INTRODUÇÃO AO SCILAB

➔ Introdução

Estas notas, de caráter introdutório e numa versão preliminar, tratam da utilização do software Scilab na disciplina Cálculo Numérico – COM400, ministrada por professores do Departamento de Computação/ICEB para vários cursos de graduação oferecidos pela UFOP.

O Scilab (http://www-rocq.inria.fr/scilab) é um ambiente voltado para o desenvolvimento de software para a resolução de problemas numéricos. É desenvolvido e mantido por um grupo de pesquisadores do INRIA (Institut de Recherche en Informatique et en Automatique), um instituto francês de pesquisa em informática e automação.

Possui um ambiente de programação numérica bastante flexível. Suas principais características são:

1. É um software de distribuição gratuita, com código fonte disponível;
2. É um ambiente poderoso para geração de gráficos;
3. Implementa diversas funções para manipulação de matrizes. As operações de concatenação, extração de elementos, transposição, adição e multiplicação de matrizes são facilmente realizadas;
4. Permite trabalhar com polinômios, funções de transferência, sistemas lineares e grafos;
5. Apresenta facilidade para a definição de funções, que podem ser passadas para outras funções como argumento de entrada ou de saída;
6. Permite interface com rotinas escritas em linguagem FORTRAN ou C;
7. Suporta o desenvolvimento de conjunto de funções para aplicações específicas (*toolboxes*).

Estas notas tratam da utilização do Scilab na sua versão 2.6. Sua tela de abertura é apresentada na figura 1.

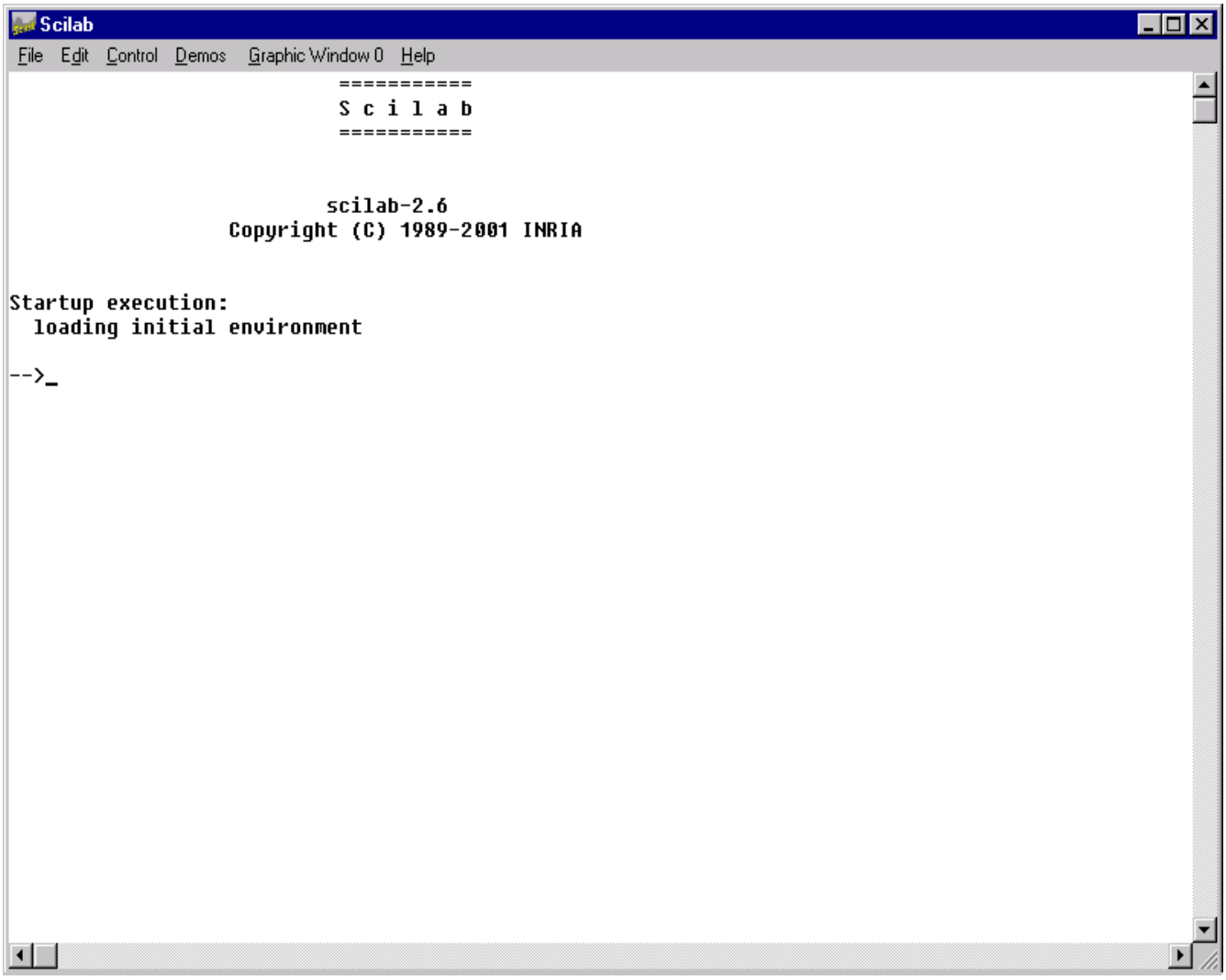

Figura 1: Tela inicial do Scilab 2.6

Observe-se que o *prompt* do Scilab é representado por uma seta, -->. É chamado de *prompt* de primeiro nível.

➔ Convenções básicas

1. Para EXECUTAR um comando digitado basta pressionar a tecla ***enter***. A execução de um comando apresenta o resultado da sua avaliação;

2. Um único comando em várias linhas é possível com a utilização de **...** ao final do comando;

3. Um ponto e vírgula no final do comando suprime a apresentação do resultado, mas, não inibe o seu cálculo internamente;

4. Letras maiúsculas e minúsculas são distintas dentro do Scilab;

5. Argumentos de funções devem vir entre parênteses;

6. Comentários podem ser inseridos em qualquer ponto utilizando-se //.

Exemplo

```
// este é um comentário
```

➔ Definição de variáveis

O Scilab sempre interpreta uma letra como sendo uma variável. Para atribuir um valor a uma varável "x" faz-se:

x = valor

➔ Operadores aritméticos

O Scilab indica as operações aritméticas por símbolos usuais como:

x + y + z ⇒ "+" para adição;

x - t ⇒ "-" para subtração;

x*z ⇒ "*" para multiplicação;

x/y ⇒ "/" para divisão;

x^y ⇒ "^" para potenciação;

A ordem de precedências, na realização dos cálculos, é a usual, mas parênteses podem ser empregados para indicar a ordem desejada, sempre que houver necessidade.

➔ Funções pré-definidas

O Scilab é carregado com algumas funções pré-definidas, chamadas de primitivas, a seguir alguns exemplos.

↳ **sqrt(x)** → raiz quadrada de x

```
-->sqrt(16)
 ans  =

   4.
```

```
-->sqrt(-2)
 ans  =

   1.4142136i
```

↳ **log(x)** → logaritmo neperiano de x

```
 -->log(%e)
 ans  =

   1
```

↳ **log10(x)** → logaritmo de x na base 10

```
-->log10(100)
 ans  =

   2.
```

↳ **exp(x)** → **e** elevado a **x**

```
-->exp(2)
 ans  =

   7.3890561
```

↳ **sin(x)** → **seno** de **x**

↳ **cos(x)** → **cosseno** de **x**

↳ **tan(x)** → **tangente** de **x**

↳ **cotg(x)** → **cotangente** de **x**

➔ **<u>Como obter ajuda</u>**

Em caso de dúvida, há várias formas de se obter ajuda.

(i) Utilizando a opção **Help** na barra de ferramentas;

(ii) Digitando **help** e o nome da função.

CÁLCULO NUMÉRICO UTILIZANDO O SCILAB

➔ Pacotes no Scilab

Se por um lado o Scilab pode ser considerado um *software* bastante profissional, por outro não tem qualquer preocupação com o aspecto didático, por este motivo o aprendizado da sua utilização não é tão simples, principalmente no que se refere à programação. Além destes inconvenientes, há também, o fato de que não disponibiliza funções que utilizam alguns dos métodos numéricos tratados na disciplina Cálculo Numérico.

Com o objetivo de suprir estas deficiências e de facilitar o uso do Scilab como ferramenta de auxílio na aprendizagem de Cálculo Numérico, foram implementadas algumas funções, utilizando a linguagem de programação do *software*, sobre os seguintes assuntos:

- Resolução de sistemas de equações lineares simultâneas;
- Interpolação polinomial;
- Resolução de equações algébricas e transcendentes.

Este texto tem a finalidade de ser um manual de utilização das funções implementadas. Os códigos fonte estão disponíveis, podendo ser livremente copiados e modificados, desde que seja feita referência ao autor; que receberá de bom grado qualquer sugestão que possibilite aperfeiçoar as rotinas implementadas e, de um modo geral, este trabalho.

Todas as funções implementadas estão armazenadas na forma de pacotes. Pacotes são arquivos de extensão **sci**, em código ASCII, de funções que não são intrinsecamente construídas no núcleo do Scilab.

Todos os pacotes do Scilab estão armazenados em pastas em:

C:\Arquivos de programas\Scilab-2.6\macros.

Para disponibilizar ao usuário as funções contidas em um pacote, basta acessar o comando **getf** através da opção **File** do menu horizontal.

Outra forma é a utilização do comando **getf**, que tem a seguinte sintaxe

getf('C:\Arquivos de programas\Scilab-2.6\macros**pasta do pacote****nome do pacote**');

Por exemplo, para carregar o pacote **gauss.sci**, disponibilizando, assim, a função nele definida, é usada a sintaxe:

getf('C:\Arquivos de programas\Scilab-2.6\macros**com400\gauss.sci**');

É importante observar que o Scilab não exibe qualquer mensagem confirmando que o pacote foi carregado com sucesso; por outro lado, exibe mensagem quando há algum problema.. Neste texto está sendo tomado como referência que os pacotes estão armazenados na pasta **com400** em **macros**.

1. Pacotes para a resolução de sistemas de equações lineares simultâneas

1.1 – Método de eliminação de Gauss

Este pacote está contido no arquivo **gauss.sci**. Possibilita a resolução de um sistema de equações lineares simultâneas utilizando o método da eliminação de Gauss, com pivotação parcial. Disponibiliza a **função gauss(Ab)**. Para utilizá-la basta fornecer a matriz aumentada do sistema de equações, e ter-se-á como retorno o vetor x, solução do sistema Ax = b.

Exemplo

Seja resolver o sistema de equações mx = b, a seguir, utilizando o método de eliminação de Gauss com pivotação parcial.

$$m = \begin{vmatrix} 1 & 5 & 1 \\ 10 & 2 & 1 \\ 2 & 3 & 10 \end{vmatrix} \quad e \quad b = \begin{vmatrix} -8 \\ 7 \\ 6 \end{vmatrix}$$

→ Entrando com a matriz aumentada do sistema de equações

```
-->ab=[1,5,1,-8;10,2,1,7;2,3,10,6]
 ab  =

!  1.    5.    1.   - 8. !
!  10.   2.    1.     7. !
!  2.    3.    10.    6. !
```

Observe-se que os elementos de uma linha da matriz são separados por vírgula e que as linhas por ponto e vírgula.

→ Carregando a função gauss.sci

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\gauss.sci');
```

→ Resolvendo o sistema de equações usando a função gauss.sci

-->gauss(ab)

Matriz aumentada do sistema de equações

```
!  1.    5.    1.   - 8. !
!  10.   2.    1.     7. !
!  2.    3.    10.    6. !
```

Troca de posição entre as linhas 1 e 2

```
!  10.   2.    1.     7. !
!  1.    5.    1.   - 8. !
!  2.    3.    10.    6. !
```

Passo 1 do processo de eliminação
O pivô neste passo é 10.000000

```
!  10.   2.     1.     7.  !
!  0.    4.8     .9  - 8.7 !
!  0.    2.6    9.8    4.6 !
```

Passo 2 do processo de eliminação
O pivô neste passo é 4.800000

```
!  10.   2.     1.        7.     !
!  0.    4.8     .9     - 8.7    !
!  0.    0.     9.3125    9.3125 !
```

O vetor solução é

```
!  1.  - 2.    1. !
```

1.2 – Método da decomposição LU

Este pacote está contido no arquivo **ludec.sci**. Possibilita a resolução de um sistema de equações lineares simultâneas utilizando o método da decomposição LU, com pivotação parcial. Disponibiliza a **função ludec(A,b)**. Para utilizá-la basta fornecer a matriz dos coeficientes do sistema de equações e vetor dos termos independentes; ter-se-á como retorno o vetor x, solução do sistema Ax = b.

Exemplo

Será utilizado o mesmo sistema de equações do item 1.1.

→ Entrando com a matriz dos coeficientes e vetor dos termos independentes

-->a=[1,5,1;10,2,1;2,3,10]

```
 a  =
```

```
!   1.     5.    1.  !
!   10.    2.    1.  !
!   2.     3.    10. !

-->b=[-8,7,6]
b  =

! - 8.    7.    6. !
```

→ Carregando a função ludec.sci

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\ludec.sci');
```

→ Resolvendo o sistema de equações usando a função ludec.sci

```
-->ludec(a,b)

 Matriz dos coeficientes

!   1.     5.    1.  !
!   10.    2.    1.  !
!   2.     3.    10. !
```

Troca de posição entre as linhas 1 e 2

```
!   10.    2.    1.  !
!   1.     5.    1.  !
!   2.     3.    10. !
```

Passo 1 do processo de eliminação
O pivô neste passo é 10.000000

```
!   10.    2.     1.  !
!    .1    4.8     .9 !
!    .2    2.6    9.8 !
```

Passo 2 do processo de eliminação
O pivô neste passo é 4.800000

```
!   10.    2.          1.     !
!    .1    4.8          .9    !
!    .2     .5416667    9.3125 !
```

Termos independentes na entrada

```
! - 8.    7.    6. !
```

Vetor de pivotação

```
!   2.    1.    3. !
```
Termos independentes na saída

```
!   7.  - 8.    6. !
```

[More (y or n) ?]

Matriz L

```
!   1.     0.          0. !
!    .1    1.          0. !
!    .2     .5416667    1. !
```

Matriz U

```
!   10.    2.     1.     !
!   0.     4.8     .9    !
!   0.     0.     9.3125 !
```

O vetor y é

```
!   7.  - 8.7    9.3125 !
```

A solução do sistema é

```
!   1.  - 2.    1. !
```

1.3 – Método de Jacobi

Este pacote está contido no arquivo **jacobi.sci**. Disponibiliza a **função jacobi(A,b)**, que permite resolver um sistema de equações lineares, AX=B, usando o método iterativo de Jacobi. Para a construção da função foi utilizada uma formulação matricial do método.

Para utilizar esta função basta fornecer a matriz dos coeficientes e o vetor dos termos independentes e ter-se-á como retorno o vetor solução do sistema.

Exemplo

Será utilizado o mesmo sistema de equações do item 1.1 trocando a ordem das equações 1 e 2.

→ Entrando com a matriz dos coeficientes e vetor dos termos independentes

```
-->a=[10,2,1;1,5,1;2,3,10]
```

```
 a  =

!   10.    2.    1.  !
!   1.     5.    1.  !
!   2.     3.    10. !
-->b=[7,-8,6]
 b  =

!   7.  - 8.    6. !
```

→ Carregando a função jacobi.sci

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\jacobi.sci');
```

→ Resolvendo o sistema de equações usando a função jacobi.sci

```
-->jacobi(a,b)

Numero maximo de iteracoes:
-->10

 Precisao desejada:
-->0.05

 Entre com o vetor x0 na forma [x1 x2...xn]:
-->[0,0,0]
Iteração 1
  .7000
 -1.6000
  .6000
Maior diferença entre o vetor 1 e 0 = 1.6000

Iteração 2
  .9600
 -1.8600
  .9400
Maior diferença entre o vetor 2 e 1 =  .3400

Iteração 3
  .9780
 -1.9800
  .9660
Maior diferença entre o vetor 3 e 2 =  .1200
```

Iteração 4
 .9994
-1.9888
 .9984
Maior diferença entre o vetor 4 e 3 = .0324

Para a precisão .05000 a solução é o vetor:

! .9994 - 1.9888 .9984 !

1.4 – Método de Gauss - Seidel

Este pacote está contido no arquivo **gseidel.sci**. Disponibiliza a **função gseidel(A,b)**, que permite resolver um sistema de equações lineares, AX=B, usando o método iterativo de Gauss - Seidel. Para a construção da função foi utilizada uma formulação matricial do método.
Para utilizar esta função basta fornecer a matriz dos coeficientes e o vetor dos termos independentes e ter-se-á como retorno o vetor solução do sistema.

Exemplo

Será utilizado o mesmo sistema de equações do item 1.1 trocando a ordem das equações 1 e 2.

→ Entrando com a matriz dos coeficientes e vetor dos termos independentes

```
-->a=[10,2,1;1,5,1;2,3,10]
 a  =

!   10.    2.    1.  !
!   1.     5.    1.  !
!   2.     3.    10. !

-->b=[7,-8,6]
 b  =

!   7.  - 8.    6. !
```

→ Carregando a função gseidel.sci

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\gseidel.sci');
```

→ Resolvendo o sistema de equações usando a função gseidel.sci

```
-->gseidel(a,b)
Numero maximo de iteracoes:
-->10
Precisao desejada:
```

```
-->0.05

 Entre com o vetor x0 na forma [x1 x2...xn]:
-->[0 0 0]
Iteração 1
  .7000
 -1.7400
  .9820
Maior diferença entre o vetor 1 e 0 = 1.7400
Iteração 2
  .9498
 -1.9864
 1.0059
Maior diferença entre o vetor 2 e 1 =  .2498

Iteração 3
  .9967
 -2.0005
 1.0008
Maior diferença entre o vetor 3 e 2 =  .0469


Para a precisao  .05000 a solução é o vetor:

!    .9966772  - 2.000525    1.0008221 !
```

2. Pacotes para interpolação polinomial

2.1 – Método de Lagrange

Este pacote está contido no arquivo **lagrange.sci**. Disponibiliza a **função lagrange(x,y)**, que permite obter a equação do polinômio que interpola uma função em um conjunto de pontos dados e, ainda, determinar um ponto não tabelado. Se o valor a ser interpolado estiver fora do intervalo dado para x, é emitida uma mensagem e a execução é abortada. Para utilizar esta função basta fornecer os vetores x e y.

Exemplo

Seja y = f(x) uma função conhecida nos pontos a seguir.

i	0	1	2	3
x_i	0	1	2	4
y_i	4	11	20	44

→ Entrando com os vetores x e y

```
-->x=[0 1 2 4]
 x  =

!  0.    1.    2.    4. !

-->y=[4 11 20 44]
 y  =

!  4.    11.    20.    44.
```

→ Carregando a função lagrange.sci

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\lagrange.sci');
```

→ Realizando a interpolação utilizando a função lagrange.sci

```
-->lagrange(x,y)

 O polinômio interpolador é

                    2
   4 + 6w + w

 Deseja fazer interpolação (s ou n entre apóstrofo)?
-->'s'

 Valor a ser interpolado?
-->3
Para x = 3.00000,     y =31.00000

 Outra interpolação (s ou n entre apóstrofo)?
-->'n'

 Sessão encerrada
```

2.2 – Método das diferenças divididas

Este pacote está contido no arquivo **difdiv.sci**. Disponibiliza a **função difdiv(x,y)**, que permite obter a equação do polinômio que interpola uma função em um conjunto de pontos dados e, ainda, determinar um ponto não tabelado. Se o valor a ser interpolado estiver fora do intervalo dado para x, é emitida uma mensagem e a execução é abortada. Para utilizar esta função basta fornecer os vetores **x** e **y**..

Exemplo

Seja y = f(x) uma função conhecida nos pontos a seguir.

i	0	1	2	3
x_i	3	5	6	8
y_i	1.10	1.27	1.36	1.53

→ Entrando com os vetores x e y

```
-->x=[3 5 6 8]
 x  =

!  3.    5.    6.    8. !

-->y=[1.1 1.27 1.36 1.53]
 y  =

!   1.1    1.27    1.36    1.53 !
```

→ Carregando a função difdiv.sci

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\difdiv.sci');
```

→ Realizando a interpolação usando a função difdiv.sci

```
-->difdiv(x,y,)

Diferenças divididas de ordem zero

!   1.1    1.27    1.36    1.53 !

Diferenças divididas de ordem 1
 .08500
 .09000
 .08500

Diferenças divididas de ordem 2
 .00167
- .00167

Diferenças divididas de ordem 3
- .00067
A equação do polinômio interpolador é

                          2           3
    .93 +  .0296667p +  .011p -  .0006667p
Deseja fazer interpolação (s ou n entre apóstrofo)?
```

-->'s'

Valor a ser interpolado?
-->4
Para x = 4.00000, y = 1.18200

Outra interpolação (s ou n entre apóstrofo)?
-->'n'

Sessão encerrada

2.3 – Método das diferenças finitas ascendentes

Este pacote está contido no arquivo **difasc.sci**. Disponibiliza a **função difasc(x,y)**, que permite obter a equação do polinômio que interpola uma função em um conjunto de pontos dados e, ainda, determinar um ponto não tabelado. Se o valor a ser interpolado estiver fora do intervalo dado para x, é emitida uma mensagem e a execução é abortada. Para utilizar esta função basta fornecer os vetores **x**, com elementos igualmente espaçados, e **y**.

Exemplo

Seja y = f(x) uma função conhecida nos pontos a seguir.

i	0	1	2	3
x_i	0	1	2	3
y_i	0	5	10	3

→ Entrando com os vetores x e y

-->x=[0 1 2 3]
 x =

! 0.   1.   2.   3. !

-->y=[0 5 10 3]
 y =

! 0.   5.   10.   3. !

→ Carregando a função difasc.sci

-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\difasc.sci');

→ Realizando a interpolação usando a função difasc.sci

-->difasc(x,y)

```
 Diferenças finitas ascendentes de ordem zero

!  0.    5.    10.    3. !
Diferenças finitas ascendentes de ordem 1
5.00000
5.00000
-7.00000

Diferenças finitas ascendentes de ordem 2
0.00000
-12.00000

Diferenças finitas ascendentes de ordem 3
-12.00000

A equação do polinômio interpolador é

            2    3
   z + 6z - 2z

 Deseja fazer interpolação (s ou n entre apóstrofo)?
-->'s'

 Valor a ser interpolado?
-->1.5
Para x = 1.50000,     z=1.50000  e  y = 8.25000

 Outra interpolação (s ou n entre apóstrofo)?
-->'n'
Sessão encerrada
```

3. Pacotes para o cálculo das raízes reais de uma equação

3.1 – Equações algébricas polinomiais

3.1.1 – Limites das raízes reais

Este pacote está contido no arquivo **limites.sci**. Trata-se de uma implementação do teorema de Lagrange para delimitação das raízes reais de uma equação algébrica polinomial.

Disponibiliza a **função limites(coef)**, onde "coef" é um vetor que contém os coeficientes da equação, que deve ser passado de forma completa, ou seja, coeficientes nulos, devem ser considerados.

Exemplo

Seja determinar os limites das raízes reais da equação $y = x^3 – 6x + 2 = 0$.

➔ Entrando com o vetor dos coeficientes

```
-->c=[1 0 -6 2]
 c  =
!   1.    0.  - 6.    2. !
```

➔ Carregando a função limites(coef)

```
-->getf('C:\Program Files\Scilab-2.6\macros\com400\limites.sci');
```

➔ Determinando os limites utilizando a função limites(coef)

```
-->limites(c)
A equação dada é de grau 3
 Equação dada. Usada no cálculo do LSRP

                  3
   2 - 6R + R

k = 1    M = 6.000000

Equação auxiliar para o cálculo do LIRP

 f1(x) = f(1/x) = 0

             2      3
   1 - 6R + 2R

k = 2    M = 6.000000
Equação auxiliar para o cálculo do LIRN

 f2(x) = f(-x) = 0

                  3
  - 2 - 6R + R

k = 1    M = 6.000000
```

Equação auxiliar para o cálculo do LSRN

f3(x) = f(-1/x) = 0

$$-1 + 6R + 2R^2 + 3$$

k = 0 M = 1.000000

As raízes positivas, se existirem, estão no intervalo:

[.25 3.45]

As raízes negativas, se existirem, estão no intervalo:

[-3.45 - .56]

3.1.2 – Enumeração das raízes reais

Este pacote, contido no arquivo **enumera.sci**, é a implementação de um processo para a enumeração das raízes reais de uma equação algébrica polinomial que utiliza uma das sucessões de Sturm. Disponibiliza a **função enumera(coef,inf,sup)**, onde:

coef → vetor com os coeficientes do polinômio. Deve ser dado completo, ou seja, coeficientes nulos, devem ser considerados;

inf → limite inferior do intervalo no qual serão enumeradas as raízes;

sup → limite superior do intervalo no qual serão enumeradas as raízes.

Exemplo

Seja enumerar as raízes reais da equação $y = x^3 – 6x + 2 = 0$, sabendo-se que estão nos intervalos:

[.25 3.45] e [-3.45 - .56]

➔ Entrando com o vetor dos coeficientes

```
-->c=[1 0 -6 2]
 c  =
!  1.    0.  - 6.    2. !
```

➔ Carregando a função enumera(coef,inf,sup)

```
-->getf('C:\Program Files\Scilab-2.6\macros\com400\enumera.sci');
```

➔Enumerando as raízes utilizando a função enumera(coef,inf,sup)

a) Enumerando as raízes positivas

-- Entrando com os limites inferior e superior

```
-->li=0.25
 li  =
     .25
-->ls=3.45
 ls  =
    3.45
```

- Usando a função enumera(coef,inf,sup)

```
-->enumera(c,li,ls)
Sucessão de Sturm
                3
   2 - 6x + x

           2
  - 6 + 3x

  - 2 + 4x

    5.25

Sinal em ----------> .3 e 3.5
                     +     +
                      -     +
                      -     +
                     +     +
Variações de sinal          2     0
O número de raízes no intervalo [ .3  3.5] é 2
```

b) Enumerando as raízes negativas

-- Entrando com os limites inferior e superior

```
-->ls=-0.56
 ls  =

  -  .56
```

- Usando a função enumera(coef,inf,sup)

```
-->enumera(c,li,ls)
Sucessão de Sturm
```

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

$$-6 + 3x^2$$

$$-2 + 4x$$

5.25

Sinal em ----------> -3.5 e - .6

-	+
+	-
-	-
+	+

Variações de sinal 3 2

O número de raízes no intervalo [-3.5 - .6] é 1

3.1.3 – Separação das raízes reais

Para separar as raízes reais de uma equação f(x) = 0, situadas em um intervalo [a, b],utilizando-se um recurso computacional, ao invés do método da bisseção pode-se fazer uso do procedimento descrito a seguir.

(i) estabelece-se um passo h;

(ii) calcula-se o valor de f(x) em cada ponto **a, a + h, a + 2h, ..., b – h, b**;

(iii) como já se sabe o número de raízes reais que a equação tem no intervalo [a, b], basta observar o sinal da f(x) em cada ponto para fazer a separação.

Para executar este procedimento no Scilab é necessário, antes, definir a função que dá origem à equação. A definição de uma função on line no Scilab é feita utilizando-se a linha de comando cuja sintaxe é apresentada a seguir.

deff('[var dep]=nome da função(var indep)',['forma analítica da função'])

Exemplo

Seja definir a função y = f(x) = x seno(x) – 1. Neste caso, tem-se que

→ Variável dependente: y

→ Nome da função: f

→ Variável independente: x

→ Forma analítica da função: y = x * seno(x) - 1

Desta forma, para este exemplo, a linha de comando que define a função é:.

deff('[y]=f(x)',['y=x*sin(x)-1'])

O nome da função e das varáveis são estabelecidos a critério do usuário.

Uma vez que a função foi definida, basta utilizar a linha de comando a seguir para gerar uma tabela com duas colunas 'x f(x)'. Na linha de comando há um espaço em branco entre **x** e **f(x)**.

for x=a:h:b,disp([x f(x)]),end

Exemplo

Seja separar as raízes reais da equação $f(x) = x^2 - 5x + 6 = 0$. Consideremos o intervalo [0; 4,2] e um passo h = 0,7. Defina a função, de nome é f, e digite a linha de comando:

for x=0:0.7:4.2,disp([x f(x)]),end

```
!  0.    6. !
!   .7   2.99 !
!  1.4    .96 !
!  2.1  -  .09 !
!  2.8  -  .16 !
!  3.5    .75 !
!  4.2   2.64 !
```

O Scilab produz os resultados ao lado, onde se pode observar que para x = 1.4 tem-se f(x) = 0.96 e para x = 2.1 f(x) = -0.09, logo há uma raiz no intervalo [1.4; 2.1]. Da mesma forma, para x = 2.8 f(x) = -0.16 e para x = 3.5 f(x) = 0.75 sendo assim, há outra raiz no intervalo [2.8; 3.5]. De fato, é facilmente verificado que as raízes desta equação são **2** e **3**. Se na primeira tentativa as raízes não forem separadas, basta diminuir o tamanho do passo.

3.2 – Métodos de cálculo das raízes reais de equações algébricas e transcendentes

3.2.1 – Método da bisseção

Este pacote está contido no arquivo **bissec.sci**. Trata-se de uma implementação do método da bisseção. Disponibiliza a função **bissec()**.

Exemplo

Seja calcular a raiz da equação f(x) = x seno(x) – 1 = 0 pertencente ao intervalo [0, 2].

➔ Carregando a função bissec()

-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\bissec.sci');

➔ Calculando a raiz usando a função bissec()

```
-->bissec()
**********************************
  ******** Método da Bisseção ********
**********************************
Limite inferior do intervalo que contém a raiz :
-->0
Limite superior do intervalo que contém a raiz :
-->2
Precisão desejada :
-->0.05
Número máximo de iterações :
-->10
Resultados fornecidos pelo Método da Bisseção
 Iter.       a          b          x          fx         deltax
   1      0.00000    2.00000    1.00000    - .15853     1.00000
   2      1.00000    2.00000    1.50000     .49624      .50000
   3      1.00000    1.50000    1.25000     .18623      .25000
   4      1.00000    1.25000    1.12500     .01505      .12500
   5      1.00000    1.12500    1.06250    - .07183      .06250
   6      1.06250    1.12500    1.09375    - .02836      .03125

Para a precisão estabelecida, qualquer valor do intervalo
[1.06250   1.12500] pode ser tomado como raiz.
```

3.2.2 – Método da falsa posição

Este pacote está contido no arquivo **falpos.sci**. Trata-se de uma implementação do método da falsa posição. Disponibiliza a função **falpos()**.

Exemplo

Seja calcular a raiz da equação x – 2*seno(x) – 0.5 = 0 sabendo-se que é próxima de 2.

➔ Carregando a função falpos()

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\falpos.sci');
```

➔ **Calculando a raiz usando a função falpos()**

```
-->falpos()

****************************************

   ******** Método da falsa posição ********

****************************************

 Limite inferior do intervalo que contém a raiz :
-->1.8

Limite superior do intervalo que contém a raiz :
-->2.3

 Precisão desejada :
-->0.001

 Número máximo de iterações :
-->10

 Resultados fornecidos pelo Método da falsa posição

 Iter.      a         b         x         fx
  1     1.80000    2.30000   2.13865   - .04746
  2     2.13865    2.30000   2.16016   - .00243
  3     2.16016    2.30000   2.16125   - .00012
```

Para a precisão estabelecida, 2.16125 é raiz.

3.2.3 – Método de Newton-Raphson

Este pacote está contido no arquivo **nr.sci**. Trata-se de uma implementação do método da falsa posição.

Disponibiliza a função **nr()**.

Exemplo

Seja calcular a raiz da equação $x^2 - 2 = 0$. Claramente, as raízes desta equação são $-\sqrt{2}$ e $+\sqrt{2}$.

➔ **Carregando a função nr()**

```
-->getf('C:\Arquivos de programas\Scilab-2.6\macros\com400\nr.sci');
```

➔ **Calculando a raiz usando a função nr()**

```
-->nr()

************************************
```

***** Método de Newton-Raphson *****

****
Estimativa inicial para a raiz:
-->1

Precisão desejada:
-->0.001

Número máximo de iterações:
-->10

Iter.	x	f(x)	df(x)	deltax
0	1.00000	-1.00000	2.00000	*******
1	1.50000	.25000	3.00000	.50000
2	1.41667	.00694	2.83333	- .08333
3	1.41422	.00001	2.82843	- .00245
4	1.41421	0.00000	2.82843	0.00000

Para a precisão estabelecida, 1.41421 é raiz.