Modulo 7 – Princípios básicos de Scilab

Prof: Rafael Lima

Introdução

- Ambiente voltado para resolução de problemas numéricos
- Criado em 1990 por um grupo de pesquisadores do INRIA (*Institut de Recherche en Informatique et en Automatique*) e ENPC (*Ecole Nationale des Ponts et Chaussées*)
- Gratuito e código aberto
- http://www.scilab.org

Introdução

- Aplicações:
  - Álgebra linear
  - Polinômios e funções racionais
  - Interpolação
  - Otimização
  - Equações diferenciais
  - Estatística

Elementos básicos

- Para acessar ajuda através do console basta digitar:

```
--> help
```

- Para obter ajuda de uma função especifica basta digitar:

```
--> help sin
```

Elementos básicos

- Variáveis não precisam ser declaradas:

```
-->x = 1
 x  =
     1.
```

- Os valores das variáveis podem ser redefinidos:

```
-->x = 2
 x  =
     2.
 -->x = 5
 x  =
     5.
```

Elementos básicos

- Para limpar a memória das variáveis não protegidas use o comando `clear`
- Para limpar a tela do *prompt* do scilab use `clc`

Elementos básicos

- Operadores básicos:
  - + Adição
  - – Subtração
  - * Multiplicação
  - / Divisão a direita $x/y = xy^{-1}$
  - \ Divisão a esquerda $x\backslash y = x^{-1}y$
  - ^ Potencia x^y
  - ** Potencia x^y
  - ' Conjugado transposto

Elementos básicos

- ~~Podemos nomear variáveis com até 24 caracteres~~
- Scilab é *case sensitive*, ou seja, scilab diferencia letras maiúsculas de minúsculas. Ex.: `potencia` e `Potencia` podem ser definidas como duas variáveis diferentes

Elementos básicos

- Comentários em scilab são iniciados por / /
- Os caracteres . . . permitem que continuemos uma expressão mesmo após o *enter*

```
-->//Este é um comentario
-->x = 1 + ...
-->2 + ...
-->3
 x  =

      6.
```

Elementos básicos

- Scilab possui já implementadas algumas funções matemáticas

acos	acosd	acosh	acoshm	acosm	acot	acotd	acoth
acsc	acscd	acsch	asec	asecd	asech	asin	asind
asinh	asinhm	asinm	atan	atand	atanh	atanhm	atanm
cos	cosd	cosh	coshm	cosm	cotd	cotg	coth
cothm	csc	cscd	csch	sec	secd	sech	sin
sinc	sind	sinh	sinhm	sinm	tan	tand	tanh
tanhm	tanm						

Elementos básicos

- Essas funções aceitam como entrada vetores para evitar uso de *loops* ou estruturas de repetição

exp	expm	log	log10	log1p	log2	logm	max
maxi	min	mini	modulo	pmodulo	sign	signm	sqrt
sqrtm							

```
-->x = cos(2);

-->y = sin(2);

-->x^2 + y^2
 ans  =

    1.
```

Elementos básicos

- Scilab possui variáveis pré-definidas que são iniciadas por `%`:

`%i` Numero imaginário *i*

`%e` Constante de Euller *e*

`%pi` Constante matemática π

Exemplo:

```
-->cos(%pi)
 ans  =
   - 1.
```

Elementos básicos

- Variáveis em scilab podem assumir também valores booleanos, ou seja, lógicos
- Verdadeiro é representado pelo símbolo `%t` ou `%T`
- Falso é representado pelo símbolo `%f` ou `%F`
- Valores booleanos são resultado de operações logicas

Elementos básicos

- Operações lógicas:

`a&b` Logica **e**

`a|b` Logica **ou**

`~a` Logica **não**

`a==b` Verdadeiro se a e b são **iguais**

`a~=b` ou `a<> b` Verdadeiro se a e b são **diferentes**

`a<b` Verdadeiro se a é **menor** que b

`a>b` Verdadeiro se a é **maior** que b

`a<=b` Verdadeiro se a é **menor** ou **igual** a b

`a>=b` Verdadeiro se a é **maior** ou **igual** a b

Elementos básicos

- Exemplo:

```
-->a = %t
 a  =
   T
-->b = (0 == 1)
 b  =
   F
-->a & b
ans  =
   F
```

Elementos básicos

- Scilab também trabalha com números complexos
- Existem funções especificas para tratar números complexos

`real` Parte real de um numero

`imag` Parte imaginaria de um numero

`imult` Multiplicação por i

`isreal` Verdadeiro se a variável não contem parte imaginaria

Elementos básicos

- Exemplos de operações:

```
-->x = 1 + %i;
-->real(x)
ans  =
    1.
-->x'
ans  =
     1. - i
```

Elementos básicos

- Scilab permite que *strings* sejam armazenadas em variáveis
- *Strings* devem estar entre ‘ ‘
- Operador + permite concatenar *strings*

```
-->nome = 'Rafael'
 nome  =
 Rafael
-->sobrenome = 'Lima'
 sobrenome  =
 Lima
-->nome + ' ' + sobrenome
 ans  =
 Rafael Lima
```

Matrizes

- Scilab é fortemente orientado a matrizes
- Matrizes são definidas por
  - Numero de linhas
  - Numero de colunas
  - Tipo de dados
- Vetores são casos particulares de matrizes.
- Vetor linha tem dimensão 1 por n e vetores colunas tem dimensão n por 1 em que n é um inteiro qualquer

Matrizes

- Valores escalares são matrizes 1 por 1
- Uso de matrizes ao invés de loops torna o processamento mais rápido com scilab
- Definindo matrizes
  - Use `[` e `]` para delimitar os valores da matriz
  - Use `,` ou espaço em branco para separar os diferentes valores de diferentes colunas
  - Use `;` para separar os valores de diferentes linhas

Matrizes

- Exemplo:

```
-->A = [1 2 7; 5 2 9; 6 1 9]
 A  =
     1.    2.    7.
     5.    2.    9.
     6.    1.    9.
```

Matrizes

- Outra forma alternativa de definir matrizes:

```
-->A = [ 2 7 9
-->9 5 1
-->0 6 4]
 A  =

    2.    7.    9.
    9.    5.    1.
    0.    6.    4.
```

Matrizes

- Definição de vetores através do símbolo **:**
- Valor inicial : Incremento : Valor final

```
-->a = 1:2:11
 a  =
     1.    3.    5.    7.    9.    11.
-->a = 1:6
 a  =
     1.    2.    3.    4.    5.    6.
-->a = 10:-1:6
 a  =
    10.    9.    8.    7.    6.
```

Matrizes

- Funções especiais para criar matrizes:

  `ones` Cria matrizes com valores 1

  `zeros` cria matrizes com valores 0

  `eye` Cria matrizes unitárias

- Exemplo:

```
-->ones(1,5)
 ans  =
    1.    1.    1.    1.    1.
```

Matrizes

- Matriz vazia é representada por `[]`
- Essa representação permite liberar o conteúdo de variáveis na memória
- Exemplo:

```
-->A = ones(100,100);
-->A = [];
-->A
A  =
        []
```

Matrizes

- Funções que estão relacionadas com a dimensão de matrizes: `size`, `length`, `matrix`, `resize_matrix` e `diag`
- `size` Retorna o numero de linhas e colunas de uma matriz
- `length` Retorna o numero de elementos de uma matriz
- `matrix` Reformata uma matriz

Matrizes

- `resize_matrix` Cria uma nova matriz com tamanho diferente
- `diag` Retorna a diagonal principal de uma matriz

```
-->a = [1 2 3 1 2 3];
-->matrix(a, 2, 3)
 ans  =
   1.   3.   2.
   2.   1.   3.
 -->b = resize_matrix(ans,2,5)
 b  =
    1.   3.   2.   0.   0.
   2.   1.   3.   0.   0.
```

Matrizes

- Acesso de valores de uma matriz `A`, em que `i` e `j` são escalares e `a` e `b` são vetores:
  - `A(i,j)` Valor na linha i e coluna j
  - `A(i,:)` Todos valores na linha i
  - `A(:,j)` Todos valores na coluna j
  - `A(:,:)` ou A Matriz toda
  - `A(a,b)` Elementos compreendidos nas linhas e colunas cujos índices são os elementos do vetor a e b respectivamente

Matrizes

- Exemplo:

```
-->A = [1 2 7; 5 2 9; 6 1 9];
-->A(2,3)
ans  =
    9.
-->A(2,:)
ans  =
    5.    2.    9.
-->A([1 2],[2 3])
ans  =
    2.    7.
    2.    9.
```

Matrizes

- O operador `$` permite referenciarmos matrizes do fim ao invés do inicio

```
-->A = [1 2 7; 5 2 9; 6 1 9];
-->A($,2)
ans  =
    1.
-->A($,$)
ans  =
    9.
```

Matrizes

- Operações com matrizes devem obedecer as regras matemáticas já conhecidas

```
-->A = [1 2 7; 5 2 9];
-->B = [2 6;7 2;2 2];
-->A*B
ans  =
    30.    24.
    42.    52.
-->A*B'
    !--error 10
Multiplicação incoerente.
```

Matrizes

- Se um `.` é colocado a frente dos operadores convencionais temos operações elemento por elemento

```
-->A = [1 2 7; 5 2 9];
-->A.*A
ans  =
    1.     4.     49.
   25.     4.     81.
```

Matrizes

- Algumas operações de álgebra linear:
  - `det` Determinante de uma matriz
  - `inv` Inversa de uma matriz

```
-->A = [1 2 7; 5 2 9; 6 1 9];
-->det(A)
ans  =
   - 22.
-->inv(A)
ans  =
  - 0.4090909    0.5  - 0.1818182
  - 0.4090909    1.5  - 1.1818182
    0.3181818  - 0.5    0.3636364
```