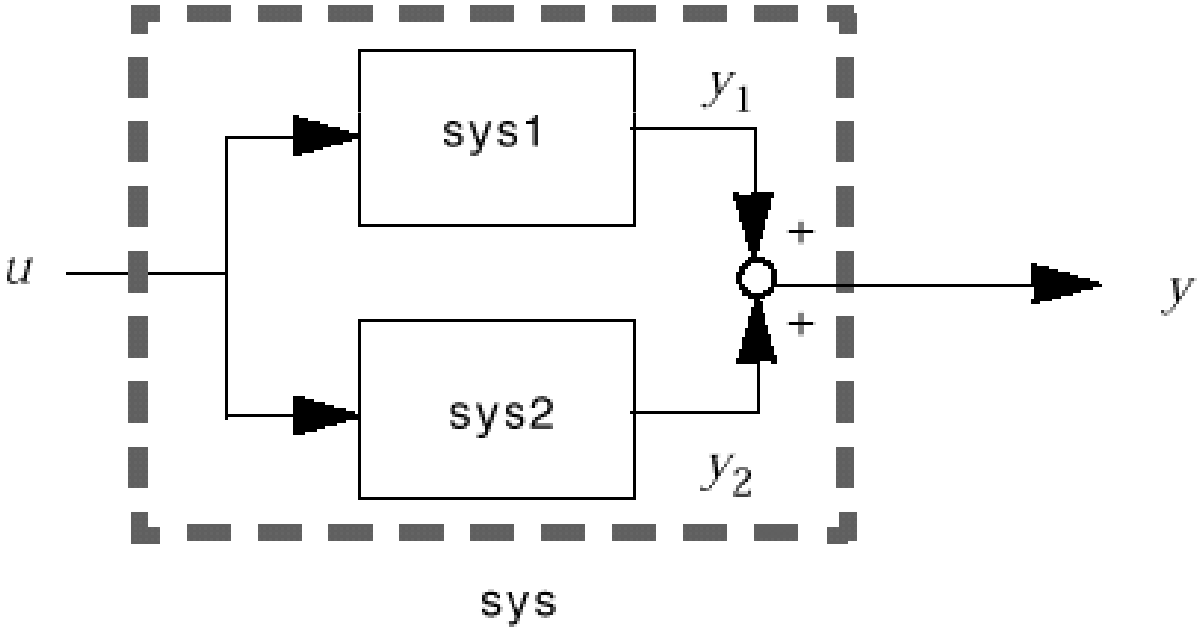
```
»
» h11 = tf(0.5,[1 0]);
» h12 = tf([1 1],[1 2]);
» H = [h11, h12]

Transfer function from input 1 to output:
0.5
---
 s

Transfer function from input 2 to output:
s + 1
-----
s + 2
```

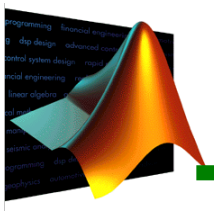
$$H(s) = \begin{bmatrix} \frac{0.5}{s} & \frac{s+1}{s+2} \end{bmatrix}$$



ToolBox de Sistemas de Controle

- Operações em Modelos LTI
  - Adição (Sistemas em Paralelo)

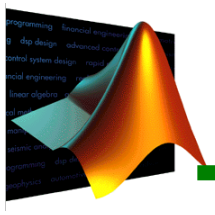




ToolBox de Sistemas de Controle

- Operações em Modelos LTI
  - Adição

```
»
» tf(1, [1 0]) + tf([1 1],[1 2]) % 1/s + (s+1)/(s+2)

Transfer function:
s^2 + 2 s + 2
-------------
  s^2 + 2 s

»
```



ToolBox de Sistemas de Controle

- Adição (Cuidados !!)

```
» tf(0.1,[1 -1], 0.1) + tf(1,[1 0.5],-1)

Transfer function:
  1.1 z - 0.95
-----------------
z^2 - 0.5 z - 0.5

Sampling time: 0.1
```

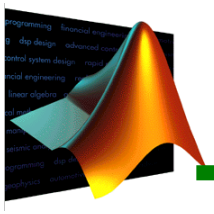
Tempo de amostragem indefinido

```
» tf(0.1,[1 -1], 0.1) + tf(1,[1 0.5],0.1)

Transfer function:
  1.1 z - 0.95
-----------------
z^2 - 0.5 z - 0.5

Sampling time: 0.1
```

Tempos de amostragem idênticos

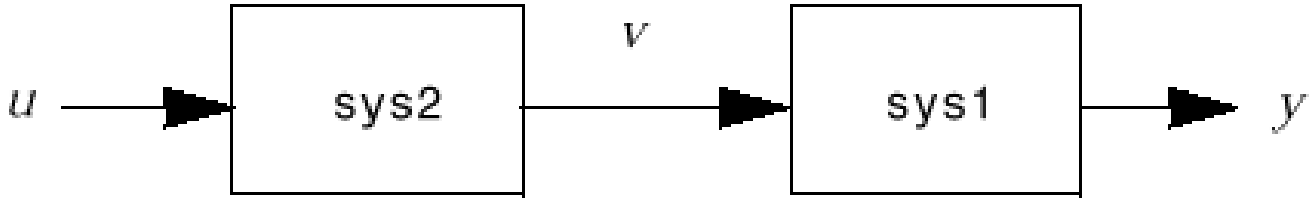
Tempos de amostragem diferentes! ERRO!!!

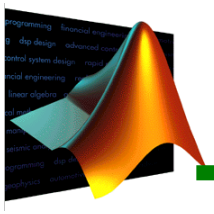
```
» tf(0.1,[1 -1], 0.1) + tf(1,[1 0.5],0.5)
??? Error using ==> plus
In Sys1+Sys2, systems must have identical sampling times.
```



ToolBox de Sistemas de Controle

- Operações em Modelos LTI
  - Multiplicação (Sistemas em Série)

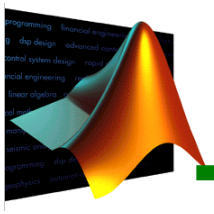




ToolBox de Sistemas de Controle

- Operações em Modelos LTI
  - Multiplicação

```
» 2*tf(1, [1 0])*tf([1 1],[1 2]) % 2*1/s*(s+1)/(s+2)

Transfer function:
 2 s + 2
---------
s^2 + 2 s
```



ToolBox de Sistemas de Controle

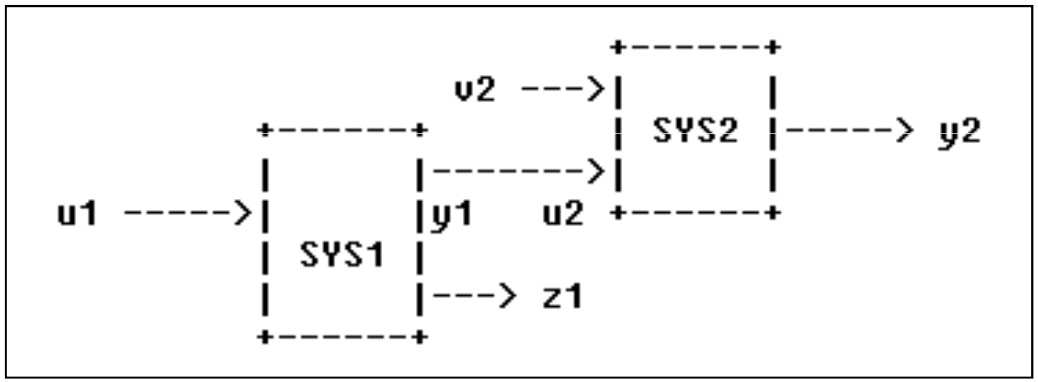
- Conexões entre Sistemas
  - Sejam os sistemas:

```
» sys1=tf(1, [1 2])

Transfer function:
  1
-----
s + 2

» sys2 = tf(1, [1 1 10])

Transfer function:
      1
-------------
s^2 + s + 10
```

ToolBox de Sistemas de Controle

Conexões em Série

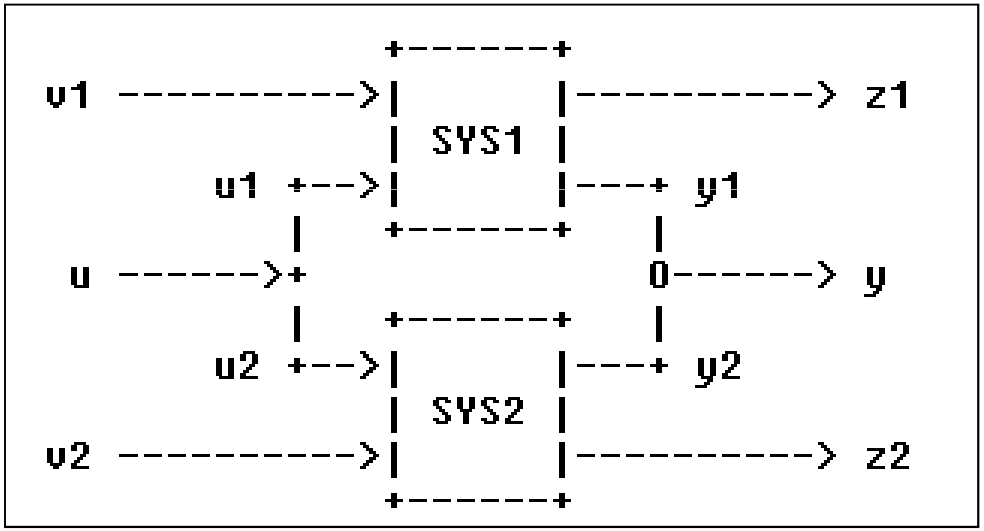
```
» sys = series(sys1, sys2)

Transfer function:
            1
-----------------------
s^3 + 3 s^2 + 12 s + 20

»
»
» sys = sys1*sys2

Transfer function:
            1
-----------------------
s^3 + 3 s^2 + 12 s + 20
```



ToolBox de Sistemas de Controle

Conexões em Paralelo

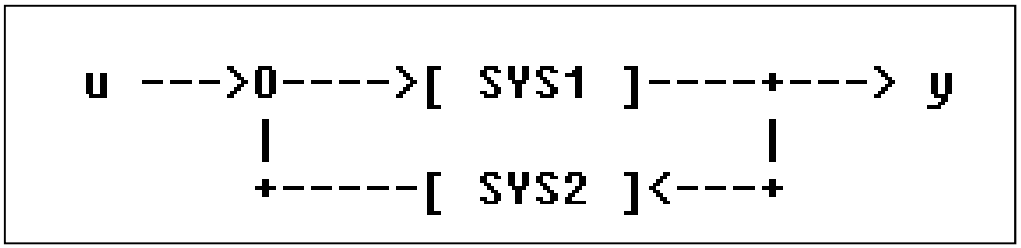
```
>
> sys = parallel(sys1,sys2)

Transfer function:
    s^2 + 2 s + 12
-----------------------
s^3 + 3 s^2 + 12 s + 20

> sys = sys1 + sys2

Transfer function:
    s^2 + 2 s + 12
-----------------------
s^3 + 3 s^2 + 12 s + 20
```

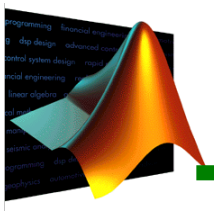


ToolBox de Sistemas de Controle

- Feedback

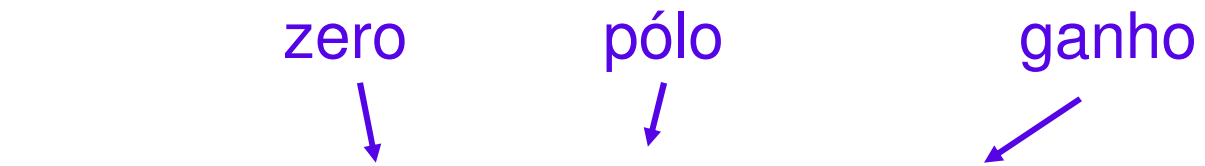
```
» sys = feedback(sys1, sys2)

Transfer function:
     s^2 + s + 10
-----------------------
s^3 + 3 s^2 + 12 s + 21
```





ToolBox de Sistemas de Controle

- Modelo zero/pólo/ganho
- Modelos SISO (Simple Input, Simple Output)

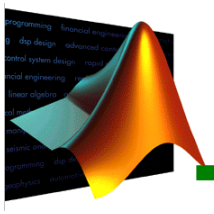



```
» h = zpk(0, [1-i 1+i 2], -2)

Zero/pole/gain:
          -2 s
----------------------
(s-2) (s^2 - 2s + 2)
```

ToolBox de Sistemas de Controle

- Modelos MIMO (Multiple Input, Multiple Output)

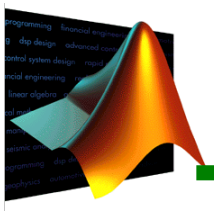
```
» Z={[], -5;[1-i 1+i] []};
» P={0,[-1 -1];[1 2 3], []};
» K=[-1 3; 2 0];
» H = zpk(Z,P,K)

Zero/pole/gain from input 1 to output...
       -1
 #1:   --
       s

       2 (s^2 - 2s + 2)
 #2:   ----------------
       (s-1) (s-2) (s-3)

Zero/pole/gain from input 2 to output...
       3 (s+5)
 #1:   -------
       (s+1)^2

 #2:   0
```



ToolBox de Sistemas de Controle

- Os comandos anteriores criam o modelo de zero-pólo-ganho de duas entradas e duas saídas:

$$H(s) = \begin{bmatrix} \dfrac{-1}{s} & \dfrac{3(s+5)}{(s+1)^2} \\ \dfrac{2(s^2-2s+2)}{(s-1)(s-2)(s-3)} & 0 \end{bmatrix}$$

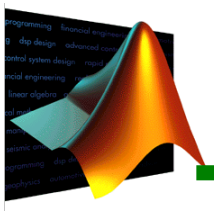


ToolBox de Sistemas de Controle

- Sistemas com múltiplas entradas e múltiplas saídas (modelo de zero/pólo/ganho):

```
»
» zeros = {[], -1};    % Usa [] quando não há zero !
» polos = {0, -2};
» ganhos = [0.5, 1];
» H = zpk(zeros,polos, ganhos)

Zero/pole/gain from input 1 to output:
0.5
---
 s

Zero/pole/gain from input 2 to output:
(s+1)
-----
(s+2)
```

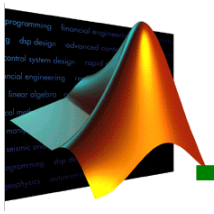
$$H(s) = \begin{bmatrix} \frac{0.5}{s} & \frac{s+1}{s+2} \end{bmatrix}$$



ToolBox de Sistemas de Controle

- Modelo zero/pólo/ganho
- Exemplo de um modelo discreto no tempo:

```
»
» sys = zpk(0.5,[-0.1 0.3],1,0.05)

Zero/pole/gain:
     (z-0.5)
---------------
(z+0.1) (z-0.3)

Sampling time: 0.05
»
```



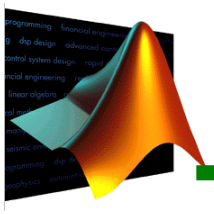
ToolBox de Sistemas de Controle

- Modelos Estado-Espaço
  - Lidam com equações diferenciais para descrever sistemas dinâmicos

$$\frac{dx}{dt} = Ax + Bu$$

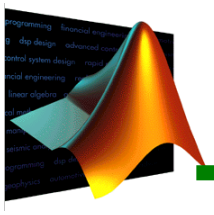
$$y = Cx + Du$$

  - onde x é o vetor de estado e u e y são os vetores de entrada e saída



ToolBox de Sistemas de Controle

- Modelos Estado-Espaço
  - No MatLab, usa-se o comando
    - sys = ss(A,B,C,D)
  - para um modelo com Nx estados, Ny saídas e Nu entradas
    - A é uma matriz Nx por Nx
    - B é uma matriz Nx por Nu
    - C é uma matriz Ny por Nx
    - D é uma matriz Ny por Nu

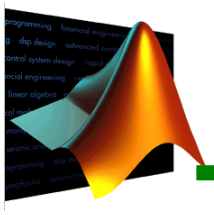


ToolBox de Sistemas de Controle

- Modelos Estado-Espaço
- Exemplo:
  - Considere o seguinte modelo de um motor:

$$\frac{d^2\theta}{dt^2} + 2\frac{d\theta}{dt} + 5\theta = 3I$$

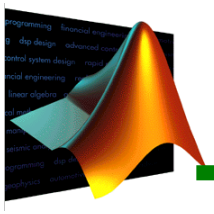
  - onde teta é o deslocamento angular do motor e I é a corrente



ToolBox de Sistemas de Controle

- Exemplo (cont.):
  - A relação entre a corrente de entrada u=I e a velocidade angular $y = d\theta/dt$ é descrita pela equação de estado-espaço:
    - dx/dt = Ax + Bu
    - y = Cx
  - onde

$$x = \begin{bmatrix} \theta \\ \frac{d\theta}{dt} \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

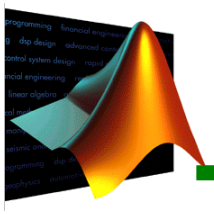


ToolBox de Sistemas de Controle

- Exemplo (cont.):
  - No MatLab, o modelo é especificado como:

```
» sys = ss([0 1;-5 -2], [0;3], [0 1], 0)
```

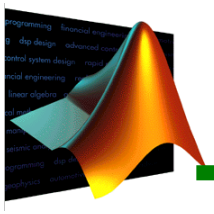
A B C D



ToolBox de Sistemas de Controle

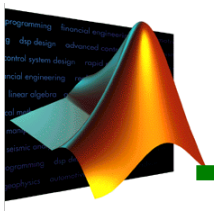
- Exemplo (cont.):
  - Tendo como resposta:

```
a =
                x1           x2
   x1            0      1.00000
   x2     -5.00000     -2.00000

b =
                u1
   x1            0
   x2      3.00000

c =
                x1           x2
   y1            0      1.00000

d =
                u1
   y1            0

Continuous-time system.
```



ToolBox de Sistemas de Controle

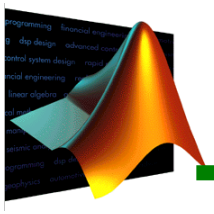
- Modelo de Estado-Espaço
  - drss(N)
    - Cria um modelo aleatório discreto de estado-espaço de N-ésima ordem
    - Exemplo:
      - >> drss(2)
      - >> drss(3)
      - ....



ToolBox de Sistemas de Controle

- drss(N)

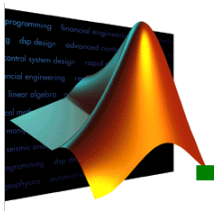
```
» sys = drss(2)

a =
                    x1          x2
          x1      0.06203     0.68257
          x2      0.68257     0.44036

b =
                    u1
          x1      0.01176
          x2      0.89390

c =
                    x1          x2
          y1      0.19914           0

d =
                    u1
          y1           0

Sampling time: unspecified
Discrete-time system.
```



ToolBox de Sistemas de Controle

- filt
  - Especificação de funções de transferência discretas

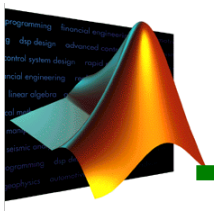
```
» num = 1;
» den = [1 1];
» tf(num,den)

Transfer function:
  1
-----
s + 1

» sys = filt(num, den)

Transfer function:
   1
--------
1 + z^-1

Sampling time: unspecified
```



ToolBox de Sistemas de Controle

- filt
  - Outro exemplo:

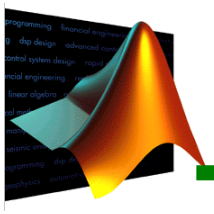
```
..
» num = [1 1];
» den = [1 1 10];
» tf(num, den)

Transfer function:
   s + 1
-----------
s^2 + s + 10

» sys = filt(num, den)

Transfer function:
     1 + z^-1
-----------------
1 + z^-1 + 10 z^-2

Sampling time: unspecified
```



ToolBox de Sistemas de Controle

- Resposta no Tempo e na Freqüência

```
step(sys)          % step response
impulse(sys)       % impulse response
initial(sys,x0)    % undriven response to initial condition
lsim(sys,u,t,x0)   % response to input u

bode(sys)          % Bode plot
nyquist(sys)       % Nyquist plot
nichols(sys)       % Nichols plot
sigma(sys)         % singular value plot
freqresp(sys,w)    % complex frequency response
```

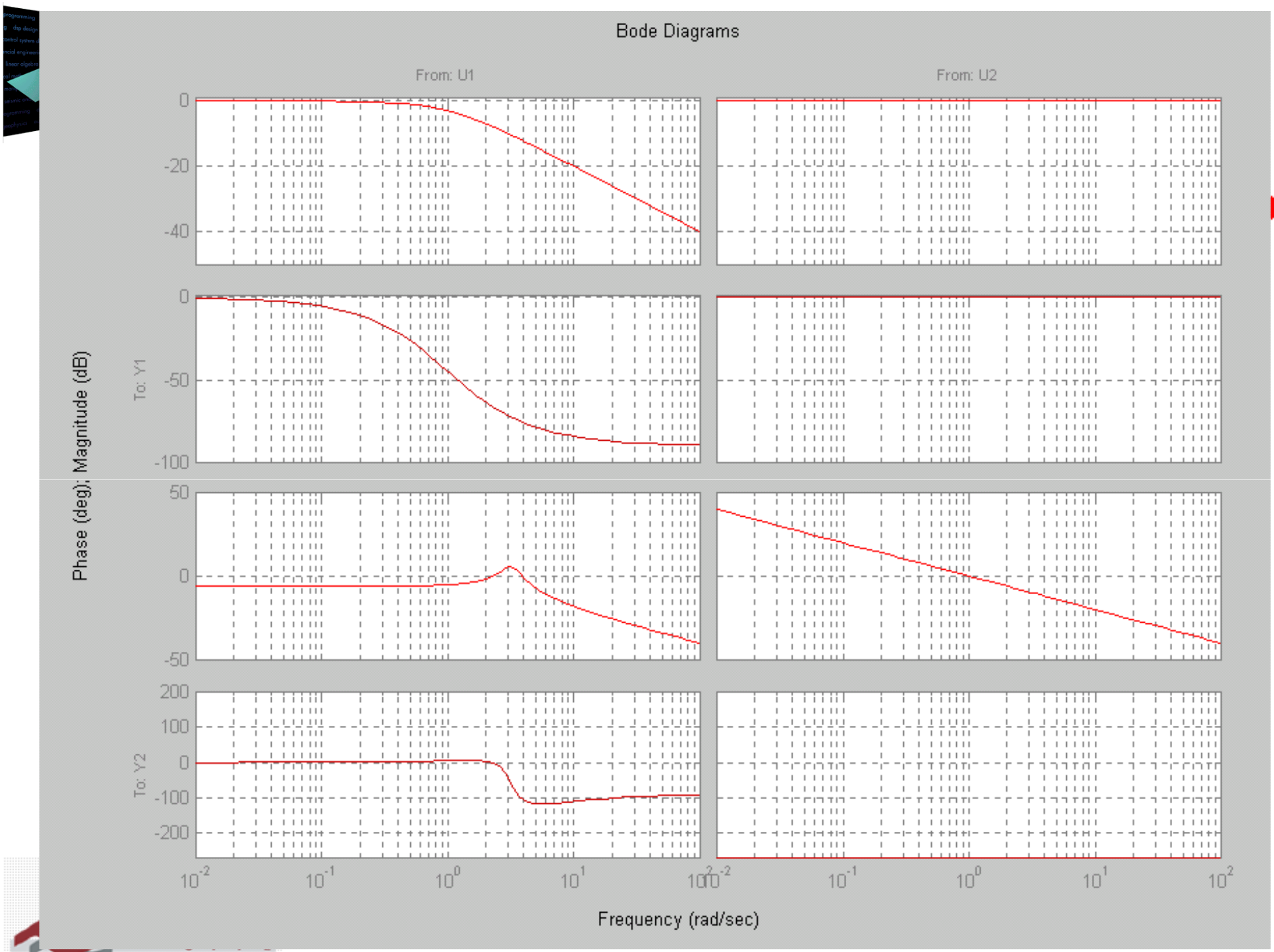


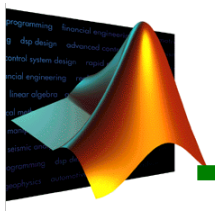
ToolBox de Sistemas de Controle

- Resposta no Tempo e na Freqüência

```
»
» sys = [tf(1,[1 1]) 1; tf([1 5],[1 1 10]) tf(-1, [1 0])]

Transfer function from input 1 to output...
         1
 #1:  -----
      s + 1

         s + 5
 #2:  ------------
      s^2 + s + 10

Transfer function from input 2 to output...
 #1:  1

      -1
 #2:  --
      s
```

```
» bode(sys)
»
```


Bode Diagrams
From: U1
From: U2
To: Y1
To: Y2
Phase (deg); Magnitude (dB)
Frequency (rad/sec)
0
-20
-40
-50
-100
50
200
100
-200
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

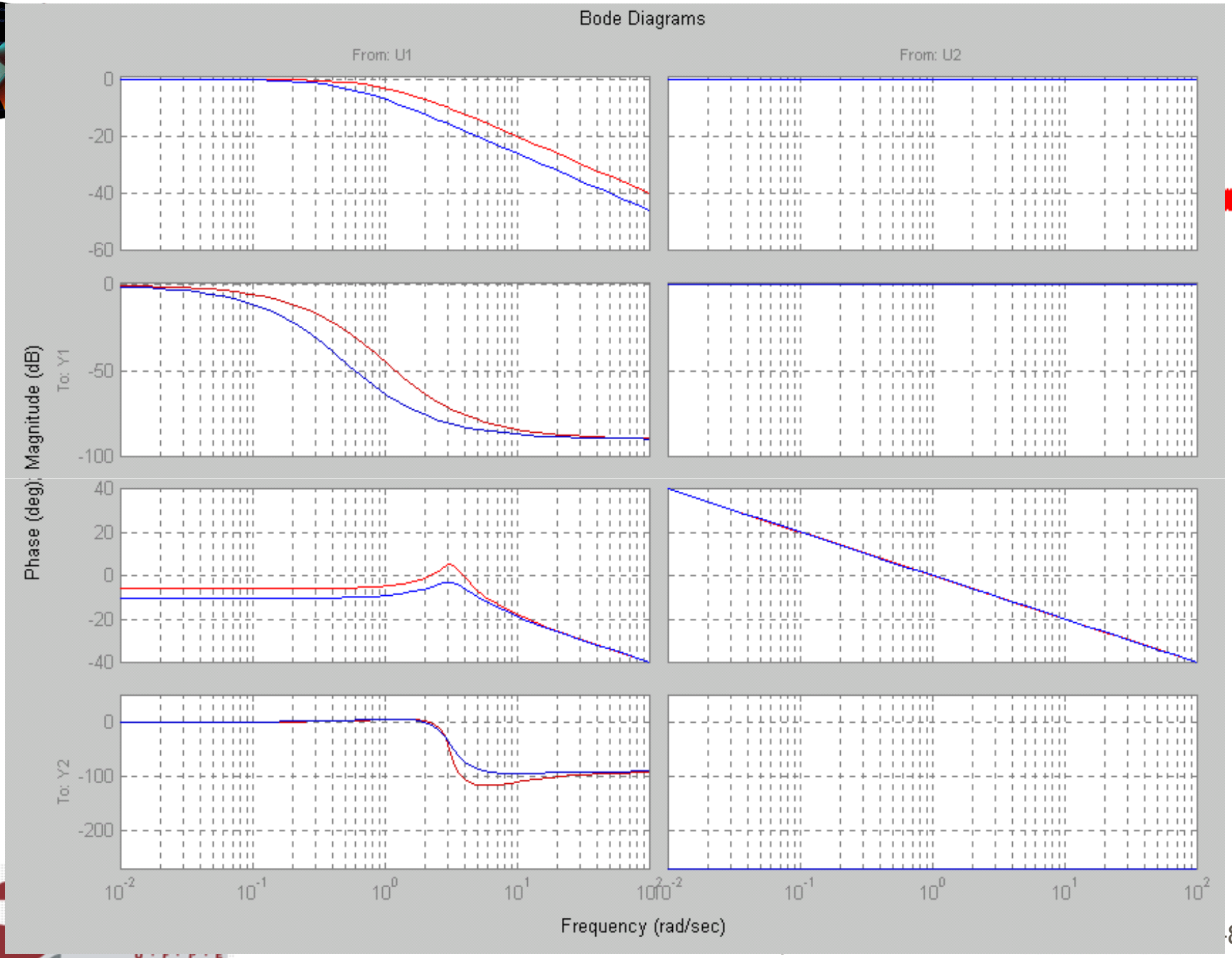


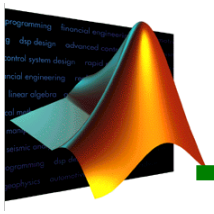
ToolBox de Sistemas de Controle

- Para sobrepor e comparar as respostas de vários sistemas LTI, basta escrever:
  - >> bode (sys1,sys2,sys3,....)
  - Exemplo:

```
» sys2 = [tf(1,[2 1]) 1; tf([1 3],[1 2 10]) tf(-1, [1 0])];
»
» bode(sys,sys2)
»
```

  - ou bode (sys1, 'r', sys2, 'b')
    - Definindo a cor de cada gráfico


Bode Diagrams
From: U1
From: U2
Magnitude (dB)
Phase (deg)
To: Y1
To: Y2
0
-20
-40
-60
-50
-100
40
20
-200
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Frequency (rad/sec)



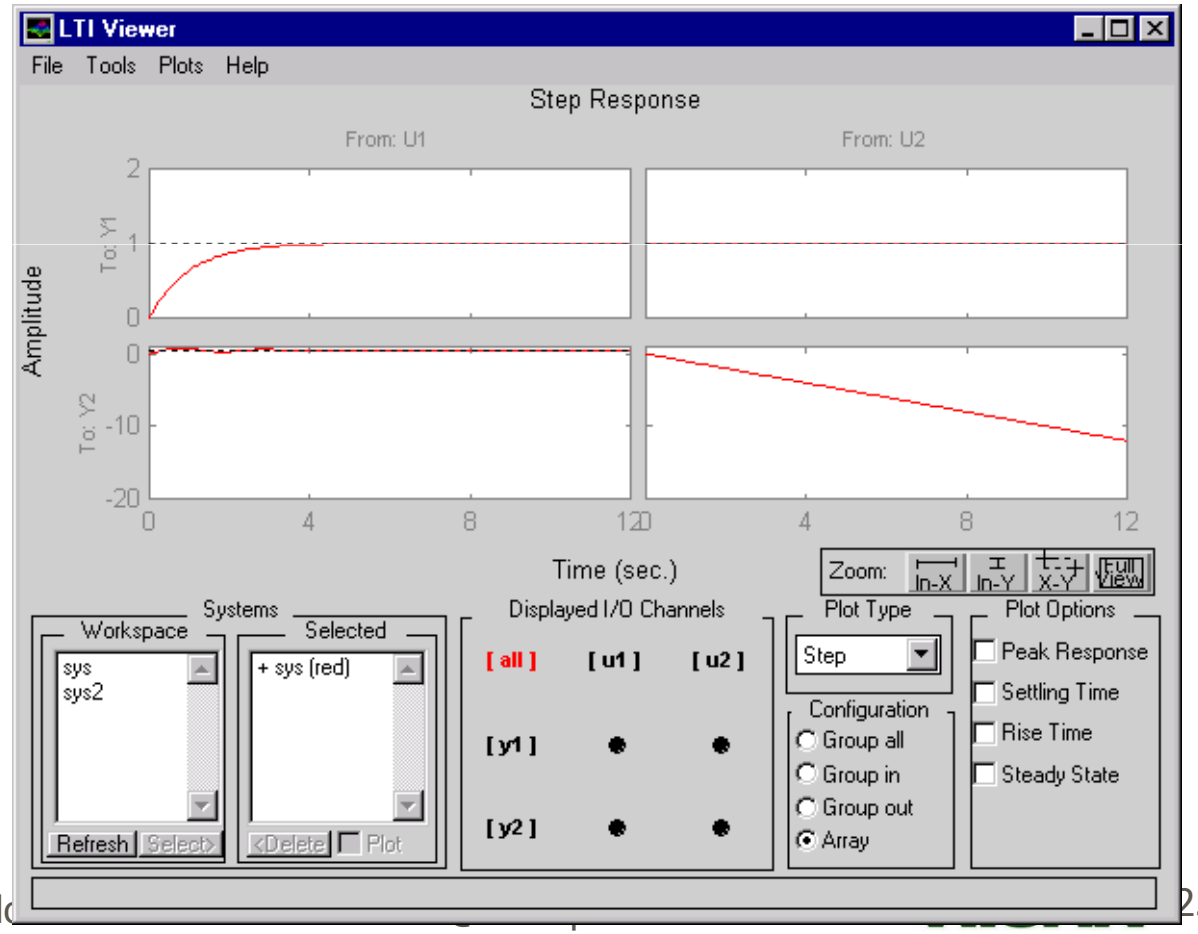
ToolBox de Sistemas de Controle

- LTI Viewer (Visualizador)
  - Interface gráfica para visualização e manipulação de gráficos de modelos LTI
  - Uso:
    - ltiview(tipo_de_grafico, sys1,sys2,..)
    - Tipo de Grafico: step, impulse, initial, lsim, pzmap, bode, nyquist, nichols, sigma

ToolBox de Sistemas de Controle

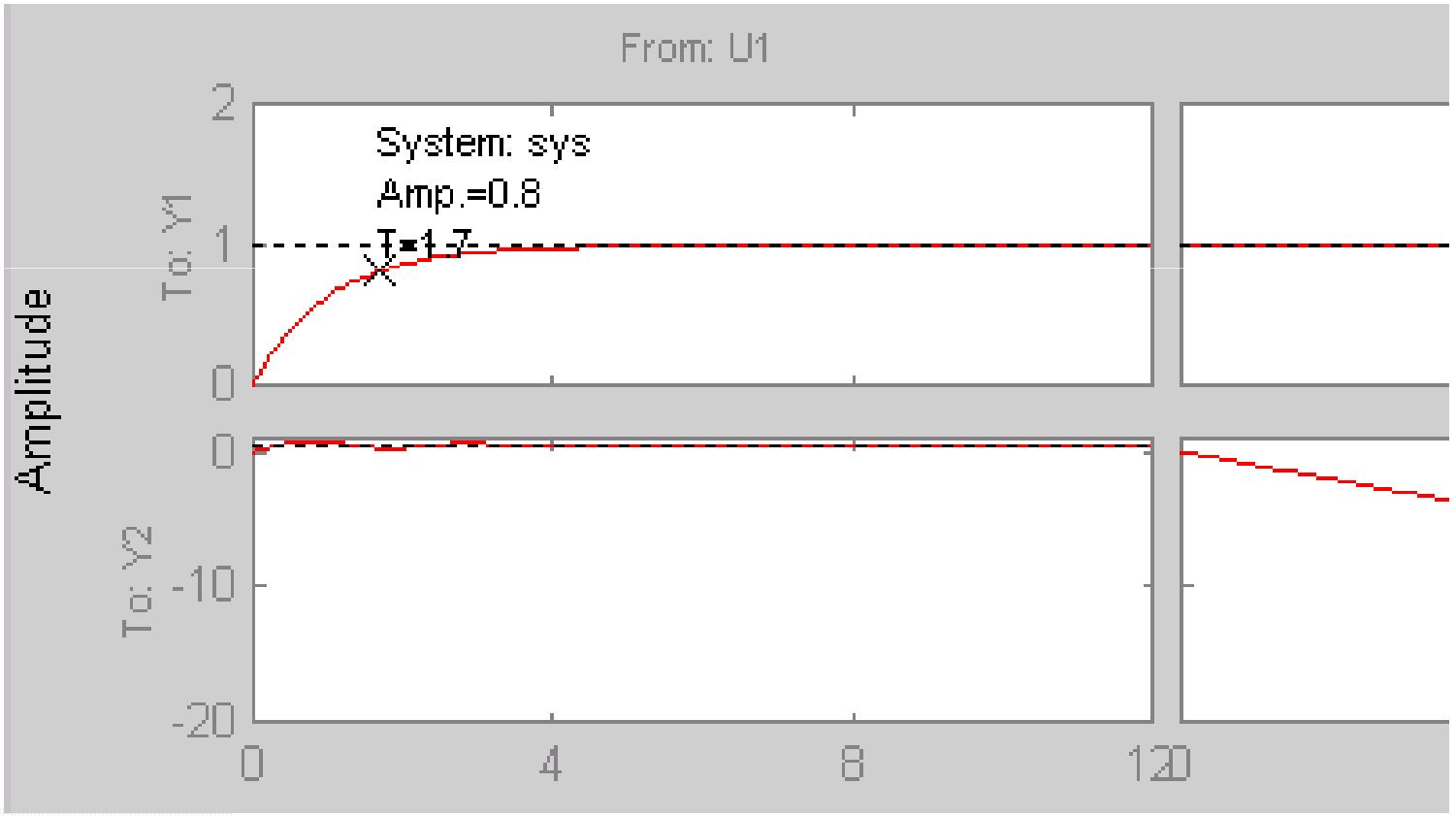
- Exemplo: Para o sistema definido por sys:

```
» ltiview('step', sys)
»
```



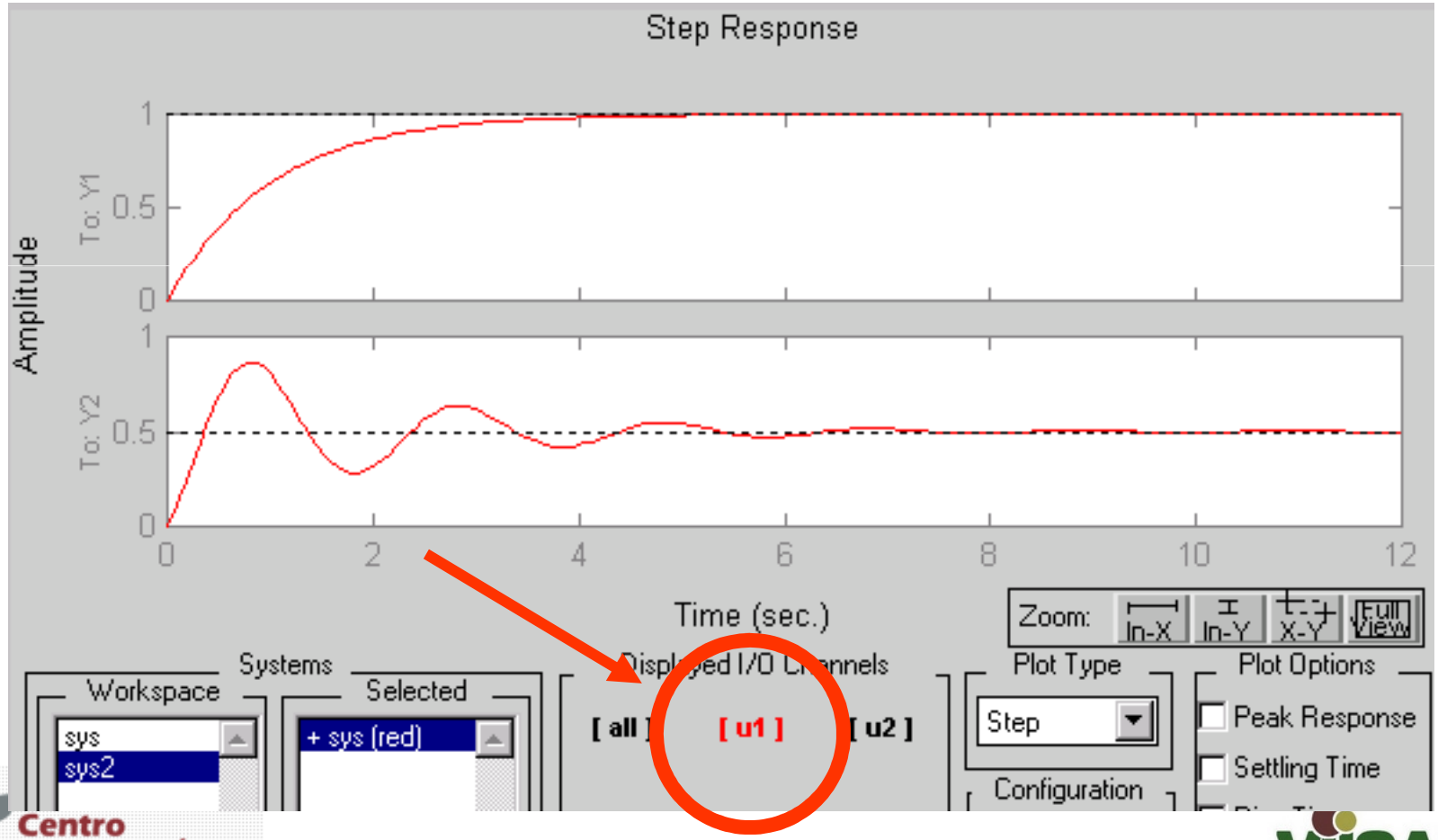
ToolBox de Sistemas de Controle

- Clicando sobre o gráfico na janela....



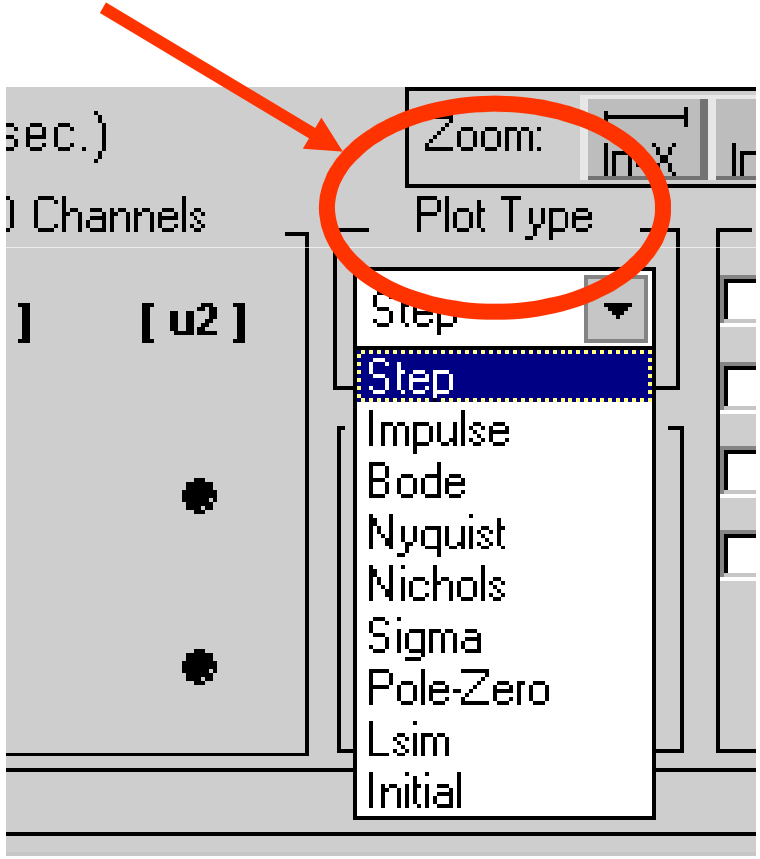
ToolBox de Sistemas de Controle

- Destacando um gráfico....



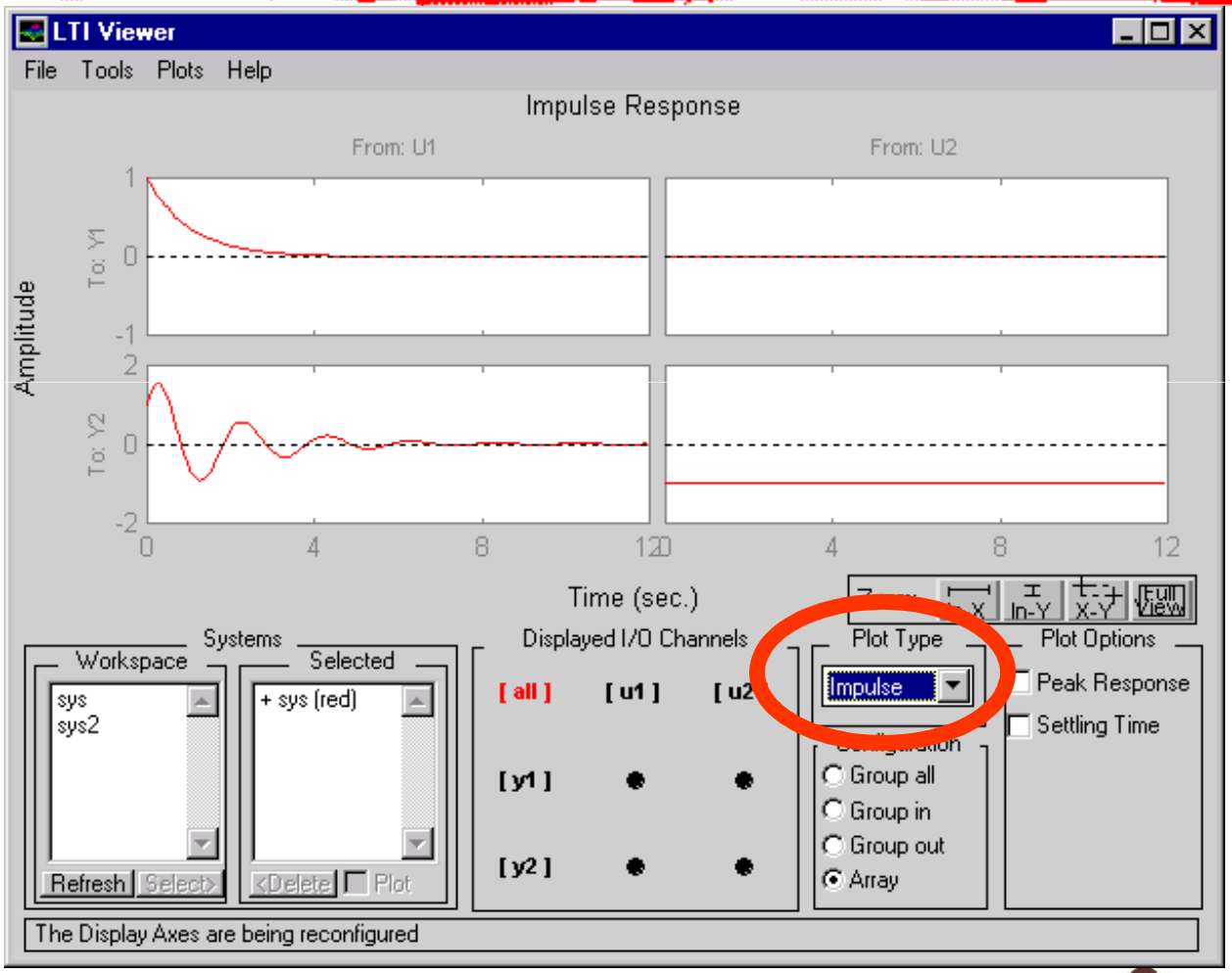
ToolBox de Sistemas de Controle

- Mudando o tipo de gráfico....



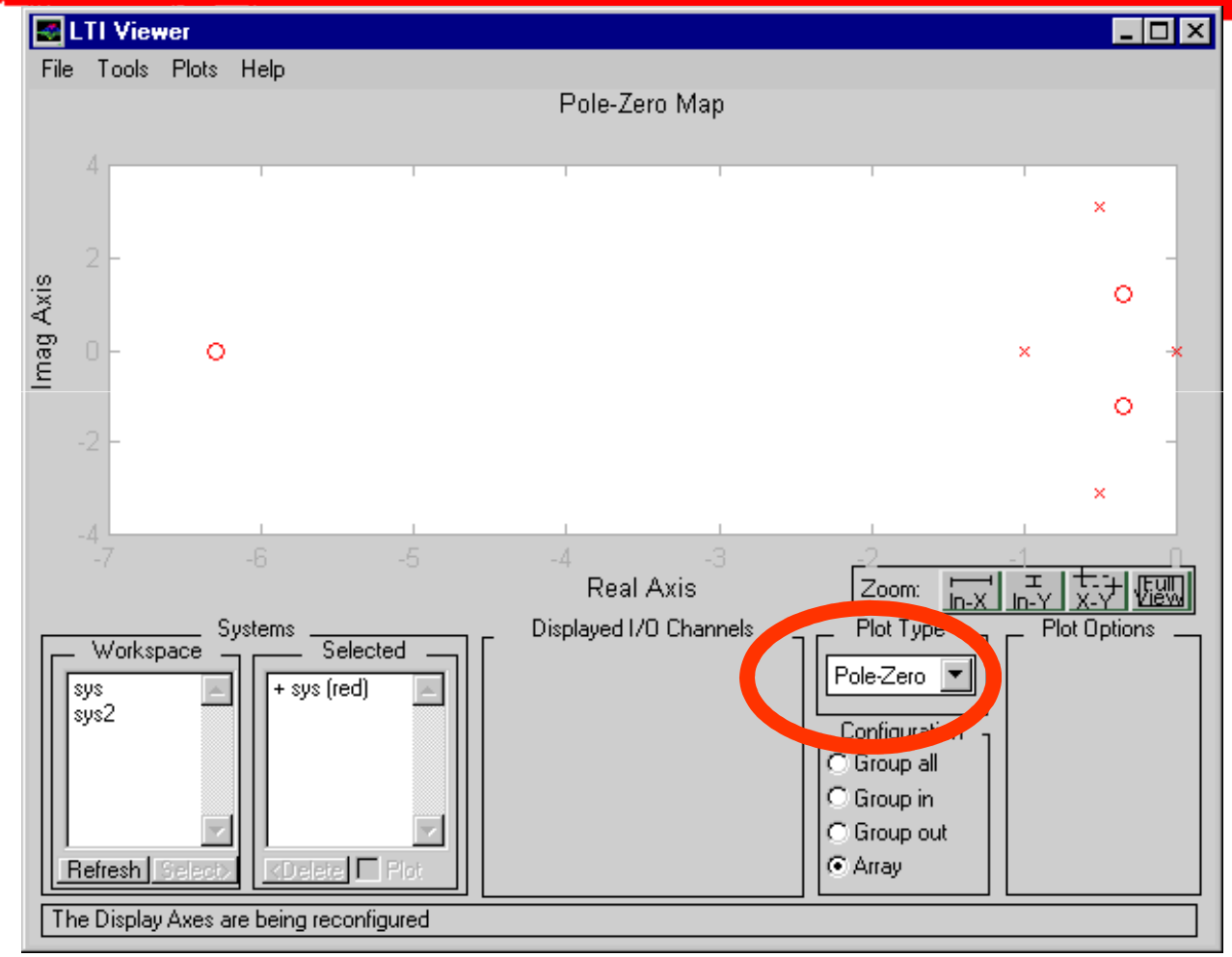
ToolBox de Sistemas de Controle

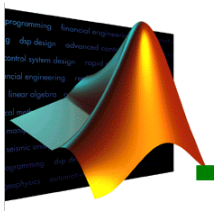
Impulse



ToolBox de Sistemas de Controle

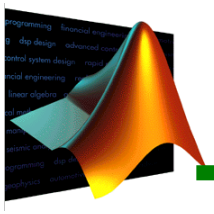
Pólos e Zeros





ToolBox de Sistemas de Controle

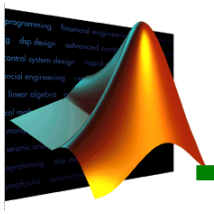
- Recuperação de Dados
  - As funções tf, zpk, ss e frd empacotam os dados e tempo de amostragem em um único objeto
  - É possível fazer o processo inverso e recuperar os dados que geraram um objeto:
    - [num, den, ts] = tfdata(sys)
    - [z,p,k,ts] = zpkdata(sys)
    - [a,b,c,d]=ssdata(sys)



ToolBox de Sistemas de Controle

- Recuperação de Dados
  - Exemplo:

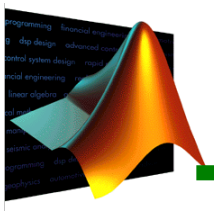
```
» sys = [tf(1,[2 1]) 1; tf([1 3],[1 2 10]) tf(-1, [1 0])]

Transfer function from input 1 to output...
          1
 #1:  -------
      2 s + 1

          s + 3
 #2:  --------------
      s^2 + 2 s + 10

Transfer function from input 2 to output...
 #1:  1

      -1
 #2:  --
      s
```



ToolBox de Sistemas de Controle

- Recuperação de Dados
  - Exemplo: (cont)

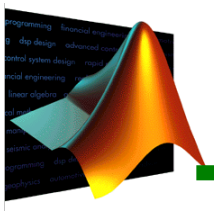
```
» [num,den,ts]=tfdata(sys)

num =

    [1x2 double]    [          1]
    [1x3 double]    [1x2 double]


den =

    [1x2 double]    [          1]
    [1x3 double]    [1x2 double]


ts =

     0
```



ToolBox de Sistemas de Controle

- Recuperação de Dados
  - Exemplo (cont): Para acessar os dados...

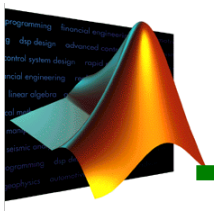
```
» num{1,1}, den{1,1}

ans =

     0     1


ans =

     2     1
```



ToolBox de Sistemas de Controle

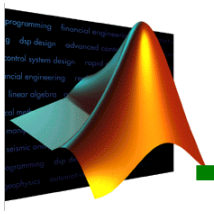
- Recuperação de Dados
  - Exemplo (cont): O mesmo pode ser conseguido com...

```
» sys.num{1,1}

ans =

     0     1

» sys.den{1,1}

ans =

     2     1
```



ToolBox de Sistemas de Controle

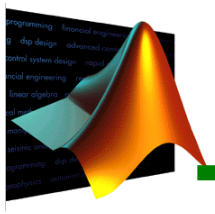
- Conversão de Modelos
  - sys = tf(sys) % Conversão para TF
  - sys = zpk(sys) % Conversão para ZPK
  - sys = ss(sys) % Conversão para Estado-Espaço



ToolBox de Sistemas de Controle

- Conversão de Modelos
  - Exemplo:

```
» sys = [tf(1,[2 1]) 1; tf([1 3],[1 2 10]) tf(-1, [1 0])]

Transfer function from input 1 to output...
          1
 #1:  -------
      2 s + 1

          s + 3
 #2:  --------------
      s^2 + 2 s + 10

Transfer function from input 2 to output...
 #1:  1

      -1
 #2:  --
      s
```

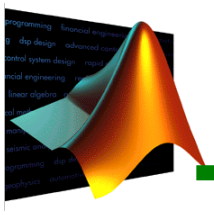


ToolBox de Sistemas de Controle

- Conversão de Modelos
  - Exemplo:

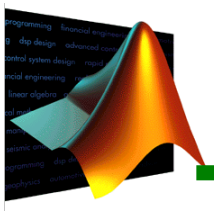
Converte de TF para ZPK

```
»
» sys2= zpk(sys)

Zero/pole/gain from input 1 to output...
          0.5
 #1:  -------
      (s+0.5)

            (s+3)
 #2:  ---------------
      (s^2 + 2s + 10)

Zero/pole/gain from input 2 to output...
 #1:  1

      -1
 #2:  --
      s
```



ToolBox de Sistemas de Controle

- Conversão de Modelos
- Cuidados!!
  - A conversão nem sempre é precisa!
  - Pode haver perda por arredondamento de alguns valores, principalmente na conversão para TF

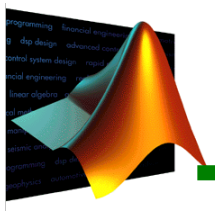


ToolBox de Sistemas de Controle

- Conversão AD/DA

```
»
» sys1=tf(1, [1 2])

Transfer function:
  1
-----
s + 2

» sys1d = c2d(sys1,0.1)

Transfer function:
 0.09063
----------
z - 0.8187

Sampling time: 0.1
»
```

Conversão Contínuo-Discreto com período de amostragem de 0.1



ToolBox de Sistemas de Controle

- Conversão AD/DA

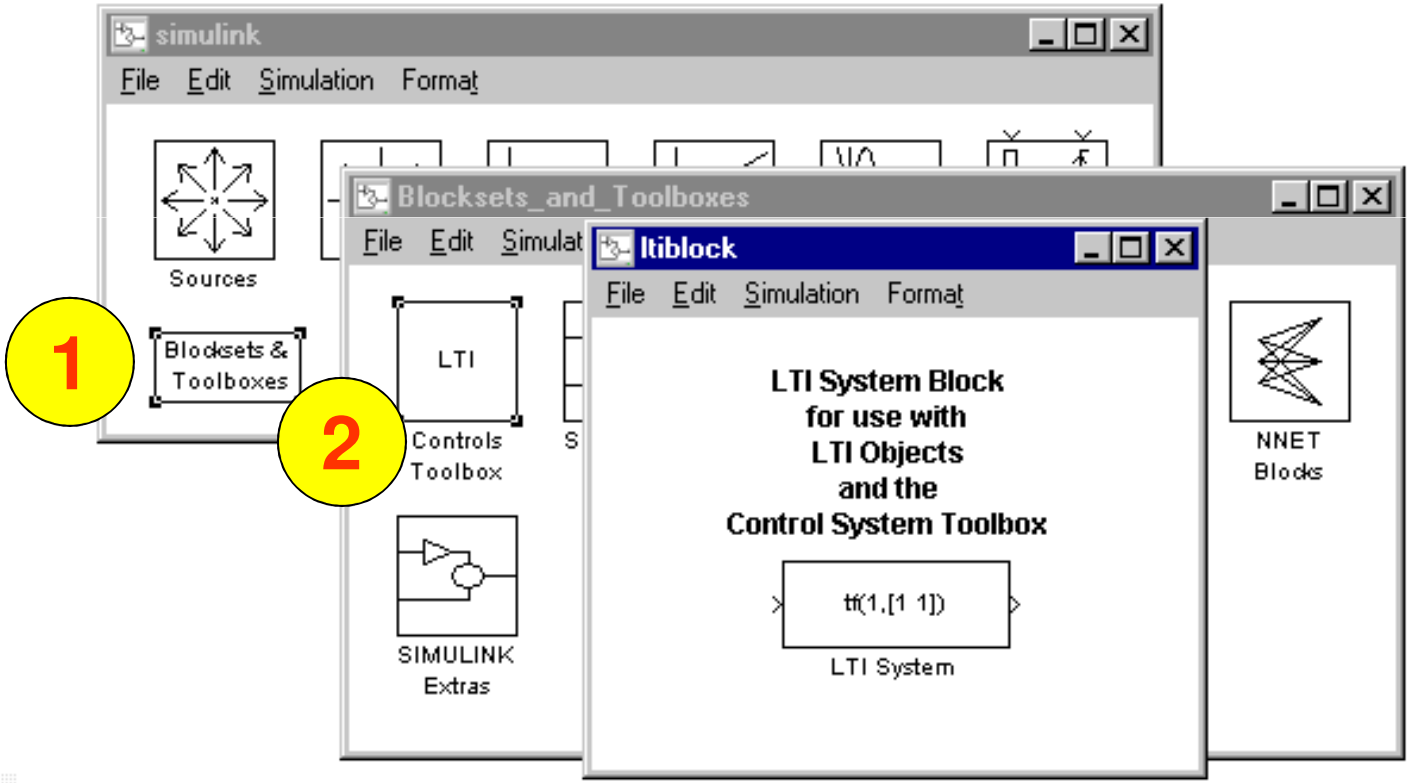
```
..
» sys2 = d2c(sys1d)

Transfer function:
   1
-----
s + 2
```

Conversão Discreto-Contínuo

Igual a sys1!!

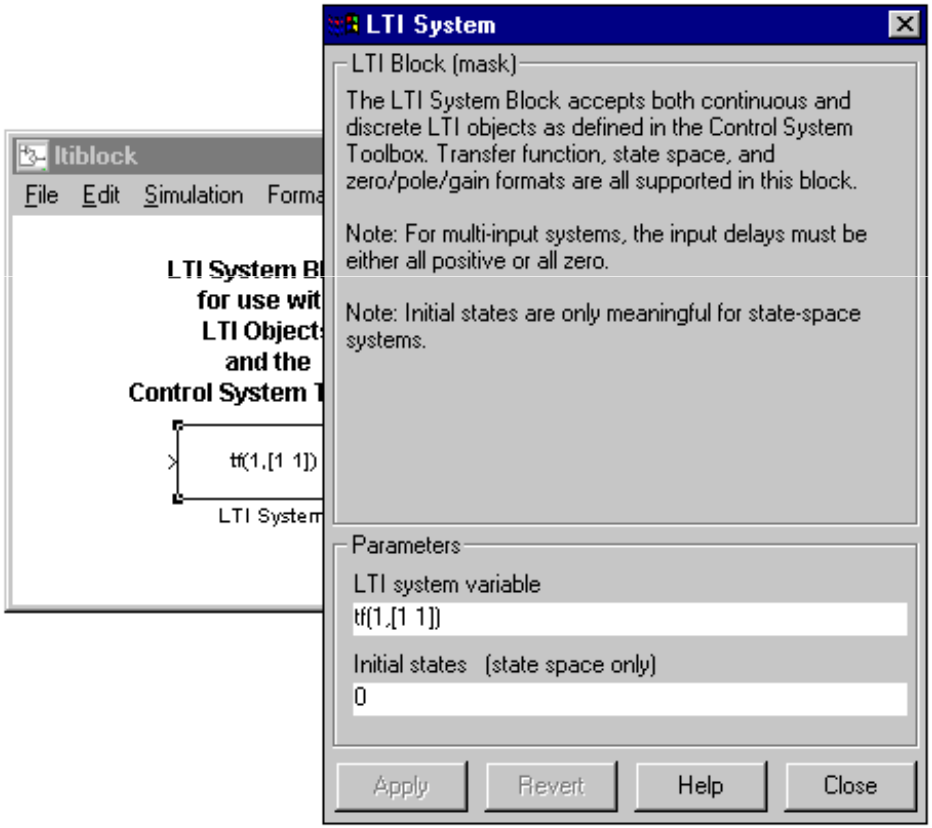
ToolBox de Sistemas de Controle

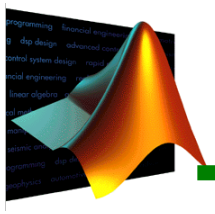
- Criando blocos no Simulink



ToolBox de Sistemas de Controle

- Clicando duas vezes em ltiblock...





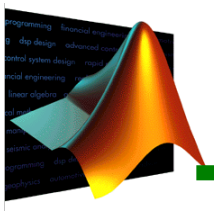
ToolBox de Sistemas de Controle

- Discretização de Sistemas

Função c2d: Contínuo para Discreto

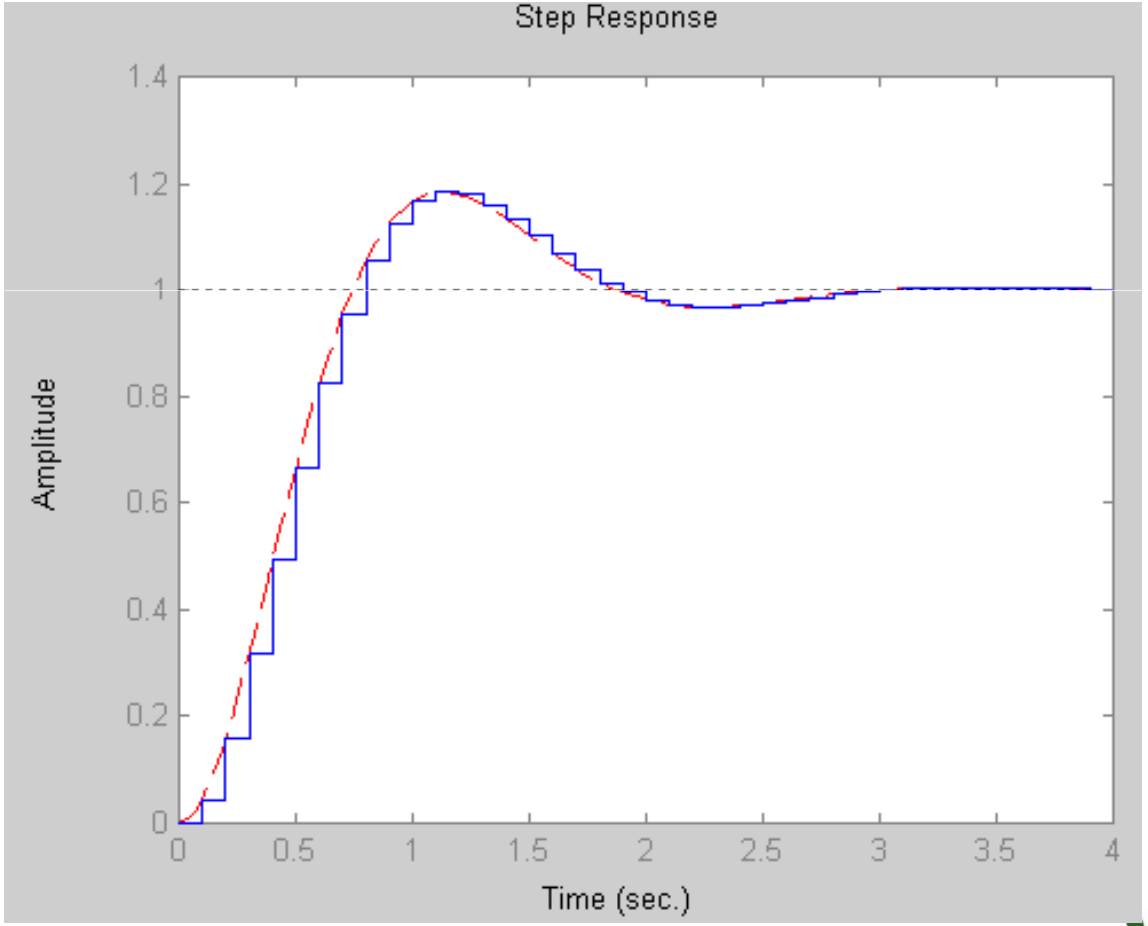
Plotagem

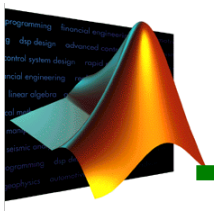
```
» h = tf(10, [1 3 10])

Transfer function:
      10
--------------
s^2 + 3 s + 10

» hd = c2d(h,0.1)

Transfer function:
 0.04498 z + 0.04069
-----------------------
z^2 - 1.655 z + 0.7408

Sampling time: 0.1
» step(h, '--', hd, 'b-')
```



ToolBox de Sistemas de Controle

- Discretização de Sistemas





ToolBox de Sistemas de Controle

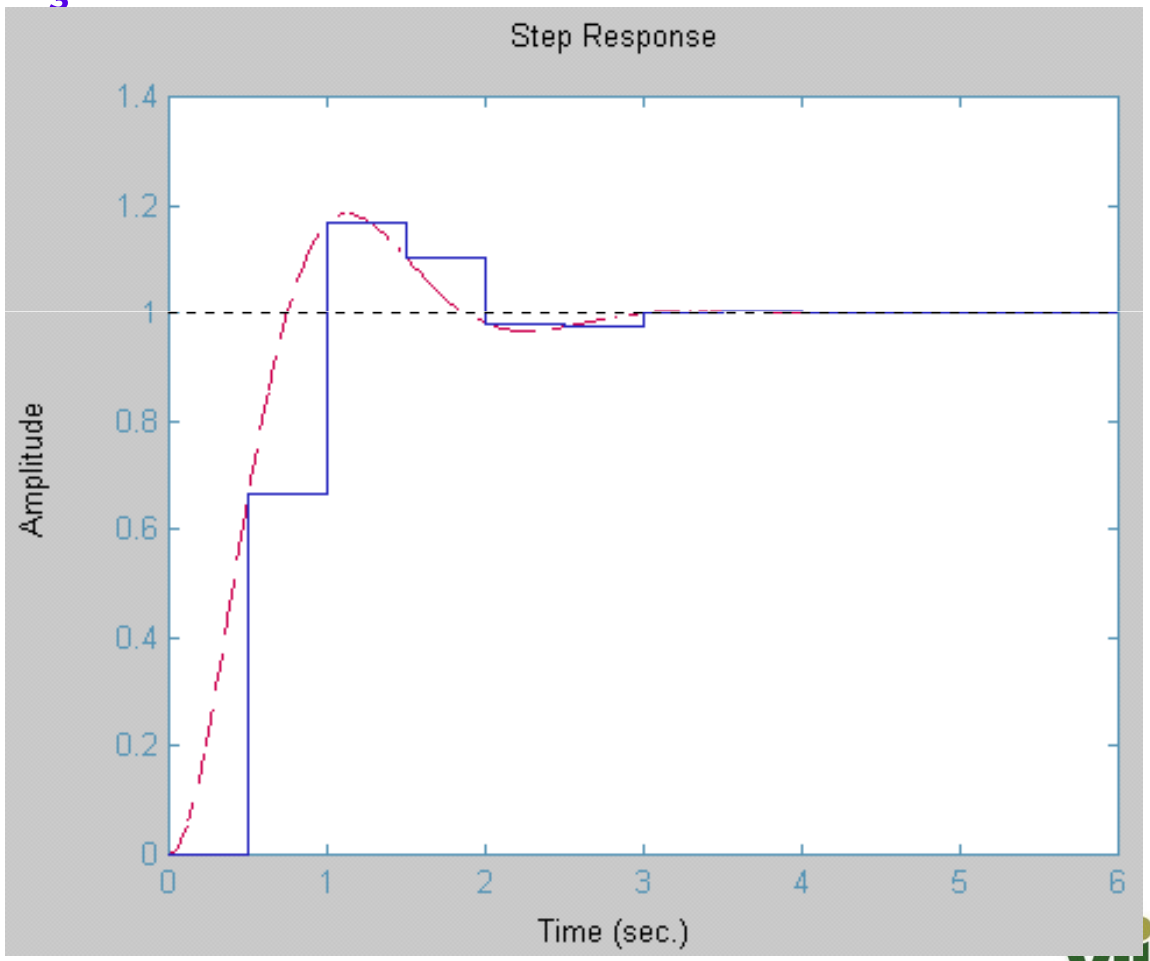
- Discretização de Sistemas
  - Cuidados com a taxa de amostragem...

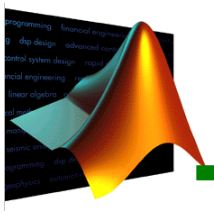
```
» h = tf(10,[1 3 10])

Transfer function:
      10
--------------
s^2 + 3 s + 10

» hd = c2d(h, 0.5)

Transfer function:
   0.6655 z + 0.3896
-----------------------
z^2 - 0.1681 z + 0.2231

Sampling time: 0.5
» step (h, '--', hd, 'b-')
```

ToolBox de Sistemas de Controle

- Discretização de Sistemas

Baixa taxa de Amostragem !

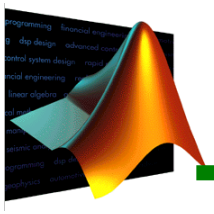




ToolBox de Sistemas de Controle

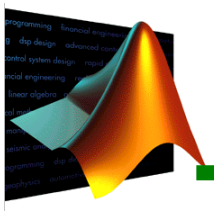
- Ferramentas para Análise de Modelos
  - Suponha o seguinte sistema:

```
» H = tf({1 [1 -1]},{[1 0.1] [1 2 10]})

Transfer function from input 1 to output:
   1
-------
s + 0.1

Transfer function from input 2 to output:
    s - 1
-------------
s^2 + 2 s + 10
```



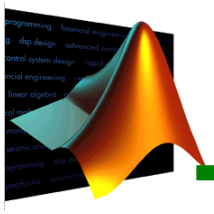
ToolBox de Sistemas de Controle

- Ferramentas para Análise de Modelos:
- Comando *class*
  - Fornece o tipo do modelo
    - >> class(H)
    - ans = tf
- Comando *size*
  - Fornece as dimensões de entrada e saída
    - >> size(H)
    - Transfer function with 2 input(s) and 1 output(s).



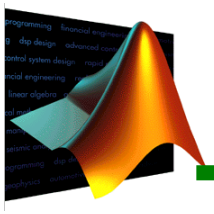
ToolBox de Sistemas de Controle

- Ferramentas para Análise de Modelos:
- Comando *size*
  - >> [ny, nu] = size(H)
  - ny = 1 % número de saídas
  - nu = 2 % número de entradas



ToolBox de Sistemas de Controle

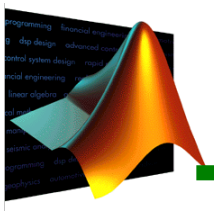
- Ferramentas para Análise de Modelos:
- Comandos Lógicos:
  - isempty (H)
    - Verdadeiro se o modelo está vazio
  - isct (H)
    - Verdadeiro se o modelo é contínuo
  - isdt (H)
    - Verdadeiro se o modelo é discreto



ToolBox de Sistemas de Controle

- Ferramentas para Análise de Modelos:

```
» H = tf({1 [1 -1]},{[1 0.1] [1 2 10]})

Transfer function from input 1 to output:
   1
-------
s + 0.1

Transfer function from input 2 to output:
    s - 1
--------------
s^2 + 2 s + 10
```

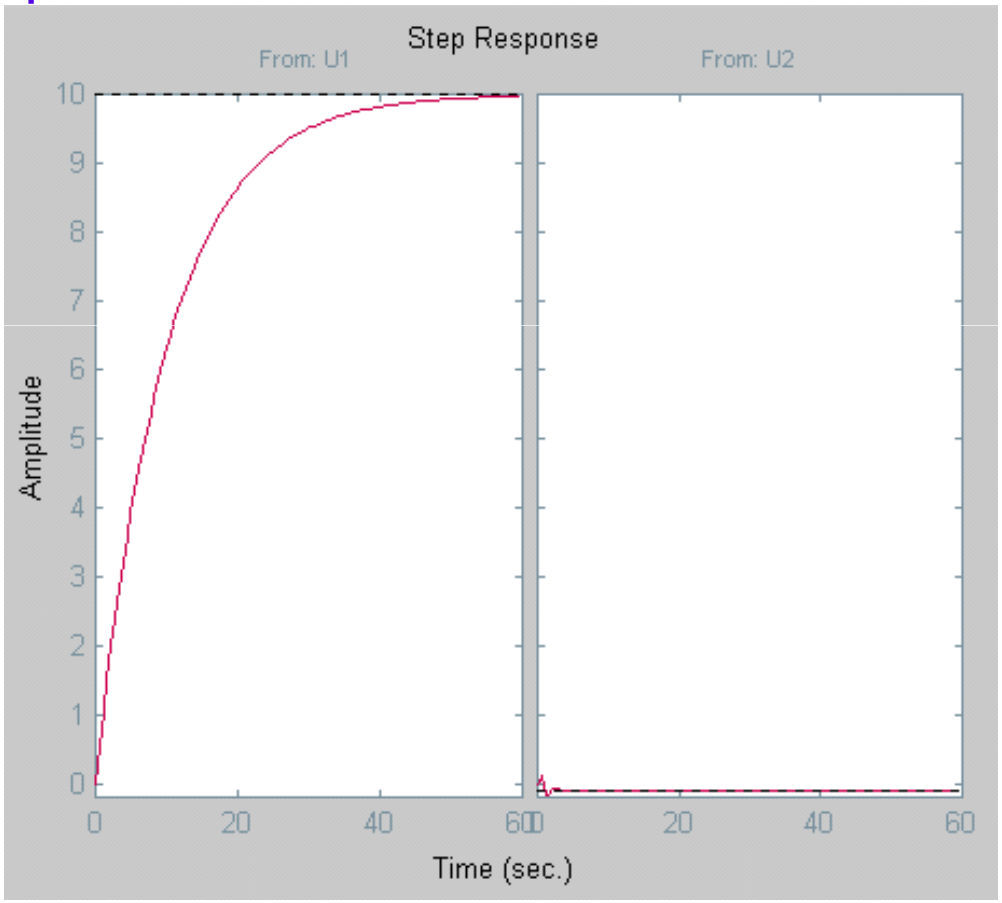


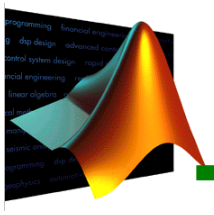
ToolBox de Sistemas de Controle

```
» pole(H)

ans =

   -1.0000+ 3.0000i
   -1.0000- 3.0000i
   -0.1000

» tzero(H)

ans =

    Empty matrix: 0-by-1

» dcgain(H)

ans =

   10.0000   -0.1000
```

ToolBox de Sistemas de Controle

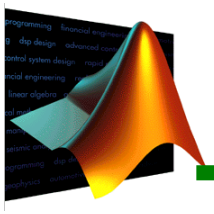
- Respostas no Tempo
  - >> step(H)





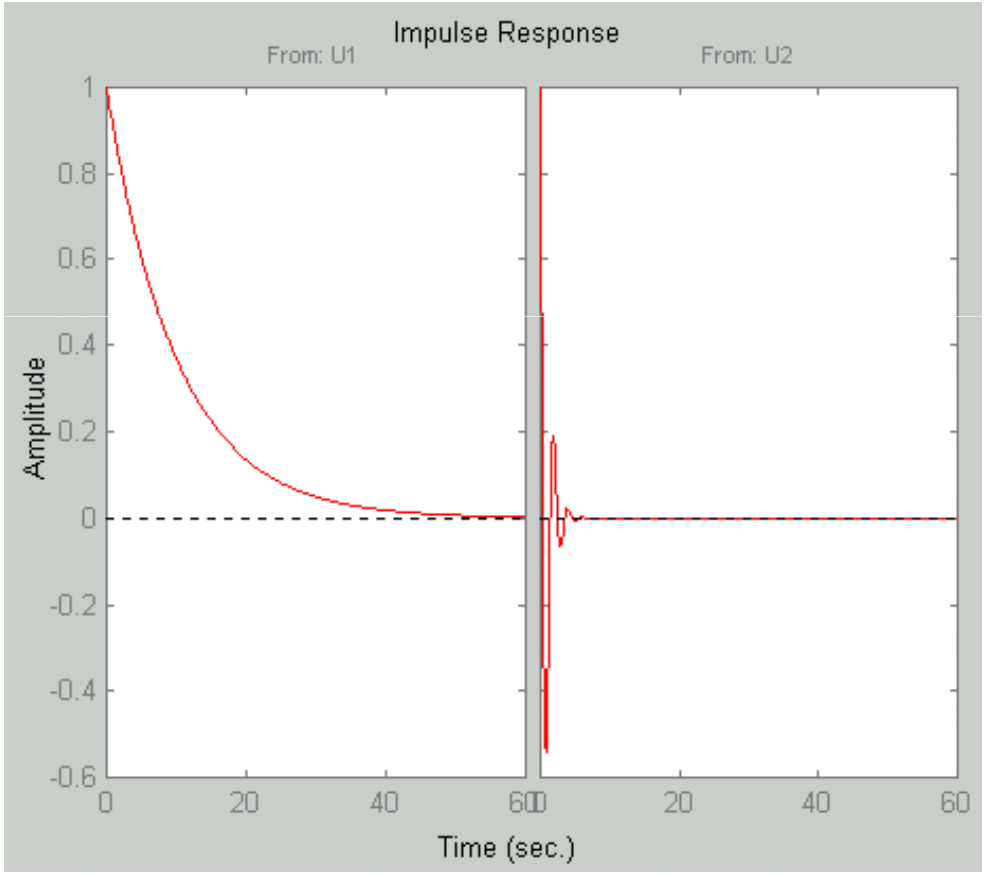
ToolBox de Sistemas de Controle

- Respostas no Tempo
  - Simula de 0 a 10 segundos
    - >> step(H,10)
  - Amostras espaçadas de 0.01 segundo
    - >> t=0:0.01:10
    - >> step(H, t)



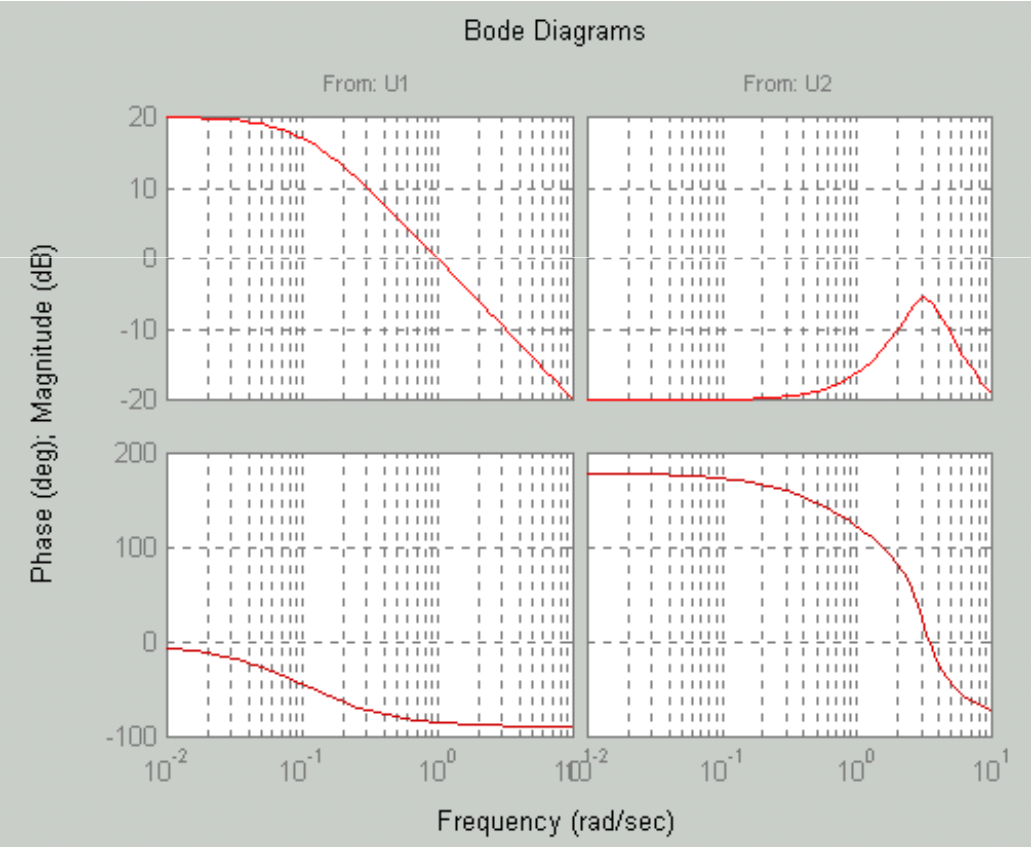
ToolBox de Sistemas de Controle

- Respostas no Tempo
  - >> impulse(H)



ToolBox de Sistemas de Controle

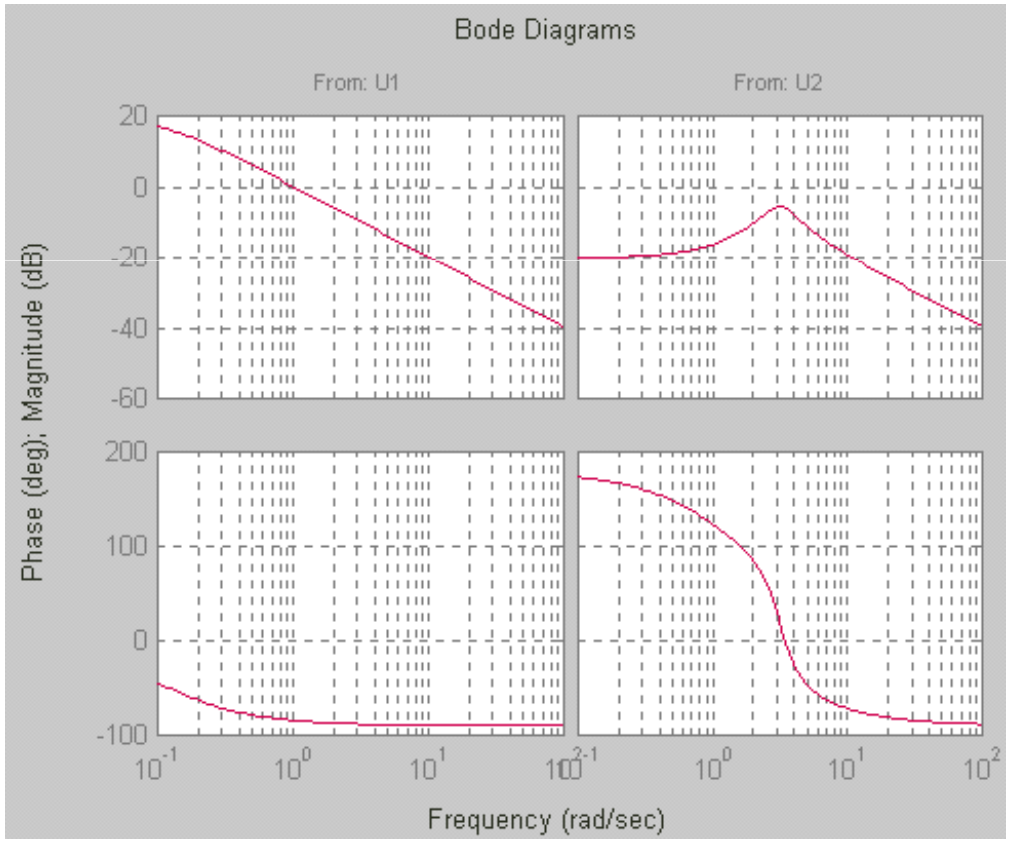
- Resposta em Freqüência
  - Diagrama de Bode: bode(H)

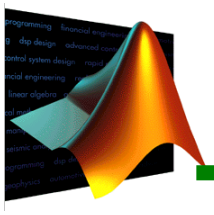


ToolBox de Sistemas de Controle

- Resposta em Freqüência
  - Diagrama de Bode em escala logarítmica:

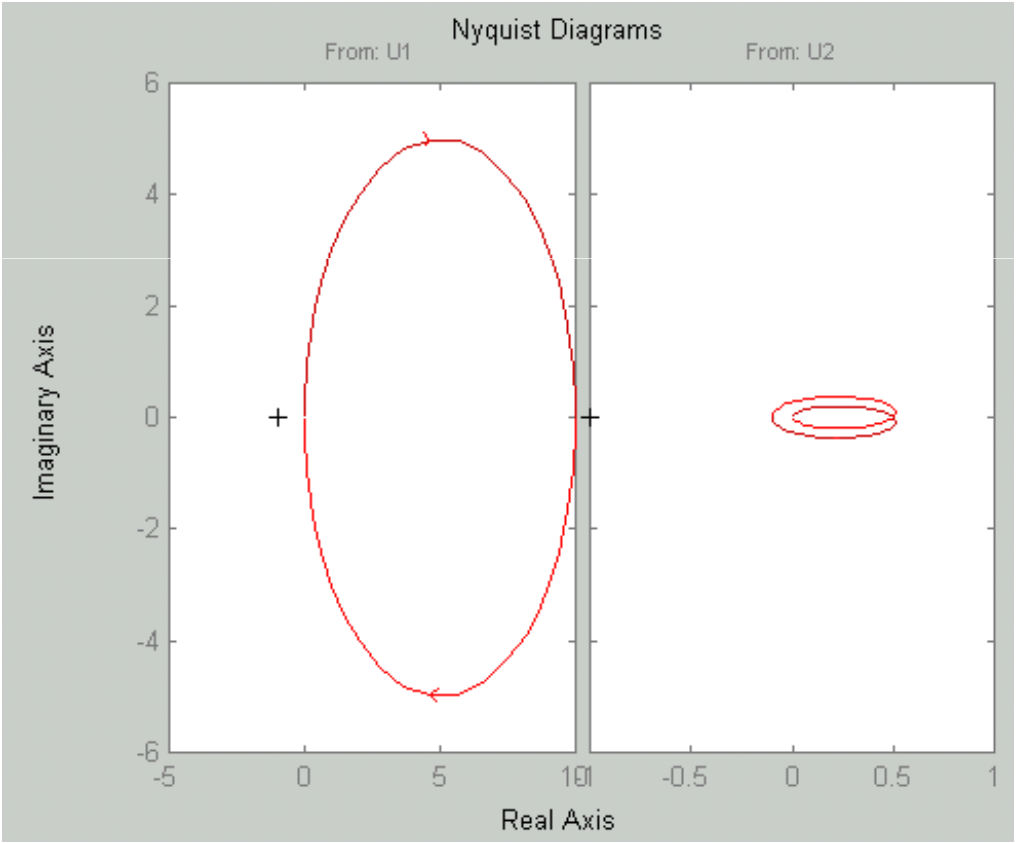
```
» w=logspace(-1,2,100);
» bode(H,w)
»
```

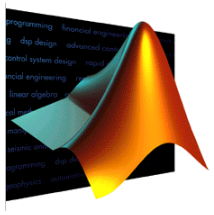




ToolBox de Sistemas de Controle

- Resposta em Freqüência
  - Diagrama de Nyquist: >> nyquist(H)





A Seguir....

- Modelagem no Domínio da Frequência