1.

this is my "myjacobi" function:

```
function [ X ] = myjacobi( A, b, x0, tol, Niter)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
[n,n] = size(A);
Results = [];
finished = 0;
X = x0;
xs = A \ b;

k = 1;

while k <= Niter
    R = A * X - b;
    maxR = 0;

    for i = 1 : 1 : n
        if abs(R(i,1)) > maxR
            maxR = abs(R(i,1));
        end
    end

    error = X - xs;
    maxE = 0;

    for i = 1 : 1 : n
        if abs(error(i,1)) > maxE
            maxE = abs(error(i,1));
        end
    end

    Results = [Results; k, maxR, maxE];

    for i = 1 : 1 : n
        temp = b(i,1);
        for j = 1 : 1 : n
            if j ~= i
                temp = temp - A(i,j) * x0(j,1);
            end
        end

        X(i,1) = temp / A(i,i);
    end

    diff = X - x0;
    maxD = 0;
    for i = 1 : 1 : n
        if abs(diff(i,1)) > maxD
            maxD = abs(diff(i,1));
        end
    end

    maxX = 0;
```

```
    for i = 1 : 1 : n
        if abs(X(i,1)) > maxX
            maxX = abs(X(i,1));
        end
    end

    if (maxD / maxX) < tol
        finished = 1;
        break;
    end

    k = k + 1;
    x0 = X;

end

if finished == 1

else
    disp('Jacobi failed to converge after maximum iterations');
end

disp(Results);

end
```

This is my script for question 1

```
N = [50,100,200];

format long e;

for i = 1 : 1 : 3
    n = N(i);

    D = zeros(n,n);
    L = zeros(n,n);
    U = zeros(n,n);

    for row = 1 : 1 : n
        if row - 1 > 0
            L(row, row-1) = 1/2;
        end

        D(row,row) = 2;

        if row + 1 <= n
            U(row, row+1) = 1/2;
        end
    end
```

```
    A = D - L - U;

    MJ = D \ (L + U);
    disp(max(abs(eig(MJ))));

    b = zeros(n,1);
    for j = 1 : 1 : n
        if j == 1
            b(j,1) = 1;
        else
            b(j,1) = 2;
        end
    end

    x0 = zeros(n,1);
    Niter = 30;
    tol = 1.0000000000e-05;
    myjacobi(A,b,x0,tol,Niter);
end
```

so I first calculate the spectral radius for Jacobi iteration matrix and then run my "myjacobi" function to test it. As you can see, when n = 50,100,200, the spectral radius are all less than 1, so I expected Jacobi method will converge, and here is the result I got :

```
  4.990516643685221e-01

1.000000000000000e+00    2.000000000000000e+00    1.999999999999985e+00
2.000000000000000e+00    1.000000000000000e+00    9.999999999999847e-01
3.000000000000000e+00    5.000000000000000e-01    4.999999999999847e-01
4.000000000000000e+00    2.500000000000000e-01    2.499999999999847e-01
5.000000000000000e+00    1.250000000000000e-01    1.249999999999847e-01
6.000000000000000e+00    6.250000000000000e-02    6.249999999998468e-02
7.000000000000000e+00    3.125000000000000e-02    3.124999999998468e-02
8.000000000000000e+00    1.562500000000000e-02    1.562499999998468e-02
9.000000000000000e+00    7.812500000000000e-03    7.812499999984679e-03
1.000000000000000e+01    3.906250000000000e-03    3.906249999984679e-03
1.100000000000000e+01    1.953125000000000e-03    1.953124999984679e-03
1.200000000000000e+01    9.765625000000000e-04    9.765624999846789e-04
1.300000000000000e+01    4.882812500000000e-04    4.882812499846789e-04
1.400000000000000e+01    2.441406250000000e-04    2.441406249846789e-04
1.500000000000000e+01    1.220703125000000e-04    1.220703124846789e-04
1.600000000000000e+01    6.103515625000000e-05    6.103515623467892e-05
1.700000000000000e+01    3.051757812500000e-05    3.051757810967892e-05

4.997581411459940e-01

1.000000000000000e+00    2.000000000000000e+00    2.000000000000000e+00
2.000000000000000e+00    1.000000000000000e+00    1.000000000000000e+00
3.000000000000000e+00    5.000000000000000e-01    5.000000000000000e-01
4.000000000000000e+00    2.500000000000000e-01    2.500000000000000e-01
5.000000000000000e+00    1.250000000000000e-01    1.250000000000000e-01
6.000000000000000e+00    6.250000000000000e-02    6.250000000000000e-02
7.000000000000000e+00    3.125000000000000e-02    3.125000000000000e-02
8.000000000000000e+00    1.562500000000000e-02    1.562500000000000e-02
```

```
9.000000000000000e+00     7.812500000000000e-03     7.812500000000000e-03
1.000000000000000e+01     3.906250000000000e-03     3.906250000000000e-03
1.100000000000000e+01     1.953125000000000e-03     1.953125000000000e-03
1.200000000000000e+01     9.765625000000000e-04     9.765625000000000e-04
1.300000000000000e+01     4.882812500000000e-04     4.882812500000000e-04
1.400000000000000e+01     2.441406250000000e-04     2.441406250000000e-04
1.500000000000000e+01     1.220703125000000e-04     1.220703125000000e-04
1.600000000000000e+01     6.103515625000000e-05     6.103515625000000e-05
1.700000000000000e+01     3.051757812500000e-05     3.051757812500000e-05

4.999389284703266e-01

1.000000000000000e+00     2.000000000000000e+00     2.000000000000000e+00
2.000000000000000e+00     1.000000000000000e+00     1.000000000000000e+00
3.000000000000000e+00     5.000000000000000e-01     5.000000000000000e-01
4.000000000000000e+00     2.500000000000000e-01     2.500000000000000e-01
5.000000000000000e+00     1.250000000000000e-01     1.250000000000000e-01
6.000000000000000e+00     6.250000000000000e-02     6.250000000000000e-02
7.000000000000000e+00     3.125000000000000e-02     3.125000000000000e-02
8.000000000000000e+00     1.562500000000000e-02     1.562500000000000e-02
9.000000000000000e+00     7.812500000000000e-03     7.812500000000000e-03
1.000000000000000e+01     3.906250000000000e-03     3.906250000000000e-03
1.100000000000000e+01     1.953125000000000e-03     1.953125000000000e-03
1.200000000000000e+01     9.765625000000000e-04     9.765625000000000e-04
1.300000000000000e+01     4.882812500000000e-04     4.882812500000000e-04
1.400000000000000e+01     2.441406250000000e-04     2.441406250000000e-04
1.500000000000000e+01     1.220703125000000e-04     1.220703125000000e-04
1.600000000000000e+01     6.103515625000000e-05     6.103515625000000e-05
1.700000000000000e+01     3.051757812500000e-05     3.051757812500000e-05
```

as you can see, when n = 50,100,200, my Jacobi function all use 17 steps to converge into the error limit we want. Which is comply with my expectation since all 3 spectral radius are less than 1, also as you can see, n = 50 has the least spectral radius whis is `4.990516643685221e-01`, n = 100 has the medium spectral radius which is `4.997581411459940e-01`, and n = 200 has the largest spectral radius, which is `4.999389284703266e-01`.

2.

this is my gauss-seidel function:

```
function [ X ] = mygauseidel( A, b, x0, tol, Niter)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
[n,n] = size(A);
Results = [];
finished = 0;
X = x0;
xs = A \ b;

k = 1;

while k <= Niter
    R = A * X - b;
    maxR = 0;

    for i = 1 : 1 : n
```

```
        if abs(R(i,1)) > maxR
            maxR = abs(R(i,1));
        end
    end

    error = X - xs;
    maxE = 0;

    for i = 1 : 1 : n
        if abs(error(i,1)) > maxE
            maxE = abs(error(i,1));
        end
    end

    Results = [Results; k, maxR, maxE];

    for i = 1 : 1 : n
        temp = b(i,1);
        for j = 1 : 1 : i - 1
            temp = temp - A(i,j) * X(j,1);
        end
        for j = i + 1 : 1 : n
            temp = temp - A(i,j) * x0(j,1);
        end

        X(i,1) = temp / A(i,i);
    end

    diff = X - x0;
    maxD = 0;
    for i = 1 : 1 : n
        if abs(diff(i,1)) > maxD
            maxD = abs(diff(i,1));
        end
    end

    maxX = 0;
    for i = 1 : 1 : n
        if abs(X(i,1)) > maxX
            maxX = abs(X(i,1));
        end
    end

    if (maxD / maxX) < tol
        finished = 1;
        break;
    end

    k = k + 1;
    x0 = X;

end

if finished == 1

else
    disp('Jacobi failed to converge after maximum iterations');
end
```

```
disp(Results);

end
```

And here is my script for question2:

```
N = [50,100,200];

format long e;

for i = 1 : 1 : 3
    n = N(i);

    D = zeros(n,n);
    L = zeros(n,n);
    U = zeros(n,n);

    for row = 1 : 1 : n
        if row - 1 > 0
            L(row, row-1) = 1/2;
        end

        D(row,row) = 2;

        if row + 1 <= n
            U(row, row+1) = 1/2;
        end
    end

    A = D - L - U;

    MGS = (D - L) \ U;
    disp(max(abs(eig(MGS))));

    b = zeros(n,1);
    for j = 1 : 1 : n
        if j == 1
            b(j,1) = 1;
        else
            b(j,1) = 2;
        end
    end

    x0 = zeros(n,1);
    Niter = 30;
    tol = 1.0000000000e-05;
    mygauseidel(A,b,x0,tol,Niter);
end
```

Here is the result:

```
2.490525637089925e-01

     1.000000000000000e+00     2.000000000000000e+00     1.999999999999985e+00
     2.000000000000000e+00     6.666666666666670e-01     6.666666666666521e-01
     3.000000000000000e+00     2.222222222222228e-01     2.222222222222090e-01
     4.000000000000000e+00     7.407407407407418e-02     7.407407407406441e-02
```

```
5.000000000000000e+00     2.469135802469147e-02     2.469135802468503e-02
6.000000000000000e+00     8.230452674897304e-03     8.230452674892863e-03
7.000000000000000e+00     2.743484224966064e-03     2.743484224962511e-03
8.000000000000000e+00     9.144947416555027e-04     9.144947416535043e-04
9.000000000000000e+00     3.048315805518342e-04     3.048315805509461e-04
1.000000000000000e+01     1.016105268507594e-04     1.016105268500933e-04
1.100000000000000e+01     3.387017561706784e-05     3.387017561684580e-05
1.200000000000000e+01     1.129005853917064e-05     1.129005853894860e-05

2.497579396626840e-01

1.000000000000000e+00     2.000000000000000e+00     2.000000000000000e+00
2.000000000000000e+00     6.666666666666670e-01     6.666666666666667e-01
3.000000000000000e+00     2.222222222222228e-01     2.222222222222225e-01
4.000000000000000e+00     7.407407407407418e-02     7.407407407407440e-02
5.000000000000000e+00     2.469135802469147e-02     2.469135802469169e-02
6.000000000000000e+00     8.230452674897304e-03     8.230452674897526e-03
7.000000000000000e+00     2.743484224966064e-03     2.743484224966286e-03
8.000000000000000e+00     9.144947416555027e-04     9.144947416559468e-04
9.000000000000000e+00     3.048315805518342e-04     3.048315805522783e-04
1.000000000000000e+01     1.016105268507594e-04     1.016105268512035e-04
1.100000000000000e+01     3.387017561706784e-05     3.387017561751193e-05
1.200000000000000e+01     1.129005853917064e-05     1.129005853961473e-05

2.514836255783694e-01

1.000000000000000e+00     2.000000000000000e+00     2.000000000000000e+00
2.000000000000000e+00     6.666666666666670e-01     6.666666666666667e-01
3.000000000000000e+00     2.222222222222228e-01     2.222222222222225e-01
4.000000000000000e+00     7.407407407407418e-02     7.407407407407440e-02
5.000000000000000e+00     2.469135802469147e-02     2.469135802469169e-02
6.000000000000000e+00     8.230452674897304e-03     8.230452674897526e-03
7.000000000000000e+00     2.743484224966064e-03     2.743484224966286e-03
8.000000000000000e+00     9.144947416555027e-04     9.144947416559468e-04
9.000000000000000e+00     3.048315805518342e-04     3.048315805522783e-04
1.000000000000000e+01     1.016105268507594e-04     1.016105268512035e-04
1.100000000000000e+01     3.387017561706784e-05     3.387017561751193e-05
1.200000000000000e+01     1.129005853917064e-05     1.129005853961473e-05
```

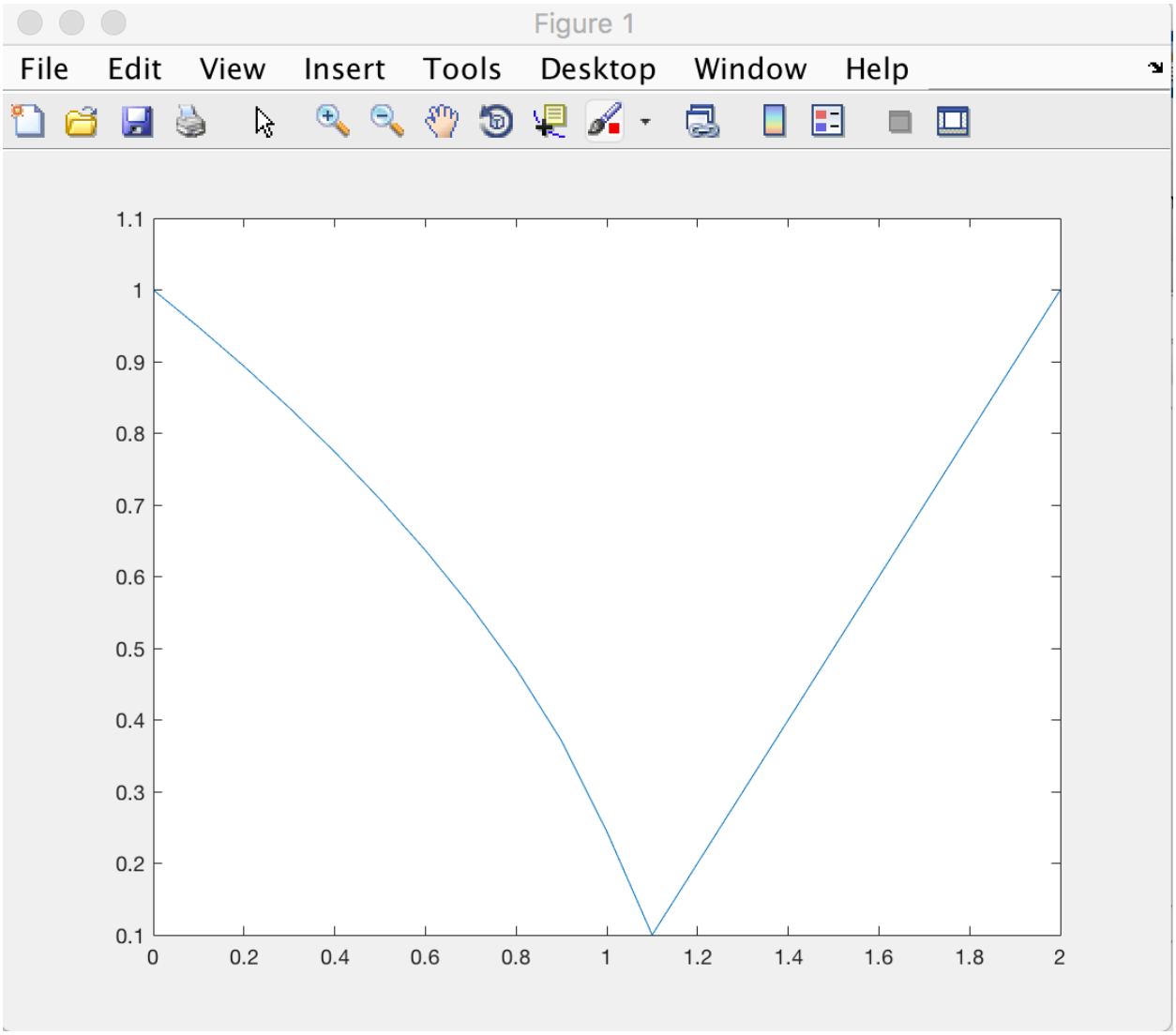
as you can see I first display all the spectral radius of 3 gauss-seidel matrix, all of 3 radius are less than 1, and I use my gauss-seidei function, the result is within 12 steps, they all converges into the right limit. Which is comply with my expectation since all 3 radius are less than 1. Also as you can see when n = 50, again I have the least spectral radius and when n = 200 again I have the largest spectral radius. Another observation is for n = 50, n = 100 and n = 200, gauss-seidal's spectral radius are all less than Jacobi matrix's spectral radius, and gauss-seidal method require less steps than Jacobi in order to converge.

3.

```
format long e;
W = 0:0.1:2;
radius = 1:21;
```

```
n = 20;

D = zeros(n,n);
L = zeros(n,n);
U = zeros(n,n);

for row = 1 : 1 : n
   if row - 1 > 0
      L(row, row-1) = 1/2;
   end

   D(row,row) = 2;

   if row + 1 <= n
      U(row, row+1) = 1/2;
   end
end

A = D - L - U;

for i = 1 : 1 : 21
    w = W(1,i);

    SOR = (D - w*L) \ ((1-w)*D + w*U);
    radius(1,i) = max(abs(eig(SOR)));
end

plot(W, radius);
```

Here is my script 3 code, and this is the gragh I generated:



as you can see in the gragh, the x-coordinate is W, the y coordinate is the spectral radius of SOR matrix. When w = 0 and w = 2, the spectral radius is 1 which means the SOR matrix does not converge, but when w is close to 1.1, the spectral radius is getting really close to 0 which means the SOR method will converge very fast.