Métodos Computacionais de Álgebra Linear

Profª Márcia Ruggiero

Relatório – ATC1

Angelica Lourenço Oliveira

Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
IMECC

Trabalho apresentado à disciplina de Métodos Computacionais de Álgebra Linear, ministrado ao 2º semestre, do Curso de Doutorado em Matemática Aplicada, do Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas.

Professora: Dra. Márcia Ruggiero

Aluna: Angélica Lourenço Oliveira

Sumário

ATC1 – Met. Com. de Álgebra Linear

Para alguns dos exercícios que devem ser feito algum programa, fizemos inicialmente um código que faz o cálculo para qualquer dado de entrada desejado, e o chamamos de Caso Geral. Em seguida, foi feito um programa auxiliar, que utiliza o programa do Caso Geral para fazer os calculos somente para os dados solicitados no exercício, no qual chamamos de Programa Auxiliar. Por último, foi criado um código que faz o cálculo somente para os dados solicitados no exercício no mesmo programa. Neste caso, o chamamos de Caso Específico.

Exercício 1

Programa 11 – Produto entre Matriz e vetor.

O Programa 11 consiste em fazer o cálculo do produto entre uma matriz ***A*** e um vetor ***v***. Em seguida, analisar o tempo de execução para diversas dimensões de *A* e comparar com o tempo de execução do produto ***A*v***, já estabelecido pelo MatLab. Para isso, foram utilizados os seguintes comandos do MatLab.

```
>> rand                    %cria um número real aleatório
                           %entre 0 e 1.
>> rand(m,n)               %Cria uma matriz de ordem mxn
                           %cujas entradas são números
                           %reais entre 0 e 1.

>> (b-a)*rand(m,n)+a       %Cria uma matriz de ordem mxn
                           %cujas entradas são números
                           %entre a e b. Os números a e
                           %b são reais e a < b.
```

Caso Geral

Este programa calcula o produto entre uma matriz quadrada ***A*** de ordem *n* qualquer, por um vetor ***v*** de ondem *nx1*.

```
function [w] = Prog11(n)      %retorna o vetor w
A=20*rand(n,n)-8;
v=6*rand(n,1)-3;
z=zeros(n,1);
tic
for i=1:n
    s=0;
    for j=1:n
       s=s+A(i,j)*v(j);
    end
```

```
    z(i)=s;
end
tp=toc;                         %salva na variável tp o tempo
                                %de execução do programa.
tic
A*v;
tm=toc;                         %salva na variável tm o tempo
                                %de execução do MatLab.

w=[tp; tm];
```

Programa auxiliar

O Exercício 1 pede que seja rodado o Programa 11 para dimensões *n* iguais a 1000, 2000, 3000, 4000 e 5000.

```
clear           %limpa todas as variáveis salvas até o
                %momento.
for i=1000:1000:5000
v  = Prog11(i);
j=i/1000;       %faz-se j=1:5, afim de criar um vetor com o
                %tempos de execução para valor de n.
w(j)=v(1);
u(j)=v(2);
end

B=[w; u];       %Matriz dos tempos do programa e do MatLab.
                %A primeira linha (vetor w) são os tempos do
                %programa para cada dimensão. A segunda
                %linha (vetor u) são os tempos dos cálculos
                %feitos pelo MatLab.

plot(B.','*')   %B.' é a transposta de B
                %o comando plot(M) cria um gráfico cujas
                %curvas são as colunas de uma matriz M.
```

Caso Específico

Este programa, uma vez executado, faz o produto para as dimensões iguais a 1000, 2000, 3000, 4000 e 5000. E em cada loop salva o tempo de execução para cada dimensão desejada, tanto para o tempo da execução do programa quanto para o tempo de execução do produto feito pelo MatLab.

```
function [B] = Prog112
for n=1000:1000:5000
    A=20*rand(n)-8;
    v=6*rand(n,1)-3;
```

```
    z=zeros(n,1);

    l=n/1000;

    tic
    for i=1:n
        s=0;
        for j=1:n
            s=s+A(i,j)*v(j);
        end
        z(i)=s;
    end
    tp(l)=toc;

    tic
    A*v;
    tm(l)=toc;

    end

    B=[tp; tm];
    plot(B.','*')
end
```

Tabela de tempos e gráficos

No programa, a matriz *B* contém todos tempos de execução para cada dimensão. A primeira linha são os tempos execução do programa para cada dimensão. A segunda linha são os tempos execução do MatLab para cada dimensão.

```
B =
    0.0133    0.1220    0.1608    0.2955    0.4767
    0.0009    0.0032    0.0072    0.0121    0.0190
```

Tabela de Tempos – Programa e MatLab					
N	1000	2000	3000	4000	5000
T. Prog	0.0133	0.1220	0.1608	0.2955	0.4767
T. MatLab	0.0009	0.0032	0.0072	0.0121	0.0190

```
>> plot(B.')
```

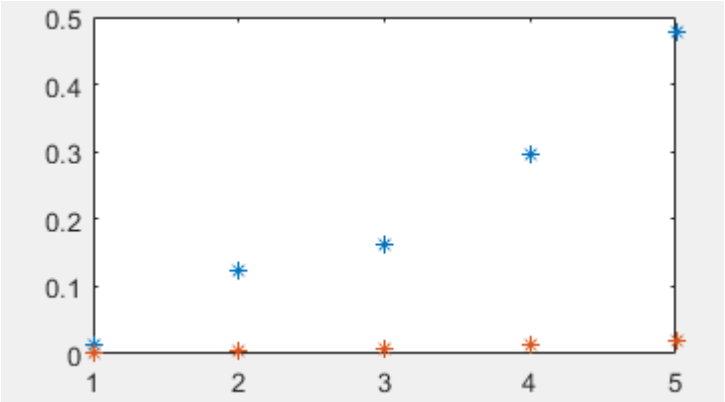

Figura 1 - Eixo horizontal é a dimensão e o vertical o tempo.

Observe que cada vez que aumenta a dimensão, o tempo de execução também aumenta. Pelo gráfico, podemos perceber que o tempo do programa cresce aparentemente exponencialmente ou quadrática.

Exercício 2

Os comandos round, floor, ceil e fix.

Definição: os comandos `round, floor, ceil` e `fix` são usados para arredondar números. Cada um utiliza um critério específico que explicaremos a seguir.

Diferenças:

- Comando `round:` arredonda para o inteiro mais próximo.
- Comando `floor:` arredonda para o menor inteiro mais próximo. Ou seja, para o inteiro mais próximo à esquerda.
- Comando `ceil:` arredonda para o maior inteiro mais próximo. Ou seja, para os inteiro mais próximo à direita.
- Comando `fix:` arredonda para o inteiro mais próximo que tem menor valor em módulo. Em outras palavras, arredonda para o inteiro mais próximo em direção ao zero.

Exemplos com -3.8 -3.2 3.2 3.8

Exercício 2: Exemplifique os comandos utilizando os valores `-3.8, -3.2, 3.2` e `3.8.`

Utilizando o MatLab

Afim de agilizar os cálculos, utilizamos o MatLab da seguinte forma:

```
T = [-3.8 -3.2 3.2 3.8];
for i=1:4
    Round_T(i)=round(t(i));
    Floor_T(i)=floor(t(i));
    Ceil_T(i)=ceil(t(i));
    Fix_T(i)=fix(t(i));
```

```
end
          %arredondamentos dos elementos de T em cada caso
```

Disso, temos que

```
Round_T =
    -4    -3     3     4
```

Arredondou os números `-3.8, -3.2, 3.2` e `3.8` para o inteiro mais próximo de cada um.

```
Floor_T =
    -4    -4     3     3
```

Arredondou os números `-3.8, -3.2, 3.2` e `3.8` para o menor inteiro mais próximo de cada um.

```
Ceil_T =
    -3    -3     4     4
```

Arredondou os números `-3.8, -3.2, 3.2` e `3.8` para o maior inteiro mais próximo de cada um.

```
Fix_T =
    -3    -3     3     3
```

Arredondou os números `-3.8, -3.2, 3.2` e `3.8` para o inteiro próximo de cada um e que tem menor valor em módulo.

Exercício 3

O comando diag(X)

No Matlab, o comando `diag()` pode ser utilizado tanto em matrizes quadradas quanto em vetores. Cada caso, resulta em elementos de estrutura diferentes.

Se aplicado numa matriz ***A***, o comando `diag(A)` nos retornará ***um vetor*** cujos elementos são as entradas da diagonal principal de ***A***. Agora, se for aplicado num vetor ***v***, o comando `diag(v)` nos retornará ***uma matriz diagonal*** cujos elementos de sua diagonal principal são as entradas do vetor **v**.

Por exemplo, tome a matriz ***A*** da seguinte forma:

```
     0     1     4
A =  4     8     8
     3     1     4
```

O comando `v = diag(A)` nos retorna um vetor `v = [0; 8; 4]`. Agora, se tomarmos o vetor `v = [0; 8; 4]`, o comando `B = diag(v)` nos retorna uma matriz quadrada **B** cujos elementos da diagonal principal são `0, 8` e `4`.

```
     0     0     0
B =  0     8     0
     0     0     4
```

Disso, podemos concluir que se fizermos `F = diag(diag(A))` teremos como resultado, *uma matriz diagonal F cuja diagonal principal é exatamente a diagonal principal de A.*

O comando diag(X, n)

Vimos que o comando `diag()`, quando aplicado numa matriz, nos retorna o vetor das entradas da diagonal principal desta matriz e se aplicado num vetor, retorna uma matriz diagonal, cuja diagonal são os elementos do vetor.

O comando `diag()` pode ser utilizado também para criar vetores com as entradas das subdiagonais de uma matriz. Por exemplo, se quisermos um vetor cujos elementos são as entradas da diagonal acima da diagonal de uma matriz ***A***, usamos o comando `diag(A,1)`. Mas, se quisermos um vetor cujos elementos são as entradas da diagonal abaixo da diagonal de ***A***, usamos o comando `diag(A,-1)`.

Esse comando pode ser usado pra criar vetor com entradas de qualquer diagonal de uma matriz a. Caso quisermos um vetor cujas entradas são o elementos da n-ésima diagonal acima da diagonal principal utilizamos o comando `diag(A,n)`. De forma análoga, se quisermos um vetor cujos elementos são as entradas da n-ésima diagonal abaixo da diagonal principal de A, usamos o comando `diag(A,-n)`.

Por exemplo, tome a matriz ***A*** do exemplo anterior. Então `diag(A,2)=[4]` e `diag(A,-2)=[3]`.

Programa 12 – Criando Matrizes Diagonais

Neste programa, utilizamos os comandos `diag()`, `round()` e `rand()`. Tais comando foram explicados anteriormente. Vamos agora, apenas aplicá-los ao Exercício 3.

Neste exercício, devemos criar uma matriz ***D*** com elementos no intervalo [-2 7], todos inteiros. Para isso, utilizamos os comandos mencionados acima.

Escolhemos o comando round para tornar os elementos inteiros, mas poderíamos usar qualquer um dos outros comandos definidos no Exercício 2.

Utilizando o MatLab

```
v=[1; 3; 5; 7];

A=diag(v)
B=diag(v,3)
C=diag(v,-3)

D=round(9*rand(5)-2)
```

```
v=diag(D)
w=diag(D,3)
z=diag(D,-2)
```

Resultados

```
A =
     1     0     0     0
     0     3     0     0
     0     0     5     0
     0     0     0     7

B =
     0     0     0     1     0     0     0
     0     0     0     0     3     0     0
     0     0     0     0     0     5     0
     0     0     0     0     0     0     7
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
C =
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     3     0     0     0     0     0
     0     0     5     0     0     0     0
     0     0     0     7     0     0     0

D =
     2     1     2     1     5
     0     2     0     6     5
    -1     3     3    -1     4
     3     4     3     5     6
     6     5     2    -2     6

v = [2; 2; 3; 5; 6] %diagonal principal de D

w = [1; 5]          %3ª diagonal acima da principal de D

z = [-1; 4; 2]      %2ª diagonal abaixo da principal de D
```

Exercício 4

Programa 13 – Produto entre Matrizes Diagonais

O Programa 13 consiste em fazer o cálculo do produto entre uma matriz diagonal, gerada a partir do comando `diag(v)`, e uma matriz **A**, com elementos inteiros entre -5 e 10. O vetor ***v*** é da forma [2 3 ... *n*] e *n* varia entre 2, 3 e 4. Em seguida encontrar uma relação entre a matriz ***A*** e as matrizes ***B*** e ***C***, onde `B=diag(v)*A` e `C=A*diag(v);`

Em geral, se temos um produto entre uma matriz diagonal ***D*** e uma matriz ***A*** qualquer, acontece algo bem específico da seguinte forma:

- Se a matriz **A** for multiplicada pela esquerda por ***D***, então cada *i*-ésima linha de ***A*** será multiplicada pelo elemento d_{ii} de ***D***.
- Se a matriz ***A*** for multiplicada pela direita por ***D***, então cada *j*-ésima coluna de ***A*** será multiplicada pelo elemento d_{jj} de ***D***.

Caso Geral

Este programa calcula o produto entre uma matriz A quadrada de ordem *n* e uma matriz diagonal gerada a partir do comando `diag(v)`, com `v = [ 2 3 ... n]` e *n* qualquer.

```
function [A, B, C] = Prog13(n)

A=round(15*rand(n)-5);  % cria uma matriz de elementos
inteiros entre -5 e 10

for i=1:n
    v(i)=i;              % v=[1; 2; ... ; n]
end

B=diag(v)*A;
C=A*diag(v);
end
```

Programa auxiliar

O Exercício 3 pede que seja rodado o Programa 13 para *n* igual a 2, 3 ou 4. Desta forma, restringimos os dados de entrada do Prog13.

```
n=input('Entre com a dimensão: ');

while n==1 | n>4    % enquanto n=1 ou n>4 diga
n=input('Dimensão inválida! Entre com um valor entre 2, 3 ou
4: ');
end
                    %caso contrario rode o Prog13

[A,B,C] = Prog13(n) %retorna as matrizes A,B e C do Prog13.
                    %Neste caso utilizamos as mesmas letras.
```

Caso Específico

Este programa, uma vez executado, faz o que é solicitado no exercício, e o usuário entra com o valor *n*, que *n* só pode ser 2, 3 ou 4.

```
clear
n=input('Entre com a dimensão: ');

while n==1 | n>4          % | = ou && = e
n=input('Dimensão inválida! Entre com um valor entre 2, 3
ou 4: ');
end

A=round(15*rand(n)-5)     % cria uma matriz de elementos
                          %inteiros entre -5 e 10

for i=1:n
    v(i)=i;               % v=[1; 2; ... ; 4]
end

D=diag(v)
B=D*A
C=A*D
```

Resultados para n=3

```
A =                       D =
    -3     8    10             1     0     0
     7     9     4             0     2     0
     7    -2     8             0     0     3
B =                       C =
    -3      8    10            -3    16    30
    14    18     8             7    18    12
    21    -6    24             7    -4    24
```

Como mencionado anteriormente, já que a matriz ***B*** é a matriz ***A*** multiplicada à esquerda por uma matriz diagonal, temos que as *i*-ésima linha de ***A*** foi multiplicada por *i* ($d_{ii} = i$). E, como a matriz ***C*** é a matriz ***A*** multiplicada à direita por uma matriz diagonal, temos que a *j*-ésima coluna de ***A*** foi multiplicada por *j* ($d_{jj} = j$).

Exercício 5

Respostas já constam no corpo do relatório. Em particular, no exercício 3.