UNIVERSIDADE FEDERAL DE OURO PRETO

Instituto de Ciências Exatas e Biológicas

Departamento de Computação

José Álvaro Tadeu Ferreira

O uso do Scilab na disciplina Cálculo Numérico

Ouro Preto

2009

1 - Introdução

O Scilab é um ambiente voltado para o desenvolvimento de software para a resolução de problemas numéricos. Foi criado em 1990 e mantido por pesquisadores pertencentes ao Institut de Recherche em Informatique et en Automatique, INRIA, através do Projeto MÉ-TALAU (Méthods, algorithmes et logiciels pour l'automatique) e à Ecole Nationale des Ponts et Chaussées, ENPC. Scilab é distribuído gratuitamente (free software) e em código aberto (open source software), via internet em www.scilab.org, desde 1994.

É um programa desenvolvido de forma a dispor, em um só ambiente, ferramentas de cálculo numérico, programação e gráficos. É similar ao MATLAB e outros programas de cálculo numérico. Pode ser utilizado em uma variedade de sistemas operacionais tais como U-NIX, Windows, Linux, etc.

A partir de maio de 2003, Scilab passou a ser mantido por um consórcio de empresas e instituições francesas denominado de Consórcio Scilab[1]. Os objetivos principais deste consórcio são:

⇒ Organizar a cooperação e intercâmbio entre os desenvolvedores do Scilab visando incorporar ao software os últimos avanços científicos na área de computação numérica;

⇒ Organizar a cooperação e intercâmbio entre os usuários do Scilab com o objetivo de fazer com que o software cumpra requisitos necessários para que possa ser utilizado com eficiência na indústria, pesquisa e educação;

⇒ Obter os recursos necessários para a manutenção da equipe de desenvolvedores e para garantir suporte adequado aos usuários

Do ponto de vista do usuário, o Scilab apresenta algumas vantagens tais como:

1- A última versão do software está sempre disponível, geralmente via Internet;

2- O software pode ser legalmente utilizado, copiado, distribuído, modificado;

3- Os resultados obtidos podem ser divulgados sem nenhuma restrição;

4- O acesso ao código fonte, evitando surpresas desagradáveis;

5- Os programas desenvolvidos podem ser transferidos para outras pessoas sem imposições de quaisquer natureza;

6- O acesso a informação de alta qualidade, e

7- A certeza de estar participando de uma comunidade cujo valor principal é irrestrita difusão do conhecimento.

[1]Anagram Technologies, Appedge, AXS Ingénierie, Cril Technology, CEA, CNES, Dassault-Aviation, EADS, Ecole Polytechnique, EDF, ENPC, Esterel Technologies, IFP, INRIA, Klippel, PSA, Renault, Styrel Technologies, Thales, TNI-Software

O Scilab é um ambiente de programação numérica bastante flexível. Suas principais características são:

1. Sua linguagem de programação é simples e de fácil aprendizado;
2. Possui um sistema de auxílio ao usuário, help;
3. É um ambiente poderoso para geração de gráficos bidimensionais e tridimensionais, inclusive com animação;
4. Possui muitas funções pré-definidas para a manipulação de matrizes. As operações de concatenação, acesso e extração de elementos, transposição, adição e multiplicação de matrizes são facilmente realizadas;
5. Permite trabalhar com polinômios, funções de transferência, sistemas lineares e grafos;
6. Apresenta facilidades para a definição de funções;
7. Permite o acesso a rotinas escritas nas linguagens FORTRAN ou C;
8. Pode ser acessado por programas de computação simbólica como o Maple, que é um software comercial, ou o MuPAD, que é livre para uso em instituições de ensino/pesquisa;
9. Suporta o desenvolvimento de conjuntos de funções voltadas para aplicações específicas, os chamados toolboxes.

Os objetivos principais deste texto são: (i) rever conceitos de cálculo numérico para aplicá-los na resolução de problemas e (ii) apresentar o Scilab tendo-se em vista o conteúdo programático da disciplina Cálculo Numérico – CIC170, ministrada para diversos cursos oferecidos pela Universidade Federal de Ouro Preto.

2 - Preliminares

2.1 – O Ambiente Scilab

Uma vez inicializado o Scilab, aparecerá na janela de comandos, mostrada na figura 2.1, um prompt -->. O prompt indica que o Scilab está esperando um comando. Todo comando deve ser finalizado teclando-se Enter.

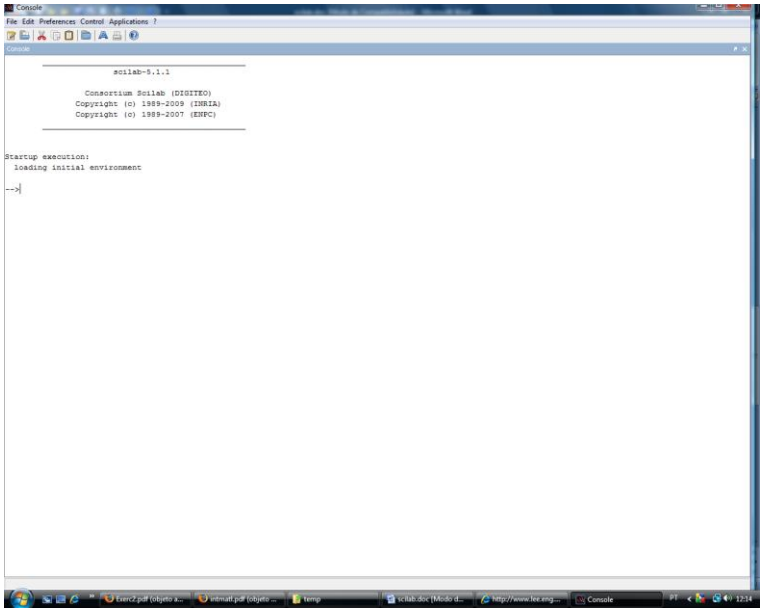


Figura 2.1: Janela de comandos do Scilab

No Scilab, pode-se obter ajuda sobre qualquer comando ou função utilizando a opção help (?) da barra de menus. O comando --> help nome (sem o prompt -->) permite obter ajuda sobre um pacote específico ou sobre um comando ou função específica.

2.2 – Operações Básicas

O Scilab faz cálculos simples e científicos como uma calculadora. Por exemplo, suponha que você vai a uma loja e compra 3 objetos que custam 25 reais cada e 5 objetos que custam 12 reais cada. Quanto custou a sua compra?

Este problema pode ser resolvido de pelo menos duas maneiras. A mais simples é

```
-->3*25+5*12
 ans  =
    135.
```

Observe que no Scilab a multiplicação tem precedência sobre a adição. Note-se, também, que o resultado foi chamado de ans.

Alternativamente, podem ser usadas variáveis para armazenar informações.

```
-->a=3, b=25, c=5, d=12
 a  =
    3.
 b  =
    25.
 c  =
    5.
```

```
 d  =
   12.
-->total=a*b+c*d
 total  =
   135.
```

Primeiro, foram criadas quatro variáveis, a, b, c e d, atribuindo a elas os seus valores respectivos. Observe que no Scilab o sinal de igual tem um sentido diferente daquele da Matemática. Aqui, igual significa atribuição. O que estiver à direita do sinal de igual é atribuído como conteúdo da variável que estiver à a esquerda. Finalmente, foi criada uma variável chamada total que recebeu o total da compra. Foi usada a vírgula para separar os comandos que são dados em uma mesma linha. Esta separação poderia ser feita com ponto e vírgula. Mas, neste caso, os resultados dos comandos não são mostrados.

Os operadores aritméticos são:

"+" ⇒ soma

"-" ⇒ subtração

"*" ⇒ multiplicação

"/" ⇒ divisão

"^" ⇒ potenciação

A ordem com que as expressões são avaliadas é dada pela seguinte regra: expressões são avaliadas da esquerda para a direita, com a potenciação tendo a mais alta precedência, seguida pela multiplicação e a divisão que têm igual precedência, seguidas pela adição e subtração que têm igual precedência. Parêntesis podem ser usados para alterar esta ordem. Sendo que neste caso, os parêntesis mais internos são avaliados antes dos mais externos.

2.3 – Área de Trabalho

Quando um comando de atribuição como este:

```
--> x = 3
```

é digitado no Scilab, a variável x é armazenada em uma área da memória do Scilab denominada de Área de Trabalho (do inglês, Workplace). A Área de Trabalho é uma parte da memória do computador que armazena as variáveis criadas no prompt de comando e pelos arquivos de *Script* (que serão tratados posteriormente).

Comandos anteriores podem ser obtidos novamente usando as teclas ↑ e ↓. Por exemplo, pressionando a tecla ↑ uma vez obtém-se o último comando digitado no prompt. Pressionando repetidamente a tecla ↑ são obtidos os comandos digitados anteriormente, um de

cada vez na direção para trás. Tem-se o mesmo pressionando-se a tecla ↓, mas na direção para frente. Em qualquer momento, as teclas ← e → podem ser usadas para a movimentação do cursor dentro de um comando, no prompt.

2.3.1 - O Comando clear

O comando clear apaga todas as variáveis da Área de Trabalho criadas pelo usuário.

Exemplo:

-->clear // Apaga todas as variáveis

O comando clear seguido de nome de uma variável apaga somente a variável nominada.

-->a = 2;

-->b = 3;

-->c = 4;

-->clear b; // Apaga somente a variável b.

2.4 - Formato de Visualização dos Números

O comando **format** modifica a quantidade de dígitos com que os números são mostrados no Scilab. Por exemplo, o comando

--> format(5)

fará com que todas os números sejam visualizados em 5 posições (incluindo o ponto decimal e um espaço para o sinal). Por exemplo,

-->sqrt(3)

ans =

1.73

Para aumentar o número de posições para 16, usa-se

-->format(16)

-->sqrt(3)

ans =

1.7320508075689

A raiz de 3 foi mostrada ocupando 16 posições (sendo uma posição para o ponto, um espaço reservado para o sinal, uma posição para a parte inteira e 13 posições para a parte fracionária).

O comando format('e') mostra os números em notação científica. Por exemplo,

-->format('e')

-->2*%pi/10

ans =

6.3D-01

Para retornar ao formato inicial, basta usar o comando:

--> format('v')

que é chamado de "formato de variável".

O comando:

-->format('v',10)

mostra os números em formato de variável com 10 posições, e

-->format('e',8)

Mostra os números em notação científica com 8 posições.

2.5 - Constantes Especiais

O Scilab possui um conjunto de constantes pré-definidas e que, normalmente, não podem ser alteradas. A seguir são apresentados alguns exemplos.

%pi - valor de π (3,1415926...)

%e - número de Euler (2,7182818...)

%eps - precisão da máquina (%eps+1=1)

%inf - infinito

%i - unidade imaginária ($\sqrt{-1}$)

%nan - significa não é um número, por exemplo, 0/0

%s - um polinômio com uma única raiz em zero e "s" como o nome da variável. A constante "s" é definida como poly(0,"s").

2.6 – Variáveis e o comando de atribuição

Uma variável é uma abstração de uma célula ou um conjunto de células na memória do computador. Informações são armazenadas em variáveis para posterior uso.

Fundamental na programação, o comando de atribuição é usado para atribuir ou modificar a informação contida na variável. No Scilab, usa-se o símbolo "=" para o comando de atribuição. O símbolo de atribuição "=" não significa igualdade matemática, uma que o comando d atribuição "i = i + 1" é válido, mas não representa igualdade matemática.

Exemplo: digite estes comandos no prompt do Scilab:

```
-->a = 2 // Atribui 2 para variável a
a =
2.
-->b = 4 // Atribui 4 para variável b
a =
4.
-->area = a*b // Atribui o produto de a e b para a variável área.
area =
8.
```

A variável ans (abreviação da palavra inglesa answer) armazena o valor corrente de saída do Scilab. Pode-se usar ans para efetuar cálculos porque ela armazena o valor do último cálculo realizado. Exemplo:

```
-->4+5
ans =
9.
-->cos(ans)+3
ans =
```

2.6.1 Regras para Formação de Nomes de Variáveis

A formação de nomes de variáveis (também conhecidos por identificadores) deve obedecer às seguintes regras[2]:

1. Nomes de variáveis começam com uma letra seguido de letras, algarismos ou sublinhados. Por exemplo: Alpha, notas, A1, B23 e cor_do_objeto;
2. Caracteres especiais não são permitidos;
3. Caracteres acentuados não são permitidos;
4. Há diferença entre maiúsculas e minúsculas. Por exemplo, variável Alpha é diferente das variáveis ALPHA, alpha e AlPhA.

De acordo com as regras acima, os seguintes nomes de variáveis são válidos:

ALPHA, X, B1, B2, b1, matricula e MEDIA.

Porém, estes nomes de variáveis são inválidos: 5B, Nota[1], A/B, X@Z.

[2] Esta regra é única, ou seja, é utilizada para nomear qualquer objeto trabalhado pelo Scilab. Por padrão, todos os objetos pré-definidos têm seus nomes em minúsculo.

2.6.2 O Ponto e Vírgula

O ponto-e-vírgula no final de um comando do Scilab faz com que o resultado da execução não seja apresentado. Seja, por exemplo, o comando:

```
-->A = 4+4^2
A =
20.
```

No entanto, o mesmo comando, digitado com ponto e vírgula, não tem seu resultado apresentado. é suprimido:

```
-->A = 4+4^2;
-->
```

2.7 - Funções

O Scilab possui um conjunto de funções pré-definidas. A seguir é apresentada uma lista de algumas das funções disponíveis.

abs(x) - valor absoluto de x.
sin(x) - seno de x.
asin(x) - arco cujo seno é x.
cos(x) - cosseno de x.
acos(x) - arco cujo cosseno é x.
tan(x) - tangente de x.
atan(x) - arco cuja tangente é x.
cotg(x) - cotangente de x.
exp(x) - exponencial e^x.
log(x) - logaritmo de x na base e.
log10(x) - logaritmo de x na base 10.
log2(x) - logaritmo de x na base 2.
sqrt(x) - raiz quadrada de x.
modulo(x,y) - resto da divisão inteira de x por y.
int(x,y) – quociente da divisão inteira de x por y.

2.7 – Expressões aritméticas

As expressões aritméticas são formadas pela combinação de constantes, variáveis e funções. Exemplos:

A+B*C

(NOTA1+NOTA2)/2

1/(a^2+b^2)

2+3*cos(x)

3 – Vetores e Matrizes

3.1 – Vetores

Os componentes de um vetor são escritos entre colchetes e separados por um espaço, vírgula ou ponto-e-vírgula. Quando separados por espaço ou vírgula dão origem a um vetor linha. Se forem separados por ponto-e-vírgula, geram um vetor coluna. Sendo assim, para criar um vetor v, de n elementos, as sintaxes possíveis são as apresentadas a seguir.

(i) b = [b_1 b_2 b_3 ... b_n] → vetor linha

(ii) b = [b_1, b_2, b_3, ..., b_n] → vetor linha

(iii) b = [b_1; b_2; b_3; ...; b_n] → vetor coluna

3.2 - Matrizes

Os elementos de cada linha de uma matriz são escritos separados por um espaço ou vírgula, enquanto a separação entre linhas é feita com ponto-e-vírgula (ou seja, o ponto-e-vírgula indica o final de uma linha. Logo, para criar uma matriz **m**, de **k** linhas e **c** colunas, as sintaxes possíveis são as apresentadas a seguir.

(i) m = [m_{11} m_{12} ... m_{1c}; m_{21} m_{22} ... m_{2c}; ... ; m_{k1} m_{k2} ... m_{kc}]

(ii) m = [m_{11}, m_{12}, ..., m_{1c}; m_{21}, m_{22}, ..., m_{2c}; ... ; m_{k1}, m_{k2}, ..., m_{kc}]

O quadro 3.1 apresenta alguns comandos para definir matrizes especiais e para realizar algumas operações sobre matrizes.

Função	Resultado
zeros(n,m)	Uma matriz n x m com elementos nulos.
ones(n, m)	Uma matriz n x m com todos os elementos iguais a um.
eye(n,m)	Uma matriz identidade n x m.
length(a)	O número total de elementos da matriz a.
size(a)	n, número de linhas, e m o número de colunas da matriz a.
diag(a)	O vetor correspondente à diagonal principal da matriz a.
diag(c)	Sendo c um vetor, a matriz diagonal com os elementos de c na diagonal principal.
triu(a)	A parte triangular superior da matriz a.
tril(a)	A parte triangular inferior da matriz a.
trace(a)	A soma dos elementos da diagonal principal de a (que é o traço).
max(a)	O valor máximo dos elementos da matriz a.
max(a,'r')	Um vetor linha com os elementos máximos de cada coluna.
max(a,'c')	Um vetor coluna com os elementos máximos de cada linha.
det(a)	O determinante de a.
inv(a)	A inversa de a.
a'	A transposta de a.

Quadro 3.1

4 – Gráficos em duas dimensões

Nesta seção será tratada a elaboração de gráficos em duas dimensões no ambiente do Scilab. Com este objetivo, serão apresentadas algumas funções pré-definidas que permitirão elaborar gráficos básicos.

A forma mais simples de produzir um gráfico em duas dimensões no Scilab é por meio do comando ***plot(.)***, cuja sintaxe básica é **plot([x],y)**. Esta função foi construída de modo a ter uma sintaxe próxima daquela utilizada pelo Matlab.

Os parâmetros x e y podem ser matrizes ou vetores reais. Observe-se que o parâmetro x, entre colchetes, é opcional, se omitido, é assumido como sendo o vetor (1x n) onde n é o número de pontos da curva dados pelo parâmetro y. Podem ocorrer os casos a seguir.

1. Se x e y são vetores, a função plot(.) permite traçar o gráfico de y em função de x. É importante observar que os dois vetores devem ter a mesma dimensão, isto é, os dois vetores devem ter o mesmo número de elementos;
2. Se x é um vetor e y é uma matriz, a função plot(.) permite traçar o gráfico de cada coluna da matriz y em função do vetor x. Neste caso, o número de elementos das colunas da matriz y deve ser igual ao número de elementos do vetor x;
3. Se x e y são matrizes, a função plot(.) permite traçar o gráfico de cada coluna da matriz y em função de cada coluna da matriz x. Neste caso, as matrizes devem ter as mesmas dimensões;
4. Se y é um vetor e x é omitido, a função plot(.) permite traçar o gráfico do vetor y em função do vetor [1:size(y)], onde size(y) é o número de elementos do vetor y,e

5. Se y é uma matriz e x é omitido, a função plot(.) permite traçar o gráfico de cada coluna de y em função do vetor [1:size(y)], onde size(y) é o número de elementos de cada linha de y.

O *script* apresentado no código 4.1, a seguir, permite elaborar gráficos considerando os cinco casos. A figura 4.1apresenta os resultados obtidos. Neste *script* foram utilizadas as funções *subplot(.)* e *xtitle(.)* que serão posteriormente detalhadas.

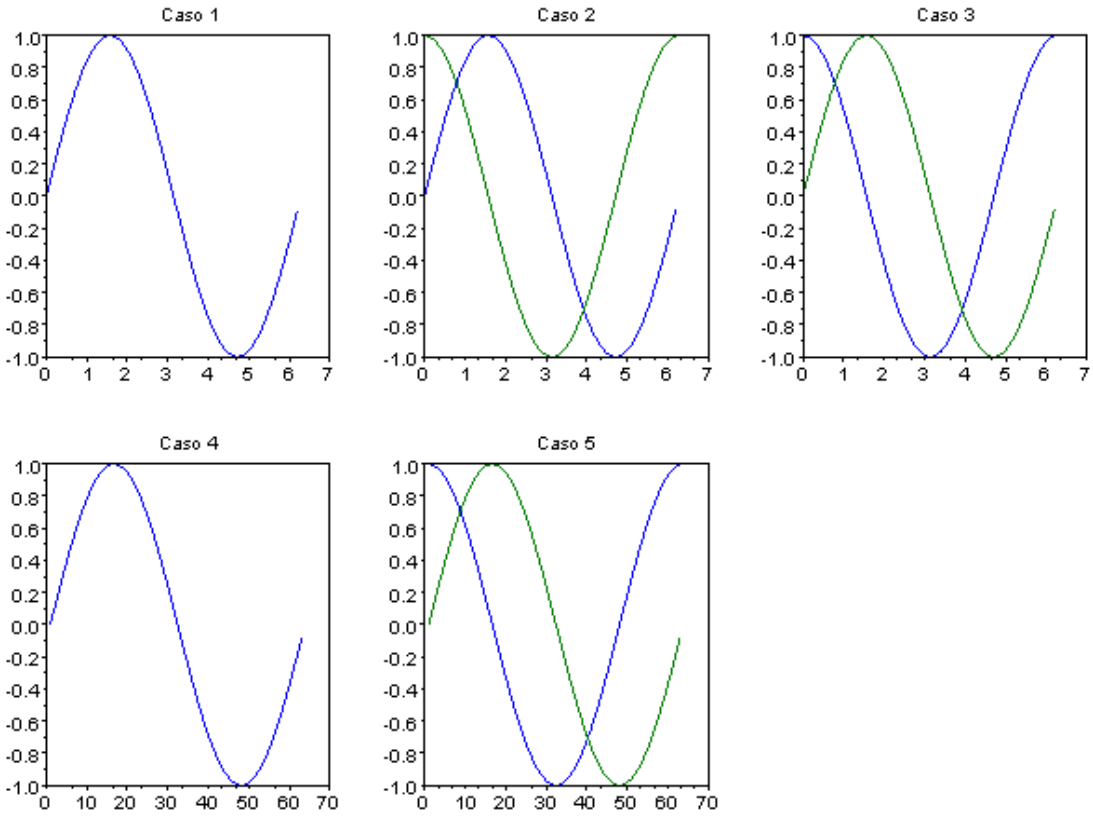
Um *scrit* é um conjunto de comandos do Scilab que podem ser executados, de imediato, na área de trabalho do Scilab, ou podem ser armazenados em um arquivo para posterior, execução. Os scripts são formados por texto puro, sem acentuação, contendo uma sequência de comandos que o usuário digitaria em uma sessão interativa no *prompt* do Scilab. Por convenção, os arquivos de scripts do Scilab possuem extensão *sce* e são executados por meio da opção *Execute* do menu *File*.

São características dos arquivos scripts:

- as variáveis definidas num script são globais, isto é, depois da chamada e execução do script estas variáveis permanecem ativas;
- não têm parâmetros (“argumentos”) de entrada nem de saída, esse fato pode dificultar a correção de erros.

```
// Script para gerar a figura 4.1
// Adaptado de: Prof. Paulo Sérgio da Motta Pires
// http://www.dca.ufrn.br/~pmotta

// Definindo o vetor das abcissas, x
x = [0:0.1:2*%pi];

// Caso 1 – x e y são vetores
y = sin(x);
subplot(231)
plot(x,y)
xtitle("Caso 1");

// Caso 2 – x é um vetor e y uma matriz
// Definindo a matriz yy
yy = [sin(x)' cos(x)'];
subplot(232)
plot(x,yy)
xtitle("Caso 2");

// Caso 3 - x e y sao matrizes
// Definindo uma variavel auxiliar
t = [0:0.1:2*%pi];
// Criando a matriz xx
xx = [t' t'];
// Criando a matriz yy
yy = [cos(t)' sin(t)'];
subplot(233)
plot(xx,yy)
xtitle("Caso 3");

// Caso 4 - y vetor (x é omitido)
subplot(234)
plot(sin(x))
xtitle("Caso 4");

// Caso 5 - y matriz (x é omitido)
subplot(235)
plot(yy)
xtitle("Caso 5");
```

Código 4.1: Script para elaborar os gráficos apresentados na figura 4.1



Figural 4.1:

Observe-se que os gráficos dos casos 4 e 5 possuem valores de abscissas diferentes dos demais.

4.1 - Especificação do tipo e cor da linha e do marcador

O Scilab permite customizar a aparência da linha do gráfico, como cor, tipo da linha e tipo de marcador a ser utilizado. A referência deve ser feita na forma de um *string* (a ordem não é importante. Por exemplo, para especificar uma linha vermelha tracejada e marcador em forma de diamante, pode ser escrito: “r--d” ou “--dire” ou “--reddiam” ou “diamondred--“.

Exemplo – 4.1

Seja construir o gráfico da função y = sen(x) no intervalo [0, 10] com linha vermelha tracejada e marcadores em diamante.

```
--> x=1:0.1:10;
--> plot(x,sin(x),”r.-d”)
```

A figura 4.2 apresenta o gráfico obtido.

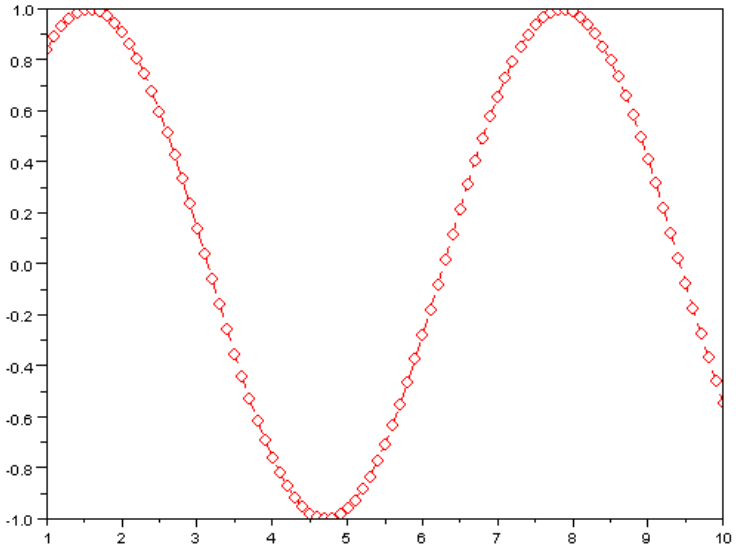


Figura 4.2: gráfico com linha vermelha tracejada e marcadores em diamante.

Podem ser construídos mais de um gráfico em um mesmo sistema de eixos utilizando diferentes especificações para cada um.

Exemplo – 4.2

Seja construir, em um mesmo sistema de eixos e no intervalo [0, 10], o gráfico da função y = sen(x) com a linha contínua em vermelho e marcador na forma de círculo e da função y = cos(x) com o marcador na forma de "+" em azul sem linha.

```
--> t=0:%pi/20:2*%pi;
--> plot(t,sin(t),"ro-",t,cos(t),"b+")
```

A figura 4.3 mostra o resultado obtido.

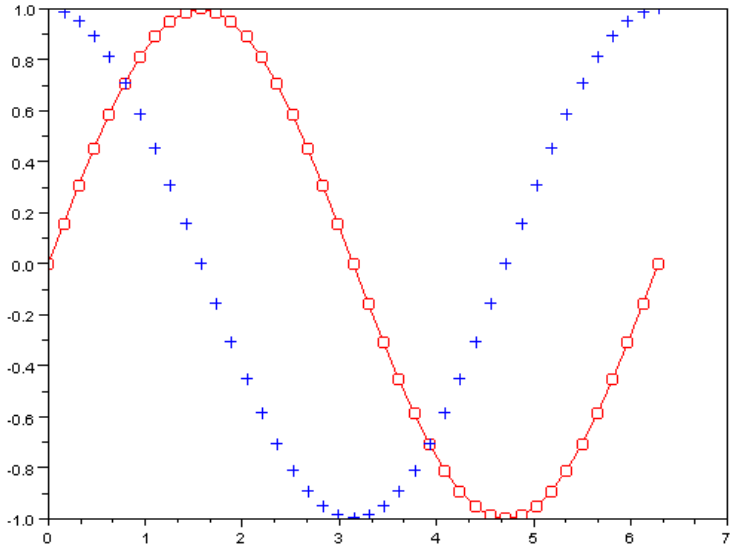


Figura 4.3: gráficos das funções y = sem(x) e y = cós(x).

A seguir, são apresentadas as tabelas de cores, de tipos de linhas e marcadores disponíveis.

Especificação	Cor
r	Vermelho (Red)
g	Verde (Green)
b	Azul (Blue)
c	Ciano (Cyan)
m	Magenta
y	Amarelo (Yellow)
k	Preto (Black)
w	Branco (White)

Tabela 4.1: Cores

Especificação	Estilo da linha
-	Linha cheia (default)
--	Linha tracejada
:	Linha pontilhada
-.	Linha cheia com ponto

Tabela 4.2: Linhas

Especificação	Tipo de marcador
+	Sinal de amis
o	Círculo
*	Asterisco
.	Ponto
x	Cruz
'square' or 's'	Quadrado
'diamond' or 'd'	Diamante
^	Triangular para cima
v	Triangular para baixo
>	Triângulo para a direita
<	Triângulo para a esquerda
'pentagram'	Estrela de cinco pontas
'none'	No marker (default)

Tabela 4.3: Marcadores

4.2 - Adicionando títulos

Para colocar título no gráfico, assim como nos eixos é utilizada a função *xtitle(.)*, cuja sintaxe é:

xtitle(título do gráfico,[título do eixo x,[título do eixo y]],[arg_opc])

Note-se que argumentos entre colchetes são opcionais. Se arg_opc for boxed = 1, um retângulo é desenhado ao redor de cada título.

Exemplo – 4.3

Seja construir o gráfico da função y = sen(x) no intervalo [0, 10] com a linha em vermelho e colocando os títulos.

```
-->x=1:0.1:10;
-->plot(x,sin(x),"r-")
-->xtitle("Função y = sen(x) no intervalo [0, 10]", "Eixo x", "Eixo Y", boxed = 1)
```

A figura 4.4 apresenta o gráfico obtido.

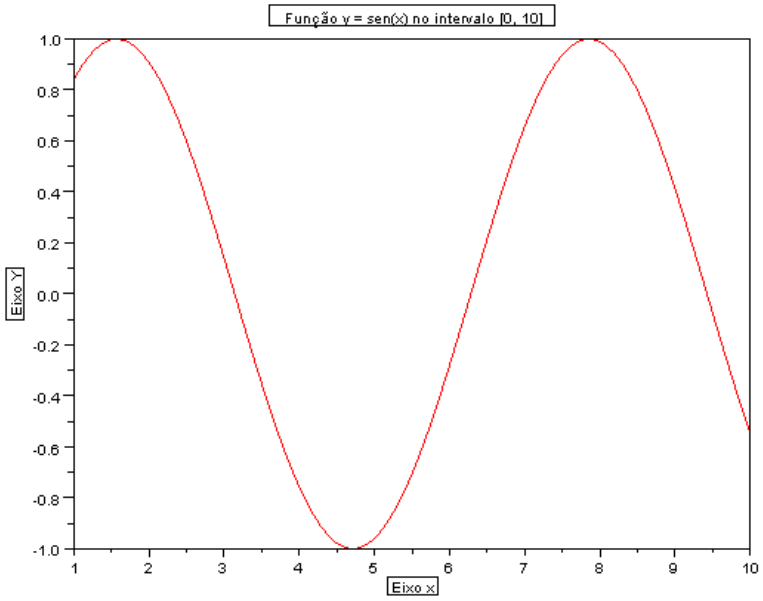


Figura 4.4: Gráfico com títulos.

4.3 - Adicionando linhas de grade

Neste caso, utiliza-se a função xgrid(.), cuja sintaxe é

xgrid([*estilo*])

onde *estilo* é um parâmetro opcional para identificar a cor a ser utilizada.

Exemplo – 4.4

Seja construir o gráfico da função y = sen(x) no intervalo [0, 10] com a linha em vermelho, colocando os títulos e as linhas de grade na cor azul.

```
-->x=1:0.1:10;
-->plot(x,sin(x),"r-")
-->xtitle("Função y = sen(x) no intervalo [0, 10]", "Eixo x", "Eixo Y", boxed = 1)
-->xgrid(2)
```

O gráfico obtido é apresentado na figura 4.5.

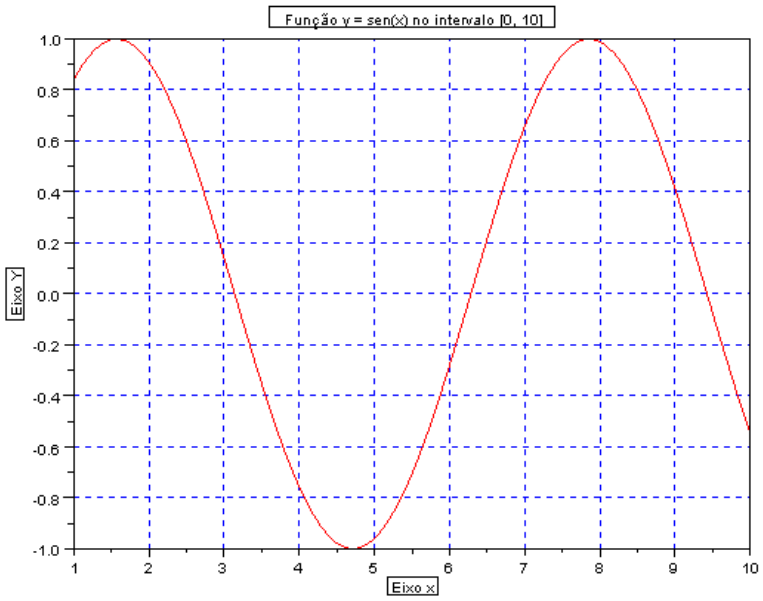


Figura 4.5: Gráfico com títulos e linhas de grade.

4.4 - Adicionando legendas

Quando se traça mais de um gráfico em um mesmo sistema de eixos, é possível definir legendas para identificar cada curva utilizando a função *legend(.)*, cuja sintaxe é a apresentada a seguir.

legend(strings [,pos] [,boxed]

strings : vetor com n strings, strings(i) é a legenda da i-ésima curva;

pos : parâmetro opcional que especifica onde deve ser colocada a legenda; pode ser um valor inteiro ou um vetor [x,y] o qual dá as coordenadas para colocação da legenda. No primeiro caso os valores possíveis são:

1 ou "ur" a legenda é colocada no canto superior direito;
2 or "ul" a legenda é colocada no canto superior esquerdo;
3 or "ll" a legenda é colocada no canto inferior esquerdo;
4 or "lr" a legenda é colocada no canto inferior direito;
5 or "?" a legenda é colocada de forma interativa com o mouse (default).

boxed: é um parâmetro booleano (o valor *default* é %t) para colocar, ou não, um box na legenda.

Exemplo – 4.5

Seja construir, em um mesmo sistema de eixos e no intervalo [0, 2pi], o gráfico da função y = sen(x) com a linha contínua em vermelho, da função y = cos(x) com linha tracejada em azul e colocando legendas.

```
--> t=0:%pi/20:2*%pi;
--> plot(t,sin(t),"r-",t,cos(t),"b--")
-->legend(["y = sin(t)"; "y = cos(t)"],5)
```

A figura 4.6 mostra o resultado obtido.

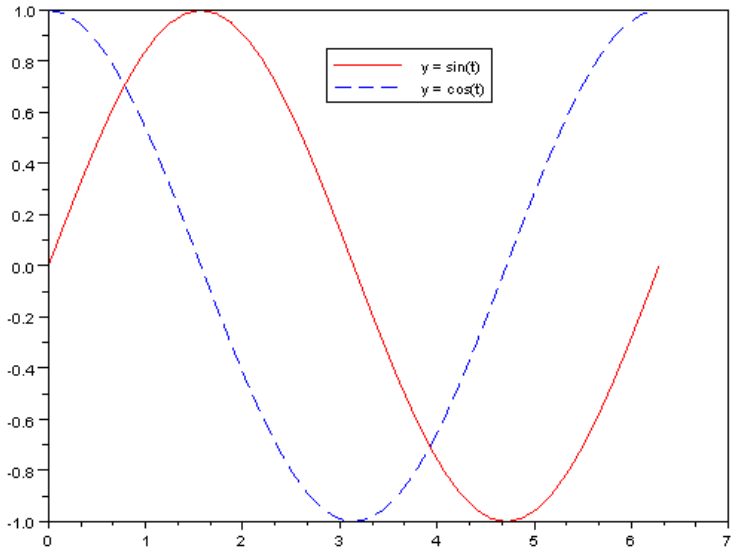


Figura 4.6: Gráfico com legendas.

4.5 - Apresentando vários gráficos em uma mesma janela

O comando **subplot(*m*,*n*,*p*)** permite dividir a janela gráfica do Scilab em uma matriz de *m* linhas por *n* colunas. Em cada um dos elementos da "matriz", identificado por *p*, pode ser colocado um gráfico. Um exemplo está mostrado na figura 4.1, que foi construída utilizando o código 4.1.

Exemplo – 4.6

Para envolver o que foi tratado nesta seção, seja construir, em um mesmo sistema de eixos e no intervalo [0, 2pi], o gráfico da função y = sen(x) com a linha contínua em vermelho, da função y = cos(x) com linha tracejada em azul, colocando legendas, linhas de grade e títulos.

```
--> t=0:%pi/20:2*%pi;
--> plot(t,sin(t),"r-",t,cos(t),"b--")
-->legend(["y = sin(t)"; "y = cos(t)"],5)
-->xtitle("Exemplo final","x","y")
-->xgrid()
```

O gráfico obtido é o que está apresentado na figura 4.7.

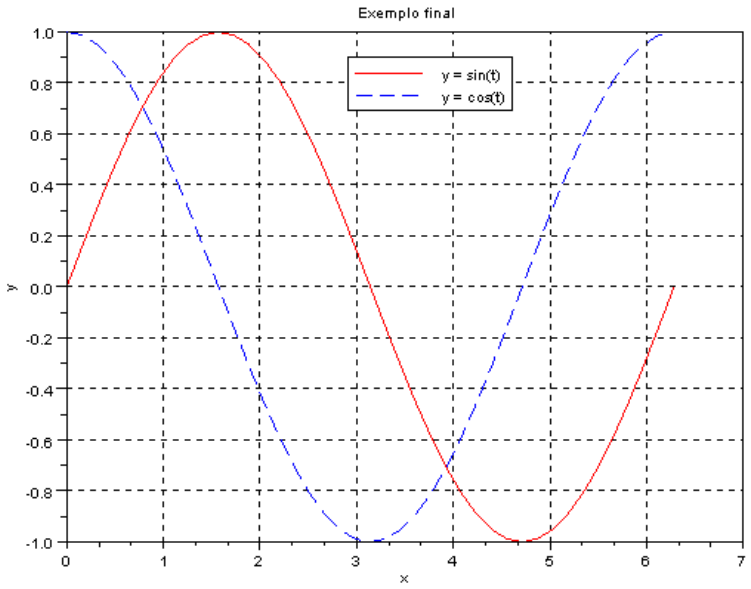


Figura 4.7: Gráfico com elementos abordados nesta seção

5 – Resolução de Sistemas de Equações Lineares Simultâneas

5.1 – Eliminação de Gauss com Pivotação Parcial

O SCILAB contém uma implementação do método de eliminação de Gauss com pivotação parcial, por meio da função \. Essa função implementa a "divisão" a esquerda de matrizes. Desta forma a\b produz a matriz **inv(a)*b** Consequentemente, para resolver o sistema de equações lineares **a.*x***=b, basta utilizar o comando x=a\b.

Exemplo – 5.1
Seja resolver o sistema de equações a.x = b, onde

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

→ Entrando com a matriz dos coeficientes

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

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

→ Entrando com os termos independentes

```
-->b=[-8 ;7; 6]
 b  =
!  -8. !
!   7. !
!   6. !
```

→ Resolvendo o sistema

```
-->x=a\b
 x  =
!   1. !
! - 2. !
!   1. !
```

5.2 – Decomposição LU com Pivotação Parcial

A decomposição **LU** com pivotação parcial encontra-se implementada na função **lu**, cuja sintaxe é a apresentada a seguir.

[l u p]=lu(a)

A matriz ***p*** é a matriz de permutação, ela é formada pela permutação das linhas da matriz identidade de mesma ordem que a matriz dos coeficientes das incógnitas. De forma que em cada linha o único elemento não nulo é igual a **1**. Se esse elemento estiver na diagonal, significa que esta linha não sofreu nenhuma permutação, se este elemento da linha ***i*** estiver na coluna ***j*** significa que durante o processo as linhas ***i*** e ***j*** foram permutadas.

Uma vez obtidos l, u e p, para resolver o sistema de equações Ax = b, basta:

(i) resolver o sistema de equações **l.y = p.b** (b na forma coluna) e, a seguir;

(ii) resolver u.**x = y**.

Exemplo – 5.2

Seja o sistema de equações utilizado no exemplo 5.1

→ Obtenção das matrizes l, u e p

```
-->[l u p]=lu(a)
 p  =

!   0.    1.    0. !
!   1.    0.    0. !
!   0.    0.    1. !

 u  =

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

 l  =

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

→ Resolvendo o sistema l.y = p.b

```
--> y=l\(p*b)

 y  =

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

→ Resolvendo o sistema u.x = y

```
-->x=u\y
 x  =

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

6 – Interpolação Polinomial

O Scilab possui um conjunto de funções pré-definidas que são implementações de vários métodos de interpolação. Digitando a linha de comando "apropos interpolation" (sem as-pas) será obtida a lista completa no navegador da janela de Ajuda (Browse Help). Neste texto será abordada a função **interp1**.

A sintaxe de utilização desta função é:

yp = **interp1**(x, y, xp, "método")

Dados (x,y,xp), esta função determina o yp correspondente a cada xp, por interpolação, linear por padrão, utilizando os pares (x, y).

Se yp é um vetor, então o comprimento de xp deve ser igual ao comprimento de yp. Se yp é uma matriz, então xp deve ter o mesmo comprimento que cada uma das colunas de yp.

O parâmetro "método" tem por função estabelecer o tipo de interpolação a utilizar, que pode ser:

linear: a interpolação é definida pelo método linear, ou seja, cada par de pontos é unido por uma reta;

spline: definição de interpolação por spline cúbico, o que significa que cada par de pontos é unido por um polinômio de grau três;

nearest: para cada valor xp(j), yp(j) toma o valor ou y(i) correspondente ao x(i) o vizinho mais próximo de xp(j).

Exemplo – 6.1

Considere-se o tanque com água mostrado na figura 5.1.

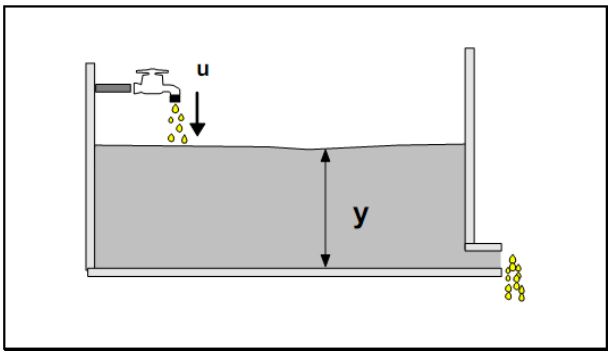


Figura 6.1: Tanque com água

Para estudar o sistema, abriu-se a torneira e anotaram-se os valores da altura da água, em metros, durante os primeiros dez minutos, conforme mostrado na tabela 5.1.

Tempo (t)	0	1	2	3	4	5	6	7	8	9	10
Altura (y)	0	0,7	2,4	3,1	4,2	4,8	5,7	5,9	6,2	6,4	6,4

Tabela 6.1: Valores da evolução da altura da água

O objetivo é estimar o valor da altura da água em pontos intermediários. Pretende-se, por exemplo, estimar a altura da água no instante 3,5 minutos.

Começa-se efetuando a entrada dos vetores t e y.

```
-->t=0:1:10
 t  =

    0.    1.    2.    3.    4.    5.    6.    7.    8.    9.    10.

-->y = [0 0.7 2.4 3.1 4.2 4.8 5.7 5.9 6.2 6.4 6.4]
 y  =

    0. 0.7 2.4 3.1 4.2 4.8 5.7 5.9 6.2 6.4 6.4
```

→ Interpolação linear

```
-->y1=interp1(t,y,3.5,"linear")
 y1  =

    3.65m
```

É fácil de entender graficamente o resultado da interpolação de todos os pontos: retas que unem cada um dos pontos adjacentes, como se mostra na figura 5.2.

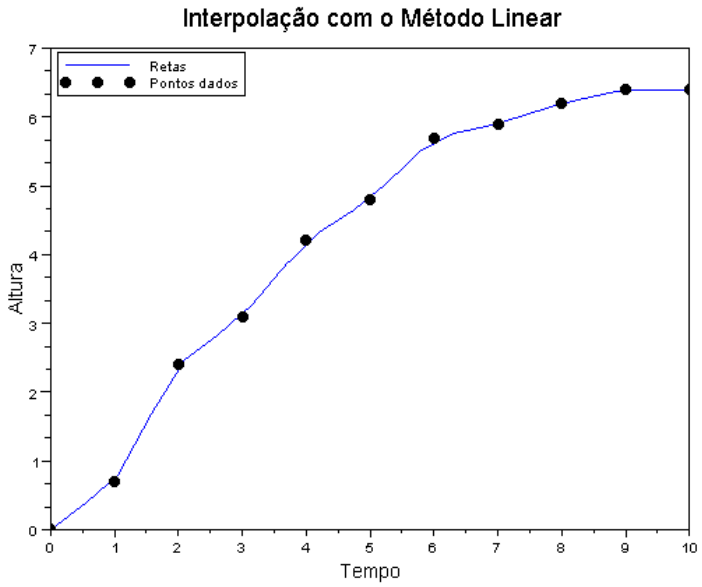


Figura 6.2: Interpolação da altura da água com o método linear

→ Interpolação com spline cúbico

```
-->y2=interp1(t,y,3.5,"spline")
 Y2  =

    3.645691m
```

Como já foi citado anteriormente, neste caso, os pontos adjacentes são unidos por polinômios de grau três, daí o nome spline cúbico. A figura 6.3 ilustra o processo.

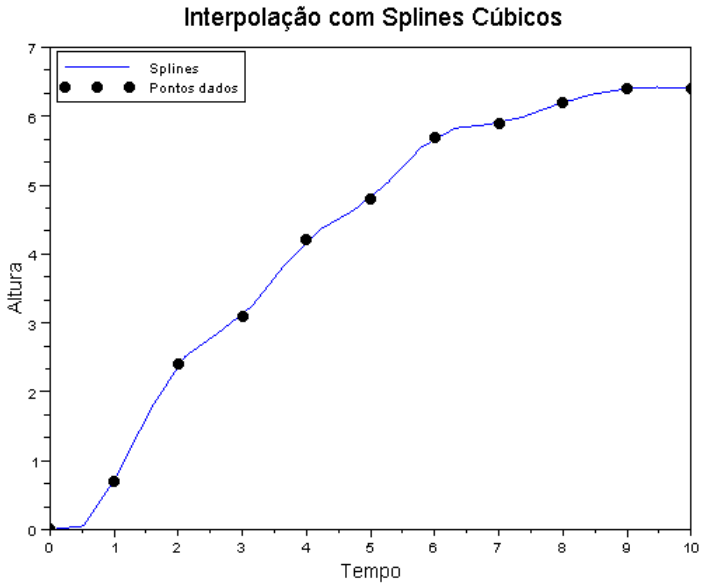


Figura 6.3: Interpolação da altura da água com o método spline

Exemplo – 6.2

Para o exemplo do tanque de níveis, foram recolhidos valores para a altura do nível da água considerando variável o diâmetro da torneira (de 1 a 1,4 centímetros). Esses resultados mostram-se na tabela seguinte.

	Diâmetro				
Tempo	**1,0**	**1,1**	**1,2**	**1,3**	**1,4**
0	0,0	0,0	0,0	0,0	0,0
1	0,7	4,0	3,3	3,7	3,8
2	2,4	4,2	2,4	5,2	5,0
3	3,1	4,1	3,5	6,1	5,8
4	4,2	4,8	4,5	6,2	6,3
5	4,8	5,2	5,2	7,2	7,4
6	5,7	5,4	5,5	7,7	7,7
7	5,9	6,1	6,5	7,9	8,0
8	6,2	6,5	6,9	8,2	8,3
9	6,4	6,7	7,4	8,2	8,4
10	6,4	6,8	7,5	8,1	8,3

Com estes dados pretende-se estimar o valor da altura da água, num instante qualquer, tendo em vista, também, o diâmetro da torneira. Trata-se, então, de interpolação a duas dimensões, isto é, o valor da variável a estimar depende de duas variáveis, neste caso, o tempo e a temperatura.

Para resolver este tipo de problema, é necessário, inicialmente, utilizar a função **splin2d**. A sintaxe básica para a utilização desta função é

c = splin2d(x, y, z)

x, y: vetores linhas estritamente crescentes (com pelo menos dois componentes) definindo o grid de interpolação

z: matriz nx x ny (nx sendo o comprimento de x e ny o comprimento de y)

c: um vetor grande com os coeficientes dos elementos de área bicúbicos.

Esta função computa um spline bicúbico, **s**, que interpola os pontos (xi,yj,zij), isto é, tem-se s(xi,yj)=zij para todo i=1,..,nx e j=1,..,ny. O spline resultante, **s**, fica definido, de forma única, pela tripla (x, y, c) onde c é o vetor com os coeficientes de cada um dos (nx-1)(ny-1) elementos de área bicúbicos.

A seguir, para avaliar splin2d, em um ponto xp, yp; é utilizada a função **interp2d**, cuja sintaxe básica é:

zp=interp2d(xp, yp, x, y, c)

xp, yp: vetores ou matrizes de reais de mesmo tamanho

x,y,c: vetores de reais definindo uma função de spline bicúbico, s

zp: vetor ou matriz com o mesmo formato que xp e yp, avaliação elemento a elemento de s nestes pontos.

Para maiores detalhes a respeito destas duas funções, deve ser consultado o help do Scilab e a sua documentação.

Seja, então, estimar o valor do nível da água no tanque no instante 3,5min e diâmetro da torneira 1,25cm.

Deve ser feita, inicialmente, a entrada dos dados. Seja **t** o vetor de temperatura, **d** o vetor de diâmetro e **h** a matriz de altura.

```
-->t=read("t.txt",1,11)
 t  =
   0.   1.   2.   3.   4.   5.   6.   7.   8.   9.   10.

-->d=read("yd.txt",1,5)
 d  =
   1.   1.1  1.2   1.3   1.4
```

```
-->h=read("zh.txt",11,5)
 h  =
   0.    0.    0.    0.    0.
   0.7   4.    3.3   3.7   3.8
   2.4   4.2   2.4   5.2   5.
   3.1   4.1   3.5   6.1   5.8
   4.2   4.8   4.5   6.2   6.3
   4.8   5.2   5.2   7.2   7.4
   5.7   5.4   5.5   7.7   7.7
   5.9   6.1   6.5   7.9   8.
   6.2   6.5   6.9   8.2   8.3
   6.4   6.7   7.4   8.2   8.4
   6.4   6.8   7.5   8.1   8.3
```

Observe-se que os vetores **t** e **d** e matriz **h** foram armazenados em um arquivo texto para posterior leitura, consulte o help do Scilab para aprender com isto é feito.

Agora o problema proposto pode ser resolvido.

```
-->c = splin2d(t,d,h);

-->hp=interp2d(3.5,1.25,t,d,c)
 hp  =
   4.960106
```

Portanto, a estimativa obtida para o n´vel de água é **hp = 4.960106m**.

7 – Integração Numérica

7.1 – Definição de funções

A definição de funções *on-line* é feita utilizando-se a função **deff**, cuja sintaxe é apresentada a seguir.

deff(“vardep = nomefunc(varindep)”,”vardep = forma analítica da função”)

Onde

vardep: é o nome da variável dependente;

varindep: é o nome da variável independente e

nomefunc: é o nome da função.

Exemplo – 7.1

Seja definir a função $f(x) = 25 – 4.x – 10.e^{-0,4.x}$. Fazendo

vardep: y, varindep: x e nomefunc: f

Sendo assim, a função é definida por meio da linha de comando a seguir.

```
-->deff("y = f(x)", "y = 25 - 4*x - 10*exp(-0.4*x)")
```

7.2 – Cálculo da integral definida dada a forma analítica da função

Existem duas funções pré-definidas para o cálculo da integral definida de uma função.

7.2.1 - Função integrate

A sintaxe desta função é a apresentada a seguir.

--> integrate('função','variável',limite inferior,limite superior,erro absoluto, erro relativo)

Os erros absoluto e relativo são parâmetros opcionais. Caso não sejam especificados, o erro absoluto é tomado como 10^{-8} e o erro relativo como 10^{-14}.

Exemplo – 7.2

Seja estimar a integral definida da função $f(x) = - 0{,}05.\log(0{,}4076).e^{- 0{,}05.x}$ considerando o intervalo [0, 50].

```
--> v = integrate('-0.05*log(0.4076)*exp(-0.05*x)',"x",0, 50)
 v  =
   0.8238002
```

Observe-se que não é necessário definir a função a ser integrada previamente, caso o seja, a sua expressão analítica deve ser substituída pelo seu nome colocado entre aspas.

7.2.2 Função intg

Para utilizar esta função é necessária a definição prévia da função a ser integrada. A sintaxe é a seguinte.

--> [valor erro] = intg(limite inferior,limite superior, nome da função, ea, er)

Onde
valor é o resultado obtido;
erro é o valor estimado do erro absoluto;
ea é o erro absoluto requerido no resultado (opcional, o valor *default* é 1.D-14);
er é o erro relativo requerido no resultado (opcional, o valor *default* é 1.D-8).

Exemplo – 7.3

Seja estimar a integral definida proposta no exemplo 7.2.

Vejamos a sua utilização:

```
-->deff('y=f(x)','y=-0.05*log(0.4076)*exp(-0.05*x)')
-->[v ea]=intg(0,50,f)
 ea  =

   9.146D-15
 v  =

   0.8238002
```

Observe-se que resultado é o mesmo que o do exemplo 7.2.

7.3 – Cálculo da integral definida conhecendo-se um conjunto de pontos

É também possível calcular uma integral definida quando se tem um conjunto de pontos. É o que fazem as funções **inttrap**, que emprega a interpolação trapezoidal, e **intsplin** que usa a interpolação pelo método spline.

7.3.1 – Função inttrap

A sintaxe desta função é:

--> integral = inttrap(abscissas, ordenadas).

Exemplo - 7.4

Seja estimar a integral definida da função f(x) = ln(x + 2) – 1 no intervalo [2; 3,2], sendo h=0,24.

```
-->x = 2:0.24:3.2]
 x  =

   2.    2.24    2.48    2.72    2.96    3.2

-->deff("y=f(x)","y=log(x + 2) - 1")

-->y = f(x)
 y  =

   0.3863    0.4446    0.4996    0.5518    0.6014    0.6487

-->integral = inttrap(x, y)
 integral  =

   0.6276
```

7.3.2 – Função intsplin

A sintaxe do comando intsplin é semelhante à de inttrap, porém fornece melhor resultado.

--> integral = intsplin(abscissas, ordenadas).

Exemplo - 7.5

Seja a mesma integral do exemplo 6.3.

-->integral = intsplin(x, y)

integral =

0.6278

7.4 – Cálculo de integrais duplas

É utilizada a função pré-definida **int2d(.)**. Esta função avalia a integral bidimensional de uma função, z = f(x, y), sobre uma região constituída de dois triângulos. Sua sintaxe básica é

[valor erro] = int2d(x, y, f)

valor: resultado obtido.

erro: estimativa para o erro cometido.

x: matriz 3x2 que contém as abscissas dos vértices dos dois triângulos.

y: matriz 3x2 que contém as ordenadas dos vértices dos dois triângulos.

f:função integranda (deve ser definida previamente).

Sendo $I = \int_{x_i}^{x_s} \int_{y_i}^{y_s} f(x,y)\,dy\,dx$, as matrizes **x** e **y** são da forma:

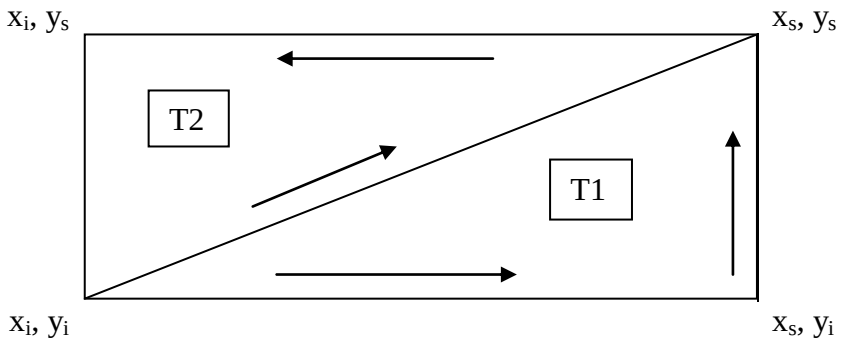
$$x = \begin{bmatrix} x_i & x_i \\ x_s & x_s \\ x_s & x_i \end{bmatrix} \qquad y = \begin{bmatrix} y_i & y_i \\ y_i & y_s \\ y_s & y_s \end{bmatrix}$$

Que, no ambiente do Scilab, são digitadas da seguinte maneira:

--> x = [x_i x_i; x_s x_s; x_s x_i]

--> y = [y_i y_i; y_i y_s; y_s y_s]

A figura a seguir ilustra a divisão da região de integração em dois triângulos, T1 e T2, e a sequência de leitura dos vértices para a montagem das matrizes x e y.



Exemplo

(a) Sendo $f(x,y)=\dfrac{\text{sen}(x.y)}{x^2+y}$ estime $I=\int_{0,1}^{0,9}\int_{0,2}^{0,5} f(x,y)\,dy\,dx$.

Definição da função

--> deff("y=f(x,y)","y=sin(x*y)/(x^2 + y)")

Entrada das matrizes **x** e **y**

-->x=[.1 .1;.9 .9;.9 .1];

-->y=[.2 .2;.2 .5;.5 .5];

Estimativa da integral

-->[v erro]=int2d(x,y,f)

erro =

4.886D-11

v =

0.0601534

(b) Sendo $f(x,y)=\dfrac{1}{(x+y)^2}$ estime $I=\int_{3}^{4}\int_{1}^{2} f(x,y)\,dy\,dx$.

Definição da função

--> deff("y=f(x,y)","y = 1/(x + y)^2")

Entrada das matrizes **x** e **y**

--> x=[3 3;4 4;4 3];

--> y=[1 1;1 2;2 2];

Estimativa da integral

-->[v erro]=int2d(x,y,f)

erro =

1.236D-11

v =

0.0408220

(c) Sendo $f(x,y) = x^2(x+y)\sqrt{x+y^2}$ estime $I = \int_0^1 \int_0^3 f(x,y)\,dy\,dx$.

Definição da função

--> deff("y=f(x,y)","y = x^2*(x + y)*sqrt(x + y^2)")

Entrada das matrizes **x** e **y**

--> x=[0 0;1 1;1 0];

--> y=[0 0;0 3;3 3];

Estimativa da integral

-->[v erro]=int2d(x,y,f)

erro =

9.268D-11

v =

4.6744419

(d) Sendo $f(x,y) = \frac{x^2}{1+y^2}$ estime $I = \int_0^1 \int_{0,5}^{1,25} f(x,y)\,dy\,dx$.

Definição da função

--> deff("y=f(x,y)","y = x^2/(1 + y^2)")

Entrada das matrizes **x** e **y**

-->x=[0 0;1 1;1 0];

-->y=[0.5 0.5;0.5 1.25;1.25 1.25];

Estimativa da integral

-->[v erro]=int2d(x,y,f)

erro =

1.435D-11

v =

0.1441359

8 – Resolução de Equações Não Lineares

8.1 – Polinômios

Nesta seção é abordado o uso do Scilab para efetuar algumas operações polinomiais.

8.1.1 – Definição de um polinômio

Um polinômio é definido utilizando-se a função **poly**. Para tanto, existem três maneiras.

(a) Definindo um polinômio com base em uma variável polinomial

Inicialmente, utilizando-se a função poly, define-se a variável polinomial "x".

Sintaxe

```
-->x=poly(0,"x")
 x =
   x
```

Para definir, por exemplo, o polinômio $p(x) = x^2 + x + 6$, utiliza-se a variável polinomial "x", conforme mostrado a seguir.

```
-->p=x^2-5*x+6
 p =
                2
   6 - 5x + x
```

(b) Definindo um polinômio com base nos seus coeficientes

Sintaxe

--> p = poly([coeff, "var", "c")

Onde c é um vetor que contém os coeficientes do polinômio em ordem crescente de grau e var é o nome da variável.

Seja, por exemplo, definir o polinômio $p(x) = x^2 + x + 6$.

```
-->p=poly([6 1 1],"x","c")
 p =
               2
   6 + x + x
```

Mais um exemplo, seja o polinômio $p(x) = x^3 – x$.

```
-->p=poly([0 -1 0 1],"x","c")
 p  =
          3
  - x + x
```

(c) Definindo um polinômio com base nas suas raízes

Sintaxe

--> p = poly([raiz, "var", "r")

Onde raiz é um vetor que contém as raízes do polinômio.

Seja, por exemplo, definir o polinômio $p(x) = x^2 – 5.x + 6$.

```
-->p=poly([2 3],"x","r")
 p =

               2
   6 - 5x + x
```

8.1.2 – Cálculo do valor numérico de um polinômio

É utilizada a função **horner**.

Sintaxe

--> valor = horner(polinômio, x)

Onde "polinômio" é o nome do polinômio e "x" é o ponto no se deseja avaliá-lo. Observe-se que x pode ser um escalar ou um vetor.

Seja o polinômio $p(x) = x^2 – 5.x + 6$ avaliado no ponto x = 5.

```
-->valor = horner(p, 5)
 valor  =
   6.
```

8.1.3 – Operações com polinômios

(a) Operações algébricas

Sejam os polinômios, $p1 = x^5 - 2.x^4 – 7.x^3 + 9.x^2 + 8.x - 6$ e $p2 = x^3 - x^2 – 4.x + 4$.

```
-->p1=poly([-6 8 9 -7 -2 1],"x","c")
 p1  =
                2     3     4     5
  - 6 + 8x + 9x - 7x - 2x + x

-->p2=poly([4 -4 -1 1],"x","c")
 p2  =
              2    3
   4 - 4x - x + x
```

Soma

```
-->ps = p1 + p2
 ps  =
                2     3     4     5
  - 2 + 4x + 8x - 6x - 2x + x
```

Da mesma forma pode ser feita a subtração.

Multiplicação

```
-->pm = p1 * p2
 pm  =
                    2       3      4       5    6     7    8
 - 24 + 56x + 10x - 78x + 19x + 28x - 9x - 3x + x
```

Divisão

Para esta operação é utilizada a função **pdiv**, cuja sintaxe é a mostrada a seguir.

--> [r q] pdiv(poli1, poli2)

Onde:

r é o resto da divisão;

q é o quociente;

pol1 é o nome do polinômio dividendo e

poli2 é o nome do polinômio divisor.

Seja dividir o polinômio p1 por p2.

```
-->[r q]=pdiv(p1,p2)
 q  =
          2
 - 4 - x + x
r  =
            2
   10 - 4x - 3x
```

(b) Derivada de um polinômio

Para obter a derivada de um polinômio basta utilizar a função **derivat(.)**, sua sintaxe é

d= derivat(poli)

Onde d é a variável para receber o resultado e poli é o nome do polinômio. Seja obter a primeira derivada de p1.

```
-->dp1 = derivat(p1)
 dp1  =
                    2    3      4
   8 + 18x - 21x - 8x + 5x
```

Observe-se que para as operações soma, subtração, multiplicação e divisão os polinômios devem ser da mesma variável.

8.1.4 – Zeros de polinômios

Para calcular todos os zeros de um polinômio, de grau menor ou igual a 100, com coeficientes reais ou complexos, é utilizada a função **roots**, cuja sintaxe é a apresentada a seguir.

--> raiz = roots(poli)

Onde raiz é o vetor para receber o resultado e poli o nome do polinômio.

Seja determinar os zeros de p1.

```
-->r=roots(p1)
 r  =

   0.5664957
 - 1.0543445
 - 2.1377228
   1.5066762
   3.1188954
```

Seja, agora, o cálculo dos zeros do polinômio $p(x) = x^4 – 12.x^3 + 22.x^2 – 20.x$.

```
-->p=poly([0 -20 22 -12 1],"x", “c”)
 p  =

              2       3     4
 - 20x + 22x - 12x + x
-->roots(pc)
 ans  =

   0
   1. + i
   1. - i
   10.
```

8.2 – Funções

Na seção anterior foi abordado um caso particular de funções, as polinomiais. Nesta, é tratado do cálculo de zeros de funções genéricas.

8.2.1 – Determinação dos zeros de uma função

Para a determinação dos xeros de uma função qualquer, o Scilab disponibiliza a função pré-definida **fsolve**, cuja sintaxe é:

[zero valorfunc] = fsolve(x0, nomefunc, prec)

Onde

x0 é a estimativa inicial da raiz;

nomefunc é o nome da função (deve ser definida anteriormente);

prec é a precisão desejada (opcional, se não for especificado é tomado como 1.D-10);

zero é a estimativa obtida para um zero da função e

valorfunc é o valor numérico da função em x = zero.

Esta função é baseada no método iterativo de Newton-Raphson.

Exemplo – 8.1

Seja obter estimativas para os zeros da função $f(x) = 25 – 4.x – 10.e^{-0,4.x}$. O gráfico 8.1 mostra que esta função tem dois zeros, um próximo de (- 4) e outro próximo de 6.

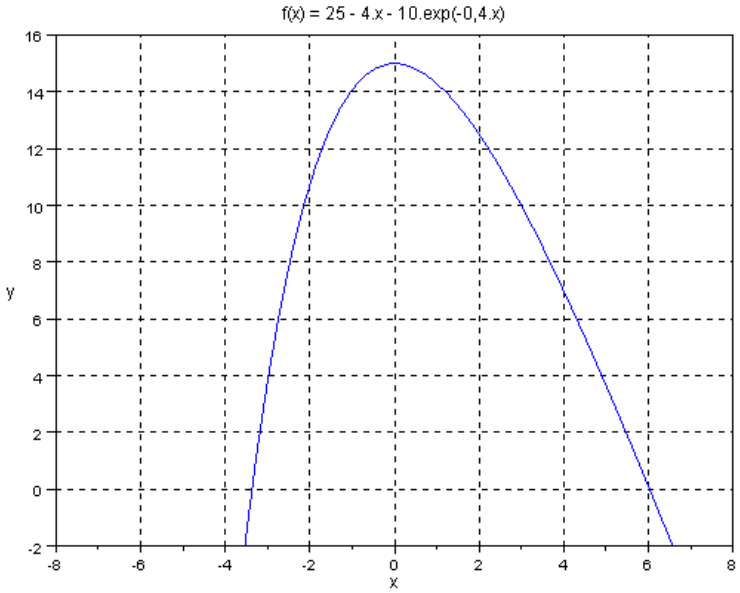


Gráfico 8.1: Zeros da função f(x) = 25 -4.x – 10.exp(-0,4.x)

A seguir, são obtidas estimativas para os dois zeros da função.

```
-->[x v] = fsolve(-4,f)
 v  =
   0.
 x  =
  - 3.3684919
-->[x v] = fsolve(6,f)
 v  =

  - 3.331D-16
 x  =

   6.0255073
```

Referências

http://www.lenep.uenf.br/~nivaldo/disciplinas/softLivre/turma2008/apresentacoes/LucasPriscila/scilab.pdf - Acessado em 13 de maio de 2009.

http://www.ing.una.py/DIREC_PPAL/ACADEMICO/APOYO/calculo%20numerico/Scilab/CURSO%20DE%20SCILAB.pdf – Acessado em 12 de maio de 2009.

http://www.mat.ufmg.br/~regi/topicos/intmatl.pdf - Acessado em 12 de maio de 2009.

http://www.lee.eng.uerj.br/~elaine/aula1_2007.pdf - Acessado em 12 de maio de 2009.

http://www.ceit.com.br/download/MatScilab_01.pdf - Acessado em 17 de maio de 2009.

www.**scilab**.org/publications/SB**Scilab**/livroSci.pdf - Acessado em 10 de abril de 2009.

http://www.dca.fee.unicamp.br/~moraes/softII/aulas/Exerc2.pdf - Acessado em 20 de maio de 2009.

http://www.dca.ufrn.br/~pmotta/sciport-3.0.pdf - Acessado em 27 de maio de 2009.