Quarto roteiro de exercícios no Scilab
Cálculo Numérico

Rodrigo Fresneda

24 de abril de 2012

Guia para respostas:

- Entregue suas respostas às tarefas contidas no roteiro de cada uma das quatro atividades, incluindo quaisquer algoritmos pedidos.
- Não copie do pdf e cole no Scilab para evitar erros de compilação: digite todos os comandos novamente.
- Data limite para entrega: 04/05/2012

Parte I
Interpolação polinomial

Nesta atividade iremos calcular o polinômio interpolador p de grau $n-1$ ou menor aos n pontos distintos (x_i, y_i),

$$p(x_i) = y_i\,,\ i = 1, ..., n$$

Para tanto iremos nos valer de duas formas para p vistas em sala: aquela dada pela fórmula de Lagrange, e aquela dada pela fórmula de Newton.

1 Polinômios interpoladores de Lagrange

A fórmula de Lagrange para o polinômio interpolador de grau $n-1$ é dada por

$$p(x) = \sum_{i=1}^{n} y_i L_i(x)$$

em que y_i, $i = 1, ..., n$, são as coordenadas "dependentes" na tabela (x_i, y_i), e os $L_i(x)$ são os polinômios

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

A seguinte função do Scilab obtém a interpolação num ponto *xin* usando a fórmula de Lagrange para o polinômio interpolador:

Algoritmo 1 Fórmula de Lagrange

```
function [yin] = ilagrange(x,y,xin)
//ilagrange ajusta um polinomio de Lagrange a um conjunto de pontos dado
//e usa o polinomio para determinar o valor interpolado de um ponto.
//Variaveis de entrada:
//x Vetor coluna com as coordenadas x dos pontos dados
//y Vetor coluna com as coordenadas y dos pontos dados
//xin  A coordenada x do ponto a ser interpolado
//Variavel de saida:
//yin O valor interpolado de xin
// O comprimento do vetor x fornece o numero de termos do polinomio
n = length(x);
for i = 1:n,
        //Calcula os termos Li do produtorio
        L(1,i) = 1;
        for j = 1:n,
                if j ~= i,
                        L(1,i) = L(1,i)*(xin - x(j))/(x(i) - x(j));
                end
        end
end
//Calcula o valor do polinomio da Eq. (16)
yin = **********;
endfunction
```

2 Polinômios interpoladores de Newton

A formula de Newton para o polinômio interpolador de grau $n-1$, para os n pontos de interpolação (x_i, y_i), $i = 1, ..., n$, e

$$p(x) = \sum_{i=1}^{n} [y_1, ..., y_i] (x - x_1) (x - x_1) \cdots (x - x_{i-1})$$

em que os coeficientes $[y_1, ..., y_{i+1}]$ são as diferenças divididas de ordem i dos y_i em relação aos x_i. As diferenças divididas de ordens 0, 1 e 2 são:

$$\begin{aligned}
[y_1] &= y_1 \\
[y_1, y_2] &= \frac{y_2 - y_1}{x_2 - x_1} \\
[y_1, y_2, y_3] &= \frac{1}{x_3 - x_1}\left(\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}\right) \\
&= \frac{[y_2, y_3] - [y_1, y_2]}{x_3 - x_1}
\end{aligned}$$

Seguindo essa relação de recorrência, podemos escrever

$$[y_1, y_2, y_3, y_4] = \frac{[y_2, y_3, y_4] - [y_1, y_2, y_3]}{x_4 - x_1}$$

e de modo geral,

$$[y_1, ..., y_k] = \frac{[y_2, ..., y_k] - [y_1, ..., y_{k-1}]}{x_k - x_1}$$

O seguinte programa pode ser utilizado para interpolar um ponto xin usando polinômios de Newton.

Algoritmo 2 Fórmula de Newton

```
function yin = inewton(x,y,xin)
//inewton ajusta um polinomio de Newton a um dado conjunto de pontos e
//usa esse polinomio para determinar o valor interpolado de um ponto.
//Variaveis de entrada:
//x Vetor com as coordenadas x dos pontos dados
//y Vetor com as coordenadas y dos pontos dados
//xin Coordenada x do ponto a ser interpolado
//Variavel de saida:
//yin O valor interpolado de xin.
n = length(x); //Comprimento do vetor x fornece o numero de coeficientes
//(e termos) do polinomio
a(1) = y(1); // Primeiro coeficiente a1
for i = 1:n-1, //Calcula as diferencas divididas de ordem 1
        //Elas sao armazenadas na 1a coluna de dif
        dif(i,1) = ********************;
end
for j = 2:n-1, //Calcula as diferencas divididas de ordem 2 ate (n-1)
        //Elas sao armazenadas nas colunas de dif
        for i = 1:n-j,
                dif(i,j) = (dif(i+1,j-1)-dif(i,j-1))/(x(i+j)-x(i));
        end
end
for j = 2:n, //Atribui os coeficientes a2 a an ao vetor a
        a(j) = dif(1,j-1);
end
//Calcula o valor interpolado de xin
yin = a(1);
xn=1;
for k = 2:n,
        xn = xn*(xin-x(k-1));
        yin = *********;
end
endfunction
```

Atividade 1

1. Complete as linhas com *** das funções ilagrange e inewton.

2. Sabe-se que um dispositivo não-linear, quando alimentado por uma tensão U (em volts) apresenta como saída a corrente I (em mA), dada por

$$I = 5U^3 - 8U^2 + 20U \tag{1}$$

Usando a equação acima, preencha seguinte tabela com os valores esperados para a corrente I:

$U\,(V)$	-7.00	-5.00	-3.00	-1.00	1.00	3.00	5.00	7.00	9.00
$I\,(mA)$									

3. Qual o valor de I para $U = 0V$?

4. Usando apenas os $n = 2$ primeiros pontos da tabela, obtenha por extrapolação uma estimativa para a corrente quando $U = 0V$ usando a função ilagrange. Qual o erro absoluto da estimativa?

5. Repita o item 4 para $n = 3, 4, 5, 6, 7, 9$. Comente os resultados obtidos. Se você usasse o programa inewton, os resultados seriam diferentes? Justifique.

6. O programa abaixo traça num mesmo gráfico a função dada na (1) para $-10 \leq U \leq 10$ e o polinômio interpolador utilizando os $n = 3$ primeiros pontos da tabela. Comente os resultados obtidos.

Algoritmo 3

```
x = linspace(-10,10,100);
y = 5*x.^3-8*x.^2+20*x;
v = [-7 -5 -3];-
i = 5*v.^3-8*v.^2+20*v;
for int = 1:length(x)
        xin = x(int);
        yin(int) = ilagrange(v,i,xin);
end
plot(x,y,x,yin,'r',v,i,'*');
xgrid;
xlabel('v');
ylabel('i');
legend('Valores exatos', 'Polinomio de Lagrange', 'Dados usados na interp.');
```

7. Modifique o programa do item 6 de forma a usar os $n = 4$ primeiros pontos da tabela. Comente os resultados obtidos.

3 Aproximação polinomial a um conjunto de $n+1$ pontos pelo método dos mínimos quadrados

Dado um vetor x com n componentes distintas e um vetor y com n valores, queremos encontrar o polinômio p de grau $m < n - 1$ que melhor se ajusta aos pontos (x_i, y_i) no sentido de minimizar a distância

$$\|y - p\| = \sqrt{\sum_{i=1}^{n} \left(y_i - p\left(x_i\right)\right)^2} \tag{2}$$

Algoritmo 4 Calcula o polinômio P nos pontos x.

```
P=poly(a,'x','coeff')
horner(P,x)
```

Como vimos em sala, esse problema é equivalente a resolver o sistema linear

$$\begin{pmatrix} (e_0,e_0) & (e_1,e_0) & (e_2,e_0) & \cdots & (e_m,e_0) \\ (e_0,e_1) & (e_1,e_1) & (e_2,e_1) & \cdots & (e_m,e_1) \\ (e_0,e_2) & (e_1,e_2) & (e_2,e_2) & \cdots & (e_m,e_2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (e_0,e_m) & (e_1,e_m) & (e_2,e_m) & \cdots & (e_m,e_m) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} (y,e_0) \\ (y,e_1) \\ (y,e_2) \\ \vdots \\ (y,e_m) \end{pmatrix} \tag{3}$$

onde os vetores e_i, $i = 0, ..., m$ dados abaixo são linearmente independentes,

$$e_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \; e_1 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, \; e_2 = \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ \vdots \\ x_n^2 \end{pmatrix}, ..., \; e_m = \begin{pmatrix} x_1^m \\ x_2^m \\ x_3^m \\ \vdots \\ x_n^m \end{pmatrix},$$

e o produto escalar no $\mathbb{R}^n$ é

$$(x,y) = \sum_{i=1}^{n} x_i y_i \,, \;\; x, y \in \mathbb{R}^n$$

Atividade 2

1. Escreva uma rotina que recebe os vetores x e y de mesmo comprimento como dados de entrada e calcula a matriz de coeficientes e o vetor de constantes do sistema de equações normais para um dado grau m do polinômio de ajuste (3).

2. Utilize sua rotina, juntamente com a rotina de eliminação gaussiana da aula passada para obter o polinômio de grau 5 que melhor se ajusta aos dados contidos no arquivo dados.txt que se encontra na página do curso.

3. Finalmente, calcule o erro na sua aproximação de acordo com a expressão (2). Dica: utilize a função "poly" para criar um polinômio em "x" cujos coeficientes são as componentes do vetor "a", e a função "horner" para calcular o polinômio nos pontos dados pelo vetor "x":

Para ler uma tabela de dados com 2 colunas de números a partir de um arquivo, utilize os comandos:

Algoritmo 5 Como ler dados a partir de um arquivo

```
dados = mopen('/caminho/para/arquivo/dados.txt','r');    //abre arquivo
if (dados == -1)
error('nao e possivel ler arquivo');
end
[num, x, y] = mfscanf(-1,dados,'%f %f');  //ler tabela com duas colunas
mclose(dados); //fecha arquivo
```

Parte II
Integração Numérica

4 integração por trapézios

A regra do trapézio para a integração numérica de funções é baseada na aproximação da integral a partir da interpolação de uma reta que passa pelos limites de integração a e b. Na figura abaixo podemos observar que a integral neste caso equivale à área de um trapézio com bases $f(a)$ e $f(b)$ e altura $(b-a)$.

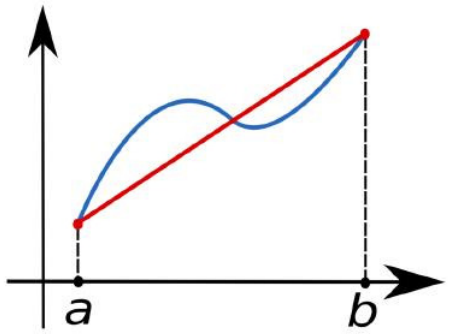

Figura 1: Integração por trapézio em 2 pontos

A área abaixo da curva é dada então por:

$$I \approx \frac{(b-a)}{2}\left[f(a)+f(b)\right]$$

Dividindo o intervalo $[a,b]$ em n sub-intervalos podemos calcular a integral de $f(x)$ pela soma do valor da integral em cada subintervalo.

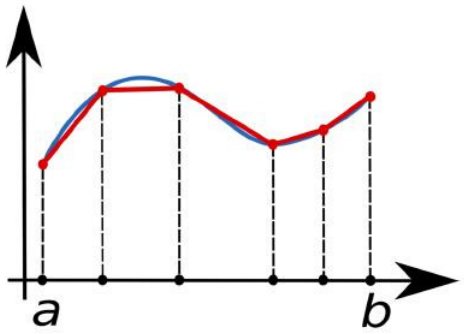

Figura 2: Integração por trapézios em $n+1$ pontos

$$I \approx \int_{x_1=a}^{x_2} f(x)\,dx + \int_{x_2}^{x_3} f(x)\,dx + \cdots + \int_{x_n}^{x_{n+1}=b} f(x)\,dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx$$

Aplicando a regra do trapézio a cada um dos subintervalos, temos:

$$I \approx \sum_{i=1}^{n} \left(\frac{x_{i+1} - x_i}{2} \right) [f(x_{i+1}) + f(x_i)]$$

No caso dos intervalos serem igualmente espaçados temos que:

$$x_{i+1} - x_i = h$$

Finalmente:

$$I \approx \frac{h}{2} \sum_{i=0}^{n} [f(x_{i+1}) + f(x_i)]$$

A expressão acima pode ser facilmente implementada no Scilab para efetuar o cálculo da integral $f(x)$.

Atividade 3

1. Escreva uma função no Scilab para calcular, por meio da fórmula dos trapézios, a integral dos dados tabulados abaixo. Os argumentos da função devem ser os vetores x e y dos dados.

2. Utilizar a função acima para calcular a área contida sobre os pontos da tabela abaixo (Resp. $0,32146$).

x	1.00	1.05	1.10	1.15	1.20	1.25	1.30
y	1.000	1.0247	1.0488	1.0723	1.0954	1.1180	1.1401

3. Reescreva a função que você criou para o cálculo do valor da integral de uma função. Os argumentos da função devem ser: a própria função a ser integrada, os limites de integração e

o número de subintervalos. Utilize a função recém criada para o cálculo das integrais abaixo:

$$\int_0^1 e^{-x^2} dx \qquad resp.0,746818$$

$$\int_0^1 e^{-x} \sin(x)\, dx \qquad resp.\ 0,24584$$

Dica: seu algoritmo pode receber como parâmetro uma string como "exp(-x^2)" e calcular a integral da função representada por esse string. Para isso, dentro do seu algoritmo você pode criar uma outra rotina que recebe a string e um número, e calcula a função representada por essa string no ponto especificado:

Algoritmo 6 Exemplo de implementação de função genérica

```
function [y]=F(z,string)
        x=z
        y=evstr(string) \\calcula a string (funcao de x) no ponto z
endfunction
```

5 regra de Simpson 1/3

A fórmula de integração por trapézios se baseia na aproximação do integrando por um polinômio de primeira ordem, para então integrar o polinômio no intervalo desejado. A regra de Simpson 1/3 é uma extensão da regra dos trapézios em que o integrando é aproximado por um polinômio de grau 2 (ver figura abaixo). Assim:

$$I = \int_a^b f(x)\, dx \approx \int_a^b p(x)\, dx$$

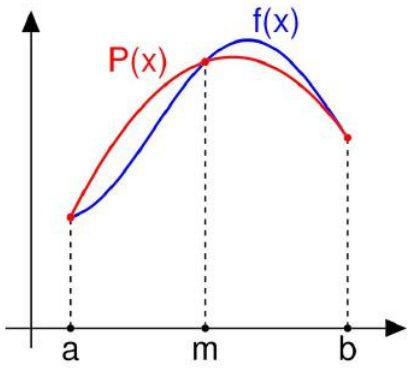

Figura 3: Simpson 1/3: $p(x)$ é um polinômio de grau 2

Utilizando um polinômio interpolador de Lagrange, para os pontos de interpolação a, b e o ponto médio $m = (a+b)/2$ temos

$$p(x) = f(a)\frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m)\frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b)\frac{(x-a)(x-m)}{(b-a)(b-m)}$$

Após integrar, temos:

$$I = \approx \int_a^b p(x)\,dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Observe que como são necessários 3 pontos para a definição de um polinômio de grau 2, o intervalo de integração $[a,b]$ é dividido em 2 intervalos adjacentes. Desta forma a equação acima pode ser escrita como:

$$I \approx \int_a^b p(x)\,dx = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

em que

$$h = \frac{b-a}{2}$$

De maneira análoga ao que foi feito para a regra dos trapézios, podemos dividir o intervalo $[a,b]$ em n subintervalos e calcular a integral em cada um destes subintervalos.

$$I \approx \int_{x_1=a}^{x_3} f(x)\,dx + \int_{x_3}^{x_5} f(x)\,dx + \cdots + \int_{x_{n-1}}^{x_{n+1}=b} f(x)\,dx = \sum_{i=2,4,6,\ldots}^{n} \int_{x_{i-1}}^{x_{i+1}} f(x)\,dx$$

Aplicando a regra de Simpson temos:

$$I \approx \frac{h}{3}\left[f(a) + 4\sum_{i=2,4,6,\ldots}^{n} f(x_i) + 2\sum_{i=3,5,7,\ldots}^{n-1} f(x_i) + f(b)\right]$$

Note que como o método de Simpson é aplicado em dois intervalos adjacentes de uma só vez, o intervalo de integração deve ser dividido em um número par de subintervalos para a utilização deste método.

Atividade 4

1. Escreva uma função no Scilab para calcular por meio do método "Simpson 1/3" a integral de dados tabulados. Os argumentos da função devem ser os vetores x e y dos dados.

2. Utilizar a função acima para calcular a integral nos pontos da tabela apresentada na atividade anterior.

3. Reescreva a função que você criou para o cálculo do valor do integral de uma função. Os argumentos da função devem ser: a própria função a ser integrada, os limites de integração e

o número de sub-intervalos. Utilize a função recém criada para o cálculo das integrais abaixo:

$$\int_0^1 e^{-x^2}\,dx \tag{4}$$

$$\int_0^1 e^{-x}\sin(x)\,dx \tag{5}$$