Homework 2.

Amath 352
Applied Linear Algebra and Numerical Analysis

Due: 1/20/22 at 11:59pm to Gradescope

Directions:

Complete all component skills exercises and the multi-step problem as neatly as possible. Up to 2 points may be deducted for homework that is illegible and/or poorly organized. You are encouraged to type homework solutions, and a half bonus point will be awarded to students who use LaTeX. (Check out my LaTeX beginner document and overleaf.com if you are new to LaTeX.) If you prefer not to type homeworks, I ask that **homeworks be scanned.** (I will not accept physical copies.) In addition, **homeworks must be in .pdf format.** For MATLAB questions, **please attach your codes and their output** at the end of your assignment.

Pro-Tips:

- You have access to some solutions of the textbook exercises and are encouraged to use them. Note that these solutions are not always correct, so double check your work just in case.
- Teamwork makes the dream work, but please indicate at the top of your assignment who your collaborators are.
- Don't wait until the last minute. ☺

Component Skills Exercises

Exercise 1. (CS1.5)

Determine the output of each of the following MATLAB codes.

a.
```
x = [0,-1,2,-3,4,-5];
m = length(x);
SUM = 0;
for j=m-1:-1:1 % j counts down from m-1 to 1 in intervals of 1.
   if x(j) > 1
      SUM = SUM + 1;
   end
end
SUM % What is the value of SUM?
```

SUM = 2

b.
```
x = [1,2,-3,4,3,-5,-8,9,9,1];
j = 1;
PROD = 1;
while abs(x(j)) < abs(x(j+1)) % abs(x(j)) is the absolute value of entry x(j)
   PROD = PROD*x(j);
   j = j + 1;
end
PROD % What is the value of PROD?
```

PROD = -6

c.
```
A = [1,7,-3, 4; 2,-5,-8, 6; 3,9,9,6];
[m n] = size(A);
SUM = 0;
for i = 1:m
   for j = i:n
      SUM = SUM + A(i,j);
   end
end
SUM % What is the value of SUM?
```

sum = 17

d.
```
A = [1,7,0,4;2,-5,-8,0;3,0,9,2];
B = [1, 2; 3, 0; 0, 8];
C = [B,B(:,2)];
D = C*A;
E = D(2:end,1:end);
F = E.^2 % What are the dimensions of F? What are its entries?
```

F is 2x4

$$F = \begin{bmatrix} 9 & 441 & 0 & 144 \\ 1600 & 1600 & 64 & 256 \end{bmatrix}$$

Exercise 2. (CS1.5)

Consider the following matrix:

$$A = \begin{pmatrix} 8 & -7 & 10 \\ 3 & 2 & -4 \\ 12 & 4 & 4 \\ 1 & -1 & 3 \end{pmatrix}.$$

a. Write a MATLAB script that determines the largest entry of A. Please do not use MATLAB built-in functions like max or find. Use loops and/or if statements.

```
A = [8 -7 10;3 2 -4; 12 4 4; 1 -1 3];
[m, n] = size(A);
max = A(1, 1);
for i = 1:m
   for j = 1:n
      if A (i, j) > max
         max = A(i, j);
      end
   end
end
max
```

b. Write a MATLAB script that determines the smallest entry of A. Please do not use MATLAB built-in functions like min or find. Use loops and/or if statements.

```
A = [8 -7 10;3 2 -4; 12 4 4; 1 -1 3];
[m, n] = size(A);
min = A(1, 1);
for i = 1:m
   for j = 1:n
      if A (i, j) > min
```

```
            min = A(i, j);
        end
    end
end
min
```

Exercise 3. (CS1.6)

Suppose U is a $m \times (m+1)$ augmented, upper triangular matrix that represents a triangular linear system with m equations and m unknowns. Assuming the entries along the main diagonal of U are nonzero, we can perform back substitution on U to obtain the unique solution to the linear system. In MATLAB, the code that performs back substitution is given below.

```
% Back substitution on a m by m+1 upper triangular matrix U.

[m,n] = size(U);
x = U(:,m+1); % Initialize column vector of unknowns (to be updated).
x(m) = U(m,m+1)/U(m,m); % Solve last equation of augmented matrix.
for i = m-1:-1:1 % i counts down from m-1 to 1 in intervals of 1.
   SUM = 0;
   for j = i+1:m
      SUM = SUM + U(i,j)*x(j);
   end
   x(i) = (U(i,n) - SUM)/U(i,i); % Updates the ith entry of x.
end
x % x is the solution of the linear system.
```

a. How many flops occur per pass through the inner for-loop?

   2 flops occur per pass through the inner for loop.

b. How many times does one pass through the inner for-loop? Using your answer to the previous question, what is the total flop count of the inner for-loop?

   Note: Your answers should depend on the index i.

   $\sum_{j=i+1}^{m} 2$
   $2\sum_{j=i+1}^{m} 1$
   2(m-(i+1)+1)
   2m - 2i flops

c. How many flops are needed to update the ith entry of the vector x? Using this answer and your answer to the previous question, how many flops occur per pass through the outer for-loop?

Note: Your answer for the flops per pass through the outer for-loop should still depend on the index i.

2m - 2i + 2 flops

d. Show that the total number of flops that occur within the outer for-loop is $m^2 + m - 2$.

Hint: Recall the useful identity $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$.

from c) 2m - 2i + 2 flops per pass
m-1 to 1 times which is equivalent to a sum from 1 to m-1
m-1 + m-2 .... + 2 + 1 = 1 + 2 ... + m-2 +m-1 =

$\sum_{i=1}^{m-1} 2m - 2i + 2$

$\sum_{i=1}^{m-1} 2m + \sum_{i=1}^{m-1} -2i + \sum_{i=1}^{m-1} +2$

$(2m)(m-1) - \frac{2(m-1)(m-1+1)}{2} + 2(m-1)$

$2m^2 - 2m - m^2 + m + 2m - 2$

$m^2 + m - 2$

e. Solving for $x(m)$ requires one additional flop so that the total flop count of back substitution is $m^2 + m - 1$. For very large matrices, what is the asymptotic flop count of back substitution?

For asymptotic flop count the higher power m will dominate the flop count. so it will be m^2

Exercise 4. (CS2.1-2.2)

What row operations do each of the following elementary matrices E represent? For each E, compute its inverse E^{-1}.

a. $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ $E^{-}1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ $R_1 <> R_2$

b. $\begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $E^{-}1 = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $R_1 = R_1 - 2R_2$

c. $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}$ $E^{-}1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}$ $R_3 = R_3 + 3R_1$

Multi-Step Problem

Note: This problem will involve programming. Please attach your code and its output to the end of your homework assignment.

Suppose we have a linear system with m equations and m unknowns. Let A be the $m \times m$ coefficient matrix associated with this system, and b be the $m \times 1$ column vector of knowns. The following MATLAB code performs regular Gaussian elimination on this system, when possible.

```
M = [A,b];
[m,n] = size(M);
for j = 1:m
   if M(j,j)==0
      error('System cannot be solved by regular Gaussian elimination.');
   end
   for i = j+1:m
      l_ij = M(i,j)/M(j,j);
      M(i,j:n) = M(i,j:n)-l_ij*M(j,j:n);
   end
end
```

To test out this algorithm, let's consider the following linear system:

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \frac{1}{6^3} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}.$$

For those familiar with numerical methods for ordinary differential equations, the system above is a finite-difference discretization of the boundary value problem

$$\begin{cases} u'' = x & 0 < x < 1 \\ u(0) = u(1) = 0 \end{cases}$$

with five equally spaced interior nodes.

Remark: the coefficient matrix above is called a *tridiagonal matrix* because all of its nonzero entries are concentrated along the three main diagonals of the matrix.

a. Create a new script in MATLAB called hw2_msp.m. In your new script, create a section titled "Part 1." In this section, copy the provided code to perform regular Gaussian elimination on the augmented matrix that represents the linear system above. In your code, display the resulting upper triangular augmented matrix to the Command Window. Use format long to show all significant digits of this matrix.

b. Create a new section of your code titled "Part 2." Copy the code provided in **Exercise 3** to perform back substition on the upper triangular augmented matrix obtained from Part 1. Display the resulting solution vector to the Command Window. Use format long to show all significant digits of this matrix.

```
%% PART 1 Gaussian Elimination
format long;
A = [-2 1 0 0 0;1 -2 1 0 0; 0 1 -2 1 0; 0 0 1 -2 1; 0 0 0 1 -2];
b = (1/6^3) * [1;2;3;4;5];

M = [A,b];
[m,n] = size(M);
for j = 1:m
    if M(j,j)==0
        error('System cannot be solved by regular Gaussian elimination.');
    end
    for i = j+1:m
        l_ij = M(i,j)/M(j,j);
        M(i,j:n) = M(i,j:n) - l_ij * M(j,j:n);
    end
end
M

%{
M =

   Columns 1 through 5

   -2.000000000000000    1.000000000000000                    0
0                    0
                    0   -1.500000000000000    1.000000000000000
0                    0
                    0                    0   -1.333333333333333    1.0000000000
0
                    0                    0                    0   -1.2500000000
1.000000000000000
                    0                    0                    0
0   -1.200000000000000

   Column 6

    0.004629629629630
    0.011574074074074
    0.021604938271605
    0.034722222222222
    0.050925925925926
%}
```

```
%% PART 2 Back Substituion
[m,n] = size(M);
x = M(:,m+1); % Initialize column vector of unknowns (to be updated).
x(m) = M(m,m+1)/M(m,m); % Solve last equation of augmented matrix.
for i = m-1:-1:1 % i counts down from m-1 to 1 in intervals of 1.
    SUM = 0;
    for j = i+1:m
        SUM = SUM + M(i,j)*x(j);
    end
    x(i) = (M(i,n) - SUM)/M(i,i); % Updates the ith entry of x.
end
x

%{

x =

   -0.027006172839506
   -0.049382716049383
   -0.062500000000000
   -0.061728395061728
   -0.042438271604938
%}

.
```