LAB 2 - WILL TIRONE

Question 1 (a) Set n = 700. The commands below ...

(i) Which method is faster?

Answer: matrix left division is faster

```
n = 700;
A = floor(25*rand(n));
z = ones(n,1);
b = A*z;

tic, x = A\b; toc
```

```
Elapsed time is 0.007782 seconds.
```

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

```
Elapsed time is 0.013619 seconds.
```

(ii) Which is more accurate?

Answer: the smaller number is more accurate (I think this is representative of a smaller error) so the first method below, using A\b is more accurate

```
a = sum(abs(x - z))
```

```
a = 7.8212e-11
```

```
b = sum(abs(y - z))
```

```
b = 1.2236e-09
```

```
a<b
```

```
ans = logical
   1
```

(b) Repeat part (a) using n = 1400 and n = 2800.

In both cases, using n=1400 and 2800, is faster and more accurate.

```
n_values = [1400, 2800];
len = length(n_values);

for i=1:len
    A = floor(25*rand(n));
    z = ones(n,1);
    b = A*z;

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

    a = sum(abs(x - z))
    b = sum(abs(y - z))
    if a<b
        disp('A\b is more efficient than the inverse method')
        disp(' ')
    else
        disp('inv(A) is more efficient than A\b')
    end

end
```

```
Elapsed time is 0.008345 seconds.
Elapsed time is 0.014773 seconds.
a = 2.6340e-10
b = 9.8917e-10
A\b is more efficient than the inverse method

Elapsed time is 0.008387 seconds.
Elapsed time is 0.016707 seconds.
a = 9.8925e-10
b = 1.5833e-08
A\b is more efficient than the inverse method
```

(c) Explain why the exact solution of the system Ax = b is the vector z?

Because b was created using Az, substituting Az into the equation for z, we can see that x must equal z. Alternatively, using the approaches above, x = A\A z reduces to x = z.

$$Ax = b$$
$$Ax = Az$$

Question 2 - The goal of this exercise is to emphasize how, when solving a linear system, the choice of which method to use is extremely important especially in the case of matrices which are close to singular (these matrices are often refereed to as ill conditioned.

Answer: it looks like A\b produced a more accurate result again; based on the prompt above, it looks like this is supposed to be an ill conditioned or almost singular matrix. The inverse method produces a huge deviation from the exact solution.

```
n = 80 ;
B = eye(n) - triu(ones(n),1);
A=B'*B;
z = ones(n,1);
b= A*z;
x = A\b;
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.331182e-51.
```

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

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.331182e-51.
```

```
a = sum(abs(x - z))
```

```
a = 0
```

```
b = sum(abs(y - z))
```

```
b = 1.3871e+34
```

```
a<b
```

```
ans = logical
   1
```

Question 3 - 3. Generate a random 7 x 7 matrix A with integer entries...

a)

```
A = floor(10*rand(7));
b = floor(20*rand(7,1))-10;


x = A\b
```

```
x = 7×1
    1.5044
   -2.4330
    0.6699
    0.3418
    1.4104
   -0.9372
   -0.9220
```

b)

```
U = rref([A, b])
```

```
U = 7×8
    1.0000         0         0         0         0         0         0    1.5044
         0    1.0000         0         0         0         0         0   -2.4330
         0         0    1.0000         0         0         0         0    0.6699
         0         0         0    1.0000         0         0         0    0.3418
         0         0         0         0    1.0000         0         0    1.4104
         0         0         0         0         0    1.0000         0   -0.9372
         0         0         0         0         0         0    1.0000   -0.9220
```

c)

```
U(:,8) - x
```

```
ans = 7×1
10^-5 ×
    0.5348
    0.2894
   -0.8314
    0.2869
   -0.6321
    0.1694
    0.9950
```

d) How many solutions will the system Ax = b have? Explain.

Because this is now an augmented matrix, comparing the last row shows that 0=1 which means there is no solution to the equation

```
A(:,5) = 7*A(:,4)+4*A(:,3);
rref([A b])
```

```
ans = 7×8
     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     1     0     4     0     0     0
     0     0     0     1     7     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     1
```

e) The way the vector c is defined guarantees that the system Ax = c is consistent, that

is, it has at least one solution. Explain why that is the case.

Answer: from the below equation, using substitution, the system is consistent because y is a guaranteed solution based on how c is defined.

$$\text{Ax} = c$$
$$\text{Substituting Ay for } c:$$
$$\text{Ax} = \text{Ay}$$
$$\text{Solving for } x:$$
$$x = A\backslash\text{Ay}$$
$$x = y$$

```
y = floor(20*rand(7,1)) - 10;
c = A*y;
```

f) How many solutions does the system Ax = c have? Explain.

Answer: it looks like column 5 is a free variable, meaning the system has infinitely many solutions.

```
U = rref([A c])
```

```
U = 7×8
     1     0     0     0     0     0     0    -8
     0     1     0     0     0     0     0     0
     0     0     1     0     4     0     0     0
     0     0     0     1     7     0     0   -15
     0     0     0     0     0     1     0     2
     0     0     0     0     0     0     1     9
     0     0     0     0     0     0     0     0
```

Question 4:

```
type myrowproduct.m
```

```
function y = myrowproduct (A,x)

% The command myproduct (A,x) computes the product
% of the matrix A and the vector x by row .

[m,n] = size (A); % [m=2, n=7]
[p,q] = size (x); % [p=7, q=1]
    if (q ==1 && p==n) % passes the check above
        y = zeros (m ,1); % y = zeros(2,1)
        for i = 1:m % i from 1 to 7
            y(i) = y(i) + A(i,:)*x; % y + x(1) * A(1,all columns)
        end
    else
        disp (' dimensions do not match ')
    y = [];
    end
end
```

Output from command line:

```
>> A=rand(4,4); x=rand(4,1);

>> myrowproduct(A,x)


ans =

0.5551

1.0417

0.7477

0.9857


>> A*x

ans =

0.5551

1.0417

0.7477

0.9857
```

```
>> B=rand(2,7);z=rand(7,1); t=rand(1,7);

>> myrowproduct(B,z)


ans =

1.1638

1.7430


>> myrowproduct(B,t)

dimensions do not match


ans =

[]

>> B*z

ans =

1.1638

1.7430


>> B*t

Error using  *

Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in

the second matrix. To perform elementwise multiplication, use '.*'.
```

Question 5:

```
type columnproduct.m
```

```
function [C] = columnproduct(A,B)

% The command columnproduct (A,B) computes the product
% of the matrix A and the matrix B by column .

[m,n] = size(A);
[p,q] = size(B);
    if (p==n)
        C = zeros (m ,q);
        for i = 1:q
            C(:,i) = C(:,i) + A*B(:,i);
        end
    else
    disp (' dimensions do not match ')
    C = [];
    end
end
```

a)

Output from command line:

```
>> A=rand(3,4); B=rand(4,5);

>> columnproduct(A,B)

ans =

0.6359    0.6239    1.0633    0.4127    0.7110

0.2511    0.1770    0.1325    0.2828    0.1812

0.4186    0.3946    0.7853    0.2926    0.4061

>> A*B


ans =

0.6359    0.6239    1.0633    0.4127    0.7110

0.2511    0.1770    0.1325    0.2828    0.1812

0.4186    0.3946    0.7853    0.2926    0.4061
```

Repeating question 5 a) with different matrices:

```
>> A=rand(3,6); B=rand(6,5); D=rand(5,6);

>> columnproduct(A,B)


ans =

0.6166    1.2164    0.5213    1.0818    0.5719

1.4235    2.1813    1.5400    1.8692    1.7351

1.2488    1.9242    1.1554    1.8377    1.4035

>> columnproduct(A,D)

dimensions do not match
```

ans =

[]

>> A*B

ans =

```
0.6166    1.2164    0.5213    1.0818    0.5719

1.4235    2.1813    1.5400    1.8692    1.7351

1.2488    1.9242    1.1554    1.8377    1.4035
```

>> A*D

Error using *

Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in

the second matrix. To perform elementwise multiplication, use '.*'.

5 b)

```
type rowproduct.m
```

```
function [C] = rowproduct(A,B)
% The command columnproduct (A,B) computes the product
% of the matrix A and the matrix B by column .

[m,n] = size(A);
[p,q] = size(B);
    if (p==n)
        C = zeros (m ,q);
        for i = 1:m
            C(i,:) = C(i,:) + A(i,:)*B;
        end
    else
    disp (' dimensions do not match ')
    C = [];
    end
end
```

From command line:

>> A=rand(4,5); B=rand(5,2);

>> rowproduct(A,B)

ans =

```
1.7143    1.7365

1.5441    2.2160
```

```
1.8142    2.3172

1.3129    1.8069


>> A*B

ans =

1.7143    1.7365

1.5441    2.2160

1.8142    2.3172

1.3129    1.8069


>> A=rand(3,6); B=rand(6,5); D=rand(5,6);

>> rowproduct(A,B)


ans =

1.3152    0.7335    1.5242    1.9414    1.5972

0.9735    0.2985    1.3361    1.6781    1.1031

1.1290    0.9122    1.8262    1.7658    1.6433

>> rowproduct(A,D)

dimensions do not match


ans =

[]

>> A*B

ans =

1.3152    0.7335    1.5242    1.9414    1.5972

0.9735    0.2985    1.3361    1.6781    1.1031

1.1290    0.9122    1.8262    1.7658    1.6433

>> A*D

Error using  *
```

```
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the
number of rows in

the second matrix. To perform elementwise multiplication, use '.*'.
```