Experiment:4
Solving systems using Direct methods

1. Row-Reduced Echelon Form

Create a code in MATLAB that can be called as a function (name it *r_r_e_f.m*) that gives row reduced echelon form of any given vector and also its rank. Find the rref of A and B using the above mentioned function:

$$A = \begin{bmatrix} 1 & 3 & 4 \\ 5 & -9 & -8 \\ 4 & 7 & 8 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 6 & 7 \\ -1 & -5 & 1 \\ 1 & 9 & 0 \end{bmatrix}$$

Algorithm

// Input: $m \times n$ matrix A

// Output:$m \times n$ matrix in reduced row echelon form and rank of matrix

(a) Create a function which take the matrix A as input

(b) Set i = 1, j = 1, rank = 0

(c) While $i \leq m$ and $j \leq n$ (m=number of rows and n=number of columns)

(d) Find position of maximum absolute value(Pivot element and position) from $A(i, j)$, $i \geq j$.

(e) If pivot is equal to zero(or less than some tolerance value)

i. Set j = j + 1
else

ii. Perform $R_i \longleftrightarrow R_{pivot}$

iii. Divide each element of row i by a_{ij}, thus making the pivot a_{ij} equal to one

iv. For each row k from 1 to m, with $k \neq i$ subtract row i multiplied by a_{kj} from row k.

v. Set $i = i + 1$, $j = j + 1$, $rank = rank + 1$;
end if
end while

(f) Return transformed matrix A and rank.

Solution. Code for the function of rref (r_r_e_f.m)

```
function [B,r]=r_r_e_f(A)
[imax,jmax]=size(A); %dimension of A
i=1; j=1; %loop varriables initialisation
```

```
rank=0;
while i<=imax && j<=jmax
    [p, k] = max(abs(A(i:imax,j))); %searching for pivot element and position
    k=k+i-1;
    if  p<=0.00000000000001 || p==0 %testing for non zero pivot
        j=j+1;
    else
        rank=rank+1;
        C=A(k,:);  %row swaping
        A(k,:)=A(i,:);
        A(i,:)=C;
        A(i,:) = A(i,:)./A(i,j); %making pivot equal to 1
        for k = [1:imax]
            if k~=i
                A(k,:) = A(k,:) - A(k,j).*A(i,:);  %making non pivot entries of
            end
        end
        i = i + 1; %incrementing loop varriable
        j = j + 1;
    end
end
B=A;   % Returning reduced row echelon form of A
r=rank;  %Returning rank of A
```

Code for finding rref form of given matrices:

```
A=[1 3 4;5 -9 -8;4 7 8];
r_r_e_f(A)
A=[1 6 7;-1 -5 1;1 9 0];
r_r_e_f(A)
```

```
ans =

     1     0     0
     0     1     0
     0     0     1


ans =

     1     0     0
     0     1     0
     0     0     1
```

■

2. Solving system using Gauss-Elimination method

Create a code in MATLAB that can be called as a function (name it *Gauss_El.m*) that;

(a) gives the solution of a system of linear equation $Ax = b$ using Gauss-elimination method, where A is a matrix of order $m \times n$, x is the array of n unknowns and b is a $m-$array, check whether the solution of the system $Ax = b$ is inconsistent, infinitely many or unique solution and, compute the solution in the case of unique solution.

(b) call this function in MATLAB to solve for following:

$$A = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 2 & 1 & 1 & 1 \\ 7 & -2 & -1 & -1 \end{bmatrix}, b = \begin{bmatrix} -2 \\ 5 \\ 5 \end{bmatrix} \text{ and } A = \begin{bmatrix} 0 & 2 & 4 & 1 \\ 2 & 0 & 5 & 0 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 3 & 1 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Algorithm
// Input: $m \times n$ matrix A and $m \times 1$ matrix b
// Output: $n \times 1$ matrix x such that $Ax = b$

(a) Create a function which take the matrix A and b as input.

(b) Call the *r_e_f* function (discussed earlier in the class) for the matrix A and get the rank of matrix A.

(c) Call the *r_e_f* function for the augmented matrix $[A \mid b]$ and get the ref form and rank.

(d) Assign first n columns of the augmented matrix to a new matrix C and its last column to a new column vector d (to get equivalent system $Cx = d$).

(e) If ranks of A and $[A \mid b]$ are not equal:

i. return *Inconsistent Solution - No Solution*

(f) If ranks are equal and rank of A is less than number of unknowns:

i. return *Consistent Solution - Infinitely Many Solutions*

(g) Otherwise, there will be a unique solution:

i. define solution vector as $n \times 1$ vector

ii. apply back substitution method and assign the values in solution vector.

A. get the value of last variable.

B. get the value of second last variable by substituting the value of last variable.

C. continue this till you get the value of first variable.

iii. return the solution vector.

Solution. Code for the function of Gauss Elimination Method

```
function sol= Gauss_El(A,b)
n=size(A,2); %number of columns of matrix A
[~,rankA]=r_e_f(A); %rank of matrix A
[B,rankAugA]=r_e_f([A b]); %rank of Augmented matrix [A|b]
C=B(:,[1:end-1]);
d=B(:,end);
if rankA~=rankAugA %rank([A|b])>rank(A), that is, 0=c, where c~=0
    sol='Inconsistent Solution - No Solution';
elseif rankA==rankAugA && rankA<n % there exist non-pivotal unknowns
```

```
    sol='Consistent Solution - Infinitely Many Solutions';
else %rank(A)=n
    sol=zeros(n,1); %define solution vector
    for i = n:-1:1 %every row after n-1 to row 1
        sum=0;
        for j=i+1:n %summing the non-pivotal weighted values of row i
            sum= sum+sol(j)*C(i,j);
        end
        sol(i)=d(i)-sum;
    end
end
```

Code for solving given systems:

```
A=[1 -1 1 -1;2 1 1 1;7 -2 -1 -1];
b=[-2 5 5]';
S1=Gauss_El(A,b)
A=[0 2 4 1;2 0 5 0;1 0 1 2;1 1 3 1];
b=[4 3 2 1]';
S2=Gauss_El(A,b)
```

```
S1 =

    'Consistent Solution - Infinitely Many Solutions'


S2 =

   -4.3333
   -3.6667
    2.3333
    2.0000
```

■

3. Solving system using Gauss-Jordan-Elimination

Create a code in MATLAB that can be called as a function (name it *Gauss_Jord.m*) that;

(a) gives the solution of a system of linear equation $Ax = b$ using Gauss-Jordan Elimination method, where A is a matrix of order $m \times n$, x is the array of n unknowns and b is a $m-$array, check whether the solution of the system $Ax = b$ is inconsistent, infinitely many or unique solution and, compute the solution in the case of unique solution.

(b) call this function in MATLAB to solve for following:

i. $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 1 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$ and $b = \begin{bmatrix} 7 \\ 8 \\ 9 \\ 11 \end{bmatrix}$

ii. $A = \begin{bmatrix} 2 & 1 & 1 & 3 \\ 2 & 2 & 0 & 3 \\ 1 & 1 & -2 & 0 \end{bmatrix}$ and $b = \begin{bmatrix} 10 \\ 11 \\ -1 \end{bmatrix}$

Algorithm

// Input: $m \times n$ matrix A and $m \times 1$ matrix b

// Output: $n \times 1$ matrix x such that $Ax = b$

(a) Create a function which take the matrix A and b as input.

(b) Call the *r_r_e_f* function (discussed earlier in the class) for the matrix A and get the rank of matrix A.

(c) Call the *r_r_e_f* function for the augmented matrix $[A \mid b]$ and get the rref form and rank.

(d) If ranks of A and $[A \mid b]$ are not equal:

i. return *Inconsistent Solution - No Solution*

(e) If ranks are equal and rank of A is less than number of unknowns:

i. return *Consistent Solution - Infinitely Many Solutions*

(f) Otherwise, there will be a unique solution:

i. return the last column of rref of the augmented matrix.

Solution. Code for the function of Gauss-Jordan Elimination Method

```
function sol=Gauss_Jord(A,b)
n=size(A,2); %number of columns of matrix A
[~,rankA]=r_r_e_f(A); %rank of matrix A
[B,rankAugA]=r_r_e_f([A b]); %rank of Augmented matrix [A|b]
if rankA~=rankAugA %rank([A|b])>rank(A), that is, 0=c, where c~=0
    sol='Inconsistent Solution - No Solution';
elseif rankA==rankAugA && rankA<n % there exist non-pivotal unknowns
    sol='Consistent Solution - Infinitely Many Solutions';
else %rank(A)=n
    sol=B(:,end);
end
```

Code for solving given systems:

```
A=[1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16];
b=[7 8 9 11]';
S3=Gauss_Jord(A,b)
A=[1 2 3 4;3 -1 5 -7;-1 -2 -4 2;6 5 15 -4];
b=[8 -16 -2 0]';
S4=Gauss_Jord(A,b)
```

```
S3 =

    'Inconsistent Solution - No Solution'


S4 =

    1.2381
    5.0476
   -2.0000
    0.6667
```

■