Curso de MATLAB
Análise de Sinais e Sistemas

Fabricio Breve
fabricio@dc.ufscar.br

Visão Geral

- MATrix LABoratory
- Ambiente interativo para computação envolvendo matrizes
- Desenvolvido no início da década de 80 por Cleve Moler, no Departamento de Ciência da Computação da Universidade do Novo México, EUA
- Versões mais recentes (4.0 em diante) foram desenvolvidas pela MathWorksInc., que detêm os direitos autorais destas implementações
- Multiplataforma:
  - Windows
  - Linux / Unix
  - MacOS
  - Solaris

Uso do MATLAB

- Matemática e Computação
- Desenvolvimento de Algoritmos
- Aquisição de Dados
- Modelagem, Simulação e Prototipação
- Análise, Exploração e Visualização de Gráficos Científicos e de Engenharia
- Desenvolvimento de Aplicativos, incluindo desenvolvimento de interface gráfica

Características

- É um sistema interativo cujo elemento de dado básico é um array que não requer dimensionamento
  - permite resolver muitos problemas técnicos computacionais, especialmente aqueles com formulação de matrizes e vetores, em uma fração de tempo que levaria para escrever um programa em uma linguagem escalar não interativa(como C ou Fortran)
- Tem diversos toolboxes para aplicações específicas, incluindo:
  - Processamento de Sinais
  - Sistemas de Controle
  - Redes Neurais
  - Lógica Fuzzy
  - Wavelets
  - Simulação

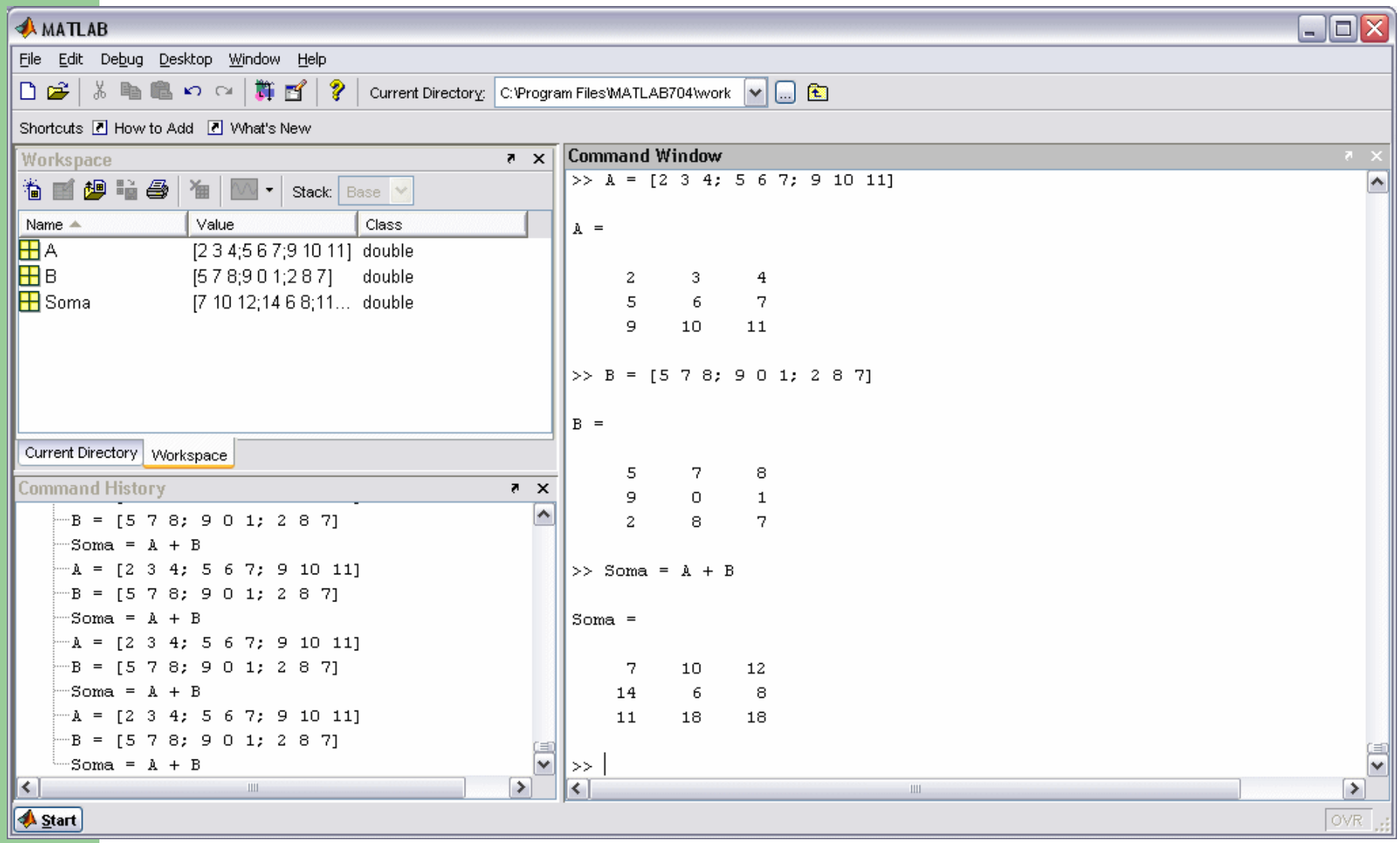
Características

- Facilita programação
  - Economia de tempo de programação
- Mais lento que linguagens tradicionais
  - Interpretado
  - Java

Sistema MATLAB

- **Ambiente de Desenvolvimento:** ferramentas que facilitam o uso do MATLAB, incluindo janela de comandos, histórico, área de trabalho, arquivos, etc.
- **Biblioteca de Funções Matemáticas:** coleção de algoritmos computacionais, variando de funções elementares como soma, seno, cosseno e aritmética complexa até funções mais sofisticadas como matriz inversa, funções de Bessel e Transformada Rápida de Fourier.
- **Linguagem Matlab:** linguagem de vetores/matriz de alto nível com controle de fluxo, funções, estrutura de dados, entrada/saída, e características de programação orientada a objetos.
- **Gráficos:**facilidades para mostrar vetores e matrizes como gráficos, e também anotações e impressões destes gráficos. Funções de alto nível para visualização de dados bidimensionais e tridimensionais, processamento de imagens, animação e apresentação de gráficos. Inclui também funções de baixo nível para personalizar a aparência de gráficos e para implementar interface gráfica para aplicações.
- **MATLAB API:** biblioteca que permite escrever programas em C e Fortran para interagir com o MATLAB. Inclui facilidades de chamar rotinas do MATLAB (elo dinâmico), chamada do MATLAB como motor computacional e leitura e escrita de arquivos MAT.

Ambiente de Desenvolvimento



Linguagem MatLab

- Versões da Mathworks incluem facilidades gráficas, de visualização e impressão
- Entretanto existem interpretadores da “linguagem Matlab”em domínio público:
  - MATLAB 1.0
  - Octave
  - Rlab
  - SciLab
- Existem ainda outros interpretadores comerciais de Matlab

Biblioteca de Funções Matemáticas

- O MATLAB incorpora diversas toolboxes com funções genéricas e específicas para diversas áreas
- Apesar de o MATLAB ser um software proprietário, os códigos-fonte das toolboxes podem ser livremente visualizados e editados.
- Existem diversas toolboxes de terceiros, tanto comerciais quanto livres.

Matrizes, Vetores e Escalares

- **Atribuição**

  escA = 3.5

  vetB = [3.5, 2.1, 10]

  matC = [3.5 2.1

  4 5]

  matD = [3.5 2.1; 4 5]

- **Busca de Valores por Índice**

  vetB(3) = 10

  matC(2,2) = 5

  matC(1,2) = 2.1

Executando Funções e Iniciando Variáveis

- Para digitar múltiplas linhas antes de executá-las, digite a linha e pressione SHIFT+ENTER.
- Para que o MATLAB execute um comando sem exibir resultado basta colocar um ponto-e-vírgula no final do comando.
- As funções e variáveis são case-sensitive
- O editor possui Syntax Highlighting e Parentheses Matching

Operações com Matrizes

```
A = [1 2 3; 4 5 6; 7 8 9];
B = [2 1 2; 4 1 7; 2 5 3];
X = A + B
X =
    3     3     5
    8     6    13
    9    13    12
```

Operadores:

+	Adição
-	Subtração
*	Multiplicação
/	Divisão à esquerda
\	Divisão à direita
^	Potência
‘	Transposição

Operação Pontual

- Operação com cada elemento da matriz de forma individual:
  - Multiplicação, Divisão e Potenciação
- Colocar ponto na frente do operador

A=[1 2
1 2]

B=[2 2
2 2]

A*B = [6 6
6 6]

A.*B=[2 4
2 4]

Trabalhando com Matrizes

- Operador dois pontos (:)
  - Criação de matriz/vetor
  - Seleção de partes da matriz
- Vetor de 0 a 1, incremento 0.2:

  >> vetA = 0:0.2:1

  vetA= 0 0.2 0.4 0.6 0.8 1.0
- Matriz 3x3 com valores de 1 a 9:

  >> matA=[1:1:3; 4:1:6;7:1:9]

  matA =

  1 2 3

  4 5 6

  7 8 9

Trabalhando com Matrizes

- Operador dois pontos (:)
- Matriz 3x3, valores de 10 a 90:

```
>> matB= [10:10:30; 40:10:60; 70:10:90]
matB =      10      20      30
            40      50      60
            70      80      90
```

- Pegando apenas uma parte da matriz acima:

```
>> matC = matB(1:2,1:3)
      matC =        10      20      30
                    40      50      60
>> matC = matB(1,:)
      matC =        10      20      30
```

Trabalhando com Matrizes

- Sendo A uma matriz 4X4:
  - Soma = A(1,4) + A(2,4) + A(3,4) + A(4,4)
  - Soma = sum(A(1:4,4))
  - Soma = sum(A(:,4))
  - Soma = sum(A(:,end))
- Todos os comandos fazem o mesmo cálculo

Exercício 1

- Criar vetor com os 6 primeiros quadrados perfeitos
- Solução:

  >>QuadPerf = 1:6;

  QuadPerf = 1 2 3 4 5 6

  >>QuadPerf = QuadPerf.^2;

  QuadPerf = 1 4 9 16 25 36
- Porque QuadPerf^2 = erro???

  QuadPerf^2 = QuadPerf * QuadPerf;

Expandindo o tamanho de uma matriz

- Se você tentar acessar um elemento fora da matriz, receberá uma mensagem de erro:

```
B = A(4,5)
Index exceeds matrix dimensions
```

- Se você tentar armazenar um elemento fora da matriz, a matriz se expande para acomodar o elemento:

```
B = A;
B(4,5) = 17
B =
    16     2     3    13     0
     5    11    10     8     0
     9     7     6    12     0
     4    14    15     1    17
```

Excluindo Linhas e Colunas

- Você pode excluir linhas de uma matriz usando apenas um par de colchetes. Começando com:

  X = A;

- Para deletar a segunda coluna de X usa-se:

  X(:,2) = [ ]

- Se você tentar excluir o único elemento de uma matriz, ela deixará de ser uma matriz, portanto essa operação não é permitida e retorna um erro.

Criando Matrizes

- **magic(n)**: cria um matriz nxn onde a soma de qualquer linha ou qualquer coluna é sempre igual
- **zeros(m,n)**: matriz mxn preenchida por 0s
- **ones(m,n)**: matriz mxn preenchida por 1s
- **eye(m,n)**: matriz mxn identidade
- **pascal(n)**: matriz de pascal

Comando Format

format short;
sqrt(2)

- **short:** 1.4142
- **long:** 1.41421356237310
- **short e:** 1.4142e+000
- **long e:** 1.41421356237309 5 e+000
- **+:** + *(sinal)*
- **rat:** 1393/985 *(aproximação)*
- **hex:** 3ff6a09e667f3bcd

Valores e Matrizes Especiais

- pi = 3,1416
- i, j = sqrt(-1)
- +Inf = +∞
- -Inf = -∞
- NaN = *not a number*
- Clock = [ano mês hora min seg]
- Date = ‘dia-mês-ano’
- Ans = *armazena resposta mais recente quando não há atribuição*

Overflow e Underflow (Limites)

- **Overflow** (10^{308})
  - Aproxima para ∞
  - **x = 2e200**
  - **y = 1e200**
  - **z = x*y (2e400)**
  - **z = Inf**
- **Underflow** (10^{-308}):
  - Aproxima para 0
  - **x = 2e-200**
  - **y = 1e200**
  - **z = x/y (2e-400)**
  - **z = 0**

Entrada e Saída

- Comando INPUT:

  >> A = input('Digite o valor de A: ')

- Saída:

  >> fprintf('texto')

- Arquivos .MAT

  - **save**

    >>save matrizes A B

  load

    >> load matrizes

Exercícios

- 2) Calcular distância euclidiana entre dois pontos (X e Y):
  - Usuário entra com os pontos
  - Exibir resultado na tela

$$d_{x,y} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- 3) Criar vetor de 0 a 2 com incremento de 0.2. (Este vetor pode ser utilizado como tempo de 0 a 2s, com amostragem a cada 0.2)
  - Salvar vetor em um arquivo

Criando funções em arquivos .m

```
function [s1,s2] = nome(e1,e2)
% Ajuda da função
```

- **Entradas e saídas**:
  - s1, s2 = saídas
  - e1, e2 = entradas
- **Variáveis:**
  - **nargin** = número de argumentos de entrada passados
  - **nargout** = número de argumentos de saída requeridos

Criando funções em arquivos .m

- Exemplo: função para calcular a distância:

```
function d=diste(p1,p2)
%Calcula e imprime
%distância entre 2 pontos
if nargin = 2
      d = (p1(1)-p2(1)).^2
      d = d + (p1(2)-p2(2)).^2
      d = sqrt(d)
end
```

Funções do Matlab

- **abs**(x): número absoluto
- **sqrt**(x): raiz quadrada
- **rand**(n,m): matriz aleatória
- [n,m] = **size**(X): tamanho de X
- **round**(x): inteiro + próximo
- **fix**(x): inteiro + próximo na direção de 0
- **floor**(x):inteiro mais próximo na direção de –Inf
- **ceil**(x): inteiro mais próximo na direção de +Inf

Funções do Matlab

- **sin**(x), **cos**(x), **tan**(x)
- **asin**(x), **acos**(x), **atan**(x)
- **sign**(x): retorna –1 para negativo, 0 para zero, 1 para positivo
- **rem**(x,y): resto da divisão entre x e y (x/y)
- **any**(x): verdadeiro se um elemento de x for <> 0
- **inv**(x): inversa
- **det**(x): determinante

Números Complexos

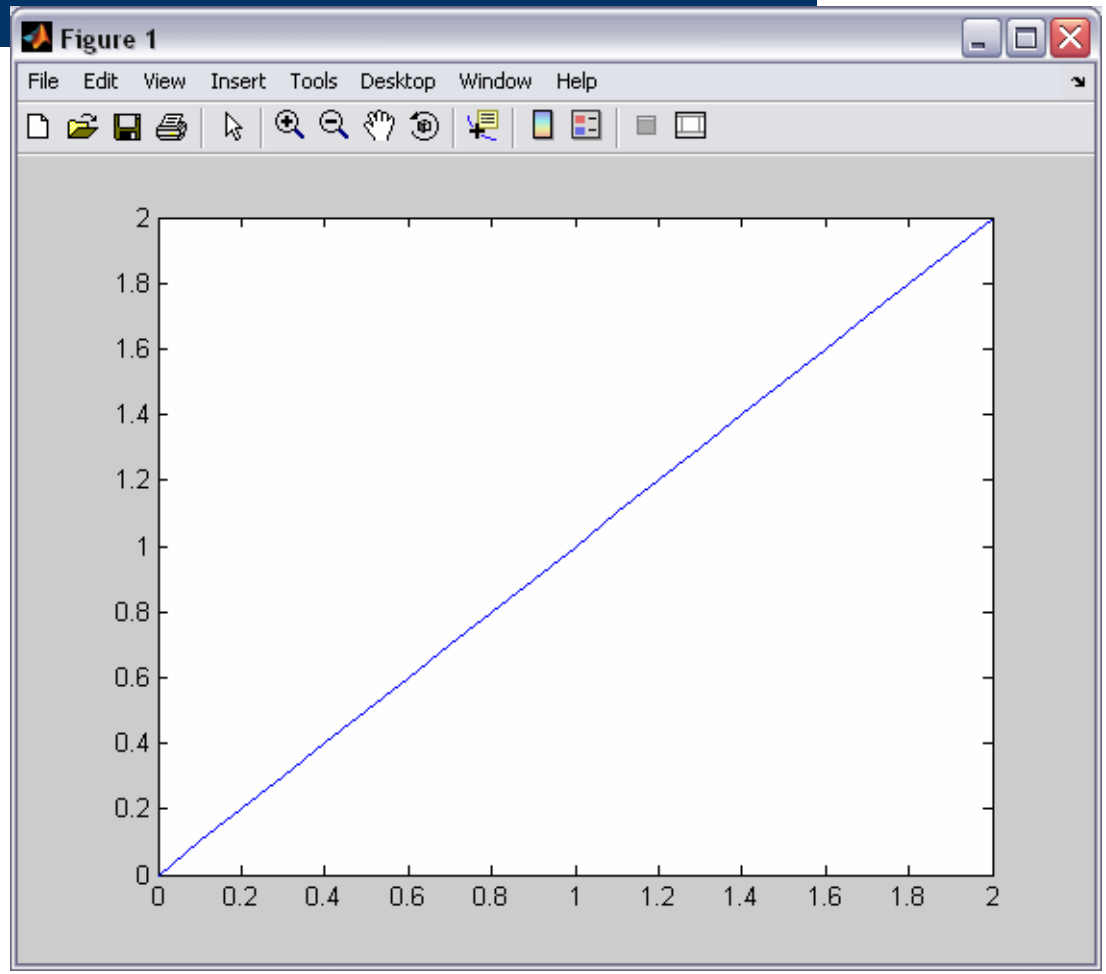
- **real**(x): retorna a parte real de x
- **imag**(x): retorna a parte imaginária de x
- **conj**(x): retorna o complexo conjugado de x
- **abs**(x): retorna o modulo complexo (magnitude) de x

Gráficos

- plot(x,y,' ');
  - x e y = vetores de mesmo tamanho

  Ex:

  ```
  x = 0:.1:2;
  y = x;
  plot(x,y);
  ```

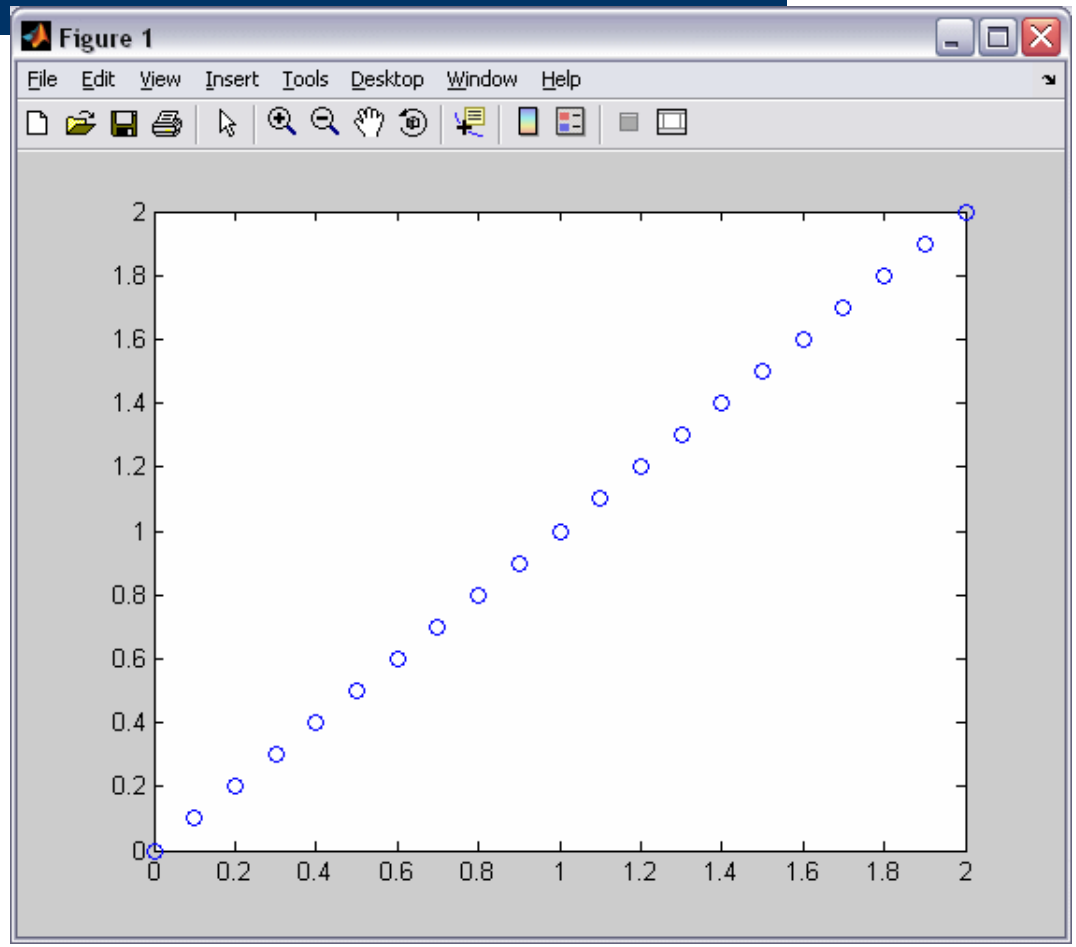


Gráficos

- plot(x,y,’ ‘);
  - x e y = vetores de mesmo tamanho

  Ex:

  ```
  x = 0:.1:2;
  y = x;
  plot(x,y,'o');
  ```

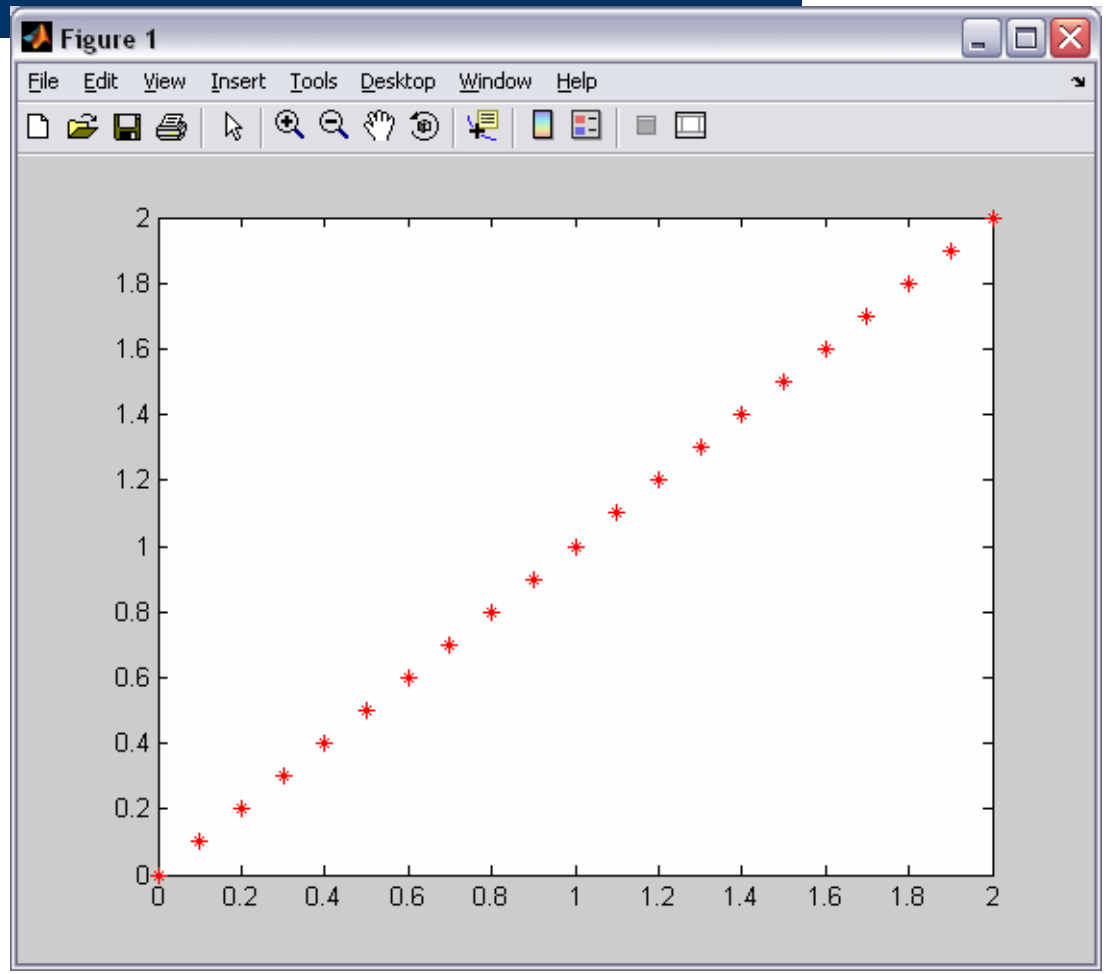


Gráficos

- plot(x,y,' ');
  - x e y = vetores de mesmo tamanho

  Ex:

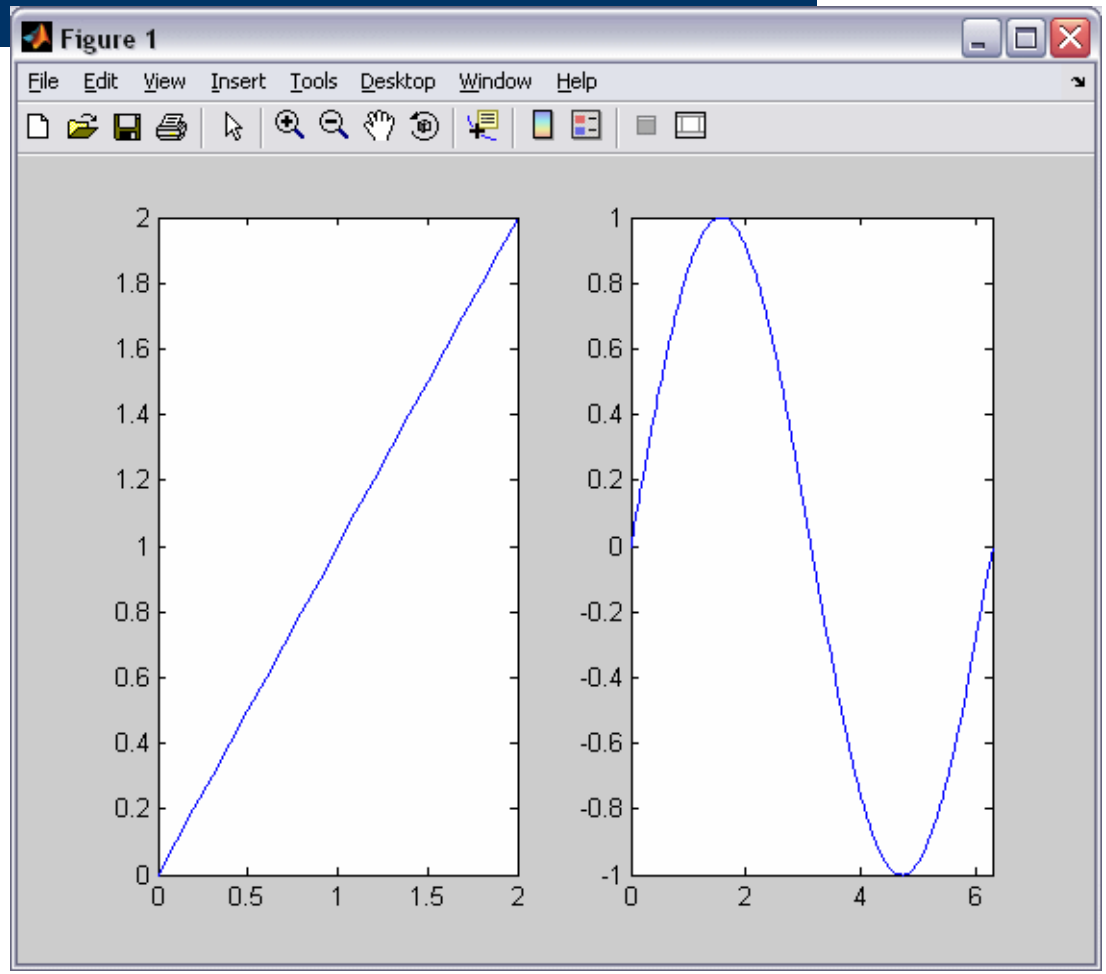
  ```
  x = 0:.1:2;
  y = x;
  plot(x,y,'r*');
  ```



Gráficos

- **subplot(m,n,p)**
  - Divide a janela em **mxn** partes para exibir vários gráficos ao mesmo tempo, escolhendo a parte **p** para a próxima exibição

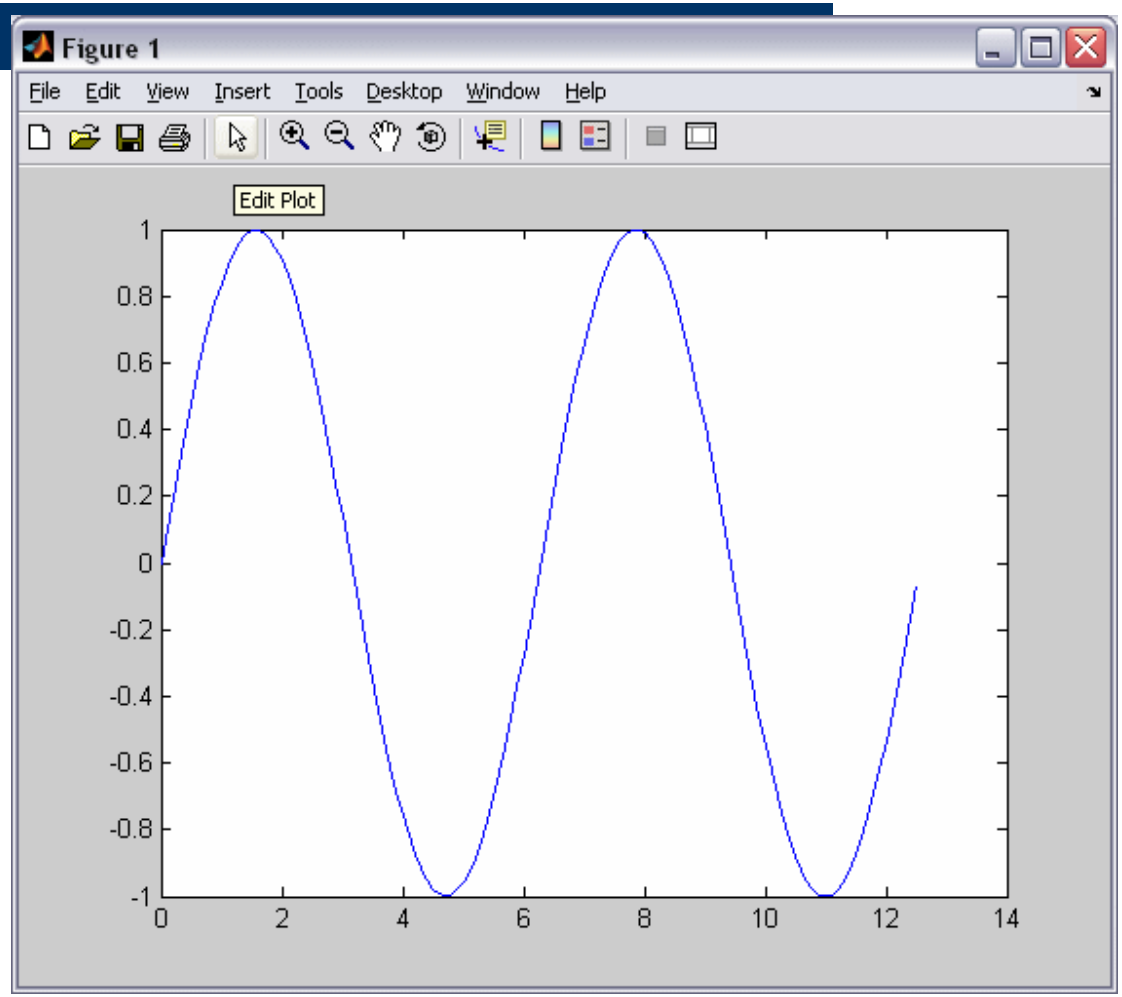
```
subplot(1,2,1);
plot(x,y);
subplot(1,2,2);
fplot('sin',[0 2*pi]);
```



Exercício 4

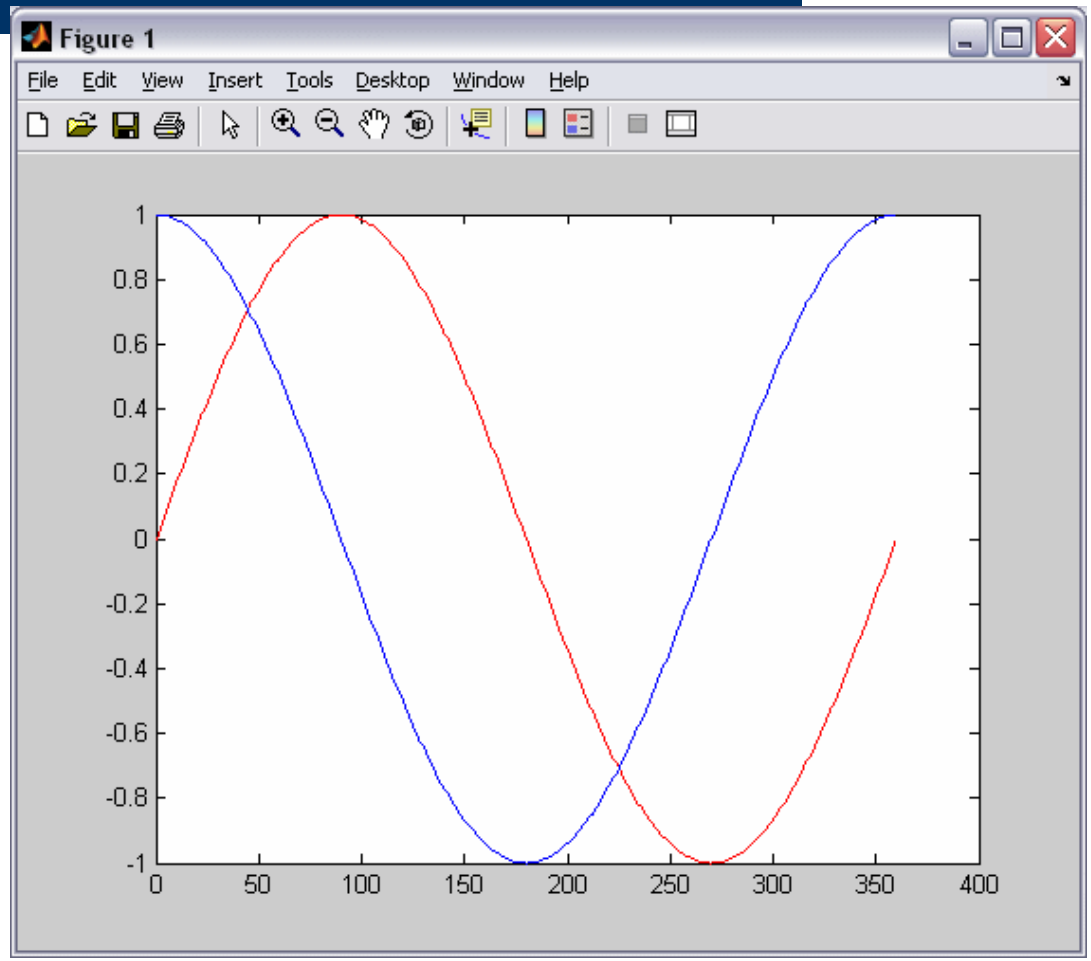
- Plotar gráfico do Seno
  - Intervalo: 0 a 4¶
  - Amostragem: 0.1

```
t = 0:0.1:4*pi;
y = sin(t);
plot(t,y);
```



Gráficos – Múltiplas Linhas

```
t=0:1:360;
y1 = sin(deg2rad(t));
y2 = cos(deg2rad(t));
plot(t,y1,'r-',t,y2,'b-');
```



Desenvolvimento de Algoritmos

- Operadores relacionais:
  - **>** - maior
  - **<** - menor
  - **>=** - maior ou igual
  - **<=** - menor ou igual
  - **==** - igual
  - **~=** - diferente
- Operadores lógicos:
  - **&** - E
  - **|** - Ou
  - **~** - Não

- Estruturas condicionais:

if (condicao)

...

elseif (condicao)

...

else

...

end

Desenvolvimento de Algoritmos

- Estruturas condicionais:

```
switch (argumento)
        case (condicao)
        ...
        otherwise
        ...
end
```

- Estruturas de repetição:

```
for (i=n : N)
        [break]
        [continue]
end
while (condição)
        [break]
        [continue]
end
```

Evitando Loops

- Loops são lentos:

  for i=1:1000, x(i) = sin(2*pi*i/100);

- Pode ser substituído por:

  x = sin(2*pi*[1:1000]/100);

Estatística

- **max**(x) máximo
- **min**(x) mínimo
- **mean**(x) média
- **median**(x) mediana
- **std**(x) desvio padrão
- **var**(x) variância
- **sum**(x) soma
- **cumsum**(x) soma cumulativa
- **prod**(x) produto
- **cumprod**(x) produto cumulativo
- **sort**(x): ordena x

```
>> x = 1:4;
 x = [1 2 3 4]
>> sum (x)
10
>> cumsum(x)
 [1 3 6 10]
>> prod(x)
 24
>> cumprod(x)
 [1 2 6 24]
```

Outros Comandos

- **who**: exibe as variáveis
- **whos**: exibe as variáveis e seus conteúdos
- **clear**: limpa a memória
- **exit/quit**: finaliza o Matlab
- **clf**: limpa a tela de trabalho
- **gcf**: coloca a figura atual (gráficos, etc.) em 1° plano
- **cfg**: limpa a figura atual
- **ls / dir**: lista o conteúdo da pasta atual
- **help / help** [comando]: exibe ajuda

Toolbox: Processamento de Sinais

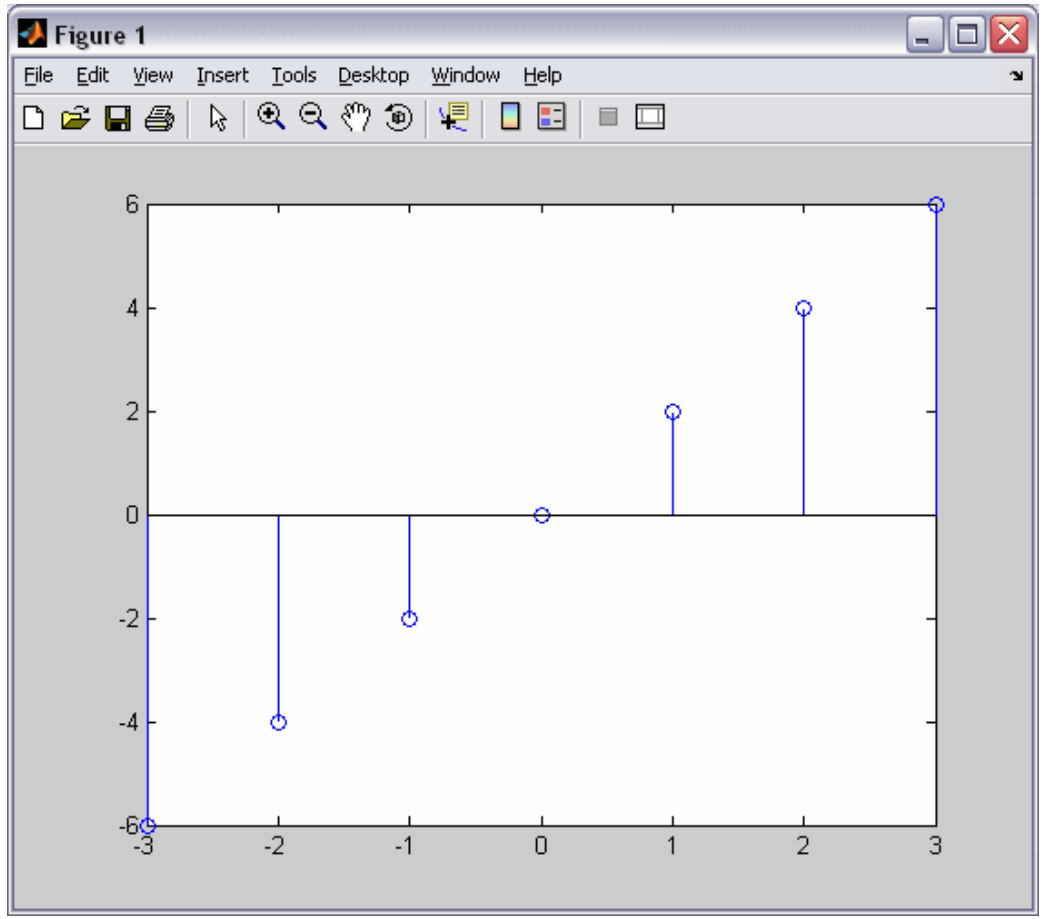
- Suporta operações com sinais, desde a geração até filtragem, modelagem e análise espectral
  - *Filtros digitais e analógicos*
  - *Implementação de filtros digitais*
  - *Transformadas*
  - *Processamento estatístico*
  - *Modelagem paramétrica*
  - *Predição linear*
  - *Geração de sinais*

Representaçao de Sinais

- Sinais são representados por vetor

$$x[n]\begin{cases} 2n, & -3 \le n \le 3 \\ 0, & otherwise \end{cases}$$

```
n = -3:3
x = 2*n;
stem(n,x);
```



Geração de Sinais

$$x(t) = \sin\left(\frac{\pi t}{4}\right)$$

- **cria um vetor tempo**

  t = -5:5;

- **e a representação do sinal**

  s = sin(pi*t/4);

$$x(n) = e^{j\left(\frac{\pi}{\frac{8}{n}}\right)}$$

- **cria um vetor tempo**

  t = 0:32;

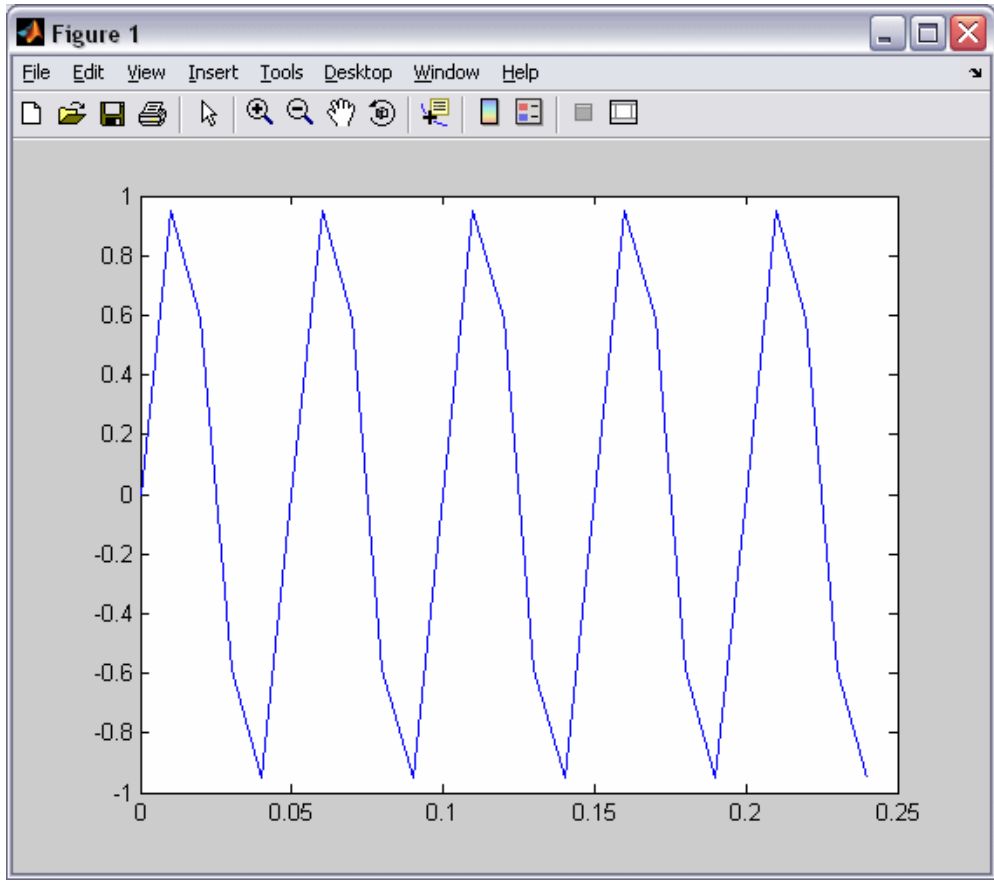
- **e a representação do sinal**

  x = exp(j*(pi/8)/n);

Geraçao de Sinais

- Gerando dados com freqüência de amostragem 100Hz
  - Vetor tempo: **t=0:0.01:1**
  - Sinal constituído de 2 senoidais com frequência de 50Hz e 120:

```
y = sin(2*pi*50*t) +
    sin(2*pi*120*t);
plot(t(1:25),y(1:25));
```

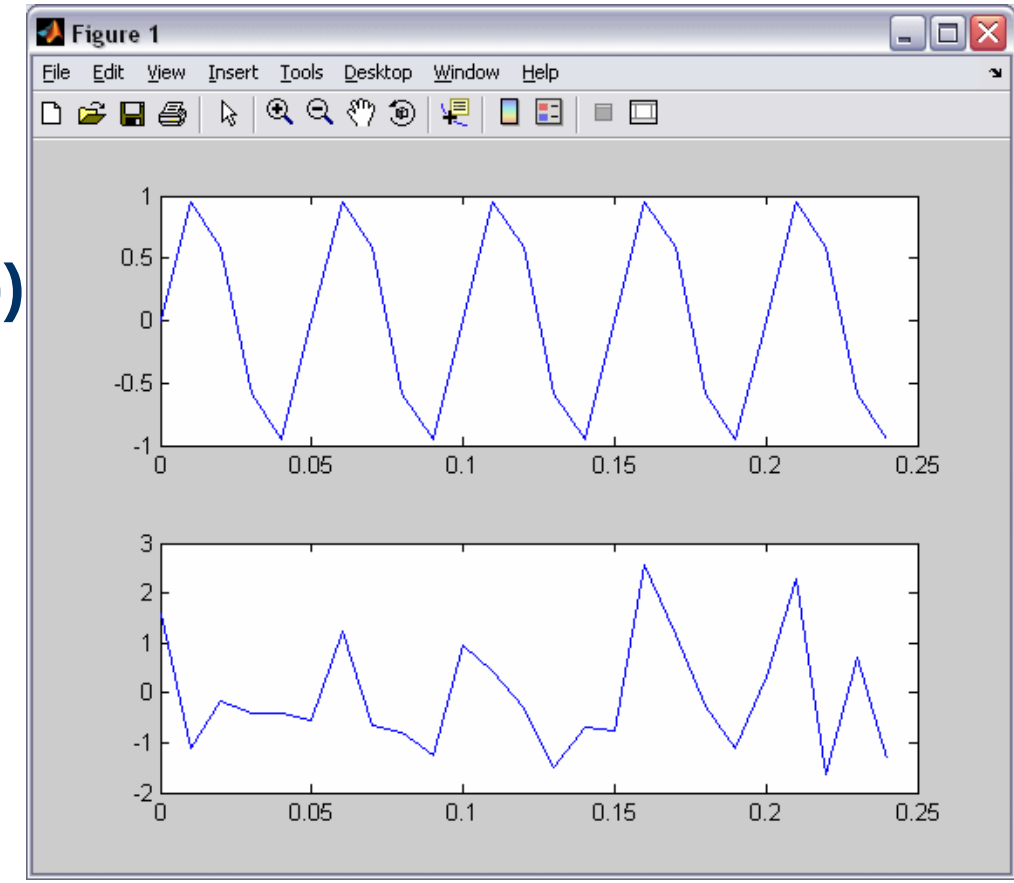


Geração de Ruído

- **randn**: gera matriz aleatória com distribuição gaussiana

  yn = y + randn(size(t))

  % adiciona ruído aleatório



Seqüências Comuns

- **Impulso unitário:**

  t = 1:100;

  y = [1; zeros(99,1)];
- **Degrau:**

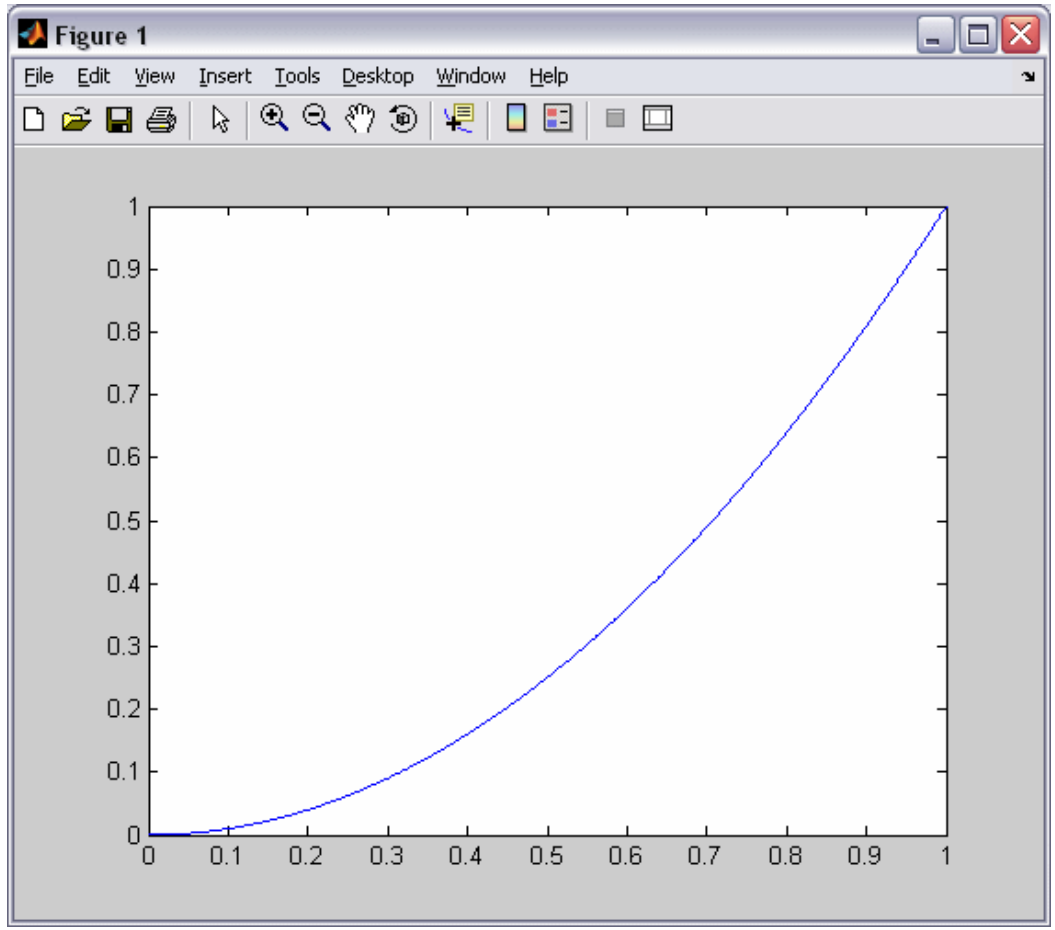
  t = 1:100;

  y = ones(100,1);
- **Rampas:**

  t = (0:0.001:1)';

  y = t;

  y = t.^2;



Convolução e Filtragem

- **conv**(x,y)
  - A saída de um filtro digital y(k) representa a sua entrada x(k) convoluída com sua resposta impulsiva h(k)
  - se a entrada x(k) e a resposta impulsiva h(k) do filtro são finitas, podemos usar **conv** para implementar um filtro

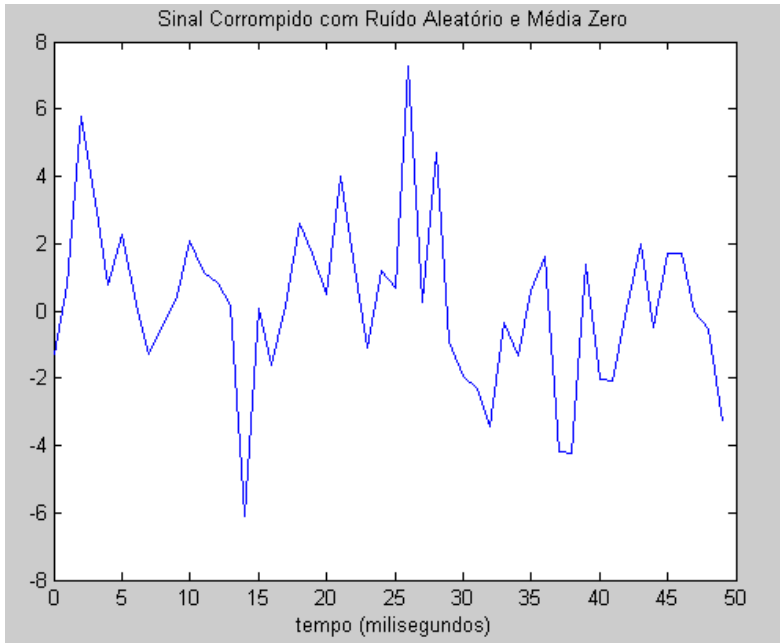
```
x = randn(5,1);
h = [1 1 1 1]/4;
y = conv(h,x);
```

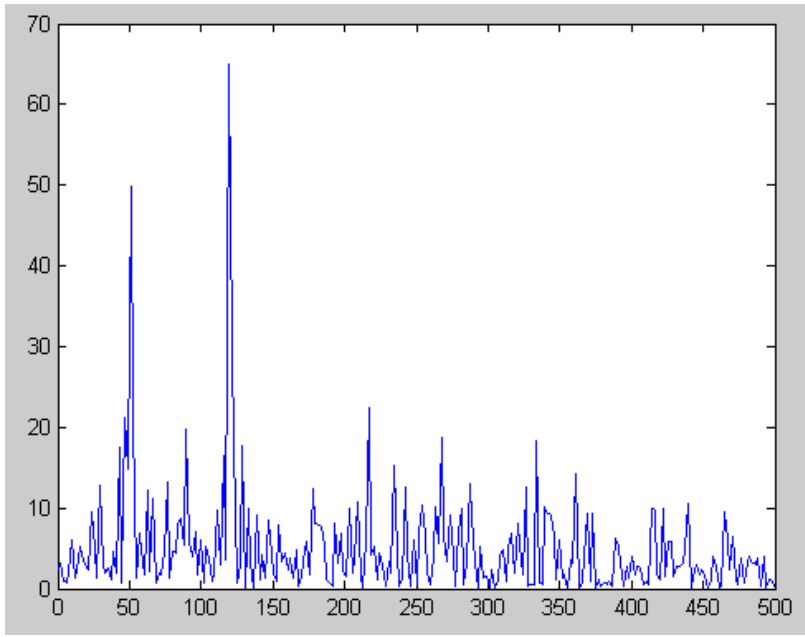
Transformada Rápida de Fourier

- fft(x,N) - transformada rápida de Fourier
- ifft(X,N) - transformada rápida de Fourier inversa

- x - sinal
- N - quantidade de pontos (deve ser potência de 2 para FFT)

Transformada Rápida de Fourier






```
t = 0:0.001:0.6;
x = sin(2*pi*50*t)+sin(2*pi*120*t);
y = x + 2*randn(size(t));
plot(1000*t(1:50),y(1:50))
title('Sinal Corrompido com Ruído Aleatório e Média Zero')
xlabel('tempo (milisegundos)')
```

```
Y = fft(y,512);
Pyy = Y.* conj(Y) / 512;
f = 1000*(0:256)/512;
plot(f,Pyy(1:257))
title(Conteudo de Frequencia de y')
xlabel('frequencia (Hz)')
```

Mais MATLAB

- http://www.mathworks.com
- http://www.mat.ufmg.br/~regi/topicos/intmatl.html
- http://www.math.ufl.edu/help/matlab-tutorial/
- http://www.dc.ufscar.br/~mauricio/