MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

1. (a) Set $n = 1000$ and generate an $n \times n$ matrix and two vectors in $\mathbb{R}^n$, all having integer entries, by setting

```
A = floor(10*rand(n));
b = sum(A')';
z = ones(n,1);
```

The exact solution of the system $A\mathbf{x} = \mathbf{b}$ is the vector $\mathbf{z}$. Why? Explain.
One could compute the solution in MATLAB using the "\" operation or by computing A^{-1} and then multiplying A^{-1} times $\mathbf{b}$. Let us compare these two computational methods for both speed and accuracy. The inverse of the matrix A can be computed in MATLAB by typing `inv(A)`. One can use MATLAB `tic` and `toc` commands to measure the elapsed time for each computation. To do this, use the commands

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

Which method is faster?
To compare the accuracy of the two methods, we can measure how close the computed solutions $\mathbf{x}$ and $\mathbf{y}$ are to the exact solution $\mathbf{z}$. Do this with the commands:

```
sum(abs(x - z))
sum(abs(y - z))
```

What method produces the most accurate solution?

(b) Repeat part (a) using $n = 2000$ and $n = 5000$.

%EXERCISES GROUP 1

%Set n = 1000 and generate an n x n matrix.

n = 1000;A=floor(10*rand(n));b=sum(A')';z=ones(n,1);

% The exact solution of the system Ax = b is the vector z.

% The operation A\b takes each element of b, which is a 1000x1

% vertical vector and divides to matrix A's sums of columns,

% which is a 1x1000 vertical vector. The net effect is a

% 1000x1 vertical vector of 1's.

% Elapse times for '\' and inv() operations

tic, x = A\b; toc

elapsed_time =

```
  0.2930

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

elapsed_time =

   0.7440

% The data showed a faster execution time using the '\'
% operation.
% Compare the accuracy of the two methods.
sum(abs(x - z))

ans =

  1.4653e-009

sum(abs(y - z))

ans =

  1.8085e-008

% As shown above, the '\' operation was more accurate
% at least by an order of magnitude (i.e smaller error band).
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
% n = 2000
n = 2000;A=floor(10*rand(n));b=sum(A')';z=ones(n,1);
tic, x = A\b; toc

elapsed_time =

    1.7640

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

elapsed_time =

    5.1530

% n = 5000
n = 5000;A=floor(10*rand(n));b=sum(A')';z=ones(n,1);
??? Error using ==> *
Out of memory. Type HELP MEMORY for your options.

% It seemed that MatLab was not capable of handling a 5000x5000
% square matrix, but I expected a similar result as n = 2000.
% (In my laptop only).
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

2. Consider the 100×100 matrix:

$$A = \begin{bmatrix} 1 & -1 & \dots & -1 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & -1 \\ 0 & \dots & 0 & 1 \end{bmatrix}$$

Use the following MATLAB commands to generate the matrix A and two vectors $\mathbf{b}$ and $\mathbf{z}$

```
n = 100;
A = eye(n,n) - triu(ones(n,n),1);
b = sum(A')';
z = ones(n,1);
```


Similarly to the previous problem, the exact solution of the system $A\mathbf{x} = \mathbf{b}$ is the vector $\mathbf{z}$. Compute the solution using the "\" and using the inverse:

```
x = A\b;
y = inv(A)*b;
```

Compare the accuracy of the two methods as in the previous problem.
What method produces the most accurate solution?

% PART 2

n = 100;A = eye(n,n) - triu(ones(n,n),1);

n = 100;A = eye(n,n) - triu(ones(n,n),1);b = sum(A')';z = ones(n,1);

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
x = A\b;

y = inv(A)*b;

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.577722e-032.
sum(abs(x - z))


ans =


     0


sum(abs(y - z))


ans =


    45


% The '\' method is still more accurate than the inv() method.
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

3. Generate a matrix A by setting `A = floor(10*rand(6));`
and generate a vector $\mathbf{b}$ by setting `b = floor(20*rand(6,1))-10;`

   (a) Since A was generated randomly, we would expect it to be nonsingular. The system $A\mathbf{x} = \mathbf{b}$ should have a unique solution. Find the solution using the "\" operation (if MATLAB gives a warning about the matrix being close to singular, generate the matrix A again).

   (b) Use MATLAB to compute the reduced row echelon form, U, of the augmented matrix `[A b]` (use the command `rref`).
   Note that, in exact arithmetic, the last column of U and the solution $\mathbf{x}$ from part (a) should be the same since they are both solutions to the system $A\mathbf{x} = \mathbf{b}$.

   (c) To compare the solutions from part (a) and part (b), compute the difference `U(:,7) - x` or examine both using `format long` (when you are done, type `format short` to go back to the original format).

   (d) Let us now change A so as to make it singular. Set

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

   (the above command replaces the third column of A with a linear combination of the first two: $\mathbf{a}_3 = 4\mathbf{a}_1 + 3\mathbf{a}_2$ where $\mathbf{a}_i$ is the ith column of A.)
   Use MATLAB to compute `rref([A b])`. How many solutions will the system $A\mathbf{x} = \mathbf{b}$ have? Explain.

   (e) Generate two vectors, $\mathbf{y}$ and $\mathbf{c}$ by setting

   ```
   y = floor(20*rand(6,1)) - 10;
   c = A*y;
   ```

   (here `A` is the matrix from part (d)).
   Why do we know that the system $A\mathbf{x} = \mathbf{c}$ must be consistent? Explain.
   Compute the reduced row echelon form U of `[A c]`. How many solutions does the system $A\mathbf{x} = \mathbf{c}$ have? Explain.

% PART 3

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

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

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

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

x=A\b;

ar=rref([A b]);

ar

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
ar =

  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

    0.0413
   -2.5688
   -1.3899
    7.7202
   -0.0092
   -3.9862

x

x =

    0.0413
   -2.5688
```

```
   -1.3899
    7.7202
   -0.0092
   -3.9862

% rref([A b]) = A\b
ar(:,7)-x

ans =

  1.0e-014 *

    0.0756
   -0.1332
   -0.0888
   -0.0888
    0.0673
    0.1332

% Observation: Very small error band.
A(:,3) = A(:,1:2)*[4,3]'

A =

     0     8    24     7     6     9
     8     6    50     5     3     5
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
     9     8    60     7     1     7
     7     8    52     8     4     7
     0     6    18     9     8     9
     8     9    59     5     8     3
```

```
ar=rref([A b])

ar =

     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

% The last row of all zeros in the coeefficient side means
% a mztrix with inconsistent solution set.
% In this case, no solution.
y = floor(20*rand(6,1)) - 10;
c = A*y;
A\c
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 3.610904e-018.

ans =
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
   24.6250
   21.2187
    1.5938
   -6.0000
    2.0000
    1.0000

c

c =

   187
   385
   454
   392
   127
   471

A

A =

     0     8    24     7     6     9
     8     6    50     5     3     5
     9     8    60     7     1     7
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

7	8	52	8	4	7
0	6	18	9	8	9
8	9	59	5	8	3

y

y =

7
8
6
-6
2
1

U=rref([A c])

U =

1	0	4	0	0	0	31
0	1	3	0	0	0	26
0	0	0	1	0	0	-6
0	0	0	0	1	0	2
0	0	0	0	0	1	1
0	0	0	0	0	0	0

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

% Ax = c is consistent because with many solutions

% because the last row is [0,0,0,...,0] and we don't

% have a pivot on the third column.

% U of [A c] has many solutions due to a 'free' term

% in column 3.

myrowproduct.m file

%---

%Write a function M-file that takes as input a matrix A and a vector x, and as output %gives the

% product y = Ax by row, as defined above (Hint: use a for loop to define each entry of %the vector

% y.) The M-file should perform a check on the dimensions of the input variables A and %x and return

% a message if the dimensions do not match. Call the file myrowproduct.m.

```
function [y] = myrowproduct(A,x)
[rA,cA] = size(A); % determine the dimension of A
[rx,cx] = size(x); % determine the dimension of x

if(cA == rx)            % check the dimensions, columns of A
                        % must equal rows of x
   y = zeros(rA,1);     % initialize the vector y, a 6x1 vector
   for i = 1:rA
      y(i) = A(i,:)*x;  % implement the formula
   end                  % End for loop
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
else                    % if dimension not agreeable
   disp('dimensions do not match')% throw an error message
   y = [];              % return an empty matrix
end                     % end if
end                     % end function
%--------------------------------------------------------------------------------
```

```
Command Window
>> A

A =

     9     2     7     4     8     1
     0     0     5     0     2     3
     0     1     0     9     0     4
     1     8     3     9     4     8
     7     2     2     2     3     3
     3     6     3     3     4     2

>> x

x =

    19
     8
     4
    18
    19
     8

>> [y]=myrowproduct(A,x)

y =

   447
    82
   202
   397
   274
   263

>>
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

rowproduct.m

```
%-------------------------------------------------------------------------------------------
% Write a function M-file that takes as input two matrices A and B, and
% as output produces the product by rows of the two matrices

function [C] = rowproduct(A,B)
[rA,cA] = size(A); % determine the dimension of A
[rB,cB] = size(B); % determine the dimension of B

if(cA == rB)            % check the dimensions, columns of A
                        % must equal rows of B
   C = zeros(rA,cB);    % initialize    C, a mxq matrix
   for i = 1:rA
      C(i,:) = A(i,:)*B;  % implement the formula
   end                  % End for loop
else                    % if dimension not agreeable
   disp('dimensions do not match')% throw an error message
   C = [];              % return an empty matrix
end                     % end if
end                     % end function
%-------------------------------------------------------------------------------------------
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

Command Window

```
x =

    19
     8
     4
    18
    19
     8

>> [y]=myrowproduct(A,x)

y =

   447
    82
   202
   397
   274
   263

>> A = floor(10*rand(6,7));B = floor(20*rand(7,5));
>> [C]=rowproduct(A,B)

C =

   344   306   335   403   128
   367   281   407   431   228
   425   342   411   399   232
   363   205   344   413   214
   244   236   232   200    87
   202   169   106   200    70

>>
```

columnproduct.m

%---

% Write a function M-file that takes as input two matrices A and B,

% and as output produces the product by columns of the two matrix.

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

```
function [C] = rowproduct(A,B)
[rA,cA] = size(A); % determine the dimension of A
[rB,cB] = size(B); % determine the dimension of B

if(cA == rB)              % check the dimensions, columns of A
                          % must equal rows of B
    C = zeros(rA,cB);     % initialize   C, a mxq matrix
    for i = 1:cB
        C(:,i) = A*B(:,i);  % implement the formula
    end                   % End for loop
else                      % if dimension not agreeable
    disp('dimensions do not match')% throw an error message
    C = [];               % return an empty matrix
end                       % end if
end                       % end function
%-------------------------------------------------------------------------------------------
```

MAT 343 MATLAB LAB 2 **NAME: _Hieu Pham____**

Command Window

```
>> A = floor(10*rand(6,7));B = floor(20*rand(7,5));
>> [C]=rowproduct(A,B)

C =

   344   306   335   403   128
   367   281   407   431   228
   425   342   411   399   232
   363   205   344   413   214
   244   236   232   200    87
   202   169   106   200    70

>> [C]=columnproduct(A,B)

C =

   344   306   335   403   128
   367   281   407   431   228
   425   342   411   399   232
   363   205   344   413   214
   244   236   232   200    87
   202   169   106   200    70

>> A = floor(10*rand(3,4));B = floor(20*rand(4,5));
>> [C]=columnproduct(A,B)

C =

   159   149   122   105   243
   105   153   108    37   135
   283   270   231   191   397

>>
```