Name: Kamal Giri
Homework3

Qn.2:
Recursive Bisection Method:

Filename: recursiveBisection.m

```
function[xRoot, xFRoot, noOfIter0] = recursiveBisection(f, xL,xR, noOfIterI)

xM = (xL +xR)/2;
fxM = feval(f, xM);

if abs(fxM) < 10^(-6)
    xRoot = xM;
    xFRoot = fxM;
    noOfIter0 = noOfIterI;
    return;
else
    fxL = feval(f, xL);
    if (fxL * fxM) < 0
        xR = xM;
    else
        xL = xM;
    end
    noOfIterI = noOfIterI + 1;
    [xRoot, xFRoot, noOfIter0] = recursiveBisection(f, xL,xR, noOfIterI);

end
```

scriptfile: scriptBisectionRecursive.m

```
clear all;
format long;

f = @(x) x^2-7;

[root, fAtRoot, noOfIter0] = recursiveBisection(f,1,3,1);

root
fAtRoot
noOfIter0
```

Output:

scriptBisectionRecursive

root =

2.645751476287842

fAtRoot =

8.742792942939559e-07

noOfIter0 =

22

Qn.3:

Jacobi Method:

Main file : jacobiMethod.m

```
function[result] =  jacobiMethod(A, b, x0, maxIter, n)

x1 = x0;

disp('       Iteration                x                   y
z                   w' )


for k= 1:maxIter
    for i = 1:n
        s = 0;
        for j = 1 : (i-1)
            s =  s + A(i,j)* x0(j);
        end
        for j = (i+1):n
            s = s + A(i*j) * x0(j);
        end
        x1(i) = (b(i)-s)/A(i,i);

    end

    res = A * x1 - b;

    if norm(res)< 10^(-6)
        break;
```

```
    end

    x0 = x1;
    if(k<5)
        print = [k,  x1'];

        disp(print)
    end
    %to graph
    yaxis(k) = norm(res);
    xaxis(k) = k;
end
figure
plot(xaxis, log(yaxis), '-o');

title("Norm of the residual versus iteration number");
xlabel("Number of iteration");
ylabel('Norm of the residual');
legend('residual vs iter line');

result = x1';
```

scriptfile: scriptJacobimethod.m

```
clear all;
format long;

A = [5 1 1 1 ;1 -7 2 2; 2 1 6 1; 1 -1 1 5];
b = [7;-4;1;9];
x0 = [0;0;0;0];
maxIter = 10;
n = 4;

[soln] = jacobiMethod(A,b, x0, maxIter,n);

soln
```

Output:

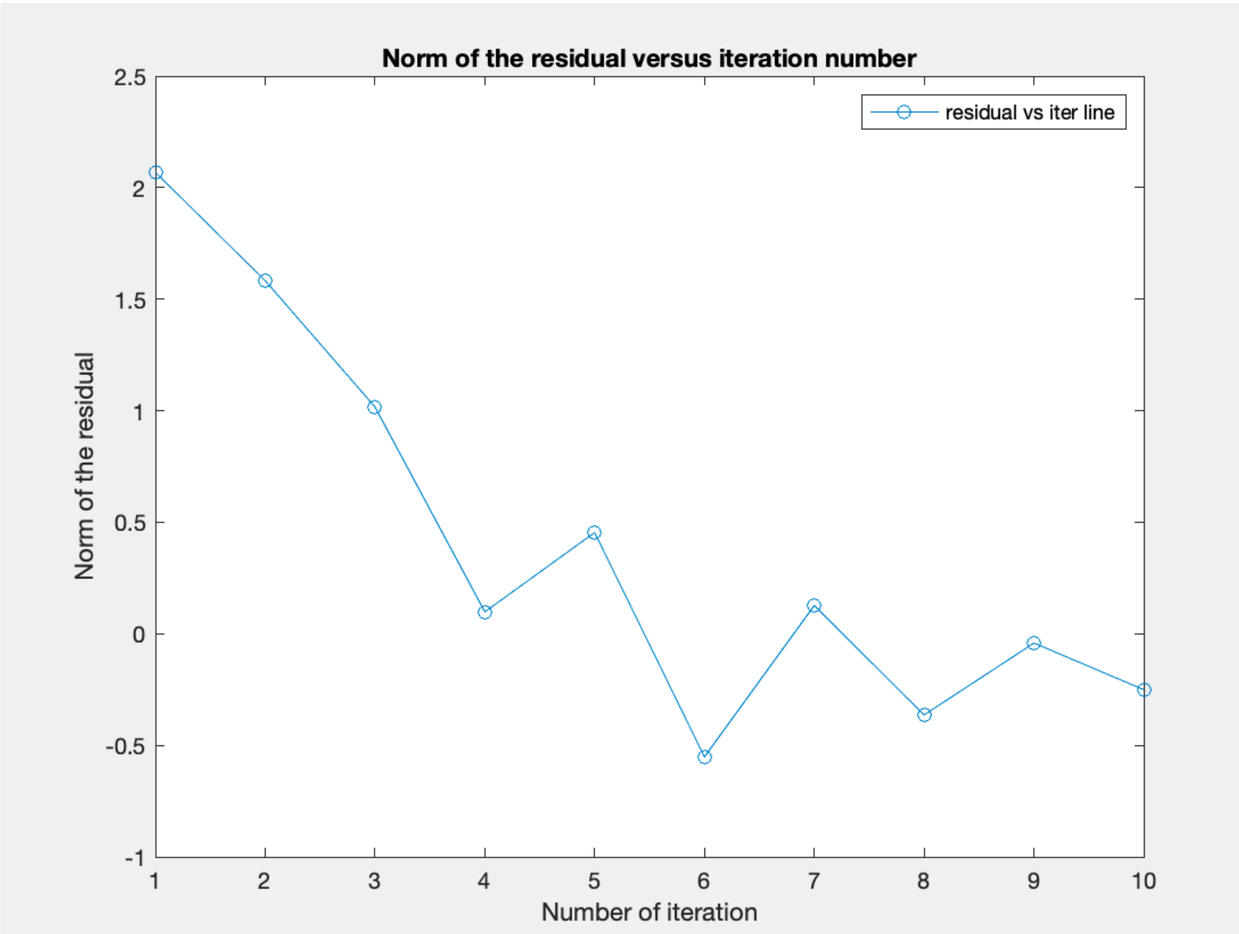
scriptJacobiMethod

Iteration	x	y	z	w
1.000000000000000	1.400000000000000	0.571428571428571	0.166666666666667	1.800000000000000
2.000000000000000	0.859047619047619	0.347619047619048	-0.695238095238095	1.600952380952381
3.000000000000000	1.288380952380952	1.160680272108843	-0.444444444444444	1.836761904761905
4.000000000000000	0.978289342403628	0.937532879818594	-0.762367346938775	1.863348752834467
5.000000000000000	1.144770612244898	1.207358859734370	-0.626243386243386	1.944322176870748

soln =

   1.052959477196202   1.138354243375540 -0.711396994270753   1.956607544058489

Figure:



Qno.4

Gauss-Seidel Method:

MainCode: gaussMethod.m

```
function[finalSolution] = gaussMethod(A,b ,x0, maxIter, n)
x1 =x0;
```

```
disp('       Iteration                x                  y
z                  w' )

for k = 1: maxIter
    x1_prev = x1;
    for i = 1:n
        x1(i) = (b(i)- A(i, 1:i-1)*x1(1:i-1)-
A(i,i+1:n)*x1_prev(i+1:n))/A(i,i);
    end

    res =  x1- x1_prev;
    if norm(res) < 10^(-6)
        break;
    end
    if(k<=5)
        x = [k, x1'];
        disp(x);
    end

    yaxis(k) = norm(res);
    xaxis(k) = k;


end


figure
plot(xaxis, log(yaxis), '-o');

title("Norm of the residual versus iteration number");
xlabel("Number of iteration");
ylabel('Norm of the residual');
legend('residual vs iter line');

 finalSolution = x1';
```

scriptFile: sciptgaussmethod.m

```
clear all;
format long;

A = [5 1 1 1 ;1 -7 2 2; 2 1 6 1; 1 -1 1 5];
b = [7;-4;1;9];
x0 = [0;0;0;0];
maxIter = 100;
n = 4;

[finalSolution] = gaussMethod(A,b, x0, maxIter,n);

finalSolution
```

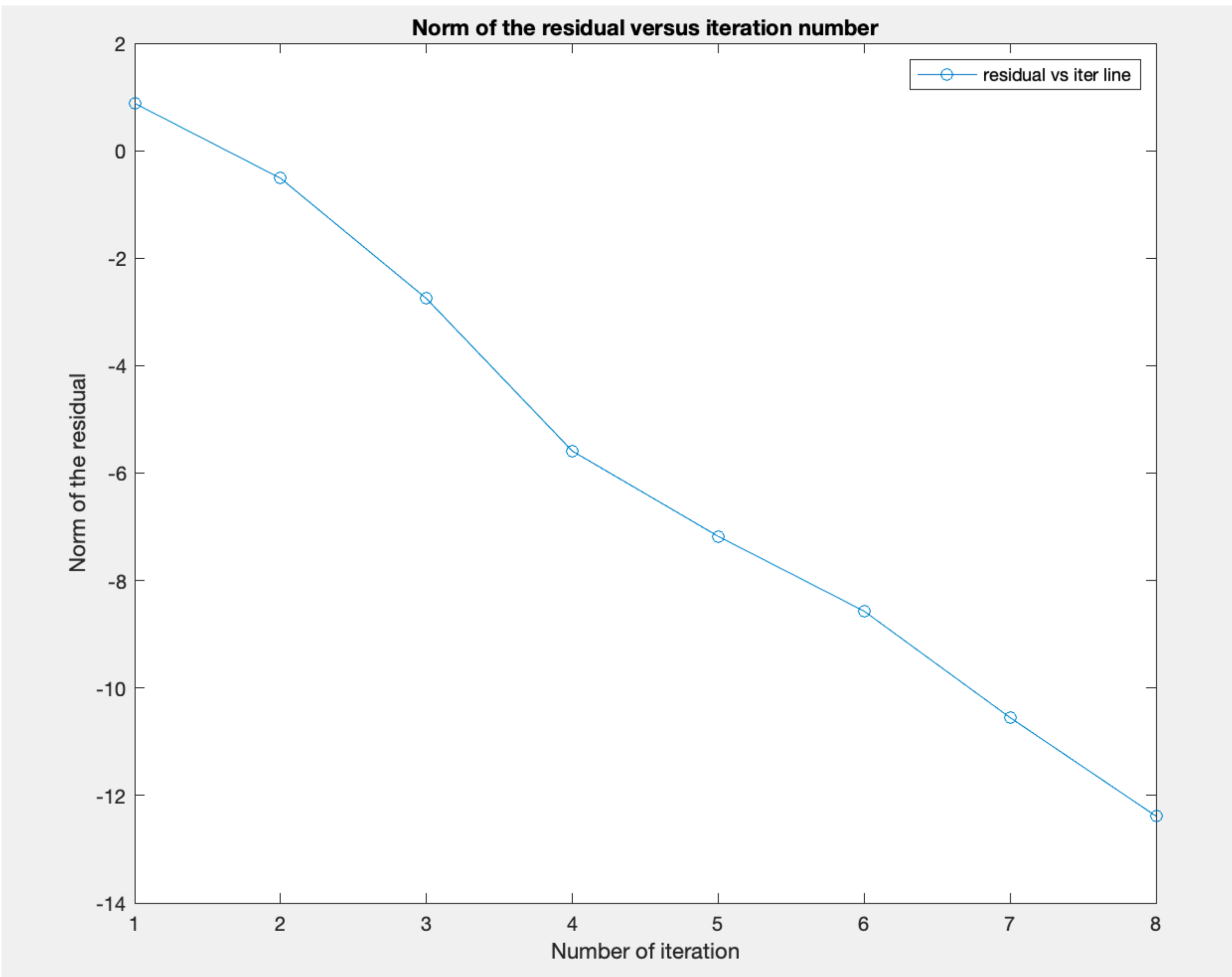
Output:

scriptgausssidel

```
       Iteration                x                   y                   z                    w
   1.000000000000000   1.400000000000000   0.771428571428571  -0.428571428571429   1.760000000000000

   2.000000000000000   0.979428571428571   1.091755102040816  -0.635102040816326   1.949485714285714

   3.000000000000000   0.918772244897959   1.078219941690962  -0.644208357628766   1.960731210884354

   4.000000000000000   0.921051441010690   1.079156735360266  -0.646998471377667   1.961020753145448

   5.000000000000000   0.921364196574390   1.078486965729993  -0.647039352004037   1.960832424231928
```

```
finalSolution =

   0.921568571581403   1.078431381698982  -0.647058830612082   1.960784328145933
```

Figure:



Problem 5:

Spectral radii for Jacob Method:

Code:

spectralradiiJacob.m

```
clear all;


A = [5 1 1 1 ;1 -7 2 2; 2 1 6 1; 1 -1 1 5];

Da = diag(diag(A));
La = tril(A) - Da;
Ua = triu(A) - Da;
GJ = -inv(Da) * (Ua +La);

%spectral radii
rhoGJ = max(abs(eig(GJ)));
spectralRadii = rhoGJ
```

Output:

```
>> spectralradiiJacobi

spectralRadii =

  0.324019645579333
```

Spectral Radii for Gauss-Sidel Method

Code:

```
clear all;
A = [5 1 1 1 ;1 -7 2 2; 2 1 6 1; 1 -1 1 5];

spectralRadii = max(abs(eig(tril(A)\ (-triu(A,1)))))
```

Output:

spectralRadii =

0.324019645579333

```
clear all;
A = [5 1 1 1 ;1 -7 2 2; 2 1 6 1; 1 -1 1 5];

spectralRadii = max(abs(eig(tril(A)\ (-triu(A,1)))))

Output:

spectralRadii =

   0.125960819158416
```

Spectral radius provides useful information on whether the iterative methods converges or not. Specially, if the spectral radius is less than 1 then it conforms that the method will converge. This is also proved in the above problems which has spectral radius less than 1. Hence, they are converging.

Yes, looking at the spectral radii, we can conclude that gauss seidal method will converge first as its decreasing quickly than Jacob Method.

Gauss-Seidel will converge fast, as we have seen in the above figures that Gauss was converging in just 8 iterations where it is taking 10 iteration for the Jacob to converge.