Mini-Curso: Introdução ao Matlab como Ferramenta para Uso de Engenheiros - Parte III

© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br

Módulo 3: Controle de fluxo e gráficos no Matlab

- **Ao final desse módulo estaremos preparados a:**
  - Entender os principais comandos de controle de fluxo no Matlab (Controle por laços; Controle de fluxo encadeado)
  - Gráficos bidimensionais;
  - Gerenciamento dos elementos gráficos no Matlab;
  - Gráficos tridimensionais.
  - Gráficos muito úteis em engenharia;
  - Graphical User Interface (GUI)
  - Criar arquivos executáveis
  - Otimização de códigos do Matlab
  - Toolboxes

© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br

Controle de Fluxo

1. **IF – END**

```
if expressão
        comandos
end
```

Se a expressão for verdadeira, os comando são executados.

Controle de Fluxo

3. Encadeamento de **IF**s

```
if expressão1
        comandos Xs
        if expressão2
                commandos Zs
        end
else
        comandos Ys
end
```

Se a *expressão 1* for verdadeira, os comandos Xs serão executados e ainda se a *expressão 2* for verdadeira, os comandos Zs também serão executados. Se a *expressão 1* for falsa, os comandos Ys serão processados.

Controle de Fluxo

4**. IF – ELSEIF – ELSE**

```
if expressão1
        comandos Xs
 elseif expressão2
        commandos Zs
else
        comandos Ys
end
```

Se a *expressão 1* for verdadeira, os comandos Xs serão executados. Se a *expressão 1* for falsa e a *expressão 2* for verdadeira então Zs serão executados. Se a expressão 1 e 2 forem falsas, os comandos Ys serão executados.

Vamos praticar

Fluxo_exer1.m

Crie uma função que converta a temperatura de graus Celsius para Fahrenheit ou Kelvin, utilizando as seguintes fórmulas:

$$F = 9C/5 + 32$$
$$K = C + 273.15$$

Use o controle de fluxo **if-elseif-else** e uma maneira de perguntar ao usuário para qual escala ele quer converter a temperatura. Use “digite 0” para Fahrenheit e “digite 1” para Kelvin. Imprima “opção inválida” se outro número for digitado.

Vamos praticar

Fluxo_exer2.m

Escreva uma função que recebe peso e altura de uma pessoa e calcule o Índice de Massa Corpórea (IMC)* definido como:

IMC = Peso (em kg) / Altura ao quadrado (em metros)

Essa função deve mostrar a situação de obesidade de acordo com a tabela da Associação Brasileira para o Estudo da Obesidade:

Cálculo IMC	Situação
Abaixo de 18,5	Você está abaixo do peso ideal
Entre 18,5 e 24,9	Parabéns — você está em seu peso normal!
Entre 25,0 e 29,9	Você está acima de seu peso (sobrepeso)
Entre 30,0 e 34,9	Obesidade grau I
Entre 35,0 e 39,9	Obesidade grau II
40,0 e acima	Obesidade grau III

*** O Índice de Massa Corporal (IMC) é uma medida do grau de obesidade uma pessoa. Através do cálculo de IMC é possível saber se alguém está acima ou abaixo dos parâmetros ideais de peso para sua estatura.**

Controle de Fluxo

5. Loop FOR

```
for VAR=INICIO:passo:FIM
        comandos
end
```

Executa os comandos diversas vezes até que a variável VAR possua o valor de FIM. A variável VAR é incrementada, por padrão, de uma unidade.

```
% Objetivo: Demonstrar o funcionamento do loop for

for h=1:10
  disp(['Imprime h: ' num2str(h)])
end

for h=1:2:10
  disp(['Imprime h: ' num2str(h)])
end
```


vicente.sousa@ct.ufrn.br

Controle de Fluxo

Cuidado com Loops FOR, as vezes eles são bastante ineficientes:

```
tic
for i=1:1000
   for j=1:1000
      z2(i,j)=z(j,i);
   end
end
toc
```

Execução em 6.576671 segundos.

```
tic
z2=ones( size( z ) );
for i=1:1000
   for j=1:1000
      z2(i,j)=z(j,i);
   end
end
toc
```

Execução em 0.041072 segundos.

160 vezes mais rápido

```
tic
z3 = z';
toc
```

Execução em 0.012694 segundos.

3,2 vezes mais rápido

Controle de Fluxo

6. Loop WHILE

```
while expressao1
    comandos
end
```

Executa os comandos enquanto a expressão1 for verdadeira.

```
% Objetivo: Demonstrar o funcionamento do loop while
h = 0;
while h < 10
  disp(['Imprime h: ' num2str(h)])
  %Incrementa I
  h = h + 2;
end
```

Controle de Fluxo

Exemplo: Uma máquina de café automática normalmente é capaz de preparar café, capuccino e chocolate quente. Faça um programa que pergunte ao cliente qual o seu pedido, misture os ingredientes (café, açúcar, água, chocolate ou leite), informe quais ingredientes misturou e quando a operação terminou e volte ao seu estado inicial. Caso o pedido não exista, informe o cliente.


```
sempre = 1;
while sempre == 1; % faça sempre
    %strvcat = Cria um array vertical de string
    mensagem = strvcat('Faça o seu pedido:',...
                       '(1) Café',...
                       '(2) Cappucino',...
                       '(3) Chocolate Quente',...
                       'Opção:')

    pedido = input(''); % O usuario digital alguma opcao que é guardada na variável "pedido"

    if pedido==1
        ['Preparando...']
        ['Estará pronto em 2 segundos! Aguarde.']
        pause(2);
        %misture agua+cafe+acucar
        ['CAFE PRONTO']
    elseif pedido == 2
        ['Preparando...']
        ['Estará pronto em 5 segundos! Aguarde.']
        pause(5);
        %misture agua+cafe+acucar+chocolate
        ['CAPPUCINO PRONTO']
    elseif pedido == 3
        ['Preparando...']
        ['Estará pronto em 4 segundos! Aguarde.']
        pause(4);
        %misture agua+cafe+acucar+chocolate
        ['CHOCOLATE PRONTO']
    else
        I = 0;
        while I<3 %mostre a mensagem 3 vezes
            ['PEDIDO DESCONHECIDO!!!']
            %espere 1 segundo
            pause(1)
            ['FAÇA OUTRO PEDIDO...']
            I = I + 1;
        end
    end
    ['MAQUINA PRONTA NOVAMENTE!!!']
end
```

Vamos praticar

loop_exer1.m

- Faça um programa que conte de 0 a 10 e faça a contagem regressiva até atingir 0 novamente. Faça a operação somente depois que o usuário pressionar 1.
  - Faça com que os números apareçam de um em um segudo
- Uma bola de borracha é solta de uma altura D, ao chocar-se contra o chão, esta retorna a uma altura D/2. Faça um programa que calcule a distância da bola ao solo após N choques e a distância percorrida. Os parâmetros D e N devem ser atribuídos pelo usuário.

loop_exer2.m

loop_exer3.m

Vamos praticar

Faça um programa bancário simples capaz de mostrar o saldo, extrato, realizar saques e depósitos.

Use o *template* banco.m como ponto de partida.

Construa uma função chamada **deposito** para realizar depósitos na conta

[saldo, extrato] = deposito(saldo, extrato);

Construa uma função chamada **saque** para realizar retiradas na conta. A função deve ter os seguinte formato:

[saldo, extrato] = saque(saldo, extrato);

Observação: saques só devem ser realizados se a conta tiver saldo positivo.

Compilar códigos em Matlab

- É possível criar códigos executável em Matlab
- Processo via ***MATLAB Compiler Runtime (MCR)***
  - Código é convertido para linguagem C ou C++
  - Executável é gerado
- Benefícios
  - Distribuir seu programa com pessoas que não tem o Matlab
  - Infelizmente, não será experimentado ganho de desempenho
  - Gerar códigos para serem usados em outros programas já feitos em C++ ou Simulink
  - Encapsula o código de maneira a ele não ser modificado por tereceiros

Compilar códigos em Matlab

- Instalação
  - No momento de instalação do Matlab escolher para instalar o via ***MATLAB Compiler Runtime (MCR)***
  - Compilar c ou c++
    - O compilador **lcc C version 2.4.1** é incluiu na instalação do Matlab. lcc só compila códigos para C (não para C++)
  - A lista completa (atualizada) de compiladores suportados pode ser acessada em:
    - http://www.mathworks.com/support/compilers/current_release/

Compilar códigos em Matlab

- Sistemas Operacionais suportados:
  - Windows
  - Linux
  - Linux x86-64
  - Mac OS® X
- Aula adicional sugerida (em vídeo): **MATLAB para Programadores C/C++**
  - http://www.mathworks.com/company/events/webinars/wbnr33559.html?id=33559&p1=583262549&p2=583262561

Compilar códigos em Matlab

- Para verificar e configurar o compilador
  - mbuild –setup (procura os compiladores instalados)
  - Ou você pode direcionar a pasta que o compilador está instalado
- Macro para compilar
  - mcc -m -v **<arquivo .m>**
  - **Exemplo:** mcc -m -v banco.m

Exportar dados do Matlab para o excel

- Follow these steps to install Spreadsheet Link EX with Microsoft Excel 2007:

  1. Start Microsoft Excel.
  2. Click the Microsoft Office Button, and then click Excel Options.
  3. Click the Add-Ins category.
  4. In the Manage box, select Excel Add-ins, and then click Go.
  5. Click Browse.
  6. Select the Excel Link add-in excllink.xla under $MATLABROOT/toolbox/exlink. (where $MATLABROOT is the MATLAB root directory on your machine, as returned by typing: matlabroot at the MATLAB command prompt.)
  7. Click OK.
  8. Back in the Add-Ins dialog box, make sure that the Excel Link check box is selected, and click OK. The Excel Link add-in loads now (and automatically loads with each subsequent invocation of Excel).
  9. Watch for the appearance of the MATLAB button on the Windows taskbar.
  10. Watch for the appearance of the "Add-Ins" menu item.
  11. Click on the "Add-Ins" menu item. The Excel Link toolbar is displayed.

Até gráficos são transferidos do Matlab para o Excel

Link com excel

- Usar o matlab dentro do Excel
  - http://www.mathworks.com/products/datasheets/pdf/spreadsheet-link-ex.pdf

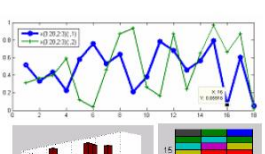
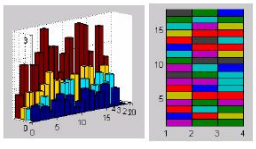
vicente.sousa@ct.ufrn.br

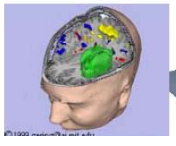
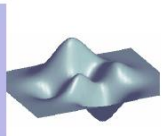
Link com excel

- Aula adicional sugerida (em vídeo): **MATLAB® para Usuários de Excel**
  - http://www.mathworks.com/company/events/webinars/wbnr30662.html?id=30662&p1=49734&p2=49735

Gráficos

- **O Matlab permite criar gráficos de vários tipos, entre eles:**
  - Visualizar conteúdo das variáveis
  - Criar imagens e vídeos
  - Gerar Interface Gráfica do Usuário (GUI)

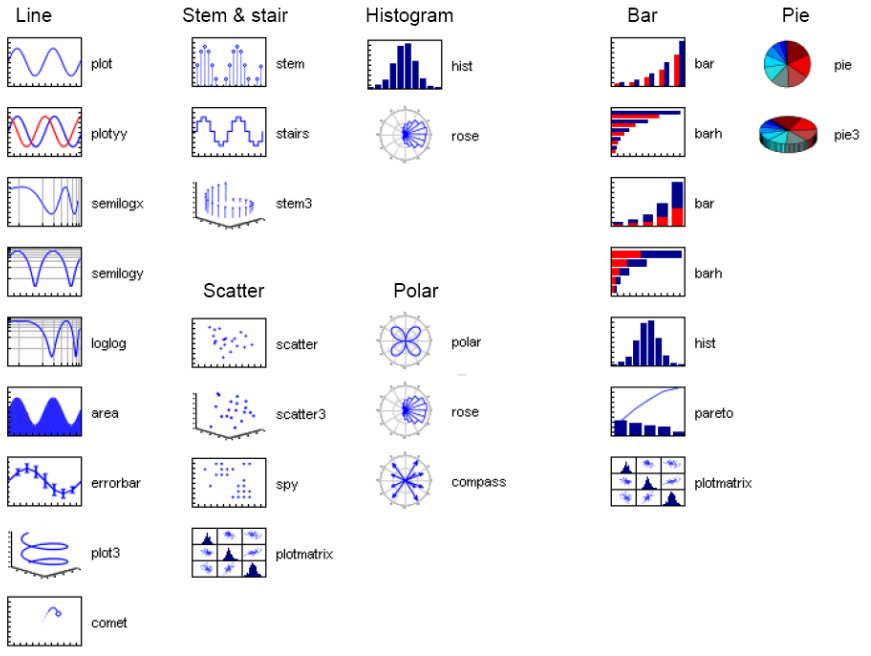




Gráficos

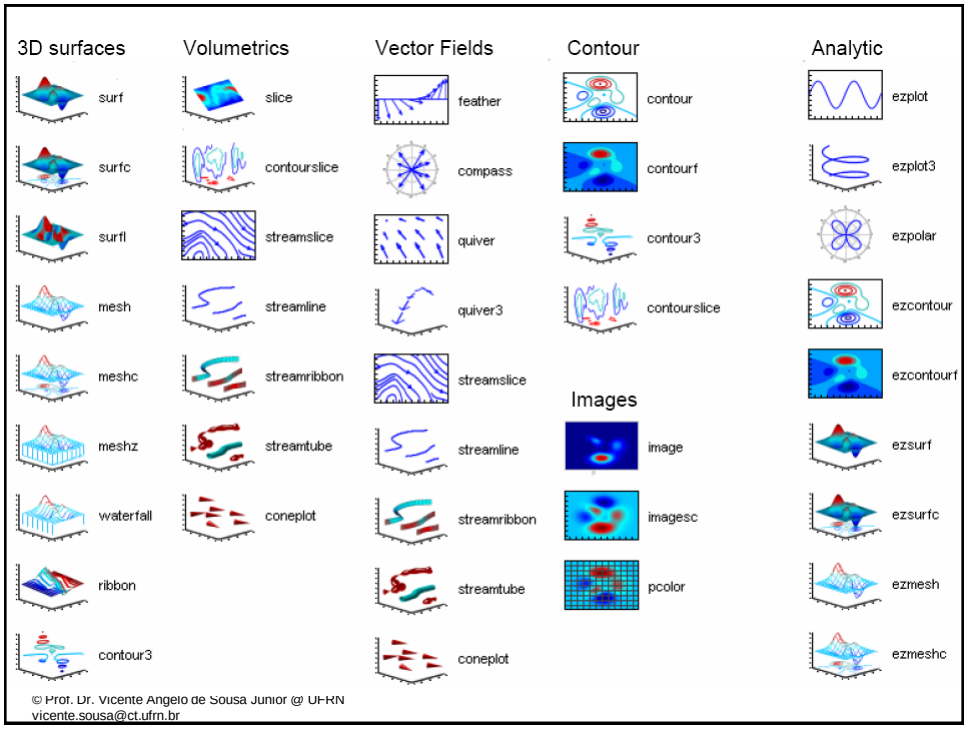
O Matlab possui diversas bibliotecas para realização dos mais variados tipos de gráficos, destacam-se:

Comando	Descrição
plot	Gráfico linear
loglog	Gráfico em escala logarítmica
semilogx	Gráfico em escala semilog
semilogy	Gráfico em escala semilog
fill	Preenche polígonos 2D
polar	Gráfico em coordenada polar
bar	Gráfico em barras
stem	Gráfico de seqüência discreta
stairs	Gráfico em degraus
hist	Gráfico em forma de histograma
cdfplot	Gráfico em forma de distribuição de probabilidade

© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br



© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br



Como fazer gráficos

Eixo horizontal **Eixo vertical**

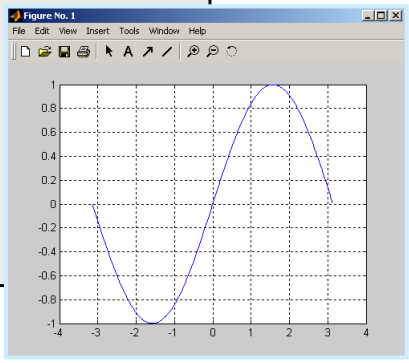
```
plot(x,y,'rx-');
```

cor **Tipo de linha**

y	yellow	.	**marcadores**	-	**Linha contínua**
m	magenta	o		:	**Linha de pontos**
c	cyan	x		-.	**Linha com pontos**
r	red	+		--	**Linha tracejada**
g	green	*			
b	blue				
w	white				
k	black				

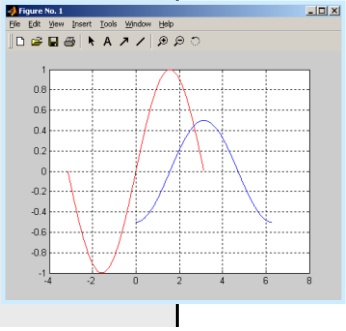
Gráficos

```
% Objetivo: Demonstrar o funciona-
% mento do comando plot.

% Definindo o vetor - π ≤ t ≤ π
t = [-pi:0.01:pi];

% Definindo o vetor y = sen(t)
y = sin(t);

% Criando o gráfico
plot(t,y);

% Criando a grade do gráfico
grid;
```



Gráficos

```
% Objetivo: Demonstrar o funcionamento dos
% recursos de cores e múltiplos gráficos
% associados ao comando plot.

% Definindo o vetor - π ≤ t1 ≤ π
t1 = [-pi:0.01:pi];

% Definindo o vetor y1 = sen(t1)
y1 = sin(t1);

% Definindo o vetor - π ≤ t2 ≤ π
t2 = [-pi:0.01:pi];

% Definindo o vetor y2 = cos(t2)/2
y2 = cos(t2)/2;

% Criando o gráfico
plot(t1,y1,'r',t2,y2,'b');
```




```
% Também pode ser feito:
plot(t1,y1,'r');
Hold on;
plot(t2,y2,'b');
```

Gráficos

Cores associadas aos gráficos

Parâmetro	Descrição
b	Azul (*blue*)
g	Verde (*green*)
r	Vermelho (*red*)
c	Ciam (*cyan*)
m	Magenta
y	Amarelo (*yellow*)
k	Preto (*black*)

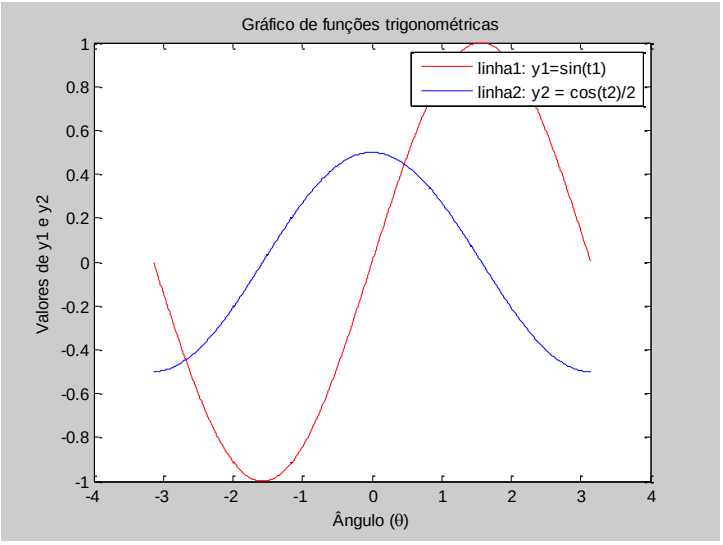
Texto em Gráficos

Título, nome dos eixos e legenda:

```
t1 = [-pi:0.01:pi];
y1 = sin(t1);
t2 = [-pi:0.01:pi];
y2 = cos(t2)/2;

% Criando o gráfico
plot(t1,y1,'r');
Hold on;
Plot(t2,y2,'b');

% Incluindo nome dos eixos
xlabel('Ângulo (\theta)');
ylabel('Valores de y1 e y2');

% incluindo título
title('Gráfico de funções trigonométricas');

% inlcuindo legenda
legend('linha1: y1=sin(t1)', 'linha2: y2 = cos(t2)/2');
```

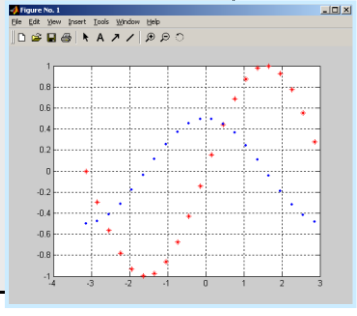
Aceita comando em Latex

Texto em Gráficos



Gráficos

```
% Objetivo: Demonstrar o funcionamento dos
% recursos de símbolos e múltiplos gráficos
% associados ao comando plot.

% Definindo o vetor - π ≤ t1 ≤ π
t1 = [-pi:0.3:pi];

% Definindo o vetor y1 = sen(t1)
y1 = sin(t1);

% Definindo o vetor - π ≤ t2 ≤ π
t2 = [-pi:0.3:pi];

% Definindo o vetor y2 = cos(t2)/2
y2 = cos(t2)/2;

% Criando o gráfico
plot(t1,y1,'r*',t2,y2,'b.');
```



Gráficos

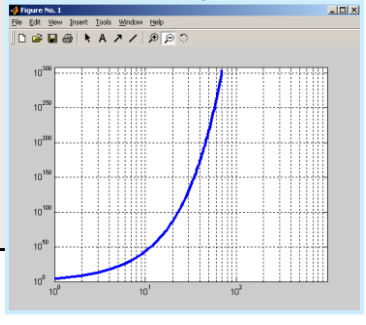
Símbolos associadas aos gráficos

Parâmetro	Descrição
.	Ponto
o	Circulo
*	Estrela
^	Diamante (cima)
v	Diamante (baixo)
>	Diamante (direita)
<	Diamante (esquerda)

© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br

Gráficos

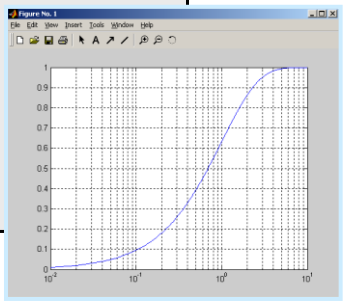
```
% Objetivo: Demonstrar o funcionamento do
% comando loglog. Este cria o gráfico no
% domínio logarítmico tanto no eixo das
% coordenadas como abscissas.

% Definindo o vetor 0 ≤ t ≤ 1000
t = [0:1000];

% Definindo o vetor y
y = exp(10*t);

% Criando o gráfico
loglog(t,y);
```



© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br

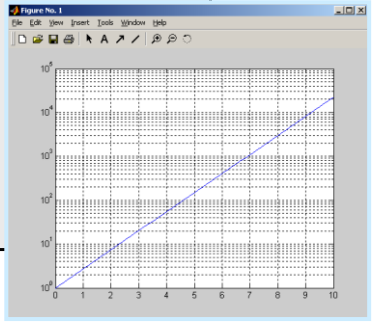
Gráficos

```
% Objetivo: Demonstrar o funcionamento do
% comando semilogx. Este cria o gráfico no
% domínio logarítmico somente no eixo das
% coordenadas.

% Definindo o vetor 0 ≤ t ≤ 10
t = [0:0.01:10];

% Definindo o vetor y
y = 1-exp(-t);

% Criando o gráfico
semilogx(t,y);
```



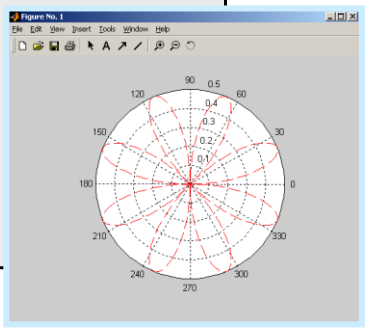
Gráficos

```
% Objetivo: Demonstrar o funcionamento do
% comando semilogy. Este cria o gráfico no
% domínio logarítmico somente no eixo das
% abscissas.

% Definindo o vetor 0 ≤ t ≤ 10
t = 0:0.1:10;

% Definindo o vetor y
y = exp(t);

% Criando o gráfico
semilogy(t,y);
```



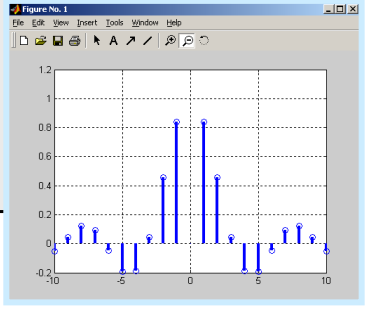
Gráficos

```
% Objetivo: Demonstrar o funcionamento do
% comando polar. Este desenha gráficos
% na forma polar.

% Definindo o vetor 0 ≤ θ ≤ 2π
teta = [0 : 0.01 : 2*pi];

% Definindo o vetor
% ρ = sen(2θ).cos(2θ)
rho=sin(2*teta).*cos(2*teta)

% Criando o gráfico
polar(teta,rho,'--r')
```



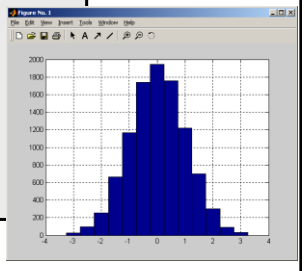
Gráficos

```
% Objetivo: Demonstrar o funcionamento do
% comando stem. Este desenha gráficos
% de seqüências discretas..

% Definindo o vetor -10 ≤ n ≤ 10
n=[-10 : 10];

% Define o vetor y. Função sinc.
y=sin(n)./n;

% Criando o gráfico
stem(n,y);
```



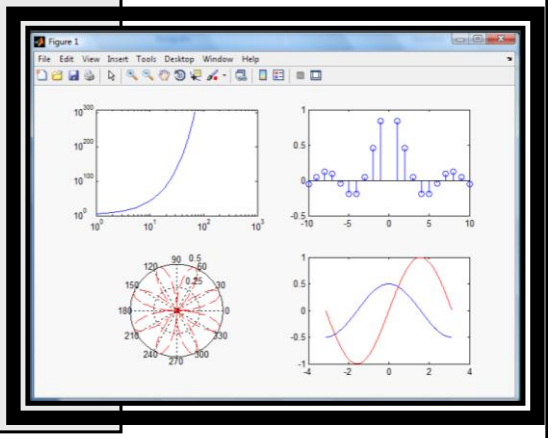
Gráficos

```
% Objetivo: Demonstrar o funcionamento do
% comando hist. Este desenha gráficos
% na forma de histogramas.

% Definindo o vetor -3 ≤ t ≤ 3 com
% intervalo de 0,5.
t = [-3 : 0.5 : 3];

% Definindo um vetor randômico de 1000
% elementos
y = randn(10000,1);

% Criando o gráfico
hist(y,t);
```



Vários gráficos na mesma figura

```
% Objetivo: Demonstrar o funcionamento do
% comando subplot. Este faz gráficos
% diferentes em uma matriz  m por n de
% pequenos axis

% grafico 1
t = [0:1000];
y = exp(10*t);
subplot(2,2,1);
loglog(t,y);
% grafico 2
subplot(2,2,2);
n=[-10 : 10];
y=sin(n)./n;
stem(n,y);
% grafico 3
subplot(2,2,3);
teta = [0 : 0.01 : 2*pi];
rho=sin(2*teta).*cos(2*teta)
polar(teta,rho,'--r')
% grafico 4
subplot(2,2,4);
t1 = [-pi:0.01:pi];
y1 = sin(t1);
t2 = [-pi:0.01:pi];
y2 = cos(t2)/2;
plot(t1,y1,'r',t2,y2,'b');
```



Vamos praticar

graficos_exer2.m

Refazer os gráficos anteriores, mais posicioná-los em uma só coluna

```
% grafico 1
t = [0:1000];
y = exp(10*t);
subplot(2,2,1);
loglog(t,y);
% grafico 2
subplot(2,2,2);
n=[-10 : 10];
y=sin(n)./n;
stem(n,y);
% grafico 3
subplot(2,2,3);
teta = [0 : 0.01 : 2*pi];
rho=sin(2*teta).*cos(2*teta)
polar(teta,rho,'--r')
% grafico 4
subplot(2,2,4);
t1 = [-pi:0.01:pi];
y1 = sin(t1);
t2 = [-pi:0.01:pi];
y2 = cos(t2)/2;
plot(t1,y1,'r',t2,y2,'b');
```

Comandos que devem mudar

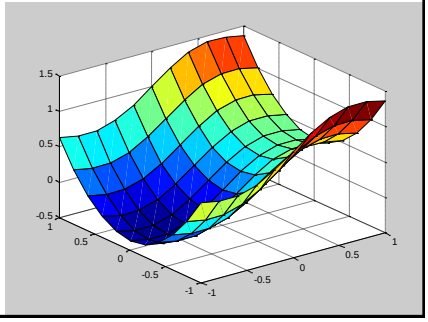
Gráficos em três dimensões

Para fazer gráficos em 3D, geralmente se usa a função **meshgrid** para criar as matrizes em duas dimensões que são apropriadas para as funções que fazem gráficos em 3D

Exemplo:

Para representar graficamente a função $x*\exp(-x^2) + y^2$ nos intervalos $-1 < x < 1$, $-1 < y < 1$, podemos fazer:

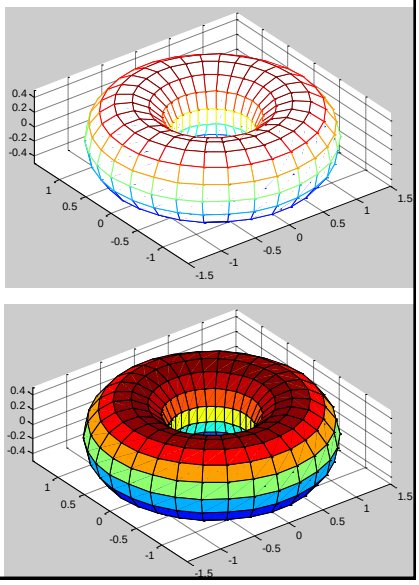
```
[X,Y] = meshgrid(-1:0.2:1, -1:0.2:1);
Z = X .* exp(-X.^2)+ Y.^2;
surf(X,Y,Z)
```



Gráficos em três dimensões

Outro exemplo: **Toróide**

```
r=0.5; %radio lateral
n=30; %número de elementos
a=1; %radio central

% Cálculo dos angulos
theta=pi*(0:2:2*n)/n;
phi=2*pi*(0:2:n)'/n;

% Calculo y proyecto en x,y,z.
xx=(a + r * cos(phi))*cos(theta);
yy=(a + r * cos(phi))*sin(theta);
zz=r * sin(phi)*ones(size(theta));
mesh(xx,yy,zz);
axis equal;
figure;
surf(xx,yy,zz);
axis equal;
```

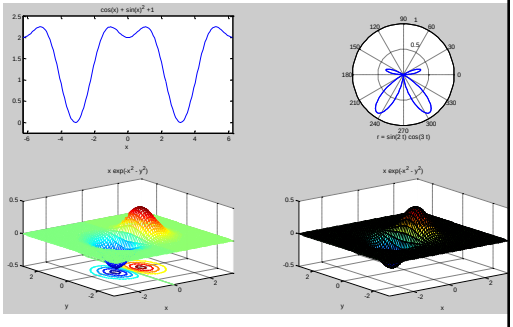


Gráficos

Easy plot

- **Comandos gráficos com o prefixo EZ servem para gerar gráficos de maneira simples no domínio** $-2\pi < x < 2\pi$

```
clc;
clear all;
close all;

% grafico 1
subplot(2,2,1);
ezplot('cos(x) + sin(x)^2 +1');

% grafico 2
subplot(2,2,2);
ezpolar('sin(2*t)*cos(3*t)')

% grafico 3
subplot(2,2,3);
ezmeshc('x.*exp(-x.^2 - y.^2)')

% grafico 4
subplot(2,2,4);
ezsurf('x.*exp(-x.^2 - y.^2)')
```



Exercícios sobre Gráficos

Crie os tipos de gráficos indicados para as equações abaixo:

$a) y = x^2 + 1$

Gráfico de barras com x de 0 a 100 com passo de 10 em 10

Adicione o gráfico de erro considerando o intervalo igual ao desvio padrão de y

$b) z = \sin(x) + \cos(y)$

Gráfico de superfície variando x e y de -4π a 4π

graficos_exer3.m

Análise de Fourier

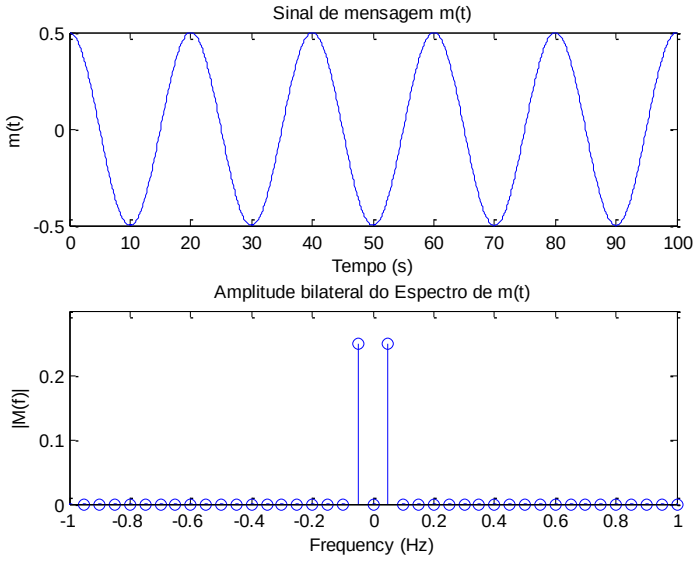
- **Parâmetros da modulação AM DSB:**

```
% Objetivo: demonstrar como calcular a magnitude do espectro de
    frequência de uma modulação AM DSB

% Resolução em frequencia
fsampling = 10;

% Parametros da modulação AM
ma = 0.5; % Índice de modulação  = Am/Ac
Ac = 1; % Amplitude da portadora
Am = ma * Ac; % amplitude do sinal mensagem

% Parametros do sinal mensagem
fm = 0.05;

% Parametros da Portadora
fc = 0.4;
t = 0 : (1/fsampling) : 5*(1/fm);
```

Análise de Fourier sinal AM DSB

```
% Objetivo: demonstrar como calcular a magnitude do espectro de frequência
    de uma modulação AM DSB

% sinal mensagem
m = Am*cos(2*pi*fm*t);

% plot do sinal no tempo
subplot(2,1,1);
plot(t,m);
title('Sinal de mensagem m(t)')
xlabel('Tempo (s)')
ylabel('m(t)')

% Plot double-sided amplitude spectrum.
subplot(2,1,2);
lfft = 200;
nSamples = length(t);
freq = [-fsampling/2 : fsampling/lfft : fsampling/2 - fsampling/lfft];
ym = fftshift(fft(m,lfft))/lfft;
stem(freq,abs(ym))
title('Amplitude bilateral do Espectro de m(t)')
xlabel('Frequency (Hz)')
ylabel('|M(f)|')
set(gcf,'color',[1 1 1])
axis([ -1 1 0 0.3])
```

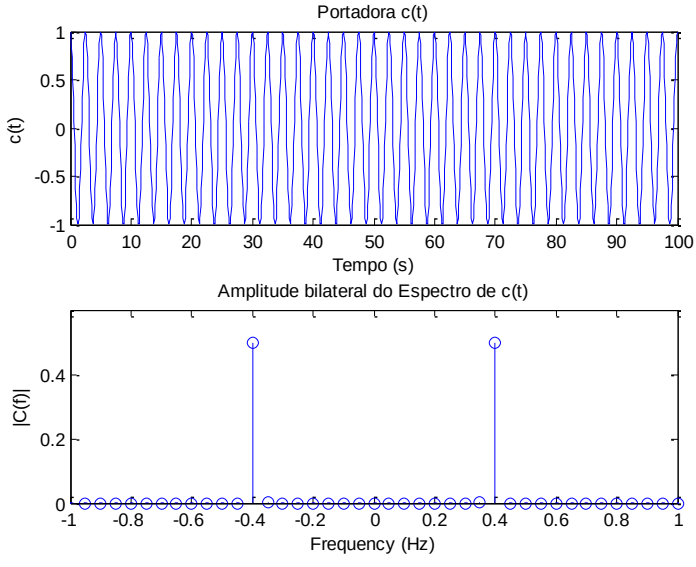

Análise de Fourier sinal AM DSB



Análise de Fourier sinal AM DSB

```
% Objetivo: demonstrar como calcular a magnitude do espectro de frequência
    de uma modulação AM DSB

% Portadora
c = Ac*cos(2*pi*fc*t);
figure;
% plot signal message in time
subplot(2,1,1);
plot( t, c );
title('Portadora c(t)')
xlabel('Tempo (s)')
ylabel('c(t)')

% Plot double-sided amplitude spectrum.
subplot(2,1,2);
lfft = 200;
nSamples = length(t);
freq = [-fsampling/2 : fsampling/lfft : fsampling/2 - fsampling/lfft];
yc = fftshift(fft(c,lfft))/lfft;
stem(freq,abs(yc))
title('Amplitude bilateral do Espectro de c(t)')
xlabel('Frequency (Hz)')
ylabel('|C(f)|')
set(gcf,'color',[1 1 1])
axis([ -1 1 0 0.6])
```

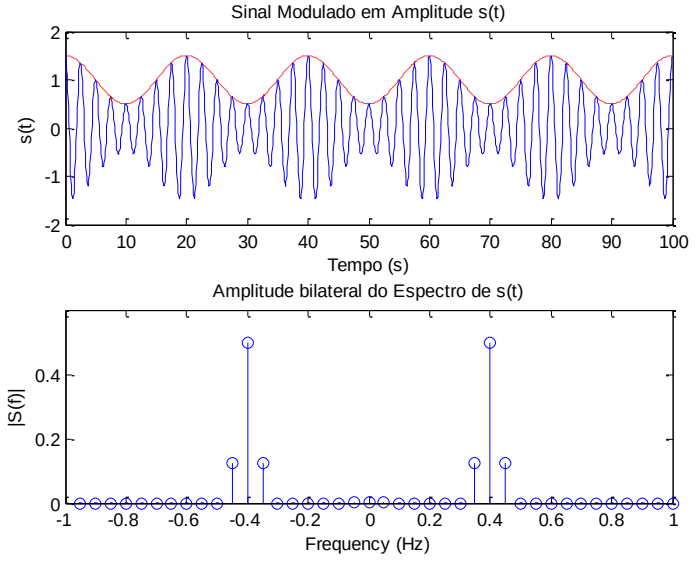

Análise de Fourier sinal AM DSB



Análise de Fourier sinal AM DSB

```
% Objetivo: demonstrar como calcular a magnitude do espectro de frequência
    de uma modulação AM DSB

% onda modulada
am = Ac*(1 + ma*cos(2*pi*fm*t) ).*cos(2*pi*fc*t)

% plot signal message in time
figure;
subplot(2,1,1);
plot( t, am );
hold on;
plot(t,Ac+m,'--r');
title('Sinal Modulado em Amplitude s(t)')
xlabel('Tempo (s)')
ylabel('s(t)')

% Plot double-sided amplitude spectrum.
subplot(2,1,2);
lfft = 200;
freq = [-fsampling/2 : fsampling/lfft : fsampling/2 - fsampling/lfft];
yam = fftshift(fft(am,lfft))/lfft;
stem(freq,abs(yam))
title('Amplitude bilateral do Espectro de s(t)')
xlabel('Frequency (Hz)')
ylabel('|S(f)|')
axis([ -1 1 0 0.6])
set(gcf,'color',[1 1 1])
```

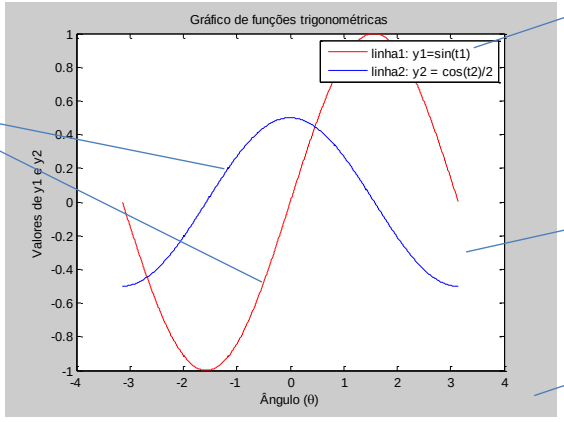
Análise de Fourier sinal AM DSB



vicente.sousa@ct.ufrn.br

Elementos Gráficos

- **Cada objeto gráfico possui um tipo (figure, axes, line, etc)**
- **Todas as propriedades de cada objeto gráfico estão armazenadas em "handles".**



Displayname da line 1

line

axis

figure

Gerenciamento de Elementos Gráficos

Para cada tipo de elemento gráfico é associado um número de identificação (handle) e existem comandos para acessá-los e mudar suas propriedades, entre eles:

-gcf: handle da **figura** corrente
-gca: handle do **axis** corrente
-get(*h*): mostra as propriedades do objeto identificado com o handle *h*
-*get(h,* 'PropertyName'): visualiza a propriedade 'PropertyName' do objeto *h*
- set(handle,'PropertyName','Value',...) : muda a propriedade de nome **PropertyName** para o valor **Value** do objeto identificado por **handle.**

Vamos praticar

graficos_exer1.m

Execute as linhas de comando a seguir:

```
close all; clc; clear all;
t1 = [-pi:0.3:pi];
y1 = sin(t1);
plot(t1,y1,'r-')
Hold on;
y2 = cos(t1);
plot(t1,y2, 'b-')
legend('seno', 'coseno')
xlabel('Tempo')
```

1. Use o comando **get** para ver as propriedades da figura criada
2. Use o comando **set** para mudar a cor da figura para branca (use o padrão RGB: [1 0 0] é vermelho)
3. Use o comando **get** para ver as propriedades do *axes* criado
4. Use o comando **get** acessar o objeto line (ver propriedade "children" do axes)
5. Use o comando **set** para mudar a legenda **coseno** para **casino**
6. Use o comando **get** acessar o objeto xlabel do axis
7. Use o comando **set** para mudar o título do eixo x de **Tempo** para **Ângulo (β)**

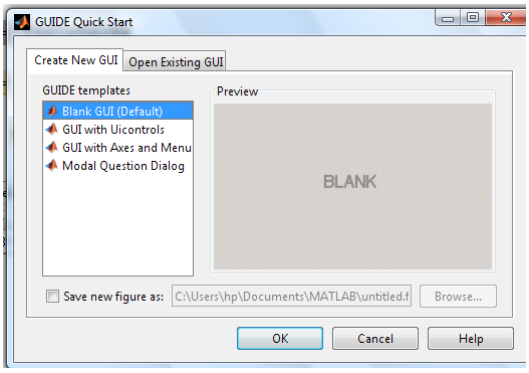
GUI (Interface Gráfica do Usuário)

- **O Matlab permite criar Interface Gráfica do Usuário**
  - Facilita a visualização da entrada e saída de dados
  - Personaliza programas
  - Facilita a interação homem-máquina

Usando o GUI (Interface Gráfica do Usuário) do Matlab

- Como invocar: digite o comando **guide** no workspace
- Um assistente abrirá perguntando se você deseja usar algum

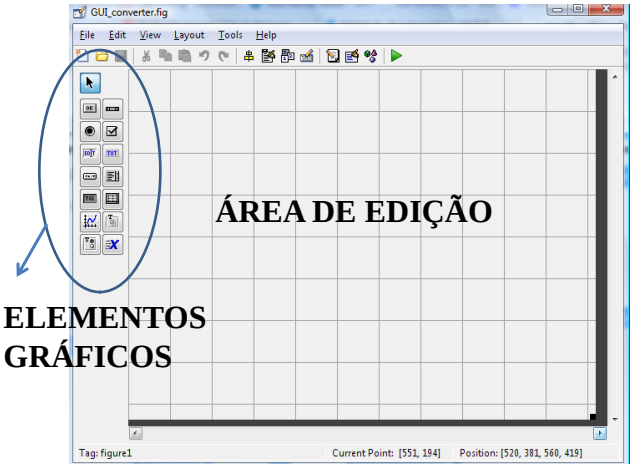
  template



Usando o GUI do Matlab

- Após escolher qual *template* usar a janela de desenvolvimento

  abrirá:



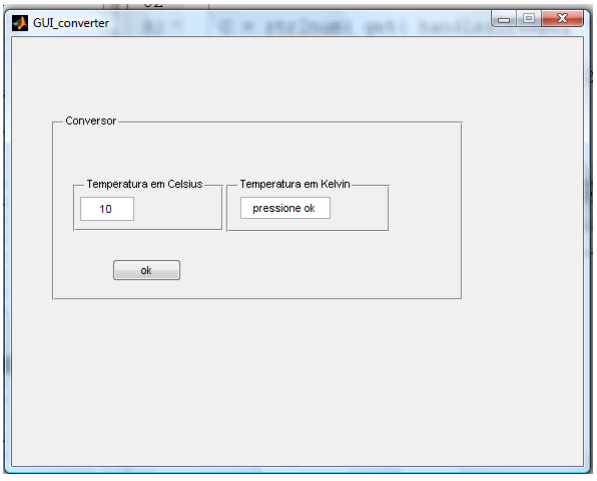
Ao salvar seu projeto será criado dois arquivos:

.fig: figura com elementos gráficos;
.m: conjunto de funções para manipulação da GUI

Exemplo

GUI_converter.m
GUI_converter.fig

- **GUI_converter**



Vamos praticar

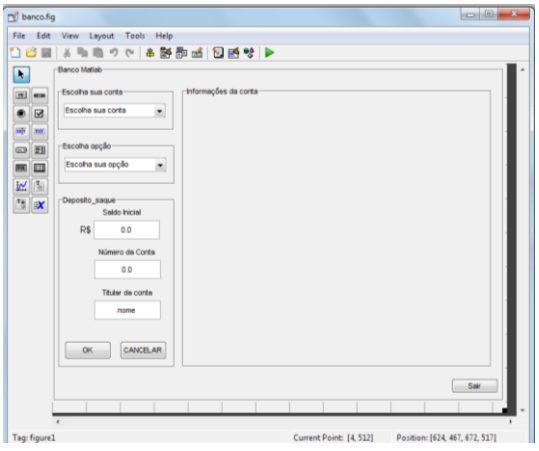
GUI_converter2.m
GUI_converter2.fig

- **Insira um novo display para mostrar a temperatura em Fahrenheit**

Vamos praticar mais ainda

banco.m
banco.fig

- **Faça o exemplo do banco em interface gráfica como a seguir:**



Otimização de códigos em Matlab

- Regra nº1: evitar usar ciclos for em MATLAB
  - Use vectorização

Exemplo:

```
x=0
for k=1:1001
   y(k)=log10(x)
   x=x+0.01
end
```

```
x=0:0.01:10
y=log10(x)
```

Otimização de códigos em Matlab

- Regra nº2: operações eficientes sobre vectores/matrizes
  - vectorização de operações

Exemplo:

```
j=0
for i=1:length(yr)
   if(yr(i)>1978) j=j+1;
   end
end
```

```
length(find(yr>1978))
```


Otimização de códigos em Matlab

- Outro exemplo:

```
tic
dx=pi/30
nx=1+2*pi/dx
for i=1:nx
   x(i)=(i-1)*dx
   y(i)=sin(3*(x(i)))
end
toc
```

```
toc=0.29
```

```
tic
x=0:pi/30:2*pi
y=sin(3*x)
toc
```

```
toc=0.005
```

Otimização de códigos em Matlab

- Regra nº3: se não for possível evitar um ciclo for
  - Use pré-alocação

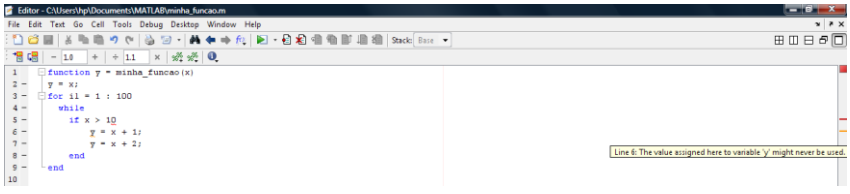
Exemplo:

```
v=zeros(200,1)

for i=1:200
   v(i)=rank(rand(i).^2);
end
```

```
for i=1:200
   v(i)=rank(rand(i).^2);
end
```

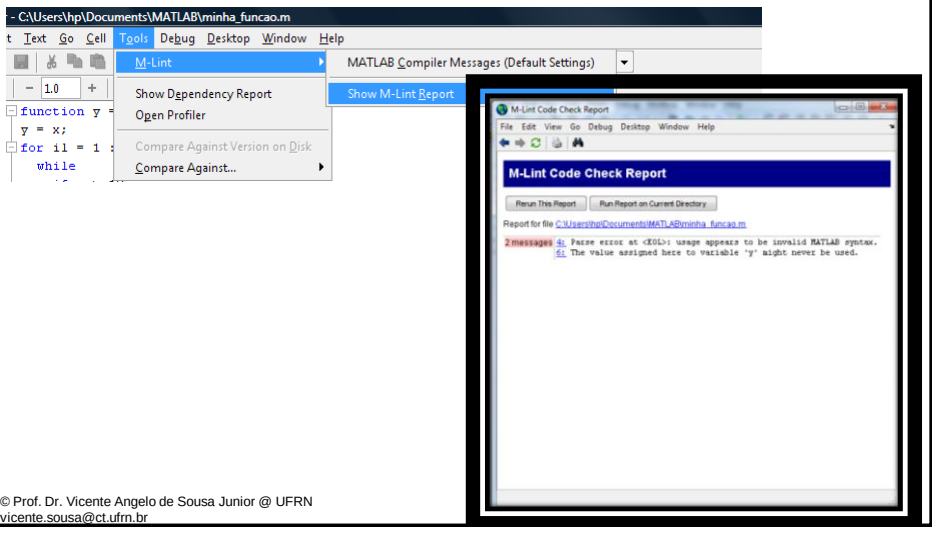

Profiling com o Matlab

- M-LINT: dicas e warnings para melhoria de códigos são sugeridos inline através de faixas coloridas na barra direita do editor do Matlab



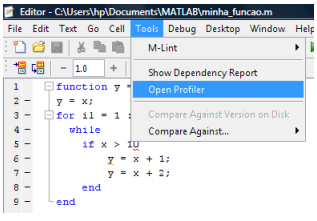
Profiling com o Matlab

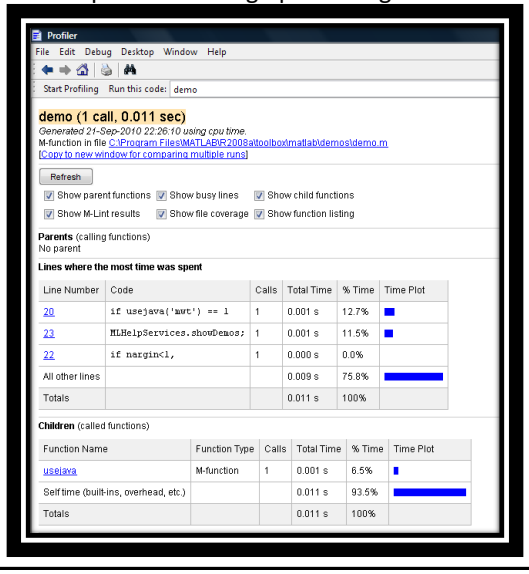
- Um relatório completo com todos as sugestões de melhoria pode ser gerado



Profiling com o Matlab

- Um relatório com a análise de desempenho do código pode ser gerado através do **profiler**





Boas práticas

- Comentar o código
- Preferir funções a scripts
- Vectorizar operações
- Evitar ciclos ou utilizar pré-alocação em memória
- Mais importante do que a velocidade:
  - código correto e legível

Toolboxes do Matlab

- A Empresa Mathworks paga vários pesquisadores para criar uma série de comandos a serem incorporados (e vendidos) junto com o Matlab
  - Um conjunto de comandos voltados a uma área específica é chamado de toolbox
  - O próximo slide mostra a lista de toolboxes disponíveis para o Matlab 2010b

Toolboxes do Matlab (2010b)

Aerospace Toolbox

Bioinformatics Toolbox

Communications Toolbox

Control System Toolbox

Curve Fitting Toolbox

Data Acquisition Toolbox

Database Toolbox

Datafeed Toolbox

Econometrics Toolbox

Filter Design Toolbox

Filter Design HDL Coder

Financial Toolbox

Financial Derivatives Toolbox

Fixed-Income Toolbox

Fixed-Point Toolbox

Fuzzy Logic Toolbox

Toolboxes do Matlab (2010b)

Global Optimization Toolbox

Image Acquisition Toolbox

Image Processing Toolbox

Instrument Control Toolbox

Mapping Toolbox

Model-Based Calibration Toolbox

Model Predictive Control Toolbox

Neural Network Toolbox

OPC Toolbox

Optimization Toolbox

Partial Differential Equation Toolbox

RF Toolbox

Robust Control Toolbox

Signal Processing Toolbox

Spreadsheet Link EX

Statistics Toolbox

Symbolic Math Toolbox

System Identification Toolbox

Vehicle Network Toolbox

Wavelet Toolbox

Toolboxes do Matlab (2010b)

Exemplos interessantes:

Optimization/Minimization of the Banana Function

Image processing toolbox/Image segmentation/Detecting a Cell Using Image Segmentation

Image processing toolbox/Measuring images features/Finding the Length of a Pendulum in Motion

Genetic Algorithm and direct search/genetic algorithm/Genetic Algorithm Options

Genetic Algorithm and direct search/Pattern Search /Pattern Search Climbs Mount Washington

Signal processing/Transforms/Discrete Fourier Transform

Signal processing/Miscellanneous/Modulation/Demodulation

Statistical/Probabillity distribution/Random Number Generation

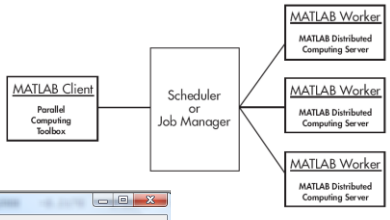
Statistical/Probabillity distribution/Distribution Functions

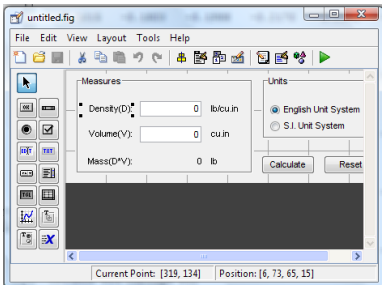
Statistical/Hypothesis test/Selecting a Sample Size

© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br

Outras facilidades do Matlab

- Operação conjunta de códigos C++ e Matlab
- MATLAB Compiler (standalone application)
- MATLAB Distributed Computing Server
- Parallel Computing Toolbox
- Interface gráfica do usuário





© Prof. Dr. Vicente Angelo de Sousa Junior @ UFRN
vicente.sousa@ct.ufrn.br

Livros sobre Matlab

Existe uma lista muito vasta de livros que utilizam o Matlab para fins didáticos.

Esses livros servem de apoio apoiar:

1. Aprendizagem em sala de aula
2. Pesquisa científica
3. Laboratórios de simulação
4. Estudo individual