Experiment:5
Solving systems using Iterative methods

1. Vector and Matrix Norms

(a) Find the $||.||_1, ||.||_2, ||.||_3$ and $||.||_\infty$ norms of the following vectors

i. $v_1 = (1, 2, 3)$

ii. $v_2 = (-4, 5, 6)$

(b) Find the $||.||_1, ||.||_2$ and $||.||_\infty$ norm of the Matrix

$$A = \begin{bmatrix} \frac{53}{360} & -\frac{13}{90} & \frac{23}{360} \\ -\frac{11}{180} & \frac{1}{45} & \frac{19}{180} \\ -\frac{7}{360} & \frac{17}{90} & -\frac{37}{360} \end{bmatrix}$$

Solution.

```
v1=[1 2 3];
v2=[-4 5 6];
b1=norm(v1,1)
```

```
b1 = 6
```

```
c1=norm(v1,2)
```

```
c1 = 3.7417
```

```
d1=norm(v1,3)
```

```
d1 = 3.3019
```

```
e1=norm(v1,"inf")
```

```
e1 = 3
```

```
b2=norm(v2,1)
```

```
b2 = 15
```

```
c2=norm(v2,2)
```

```
c2 = 8.7750
```

```
d2=norm(v2,3)
```

```
d2 = 7.3986
```

```
e2=norm(v2,"inf")
```

```
e2 = 6
```

```
A=[53/360 13/90 23/360; -11/180 1/45 19/180; -7/360 17/90 -37/360];
l_1=norm(A,1)
```

```
l_1 = 0.3556
```

```
l_2=norm(A,2)
```

```
l_2 = 0.2543
```

```
l_i=norm(A,"inf")
```

```
l_i = 0.3556
```

■

2. Iterative Method(s) for solving system of equations

Creates code in MATLAB that can be called as a function(s) (name it *Gauss_Jacobi.m* and *Gauss_Seidel.m*) that gives the computational solution of system(s) for given tolerance level, and the no. of iterations. Solve the following system(s) of equations using both the above function(s) and check which method converges faster.

(a) Using Jacobi method and Guass Seidel method find approximating solution with initial guess as $x^{(0)} = (0, 0, 0)^T$ and tolerance 0.00001.

$$\begin{aligned} 5x - y + z &= 10 \\ 2x + 4y &= 12 \\ x + y + 5z &= -1 \end{aligned}$$

(b) Using Jacobi method and Guass Seidel method find approximating solution with initial guess as $x^{(0)} = (0, 0, 0, 0)^T$ and tolerance 0.001

$$\begin{aligned} 29x + 2y + z &= 7 \\ 2x + 6y + z &= 5 \\ x + y + \frac{z}{5} &= 4 \end{aligned}$$

Algorithm for Gauss-Jacobi method

// Input: a $m \times n$ matrix A, $n \times 1$ column vector b, tol, N(Number of iterations), $x^{(0)}$(initial guess)

// Output: Jacobi method won't be applied or $n \times 1$ column vector along with number of iterations consisting computational solution of the system

(a) Create a function which take the matrix A, b, tol, N, $x^{(0)}$ as input

(b) Evaluate size of matrix A and store it in variables m, n

(c) Define zero matrices D, L & U of order $m \times n$ % for creating A=D+L+U, D is diagonal matrix, L is strictly lower triangular matrix and U is strictly upper triangular matrix

(d) `If` a given matrix A is a square matrix

i. `for` $i < j$ store the values of matrix A to the matrix U;
`end for`

ii. `for` $i > j$ store the values of matrix A to the matrix L;
`end for`

iii. `for` $i = j$ store the values of matrix A to the matrix D;
`end for`

(e) Set $s = 1$;

(f) `While` $s <= N$ % for running iterations

i. $x^{(k+1)} = D^{-1}(b - (L + U)x^{(k)})$

ii. Define $||x^{(k+1)} - x^{(k)}||_\infty$

iii. `if` $||x^{(k+1)} - x^{(k)}||_\infty$ is less than tol

iv. `Print` the solutions and no. of iterations

v. `return`;
`end if`

vi. Set $s = s + 1$

vii. Repeat the above steps (i)-(vi)

`end while`

(g) `Print` "Method fails to converge to given tol in given number of iterations"

(h) `else Print` "Jacobi method will not work for given system"
`end if`

Algorithm for Gauss Seidel method

// Input: a $m \times n$ matrix A, $n \times 1$ column vector b, tol, N(Number of iterations), $x^{(0)}$(initial guess)

// Output: Jacobi method won't be applied or $n \times 1$ column vector along with number of iterations consisting computational solution of the system

(a) Create a function which take the matrix A, b, tol, N, $x^{(0)}$ as input

(b) Evaluate size of matrix A and store it in variables m, n

(c) Define zero matrices D, L & U of order $m \times n$ % for creating A=D+L+U, D is diagonal matrix, L is strictly lower triangular matrix and U is strictly upper triangular matrix

(d) `If` a given matrix A is a square matrix

i. `for` $i < j$ store the values of matrix A to the matrix U;
`end for`

ii. `for` $i > j$ store the values of matrix A to the matrix L;
`end for`

iii. `for` $i = j$ store the values of matrix A to the matrix D;
`end for`

(e) Set $s = 1$;

(f) `While` $s <= N$ % for running iterations

i. $x^{(k+1)} = (D + L)^{-1}(b - Ux^{(k)})$

ii. Define $||x^{(k+1)} - x^{(k)}||_\infty$

iii. `if` $||x^{(k+1)} - x^{(k)}||_\infty$ is less than tol

iv. `Print` the solutions and no. of iterations

v. `return`;
`end if`

vi. Set $s = s + 1$

vii. Repeat the above steps (i)-(vi)

`end while`

(g) `Print` "Method fails to converge to given tol in given number of iterations"

(h) `else Print` "Gauss Seidel method will not work for given system"
`end if`

Solution. Code for the function of Gauss Jacobi (Gauss_Jacobi.m)

```
function Y=Gauss_Jacobi(A,b,tol,N,x0)
[n,m]=size(A);
D=[];
L=zeros(n,m);
U=zeros(n,m);
if m==n %for checking rows and columns are equal
for i=1:n %for creating strictly Upper triangular matrix
    for j=i+1:m
        U(i,j)=A(i,j);
    end
end
%disp(U);
for k=1:m %for creating strictly lower triangular matrix
    for l=k+1:n
        L(l,k)=A(l,k);
    end
end
%disp(L);
for p=1:n %for creating strictly Diagonal matrix
    D(p,p)=A(p,p);
end
%disp(D);
s=1;
while s<=N %for running iterations
    x1=inv(D)*(b-(L+U)*x0);
    norm= max(abs(x1-x0));
        if norm<tol
            sol=x1;
            fprintf('Method converges to given tol in %d no. of iterations',s)
            fprintf(' \n and the sol is ')
            Y=sol;
            return;
        end
    s=s+1;
    x0=x1;
end
 fprintf('Method fails to converge to given tol after %d no. of iterations',N)
else
    disp("Jacobi method will not work as no. of equations" + ...
        " and variables are not same")
end
```

Code for the function of Gauss Seidel (Gauss_Seidel.m)

```
function Y=Gauss_Seidel(A,b,tol,N,x0)
[n,m]=size(A);
D=[];
L=zeros(n,m);
U=zeros(n,m);
if m==n %for checking rows and columns are equal
for i=1:n %for creating strictly Upper triangular matrix
    for j=i+1:m
        U(i,j)=A(i,j);
    end
end
%disp(U);
for k=1:m %for creating strictly lower triangular matrix
    for l=k+1:n
        L(l,k)=A(l,k);
    end
end
%disp(L);
for p=1:n %for creating strictly Diagonal matrix
    D(p,p)=A(p,p);
end
%disp(D);
s=1;
while s<=N %for running iterations
    x1=inv(D+L)*(b-U*x0);
    norm= max(abs(x1-x0));
        if norm<tol
            sol=x1;
            fprintf('Gauss Seidel Method converges to given tol in %d no. of iter
            fprintf(' \n and the sol is ')
            Y=sol;
            return;
        end
    s=s+1;
    x0=x1;
end
 fprintf('Gauss Seidel Method fails to converge to given tol after %d no. of iter
else
    disp("Gauss Seidel method will not work as no. of equations" + ...
        " and variables are not same")
end
```

a)

```
A=[5 -1 1;2 4 0;1 1 5];
b=[10 12 -1]';
x0=[0,0,0]';
C=Gauss_Jacobi(A,b,0.00001,1000,x0)
```

```
Jacobi method converges to given tol in 12 no. of iterations
 and the sol is
C = 3x1
    2.5556
    1.7222
   -1.0556
```

```
A=[5 -1 1;2 4 0;1 1 5];
b=[10 12 -1]';
x0=[0,0,0]';
D=Gauss_Seidel(A,b,0.00001,1000,x0)
```

```
 Gauss Seidel Method converges to given tol in 7 no. of iterations
 and the sol is
D = 3x1
    2.5556
    1.7222
   -1.0556
```

b)

```
A=[29 2 1;2 6 1;1 1 1/5];
b=[7 5 4]';
x0=[0,0,0]';
D=Gauss_Jacobi(A,b,0.00001,100000,x0)
```

```
Jacobi method fails to converge to given tol after 100000 no. of iterations
```

```
A=[29 2 1;2 6 1;1 1 1/5];
b=[7 5 4]';
x0=[0,0,0]';
D=Gauss_Seidel(A,b,0.00001,1000,x0)
```

```
 Gauss Seidel Method converges to given tol in 149 no. of iterations
 and the sol is
```

```
D = 3x1
  -3.8667
  -26.6000
  172.3332
```

■