Tanweer Molla

Lab 2

Instructor: Rebecca Everett/Dr. Alex Mahalov

October 4, 2012

Partners: Alex King and Josh Rood

**Question 1:**

For this question, we were to generate a n x n matrix for n=1000, n=2000, n=5000. The solution to Ax=b is the vector z because B can be set up as a vector of A.

The following commands were inserted into the command window:

n=1000;

>> A=floor(10\*rand(n));

>> b=sum(A')';

>> z=ones(n,1);

>> tic, x=A\b; toc

Elapsed time is 0.955327 seconds.

>> tic,y=inv(A)\*b;toc

Elapsed time is 0.421522 seconds.

For this particular problem, tic, y is the faster solution. We then inputted the following command:

sum(abs(x-z))

ans =

1.7694e-10

>> sum(abs(y-z))

ans =

2.3999e-09

This particular solution was the backslash method, and was the fastest and most accurate solution. The same steps were done for parts B and C to the question, where n=2000, and n=5000.

For n = 2000:

%

>> n=2000;

>> A=floor(10\*rand(n));

>> b=sum(A')';

>> z=ones(n,1);

>> tic, x=A\b; toc

Elapsed time is 0.531257 seconds.

>> tic,y=inv(A)\*b;toc

Elapsed time is 1.351847 seconds.

>> %Tic X is faster

>> sum(abs(x-z))

ans =

9.8107e-10

>> sum(abs(y-z))

ans =

8.0743e-09

For this solution the top x backslash solution was smaller, therefore the most accurate, and tic x was the faster solution.

For n=5000:

>> n=5000;

>> A=floor(10\*rand(n));

>> b=sum(A')';

>> z=ones(n,1);

>> tic, x=A\b; toc

tic,y=inv(A)\*b;toc

Elapsed time is 6.756629 seconds.

Elapsed time is 19.152842 seconds.

>> %X faster

>> sum(abs(x-z))

ans =

1.0775e-08

>> sum(abs(y-z))

ans =

4.0195e-08

For this solution, the backslash solution represented the most accurate solution, as tic x was also the faster solution.

**Question 2:**

For this problem, we generated another matrix and inputted the following into the command window:

n=100;

>> A=eye(n,n)-triu(ones(n,n),1);

>> b=sum(A')';

>> z=ones(n,1);

>> x=A\b;

>> y=inv(A)\*b;

Warning: Matrix is close to singular or badly scaled. Results may

be inaccurate. RCOND = 1.577722e-32.

>> sum(abs(x-z))

ans =

0

>> sum(abs(y-z))

ans =

45

>> %backslash most accurate

We found that once again, the backslash solution gave the most accurate solution to the above matrix.

**Question 3:**

There were several parts to this question. We were first asked to generate the following matrix:

A=floor(10\*rand(6));

>> b=floor(20\*rand(6,1))-10;

>> x=A\b

x =

-7.0380

3.0815

-4.6260

1.8456

8.2492

-1.2694

Part B:

We were asked to put the following command in row echelon form:

U=rref([A,b])

U =

Columns 1 through 6

1.0000 0 0 0 0 0

0 1.0000 0 0 0 0

0 0 1.0000 0 0 0

0 0 0 1.0000 0 0

0 0 0 0 1.0000 0

0 0 0 0 0 1.0000

Column 7

-7.0380

3.0815

-4.6260

1.8456

8.2492

-1.2694

Part C:

We computed the difference using the format long and short command functions on MATLAB:

format long

>> U(:,7)-x

ans =

1.0e-04 \*

0.021273020101020

-0.026945825450930

0.295746864384938

0.080243083027121

-0.011926840794985

-0.020134795737370

U(:,7)-x

ans =

1.0e-04 \*

0.0213

-0.0269

0.2957

0.0802

-0.0119

-0.0201

Part D:

The following was inputted into the command window:

A(:,3)=A(:,1:2)\*[4,3]'

A =

3 7 33 3 1 6

5 5 35 7 5 0

0 6 18 3 0 1

7 9 55 9 0 4

8 5 47 7 8 7

4 4 28 0 6 2

>> rref([A,c])

rref([A,b])

ans =

1 0 4 0 0 0 0

0 1 3 0 0 0 0

0 0 0 1 0 0 0

0 0 0 0 1 0 0

0 0 0 0 0 1 0

0 0 0 0 0 0 1

We found that there will be infinitely many solutions to this particular matrix.

Part E:

We know that Ax=C must be consistent because there is at least one solution in the system. C is a linear combination of the column vectors of A, as shown below:

y=floor(20\*rand(6,1))-10;

>> c=A\*y;

>> rref([A,c])

ans =

1 0 4 0 0 0 -33

0 1 3 0 0 0 -22

0 0 0 1 0 0 -4

0 0 0 0 1 0 6

0 0 0 0 0 1 0

0 0 0 0 0 0 0

For this particular problem, we found that there are infinitely many solutions.

**Question 4:**

This problem introduced us to script files. The following script was made in a separate m file for this particular problem:

function [y]=myproductroww(A,x)

[m,n]=size[A];

[p,q]=size(x);

y=zeros(m,q);

if (n==p&&q==1)

for i=l:m

y(i)=A(i,:)\*x;

end

else

disp('dimensions do not match')

y=[];

end

A is a matrix, x is a vector. With this script, we got the following as the output:

myrowproduct(A,x)

ans =

44 76

A\*x

ans =

44

76

A\*x and the myrowproduct script gave the same answers. We did it for another matrix example:

myproductrow(A,x)

Dimensions do not match

ans =

0

**Question 5:**

This problem introduced us to more scripts, functions, column products, and row products.

For the row product, the following was entered into a script:

function [ C ] = rowproduct( A,B )

m = length(A(:,1));

n = length(A(1,:));

p = length(B(:,1));

q = length(B(1,:));

C = zeros(m,q);

if(n ~= p)

disp('Dimensions of A and B are incompatible');

else

for i = 1:m

C(i,:) = A(i,:)\*B;

end

end

end

We got the following output:

A = floor(10\*rand(2,4))

A =

3 1 0 6

2 5 9 7

>> B = floor(10\*rand(4,3))

B =

0 6 7

5 3 8

9 4 3

0 9 3

>>

rowproduct(A,B)

ans =

5 75 47

106 126 102

columnproduct(A,B)

ans =

5 75 47

106 126 102

A\*B

ans =

5 75 47

106 126 102

A = floor(10\*rand(2,7))

A =

8 2 9 4 6 7 5

6 7 9 5 2 1 8

rowproduct(A,B)

Dimensions do not match

ans =

0 0 0

0 0 0

columnproduct(A,B)

Dimensions do not match

ans =

0 0 0

0 0 0

Row products, column products gave the exact same answers, and while extending the matrix, we got all zeros for answers because the dimensions did not match.