

# 1 Lista de Exercícios de MAC 315 - Primeiro semestre de 2002

Leônidas de Oliveira Brandão

## 1.1 Exercícios “teóricos”

**Exercício 1.1** Sejam  $B := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|^2 \leq r^2\}$ ,  $\mathbf{1} := (1, 1, \dots, 1)' = \sum_{i=1}^n \mathbf{e}_i$  e  $H := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{1}'\mathbf{x} \leq r\sqrt{n}\}$ . Prove que  $B \subset H$  (use desenho no  $\mathbb{R}^2$  para ilustrar sua prova).

**Exercício 1.2** Sejam  $X$  e  $I$  tais que:  $\mathbf{b} \in C(\{\mathbf{a}^i\}_{i \in I})$ . Prove que: para todo  $B \subseteq \{1, 2, \dots, n\}$ , tal que  $\{\mathbf{a}^i\}_{i \in B}$  é l.i.,  $\#B = m$  e  $I \subseteq B$ , então  $\langle x_i = 0, \forall i \notin I, \mathbf{x} \text{ gerado por } B \rangle$ .

**Exercício 1.3** Coloque na forma canônica os seguintes poliedros (mostre cada passo da transformação):

$$1. \begin{cases} \max & 2x_1 - x_2 + x_3 \\ \text{s.a.} & 2x_1 + x_2 - 2x_3 \leq 8 \\ & 4x_1 - x_2 + 2x_3 \geq 2 \\ & 2x_1 + 3x_2 - x_3 \geq 4 \\ & \mathbf{x} \geq \mathbf{0}. \end{cases}$$

$$2. \begin{cases} \max & 5x_1 - 2x_2 + x_3 \\ \text{s.a.} & 2x_1 + 4x_2 + x_3 \leq 6 \\ & 2x_1 + x_2 + 3x_3 \geq 2 \\ & (x_1; x_2) \geq \mathbf{0} \\ & x_3 \in \mathbb{R}. \end{cases}$$

**Exercício 1.4** Sejam  $V := \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$  l.i.,  $\mathbf{A} := [\mathbf{a}_1 | \mathbf{a}_2 | \dots | \mathbf{a}_n]$  e  $\mathbf{B} := \mathbf{A}^{-1}$ . Para algum  $\mathbf{b} \in \mathbb{R}^n$ , se  $\{\mathbf{a}_j\}_{j=1}^n \cup \{\mathbf{b}\}$  é l.i., mostre como obter  $\mathbf{C}$  a inversa de  $\mathbf{D} := [\mathbf{a}_1 | \mathbf{a}_2 | \dots | \mathbf{a}_{n-1} | \mathbf{b}]$  em  $O(n^2)$  operações elementares.

**Exercício 1.5** Para  $\mathbf{A} \in \mathbb{R}^{m \times n}$  e  $\mathbf{b} \in \mathbb{R}^n$ , mostre que o sistema abaixo não pode ter solução  $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m$ ,

$$\mathbf{A}\mathbf{x} \leq \mathbf{0} \tag{1}$$

$$\mathbf{A}'\mathbf{y} = \mathbf{b} \tag{2}$$

$$\mathbf{b}'\mathbf{x} > 0 \tag{3}$$

$$\mathbf{y} \geq \mathbf{0} \tag{4}$$

justificando cada passagem empregada (use os rótulos das equações para isso).

**Exercício 1.6** Dê uma interpretação geométrica para o resultado obtido no exercício acima (utilize um desenho em  $\mathbb{R}^2$  para facilitar sua explicação).

**Exercício 1.7** Prove ou ache um contra-exemplo para a seguinte implicação: para  $\mathbf{A} \in \mathbb{R}^{m \times n}$  e  $\mathbf{b} \in \mathbb{R}^m$ ,

$$\left. \begin{matrix} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} > \mathbf{0} \end{matrix} \right\} \implies \mathbf{b}'\mathbf{A} > \mathbf{0}.$$

**Exercício 1.8** Seja  $\mathcal{S}$  o conjunto das matrizes quadradas tais que  $\mathbf{A} \in \mathcal{S} \iff \begin{cases} a_{ij} > 0 & \Leftrightarrow i = j, \text{ e} \\ i < j & \Rightarrow a_{ij} = 0. \end{cases}$

Prove que  $\mathbf{A} \in \mathcal{S}$  implica que  $\mathbf{A}$  é não singular e que  $|\mathbf{A}| > 0$ .

**Dica:** Utilize indução na dimensão  $n$  das matrizes  $\mathbf{A}$ .

## 1.2 Exercícios no Scilab

Entregar um arquivo de nome `seu_nome_com_no_máximo_10_letras.tar`, “targado”, contendo:

- um arquivo `todos.txt` no formato ASCII com todos os exercícios abaixo, separados por linha em branco e cabeçalho (`\\ Exercício N`);
- um arquivo de nome `ex-X.sci`, para cada exercício (onde `X` é o número do exercício).

**Atenção**, também deve ser entregue uma listagem do arquivo `todos.txt` (com seu nome na primeira linha).

**Exercício 1.1** Uma TRANSFORMAÇÃO ELEMENTAR sobre uma matriz  $\mathbf{A}$  equivale a somar numa linha (coluna) de  $\mathbf{A}$  o produto de um escalar por outra linha (coluna). Uma matriz  $\mathbf{B}$  que realiza uma transformação elementar ( $\mathbf{BA}$ ) é dita de TRANSFORMAÇÃO ELEMENTAR. Podemos dizer também que uma matriz que seja o produto de transformações elementares é de TRANSFORMAÇÃO.

Utilizando o programa Scilab no apêndice A.2 (estude-o para entender sua sintaxe) e a matriz “default” no mesmo, construa (deduza) as três matrizes de transformação  $\mathbf{T}_1$ ,  $\mathbf{T}_2$  e  $\mathbf{T}_3$  que, respectivamente, “pivota” a coluna 1 (os elementos abaixo) por  $b_{11}$ , coluna 2 por  $b_{22}$  e coluna 3 por  $b_{33}$  ( $\mathbf{B}$  são as matrizes obtidas em cada iteração do algoritmo).

Entregue um código Scilab, definindo as matrizes  $\mathbf{T}_1$ ,  $\mathbf{T}_2$  e  $\mathbf{T}_3$  e imprimindo  $|\mathbf{T}_1|$ ,  $|\mathbf{T}_2|$ ,  $|\mathbf{T}_3|$ ,  $|\mathbf{A}|$  e  $|\mathbf{B}|$  e também o produto  $\mathbf{T}_3 * \mathbf{T}_2 * \mathbf{T}_1 * \mathbf{A}$ .

**Exercício 1.2** Construa um programa Scilab que realize a operação do exercício 1.1 (obtendo as correspondentes  $\mathbf{D}$  e  $\mathbf{C}$ ) e, ao final, imprima as matrizes  $\mathbf{C}$  e  $\mathbf{C} * \mathbf{D}$ . Use o exemplo do apêndice A.2 como modelo, notando que uma operação efetuada numa linha de matriz (como  $\mathbf{B}(i,:) = \mathbf{B}(i,+)/\mathbf{B}(i,i)$ ) gasta da  $O(n)$  operações.

**Exercício 1.3** Teste o programa de seu exercício acima para as matrizes abaixo, aplicando seu algoritmo para obter  $\mathbf{C}$  e, ao final, imprima as matrizes  $\mathbf{C}$  e  $\mathbf{C} * \mathbf{D}$ .

$$(\mathbf{A}, \mathbf{b}) := \left( \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \\ 3 & 0 & 2 & 0 & 3 \\ 0 & 2 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right), \quad (\mathbf{A}, \mathbf{b}) := \left( \begin{bmatrix} 2 & -1 & 1 & 1 & 0 \\ -1 & 2 & 0 & 2 & 0 \\ 2 & 4 & -1 & 1 & 1 \\ 2 & 0 & 1 & 0 & 3 \\ 0 & 3 & 0 & 1 & 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right),$$

$$(\mathbf{A}, \mathbf{b}) := \left( \begin{bmatrix} 1 & -1 & 1 & 1 & 0 \\ -1 & 2 & 0 & 2 & -1 \\ 1 & 1 & -1 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ -1 \end{bmatrix} \right).$$

## A Exemplos de programas no Scilab

### A.1 Atribuições, recorrências e instâncias

```
function []=nada()
//
// MAC 315 - 2000 - Scilab - Prof. Leônidas
// Exemplo: funções contadoras, métodos recursivos e iterativos
//
// -----
// Algumas dicas básicas sobre o Scilab (mais detalhes consulte o "Help")
//
// A primeira linha é necessária, pois uma função Scilab NÃO pode começar
// com comentários, precisa do 'function'
//
// Para definir variáveis 'a' e 'b' com os valores '1' e '2.5': a=1, b=2.5
// Para definir variáveis 'a' e 'b' sem "echo" na tela:      a=1; b=2.5;
// Para ver todas as variáveis e funções já definidas, digite:      who
// Para "limpar" o Scilab, removendo todas as variáveis e funções:  clear
// Para remover apenas a variável 'a':                             clear a
// Para "rodar" este exemplo, pegue-o na página de MAC315 e digite:
//      ;getf("/todo_o_caminho_no_Linux/ex-contadores.sci");
// Agora você dispõe das funções: contador_it, contador_rec e galho_rec.
// Para testar a função 'contador_it', somando de 2 até 10, digite
//      contador_it(2,10)
// se quiser o resultado na variável 's' digite
//      s=contador_it(2,10)
// -----

function [cont]=contador_it(k,fim)
    cont=0;
    for j = k:fim, cont=cont+1; end;

// Para usar esta função, após o 'getf', digite por exemplo
//      soma=contador_rec(3,10), soma
//
function [c]=contador_rec(k,fim)
    if (k<fim)
        c=contador_rec(k+1,fim)+1; // Cuidado: retirando-se "c=...", a chamada
                                   // externa ficará com o valor da primeira
        // printf("%f ",cont);    // chamada !!
    else c=1;
    end;

// Cuidado com a recorrência: a = a+EXP => cria nova variável a no nível
//                                     atual da recursão
// Para usar esta função, após o 'getf', digite por exemplo
//      cont=0, galho_rec(1,10)
//
function []=galho_rec(k,fim)
    if (k<fim)
        cont = cont+1;           // (*)
        printf("E: %f ",cont);
        galho_rec(k+1,fim);      // apesar de cont ter sido acresc. em recursão
                                   // acima, aqui ela volta a ter o valor de (*)
        printf("S: %f ",cont);
    end;

end
```

## A.2 Como “pivotar” uma matriz

```
function []=pivo()
//
// MAC 315 - 200 - Prof. Leônidas O. Brandão
//
// Pivoteamento abaixo da diagonal principal
//
// Dicas;
// - acentos não são reconhecidos nos print's !!!
// - obrigatório a primeira linha começar comando e não com comentário.
// - operações em linhas (colunas)
//   B(i,:) = B(i,:) / B(i,i) dividiria toda a linha i de B por b_{ii}

A = [-1,1,0,0; 1,0,1,0; 3,0,0,3; -3,0,1,1];
dim=size(A); n=dim(1); m=dim(2); // pega as dimensões de A
if (n<>m)
    printf("Erro: matriz A(%d,%d) deve ser quadrada",m,n)
    exit;
end;

// Entrada de uma matriz via "diálogo gráfico"
row='linha ' ; labelv = row(ones(1,n))+string(1:n);
col='coluna ' ; labelh = col(ones(1,m))+string(1:m);
new=evstr(x_mdialog('<< Edite a matriz A >>', labelv, labelh, string(A)))

B = A // matrix("empty",n) definiria uma matriz quadrada vazia

for i=1:n
    if (B(i,i)==0)
        printf("Erro: B(%d,%d)=%f",i,i,B(i,i))
        return
    end
    for k=i+1:n // pivoteamento segundo B(i,i)
        B(k,:) = B(k,:) - B(k,i)/B(i,i) * B(i,:)
    end
end

print(%io(2),B,det(B),det(A));

end
```