Linear Systems

Given:

A=[5 1 2 ; -3 9 4; 1 2 -7]

X=[x1;x2;x3]

B=[10 -14 33]

Aim:

To solve the given system using Jacobi, Gauss Siedel and SOR

And determine the following;

1. Eigenvalues of the iteration matrix.
2. Spectral radius of iteration matrix.
3. Solve the system using Jacobi, Gauss-seidel and SOR methods
4. Compute the iteration Matrix for the above mentioned iterative methods.
5. Increase the Spectral radius to 1.1 by changing the diagonal terms suitably.

Solution:

In matrix form, the given system can be written of the form;

$$Ax=B$$

Where,

A can be represented as D+L+U Or A=D-L-U

D = Diagonal Matrix

L = -(Lower Diagonal Matrix)

U = -(Upper Diagonal Matrix)

Jacobi Iteration Method:

- In the Jacobi method, a guess value for x is considered.
- Let it be (x1 x2 x3) = 0.

- The same value is applied to all the 3 system of equations to extract the value of unknown x. The obtained value of x is further used in subsequent iterations in a similar manner as previously done.
- In short, the latest updated value is used in immediate iteration only.
- The value of unknown is obtained by iterating over the below matrix,

$$\mathbf{x} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b}$$

- The iteration matrix is of the form,

$$\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$$

Gauss Siedel Method:

- In GS method, a guess value for x is considered.
- Let it be (x1 x2 x3) = 0.
- This value is applied to the first equation of the system and then the new value obtained is utilised in remaining equations.
- In short, the latest updated value is used in immediate computation itself.
- The value of unknown is obtained by iterating over the below matrix,

$$\mathbf{x} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}\mathbf{x} + (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}$$

- The iteration matrix is of the form,

$$(\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}$$

Successive Over Relaxation:

- This is just an extension of the GS method with a modifiable convergence rate based on a factor called the relaxation factor w(omega).
- When w<1, under relaxation, This is used when there is no convergence using GS method.
- When w>1, over relaxation, This is used when there is a need to accelerate the rate of convergence.
- When w=1, GS method.
- The value of unknown is obtained by iterating over the below matrix,

$$\mathbf{x}^{(k+1)} = (\mathbf{D} - \omega\mathbf{L})^{-1}((1-\omega)\mathbf{D} + \omega\mathbf{U})\mathbf{x}^{(k)} + (\mathbf{D} - \omega\mathbf{L})^{-1}\omega\mathbf{b}$$

- The iteration matrix is of the form,

$$(\mathbf{D}-\omega\mathbf{L})^{-1}\left[(1-\omega)\mathbf{D}+\omega\mathbf{U}\right]$$

Eigen Values:

Eigen Values are the deterministic roots of a matrix.

They are represented using the symbol lambda λ. There can be more number of eigen values than the number of equations.

Spectral Radius:

Maximum of all the absolute value of eigen values of a matrix is the spectral radius.

Program:

Main Program:

```
clear all
close all
clc

%Given linear equations in Matrix form
a=[5 1 2; -3 9 4; 1 2 -7];
B=[10; -14; 33];

%LU decomposition A=D-L-U
L=[0 0 0; 3 0 0; -1 -2 0];%negative of Lower Diagonal Matrix
U=[0 -1 -2; 0 0 -4; 0 0 0];%negative of Upper Diagonal Matrix
D=[5 0 0; 0 9 0; 0 0 -7];%Main Diagonal Matrix

%Relaxation Factor for SOR
w=0.9;

%iteration matrix of jacobi,gs,sor
Tjac=inv(D)*(L+U);
Tgs=inv(D-L)*U;
Tsor=inv(D-(w*L))*(((1-w)*D)+(w*U));

%for verification
Spectral_radius_test1_Tjac=max(abs(eig(Tjac)));
Spectral_radius_test2_Tgs=max(abs(eig(Tgs)));
```

```
Spectral_radius_test3_Tsor=max(abs(eig(Tsor)));

%manual calculation to find the eigen values to extract spectral radius
%Assigning a symbolic variable 'x' to find the eigen value lambda
syms x
%Identity matrix
I=[x 0 0; 0 x 0; 0 0 x];

%to find eigen value: Det(A-x*I)=0
eigenTjac=Tjac-I;
eigenTgs=Tgs-I;
eigenTsor=Tsor-I;

%Determinant
roots_jac=eigenTjac(1,1)*(eigenTjac(2,2)*eigenTjac(3,3)-eigenTjac(3,2)*eigen
Tjac(2,3))-eigenTjac(1,2)*(eigenTjac(2,1)*eigenTjac(3,3)-eigenTjac(3,1)*eige
nTjac(2,3))+eigenTjac(1,3)*(eigenTjac(2,1)*eigenTjac(3,2)-eigenTjac(3,1)*eig
enTjac(2,2));
roots_gs=eigenTgs(1,1)*(eigenTgs(2,2)*eigenTgs(3,3)-eigenTgs(3,2)*eigenTgs(2
,3))-eigenTgs(1,2)*(eigenTgs(2,1)*eigenTgs(3,3)-eigenTgs(3,1)*eigenTgs(2,3))
+eigenTgs(1,3)*(eigenTgs(2,1)*eigenTgs(3,2)-eigenTgs(3,1)*eigenTgs(2,2));
roots_sor=eigenTsor(1,1)*(eigenTsor(2,2)*eigenTsor(3,3)-eigenTsor(3,2)*eigen
Tsor(2,3))-eigenTsor(1,2)*(eigenTsor(2,1)*eigenTsor(3,3)-eigenTsor(3,1)*eige
nTsor(2,3))+eigenTsor(1,3)*(eigenTsor(2,1)*eigenTsor(3,2)-eigenTsor(3,1)*eig
enTsor(2,2));

%Simplication of eqn
simp1=simplify(roots_jac);
simp2=simplify(roots_gs);
simp3=simplify(roots_sor);

%solving the cubic equation
f1=solve(simp1);
f2=solve(simp2);
f3=solve(simp3);

%extraction of roots from the cubic eqn
Zroots1 = vpa(f1);
Zroots2= vpa(f2);
Zroots3 = vpa(f3);
eigen_values1 = double(Zroots1); %Reduction of digits after decimal
eigen_values2 = double(Zroots2);
eigen_values3 = double(Zroots3);
```

```
% calculated spectral radius
Spectral_radius_calc_Tjac= max(abs(eigen_values1));
Spectral_radius_calc_Tgs= max(abs(eigen_values2));
Spectral_radius_calc_Tsor= max(abs(eigen_values3));

% %solving using jacobi method
error=9e3;
tol=1e-3;
x=[0;0;0];

magfac=0.1:0.1:2;
for i=1:length(magfac)
[jac_iter,jac_iternew(i),new_specrad_value_jac(i)]=jacobi(Tjac,L,U,D,B,magfac(
i));
[gs_iter,gs_iternew(i),new_specrad_value_gs(i)]=gs(Tgs,L,U,D,B,magfac(i));
[sor_iter,sor_iternew(i),new_specrad_value_sor(i)]=sor(Tsor,L,U,D,B,magfac(i))
;
end

figure(1)
plot(new_specrad_value_jac,magfac,'*-');
hold on
plot(new_specrad_value_gs,magfac,'*-');
hold on
plot(new_specrad_value_sor,magfac,'*-');
title('Magnification Factor Vs Spectral Radius');
xlabel('Spectral Radius');
ylabel('Mag. Fac');
legend('Jacobi','GS','SOR');

figure(2)
plot(new_specrad_value_jac,jac_iternew);
hold on
plot(new_specrad_value_gs,gs_iternew);
hold on
plot(new_specrad_value_sor,sor_iternew);
title('Spectral Radius Vs Iterations');
xlabel('Spectral Radius');
ylabel('Iterations');
legend('Jacobi','GS','SOR');

  figure(3)
  plot(new_specrad_value_jac,jac_iternew,'*-');
  title('Spectral Radius Vs Iterations');
  xlabel('Spectral Radius');
  ylabel('Iterations');
  legend('Jacobi');
```

```
figure(4)
plot(new_specrad_value_gs,gs_iternew,'*-');
title('Spectral Radius Vs Iterations');
xlabel('Spectral Radius');
ylabel('Iterations');
legend('GS');

figure(5)
plot(new_specrad_value_sor,sor_iternew,'*-');
title('Spectral Radius Vs Iterations');
xlabel('Spectral Radius');
ylabel('Iterations');
legend('SOR');
```

Function for Jacobi:

```
function
[jac_iter,jac_iternew,new_specrad_value_jac]=jacobi(Tjac,L,U,D,B,magfac)

% %solving using jacobi method
error=9e3;
tol=1e-3;
x=[0;0;0];
xold=x;
jac_iter=0;
while error>tol
   x_jac=(Tjac*xold)+(inv(D)*B);
   error=max(abs(x_jac-xold));
   xold=x_jac;
   jac_iter=jac_iter+1;
end
disp(jac_iter);

%solving using jacobi method with diagonal mag.
newD=magfac*D; %magnifying diagonal terms
newTjac=inv(newD)*(L+U);  %new iteration matrix of jacobi
new_specrad_value_jac=max(abs(eig(newTjac)));
   x_new=[0;0;0];
   xold1=x_new;
   error=9e3;
   tol=1e-3;
   jac_iternew=0;
   while error>tol
       x_jac_new=(newTjac*xold1)+(inv(newD)*B);
```

```
        error=max(abs(x_jac_new-xold1));
        xold1=x_jac_new;
        jac_iternew=jac_iternew+1;
    end
    disp(jac_iternew);
end
```

Function for GS:

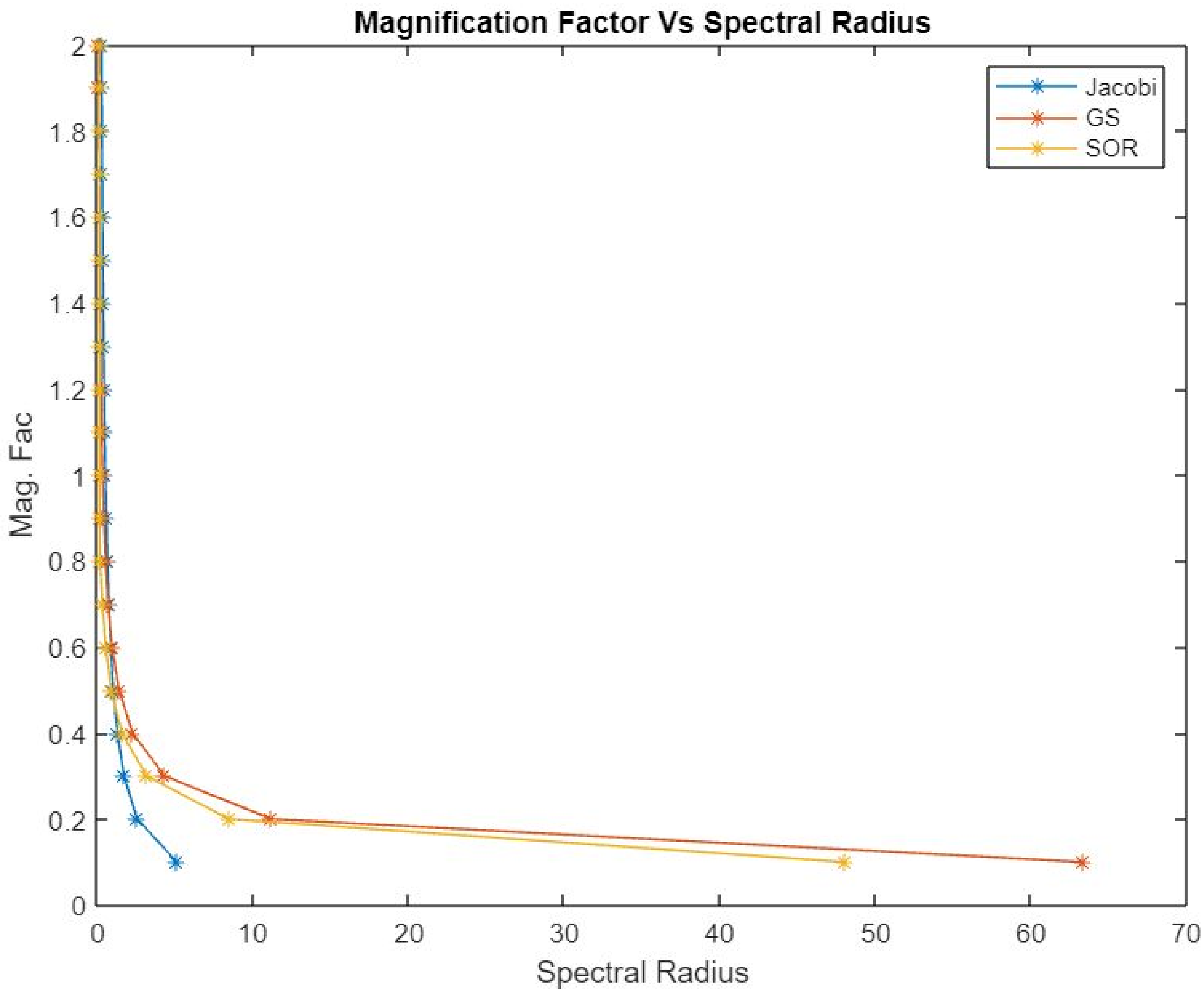
```
function [gs_iter,gs_iternew,new_specrad_value_gs]=gs(Tgs,L,U,D,B,magfac)

% %solving using gs method
error=9e3;
tol=1e-3;
x=[0;0;0];
xold=x;
gs_iter=0;
while error>tol
    x_gs=(Tgs*xold)+(inv(D-L)*B);
    error=max(abs(x_gs-xold));
    xold=x_gs;
    gs_iter=gs_iter+1;
end
disp(gs_iter);

%solving using gs method with diagonal mag.
newD=magfac*D; %magnifying diagonal terms
newTgs=inv(newD-L)*U;  %new iteration matrix of gs
new_specrad_value_gs=max(abs(eig(newTgs)));
x_new=[0;0;0];
xold1=x_new;
error=9e3;
tol=1e-3;
gs_iternew=0;
while error>tol
    x_gs_new=(newTgs*xold1)+(inv(newD-L)*B);
    error=max(abs(x_gs_new-xold1));
    xold1=x_gs_new;
    gs_iternew=gs_iternew+1;
end
disp(gs_iternew);
end
```

Function for SOR:

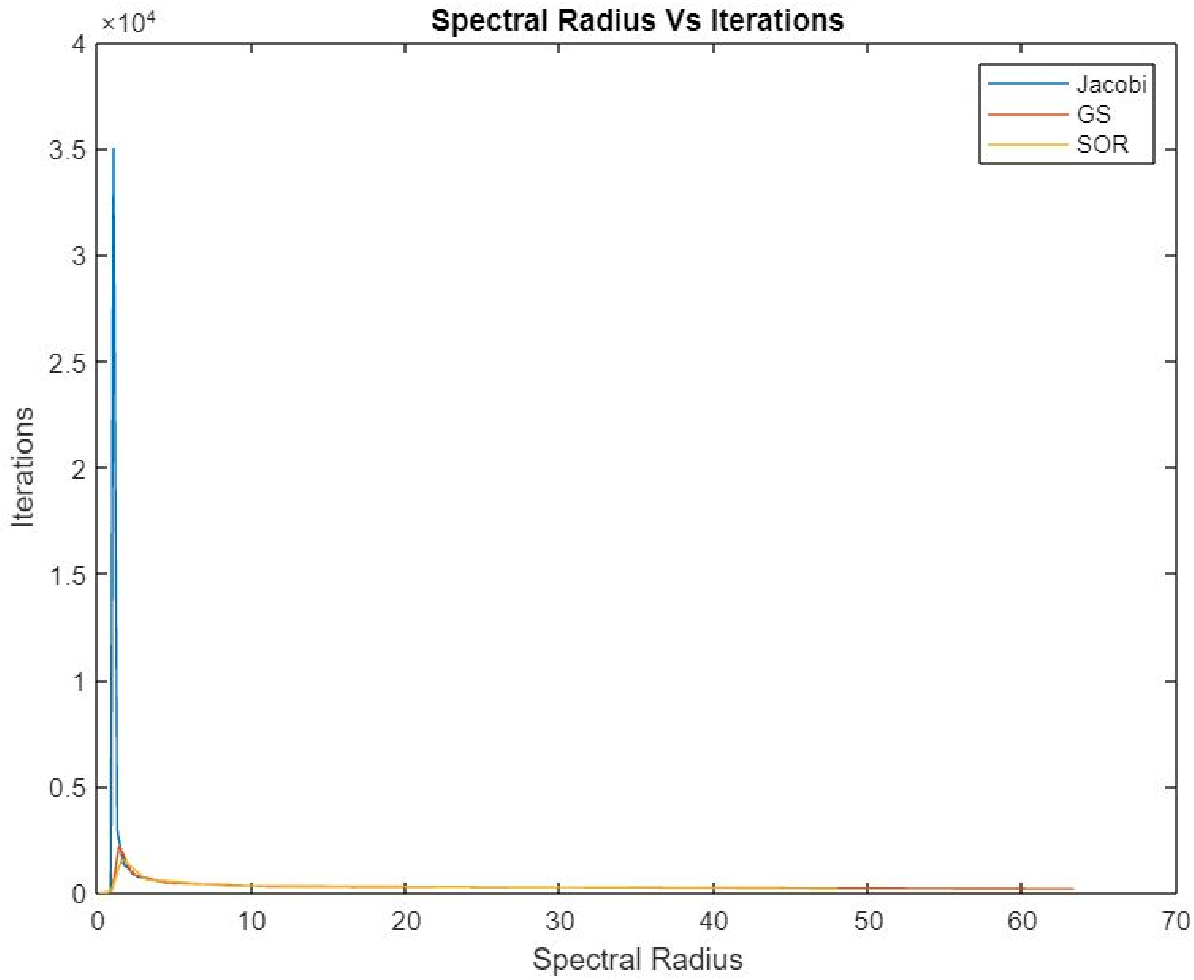
```
function
[sor_iter,sor_iternew,new_specrad_value_sor]=sor(Tsor,L,U,D,B,magfac)

% %solving using sor method
```

```
error=9e3;
tol=1e-3;
x=[0;0;0];
xold=x;
sor_iter=0;

%Relaxation Factor for SOR
w=0.9;

while error>tol
   P=inv(D-(w*L));
   Q=((1-w)*D)+(w*U);
   R=inv(D-(w*L))*w*B;
   x_sor=(P*Q*xold)+R;
   %x_sor=inv(D-(w*L))*((1-w)*D+(w*U))*xold+inv(D-w*L)*w*B;
   error=max(abs(x_sor-xold));
   xold=x_sor;
   sor_iter=sor_iter+1;
end
disp(sor_iter);

%solving using sor method with diagonal mag.
newD=magfac*D; %magnifying diagonal terms
newTsor=inv(newD-(w*L))*(((1-w)*newD)+(w*U));  %new iteration matrix of sor
new_specrad_value_sor=max(abs(eig(newTsor)));
x_new=[0;0;0];
xold1=x_new;
error=9e3;
tol=1e-3;
sor_iternew=0;

while error>tol

x_sor_new=(inv(newD-(w*L))*(((1-w)*newD)+(w*U))*xold1)+(inv(newD-(w*L))*w*B)
;
   error=max(abs(x_sor_new-xold1));
   xold1=x_sor_new;
   sor_iternew=sor_iternew+1;
end
disp(sor_iternew);

end
```

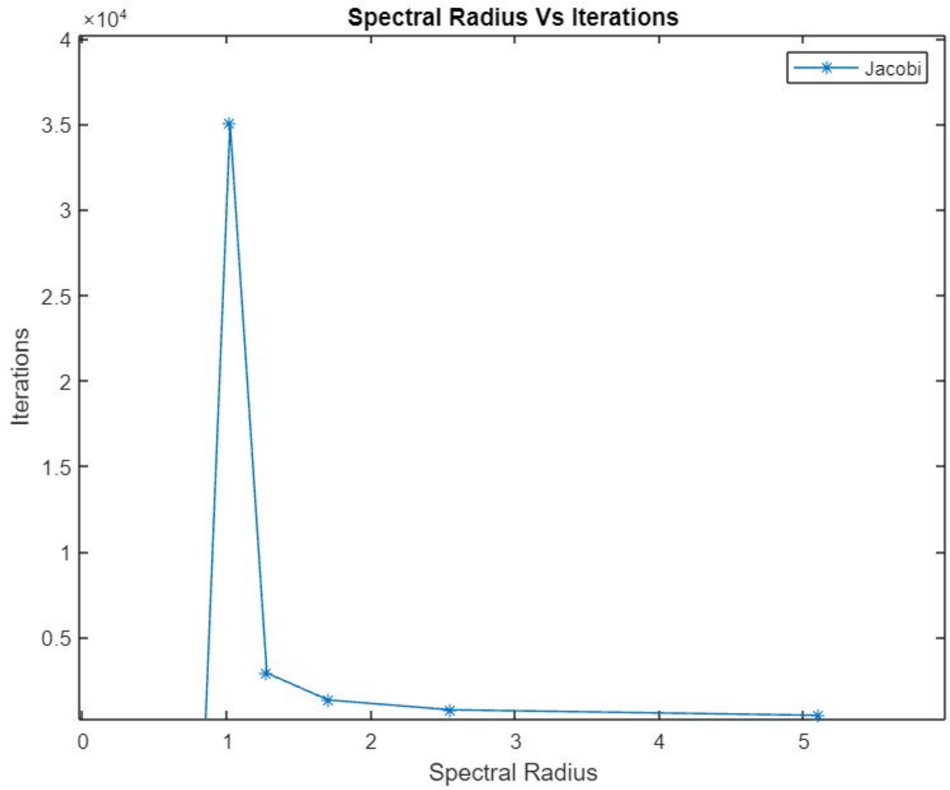
Results:

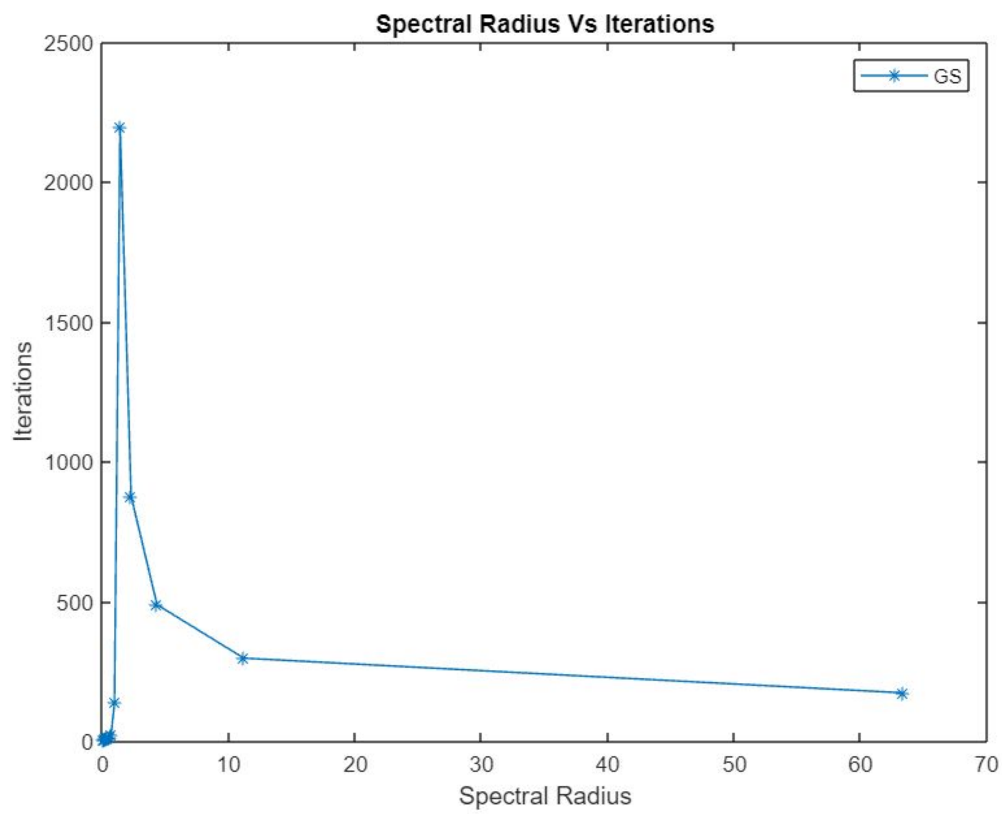


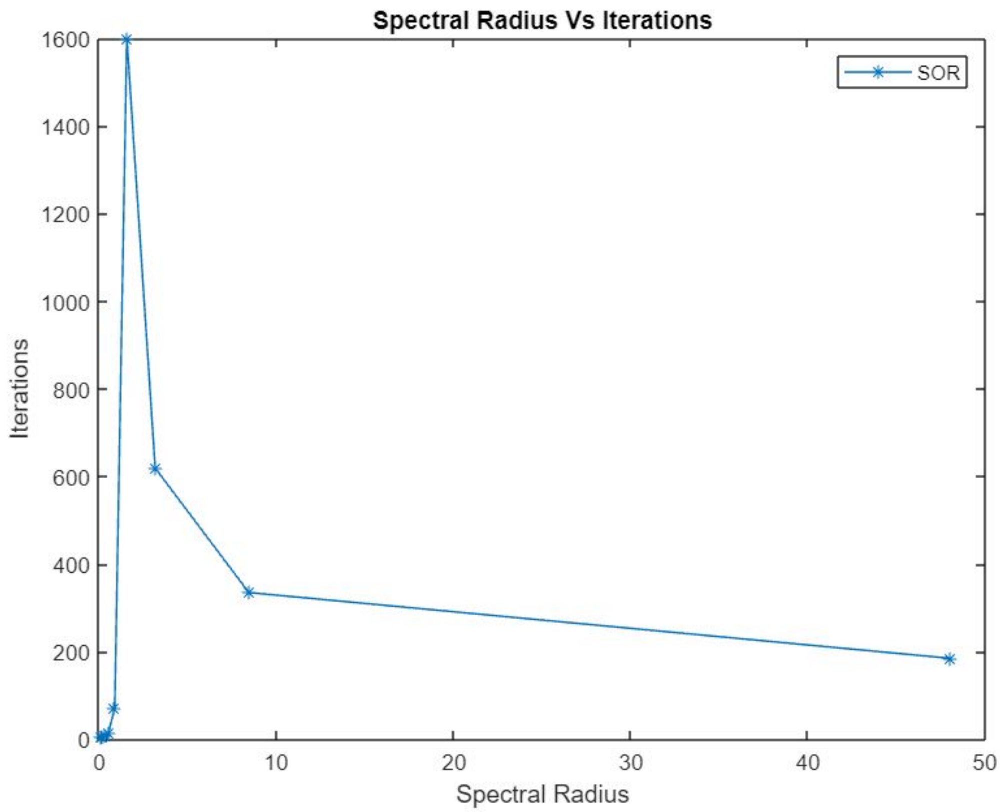
- As the Magnification factor increases, Spectral Radius decreases.
- This is more significant in GS and SOR whereas it is relatively less in Jacobi.



- As the spectral radius decreases, Convergence is faster (i.e., iterations are reduced). This is more evident when the spectral radius is less than 1.
- The effect of diagonal dominance is clearly visible here. Magnification Factors in a range [0.1:0.1:2] is introduced. When magnifying factor increases, spectral radius decreases and the convergence is achieved faster.
- Convergence rate is of the order SOR>GS>Jacobi. This is more evident from the below images where the max iterations taken by Jacobi is 3.5*10^4, GS is 2500 and SOR is 1000 (approx.)


Spectral Radius Vs Iterations
×10^4
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
Iterations
Spectral Radius
Jacobi


Spectral Radius Vs Iterations
2500
2000
1500
1000
500
0
10
20
30
40
50
60
70
Iterations
Spectral Radius
GS



Spectral Radius Vs Iterations
SOR
Iterations
Spectral Radius
0
200
400
600
800
1000
1200
1400
1600
0
10
20
30
40
50