GMA
DEPARTAMENTO DE MATEMÁTICA APLICADA

Aplicações de Álgebra Linear

Lista de Exercícios

Humberto José Bortolossi
http://www.professores.uff.br/hjbortol/

04

Ainda sobre Matrizes e Aplicações

[01] Efetue a multiplicação $\mathbf{A} \cdot \mathbf{B}$ das duas matrizes $\mathbf{A}$ e $\mathbf{B}$ dadas abaixo usando (1) o método da multiplicação por blocos e (2) o produto convencional de matrizes.

$$\mathbf{A} = \left[\begin{array}{cc|cccc} 1 & 2 & 1 & -1 & 0 & 1 \\ \hline -1 & 1 & 1 & 0 & -1 & 1 \\ 0 & 1 & -1 & 1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 2 & 1 & 0 \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{ccc|cc} 1 & 0 & 1 & 2 & 1 \\ -1 & 1 & 2 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 & 2 \\ -1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & -1 & 1 \end{array}\right].$$

[02] Seja

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

onde $\mathbf{B}$ e $\mathbf{C}$ são matrizes $n \times n$, $\mathbf{0}$ é a matriz nula $n \times n$ e $\mathbf{I}$ é a matriz identidade $n \times n$. Mostre que se $\mathbf{B} - \mathbf{I}$ for não singular (isto é, se $\mathbf{B} - \mathbf{I}$ for inversível), então, para todo $k \geq 1$,

$$\mathbf{A}^k = \begin{bmatrix} \mathbf{B}^k & (\mathbf{B}^k - \mathbf{I})(\mathbf{B} - \mathbf{I}^{-1})\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

[03] Seja $\mathbf{A}$ uma matriz $n \times n$ inversível, e sejam $\mathbf{u}$ e $\mathbf{v}$ dois vetores em $\mathbb{R}^n$. Encontre condições necessárias e suficientes para que a matriz

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{u} \\ \mathbf{v}^T & 0 \end{bmatrix}$$

seja inversível, e apresente uma fórmula para a inversa quando ela existir.

[04] Usando a rotina `MATLAB` que desenvolvemos em sala de aula, calcule o polinômio de Lagrange de grau ≤ 4 que interpola os pontos:

$$(0.00,\ \ 1.00), \quad (0.25,\ \ 1.00), \quad (0.50,\ \ 0.00), \quad (0.75,\ \ 2.00), \quad (1.00,\ \ 0.00).$$

[05] Dependendo dos valores de (x_i, y_i), a matriz de Vandermonde associada à interpolação de Lagrange pode ficar mal-condicionada. Veja o `MATLAB` reclamar deste fato usando o código abaixo para diferentes valores de `n`.

```
n = 10; x = [1:(3 - 1)/(n - 1):3]; y = 10*rand(1, n);
[A, c] = my_polyfit(x, y);
```

Por este motivo, obter o polinômio interpolador de Lagrange através da resolução de um sistema linear via matriz de Vandermonde não é recomendado. Outros métodos numéricos existem (forma de Newton com diferenças divididas).

[06] Sejam $x_0 < x_1 < x_2 < \cdots < x_n$ números reais. Defina os $n+1$ polinômios de grau n:

$$p_i(x) = \frac{\displaystyle\prod_{\substack{k=0\\k\neq i}}^{n}(x - x_k)}{\displaystyle\prod_{\substack{k=0\\k\neq i}}^{n}(x_i - x_k)} = \frac{(x-x_0)\cdots(x-x_{i-1})\cdot(x-x_{i+1})\cdot(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})\cdot(x_i-x_{i+1})\cdot(x_i-x_n)}, \qquad \text{para } i = 0, \ldots, n.$$

Suponha que $n = 3$, $x_0 = 1$, $x_1 = 2$, $x_2 = 3$ e $x_3 = 4$.

(a) Calcule $p_0(x)$, $p_1(x)$, $p_2(x)$ e $p_3(x)$.

(b) Mostre que $\{p_0, p_1, p_2, p_3\}$ forma uma base para o espaço vetorial $V = \mathcal{P}_3(\mathbb{R})$ das funções polinomiais reais de grau ≤ 3. Dica:

$$p_i(x_j) = \delta_{ij} = \begin{cases} 0, & \text{se } i \neq j, \\ 1, & \text{se } i = j. \end{cases}$$

(c) Calcule as coordenadas de $q(x) = 2 + 3\,x + 4\,x^2 + 5\,x^3$ nesta base.

(d) Mostre que o item (b) vale para um $n \geq 1$ qualquer e conclua que

$$p(x) = \sum_{i=0}^{n} y_i p_i(x) = \sum_{i=0}^{n} y_i \frac{\displaystyle\prod_{\substack{k=0\\k\neq i}}^{n}(x - x_k)}{\displaystyle\prod_{\substack{k=0\\k\neq i}}^{n}(x_i - x_k)}$$

é uma expressão para o polinômio de Lagrange que interpola os pontos $(x_0, y_0), \ldots, (x_n, y_n)$.

[07] Usando o código `MATLAB` que desenvolvemos para o método de Newton em sala de aula, calcule uma aproximação do sistema não-linear abaixo. Use `x0 = [-1; 4]` como valor inicial.

$$\begin{cases} 1 + x^2 - y^2 + e^x \cos(y) = 0, \\ 2\,xy + e^x \operatorname{sen}(y) = 0. \end{cases}$$

[08] Usando o código `MATLAB` que desenvolvemos para o método de Newton em sala de aula, calcule uma aproximação do sistema não-linear abaixo. Use `x0 = [1; 1; 1]` como valor inicial.

$$\begin{cases} xy - z^2 = 1, \\ xyz - x^2 + y^2 = 1, \\ e^x - e^y + z = 3. \end{cases}$$

O que acontece, se você iniciar o algoritmo com `x0 = [0; 0; 1]`? Explique!

[09] Considere a matriz

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 2 & 0 & 5 \\ 3 & 0 & 4 & 0 \\ 0 & 5 & 0 & 6 \end{bmatrix}.$$

Use o comando `chol(A)` do `MATLAB` para determinar se **A** possui uma decomposição de Cholesky e, assim, verificar se **A** é positiva definida. Você também pode usar a rotina `LU_naive(A)` que calcula a decomposição **LU** de **A** sem pivotamento.

[10] Determine se a matriz **A** definida pelo código `MATLAB` abaixo é positiva definida.

```
B = fix(5*rand(3, 3)); A = B + B' - ones(3,3)
```

Respostas dos Exercícios

[02] Note que

$$\mathbf{A}^k = \begin{bmatrix} \mathbf{B}^k & \mathbf{B}^{k-1}\mathbf{C} + \cdots + \mathbf{B}^1\mathbf{C} + \mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^k & (\mathbf{B}^{k-1} + \cdots + \mathbf{B}^1 + \mathbf{I})\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

Agora, como $(\mathbf{B}^{k-1} + \cdots + \mathbf{B}^1 + \mathbf{I})(\mathbf{B} - \mathbf{I}) = \mathbf{B}^k - \mathbf{I}$ e, por hipótese, $\mathbf{B} - \mathbf{I}$ é inversível, vemos então que

$$\mathbf{B}^{k-1} + \cdots + \mathbf{B}^1 + \mathbf{I} = (\mathbf{B}^k - \mathbf{I})(\mathbf{B} - \mathbf{I})^{-1}.$$

Desta maneira,

$$A^k = \begin{bmatrix} \mathbf{B}^k & (\mathbf{B}^k - \mathbf{I})(\mathbf{B} - \mathbf{I}^{-1})\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

[03] Para que a matriz $\widetilde{\mathbf{A}}$ seja inversível, é necessário e suficiente que sua última coluna não seja combinação linear das demais colunas. Como $\mathbf{A}$ é inversível, existe um único $\mathbf{x} \in \mathbb{R}^n$ tal que $\mathbf{u} = \mathbf{A}\mathbf{x}$. De fato, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{u}$. Assim, para que a última coluna de $\widetilde{\mathbf{A}}$ não seja combinação linear das demais, é necessário e suficiente que a combinação dos elementos de $\mathbf{v}^T$ com coeficientes dados pelas entradas de $\mathbf{x} = \mathbf{A}^{-1}\mathbf{u}$ sejam diferentes do número 0, isto é,

$$\mathbf{v}^T\mathbf{x} = \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u} \neq 0.$$

Suponha agora que $\widetilde{\mathbf{A}}$ seja inversível ($\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u} \neq 0$). Vamos calcular uma fórmula para a inversa de $\widetilde{\mathbf{A}}$. Escrevendo

$$\begin{bmatrix} \mathbf{A} & \mathbf{u} \\ \mathbf{v}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{c} \\ \mathbf{d}^T & e \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$$

concluímos que

$$\mathbf{A}\mathbf{B} + \mathbf{u}\mathbf{d}^T = \mathbf{I}, \quad \mathbf{A}\mathbf{c} + e\mathbf{u} = \mathbf{0}, \quad \mathbf{v}^T\mathbf{B} = \mathbf{0}^T, \quad \mathbf{v}^T\mathbf{c} = 1.$$

Da segunda equação, segue-se que $\mathbf{c} = -e\mathbf{A}^{-1}\mathbf{u}$. Substituindo este valor na última equação, obtemos que $\mathbf{v}^T(e\mathbf{A}^{-1}\mathbf{u}) = -1$. Logo

$$e = -\frac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}} \qquad \text{e} \qquad \mathbf{c} = \frac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\mathbf{A}^{-1}\mathbf{u}.$$

Da primeira equação temos que $\mathbf{B} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{u}\mathbf{d}^T$. Substituindo este valor na terceira equação, obtemos que $\mathbf{v}^T\mathbf{A}^{-1} - \mathbf{v}^T(\mathbf{A}^{-1}\mathbf{u}\mathbf{d}^T) = \mathbf{v}^T\mathbf{A}^{-1} - (\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u})\mathbf{d}^T = \mathbf{0}^T$. Logo,

$$\mathbf{d}^T = \frac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\mathbf{v}^T\mathbf{A}^{-1} \qquad \text{e} \qquad \mathbf{B} = \mathbf{A}^{-1} - \frac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}.$$

Portanto,

$$\begin{bmatrix} \mathbf{A} & \mathbf{u} \\ \mathbf{v}^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} - \dfrac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1} & \dfrac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\mathbf{A}^{-1}\mathbf{u} \\ \dfrac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\mathbf{v}^T\mathbf{A}^{-1} & -\dfrac{1}{\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}} \end{bmatrix}.$$

Texto composto em LaTeX2e, HJB, 12/01/2009.

Apêndice: código MATLAB para a decomposição LU sem pivotamento

```
function A=LU_naive(A)
[m,n]=size(A);
A = sym(A);
disp('Begin');
for k=1:min(m, n)-1
    for i=k+1:m
        p = sym(-A(i,k)/A(k,k));
        display(['L' num2str(i) '<- L' num2str(i) ' + ...
                 (' char(p) ')*L' num2str(k)]);
        for j=k:n
           A(i, j) = A(i, j) + p*A(k, j); % Be careful:
                                          % the "-" sign is built in
                                          % the definition of p
        end
        disp(A)
        answer = input('Continue? (y/n) ', 's');
        if (answer == 'n')
           break;
        end
    end
end
disp('End');

function A=LU_golub_kij(A)
[m,n]=size(A);
A = sym(A);
disp('Begin');
for k=1:min(m, n)-1
    for i=k+1:m
        A(i,k)=A(i,k)/A(k,k);
        for j=k+1:n
           A(i, j) = A(i, j) - A(i,k)*A(k, j);
        end
    end
end
disp(A);
disp('End');
```

Apêndice: código MATLAB para a interpolação de Lagrange

```
% Example:
% n = 10; x = [1:(3 - 1)/(n - 1):3]; y = 10*rand(1, n);
% [A, c] = my_polyfit(x, y);
%
% http://www-hm.ma.tum.de/archiv/in1/ws0102/links/Interpol/Lagrange.html
function [A, c] = my_polyfit(x, y)
[m n] = size(x);
A = zeros(n, n);
for k=1:n
    A(:,k)=x'.^(k - 1);
end
c = A\y';
xb = x(1, 1);
xe = x(1, n);
xp = xb:(xe - xb)/100:xe;
yp = polyval((c(n:-1:1))', xp);
plot(x, y, 'o', xp, yp, '-')
```

Apêndice: código MATLAB para o método de Newton

```
function FF=newton_ff(x)
FF(1,1) = x(1)^2 + x(2)^2 - 1;
FF(2,1) = sin(pi*x(1)/2) + x(2)^3;
return

function DF=newton_df(x)
pi2 = 0.5* pi;
DF(1,1) = 2*x(1);
DF(1,2) = 2*x(2);
DF(2,1) = pi2*cos(pi2*x(1));
DF(2,2) = 3*x(2)^2;
return

% Example:
%
% x0 =[1;1]; tol=1e-5; maxiter =10;
% [x,F,iter] = newton_sm(@newton_ff, @newton_df, x0, tol, maxiter);
%
function [x,R,iter] = newton_sm(newton_ff, newton_df, x0, ...
                                tol, nmax, varargin)
iter = 0; err = tol + 1; x = x0;
while err > tol & iter <= nmax
    FF = feval(newton_ff, x, varargin {:});
    DF = feval(newton_df, x, varargin {:});
    delta = - DF\FF;
    x = x + delta;
    err = norm(delta);
    iter = iter + 1;
end
R = norm(feval(newton_ff, x, varargin {:}));
if iter >= nmax
    fprintf('Too many iterations. Residual: %e.\n', R);
else
    fprintf('Success with %i step(s). Residual: %e.\n', iter, R);
end
return
```