CCI-22 Trabalho 4
Interpolação

10 de abril de 2018

1 Notação

- escalares: letras minúsculas em itálico: n
- vetores: letras minúsculas em bold: $\mathbf{p}$
- matrizes: letras maiúsculas em bold: $\mathbf{T}$

2 Implementação

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado), valem 1 ponto cada:

2.1 `[p, cA] = gPolinomioInterpolador(x, y, maxCond)`

determina o polinômio interpolador $p_n(x)$ a partir de $n+1$ pontos de interpolação, definidos pelos vetores coluna $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ e $\mathbf{y} = [f(x_0), f(x_1), ..., f(x_n)]^T$. O vetor linha $\mathbf{p}$ é escrito na seguinte forma:

$$\mathbf{p} = [a_n, a_{n-1}, a_{n-2}, ..., a_0] \tag{1}$$

De modo a representar um polinômio interpolador $p_n(x)$ na forma:

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + ... + a_0 \tag{2}$$

1. Utilize a matriz de Vandermonde para calcular o polinômio interpolador.

2. Note que este formato de **p** é coerente com o esperado pela função `polyval`.

3. o parâmetro `maxCond` é um limite para mal condicionamento da matriz de Vandermonde. Para evitar mensagens de erro e soluções sem sentido, se `cond(A)` > `maxCond`, então não é necessário calcular o resultado. Código nesse sentido já foi entregue para os alunos.

4. maxCond é opcional, se omitido, é preciso definir um limite default muito grande. Código nesse sentido já foi entregue para os alunos.

5. A resposta `cA` é o valor de `cond(A)`. Onde A é a matriz de Vandermonde.

2.2 `T = TabelaDiferencasDivididas(x, y)`

calcula a Tabela de Diferenças Divididas **T** a partir de $n+1$ pontos de interpolação, definidos pelos vetores coluna

$\mathbf{x} = [x_0, x_1, ..., x_n]^T$ e $\mathbf{y} = [f(x_0), f(x_1), ..., f(x_n)]^T$. Adota-se o seguinte formato para **T**:

$$\mathbf{T} = \begin{bmatrix} f[x_0] & f[x_0,x_1] & f[x_0,x_1,x_2] & \cdots & f[x_0,x_1,\cdots,x_n] \\ f[x_1] & f[x_1,x_2] & f[x_1,x_2,x_3] & \cdots & \\ f[x_2] & f[x_2,x_3] & f[x_2,x_3,x_4] & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f[x_n] & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (3)$$

Note ainda que T tem dimensão $(n+1) \times (n+1)$.

2.3 `[yq,pn] = InterpolacaoFormaNewton(dd, x, xq)`

calcula os valores interpolados $\mathbf{yq} = p_n(\mathbf{xq})$ utilizando polinômio interpolador $p_n(x)$ na Forma de Newton. O vetor
$\mathbf{dd} = [f[x0], f[x0,x1], ..., f[x0,x1,...,xn]]$ contém os operadores de diferenças divididas necessários para o cálculo e $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ são os $n+1$ pontos de interpolação. **xq** é um vetor coluna (que pode ter apenas um elemento), e **yq** é um vetor coluna tal que $\mathbf{yq}(i) = p_n(\mathbf{xq}(i))$.

O vetor linha **pn** deve representar o polinômio interpolador, no mesmo formato definido na questão 2.1, e deve ser calculado usando as diferenças divididas e o método de Newton, e não por outros métodos.

2.4 `x = SolucaoTridiagonal(a, b, c, d)`

1. utiliza o algoritmo de Thomas, cuja complexidade é $\mathcal{O}(n)$, para resolver um sistema tridiagonal na seguinte forma:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, i = 1, 2, ..., n \tag{4}$$

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

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & \cdots & 0 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{5}$$

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 $\mathbf{x} = [x_1, x_2, ..., x_n]^T$.

2.5 `[a, b, c, d] = SistemaSplineCubica(x, y)`

monta o seguinte sistema tridiagonal associado ao método Spline Cúbica:

$$\begin{bmatrix} 2h_{12} & h_2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ h_2 & 2h_{23} & h_3 & 0 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2h_{34} & h_4 & 0 & \cdots & 0 & 0 \\ 0 & 0 & h_4 & 2h_{45} & h_5 & \cdots & 0 & 0 \\ 0 & 0 & 0 & h_5 & 2h_{56} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 2h_{n-2,n-1} & h_{n-1} \\ 0 & 0 & 0 & 0 & 0 & \cdots & h_{n-1} & 2h_{n-1,n} \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \vdots \\ \Phi_{n-2} \\ \Phi_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{6}$$

Em que:

$$h_i = x_i - x_{i-1}, i = 1, 2, 3, ..., n \tag{7}$$

$$h_{ij} = h_i + h_j \tag{8}$$

$$v_i = 6 \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right), i = 1, 2, 3, ..., n-1 \tag{9}$$

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`.

2.6 `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.

3 Análise

A tabela na última seção serve para concentrar e organizar as respostas númericas de todas as perguntas. A nota depende da resposta número e das respostas textuais, discussões e gráficos correspondentes, em conjunto. Por isso a tabela e os gráficos em si, sozinhos, não vale ponto. Vale apenas responder as perguntas com base em dados correspondentes às respostas.

3.1 Comparação e Fenômeno de Runge

Seja a função de Runge:

$$f(x) = \frac{1}{1 + 25x^2} \tag{10}$$

E a função :

$$g(x) = \exp(x)\sin(5x) \tag{11}$$

Considere ainda $n + 1$ pontos equidistantes no intervalo $[-1, 1]$. Pode-se obter o polinômio interpolador $p_n(x)$ a partir destes $n + 1$ pontos de interpolação. Mostre $p_n(x)$ para $n \in 1, 2, 4, 8, 16$ juntamente com $f(x)$ num único gráfico. Utilize as funções `PolinomioInterpolador`, `InterpolacaoFormaNewton` e `InterpolaçãoSplineCúbica` para realizar a interpolação.

Então, considere $E(n) = \max_{x \in [-1,1]} |f(x) - p_n(x)|$, a função que mede o máximo desvio de $p_n(x)$ em relação a $f(x)$ no intervalo $[-1, 1]$. Plote um gráfico de como varia $E(n)$ com $n \in 1, 2, ..., 20$ e determine o n que minimiza $E(n)$. Use como aproximação $\tilde{E}(n) \max_{x \in \{-1:h:1\}} |f(x) - p_n(x)|$ com h pequeno, ou seja, calcule a diferença entre as funções para uma série de pontos igualmente espaçados no intervalo, e suponha que a maior diferença encontrada aproxima a diferença máxima entre as funções.

Observe as curvas e resultados obtidos e discuta-os com base no Fenômeno de Runge. Considere:

- [1pt] Aumentar n sempre melhora a qualidade da interpolação? Analise os gráficos com a evolução do erro com o aumento de n .
- [1pt] Especialmente: preencha a coluna "converge p/" considerando, se ao chegar em $n = 50$, o erro $E(n)$:
  - parece estar convergindo para zero (preencha 0);
  - parece estar convergindo para uma constante (preencha com o valor da constante e considere qualquer coisa próxima do epsilon da máquina como zero);
  - parece estar aumentando ou variando rapidamente em valores altos (preencha com ∞. Isto seria um sinal claro do Fenômeno de Runge!);
  - parece estar começando a aumentar próximo ao 50, o que seria um sinal que: ou Fenômeno de Runge ainda existe, embora diminuído, ou existe algum outro problema, mas de qualquer forma não conseguimos discernir com clareza com n até 50. (preencha com "?", indicando cuidado!).

Note que a função `Runge` plota os gráficos pedidos neste item, desde que seja fornecida uma função capaz de realizar a interpolação com cabeçalho adequado. Como funções auxiliares, também são fornecidas funções que adequam o cabeçalho dos métodos implementados pelo aluno à função `Runge`.

ATENÇÃO: não olhe os gráficos superficialmente e diga que são iguais apenas porque ambos parecem ter a mesma forma se vistos de uma escala maior. Não diga que converge para zero só porque parece diminuir muito nos primeiros n. Verifique com zoom, e/ou conferindo os números das tabelas, e considere se as diferenças são ou não significativas e consistentes.

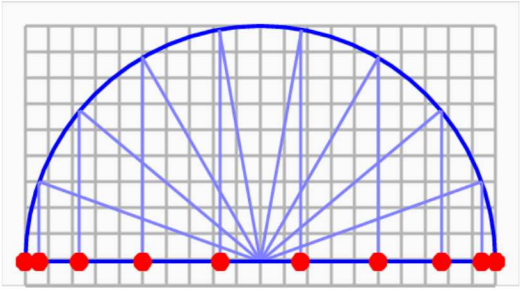

Figura 1: Nodes de Chebyshev

3.2 Nós de Chebyshev

Ao invés de utilizar nós de interpolação igualmente espaçados, podemos utilizar os nós de chebyshev. Dado um intervalo no eixo das abcissas e um semicírculo cujo diâmetro corresponde ao intervalo, as abcissas dos nodes de chebyshev são obtidas tomando-se um espaçamento regular no semicírculo e projetando os pontos do semicírculo para a reta abcissa, conforme a figura 1.

No intervalo $\left[\begin{array}{cc} -1 & 1 \end{array}\right]$, os nós de Chebyshev são:

$$x_j = \cos\left(j\pi/N\right),\ j = 0, 1, \ldots, N$$

Em um intervalo geral $\left[\begin{array}{cc} a & b \end{array}\right]$, os nós de Chebyshev são:

$$x_j = \frac{a+b}{2} + \frac{\pi\left(b-a\right)}{2}\cos\left(\frac{2j+1}{2N+2}\right),\ j = 0, 1, \ldots, N$$

Foi fornecida uma função que calcula os nós de Chebyshev dado o intervalo e o número de nós. Repita o exercício anterior utilizando os nós de Chebyshev, inclua medidas dos novos erros e analise a diferença:

- [0.5] O resultado eh melhor para todos os métodos de interpolação por polinomio (sem contar spline), comparando com nós de chebyshev com nós regulares? Considere os valores dos erros em si mesmos, mas especialmente a existência e gravidade do Fenômeno de Runge.

- [0.5] Considerando apenas os métodos de interpolação por polinomio (sem contar spline), há diferenças visíveis no resultado entre um método e outro? Porque deve ser assim?

- [0.25] Quanto à spline, o que acontece? Porque? Dica: olhe o desenvolvimento da prova, o que foi suposto que não é mais válido?

- [0.5] Comparando interpolação por polinômio único versus splines: Quais as vantagens e desvantagens? Inclua comparação do resultado da spline com nós regulares versus o resultado da interpolação com nós de chebyshev.
- [0.25] Há alguma desvantagem em se utilizar nós de chebyshev - não é uma pergunta teórica, na prática, qual a dificuldade possível?

3.3 Tabela de Resultados

função de Runge	nós	$\min E(n)$	n para $\min E(n)$	converge para
PolinomioInterpolador	regular			
InterpolacaoNewton	regular			
SplineCubica	regular			
PolinomioInterpolador	Chebyshev			
InterpolacaoNewton	Chebyshev			
SplineCubica	Chebyshev			

função g(x)	nós	$\min E(n)$	n para $\min E(n)$	converge para
PolinomioInterpolador	regular			
InterpolacaoNewton	regular			
SplineCubica	regular			
PolinomioInterpolador	Chebyshev			
InterpolacaoNewton	Chebyshev			
SplineCubica	Chebyshev			

3.4 Gráficos

Inclua os seguintes gráficos para a funcao de runge.

PolinomioInterpolador com nós regulares
InterpolacaoNewton com nós regulares
SplineCubica com nós regulares
PolinomioInterpolador com nós Chebyshev
InterpolacaoNewton com nós Chebyshev

4 Instruções

- 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. Por exemplo, na implementação da função `PolinomioInterpolador`, é necessário resolver um sistema linear de equações, de modo que é permitido usar diretamente a função `linsolve` do MATLAB.

- Não é necessário se preocupar com verificação dos dados de entrada: assuma que x e y tem mesma dimensão, que dd esteja conforme formato especificado no cabeçalho da função, etc.

- Antes de entregar o seu código, certifique-se de que ele não imprime nenhum texto na tela, apenas retorna o resultado correto para passar nos testes. Acrescente ; ao final das linhas de código para evitar impressões.

Não *pode usar a função* `vander` *para calcular a matriz de Vandermonde.* Primeiro, porque resolve boa parte do problema diretamente. Segundo, pois assim há a oportunidade de ser esperto e construir a matriz de Vandermonde em $O(n^2)$ e não em $O(n^3)$. Será $O(n^3)$ (ou no mínimo $O(n^2 \log n)$ dependendo da exponenciação) se para cada elemento fizer a exponenciação diretamente.

5 Dicas:

- Após ter obtido o polinômio interpolador com `PolinomioInterpolador`, utilize `polyval` para calcular os valores de $p_n(x)$ desejados.

- `fliplr()` inverte um vetor: o último elemento passa a ser o primeiro, e vice-versa. Isso é útil para converter o vetor que representa o polinômio de uma convenção onde x_n é o primeiro elemento (nosso código

e `polyval`) e outra convenção onde x_n é o último elemento, que foi utilizada nos slides.

- Para a implementação de algumas das funções, pode ser conveniente utilizar operações elemento a elemento de matrizes (também para vetores): `a .* b`, `1 ./ a`, `a.^2` ,etc. Ou um dos operadores é um escalar, ou as matrizes (vetores) de ambos operadores tem o mesmo tamanho. A notação matemática para produto elemento a elemento de matrizes, ou produto de Hadamard, ou produto de Schur, é $A \circ B$, definido como $(A \circ B)_{ij} = a_{ij} \cdot b_{ij}$

- Perceba que o vetor dd passado como parâmetro para `InterpolacaoFormaNewton` é a primeira linha da tabela $\mathbf{T}$ retornada por TabelaDiferencasDivididas, i.e. $\mathbf{dd} = \mathbf{T}(1, :)$

- É muito simples calcular uma sequencia de pontos igualmente espaçados em um intervalo dado, mas há uma função que faz isso: `linspace(a,b,n)` gera `n` pontos igualmente espaçados entre `a` e `b`.

- A função `diff` é muito apropriada para calcular diferenças divididas. Também pode fazer `v(2:end) - v(1:end-1)`

- No matlab para definir a função g, use `g = @(x) (exp(x).*sin(5.*x))`. Note o operador de Hadamard product ‘‘`.*`”, é necessário para permitir que a função funcione com um vetor inteiro de uma vez, o que por sua vez é necessário para otimizar a função Runge.

- Se o método de interpolação é baseado em resolver um sistema linear de equações, à medida que o numero de equações aumenta, a matriz pode ficar mal condicionada e prejudicar a solução. Se calcular `cond(A)`, onde A é a matriz de coeficientes do sistema linear, pode-se verificar se isso acontece. Se um sistema é tridiagonal, é mais fácil ou mais dificil de se tornar mal condicionado quando o tamanho aumenta?

5.1 Multiplicar polinômio

Dados 2 polinomios representados por vetores de coeficientes `p1` e `p2`, na forma e ordem que a funcao polyval aceita. O polinomio `pm` igual a multiplicacao de `p1` e `p2` pode ser calculado por `pm = conv(p1,p2)` Onde `pm` tem a mesma representacao como polinomio.

Eh um tanto mais fácil do que fazer um loop duplo multiplicando cada termo por todos os termos do outro polinomio. Nao sei se voces sabem o que

é convolução (se nao sabem saberão, principalmente o povo da ELE), mas pode olhar no help do matlab pra saber.

E forneci uma função sumpoly pra somar polinomios. Notem que ela automaticamente ajusta os tamanhos dos polinomios.

5.2 convenções de spline[1]

Podemos verificar os resultados das nossas implementações tambem com as implementações prontas do matlab. A função `interp1` possui interpolação linear (linhas entre pontos), nearest (toma o ponto mais próximo, a escadinha), e inclusive faz spline. Por exemplo, para interpolação linear:

```
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.

Na dedução do spline que fizemos em sala, nós forçamos como condições extras que as segundas derivadas nos extremos fossem nulas, completando as últimas 2 equações para o sistema ficar determinado. No caso, `interp1` não força esta condição, mas sim que o spline seja de classe C^3 (i.e. terceira derivada contínua) no primeiro e no último pontos de quebra interiores. Assim, o spline resultante é ligeramente diferente do determinado conforme a formulação de CCI22.

Descobri também a função csape, que é uma função de spline cúbica mais avançada que permite que você escolha qual é a condição extra usada para restringir os dois graus de liberdade faltantes do spline. Usando ela com condição `'variational'`, o resultado obtido coincide com o resultado de `InterpolacaoSplineCubica`. Vocês podem então usar esta função `csape` para verificar o lab de vocês com uma precisão maior. Exemplo de uso da csape:

```
pp = csape(x, y, 'variational');
yq = ppval(pp, xq);
```

Interessante: `doc csape` lista 6 diferentes convenções que o `csape` implementa para as últimas equações necessárias para a spiline cúbica.

Outra coisa interessante é a variavel `pp` acima. Ela é um vetor de structs onde cada elemento contém um intervalo e um polinomio. Assim, `pp` representa todo o resultado da spline! E `ppval` é uma função pronta que reconhece este formato!

[1] a aluna Ana Paula e o prof. Máximo descobriram estes detalhes.

5.3 Erros comuns

Max x abs: a ordem importa... compare e note que são diferentes:

```
max(abs( xnovo - xvelho ) )
abs(max( xnovo - xvelho ) )
```