

Computational Electromagnetics and Applications
Professor Krish Sankaran
Indian Institute of Technology Bombay
Lecture 05/Exercise 02
Finite Difference Methods –1

So we will now look into yet another equation which is building on the first equation, Laplace equation. So now we are going to extend Laplace equation to Poisson equation. As we know in the Poisson equation we will have the right hand side term. So let us start looking into the equation itself.

(Refer Slide Time: 00:37)

$\nabla^2 \phi = \frac{\rho}{\epsilon}$

$\left[\frac{V}{m^2} \right]$

$= \frac{\rho}{\epsilon} \Rightarrow$ if constant

\Downarrow variable

$f = \frac{\rho}{\epsilon}$
 $f(x, y)$

entire domain with certain $\frac{\rho}{\epsilon}$

So initially we had the term Del square Phi equal to 0 in the case of Laplace equation right now we are going to look into Poisson equation. So we will have Del square Phi is equal to Rho by Epsilon. And again when you look at the units so we are talking about units volt per meter square. So this is going to be the unit of this right hand side term. And we are going to put certain value. So we assume that Rho by Epsilon is equal to a constant. At certain points we can basically assign that constants in the domain.

It need not be a constant it can vary across domain. so what we are going to do is, so this is a special case if this is a constant then you can define the entire domain with certain Rho by Epsilon. Whereas in our case we are going to keep this one as variable. So if we choose this path then you can assign the entire domain with certain value for Rho by Epsilon. But if we choose the path of the variable then the function , so this is going to be a function f is equal to Rho by Epsilon, and f is going to be a function of x,y.

(Refer Slide Time: 02:32)

$\nabla^2 \varphi = \frac{\rho}{\epsilon}$ $\left[\frac{V}{m^2} \right]$

$= \frac{\rho}{\epsilon} \Rightarrow$ if Constant

\Downarrow Variable

$f = \frac{\rho}{\epsilon} = f(x, y)$

$\varphi = 0$

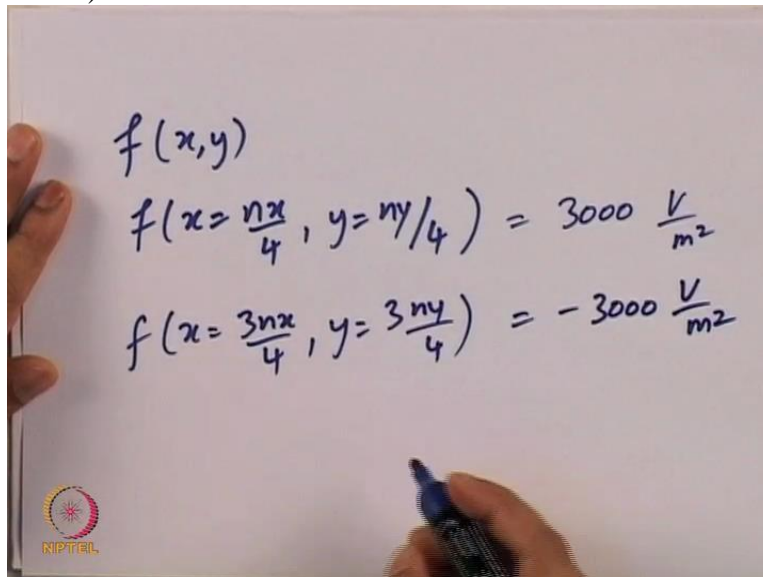
entire domain with certain $\frac{\rho}{\epsilon}$

In other words along the domain it is going to have certain values so what we are going to do is we are going to say at certain point we are putting the potential in terms of Rho by Epsilon. So in certain point we are assigning the source function to be certain value and the rest of the points we keep them as 0.

So in the example what we are going to have, we are going to run the particular simulation and let us say define the domain boundaries accordingly. So we have (0,0), we have (10,0), we have (10,10) and we have (0,10). So this is going to be the domain boundary and we are setting all the boundary conditions where Phi equal to 0, of course we can set other boundary condition as well for in this particular problem we are going to set all the boundary conditions going to be equal to 0.

And the right hand side term, which we are saying to be a variable and we are assigning that variable to have certain functional values depending on the x and y coordinates. So let us take this further and we say we will assign certain values for the source itself.

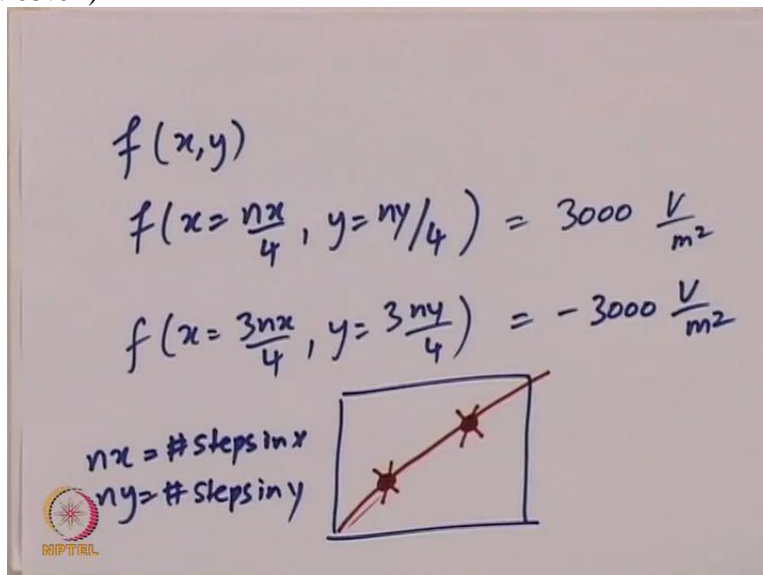
(Refer Slide Time: 03:58)


$$f(x,y)$$
$$f\left(x = \frac{nx}{4}, y = \frac{ny}{4}\right) = 3000 \frac{V}{m^2}$$
$$f\left(x = \frac{3nx}{4}, y = \frac{3ny}{4}\right) = -3000 \frac{V}{m^2}$$

So the source is going to be defined as f of (x,y) and let us say at the point where f of $(x$ equal to nx by 4 and y equal to ny by 4). So we are going to define the value of the source at two points, the first point is going to be f of $(x$ equal to nx by 4 and y equal to ny by 4) where we are going to set the value as 3000 volts per meter square.

And Let us say we are choosing another point, where f of $(x$ equal to $3nx$ by 4 , y equal to $3ny$ by 4) equal to minus 3000 volt per meter square.

(Refer Slide Time: 05:02)

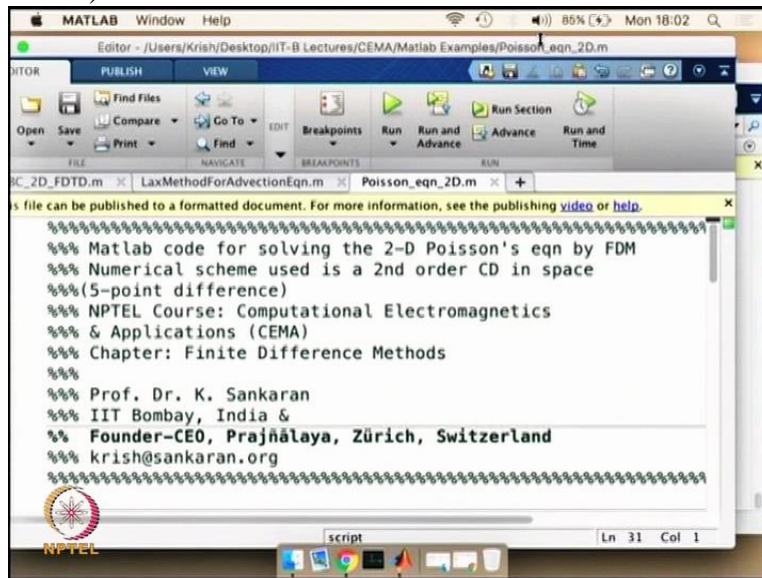

$$f(x,y)$$
$$f\left(x = \frac{nx}{4}, y = \frac{ny}{4}\right) = 3000 \frac{V}{m^2}$$
$$f\left(x = \frac{3nx}{4}, y = \frac{3ny}{4}\right) = -3000 \frac{V}{m^2}$$

$nx = \# \text{ steps in } x$
 $ny = \# \text{ steps in } y$

So if you look at your domain nx and ny are going to be number of steps. So these are number of steps in x . similarly this is going to be number of steps in y .

So now what we are saying that for certain two points. And we have chosen two points in such a manner that it is going to be in a linear line and they are going to be at two locations and those locations are given by these two points. And we are assigning certain value for the source.

(Refer Slide Time: 05:02)

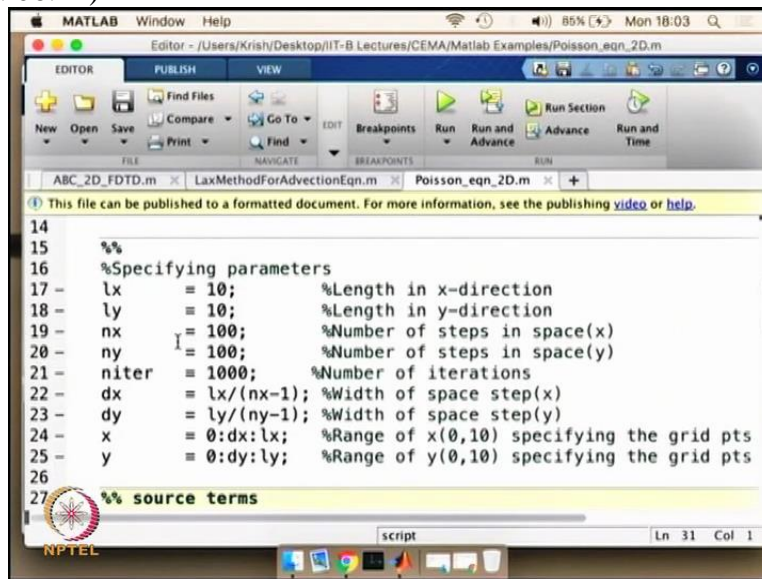


The screenshot shows the MATLAB editor window with the following content:

```
file can be published to a formatted document. For more information, see the publishing video or help.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matlab code for solving the 2-D Poisson's eqn by FDM
%% Numerical scheme used is a 2nd order CD in space
%% (5-point difference)
%% NPTEL Course: Computational Electromagnetics
%% & Applications (CEMA)
%% Chapter: Finite Difference Methods
%%
%% Prof. Dr. K. Sankaran
%% IIT Bombay, India &
%% Founder-CEO, Prajñālaya, Zürich, Switzerland
%% krish@sankaran.org
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

So let us go into the code itself and we will see certain parameters that we are declaring in the code. So this is going to be the Matlab program for solving a simple 2D Poisson equation using Finite Differencing method and we are going to use the central order Central Differencing method like the way we discussed in the Laplace equation problem. And we are going to use the 5 point differencing scheme.

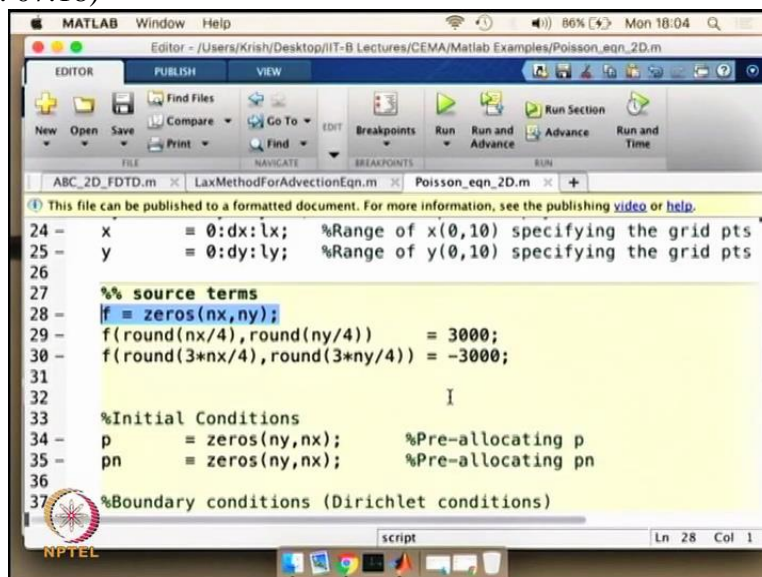
(Refer Slide Time: 06:24)



```
14
15 %%
16 %Specifying parameters
17 - lx      = 10;      %Length in x-direction
18 - ly      = 10;      %Length in y-direction
19 - nx      = 100;     %Number of steps in space(x)
20 - ny      = 100;     %Number of steps in space(y)
21 - niter   = 1000;    %Number of iterations
22 - dx      = lx/(nx-1); %Width of space step(x)
23 - dy      = ly/(ny-1); %Width of space step(y)
24 - x       = 0:dx:lx; %Range of x(0,10) specifying the grid pts
25 - y       = 0:dy:ly; %Range of y(0,10) specifying the grid pts
26
27 %% source terms
```

So let us look at the initial specifications of parameters. So the length is going to be 10 in the x direction 10 in the y direction. And the number of steps in x and y direction are going to be given by nx and ny. In this case we are choosing 100 steps in each of the direction. And we are going to iterate the problem for 1000 steps. And the special disciazation in x and y directions are going to be given by dx and dy. And each of the points in x and y directions are going to be given in the vector x and y which goes from 0 to lx which is 10 in our case and 0 to ly which is also 10 in our case and going to have a step size of dx defined by this particular equation.

(Refer Slide Time: 07:18)

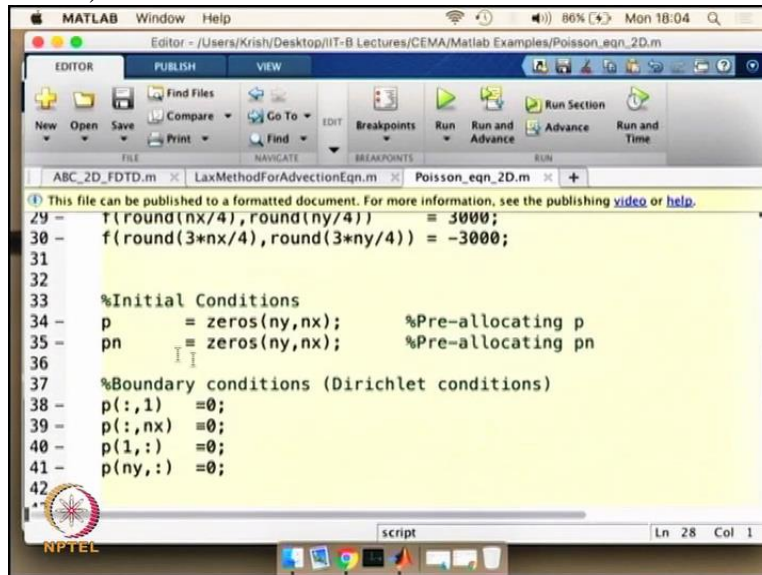


```
24 - x       = 0:dx:lx; %Range of x(0,10) specifying the grid pts
25 - y       = 0:dy:ly; %Range of y(0,10) specifying the grid pts
26
27 %% source terms
28 - f = zeros(nx,ny);
29 - f(round(nx/4),round(ny/4)) = 3000;
30 - f(round(3*nx/4),round(3*ny/4)) = -3000;
31
32
33 %Initial Conditions
34 - p       = zeros(ny,nx); %Pre-allocating p
35 - pn      = zeros(ny,nx); %Pre-allocating pn
36
37 %%Boundary conditions (Dirichlet conditions)
```

And now we are going to define the source term itself, so we are saying that the source is going to be the function of x and y along the entire domain. Since we assume that f is going to be a

variable. we are defining it in each of the points. if we say that f is going to be constant in the entire domain, we do not need to do that we can directly say f is equal to that particular value we want but in this case we have taken a more complicated problem. Where we have set f is going to be a variable in the entire domain. And as we said we are going to keep the value of f source function as 3000 volt per meter square in two of the points. So in one point we are going to keep it as 3000, the other point we are going to keep it as minus 3000 volt per meter square.

(Refer Slide Time: 08:13)



```
29 - T(round(nx/4), round(ny/4)) = 3000;  
30 - f(round(3*nx/4), round(3*ny/4)) = -3000;  
31  
32  
33 %Initial Conditions  
34 - p = zeros(ny,nx); %Pre-allocating p  
35 - pn = zeros(ny,nx); %Pre-allocating pn  
36  
37 %Boundary conditions (Dirichlet conditions)  
38 - p(:,1) = 0;  
39 - p(:,nx) = 0;  
40 - p(1,:) = 0;  
41 - p(ny,:) = 0;  
42
```

And we are going to initialize a condition for the potentials and the boundary conditions like in the case before; we are going to put them all equal to 0. Of course we can vary the boundary condition as we want. But in this case we have set it to 0. Since we have defined the boundaries on the first and the last points we are going to run the iteration only for i equal to 2 to n_x minus 1; Similarly j equal to 2 to n_y minus 1.

(Refer Slide Time: 08:43)

```

MATLAB Window Help
Editor - /Users/Krish/Desktop/IT-B Lectures/CEMA/Matlab Examples/Poisson_eqn_2D.m
EDITOR PUBLISH VIEW
New Open Save Compare Go To EDIT Breakpoints Run Run and Advance Run and Time
ABC_2D_FDTD.m LaxMethodForAdvectionEqn.m Poisson_eqn_2D.m
This file can be published to a formatted document. For more information, see the publishing video or help.
46 - i = 2:nx-1;
47 - j = 2:ny-1;
48
49 - for it = 1:niter
50 -     pn = p;
51 -     p(i,j) = ((dy^2*(pn(i+1,j)+pn(i-1,j)))+...
52 -             (dx^2*(pn(i,j+1)+pn(i,j-1)))-...
53 -             (f(i,j)*dx^2+dy^2))/(2*(dx^2+dy^2));
54 - end
55
56 %Plotting the solution
57 - surf(x,y,p,'EdgeColor','none');
58 - shading interp
59 - title('2-D Poisson's equation';...
script Ln 53 Col 15
  
```

And the iteration goes except for this particular term the entire equation is same as Laplace equation. In the Laplace equation the source function is going to be 0, so this is not there. Whereas in this case we are going to have it on right hand side.

(Refer Slide Time: 09:07)

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f(x,y) \Rightarrow \frac{\rho}{\epsilon}$$

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

So when we are going to see the differencing analog of this equation what we get is (do square Phi by do x square) plus (do square Phi by do y square) is equal to f of (x,y) which is nothing but the Rho by Epsilon. And when you are using the central differencing scheme as we have seen this will be equal to Phi of (i plus 1,j) minus 2 Phi of (i,j) plus Phi of (i minus 1,j) divided by Del

x square plus Phi of (i,j plus 1) minus 2 Phi (i,j) plus Phi of (i,j minus 1) divided by Del y square is going to be equal to f of (x,y) which is equal to f of (i,j).

And now when you rearrange the term you are going to get on the right hand side the f term as well. and that is what you see in this particular equation here

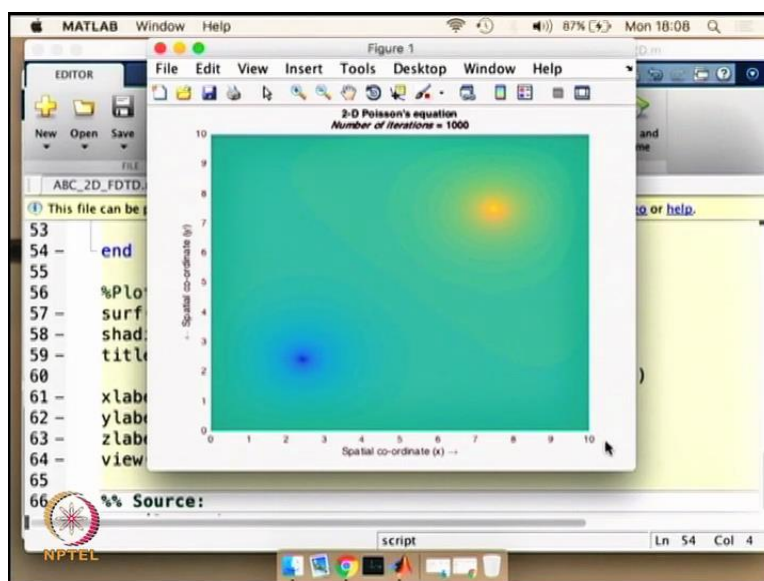
(Refer Slide Time: 10:43)

```

53         (f(i,j)*dx^2+dy^2))/(2*(dx^2+dy^2));
54     end
55
56     %Plotting the solution
57     surf(x,y,p,'EdgeColor','none');
58     shading interp
59     title({'2-D Poisson's equation';...
60         ['\itNumber of iterations = ',num2str(it)]})
61     xlabel('Spatial co-ordinate (x) \rightarrow')
62     ylabel('\leftarrow Spatial co-ordinate (y)')
63     zlabel('Solution profile (P) \rightarrow')
64     view(0,90);
65
66 %% Source:
  
```

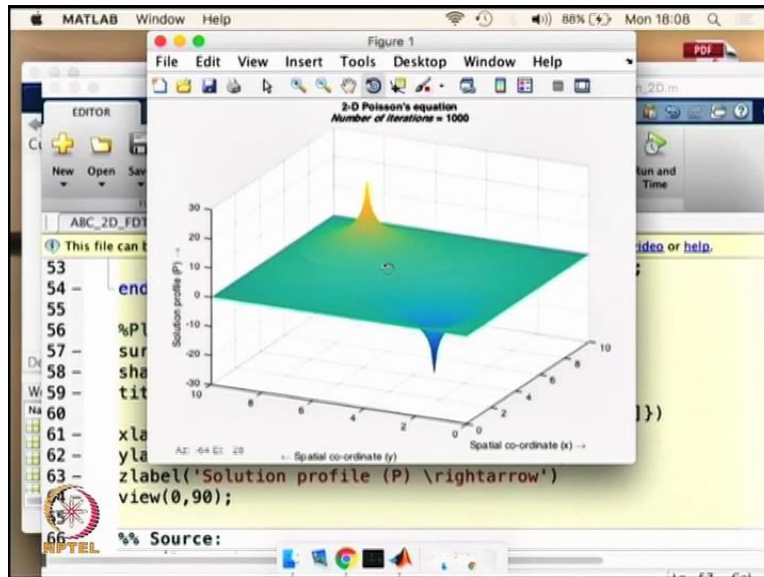
So now we are going to plot the same function p which we are computing and we going to look at it from the top angle (0, 90). And let us simulate it.

(Refer Slide Time: 11:01)



What we see here is the potentials on the left and the right, similarly at the bottom and the top, they are all set to 0. What we see is two sources which are sitting in two points.

(Refer Slide Time: 11:20)



So what you see is the solution what we have simulated may be one test condition is if we can reduce the resolution and we can see what is the impact of the resolution.

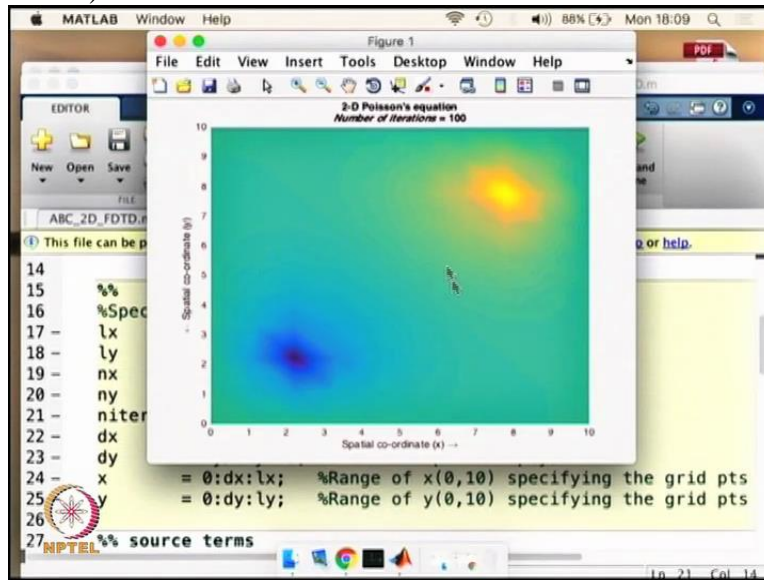
(Refer Slide Time: 11:35)

The screenshot shows the MATLAB editor window with the source code for the 3D Poisson's equation simulation. The code is as follows:

```
14  
15  
16 %% Specifying parameters  
17 - lx = 10; %Length in x-direction  
18 - ly = 10; %Length in y-direction  
19 - nx = 10; %Number of steps in space(x)  
20 - ny = 10; %Number of steps in space(y)  
21 - niter = 100; %Number of iterations  
22 - dx = lx/(nx-1); %Width of space step(x)  
23 - dy = ly/(ny-1); %Width of space step(y)  
24 - x = 0:dx:lx; %Range of x(0,10) specifying the grid pts  
25 - y = 0:dy:ly; %Range of y(0,10) specifying the grid pts  
26  
27 %% source terms
```

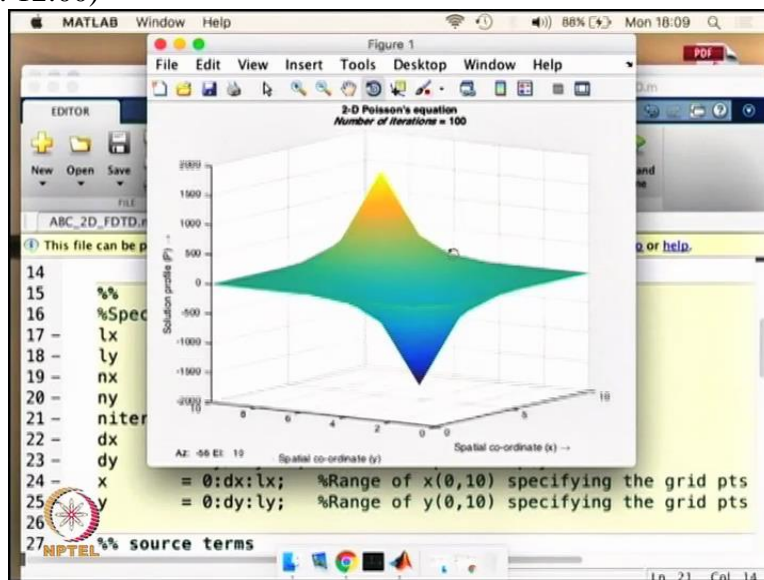
Like the way we saw before, instead of having 100 steps let us say we are having only 10 steps in n_x and n_y and we are simulating it only for 100 iterations.

(Refer Slide Time: 11:45)



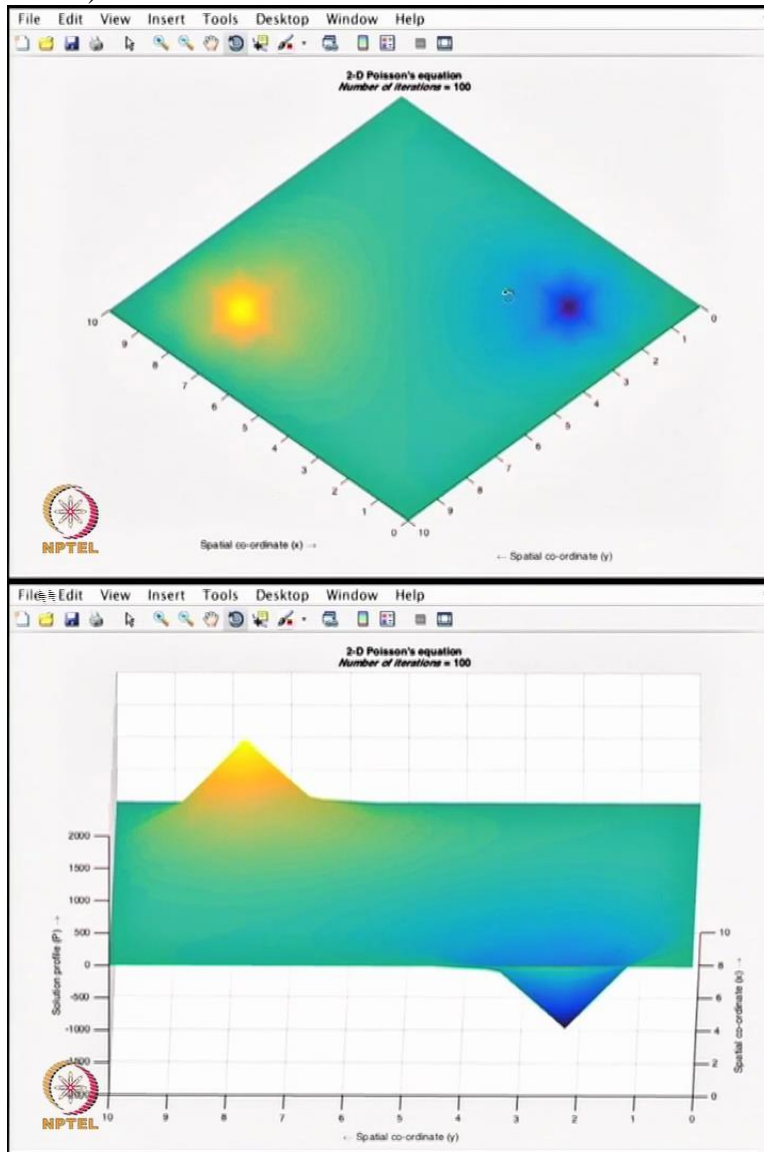
You will see that the result is very crude and you can already see the result will have certain pattern of the discretization in the way we see the potential distribution.

(Refer Slide Time: 12:00)



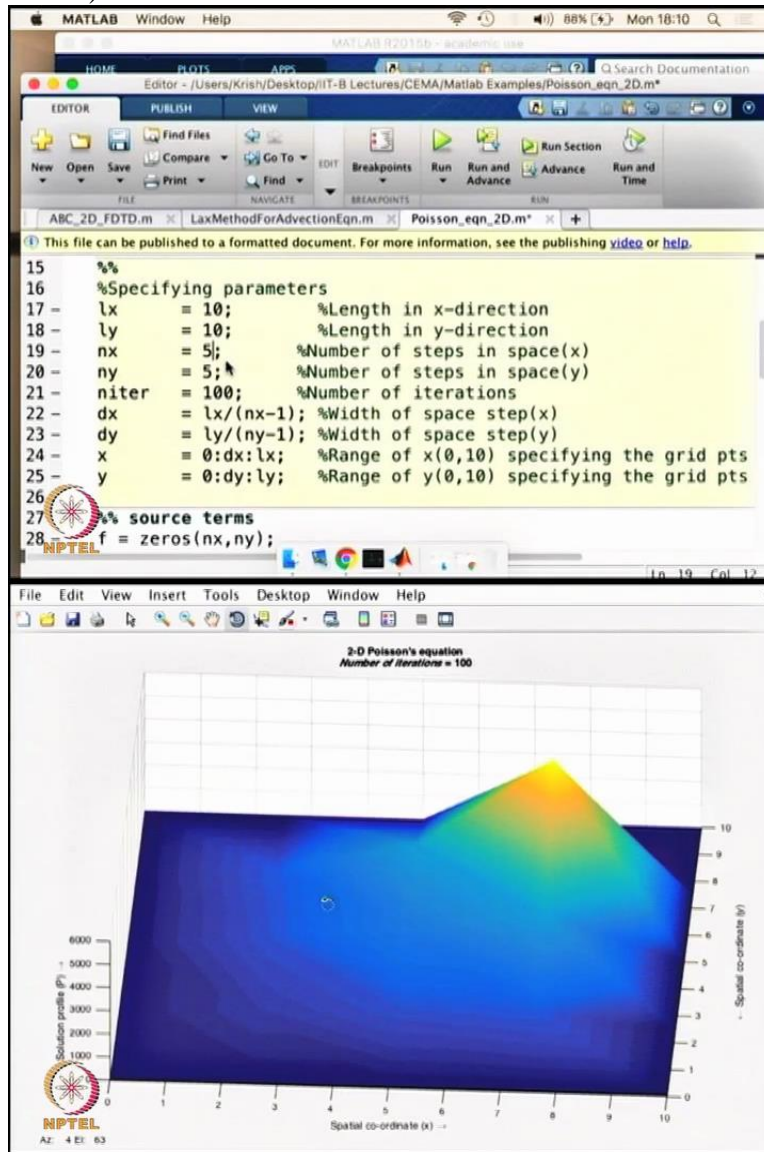
This is not very accurate as we can already appreciate that we already see the discretization impact in the solution. If I zoom it you will be able to see it what you see is pretty much the discretization impact is seen.

(Refer Slide Time: 12:09)



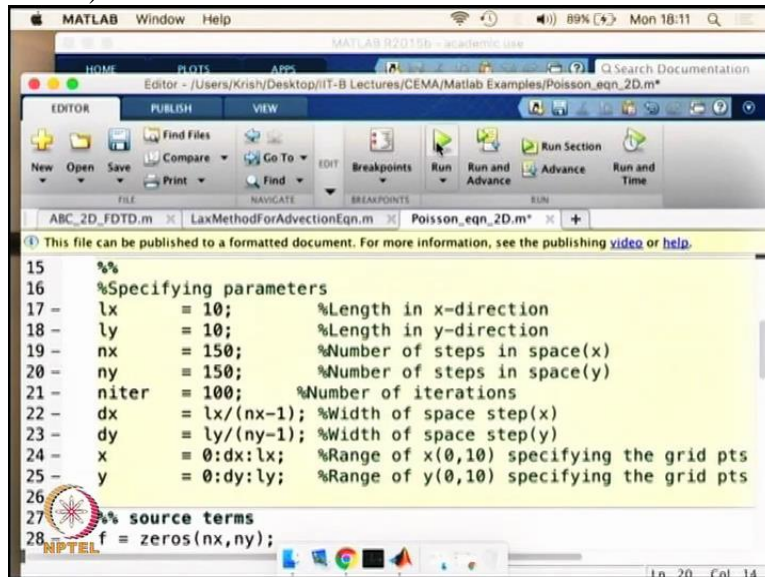
And when we reduce the number of term even further, we will see that we are not able to get good results. For example if i go even lower, let us say i am having only 5 steps in n_x and n_y .

(Refer Slide Time: 12:35)



This is not accurate as you see the other side the other source function is not able to be seen at all. So what we wanted to showcase here is even though the differencing method itself will have certain impact. This impact is even strongly seen once you have certain number of cells assigned.

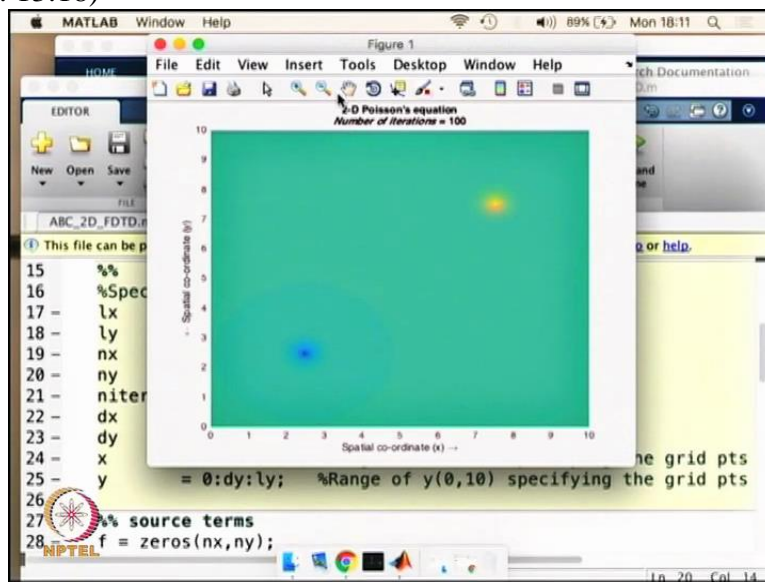
(Refer Slide Time: 13:00)



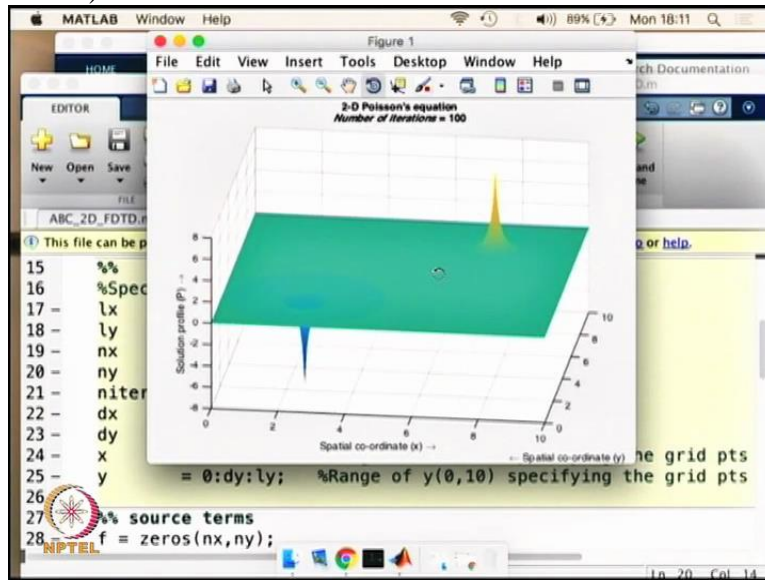
```
15 %  
16 %Specifying parameters  
17 - lx = 10; %Length in x-direction  
18 - ly = 10; %Length in y-direction  
19 - nx = 150; %Number of steps in space(x)  
20 - ny = 150; %Number of steps in space(y)  
21 - niter = 100; %Number of iterations  
22 - dx = lx/(nx-1); %Width of space step(x)  
23 - dy = ly/(ny-1); %Width of space step(y)  
24 - x = 0:dx:lx; %Range of x(0,10) specifying the grid pts  
25 - y = 0:dy:ly; %Range of y(0,10) specifying the grid pts  
26  
27 % source terms  
28 f = zeros(nx,ny);
```

For example if we increase the solution we will be able to see that we are able to get much much more finer results. So let us say we go for 150 cells in the x and y direction.

(Refer Slide Time: 13:18)

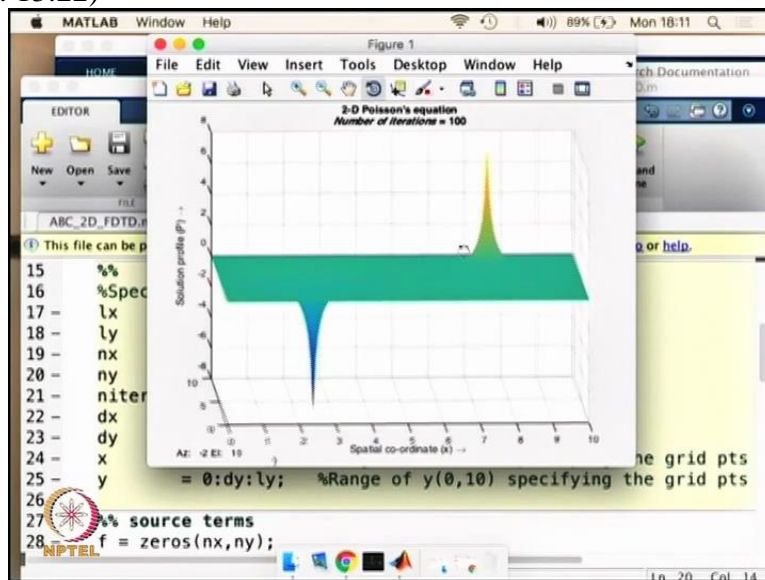


(Refer Slide Time: 13:19)



You will be able to see that you are able to very very accurately replicate the potential. And this is what we want in our simulation method that we want to have a good resolution at the same time. We have to compromise on the number of iteration.

(Refer Slide Time: 13:22)



When you are doing simple problems like this the most important thing is to make a good sense of judgment. How many iteration do i need? What is the number of cells i need in x and y direction? And also what is going to be the method i am going to use? in this case we have chosen Central Differencing scheme because it has a better accuracy than forward and backward differencing which we saw in our earlier modules.

However only using central differencing scheme is not enough, we need to set certain parameters like the special discretization in x and y direction and also the number of iteration what we need to resolve this problem

So we have showcased a classical problem of Poisson for understanding how a simple problem can teach us a lot about Finite Differencing method. I would encourage you to take this code and practice it for yourself. And play a little bit with various parameters so as to get a physical sense of what we are simulating? And how we can use this information and knowledge for solving much complex problem at later stage.