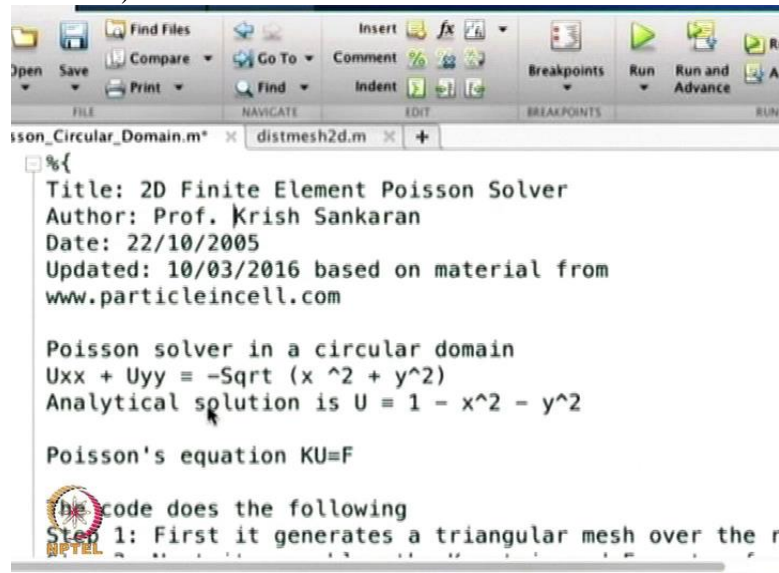


Computational Electromagnetics and Applications
Professor Krish Sankaran
Indian Institute of Technology Bombay
Lecture No. 22
Finite Element Method (FEM)

We will do some simple simulation on Matlab, how Poisson equation is going to pan out. We did a lot of mathematics now but let us look into a practical way of solving this using Matlab. (Refer Slide Time: 00:29)



```
%{
Title: 2D Finite Element Poisson Solver
Author: Prof. Krish Sankaran
Date: 22/10/2005
Updated: 10/03/2016 based on material from
www.particleincell.com

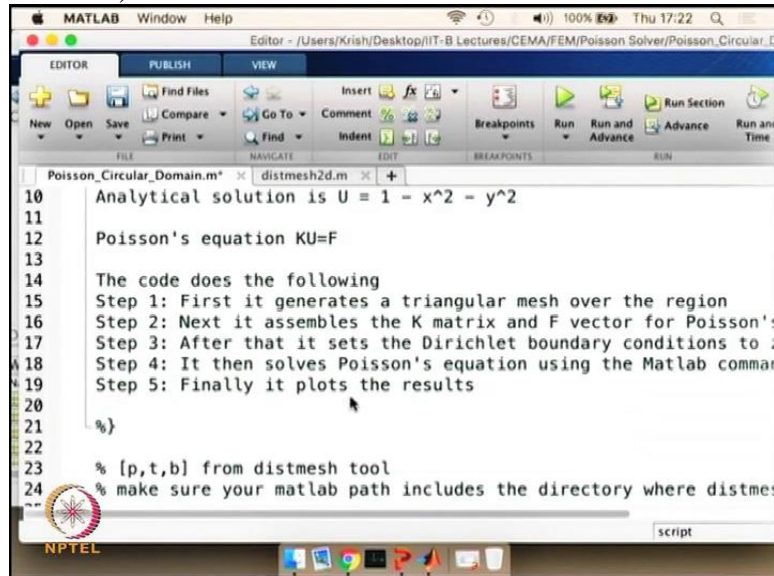
Poisson solver in a circular domain
Uxx + Uyy = -Sqrt (x ^2 + y^2)
Analytical solution is U = 1 - x^2 - y^2

Poisson's equation KU=F

code does the following
Step 1: First it generates a triangular mesh over the r
```

So Matlab luckily has some of the inbuilt functions which gives you easy way to get certain results of matrices manipulation. So I am going to use that to make things simple. But the goal of this demonstration is to show how the entire field distribution whether it is potential or it is electric field its going to change as a function of the discretization itself. So with that in goal let us look into the solver which I have got. So it is based on the material which is freely available online particle in sell so I have a Poisson solver on circular domain and the Poisson equation is given as $u_{xx} + u_{yy} = -\sqrt{x^2 + y^2}$ is equal to the square root of certain value. So the Rho value which is nothing but surface charge density is going to depend on x and y. So that is the idea and there is an analytical solution that one can derive for this but let us go into the step one by one.

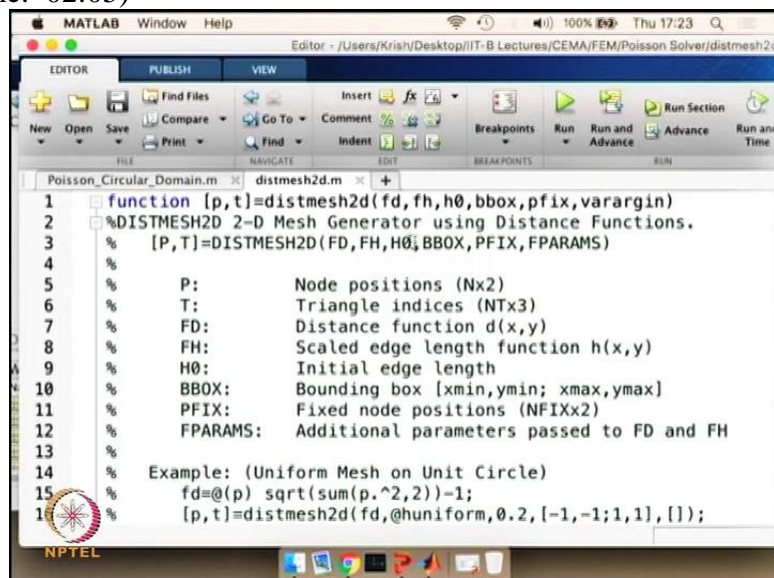
(Refer Slide Time: 01:32)



```
10 Analytical solution is U = 1 - x^2 - y^2
11
12 Poisson's equation KU=F
13
14 The code does the following
15 Step 1: First it generates a triangular mesh over the region
16 Step 2: Next it assembles the K matrix and F vector for Poisson's
17 Step 3: After that it sets the Dirichlet boundary conditions to z
18 Step 4: It then solves Poisson's equation using the Matlab comman
19 Step 5: Finally it plots the results
20
21 %}
22
23 % [p,t,b] from distmesh tool
24 % make sure your matlab path includes the directory where distmes
```

So the Poisson equation is of the form $k u$ is equal to F . Remember in our earlier example we said it will be $K \Phi$ is equal to b . What the code actually does is first it generates a triangular mesh over a region. And for this I am going to use the mesh generator which is called as the distmesh, so it is freely available online. So it is going to give us certain mesh which is based on the amount what we are putting in.

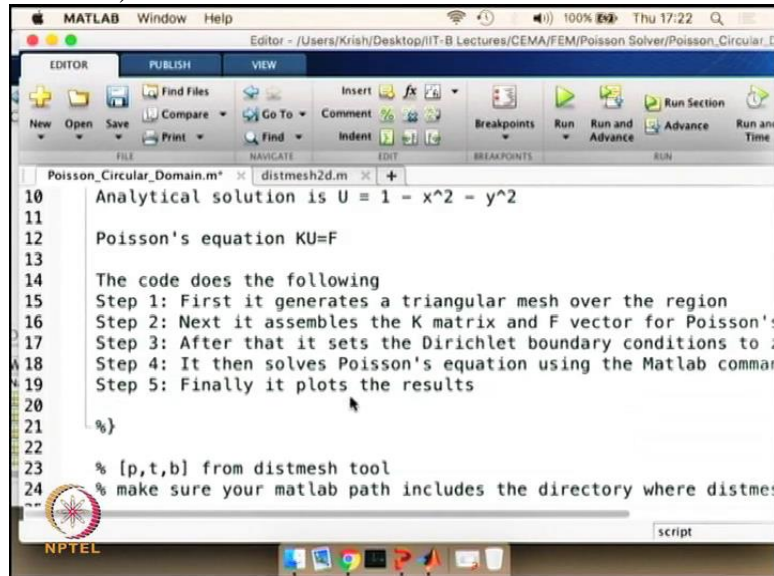
(Refer Slide Time: 02:05)



```
1 function [p,t]=distmesh2d(fd,fh,h0,bbox,pfix,varargin)
2 %DISTMESH2D 2-D Mesh Generator using Distance Functions.
3 [P,T]=DISTMESH2D(FD,FH,H0,BBOX,PFIX,FPARAMS)
4
5 % P: Node positions (Nx2)
6 % T: Triangle indices (NTx3)
7 % FD: Distance function d(x,y)
8 % FH: Scaled edge length function h(x,y)
9 % H0: Initial edge length
10 % BBOX: Bounding box [xmin,ymin; xmax,ymax]
11 % PFIX: Fixed node positions (NFIXx2)
12 % FPARAMS: Additional parameters passed to FD and FH
13
14 % Example: (Uniform Mesh on Unit Circle)
15 % fd=@(p) sqrt(sum(p.^2,2))-1;
16 % [p,t]=distmesh2d(fd,@huniform,0.2,[-1,-1;1,1],[]);
```

So if I say my discretization is going to be H 0 H length is going to be certain number. So it is going to generate the edges of that length. And what is interesting here is I am going to give also the starting and ending point so the bounding domain. So the bounding domain is going to be the x minimum and why minimum and x max and y max. So once I give that it is going to give me a domain with certain grids so let us see how it does using this example.

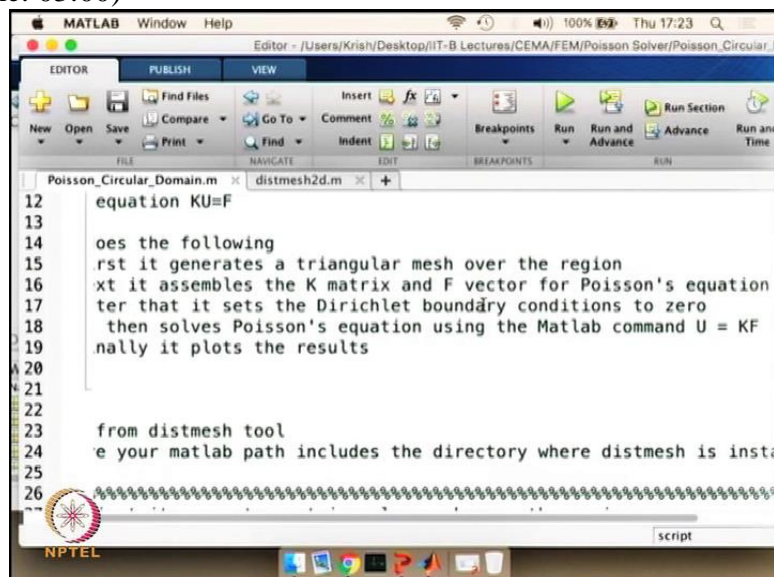
(Refer Slide Time: 02:39)



```
10 Analytical solution is U = 1 - x^2 - y^2
11
12 Poisson's equation KU=F
13
14 The code does the following
15 Step 1: First it generates a triangular mesh over the region
16 Step 2: Next it assembles the K matrix and F vector for Poisson's
17 Step 3: After that it sets the Dirichlet boundary conditions to zero
18 Step 4: It then solves Poisson's equation using the Matlab command U = KF
19 Step 5: Finally it plots the results
20
21 %}
22
23 % [p,t,b] from distmesh tool
24 % make sure your matlab path includes the directory where distmesh is installed
```

So first it generates the mesh and then it goes into the k matrix assembly remember I explained you how the k matrix is done step by step. We take each of the local elements and then we put them on the global matrix and we construct the entire matrix. So that is what we are going to do here.

(Refer Slide Time: 03:00)

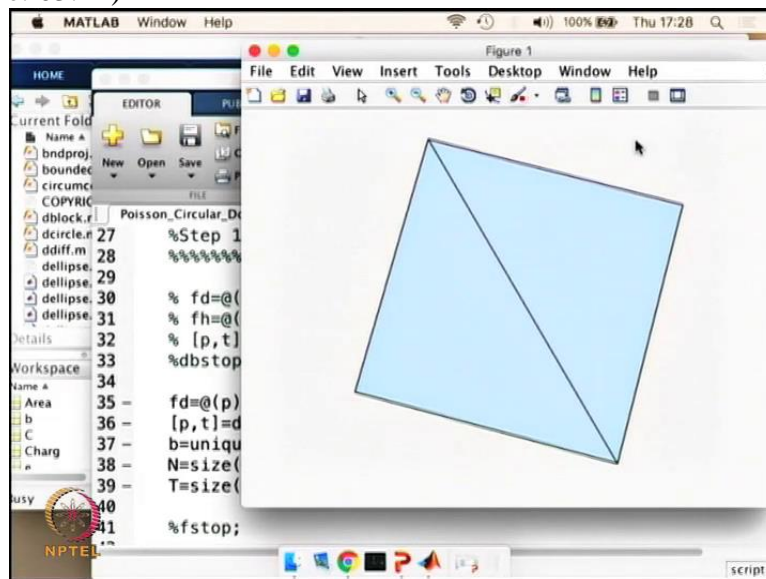


```
12 equation KU=F
13
14 oes the following
15 rst it generates a triangular mesh over the region
16 xt it assembles the K matrix and F vector for Poisson's equation
17 ter that it sets the Dirichlet boundary conditions to zero
18 then solves Poisson's equation using the Matlab command U = KF
19 nally it plots the results
20
21
22
23 from distmesh tool
24 e your matlab path includes the directory where distmesh is installed
```

And after that I am assigning certain boundary condition which is I am saying Dirichlet boundary condition and I am assigning the value of potential unknown to be 0 and those boundaries and then I am trying to solve the Poisson equation. So I am going to show you how the domain itself is going to look like based on the input values I am giving here. So the domain bounding values are here. The minimum values of x the minimum value of y, the maximum value of x and maximum value of y. I make a functional stop here so that you can

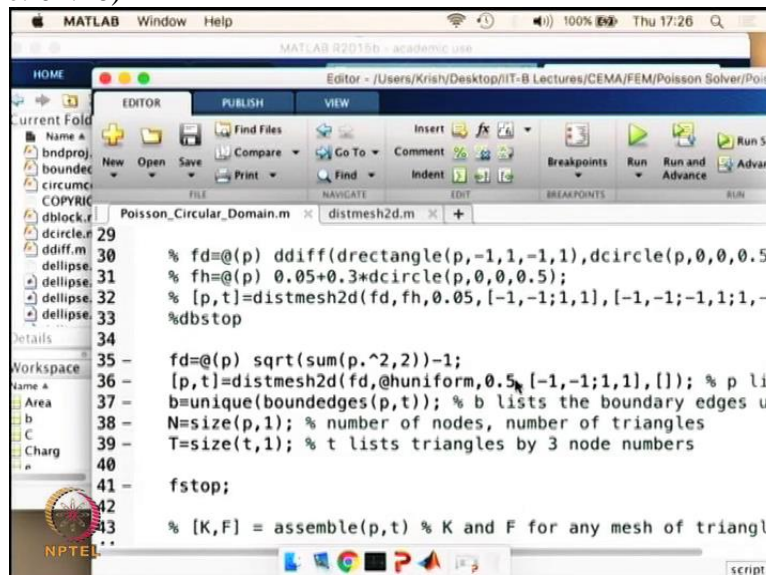
see the simulation and the domain itself let us put point 8 and see how the discretization is going to look like.

(Refer Slide Time: 03:44)



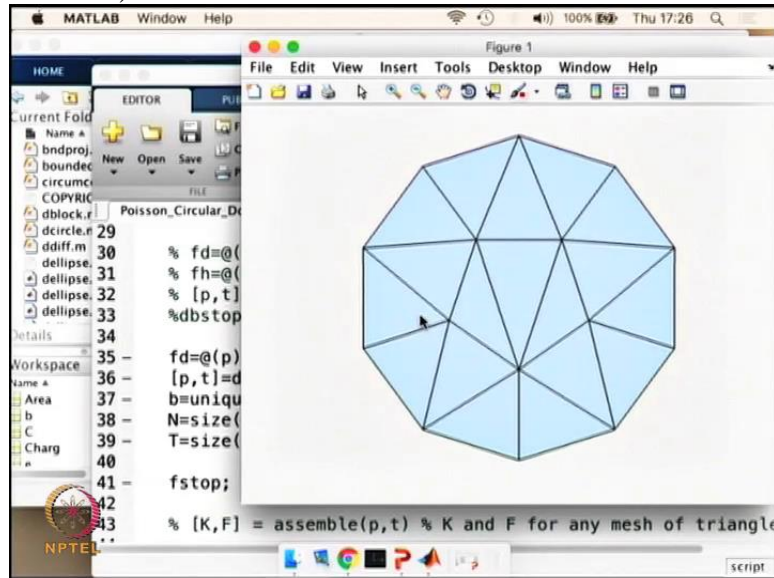
This is not a very good discretization the reason is I say the starting and ending point is minimum value is minus 1 and the maximum value is 1. So inside a particular domain I can have maximum only two cells because $0.8 + 0.8$ is going to be give me 1.6 and the domain maximum limits is going to be 2. So it is not very good. So putting it 0.8 is going to be very very coarse discretization.

(Refer Slide Time: 04:16)



So having 0.8 is not a good solution so I am going to refine the mesh by reducing the edge length to 0.5, let me see how the domain is going to look like.

(Refer Slide Time: 04:30)



You can see the length of the each of the edges is going to be approximately 0.5 and based on that the dish mesh is going to give me domain. And of course what you also see is the domain is no longer circular. In the initial case it was really bad. But here you see that it is confirming more and more to the circular area. So what I am going to do is I am going to keep 0.5 as my domain discretization maximum edge length and I am going to run the code without stopping it and then see what will be the result of the simulation.

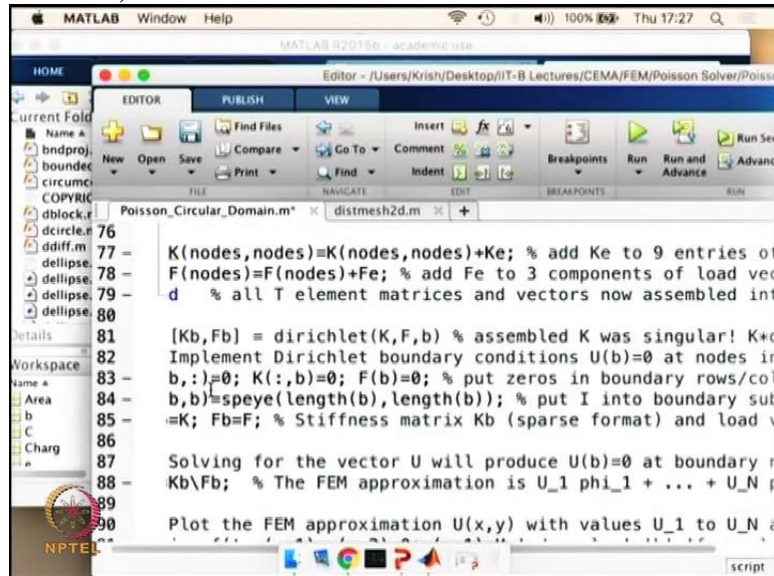
(Refer Slide Time: 05:07)

A screenshot of the MATLAB software interface showing a script editor. The script is titled 'Poisson_Circular_Domain.m' and contains the following code:

```
10 Analytical solution is U = 1 - x^2 - y^2
11
12 Poisson's equation KU=F
13
14 The code does the following
15 Step 1: First it generates a triangular mesh over the region
16 Step 2: Next it assembles the K matrix and F vector for Poisson's
17 Step 3: After that it sets the Dirichlet boundary conditions to 2
18 Step 4: It then solves Poisson's equation using the Matlab comman
19 Step 5: Finally it plots the results
20
21 %}
22
23 % [p,t,b] from distmesh tool
24 % make sure your matlab path includes the directory where distmes
```

So basically I said I have defined the domain I am assigning the k matrix to be 0 initially and I am going inside each of the triangular elements, each element at a time I go and load the values one by one and then I compute the value finally and I know the integration I integrated over each of the triangles to get the value of f which is on the right hand side

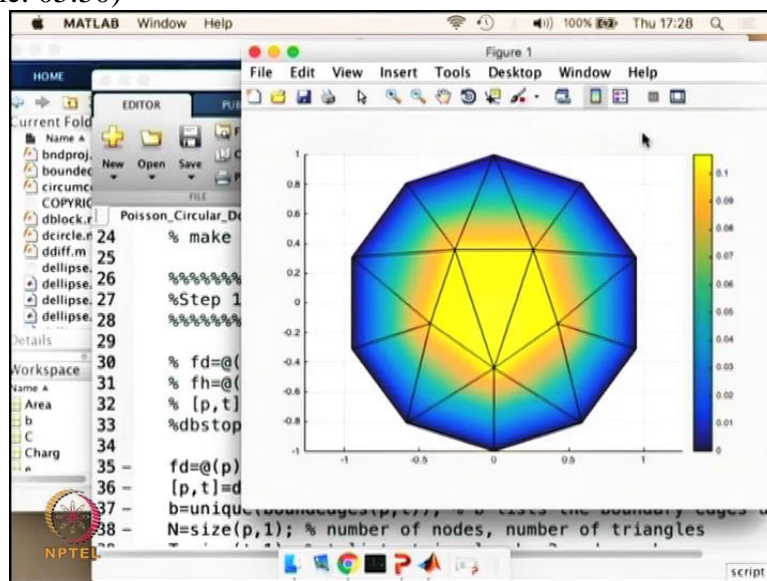
(Refer Slide Time: 05:34)



```
76 K(nodes,nodes)=K(nodes,nodes)+Ke; % add Ke to 9 entries of
77 F(nodes)=F(nodes)+Fe; % add Fe to 3 components of load vec
78 d % all T element matrices and vectors now assembled into
79
80
81 [Kb,Fb] = dirichlet(K,F,b) % assembled K was singular! K+d
82 Implement Dirichlet boundary conditions U(b)=0 at nodes in
83 b,:)=0; K(:,b)=0; F(b)=0; % put zeros in boundary rows/col
84 b,b)=speye(length(b),length(b)); % put I into boundary sub
85 =K; Fb=F; % Stiffness matrix Kb (sparse format) and load v
86
87 Solving for the vector U will produce U(b)=0 at boundary n
88 Kb\Fb; % The FEM approximation is U_1 phi_1 + ... + U_N p
89
90 Plot the FEM approximation U(x,y) with values U_1 to U_N a
```

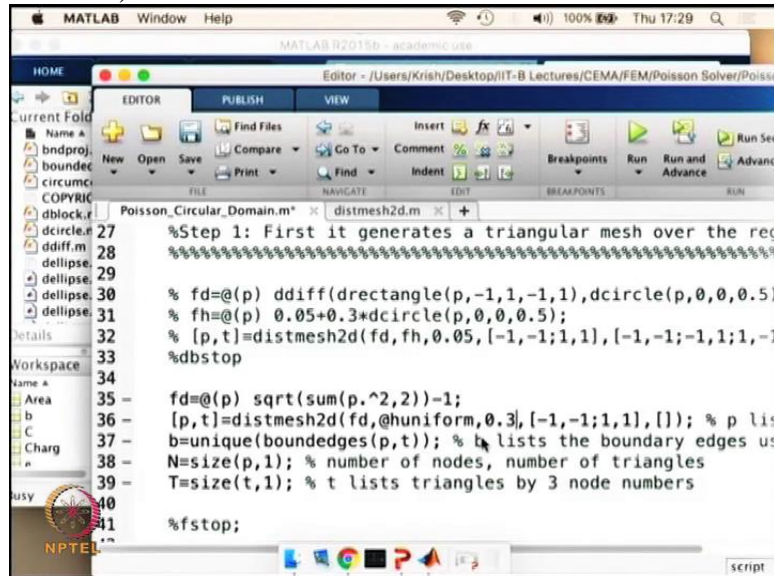
And finally I am going to invert the matrix by solving this equation so initially it is going to be k of u is equal to b and I can get the k inverse by this Matlab function. So let us run this code and see how the result is going to look like.

(Refer Slide Time: 05:50)



So what we have got is we have got a very very crude discretization and the maximum is at the centre. The ρ maximum is at the centre and for the function what we have chosen the value of ρ is going to change as a function of minus square root of x square plus y square so it is going to be a circular charge variation and you have the eq potential lines which are circular lines. That is what you see and that is how it should look.

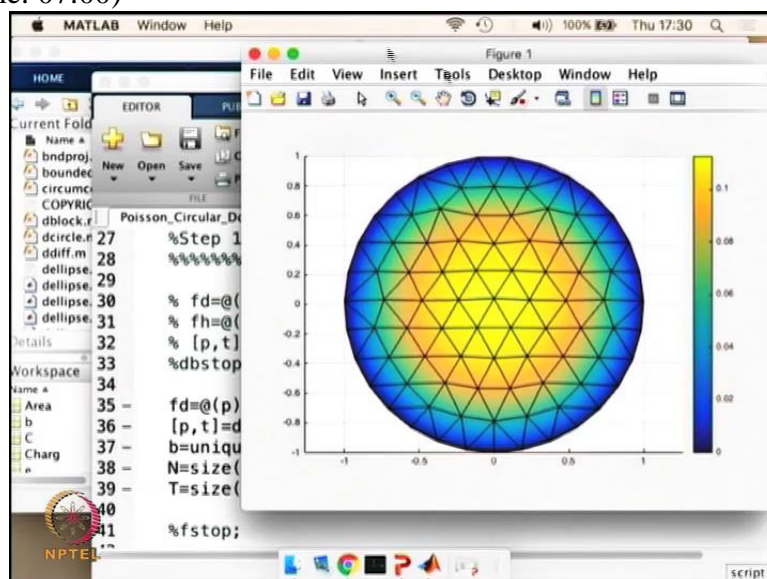
(Refer Slide Time: 06:23)



```
27 %Step 1: First it generates a triangular mesh over the reg
28 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29
30 % fd=@(p) ddiff(drectangle(p,-1,1,-1,1),dcircle(p,0,0,0.5))
31 % fh=@(p) 0.05+0.3*dcircle(p,0,0,0.5);
32 % [p,t]=distmesh2d(fd,fh,0.05,[-1,-1;1,1],[-1,-1;-1,1;-1,1])
33 %dbstop
34
35 fd=@(p) sqrt(sum(p.^2,2))-1;
36 [p,t]=distmesh2d(fd,@uniform,0.3,[-1,-1;1,1],[]); % p lis
37 b=unique(boundedges(p,t)); % b lists the boundary edges us
38 N=size(p,1); % number of nodes, number of triangles
39 T=size(t,1); % t lists triangles by 3 node numbers
40
41 %fstop;
```

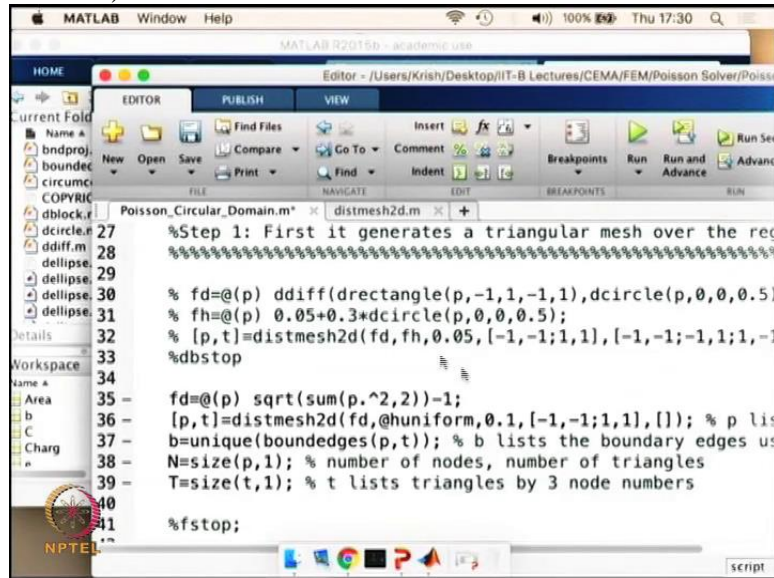
So let us go to 0.3. So the maximum edge length is going to be 0.3 now. And I am running the same program so you see the value is getting more and more finer you see the nuances of the equipotential areas initially when we had 0.5 we did not see the green equipotential line we start to see it now. So if I refine it even further let us say I am going to 0.2.

(Refer Slide Time: 07:00)



I might be able to see more refinement.

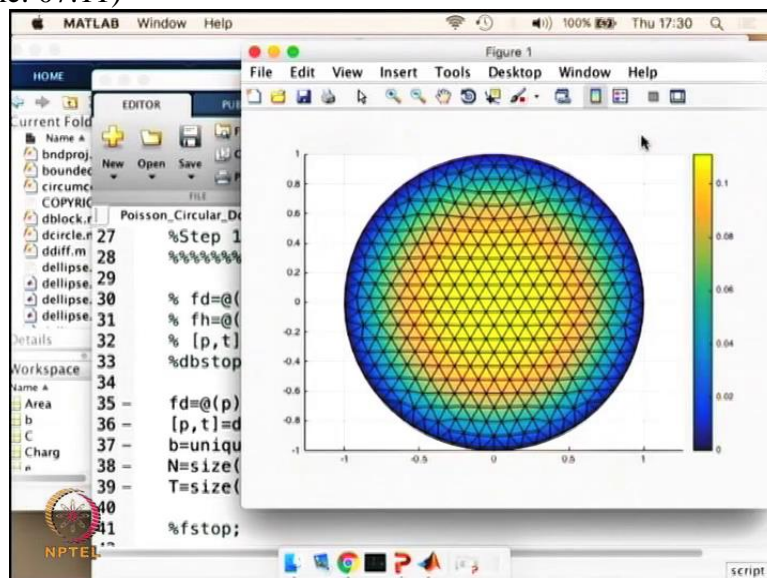
(Refer Slide Time: 07:09)



```
27 %Step 1: First it generates a triangular mesh over the reg
28 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29
30 % fd=@(p) ddiff(drectangle(p,-1,1,-1,1),dcircle(p,0,0,0.5)
31 % fh=@(p) 0.05+0.3*dcircle(p,0,0,0.5);
32 % [p,t]=distmesh2d(fd,fh,0.05,[-1,-1;1,1],[-1,-1;-1,1;-1
33 %dbstop
34
35 fd=@(p) sqrt(sum(p.^2,2))-1;
36 [p,t]=distmesh2d(fd,@huniform,0.1,[-1,-1;1,1],[]); % p lis
37 b=unique(boundedges(p,t)); % b lists the boundary edges us
38 N=size(p,1); % number of nodes, number of triangles
39 T=size(t,1); % t lists triangles by 3 node numbers
40
41 %fstop;
```

And likewise I can go to 0.1 also.

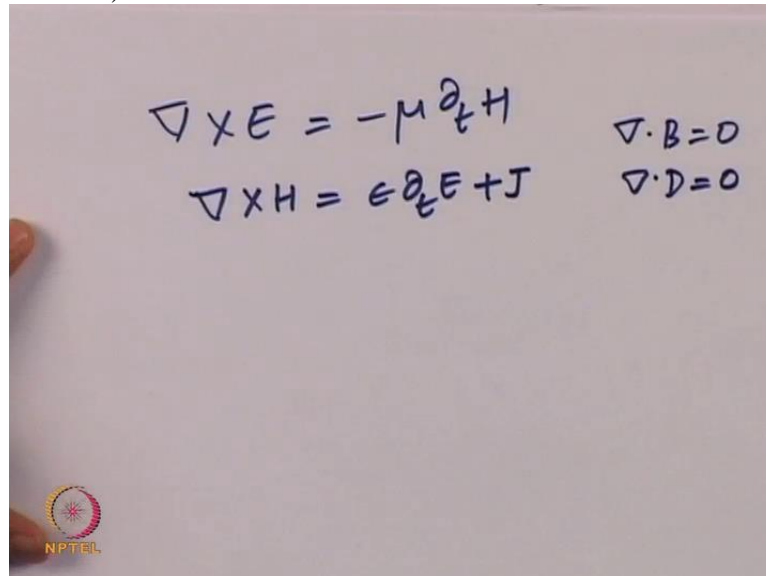
(Refer Slide Time: 07:11)



So you are able to see the various accurate results for various discretization obviously the more finer our element dimensions are going to be the domain is going to be accurately also represented. So what we have demonstrated in this example is to get a physical sense of how the potential fields are going to be and also how to do it in a simple way using Matlab. In the beginning remember I said one thing is missing which is the domain approach of the Maxwell equation itself.

So that is what we will focus in next part of this module. We will start with the Maxwell equation itself and in time domain and we will model it for the finite element approach.

(Refer Slide Time: 08:00)



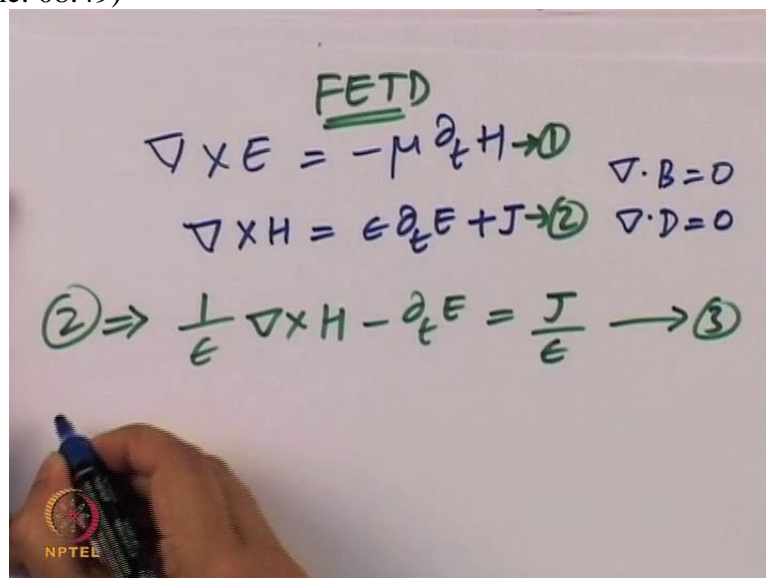
Handwritten Maxwell equations on a whiteboard:

$$\begin{aligned}\nabla \times E &= -\mu \partial_t H & \nabla \cdot B &= 0 \\ \nabla \times H &= \epsilon \partial_t E + J & \nabla \cdot D &= 0\end{aligned}$$

The NPTEL logo is visible in the bottom left corner.

So let us start with the basic Maxwell equation itself which is given by the two curl equations. The curl of E is equal to minus Mu d t H and the curl of H is equal to epsilon d t E plus J. So what we have said now we are going to start with these two curl equations. Now obviously there are going to be the diversions conditions the diversions of B is equal to 0 and the diversions of D is also we have set it 0.

(Refer Slide Time: 08:49)



Handwritten Maxwell equations on a whiteboard, with the second equation being manipulated:

FETD

$$\begin{aligned}\nabla \times E &= -\mu \partial_t H \rightarrow \textcircled{1} & \nabla \cdot B &= 0 \\ \nabla \times H &= \epsilon \partial_t E + J \rightarrow \textcircled{2} & \nabla \cdot D &= 0\end{aligned}$$
$$\textcircled{2} \Rightarrow \frac{1}{\epsilon} \nabla \times H - \partial_t E = \frac{J}{\epsilon} \rightarrow \textcircled{3}$$

A hand holding a blue marker is visible at the bottom left.

And we can start manipulating the second equation, so let us say this is 1 this is 2. So our approach is we are going to go for finite element time domain. So what we are going to do is from 2 we have 1 by Epsilon curl of h minus d by dt of E is equal to J by epsilon so this is 3. So I am going to multiply 3 by the curl.


(Refer Slide Time: 09:30)

FETD

$$\nabla \times E = -\mu \frac{\partial H}{\partial t} \rightarrow (1) \quad \nabla \cdot B = 0$$

$$\nabla \times H = \epsilon \frac{\partial E}{\partial t} + J \rightarrow (2) \quad \nabla \cdot D = 0$$

$$(2) \Rightarrow \frac{1}{\epsilon} \nabla \times H - \frac{\partial E}{\partial t} = \frac{J}{\epsilon} \rightarrow (3)$$

$$(3) \times \nabla \times \Rightarrow \nabla \times \frac{1}{\epsilon} \nabla \times H - \frac{\partial}{\partial t} \nabla \times E = \frac{1}{\epsilon} \nabla \times J$$


So I am going to multiply equation 3 by curl. So curl of 1 by epsilon of H minus d dt curl of E is equal to 1 by epsilon Curl of J.

So for two dimensional problems let us say we will start with a simplification that we are only interested in the x y plane and your magnetic field is going to have only the z component so it is going to be a transverse electric case. So we will reduce this entire equation into more simplified form. So we will use the simplified equation so we are going from here to the form that we have got for the transverse electric case. So this equation can be written using the values of the curl. So I have the curl value which is here I am going to substitute it here and write it in the value which is only in terms of H.

(Refer Slide Time: 00:52)


FETD

$$\nabla \times E = -\mu \frac{\partial H}{\partial t} \rightarrow (1) \quad \nabla \cdot B = 0$$

$$\nabla \times H = \epsilon \frac{\partial E}{\partial t} + J \rightarrow (2) \quad \nabla \cdot D = 0$$

$$(2) \Rightarrow \frac{1}{\epsilon} \nabla \times H - \frac{\partial E}{\partial t} = \frac{J}{\epsilon} \rightarrow (3)$$

$$(3) \times \nabla \times \Rightarrow \nabla \times \frac{1}{\epsilon} \nabla \times H - \frac{\partial}{\partial t} \nabla \times E = \frac{1}{\epsilon} \nabla \times J$$

$$\nabla \times \frac{1}{\epsilon} \nabla \times H + \mu \frac{\partial^2 H}{\partial t^2} + \frac{\mu \sigma}{\epsilon} \frac{\partial H}{\partial t} = \frac{1}{\epsilon} \nabla \times J$$


So what I will have is curl of 1 by Epsilon curl of H plus Mu d square dt of H plus Mu sigma of E dt H is equal to 1 by Epsilon curl of J. So what I have done is I have used the value here

and I have exchanged the value of ϵ using the expression which is here. And what I have also done is I have made the right hand side as J and I have kept the source as the J function.

(Refer Slide Time: 11:48)

$$\frac{1}{\epsilon_r \epsilon_0} \nabla^2 H_z - \sigma \frac{\mu_r \mu_0}{\epsilon_r \epsilon_0} \partial_t H_z - \mu_r \mu_0 \partial_t^2 H_z = -\frac{1}{\epsilon_r \epsilon_0} (\nabla \times J)$$

So what we have got is $\frac{1}{\epsilon_r \epsilon_0} \nabla^2 H_z - \sigma \frac{\mu_r \mu_0}{\epsilon_r \epsilon_0} \partial_t H_z - \mu_r \mu_0 \partial_t^2 H_z = -\frac{1}{\epsilon_r \epsilon_0} (\nabla \times J)$.

(Refer Slide Time: 12:30)

$$\frac{1}{\epsilon_r \epsilon_0} \nabla^2 [H_z] - \sigma \frac{\mu_r \mu_0}{\epsilon_r \epsilon_0} \partial_t [H_z] - \mu_r \mu_0 \partial_t^2 [H_z] = -\frac{1}{\epsilon_r \epsilon_0} (\nabla \times J)$$

$$[T] \partial_t^2 P + [B] \partial_t P + [G] P + [F] = 0$$

$P = \text{coefficient vector } H_z, J$

So this is a simple equation where the coefficients have to be assigned so what we can write as this is the second order equation in time. So what we see is the second order equation in time so I can write it as some T matrix multiplied by let us say the second order equation in v plus some B matrix multiplied by first order derivative of v with respect to t plus some G matrix into v itself plus some forcing function which is F equal to 0. So this entire expression can be

written in this form where v is same coefficient vector. So what you see here is this p is actually the coefficient vector for H_z we are using this as a coefficient matrix.

So what you can see here is these are the values and F is going to be directly given for the value of J . So J is the forcing function and for which you have F . So what we have written is basically an equation which is second order in time derivative and we have written it in this form obviously we have not said anything about T , B , G and F .

(Refer Slide Time: 14:07)

$$[T] \frac{d^2 P}{dt^2} + [B] \frac{dP}{dt} + [G]P + [F] = 0$$

$P = \text{coefficient vector } H_z, J$

$$T_{ij} = \iint \frac{\mu r}{c^2} w_i w_j dx dy$$

So we will see the value of T which is in this equation is going to be given by a second surface integral so it is in two dimension so you have the x and y integral of μr by c square $w_i w_j dx dy$. So w_i and w_j are the weighting function in i and j . So in x and y direction respectively. Similarly what we are going to have is we are going to have the value for B_j which is in this part is going to be the equation for this one which is given by the double integral. So one for x and one for y it is going to be μr divided by C Square $(2j \text{ omega } c \text{ plus alpha}) w_i w_j dx dy$. Similarly we can also have the expression for j so on and so forth.

(Refer Slide Time: 15:11)

$$T_{ij} = \iint_{xy} \frac{\mu_r}{c^2} w_i w_j dx dy$$

$$B_{ij} = \iint_{xy} \frac{\mu_r}{c^2} (z_j w_c + d) w_i w_j dx dy$$

$$G_{ij} = \iint_{xy} \frac{1}{\epsilon_r} \nabla w_i \cdot \nabla w_j - \frac{\mu_r}{c^2} (w_i^2 - j\omega w_c) w_i w_j dx dy$$

$$F_i = \iint_{xy} \frac{1}{\epsilon_r} w_i \cdot (\nabla \times j_z) dx dy$$

$w_i = -2D$

So what we have done is we have basically given the expressions for individual components of those matrices.

(Refer Slide Time: 15:20)

$$\frac{1}{\epsilon_r \epsilon_0} \nabla^2 [H_z] - \frac{\mu_r \mu_0}{\epsilon_r \epsilon_0} \partial_t^2 [H_z] - \mu_r \mu_0 \partial_t^2 [H_z]$$

$$= -\frac{1}{\epsilon_r \epsilon_0} (\nabla \times J)$$

$$[T] \partial_t^2 P + [B] d_t P + [G] P + [F] = 0$$

$P =$ coefficient vector H_z, J

So the components are basically given from this particular equation and they are here.

(Refer Slide Time: 15:28)

$$T_{ij} = \iint_{xy} \frac{Mv}{c^2} w_i w_j dx dy$$

$$B_{ij} = \iint_{xy} \frac{Mv}{c^2} (z_j w_c + \alpha) w_i w_j dx dy$$

$$G_{ij} = \iint_{xy} \frac{1}{\epsilon_r} \nabla w_i \cdot \nabla w_j - \frac{Mv}{c^2} (w_i^2 - j\omega w_c) w_i w_j dx dy$$

$$F_i = \iint_{xy} \frac{1}{\epsilon_r} w_i \cdot (\nabla \times jz) dx dy$$

$\alpha = \sigma / \epsilon_r \epsilon_0$ $c = \text{vel. wave}$ $w_{ij} = 2D \text{ basis funt}$

There are some coefficients which we have used alpha which is equal to sigma divided by epsilon r epsilon 0 we have used the terms c which is the velocity of the wave and obviously w i and w j are the 2 D basis functions which we have used.

(Refer Slide Time: 15:48)

$$\frac{1}{\epsilon_r \epsilon_0} \nabla^2 [H_z] - \sigma \frac{Mv \mu_0}{\epsilon_r \epsilon_0} \partial_t [H_z] - Mv \mu_0 \partial_t^2 [H_z] = - \frac{1}{\epsilon_r \epsilon_0} (\nabla \times J)$$

$$[T] \partial_t^2 P + [B] \partial_t P + [G] P + [F] = 0$$

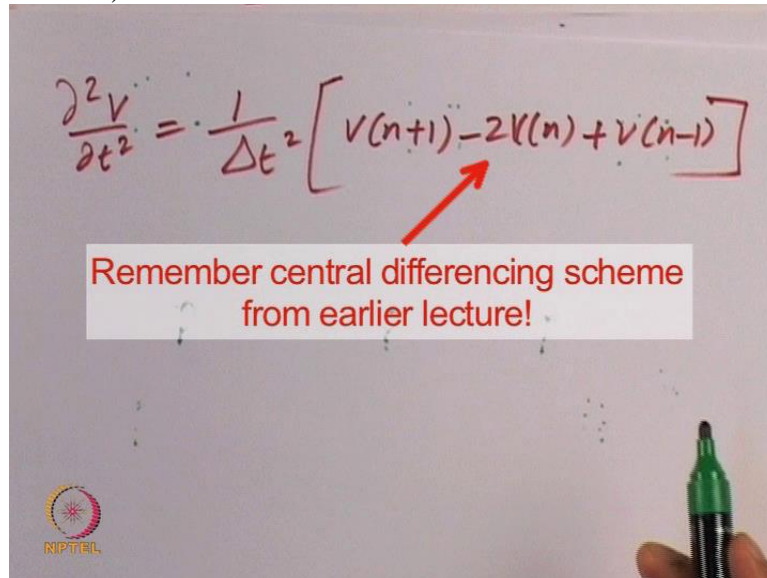
$P = \text{coefficient vector } H_z, J$

So once we have these integrals we can compute the individual matrices for T, B G and F and plug it in so you will have a simple form which is given here. One thing that is still not complete is, we still have a second order derivative in time first order derivative in time. So we need to find a way to manipulate them. So that is what we are going to do but for that we are already in safe hands because,

(Refer Slide Time: 16:12)

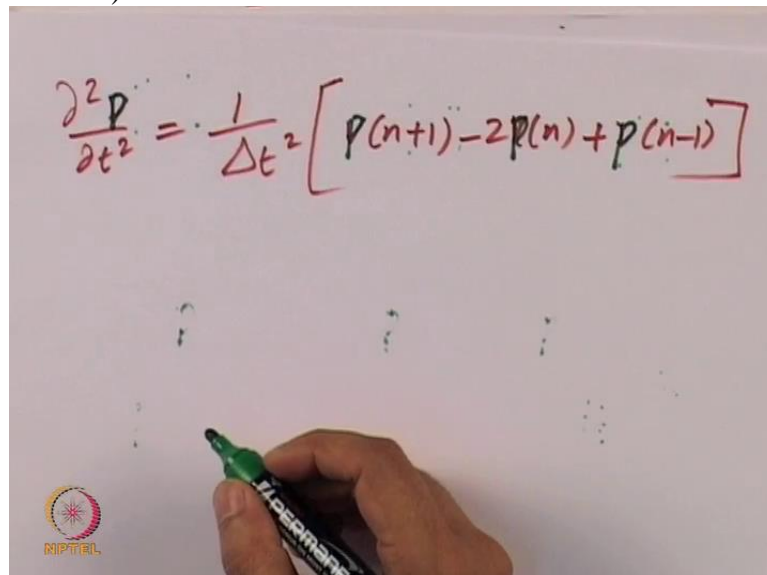
$$\frac{\partial^2 v}{\partial t^2} = \frac{1}{\Delta t^2} [v(n+1) - 2v(n) + v(n-1)]$$

Remember central differencing scheme from earlier lecture!



We have an expression for a particular function let us say a Phi is equal to,

(Refer Slide Time: 16:15)

$$\frac{\partial^2 p}{\partial t^2} = \frac{1}{\Delta t^2} [p(n+1) - 2p(n) + p(n-1)]$$


So we are computing the value of the second order derivative based on this particular expression. So what we have is for second order derivative Δt^2 so we are taking the value that is on the forward and the reverse and the central point. This stencil is something which we have already looked into in our earlier lecture on the finite difference.

(Refer Slide Time: 16:50)

$$\frac{\partial^2 P}{\partial t^2} = \frac{1}{\Delta t^2} [P(n+1) - 2P(n) + P(n-1)]$$
$$\frac{\partial P}{\partial t} = \frac{1}{2\Delta t} [P(n+1) - P(n-1)]$$

So we can also see that we can write the first derivative of t is equal to $\frac{1}{2\Delta t} [P(n+1) - P(n-1)]$. So what I have done is I have taken a central differencing between $[P(n+1) - P(n-1)]$ so that is why we have $2\Delta t$ here. I can also do some kind of in between waiting functions.

(Refer Slide Time: 17:20)

$$\frac{\partial^2 P}{\partial t^2} = \frac{1}{\Delta t^2} [P(n+1) - 2P(n) + P(n-1)]$$
$$\frac{\partial P}{\partial t} = \frac{1}{2\Delta t} [P(n+1) - P(n-1)]$$
$$P(n) = \beta P(n+1) + (1 - 2\beta) P(n) + \beta P(n-1)$$

$\beta = \frac{1}{4}$ good choice

So I can say $P(n)$ is equal to some value β multiplied by $P(n+1)$ plus $(1 - 2\beta) P(n)$ plus $\beta P(n-1)$. So what I have done is I have taken $P(n+1)$, $P(n)$ and $P(n-1)$ and I am using some functions of β . In the earlier lectures we used the term r or some of the ways in which we can compute r is also given by the PDE itself. One good approach is we can choose β is equal to $\frac{1}{4}$ so this is a good choice for numerical modelling. So if you choose β is equal to $\frac{1}{4}$.

(Refer Slide Time: 18:23)

$$\frac{1}{\epsilon_r \epsilon_0} \nabla^2 [H_2] - \frac{\mu_r \mu_0}{\epsilon_r \epsilon_0} \partial_t [H_2] - \mu_r \mu_0 \partial_t^2 [H_2]$$

$$= -\frac{1}{\epsilon_r \epsilon_0} (\nabla \times J)$$

$$[T] \partial_t^2 P + [B] \partial_t P + [G] P + [F] = 0$$

$P = \text{coefficient vector } H_2, J$

Our expression which is here should be changed into a simple more easily approachable form so we are going to write it in a easier form as follows.

(Refer Slide Time: 18: 38)

$$\left(\frac{[T]}{\Delta t^2} + \frac{[B]}{2\Delta t} + \frac{[G]}{4} \right) P(n+1)$$

$$= \left(\frac{2[T]}{\Delta t^2} - \frac{[G]}{2} \right) P(n)$$

$$+ \left(\frac{-[T]}{\Delta t^2} + \frac{[B]}{2\Delta t} - \frac{[G]}{4} \right) P(n-1)$$

$$- [F]$$

So what we will have is [T] divided by delta t square plus [B] divided by 2 delta t plus [G] divided by 4 the entire thing multiplied by p(n plus 1) is equal to (2[T] divided by delta t square minus [G] divided by 2) multiplied by P (n) minus [T] divided by delta t square plus [B] divided by 2delta t minus [G] divided by 4) into P (n minus 1) minus the forcing function [F]. So this is nothing but what I have got in this equation. So I have taken this equation and I have substituted the value of time stepping using the approach what I have explained in the time stepping algorithm. So I have chosen beta is equal to 1 by 4.

And accordingly what I have got is an expression for the value of the update equation. So it is logical to see wherever there are

(Refer Slide Time: 20 :30)

$$[T] \frac{d^2 P}{dt^2} + [B] \frac{dP}{dt} + [G] P + [F] = 0$$

$P = \text{coefficient vector } H_z, J$

$$\left(\frac{[T]}{\Delta t^2} + \frac{[B]}{2\Delta t} + \frac{[G]}{4} \right) P(n+1) = \left(\frac{2[T]}{\Delta t^2} - \frac{[G]}{2} \right) P(n) + \left(\frac{-[T]}{\Delta t^2} + \frac{[B]}{2\Delta t} - \frac{[G]}{4} \right) P(n-1)$$

FETD
Maxwell
Eqn

[T] The value is the central differencing which is basically the second order derivative of t is here. And then for B you have two T values and you have the value for G which is given by 1 by 4 and they are all multiplying on the time stepping n plus 1 . So what you have got in the left hand side of this equation is nothing but the update equation for n plus 1 th step based on the values of P which is (n) and $(n$ minus $1)$ steps. So it is an explicit model to compute the value of the equation. Obviously there is some tradeoffs here you have to compute these matrices $[T]$, $[B]$, $[G]$ and $[F]$ separately. But it is going to create a lot of computational effort but this is the way you can model the finite element time domain for Maxwell equations.

We will look more into this in the exercises but at least now what you have got is a basic understanding of how to get into the Maxwell equations in the finite element method for explicit time stepping algorithm. We have done a lot of mathematics here in fact too much mathematics it is good that you at least know step by step how the process of doing update equations look like. I did not want to just stop at the point of saying that this is the update equation you can just feed it into a Matlab code and do it.

Because the devil is always in the detail. So I wanted to give you step by step approach on get into the point where you can use Matlab to compute obviously when you do coding yourself, you will inevitably come into a lot of challenges where you have to find ways to do it. So with that we have come to an end of this particular module and also the chapter of finite element itself. We have looked into the variational method in general, we looked into Rayleigh Ritz method we also looked into the method of weighted residuals and also into the method of Galerkin and we also looked at various aspect of finding variational principle or the action of a particular PDE and how to go backward to a PDE to action function. So in that

sense we have covered quite a bit of work on the variational method including finite element approach. We have seen also some examples that can demonstrate how the method actually can be executed using commercial packages like Matlab.

So with that being said I would like to thank you for listening to this module and we will look into the method of moments and applications of method of moments in the next modules to come. So thank you and see you in the next module. Bye Bye!