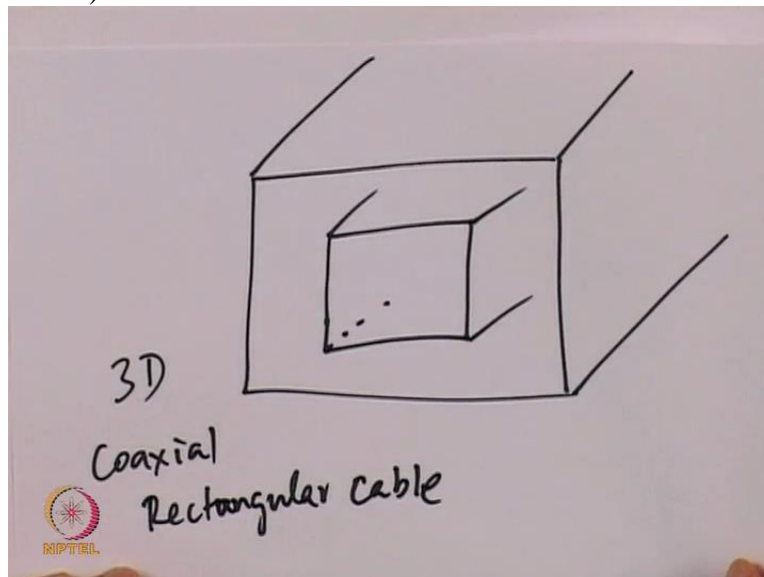


Computational Electromagnetics and Applications
Professor Krish Sankaram
Indian Institute of Technology, Bombay
Exercise No 5
Finite Difference Methods-II

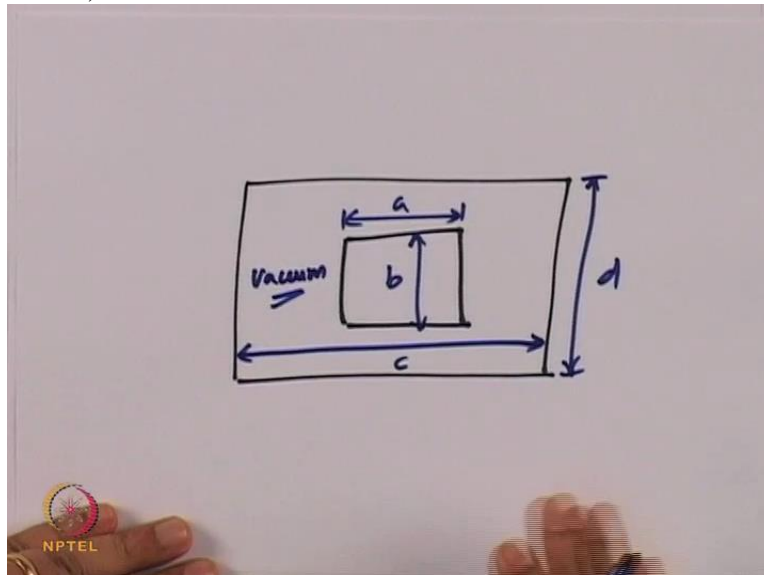
We are going to look into one of the very common problems in Electromagnetics which is capacitance problem. So today we are going to use this capacitance problem and we are going to use Finite Difference Method to solve for capacitance in a coaxial rectangle. So a problem geometry for today is going to be as follows:

(Refer Slide Time: 00:40)



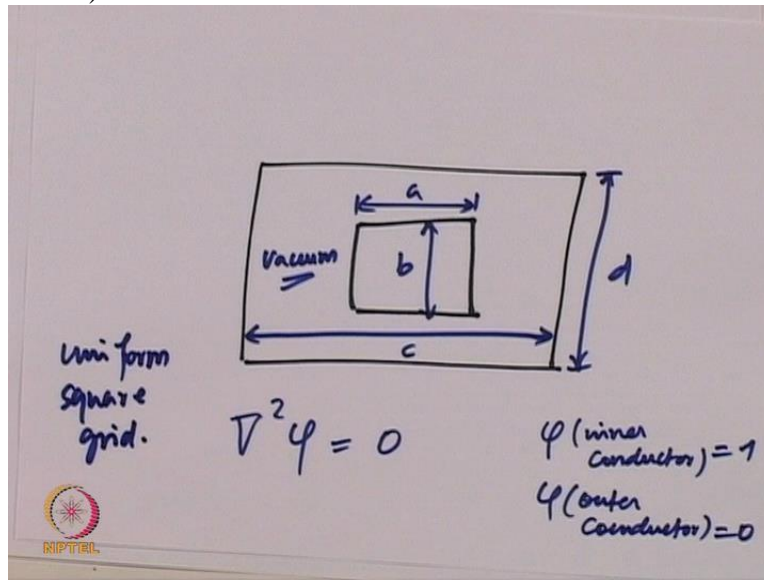
So we have coaxial rectangle and it is extending infinitely long in one direction, and we have a inner conductor which is also extending infinitely long in one direction. So this problem can be simplified into a two dimensional problem, and that is what we are going to do. So this is a 3D coaxial rectangular cable. What we are interested is finding the capacitance per unit length of this coaxial pair of rectangles and we are going to approximate this into a 2D problem. And our problem is going to be as follows:

(Refer Slide Time: 01:43)



So you will have same rectangle you are taking one cross section and we are approximating it using the 2D approximation so that once we find the solution in this case it will give us also the 3D solution accordingly. So what is happening here is we are taking a cross section and the dimensions of this particular cross section are as follows. The width is going to be given by a , the height is going to be given by b , for the inner rectangle; and similarly for the outer rectangle the width is going to be c , and the height is going to be given by d , and in between we have vacuum. So once we have this problem we are going to use Finite Difference method but what is going to be the governing equation.

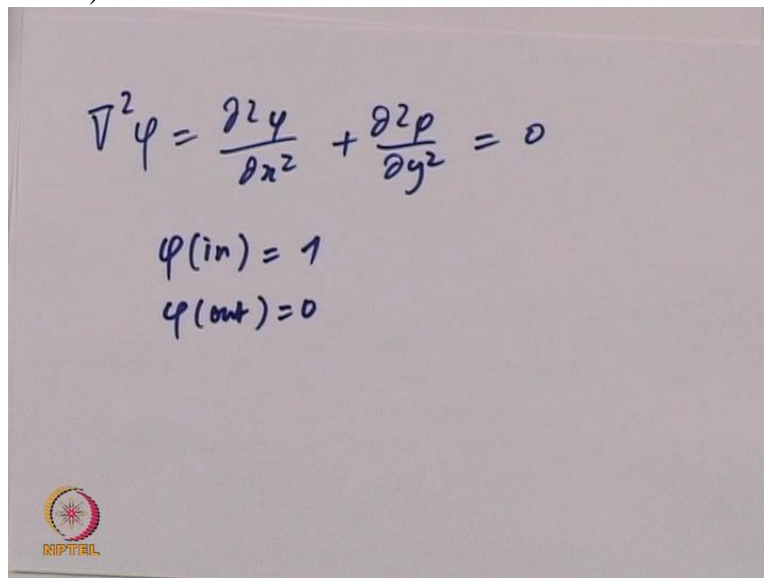
(Refer Slide Time: 02:52)



The governing equation for this particular problem is going to be the Laplace equation which is $\nabla^2 \phi = 0$. Of course we have certain potential values that we are going to assign to the inner conductor and the outer conductor. So we are going to keep the potential of the inner conductor equal to 1 volt and the potential of the outer conductor is going to be 0.

So this is the boundary condition we are choosing and we are going to use Finite Difference Method so where we are going to discretize this particular problem using uniform square grids.

(Refer Slide Time: 03:55)



So let us go to the 2D form of this Laplace equation so the 2D form of the Laplace equation is going to be $\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$.

Do y square is equal to 0. With the potential Φ inner is going to be 1 and Φ outer conductor is going to be 0. So this is the boundary condition.

(Refer Slide Time: 04:28)

A hand is holding a black marker, pointing to the equation. The equation is written on a whiteboard. In the bottom left corner, there is a logo for NIPTEEL.

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(i+1, j) - 2\phi(i, j) + \phi(i-1, j)}{\Delta x^2}$$

CD

So the central differencing method is going to approximate the value of for example, this particular term we are going to (Do x square Φ by Do x square equal to Φ (i plus 1, j minus 2 Φ (i , j) plus Φ of (i minus 1, j) divided by Δx square). So this is the Central Differencing scheme which we are using.

(Refer Slide Time: 05:10)

A hand is holding a black marker, pointing to the equations. The equations are written on a whiteboard. In the bottom left corner, there is a logo for NIPTEEL.

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(i+1, j) - 2\phi(i, j) + \phi(i-1, j)}{\Delta x^2}$$

CD

$$\frac{\partial^2 \phi}{\partial y^2} = \frac{\phi(i, j+1) - 2\phi(i, j) + \phi(i, j-1)}{\Delta y^2}$$

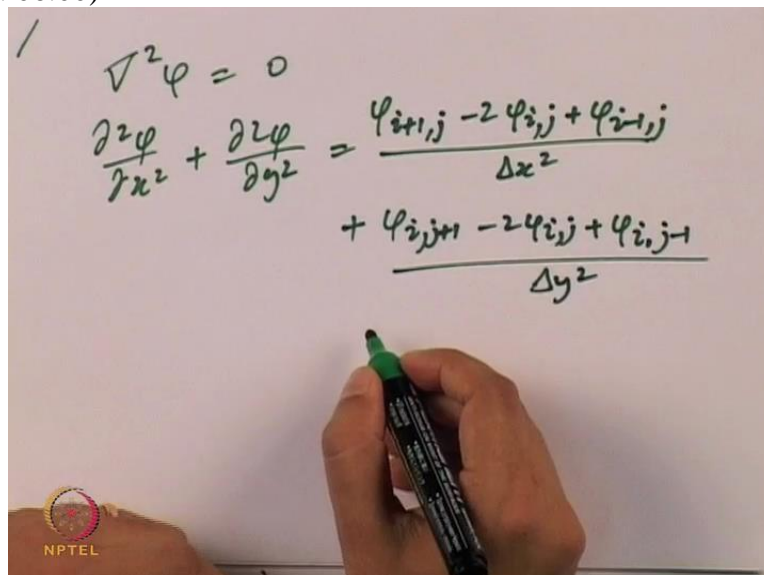
And Similarly for the sake of the other one we will do the same but for the wide term it will be $\Delta y^2 \Phi_{i,j} = \Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}$ divided by Δy^2 .

(Refer Slide Time: 05:30)



In this problem we are going to use a special type of iterative solver called as the Gauss Seidel Solver. The reason for using a special type of solver is to improve the order of convergence and also to quicken the computational time itself. So we will briefly introduce the method of Gauss Seidel which is used in this particular case.

(Refer Slide Time: 06:00)



So we have the Laplacian equation $\nabla^2 \phi = 0$. So if we do that in discrete two dimensional space, what you will get is $\Delta x^2 \phi_{i,j} + \Delta y^2 \phi_{i,j} = \phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j} + \phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}$ divided by Δx^2 plus Δy^2 . So we are taking the central differencing here for the first term with respect to x. So we have $i+1$ term here, similarly we do that for the j term plus $\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}$ divided by Δy^2 . So this is the discrete version using the Central Differencing scheme.

(Refer Slide Time: 06:55)

The image shows a handwritten derivation on a whiteboard. At the top left, it says $\nabla^2 \phi = 0$. Below that, the Laplacian is expressed as the sum of second-order partial derivatives: $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}$. This is equated to the central difference approximation: $\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2}$. The next line states $\Delta x = \Delta y = h$. Finally, the equation is simplified to $\nabla^2 \phi \approx \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{h^2} = 0$. An NPTEL logo is visible in the bottom left corner of the whiteboard image.

Now we are going to set Δx equal to Δy equal to h , if we do that what we can write is basically the equation purely in terms of h instead of Δx and Δy so we will get the Laplacian $\nabla^2 \phi$ equal to or approximately equal to $\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}$ divided by h^2 , this minus 4 comes by adding this $2\phi_{i,j}$. So once we do that we set this equal to 0, because the Laplacian should be equal to 0. If we do that we can see that the numerator should be 0 in order for us to satisfy this particular equation.

(Refer Slide Time: 08:08)

$$\nabla^2 \phi \approx \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{h^2} = 0$$
$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} = 4\phi_{i,j}$$
$$\phi_{i,j} = \frac{1}{4} (\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1})$$

So if we do that numerator should become equal to 0, so we will set the entire equation equal to 0 and we bring that last term to the other side. So what we will get is Phi of i plus 1,j plus Phi of i minus 1,j plus Phi i,j plus 1 plus Phi of i,j minus 1 is equal to 4 Phi of i,j. So now we can set Phi of i, j is equal to I am just bringing the left hand side to the right hand side and right hand side to the left hand side. What we will get is 1 by 4 of (Phi of i plus 1,j plus Phi i minus 1,j plus Phi of i,j plus 1 plus Phi of i,j minus 1). So now you know while we are doing the Matlab equation.

So this particular expression does not require the system of linear equation to be formed. And we can solve them explicitly. So whatever values we are having here on the left hand side of a time step n plus 1 will have only dependence on the time step at n. So this is the way we do the explicit formulation. Because of this only the solution must be stored in the computer memory and it allows us to solve larger problem given the amount of memory available on our computer. So that way it is memory wise efficient. But the problem with this is there is a slow convergence.

So this is something that is inherent to the Jacobi scheme. To solve that what we can do is we can go to the Gauss Siebel iteration where the old values of Phi are immediately overwritten as soon as the new ones are computed. So if Phi is updated in the order of increasing i and j.

(Refer Slide Time: 10:29)

$$\begin{aligned}\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1} &= 4\varphi_{i,j} \\ \varphi_{i,j} &= \frac{1}{4} (\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1}) \\ \varphi_{i,j}^{n+1} &= \frac{1}{4} (\varphi_{i+1,j}^{n+1} + \varphi_{i-1,j}^n + \varphi_{i,j+1}^{n+1} + \varphi_{i,j-1}^n)\end{aligned}$$

The Gauss Siebel method will be written as Φ of i,j at n plus 1 is equal to 1 by 4 Φ of i minus i,j n plus 1 plus Φ of i plus 1, j n plus Φ of i,j minus 1 n plus 1, Φ of i,j plus 1 at n). So this is the way we compute the Gauss Siebel scheme.

(Refer Slide Time: 11:13)

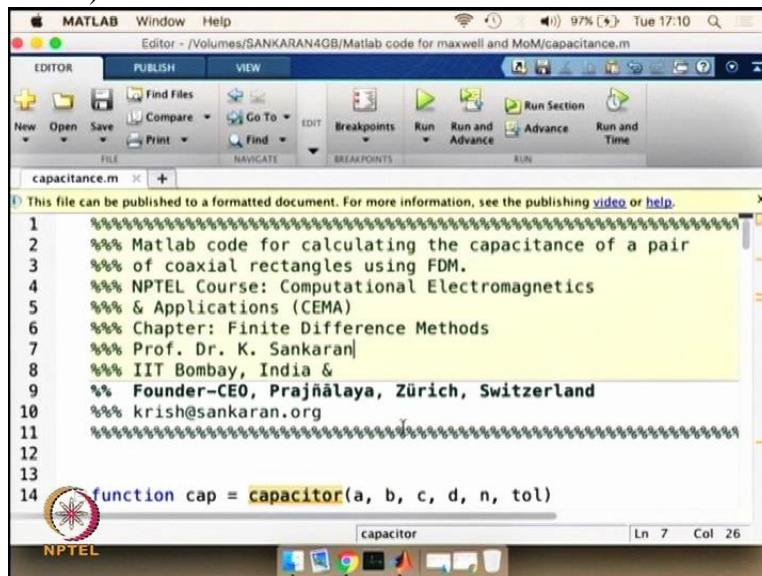
$$\begin{aligned}\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1} &= 4\varphi_{i,j} \\ \varphi_{i,j} &= \frac{1}{4} (\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1}) \\ \varphi_{i,j}^{n+1} &= \frac{1}{4} (\varphi_{i+1,j}^{n+1} + \varphi_{i-1,j}^n + \varphi_{i,j+1}^{n+1} + \varphi_{i,j-1}^n)\end{aligned}$$

$R > 1$ Improve ~~convergence~~ ^{convergence}
 $R < 2$ for stability

And for us the relaxation parameter to both the give us the convergence and also to serve us as a stability condition. So we want the R to be such that it also serves both purposes. But what happens when R goes more than 1 it improves the convergence and R less than 2 will provide the right stability. So what we want is R should be bounded between these two limits so that it serves both the convergence and also the stability. So that is the background of the Matlab code that we

are going to use and we are seeing now. And in case of need of certain clarity this particular discussion will help you for that.

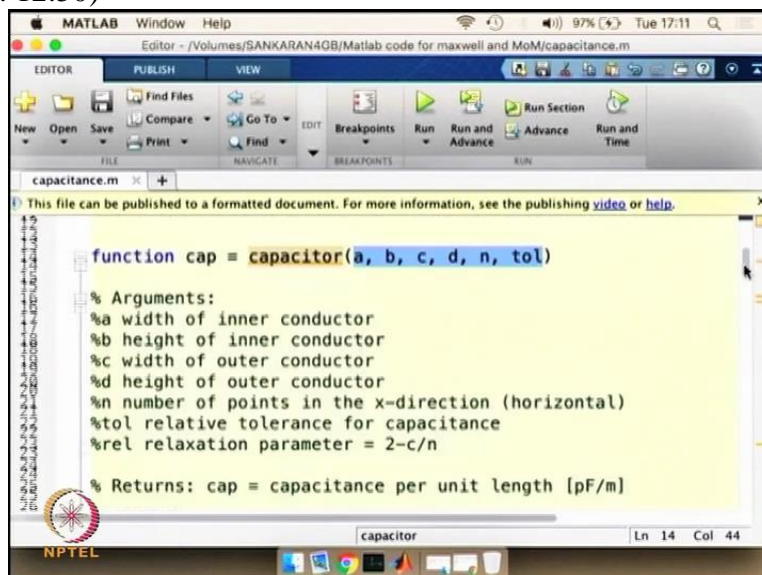
(Refer Slide Time: 12:16)



```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
2  %%% Matlab code for calculating the capacitance of a pair  
3  %%% of coaxial rectangles using FDM.  
4  %%% NPTEL Course: Computational Electromagnetics  
5  %%% & Applications (CEMA)  
6  %%% Chapter: Finite Difference Methods  
7  %%% Prof. Dr. K. Sankaran  
8  %%% IIT Bombay, India &  
9  %%% Founder-CEO, Prajñālaya, Zürich, Switzerland  
10 %%% krish@sankaran.org  
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
12  
13  
14 function cap = capacitor(a, b, c, d, n, tol)
```

So now let us go into the Matlab code itself and we are trying to use this geometry and this particular problem governing equation to solve the Matlab program which we have written. So the Matlab code what we have got is a code for calculating the capacitance of a pair of coaxial rectangles using Finite Differencing Method.

(Refer Slide Time: 12:50)



```
function cap = capacitor(a, b, c, d, n, tol)  
  
% Arguments:  
%a width of inner conductor  
%b height of inner conductor  
%c width of outer conductor  
%d height of outer conductor  
%n number of points in the x-direction (horizontal)  
%tol relative tolerance for capacitance  
%rel relaxation parameter = 2-c/n  
  
% Returns: cap = capacitance per unit length [pF/m]
```

And the code accepts certain inputs here a, b, c, d, n and tol.

And we have described these inputs as follows:

a is the width of the inner conductor

b is the height of the inner conductor

c is the width of the outer conductor

And d is the height of the outer conductor

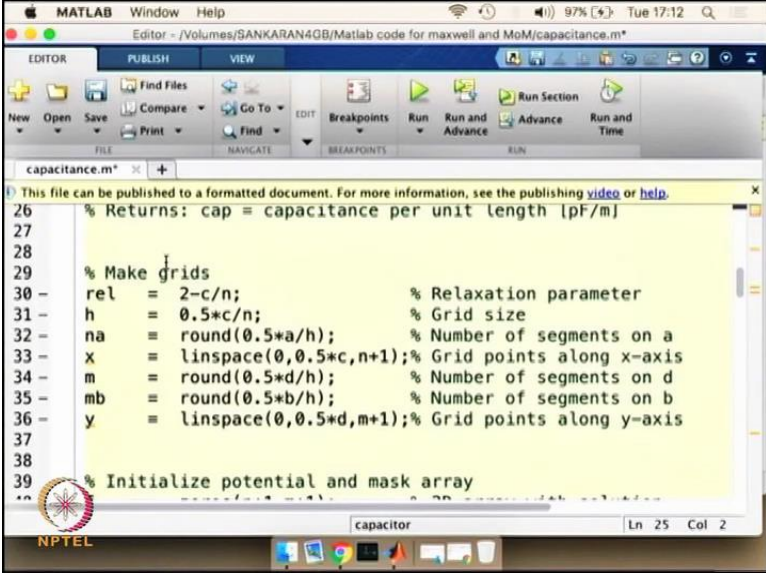
n is the number of points in the x direction horizontal direction, so n is going to be a variable that we can change for us to get different discretization

And tol is going to be the relative tolerance of the capacitance. So normally we choose a value which is in the range of 1 multiplied by 10 to the power minus 9. So we can put them in here approximately.

And there is going to be a value that is going to be calculated in the code and this value is going to depend on the inputs what we are giving. Value calculated based on input arguments. And this value is called as relaxation parameter. And the relaxation parameter is going to depend on two of the input parameters c and n. As you will see when we change c and n the value is going to change accordingly.

The program returns cap which is the capacitance per unit length which is measured in Pico farad per meter.

(Refer Slide Time: 14:20)



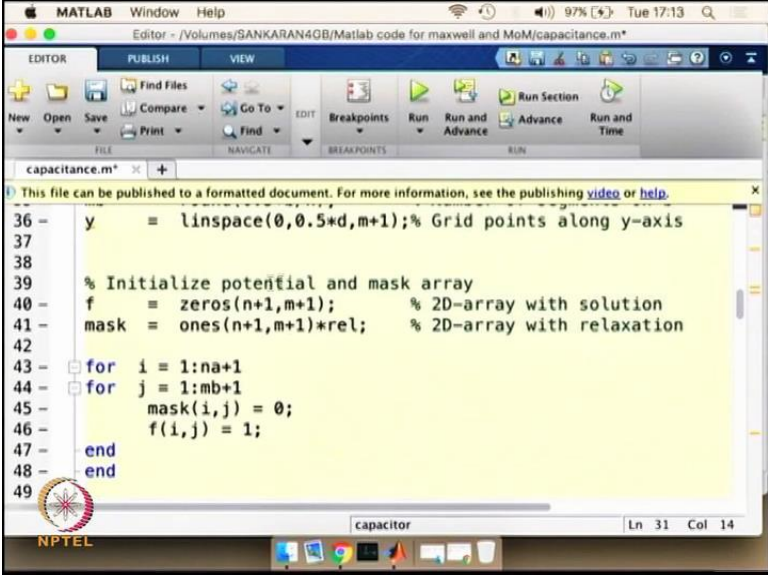
```
26 % Returns: cap = capacitance per unit length [pF/m]
27
28
29 % Make grids
30 - rel = 2-c/n; % Relaxation parameter
31 - h = 0.5*c/n; % Grid size
32 - na = round(0.5*a/h); % Number of segments on a
33 - x = linspace(0,0.5*c,n+1); % Grid points along x-axis
34 - m = round(0.5*d/h); % Number of segments on d
35 - mb = round(0.5*b/h); % Number of segments on b
36 - y = linspace(0,0.5*d,m+1); % Grid points along y-axis
37
38
39 % Initialize potential and mask array
```

So the input parameters are given and then based on that we are computing various parameters.

So h is going to be the grid size which is going to depend on c and n . The outer conductor with divided by the number of points the grid points that we are going to give in the x direction. Similarly various other parameters are described as follows.

And there are some internal number of segments that we are calculating based on the h value and d value. Again once we compute h value from the value of c and n all other parameters are calculated as given in the code.

(Refer Slide Time: 15:07)

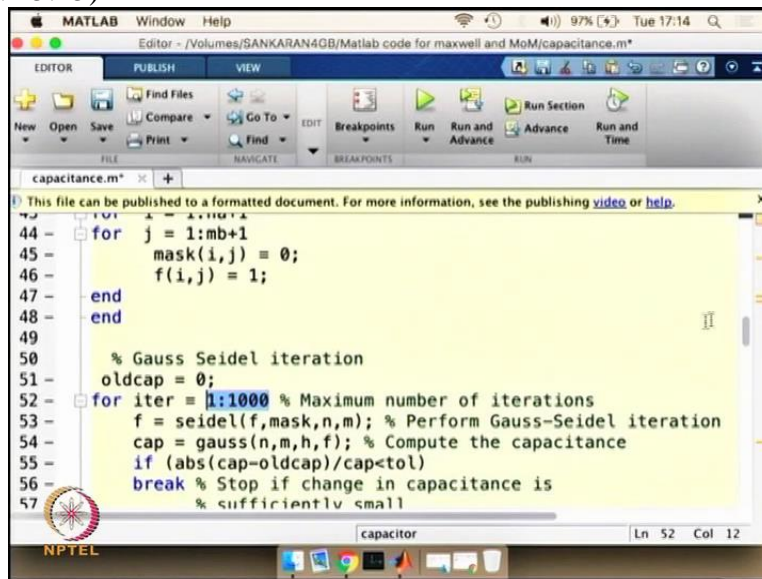
A screenshot of the MATLAB editor window. The title bar reads 'MATLAB Window Help' and the editor title is 'Editor - /Volumes/SANKARAN4GB/Matlab code for maxwell and MoM/capacitance.m*'. The code in the editor is as follows:

```
36 - y = linspace(0,0.5*d,m+1);% Grid points along y-axis
37
38
39 % Initialize potential and mask array
40 - f = zeros(n+1,m+1); % 2D-array with solution
41 - mask = ones(n+1,m+1)*rel; % 2D-array with relaxation
42
43 - for i = 1:na+1
44 - for j = 1:mb+1
45 -     mask(i,j) = 0;
46 -     f(i,j) = 1;
47 - end
48 - end
49
```

The status bar at the bottom shows 'capacitor' and 'Ln 31 Col 14'. An NPTEL logo is visible in the bottom left corner of the editor window.

We are initializing the value that we are going to use internally in the code.

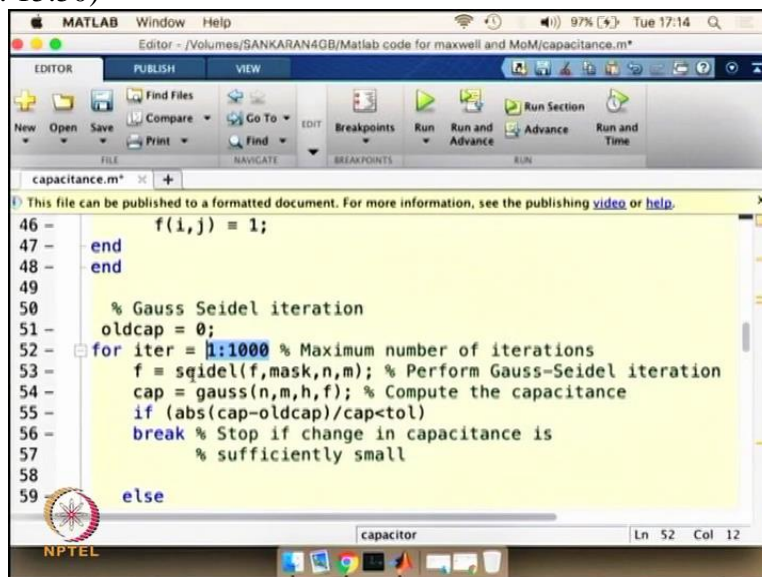
(Refer Slide Time: 15:15)



```
capacitance.m* x +
This file can be published to a formatted document. For more information, see the publishing video or help.
44- for j = 1:mb+1
45-     mask(i,j) = 0;
46-     f(i,j) = 1;
47- end
48- end
49
50 % Gauss Seidel iteration
51- oldcap = 0;
52- for iter = 1:1000 % Maximum number of iterations
53-     f = seidel(f,mask,n,m); % Perform Gauss-Seidel iteration
54-     cap = gauss(n,m,h,f); % Compute the capacitance
55-     if (abs(cap-oldcap)/cap<tol)
56-         break % Stop if change in capacitance is
57-         % sufficiently small
58
59
```

And we are running the program for certain number of iterations. One of the things that we have to give here is also the number of iterations and for a problem of this type we set the iteration to be 1000. I mean of course we can change the iterations to 10 or 100 but it is going to affect the convergence of the result. So we keep 1000 as a namesake value here obviously you can change that and find out what is good for your particular case.

(Refer Slide Time: 15:50)

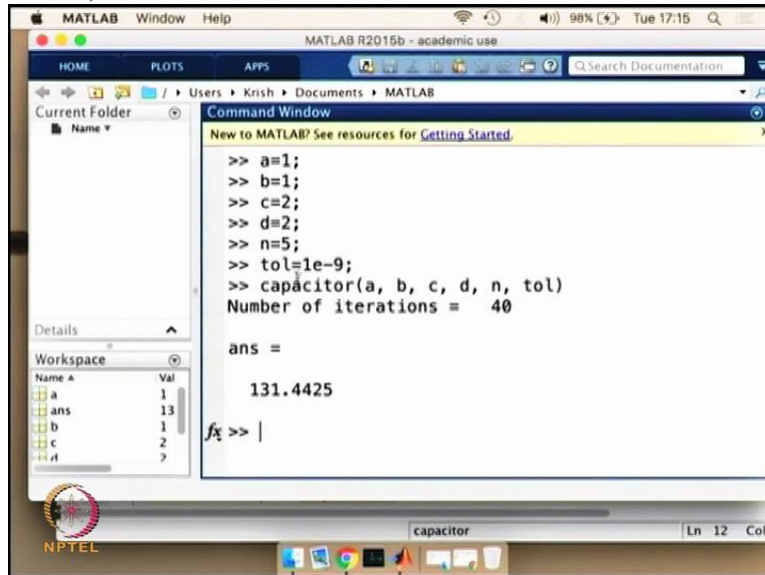


```
capacitance.m* x +
This file can be published to a formatted document. For more information, see the publishing video or help.
46-     f(i,j) = 1;
47- end
48- end
49
50 % Gauss Seidel iteration
51- oldcap = 0;
52- for iter = 1:1000 % Maximum number of iterations
53-     f = seidel(f,mask,n,m); % Perform Gauss-Seidel iteration
54-     cap = gauss(n,m,h,f); % Compute the capacitance
55-     if (abs(cap-oldcap)/cap<tol)
56-         break % Stop if change in capacitance is
57-         % sufficiently small
58
59 else
```

We are going to use some internal functions from Matlab which is a Gauss Seidel function. And we are using that to compute the capacitance as given here in the program. And there are other ways of doing the iteration as well, so you can also go for the direct Seidel iteration instead of

Gauss Seidel iteration and there are different ways of doing that, let us not go into it. For now we are going to use them as a inbuilt function. And we are going to calculate the value of the capacitance based on that.

(Refer Slide Time: 16:26)



The image shows a MATLAB R2015b Command Window with the following code and output:

```
>> a=1;
>> b=1;
>> c=2;
>> d=2;
>> n=5;
>> tol=1e-9;
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 40

ans =

    131.4425
```

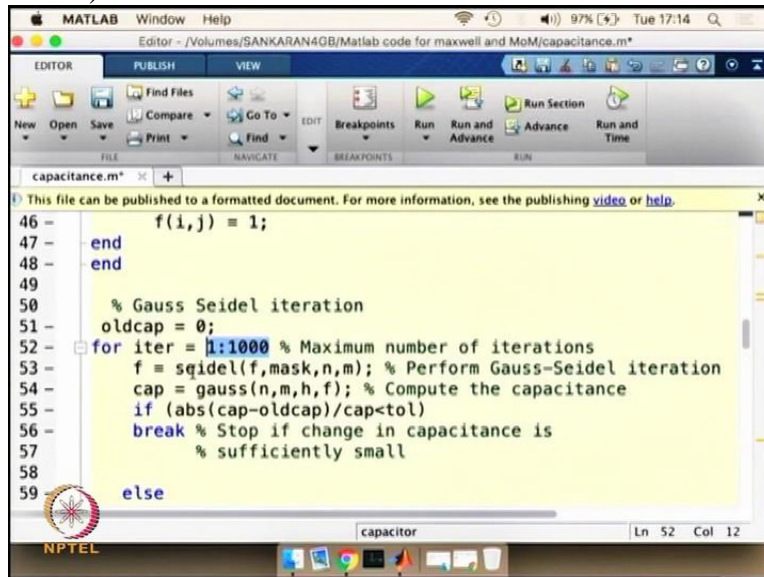
The workspace window on the left shows the following variables:

Name	Value
a	1
ans	131.4425
b	1
c	2
d	2

So let us look at the result of the simulation for various parameters that we are going to set. So we are going to set the value of a which is going to be 1, we are going to set the value of b which is also 1. So the inner coaxial is going to be a square of 1 by 1. We are going to set the outer conductor dimension as c equal to 2, d is also equal to 2. And we are going to also set the value of n and tol. So n is the number of grid points that we are going to choose, so let us say for the sake of initial problem we will choose 5 points and then the tolerance as we said we want 1e power minus 9.

And now we can basically run this code to get the value of the capacitance. So what we are getting is the number of iteration is now 40.

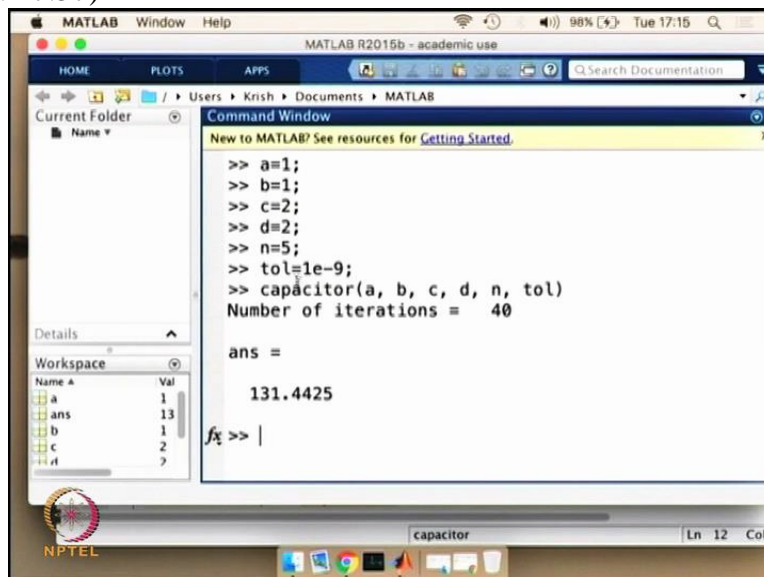
(Refer Slide Time: 17:50)



```
46 -         f(i,j) = 1;
47 -     end
48 - end
49
50     % Gauss Seidel iteration
51     oldcap = 0;
52     for iter = 1:1000 % Maximum number of iterations
53         f = sgidel(f,mask,n,m); % Perform Gauss-Seidel iteration
54         cap = gauss(n,m,h,f); % Compute the capacitance
55         if (abs(cap-oldcap)/cap<tol)
56             break % Stop if change in capacitance is
57                 % sufficiently small
58         else
59
```

Obviously if you see here the maximum iteration that the program runs is 1000, but if the value is converging already then the iteration stops at certain minimum number than 1000.

(Refer Slide Time: 17:57)



```
>> a=1;
>> b=1;
>> c=2;
>> d=2;
>> n=5;
>> tol=1e-9;
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 40

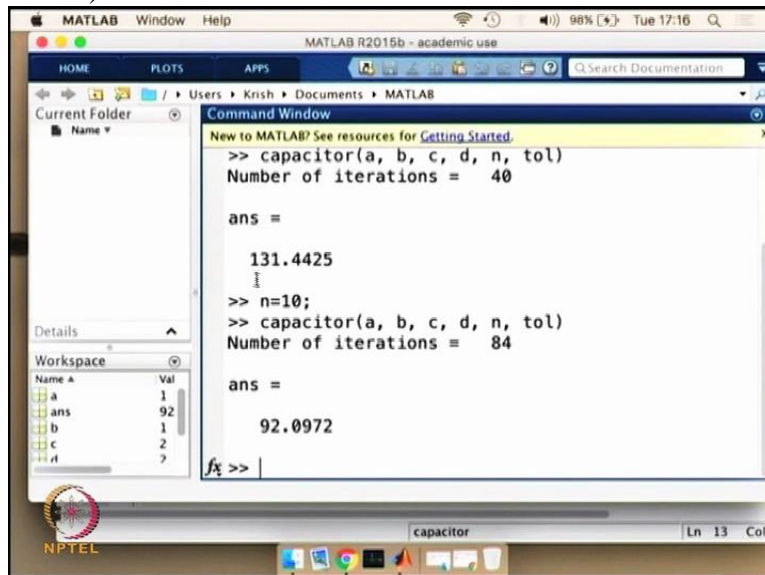
ans =

    131.4425

fx >> |
```

That is what we are seeing here. After 40 iterations the solution is already converging to 131. And of course what we are not very sure about is the number of points we are choosing is correct or not. So what is important is if the number of point is not enough the results will not be as accurate as we want. So we are going to increase the number of points which is going to be n in our case to double the number.

(Refer Slide Time: 18:21)



The screenshot shows the MATLAB R2015b Command Window. The current folder is 'MATLAB'. The Command Window displays the following code and output:

```
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 40

ans =

    131.4425

>> n=10;
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 84

ans =

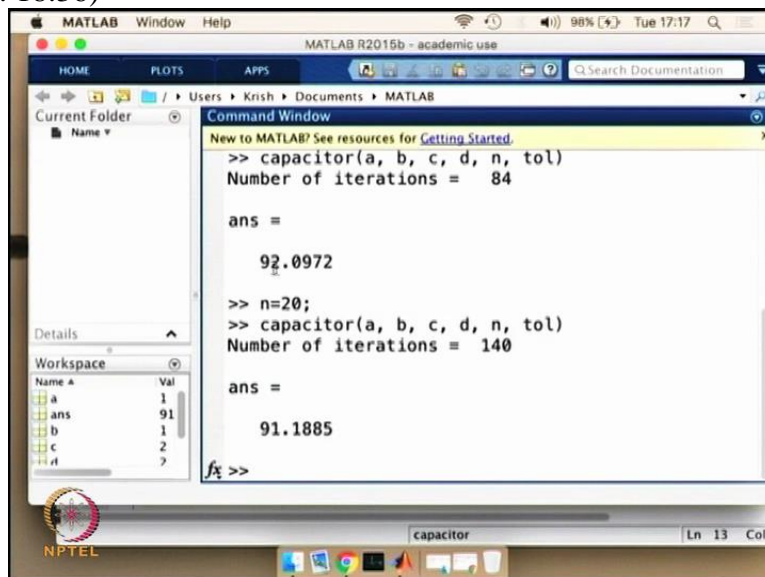
    92.0972
```

The workspace table shows the following variables:

Name	Val
a	1
ans	92
b	1
c	2
d	?

So let us say we are choosing double 10 points. And now let us run the code again and see the result is going to be much lower than what it was before.

(Refer Slide Time: 18:36)



The screenshot shows the MATLAB R2015b Command Window. The current folder is 'MATLAB'. The Command Window displays the following code and output:

```
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 84

ans =

    92.0972

>> n=20;
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 140

ans =

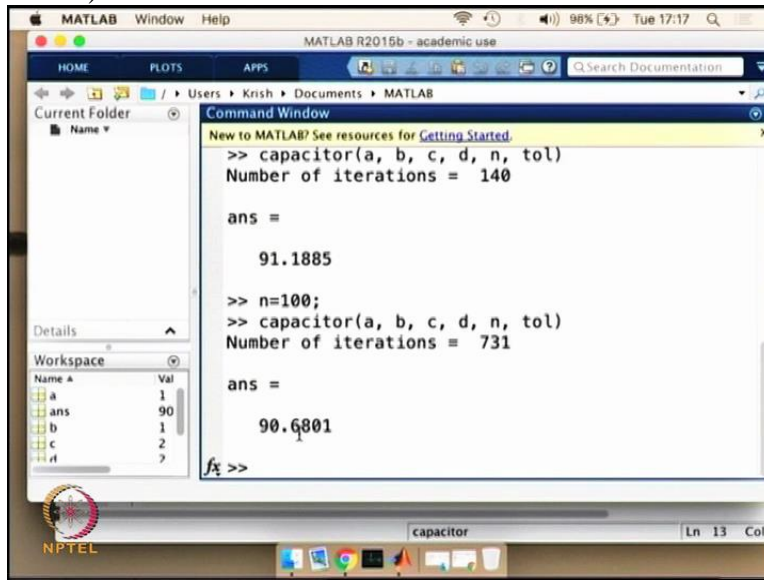
    91.1885
```

The workspace table shows the following variables:

Name	Val
a	1
ans	91
b	1
c	2
d	?

So now let us increase double it again, let us go to 20, we see the result is converging but in a slow manner.

(Refer Slide Time: 18:50)



```

>> capacitor(a, b, c, d, n, tol)
Number of iterations = 140

ans =

    91.1885

>> n=100;
>> capacitor(a, b, c, d, n, tol)
Number of iterations = 731

ans =

    90.6801

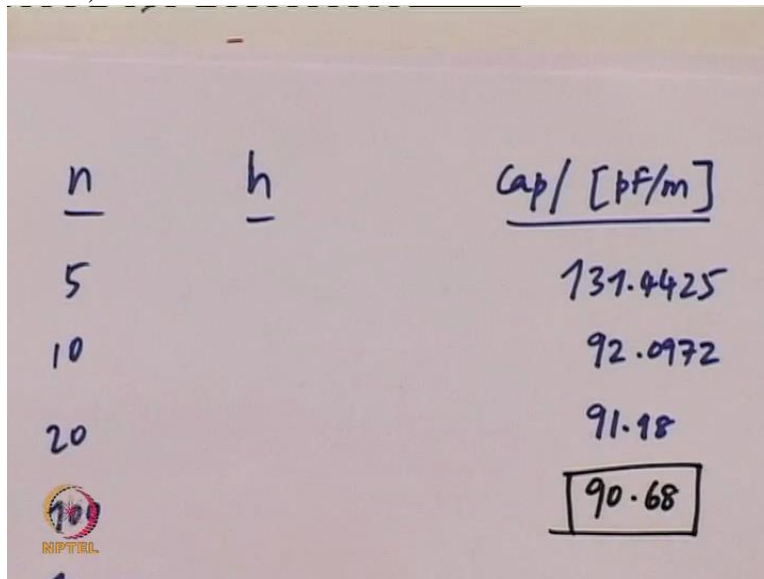
```

The screenshot shows the MATLAB Command Window with the following workspace variables:

Name	Val
a	1
ans	90
b	1
c	2
d	?

Let us say we go 5 times of that, we are coming to 90.

(Refer Slide Time: 19:05)



<u>n</u>	<u>h</u>	<u>Cap/ [pF/m]</u>
5		131.4425
10		92.0972
20		91.18
		90.68

And what we can also write is the number of points n and for that we can also compute the h value which is going to be the grid element size and the capacitance value which is measured in Pico farad per meter. You write it down in this form what we can see is various values for n equal to 5, n equal to 10, n equal to 20, n equal to 100.

For n equal to 5 as we can see in the code

We are getting a value which is 131.4425,

For n equal to 10 We got 92.0972

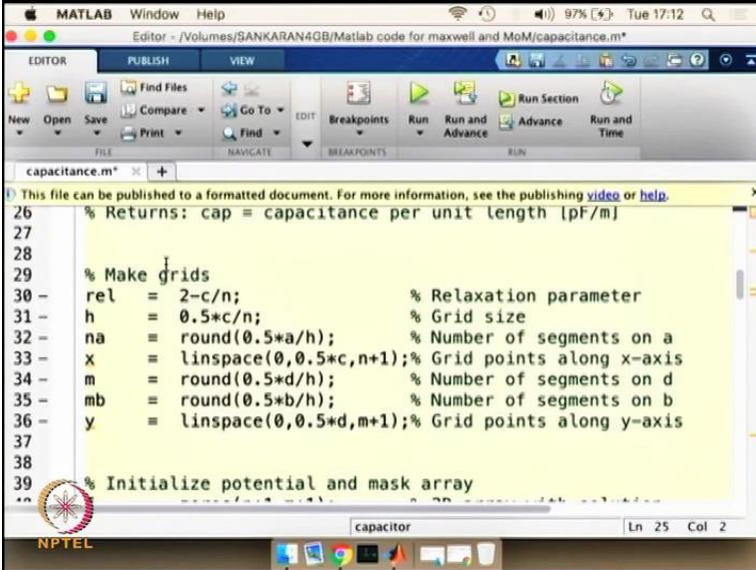
For n equal to 20 We got 91.18

So what we are going to see now is certain convergence patterns for various values of n and we are going to see what are those convergence values.

And for n equal to 100

We got a value which is 90.68. And we say that it is almost converging this limit.

(Refer Slide Time: 20:45)

A screenshot of the MATLAB editor window. The title bar shows 'MATLAB Window Help' and the file path is '/Volumes/SANKARAN4GB/Matlab code for maxwell and MoM/capacitance.m'. The editor displays the following code:

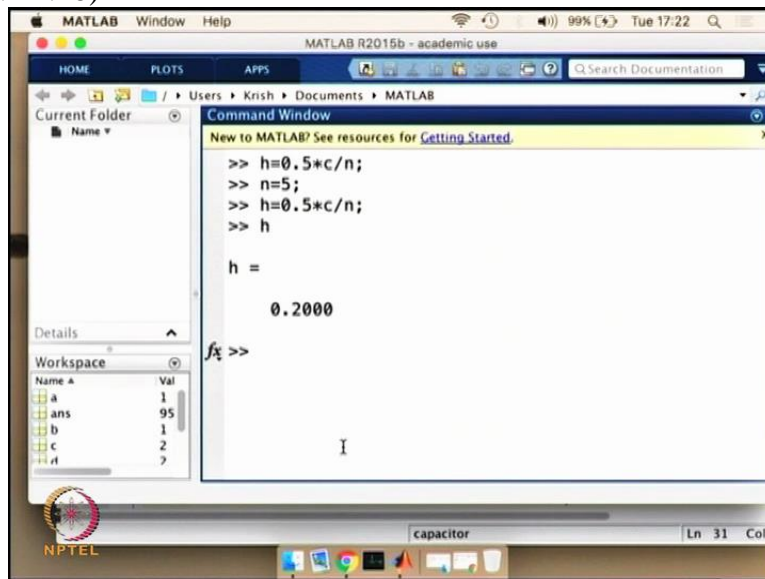
```
26 % Returns: cap = capacitance per unit length [pF/m]
27
28
29 % Make grids
30 - rel = 2-c/n; % Relaxation parameter
31 - h = 0.5*c/n; % Grid size
32 - na = round(0.5*a/h); % Number of segments on a
33 - x = linspace(0,0.5*c,n+1); % Grid points along x-axis
34 - m = round(0.5*d/h); % Number of segments on d
35 - mb = round(0.5*b/h); % Number of segments on b
36 - y = linspace(0,0.5*d,m+1); % Grid points along y-axis
37
38
39 % Initialize potential and mask array
```

The status bar at the bottom indicates 'capacitor' and 'Ln 25 Col 2'. An NPTEL logo is visible in the bottom left corner of the editor window.

And obviously this particular code has certain parameters that are dependent on n for example the value that we are going to get depends on n in terms of the Gauss Seidel method as well. So we see that once we are in 100 points limit it is already a good value for us to see that the result is converging.

And one more thing is the value of the step size or the grid size itself, h value and h value is going to be dependent on n itself. So if you compute the value of h for various values of n and see so we can take that.

(Refer Slide Time: 21:28)



The screenshot shows the MATLAB R2015b Command Window. The Command Window contains the following code and output:

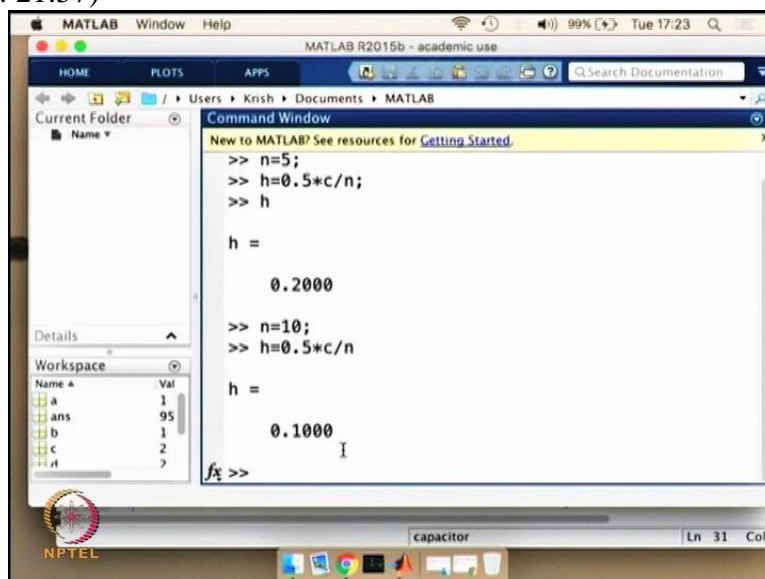
```
>> h=0.5*c/n;  
>> n=5;  
>> h=0.5*c/n;  
>> h  
  
h =  
  
    0.2000  
  
fx >>
```

The Workspace window on the left shows the following variables:

Name	Val
a	1
ans	95
b	1
c	2
r1	?

We can clear the screen here. So we can write h is equal to 0.5 and then we can see what is the h value for various values of n. So if n equal to 5 h value is 0.2.

(Refer Slide Time: 21:57)



The screenshot shows the MATLAB R2015b Command Window. The Command Window contains the following code and output:

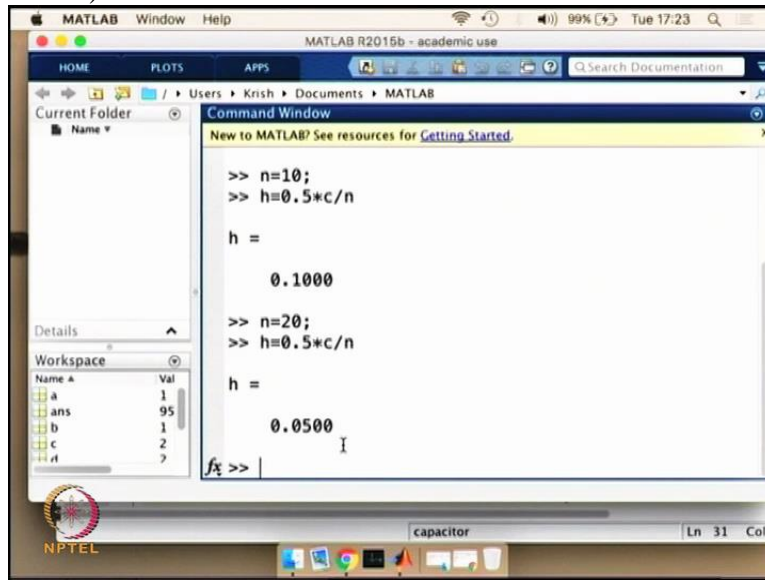
```
>> n=5;  
>> h=0.5*c/n;  
>> h  
  
h =  
  
    0.2000  
  
>> n=10;  
>> h=0.5*c/n  
  
h =  
  
    0.1000  
  
fx >>
```

The Workspace window on the left shows the following variables:

Name	Val
a	1
ans	95
b	1
c	2
r1	?

If n equal to 10, h value is 0.1

(Refer Slide Time: 22:08)



The image shows a MATLAB R2015b Command Window. The window title is "MATLAB R2015b - academic use". The Command Window contains the following code and output:

```
>> n=10;
>> h=0.5*c/n

h =

    0.1000

>> n=20;
>> h=0.5*c/n

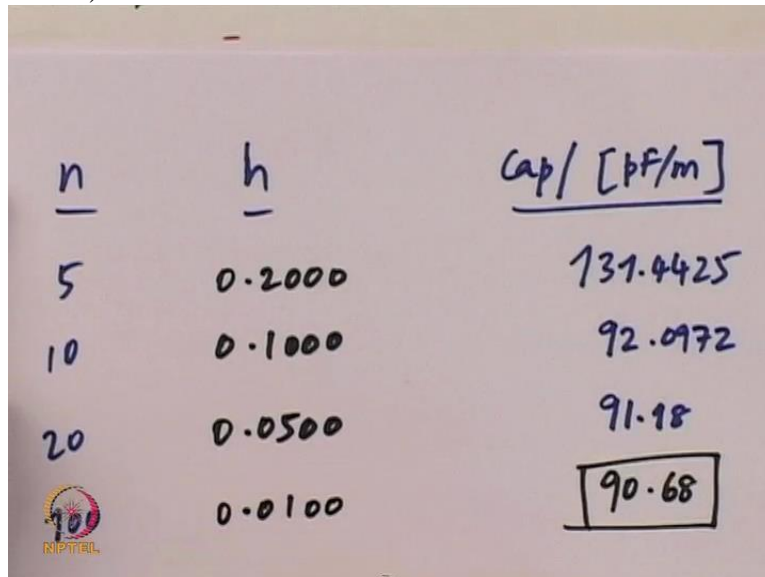
h =

    0.0500
```

The workspace on the left shows variables: a (1), ans (95), b (1), c (2), and r (?). The current folder is "/Users/Krish/Documents/MATLAB". The status bar at the bottom shows "capacitor Ln 31 Col".

And n equal to 20, h value is 0.05

(Refer Slide Time: 22:17)



The image shows a handwritten table with three columns: n , h , and $Cap / [pF/m]$. The values are as follows:

n	h	$Cap / [pF/m]$
5	0.2000	137.4425
10	0.1000	92.0972
20	0.0500	91.98
	0.0100	90.68

The value 90.68 is enclosed in a box. The NPTEL logo is visible in the bottom left corner.

So let me write it down here. So when n equal to 5 what we got is the h value is going to be 0.2000

And when n is equal to 10 we have 0.1000

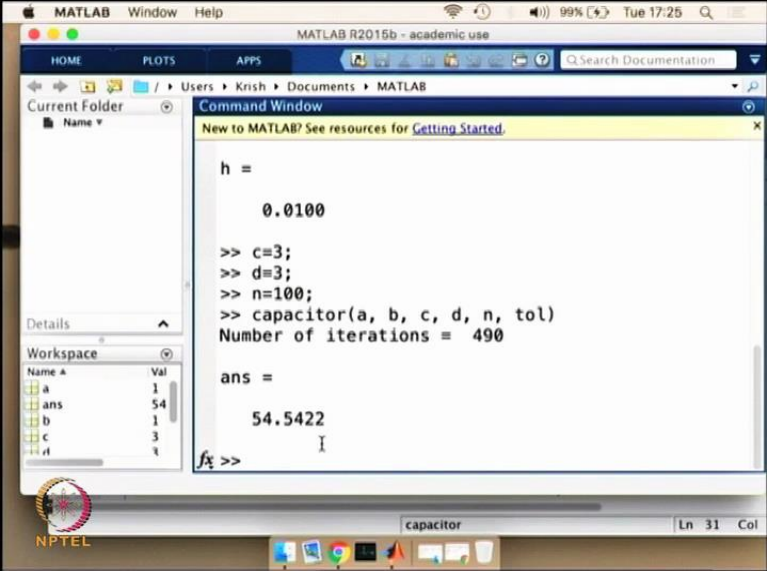
And when n is equal to 20 we get 0.0500

And when n is equal to 100 we can compute the value of h which is going to be 0.0100

So this particular problem is already converging around this point. Obviously when you go higher in this number of n the value is going to go down. But be careful that sometimes for some

values of n the value might increase because we are going to depend on n for the case of the Gauss Seibel method as well. So once you see that the value is converging towards the limit, then it is already good enough for you to stop there.

(Refer Slide Time: 24:15)



The screenshot shows the MATLAB Command Window with the following code and output:

```
h =  
    0.0100  
  
>> c=3;  
>> d=3;  
>> n=100;  
>> capacitor(a, b, c, d, n, tol)  
Number of iterations = 490  
  
ans =  
  
    54.5422
```

The workspace on the left shows variables: a (1), ans (54.5422), b (1), c (3), and d (3). The file name at the bottom is 'capacitor'.

So we say that with the problem of this sort with n is equal to 100 where the grid element size is roughly 0.01 the value is 90.68 Pico farad per meter. And this is going to be the capacitance value. And obviously we can run the same problem for various values of a and b , c and d . For example if I choose a and b as a same whereas I change c to be 3 and d to be 3 as well and the rest of the values n is equal to 100 and I am running the code again you see that the capacitance value is changing accordingly and which is a different problem altogether because the geometry itself is changed.

So what I would suggest is you take this code and run for various values of a , b , c and d and also for various values of tolerance and relaxation parameters and get a good sense of how this code is running and for people who are more interested in programming such methods, the sources of code is also given at the bottom. So I would definitely recommend you to look into the source for further instructions.

But for now it is good enough to practice this code and get a good sense of the physical things what we are trying to model and also the value of the method itself. Because we are going to use central differencing method. So I encourage you to practice this code and get a good sense of how the problem is modeled. Thank you