SLCO4A - Aula Prática 3

Prof. Willian R. Bispo M. Nunes

Sumário

Funções (M files)

São arquivos de extensão .m e devem ser salvas no diretório atual (*Current folder*) do MATLAB. A diferença entre funções e *scripts* é que funções aceitam um ou mais argumentos de entrada e retornam um ou mais argumentos de saída.

A primeira linha de uma função .m deve ser da forma `function [y1, y2,...yn]=name(x1,x2,...,xn).`

Criando uma função m.file

- Acesse *New>>Function* e verá o seguinte *script*

```
function [outputArg1,outputArg2] = untitled2(inputArg1,inputArg2)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
outputArg1 = inputArg1;
outputArg2 = inputArg2;
end
```

```
[soma,produto]=oper(2,10)
```

- Faça as alterações no *template* de forma a criar uma função que desenvolva o cálculo de soma e produto de duas variáveis. O código deve ficar da seguinte forma:

```
function [sm,pr] = oper(a,b)
%OPER Função que realiza soma e multiplicação de duas matrizes
sm = a+b;
pr = a*b;
end
```

Testando a função

- Sem variável para armazenar os resultados

```
C = [1 2; 1 2];
D = [2 3; 2 3];
oper(C,D)
```

```
C = [1 2; 1 2];
D = [2 3; 2 3];
oper(C,D)
```

- Armazenando os resultados

```
E = [1 1; 2 3];
F = [1 2; 3 4];
[AD, MUL] = oper(E,F)
```

```
E = [1 1; 2 3];
F = [1 2; 3 4];
[AD, MUL] = oper(E,F)
```

- Descrição detalhada de ajuda para função

```
help oper
```

```
help oper
```

- Abrir o script de qualquer função

```
open oper
```

```
%open oper
```

Comandos de entrada e saída

- O comando `input` é empregado para obter dados de entrada do usuário. O comando mostra no *command window* o texto `"string"` e então espera-se uma entrada de dados do usuário proveniente do teclado. A entrada de dados do usuário é atribuída à variável `a`.

```
a=input('string')
```

```
a=input('Digite um número: ')
```

- A entrada do usúario é considerada uma string.

```
b=input('string','s')
```

```
b=input('Digite um número: ','s')
```

- O comando `disp` mostra o resultado da variável `a`.

```
disp(a)
```

```
disp(a)
```

- O comando disp mostra uma `string`.

```
disp('string')
```

```
disp('teste')
```

- Para concatenar um texto com valores numéricos de uma variável deve-se realizar a conversão do formato numérico para string por meio da função `num2str`.

```
disp(['O número é' num2str(a)])
```

```
disp(['O número é: ' num2str(a)])
```

- Um comando equivalente é o `fprintf`

```
fprintf('%s %f', 'O número é', a)
```

```
fprintf('%s %f', 'O número é: ', a)
```

Operadores relacional e lógico

Uma expressão lógica é uma expressão cujo resultado é verdadeiro ou falso.

- Os principais operadores relacionais são os seguintes:

```
==      eq()
>=      gt()
<=      lt()
~=      ne()
>=      ge()
<=      le()
```

```
eq(10,5)
gt(10,5)
lt(10,5)
ne(10,5)
ge(10,5)
le(10,5)
```

- Operadores lógicos de matrizes. Alguns comandos são os seguintes:

```
isequal(a,b) %retorna 1 se as matrizes a e b são iguais

isempty(a) %retorna 1 se a matriz a não contém elementos
```

```
isnan(a) %retorna 1 se o argumento de a é NaN

isinf(a) %retorna 1 se o argumento de a é infinito
```

- Operadores lógicos

```
&        %AND
|        %OR
~        %NOT
xor()    %XOR
all(a<b) %retorna 1 se a condição é satisfeita para todos os elementos do vetor
all(a)   %retorna 1 se todos os elementos de a são não-nulos
any(a<b) %retorna 1 se a condição é satisfeita para qualquer elemento do vetor
```

Operadores de controle

`if-ifelse-else, switch-case-otherwise, for, while, break, continue, return`

```
if condition1
  statements
elseif condition2
  statements
else
  statements
end
```

```
switch(variable)
case{value1}
   statements
case{value2}
   statements
otherwise
   statements
end
```

Exercício: Faça uma função no qual o usuário deve digitar um número entre 1 e 10. Dependendo do número digitado pelo usuário o programa deve retornar uma resposta de *sucesso* para um número maior ou igual a 5 e *falha* para número menor que 5.

a) Crie uma função **local** com o nome `if1` utilizando condicionais do tipo `if-elseif-else`

```
function if1
a=input('Digite um número: ')
if (a>0 && a<5)
    disp('Falha')
elseif (a>=5 && a<=10)
    disp('Sucesso')
else
```

```
        disp('Escolha novamente')
    end
    end
```

```
function if1
a=input('Digite um número: ')
if (a>0 && a<5)
    disp('Falha')
elseif (a>=5 && a<=10)
    disp('Sucesso')
else
    disp('Escolha novamente')
end
end
```

b) Crie uma função **.mlx** com o nome `swi` utizando em sua lógica a estrutura `switch`

```

```

```
    for counter = vector
    statements
    end
```

```

```

```
    while (condition)
    statements
    end
```

```

```

Variáveis simbólicas

Uma variável simbólicas é definida pelos comandos `sym` e `syms`.

```
    x=sym('x')
    syms x y z w

    a=limit(sin(x)/x,0)

    A=[x 2*y;z-w z+w]
    det(A)

    F=x^2+5*x+3
```

```

```

Diferenciação

```
    diff(f)
    diff(f,n)
    diff(f,variable,n)
```

Exemplo: Compute a derivada parcial da função $f(x,y) = e^{-x}\cos(y)$.

Integração

```
int(f,variable)
int(f,variable, lower-limit, upper-limit)
```

Somatório de uma função

```
symsum(expression, index, lower-limit, upper-limit)
```

Exemplo: compute $\sum_{k=0}^{\infty} w^k$

Resolução de equações algébricas

O comando `solve` computa as raízes da expressão simbólica `f`.

Resolução equações diferenciais

```
dsolve('f','initial-conditions','independ-variable')
% A primeira derivada é denotada por D
% A segunda derivada é denotada por D2
```

Exemplo: Compute a solução da equação diferencial $y'(t) + y(t) + 1 = 0, y(0) = 0$

Comando de substituição

```
subs(f,old,new)
```

Polinômios

Há diversas formas de representar polinômios de representar um polinômio no MATLAB. A forma mais conveniente é representar em um vetor linha cujos elementos representam os coeficiente do polinômio em ordem descendente.

Por exemplo, o polinômio $z(x) = 2x^3 + 5x + 6$ é representado como `z=[2 0 5 6]`.

As seguintes operações podem ser realizadas entre polinômios $z(x)$ e $p(x)$:

- Soma/subtração: desde que possuam o mesmo tamanho.

- Multiplicação: o produto $z(x) \cdot p(x)$ é realizado com o comando `conv(z,p)`
- Divisão: o comando [q,r]=`deconv(z,p)` retorna o quociente q e o resto `r` da divisão $\frac{z(x)}{p(x)} = q(x) + \frac{r(x)}{p(x)}$

Outras operações que podem ser realizadas com polinômios:

- Raízes de um polinômio: por meio do comando r=`roots(p)`
- Se as raízes são conhecidas pode-se determinar o polinômio p=`poly(r)`
- A derivada do polinômio é computada por meio do comando h=`polyder(p)`
- Para avaliar o resultado de um polinômio $p(x_0)$ para um valor específico x_0 utiliza-se o comando `polyval(p,x0)`

```
ans = 2
```