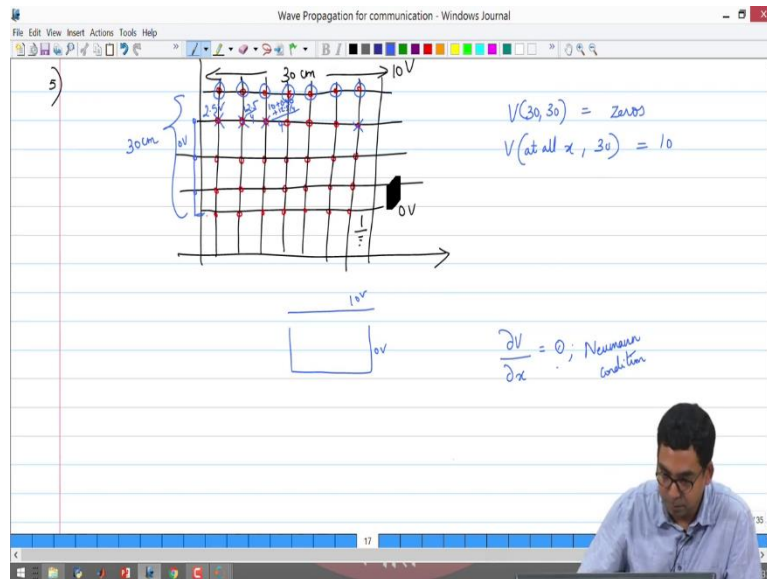


Transmission Lines and Electromagnetic Waves
Prof. Ananth Krishnan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture - 04
Octave simulation of wave equation

(Refer Slide Time: 00:15)



Ok We will begin where we left off, right. The last time we had this structure where we had a 10 volts plate on the top and we had a 0 volts plate at the bottom and we wanted to just use the a difference method to solve for the voltage, right So, we began with the algorithm, but did not have time to go over it completely. So, I will go over the algorithm fully and then we will start an octave code to solve for the voltage in the capacitor.

So, what we knew last time was that the top plate was at 10 volts, so the plate here. These points were at 10 volts. The remaining points are not known to us. The bottom plate is going to be at 0 volts. We have divided this region into a number of x y grid uniformly sampled, right and we started to begin, all right.

So, what we would do is we will come to the first point where we do not know the value of the voltage. It was originally 0, but we look at the neighbouring points 10 volts on the top. 0 volts is what we had initialized to the right. 0 volts at the bottom and on the left hand side there was no point. We assume that if we are not aware of the voltage at any point we are going to make it 0, all right

So, the left hand side since there is no point we make it 0 volts and this particular boundary condition is known as a Dirichlet boundary condition. It is a special boundary condition that

we are using here that the boundary is going to be having the unknown value equal to 0. So, it became $10 + 0 + 0 + 0$ divided by 4. So, we mark 2 and a half volts at this point.

Then we went to the second point which where we needed to estimate the voltage we started at this point, so that will be 10 on the top plus 2.5 plus 0 plus 0 divided by 4 and we keep going to the next point, so here it will be $10 + 0 + 0 + 12.5$ divided by 4, the whole thing divided by 4. And we keep going along the row. And once you finish that particular row on the right extreme, once again we do not know the value of the voltage to the right side because there is no point over there. So, if there is an unknown voltage we consider that to be 0, right

Once you finish the first row you will go to the second row, you will keep going to the third row and you will keep going till you reach the last unknown point. But while we are doing this something has happened when we began computing the voltage at the first point mark 2 and a half volts over here the right hand side point over here was at 0 volts, that is how you arrived at $10 + 0 + 0 + 0$ divided by 4. But now it has a voltage of 12.5 divided by 4, so you began with something, but you ended up with something else, all right. So, we need to rectify this

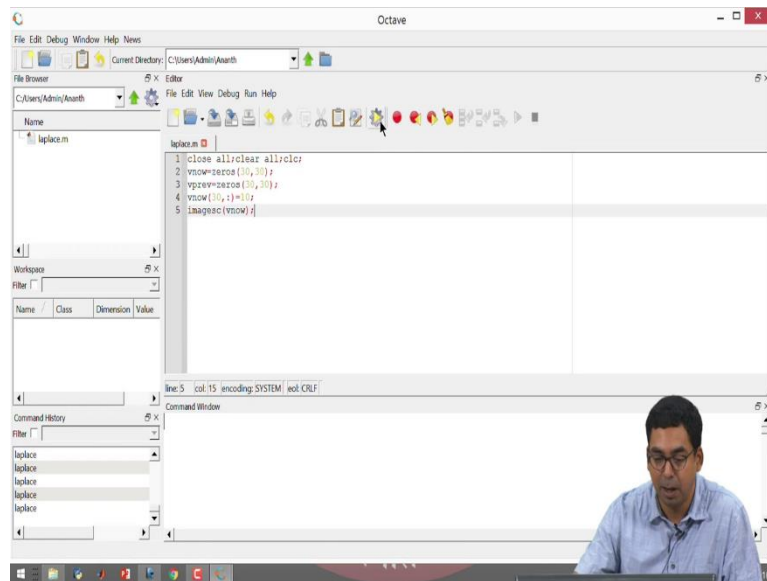
So, now what the algorithm will do is when you go from top to bottom for all the unknown points we will say that we have finished one iteration of the calculation, all right. So, we will go to the second iteration you will start over here you will once again compute the average of all the neighbouring points. So, we will have $10 + 12$ and a half divided by 4 plus whatever you had computed in the previous iteration. Left hand side there is no point, so we are going to assume it has 0 and divided by 4.

And you keep doing this across all the rows, all right going from top to bottom that we constitute your second iteration. Once again you have to keep going to the top and start doing this all over again, and you keep repeating this till between successive iterations the value of the voltage at a given point does not change significantly, ok.

Now, how do we determine what is significant and what is insignificant? Well, it depends on the application. If you wanted to know the voltage distribution inside the capacitor to some precision say that you need to know about 50 millivolts or 500 millivolts that the accuracy you need, you do not need more than that then you set that as the error. If you want extremely precise calculation then you say that let the error be very small like 1 nanovolt extra, all right.

Now, as you said the error between consecutive iterations to be smaller and smaller. We know that the number of iterations that you need to perform will become larger and larger and larger, ok. So, that is the downside that the other thing is it is perfectly controllable, right. I hope you are able to follow the algorithm now and we will just fire octaves on the computers and write the programmatic version for whatever we have done now, right.

(Refer Slide Time: 05:07)



So, I will start octave. I begin all my programs with the close all, clear all and clc command, all right. Close all will close all active figure windows and any other windows that have popped up in the previous run. Clear all will clear the variables in the workspace. So, there will be no variables in the workspace and the memory will be freed. Clc will just clear off my display buffer or the command window over here. So, I start all my octave programs like this. You could do the same, right.

Now, I need to define my voltage matrix, all right. v is going to be zeroes of 30 comma 30. What this does is it creates a two-dimensional matrix filled with zeroes, all right and the size of the matrix is going to be 30 rows by 30 columns, ok.

Now, one of the things that we saw in the algorithm was as you start computing the value of the voltage you have to come you have to compare the current value that you have calculated to the previous value that you had calculated in the previous iteration; that means, you need to store the current and the previous value, all right. So, I am going to make a distinction on the voltage. I am going to call this as v now; this means that the voltage that I am calculating now in the current iteration is going to be stored in v now, all right. And I am going to create one more voltage variable, all right, v previous, all right.

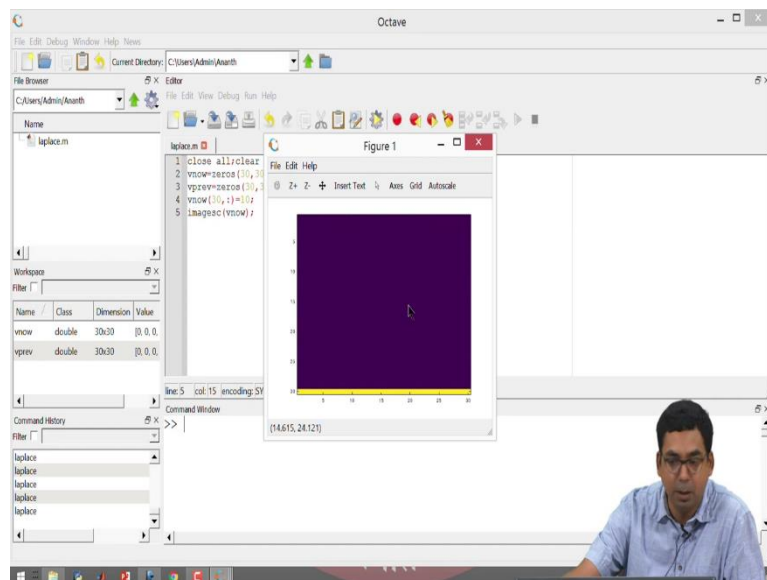
Since I have zeroes for all the values a I mean all the rows and the columns, this made its zeroes of 30 comma 30, ok. So, I will just run it for the sake of completeness. So, on the left hand side I am having a workspace which tells me that I have two variables defined v now and v previous, dimension is 30 by 30 and I can check the values and it is going to be zeroes for all the rows and all the columns, ok.

Now, I want to set the voltage at the top of this capacitor configuration to be equal to 10 volts, ok. So, I am going to say that v now of. What this one does is it takes the declared

matrix from line number 2, takes the 30th row of that matrix, colon represents all columns in that particular row and I am going to set that to value 10, all right. So, just missed an equal to sign. So, v now of 30 comma colon is going to be equal to 10, all right. So, the 30th row of all columns is going to be set at some value 10, all right.

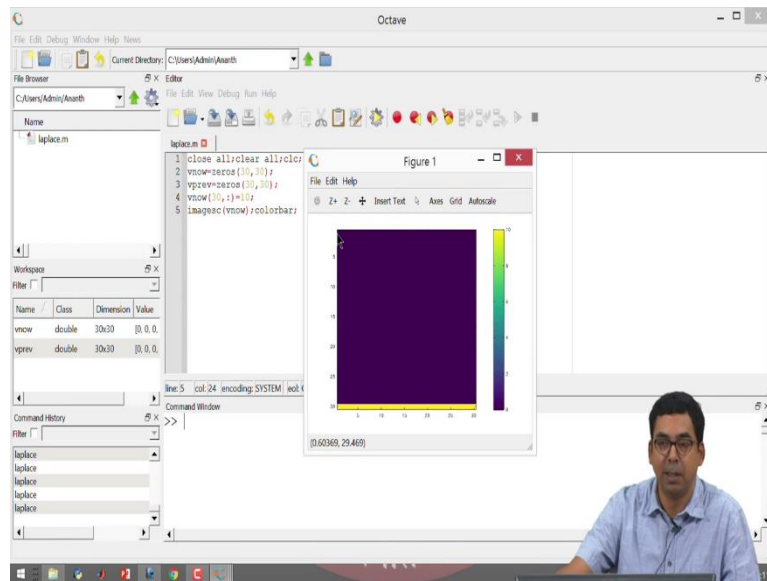
Just to see what we are doing at this stage, I will just use a command to see visually what we are doing. Imagesc is a command that will allow you to look at a surface that you are defining. So, in this case we have a two-dimensional matrix and I want to look at the values held by the matrix. So, imagesc is nothing but an image scale. It will give you an image. It is a 2D matrix and it will scale the colour corresponding to the value stored in the matrix. So, as we run it you will be able to understand. So, I will just go to this button save file and run and I will run, ok. I have this, all right.

(Refer Slide Time: 08:45)



Now, there are a few things that we need to look at, all right. There is some yellow colour over here and there is some dark colour on the remaining portion. I need to understand what colour means what, ok.

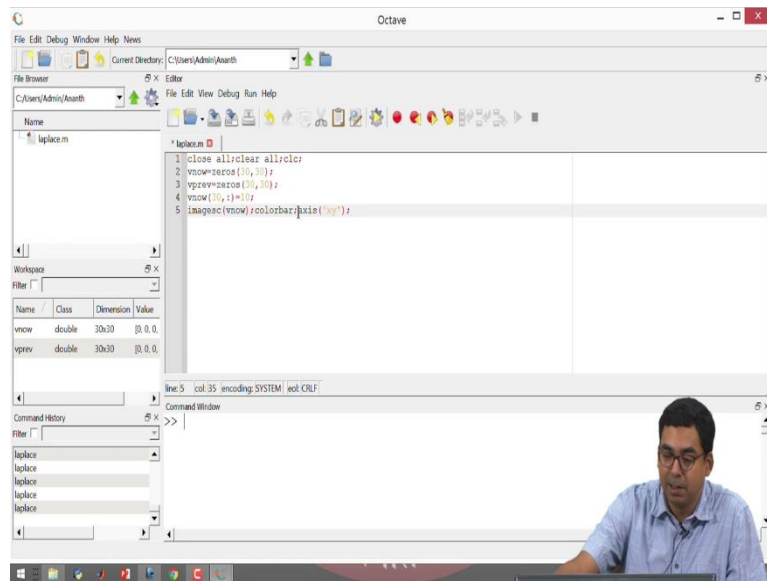
(Refer Slide Time: 09:01)



So, I am going to go back to this code and place a command called colour bar. This will allow me to figure out the values corresponding to different colours. On the right hand side, I have now a colour bar appearing saying that the yellow colour corresponds to value 10 and the darker colour corresponds to value 0, ok.

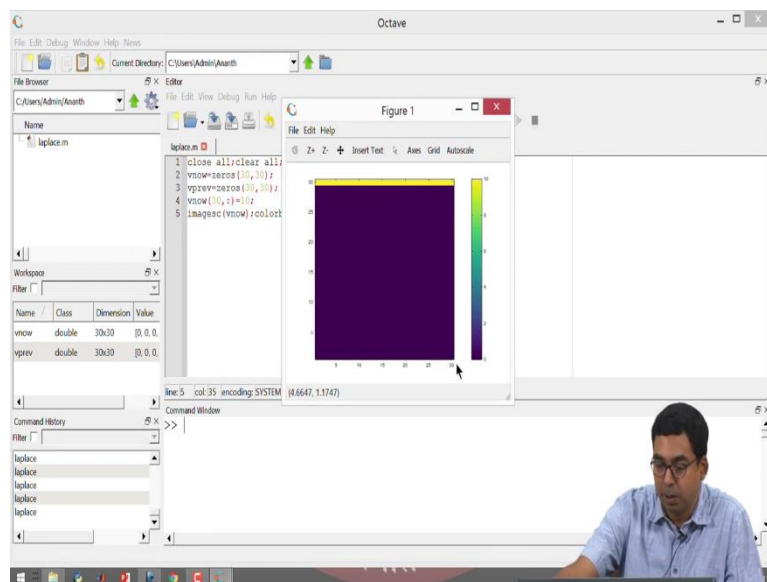
The other thing that we notice over here is that the axis is very strange. The x axis goes from 0 to 30 and the y axis goes from 0 to 30, but from top to bottom, it is not the conventional way you draw the axis. So, the 30th row has come down. I do not, I have a problem in visualizing this because the way we have drawn the diagram in the class is different and the way we are used to drawing axes is different. Simply because in image processing this is a conventional way of representing axes, so we will go for some normal access so that we can make a one to one correlation.

(Refer Slide Time: 09:51)



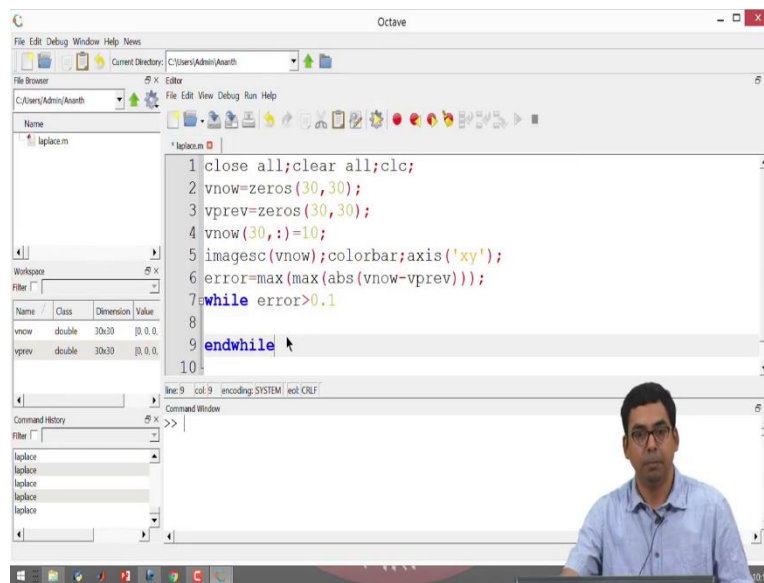
So, I will just say in my command axis x y, it says that octaves can draw the axis in a regular x y graph, right.

(Refer Slide Time: 10:03)



So, if I go now I have the line flip. So, I have 0 to 30 on the x axis, 0 to 30 on the y axis, top row is set to 10 volts, right. Now that I have my preliminary setting clear I need to write the code for solving the Laplace equation that is all, all right.

(Refer Slide Time: 10:29)



```
1 close all;clear all;clc;
2 vnow=zeros(30,30);
3 vprev=zeros(30,30);
4 vnow(30,:)=10;
5 imagesc(vnow);colorbar;axis('xy');
6 error=max(max(abs(vnow-vprev)));
7 while error>0.1
8
9 endwhile
10
```

Name	Class	Dimension	Value
vnow	double	30x30	[0 0 0]
vprev	double	30x30	[0 0 0]

So, let us go ahead and let us say that I am going to open a while loop and I am going to say, ok. So, this error that I am writing down here is going to be the difference between v now and v previous. It is going to be greater than 0.1. I want to start a while loop, but before that I need to define what this error is going to be, all right.

So, the error that I wish to calculate is actually the difference between two matrices v now and v previous. The difference between the two matrices will also be a matrix, ok. So, now, we have to figure out whether we want to take each and every point and compare or whether we make some maximum error admissible in the entire difference matrix that you have got. So, the simplest way to do it is you take the difference between two matrices and you take the maximum element.

But now the question is since you are having v now minus v previous, it is quite possible that between the iterations there is a sign change, maximum will capture only the largest positive number, it is also possible that you have a large negative number which is coming as a difference between the two matrices. So, we have to deal with the absolute value of the difference, right.

In order to do this I am going to say max, ok. Now, this is the slightly weird part of max. Max of max means that, usually when we say max it means that it accepts a vector one-dimensional vector and it picks up the maximum element present in that one-dimensional vector. When you have a two-dimensional matrix, I have to use an octave max of max to find out the maximum in each column and maximum in each row, all right and then give me a single value, ok.

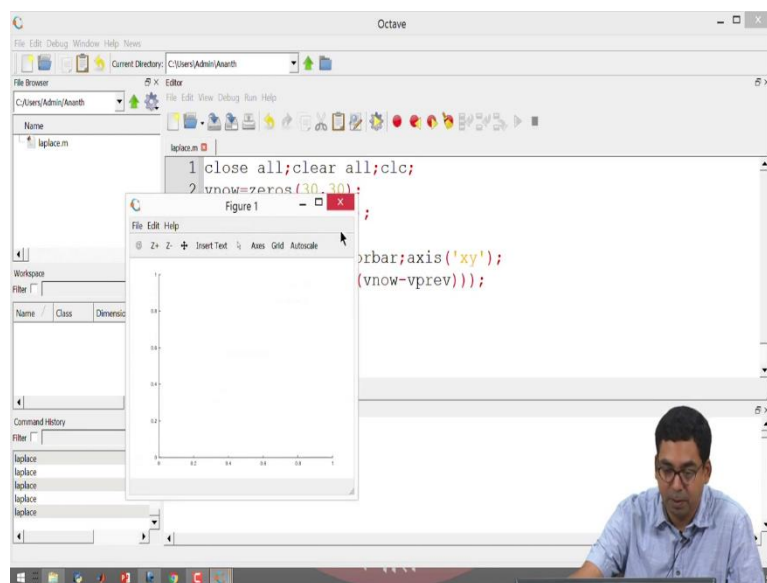
So, max of max of and I want to make sure that I am capturing both the positive and the negative signs. So, I am going to write down the absolute value of v now minus v previous. Are you guys able to see what is being written?

Student: No.

No. Now, all right. So, I have a variable error defined as max of max of absolute value of v now minus v pre. v now is a matrix, v pre is a matrix, the difference between the two will give you a matrix. And I am going to use only the absolute value of all the elements, so I have abs, all right which will remove the negative signs and then I am going to take the max of max, right that is how it says that we get the single maximum element from the difference.

So, while this error is greater than 0.1; that means, if I start from line number 1 and progress towards line number 6 which defines the error, in order for me to write anything within this the error has to be greater than 0.1, right now it is, because I have made v now of 30 comma colon line number 4 to be equal to 10, ok.

(Refer Slide Time: 14:05)



So, if I just run this program, ok, so it is an infinite loop because error is continuously greater than 10, does not matter, all right.

(Refer Slide Time: 14:5)

```
4 vnow(30,:)=10;
5 imagesc(vnow);colorbar;axis('xy');
6 error=max(max(abs(vnow-vprev)));
7 while error>0.1
8   for i=1:30
9     for j=1:30
10      vnow(i,j)=(vnow(i+1,j)+vnow(i-1,j)+vnow(i,j+1)+vnow(i,j-1))/4;
11    endfor
12  endfor
13 endwhile
```

line:10 col:63 encoding:SYSTEM seek CRLF

>> error: close: first argument must be "all", a figure handle, or a figure object
error: parse error
error: called from
close at line 81 column 7

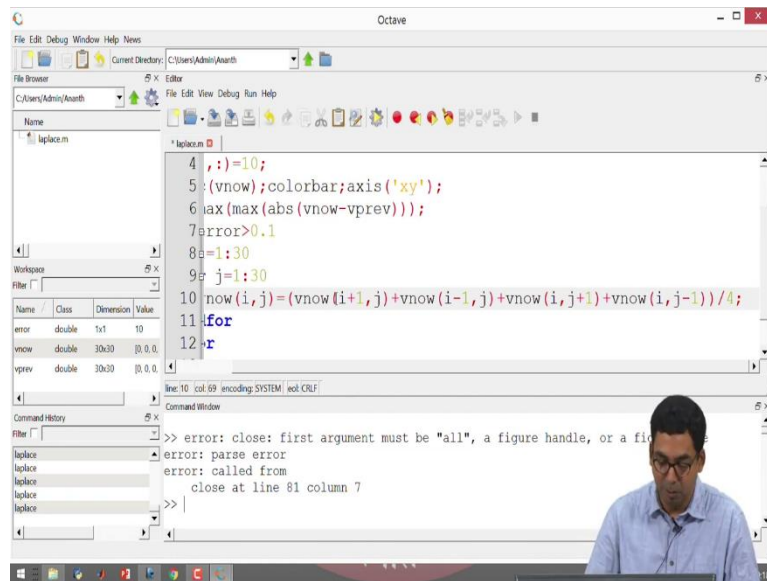
So, error is equal to 10, all right, so it will enter this loop and I would like to write the code for calculating the voltages within this loop, ok.

Now, for doing this calculation I am going to go across each row, each column and write down the average of the neighbouring points for each of the points that I am wishing to calculate the voltage, ok. So, I am going to start with for i equal to 1 to 30, ok and for j equal to 1 to 30. The syntax here is if I put one colon 31 signifies the starting point for that for array a variable or i, and 30 signifies the end point and if I do not specify anything else it means that will going increments of one, so it will be starting with i equal to 1, the next round it will be going i equal to 2, next round it will be going i equal to 3, so on and so forth.

If I want to specify specific increments we can always do that, but in this course we are not going to be using any specific increments, always it is going to be units of 1, it is going to be integer, ok.

Now, I would like to write down the Laplace solver that we have already arrived at. So, I would like to take the value of the voltage at i comma j which is the unknown quantity and I would like to say that this is going to be equal to v now of i plus 1 comma j plus v now of i minus 1 comma j plus v now of i comma j plus 1 plus v now of i comma j minus 1, the whole thing divided by 4.

(Refer Slide Time: 16:11)



The screenshot shows the Octave IDE interface. The main editor window contains the following code:

```
4  ,:)=10;  
5  :(vnow);colorbar;axis('xy');  
6  (ax(max(abs(vnow-vprev))));  
7  error>0.1  
8  =1:30  
9  j=1:30  
10 now(i,j)=(vnow(i+1,j)+vnow(i-1,j)+vnow(i,j+1)+vnow(i,j-1))/4;  
11 for  
12 r
```

The Command Window at the bottom displays the following error message:

```
>> error: close: first argument must be "all", a figure handle, or a figure  
laplace  
error: parse error  
laplace  
error: called from  
laplace  
close at line 81 column 7  
>> |
```

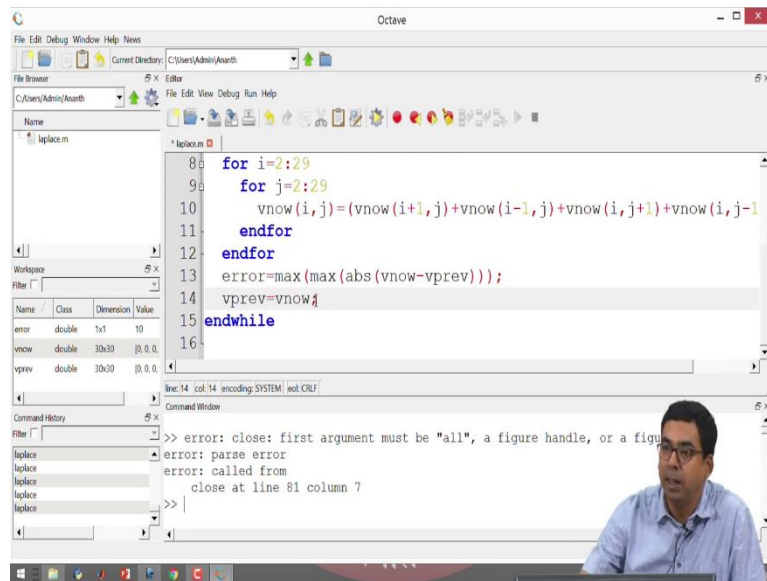
A person's head and shoulders are visible in the bottom right corner of the screenshot, looking at the screen.

So, very simple you know calculation and you have your skeleton for your Laplace solution ready, all right. But there are some small niggling issues that we will have to fix.

If we examine this line, all right, I start the loop at I equal to 1 and then j equal to 1, all right. So, I will be calculating v now at 1 comma 1 and it is now taking v now of 1 plus 1 comma 1 is 2 comma 1, but then I have 1 minus 1 comma 1 also which is going to index 0. Unlike C or C plus plus here the starting index in Octave or MATLAB is going to be 1, so this is going to throw an array out of bounds or index out of bounds error, ok.

So, this means we need to start our loop at 2, ok. The same thing is going to happen when it reaches 30, we will try to search for 30 plus 1, comma 1. So, it will be 31, so it is going to be indexed out of your known bounds of the array. So, we have to make this 29, 2 to 29, right, ok ok.

(Refer Slide Time: 17:21)



```
8 for i=2:29
9   for j=2:29
10    vnow(i,j)=(vnow(i+1,j)+vnow(i-1,j)+vnow(i,j+1)+vnow(i,j-1));
11   endfor
12 endfor
13 error=max(max(abs(vnow-vprev)));
14 vprev=vnow;
15 endwhile
16
```

line:14 col:14 encoding:SYSTEM text CRLF

>> error: close: first argument must be "all", a figure handle, or a figure
error: parse error
error: called from
close at line 81 column 7
>>

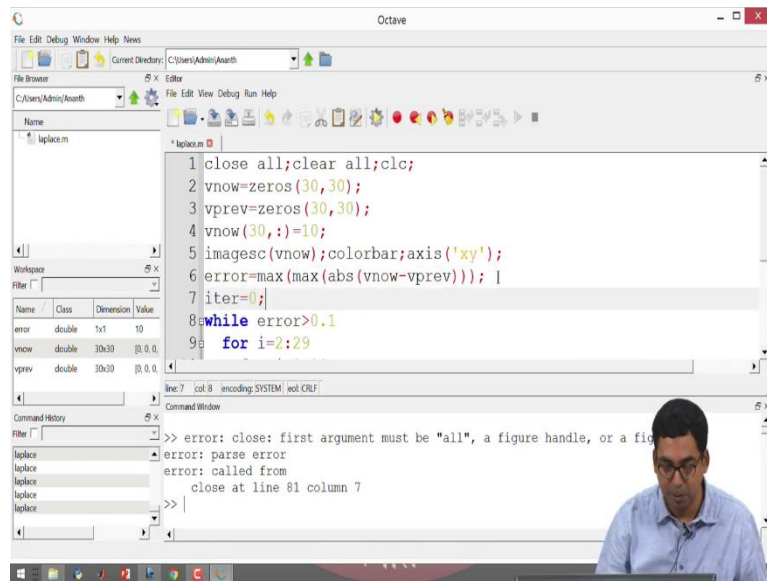
Name	Class	Dimension	Value
error	double	1x1	10
vnow	double	30x30	[0 0 0]
vprev	double	30x30	[0 0 0]

Now, at this stage what has happened is two for loops will be executed for each row and each column, all right. And once this is done, I would like to calculate the error between the current value of voltages that we have found and the previous value of voltage that has been computed prior to starting this particular iteration. So, this entire for loop, two for loops, constitutes one iteration, ok. So, we make the error calculation once again. So, I am just going to copy and paste line number 6, ok.

Once I do this it is going to remain within the loop and I have to give some idea to what the v previous is going to be. So, v previous is now going to become equal to v now because the next iteration will begin from the top once again, right, ok.

I would also like to have more information when we are doing this. I would like to know how many iterations it takes, all right. So, we can keep a counter variable.

(Refer Slide Time: 18:53)



```
1 close all;clear all;clc;
2 vnow=zeros(30,30);
3 vprev=zeros(30,30);
4 vnow(30,:)=10;
5 imagesc(vnow);colorbar;axis('xy');
6 error=max(max(abs(vnow-vprev)));
7 iter=0;
8 while error>0.1
9     for i=2:29
```

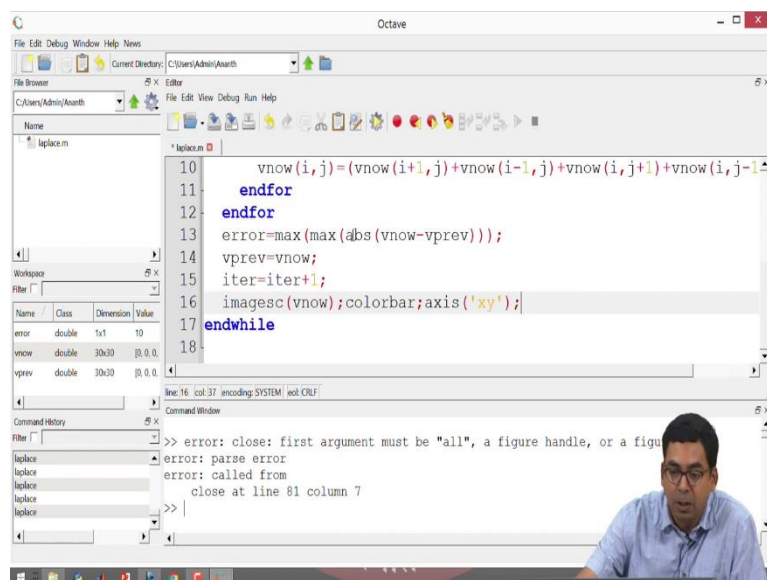
line:7 col:8 encoding:SYSTEM iec:CRLF

Command Window

```
>> error: close: first argument must be "all", a figure handle, or a figure
error: parse error
error: called from
    close at line 81 column 7
>> |
```

Before I begin the loop I can say iter is equal to 0, and each time when I finish one calculation and make v previous equal to v now I would like to increment this iteration counter, so I will say that iter is equal to iter plus 1, ok.

(Refer Slide Time: 19:03)



```
10     vnow(i,j)=(vnow(i+1,j)+vnow(i-1,j)+vnow(i,j+1)+vnow(i,j-1))/4;
11     endfor
12 endfor
13 error=max(max(abs(vnow-vprev)));
14 vprev=vnow;
15 iter=iter+1;
16 imagesc(vnow);colorbar;axis('xy');
17 endwhile
18
```

line:16 col:37 encoding:SYSTEM iec:CRLF

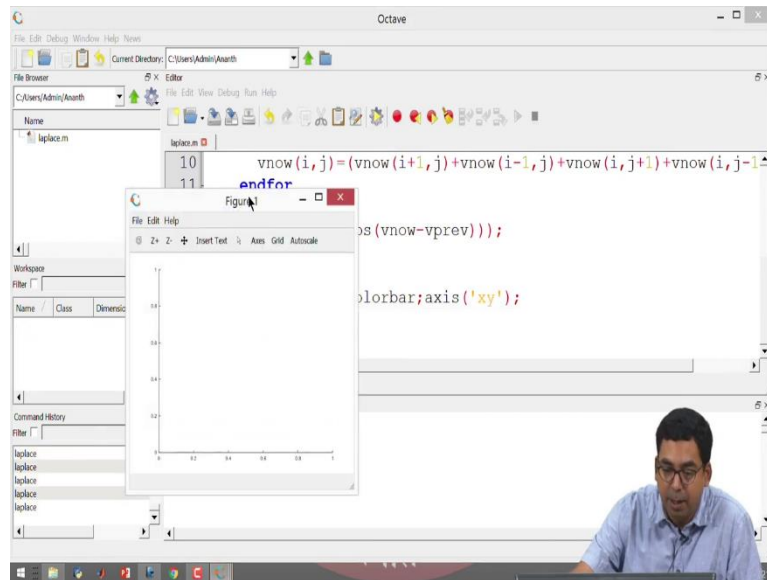
Command Window

```
>> error: close: first argument must be "all", a figure handle, or a figure
error: parse error
error: called from
    close at line 81 column 7
>> |
```

Now, when you go from top to bottom within one a while loop as you know within one iteration you would have calculated the value of the voltages at all the points and I want the system to display what has been calculated in every iteration, ok. So, I am going to have a

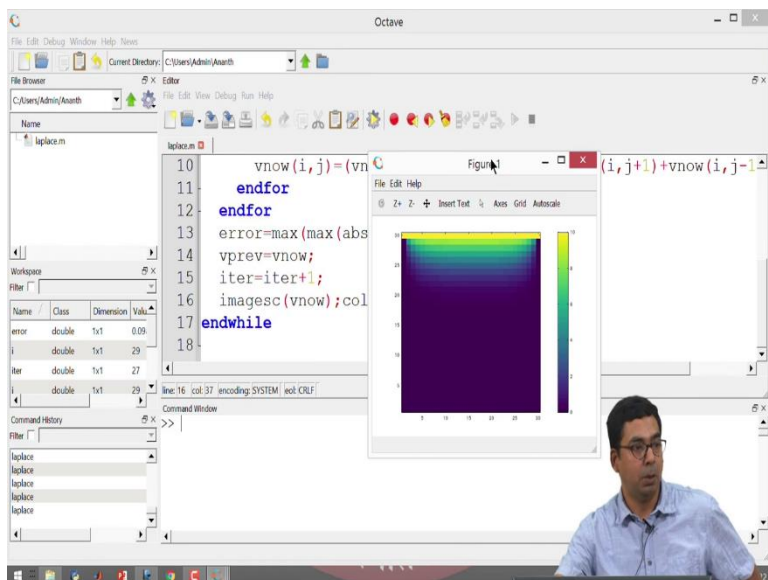
command over here which will allow me to do that, right. So, I am going to have imagesc of v now. So, this is the command that we have over here, I am going to cut this, I do not need it there, right and I am going to simply paste it over here, ok. I am going to run this once.

(Refer Slide Time: 20:05)



I immediately noticed that there is a problem.

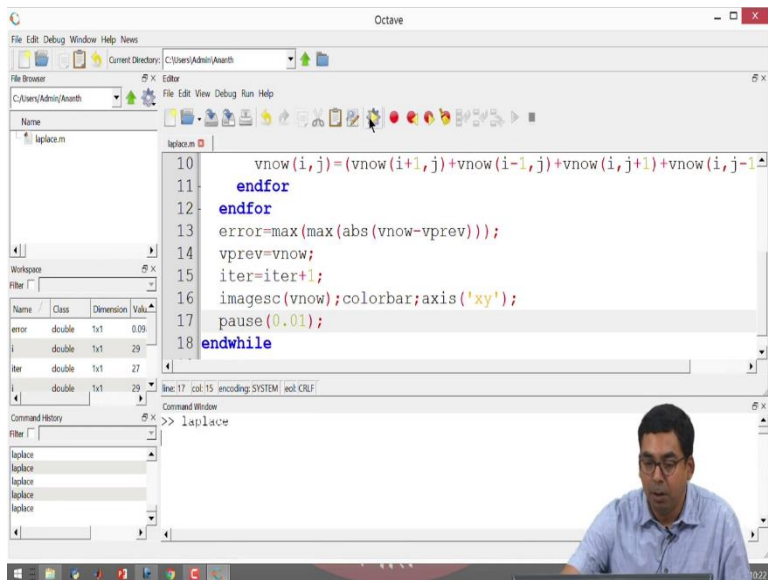
(Refer Slide Time: 20:13)



I get the value of the voltages that have been calculated only after all the iterations have finished, all right.

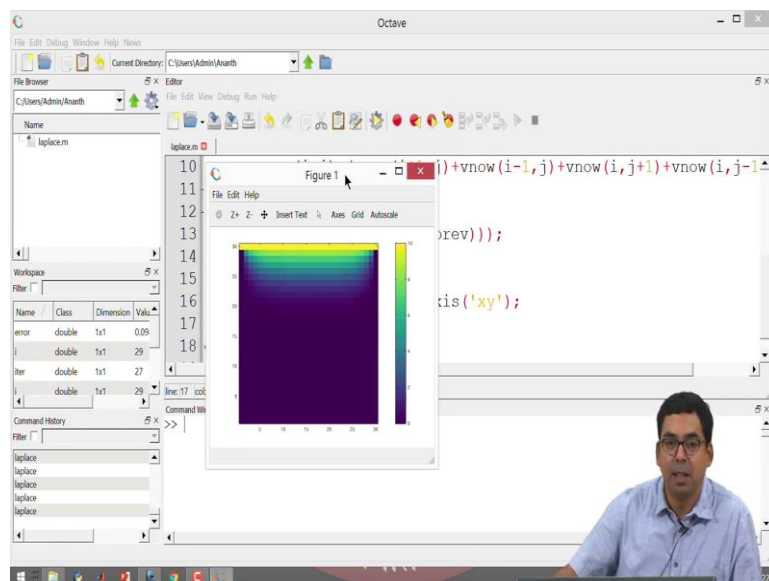
This is a problem because the time it takes to refresh the screen is larger than the time it takes to go for the next iteration. So, the CPU throws out the command to draw the picture on the screen, but before it attempts to draw it has gone to the next iteration and it has completed, so it tries to keep on the display buffer and tries to keep on with what is happening with the CPU. So, we want to slow this down, all right.

(Refer Slide Time: 20:43)



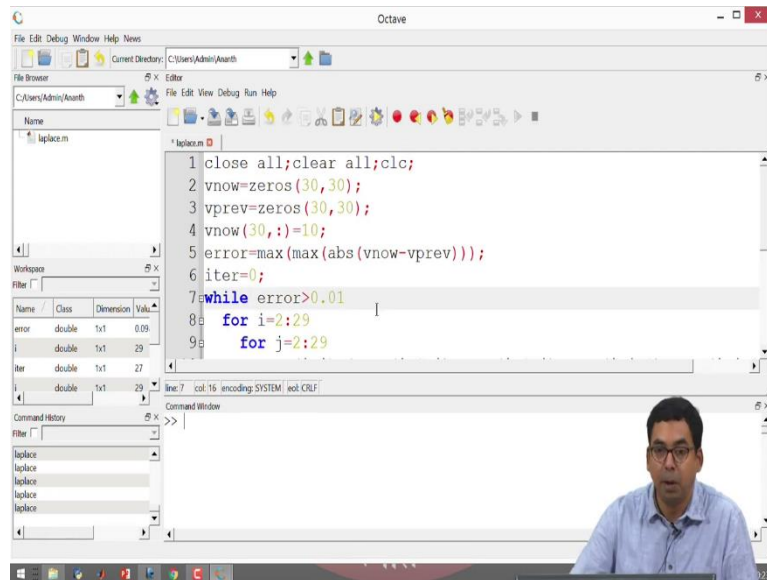
So, we will say that after every iteration, pause for a small amount of time, so that I can have a look at what is going on. So, I am just going to put that pause 0.01. I do not remember the units of this pause command, maybe it is in seconds, maybe it is in milliseconds, but it seems to work, ok. So, going to go ahead do this.

(Refer Slide Time: 21:03)



Now, I have something happening, all right, ok. There have been some calculations done, all right. However, I am not very satisfied because it is too little. I want to see the voltages that have been calculated you know fairly more accurately.

(Refer Slide Time: 21:23)

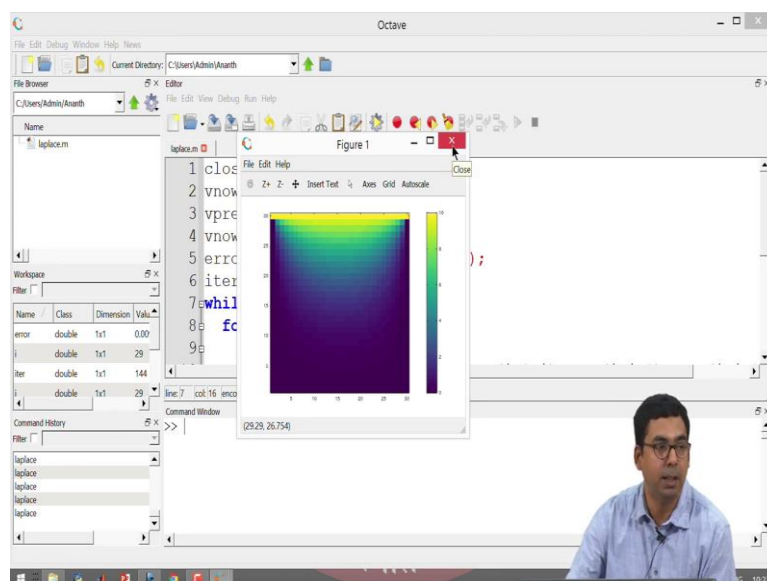


```
1 close all;clear all;clc;
2 vnow=zeros(30,30);
3 vprev=zeros(30,30);
4 vnow(30,:)=10;
5 error=max(max(abs(vnow-vprev)));
6 iter=0;
7 while error>0.01
8     for i=2:29
9         for j=2:29
```

Name	Class	Dimension	Value
error	double	1x1	0.09
i	double	1x1	29
iter	double	1x1	27
vnow	double	30x30	29

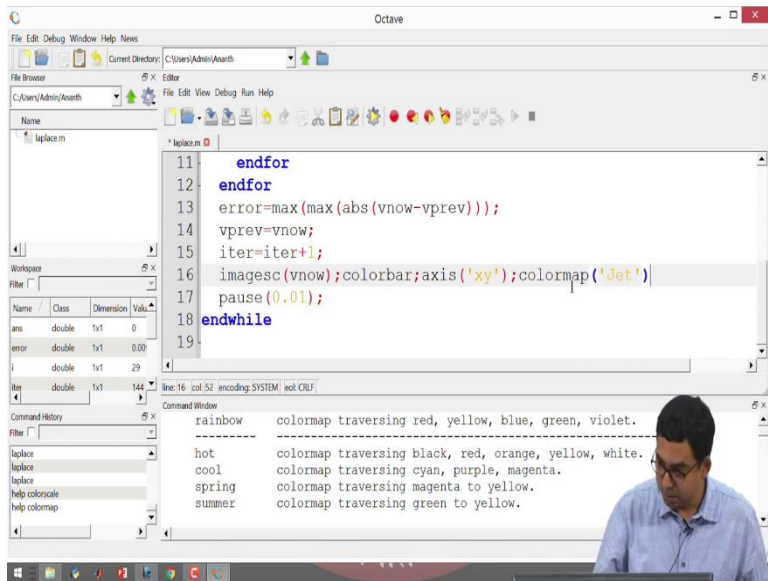
So, I am going to go ahead and say that while the error is greater than 0.01 is reducing the error by a factor of 10, right. So, I want to watch this slowly as what happens.

(Refer Slide Time: 21:31)



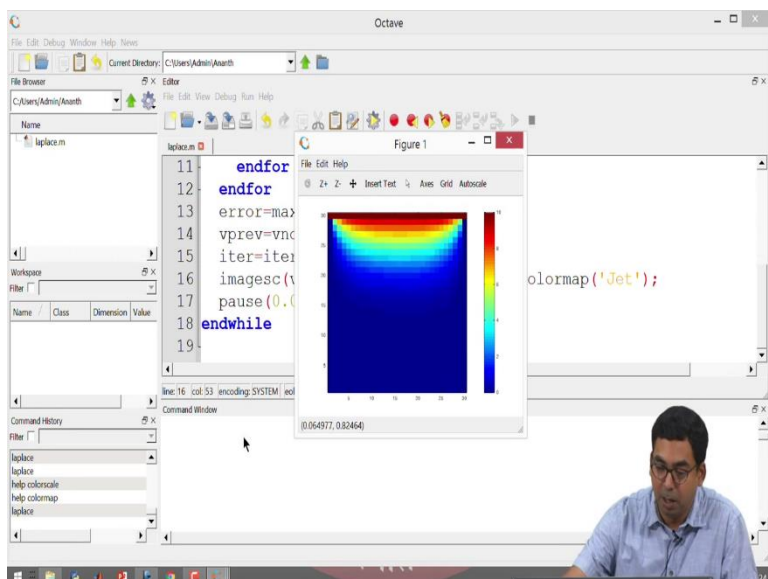
The moment I get my cursor back in the command window I know that the calculation is completed, all right. This is what the calculated voltages look like. Unfortunately, I really do not like this colour scheme at all, all right, I would like to go for something else. I did not expect that the default colour setting would be this. So, I will go ahead and change that, all right, ok.

(Refer Slide Time: 22:03)



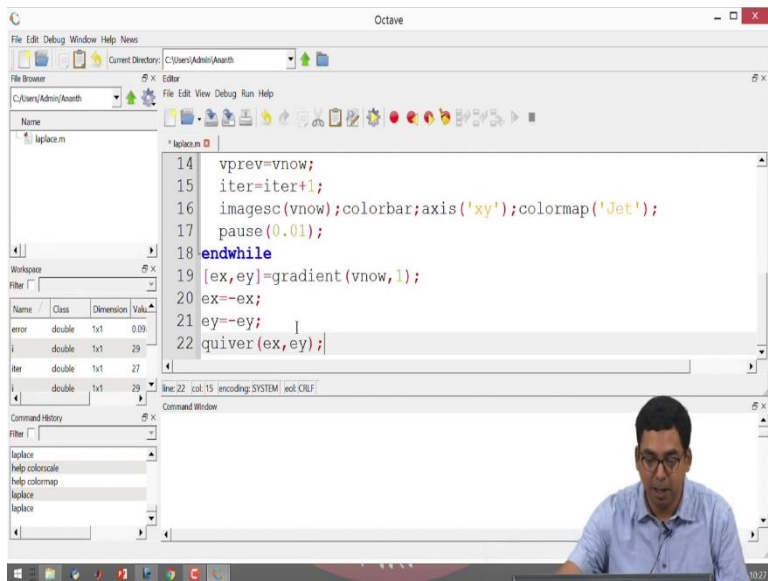
So, I am just going to change this colour map to jet, ok, ok. Little better, ok, ok.

(Refer Slide Time: 22:33)



So, it is happening if I want to terminate this abruptly I can go to the command window and press control c, all right, ok.

(Refer Slide Time: 23:01)



```
14 vprev=vnow;
15 iter=iter+1;
16 imagesc(vnow);colorbar;axis('xy');colormap('Jet');
17 pause(0.01);
18 endwhile
19 [ex,ey]=gradient(vnow,1);
20 ex=-ex;
21 ey=-ey;
22 quiver(ex,ey);
```

Name	Class	Dimension	Value
error	double	1x1	0.09
i	double	1x1	29
iter	double	1x1	27
v	double	1x1	29

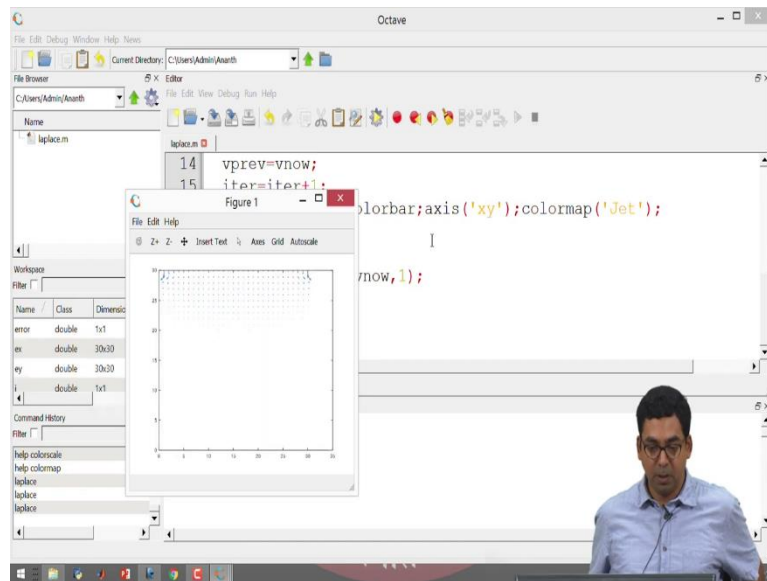
So, there you go, this is your preliminary. So, I will do this once again, ok. So, this is the calculated voltage profile inside of a capacitor that is weirdly shaped, ok.

Let us go ahead and let us start to make some more things that we can learn from this particular program, all right. First of all, I want to be able to do more than just draw you know the voltage values at each point. The course is about transmission lines and waves, all right, that means that we will have to learn about fields eventually, right. I would like to draw the electric fields inside of this capacitor and try to understand what is going on with it, ok.

So, I am going to go to the bottom, once I finished calculating the value of the voltage at each and every point, I would like to be able to draw the arrow plots of electric fields like how we do in regular class, ok. So, the electric field here is going to be given by gradient of a voltage. The gradient command takes the derivative of the voltages that we have calculated, but we know that the electric field is minus gradient of voltage. So, after I do this I will be multiplying each E_x and E_y by minus 1. So, I will just say E_x equal to minus E_x , E_y is equal to minus E_y , ok.

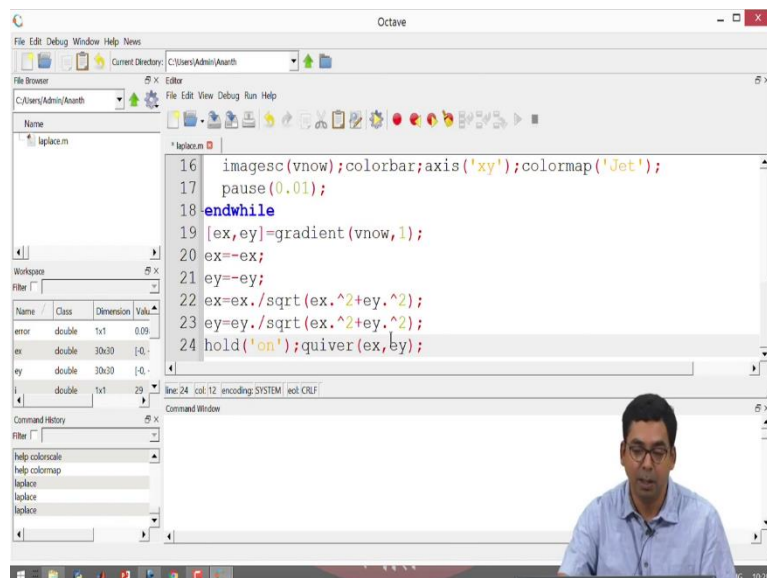
And once I have calculated this I would like to plot what the fields would look like, ok. So, I give what is known as a quiver plot. Quiver plot is nothing but a plot that tells you that it gives you an arrow plot just like how you would draw in the case of fields. So, once you run it you will understand what a quiver plot exactly is. So, I will keep my error fairly, so that there are fewer iterations, ok, ok.

(Refer Slide Time: 25:17)



Now, I noticed that I am getting some arrow plots, ok. It shows that the arrows are emanating from the top and it is doing something and it has become very very tiny, all right. I would like to have this plot on top of the voltage plot that we have calculated and I would like to see really large arrows. The default setting is the size of the arrow tells you the magnitude of the electric field, all right. So, I would like to play with all that to get a preliminary starting point, ok.

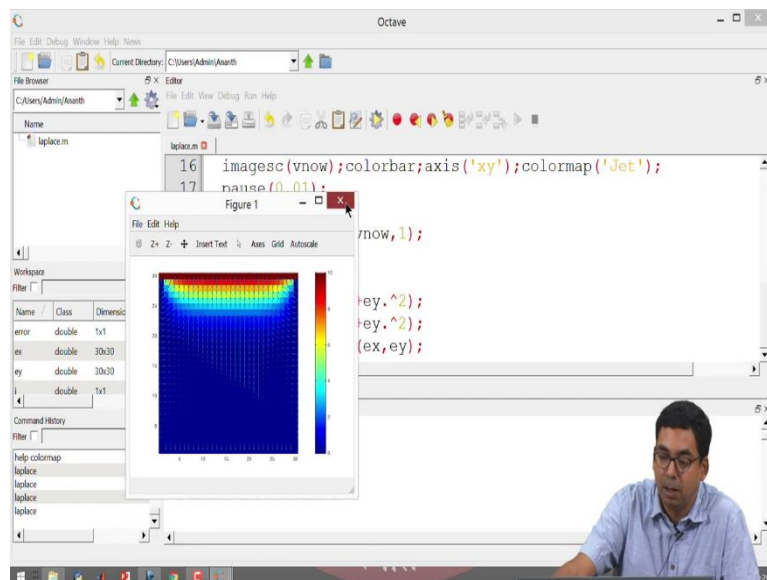
(Refer Slide Time: 25:57)



So, what I am going to do is I am going to normalize the size of the arrows. So, I am going to say that e_x , ok, is equal to E_x divided by x^2 plus y^2 . So, for each of the quiver plots I am calculating using the gradient of the voltage. I am just going to normalize the size of the arrow to its magnitude, I am going to make all the magnitudes to be equal to 1. So, this process I am normalizing all the electric fields that I am calculating. I will do the same thing for E_y , ok.

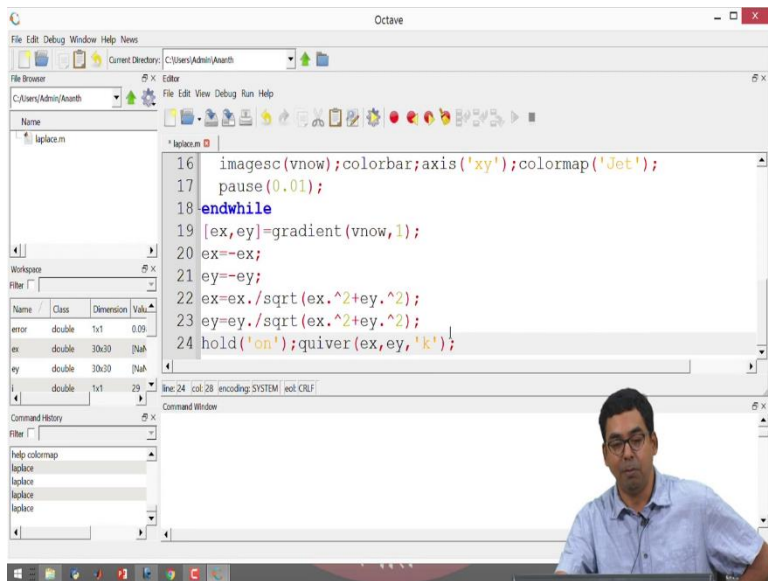
This is one part. And the other part is I wanted this to be on top of the voltages that I have calculated using this colourful plot. So, I am going to write a command saying hold. So, hold on allows me to take the last plotted graph and then add more contents to it using another frame. So, in this case I want to have an image plot of the voltage, hold it, on top of that place these quivers or the arrows for the electric field, right. So, I am going to run this.

(Refer Slide Time: 27:17)



So, now I have the arrows appearing. Unfortunately, I do not like the colour of the arrows in this computer. Default settings are all very weird in this computer.

(Refer Slide Time: 27:35)

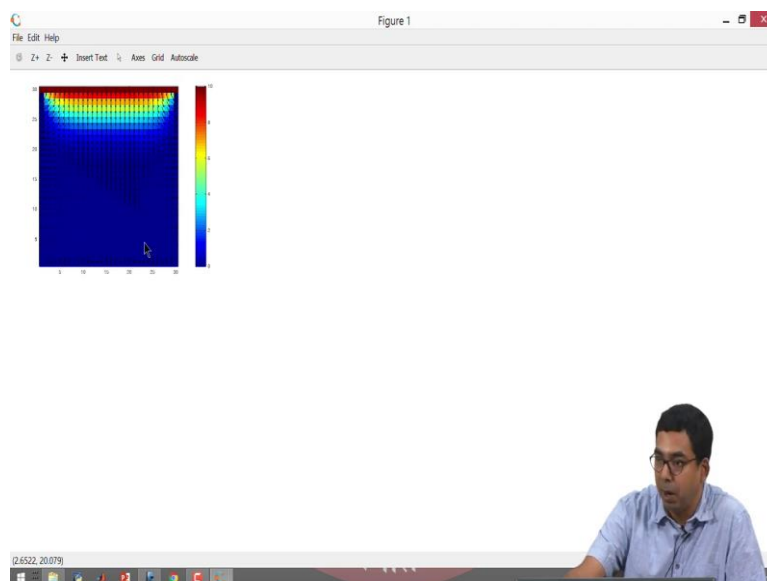


```
16 imagesc(vnow);colorbar;axis('xy');colormap('Jet');
17 pause(0.01);
18 endwhile
19 [ex,ey]=gradient(vnow,1);
20 ex=-ex;
21 ey=-ey;
22 ex=ex./sqrt(ex.^2+ey.^2);
23 ey=ey./sqrt(ex.^2+ey.^2);
24 hold('on');quiver(ex,ey,'k');
```

Name	Class	Dimension	Value
error	double	1x1	0.09
ex	double	30x30	[NaN]
ey	double	30x30	[NaN]
i	double	1x1	29

So, I am going to change this a little bit more, all right. So, I am just going to write this down as maybe, comma k, k is a short form for drawing it in black colour, ok, ok.

(Refer Slide Time: 27:41)



Now, I have the arrows starting from the top plate, all right which depicts the electric field. They are going downwards and then there is a region where the arrows are not visible at, all right or a where there are no arrows that have been calculated and then the arrows are terminating at the bottom plate, on the side plate, on the right side and the left side.

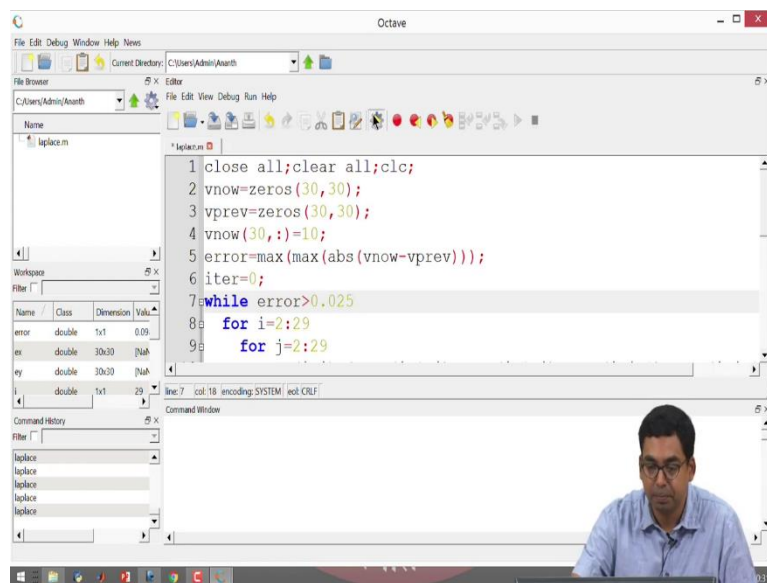
There are a few things that we can notice over here, all right. One of the things that we can notice is the arrows emerged from the top plate at right angles, all right. The arrows are

ending at the bottom plate at right angle with respect to the surface of the plate on the side the arrow will terminate at right angle, all right, both on the left hand right hand side. This gives you an idea that the condition that we are looking at is a Dirichlet condition.

The top plate is having a voltage of say 10 volts, ok, so the source of the electric field is going to emanate normally from that surface. All the other plates are at 0 volts, so the electric field has to terminate normal to the surface, all right in between it will curve. The electric field lines will not cross each other, ok. All these conditions are taken care off, ok

Now, just to be a little bit more clearer, I will make the error a little more tolerable, ok.

(Refer Slide Time: 29:07)



```
1 close all;clear all;clc;
2 vnow=zeros(30,30);
3 vprev=zeros(30,30);
4 vnow(30,:)=10;
5 error=max(max(abs(vnow-vprev)));
6 iter=0;
7 while error>0.025
8   for i=2:29
9     for j=2:29
```

Name	Class	Dimension	Value
error	double	1x1	0.09
ex	double	30x30	[NaN]
ey	double	30x30	[NaN]
i	double	1x1	29

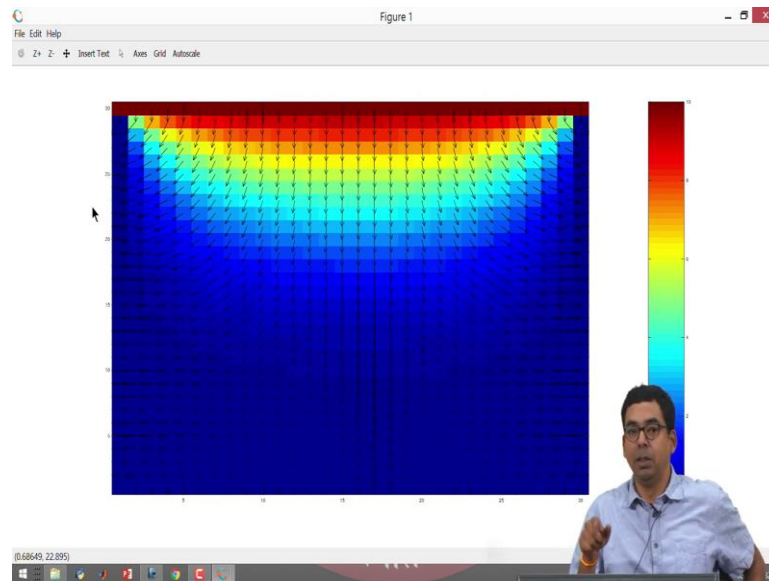
Command Window

line: 7 col:18 encoding: SYSTEM' \eol: CRLF

031

So, I am having arrows going to the bottom and then arrows going from the top plate to the left and the right hand sides, ok. This is how one would solve a partial differential equation using a computer without having any general solutions plugged in.

(Refer Slide Time: 29:11)



All you needed was definition of variables. You had some loops to cover different points, but remember that within those loops all you had was one statement saying that the voltage is going to be average of the neighbouring points and that kind of wrapped up how you would approach the Laplace equation, all right.

Now, the equations that we had seen in the prior classes for transmission line, so the wave equation is a different partial differential equation, but its structure is close to the Laplace equation. It just has got some time derivative over there. We need to get there, all right, but before we get there while we are on this let us sort out one small issue with this simulation. When I began the problem in the last class I was telling that we will start with a parallel plate capacitor, but now we do not have the parallel plate capacitor. It is a weird configuration capacitor.

We can always go back and say what should I do to get the fields of a parallel plate capacitor using this program, all right. So, one of the things that we can think about, all right is looking at our algorithm itself and trying to figure out what can be done.

Now, if we look at the calculated fields from the computer, we saw that the fields started from the top and went and terminated on the sides. In a parallel plate capacitor, you will not have the fields going to the sides, you will have the fields going from top to bottom. Secondly, we have to assume that these capacitors are infinitely wide, so that the fields actually do not bend at all they will go straight from top to the bottom, ok. So that means, the fields have to emanate from the top at 90 degrees with respect to the surface and terminate only at the bottom at 90 degrees with respect to the surface, it should not curve and go to the sides.

What is it in our program that makes it go to the sides? It is because we have assumed any unknown value of the voltage on the left and right hand side to be 0 volts. We have intentionally grounded it. So, when you have 0 volts intentionally placed your field will try to get terminated on those plates. So, if we relook at our algorithm the parts we need to fix are the left and the right hand side boundaries, ok. So, we need to look at these boundary conditions, ok.

Now, one of the things that we can do is relook at how else can I treat this boundary, ok. So, we will start with the first point again. I have a value currently unknown and I have 3 points present, the 4th point is on the left hand side of this particular place where I want to calculate the voltage is not present, ok. One of the ways to think about it is suppose I have an unknown value of the voltage here, I can make that value equal to the value on the right hand side, that is I am going to have something like a mirror image of what this voltage is going to be. So, if I have an unknown voltage at the boundary I am going to take the nearest neighbour and I am going to flip it or mirror it on this side, all right. I am going to have the same value of the voltage on the left side, ok. When I go to the right side of the unknown value on the right side instead of making it 0, I am going to make it equal to the point to the left of it, ok.

Now, that means, that each and every time I encounter the boundary the value of the voltage on the left and the value of the voltage on the right side of the boundary is going to be a mirror of what is there on the other side, which means that every iteration you are changing the boundary value, all right. It is a dynamic boundary condition, it is not a static boundary condition that you had before.

But what does that mean? That should mean something, all right. If you place a boundary in such a way that the point that you are going to place on the left side is going to be equal to the point on the right side and going by our previous lessons if I want to take a derivative of the voltage along the x direction. Since, the voltage here is identical to the voltage here the derivative of the voltage along that direction is going to become equal to 0, ok. So, the numerical way of writing that boundary condition or analytical way of writing the boundary condition is $\frac{\partial V}{\partial x} = 0$, ok.

So, when you go to the left hand side boundary you will play something over there which is the same as the value within the capacitor and when you go to the right side you will place that value equal to the value on the left, all right. That means, that on the boundary we are imposing the condition $\frac{\partial V}{\partial x} = 0$. This condition is known as the Neumann boundary condition, right.

Unlike the previous condition this is a dynamic boundary condition. Every iteration that you make will change your boundary condition, all right and this is a special form of Neumann boundary condition where the right hand side is 0, it could also be any constant, but in this particular case we are just setting the derivative of the voltage with respect to x to be equal to 0.

Now, I want to go back to the program and I want to fix the Neumann condition on the left and the right side and then I want to calculate the fields of a parallel plate capacitor, ok. So, I am going to go back to the program, I am going to close this, right and I want to have a look at what I am doing over here, all right. So, it says that from 2 to 29 is my calculation, all right that happens, ok, on the left hand side, ok, I want to make some calculations.

(Refer Slide Time: 35:47)

```

6 iter=0;
7 while error>0.025
8   for i=2:29
9     for j=2:29
10      vnow(i,j)=(vnow(i+1,j)+vnow(i-1,j)+vnow(i,j+1)+vnow(i,j-1))/4;
11    endfor
12  endfor
13  vnow(:,1)=vnow(:,2);
14  vnow(:,30)=vnow(:,29);

```

Name	Class	Dimension	Value
error	double	1x1	0.02
ex	double	30x30	[NaN]
ey	double	30x30	[NaN]
i	double	1x1	29

So, v now of colon comma 1 previously was not touched by our simulation of once you I mean you started with 0s for all the a positions of v now, you never altered v now at 1 comma something or 30 comma something, similarly you would never altered a x comma 30 and x comma 1, all right. Now, we are going to be altering that v now of colon comma 1 we are going to make it equal to v now of colon comma 2, all right. What we are doing here is on the left hand side we are taking that there is going to be no derivative; that means that you are just going to make it identical to the point next to it, all right.

So, we are just going to simply plug in the condition, v now of colon comma 1 it is going to be equal to v now of colon comma 2, it means that for all rows the first column the voltage is going to be that equal to that of second column which makes $\frac{\partial V}{\partial x} = 0$.

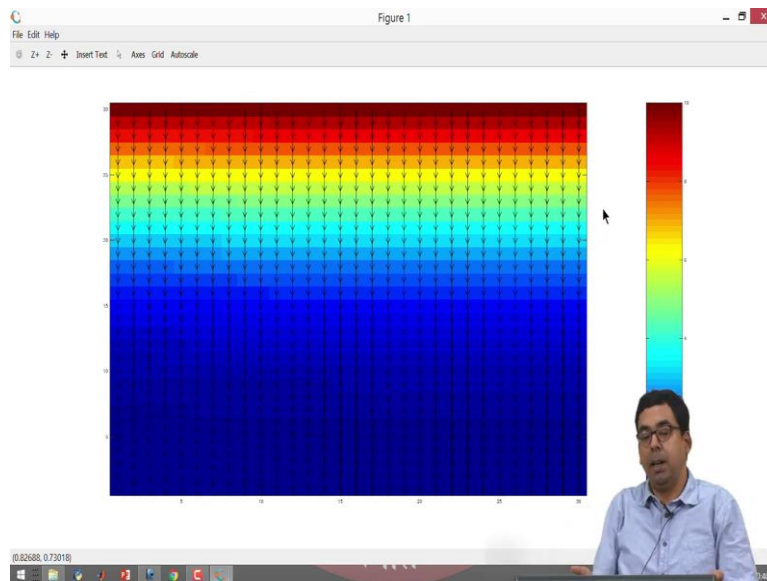
Same way on the right hand side v now of colon comma 30 will become v now of colon comma 29. Previously, v now of colon comma 1 and v now of colon comma 30 we are not touched by our program at all, but now we are explicitly setting them to be the values present in the last calculated values of the voltages, ok.

This is a simple way of writing the boundary condition. When people begin they begin to write the boundary conditions within the loop which is also perfectly fine you can say that for i equal to a 2 to I mean 1 to 30 and j equal to 1 to 30. If it is equal to 1 you can write some boundary conditions. If i equal to 30 you can write another a you know a Laplace a

equation solutions or we can say that if j is equal to 1 or j is equal to 30 you will have another statement which will tell you what v now should be equal to. So, one of the ways you could do it is v now of i plus 1 comma j multiplied by 2 plus the remaining terms, right because you are taking the value on the left to be equal to the value on right you are adding it two times, so you can always do that. So, this is a simpler way of writing the same boundary condition, right.

So, I am going to do this once again. I am going to hit run.

(Refer Slide Time: 38:13)



Now, I suddenly notice that the voltages are appearing as straight lines. There is no derivative of the voltage across the x direction and the voltage is linearly changing from 10 volts to 0 volts at the bottom, all right and my fields are absolutely straight going from top to bottom. The fields do not curve or do anything. They do not have ground plates on the sides, also we are considering that there is no derivative on the x side which means that it is infinitely large and there are no fringing fields or fringing capacitance present over here, ok. So, this would be the definition of Neumann condition.

I am not sure how many of you have solved partial differential equations before on the computer, ok, but by looking at this I hope you will be convinced that there are some merits, all right in doing it in this way because visualization becomes easier than actually writing down a few analytical solutions.

So, in this class we will not disregard analytical solutions or use only numerical value sectors. We will try to go hand in hand. Each time when we encounter a partial differential equation system we will try to identify first what is the unknown in the equation. Next we will try to figure out how to represent the unknown in terms of known quantities.

Third thing that we will do is figure out an algorithm to calculate the unknown quantity using some iterative method. The 4th thing that you will do is you will try to figure out the boundary conditions that you need, all right. And fifth thing is write the program and just run and then try to make derivations of what is going on, physical interpretation of what is going on.

Now that you know how to solve the Laplace equation, we need to go back to our equations that we had derived before which was the wave equation and try to write down how we are going to solve that. So, let us go ahead and do that now and we will wrap up our class once we get the expression for the value of voltage in the case of a wave equation that we had derived in the first or the second class, ok. So, I will go back to my notes.

(Refer Slide Time: 40:25)

So, I have the wave equation for the voltage and the wave equation for the current. I am going to take the wave equation for the voltage.

$$\frac{\partial^2 V}{\partial z^2} = \frac{1}{u^2} \frac{\partial^2 V}{\partial t^2}$$

So, I am going to take this and I am going to start to figure out what is the unknown in that wave equation, what is the known quantity, how do we find out the unknown quantity, ok.

(Refer Slide Time: 40:57)

$$\frac{\partial^2 V}{\partial z^2} = \frac{1}{u^2} \frac{\partial^2 V}{\partial t^2}$$

Here, there are 2 independent variables.

$$\frac{V(z_0 + \Delta z, t_0) + V(z_0 - \Delta z, t_0) - 2V(z_0, t_0)}{u^2} = V(z_0, t_0 + \Delta t)$$

So, I will write this down once again.

$$\frac{\partial^2 V}{\partial z^2} = \frac{1}{u^2} \frac{\partial^2 V}{\partial t^2}$$

Now, we already know how to represent second order partial derivatives using the Taylor series expansions that we have got before. $\frac{\partial^2 V}{\partial z^2}$, all right, can be written in terms of Taylor series. So, let us go ahead and do that, all right.

Here let us note that there are two independent variables, ok, ok. I need to write down $\frac{\partial^2 V}{\partial z^2}$. So, I can go back and I can try to find out what the second order partial derivative of a function would be.

(Refer Slide Time: 41:53)

4) If x & y are independent variables & $V(x,y)$,

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$
$$\frac{\partial^2 V}{\partial x^2} \cong \frac{f(x_0 + \Delta x, y_0) + f(x_0 - \Delta x, y_0) - 2f(x_0, y_0)}{\Delta x^2} \quad \text{--- (a)}$$
$$\frac{\partial^2 V}{\partial y^2} \cong \frac{f(x_0, y_0 + \Delta y) + f(x_0, y_0 - \Delta y) - 2f(x_0, y_0)}{\Delta y^2} \quad \text{--- (b)}$$

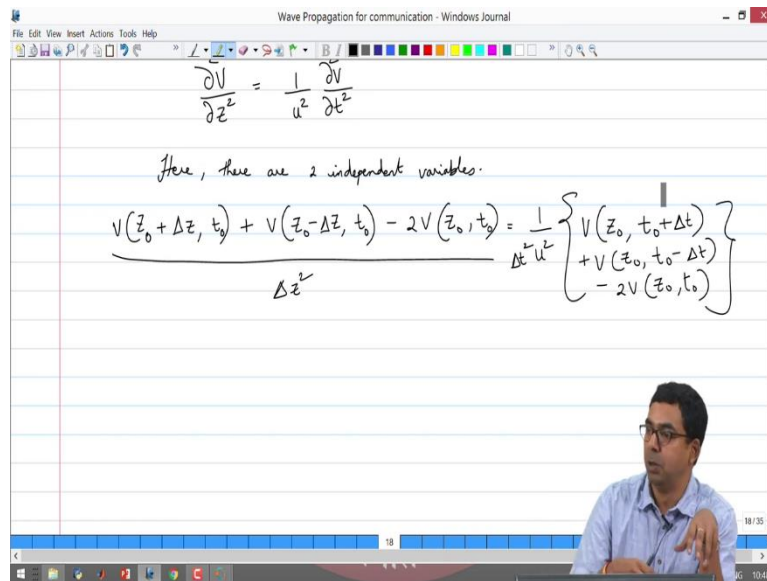
Let $\Delta x = \Delta y = h,$

So, I had $\frac{\partial^2 V}{\partial z^2}$ to be this particular expression. So, I am going to just use this expression on the left hand side. So, I have

$$V(z_0 + \Delta z, t) + V(z_0 - \Delta z, t) - 2V(z_0, t) = \frac{1}{u^2} \{V(z_0, t + \Delta t)$$

On the right hand side, I need to take the derivative with respect to time, all right. So, the spatial coordinate with remain at z_0 I will have $t_0 + \Delta t$, ok. a I think I need to put the denominator on the left hand side, right, divided by it should be Δz^2 , ok.

(Refer Slide Time: 43:27)



So, I will just

$$\frac{V(z_0 + \Delta z, t) + V(z_0 - \Delta z, t) - 2V(z_0, t_0)}{\Delta z^2} = \frac{1}{\Delta t^2 u^2} \{V(z_0, t + \Delta t) + V(z_0, t - \Delta t) - 2V(z_0, t_0)\}$$

This is how I would approach the wave equation, ok. Once you do this you need to identify what is the unknown.

In all these problems, there are some simple ways of estimating what is the unknown, all right. The simplest guideline that I can provide is in the wave equation the right hand side is a derivative with respect to time, and if you look carefully on the right hand side you have voltage to be found at $t_0 + \Delta t$. If you assume t not to be the current instant of time $t_0 + \Delta t$ is going to be what is the value of voltage at the next instant of time, ok.

So, in cases of equations which have time derivatives the unknown is simply the future values, all right. So, from this equation what you are supposed to do is take v naught of z_0 comma $t_0 + \Delta t$ on one side, all the other quantities on the right side. If all the other quantities are known then you will be able to estimate the value of the voltage v at position z_0 at the next instant of time. So, we are saying that if the past values $t_0 - \Delta t$ is known.

If the current values voltage at t naught is known then you will be able to estimate what is $t_0 + \Delta t$. It requires a rearrangement, all right and then once you rearrange you will need to plug in a few values, all right.

So, for starters just like how we did Laplace equation you can say that Δz and Δt is going to be equal to 1, that is how we did Δx and Δy in the Laplace equation. And at the velocity, we

can take some 1 meter per second or 1 grid cell per unit time step to make our calculations very simple to begin doing this analysis.

So, we will stop here. My suggestion is try to a bring the unknown to one side, known quantities to the other side, keep your equations ready, in the next class we will write a code for this and then try to understand what the wave equation that we derived in the first two classes actually meant, what are the boundary conditions for the wave equations in transmission lines, all right. And then we will proceed from there, ok.

So, I will stop here.