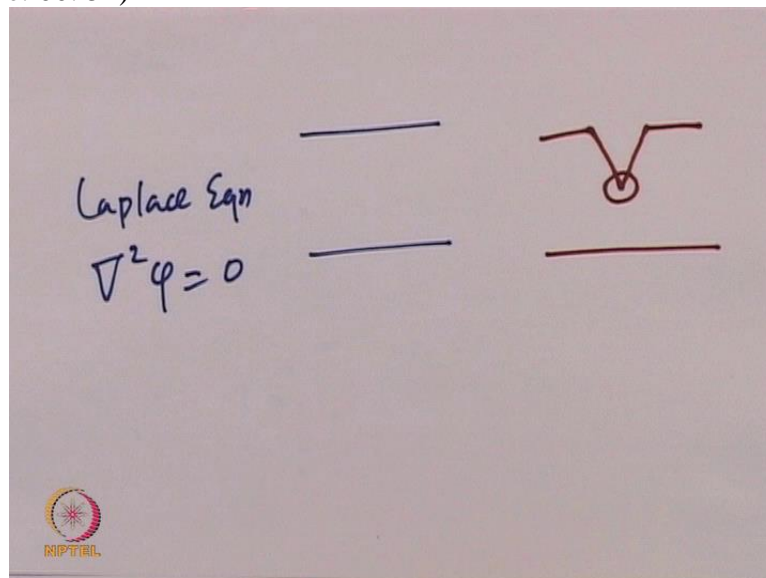


Computational Electromagnetics and Applications
Professor Krish Sankaran
Indian Institute of Technology Bombay
Exercise No. 16
Finite Element Method-II

We saw in the previous exercise that using a PDE tool box we are able to solve very simple Laplace equation which is a model for a capacitor problem. And we also saw the impact of the discretization how is it impacting on the resolution whether it makes sense to go finer for a simple problem so on and so forth.

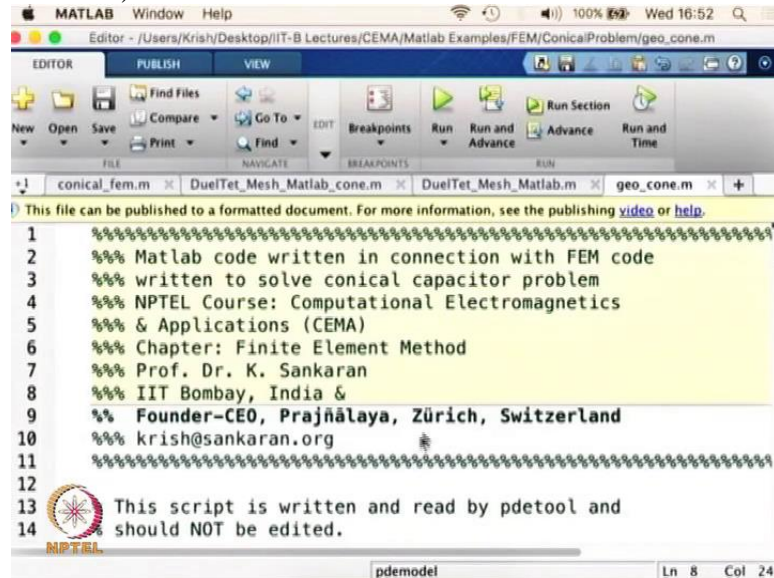
And I mentioned now we will look into a problem where refining the grid makes sense and we have chosen the problem specifically for that particular reason. So that we are exaggerating the impact of discretization in the problem resolution.


(Refer Slide Time: 00: 51)



We are going to again start with the Laplace equation. Now instead of having a parallel plate like in the case before we are going to use a different kind of capacitor, it is not a parallel plate capacitor whereas it is going to have a kind of a dip a conical dip. And the reason for doing that is in the sharp edge here what we see the resolution of electric potential and also the potential difference is going to be difficult because the sharp edge itself will create certain inaccuracy and some singularity in some case, So we are going to see how we can model this problem using the PDE tool box which is a inbuilt functionality of the Matlab.

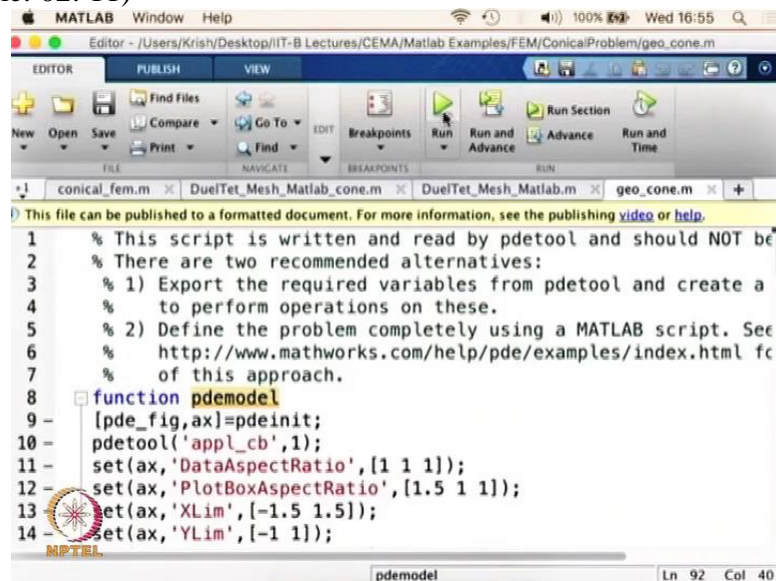
(Refer Slide Time: 01: 55)



```
MATLAB Window Help
Editor - /Users/Krish/Desktop/IIT-B Lectures/CEMA/Matlab Examples/FEM/ConicalProblem/geo_cone.m
EDITOR PUBLISH VIEW
New Open Save Compare Go To Find Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE BREAKPOINTS RUN
This file can be published to a formatted document. For more information, see the publishing video or help.
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%% Matlab code written in connection with FEM code
3 %%% written to solve conical capacitor problem
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, Prajñālaya, Zürich, Switzerland
10 %%% krish@sankaran.org
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12
13  This script is written and read by pdetool and
14 % should NOT be edited.
pdemodel Ln 8 Col 24
```

So let us start looking at the problem itself in the Matlab. So we are going to start with the Matlab code that is being written to generate the mesh in the form of a PDE model.

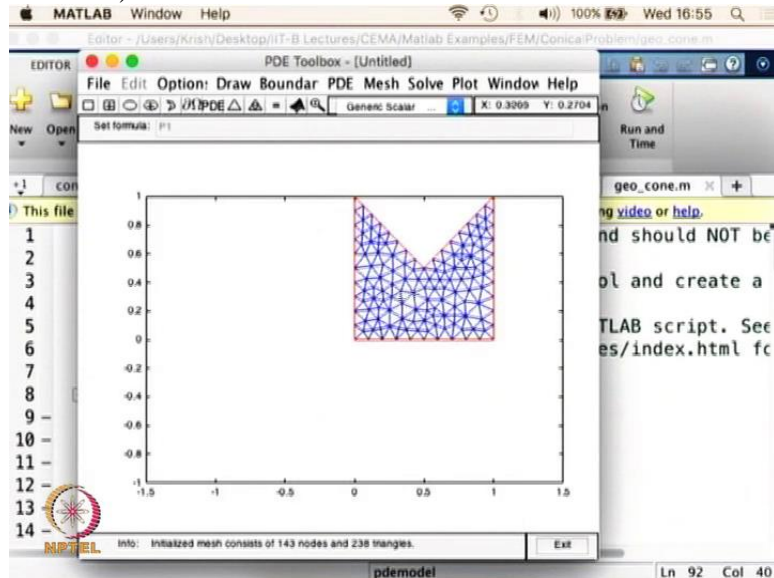
(Refer Slide Time: 02: 11)



```
MATLAB Window Help
Editor - /Users/Krish/Desktop/IIT-B Lectures/CEMA/Matlab Examples/FEM/ConicalProblem/geo_cone.m
EDITOR PUBLISH VIEW
New Open Save Compare Go To Find Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE BREAKPOINTS RUN
This file can be published to a formatted document. For more information, see the publishing video or help.
1 % This script is written and read by pdetool and should NOT be
2 % There are two recommended alternatives:
3 % 1) Export the required variables from pdetool and create a
4 % to perform operations on these.
5 % 2) Define the problem completely using a MATLAB script. See
6 % http://www.mathworks.com/help/pde/examples/index.html for
7 % of this approach.
8 function pdemodel
9 [pde_fig,ax]=pdeinit;
10 pdetool('appl_cb',1);
11 set(ax,'DataAspectRatio',[1 1 1]);
12 set(ax,'PlotBoxAspectRatio',[1.5 1 1]);
13 set(ax,'XLim',[-1.5 1.5]);
14 set(ax,'YLim',[-1 1]);
pdemodel Ln 92 Col 40
```

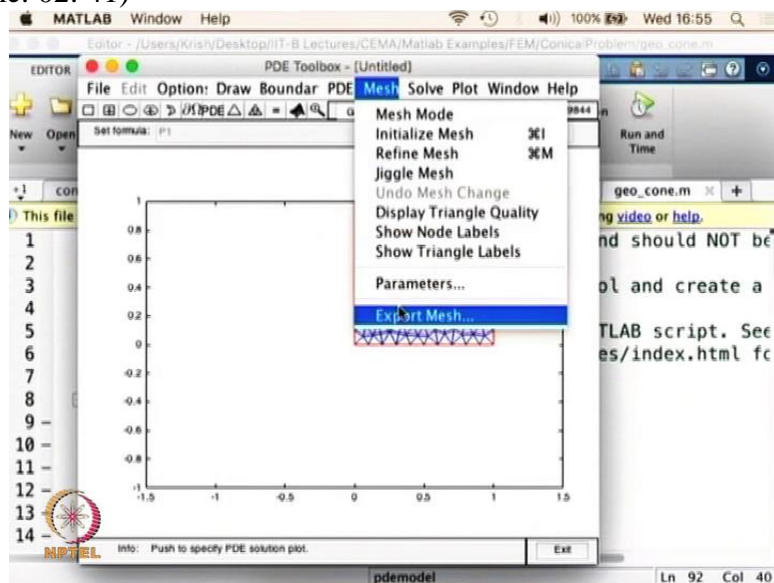
So now we are going to run the geometry generating or the PDE model generating PDE function.

(Refer Slide Time: 02: 21)



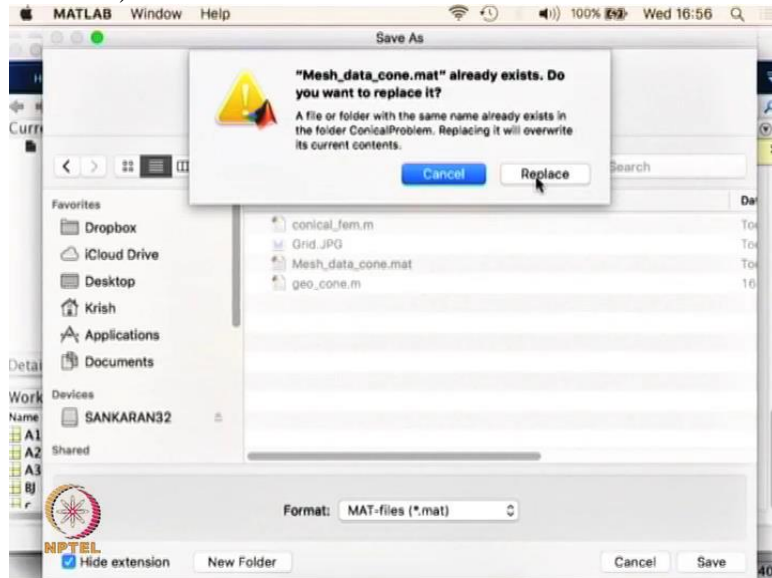
And we are going to get the mesh that we want. So initially we will start with a very coarse mesh. This is a geometry that I have described on paper. And we have now using this geometry and we are meshing it. And we have kept very coarse discretization as you can see.

(Refer Slide Time: 02: 41)



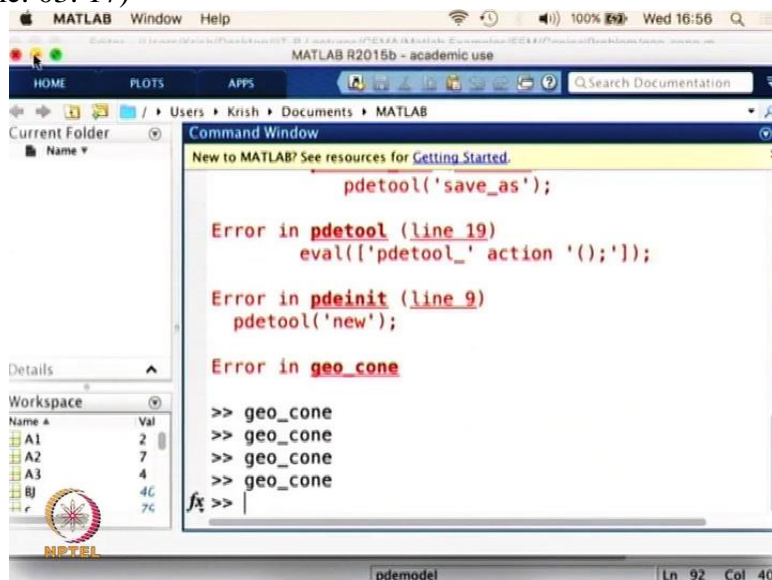
And we are now going to export the mesh as a PET matrix. And now we are going to close this one. And then we are going to go and save the home workspace in terms of a material matrix.

(Refer Slide Time: 03: 03)



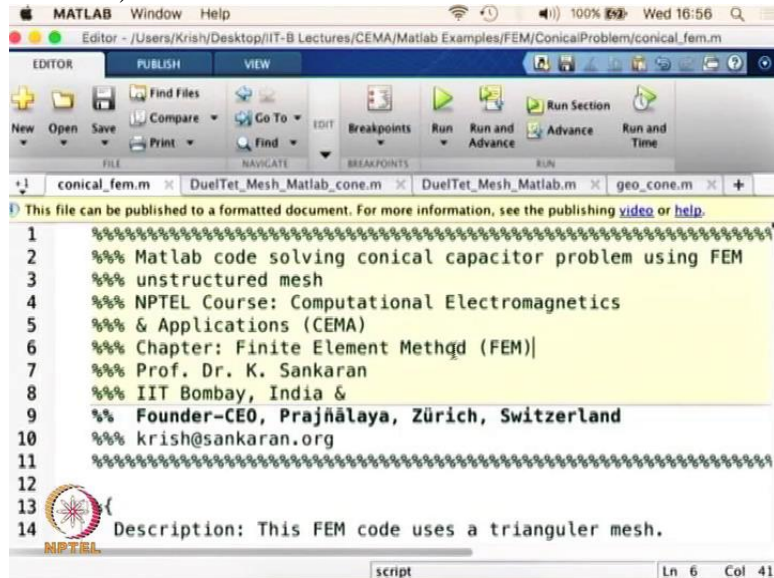
So capacitor, conical problem, mesh data cone, save; replace.

(Refer Slide Time: 03: 17)



Now we have saved it on a particular folder and now what we are going to do is we are going to go and run the Matlab code that we have written for the problem.

(Refer Slide Time: 03: 29)

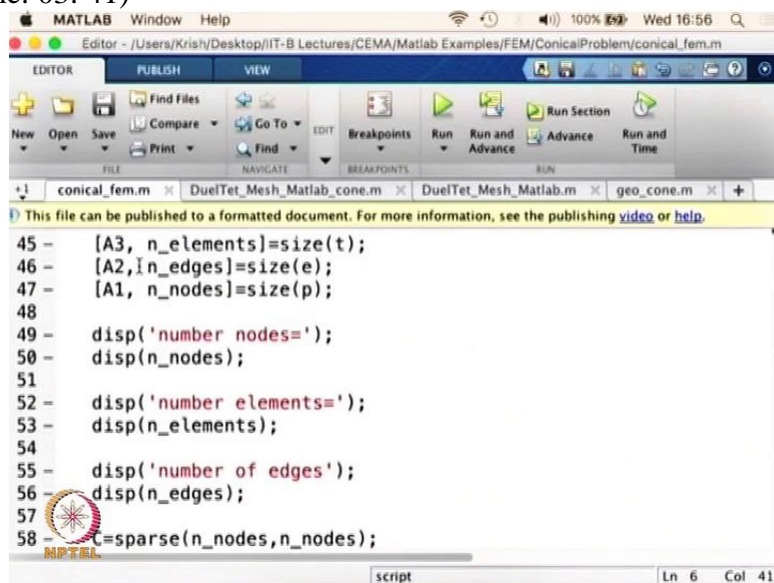


```
MATLAB Window Help
Editor - /Users/Krish/Desktop/IIT-B Lectures/CEMA/Matlab Examples/FEM/ConicalProblem/conical_fem.m

EDITOR PUBLISH VIEW
New Open Save Compare Go To Find Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE BREAKPOINTS RUN

+1 conical_fem.m x DuelTet_Mesh_Matlab_cone.m x DuelTet_Mesh_Matlab.m x geo_cone.m x +
This file can be published to a formatted document. For more information, see the publishing video or help.
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%% Matlab code solving conical capacitor problem using FEM
3 %%% unstructured mesh
4 %%% NPTEL Course: Computational Electromagnetics
5 %%% & Applications (CEMA)
6 %%% Chapter: Finite Element Method (FEM)
7 %%% Prof. Dr. K. Sankaran
8 %%% IIT Bombay, India &
9 %%% Founder-CEO, Prajnālaya, Zürich, Switzerland
10 %%% krish@sankaran.org
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12
13 (
14 Description: This FEM code uses a triangular mesh.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
script Ln 6 Col 41
```

It is a conical capacitor. And we are using FEM the unstructured mesh. And it is going to do the same thing what we saw in the earlier module, it is going to start with the PET matrix, (Refer Slide Time: 03: 41)



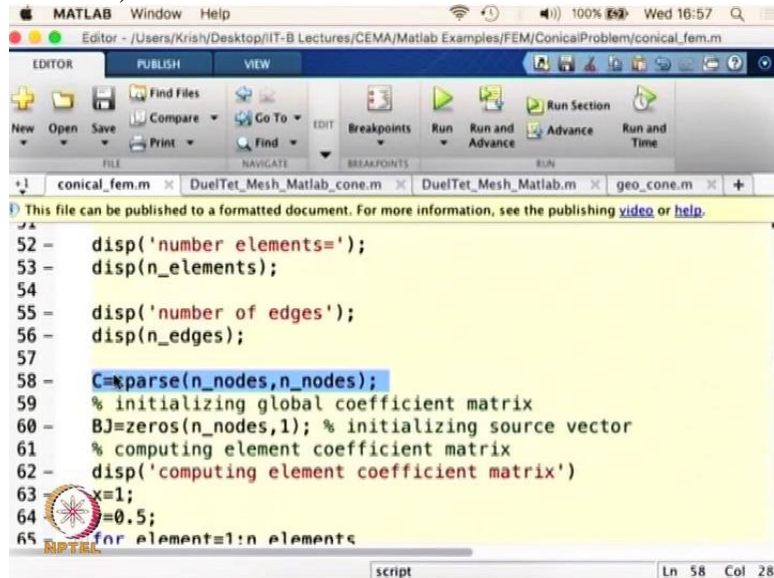
```
MATLAB Window Help
Editor - /Users/Krish/Desktop/IIT-B Lectures/CEMA/Matlab Examples/FEM/ConicalProblem/conical_fem.m

EDITOR PUBLISH VIEW
New Open Save Compare Go To Find Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE BREAKPOINTS RUN

+1 conical_fem.m x DuelTet_Mesh_Matlab_cone.m x DuelTet_Mesh_Matlab.m x geo_cone.m x +
This file can be published to a formatted document. For more information, see the publishing video or help.
45 - [A3, n_elements]=size(t);
46 - [A2, n_edges]=size(e);
47 - [A1, n_nodes]=size(p);
48
49 - disp('number nodes=');
50 - disp(n_nodes);
51
52 - disp('number elements=');
53 - disp(n_elements);
54
55 - disp('number of edges');
56 - disp(n_edges);
57
58 - C=sparse(n_nodes,n_nodes);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
script Ln 6 Col 41
```

And it is going to assign the elements, edges and nodes accordingly. It is going to display the number of elements, display the number of edges.

(Refer Slide Time: 03: 53)

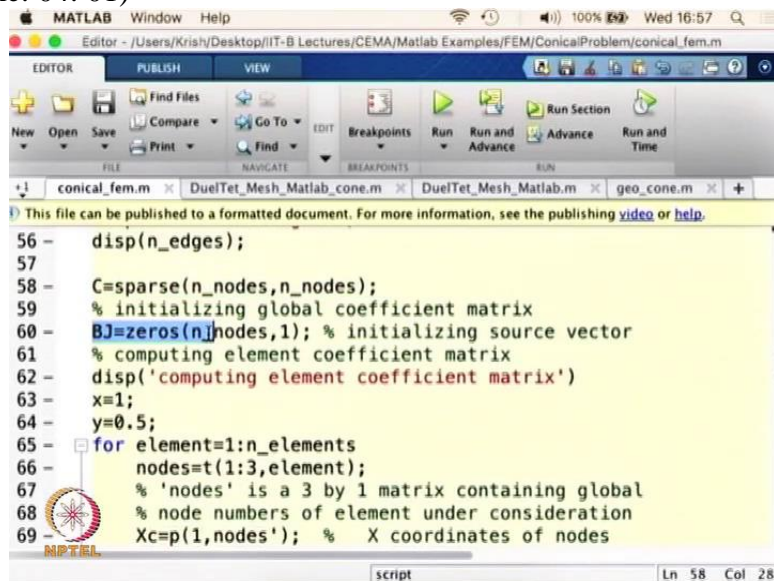


```
52 - disp('number elements=');
53 - disp(n_elements);
54
55 - disp('number of edges');
56 - disp(n_edges);
57
58 - C=sparse(n_nodes,n_nodes);
59 % initializing global coefficient matrix
60 - BJ=zeros(n_nodes,1); % initializing source vector
61 % computing element coefficient matrix
62 - disp('computing element coefficient matrix')
63 - x=1;
64 - y=0.5;
65 for element=1:n_elements
```

Initialize the c matrix as a sparse matrix; this is done in order to make the program faster.

This has been done to make the program faster.

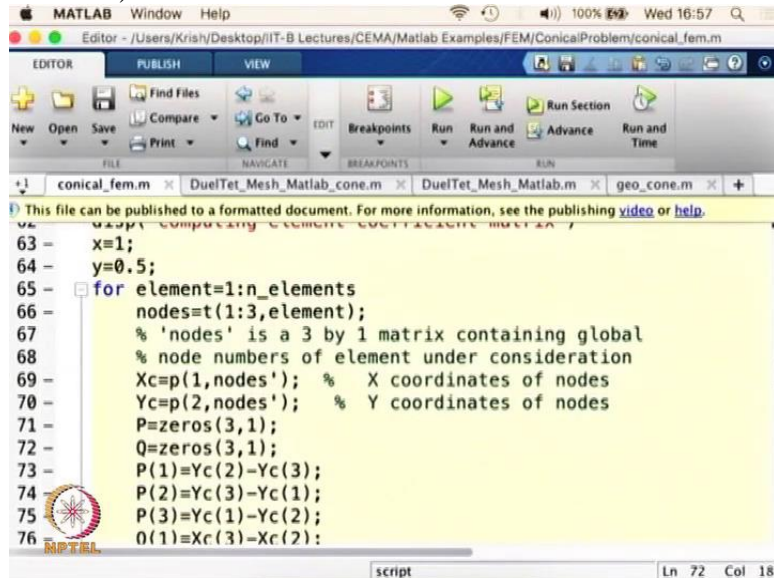
(Refer Slide Time: 04: 01)



```
56 - disp(n_edges);
57
58 - C=sparse(n_nodes,n_nodes);
59 % initializing global coefficient matrix
60 - BJ=zeros(n_nodes,1); % initializing source vector
61 % computing element coefficient matrix
62 - disp('computing element coefficient matrix')
63 - x=1;
64 - y=0.5;
65 for element=1:n_elements
66 - nodes=t(1:3,element);
67 % 'nodes' is a 3 by 1 matrix containing global
68 % node numbers of element under consideration
69 - Xc=p(1,nodes'); % X coordinates of nodes
```

And it is initializing the source vector

(Refer Slide Time: 04: 07)

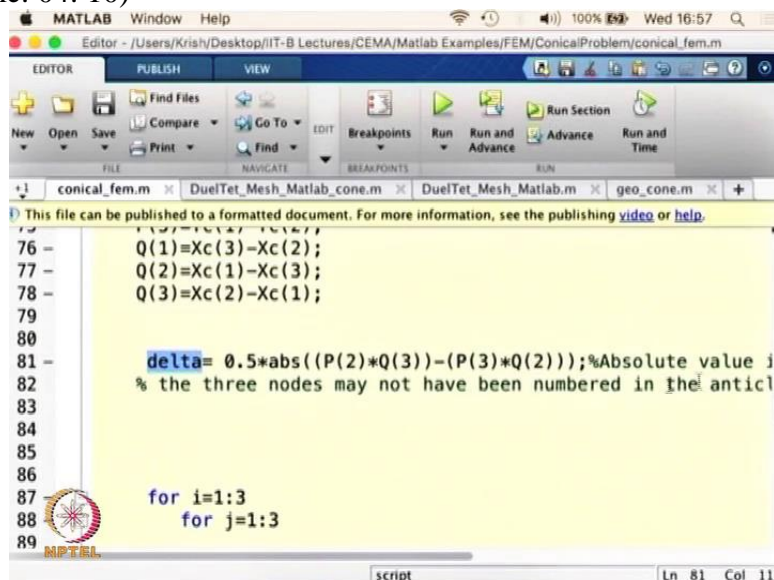


The image shows a MATLAB editor window with the following code:

```
63 - x=1;
64 - y=0.5;
65 - for element=1:n_elements
66 -     nodes=t(1:3,element);
67 -     % 'nodes' is a 3 by 1 matrix containing global
68 -     % node numbers of element under 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);
```

And it is going to go to compute the local coefficient matrix.

(Refer Slide Time: 04: 10)

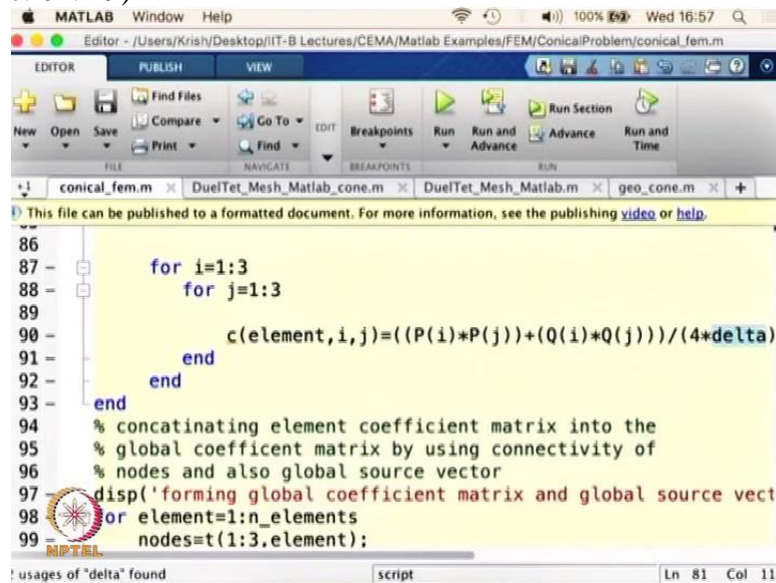


The image shows a MATLAB editor window with the following code:

```
76 -     Q(1)=Xc(3)-Xc(2);
77 -     Q(2)=Xc(1)-Xc(3);
78 -     Q(3)=Xc(2)-Xc(1);
79
80
81 -     delta= 0.5*abs((P(2)*Q(3))-(P(3)*Q(2)));%Absolute value of
82 -     % the three nodes may not have been numbered in the anticlockwise
83
84
85
86
87 -     for i=1:3
88 -         for j=1:3
89
```

It is computing the delta which is the area of each of the elements.

(Refer Slide Time: 04: 19)

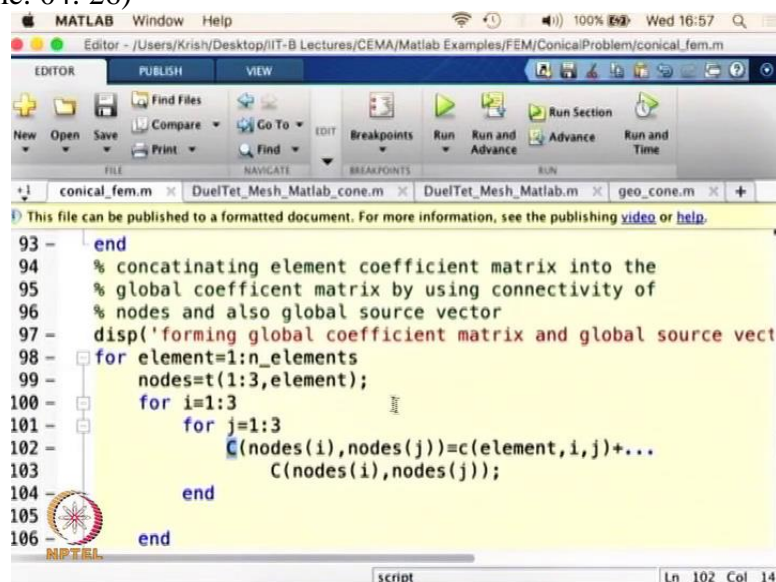


```
86 -
87 -     for i=1:3
88 -         for j=1:3
89 -             c(element,i,j)=((P(i)*P(j))+Q(i)*Q(j))/(4*delta)
90 -         end
91 -     end
92 - end
93 - end
94 - % concatenating element coefficient matrix into the
95 - % global coefficient matrix by using connectivity of
96 - % nodes and also global source vector
97 - disp('forming global coefficient matrix and global source vect
98 - for element=1:n_elements
99 -     nodes=t(1:3,element);
```

usages of "delta" found script Ln 81 Col 11

And it is substituting the value that has been computed for the local coefficient matrix.

(Refer Slide Time: 04: 26)

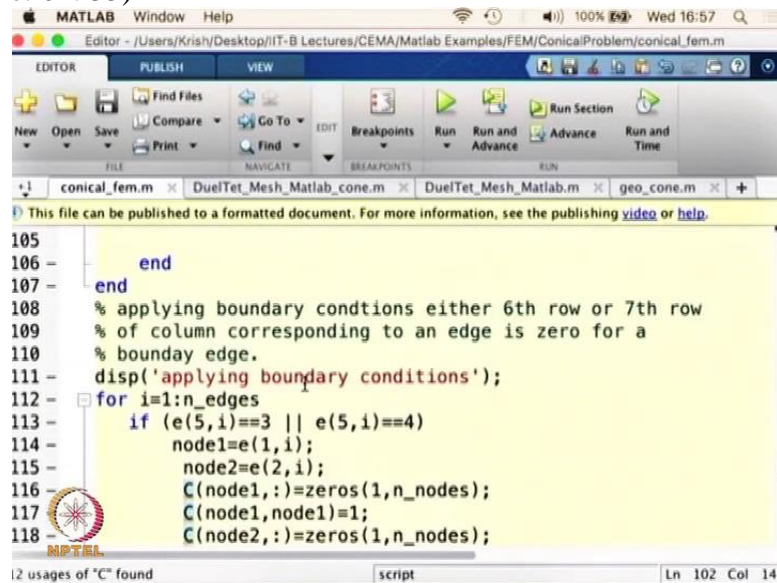


```
93 - end
94 - % concatenating element coefficient matrix into the
95 - % global coefficient matrix by using connectivity of
96 - % nodes and also global source vector
97 - disp('forming global coefficient matrix and global source vect
98 - for element=1:n_elements
99 -     nodes=t(1:3,element);
100 -     for i=1:3
101 -         for j=1:3
102 -             C(nodes(i),nodes(j))=c(element,i,j)+...
103 -                 C(nodes(i),nodes(j));
104 -         end
105 -     end
106 - end
```

script Ln 102 Col 14

And once it comes out of the loop it goes into concatenation process where it is going to join all the local coefficient matrix to create the capital c which is going to be the global coefficient matrix.

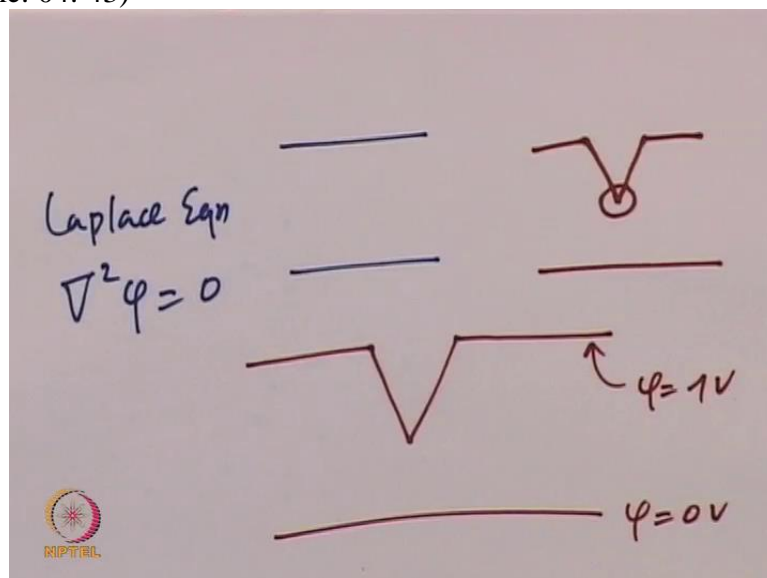
(Refer Slide Time: 04: 35)



```
105  
106 - end  
107 - end  
108 % applying boundary condntions either 6th row or 7th row  
109 % of column corresponding to an edge is zero for a  
110 % bounday edge.  
111 - disp('applying boundary condntions');  
112 - for i=1:n_edges  
113 -     if (e(5,i)==3 || e(5,i)==4)  
114 -         node1=e(1,i);  
115 -         node2=e(2,i);  
116 -         C(node1,:)=zeros(1,n_nodes);  
117 -         C(node1,node1)=1;  
118 -         C(node2,:)=zeros(1,n_nodes);
```

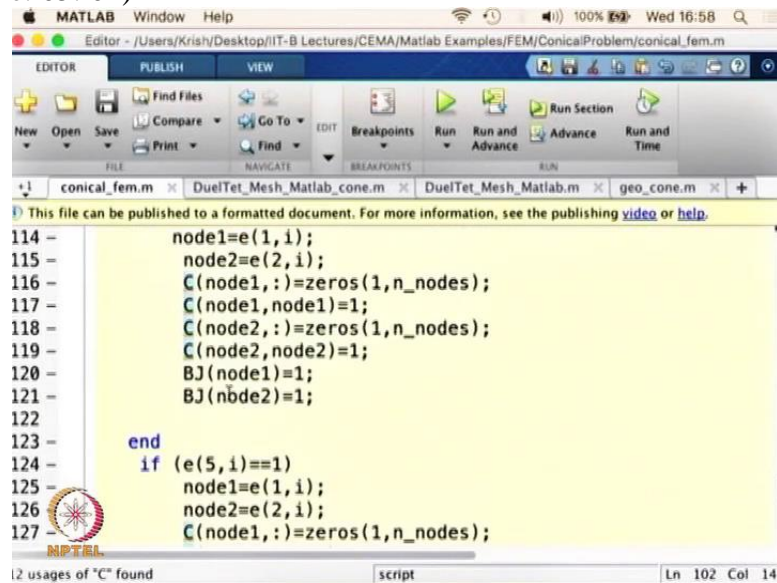
Once it is done it is going to apply the boundary conditions.

(Refer Slide Time: 04: 43)



We said that in our problem; let us go back to the sheet, so our problem is going to be this manner. We have set the top plate to be at Phi is equal to 1, the bottom plate to be at Phi equal to 0 volt. So these are the boundary conditions that we are setting.

(Refer Slide Time: 05: 02)



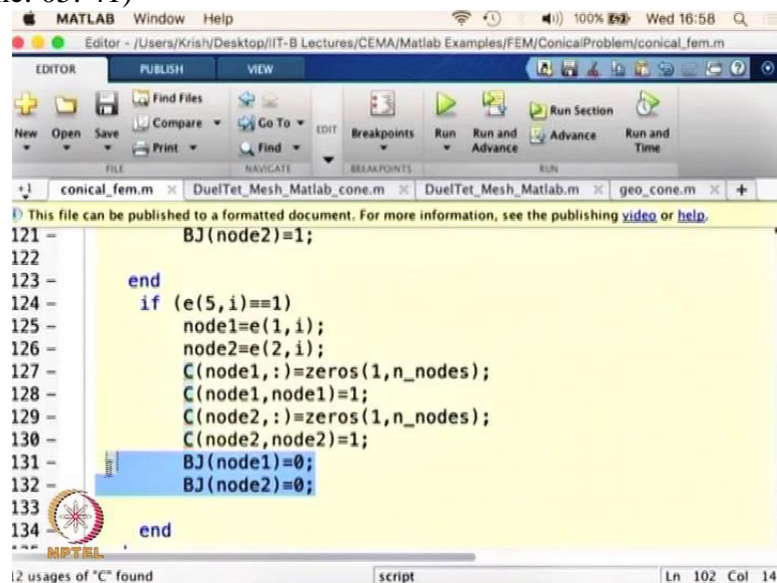
The image shows a MATLAB editor window with the following code:

```
114 -     node1=e(1,i);
115 -     node2=e(2,i);
116 -     C(node1,:)=zeros(1,n_nodes);
117 -     C(node1,node1)=1;
118 -     C(node2,:)=zeros(1,n_nodes);
119 -     C(node2,node2)=1;
120 -     BJ(node1)=1;
121 -     BJ(node2)=1;
122 -
123 -     end
124 -     if (e(5,i)==1)
125 -         node1=e(1,i);
126 -         node2=e(2,i);
127 -         C(node1,:)=zeros(1,n_nodes);
```

The status bar at the bottom indicates "12 usages of 'C' found" and "Ln 102 Col 14".

And this you can see also in the earlier part of the code where we have clearly said what is going to be our potentials. So this is going to be given as v1 and v2. So we are setting the top plate to be 1

(Refer Slide Time: 05: 41)



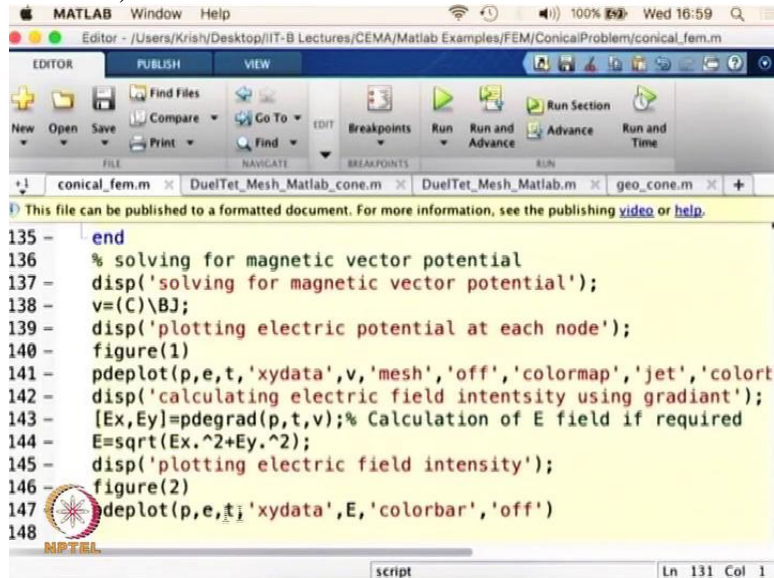
The image shows a MATLAB editor window with the following code:

```
121 -     BJ(node2)=1;
122 -
123 -     end
124 -     if (e(5,i)==1)
125 -         node1=e(1,i);
126 -         node2=e(2,i);
127 -         C(node1,:)=zeros(1,n_nodes);
128 -         C(node1,node1)=1;
129 -         C(node2,:)=zeros(1,n_nodes);
130 -         C(node2,node2)=1;
131 -         BJ(node1)=0;
132 -         BJ(node2)=0;
133 -
134 -     end
```

The status bar at the bottom indicates "12 usages of 'C' found" and "Ln 102 Col 14".

And the bottom plate to be 0. So these are the boundary values that we are giving.

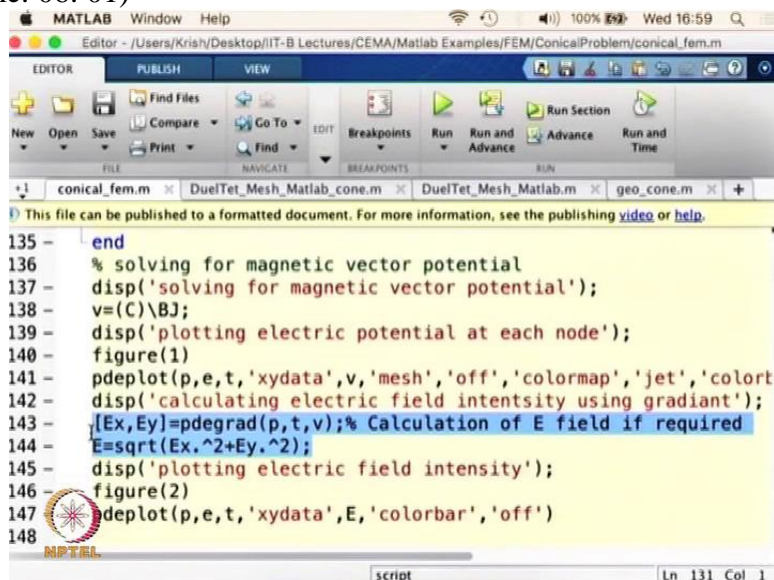
(Refer Slide Time: 05: 58)



```
135 - end
136 % solving for magnetic vector potential
137 - disp('solving for magnetic vector potential');
138 - v=(C)\BJ;
139 - disp('plotting electric potential at each node');
140 - figure(1)
141 - pdeplot(p,e,t,'xydata',v,'mesh','off','colormap','jet','color'
142 - disp('calculating electric field intensity using gradient');
143 - [Ex,Ey]=pdegrad(p,t,v);% Calculation of E field if required
144 - E=sqrt(Ex.^2+Ey.^2);
145 - disp('plotting electric field intensity');
146 - figure(2)
147 - pdeplot(p,e,t,'xydata',E,'colorbar','off')
148
```

And we are going to see the result solving for the magnetic vector potential and also plotting the electric potential at each node and we are going to see how the electric field intensity is being displayed and we are computing by calculation the gradient. And the value that we are getting is nothing but the plot for the electric field intensity. And we are plotting it as a function of PET.

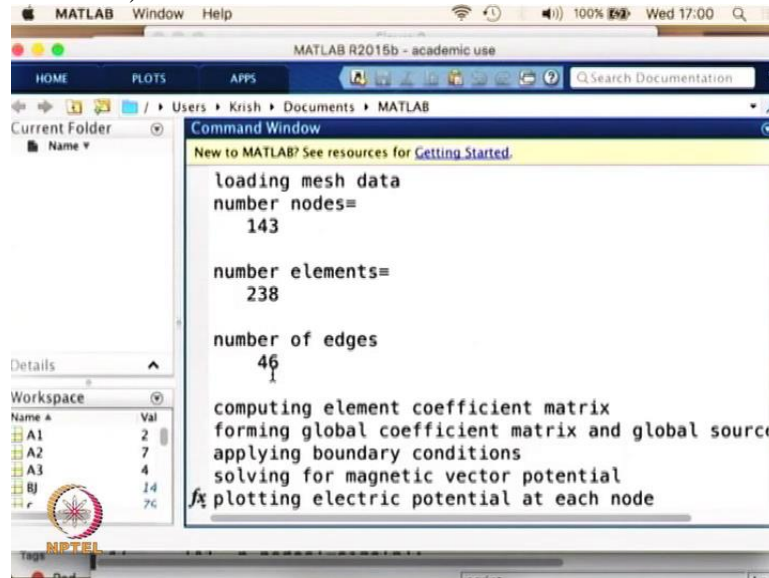
(Refer Slide Time: 06: 01)



```
135 - end
136 % solving for magnetic vector potential
137 - disp('solving for magnetic vector potential');
138 - v=(C)\BJ;
139 - disp('plotting electric potential at each node');
140 - figure(1)
141 - pdeplot(p,e,t,'xydata',v,'mesh','off','colormap','jet','color'
142 - disp('calculating electric field intensity using gradient');
143 - [Ex,Ey]=pdegrad(p,t,v);% Calculation of E field if required
144 - E=sqrt(Ex.^2+Ey.^2);
145 - disp('plotting electric field intensity');
146 - figure(2)
147 - pdeplot(p,e,t,'xydata',E,'colorbar','off')
148
```

And the value E is computed according to this equation.

(Refer Slide Time: 06: 05)



```
loading mesh data
number nodes=
    143

number elements=
    238

number of edges
    46

computing element coefficient matrix
forming global coefficient matrix and global source
applying boundary conditions
solving for magnetic vector potential
plotting electric potential at each node
```

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

```
loading mesh data
number nodes=
    143

number elements=
    238

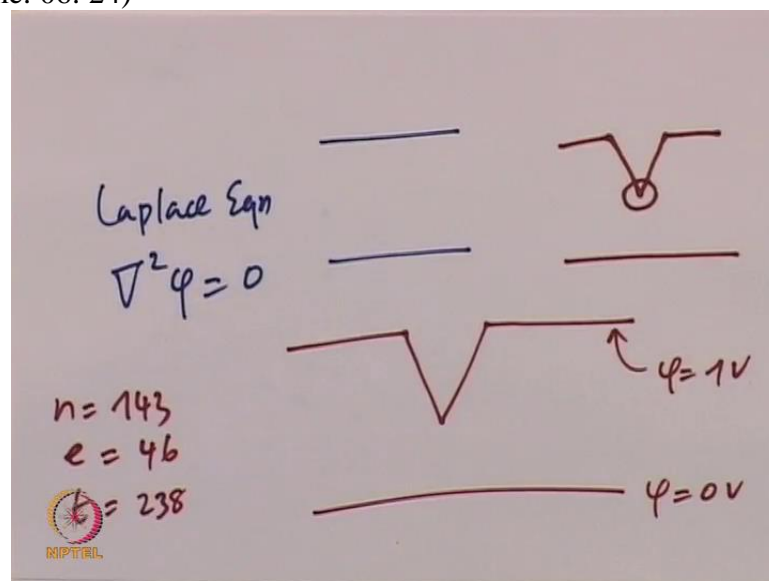
number of edges
    46

computing element coefficient matrix
forming global coefficient matrix and global source
applying boundary conditions
solving for magnetic vector potential
plotting electric potential at each node
```

The workspace on the left shows variables: A1 (2), A2 (7), A3 (4), BJ (14), and r (76).

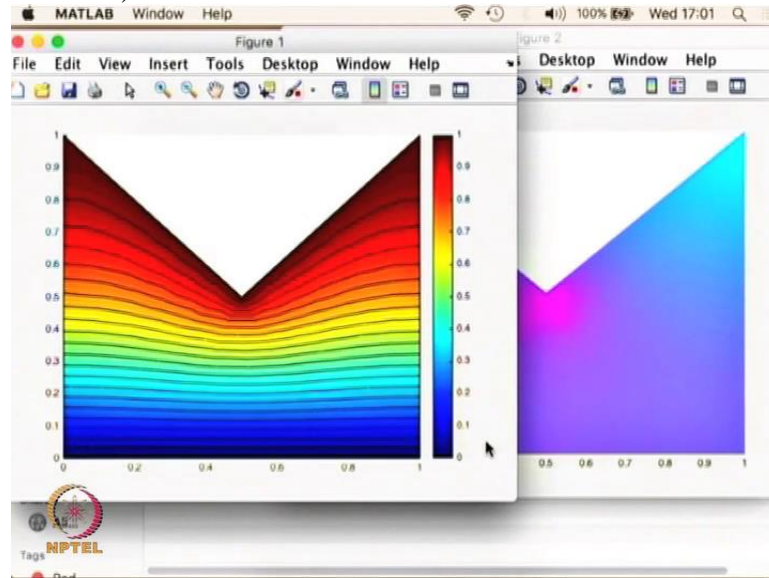
So we are going to run this program now to see the outcome of the program. So what we can first see is the number of edges nodes and the elements.

(Refer Slide Time: 06: 24)



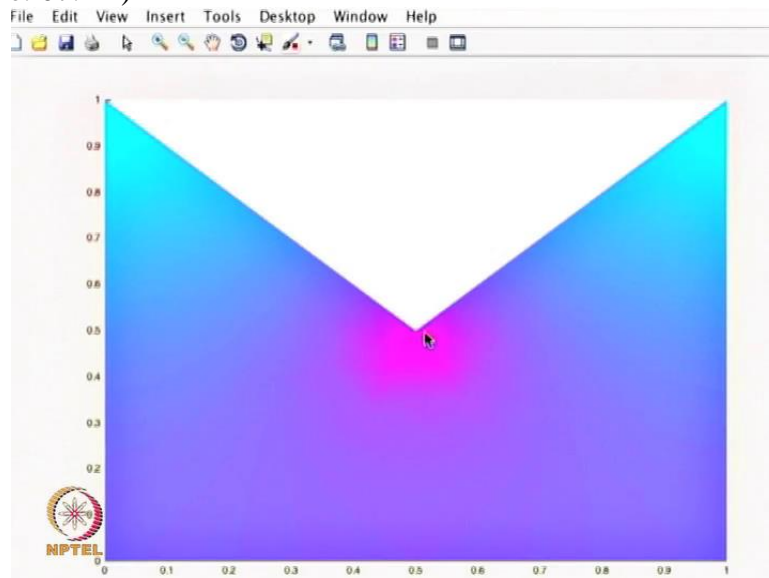
So what we have is for a coarse grid what we have chosen. The number of nodes which is written as n is going to be 143. And the number of edges e is going to be 46 and the number of elements which is a t which is going to be 238. So I have written this down so when we compare it for a next discretization we will see what is the number of nodes edges and triangles what we have got is two things solving the magnetic vector potential and plotting the electric potential at each of the nodes.

(Refer Slide Time: 07: 04)



What we see are two graphs, So let us look into the graphs or the plots what we have got, so the first one is the potential, what we see is the top plate the conical plate on the top is having 1 volt on the top. And then we see that the bottom plate has 0. And we see that the equipotential lines are no longer straight closer to the conical plate. It is going through a curve. Whereas the at the bottom plate it comes to a straight line.

(Refer Slide Time: 07: 41)

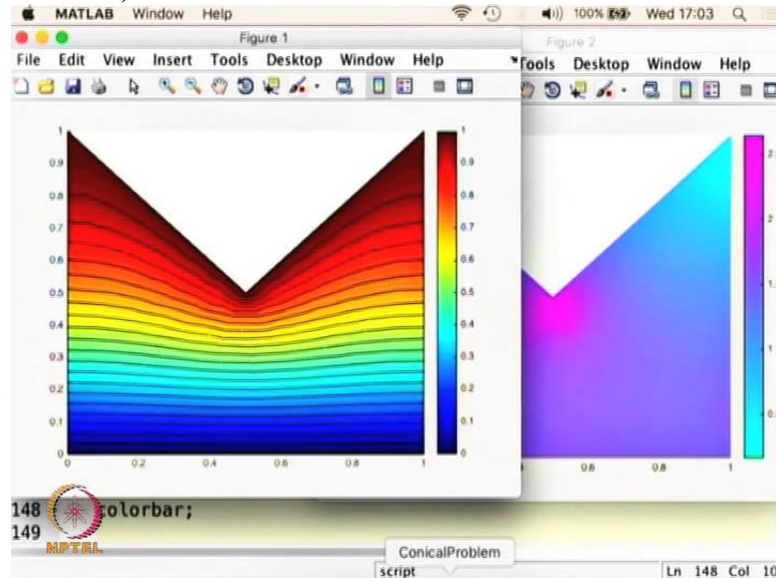


And what is also interesting to see is the electric field itself. What you see here as electric field, it shows certain high value at the tip of the cone, so we set the top plate and the bottom plate at the point of the tip of the cone is going to be between 0.5 and 0. So the distance between the top plate the conical point of this plate and the bottom plate is roughly 0.5. And we said that the top plate is at a potential 1 and the bottom plate is at a potential 0. So the

electric field at this point should be approximately 1 divided by the distance between the plate which is 0.5, which should be 2, which should be the actual value.

So now with that in mind let us see the actual scale of this thing. So let us put the scale in place so when I run the program. So I am going to run the program for the coarse discretization.

(Refer Slide Time: 08: 57)



What we have got and what we see is two plots and we also see certain parameters that are important for us to note.

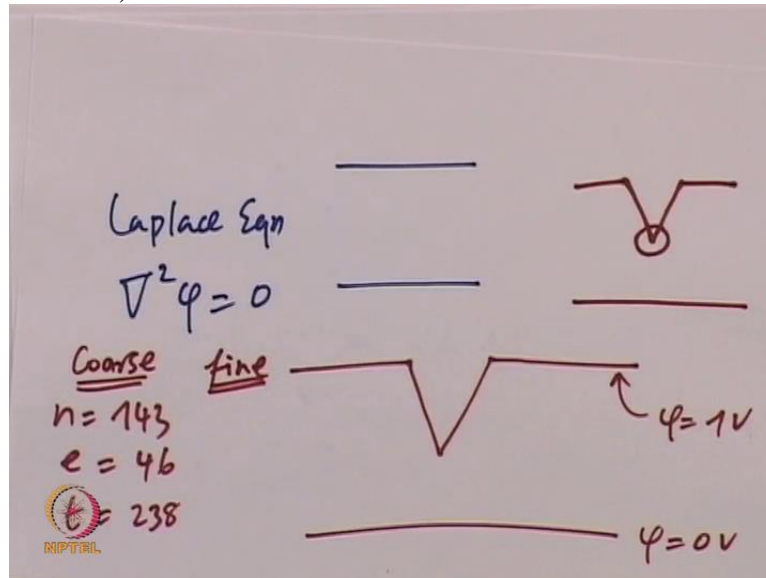
(Refer Slide Time: 09: 05)

```
loading mesh data
number nodes=
    143
number elements=
    238
number of edges
    46
computing element coefficient matrix
forming global coefficient matrix and global source
applying boundary conditions
solving for magnetic vector potential
plotting electric potential at each node
```

Name	Val
A1	2
A2	7
A3	4
BJ	14
r	7C

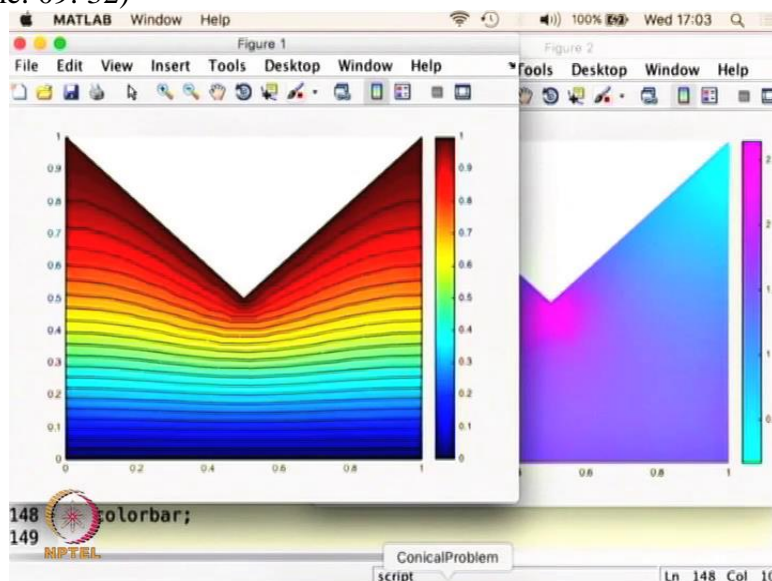
So what we get is the number of nodes is 143, the number of element is 238 and the number of edges is 46. That is what I have written here in the paper.

(Refer Slide Time: 09: 15)



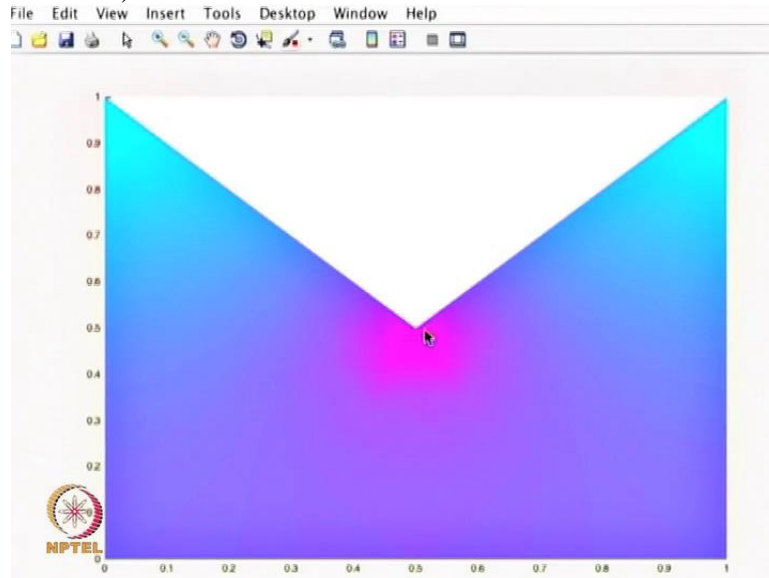
So this has been done for the coarse grid and we will do a fine grid example where we will see the number of nodes, edges and triangles are going to change.

(Refer Slide Time: 09: 32)



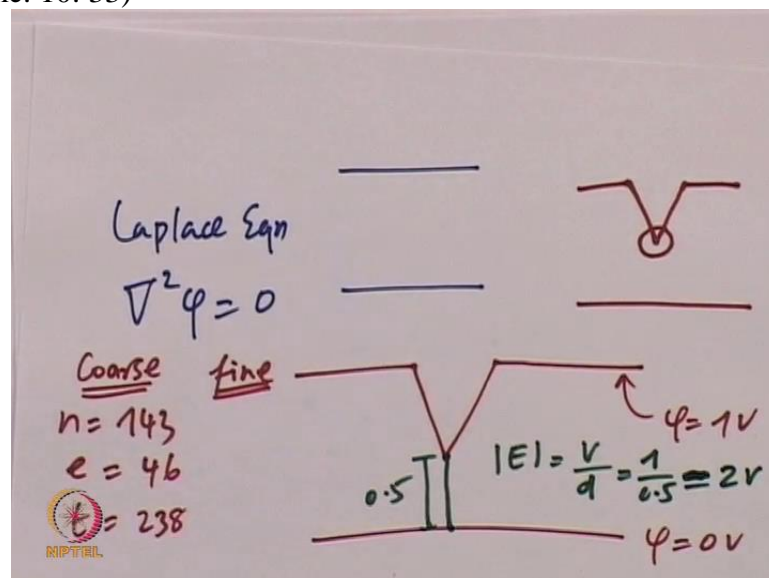
But let us see the result itself and see what is the result telling us. So you that the potentials on the top plate is 1 and the bottom plate is 0. The equipotential lines towards the conical plate is curved as we expect while it changes to a horizontal line as it comes to the bottom plate. And we also see that the curving is gradually reducing the equipotential lines curving is gradually reducing as if it comes down and it also changes the curvature changes as we go away from the conical point towards the edges either edges. This is an important point which replicates the physical meshable result.

(Refer Slide Time: 10: 24)



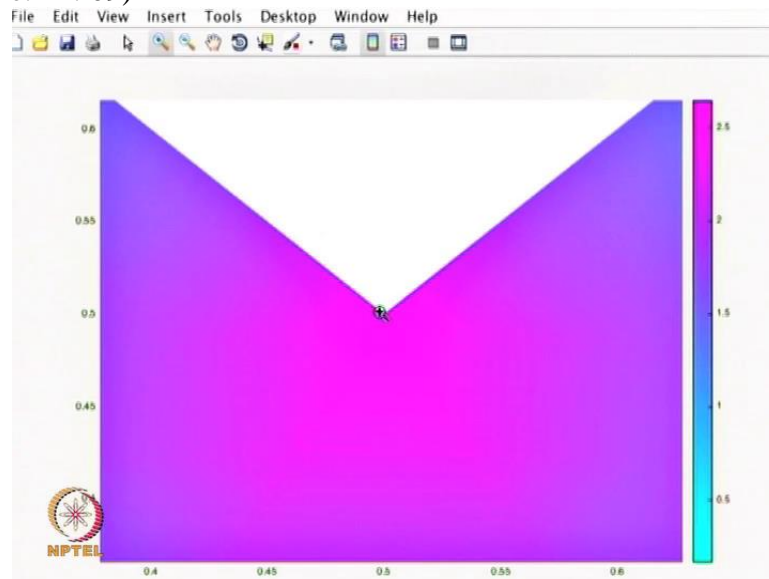
And let us look into electric field itself.

(Refer Slide Time: 10: 33)



The point that one has to know here is the potential at this point here is 1 volt, the potential here is going to be 0 volt. The distance between these two points is 0.5. So we know the electric field magnitude should be given by the voltage divided by the distance roughly. So the voltage difference is 1 and the distance is 0.5, so we should have the value in the range of 2 volt.

(Refer Slide Time: 11: 09)



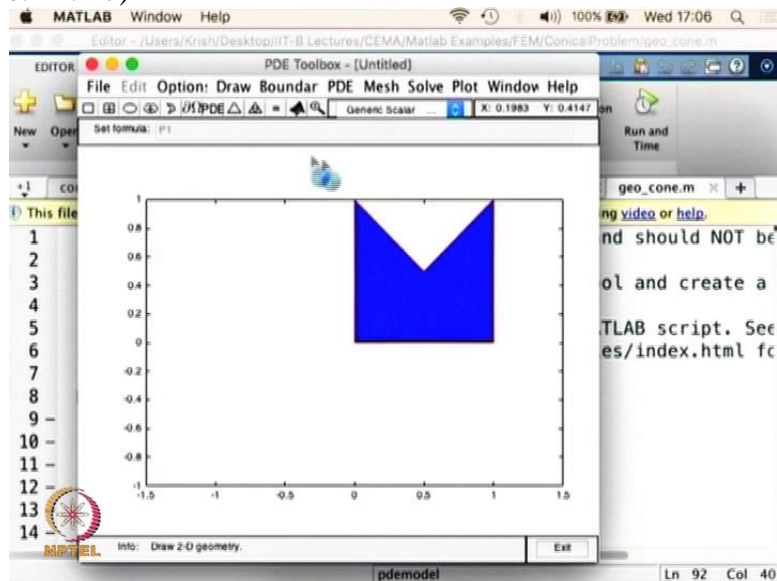
Let us look at the result itself; what you see is it is definitely more than 2 volts. It is in the range of 2.5 volt or may be even more. So that is the accuracy that one sees when we have very sharp discontinuities.

(Refer Slide Time: 11: 34)

```
136 % solving for magnetic vector potential
137 - disp('solving for magnetic vector potential');
138 - v=(C)\BJ;
139 - disp('plotting electric potential at each node');
140 - figure(1)
141 - pdeplot(p,e,t,'xydata',v,'mesh','off','colormap','jet','colort
142 - disp('calculating electric field intensity using gradient');
143 - [Ex,Ey]=pdegrad(p,t,v);% Calculation of E field if required
144 - E=sqrt(Ex.^2+Ey.^2);
145 - disp('plotting electric field intensity');
146 - figure(2)
147 - pdeplot(p,e,t,'xydata',E,'colorbar','off')
148 - colorbar;
149
```

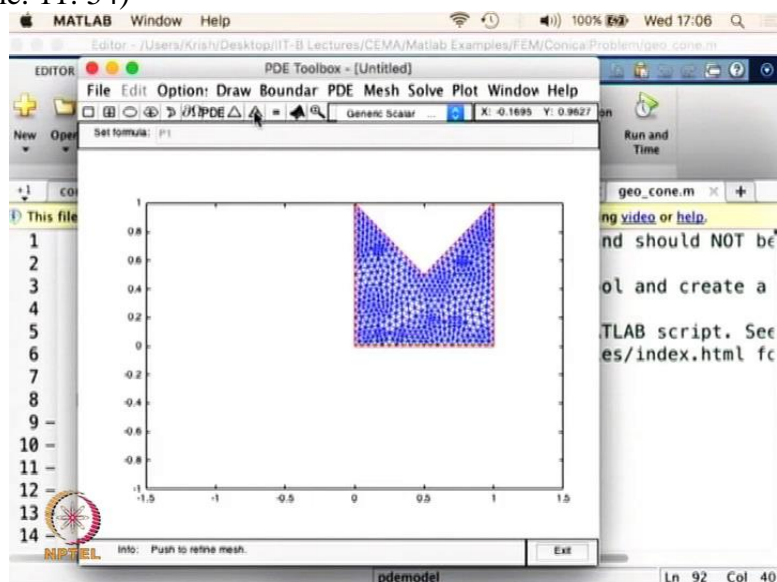
We should be in the in this range colour range whereas we see that it is definitely above 2.5 volt. With that in mind let us go now an do the same problem for a finer grid and see how the result is going to improve.

(Refer Slide Time: 11: 47)



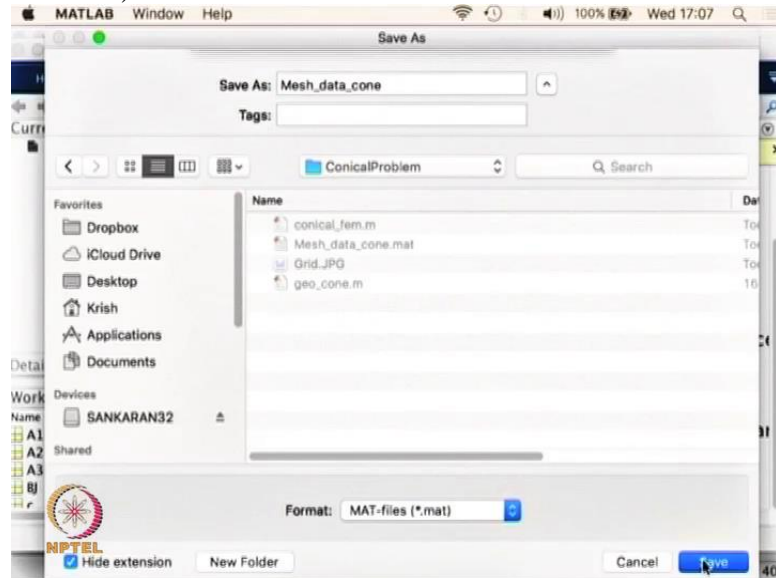
So for that I have to go back to the PDE tool box and I am going to run it again.

(Refer Slide Time: 11: 54)



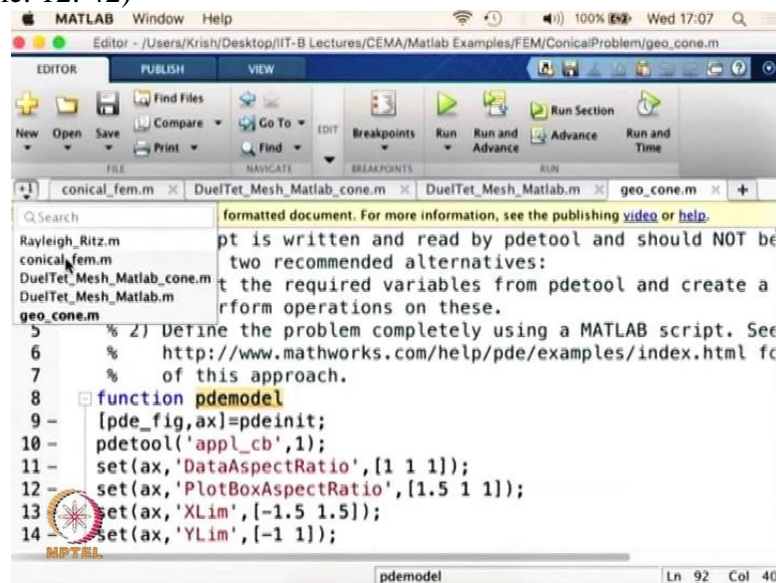
So we are refining the grid. Remember initially we had a grid of this coarse discretization, now we are going to refine it refine it refine it and now it is very refined.

(Refer Slide Time: 12: 30)



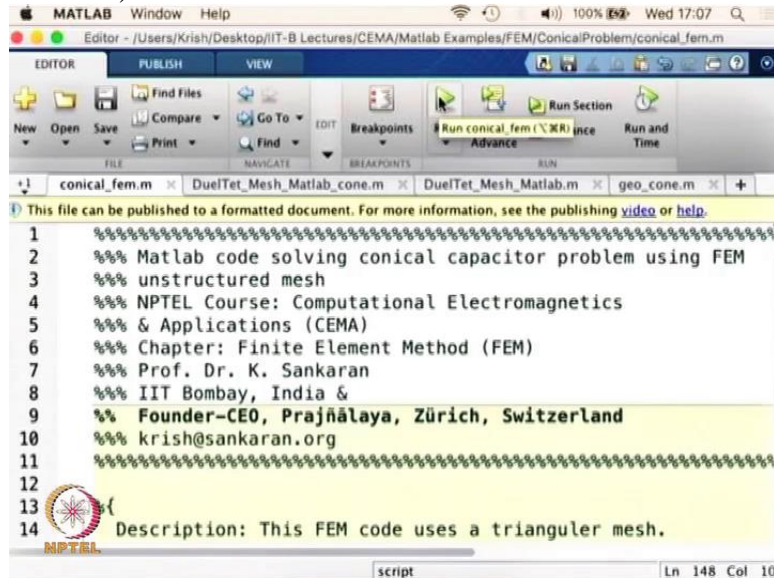
And we are going to mesh export, say ok, we exit. Now and then we go to the home, we go to the workspace and we save the workspace conical problem, mesh data cone and we are going to save it. Going to replace it.

(Refer Slide Time: 12: 42)



And we are going to go back to the finite element code. And run it conical FEM

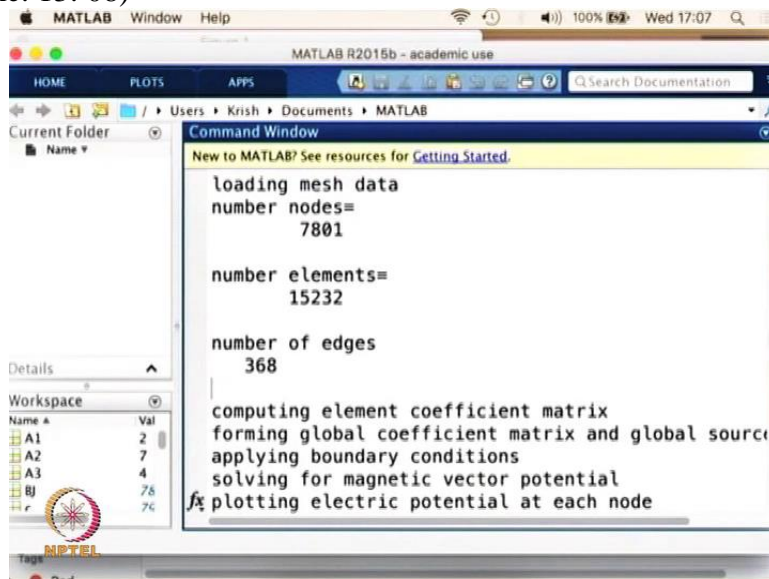
(Refer Slide Time: 12: 52)



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

Now we are going to run the program and we will start noticing that it is going to take a lot time than before, slightly longer not that much longer but slightly longer.

(Refer Slide Time: 13: 08)



```
loading mesh data
number nodes=
    7801

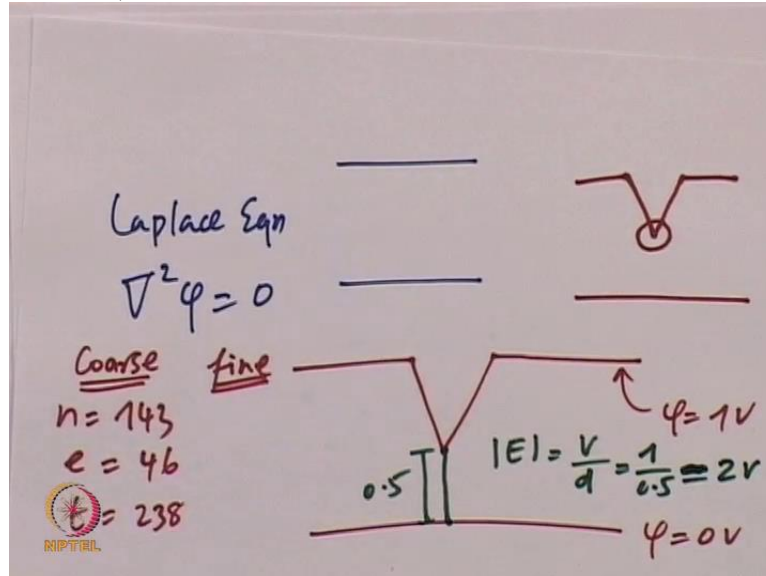
number elements=
    15232

number of edges
    368

computing element coefficient matrix
forming global coefficient matrix and global source
applying boundary conditions
solving for magnetic vector potential
plotting electric potential at each node
```

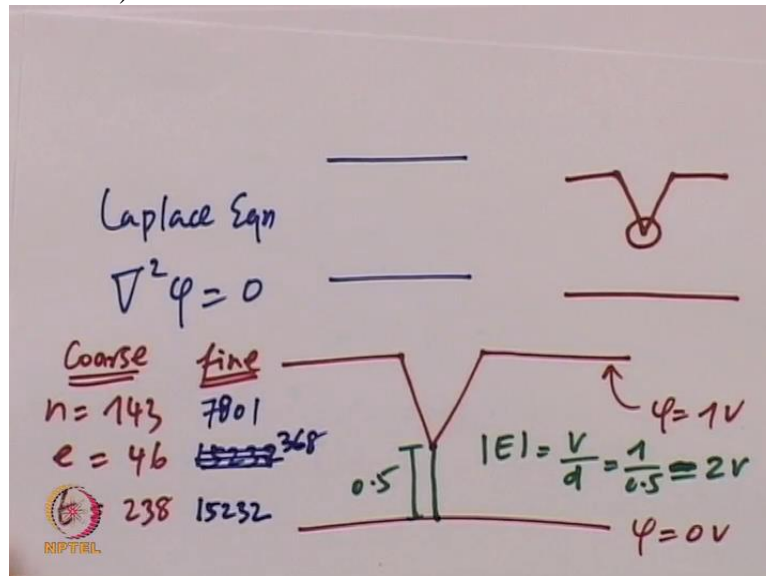
What we can see is the number of edges, number of points have changed dramatically.

(Refer Slide Time: 13: 13)



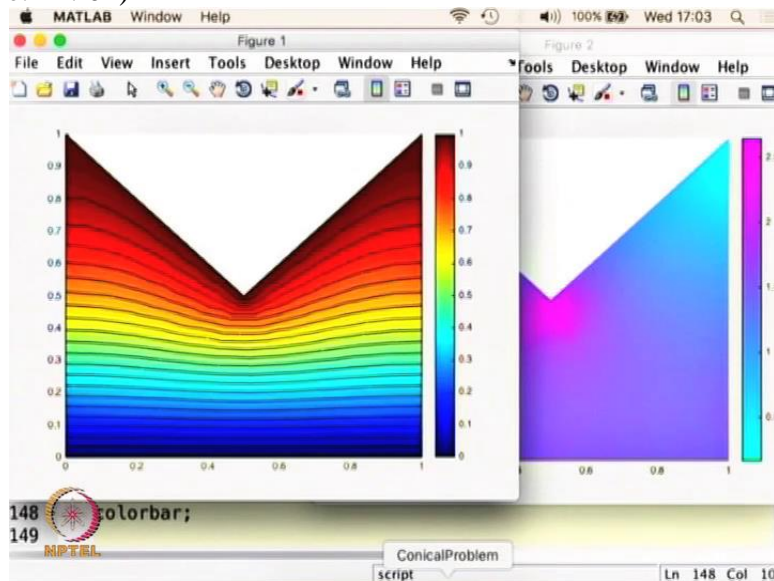
So let us go back to the paper and see that in the first example what we had we are 143 number of points nodes edges were 46 and the triangles were 238.

(Refer Slide Time: 13: 27)



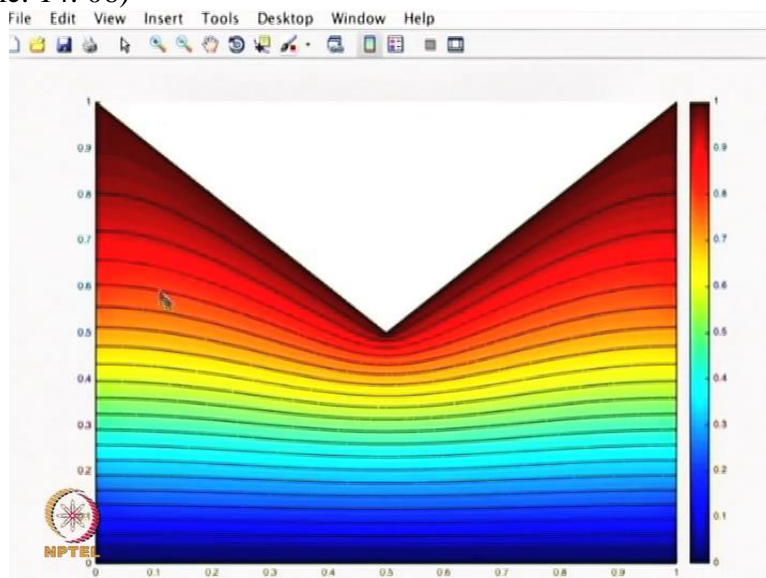
So let us look at this problem here you see the number of nodes have been increased to 7801, the number of triangles are going to be 15232 whereas the number of edges are going to be 368. So what we see is there is going to be a substantial increase in the number of elements, number of edges and number of nodes.

(Refer Slide Time: 14: 01)



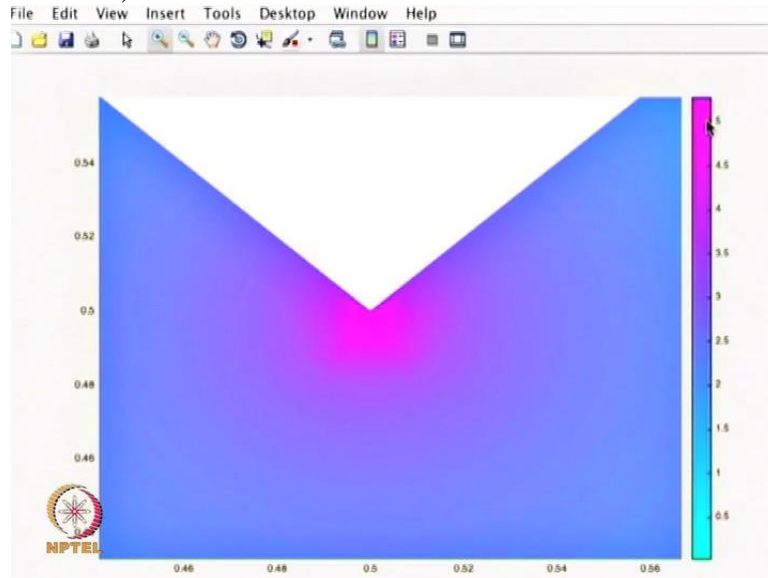
And we also see that how this is going to affect the value of our result.

(Refer Slide Time: 14: 06)



If you see the potential graph itself it does not change much, it is pretty much the same way we saw before, the equipotential lines are pretty much the same, we see that towards the lower plate it is changing to be horizontal where as when it goes towards the conical plate it sees a curve. And the potentials on the top plate and the bottom plate are exactly what we have given; the result is not categorically different. So in this case we do not win much.

(Refer Slide Time: 14: 37)



Whereas if you see the way the electric field is plotted what we also see is overall we get very good result whereas what we see in the tip is the singularity is very very sharply reflected, what you see is a value that is more than 5 in this case. In the coarse grid what we got is roughly 2.25, whereas in the fine grid we are getting a value that is more than 5 which is ridiculously wrong. So what we wanted to show here is in some of the problems you gain something by refining, whereas in some of the problems the gain is not there pretty not there. There is going to be a lot of other things happening, this is also something we saw in the case of finite difference method, by refining the grid the error is going to increase because of various other reasons. For example the number of points that we are going to use in the computer is going to increase dramatically and because of that the truncation error or the numerical accuracy is going to change accordingly.

So remember the curve we had in the case refining will lead to reduction in error because of one reason. Whereas it is going to increase the error for a different reason. And that is what we have also showcased in this case, although we have refined globally the solution is in a good way it is replicated in a good way. The electric field in this point, in this point is nice. But whereas at sharp edges the error is exaggerated much beyond our expectation you see that it is going to a very very high value.

So what will happen is if we go even finer the discretization will create a very very high value which goes contra intuitive. This is the problem that we wanted to showcase where we actually win in terms of using the discretization or increasing the discretization making the grid more finer or making the grid more coarser. In some cases we win in some case we do not win, and this is something is contra intuitive people do not really know this until they

simulate it and they think the program is wrong or the code is running wrong. No it is everything is ok the only thing is the nature of the numerical method is like that and particularly in the FEM case this is very much like this case. So one has to get a physical sense of what will be a end of discretization, what is good enough and how we can model it, depends on the experience that you gain by running such program.

So what I would encourage is I would like you to look into this program model it for yourself practice it a bit and get a good sense of how the program can be used for your type of problems. So whether it could be a simple parallel plate problem or it could be a conical plate problem or any sharp discontinuities that you might encounter in real like applications.

So with that being said we have covered enough in the last two modules using Finite element method and we have showcased the various pros and cons of numerical method of finite element and also we have seen the pros and cons of using finite grid or a coarser grid. We will use the same problem and we will approach the same problem using a different method while we come and do the module of algebraic topology later on but it is important for you to know this problem will be a test problem for you to see whether the refinement is making sense or not.

With that being said we will stop here and we will come back and do some more problems in finite element at the later modules Thank you!