Instituto Tecnológico de Aeronáutica – ITA
Matemática Computacional – CCI-22
Laboratório 6 – Spline Cúbica

Professor: Marcos Ricardo Omena de Albuquerque Maximo

22 de abril de 2019

1 Tarefas

1.1 Implementação

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado):

1. `yq = InterpolacaoLinear(x, y, xq)`: realiza interpolação linear. Os vetores coluna $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ e $\mathbf{y} = [y_0, y_1, ..., y_n]^T$ são os $n+1$ nós de interpolação. O vetor coluna $\mathbf{x}_q$ contém os valores para os quais se deseja calcular a interpolação. A saída da função é o vetor coluna $\mathbf{y}_q$ tal que $\mathbf{y}_q(j) = p(\mathbf{x}_q(j))$, em que p é a função gerada pelas funções afins construídas para cada segmento a partir dos $n+1$ nós de interpolação.

2. `x = SolucaoTridiagonal(a, b, c, d)`: utiliza o algoritmo de Thomas, cuja complexidade é $\mathcal{O}(n)$, para resolver um sistema tridiagonal na seguinte forma:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, i = 1, 2, ..., n \tag{1}$$

Em que $a_1 = c_n = 0$. Ou em representação matricial:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & \cdots & 0 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{2}$$

As entradas da função são os vetores coluna $\mathbf{a} = [a_1, a_2, ..., a_n]^T$, $\mathbf{b} = [b_1, b_2, ..., b_n]^T$, $\mathbf{c} = [c_1, c_2, ..., c_n]^T$ e $\mathbf{d} = [d_1, d_2, ..., d_n]^T$, enquanto a saída é o vetor coluna $\mathbf{x} = [x_1, x_2, ..., x_n]^T$.

3. `[a, b, c, d] = SistemaSplineCubica(x, y)`: monta o seguinte sistema tridiagonal associado ao método Spline Cúbica:

$$\begin{bmatrix} 2h_{12} & h_2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ h_2 & 2h_{23} & h_3 & 0 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2h_{34} & h_4 & 0 & \cdots & 0 & 0 \\ 0 & 0 & h_4 & 2h_{45} & h_5 & \cdots & 0 & 0 \\ 0 & 0 & 0 & h_5 & 2h_{56} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 2h_{n-2,n-1} & h_{n-1} \\ 0 & 0 & 0 & 0 & 0 & \cdots & h_{n-1} & 2h_{n-1,n} \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \vdots \\ \Phi_{n-2} \\ \Phi_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{3}$$

Em que:

$$h_i = x_i - x_{i-1}, i = 1, 2, 3, ..., n \tag{4}$$

$$h_{ij} = h_i + h_j \tag{5}$$

$$v_i = 6\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right), i = 1, 2, 3, ..., n-1 \tag{6}$$

As entradas são os vetores coluna $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ e $\mathbf{y} = [y_0, y_1, ..., y_n]^T$, que representam os $n+1$ nós de interpolação. Já as saídas são os vetores coluna $\mathbf{a} = [a_1, a_2, ..., a_{n-1}]^T$, $\mathbf{b} = [b_1, b_2, ..., b_{n-1}]^T$, $\mathbf{c} = [c_1, c_2, ..., c_{n-1}]^T$ e $\mathbf{d} = [d_1, d_2, ..., d_{n-1}]^T$, que representam o sistema tridiagonal que se deseja construir na notação requerida pela função `SolucaoTridiagonal`.

4. `yq = InterpolacaoSplineCubica(x, y, xq)`: realiza interpolação usando o método Spline Cúbica. Os vetores coluna $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ e $\mathbf{y} = [y_0, y_1, ..., y_n]^T$ são os $n+1$ nós de interpolação. O vetor coluna $\mathbf{x}_q$ contém os valores para os quais se deseja calcular a interpolação. A saída da função é o vetor coluna $\mathbf{y}_q$ tal que $\mathbf{y}_q(j) = p(\mathbf{x}_q(j))$, em que p é a função gerada pelas funções splines construídas para cada segmento a partir dos $n+1$ nós de interpolação.

1.2 Análise

O script `ComparacaoLinearSpline` foi fornecido juntamente com o roteiro do laboratório e realiza uma comparação entre interpolação linear e interpolação por Spline Cúbica usando a função de Runge:

$$f(x) = \frac{1}{1+25x^2} \tag{7}$$

Analogamente ao script `FenomenoRunge`, fornecido no laboratório anterior, o script fornecido neste laboratório primeiramente considera $n+1$ pontos equidistantes no intervalo $[-1, 1]$ e realiza interpolação. Isto é feito para $n \in \{2, 4, 8, 16\}$ e para ambos os métodos implementados neste laboratório. Dois gráficos são traçados: um com os resultados da interpolação linear e um com os da interpolação por Spline Cúbica. Observe os gráficos e compare-os. Discuta se, nestes casos, a qualidade da interpolação melhora ou piora com o aumento de n. Faça ainda uma comparação com o que foi observado no laboratório anterior, especialmente em relação ao efeito conhecido como Fenômeno de Runge.

Ainda semelhante ao laboratório anterior, considere $E(n) = \max_{x\in[-1,1]} |f(x) - p(x)|$, a função que mede o máximo desvio de $p(x)$ (função resultante da interpolação) em relação a $f(x)$ no intervalo $[-1, 1]$. O script plota num mesmo gráfico como varia $E(n)$ com n para $n \in \{1, 2, ..., 50\}$, considerando ambos os métodos de interpolação. Para o cálculo, considera-se $\max_{x\in[-1,1]} |f(x) - p(x)| \approx \max_{x\in\{-1:h:1\}} |f(x) - p(x)|$ com h pequeno. Comente o resultado obtido.

Note que o script `ComparacaoLinearSpline` plota os gráficos pedidos neste item desde que todas funções tenham sido implementadas.

2 Instruções

- A primeira etapa do processo de correção consistirá em submeter as funções implementadas a vários casos de teste de forma automatizada. Assim, os cabeçalhos das funções devem ser seguidos **rigorosamente**. Arquivos .m com os nomes destas funções e os cabeçalhos já implementados foram fornecidos juntamente com este roteiro. Dê preferência a implementar seu laboratório a partir destes arquivos .m fornecidos para evitar erros.

- **Não** é permitido o uso de funções ou comandos prontos do MATLAB que realizem toda a funcionalidade atribuída a uma certa função. Entretanto, o uso destas funções para verificação das implementações realizadas é encorajado. Em caso de dúvida quanto à permissão de uso de alguma função ou comando, recomenda-se consultar o professor.

- Não é necessário reimplementar métodos de laboratórios anteriores. Assim, funções implementadas em laboratórios anteriores podem ser utilizadas. Caso prefira, também é permitido utilizar funções equivalentes do MATLAB.

- Não é necessário se preocupar com verificação dos dados de entrada: assuma que x e y tem mesma dimensão, que vetores a, b, c e d estão na notação especificada no cabeçalho etc.

- Os arquivos .m implementados devem ser entregues juntamente com um relatório.

- No relatório, não é necessário demonstrar que as funções implementadas funcionam corretamente (isto será verificado separadamente). Basta incluir resultados e conclusões relativos à **Análise**.

- Para facilitar a correção da Análise, inclua os gráficos diretamente no relatório.

- A função `SolucaoTridiagonal` deve **necessariamente** utilizar o algoritmo de Thomas. **Não** é permitido utilizar outras técnicas de resolução de sistemas lineares.

3 Dicas:

- A função `diff` do MATLAB é útil para calcular h.

- A função `interp1` é muito útil para verificar a função `InterpolacaoLinear`:

  `yq = interp1(x, y, xq, ‘linear’);`

  Porém, o método de Spline Cúbica (`‘spline’`) da função `interp1` é ligeiramente diferente do método de Spline Cúbica deduzido em sala. Assim, prefira usar a função `csape` para verificar a função `InterpolacaoSplineCubica`:

  `pp = csape(x, y, ‘variational’);`

  `yq = ppval(pp, xq);`

- Para verificar a função `SistemaTridiagonal`, basta montar o sistema tridiagonal no formato $Ax = b$ e usar o comando `x = linsolve(A, b)`.

- Utilize as funções `SolucaoTridiagonal` e `SistemaSplineCubica` na implementação da função `InterpolacaoSplineCubica`.

- Cuidado com o formato requerido para vetores para evitar problemas na correção automática.

- Para criar gráficos com alta qualidade em formato PNG para inclusão em arquivos do Microsoft Word, utilize o comando: `print -dpng -r300 grafico.png`.

- Se utilizar LaTeX, dê preferência para incluir gráficos em formato vetorizado. No Linux, utilizando `pdflatex`, você pode gerar um gráfico em formato EPS usando “`print -depsc2 grafico.eps`” e depois convertê-lo para PDF usando o comando de terminal “`epstopdf grafico.eps`”. O arquivo PDF é aceito pelo `pdflatex`.