MÉTODOS NUMÉRICOS - LISTA DE EXERCÍCIOS III

SÉRGIO CORDEIRO

Sumário

1. Problemas

1. Sobre SVD: apresente modelagem matemática para calculo; apresente aplicações práticas; escolha uma aplicação associada à compressão de matrizes e faça a decomposição, a compressão e a analise o número de condicionamento antes e depois da compressão. Comente e analise todos os resultados.

Os **valores singulares** são uma generalização dos autovalores, pois aplicam-se não apenas a matrizes quadradas. A decomposição SVD de uma matriz $\mathbf{A}$ consiste na sua substituição pelo produto de três outras matrizes: $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{(T)}$, onde $\mathbf{U}$ e $\mathbf{V}$ são unitárias e $\mathbf{S}$ é diagonal. Pode-se demonstrar que toda matriz pode ser decomposta dessa forma, e que a decomposição é única, a menos da ordem das colunas das matrizes.
A generalização dos autovetores são os **vetores singulares**. As colunas de $\mathbf{U}$ são chamadas de vetores singulares à esquerda, e as de $\mathbf{U}$, de vetores singulares à direita.
A aplicação imediata da decomposição SVD é a solução mais fácil de um sistema linear, pois $\mathbf{U}$ e $\mathbf{V}$ são matrizes unitárias, por isso suas transpostas são as inversas; S, por sua vez, é diagonal, e encontrar sua inversa é trivial. Portanto:

$$\begin{aligned} \mathbf{A}\mathbf{x} = \mathbf{b} \implies \qquad \mathbf{x} &= \mathbf{A}^{-1}\mathbf{b} \\ &= \left(\mathbf{U}\mathbf{S}\mathbf{V}^{(T)}\right)^{-1}\mathbf{b} \\ &= \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^{(T)}\mathbf{b} \end{aligned}$$

Uma segunda aplicação é a compressão da matriz, que pode ser expressa pelo produto $\hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^{(T)}$, com as matrizes $\hat{\mathbf{U}}$, $\hat{\mathbf{S}}$ e $\hat{\mathbf{V}}^{(T)}$ derivadas das originais por eliminação de colunas menos significativas, correspondentes a valores singulares de menor valor absoluto.
Uma terceira aplicação é a diminuição do número de condicionamento da matriz, através da eliminação dos valores singulares com menor valor absoluto. A melhoria no condicionamento torna a matriz mais adequada ao trataemnto por métodos iterativos.
Finalmente, uma quarta aplicação é no cálculo da **pseudoinversa** de uma matriz, que é definida como:

$$\mathbf{A}^{\dagger} = \mathbf{A}A^{T}(\mathbf{A}\mathbf{A}^{T})^{-1} \text{ ou}$$

$$\mathbf{A}^{\dagger} = (\mathbf{A}A^{T}\mathbf{A})^{-1}\mathbf{A}^{T}$$

dependendo de qual produto, $\mathbf{A}\mathbf{A}^T$ ou $\mathbf{A}A^T\mathbf{A}$, for invertível. A pseudoinversa é dada por:

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{S}^\dagger\mathbf{U}^T$$

cujo cálculo é fácil, uma vez que S é diagnonal [FASSHAUER 2006, PRESS 1992 1, VETTERLI 2014].

A compressão de matrizes por meio da decomposição SVD usa a técnica conhecida como *low-rank approximation*: após encontrarem-se os n autovalores, selecionam-se os m maiores; todas linhas de U e V são mantidas, mas apenas as m colunas que correspondem aos maiores autovalores dessas matrizes; quanto a S, são selecionadas tanto as linhas quanto as colunas que correspondem aos autovalores mais importantes.
O resultado é uma matriz C que tem o mesmo tamanho da matriz original, mas apenas m autovalores.
A manutenção dos maiores autovalores faz com que a maior parte da variância original seja preservada, e com ela, a informação relevante. O número de condição, por sua vez, é bastante melhorado.

A aplicação escolhida foi a compressão de imagens em preto e branco, com e sem introdução de ruído. As imagens originais foram obtidas no banco de dados do Departamento de Engenharia Elétrica e de Computação do Instituto Politécnico da Universidade de Nova York.
O programa **exercmat.c**, em anexo, escrito em C, lê uma matriz em disco e comprime-a pelo método de decomposição SVD, calculando o número de condicionamento antes e depois da compressão. Basta digitar:

exercmat 20 *n*

onde n é o tamanho da matriz (n x n). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa. O próprio programa tenta descobrir qual é a maior taxa de compressão possível sem que resulte perda expressiva de dados.
O algoritmo é totalmente genérico, e pode ser aplicado diretamente a uma matriz que representa uma imagem. Para este exercício, experimentou-se com alguns valores diferentes para a taxa de compressão.
O algoritmo usa o método de Jacobi para encontrar os autovalores e autovetores da matriz $\mathbf{A}^{(T)}\mathbf{A}$, o que não é uma solução ótima.
Os resultados obtidos são mostrados e tabelados a seguir. Foi usada apenas precisão simples, de forma a favorecer a ocorrência de erros de arredondamento.

n		nome	Número de condição		iterações
antes	depois		antes	depois	
512	347 240 149 71	Lena	38.146271	1.860093 1.495879 1.273939 1.142361	406
512	336 232 145 68	Bárbara	23.812378	1.879261 1.471599 1.275514 1.138802	348

n		nome	custo [1]	
antes	depois		decomposição	compressão
512	347 240 149 71	Lena	35047344300288	304347657 184041993 100176393 41789961
512	336 232 145 68	Bárbara	35047344300288	290898441 175989257 96879113 39793161

Imagem original: Lena com 512 valores singulares

[1]Número de operações de ponto flutuante necessárias.

Imagem original: Bárbara com 512 valores singulares

Imagem comprimida: Lena com 347 valores singulares

Imagem comprimida: Bárbara com 336 valores singulares

Imagem comprimida: Lena com 240 valores singulares

Imagem comprimida: Bárbara com 232 valores singulares

Imagem comprimida: Lena com 149 valores singulares

Imagem comprimida: Bárbara com 145 valores singulares

Imagem comprimida: Lena com 71 valores singulares

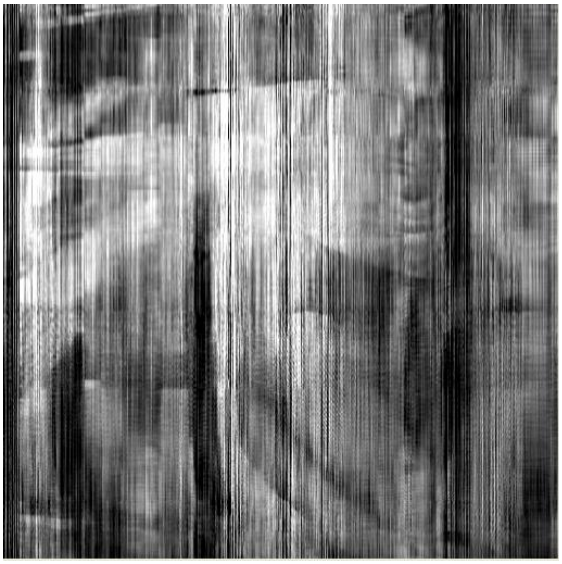
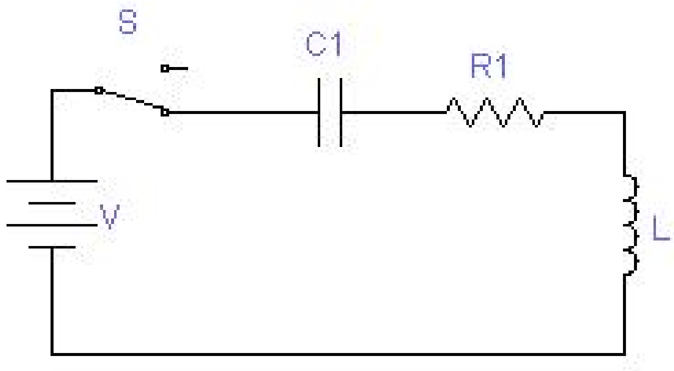


Imagem comprimida: Bárbara com 68 valores singulares

Para a pronunciada compressão obtida, as imagens mostram que a informação importante foi preservada.

2. Simule (utilizando qualquer software) o circuito da figura 1 e obtenha a tabela de valores (t (seg) e I (A)). Resolva analiticamente o circuito. Faça o ajuste da função utilizando regressão (avalie a que melhor se adeque a esse problema). Plote em um gráfico os valores simulados, calculados (solução analítica) e os valores obtidos a partir da regressão. Escolha os valores para os elementos de forma que a resposta seja oscilatória. Comente os resultados.

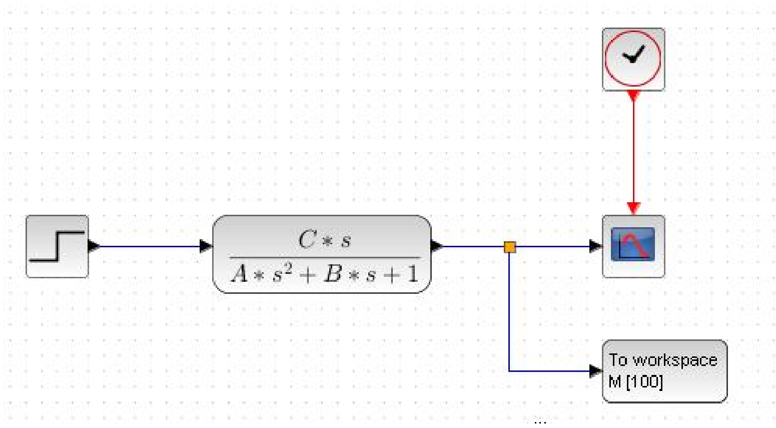


A função de transferência $\mathcal{G}(s) = \frac{\mathcal{I}(s)}{\mathcal{V}(s)}$ correspondente é:

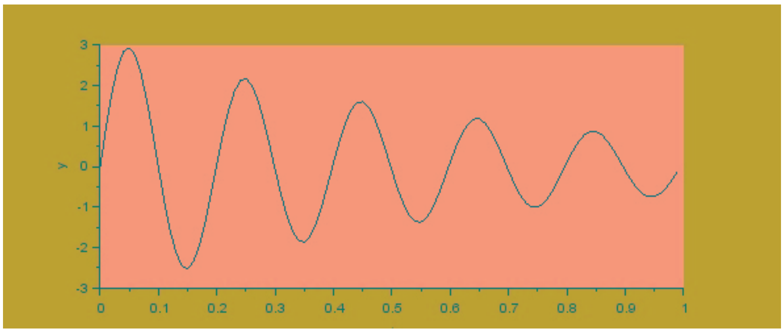
$$\mathcal{G}(s) = \frac{1}{R + Ls + \frac{1}{Cs}}$$

$$= \frac{Cs}{LCs^2 + RCs + 1} \qquad = \frac{Cs}{As^2 + Bs + 1} \qquad \boxed{LC = A, RC = B}$$

O diagrama de simulação correspondente é o seguinte:



e a saída simulada é a seguinte, para $A = 0.001$, $B = 0.003$ e $C = 0.1$:



Evidentemente, $C = 0.1\ F$, $L = \frac{A}{C} = \frac{0.001}{0.1} = 0.01\ H$ e $R = \frac{B}{C} = \frac{0.003}{0.1} = 0.03\ \Omega$, que não são valores muito realísticos, mas servem como ilustração.
As raízes da equação característica são:

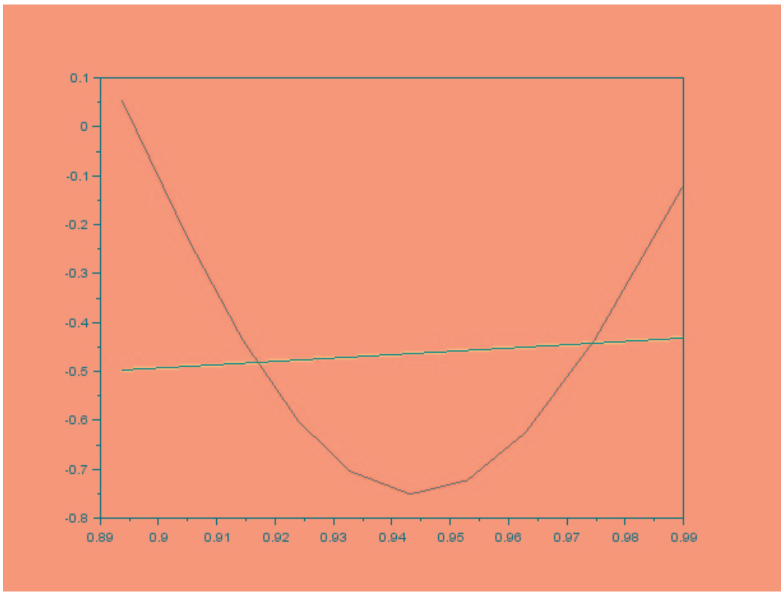
$$A\lambda^2 + B\lambda + 1 = 0 \implies \lambda = \frac{-B \pm \sqrt{B^2 - 4A}}{2A}$$
$$= \frac{-0.003 \pm \sqrt{0.003^2 - 4 \times 0.001}}{2 \times 0.001}$$
$$= \frac{-0.003 \pm \jmath 0.0632}{0.002}$$
$$= -1.5 \pm \jmath 31.6$$

Assim, a forma da resposta será $y = De^{-1.5t}\sin(31.6t + E)$. Como $y(0) = 0$, então $E = 0$. O programa **`exercmat.c`**, em anexo, escrito em C, lê uma matriz em disco e ajusta a ela um polinômio por regressão polinomial, usando o método dos mínimos quadrados. Basta digitar:

```
exercmat 32 n
```

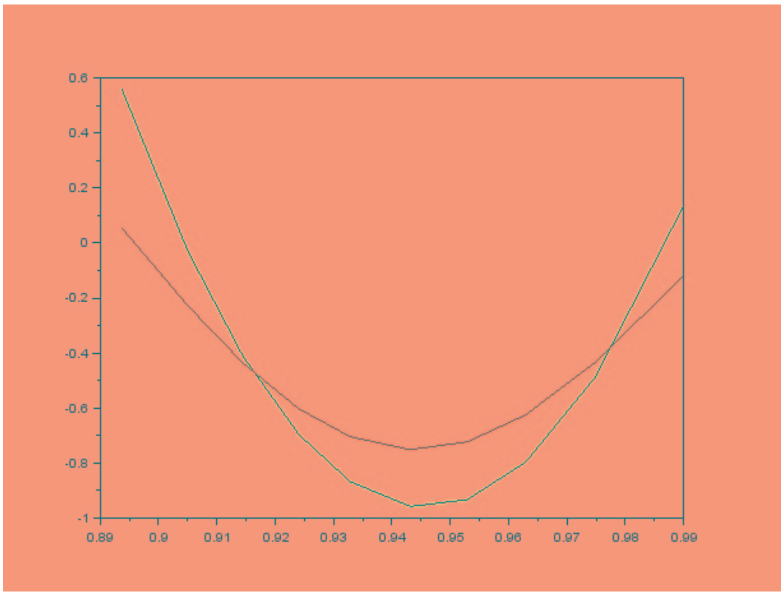
onde n é o tamanho do problema. Ele também calcula a qualidade da aproximação e o número de operações em ponto flutuante necessário para cada etapa.
Para ajustar um polinômio aos pontos disponíveis, selecionaram-se as amostras correspondentes aos 100 valores anteriores a $x = 1$ e tentaram-se graus crescentes para o polinômio. Os resultados obtidos são mostrados na tabela abaixo e ilustrados pelos gráficos que se seguem. Os gráficos foram plotados em baixa definição, usando-se apenas 10 pontos, de forma a não incorrer em custo de processamento alto.

grau	**coeficiente de determinação**	**variãncia residual**	**custo** [2]	
			ajuste	**avaliação**
1	0.006412	0.065045	739	1670
2	0.282749	0.047438	1578	2709
3	0.991840	0.000545	2632	3963
4	0.983997	0.001081	3905	5436
5	0.989866	0.000692	5401	7132

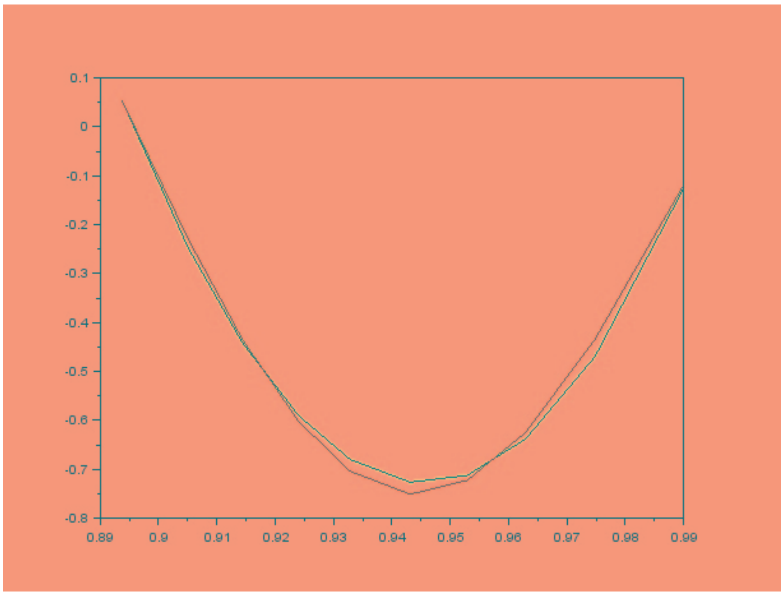


$\alpha_1 = 0.680600,\ \alpha_0 = -1.105716$

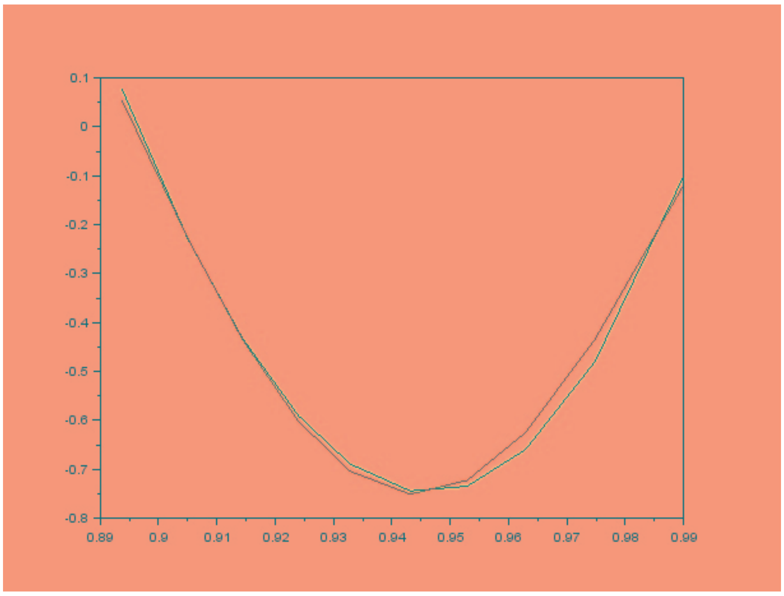
[2]Número de operações de ponto flutuante necessárias.



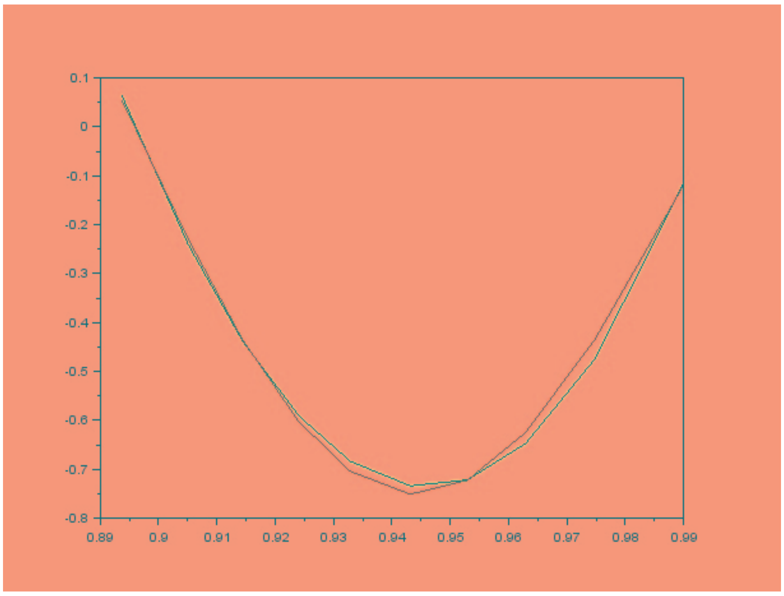
$\alpha_2 = 560.891174$, $\alpha_1 = -1060.951416$, $\alpha_0 = 500.748230$



$\alpha_3 = 156.376862$, $\alpha_2 = -144.615555$,
$\alpha_1 = -145.984055$, $\alpha_0 = 134.404083$



$\alpha_4 = 196.042389$, $\alpha_3 = -263.384857$, $\alpha_2 = 15.143293$,
$\alpha_1 = 14.279124$, $\alpha_0 = 38.165676$



$\alpha_5 = 23.204826$, $\alpha_4 = 13.646919$, $\alpha_3 = 43.347961$,
$\alpha_2 = -84.813011$, $\alpha_1 = -95.061539$, $\alpha_0 = 99.884033$

As curvas mostram que não há melhoria apreciável quando o grau do polinômio é aumentado a partir de 3. Os índices da qualidade do ajuste inclusive indicam que a melhor aproximação é obtida com um polinômio do terceiro grau. Evidentemente, se outro trecho da curva tivesse sido escolhido.

3. Para os Tópicos: Interpolação de Hermite, Interpolação com *Spline* Cúbico e Extrapolação faça:
- Descrição de um problema real;
- Modelagem matemática e numérica (código em anexo);
- Resultados e conclusões

I) Interpolação de Hermite:
A interpolação de Hermite utiliza n pontos (x, y) da função e n *pontos* (x, y') de sua derivada. Permite o uso de um polinômio de grau muito menor, $\frac{n-1}{2}$ em lugar de $n-1$, o que, entre outras coisas, implica em menor oscilação do resultado [3]. Os coeficientes são obtidos a partir da fórmula para os coeficientes de Lagrange. Evidentemente, seu emprego se limita aos casos em que os pontos da derivada são conhecidos.
O programa **exercmat.c**, em anexo, escrito em C, lê uma tabela em disco e interpola um ponto pelo método de Hermite. Basta digitar:

exercmat 21 *n*

onde n é o tamanho da tabela (n x 3). Ele também calcula o número de operações em ponto flutuante necessário.
Um caso prático, trabalhado por [TOBÓN 2011], é a aproximação da função $y = \ln(x)$ por meio de um polinômio; em todos os pontos se conhece a derivada $y' = \frac{1}{x}$. Neste caso, foram usadas tabela com 6, 10 e 20 entradas, para $1 \leq x \leq 5.5$, resultando em polinômios de grau crescente. O ponto interpolado foi $x = 3.1$; o valor exato é $y = 1.131402$. Os resultados obtidos estão mostrados abaixo. O valor obtido pela interpolação pelos métodos de Lagrange e de Neville também são mostrados, para comparação. A interpolação de Lagrange pode ser obtida digitando-se:

exercmat 19 *n*

e a interpolação de Neville é obtida digitando-se:

exercmat 24 *n*

As técnicas de Lagrange e de Neville não consideram o valor da derivada.

[3] O chamado *Fenômeno de Runge.*

n	**Hermite**	**Lagrange**	**Neville**	**custo** [4]
3	1.31482	1.138113	1.124153	81
5	1.31399	1.130888	1.131802	215
10	1.31400	1.131401	1.131402	830

O número ideal de entradas é 10, neste caso, 5 para y e 5 para y', que corresponde a um polinômio interpolador de grau 9. Mais pontos não melhoram muito a precisão e aumentam bastante o custo computacional. A técnica se mostrou superior às interpolações de Lagrange e de Neville para o mesmo número de pontos.
Uma grande desvantagem da técnica é que cada ponto a ser interpolado exige o recálculo de todos os coeficientes [FREITAS 2010, TOBÓN 2011].

II) *Spline* cúbico:
O problema anterior pode ser resolvido de forma alternativa por meio de *spline* cúbico. A técnica consiste em não ajustar um mesmo polinômio a todos os pontos conhecidos, e sim ajustar polinômios de grau 3 diferentes a grupos de 4 pontos consecutivios. Condições extras são introduzidas de forma a evitar descontinuidades nas derivadas primeiras e segundas. Condições adicionais devem ser impostas às derivadas primeira e segunda nos pontos extremos:

1. $y''(x_1) = y''(x_1) = 0$ (a chamda condição de ***spline* natural**), ou
2. $y''(x_1) = y''(x_2)$ e $y''(x_{n-1}) = y''(x_n)$, ou
3. $y''(x_1)$ e $y''(x_n)$ são obtidos por extrapolação linear, ou
4. $y'(x_1) =$ A e $y'(x_{n-1}) =$ B

O programa **`exercmat.c`**, em anexo, escrito em C, lê uma tabela em disco e interpola um ponto por tal método, considerando as derivadas ou não. Basta digitar:

```
exercmat 22 n
```

onde n é o tamanho da tabela. Ele também calcula o número de operações em ponto flutuante necessário. A condição de *spline* natural é sempre imposta.
Verifica-se que a consideração da derivada melhora bastante a precisão e diminui o custo computacional. A exatidão, contudo, é inferior à obtida com um polinômio de mais alto grau. A técnica do *spline*, além de ser mais simples, ainda permite o reaproveitamento dos coeficientes para interpolação de outros pontos [KOKKOTAS 2015, KUCKIR 2014, ONEILL 2002, OUYED 2011, PRESS 1992 2].

[4] Número de operações de ponto flutuante necessárias.

n	**derivadas**	**resultado**	**custo** [5]
5	Não	1.119492	127
10	Não	1.120845	317
5	Sim	1.31414	20
10	Sim	1.31404	20

III) Extrapolação:
As técnicas de interpolação podem também, a princípio, ser usadas para extrapolação, contanto que o ponto extrapolado não esteja muito distante do intervalo onde os pontos são conhecidos. A técnica do *spline* cúbico natural, devido às condições extras impostas aos pontos extremos, não se presta a esse uso.
O programa **`exercmat.c`**, em anexo, escrito em C, lê uma tabela em disco e extrapola um ponto por quatro métodos: o de Lagrange, o de Neville, o de Hermite e o *spline* cúbico. Basta digitar:

```
exercmat 23 n
```

onde n é o tamanho da tabela. Ele também calcula o número de operações em ponto flutuante necessário. Os resultados são mostrados abaixo:

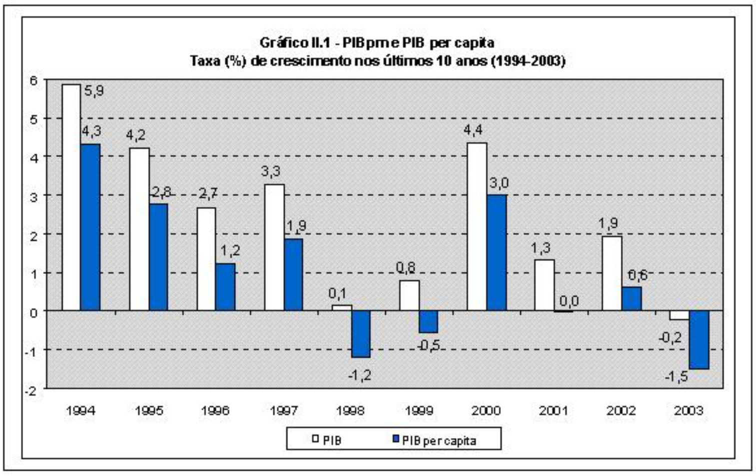
n	**valor exato**	**Lagrange**	**Hermite**	**Neville**
5	1.629240	1.626772	1.629260	1.769480
10	1.722767	1.722921	1.712949	1.713927

A tabela confirma a tendência de a técnica de Hermite ser a mais precisa e mais cara, e a de Neville, a menos precisa e mais barata.

---

[5] Número de operações de ponto flutuante necessárias.

4. O gráfico a seguir apresenta as variações do PIB e PIB/percapita.



Para esses dados (PIB e PIB per capita) faça:

1. Encontre o polinômio interpolador que melhor represente essa função (PI de maior grau possível) (Use interpolação);
2. Plote o diagrama de dispersão e o PI encontrado;
3. Comente todos os resultados

O programa **`exercmat.c`**, em anexo, escrito em C, lê uma tabela em disco e encontra o polinômio interpolador com o maior grau possível. Basta digitar:

`exercmat 18` ***`n`***

onde n é o tamanho da tabela (n x 2). Ele também calcula o número de operações em ponto flutuante necessário para cada etapa.
Os coeficientes encontrados para o PIB foram:
$a_9 = 5.900000$, $a_8 = 24.808952$, $a_7 = -60.370991$, $a_6 = 49.870708$, $a_5 = -19.498470$, $a_4 = 3.804901$, $a_3 = -0.311178$, $a_2 = -0.005871$, $a_1 = 0.002509$, $a_0 = -0.000111$
A expressão do polinômio interpolador é:
$y = \sum_{i=0}^{9} a_i x^i$
com $x = ano - 1994$. Os coeficientes encontrados para o PIB per capita foram:
$a_9 = 4.300000$, $a_8 = 26.969584$, $a_7 = -64.837807$, $a_6 = 53.848827$, $a_5 = -21.452248$, $a_4 = 4.385204$, $a_3 = -0.417683$, $a_2 = 0.005891$, $a_1 = 0.001795$, $a_0 = -0.000093$.

2. ANEXOS

Os seguintes arquivos constam do anexo (arquivo **`exercmat1.zip`**):

- arquivo fonte em C **`exercmat.c`**
- arquivos de dados:
  - An: problema 1
  - En: problema 2
  - Bn: problemas 3 e 4

REFERÊNCIAS

[FASSHAUER 2006] Greg FASSHAUER, **Numerical Linear Algebra/Computational Mathematics I**: Illinois Institute of Technology, 2006. Disponível em http://www.math.iit.edu/~fass/477577_Chapter_2.pdf, acesso em 25/03/2016.

[FREITAS 2010] Pedro Garcia FREITAS, **3.1.4 Aproximação de Funções — Interpolação — Interpolação de Hermite**. Disponível em http://www.sawp.com.br/blog/?p=880, acesso em 04/04/2016.

[KOKKOTAS 2015] Kostas KOKKOTAS, **Interpolation, Extrapolation ans Polynomial Approximation**. Disponível em http://www.tat.physik.uni-tuebingen.de/~kokkotas/Teaching/Num_Methods_files/Comp_Phys3.pdf, acesso em 17/04/2016.

[KUCKIR 2014] Ivan KUCKIR, **Interpolation with Cubic Splines**. Disponível em http://blog.ivank.net/interpolation-with-cubic-splines.html, acesso em 05/04/2016.

[ONEILL 2002] Charles O'NEILL, **Cubic Spline Interpolation MAE 5093**. Disponível em http://charles-oneill.com/projects/cubicspline.pdf, acesso em 04/04/2016.

[OUYED 2011] Rachid OUYED e Woflgang DOBLER, **Interpolation, Extrapolation ans Polynomial Approximation**. Disponível em Chap.4,2011,pp.53a55, acesso em http://pjl.ucalgary.ca/courses/physics381/computational-physics/Ouyed-Chapter-4-Interpolation-Extrapolation-Techniques.pdf.17/04/2016

[PRESS 1992 1] William H. PRESS, Saul A. TEUKOLSKY, William T. VETTERLING and Brian P. FLANNERY, **Numerical recipes in Fortran 77: The art of scientific computing Vol. 1**: Cambridge University Press, 2nd. Ed., 1992, ISBN 0-521-43064-X, Chap. 3. pp. 51 a 63. Disponível em http://www.fing.edu.uy/if/cursos/fiscomp/extras/numrec/book/f3.pdf, acesso em 06/04/2016.

[PRESS 1992 2] William H. PRESS, Saul A. TEUKOLSKY, William T. VETTERLING and Brian P. FLANNERY, *op. cit.*, Chap. 2. pp. 99 a 122.

[TOBÓN 2011] Luis E. TOBÓN, **Newton, Lagrange and Hermite Interpolation: Convergence and Runge phenomena**. Disponível em http://people.duke.edu/~let12/pdf/courses/math225/LuisTobon_HW1_Interpolation.pdf, acesso em 04/04/2016.

[VETTERLI 2014] M. VETTERLI, J. KOVACEVIC e V. K. GOYAL, **Foundations of Signal Processing**, Cambridge University Press, March 2014, ISBN 110703860X [free version], pp. 59 a 60 e 146 a 147.30/04/2016

Programas testados com **Octave** 4.0.0, **Scilab** 5.5.2 e **MinGW** C 4.8.2:
https://www.scilab.org
https://www.gnu.org/software/octave/
https://www.mingw.org
Texto formatado com **pdflatex** em ambiente **MiKTeX** 2.9:
http://miktex.org/download/