

Modulo 7 – Princípios básicos de Scilab (2)

Prof: Rafael Lima

Editor

- Scilab pode ser usado para edição
 - Através do *Console* em um modo interativo
 - Através do editor
- *Console* – Os resultados já são obtidos a medida que são requisitados
- Editor – Permite editar uma seqüência de comandos para posterior processamento. Pode ser aberto através do comando `editor`

Editor

- Existem dois tipos de extensões para os arquivos editados pelo Scilab:
 - .sce usado para criação de *scripts*
 - .sci usado para criação de funções
- *Scripts* - seqüência de comandos que o usuário digitaria em uma sessão interativa no *prompt* do Scilab
- Funções – entidade que recebe parâmetros de entrada, realiza um processamentos sobre estes, e retorna um conjunto de variáveis de saída

Scripts

- *Scripts*
 - Todas as variáveis definidas no arquivo de comandos permanecem validas no ambiente Scilab
 - Não há definição clara das entradas e saídas do *script*
- *Scripts* após salvos podem ser evocados através
 - Comando `execute`
 - Menu *Execute*

Scripts

- Comando `exec`
 - Notação: `exec("nome do arquivo")`
 - `-->exec("exemplo_script.sce")`
- Menu *Execute*
 - *Load into scilab* – Executa os comandos no arquivo como se fosse copiado/colado no *prompt* do Scilab. Comandos sem ; serão ecoados na tela.
 - *Evaluate Selection* – Permite executar os comandos que foram selecionados
 - *Execute file into scilab* – Executa um arquivo no *console* como se fosse evocado o comando `exec`

Funções

- Assim como linguagens de programação, Scilab permite a criação de funções para permitir reusabilidade de código
- A chamada de uma função no *console* do Scilab é feita da seguinte forma:

`[s1, ..., sN] = nome_da_funcao(e1, ..., eM)`

- Já vimos exemplo de funções

`--> [linha coluna] = size(A)`

Funções

- Definição de uma função:

```
function [s1, ..., sN] = nome_função(e1, ..., eM)
```

```
    instrução 1
```

```
    instrução 2
```

```
    ...
```

```
    instrução P
```

```
endfunction
```

Funções

- Exemplo de definição de uma função:

```
function [P, d] =  
    exemplo_funcao(A, B)  
    P = A*B;  
    d = det(P);  
endfunction
```


Funções

- Para que uma função seja carregada no ambiente Scilab temos três opções
 - Digitar a função diretamente no *console*
 - Através do menu *Execute->Load into scilab*
 - Através do menu *Execute->Execute file into scilab* ou pelo comando `execute`
- Uma vez que a função foi carregada ela pode ser utilizada normalmente

Funções

- Toda função deve definir o valor do seu parâmetro de saída

```
function y = myfunction ( x )
```

```
z = 2 * x
```

```
Endfunction
```

```
-->y = myfunction(2)
```

```
!--error 4
```

```
Variável indefinida: y
```

```
at line          4 of function myfunction
```

```
called by :
```

```
y = myfunction(2)
```

Funções

- As variáveis definidas dentro da função são variáveis locais e não estão disponíveis após sua execução

```
function y = myfunction ( x )
```

```
    m = 2; y = m*x;
```

```
endfunction
```

```
-->y = myfunction(5)
```

```
y =  
    10.
```

```
-->m
```

```
!--error 4
```

```
Variável indefinida: my = myfunction(5)
```

Programação

- Scilab proporciona a maioria das estruturas de linguagens de programação convencionais
 - `if-then-else`
 - `select-case`
 - `for`
 - `while`
 - `break` e `continue`

Programação

- `if` permite executar um comando se a condição for verdadeira
- Exemplos de uso:

```
if ( %t ) then  
    disp ( " Hello ! " )  
end
```

Programação

- Exemplos de uso:

```
if ( %f ) then
    disp (" Hello !")
else
    disp (" Goodbye !")
end
```

Programação

- Exemplos de uso:

```
i = 2
if ( i == 1 ) then
    disp ( " Hello !" )
elseif ( i == 2 ) then
    disp ( " Goodbye !" )
elseif ( i == 3 ) then
    disp ( " Tchao !" )
else
    disp ( "Au Revoir !" )
end
```

Programação

- `select-case` permite combinar vários ramos condicionais de uma forma bem simples
- Exemplo de uso:

```
i = 2
select i
case 1
    disp ("One")
case 2
    disp ("Two")
else
    disp (" Other ")
end
```


Programação

- `for` permite realizar *loops*, ou seja, realizar uma dada ação várias vezes
- Exemplo de uso:

```
for i = 1 : 5  
    disp (i)  
end
```

Programação

- Exemplo de uso:

```
v = [1.5 exp(1) %pi];  
for x = v  
    disp(x)  
end
```

Programação

- `while` permite executar ações enquanto uma certa condição é verdadeira
- Exemplo de uso:

```
x = 1;
```

```
while x < 14
```

```
    x = x * 2;
```

```
end
```

```
disp(x)
```

Programação

- `break` permite interromper um *loop*
- Exemplo de uso:

```
s = 0; i = 1
while ( %t )
    if ( i > 10 ) then
        break
    end
    s = s + i; i = i + 1
end
disp(s)
```

Programação

- `continue` permite que saltemos para nova iteração de forma que o restante dos comandos dentro do *loop* não são executados
- Exemplo de uso:

```
s = 0; i = 1
while ( i <= 10 )
    if ( modulo ( i , 2 ) == 0 ) then
        i = i + 1; continue
    else
        s = s + i; i = i + 1
    end
end
```

Polinômios

- Scilab também permite trabalharmos com polinômios
- Polinômios podem ser criados através do comando `poly`
- Podemos criar um polinômio através de seus coeficientes ou suas raízes

Polinômios

- Através de seus coeficientes:

```
-->p = poly([1 2 3 4], 'x', 'coeff')
```

Vetor de
coeficientes

Variável
simbólica

Tipo de entrada:
coefficients

p =

$$1 + 2x + 3x^2 + 4x^3$$

Polinômios

- Através de suas raízes:

```
-->q = poly([2 3], 'z', 'roots')
```

Vetor de raízes

Variável
simbólica

Tipo de entrada:
roots

$q =$

$$6 - 5z + z^2$$

Polinômios

- Podemos realizar diversas operações com polinômios
- Para realizar operações entre polinômios eles devem ter a mesma variável simbólica
- Exemplo considerando p e q definidos nos slides anteriores

```
-->p+q
```

```
!--error 144
```

Operação indefinida para os dados operandos.
Verifique ou defina a função `%p_a_p` para
overloading.

Polinômios

- Considere os seguintes polinômios:

```
-->p1 = poly([1 2 3 4], 'x', 'coeff')
```

p1 =

$$1 + 2x + 3x^2 + 4x^3$$

```
-->p2 = poly([2 3], 'x', 'roots')
```

p2 =

$$6 - 5x + x^2$$

Polinômios

- Soma:

-->p1+p2

ans =

$$7 - 3x + 4x^2 + 4x^3$$

- Subtração:

-->p1-p2

ans =

$$-5 + 7x + 2x^2 + 4x^3$$

Polinômios

- **Multiplicação:**

```
-->p1*p2
```

```
ans =
```

$$6 + 7x + 9x^2 + 11x^3 - 17x^4 + 4x^5$$

- **Divisão:**

```
-->[r q] = pdiv(p1,p2)
```

```
q =
```

$$23 + 4x$$

```
r =
```

$$-137 + 93x$$

p1 – Numerador

p2 – Denominador

q – Quociente

r – Resto da divisão

Polinômios

- Outros comandos:
 - Cálculo das raízes de um polinômio: `roots`
 - Calcular o valor de um polinômio para um dado valor da variável simbólica: `horner`

- Comando `roots`:

```
-->p = poly([-6 11 -6 1], 'x', 'coeff')
```

```
p =
```

$$-6 + 11x - 6x^2 + x^3$$

```
-->roots(p)
```

```
ans =
```

```
1.      2.      3.
```

Polinômios

- Comando horner:

```
-->p = poly([-6 11 -6 1], 'x', 'coeff')
```

```
p =
```

$$-6 + 11x - 6x^2 + x^3$$

```
-->horner(p,1)
```

```
ans =
```

```
0.
```

Manipulando arquivos e diretórios

- Comando `pwd`: mostra o diretório atual em que estamos trabalhando
- Comando `ls`: lista os arquivos existentes no diretório atual
- Comando `chdir`: muda o diretório que estamos trabalhando

`% Muda para diretorio \Rafael`

`chdir('C:\Documents and Settings\Rafael\')`

Manipulando arquivos e diretórios

```
% Subir um nivel na hierarquia de
  pastas
-->chdir('..');
-->ls
ans =
!Rafael          !
!NetworkService !
!LocalService   !
!Default User   !
!Cibelly        !
!All Users      !
```


Manipulando arquivos e diretórios

- Comando `save`: salva variáveis definidas no Scilab em um arquivo

```
-->a = ones(2,5);  
-->b = 'engenharia';  
-->save('variaveis.dat',a,b);  
-->ls  
ans =  
!variaveis.dat      !  
!teste.sce          !  
!scilab              !  
!exemplo_script.sce !  
!exemplo_funcao.sci !
```

Manipulando arquivos e diretórios

- Comando `load`: restaura no ambiente de trabalho do Scilab variáveis salvas em uma arquivo

```
-->clear
-->a
!--error 4  Variável indefinida: a
-->b
!--error 4  Variável indefinida: b
-->load('variaveis.dat');
-->a
a  =
    1.    1.    1.    1.    1.
    1.    1.    1.    1.    1.
-->b
b  =
engenharia
```