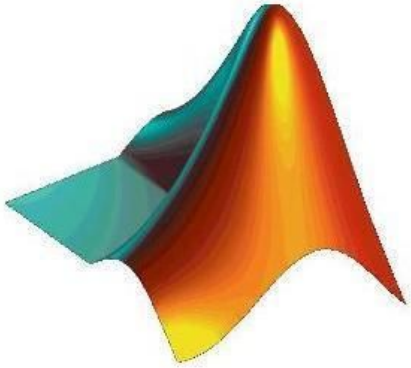
MATLAB



Prof. Diomar Cesar Lobão

Prof. José Flávio Feiteira

UFF – Volta Redonda, RJ. Nov. 2008

Introdução ao MATLAB

Conteúdo

- O que é o MATLAB?
- Trabalhando com Matrizes
- Expressões
- Controle de Fluxo
- Ambiente MATLAB
- Mais a respeito de Matrizes...
- Programação: arquivo M “Script”
- Funções
- Gráfico Básico 2-D
- Gráfico Básico 3-D
- GUI Introdução

Introdução ao MATLAB

O que é o MATLAB?

- Uma linguagem para computação de problemas técnicos
- Um sistema interativo o qual a estrutura de dados básica é a ***matriz***
- MATLAB vem de: ***''MATrix LABoratory''***

Uso Típico

- Matemática e computação, simulações em geral
- Desenvolvimento de Algoritmos
- Modelagem, simulação, e criação de protótipos
- Análise de dados, exploração e visualização de dados
- Gráficos
- Aplicações e desenvolvimento de interfaces: ***GUI***

Introdução ao MATLAB

Características

- Todo dado (objeto) em MATLAB é uma matriz
- As Funções podem retornar múltiplos valores
- As funções MATLAB são vetorizadas
- As matrizes em MATLAB são armazenadas em colunas

```
for j=1:4
   for i=1:5
        k=(j-1)*5+i
       A(k)=B(I,j);
    end
end
```

A Família MATLAB

- MATLAB
- MATLAB Extensions/ MATLAB Compiler, C/C++ Math Library
- Toolboxes
- Simulink (interactive system for simulating nonlinear dynamic systems)
- SIMULINK Extensions/ Accelerator, Real-Time Workshop
- Blocksets (são adendos do SIMULINK que prove adicionais “libraries” de blocos para aplicações especializadas como: comunicação, processamento de sinal, e sistemas de força.
- ...

Trabalhando com Matrizes

Entrando com Matrizes

```
A = [ 0.6 3 2; 5 10 2; 9 -6 7]
```

- Separe os elementos de uma linha com “blanks” ou “commas”
- Use um ‘’semicolon’’ para indicar o fim de cada linha
- Toda a lista de elementos é delimitada por [,]

Índices

```
A(1,2) + A(2,3)
```

- A(i, j) = o elemento da linha *i* e coluna *j*

Trabalhando com Matrizes

- Operador – dois pontos “Colon”

```
A(3, :)
A(:, 2)
```

- A(i, :) = a i-ésima linha de A
- A(:, j) = a j-ésima coluna de A

```
A(1:2, :)
A(1:2, 1:2)
A(4, 1) = 1
```

- A(1:2, :) = sub-matriz (1:2,:) de A

Trabalhando com Matrizes

- sum, transpose, diag: A = [0.6 3 2; 5 10 2; 9 -6 7]
  Soma, transposta e diagonal

```
sum(A)
A'
diag(A)
```

- sum(A) = soma cada elemento em coluna: 14.6000 7.0000 11.0000

- A' = A^T :0.6000 5.0000 9.0000
  3.0000 10.0000 -6.0000
  2.0000 2.0000 7.0000
- diag(A) = colhe os elementos da diagonal de A : 0.6000
  10.0000
  7.0000

Trabalhando com Matrizes

Exercício n.1

```
X = [ 1 2 3 4  5 6 7 8  9 10 11 12] (linha)
Y = X(:) (coluna) ou Y=X' (trasposta)
Z = reshape(X,3,4) , faça Z-X = ?
W = reshape(Y,4,3)
```

Problema n.1

- Qual é a diferença entre: diag([1 2]) e diag([1 0; 0 2]) :

a) Monta uma matriz (2X2) com 1 e 2 na diagonal e o restante é 0

b) Retorna os elementos da diagonal principal (1 e 2)

Mais sobre Matrizes

Algumas Matrizes úteis

```
zeros(2, 4)
ones(3)
eye(2)  ⇔ diag([1 1]); eye(3) ⇔ diag([1 1 1])
```

- zeros = gera uma matriz de zero
- ones = gera uma matriz de uns ‘’ones’’
- eye = gera uma matriz identidade

Concatenação de Matrizes

```
A = [1 2; 3 4]
B = [-1 -2; -3 -4]
C = [A B; B A] (neste caso simétrica...)
```

Mais sobre Matrizes

- Apagando Linhas e Colunas

```
A(:, 2) = [ ]
A(3, :) = [ ]
```

- Operações com Matrizes

```
dA = det(A)
 iA = inv(A)
 iA*A (resulta em...)
 ce = poly(A) => polinômio característico: det(lambda*EYE(SIZE(A)) - A)=0
 roots(ce) (Autovalores de A...)
```

  - det(A) = determinante de A
  - inv(A) = inversa de A

Mais sobre Matrizes

- Álgebra Linear, alguns exemplos

```
A = [1 2 3; 4 5 6; 7 8 9];
r = rank(A); s=cond(A);
B = lu(A);
C = svd(A);
D = qr(A);
```

- lu(A) : Fatorização LU da matriz A
- svd(A) : Decomposição do valor singular da matriz A (A = U*S*V‘)
- qr(A) : Decomposição QR da matriz A (Orthogonal-triangular decomposition.)
- rank(A) fornece uma estimativa do número de linhas ou colunas linearmente independentes da matriz A

Mais sobre Matrizes

- Arrays (operação com elementos)

```
A .* B
A .^ 2
A ^ 2
X = [2 1+i; 4i 2]
X'
X.'
```

- .* = Multiplicação de elemento-por-elemento
- .^ = Potência de elemento-por-elemento
- .' = transposta de não conjugado
- ./ .\ = divisão a esquerda; divisão a direita

A=[1 2; 3 4], B=[2 4; 6 8]; A./B=[0.5 0.5; 0.5 0.5]; A.\B=[2 2; 2 2]

Mais sobre Matrizes

Exercício n.2

```
A = [ 1+i  2+j  3; 4-4j  5i 6]
size(A) (n. de linhas, n. de colunas)
length(A)  % =max(size(A)) (a maior dimensão de A)
```

Problema n.2

- Dado a matriz A, calcule A ao cubo?

Janela de Comando – "Command Window"

- format

```
Format short => Precisão de 6 dígitos
A=[pi 1.4e4 -4/3]
Format long  => Precisão de 16 dígitos
A
```

  - format = muda entre 6 e 16 dígitos o formato de output

- ans

```
ans
```

  - ans = variável do sistema que armazena os mais recentes resultados

Janela de Comando – "Command Window"

- help

```
help format
help help
```

  - help = retorna ajuda descritiva de um tópico específico.

- ↑ (tecla)

```
A(1,2)
  ↑
```

  - ↑ = retorna o comando anterior
  - ↑ = retorna o próximo comando
  - ↑ = ... até o último comando

Janela de Comando – "Command Window"

Variáveis

```
pi_value = 3.141592
string1 = 'Hello, Matlab!'
string1
pi
eps
```

- Nenhum tipo de declaração ou dimensão é necessário
- Nome variável = a letra + letra/dígito/underscores
- Os primeiros 31 caracteres de uma variável é usado
- Os nomes das variáveis são sensíveis: *Ama ≠ ama*
- pi = 3.1415926535897.... (16 digitos formato *'long'*)
- eps = Precisão relativa de ponto flutuante (intrínseco do Matlab: 2.2204e-016).
- realmin, realmax, inf, nan

Janela de Comando – "Command Window"

- Exercício n.3

```
a = (3==2);

b = a
```

Qual é o tipo de a e b?

- Problema n.3
  - Qual é a diferença entre os comandos: cd e pwd?

Expressões e Funções Matemáticas

Números

```
0.0001
1.602e-5
3i
0.41E2j
```

- Todos os números são armazenados internamente usando: *long* formato
- i = 2; 3i; 3*i; 1.602e-5 = 1.602×10-5

```
i = 2
3i
3*i
```

- i, j = unidade imaginária

• NÃO USE VARIÄVEIS i ou j EXCETO PARA ÍNDICES DE LOOP!!!

Expressões e Funções Matemáticas

- Operadores

```
a = 4
b = 1 + 2j
c = a^2
d = a^-0.5
e = b'
```

- ^ = potência
- ' = Transposto complexo conjugado

```
A = [ 3 2; 2 1]
B = [ 2 1; 1 0]
C = A*B
A^2 - A*A  Obs: A.^2 ≠ A^2 ???
```

Expressões e Funções Matemáticas

- Funções intrínsecas

```
sin(a)
abs(b)
sqrt(c)
exp(1.0)
log(10.0)
a = sqrt(-1.0)
b = sqrt(i)
```

- sqrt = √
- exp(x) = e^x
- log = Logaritmo natural

Expressões e Funções Matemáticas

Exercício n.4

```
A = [0 30; 45 60]*pi/180
B = sin(A)
C = A^-1 - inv(A)
```

Problema n.4

1) Use: exp(A) and expm(A), com A=eye(2), qual a diferença?.
2) Escreva usando Matlab o cálculo do ângulo entre dois vetores: a=[2,1]; b=[1,2];

```
 angle=  dot(a,b)/(norm(a)*norm(b));
theta = acos(angle); Resultado no intervalo [0,pi].
```

Ambiente MATLAB

- Espaço de Trabalho – “Workspace’’

```
save my.dat
clear
clc
who
load my.dat -ascii
whos
```

- workspace = área de memória usada pelo MATLAB
- *whos* = lista as variáveis utilizadas no workspace
- *clear* = Limpa o workspace
- *save* = salva as variáveis do workspace num arquivo.
- *load* = carrega todas as variáveis de um arquivo.

Ambiente MATLAB

save

```
save myprog1 a b c
save amVar_A.dat -ascii A
```

- save myprog1 a b c = salva as variáveis a, b, c em myprog1.mat
- save amVar_A.dat -acii A = salva A em amVar_A.dat com formato de 8-dígitos texto

Disk file

- dir = dir/ls
- type filename.ext = type/cat
- delete filename.ext = del/rm
- diary (help dairy...)

Gráficos : Básico 2-D

- Criando um gráfico com comando: Plot

```
t = 0:pi/100:2*pi;
    y = sin(t);
          plot(t, y)
```

- plot(x, y) = produz um gráfico 2D de: y vs. x

```
figure(1);
   y2 = sin(t - 0.25);
          plot(t, y2)
              hold on
                  plot(t, y+y2)
```

- figure = abre uma nova janela de figura
- hold on/off = Permite adicionar gráficos aos já plotados

Gráficos: Básico 2-D

- Comando: plot

```
y3 = sin(t - 0.5);
        plot(t, y, t, y2, t, y3)
```

- plot(x1, y1, x2, y2, ...)

```
plot(t, y, 'c--', t, y2, 'r.')
```

- plot(x1, y1, 'color_style_marker', x2, y2, 'color_style_marker, ...)
- color_style_marker = a 1/2/3 caractere string.
- color = c, m, y, r, g, b, w, k
- style = -, --, :, -., none
- marker = +, o, *, x (use help plot...)

Gráficos: Básico 2-D

- Comando: Subplots

```
clf
y4 = cos(t);
subplot(2, 2, 1); plot(t, y)
    subplot(2, 2, 2); plot(t, y2)
       subplot(2, 2, 3); plot(t, y3)
          subplot(2, 2, 4); plot(t, y4)
```

- clf = limpa a janela da figura
- subplot = mostra múltiplos gráficos na mesma janela

Gráfico: Básico 2-D

- Comando: Axis

```
clf
t = -pi:pi/100:pi;
   y = sin(t);
        plot(t, y)
        grid on
              axis([-pi pi -1 1])
              xlabel('\pi \leq \itt \leq \pi')
              ylabel('sin(t)')
```

- grid on = desenha linhas de malha
- axis([xmin xmax ymin ymax]) = fixa a escala dos eixos
- xlabel/ylabel = adiciona texto (label) do lado dos eixos: X e Y

Gráfico: Básico 2-D

- ***ginput:*** [X,Y] = GINPUT(N) colhe N pontos do corrente eixo e retorna nos vetores X e Y de comprimento N.
- ***gtext:*** Escreve na figura o texto posicionado pelo cursor.

```
  clear all; clc; close all;
figure(1);
%
plot(1:2,1:2,'s'); hold on
for i=1:2
    [x1,y1]=ginput(1)
    title(['Test', ' x = ', num2str(x1),' y = ',num2str(y1)]);
    gtext([' x = ', num2str(x1),' y = ',num2str(y1)]);
end
```

Gráfico: Básico 2-D

Exercício n.5

```
figure;
plot((0:0.5:20),sin(0:0.5:20),'ro');
Hold on;
plot((0:0.5:20),sin(0:0.5:20),'b:')
figure(3); t=(0:0.5:20)';
plot(t, sin(t),'g.',t,cos(t),'m+');
title('Graph of sin(t)')
```

Problema n.5

- No exemplo use os comandos: *legend, gtext, line e text*

Arquivo do Tipo Script: myfile.m

myfile.m

- Arquivos que contém código escrito na linguagem MATLAB
- Nome de um arquivo = ***myfile.m***
- Nome de arquivos não são sensíveis a maiúsculo/minúsculo
- Há dois tipos de arquivos ***myfile.m*** => Script puro e Script função

Processo para escrever um arquivo .m

- Crie um arquivo ***myfile.m*** usando um editor de texto
- Chame o arquivo ***myfile.m*** da *janela de comando* ou de dentro de outro ***.m***

Script .m

- Crie um arquivo .m usando editor de texto

```
clear
t = -pi:pi/100:pi;
y = sin(t);
plot(t, y)
xlabel('t [rad]')
ylabel('sin(t)')
```

  - salve o arquivo com o nome myplot1.m

- Chame-o na janela de comando como:

```
>>myplot1
>>whos
```

  - O arquivo script automaticamente executa os comandos

Comandos de Controle de execução

- Quatro comandos de controle de execução
  - ***if***
  - ***switch***
  - ***while***
  - ***for***

  Obs*: break, continue, return*
- Operadores relacionais
  - < <= > >= == ~=
  - true = 1 false = 0
- Operadores Lógicos
  - & | ~

Comandos de Controle

- *if, elseif, else, end*

```
n = 3;
if n > 0
  disp('Positive')
elseif n < 0
  disp('Negative')
else
  disp('Zero')
end
```

- disp = mostre um dado string/array
- end => nunca deve ser omitido!!!
- if ~ end (abre com ***if*** e fecha com ***end***)
- if ~ else ~ end

Controle de execução

- switch

```
n = 3;
switch n
  case 1
    disp('1')
  case {2, 3, 4}
    disp('2 or 3 or 4')
  otherwise     => caso de nenhuma condição seja satisfeita
    disp(‘qualquer outra coisa')
end
```

- ‘otherwise' pode ser omitido

Controle de execução

- *for*

```
for i = 1:5
  for j = 1:3
    A(i, j) = 1/(i+j);
  end
end
```

- 'for' loop executa um número pré-determindo de repetição
- ***for índice = início:incremento:fim***
- ***comandos***
- ***end***
- O incremento default é: 1

Controle de execução

- *while*

```
sum = 0;
n = -2;
      while n <= 5
        sum = sum + n;
        n = n + 1;
      end
sum
```

- *'while'* loop executa comandos repetidamente enquanto a *expressão de controle* é true(1)
  - while *expression*
    comandos
    end

Funções definidas em: Arquivo *.M

- Exemplo de função simples

```
function mv = avg(x)
% Media de um vetor
[m, n] = size(x);
if ~(m == 1) & ~(n == 1)
    error('Input deve ser um vetor')
end
mv = sum(x) / length(x);
```

- Grave este exemplo no arquivo ***avg.m***

Exemplo de uso:

```
>>A = [1 2 3 4];  ma = avg(A)
>>B = [1 2; 3 4]; mb = avg(B)
```

Funções definidas em: Arquivo *.M

- Anatomia de uma função *"function"*

```
function [r1, r2] = fname(x)
% H1 line
% Function help text
...
Function body
...
r1 = ... Variável de retorno
r2 = ...  Variável de retorno
return;
```

- Variável de entrada: *x*
- Variável local e de retorno: *r1, r2*

Funções definidas em: Arquivo *.M

- Nome da função
  - possue as mesmas regras de definição de variáveis
  - UMA função em UM arquivo -) sub-função = subfunction
  - nome do arquivo > nome da função
  - o nome de arquivo M e função deve ser os mesmos
- Resolução
  - Checar para ver se o nome é uma variável
  - Checar para ver se o nome é uma sub-função = subfunction
  - Checar para ver se o nome é uma função no corrente diretório
  - Checar para ver se o nome é uma função definida no caminho de procura do MATLAB

Funções definidas em: Arquivo *.M

- Variáveis: local e global

```
function value=tg(x, y)
    global Z
        x1 = x^2;  y1 = y^2;  Z1 = Z^2
        value=x1+y1+z1;  return
```

- salve como: ***tg.m***
- Declarar a variável Z como *global* em toda a função que requerê-la para ter acesso a ela
- Definir o comando *global no* topo do arquivo de chamada da função

```
global Z
x = 3;  y = 4;  Z= 5;
tg(x, y) % chamada da funcao tg(x,y)
x, y, Z
```

- salvar como: ***test.m*** (c;odigo principal) executá-lo como: ***test***

Funções definidas em: Arquivo *.M

- eval e feval

```
for n = 1:12
    eval(['M' num2str(n) ' = magic(n)'])
end
```

- ***eval*** : executa conteúdo de um colchete na tela de comando

```
for n = 0:pi/10:pi
    m=feval('sin',n);
    disp([n m])
end
```

- ***feval*** : avalia uma função pelo nome e argumentos no colchete

Gráfico Básico 3-D

- plot3(x,y,z)
  - desenha x,y,z em espaço 3-dimensional

```
t=(0:pi/10:8*pi);
   x=exp(-t/20).*cos(t);
   y=exp(-t/20).*sin(t);
   z=t;
   figure(1);  plot3(x,y,z);
         xlabel('x'); ylabel('y'); zlabel('z');
```

- view(Az,El)
  - Especifica o ponto de visada em 3D

```
view(20,70);   view(2);   view(3);
```

Gráfico Básico 3-D

- meshgrid(x,y)
  - gera um domínio para ser desenhado

```
x=(-5:0.2:5);
y=(-5:0.2:5);
    [X,Y] = meshgrid(x,y);
          Z= X .* exp(-X.^2 - Y.^2);
              size(X); size(Y); size(Z);
```

- mesh(Z)
  - desenha uma malha colorida 3D

```
mesh(Z); mesh(X,Y,Z);
```

Gráfico Básico 3-D

- surf
  - desenha uma superfície 3D

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

- contour
  - desenha as isso-curvas de uma gráfico 3D

```
[cs,h]=contour(Z)
        clabel(cs,h);
            colorbar;
```

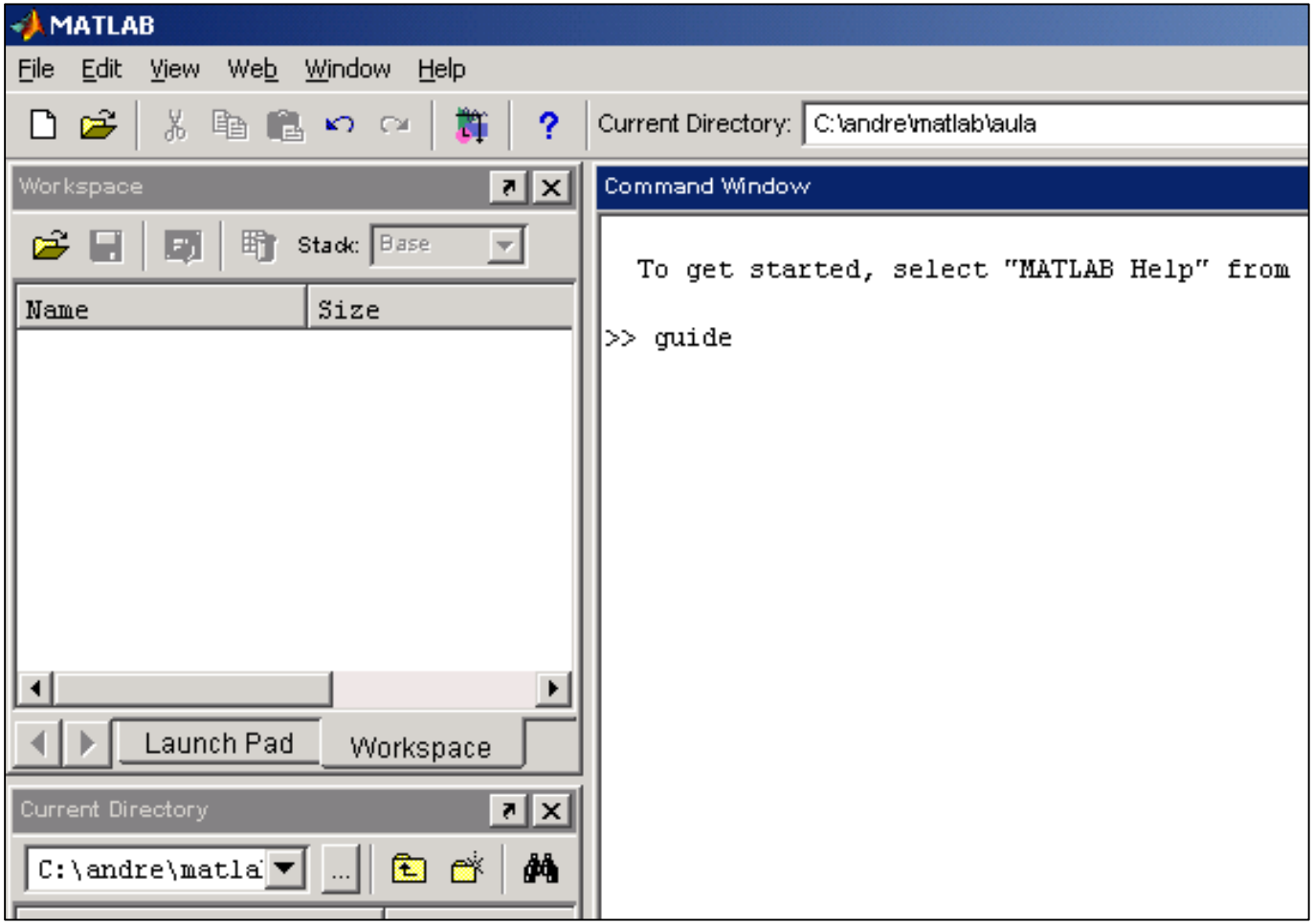
Gráfico Básico 3-D

- Exercício

```
clear all; clear all;
    [x, y, z] = sphere(25);
                surf(x-3,y-2,z);
                    hold on;
                        surf(x*2,y*2 z*2);
                            grid on; cloorbar ; hold off;
```

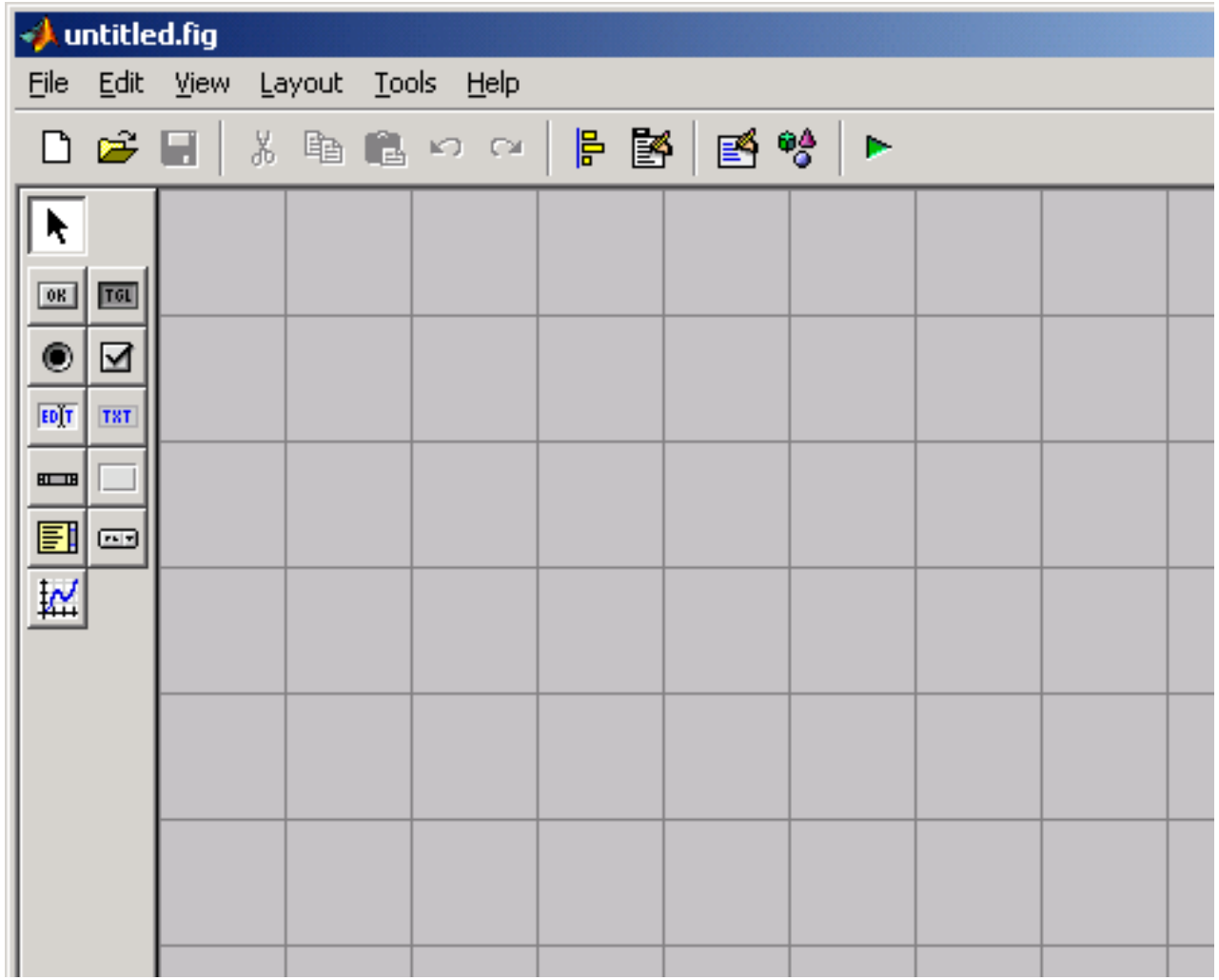
GUI – Interface Gráfica com o Usuário

- No prompt digite guide e tecle ENTER

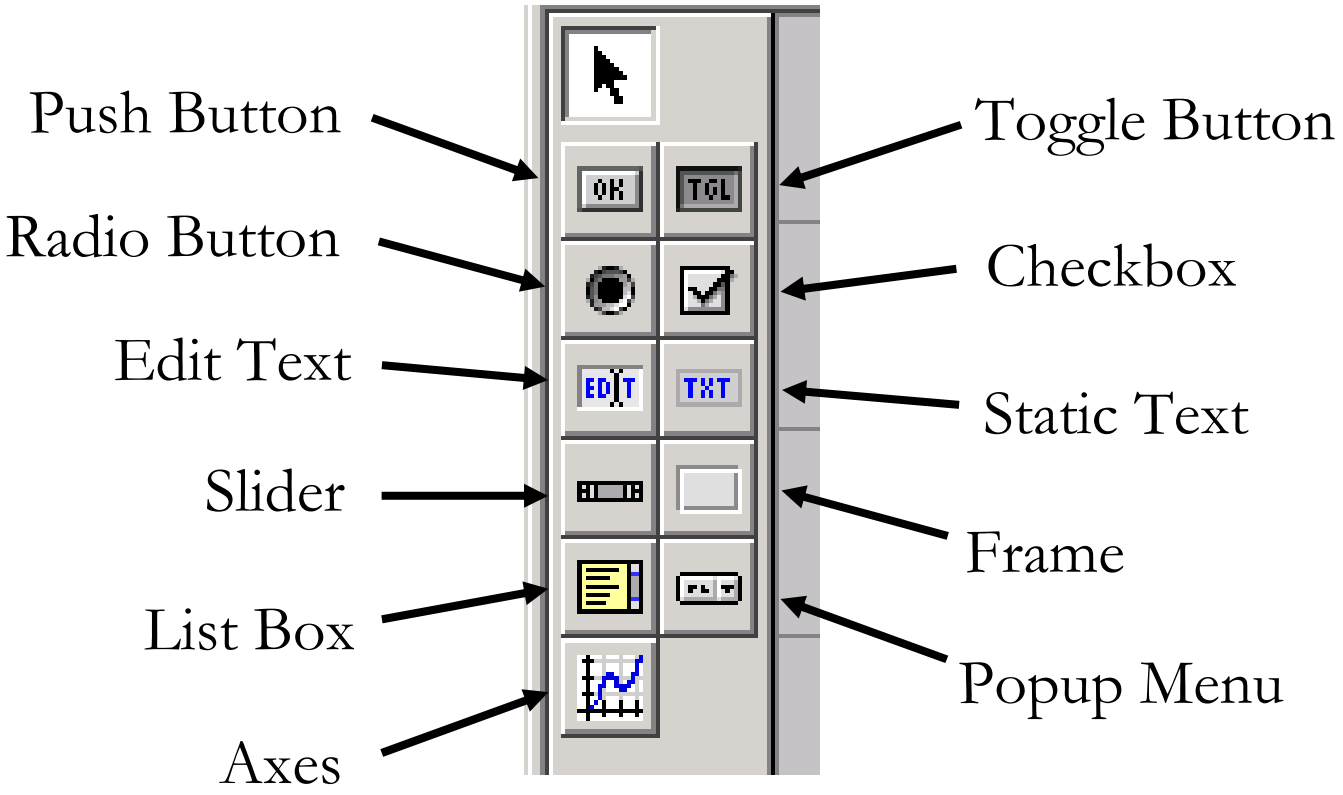


GUI – Interface Gráfica com o Usuário

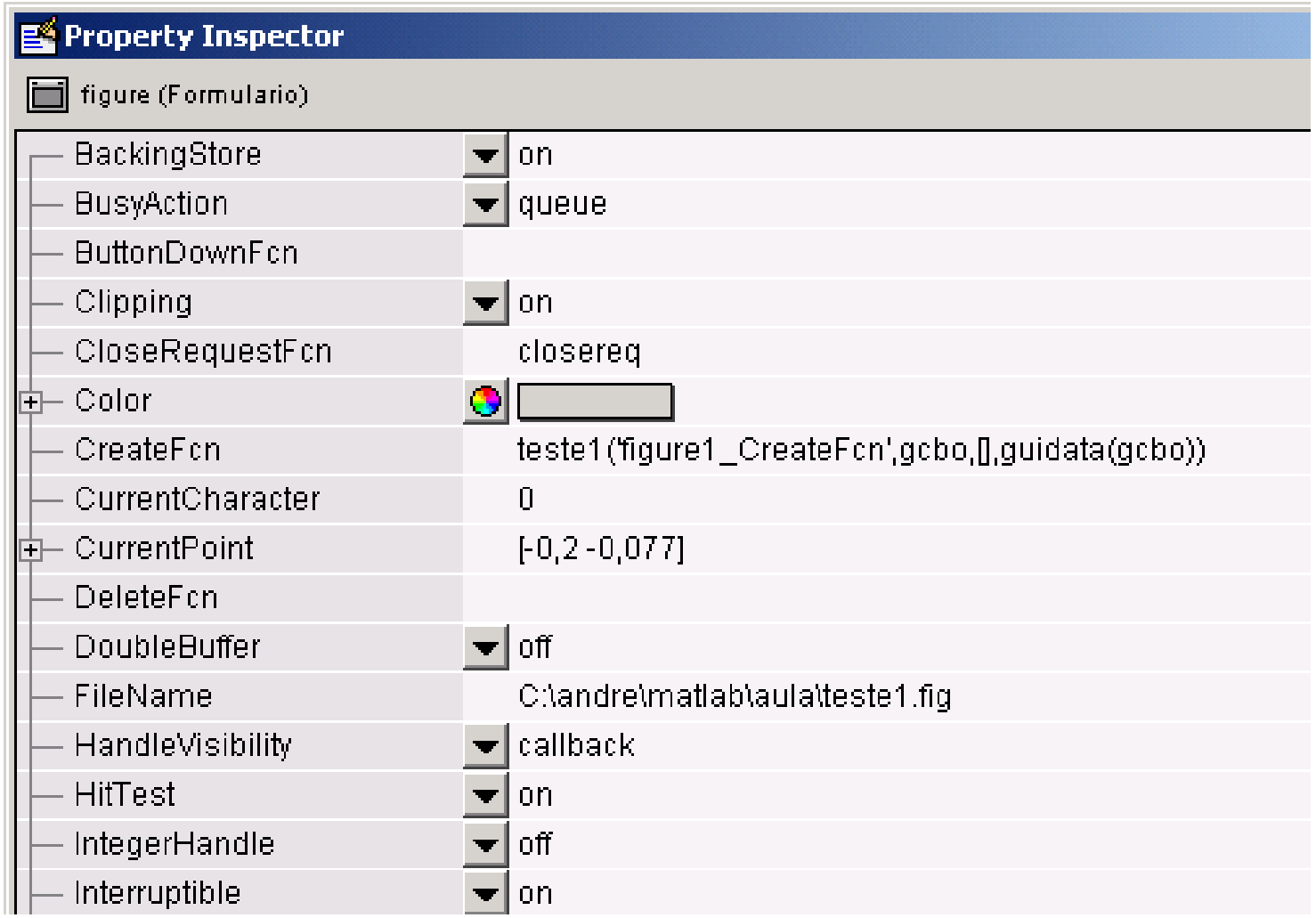
- Surge a seguinte janela:



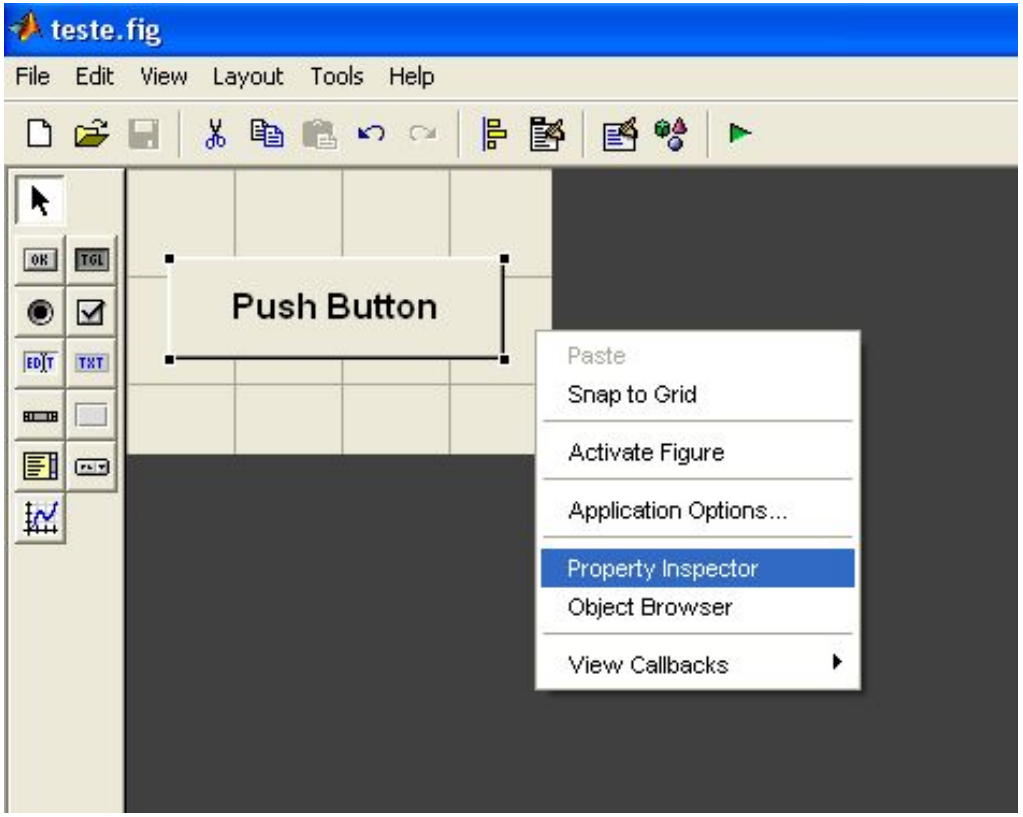
Objetos disponíveis:

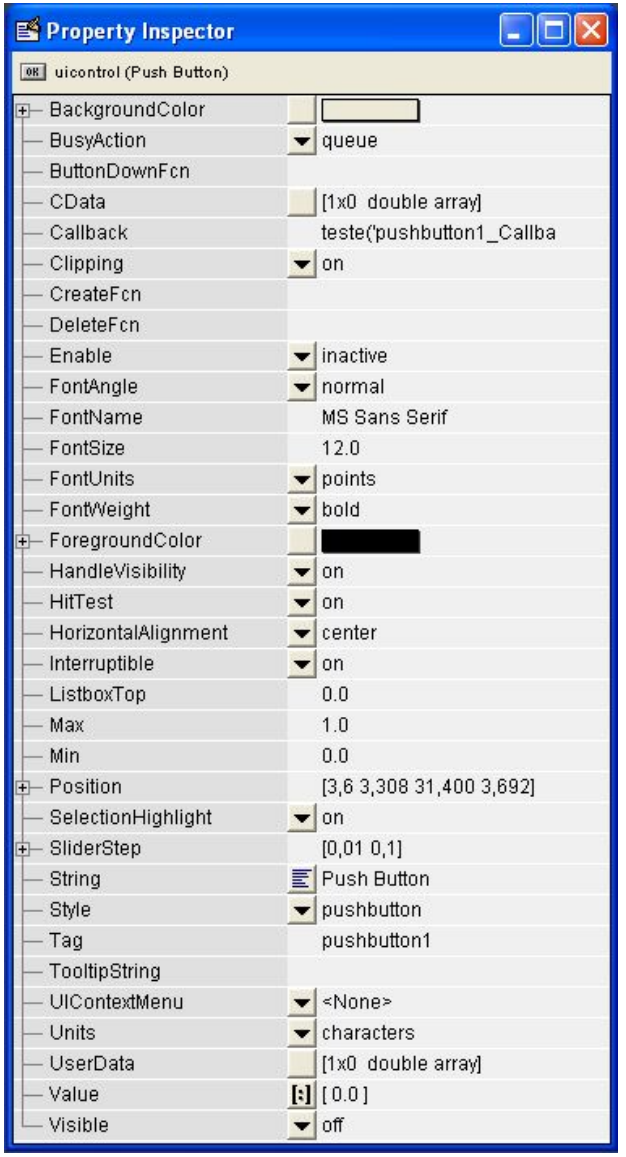


Formas de Gerenciamento:

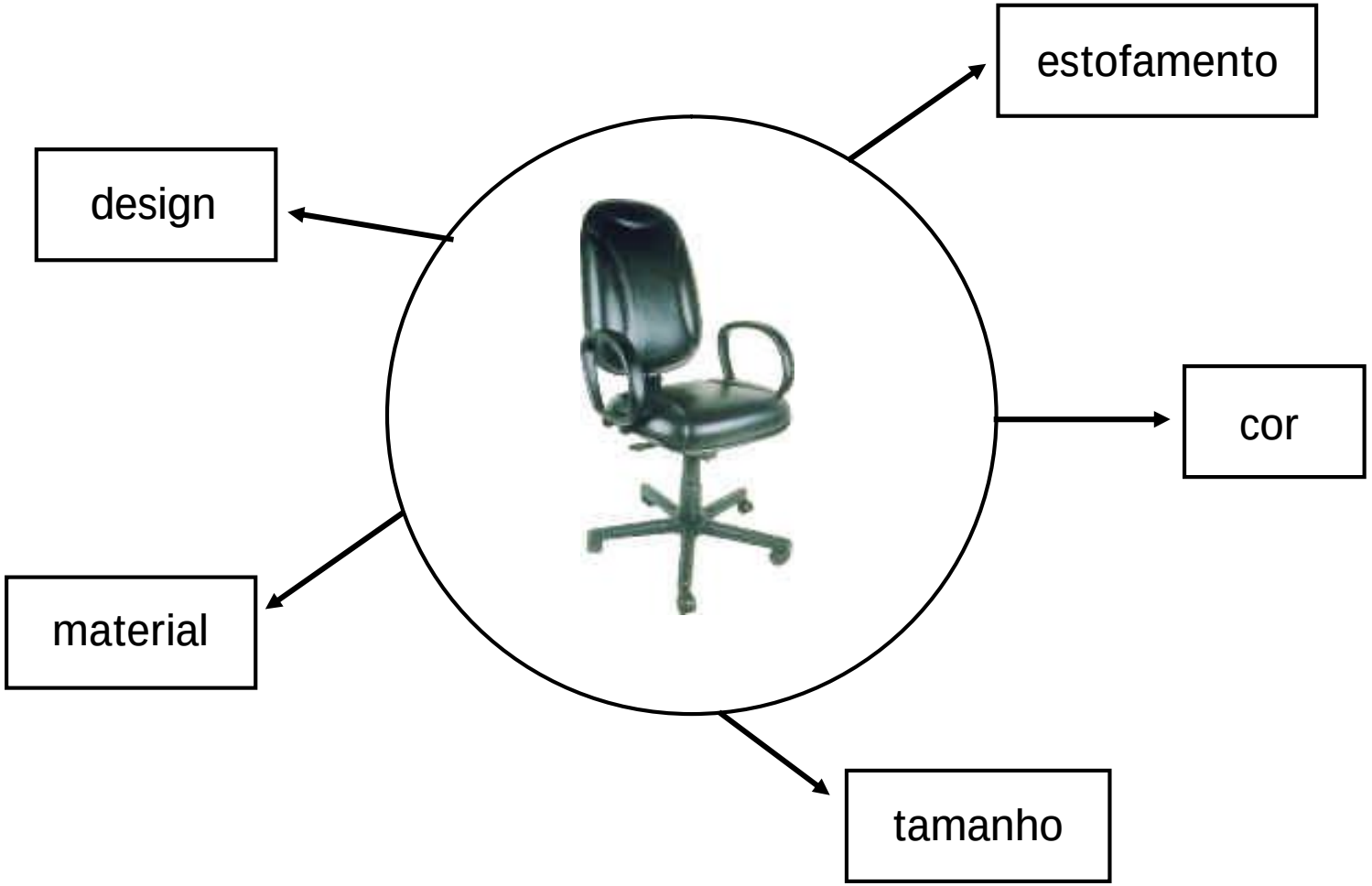


Arraste um objeto para o formulário

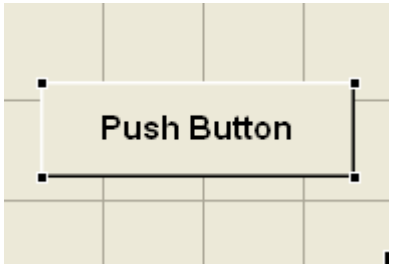


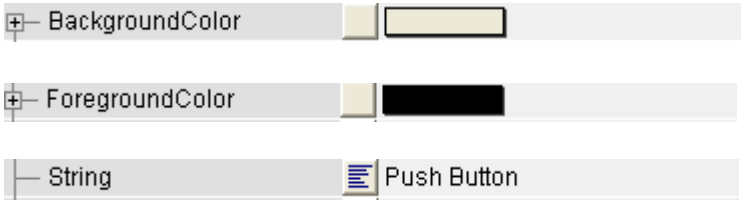


Propriedades de um Objeto:



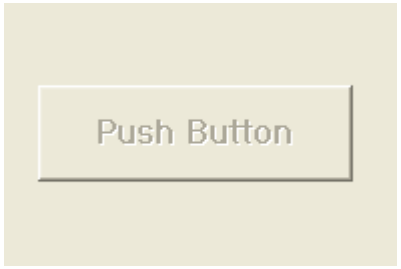
Propriedades do Objeto BOTÃO:



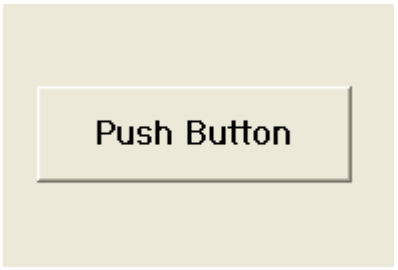


Enable

on

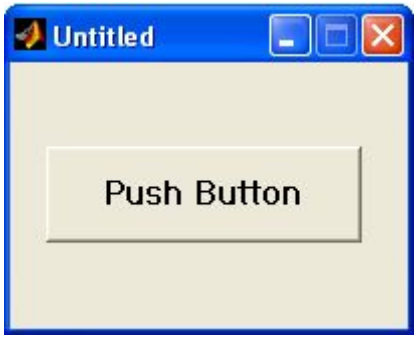


off



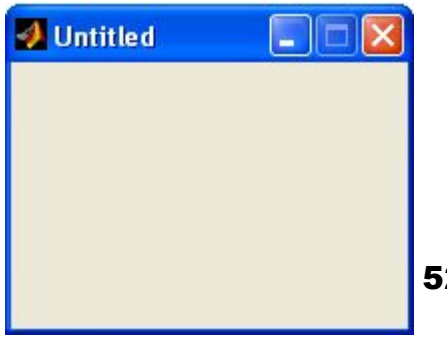
inactive

Visible

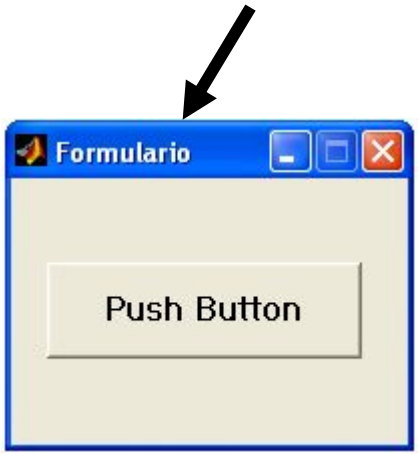


on

off



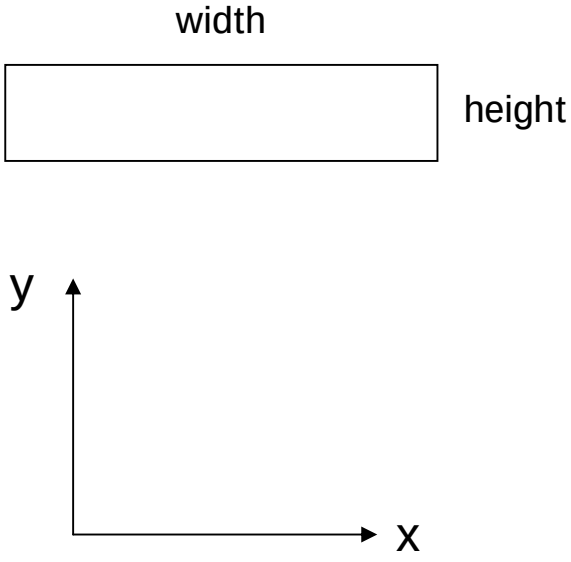
Propriedades do Objeto Formulário:



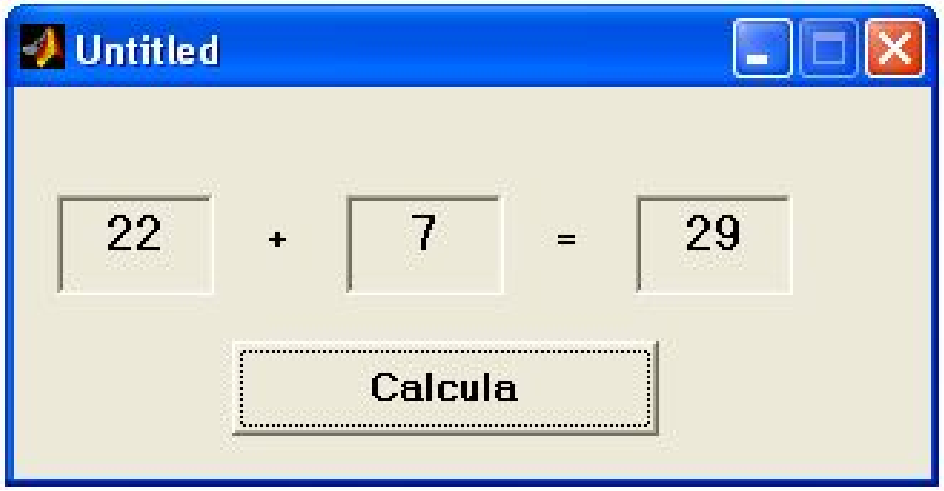
Name

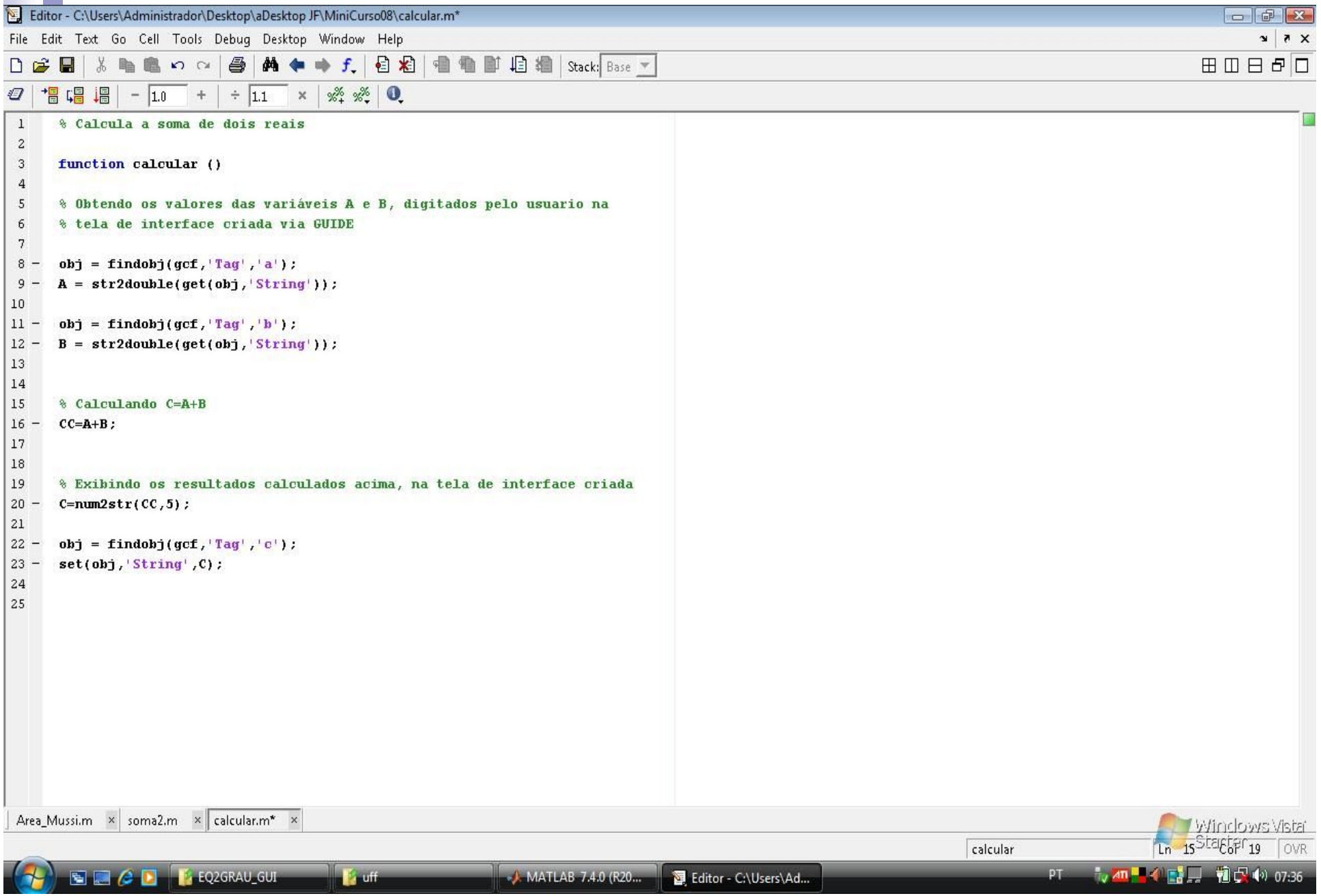
Posição e Tamanho iniciais

Position	[0,8 53,923 39,6 10,231]
x	0.8
y	53.923
width	39.6
height	10.231



- Construir o seguinte exemplo:




Editor - C:\Users\Administrador\Desktop\aDesktop JF\MiniCurso08\calcular.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
% Calcula a soma de dois reais
function calcular ()
% Obtendo os valores das variáveis A e B, digitados pelo usuario na
% tela de interface criada via GUIDE
obj = findobj(gcf,'Tag','a');
A = str2double(get(obj,'String'));
obj = findobj(gcf,'Tag','b');
B = str2double(get(obj,'String'));
% Calculando C=A+B
CC=A+B;
% Exibindo os resultados calculados acima, na tela de interface criada
C=num2str(CC,5);
obj = findobj(gcf,'Tag','c');
set(obj,'String',C);
Area_Mussi.m
soma2.m
calcular.m*
calcular
Ln 15 Col 19
OVR
EQ2GRAU_GUI
uff
MATLAB 7.4.0 (R20...
Editor - C:\Users\Ad...
PT
07:36

Referências

[1] Matsumoto, É. Y., Matlab 6.5 Fundamentos de Programação. Ed. Érica. 2002.

[2] Gilat, A., *MATLAB: An Introduction with Applications 2nd Edition*. John Wiley & Sons. 2004. ISBN 978-0-471-69420-5.

[3] Quarteroni, A.; Fausto S., *Scientific Computing with MATLAB and Octave*. Springer. 2006. ISBN 978-3-540-32612-0