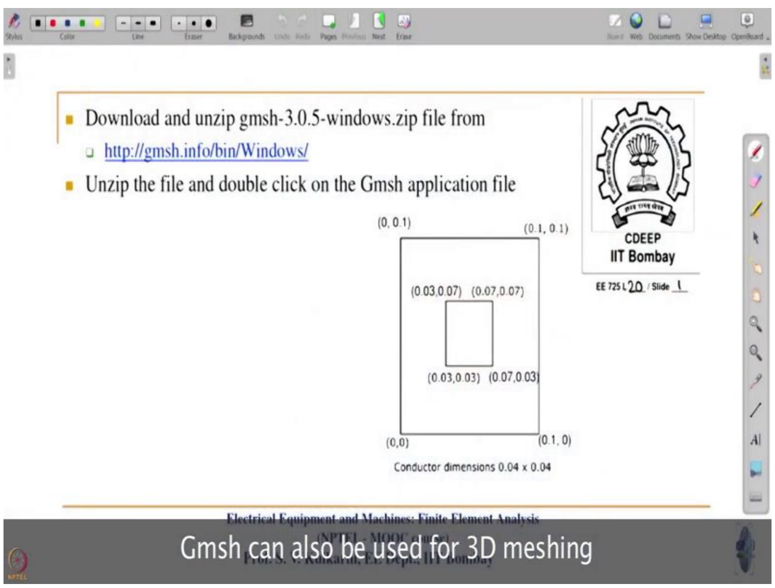
Electrical Equipment and Machines: Finite Element Analysis
Professor Shrikrishna V. Kulkarni
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Lecture 20
2D FEM Code: Gmsh and Scilab

Welcome to lecture 20. In the previous lecture, we have seen a 2D FEM code which was developed to solve a 2 dimensional magnetostatic problem, in that code we have used the mesh that was generated manually. But that kind of mesh is applicable for rectangular geometries where the discretization can be uniform.

But most of the geometries of electrical machines have circular boundaries. It will be difficult to mesh such kind of geometries using the algorithm that we saw in the previous lecture. So, for such kind of geometries we have to use some meshing software which are available online. Gmsh is one such kind of freeware which is used to mesh a 2D geometry of any complicated shape .

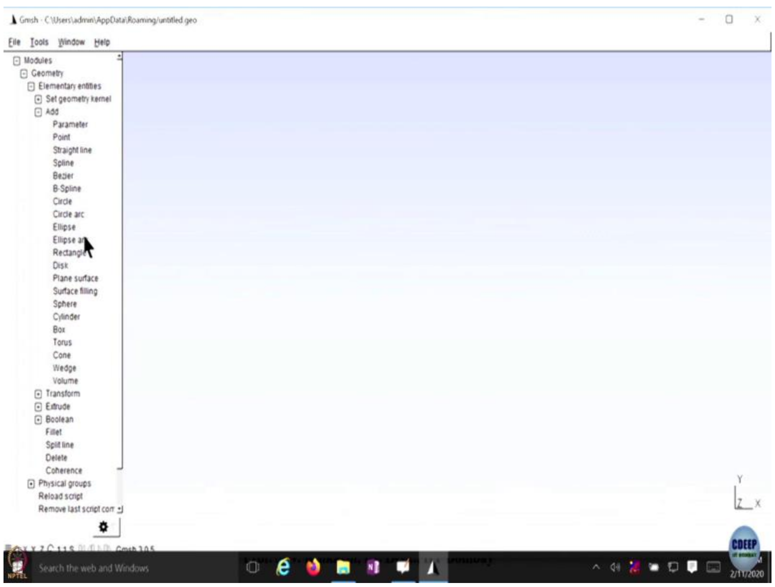
(Refer Slide Time: 1:12)



You can download the Gmsh software using the link (http://gmsh.info/bin/Windows/) which is given in the above slide. Also, preferably download version 3.0.5-windows of the software. Because the mesh data that comes out of this version of the software is easier to understand. In this lecture, we are going to solve the 2D geometry which we saw in the previous lecture using the mesh generated by using Gmsh software. The geometry we saw in the previous lecture is given in the above slide.

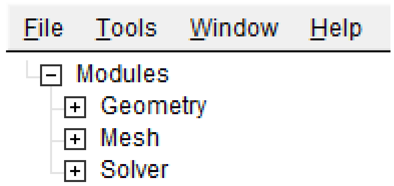
In this lecture, first we will see the interface of the Gmsh software and then we will draw the geometry in it. Then we will mesh the geometry and extract the mesh data in terms of p, e, t matrices and we use the generated p, e, t matrices in the code that we saw in the previous lecture with few modifications. Here, we are going to focus only on the modifications of the code that we saw in the previous lecture.

(Refer Slide Time: 2:48)

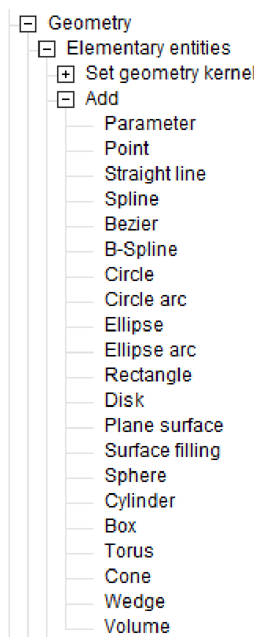


First we will go to the Gmsh software, you can directly download it from the link which is given in the previous slide. It will be downloaded in the form of a zip file. Extract that zip file and open the extracted folder. There is no need of installation for this software. The interface shown in the above figure can be opened directly by double clicking on the application icon in the folder.

On the left hand side of the Gmsh interface, you can see three different options which are shown in the following figure.

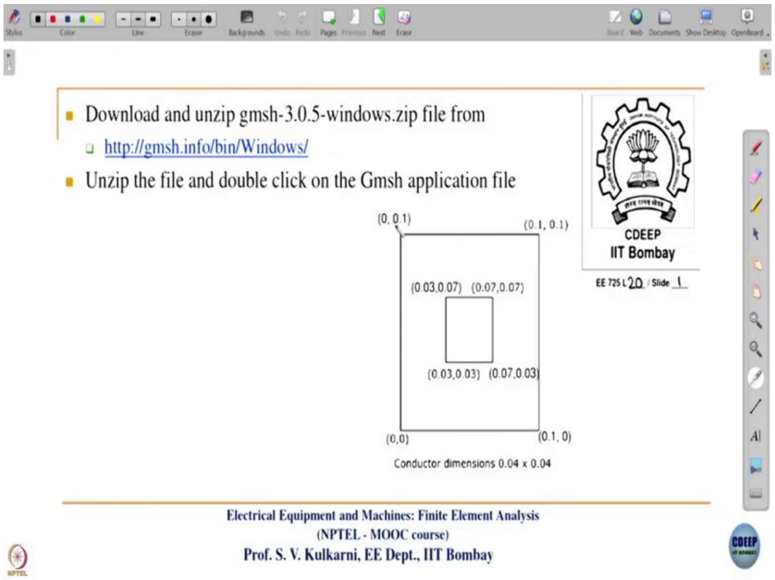


The first is geometry, under this tree you have different entities which are given in the following figure to draw a geometry.

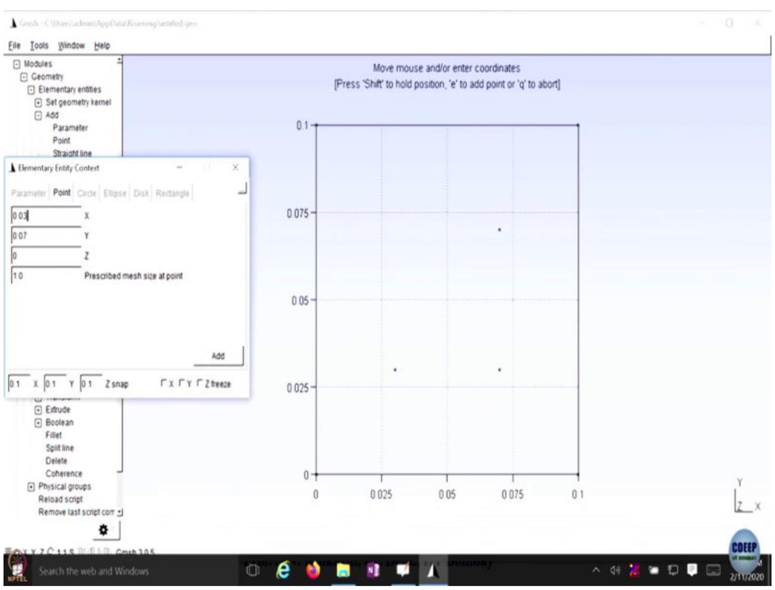


According to these entities, we can draw a geometry by using circles, rectangles, circular arcs and so on. But in this lecture, we will draw the geometry by using points and straight lines. Because geometries of most of the electrical machines are basically made of circular arcs and they are not made of complete circles and rectangles. For example, the shape of a slot is not a complete circle and it is an arc. So, it is better to a draw geometry using nodes and join them by using straight lines. That is why in this lecture we are going to do the same thing.

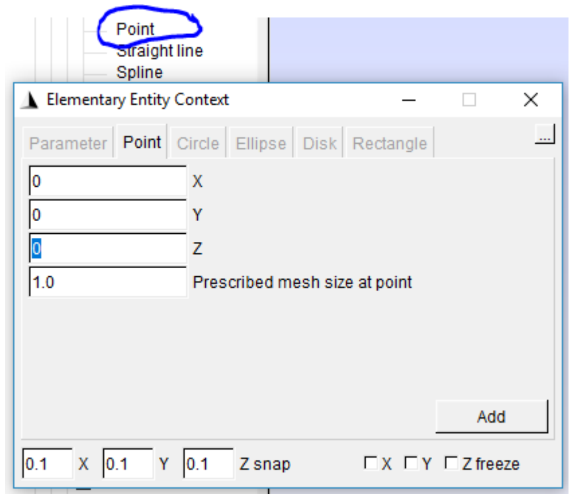
(Refer Slide Time: 4:00)



In the above slide, you can see the coordinates of the eight nodes of the two rectangles in our problem geometry.



First we are going to place these eight nodes of the geometry. For that, first you click on point, which opens the pop up which is as shown in the following figure. Once the pop up window comes out, you can enter all the coordinates.



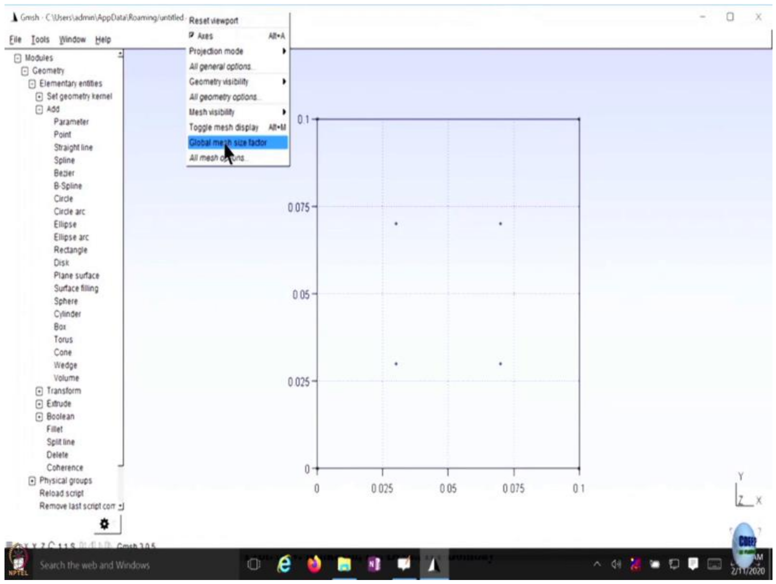
Now enter the first coordinate (0, 0) and length of the outer boundary is 0.1, so we are changing the x coordinate in the above pop up window as 0.1. The height of the boundary is 0.1, we are changing the y coordinate as 0.1. And the fourth node of the rectangle will be (0, 0.1). We have formed the nodes of the outer rectangle.

Now, we will form the nodes of the inner rectangle. Place the first node at (0.03, 0.03) and the length of the conductor is 40 mm, so you have to increase the x coordinate by 0.04 and the height is 0.04, so you have to increase y coordinate to 0.07 and the fourth node of the rectangle is (0.03, 0.07). Now you have placed all the nodes of the two rectangles. Now, you can complete the geometry by joining these nodes using straight lines, because the edges of a rectangle are straight lines. So, here we want to highlight one thing about the mesh generated using Gmsh. As we have discussed in the previous lecture, we make use of the mesh information generated from Gmsh to apply the boundary conditions.

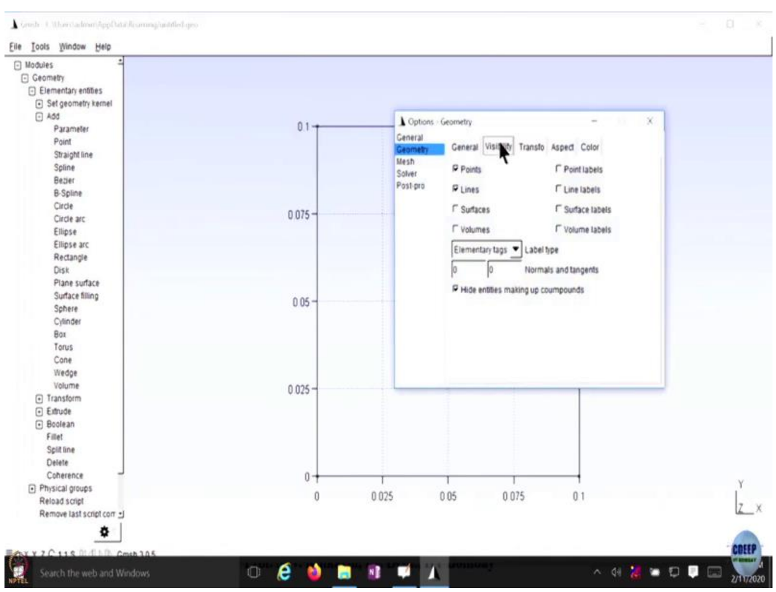
For that, we need to know something called as global edge number. So, we need to know the edge label to apply the boundary condition. In Gmsh, the edge numbers will be assigned in the same sequence as you draw the edges.

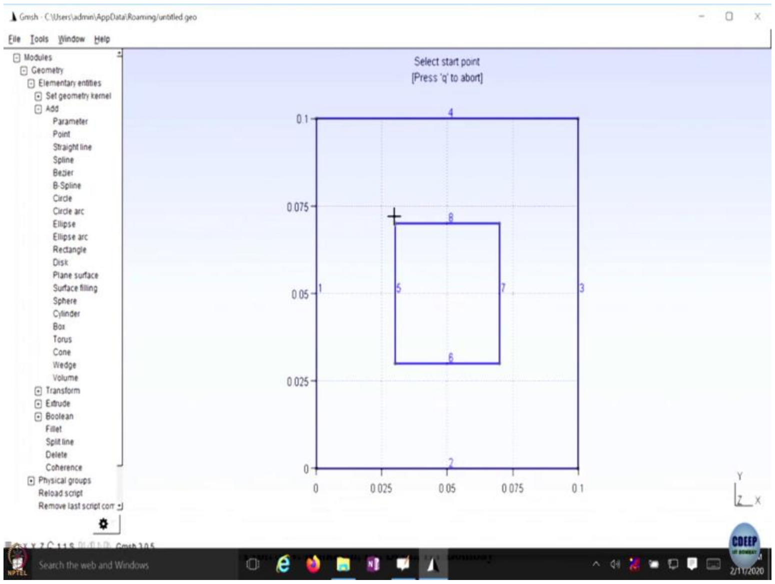
Similarly, sub-domain numbers are also assigned in the sequence as you define the sub-domains. So we can explore these features in the present code.

(Refer Slide Time: 7:05)



To see the global edge numbers and sub domain numbers, you have to right click on the space where you have drawn the geometry and go to ‘All Geometry Options’ this will pop up a window as shown in the following slide. In that window you click on the ‘Visibility’ tab and check the ‘Line Labels’, ‘Surface Label’.



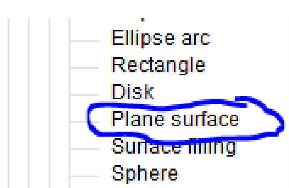


Now you draw the straight lines as shown in the above slide. If I first draw the vertical line which is joining (0,0) and (0,0.01) then you can see 1 is assigned for this edge. Similarly, if I draw an edge joining nodes at (0,0) and (0.1,0) then, 2 will be assigned to this edge.

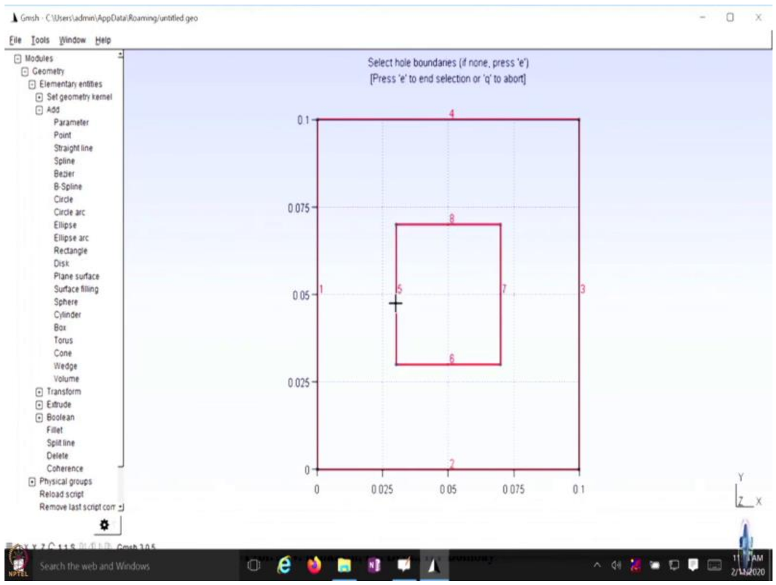
Similarly, 3, 4, 5,…,8 will be assigned to the other edges. To avoid confusion about the edge numbers, while drawing the geometry you first draw the outer boundary where in most of the cases we apply A = 0 so that numbers 1, 2, 3, 4 will be assigned to theses edges. And the part of the code where you assign the boundary conditions will remain the same for any geometry.

Now, after this, we need to assign the sub-domain number. In the present problem, there are two sub-domains, the first sub-domain is the region between the outer rectangle and the inner rectangle. So, the edges of the outer rectangle and the inner rectangle form the boundary of the air sub-domain.

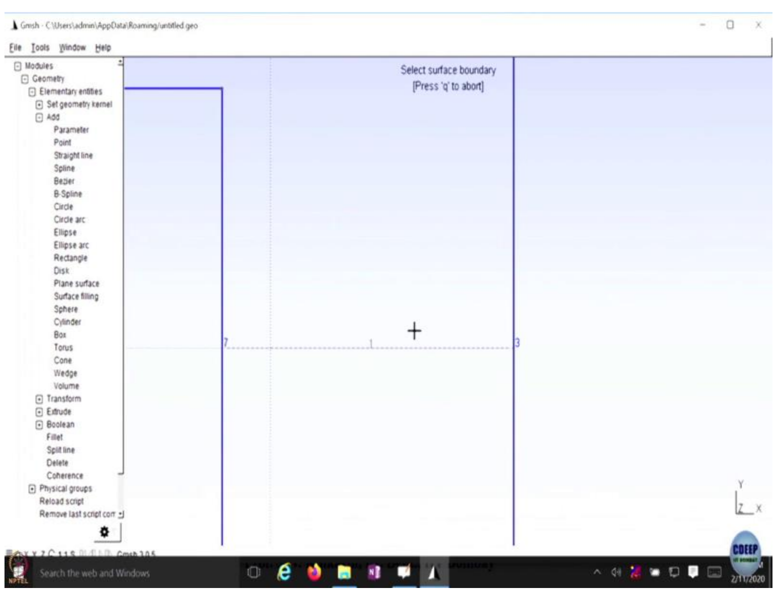
To define the sub domain, you click on the plane surface which is below and select the boundaries of the region.



The two selected boundaries of the first sub-domain are shown in the following slide. So whatever may be the sub-domain, you need to choose by clicking all the boundaries of that sub-domain.

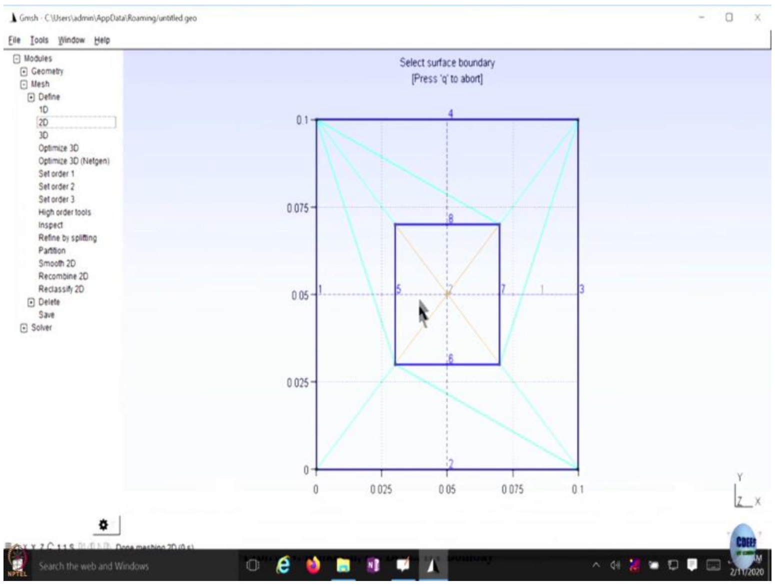


And then you click the character ‘e’ on the key board. In the above slide, you can see it is instructed that ‘press e to end’. So that, 1 will be assigned to the first sub-domain region. You can see 1 in that region. Similarly, the inner rectangle can be selected for the second sub-domain region.

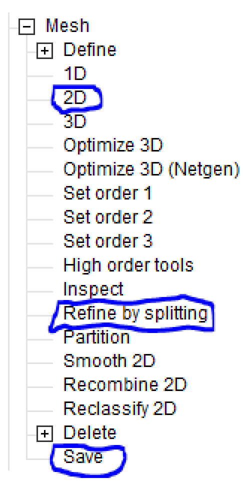


So, for the rectangular conductor, the boundary is formed by the edges of the inner rectangle. Choose the edges of inner rectangle and then click on key e. By this, the second sub-domain is selected. With this, we have assigned the sub-domain numbers also. Now we have to mesh this geometry.

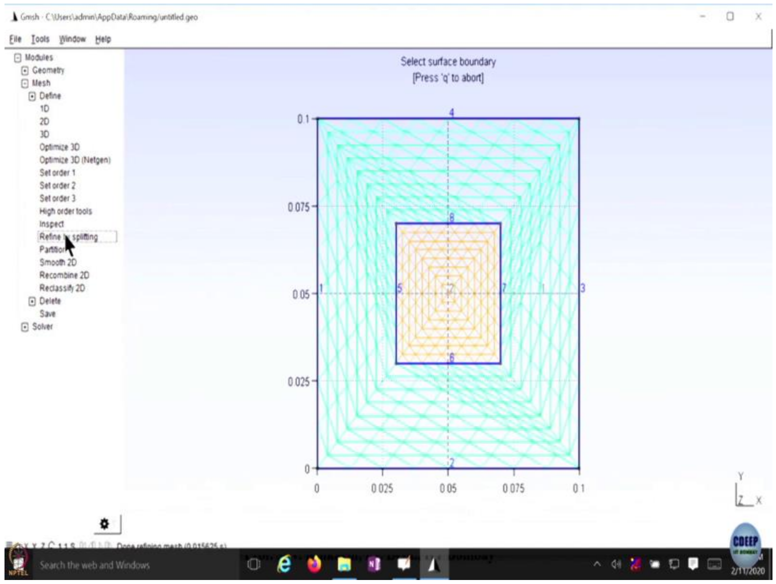
(Refer Slide Time: 10:06)



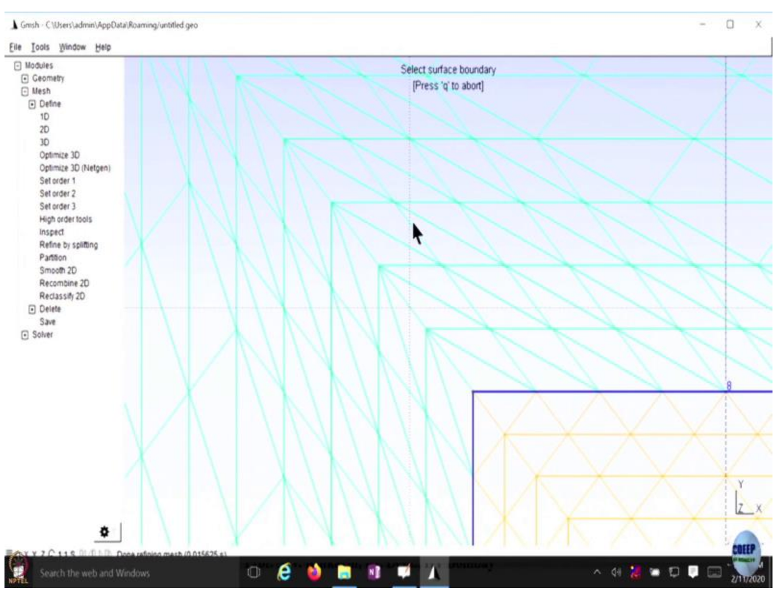
Now go to mesh in which you will find 1D, 2D, and 3D options as shown in the following figure. Using this software, you can mesh a 3D geometry also.

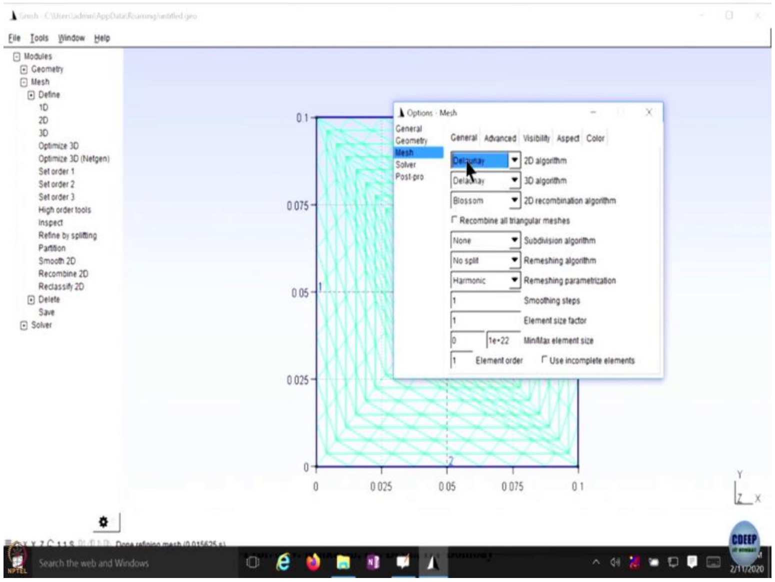


Since the problem domain is a 2D geometry, click on 2D to generate mesh which is shown in the above slide, but this is a very coarse mesh. Now you need to refine this mesh, for that you click on refine mesh. By this you can generate a refined mesh as shown in the following slide.

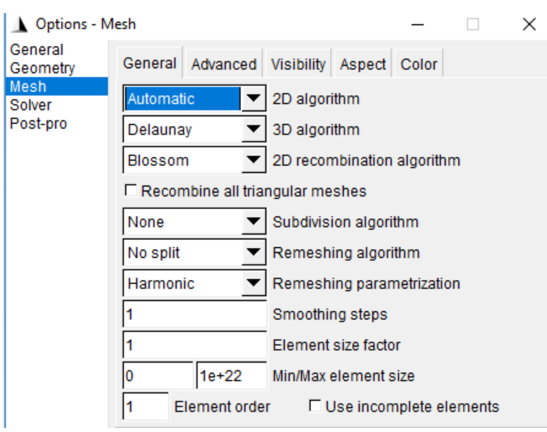


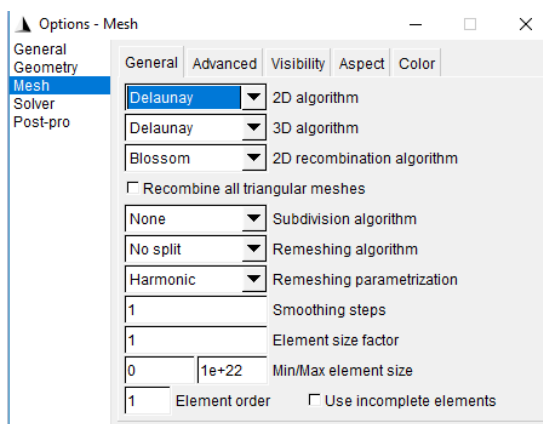
But there is a problem in this mesh. If you zoom the mesh which is shown in the following slide, you can see that the triangles in this mesh are obtuse angle triangles. But it is advisable to have equilateral triangles to get an accurate result. You need to avoid these kind of obtuse angle triangles in your discretization. For that, you need to change the meshing algorithm. There will be a default algorithm to mesh a given geometry using Gmsh. So, you need to change the algorithm of meshing.

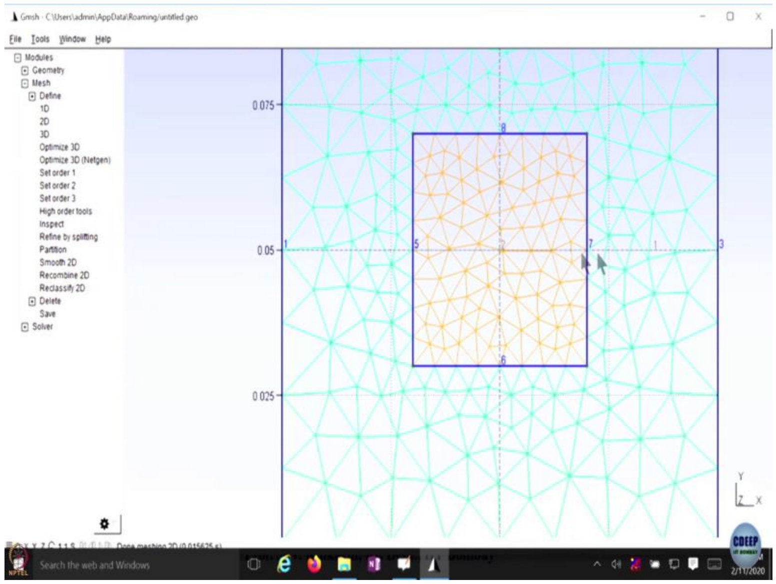


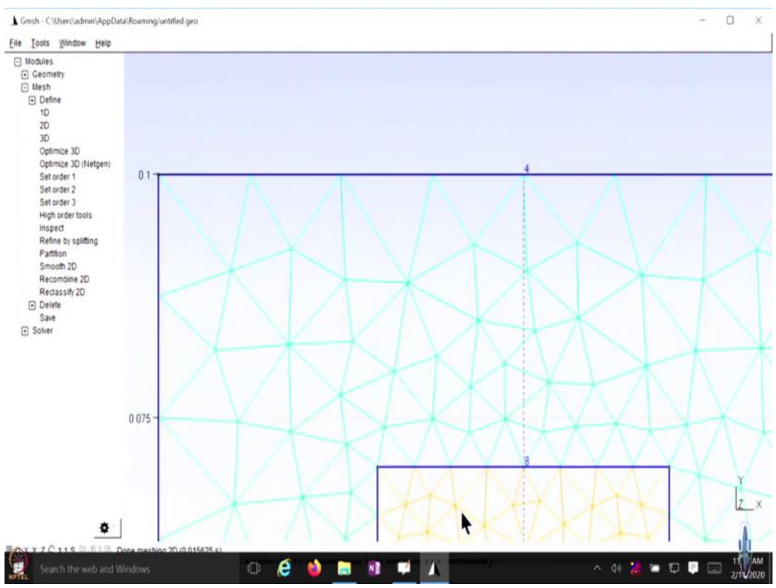


For that, you right click on the space go to 'All Mesh options' which will pop up a window as shown in the following figures. Now click on the 'General' tab. There you will find 2D algorithm, in that automatic will be selected by default as shown in the figure on the left hand side. Instead of that you select the 'Delaunay algorithm'as shown in the figure on the right hand side. So after selecting this algorithm, you click on 2D which is under the mesh tree, then you can see the mesh shape has changed as shown in the next two slides.







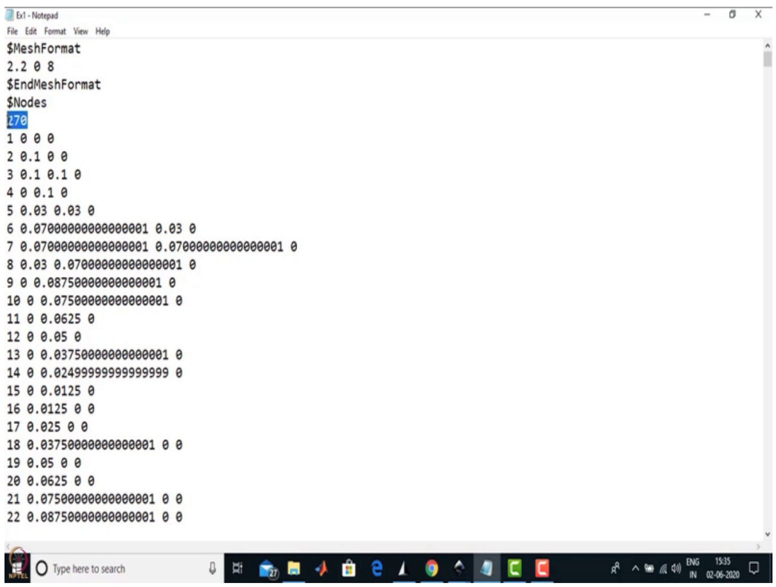


In the above two slides, more or less all elements are equilateral triangles and they are equally spaced. Also, the elements in two sub-regions are with different colors. Now, here we want to highlight one more point. In case of electrostatic problems, there is no need to mesh the geometry that corresponds to the conductor because E field is 0 in that region. So there is no need to calculate the potential values in that region.

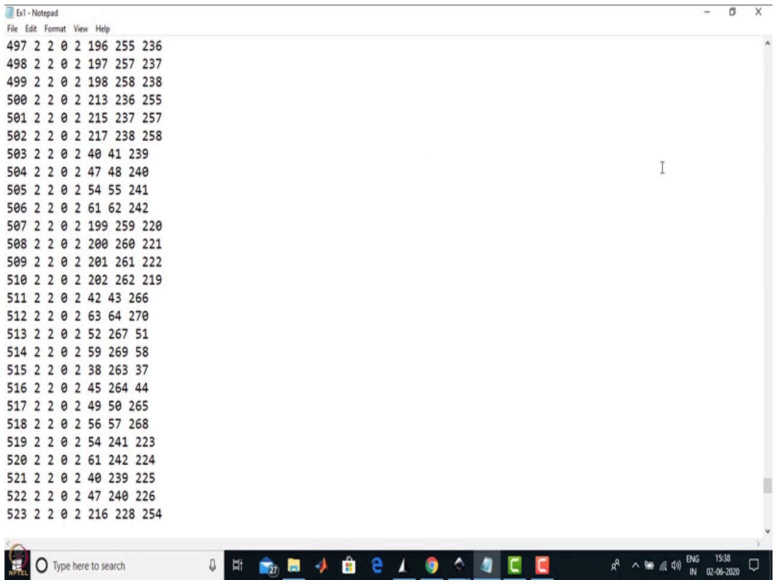
In that case, you should not select the conductor region using plane surface to define it as a sub-domain to avoid meshing that region, just like the way we selected this conductor in the present problem. So, if the problem is an electrostatic case then you should not select the conductor and by doing this, it will not mesh that region and that is how you can reduce the mesh size and computational burden.

Now we have generated the mesh. After this, we need to export this mesh information into the corresponding Scilab code. For that, first of all, we need to save this mesh data. So, under the ‘Mesh tree’, there is a save option and you click on it. The mesh data is saved in the folder where your geometry file is there.

(Refer Slide Time: 13:26)



Now you go to that folder and you can see example1.msh file. So mesh data will get saved in the .msh file. Then you right click on this file, and open it using notepad. You will get the complete information about the generated mesh in this file. This file contains two sets of information. The first set of information will be saved under ‘Section nodes’. In the above slide, you can see that there is a number 270 which corresponds to the total number of nodes in the generated mesh. The rest of the information in this section corresponds to the coordinates of the nodes in the mesh. So, here you have to copy the information that corresponds to the coordinates in a text file, and save it as p.txt in the folder where you will save the corresponding Scilab code. You have to open an empty notepad file, copy the information and save it with the name p.txt.



Ex1 - Notepad

File Edit Format View Help

```
497 2 2 0 2 196 255 236
498 2 2 0 2 197 257 237
499 2 2 0 2 198 258 238
500 2 2 0 2 213 236 255
501 2 2 0 2 215 237 257
502 2 2 0 2 217 238 258
503 2 2 0 2 40 41 239
504 2 2 0 2 47 48 240
505 2 2 0 2 54 55 241
506 2 2 0 2 61 62 242
507 2 2 0 2 199 259 220
508 2 2 0 2 200 260 221
509 2 2 0 2 201 261 222
510 2 2 0 2 202 262 219
511 2 2 0 2 42 43 266
512 2 2 0 2 63 64 270
513 2 2 0 2 52 267 51
514 2 2 0 2 59 269 58
515 2 2 0 2 38 263 37
516 2 2 0 2 45 264 44
517 2 2 0 2 49 50 265
518 2 2 0 2 56 57 268
519 2 2 0 2 54 241 223
520 2 2 0 2 61 242 224
521 2 2 0 2 40 239 225
522 2 2 0 2 47 240 226
523 2 2 0 2 216 228 254
```

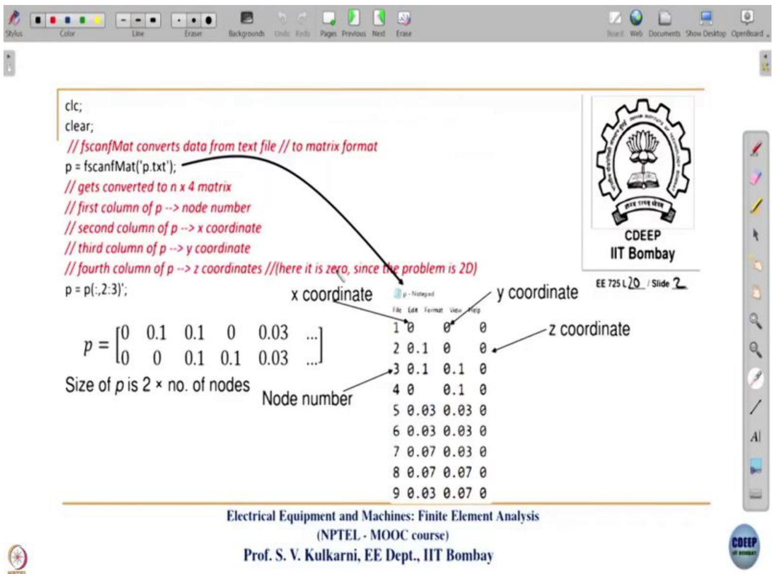
The second set of information will be saved under 'Section elements' and this section contains three sets of data. The first set of data corresponds to the global nodes which we have placed to draw the geometry, the second set of data corresponds to edges that are on the boundaries of the geometry, and the third set of data corresponds to triangles in the mesh, which is nothing but the connectivity matrix (t matrix) that we have seen in the previous lecture.

Here the question that arises is, since 3 sets of data are saved under the same section, how to distinguish these 3 sets of data? The data that corresponds to nodes will have 6 columns, the data that corresponds to edges will have 7 columns, and the data that corresponds to triangles or elements will have 8 columns. Here, we will save the information that corresponds to edges and triangles (elements).

Because the information that corresponds to nodes has no use in our code. First, you have to have copy the information that corresponds to edges in a notepad file, like the way we did for the nodes and save it as e.txt file in the same folder where the p file is saved. Similarly, you have to copy the information that corresponds to triangles or elements in a notepad file and you have to save it in the same folder where the p and e files are saved.

With this, we have completed the task of extracting the mesh information in terms of three matrices that are p, e, and t. Now we will see how to use this information in a Scilab code to solve the 2D magnetostatic problem which we saw in the previous lecture.

(Refer Slide Time: 16:25)



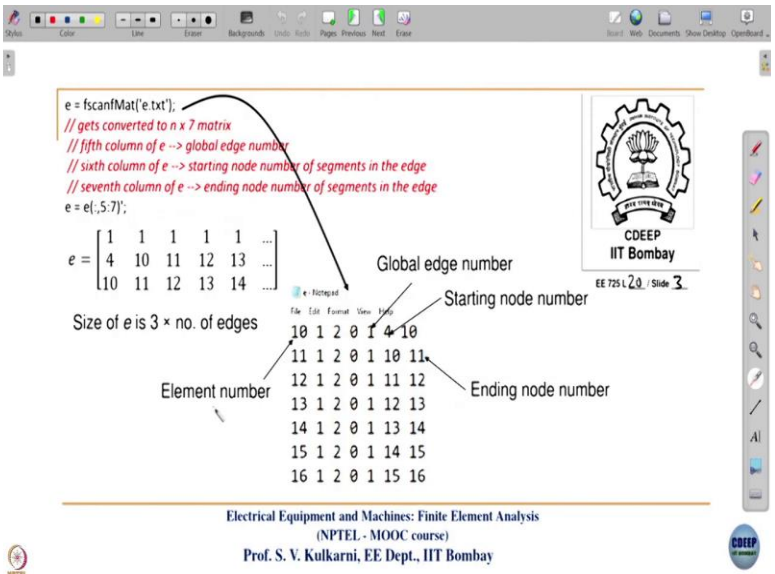
Going further, we have already seen the clc and clear all commands in the previous lecture. The next command is p = fscanfMat('p.txt'). This command will read the data in text file (p.txt file) into a matrix in Scilab. Remember that this p.txt is saved already in the folder where the Scilab code is saved. Therefore, fscanfMat will take out the information that is saved in the p.txt file into the p matrix.

The information that is saved in the p.txt is shown in the figure on the above slide. In this file, the first column is node numbers, x and y coordinates of all the nodes are saved in the second and third columns, and in the fourth column, z coordinates are saved. Since this is a 2D problem (geometry) the z coordinates are 0 by default.

Now, we have to take the things that are needed in the code, so only x and y coordinates are taken from the nodes information. Here, we are taking the second and third columns and updating the p matrix with that data and taking a transpose of it using the command p = p(:,2:3)'. Then the resultant matrix will be in line with the form of p matrix that we have seen in the previous lecture. After this operation, the p matrix will be as given below.

$$p = \begin{bmatrix} 0 & 0.1 & 0.1 & 0 & 0.03 & \dots \\ 0 & 0 & 0.1 & 0.1 & 0.03 & \dots \end{bmatrix}$$

The number of columns in this p matrix are equal to the node numbers. So the size of this matrix is 2×number of nodes.

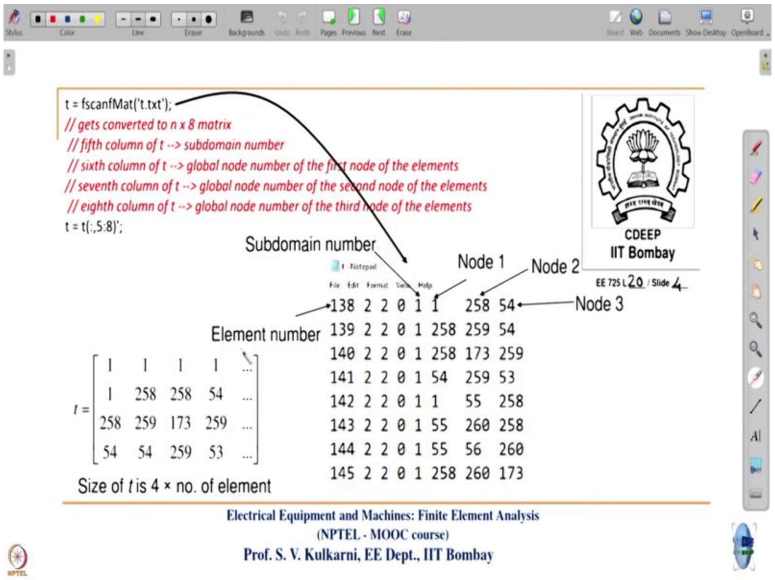


Similarly, we will read the e.txt file into the e matrix using the command e = fscanfMat('e.txt'). In the e.txt file, there are 7 columns as we have discussed earlier. Of these 7 columns, first column represents the element number. As this element number is combination of all the three sets of data, this numbering has no use in our code.

The next three columns are Gmsh generated information and this information has no use in the code. The information that we use in this code is the last 3 columns. The fifth column represents the global edge number and this will be used to impose the boundary conditions. At the starting of this lecture, you may remember that while we drawing the geometry, the edge numbers are assigned in the sequence as we draw the geometry. The starting node number and the ending node number of the edge will be placed in the sixth and seventh columns of the e.txt file. So, after the operation e = e(:,5:7)', we will update the e matrix with the information in fifth to seventh columns and transpose it, so that we will get an e matrix of the kind shown below.

$$e = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \ldots \\ 4 & 10 & 11 & 12 & 13 & \ldots \\ 10 & 11 & 12 & 13 & 14 & \ldots \end{bmatrix}$$

The first row of the updated e matrix will have the global edge numbers, the second and third rows will have the starting node number and the ending node numbers of the edge. So, finally, the size of the e matrix will be 3×number of nodes.



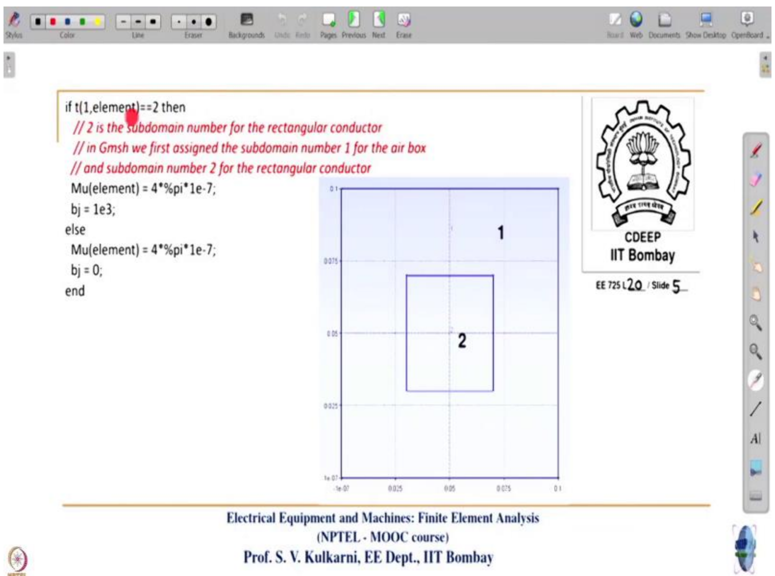
Similar to p and e matrices, we can get the data of t.txt file and save it in a matrix whose name is t. Similar to e matrix, the first 4 columns are of no use and we are going to use the information in fifth column to eighth columns. That is why we are taking columns five to eight and taking a transpose using the command t = t(:,5:8)'. The updated t matrix will be of the kind shown below.

$$t = \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots \\ 1 & 258 & 258 & 54 & \ldots \\ 258 & 259 & 173 & 259 & \ldots \\ 54 & 54 & 259 & 53 & \ldots \end{bmatrix}$$

The first row of the t matrix denotes the sub-domain number and we will use it to assign the material properties and the source conditions. The second, third, and fourth rows contain the global node numbers of each element and the element number is nothing but the column number of this matrix. The first column corresponds to the first element, the second column corresponds to the second element, and so on.

Finally, the size of the t matrix will be 4×number of elements. So, up to here we have seen meshing the geometry and exporting the mesh data into the code. This part is completely different from the code that we saw in the previous lecture. From here, the next part up to the calculation of element coefficient matrices is the same. Next thing that we need to change is assigning the source conditions.

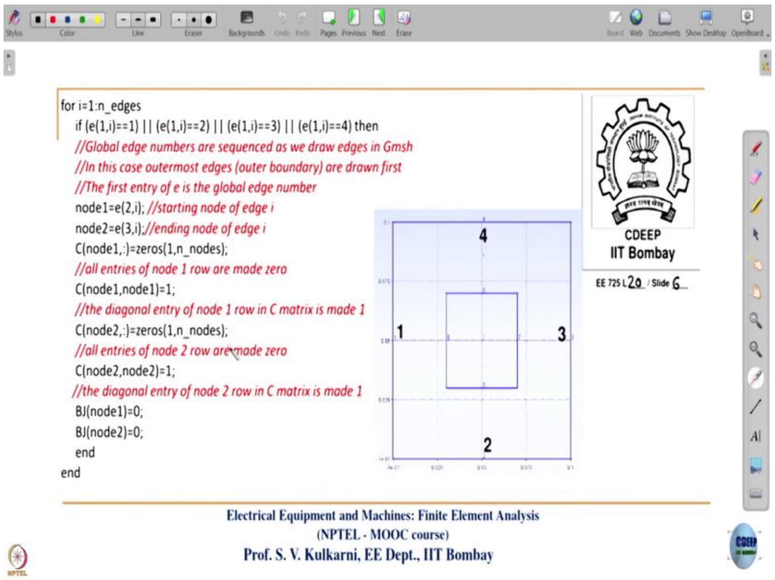
(Refer Slide Time: 22:14)



If you remember in the previous lecture, we have assigned the source conditions by verifying whether its centroid is inside the rectangular conductor region or not. But, in this code, we are not using that logic and we will just check the sub-domain number of each and assign source conditions. As we have seen earlier, the first row of each element corresponds to sub-domain number, so in the code given in the above slide, we are checking whether the sub-domain number of the considered element is 2 or not. Remember that 2 is the subdomain number of the rectangular conductor. If it is 2, then we assign the source to that element, if not, we will not assign the source. Since, there are only two subdomains, subdomain 1 corresponds to the air region and subdomain number 2 corresponds to the rectangle conductor. Here, we are assigning the source only to the rectangular conductor.

The part of the code shown in the above slide will be changed as compared to the code that we saw in the previous lecture.

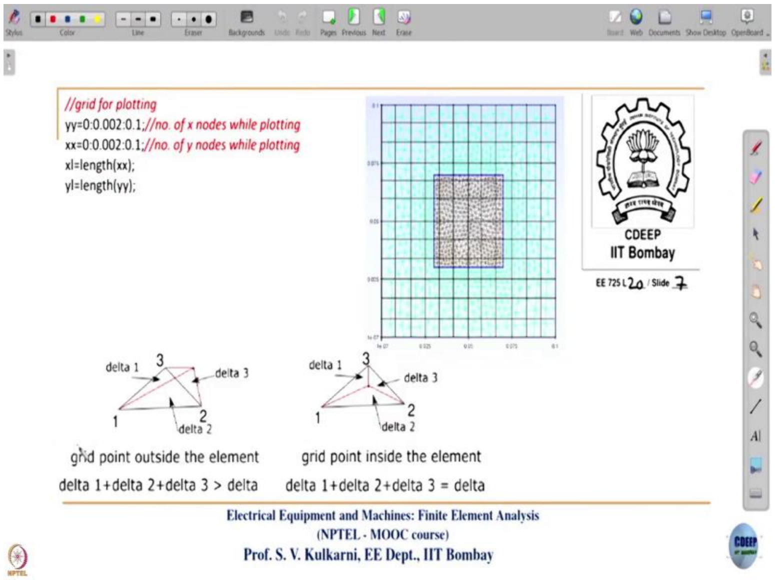
(Refer Slide Time: 22:45)



Similarly, the next part of the code that we need to change is assigning the boundary condition. In the previous lecture, we have assigned the boundary conditions node wise. For that we have to do a book keeping of all the node numbers that are on the boundary and we have made a separate logic for different edges of the outer boundary. But in the present code, boundary conditions are imposed with the help of global edge numbers information generated using the Gmsh software.

Remember that we have drawn the geometry and we have assigned 1, 2, 3, 4 as global edge numbers for the outer four edges where the value of A should be imposed. In the code given in the above slide we will check whether the global edge number of each edge under consideration is 1 or 2 or 3 or 4 and we will assign the boundary condition (A = 0). The part of the code given in the above slide is same for every code. Also, the logic with which this part is developed is same as the one we saw in the previous lecture where we applied boundary condition to one node at a time because we are assigning the boundary conditions node wise. But each edge will have two nodes, so that is why we are assigning boundary conditions to two nodes at a time. That is why in the previous code you have three steps and here you have six steps.

(Refer Slide Time: 24:12)



After this, the part of the code that corresponds to plotting the potentials has to be changed. The surf and contour plots which we saw in the previous lecture are applicable only for rectangular grids, you cannot use those commands to plot potentials of a triangular mesh. First we need to form a rectangular grid and you calculate the potential at each node of this rectangular grid using the solution that you have obtained using a triangular mesh.

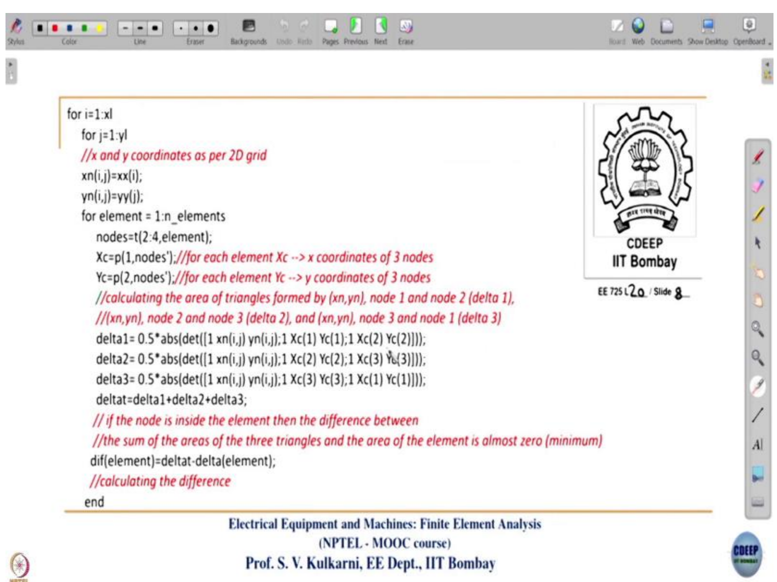
You frame a grid with many number of points as shown in the figure given in the above slide. You determine the element in which each node is present and then calculate the potential of that grid node using the potentials of the three nodes of the element. So, like this, you can calculate the potential of the complete rectangle grid and plot the potentials using surf and contour plot commands. This is the logic that we are going to use to plot the potentials in the code where the mesh generated using Gmsh is used.

How can you know the element number in which the grid node is present? Remember that a point can be inside one triangle only. You can frame three triangles as shown in the following figures using the (node) point and a triangular element that we are verifying.

The area of the triangle formed by the node under consideration and nodes 1 and 3 of the triangle is delta 1, whereas area formed by the node under consideration and nodes 1 and 2 is delta 2, and area formed by the node under consideration and nodes 2 and 3 is delta 3. If the sum of the areas of the three triangles is equal to the area of the triangular element then the node is inside the triangle as shown in the figure on the right hand side. If not, then that area will be more than the area of triangular element as shown in the figure on the left hand side. Here, to calculate the potentials on the grid which is used for plotting purpose, we are going to do the same logic in this slide. So, we are going to discretize the geometry in the same fashion that we have done in the previous code using the following commands.

```
//grid for plotting
yy=0:0.002:0.1;//no. of x nodes while plotting
xx=0:0.002:0.1;//no. of y nodes while plotting
xl=length(xx);
yl=length(yy);
```

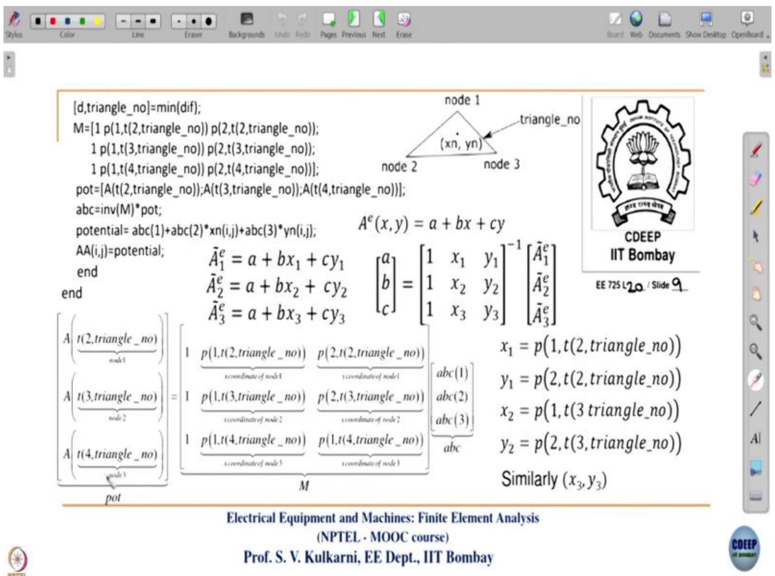
(Refer Slide Time: 26:43)



For each grid node and element, we are calculating the areas delta 1, delta 2, and delta 3 and the difference delta – (delta 1+ delta 2+ delta 3) using the code given in the above slide.

We are taking the difference because this is a computing software and sometimes delta and delta 1+ delta 2+ delta 3 may not be numerically equal. That is why we are calculating the difference between the sum of the areas and area of that triangle and we are storing it in a matrix named dif.

(Refer Slide Time: 27:25)



Once we store the value of difference for all elements, we will take the number of the triangle (or element) with minimum difference. With this we got to know in which triangle the node under consideration is present. Now using the potential approximation $(A = a + bx + cy)$ inside an element, remember that this approximation is our initial step to develop FE formulation, we will calculate the potential at the grid node using the potentials at the three nodes of the element and this approximation. Using the potential values of the three nodes, we calculate the constants a, b, c by inverting the matrix given below.

$$\tilde{A}_1^e = a + bx_1 + cy_1 \qquad \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{A}_1^e \\ \tilde{A}_2^e \\ \tilde{A}_3^e \end{bmatrix}$$
$$\tilde{A}_2^e = a + bx_2 + cy_2$$
$$\tilde{A}_3^e = a + bx_3 + cy_3$$

This step also we have seen in previous lectures while developing the FE formulation. Now once we calculate the values of a, b, and c, you substitute x and y coordinates of the point where you want to calculate the potential in the potential function $A = a + bx + cy$.

We have used this logic to develop this part of the code. The inverted matrix is named as M in the code and is given below.

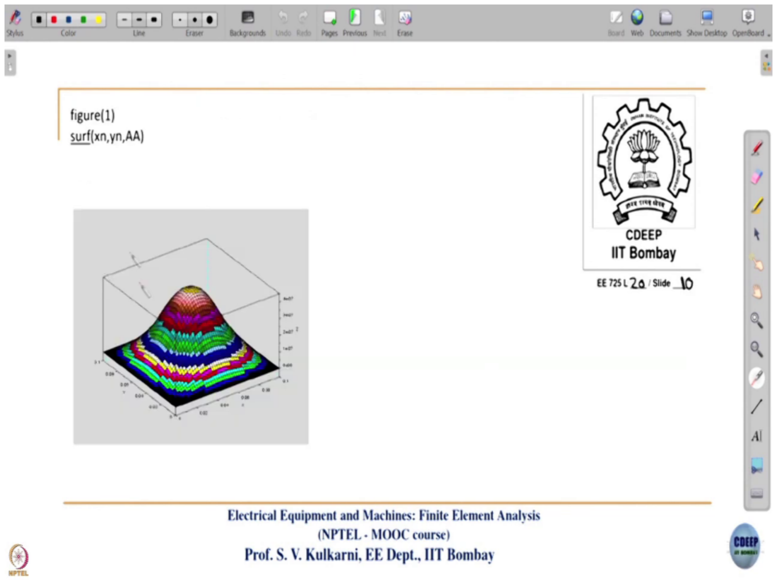
```
M=[1 p(1,t(2,triangle_no)) p(2,t(2,triangle_no));
   1 p(1,t(3,triangle_no)) p(2,t(3,triangle_no));
   1 p(1,t(4,triangle_no)) p(2,t(4,triangle_no))];
```

Here the x and y coordinates of each node are stored in p matrix, the first row of p matrix is x coordinate and second row of p matrix is y coordinate. The global node number of the vertices of a triangular element are stored in 2 to 4 rows of t matrix.

Using the commands p(1,t(2,triangle_no)) and p(2,t(2,triangle_no)), you can get the information of x_1, y_1, x_2, y_2, x_3, and y_3 from the M matrix as shown in the above figure. Calculate the values of a, b, c using the command abc=inv(M)*pot and the matrix pot (pot=[A(t(2,triangle_no));A(t(3,triangle_no));A(t(4,triangle_no))];) will have potentials of the three nodes of the element and we are storing those values in abc matrix. The abc matrix is a 3×1 matrix in which abc(1) corresponds to a, abc(2) corresponds to b, abc(3) corresponds to c. Substitute the x and y coordinates as given below to calculate the potential at the corresponding grid point.

```
potential= abc(1)+abc(2)*xn(i,j)+abc(3)*yn(i,j);
```

After this, you can plot the potentials as shown in the following slide using the same surf command. This logic of plotting potentials using a rectangular grid on a triangular mesh is used in all the codes. So, this logic will be there in all the Scilab codes. In case of MATLAB there is an inbuilt pde toolbox and that toolbox will also be used to generate same p, e, t matrices. Using that pde toolbox command which is called as pdeplot you can plot the potentials directly. But in case of Scilab we do not have such command. That is why we are using this logic to plot the potentials. Similarly you can draw a contour plot for this solution, thank you.



(Refer Slide Time: 30:09)