 Solutions to Computer Assignment #1

Instructor: Prof. Glaucio Paulino
Course: CEE 361 Fall 2000
Prepared by M. Barnard
September 10, 2000

Problem 1: Start playing with MATLAB. For fun, verify the following
properties numerically for any random square matrices A, B, and C of your
own choice. Turn in just one (and only one) MATLAB output for each problem
below.

Note: When needed, the following 2 matrices will be used to solve
all the parts of problem 1 unless otherwise noted.

\[
A = \begin{bmatrix}
2 & 1 & 3 \\
1 & 5 & 7 \\
3 & 7 & 9
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
9 & 4 & 8 \\
4 & 3 & 5 \\
8 & 5 & 7
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
1770 & 1074 & 1682 \\
1074 & 656 & 1024 \\
1682 & 1024 & 1600
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 0 \\
0 & 0
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
1 & 2 \\
1 & 2
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\]

\[
A\ast B
\]

\[
\begin{bmatrix}
1 & 2 \\
0 & 0
\end{bmatrix}
\]

\[
C\ast B
\]

\[
\begin{bmatrix}
1 & 2 \\
0 & 0
\end{bmatrix}
\]

\[
A\ast B = C\ast B \text{ but } A \neq C
\]

\[
A = \begin{bmatrix}
2 & 1 & 3 \\
1 & 5 & 7 \\
3 & 7 & 9
\end{bmatrix}
\]
\[
\begin{bmatrix}
9 & 4 & 8 \\
4 & 3 & 5 \\
8 & 5 & 7 \\
\end{bmatrix}
\]

\[
D = 
\begin{bmatrix}
46 & 26 & 42 \\
85 & 54 & 82 \\
127 & 78 & 122 \\
\end{bmatrix}
\]

\[
D' = 
\begin{bmatrix}
46 & 85 & 127 \\
26 & 54 & 78 \\
42 & 82 & 122 \\
\end{bmatrix}
\]

\[
D_{\text{transpose}} = B'A'
\]

\[
D_{\text{transpose}} = 
\begin{bmatrix}
46 & 85 & 127 \\
26 & 54 & 78 \\
42 & 82 & 122 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & -1 \\
0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 \\
1 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 \\
0 & 0 \\
\end{bmatrix}
\]

Problem 2: Verify if each statement below is true or not. If not true, provide a simple counter-example.

\[
\begin{bmatrix}
1 & 0; 1 & 0 \\
\end{bmatrix}
\]
function [C]=mmult(A,B)

% this function takes two matrices, A and B, and multiplies them together
% to get a matrix C without using built-in arithetic operations
%
% input variables
% % A = a matrix of dimension pxn
% % B = a matrix of dimension nxm
%
% internal variables
% % p = represents the number of rows in A
% % n1 = represents the number of columns in A
% % n2 = represents the number of rows in B
% % m = represents the number of columns in B
% % n = if n1 and n2 are equal, n is the number of rows in C
% % i = for loop counter, represents the number of rows in A
% % j = for loop counter, represents the number of columns in B
% % k = for loop counter, represents the number of columns in A and the
% % number of rows in B
%
% output variables
% % % C = matrix of dimension pxm
% %

[p,n1]=size(A);
[n2,m]=size(B);
C=zeros(p,m);

if n1==n2
    n=n1;
    for i=1:p
        for j=1:m
            for k=1:n
                C(i,j)=C(i,j)+A(i,k)*B(k,j);
            end
        end
    end
else
    disp 'error in matrix dimensions'
end

Problem 3: matrix multiplications

P=K*d

Use the type function while in MATLAB to list the contents of a file in your operating directory. Note: the variables inside the function do not need to have the same names as specified in the problem statement. All that matters is that the variables have the correct form as expected.

To insure that the program functions correctly, we will try an example and see if the results are consistent between our example and the program above.

check mmult.m
A =
2 1 3
1 5 7
3 7 9

B =
9 4 8
4 3 5
8 5 7

C = mmult(A, B)

C =
46 26 42
85 54 82
127 78 122

A * B

ans =
46 26 42
85 54 82
127 78 122

the MATLAB result and our program works exactly!

Check of Floating Point Operations
Built-In Operations

>> startflops2 = flops

startflops2 =
0

A =
2 1 3
1 5 7
3 7 9

B =
9 4 8
4 3 5
8 5 7

>> C2 = A * B

C2 =
46 26 42
85 54 82
127 78 122
46  26  42  
85  54  82  
127  78  122  

>> endflops2 = flops

eendflops2 =

54

Problem Discussion

Therefore, the results from our program do exactly match up with the values generated using the built-in operations. Note that the flops between our program exactly equal the number of flops of the built-in operations. This implies that our program is as efficient as the built-in operations. Flops are the number of floating point operations, i.e. the number of additions or multiplications completed. Theoretically, the number of flops required for this problem would be;

\[ A(m \times n) \times B(n \times p) \implies 2mnq \text{ flops} \]
\[ \implies \text{total number of flops} = 2mnq \]

For our example, \( n \) and \( m \) equal 3 therefore total number of flops should be \( 2 \times 3 \times 3 \times 3 = 54 \). This is exactly what the number of flops were for both methods used here.