MÉTODOS NUMÉRICOS - LISTA DE EXERCÍCIOS III

SÉRGIO CORDEIRO

SUMÁRIO

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.

A decomposição SVD de uma matriz $\mathbf{A}$ consiste na sua substituição pelo produto de três outras matrizes: $\mathbf{A} = \mathbf{USV}^{(T)}$, 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 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; $\mathbf{S}$, por sua vez, é diagonal, e encontrar sua inversa é trivial. Portanto:

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

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 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 $\mathbf{U}$ e $\mathbf{V}$ são mantidas, mas apenas as m colunas que correspondem aos maiores autovalores dessas matrizes; quanto a $\mathbf{S}$, são selecionadas tanto as linhas quanto as colunas que correspondem aos autovalores mais importantes. O resultado é uma matriz $\mathbf{C}$ 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**		**custo** [1]		**iterações**
antes	**depois**		**antes**	**depois**	**decomp.**	**¨compr.**	
512	347	Lena	38.146271	1.860093	35047344300288	304347657	406
	240			1.495879		184041993	
	149			1.273939		100176393	
	71			1.142361		41789961	
512	336	Bárbara	23.812378	1.879261	35047344300288	290898441	348
	232			1.471599		175989257	
	145			1.275514		96879113	
	68			1.138802		39793161	

Imagem original: Lena com 512 valores singulares

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

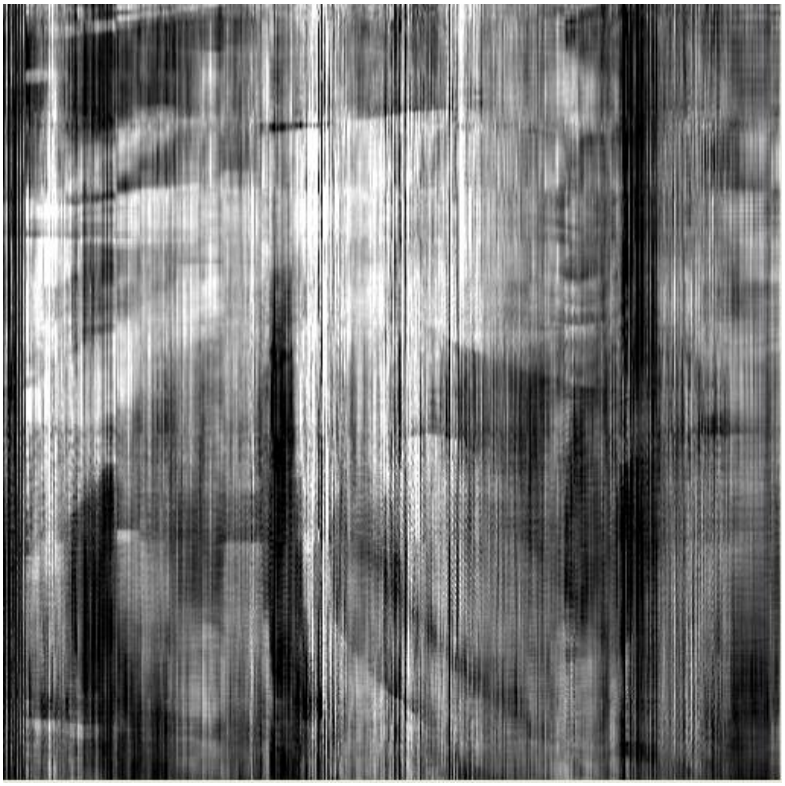
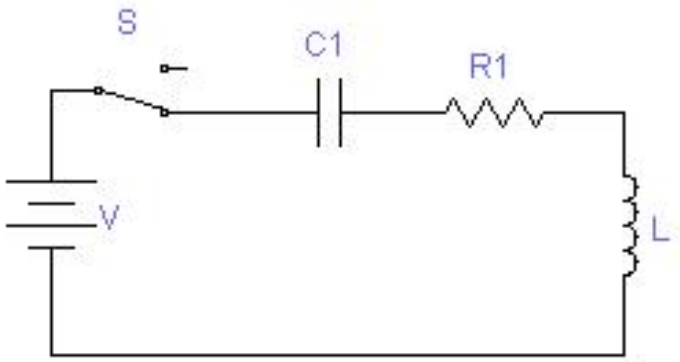


Imagem comprimida: Bárbara com 68 valores singulares

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.

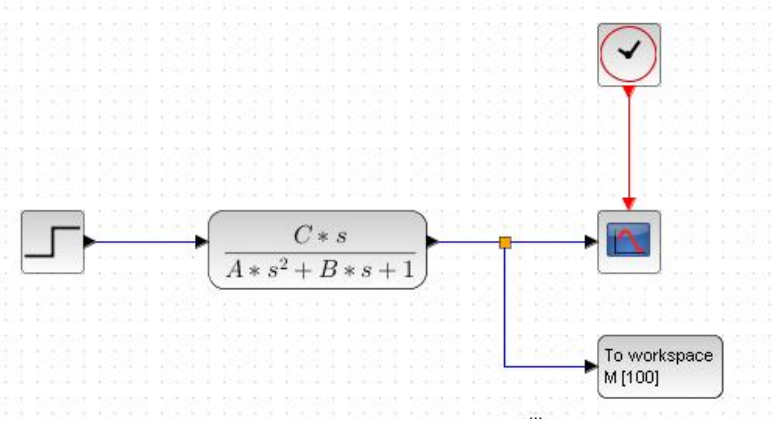


A função de transferência $\mathcal{G}(s) = \frac{I(s)}{\mathcal{V}(s)}$ correspondente é:

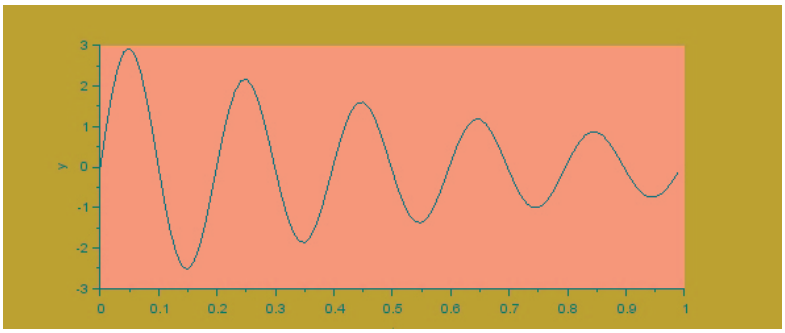
$$\mathcal{G}(s) = \frac{1}{R + Ls + \frac{1}{Cs}}$$

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

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 $E = 0$.

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:
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 de Lagrange é também mostrado, para comparação. A interpolação de Lagrange pode ser obtida digitando-se:

```
exercmat 19 n
```

A técnica de Lagrange não considera o valor da derivada.

n	**Hermite**	**Lagrange**	**custo** [2]
3	1.31482	1.138113	81
5	1.31399	1.130888	215
10	1.31400	1.131401	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 à interpolação de Lagrange para o mesmo número de pontos.
Uma grande desvantagem da técnica é que cada ponto a ser interpolado exige o recálculo de todos as grandezas [FREITAS 2010, TOBÓN 2011].

II) *Spline* cúbico:
O problema anterior pode ser resolvido de forma alternativa por meio de *spline* cúbico. 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. 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 [KUCKIR 2014, ONEILL 2002].

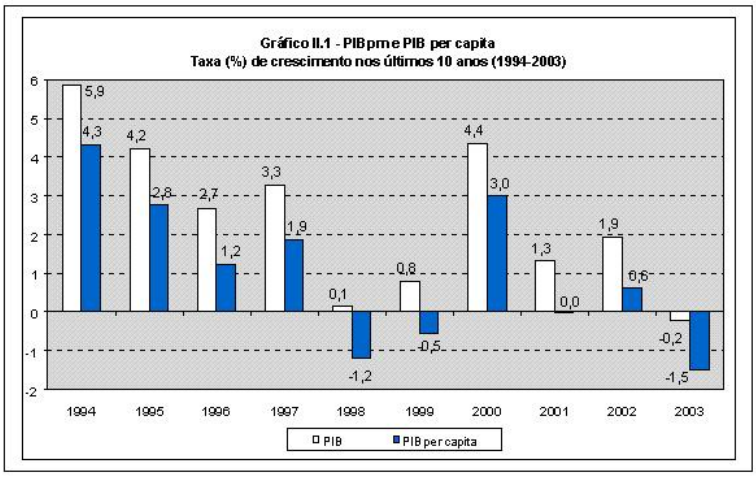
n	**derivadas**	**resultado**	**custo** [3]
5	Não	1.119492	127
10	Não	1.120845	317
5	Sim	1.31414	20
10	Sim	1.31404	20

III) Extrapolação:
O problema anterior pode também ser resolvido por meio de extrapolação. O programa **exercmat.c**, em anexo, escrito em C, lê uma tabela em disco e extrapola um ponto por dois métodos: a extrapolação polinomial normal e a extrapolação de Richardson. Basta digitar:

exercmat 23 *n*

onde n é o tamanho da tabela. Ele também calcula o número de operações em ponto flutuante necessário.

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:
$a_9 = 4.300000$, $a_8 = 26.969584$, $a_7 = -64.837807$, $a_6 = 53.848827$, $a_5 = -21.452248$, $a_4 = 4.385204$, $a_3 = -0.417683$, $a_2 = 0.005891$, $a_1 = 0.001795$, $a_0 = -0.000093$.

1. ANEXOS

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

- arquivo fonte em C **`exercmat.c`**
- arquivos fontes em MATLAB:
  - **`gerachol.m`**: problema 2
  - **`geradsis.m`**: problema 5
  - **`geraL.m`**: problema 11
- arquivos de dados:
  - S*n*: problemas 1, 3 e 4
  - C*n*: problemas 2, 7 e 8
  - D*n*: problemas 5 e 6

REFERÊNCIAS

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

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

[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.

[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.

[WEISSTEIN 2016] Eric W. WEISSTEIN, **Positive Definite Matrix *MathWorld***. Disponível em http://mathworld.wolfram.com/PositiveDefiniteMatrix.html, acesso em 17/03/2016.

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