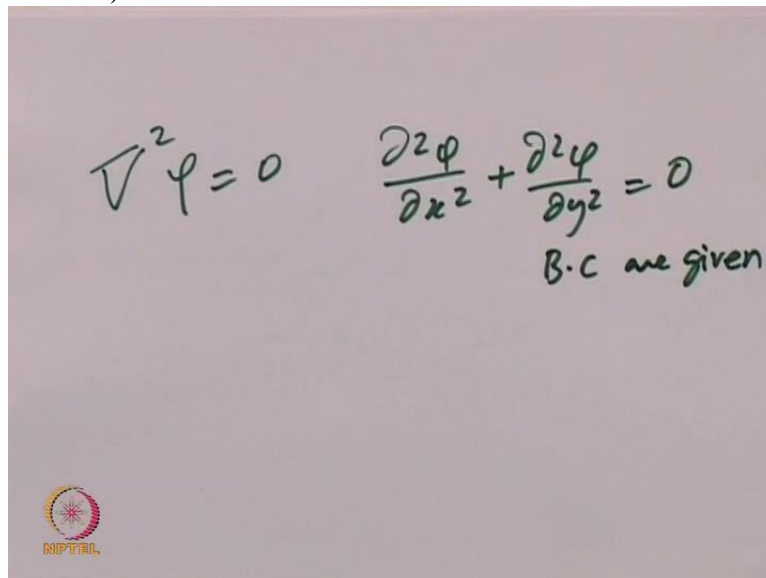


**Computational Electromagnetics and Applications**  
**Professor Krsih Sankaran**  
**Indian Institute of Technology Bombay**  
**Exercise No 11**  
**Finite Element Method -I**

Hello! So today we are going to look into a very interesting way to solve a simple simple problems like capacitor problem. So we will look into Finite element method based approach how to use Finite element to solve a very simple capacitor problem, which is a Laplace problem.

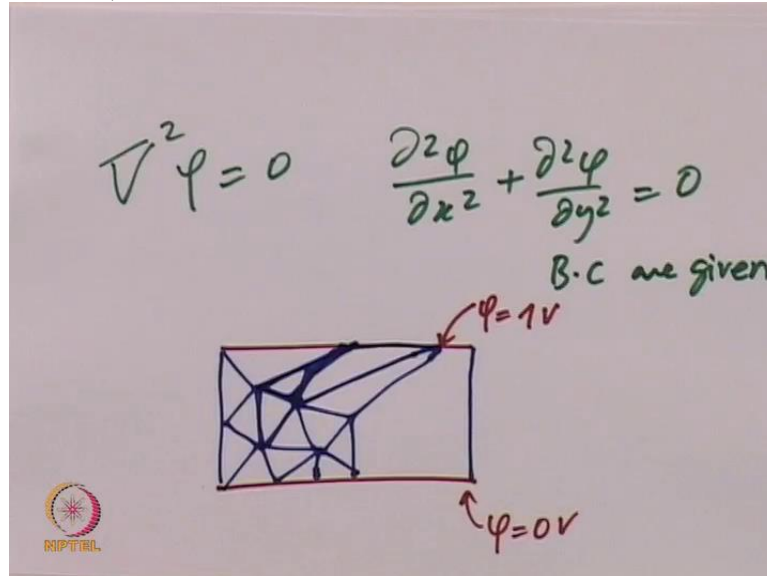
(Refer Slide Time: 00:35)


$$\nabla^2 \phi = 0$$
$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

B.C are given

So let us look into the geometry of the problem it is a Laplace equation. And in two dimension what we are interested is  $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$  and with certain boundary conditions that are given. So let us look at a standard capacitor a parallel plate capacitor.

(Refer Slide Time: 01:03)



So we have a top capacitor plate and a bottom capacitor plate and there is a potential that is applied here. So let us say Phi is equal to 1 volt and there is a bottom potential that is given Phi equal to 0 volt. So we are going to discretize it using triangular discretization. So let us look at how the triangular grid is going to look like

So for example we are interested in using unstructured grid. So the grid is going to be very irregular. So some places you have very fine cells in some places you have very coarse cells and the cell size is going to depend on the gridding or the meshing software. And this is one of the things that we should be very careful. For example here we have a triangle that is very very long in the sides. And there is another triangle which is symmetric with one or two sides whereas these triangles in the middle are very very fine.

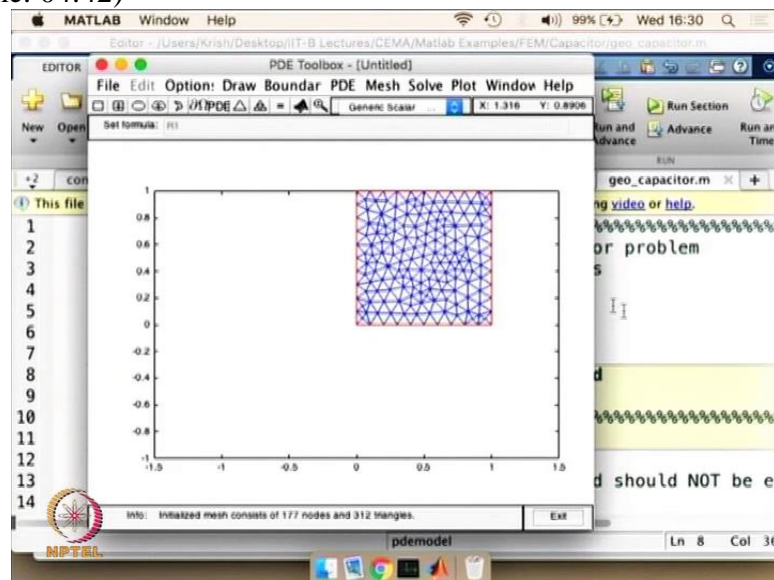
So if you are having a mesh generator and that is not going to be very accurate so your problem should be still run on such outputs of those mesh generators. So we want mesh generator that is going to be very general we cannot rely on the mesh generator accuracy itself what we can do is we can refine the grid but we do not know the quality of the grid that can be produced or that will be produced by any mesh generator.

With that being said the program that you are going to use or going to apply for solving such problem should be able to solve the problem using any mesh. Of course we need to look into the numerical accuracy that we need. If we need better accuracy we have to refine the grid. But what I am trying to say is if your program is able to run on any general grid. For example on any general triangular grid it does not need to be uniform or regular. And that is a good way to start programming. If your program is dependent on the regularity or the uniformity of the grid than you are very specific and you are not able to solve it using a generalized mesh

that can come out of any mesh generator. So when I say generalized mesh what I am pointing out here is generalized triangular grid. So once I know the shape of the grid I should be able to run it. Not to put so much emphasis on the quality of the grid itself because as I said sometimes you cannot control the quality of the meshing and this is mostly the case if you are using any freely available tool. If you are using proprietary software or software that are bought from companies there you are refinement possibilities where you can get better accuracy and the better refinement is a feature that they sell for certain free.

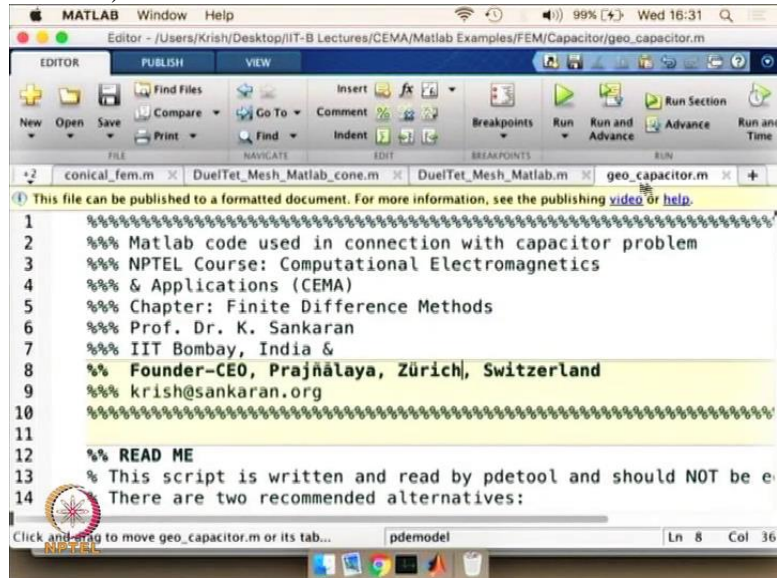
But what we are using for most of this course is a freely available tool or inbuilt tools. So in this case what we are using is a PDE tool box which is an inbuilt feature of the Matlab. And we do not have much control on the way to make the grid we cannot force the grid to be formed in certain way. Si in that case we should have a feature of programming it and able to read any data.

(Refer Slide Time: 04:42)



So let us look at this problem. So what we have as a problem is represented here by this grid so what I have done here is I have used the PDE tool box I can explain you how the PDE tool box works.

(Refer Slide Time: 04:55)

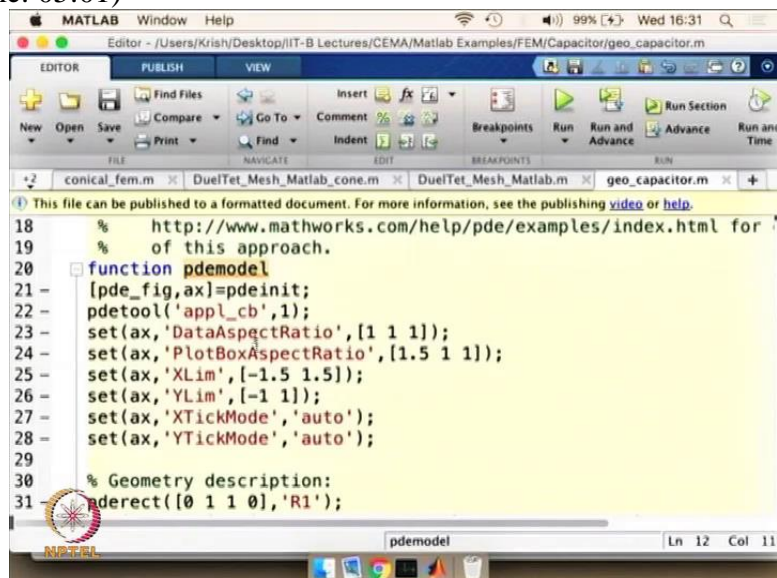


```
MATLAB Window Help
Editor - /Users/Krish/Desktop/IIT-B Lectures/CEMA/Matlab Examples/FEM/Capacitor/geo_capacitor.m

EDITOR PUBLISH VIEW
New Open Save Compare Go To Comment Breakpoints Run Run and Advance Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN
+2 conical_fem.m x DuelTet_Mesh_Matlab_cone.m x DuelTet_Mesh_Matlab.m x geo_capacitor.m x
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %% Matlab code used in connection with capacitor problem
3 %% NPTEL Course: Computational Electromagnetics
4 %% & Applications (CEMA)
5 %% Chapter: Finite Difference Methods
6 %% Prof. Dr. K. Sankaran
7 %% IIT Bombay, India &
8 %% Founder-CEO, Prajñālaya, Zürich, Switzerland
9 %% krish@sankaran.org
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 %% READ ME
13 % This script is written and read by pdetool and should NOT be e
14 % There are two recommended alternatives:
Click and drag to move geo_capacitor.m or its tab... pdemodel Ln 8 Col 36
```

So this is the geometry of the capacitor we have used the inbuilt feature of the PDE tool box.

(Refer Slide Time: 05:01)

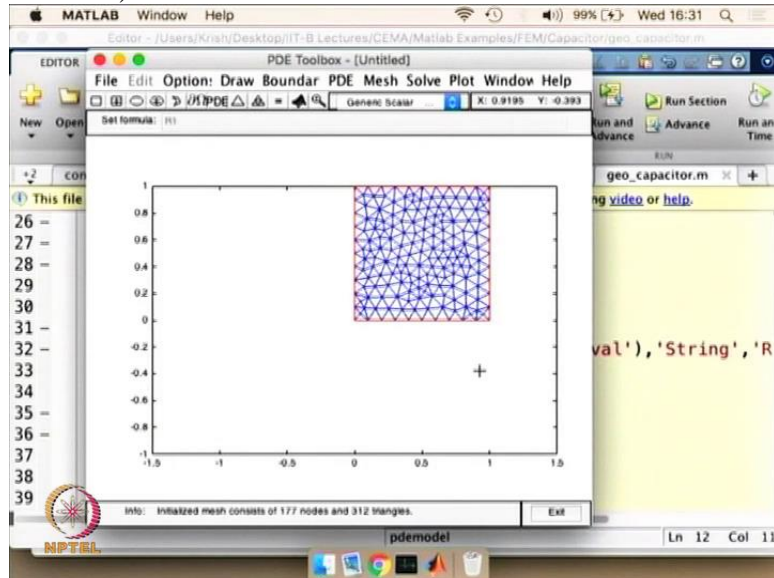


```
MATLAB Window Help
Editor - /Users/Krish/Desktop/IIT-B Lectures/CEMA/Matlab Examples/FEM/Capacitor/geo_capacitor.m

EDITOR PUBLISH VIEW
New Open Save Compare Go To Comment Breakpoints Run Run and Advance Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN
+2 conical_fem.m x DuelTet_Mesh_Matlab_cone.m x DuelTet_Mesh_Matlab.m x geo_capacitor.m x
18 % http://www.mathworks.com/help/pde/examples/index.html for
19 % of this approach.
20 function pdemodel
21 [pde_fig,ax]=pdeinit;
22 pdetool('appl_cb',1);
23 set(ax,'DataAspectRatio',[1 1 1]);
24 set(ax,'PlotBoxAspectRatio',[1.5 1 1]);
25 set(ax,'XLim',[-1.5 1.5]);
26 set(ax,'YLim',[-1 1]);
27 set(ax,'XTickMode','auto');
28 set(ax,'YTickMode','auto');
29
30 % Geometry description:
31 rderect([0 1 1 0],'R1');
pdemodel Ln 12 Col 11
```

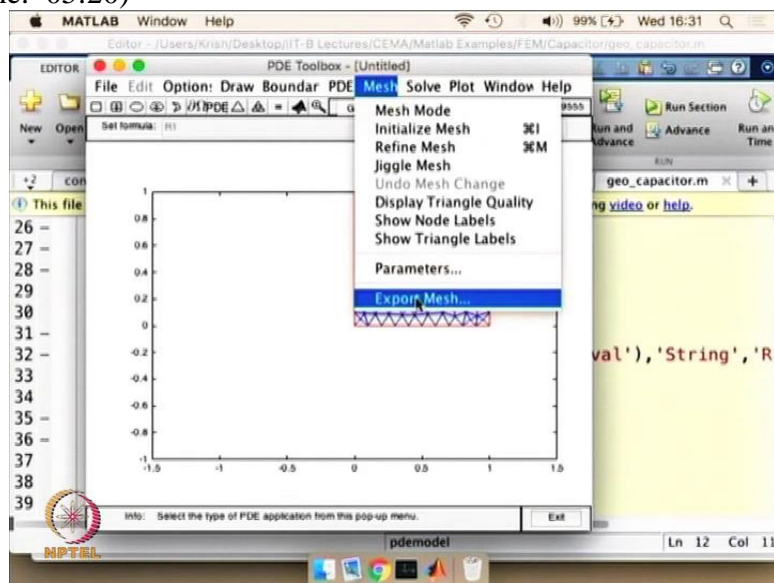
What it does is it creates a model and it assigns certain model and it exports the model the way we want. So let us run this one.

(Refer Slide Time: 05:09)



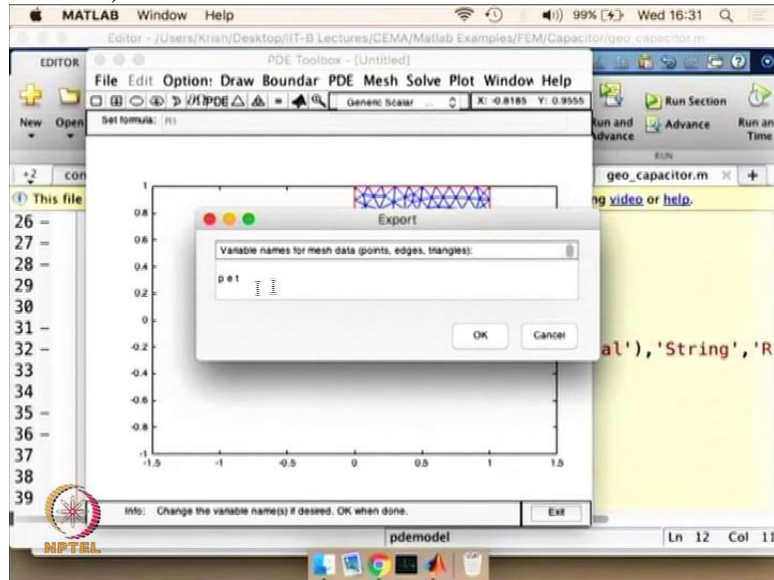
Let us say we want to see the result for a very very basic coarse discretization and we are going to export this mesh.

(Refer Slide Time: 05:20)



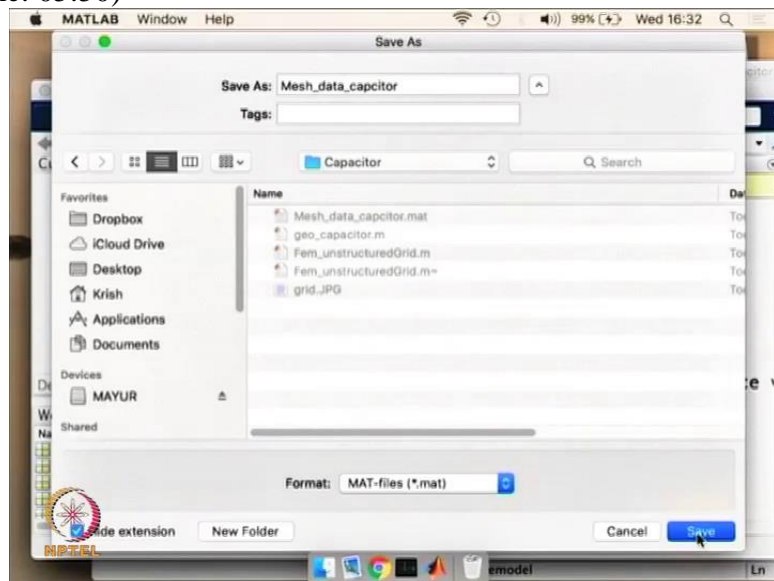
So you go into mesh, export mesh.

(Refer Slide Time: 05:22)



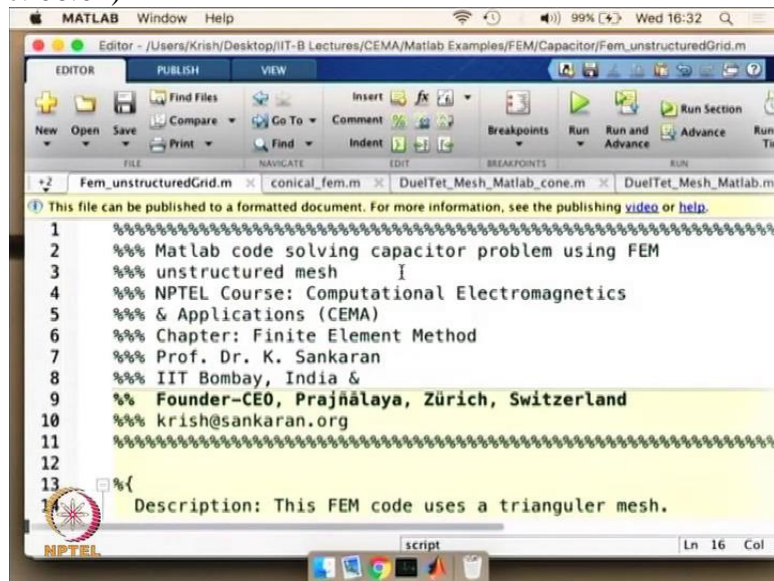
To get the variable names as PET these are the points, edges and triangles we are exporting it and once we have exported it. We can go into the workspace.

(Refer Slide Time: 05:50)



So we can save the workspace. We can give it the name.

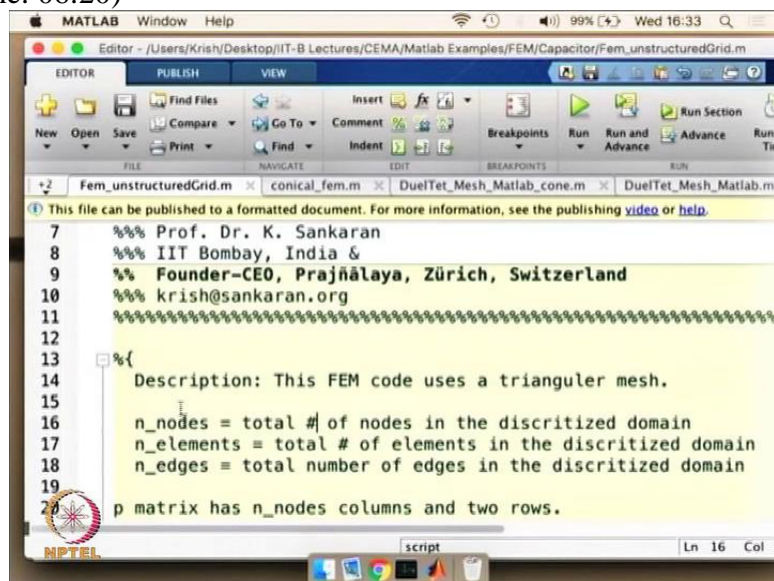
(Refer Slide Time: 06:02)



```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
2  %% Matlab code solving capacitor problem using FEM  
3  %% unstructured mesh  
4  %% NPTEL Course: Computational Electromagnetics  
5  %% & Applications (CEMA)  
6  %% Chapter: Finite Element Method  
7  %% Prof. Dr. K. Sankaran  
8  %% IIT Bombay, India &  
9  %% Founder-CEO, Prajnālaya, Zürich, Switzerland  
10 %% krish@sankaran.org  
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
12  
13  
14 %  
15 Description: This FEM code uses a trianguler mesh.
```

And now let us go into the Matlab code itself. So the Matlab code we have used as a FEM code which is able to solve the problem the Laplace problem using unstructured mesh.

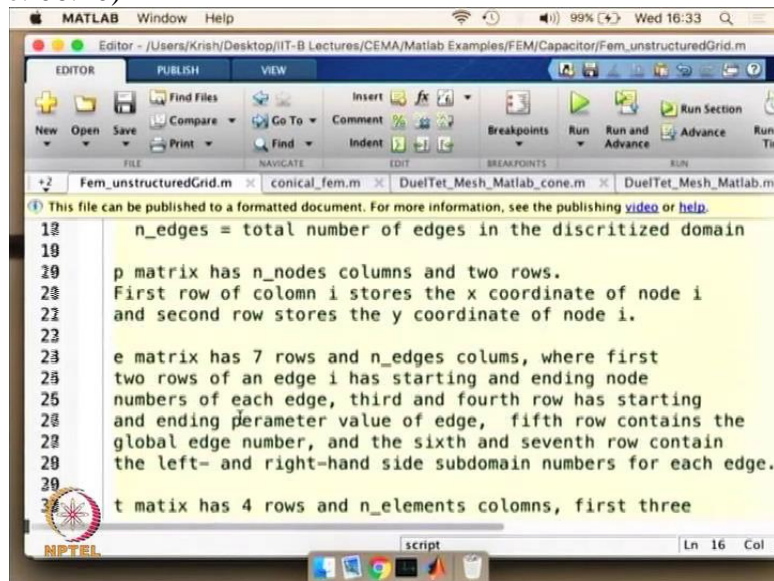
(Refer Slide Time: 06:20)



```
7  %% Prof. Dr. K. Sankaran  
8  %% IIT Bombay, India &  
9  %% Founder-CEO, Prajnālaya, Zürich, Switzerland  
10 %% krish@sankaran.org  
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
12  
13  
14 %  
15 Description: This FEM code uses a trianguler mesh.  
16  
17 n_nodes = total # of nodes in the discretized domain  
18 n_elements = total # of elements in the discretized domain  
19 n_edges = total number of edges in the discretized domain  
20  
21 p matrix has n_nodes columns and two rows.
```

And what it does is it basically reads the data from the PDE tool box and it sets certain parameters for example number of nodes, number of elements, and number of edges. And these are the values it reads from the output of the PDE tool box.

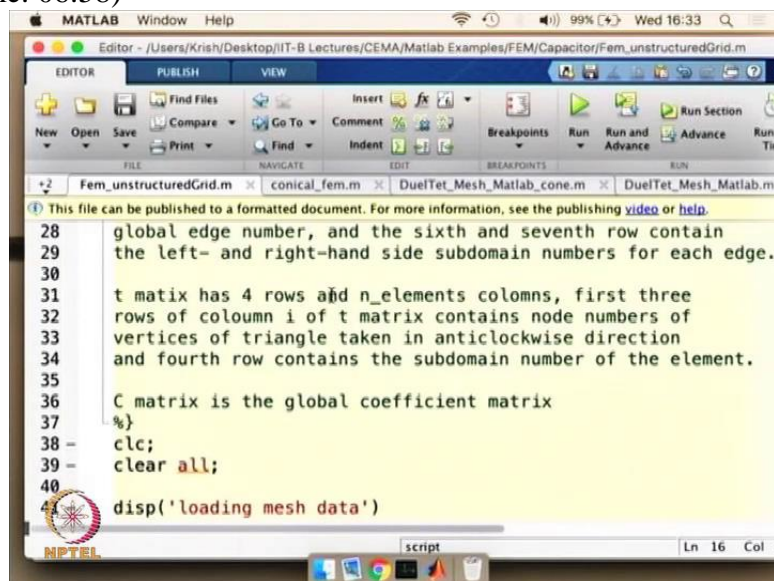
(Refer Slide Time: 06:40)



```
18 n_edges = total number of edges in the discretized domain
19
20 p matrix has n_nodes columns and two rows.
21 First row of column i stores the x coordinate of node i
22 and second row stores the y coordinate of node i.
23
24 e matrix has 7 rows and n_edges columns, where first
25 two rows of an edge i has starting and ending node
26 numbers of each edge, third and fourth row has starting
27 and ending parameter value of edge, fifth row contains the
28 global edge number, and the sixth and seventh row contain
29 the left- and right-hand side subdomain numbers for each edge.
30
31 t matrix has 4 rows and n_elements columns, first three
```

The first matrix what it creates is the P matrix, the E matrix is the second one and the T matrix is the third one. So as I said the P matrix corresponds to the point matrix the various points in the mesh. E matrix are the edges matrix.

(Refer Slide Time: 06:58)

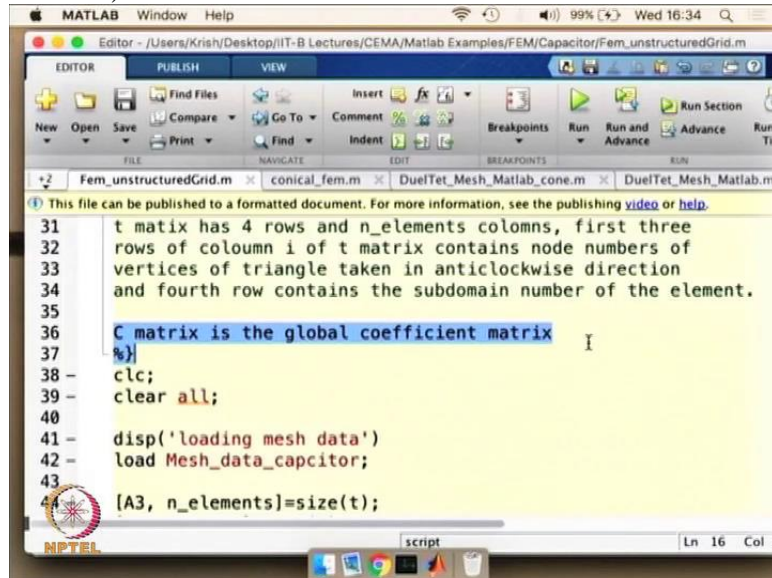


```
28 global edge number, and the sixth and seventh row contain
29 the left- and right-hand side subdomain numbers for each edge.
30
31 t matrix has 4 rows and n_elements columns, first three
32 rows of column i of t matrix contains node numbers of
33 vertices of triangle taken in anticlockwise direction
34 and fourth row contains the subdomain number of the element.
35
36 C matrix is the global coefficient matrix
37 %}
38 - clc;
39 - clear all;
40
41 disp('loading mesh data')
```

And the T matrix correspond to the different triangles. And what is important to know is there are 4 rows and n elements. So with the first three rows of the each column is corresponding to the nodes of the different triangles for example each triangle is represented as a i and the first three columns are the three columns taken in the anti clockwise. The fourth row contains sub domain information if the domain is the main domain or part of the other domain. In this case we do not have any sub domain so we do not need to worry about this.



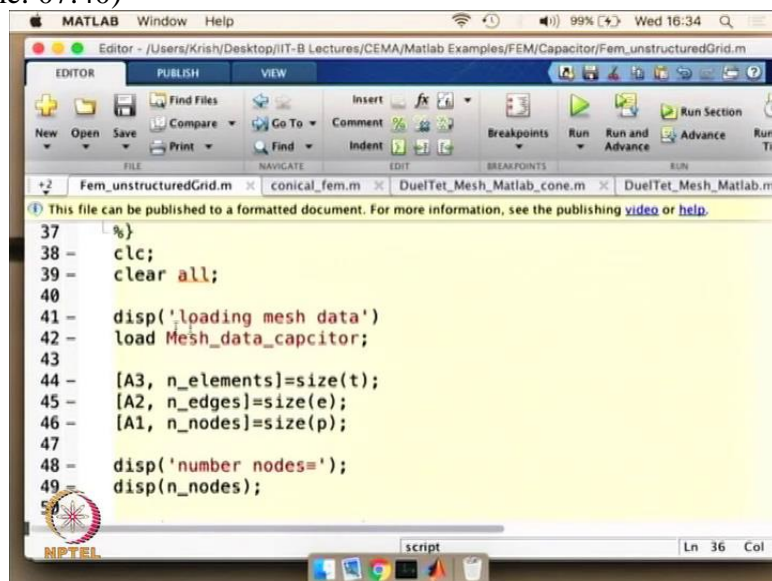
(Refer Slide Time: 07:30)



```
31 t matrix has 4 rows and n_elements columns, first three
32 rows of column i of t matrix contains node numbers of
33 vertices of triangle taken in anticlockwise direction
34 and fourth row contains the subdomain number of the element.
35
36 C matrix is the global coefficient matrix
37 %}
38 - clc;
39 - clear all;
40
41 - disp('loading mesh data')
42 - load Mesh_data_capcitor;
43
44 [A3, n_elements]=size(t);
```

And what it also does is it creates the C matrix. Which is the global coefficient matrix that we will be using.

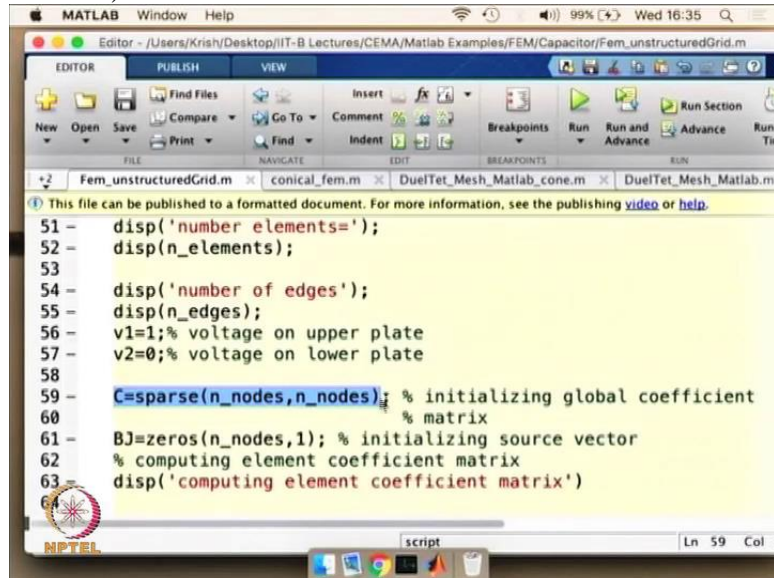
(Refer Slide Time: 07:40)



```
37 %}
38 - clc;
39 - clear all;
40
41 - disp('loading mesh data')
42 - load Mesh_data_capcitor;
43
44 [A3, n_elements]=size(t);
45 [A2, n_edges]=size(e);
46 [A1, n_nodes]=size(p);
47
48 - disp('number nodes=');
49 - disp(n_nodes);
50
```

And initially what it does is it loads the mesh data. So it prints loading mesh data and it loads the mesh data what we have stored. And it goes and assigns those T E and P matrix to various parameters so A3 N elements are the number of elements basically these are the triangles and the size of E will be the number of edges, number of nodes will be the size of P.

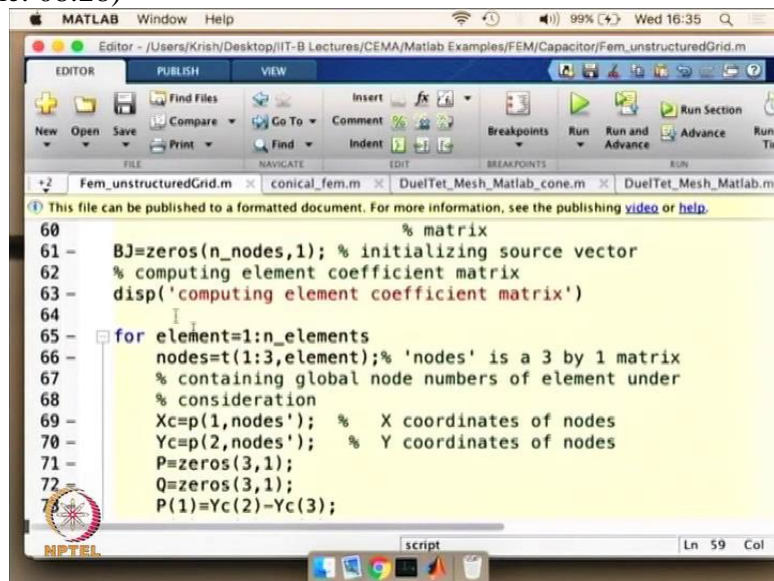
(Refer Slide Time: 08:15)



```
51- disp('number elements=');
52- disp(n_elements);
53
54- disp('number of edges');
55- disp(n_edges);
56- v1=1;% voltage on upper plate
57- v2=0;% voltage on lower plate
58
59- C=sparse(n_nodes,n_nodes); % initializing global coefficient
60-                                     % matrix
61- BJ=zeros(n_nodes,1); % initializing source vector
62- % computing element coefficient matrix
63- disp('computing element coefficient matrix')
```

And it displays those values and what it does is it creates a matrix and defines that matrix has a sparse matrix. The reason for doing that is to make the computation fast.

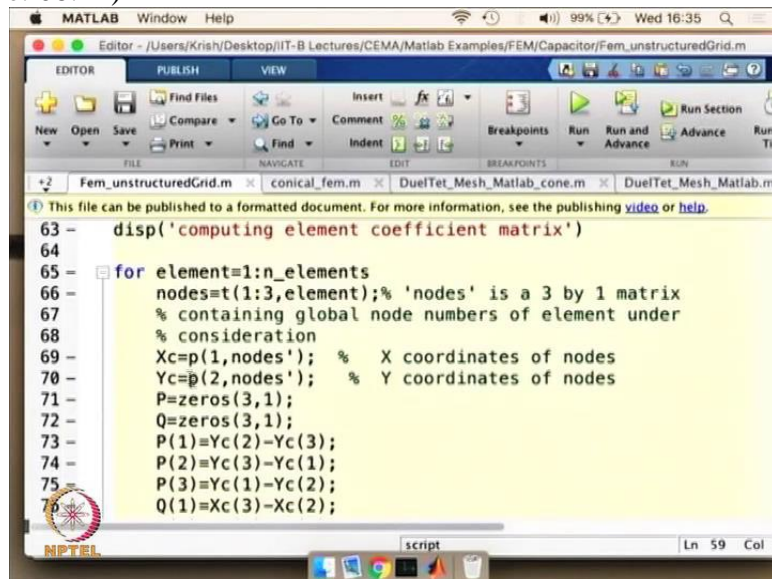
(Refer Slide Time: 08:28)



```
60-                                     % matrix
61- BJ=zeros(n_nodes,1); % initializing source vector
62- % computing element coefficient matrix
63- disp('computing element coefficient matrix')
64
65- for element=1:n_elements
66-     nodes=t(1:3,element);% 'nodes' is a 3 by 1 matrix
67-     % containing global node numbers of element under
68-     % consideration
69-     Xc=p(1,nodes'); % X coordinates of nodes
70-     Yc=p(2,nodes'); % Y coordinates of nodes
71-     P=zeros(3,1);
72-     Q=zeros(3,1);
73-     P(1)=Yc(2)-Yc(3);
```

And it creates the source matrix or the source vector to be precise. So it creates the source vector BJ and initializes it and it displays the computing element coefficient matrix.

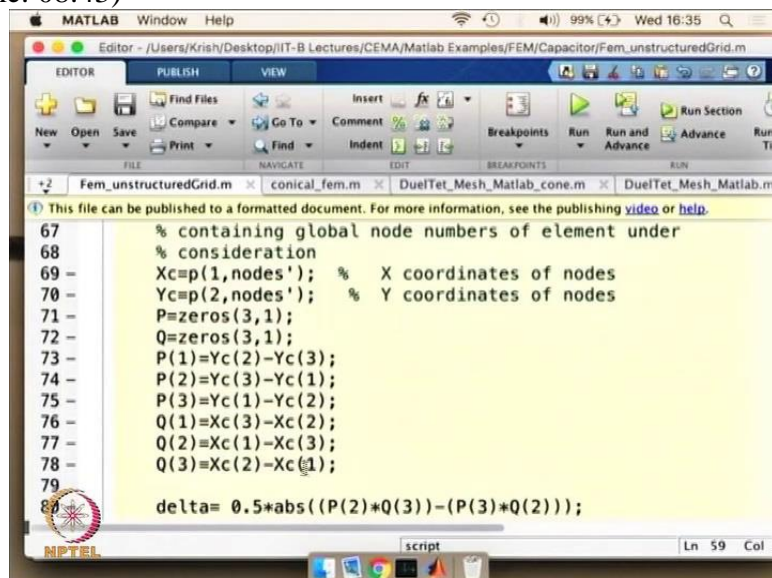
(Refer Slide Time: 08:41)



```
63 - disp('computing element coefficient matrix')
64
65 - for element=1:n_elements
66 -     nodes=t(1:3,element);% 'nodes' is a 3 by 1 matrix
67 -     % containing global node numbers of element under
68 -     % consideration
69 -     Xc=p(1,nodes'); % X coordinates of nodes
70 -     Yc=p(2,nodes'); % Y coordinates of nodes
71 -     P=zeros(3,1);
72 -     Q=zeros(3,1);
73 -     P(1)=Yc(2)-Yc(3);
74 -     P(2)=Yc(3)-Yc(1);
75 -     P(3)=Yc(1)-Yc(2);
76 -     Q(1)=Xc(3)-Xc(2);
```

So it goes into each of the elements.

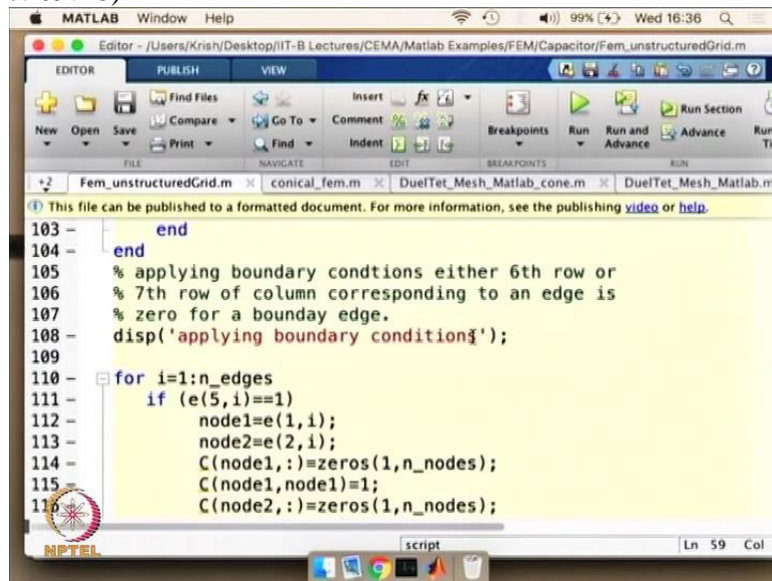
(Refer Slide Time: 08:43)



```
67 - % containing global node numbers of element under
68 - % consideration
69 - Xc=p(1,nodes'); % X coordinates of nodes
70 - Yc=p(2,nodes'); % Y coordinates of nodes
71 - P=zeros(3,1);
72 - Q=zeros(3,1);
73 - P(1)=Yc(2)-Yc(3);
74 - P(2)=Yc(3)-Yc(1);
75 - P(3)=Yc(1)-Yc(2);
76 - Q(1)=Xc(3)-Xc(2);
77 - Q(2)=Xc(1)-Xc(3);
78 - Q(3)=Xc(2)-Xc(1);
79
80 - delta= 0.5*abs((P(2)*Q(3))-(P(3)*Q(2)));
```

And from each of the elements it computes the value of the elements delta. Delta is the area of the each of the elements and it uses that value into the local matrix. And once the local matrix c is created. It concatenates the local matrix to form the global coefficient matrix. And that is what you see. You see the capital C is the global coefficient matrix and the c is the local matrix.

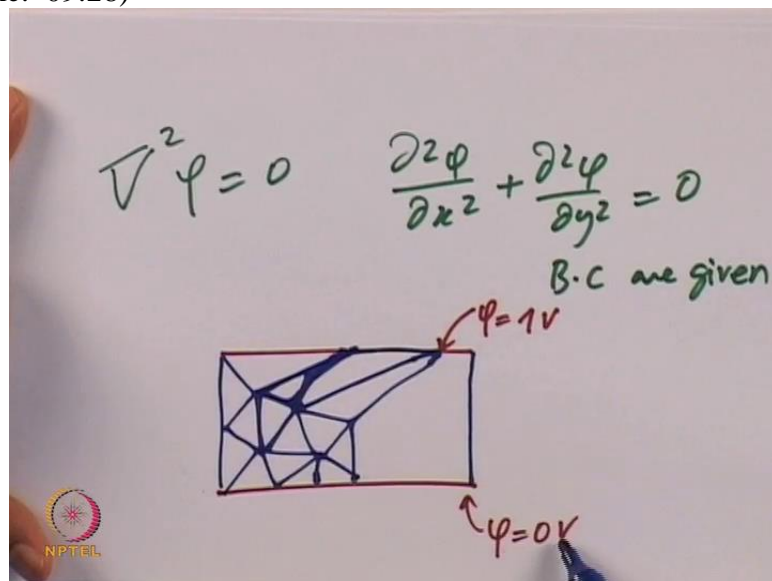
(Refer Slide Time: 09:18)



```
103 -     end
104 -   end
105   % applying boundary condtions either 6th row or
106   % 7th row of column corresponding to an edge is
107   % zero for a bounday edge.
108 -   disp('applying boundary condition$');
109
110 -   for i=1:n_edges
111 -     if (e(5,i)==1)
112 -       node1=e(1,i);
113 -       node2=e(2,i);
114 -       C(node1,:)=zeros(1,n_nodes);
115 -       C(node1,node1)=1;
116 -       C(node2,:)=zeros(1,n_nodes);
```

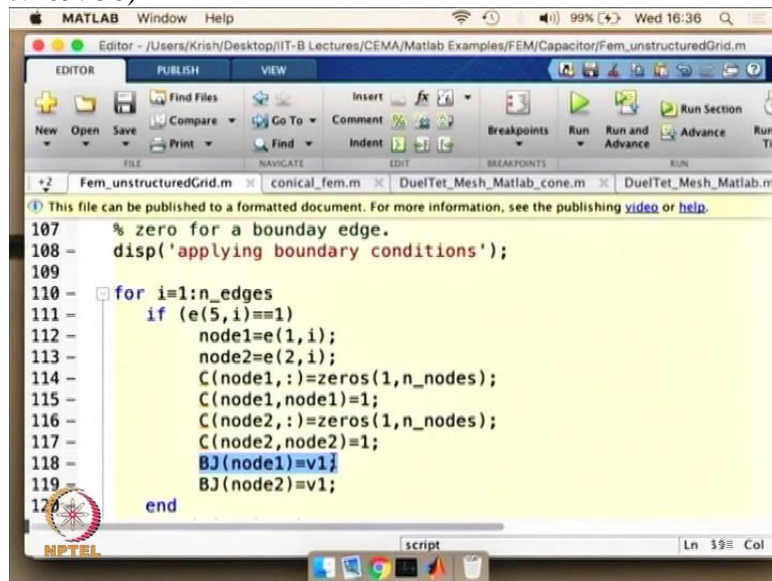
And once it does that it applies the boundary condition as I said the boundary condition what we are looking into is

(Refer Slide Time: 09:28)



As it is said in the initial sheet we have phi is equal to 1 on the top plate and Phi equal to 0 on the bottom plate. And that is what you will see here.

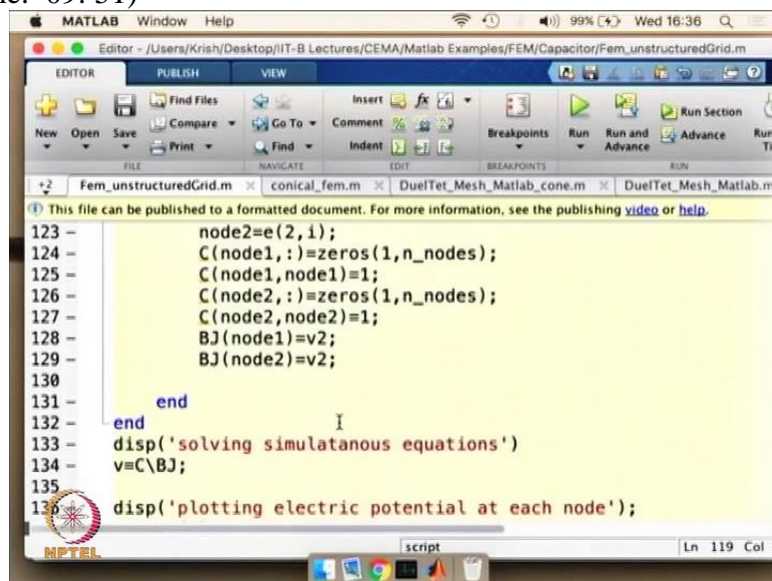
(Refer Slide Time: 09: 38)



```
107 % zero for a boundary edge.
108 disp('applying boundary conditions');
109
110 for i=1:n_edges
111     if (e(5,i)==1)
112         node1=e(1,i);
113         node2=e(2,i);
114         C(node1,:)=zeros(1,n_nodes);
115         C(node1,node1)=1;
116         C(node2,:)=zeros(1,n_nodes);
117         C(node2,node2)=1;
118         BJ(node1)=v1;
119         BJ(node2)=v1;
120     end
```

We are going to apply the boundary condition as  $v_1$  on the nodes of the top plate, so these are nodes on the top plate.

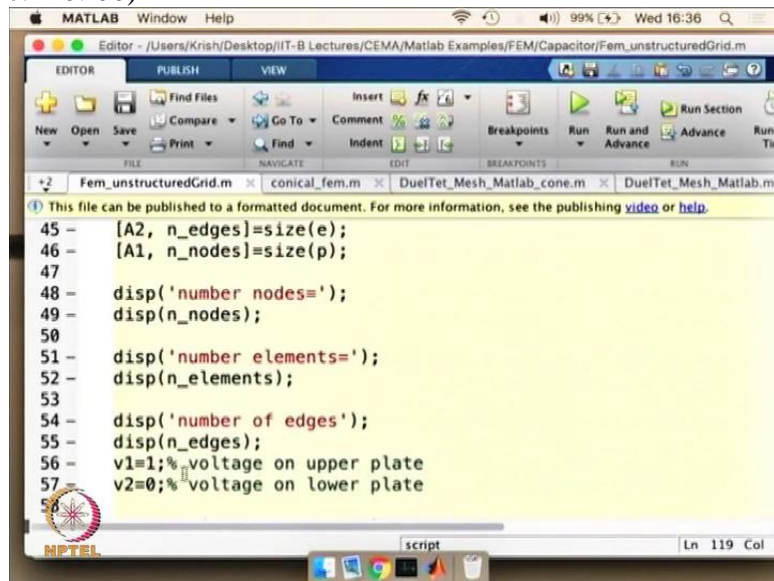
(Refer Slide Time: 09: 51)



```
123     node2=e(2,i);
124     C(node1,:)=zeros(1,n_nodes);
125     C(node1,node1)=1;
126     C(node2,:)=zeros(1,n_nodes);
127     C(node2,node2)=1;
128     BJ(node1)=v2;
129     BJ(node2)=v2;
130
131 end
132 end
133 disp('solving simulanous equations')
134 v=C\BJ;
135
136 disp('plotting electric potential at each node');
```

And we are going to set  $v_2$  for the bottom plate. And the value of  $v_1$  and  $v_2$  is given on the top.

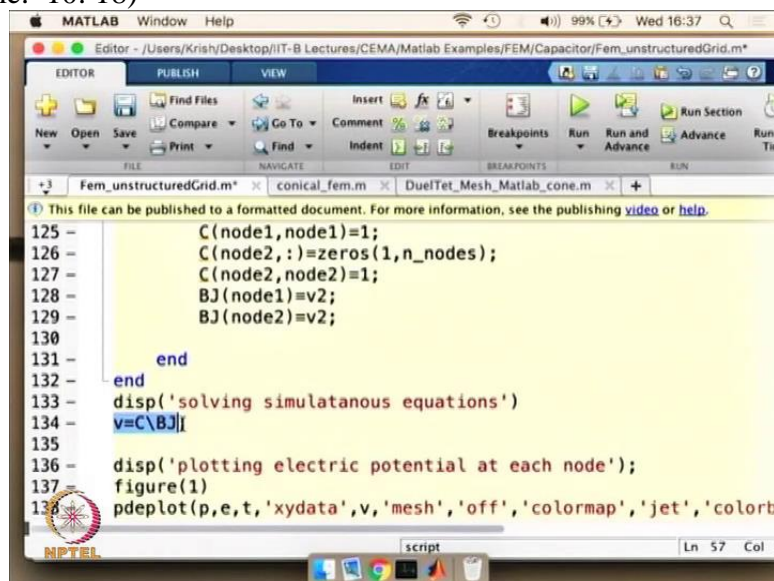
(Refer Slide Time: 10: 00)



```
45 - [A2, n_edges]=size(e);
46 - [A1, n_nodes]=size(p);
47
48 - disp('number nodes=');
49 - disp(n_nodes);
50
51 - disp('number elements=');
52 - disp(n_elements);
53
54 - disp('number of edges');
55 - disp(n_edges);
56 - v1=1;% voltage on upper plate
57 - v2=0;% voltage on lower plate
58
```

v1 is the upper plate voltage so we can write them as the potential and the lower plate potential. So these are all in volts.

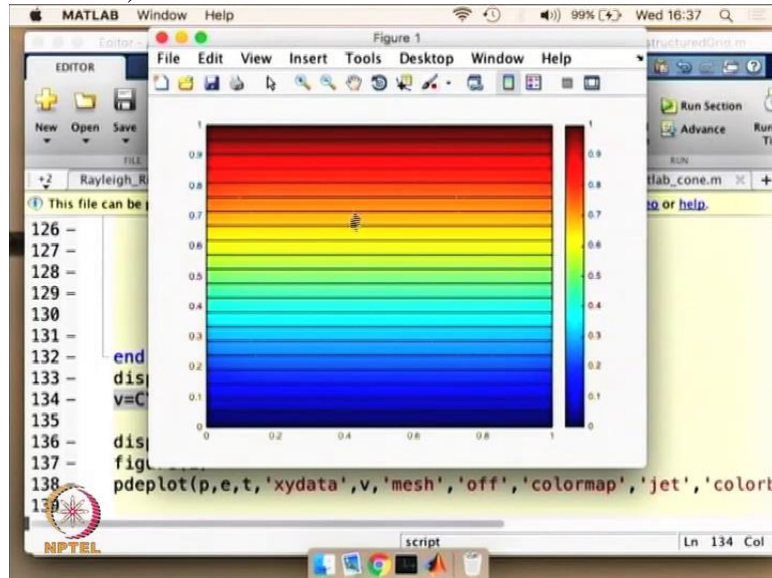
(Refer Slide Time: 10: 18)



```
125 - C(node1,node1)=1;
126 - C(node2,:)=zeros(1,n_nodes);
127 - C(node2,node2)=1;
128 - BJ(node1)=v2;
129 - BJ(node2)=v2;
130
131 - end
132 - end
133 - disp('solving simulanatous equations')
134 - v=C\BJ;
135
136 - disp('plotting electric potential at each node');
137 - figure(1)
138 - pdeplot(p,e,t,'xydata',v,'mesh','off','colormap','jet','colorb
```

And this is applied in the boundary condition and we are going to solve the simultaneous equation given by this expression. We are going to compute the various potential v and we are going to invert the matrix c in order to get the result and BJ is the source matrix. And the plotted and the computed value of the electric potential are plotted in the fig 1. So let us run this code and see what is happening.

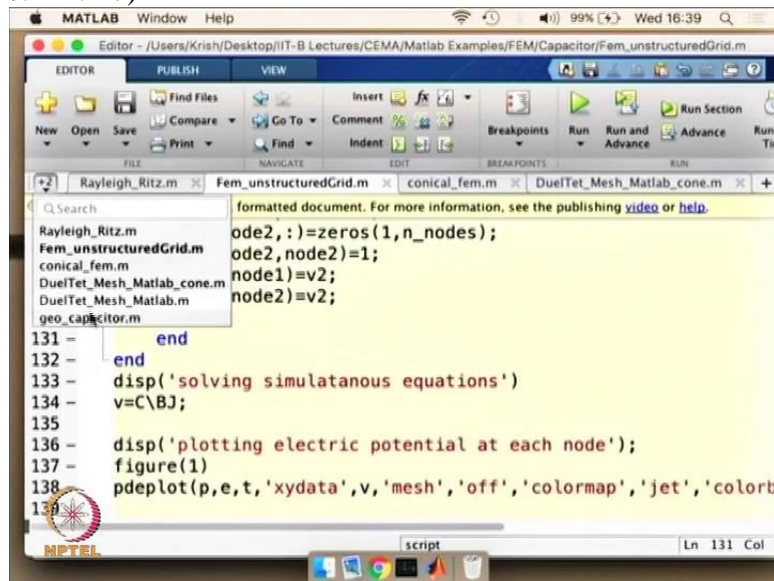
(Refer Slide Time: 10: 53)



We see that the potential is going to be 1 and the bottom plate is going to be 0 as we can see. And there is a natural degradation there is a linear degradation from top plate to bottom plate. Here there is an assumption on the left hand side and the right hand side. The assumption is the fields are going to be following certain patterns as if the plate is infinitely long in this direction and this direction. So it is going to have assuming that the top direction is y, the horizontal direction is going to be x. It assumes that the plates are infinitely long in the x direction. So the fields values here the normal component of the fields here are going to be seen continuous. So you do not see any kind of deviation, if there all looking as if the plate is infinitely long.

What is also important to know is once we refine the grid to the level what we want we can get the accuracy what we want the way we want. We can go and refine the grid even further.

(Refer Slide Time: 12: 17)

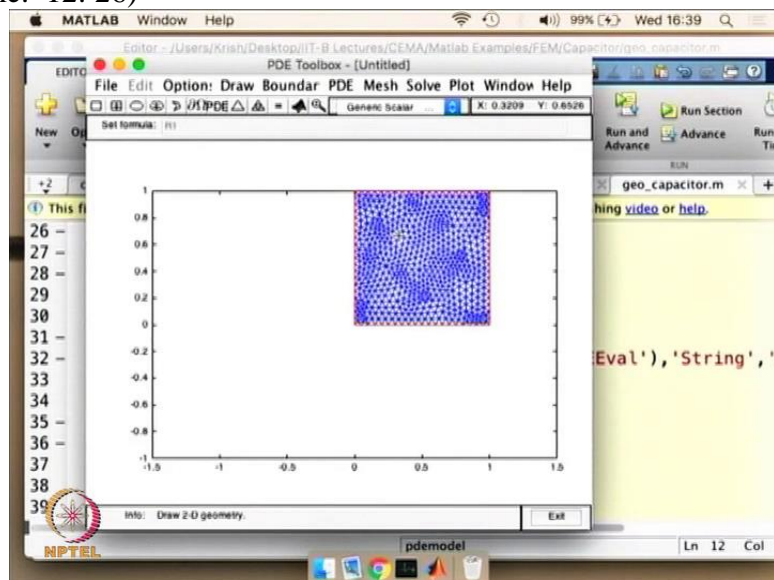


```
ode2,:)=zeros(1,n_nodes);
ode2,node2)=1;
node1)=v2;
node2)=v2;

131 - end
132 - end
133 - disp('solving simulanous equations')
134 - v=C\BJ;
135
136 - disp('plotting electric potential at each node');
137 - figure(1)
138 - pdeplot(p,e,t,'xydata',v,'mesh','off','colormap','jet','colorb
139
```

For example we can go and run this geocapacitor PDE tool box one more time.

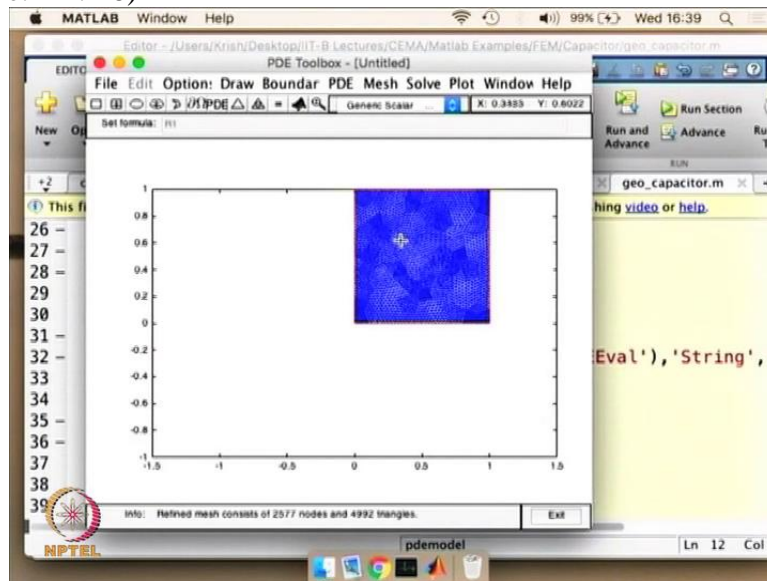
(Refer Slide Time: 12: 26)



And we can refine the grid even further

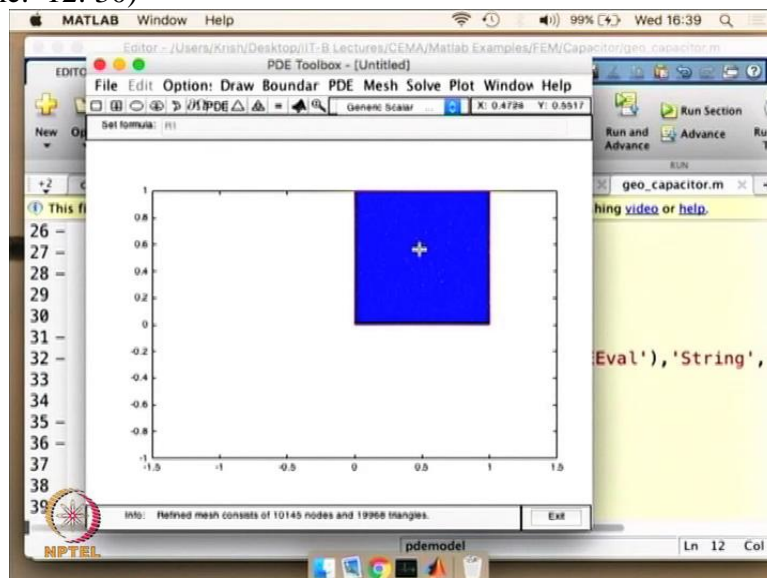


(Refer Slide Time: 12: 28)



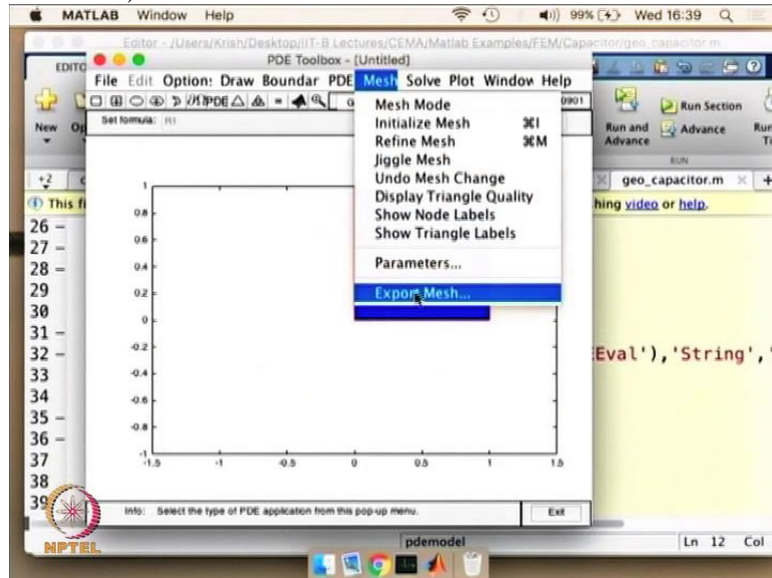
For example if you see here I have refined it even further.

(Refer Slide Time: 12: 30)



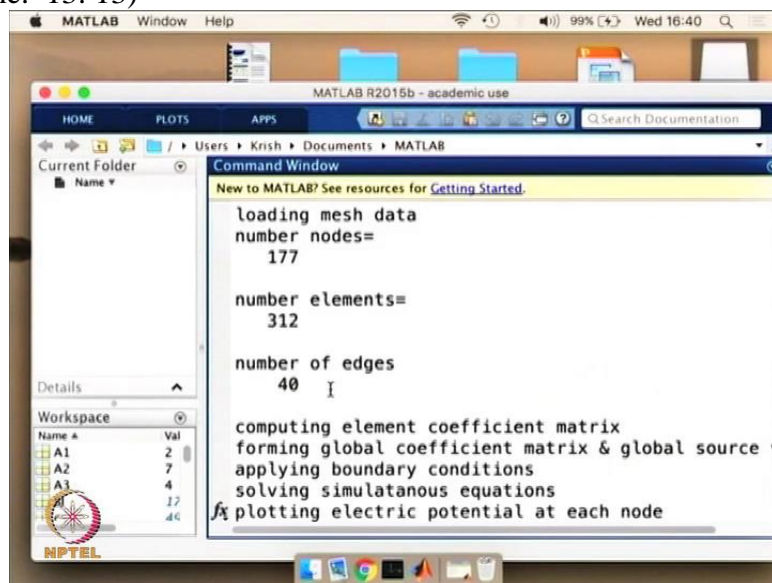
This is the maximum refinement possible but once you refine it so much what is going to happen is the time it takes for you to run the code is going to take longer. So this is the basic discretization which we ran. And we can go and refine the grid even further.

(Refer Slide Time: 12: 50)



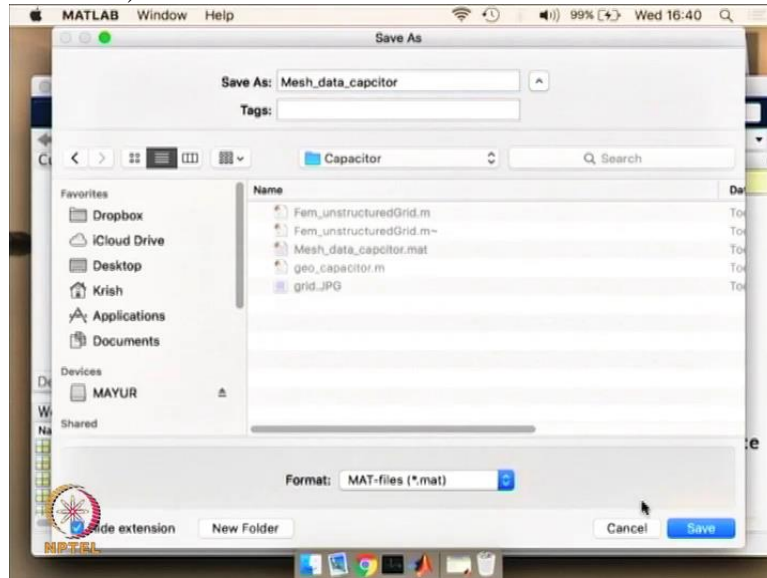
So for example in this case we can also do the same thing what we did before export the mesh you can say ok, and we can exit.

(Refer Slide Time: 13: 13)



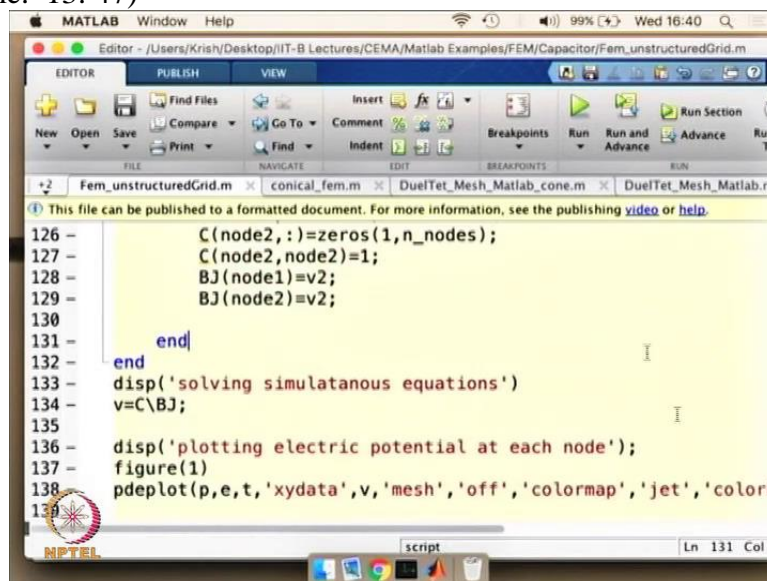
And what we can see is in the previous case when we did the computation we had 177 nodes 312 elements and 40 edges. But now we have refined it so much, we expect that the number of nodes number of edges are going to be dramatically higher.

(Refer Slide Time: 13: 24)



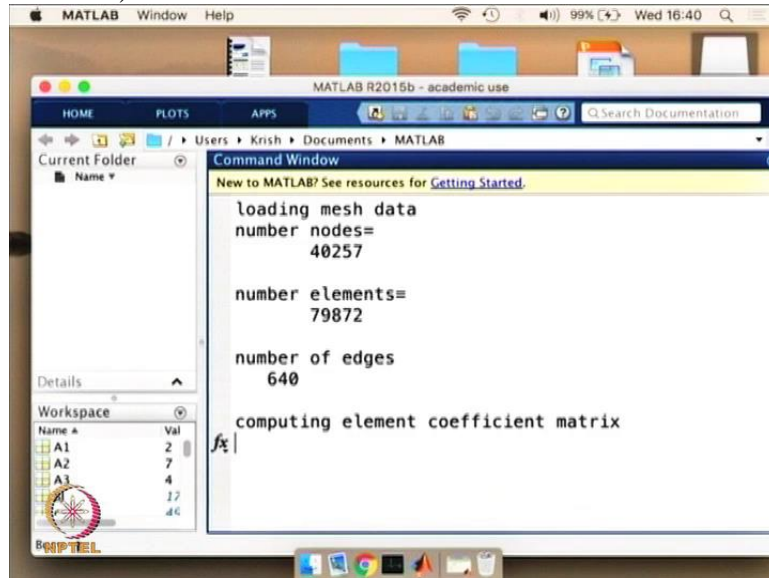
So let us go and save this particular matrix data so we are going to save it once again same name. So mesh data capacitor. Replace

(Refer Slide Time: 13: 47)



And now we are going to go and run the program one more time. So let us go and run the program FEM unstructured grid one more time. And it is going to take a long time because there are going to be so many elements. So let us run it.

(Refer Slide Time: 14: 06)



The image shows a MATLAB R2015b Command Window. The Command Window displays the following text:

```
loading mesh data
number nodes=
    40257

number elements=
    79872

number of edges
    640

computing element coefficient matrix
```

The Command Window also shows a message: "New to MATLAB? See resources for Getting Started." The Workspace window shows the following variables:

Name	Val
A1	2
A2	7
A3	4
	17
	46

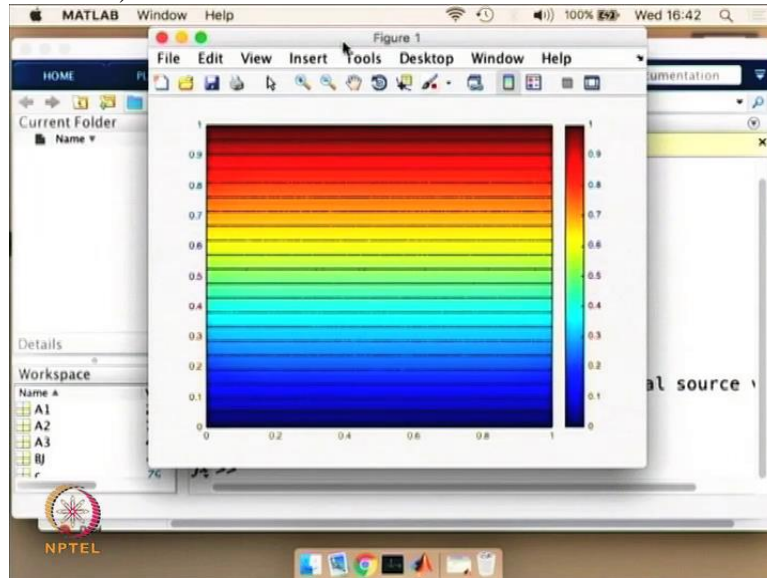
And as you can see the solution is going to take a long time. So it is very busy now it is computing the element coefficient matrix.

And as you can see compared to the last time it is going to have 640 earlier we had only 40 edges now we have 640 edges. This is a much finer discretization. So it is going to take a very long time for us to get the result. What I wanted to show is in certain problems it does not really matter how much cells you are going to have because the solution itself will be symmetric and you can use the symmetry of the solution to get a better result in a quicker way or in a very easy way. In this case I have done the problem so as to show you how one can use PDE tool box and combine it with Finite element method and as I said this grid is going to take such a long time it is still computing the coefficient matrix forming the global coefficient matrix and once it is done it is going to show the result.

And let us see what is going to be the difference between a coarse grid what we had and a very very very fine grid. And does it make sense.

And as I told you one of the thing that one has to learn during the process of this lecture is how we can make a informed choice of the discretization the method that we are going to use and the kind of resolution that we want.

(Refer Slide Time: 15: 38)



For example if you see this solution what we have got we did not gain anything that much if you see the result what we have got here looks very similar to what we have got even before. Because the problem itself is very symmetric and it is a straight forward problem there is not much material discontinuity or material changes permittivity and permeability changes or domain itself is not that complicated. So we are not gaining that much. So we will do yet another problem using the same finite element method. And we will see how the discretization going to improve our accuracy in that case.

So with that being said we will stop here. What we have seen is a very simple Laplace problem. We have used an inbuilt mesh generator called as PDE toolbox and we have used nodal based FEM method using unstructured grid to solve a very simple Laplace problem. And we have seen how the accuracy is going to change whether it makes sense to go for a finite grid or coarser grid so on and so forth. And as I told you in the next module in the next exercises what we will see is a very different type of capacitor problem where refinement helps us in gaining something. So that is the focus of the next module. See you in the next module. Thank you!