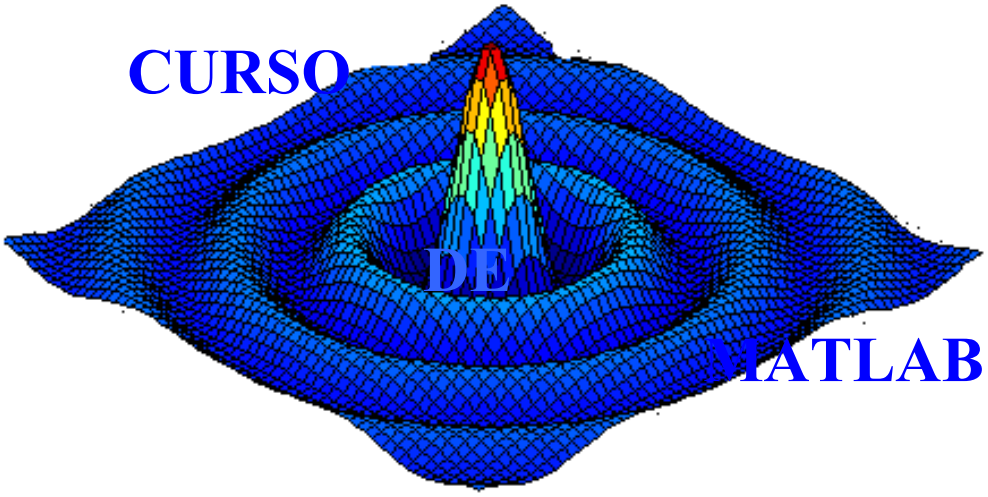
UNIVERSIDADE FEDERAL DO PARÁ
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PROGRAMA DE EDUCAÇÃO TUTORIAL
SEMANA DOS 40 ANOS DE ENGENHARIA ELÉTRICA

CURSO DE MATLAB



NÍVEL BÁSICO
CAPÍTULO IV

PROGRAMA DE EDUCAÇÃO TUTORIAL
ENGENHARIA ELÉTRICA -UFPA
BELÉM-2004

CAPÍTULO 4 – INTRODUÇÃO À PROGRAMAÇÃO EM MATLAB

ÍNDICE:

4.1 – INTRODUÇÃO

Desenvolver soluções no MATLAB é, sem dúvida, muito mais eficiente e rápido do que qualquer outra linguagem de programação. Neste capítulo, apresentaremos os conceitos básicos de programação no MATLAB. Os fluxos de controle encontrados na maioria das linguagens de programação são tratados neste capítulo.

As linhas de comando de fluxo de controle permitem que MATLAB seja utilizado como uma linguagem de programação completa de alto-nível.

4.2 – CONCEITOS E TIPOS DE M-FILES

Um dos assuntos mais importantes, a criação de arquivos-M é descrita neste capítulo. MATLAB é usualmente acionado por um comando; quando se entra com uma simples linha de comando, MATLAB a processa imediatamente e mostra o resultado. O MATLAB também pode executar uma sequência de comandos que está armazenada em um arquivo. Arquivos de programa do MATLAB têm extensão '.m' e, por este motivo, são usualmente chamadas de 'm-files'.

Dois tipos de arquivos-M podem ser usados: scripts e funções. **Scripts, ou arquivos script**, realizam longas sequências de comandos. **Funções, ou arquivos função**, permitem adicionar novas funções à funções já existentes. A maior parte do poder do MATLAB se deve ao fato de se poder criar novas funções que resolvam problemas específicos.Veremos a seguir os dois tipos de m-files.

4.2.1 - Arquivos Script

Quando um script é chamado, MATLAB simplesmente executa os comandos encontrados no arquivo. As linhas de comando de um arquivo script operam globalmente com os dados que estão no espaço de trabalho. Scripts são úteis na realização de análise, solução de problemas, ou no projeto de longas sequências de comando, o que se torna

```
f = [1 1]; i = 1;
while f(i) + f(i+1) < 1000
f(i+2) = f(i) + f(i+1);
i = i + 1;
end
plot(f)
```

cansativo para ser feito interativamente. Como um exemplo, suponha um arquivo chamado fibo.m que possui os comandos:

Digitando a linha de comando fibo faz com que MATLAB execute os comandos, calculando os 16 primeiros números da série de Fibonacci, e crie um gráfico. Após a execução do arquivo estar completa, as variáveis f e i ficam mantidas no espaço de trabalho. Os demos fornecidos pelo MATLAB são bons exemplos de como se utilizar scripts para realizar tarefas mais complicadas. Para usá-los, basta digitar demos no prompt do MATLAB.

4.2.2 - Arquivos Função

Um arquivo-M que contém a palavra function no ínicio da primeira linha é um arquivo função. Uma função difere de um script pelos argumentos que devem ser passados e pelas variáveis que são definidas e manipuladas, que são locais à função e não podem ser operadas globalmente no espaço de trabalho. O arquivo mean.m é um exemplo de um arquivo função que possui as seguintes linhas de comando:

```
function y = mean(x)
% MEAN Average or mean value
% For vectors, Mean(x) returns the mean value
% For matrices, MEAN(x) is a row vector
% containing the mean value of each column.
[m,n] = size(x);
if m == 1
m = n;
end
y = sum(x)/m;
```

4.3 – COMANDO *EDIT*

O MATLAB possui um ambiente para edição de programas que pode ser acionado de duas formas: Na linha de comando digitando **edit,** ou nos ícones da barra de ferramenta. É com ele que criaremos as nossas funções, e tem algumas características:

- O editor abre uma pasta para cada arquivo.
- [*File*], comandos de criação abertura e salvamento de arquivos de programação
- Os textos são exibidos em cores diferentes conforme sua classe sintática:
    - funções e variáveis: preto
    - textos de comentário (indicados por '%'): verde
    - palavras reservadas (ex.: for, end, if): azul
    - mensagens de erro: vermelho

4.4 – ELEMENTOS DE PROGRAMAÇÃO

4.4.1 – Sintaxe Básica

As funções no MATLAB aceitam múltiplos parâmetros de entrada e retornam múltiplos parâmetros de saída. A sintaxe básica de definição de função é:

```
function  [ Ps1, ... , PsN ] = nome-funcao ( Pe1 , … , PeM )
<expressão1>
      …
<expressãoR>
```

função.

⇨ A chamada da função segue este formato:

```
>>[Ps1, ... , PsN] = nome_função(Pe1 , … , PeM)
```

Exemplo: Criar uma função chamada **muv** com as seguintes caracteríticas:
Parâmetros de entrada => s0: espaço inicial, v0: velocidade inicial, a: aceleração constante, t: tempo.
Parâmetros de saida => s: vetor espaço, dado por s = s0 + v0 * t + a/2 * t^2
v: vetor velocidade , dado por v = v0 + a * t

4.5 – CONTROLE DE FLUXO

4.5.1 – Expressões Booleanas

O MATLAB possui características de linguagem de programação estruturada que utiliza expressões booleanas para implementar testes e tomadas de decisão.

- **Operadores Relacionais:** O MATLAB tem operadores relacionais que podem ser usados para comparar duas matrizes de mesma ordem ou para comparar uma matriz e um escalar, como os mostrados a seguir:

Operador	Descrição
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a
= =	Igual a
~ =	Não igual a (diferente)

- **Operadores Lógicos:** Podemos combinar expressões usando os operadores lógicos do MATLAB. Os operadores são representados pelos seguintes símbolos.

Operadores	Descrição
&	E(and)
|	Ou (or)

```
>>i=10; j=15;
>>i = = j
>>ans = 0
 >> i ~ = j
>> ans = 1
 >>i < j
>>ans = 1
```

4.5.2 – Estruturas de Seleção

```
If <expr.bool.1>
      <expressão1>
elseif  <expr.bool.2>
           <expressão2>
         else
           <expressão3>
end
```

4.5.3 - Estruturas de Repetição

Apresentaremos as sintaxes que implementam estruturas de repetição. Fazemos isso utilizando as expressões booleanas descritas anteriormente. Os principais comandos são:

while

```
While <expr.bool.>
                  <expressão1>
                  …
                  <expressãoN>
end
```

for

```
For <valor>=<inic>:<incr>:<fim>
          <expressão1>
...
          <expressãoN>
end
```

4.6 – Exemplos:

```
% CRIANDO UM PROGRAMA EXEMPLO DE GRÁFICO 3D

n = 30;
m = 30;
for i = 1:m
   for j = 1:n
      a(i,j) = sqrt(i+j);
   end
end
b = [a+0.5 a'-0.5;
(a.^2)/5 ((a'-0.1).^2)/2];
mesh(b)
```

```
% UTILIZANDO O LAÇO WHILE
a = l; b = 15;
while a<b,
   clc
   a = a+l
   b = b-l
   pause(l)
end
disp('fim do loop')
```

% USO DA DECLARAÇÃO IF NO MATLAB

```
for i = l:5,
   for j = l:5,
      if i = = j
         A(i,j) = 2;
      else if abs(i-j) = = 1
         A(i,j) = -1;
      else
         A(i,j) = 0;
      end
   end
end
```

PROGRAMA PARA ANÁLISE EM CONTROLE

```
a=[1 4 16 36];
b=[0.6 2 5.6 9.6];
t=0:0.1:8;
y=zeros(81,4);
for i = 1:4;
num=[0 0 a(i)];
den= [1 b(i) a(i)];
y(:,i)=step(num,den,t);
     end
plot(t,y(:,1),t,y(:,2),t,y(:,3),t,y(:,4))
grid
title('Curvas de resposta ao degrau para os 4 casos')
xlabel('t(s)')
ylabel('saídas')
```

4.7 – EXERCÍCIOS

1) Determine se as expressões nos problema 1 a 8 são verdadeiras ou falsas. Depois, verifique suas respostas usando o MATLAB. Lembre que ao verificá-las , você precisa entrar com a expressão. Suponha que as variáveis tenham os valores indicados abaixo:

a = 5.5 b = 1.5 k = -3

1. a < 10.0
2. a + b >= 6.5
3. k ~= 0
4. b – k > a
5. ~(a = = 3*b)
6. – k < = k + 6
7. a < 10 & a > 5
8. abs (k) > 3 | k < b - a

2) (**estruturas de decisão**)Nos problemas 1 a 7, dê os comandos MATLAB necessários para executar os passos
indicados. Suponha que as variáveis são escalares.

a Se *time* é maior que 50, então incremente-a por 1.

b. Quando a raiz quadrada de *poly* for menor que 0,001, imprima o valor de *poly*;

c. Se a diferença entre *volt_1* e *volt_2* for maior que 2, imprimir os valores de *volt_1* e *volt_2*;

d. Se o valor de *den* for menor que 0, 003; atribua zero a *result*; caso contrário, atribua a *result num* dividido por dez;

e. Se o logaritmo natural de *x* for maior ou igual a 10, atribua zero a *time* e incremente-o por
count;

f. Se *dist* for maior que 50 e *time* for maior que 10, incremente *time* por 2; caso contrário, incremente *time* por 5.

g. Se *dist* for maior ou igual a 100, incremente *time* por 10. Se *dist* estiver entre 50 e 100,

incremente *time* por 1. Caso contrário incremente *time* por 0,5.