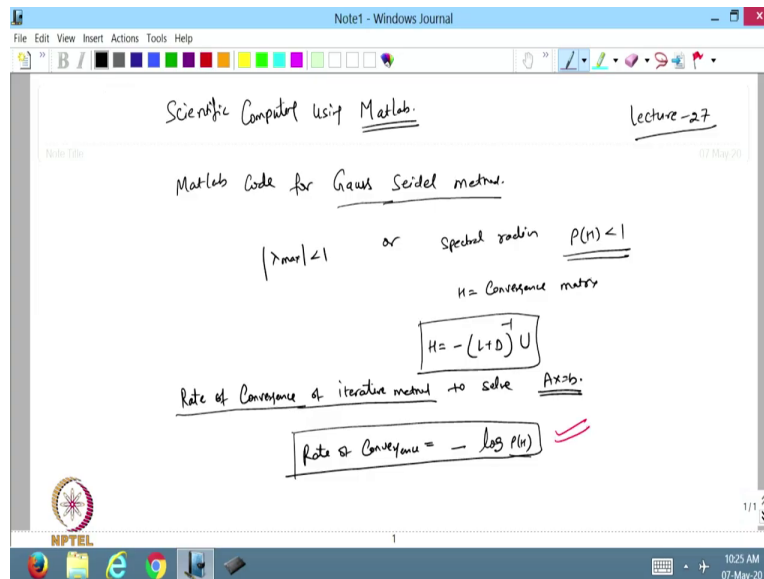


Scientific Computing Using MATLAB
Professor Vivek Aggarwal
Department of Mathematics
Indian Institute of Technology, Delhi/DTU
Lecture - 27
MATLAB Code for Gauss-Seidel Method

Hello, viewers. Welcome back to the course on Scientific Computing Using MATLAB. So, in the previous lectures, we have discussed the iterative methods for solving the linear system of equations. So, in the previous lecture, we discussed how we can make the MATLAB code for the Gauss-Jacobi method.

(Refer Slide Time: 00:44)



So, in today lecture, we will try to make the MATLAB code for Gauss-Seidel method and we know that in the Gauss-Seidel methods that the method is convergent if eigenvalues, the modulus value of the eigenvalues should be less than equal to 1 or the spectral radius that is $\rho(H) < 1$. So, and the $\rho(H)$ we know that this $\rho(H)$, where H is the convergence matrix. This one, we can delete, so this equals to less than 1.

So, and now in the Gauss-Seidel method, we know that the $H = -(L + D)^{-1}U$. So, in this case, we will try to find what is the, our convergence matrix. Now if you want to find the rate of convergence, so the rate of convergence of iterative methods. To solve the system $Ax=B$. So we find the rate of convergence.

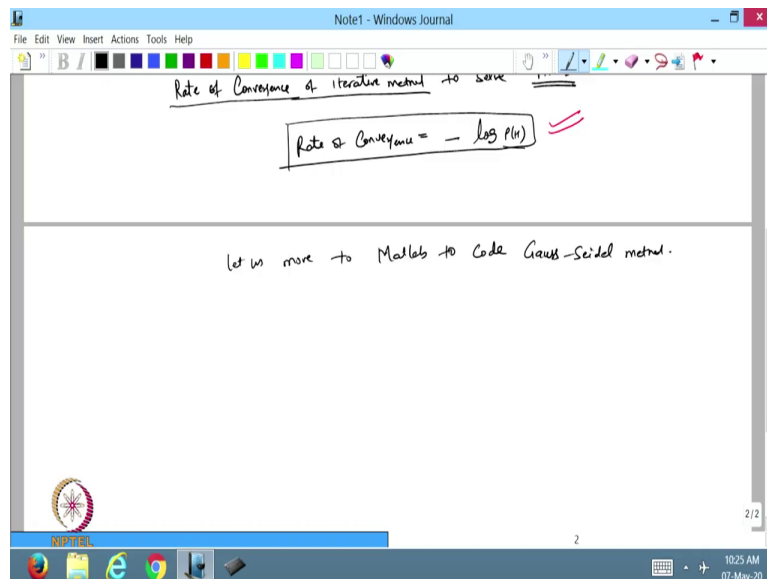
So the rate of convergence = $-\log \rho(H)$. Because the spectral radius we are taking is the

maximum value we are taking, so if you see from here, we can write this as I can write, it should be less than 1.

So if I take the maximum eigenvalues using that one, we will find the spectrum and I know that the either Gauss-Seidel method is converging that then the eigenvalue, the maximum eigenvalue will be less than 1. So, that is why the negative sign had been taken because this quantity is less than 1, so when I were to take the log of this one that we will be negative. So just to get rid of this one, I will take the negative sign from here.

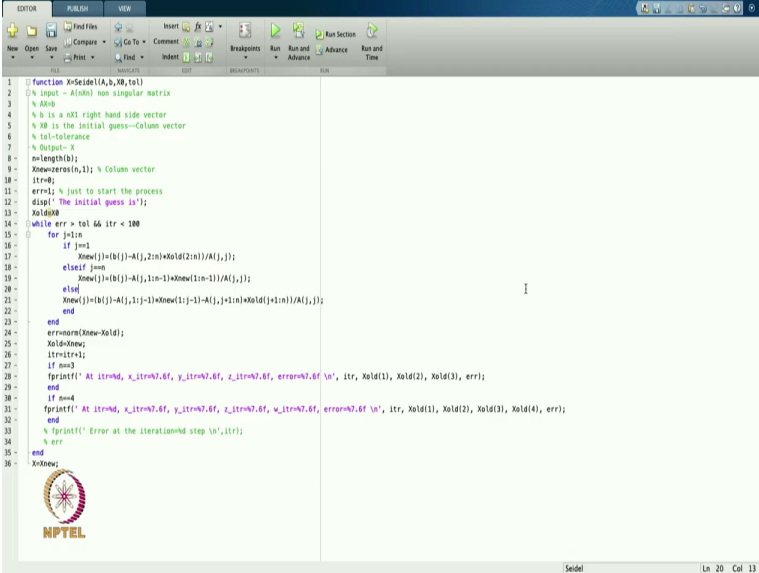
So, that is the rate of convergence of the, of any iterative methods it may be Gauss-Seidel, it may be a Gauss-Jacobian, other rate iterative method. Then the rate of convergence is given by this formula and in the code, we will try to make, so we will introduce all this in the, in the MATLAB code we will introduce that how we can calculate this convergent matrix, then how we can calculate the rate of convergence for a given initial condition.

(Refer Slide Time: 03:52)



So, let us move to then, so let us move to MATLAB to code Gauss-Seidel method. So let us move to the MATLAB. So, let us start doing the code for the Gauss-Seidel.

(Refer Slide Time: 04:21)



```
1 function X=Seidel(A,b,X0,tol)
2 % Input = A(b) non singular matrix
3 % A(b)
4 % b is a n x 1 right hand side vector
5 % X0 is the initial guess--Column vector
6 % tol--tolerance
7 % Output = X
8 n=length(b);
9 Xnew=zeros(n,1); % Column vector
10 itr=0;
11 error=1; % just to start the process
12 disp('The initial guess is:')
13 Xold=X0;
14 while error > tol && itr < 100
15     for j=1:n
16         if j==1
17             Xnew(j)=(b(j)-A(j,2:n)*Xold(2:n))/A(j,1);
18         elseif j==n
19             Xnew(j)=(b(j)-A(j,1:n-1)*Xnew(1:n-1))/A(j,j);
20         else
21             Xnew(j)=(b(j)-A(j,1:j-1)*Xnew(1:j-1)-A(j,j+1:n)*Xold(j+1:n))/A(j,j);
22         end
23     end
24     error=norm(Xnew-Xold);
25     Xold=Xnew;
26     itr=itr+1;
27     if mod(itr,4)
28         fprintf(' At itr=%d, x_itr=%f, y_itr=%f, z_itr=%f, error=%f \n', itr, Xold(1), Xold(2), Xold(3), error);
29     end
30     if mod(itr,4)
31         fprintf(' At itr=%d, x_itr=%f, y_itr=%f, z_itr=%f, error=%f \n', itr, Xold(1), Xold(2), Xold(3), Xold(4), error);
32     end
33     % fprintf(' Error at the iteration%d stop \n',itr);
34     % err
35 end
36 X=Xnew;
```

So, as we have done in the Jacobi method, so I have also made the Gauss-Seidel method. This is for you, so we can do it here. Now, it is the same as I am defining the function that $X = \text{Seidel}(A, b, X_0, \text{and tolerance})$. So in the previous one, we have changed the name to Jacobi; so now it is a Seidel, Gauss-Seidel method.

Other things are same that A is a matrix, which we are going to solve that is $n \times n$, b is the right-hand vector and X_0 is the initial approximation of the solution, and the tolerance is how much accuracy I need for the, finding the solution of the system. Now, this is the b , so find the length of b that is n . Now, from here I define a vector that is with the zero elements. So this is the dimension n and the column 1, so it is a column vector. So you can write from here that this is the column vector.

Now I start with the iteration, so that is the initial value 0. I just take the error as 1 just to start the process. Now, the display I show that the initial guess is whatever the initial guess I give, so that will be X ; I will put that one as $X_{old} = X_0$. So X_0 is coming from this, from the function where this function has been called. So this X_0 will go and it will save into the X_{old} and if I want to display this one, so I will remove these semicolons. Now this will be displayed on the screen.

Now I will go to the while-loop. So in the while-loop, I will go from here that this is the, if the

error is greater than tolerance and iteration is less than 100 then we should enter into this one. So, if anyone is false, then we will not move to the while-loop. So error should be greater than tolerance and iteration should be less than 100.

Now, from here if I go inside then I will go from j from 1 to n. Now, what will happen in the Gauss-Seidel method, that the first value that is the first equation of the system will find out the value of X. Suppose, I have the solution as x, y, z in 3 by 3, or x1, x2, x3, up to xn in n by n matrix, then x1 will be calculated same as a Jacobi method.

So, this will be j=1, so X new 1 will be

$X_{new}(j) = (b(j) - A(j, 2 : n) * X_{old}(2 : n)) / A(j, j)$. So that is the first point in the matrix, so this will be calculated.

And then the last one, j=n, so I know that when I take j=n then the new updated value will be used to find out the X new. So, that is why take

$X_{new}(j) = (b(j) - A(j, 1 : n - 1) * X_{new}(1 : n - 1)) / A(j, j)$, see from here that I have taken the X new, so this is the X new.

So whatever the solution we are getting from the previous equations we are solving that j is equal to 1 and j is equal to 2, 3 up to n-1, that all solutions have been used here and then divided A(j, j). So this and this conditions we are, we have to calculate separately. Else, means other than these first and last, we will find out the X new j is equal to b j, so this is the value we are going to use.

So what is going to happen? If we split the matrix into two parts, so the matrix below the diagonal elements will be calculated this X new and above the main diagonal, it will be used with the X old. So whatever, the same thing we have done, so we will find out the b j -A j and then all the columns from 1 to j-1, I will use the updated value that is Xnew -A j and j+1 to n because I have to leave the value on A j j.

So after that, all the values above the main diagonal, I am using the old value. So that is, from j+1 to n and then the whole thing will be divided by A j j. So in this case, we have to split the whole while-loop and then for the for-loop into three parts when j=1, j=n and other than that. Now, this is the end of if-loop and this is the end of for-loop.

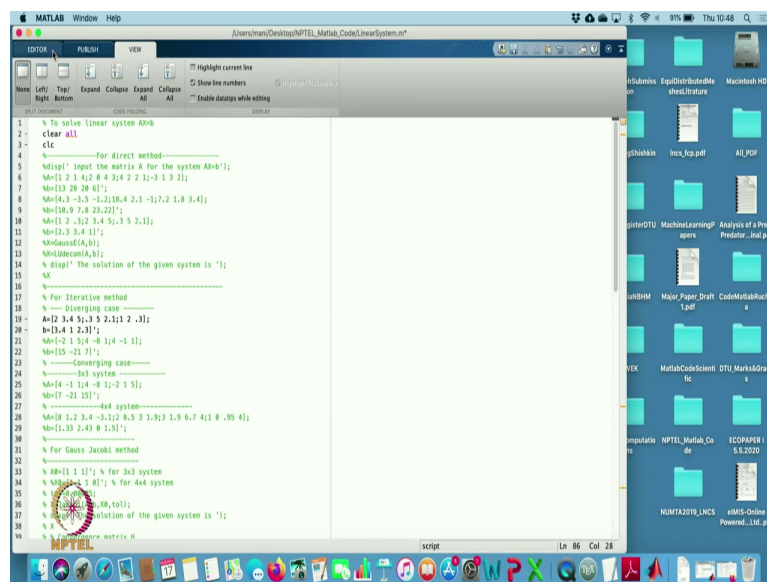
Then I will find the error. So this is the norm I am taking, so Xnew-Xold and I am taking the maximum norm, so that is the error I am taking. Then after this I will, whatever the value we are getting from this Xnew, I will use this Xnew to update my old value. So there that is this Xnew will go to the Xold. Now iteration will be increased by 1, so earlier, iteration was 0 now it is 1. So after one iteration, whatever the solution you are getting, you will get this.

Now, this is, I have written this one to find out that how the accuracy and how the solution is approaching to converge, so I have displayed only two cases, when my matrix is 3 by 3 or my matrix is 4 by 4, because other than that it is very difficult to display it on the screen. So if $n=3$, I print, at iteration so x_1 , so I am suppose I have a 3 by 3 system, so my solutions are x , y , z . So I put x at this iteration here, y at this iteration, and z , that iteration and this is the error I am finding, so I will put this here as iteration X_{old} one because all the solution is going to this as a vector.

So, that first component will be I am treating as x ; then the second component, I am treating as y ; and the third component, I am treating as z , so this is the z I am finding; and in the last time finding the, displaying the error also, and that is the back-slash \backslash is a new line. So this will end and if $n=4$, the same thing is happening except the last one, I will take that as a w , and then the solution will be $X_{old}(4)$ and this is the error.

So I will display this one on the screen and this will be updated at each iteration and that will be display in the screen as in the form of a table, and then I will end this while-loop, and then $X=X_{new}$ because I have to send back this X output, so this my X_{new} , after all this iteration, my X_{new} will go and it will save in the X and this X will be returned to the system or return to the code from where this function has been called. So this is the function we have made for the Seidel method.

(Refer Slide Time: 12:35)



```

1 % To solve Linear system Axb
2 clear all
3 %
4 % -----For direct method-----
5 % input the matrix A for the system Axb;
6 A=[1 2 1 4;2 4 3 4 2 2 1;-3 1 3 2];
7 A=[1 2 2 2 2 2 2];
8 A=[4 3 -3.5 -1.2;18.4 2.1 -1.7;2 1.8 3.4];
9 A=[18.5 7.8 25.22];
10 A=[1 2 -3.2 3.4 5;-3 5 2.1];
11 A=[2 3 3.4 1];
12 A=double(A,b);
13 % display the solution of the given system is ;
14 %
15 %
16 % -----For iterative method-----
17 % -----Diverging case-----
18 A=[2 3.4 5;-3 5 2.1;1 2 -3];
19 b=[3.4 1 2.3];
20 A=[1 2 1 4;-8 1 4 -1 1];
21 A=[1 2 1 5;4 -8 1 4 -1 1];
22 A=[15 -21 7];
23 % -----Converging case-----
24 % -----3x3 system-----
25 A=[1 1 1;4 -8 1;-2 1 5];
26 A=[17 -21 15];
27 % -----4x4 system-----
28 A=[18 1.2 3.4 -3.1;2 6.5 3 1.9;3 1.9 6.7 4;1 0 -39 4];
29 A=[1.33 2.43 8 1.33];
30 % For Gauss Jacobi method
31 %
32 % A=[1 1 1]; % for 3x3 system
33 % A=[1 1 1 1]; % for 4x4 system
34 % A=[1 1 1 1]; % for 4x4 system
35 % A=[1 1 1 1]; % for 4x4 system
36 % A=[1 1 1 1]; % for 4x4 system
37 % A=[1 1 1 1]; % for 4x4 system
38 % A=[1 1 1 1]; % for 4x4 system
39 % A=[1 1 1 1]; % for 4x4 system
40 % A=[1 1 1 1]; % for 4x4 system
41 % A=[1 1 1 1]; % for 4x4 system
42 % A=[1 1 1 1]; % for 4x4 system
43 % A=[1 1 1 1]; % for 4x4 system
44 % A=[1 1 1 1]; % for 4x4 system
45 % A=[1 1 1 1]; % for 4x4 system
46 % A=[1 1 1 1]; % for 4x4 system
47 % A=[1 1 1 1]; % for 4x4 system
48 % A=[1 1 1 1]; % for 4x4 system
49 % A=[1 1 1 1]; % for 4x4 system
50 % A=[1 1 1 1]; % for 4x4 system

```


system, so this is 3 by 3 system. So I am taking the initial condition as 1 1 0 and this is the dash means the, it will be a column vector. The tolerance I am taking 10^{-5} .

Now, I will call the function X equal to Gauss-Seidel from here, so this is the input I am passing. Then, after that, it will give me the X, so the, I will show that the display, the solution of the given system in this one. Now what I want to do, I want to find the convergence matrix, so that is the convergence matrix, n is the length of b, L is the left matrix, D is a diagonal matrix, U is upper triangular matrix, so L is the lower triangular matrix.

So, I will put the value of the given matrix into the corresponding lower, diagonal, and upper triangular matrix. So it is, I am putting for-loop, i from 1 to n and j from 1 to n. Now, if $i < j$, it means I am putting that if i is 1, then if i is 1 then j will move from 2 to n. So 1 is, $i < j$. Then I will be getting the values as an upper triangular matrix. So, now if $i = j$, then this will be saved in the diagonal matrix as a diagonal matrix and this is the else value. So if it is not this or not this, then whatever is left that will be saved in the lower triangular matrix.

So this is the way we can split the matrix A into the lower triangular, upper triangular or diagonal matrix. Now from the Gauss-Seidel method, I know that my convergence matrix is

$-inv(L + D) * U$. Now, to display the eigenvalues of the corresponding matrix, I find the eigenvalue of this matrix, so that will show the eigenvalue. Then I find the spectral radius, so the spectral radius I am finding from here and I call that as spectralRad.

So that is what I am doing, I am finding eigenvalue, then I take the absolute value because the eigenvalue may be complex also. So, in that case, I will find out the absolute value of that and then I am finding the maximum of all. Now I am, from here, I am displaying that the spectral radius and the rate of convergence of the given method is, spectral radius will be this one and $-\log(\text{spectral radius})$ that is the rate of convergence.

So, let us try to run this one. So I will run this one and so, let us see what will happen.

(Refer Slide Time: 16:39)

The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the initial guess for a system of equations, with variables x , y , and z defined as vectors of size 10. The Workspace shows the values of these variables, which are all zero.

```
The initial guess is
x =
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
y =
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
z =
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

Workspace:

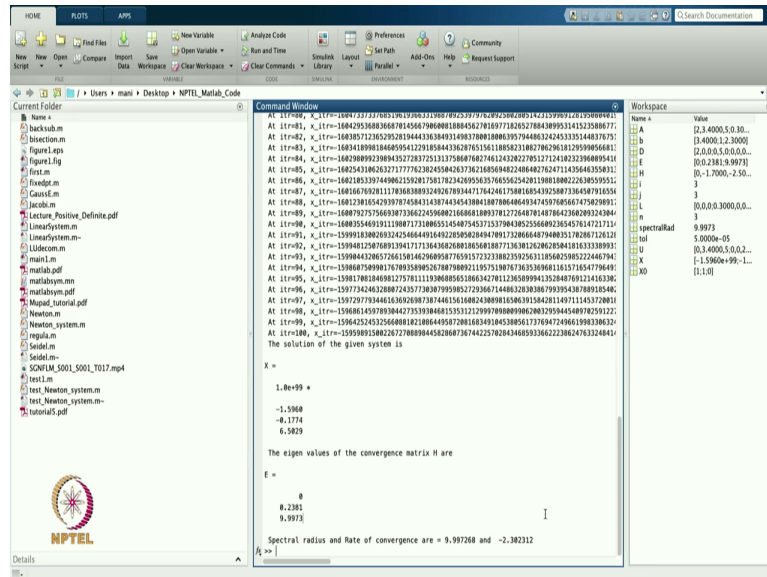
Name	Value
x	[10x1 double]
y	[10x1 double]
z	[10x1 double]
...	...

The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the results of a numerical solution, with variables x , y , and z defined as vectors of size 10. The Workspace shows the values of these variables, which are all zero.

```
The results of the numerical solution are
x =
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
y =
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
z =
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

Workspace:

Name	Value
x	[10x1 double]
y	[10x1 double]
z	[10x1 double]
...	...



So from here, if you see, it started with the iteration number 1, then x iteration, y iteration, and z iteration, so this is my initial solution and the error is this. Then it keeps increasing, then at iteration number 2, the solution goes to -14 then -161 and you see that the value is going to be infinity, -infinity in this case; and the error if you see, then the error is going to be increasing and going to be infinity. So, in this case, after 100 iterations I stopped my program, and then this is the solution.

And, in this case, I can see that the method is not converging, it is diverging. So this, for the given system of equation or our method is diverging, and from here, you can also see that the, if you find the, the eigenvalue then it is coming almost 10 eigenvalue. So the eigenvalue is coming 10, because if the system is going to converge then we know that the eigenvalue is going to be less than 1, the modulus of that eigenvalue.

(Refer Slide Time: 18:01)

```

1 % To solve Linear system A\b
2 clear all
3 clc
4 % ----- For direct method -----
5 disp(' Input the matrix A for the system A\b:');
6 M=[1 2 1 4;2 8 4 3;4 2 2 1;-3 1 3 2];
7 M=[13 28 28 61];
8 M=[4,3 -3,5 -1,2;18,4 2,1 -1,7;2 1,8 3,4];
9 M=[18,9 7,8 23,22];
10 M=[1 2 -3;2 3,4 5;3 5 2,1];
11 M=[2,3 3,4 1];
12 M=double(A,b);
13 M=double(M);
14 % disp(' The solution of the given system is ');
15 %
16 % ----- For iterative method -----
17 % ----- Diverging case -----
18 M=[2 3,4 5;3 5 2,1;1 2 -3];
19 M=[3,4 1 2,3];
20 M=[-2 1 3,4 -8 1;4 -1 1];
21 M=[15 -21 7];
22 % ----- Converging case -----
23 % ----- 3x3 system -----
24 M=[-1 1,4 -8 1;-2 1 5];
25 M=[7 -21 15];
26 % ----- 4x4 system -----
27 M=[8 1,2 3,4 -3,12 8,5 3 1,9;3 1,9 6,7 4,1 0 -95 4];
28 M=[1,33 2,43 8 1,5];
29 % For Gauss Jacobi method
30 %
31 % M=[1 1 1]; % for 3x3 system
32 % M=[2 1 8]; % for 4x4 system
33 % M=[1,33 2,43 8,1,5];
34 % M=[1,33 2,43 8,1,5];
35 % disp(' The solution of the given system is ');
36 %
37 %
38 %
39 %
40 %
41 %
42 %
43 %
44 %
45 %
46 %
47 %
48 %
49 %
50 %
51 %
52 %
53 %
54 %
55 %
56 %
57 %
58 %
59 %
60 %
61 %
62 %
63 %
64 %
65 %
66 %
67 %
68 %
69 %
70 %
71 %
72 %
73 %
74 %
75 %
76 %
77 %
78 %
79 %
80 %
81 %
82 %
83 %
84 %
85 %
86 %
87 %
88 %
89 %
90 %
91 %
92 %
93 %
94 %
95 %
96 %
97 %
98 %
99 %
100 %

```

So this is the, the first one we have taken, now I will change this one and then I will take the system. So this is also a 3 by 3 system I am taking and then I run this one. So let us see what will happen.

(Refer Slide Time: 18:20)

```

1 % To solve Linear system A\b
2 clear all
3 clc
4 % ----- For direct method -----
5 disp(' Input the matrix A for the system A\b:');
6 M=[1 2 1 4;2 8 4 3;4 2 2 1;-3 1 3 2];
7 M=[13 28 28 61];
8 M=[4,3 -3,5 -1,2;18,4 2,1 -1,7;2 1,8 3,4];
9 M=[18,9 7,8 23,22];
10 M=[1 2 -3;2 3,4 5;3 5 2,1];
11 M=[2,3 3,4 1];
12 M=double(A,b);
13 M=double(M);
14 % disp(' The solution of the given system is ');
15 %
16 % ----- For iterative method -----
17 % ----- Diverging case -----
18 M=[2 3,4 5;3 5 2,1;1 2 -3];
19 M=[3,4 1 2,3];
20 M=[-2 1 3,4 -8 1;4 -1 1];
21 M=[15 -21 7];
22 % ----- Converging case -----
23 % ----- 3x3 system -----
24 M=[-1 1,4 -8 1;-2 1 5];
25 M=[7 -21 15];
26 % ----- 4x4 system -----
27 M=[8 1,2 3,4 -3,12 8,5 3 1,9;3 1,9 6,7 4,1 0 -95 4];
28 M=[1,33 2,43 8 1,5];
29 % For Gauss Jacobi method
30 %
31 % M=[1 1 1]; % for 3x3 system
32 % M=[2 1 8]; % for 4x4 system
33 % M=[1,33 2,43 8,1,5];
34 % M=[1,33 2,43 8,1,5];
35 % disp(' The solution of the given system is ');
36 %
37 %
38 %
39 %
40 %
41 %
42 %
43 %
44 %
45 %
46 %
47 %
48 %
49 %
50 %
51 %
52 %
53 %
54 %
55 %
56 %
57 %
58 %
59 %
60 %
61 %
62 %
63 %
64 %
65 %
66 %
67 %
68 %
69 %
70 %
71 %
72 %
73 %
74 %
75 %
76 %
77 %
78 %
79 %
80 %
81 %
82 %
83 %
84 %
85 %
86 %
87 %
88 %
89 %
90 %
91 %
92 %
93 %
94 %
95 %
96 %
97 %
98 %
99 %
100 %

```

The initial guess is

Xold =

1
1
0

At itr=1, x_itr=2.000000, y_itr=3.625000, z_itr=0.875000, error=4.164883
At itr=2, x_itr=1.987500, y_itr=3.593750, z_itr=0.864375, error=0.164888
At itr=3, x_itr=1.997387, y_itr=3.584151, z_itr=0.880847, error=0.122418
At itr=4, x_itr=1.998523, y_itr=3.590268, z_itr=0.999556, error=0.885321
At itr=5, x_itr=1.999926, y_itr=3.599988, z_itr=0.999989, error=0.881844
At itr=6, x_itr=1.999988, y_itr=3.599989, z_itr=0.999984, error=0.880806
At itr=7, x_itr=1.999999, y_itr=3.599999, z_itr=0.999988, error=0.880822

The solution of the given system is

X =

2.0000
4.0000
3.0000

The eigen values of the convergence matrix H are

E =

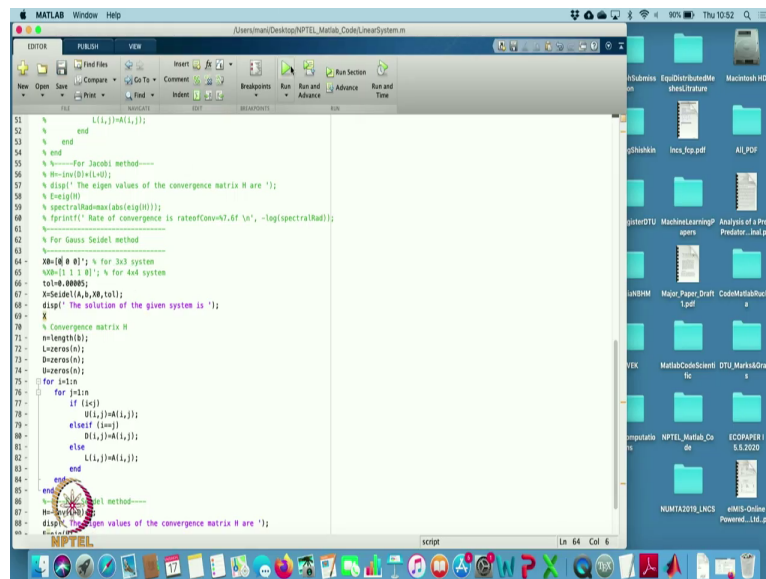
0
-0.12500
0.12500

Spectral radius and Rate of convergence are 0.125000 and 2.879442

Now, from here, this is my solution. So the initial guess I started with 1 1 0. Now, after the seven iteration, you can see that my solution is that x is converging to 2, this is converging into 4, and z is converging to 3. So that, in my system, is the solution of this system. And then, if I find the eigenvalue, then the eigenvalue is coming 0. So zero eigenvalue means this matrix is a singular matrix, then its value is minus 0.1 and this is the maximum eigenvalue this one.

So the maximum eigenvalue is 0.1250, so that shows that the spectral radius is this one, so using this spectral radius, so that is my spectral radius and you can see that the rate of convergence is very high, that is, 2. So with this help, with this Gauss-Seidel method, I am able to find the solution in the seven iteration only.

(Refer Slide Time: 19:19)



```

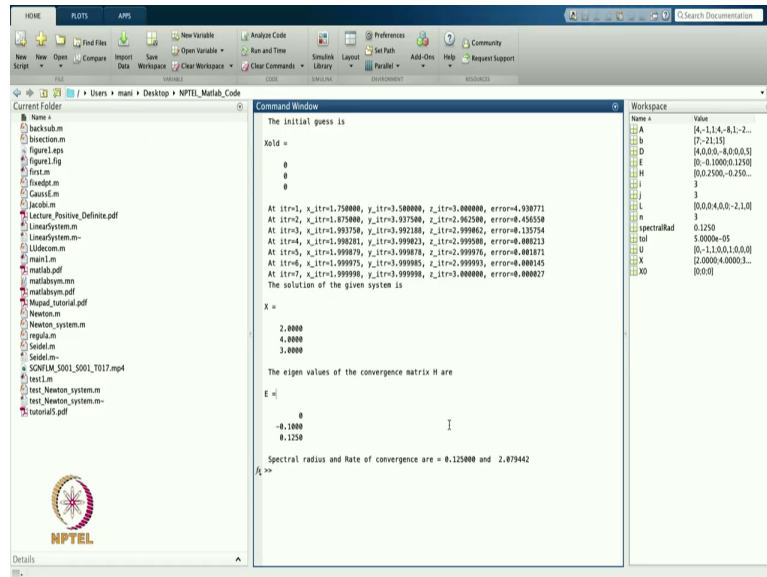
MATLAB Window Help
C:\Users\man\Desktop\NPTEL_Matlab_Code\LinearSystem.m

51 % L(i,j)=A(i,j);
52 % end
53 % end
54 % -----For Jacobi method-----
55 % B=inv(D)*(L+U);
56 % disp('The eigen values of the convergence matrix B are ');
57 % [eig(B)]
58 % spectralRadius=max(abs(eig(B)));
59 % 'Print' Rate of convergence is rateOfConv=7.0f %', -log(spectralRadius));
60 % -----
61 % For Gauss Seidel method
62 % -----
63 % X0=[0 0 0]'; % for 3x3 system
64 % X0=[1 1 0]'; % for 4x4 system
65 % tol=0.0005;
66 % [A,b]=randi(4,4,10,tol);
67 % disp('The solution of the given system is ');
68 % R
69 % A Convergence matrix H
70 % n=length(b);
71 % L=zeros(n);
72 % U=zeros(n);
73 % U=zeros(n);
74 % U=zeros(n);
75 % for i=1:n
76 %     for j=1:n
77 %         if i==j
78 %             U(i,j)=A(i,j);
79 %         elseif i>j
80 %             D(i,j)=A(i,j);
81 %         else
82 %             L(i,j)=A(i,j);
83 %         end
84 %     end
85 % end
86 % % -----For Gauss Seidel method-----
87 % B=inv(D)*(L+U);
88 % disp('The eigen values of the convergence matrix B are ');

```

Now, from here you can see what will happen if I change the initial approximation. So let us try to change the initial approximation. Suppose, I take X0 is equal to 1 0 0, or what will happen if I take all the zeros then let us see what will happen.

(Refer Slide Time: 19:40)



And if we see from here, then if I take 0 0, then the solution is still converging to this one within the seven iteration and the same eigenvalue definitely will come. But, in this case, the iterations are similar, the seven iteration. So I told you that you can choose any initial condition whatever the initial condition you want to choose.

(Refer Slide Time: 20:07)


```

51 % L(i,j)=H(i,j);
52 %
53 % end
54 % end
55 % %-----For Jacobi method-----
56 % We solve D\*(b-L\*U);
57 % disp('The eigen values of the convergence matrix H are ');
58 % eig(H)
59 % spectralRadius=max(abs(eig(H)));
60 % fprintf('Rate of convergence is rateofConv=%f \n', -log(spectralRadius));
61 %
62 % For Gauss Seidel method
63 %-----
64 % x=[0 0 0]'; % for 3x3 system
65 % x=[0 0 0 0]'; % for 4x4 system
66 % tol=0.00005;
67 % x=Seidel(A,b,x,tol);
68 % disp('The solution of the given system is ');
69 % x
70 % Convergence matrix H
71 % n=length(b);
72 % L=zeros(n);
73 % U=zeros(n);
74 % for i=1:n
75 %     for j=1:n
76 %         U(i,j)=H(i,j);
77 %     end
78 %     L(i,i)=1-U(i,i);
79 %     L(i,j)=H(i,j);
80 %     U(i,j)=H(i,j);
81 % end
82 % end
83 % end
84 % end
85 % %-----For Gauss Seidel method-----
86 % We solve D\*(b-L\*U);
87 % disp('The eigen values of the convergence matrix H are ');
88 % eig(H)

```

Now suppose I put it 0 0 0 and then 1, then let us see what will happen.

(Refer Slide Time: 20:13)

```

The initial guess is
xold =
0
0
0
1

At itr=1, x_itr=1.500000, y_itr=1.500000, z_itr=2.000000, error=4.255585
At itr=2, x_itr=1.000000, y_itr=1.000000, z_itr=2.000000, error=0.597212
At itr=3, x_itr=1.000000, y_itr=1.000000, z_itr=2.000000, error=0.104111
At itr=4, x_itr=1.000000, y_itr=1.000000, z_itr=2.000000, error=0.018186
At itr=5, x_itr=1.000000, y_itr=1.000000, z_itr=2.000000, error=0.001615
At itr=6, x_itr=1.000000, y_itr=1.000000, z_itr=2.000000, error=0.000166
At itr=7, x_itr=1.000000, y_itr=1.000000, z_itr=2.000000, error=0.000024

The solution of the given system is
x =
2.0000
4.0000
3.0000

The eigen values of the convergence matrix H are
E =
0
-0.1000
0.1250

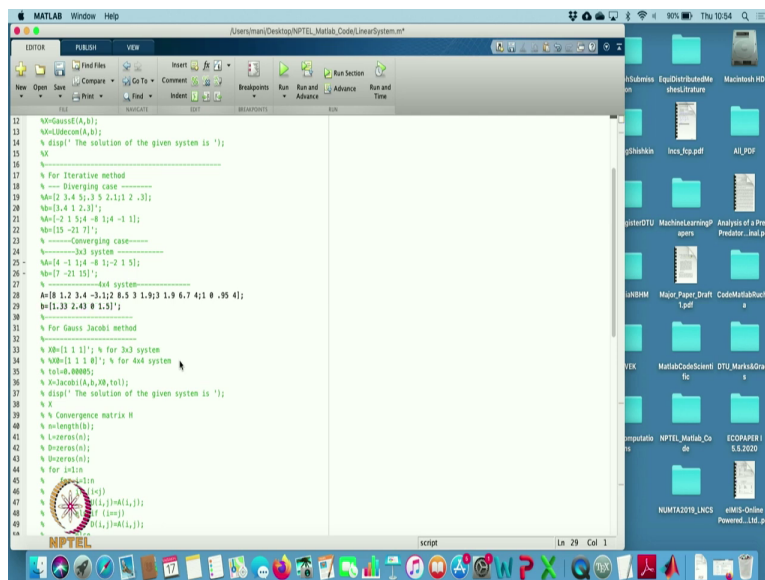
Spectral radius and Rate of convergence are = 0.125000 and 2.079442
lambda =

```

So, in this case also if you see 0 0 1, the same initial condition means changing the initial condition, but the number of iterations is the same and the rate of convergence is the same. So this is the way we can find out how for the different initial conditions our method is converging because from here we can see that if I take the convergence matrix, then the eigenvalue of the convergence matrix is less than 1, the modulus of that one.

So, and the small is the value the faster the method will be. So in this case the eigenvalue, the maximum eigenvalue is 0.125. So that is why you can see that the order of convergence is quite high and that is a second-order method in this case. So this is the 3 by 3 system I have taken.

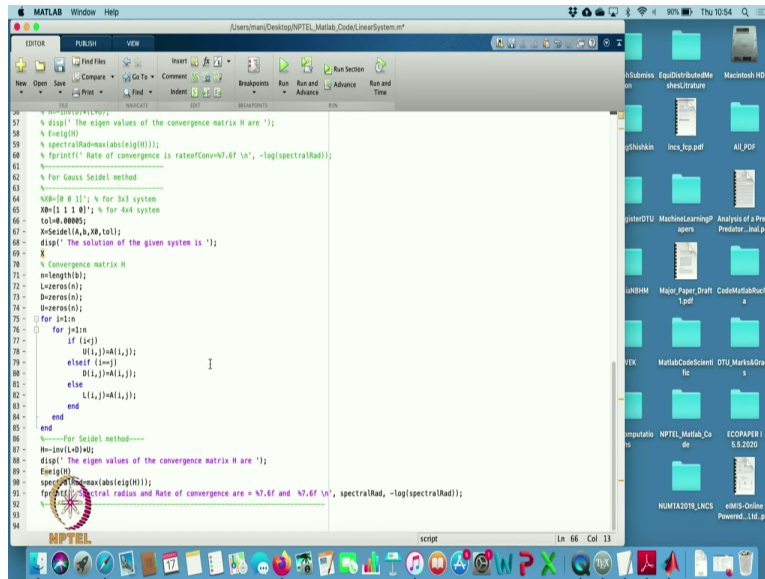
(Refer Slide Time: 21:10)



```

12 % Gauss (A,b);
13 % LU decomposition;
14 % disp('The solution of the given system is ');
15 %
16 %
17 % For Iterative method
18 % ----- Diverging case -----
19 % M=[2 3.4 5; 3 2.1 2 -3];
20 % N=[3.4 1 2.3];
21 % M=[2 1 5; 4 -8 1; 4 -1 1];
22 % M=[15 -21 7];
23 % ----- Converging case -----
24 % ----- Jacobi system -----
25 % M=[4 -1 1; 4 -8 1; -2 1 5];
26 % M=[17 -21 15];
27 % ----- Gauss-Jordan system -----
28 % A=[8 1.2 3.4 -3.12 6.5 3 1.9; 3 1.9 6.7 4; 1 0 .95 4];
29 % b=[1.33 2.43 0 1.51];
30 %
31 % For Gauss-Jacobi method
32 %
33 % M=[1 1 1]; % for Jacobi system
34 % N=[1 1 4]; % for Jacobi system
35 % tol=0.00005;
36 % x=Jacobi(A,b,N,tol);
37 % disp('The solution of the given system is ');
38 % x
39 % % Convergence matrix H
40 % x=length(x);
41 % L=zeros(x);
42 % D=zeros(x);
43 % U=zeros(x);
44 % for i=1:x
45 %     for j=1:x
46 %         L(i,j)=M(i,j);
47 %         if i==j
48 %             D(i,j)=M(i,j);
49 %         else
50 %             U(i,j)=M(i,j);
51 %         end
52 %     end
53 % end
54 %
55 %
56 %
57 %
58 %
59 %
60 %
61 %
62 %
63 %
64 %

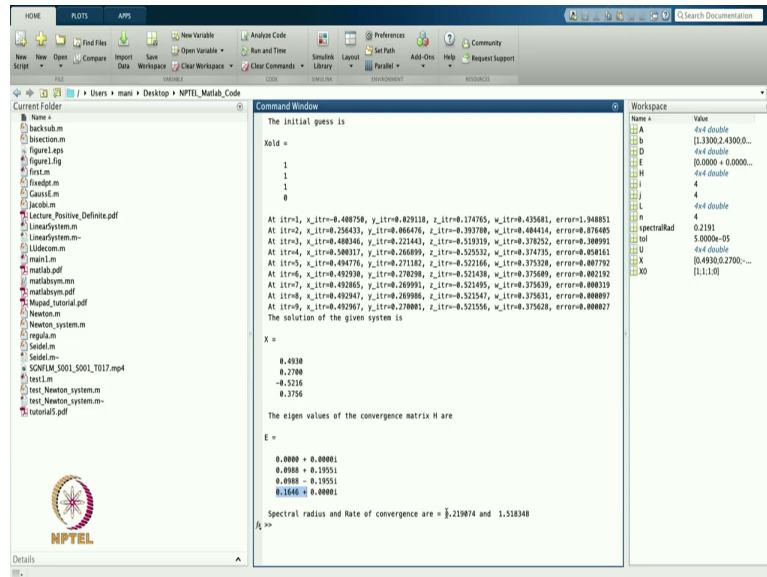
```



```
57 % disp(' The eigen values of the convergence matrix H are ');
58 % E=eig(H)
59 % spectralRad=max(abs(eig(H)));
60 % fprintf(' Rate of convergence is rateOfConv%7.6f \n', -log(spectralRad));
61
62 % For Gauss Seidel method
63
64 N=nz([0 0 1]); % for 3x3 system
65 M=[1 1 0]; % for 4x4 system
66 tol=0.00005;
67 x=Seidel(A,b,M,tol);
68 disp(' The solution of the given system is ');
69 x
70
71 % Convergence matrix H
72 n=length(b);
73 L=zeros(n);
74 D=zeros(n);
75 for i=1:n
76     for j=1:n
77         if (i==j)
78             L(i,j)=A(i,j);
79         elseif (i==j)
80             D(i,j)=A(i,j);
81         else
82             L(i,j)=A(i,j);
83         end
84     end
85 end
86 %-----for Seidel method-----
87 M=[0 0 0 0];
88 % disp(' The eigen values of the convergence matrix H are ');
89 % E=eig(H)
90 % spectralRad=max(abs(eig(H)));
91 % fprintf(' Spectral radius and Rate of convergence are = %7.6f and %7.6f \n', spectralRad, -log(spectralRad));
92
93
94
```

Now, let us try to take the, another system, so let us take 4 by 4. So 4 by 4 system, but in the initial condition, I have to change. So this is the initial condition 4 by 4, so I will take this one as a, because that vector, initial vector should be of four components, so this is x, y, z, and w and the tolerance I am not changing, I am put keeping the same tolerance. So my system is now 4 cross 4, so let us run this one.

(Refer Slide Time: 21:51)



And from here, if you see, now I have started with the 4 by 4 matrix and the initial approximation of the solution I am starting with 1 1 1 0. Now, you can see that iteration number 1, this value is at the initial point. Then after the iteration, each iteration, this is converging very fast. Even after the fifth iteration, you can see that it is going close to the root, and after a few iterations that is the iteration that our solution is now reaching to the tolerance.

So, in this case, my x component has 0.49, so 0.5 is the value, 0.27 -0.52, and 0.37. So that is this one and error, you can see that my error is 0.0002, so it is less than the given tolerance. So that is why we stopped here. And now, from here you can see that the eigenvalues I am finding. So these are the eigenvalues I am finding and from here, you can see that one eigenvalue it is finding is 0, another is the complex eigenvalue; x component is this, and y component is this.

Now again, and we know that the eigenvalues if they are complex, they appear in a conjugate form, so that is the conjugate of this complex number. And then, I am getting another real eigenvalue that is 0.1645. So now, from here I take the modulus, absolute value of this one.

(Refer Slide Time: 23:26)

The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the results of solving a system of equations using the Newton-Raphson method. The solution is given as $x = 0.4930$, $y = 0.2780$, $z = -0.5216$, and $w = 0.3756$. The eigenvalues of the convergence matrix H are listed as $0.0000 + 0.0000i$, $0.0000 + 0.1955i$, $0.0000 - 0.1955i$, and $0.1640 + 0.0000i$. The spectral radius and rate of convergence are calculated as 0.219074 and 1.518348 respectively. The Workspace shows the variables A , b , D , H , L , n , $spectralRad$, tol , U , X , and $X0$.

```

At t=0, x_itr=0.408759, y_itr=0.829118, z_itr=0.174765, w_itr=0.435681, error=1.948851
At t=2, x_itr=0.256433, y_itr=0.066476, z_itr=0.393788, w_itr=0.484414, error=0.876485
At t=4, x_itr=0.488346, y_itr=0.221443, z_itr=0.519319, w_itr=0.378752, error=0.380991
At t=6, x_itr=0.380317, y_itr=0.260899, z_itr=0.525532, w_itr=0.374735, error=0.491161
At t=8, x_itr=0.494776, y_itr=0.271182, z_itr=0.522166, w_itr=0.375328, error=0.007792
At t=10, x_itr=0.492938, y_itr=0.270288, z_itr=0.521438, w_itr=0.375689, error=0.002192
At t=12, x_itr=0.492805, y_itr=0.269991, z_itr=0.521489, w_itr=0.375639, error=0.00019
At t=14, x_itr=0.492847, y_itr=0.269986, z_itr=0.521547, w_itr=0.375631, error=0.000097
At t=16, x_itr=0.492867, y_itr=0.270001, z_itr=0.521556, w_itr=0.375628, error=0.000027

The solution of the given system is

X =

    0.4930
    0.2780
   -0.5216
    0.3756

The eigen values of the convergence matrix H are

E =

    0.0000 + 0.0000i
    0.0000 + 0.1955i
    0.0000 - 0.1955i
    0.1640 + 0.0000i

Spectral radius and Rate of convergence are = 0.219074 and 1.518348
>> E(1)

ans =

    0

>> abs(E(2))

ans =

    0.2191
  
```

So after getting the absolute value, I find that the spectral radius is coming because this is the E value I am getting, so E1 is 0. So I just check that what I will do, the absolute value of E2 is point, see this is the 0.2191, so that is the absolute value of this.

So this is, and then using this one I find the rate of convergence. So the rate of convergence, in this case, is 1.51. So because here, the eigenvalue, the spectral radius has increased, in the previous case it was 0.1, now it is 0.21, so it is going close to, means as compared to the previous one, this value is greater. So that is why the order of convergence or the rate of convergence has decreased from 2 to 1.5. So this way we can find the solution for a 4 by 4 system.

(Refer Slide Time: 24:41)

```

57 % disp('The eigen values of the convergence matrix H are ');
58 % E=eig(H);
59 % spectralRadius=eig(H));
60 % print('Rate of convergence is rateOfConv=7.6f %', -log(spectralRad));
61
62 % For Gauss Seidel method
63
64 N=[0 0 1]; % for 3x3 system
65 X=[1 1 0]; % for 4x4 system
66 tol=0.0001;
67 X=Seidel(A,B,tol);
68 disp('The solution of the given system is ');
69 X
70
71 % Convergence matrix H
72 m=length(b);
73 L=zeros(m);
74 U=zeros(m);
75 for i=1:m
76     for j=1:m
77         if i==j
78             L(i,j)=1;
79         else
80             L(i,j)=A(i,j);
81         else
82             U(i,j)=A(i,j);
83         end
84     end
85 end
86 % For Seidel method
87 H=inv(L+U);
88 disp('The eigen values of the convergence matrix H are ');
89 E=eig(H);
90 spectralRadius=max(abs(E));
91 % print('Spectral radius and Rate of convergence are = 7.6f and 7.6f %', spectralRad, -log(spectralRad));
92
93
94

```

Now the same thing I can take, now what do I do, I will reduce the tolerance now. I am keeping the same initial condition, but I am reducing the tolerance. So let us see what will happen.

(Refer Slide Time: 24:55)

```

The initial guess is
X0 =
1
1
0

At itr=1, x_itr=0.488758, y_itr=0.829118, z_itr=0.174765, w_itr=0.435681, error=1.948851
At itr=2, x_itr=0.256433, y_itr=0.866476, z_itr=0.393789, w_itr=0.484414, error=0.876485
At itr=3, x_itr=0.488346, y_itr=0.221443, z_itr=0.519319, w_itr=0.378252, error=0.380991
At itr=4, x_itr=0.388317, y_itr=0.266899, z_itr=0.525332, w_itr=0.374735, error=0.40161
At itr=5, x_itr=0.484776, y_itr=0.271182, z_itr=0.522166, w_itr=0.373328, error=0.807792
At itr=6, x_itr=0.432938, y_itr=0.270288, z_itr=0.521438, w_itr=0.375889, error=0.802192
At itr=7, x_itr=0.432865, y_itr=0.269991, z_itr=0.521495, w_itr=0.375439, error=0.808319

The solution of the given system is
X =
0.4329
0.2708
-0.5225
0.3756

The eigen values of the convergence matrix H are
E =
0.0000 + 0.0000i
0.0000 + 0.1955i
0.0000 - 0.1955i
0.1646 + 0.0000i

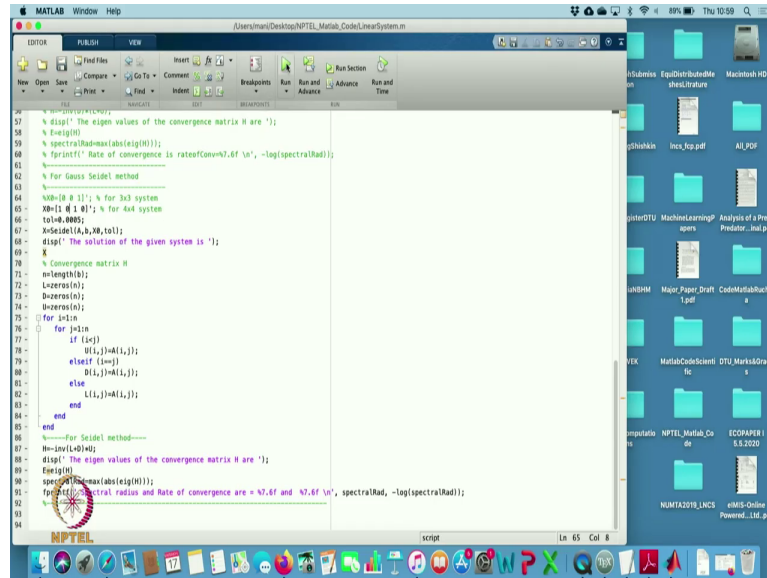
Spectral radius and Rate of convergence are = 0.219074 and 1.518348

```

So in this case, now you can see that the tolerance I have decreased by one factor only. Now, the number of iterations has reduced from the previous one, so it is, now it is seven iteration and the solution is, is accurate up to four decimal.

So this is, so less than 10^{-4} , so now this is my error. So the number of iterations has reduced and this is my solution. Now, from here you can see that this is because this is not going to change with the, with respect to the initial condition or the tolerance, so that is not going to change and the rate of convergence will be the same, that is 1.5. Only the change will be the number of iterations. So in this case, the number of iterations are the same.

(Refer Slide Time: 25:54)



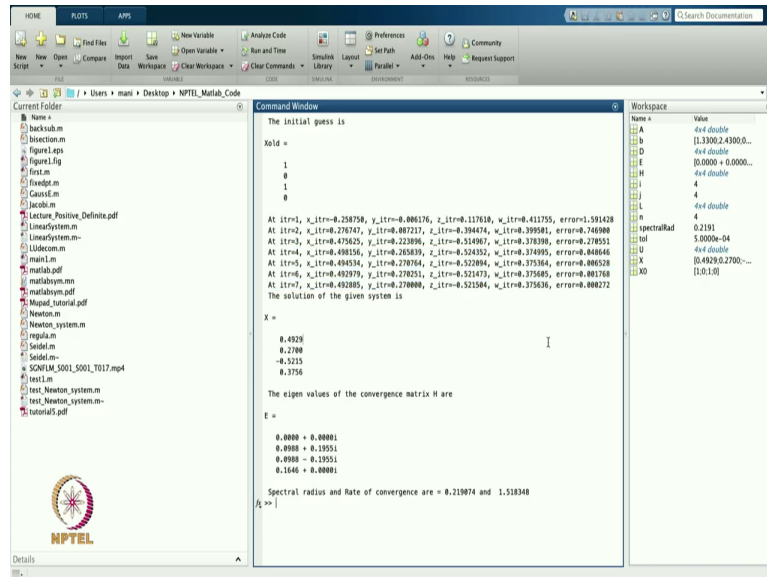
```

57 % disp('The eigen values of the convergence matrix H are ');
58 % E=eig(H);
59 % spectralRad=max(abs(eig(H)));
60 % fprintf('Rate of convergence is rateOfConv=7.6f \n', -log(spectralRad));
61 %-----
62 % For Gauss Seidel method
63 %-----
64 %N=[0 0 1]; % for 3x3 system
65 %X0=[1 0 1 0]; % for 4x4 system
66 %tol=0.0005;
67 %ndesdel(1a,2,X0,tol);
68 disp('The solution of the given system is ');
69 %
70 % Convergence matrix H
71 n=length(b);
72 L=zeros(n);
73 U=zeros(n);
74 %-----
75 for i=1:n
76     for j=1:n
77         if (i<j)
78             U(i,j)=A(i,j);
79         elseif (i==j)
80             L(i,j)=A(i,j);
81         else
82             L(i,j)=A(i,j);
83         end
84     end
85 end
86 %-----For Seidel method-----
87 H=inv(L+U)*U;
88 disp('The eigen values of the convergence matrix H are ');
89 %E=eig(H);
90 %spectralRad=max(abs(eig(H)));
91 %fprintf('Spectral radius and Rate of convergence are = 7.6f and 7.6f \n', spectralRad, -log(spectralRad));
92
93
94

```

Now what will happen, let us see, if I change the initial condition 1 0 1 0.

(Refer Slide Time: 26:05)



So, in this case, $\begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$, the same number of iterations we are getting now, 0.7, the iterations are 7 and the error has reduced from point in the previous one it was 0.003, now it is 0.002, so the solution is this one. So that is my approximate solution after seven iterations. So in this case, the solution is converging because you can see from here that after three iterations you will get the knowledge of what will be the solution.

See the power of this method that even after three iterations you can see that the solution is converging toward 0.49, this is converging to 0.27, this is converging to -0.52, and that is 0.37. So even after three iterations, you were able to see or guess that where our solution is going to have to lie. So that is the power of the Gauss-Seidel method.

(Refer Slide Time: 27:20)


```

16 % For iterative method
17 % ----- Diverging case -----
18 Mo=[2 3.4 5; 3 5 2; 1 2 -3];
19 Mo=[3.4 1 2; 3];
20 Mo=[2 1 3; 4 -8 1; 4 -1 1];
21 Mo=[15 -21 7];
22 % ----- Converging case -----
23 % ----- 3x3 system -----
24 Mo=[4 -1 1; 4 -8 1; 2 1 5];
25 Mo=[7 -21 15];
26 % ----- 4x4 system -----
27 Mo=[8 1.2 3.4 -0.1; 2 8.5 3 1.9; 3 1.9 6.7 4; 1 0 -95 4];
28 Mo=[1.33 2.43 0 1.5];
29 % let us take another 4x4 system
30 A=[1 2 3 0; 1 -1 0 -5 -1.5; 0 1 0 .2; 25 -3 0 0];
31 b=[1 0 .3 1.5];
32 % For Gauss Jacobi method
33 % Mo=[1 1 1]; % for 3x3 system
34 % Mo=[1 1 1 0]; % for 4x4 system
35 % tol=0.0005;
36 % x=Jacobi(A,b,tol);
37 % disp('The solution of the given system is ');
38 % x
39 % % Convergence matrix H
40 % n=length(b);
41 % L=zeros(n);
42 % D=zeros(n);
43 % U=zeros(n);
44 % for j=1:n
45 %     for i=1:n
46 %         if i==j
47 %             D(i,i)=1;
48 %         else
49 %             L(i,j)=A(i,j);
50 %             U(i,j)=A(i,j);
51 %         end
52 %     end
53 % end
54 % % For Gauss Seidel method
55 % Mo=[1 1]; % for 3x3 system
56 % Mo=[1 1 0]; % for 4x4 system
57 % tol=0.0005;
58 % x=Seidel(A,b,tol);
59 % disp('The solution of the given system is ');
60 % x
61 % % Convergence matrix H
62 % L=zeros(n);
63 % D=zeros(n);
64 % U=zeros(n);
65 % for j=1:n
66 %     for i=1:n
67 %         if i==j
68 %             D(i,i)=1;
69 %         else
70 %             L(i,j)=A(i,j);
71 %             U(i,j)=A(i,j);
72 %         end
73 %     end
74 % end
75 % % For Gauss Seidel method
76 % Mo=[1 1]; % for 3x3 system
77 % Mo=[1 1 0]; % for 4x4 system
78 % tol=0.0005;
79 % x=Seidel(A,b,tol);
80 % disp('The solution of the given system is ');
81 % x

```

```

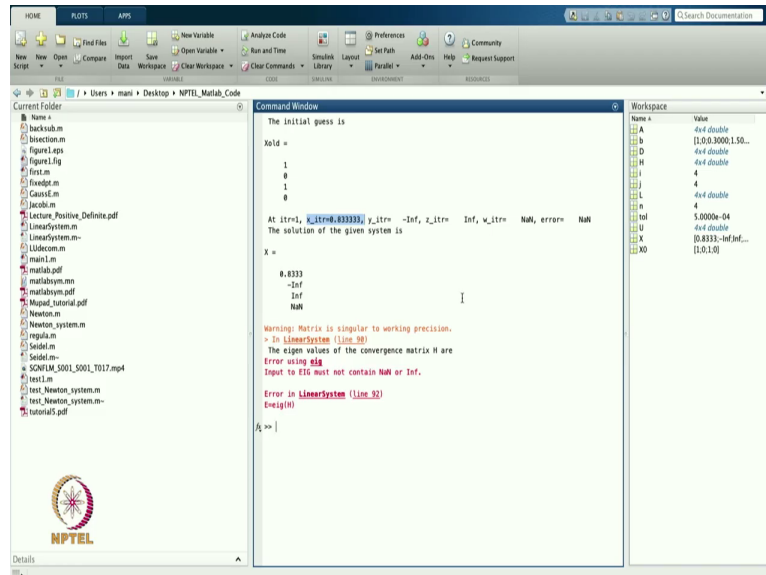
41 % % Convergence matrix H
42 % n=length(b);
43 % L=zeros(n);
44 % D=zeros(n);
45 % U=zeros(n);
46 % for j=1:n
47 %     for i=1:n
48 %         if i==j
49 %             D(i,i)=1;
50 %         else
51 %             L(i,j)=A(i,j);
52 %             U(i,j)=A(i,j);
53 %         end
54 %     end
55 % end
56 % % For Gauss Seidel method
57 % Mo=[1 1]; % for 3x3 system
58 % Mo=[1 1 0]; % for 4x4 system
59 % tol=0.0005;
60 % x=Seidel(A,b,tol);
61 % disp('The solution of the given system is ');
62 % x
63 % % Convergence matrix H
64 % L=zeros(n);
65 % D=zeros(n);
66 % U=zeros(n);
67 % for j=1:n
68 %     for i=1:n
69 %         if i==j
70 %             D(i,i)=1;
71 %         else
72 %             L(i,j)=A(i,j);
73 %             U(i,j)=A(i,j);
74 %         end
75 %     end
76 % end
77 % % For Gauss Seidel method
78 % Mo=[1 1]; % for 3x3 system
79 % Mo=[1 1 0]; % for 4x4 system
80 % tol=0.0005;
81 % x=Seidel(A,b,tol);
82 % disp('The solution of the given system is ');
83 % x
84 % % Rate of convergence is rateConver=7.67 % ln(-log(spectralRad));
85 % % For Gauss Seidel method
86 % Mo=[1 1]; % for 3x3 system
87 % Mo=[1 1 0]; % for 4x4 system
88 % tol=0.0005;
89 % x=Seidel(A,b,tol);
90 % disp('The solution of the given system is ');
91 % x
92 % % Convergence matrix H
93 % L=zeros(n);
94 % D=zeros(n);
95 % U=zeros(n);
96 % for j=1:n
97 %     for i=1:n
98 %         if i==j
99 %             D(i,i)=1;
100 %         else
101 %             L(i,j)=A(i,j);
102 %             U(i,j)=A(i,j);
103 %         end
104 %     end
105 % end
106 % % For Gauss Seidel method
107 % Mo=[1 1]; % for 3x3 system
108 % Mo=[1 1 0]; % for 4x4 system
109 % tol=0.0005;
110 % x=Seidel(A,b,tol);
111 % disp('The solution of the given system is ');
112 % x

```

So, in this case, I may, so that is a 4 by 4 system we have taken. Now, let us take another system. So let us take another 4*4 system. So I will write in front of you. So let us take the matrix a that is equal to, suppose I take the matrix 1.2, then 2.3, then 0, and then I take 1. So, that is the first row I take.

Then I take -0.1, then 0, then 0.5, then -1.5, so this is another row I am taking; then I take another row 0 1 0 0.2, so this is another one; and I now I take the last row. So the last row is 0.25 -0.3 0 0. So let us take this system where the matrix is this and then I have to take the right-hand side vector. So b, I am taking 1 0 0.3 1.5, so let us take this one, I am just taking it randomly. So let us try to solve this problem, so I run this one and then see what will happen.

(Refer Slide Time: 29:14)



The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the following text:

```
The initial guess is
xold =
    1
    0
    1
    0

At iter=1, x_iter=0.8333, y_iter=-Inf, z_iter=Inf, w_iter=NaN, error= NaN
The solution of the given system is
x =
    0.8333
   -2e2
    Inf
   NaN

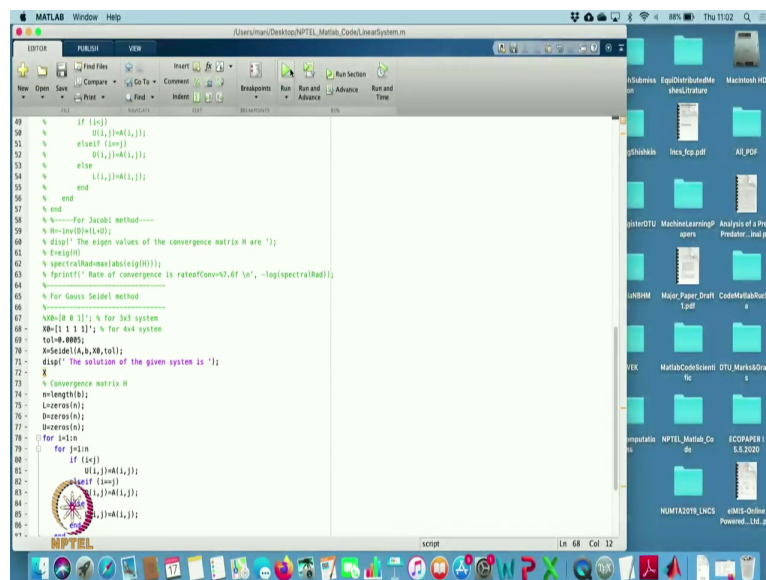
Warning: Matrix is singular to working precision.
> In LinearSystem (line 98)
The eigen values of the convergence matrix H are
Error using eig
Input to eig must not contain NaN or Inf.
Error in LinearSystem (line 92)
eig(H)
```

The Workspace shows the following variables:

Name	Value
A	4x4 double
b	[1.0;0.3000;1.50...
D	4x4 double
H	4x4 double
i	4
j	4
L	4x4 double
n	4
tol	1.0000e-04
U	4x4 double
X	[0.8333;-2e2;Inf;...
X0	[1.0;1.0]

Now, from here the matrix and the solution, it is coming that the matrix is singular and the, so you can see that even after the first iteration this solution x is gaining and the y, z, and w, they are coming not a number.

(Refer Slide Time: 29:34)

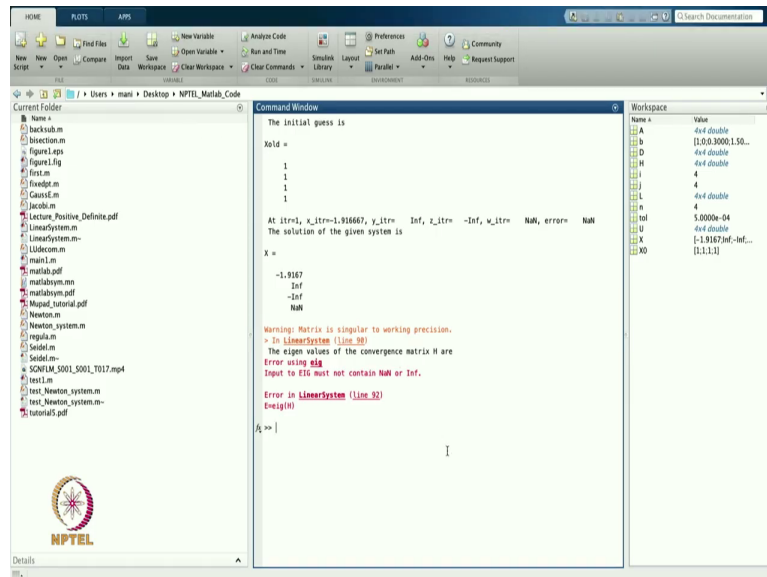


The screenshot shows the MATLAB Editor with the code for the LinearSystem.m script. The code is as follows:

```
% if (i==1)
%   H(i,j)=H(i,j);
%   elseif (i==j)
%   H(i,j)=H(i,j);
%   else
%   H(i,j)=H(i,j);
%   end
% end
% end
% For Jacobi method
% H=inv(D+(L+U));
% disp('The eigen values of the convergence matrix H are ');
% eig(H)
% spectralRadius=max(abs(eig(H)));
% print('Rate of convergence is rateoffconv=7.0f %', -log(spectralRadius));
% For Gauss Seidel method
% H=H+(L+U); % for Jac system
% H=H+(L+U); % for Jac system
% H=H+(L+U); % for Jac system
% disp('The solution of the given system is ');
% x
% Convergence matrix H
% m=length(b);
% L=zeros(m);
% U=zeros(m);
% L=zeros(m);
% U=zeros(m);
% for j=1:m
%   for i=j+1:m
%     if (i==j)
%       H(i,j)=H(i,j);
%     elseif (i==j)
%       H(i,j)=H(i,j);
%     else
%       H(i,j)=H(i,j);
%     end
%   end
% end
```

So let us change this matrix or maybe, I can take this initial condition, change in initial condition, so the same thing, so let us see.

(Refer Slide Time: 29:47)



The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the following text:

```
The initial guess is
Xold =
    1
    1
    1
    1

At itr=1, x_itr=-1.91667, y_itr= Inf, z_itr= -Inf, w_itr= NaN, error= NaN

The solution of the given system is

X =
   -1.9167
        Inf
       -Inf
        NaN

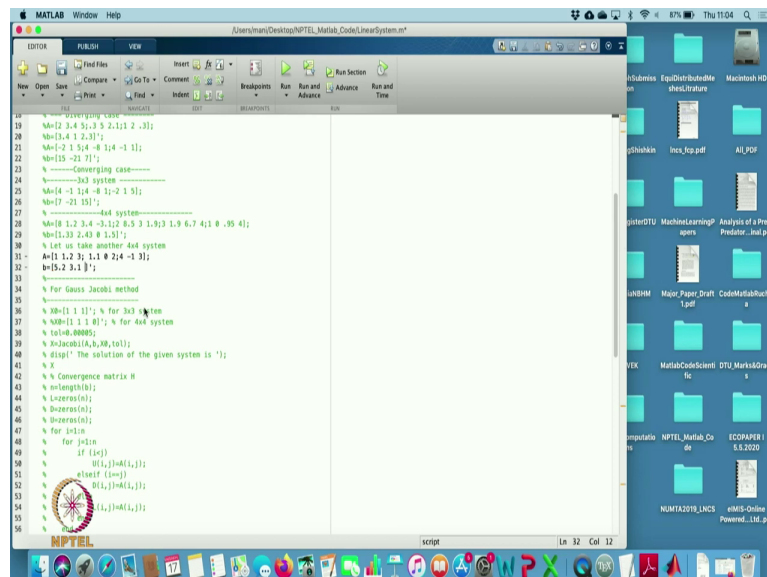
Warning: Matrix is singular to working precision.
> In LinearSystem (line 30)
The eigen values of the convergence matrix H are
Error using eig
Input to eig must not contain NaN or Inf.
Error in LinearSystem (line 32)
eig(H)
```

The Workspace shows the following variables:

Name	Value
A	4x4 double [1.0 0.3000 1.50...
b	4x4 double [1.0 0.3000 1.50...
D	4x4 double [1.0 0.3000 1.50...
H	4x4 double [1.0 0.3000 1.50...
i	4
j	4
L	4x4 double [1.0 0.3000 1.50...
n	4
tol	5.0000e-04
U	4x4 double [1.0 0.3000 1.50...
X	4x4 double [-1.9167 Inf...
X0	[1.1 1.1]

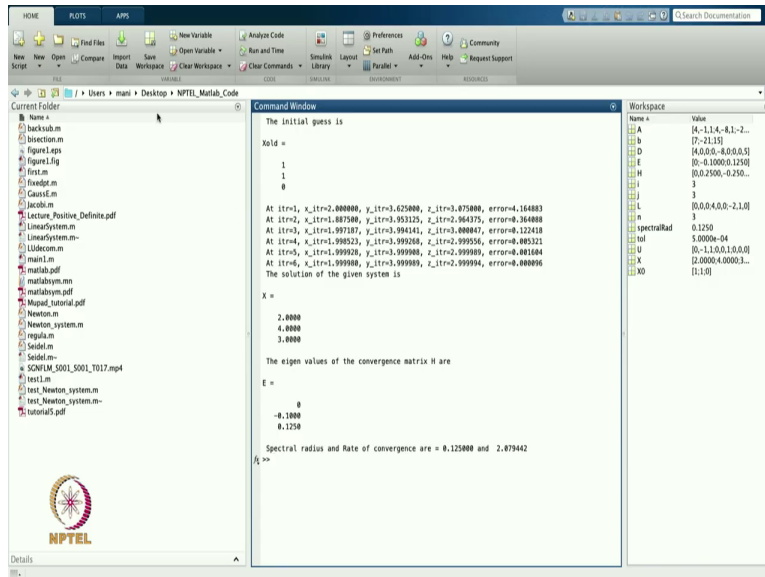
So the same thing is happening now. It means that this system is not solvable with the help of Gauss-Seidel method.

(Refer Slide Time: 30:01)



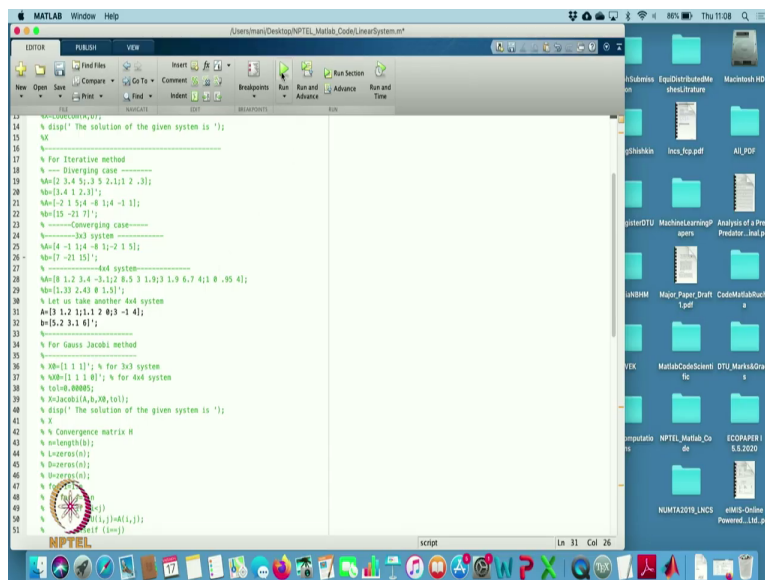
The screenshot shows the MATLAB Editor with the following code:

```
18 % Gauss-Seidel method
19 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
20 % b=[1 2 3 4];
21 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
22 % b=[1 2 3 4];
23 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
24 % b=[1 2 3 4];
25 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
26 % b=[1 2 3 4];
27 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
28 % b=[1 2 3 4];
29 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
30 % b=[1 2 3 4];
31 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
32 % b=[1 2 3 4];
33 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
34 % b=[1 2 3 4];
35 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
36 % b=[1 2 3 4];
37 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
38 % b=[1 2 3 4];
39 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
40 % b=[1 2 3 4];
41 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
42 % b=[1 2 3 4];
43 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
44 % b=[1 2 3 4];
45 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
46 % b=[1 2 3 4];
47 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
48 % b=[1 2 3 4];
49 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
50 % b=[1 2 3 4];
51 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
52 % b=[1 2 3 4];
53 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
54 % b=[1 2 3 4];
55 % A=[2 3 4 5; 3 5 2 1; 2 3 1 2; 3 1 2 3];
56 % b=[1 2 3 4];
```

So let us take the previous case, let us see what is happening because it may happen that I have, now it is giving the solution for the given matrix. So in this case, the solution is there, actually, maybe I am not taking a matrix with a diagonal dominance, so let us, so I am taking this.

(Refer Slide Time: 31:55)



So let us change this one to diagonal dominance because our matrix, if the matrix is diagonally dominant, so let us take this as a 3 and this is 1. So I take 1.1 and then 2 and then 0, and then I take this 3 and then I take 4, maybe. So, in this case, this is a diagonally dominant 1.1 2 0, so that is the matrix and 3 -1, so maybe I can take this one. So let us see what will happen.

(Refer Slide Time: 32:48)

```
The initial guess is
xold =
    1
    1
    0

At iter=1, x_iter=1.333333, y_iter=0.816667, z_iter=0.784167, error=0.000758
At iter=2, x_iter=1.171844, y_iter=0.985431, z_iter=0.847389, error=0.233326
At iter=3, x_iter=1.008695, y_iter=0.951218, z_iter=0.921283, error=0.128357
At iter=4, x_iter=1.003752, y_iter=0.974837, z_iter=0.955395, error=0.062084
At iter=5, x_iter=1.021608, y_iter=0.987828, z_iter=0.979855, error=0.032825
At iter=6, x_iter=1.012174, y_iter=0.993384, z_iter=0.989196, error=0.016528
At iter=7, x_iter=1.008389, y_iter=0.998546, z_iter=0.994427, error=0.008321
At iter=8, x_iter=1.003239, y_iter=0.998218, z_iter=0.997225, error=0.004396
At iter=9, x_iter=1.001871, y_iter=0.998081, z_iter=0.998317, error=0.002267
At iter=10, x_iter=1.000802, y_iter=0.999526, z_iter=0.999235, error=0.001178
At iter=11, x_iter=1.000445, y_iter=0.999755, z_iter=0.999685, error=0.000683
At iter=12, x_iter=1.000229, y_iter=0.999874, z_iter=0.999796, error=0.000311

The solution of the given system is

X =

    1.0002
    0.9999
    0.9998

The eigen values of the convergence matrix H are

E =

     0
    0.0000
    0.5158

Spectral radius and Rate of convergence are = 0.515833 and 0.662972
```

Yeah. So now it is giving the solution. So after the twelve iterations, you are getting the solution. So, that was the problem because I have chosen a random matrix and that was not diagonally dominant. So from here, you can see that the solution is going to be 1, close to 1 and the matrix was singular, so it is 0. So both two eigenvalues are 0 basically and this is 0.51. So if the, it is going to be very large as compared to the previous cases, so in this, you can see the rate convergence has increased to be even less than linear value, so it is 0.6 value.

So now we will stop here. So today, we have discussed the MATLAB code for Gauss-Seidel method and with the help of Gauss-Seidel method, we have tried to solve the system, 3 by 3 system or 4 by 4 system. And we also found that if a system is not diagonally dominant then the, it may or may not converge. So in that previous case, it was not converging but when we made the matrix as a diagonally dominant, then it was giving the solution in the twelve iteration.

So in the next lecture, we will continue with the other problems. So thanks for viewing. Thanks very much.