

Transmission lines and electromagnetic waves

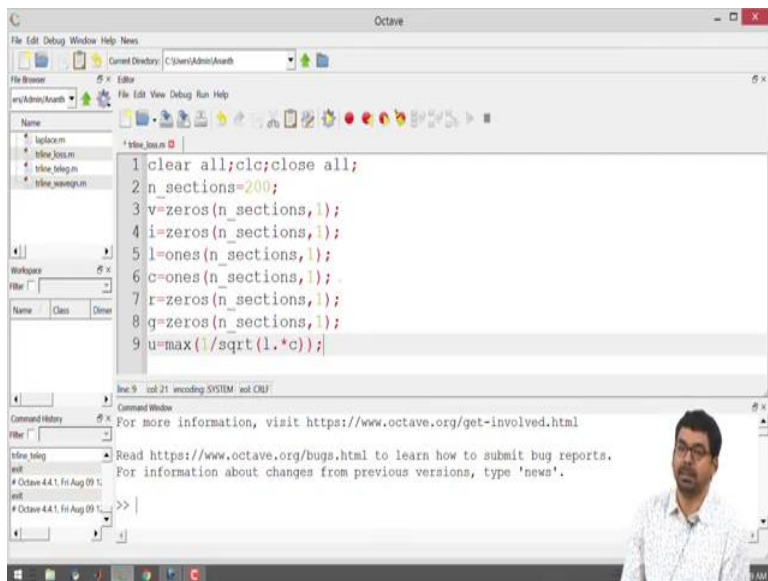
Prof. Ananth Krishnan

Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 09

Octave Simulation of Transmission Lines with Losses

(Refer Slide Time: 00:15)



```
1 clear all;clc;close all;
2 n_sections=200;
3 v=zeros(n_sections,1);
4 i=zeros(n_sections,1);
5 l=ones(n_sections,1);
6 c=ones(n_sections,1);
7 r=zeros(n_sections,1);
8 g=zeros(n_sections,1);
9 u=max(1/sqrt(1.*c));
```

The screenshot shows the Octave software window with a script editor containing the above code. The command window below shows the execution of the script, with the current line being line 9. The interface includes a menu bar, a toolbar, and a file browser on the left.

So, this class what we are going to do is, we are going to see the version of a computational programmatic version of whatever we saw for the ac excitation of the transmission line and also include the losses that are r and g into the model. And we have derived some partial differential equations that we are going to plug into the program and actually see what can happen to understand the exponential decay or exponential growth corresponding to your attenuation constant, whether it is positive or negative this is the objective.

There are some nuances to the code ok, some nuances I will explain, some of them are out of scope of this course, ok. It will be dealt with in an advanced elective that you have named computational electromagnetics, all right. But as much as possible I will try to answer your questions, ok. So, I will begin with this, clear all the variables in the workspace, clear the command window, all right, close all active figures, ok.

And we can begin with the transmission line section definition. So, I am going to have two 200 sections of a transmission line, ok. And corresponding to these 200 sections, I will be having

the definition of voltage current I and c ok, and also I will be adding r and g later. So, I will be having an array for storing the voltage at different points.

So, since I do not know the values of voltages, I am initializing into zeros, the dimensions of this will be n sections comma 1. So, the voltage is now a vector, one dimensional vector having all zeros, ok and the size of that is 200. Similarly, the current is also going to be zeros right and its dimension is going to be identical, ok. And then we are going to add a few more things to make it slightly different from the previous codes that we had written for the dc or the pulse excitation.

So, here we want to have control over the spatial profile of the transmission line all right and we want to manipulate the spatial profile and observe what is happening with respect to space and time, what that means, is in different sections I would like to be able to have different values of I and c ok and different values of r and g when possible, ok. So, I am going to define my I also to be an array, ok.

So, I will be having I as one of, note that I am using ones of a n section comma 1, I am not using zeros, ok. The reason is if the momentum you plug in is equal to 0, you will end up having velocity to be infinite and we have not dealt with any case with velocity infinite, this whole transmission line is beginning on the premise that there is a finite delay. So, to ensure that I have a finite delay, I am starting with ones, ok.

And I am going to be having capacitance for all the sections to be equal to 1, all right. So, for now I will also make the other two parameters equal to zeros. I am going to start with the lossless transmission line and slowly add losses, ok. So, I am going to start with r equal to zeros of n sections comma 1 and g to be zeros of n sections comma 1, just save this file, right ok.

So, this means that in each and every section I can control what the value of I c r and g would be later on as I go in the program, I have initialized them to be an array, all right. So, in this particular case right I and c equal to 1, all right. So, the velocity will be equal to 1 by square root of I c which is also going to be equal to 1, all right.

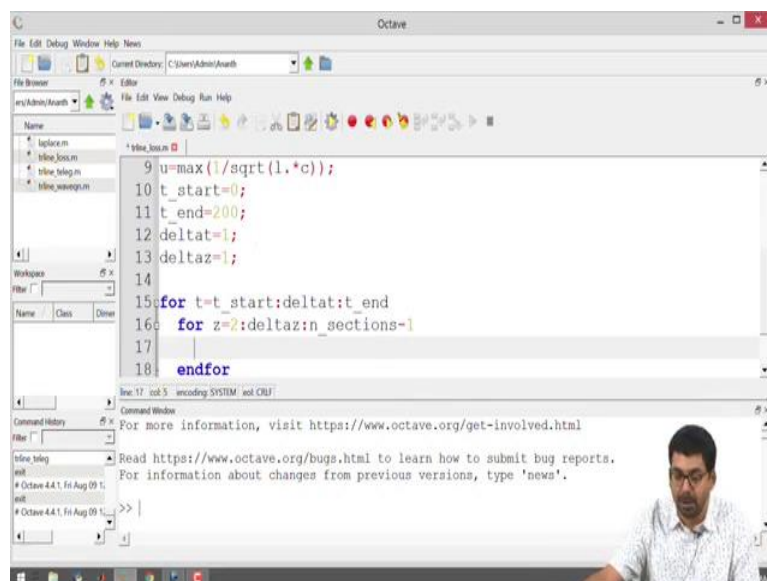
So, if I choose Δz to be equal to 1, Δt will automatically become equal to 1, because it has to go one space step and one-time step that is how you get the value of velocity to be equal to 1. So, all the calculations become simple if I choose them to be ones, ok. But later on accurate values can be plugged in, some considerations are needed ok, but this gives the essence of the theory that is more than enough, ok.

So, I am going to start with u is equal to this case you know $\frac{1}{\sqrt{lc}}$. So, that a dot star is just representing that each element of l , because it is an array will be multiplied with the corresponding element of c , all right. If that is the case the velocity will also become an array all right, which means that, in different positions you can have different velocities, I want to keep it less confusing. I want to decide the time steps and space steps based on the maximum velocity that is possible.

So, one could always say that I am not going to be going for l and c less than 1 as of now, alright. But it is always possible to fix whatever value you want and the maximum velocity that is going to be in your system actually dictates many parameters.

So, one could always say that, we can have sum ok. So, deltat , deltaz extra has to be considered for the maximum velocity in your system, ok. And these details will become apparent as you do some trials ok, ok.

(Refer Slide Time: 06:55)



```
9 u=max(1/sqrt(l.*c));
10 t_start=0;
11 t_end=200;
12 deltat=1;
13 deltax=1;
14
15 for t=t_start:deltat:t_end
16     for z=0:deltax:n_sections-1
17
18     endfor
```

The screenshot shows the Octave software interface. The main window displays a script with MATLAB-style code. The code defines the maximum velocity u as $1/\sqrt{l \cdot c}$, sets the time interval from $t=0$ to $t=200$ with a time step of 1, and sets the spatial interval from $z=0$ to $z=n_sections-1$ with a spatial step of 1. The code uses nested for loops to iterate over time and space. The interface also shows a file browser on the left, a command window at the bottom, and a small video inset of a person in the bottom right corner.

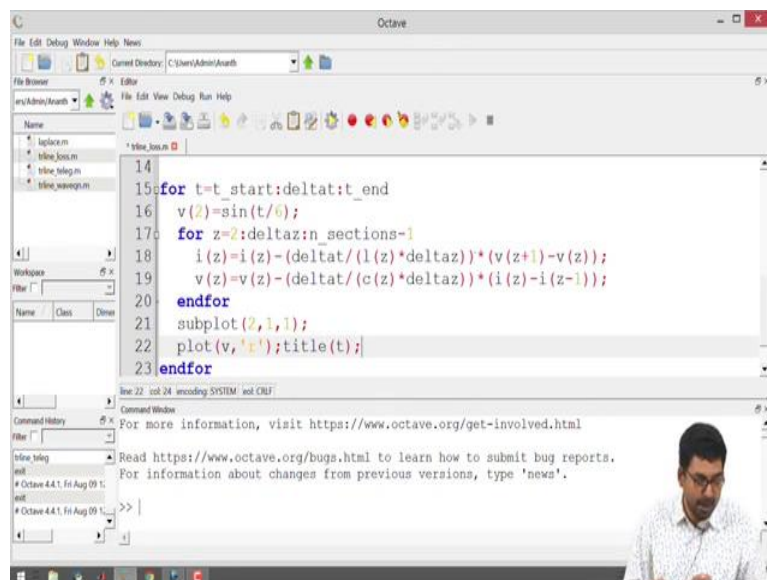
And I am going to designate my starting time to be as 0, that is the time at which I start the simulation, ok. And I would like to designate the ending time to say 200, the reason because I have chosen 200 in the beginning is their number of sections is 200. If the velocity is going to be equal to 1, then it should take me 200 instances of time to go from one side to the other side for the voltage wave, right.

So, that is why I am starting with this, later on we can always manipulate this, ok. For now, I am not going to make use of the velocity u all right, because I know that I want deltat to be equal to 1 to simplify the amount of math that I have, consequently $\text{delta } z$ also becomes equal to 1, ok. This more or less takes care of all the variables that we need to define, ok.

Once we have defined it, then we have to go for solving the coupled partial differential equations that is it. We have done this before for the pulsed case right, there are just slide variations that we will be seeing over here. So, the main program consists of a giant for loop in time, right. So, for the is starting from t_{start} right in steps of deltat all the way to t_{end} , ok. At each and every instant of time, we will be solving the telegrapher's equation for all points in space, that is the algorithm that we are going to be following.

So, I am going to start with for z is equal to 2, the reason for doing 2 to sections minus 1 should be clear by now, because we know that the differences at the edge can go to the you know index out of bound. So, to prevent that from happening, now you are doing the same as what we had done prior to this particular lecture, all right.

(Refer Slide Time: 09:15)



```
14
15: for t=t_start:deltat:t_end
16   v(2)=sin(t/6);
17:   for z=2:deltaz:n_sections-1
18     i(z)=i(z)-(deltat/(l(z)*deltaz))*(v(z+1)-v(z));
19     v(z)=v(z)-(deltat/(c(z)*deltaz))*(i(z)-i(z-1));
20   endfor
21   subplot(2,1,1);
22   plot(v,'r');title(t);
23 endfor
```

Now, for all instances of time we will have to solve for the voltage and the current, we are familiar with the equations that we have already. We will begin with them, all right, and then we will go forward and start making some extra terms into the update equations, ok. So, previously what we were doing was, we were calculating the current based on the telegrapher's equations.

So, we used to have

$$i(z) = i(z) - (\text{deltat}/(l(z) * \text{deltaz})) * (V(z + 1) - V(z))$$

This is the same update equation that we had when we were dealing with dc or pulse excitation of the transmission line before, all right.

And putting down the same equation that we had before all right, the only change that I have made with respect to the prior program is I have made this to be l of z . Previously we had a fixed value of l or fixed value of c , I have just changed it to reflect that we are going to deal with some spatial information.

Similarly, we had an update equation for the voltage

$$V(z) = V(z) - (\text{deltat}/(l(z) * \text{deltaz})) * (i(z + 1) - i(z))$$

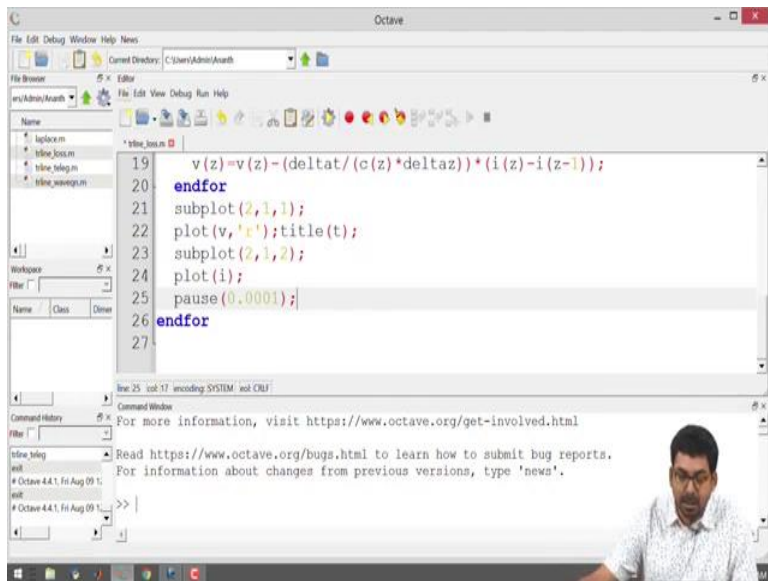
These were the two equations that we were solving before, ok. Just to be sure that we are doing some excitation, now all the voltages and currents are initializing to zero, if you solve this you will be getting all the values to be zero. So, you have to define some source of some kind all right and here we are interested in sinusoidal excitations, ok.

So, we can place a voltage source. So, we can say that with respect to time, it is going to give some sinusoid of time, but I am going to just reduce the time period all right, not the time period frequency all right, I am going to reduce the frequency. So, I am going to have $v(2) = \text{Sin}(\frac{t}{6})$, I am doing this deliberately ok you will get to know about why I did this in a few seconds as we start manipulating things.

So, this loop will keep calculating the values of voltage and current according to the telegrapher's equation. And then for every instant in time, after this loop has finished inside for the calculation of a voltage and current, we want to be able to display the plots that is all we have done. So, we will write down a subplot ok 2 rows 1 column and plot number 1 ok, that is what 2 commas 1 comma 1 means.

So, we will be plotting voltage in red colour, ok. And I want to have an estimate of how much, how many iterations are going on extra. So, I will just make the title to be the instant of time. So, if I have 200 iterations, it needs to keep displaying the current instant of time for which it shows me the voltage and the current, ok.

(Refer Slide Time: 13:09)

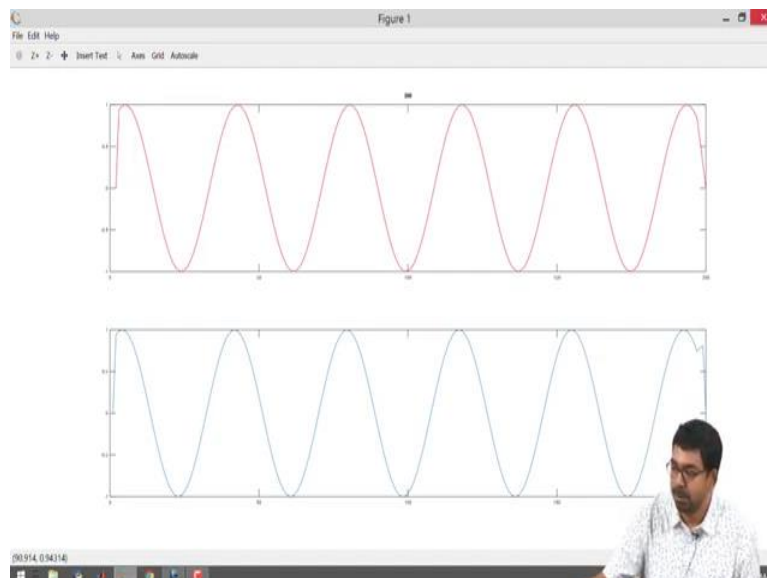


```
19 v(z)=v(z)-(deltat/(c(z)*deltaz))*(i(z)-i(z-1));
20
21 endfor
22 subplot(2,1,1);
23 plot(v,'r');title(t);
24 subplot(2,1,2);
25 plot(i);
26 pause(0.0001);
27 endfor
```

The screenshot shows the Octave environment with a script editor containing the above code. The Command Window at the bottom shows the execution progress, including the message "line 25: 100% encoding SYSTEM: not OUI".

And I will have another subplot 2 comma 1 comma 2, there I will plot the current. And we already know that if we do this and run the program, we will not be able to see the plots till the end of the program, simply because of the speeds involved with this. So, we need to introduce a small pause command. So, for a small instant of time I am going to pause, so that allows me to see what is going on. I will quickly save this and run to make sure that there are no big errors.

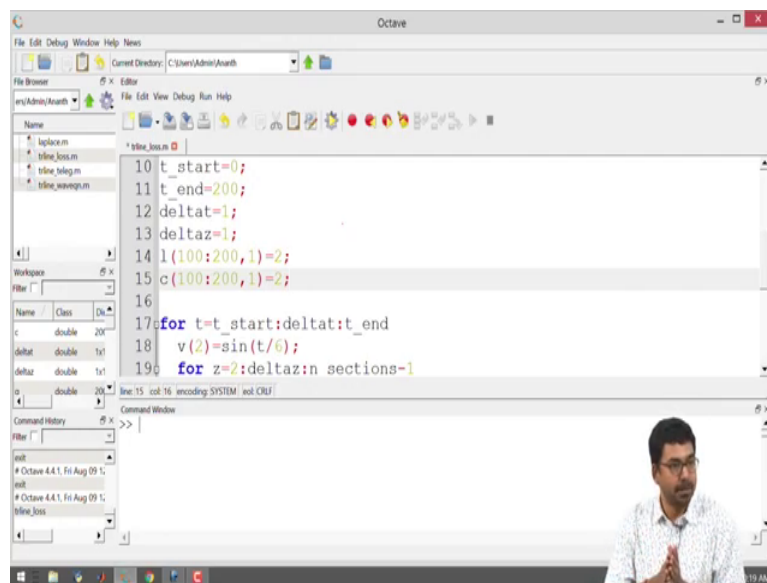
(Refer Slide Time: 14:05)



I am having voltage on the top, current at the bottom, all right ok. Once it reaches the boundary, the simulation stops, all right. And we have seen glimpses of this in the previous classes, but we did not go over it in the detail that is all, right.

So, now I will go back to the program, just wait for a couple of you to finish, then I will move on with the manipulations, ok. We will start with some minor manipulations of the program, ok. First of all, I want to change the values of l and c in the second half of the transmission line ok. So, I am going to go ahead.

(Refer Slide Time: 15:13)



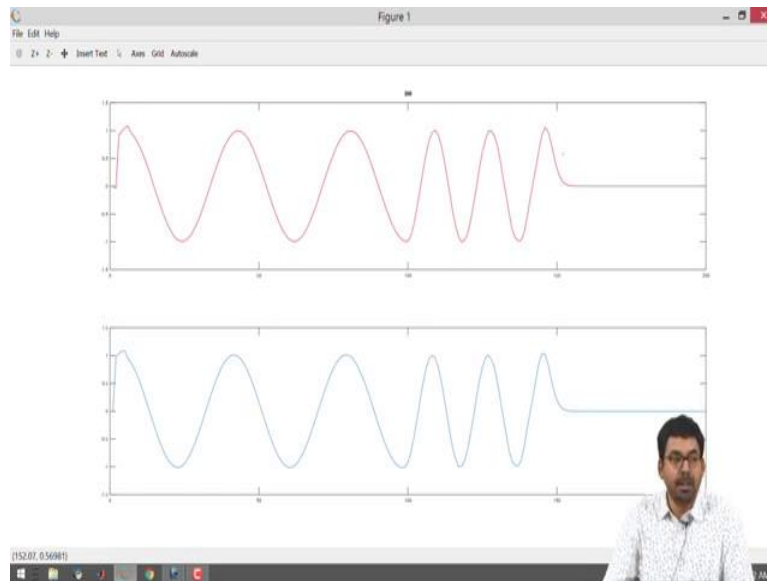
```
10 t_start=0;
11 t_end=200;
12 deltat=1;
13 deltax=1;
14 l(100:200,1)=2;
15 c(100:200,1)=2;
16
17 for t=t_start:deltat:t_end
18     v(z)=sin(t/6);
19     for z=2:deltax:n sections-1
```

After I have all the initializations done line number 14, ok, we are going to create some space over there, I am going to say that, l from 100 to 200 positions ok is going to be equal to 2, similarly, c from the position 100 to 200, so also going to be equal 2.

So, now, you have two halves of the transmission line, the left half of the transmission line will have l and c equal to 1, that is from position 1 to 100 it will have a you know value of l and c equal to 1. The second half of your graph whatever you are seeing in the simulation will be having l equal to 2 and c equal to 2, ok. So, I am going to run this, we have to keep in mind what happened in the previous simulation and then compare this.

The previous simulation was pretty much non-event full, that is the wave just travelled from left to right. In 200 instances of time, it reaches the 200 spatial point at which point the simulation stops, ok.

(Refer Slide Time: 16:23)



Now, let us have a look at what is going to happen here, ok. There are a few things that we can notice, first of all there was a vigil at that interface, which means that some reflection was happening between the left and the right side points. Next thing I notice is that the wavelength of the wave on the right hand side is shorter than the wavelength of the wave on the left hand side.

Since I have a cursor at the bottom over here all right, I can always figure out what the wavelength is and then try to see what the wavelength is on the right hand side. On the left hand side, I am going to take the position or the length distance between the two say dips. So, I am having 23.71 marked over, I am looking at this a x coordinate. So, 23.8, I will say it is 24 right, 24 to 61 right that is about 37, right.

So, on the right side I have 118 to 137 that is about 19, right. So, it is about one half. So, the wavelength on the right side is one half of the wavelength that we have on the left side, is that correct? So, we had 37 about 19 there, we did not do it very precisely, we just moved the mouse over and found. So, the wavelength on the right is one half.

So, it is my first observation, all right. So, the value of l and c in the transmission line can tell you, you know what the wavelength would be in that section. And it is clear that, if your l doubles, c also doubles wavelength is becoming half, ok.

The second thing that we have to notice is that the wave should have reached 200 spatial steps at the end of 200 time steps, it did not do that, ok. Simply because we already know that the velocity is going to be $1/\sqrt{l c}$, ok. So, in the second half of the

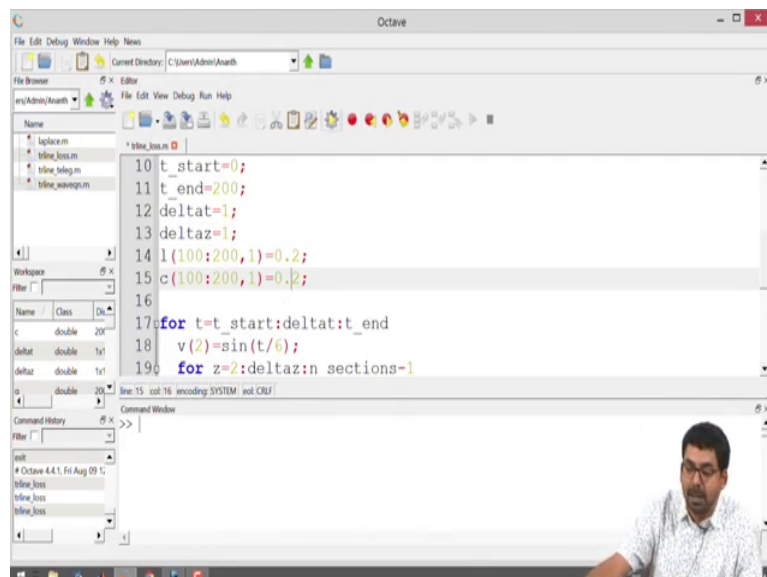
transmission line, the velocity is reduced, ok. So, we had l doubling c doubling, so the velocity reduces to half. And as a result of which it should have traveled so many time steps.

If you were to look at this carefully, you will be able to figure out that ok it travelled about from 100 to 200 is the same transmission line with identical parameters, but it moved only to 50 percent of that. So, it is travelling with about half the velocity, ok. So, the consequence of changing l and c in different portions of your transmission line is first, you are creating an interface, there is going to be a reflection, there was a small wiggle that was happening, all right.

I do not know how many of you saw that, but I will just run it again and talk. So, as this wave front reaches the midpoint right, you will see that a small wiggle goes on your voltage wave and also the current wave, ok. See something going there, that means, there was a reflection, so that we call it as a reflection due to an impedance mismatch between the transmission lines. You can always calculate the coefficient all right, the reflection coefficient and try to see what is happening, all right. On the right hand side, we are noticing that the wavelength has shrunk, velocity has reduced ok.

Now there is a tendency to see to do the exact opposite of what we did, right. There is a tendency that why can I not reduce l and c , right. In the way we have written the program right, it is not going to yield an appropriate result so, but just to make that point clear, I will just instead of making a 2, I will make it 0.2 for l , 0.2 for c .

(Refer Slide Time: 20:29)

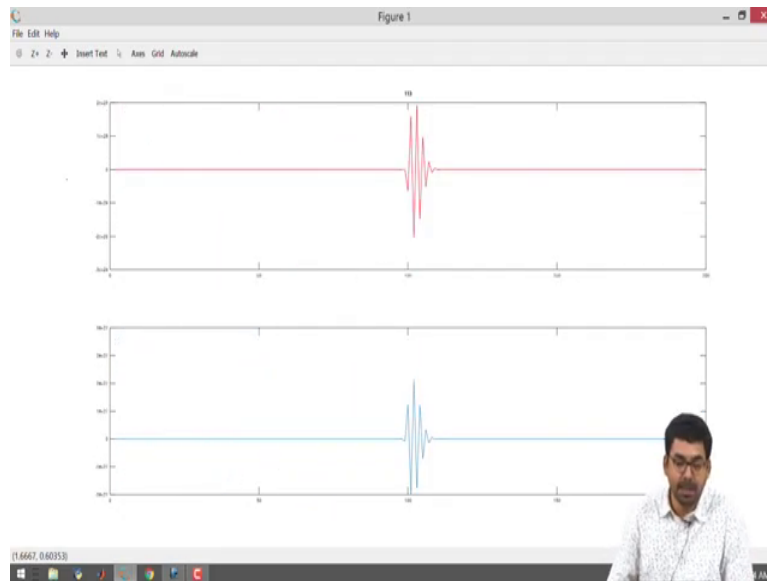


```
10 t_start=0;
11 t_end=200;
12 deltat=1;
13 deltax=1;
14 l(100:200,1)=0.2;
15 c(100:200,1)=0.2;
16
17 for t=t_start:deltat:t_end
18     v(z)=sin(t/6);
19     for z=2:deltax:n_sections-1
```

Name	Class	Size
c	double	200x1
deltat	double	1x1
deltax	double	1x1
t	double	200x1

Now also one will argue that there is an impedance mismatch and the program should capture it correctly, but the way we have written it to look slightly absurd, all right.

(Refer Slide Time: 20:41)



So, the wave travels from one medium, it enters the second medium and the y axis blows up and you do not have a graph or anything anymore. It is not because it is an incorrect physical setup or anything like that, it is just because the way we have written the program, it is a very important understanding that the choice of Δt , Δz play a very very big role in your simulation, ok.

Remember that I was saying that, the maximum velocity has to be considered, ok. The maximum velocity will give you the shortest time in which your voltage wave front or current wave front will go from one spatial grid to another spatial grid point, ok.

Suppose in one step, one time step all right, it appears that the wave has gone from the left side of the grid point to another point on the right side. Which is what is happening right now, your Δt and Δz has reduced and the wave we have written the simulation is, if I had a point at one instant of time the wave was here, at the next instant of time the wave was there without passing through this point. That is how we have written the simulation, all right.

So, such kinds of things are known as instabilities in the way you have written the program, and the specific term that is commonly used in this field is known as the Courant stability criterion commonly known as Courant Friedrichs Lewy criterion or CFL, ok. We will not go into this details about this stability, but one thing I wanted to make clear is, when you are writing these programs and manipulating to see some of the things that we have seen in the theory may be that, you encounter a situation that your y axis blows up and you do not have a graph.

In such cases you should quickly go back and see whether your choice of Δt Δz is appropriate in such a way that, in one instant of time, you are going to the next space step or not.

(Refer Slide Time: 22:59)

```

198 for z=2:deltaz:n_sections-1
20   i(z)=i(z)-(deltat/(l(z)*deltaz))*(v(z+1)-v(z));
21   v(z)=v(z)-(deltat/(c(z)*deltaz))*(i(z)-i(z-1));
22   endfor
23   subplot(2,1,1);
24   plot(v,'r');title(t);
25   subplot(2,1,2);
26   plot(i);
27   pause(0.0001);
28   endfor

```

Warning: opengl_renderer: data values greater than float capacity. (1) Scale d (2) Use gnuplot

You can always choose the value of. So, here I am having $\Delta t/\Delta z$, I am not including a velocity term get over here u , all right. So, simply put just briefly write down without going into too much details, because there is another course for that, ok.

(Refer Slide Time: 23:21)

$$\frac{u \Delta t}{\Delta z} \leq 1$$

$$\frac{u \Delta t}{\Delta z} = 1 \quad (\text{Magic time step criterion})$$

So, in this course we are using u for velocity. So, I will just make this as

$$\frac{u\Delta t}{\Delta z} \leq 1$$

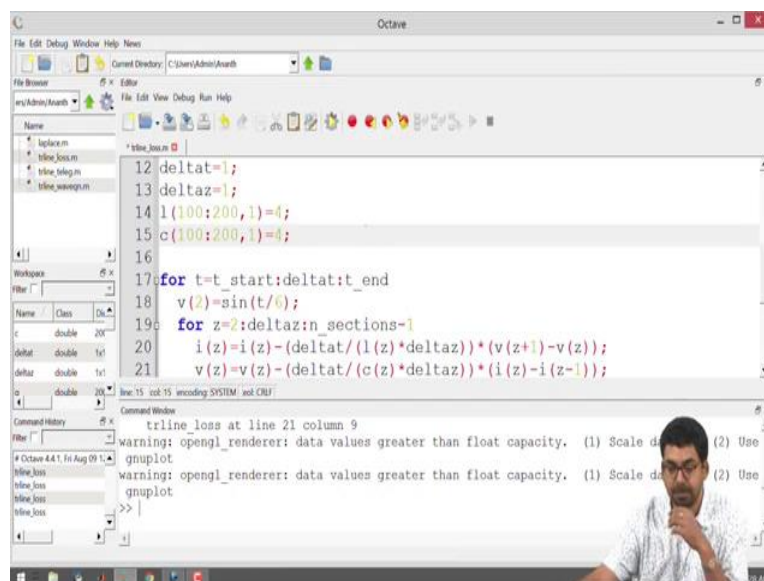
This quantity has to be less than equal to 1 for your simulation to run without blowing up the y axis, ok. If it is equal to 1, this is known as a magic time step criterion, right.

There is no big magic, except that whatever choice of Δt and Δz you make conforming to this criterion, you will have no errors in your simulation especially with respect to phase. If you did choose $u \Delta t$ divided by Δz to be less than 1, you can expect some phase errors in your simulation, because the wave will not go exactly to the next point, right. It will go in between your first and next point, so you will start having some phase errors as your simulation goes running.

It may appear to be travelling slower than it actually should be travelling extra, those are just too many details to understand transmission lines, all right. But I want you to know that, if you just did run a simulation and your y axis was blowing up, most likely you have an issue with the choice of Δt and Δz , make sure that the majority of the cases is equal to 1.

And also make sure that you are not using the criterion, I mean the Δt equal to 1, Δz equal to 1 should lead to a velocity equal to 1. And your values of l and c are the other sections of the transmission line should not be higher than what you have chosen, for us to just understand what is going on over here.

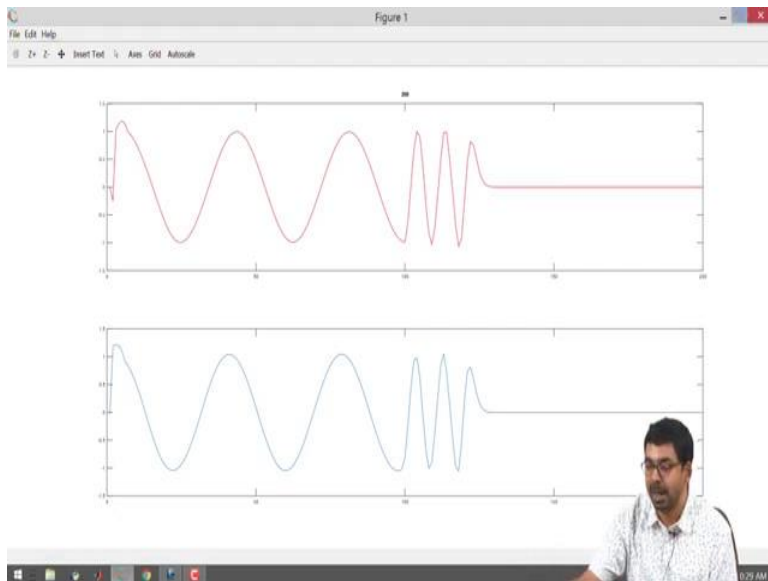
(Refer Slide Time: 25:21)



```
Octave
File Edit Debug Window Help News
Current Directory: C:\Users\Admin\Asanth
File Browser
Name
  * lplacem
  * trline_loss.m
  * trline_mag.m
  * trline_wavegen.m
Workspace
Filter
Name / Class / Dir
c double 200
delta_t double 1e-11
l double 200
Command History
trline_loss at line 21 column 9
warning: opengl_renderer: data values greater than float capacity. (1) Scale da (2) Use
gnuplot
warning: opengl_renderer: data values greater than float capacity. (1) Scale da (2) Use
gnuplot
>> |
```

You could always choose you know 4 and 4 oops, ok.

(Refer Slide Time: 25:33)



There is a bigger reflection that is happening, your wave is travelling far to slow the second medium. And it seems to be having some distortion in the way it is represented, it does not look like a full sinusoid, it looks like some points there are sharp points in your sinusoid, all right.

So, that just tells you about the way you're sampling a sinusoid in the second medium, all right. As you change the medium from one to another or the characteristic impedance from one to another, the way you sample your sinusoid will change, ok.

So, those are minor details, but l and c in this right hand side section above 1 for the simulation condition that we have chosen will work, ok. Anything below 1 will give you some access, I mean out of bounds error, right.

(Refer Slide Time: 26:43)

```

12 deltat=1;
13 deltaz=1;
14 l(100:200,1)=1;
15 c(100:200,1)=1;
16
17 for t=t_start:deltat:t_end
18     v(2)=sin(t/6);
19     for z=2:deltaz:n_sections-1
20         i(z)=i(z)-(deltat/(l(z)*deltaz))*(v(z+1)-v(z));
21         v(z)=v(z)-(deltat/(c(z)*deltaz))*(i(z)-i(z-1));

```

So, now I am putting it back to 1 right, I want to be able to add r and g into my simulation, all right. So, I go to my update equation ok and start to add the terms, all right. So, for the current calculation, the way we had calculated before, it had a term for r multiplied by I, so if I were to look at the expression that I had with the loss included, all right.

(Refer Slide Time: 27:31)

KVL:-

$$V(z) - (r\Delta z) i(z) - L\Delta z \frac{\partial i}{\partial t} - V(z+\Delta z) = 0$$

Rearrange,

$$V(z+\Delta z) - V(z) = -r\Delta z i(z) - L\Delta z \frac{\partial i}{\partial t}$$

$$\Rightarrow \frac{V(z+\Delta z) - V(z)}{\Delta z} = -r i(z) - L \frac{\partial i}{\partial t}$$

As $\Delta z \rightarrow 0$,

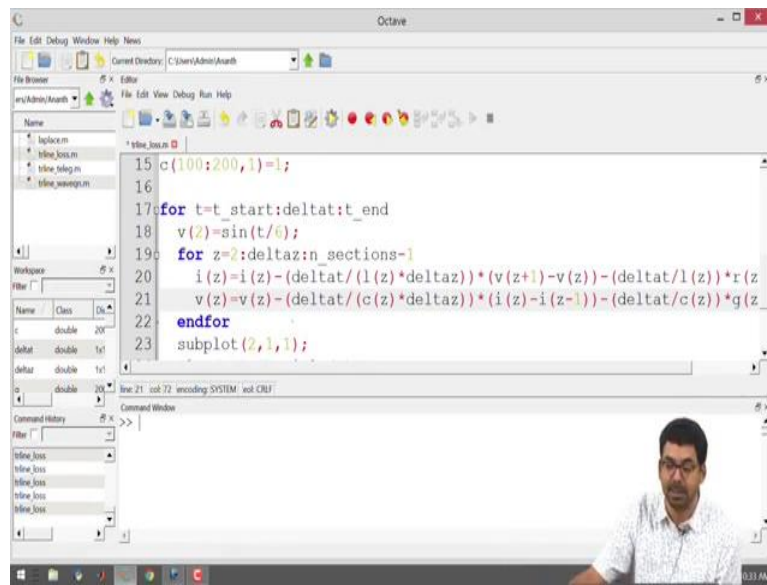
$$\Rightarrow \boxed{\frac{\partial V}{\partial z} = -r i(z) - L \frac{\partial i}{\partial t}}$$

So, from this equation the objective is to figure out the current at the next instant of time, because the time derivative is for the current, so the unknown quantity will be current at the

next instant of time. So, if you rearrange this equation, you will get the first telegrapher's equation which we already have in the program, but there is going to be another term coming there, which is going to be $-ri(z)$, right.

So, this is the part that is going to come into your code. So, we can go back to the code and plug this part in, right.

(Refer Slide Time: 28:07)



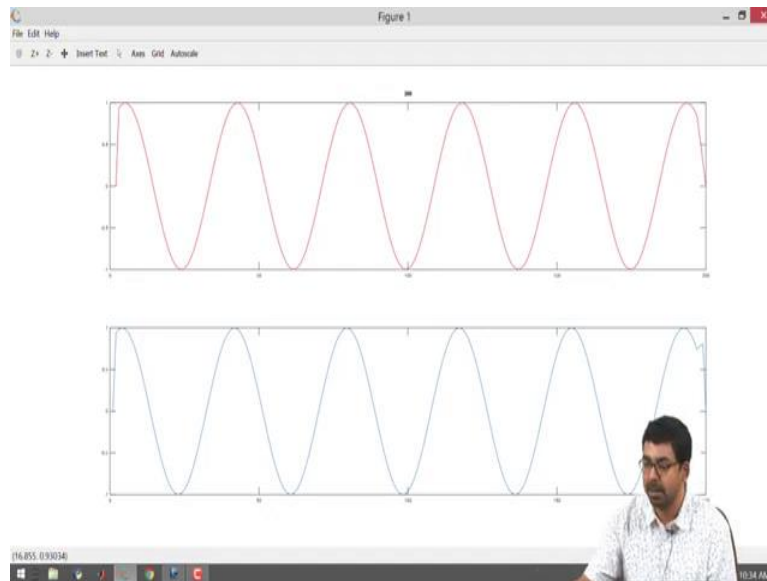
```
Octave
File Edit Debug Window Help New
Current Directory C:\Users\Admin\Asarth
File Browser
File Edit View Debug Run Help
Name
  * lplace.m
  * lplace.m
  * lplace.m
  * lplace.m
Workspace
Filter
Name Class Dbl
c double 200
deltat double 1e-1
deltaz double 1e-1
Command History
Filter
>>
line 21: col 72: encoding SYSTEM 'utf-8'
```

So, I am just adding that particular part. So, r multiplied by $i(z)$ was there, but if you were to find out the current at the next instant of time, it has some coefficients that have to be multiplied and you have to balance the equation. So, you are getting you know minus Δt by $i(z)$ multiplied by $r(z)$ times a $i(z)$, that is what we have for the expression for the current at the next instant of time, right.

Similarly, for the voltage all right you will be having an additional term coming into your update equation. So, I will be just copying this pasting and then I will be making some changes to it. So, I have Δt divided by c of z ok a multiplied with g of z instead of $r(z)$ and i goes to v , right those are the only changes.

So, I will not change anything now, I will just make sure that for r equal to 0 and g equal to 0 which is the program that we have everything works correctly and then I will start to disturb, start to disturb the values of r and g , ok.

(Refer Slide Time: 29:43)



And I have changed the l and c back to one and all the sections seem to work fine, right. So, there I do not have any errors when r is equal to 0 and g is equal to 0, but now I will start putting in values for r and g , ok. And we are going to consider transmission lines that are lossy, but you know the values of r and g are much less than the values of l and c , right. So, I introduce a small amount of loss, the second half of the transmission line.

(Refer Slide Time: 30:27)

The screenshot shows the Octave software interface. The main window displays the following code in a script editor:

```
10 t_start=0;  
11 t_end=200;  
12 deltat=1;  
13 deltax=1;  
14 l(100:200,1)=1;  
15 c(100:200,1)=1;  
16 r(100:200,1)=0.001;  
17 g(100:200,1)=0.001;  
18
```

The workspace window on the left shows variables: c (double, 20x), $deltax$ (double, 1x1), $deltat$ (double, 1x1), and t (double, 20x). The command window at the bottom shows the prompt `>>`. A person's head and shoulders are visible in the bottom right corner of the window.

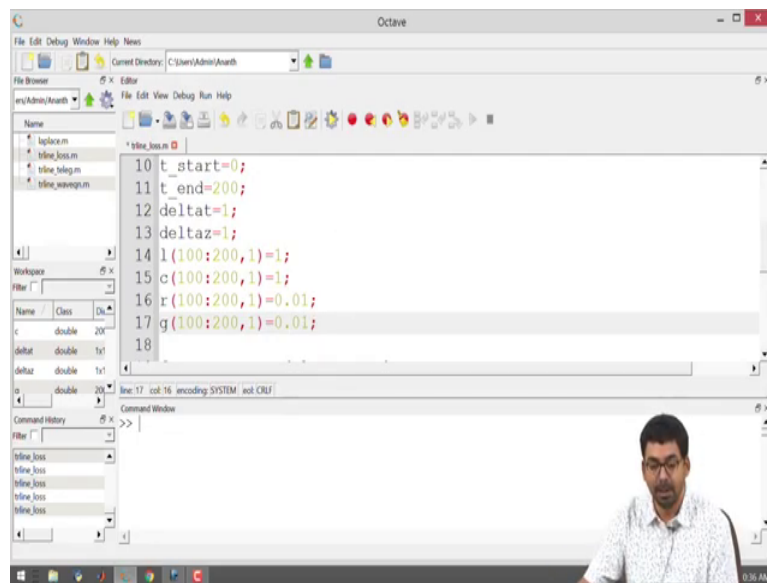
So, I will have r of 100 colon 200 comma 1 is equal to 0.001 all right all right and I am making g from 100 to 200 to be 0.001. So, the left half of the transmission line is r , l , g , c equal to, I mean r and c equal to 0, l and c is equal to 1. So, it is a lossless transmission line on the left side. On the right hand side, it is a lossy version of the same transmission line, l and c are identical, but just r and g have been added.

(Refer Slide Time: 31:09)



And I am going to run this, ok. So, we can notice that on the right hand side, the value is decreasing for sure, but something strange is happening, right. So, it means that your program is not correct, all right. We just took the equations and we actually plugged it in, we did not make any change in the way we write our update equations. It seems like something is happening on the right hand side, but when you see distortions like this, usually it is a symbol of something big, all right.

(Refer Slide Time: 32:09)

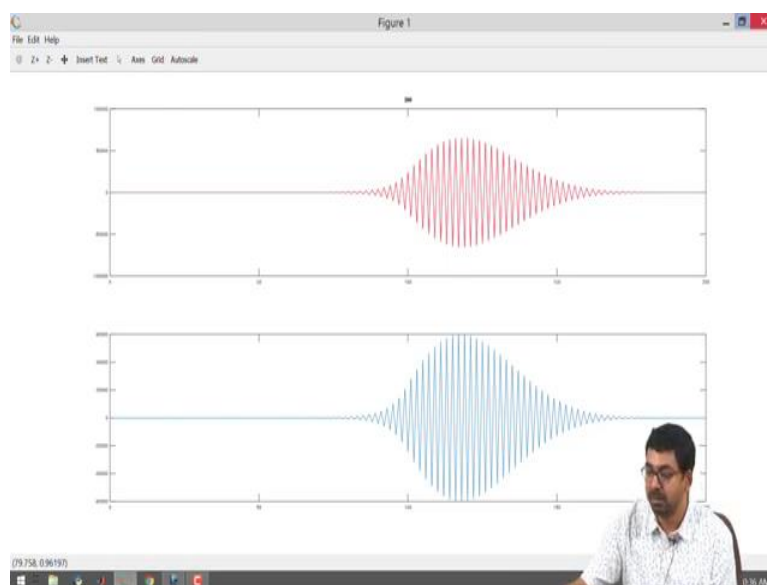


```
10 t_start=0;
11 t_end=200;
12 deltat=1;
13 deltax=1;
14 l(100:200,1)=1;
15 c(100:200,1)=1;
16 r(100:200,1)=0.01;
17 g(100:200,1)=0.01;
18
```

The screenshot shows the Octave environment with a workspace containing variables like 'c', 'deltat', and 'deltax'. The code in the editor defines a simulation with a time range from 0 to 200, a time step of 1, and a spatial step of 1. It sets initial conditions for a wave pulse and defines parameters r and g as 0.01.

So, to make it even bigger, what I am going to do is, I am going to make it 0.01 and 0.01 for r and g instead of 0.001. And it should blow up.

(Refer Slide Time: 32:15)

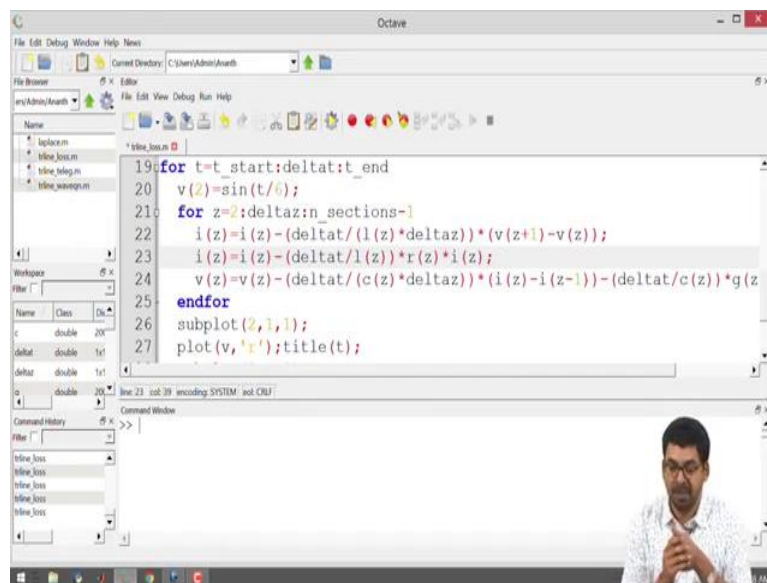


So, it is going to hit the interface and very quickly it will be distorted. While I can see that the envelope is shrinking, something absurd starts to happen and that is the problem with the way we are writing the code. Even though you have taken the exact update equations and

written, a lot of care has to be taken as to what you do, where. That is the problem with these things, every single thing you add or subtract has to be carefully considered, all right.

So, then we know that the problem cannot be with the entire update equation, it was working fine and if r is equal to 0 and g is equal to 0 it still works fine, but only when I start adding r and g it starts to do something weird. That means, the problem is with the terms that I have added, the question naturally becomes is it ok to add just terms like this to your update equation. Because you are having Δt divided by $i(z)$ multiplied by r of z , $i(z)$ this is the way we have written.

(Refer Slide Time: 33:59)

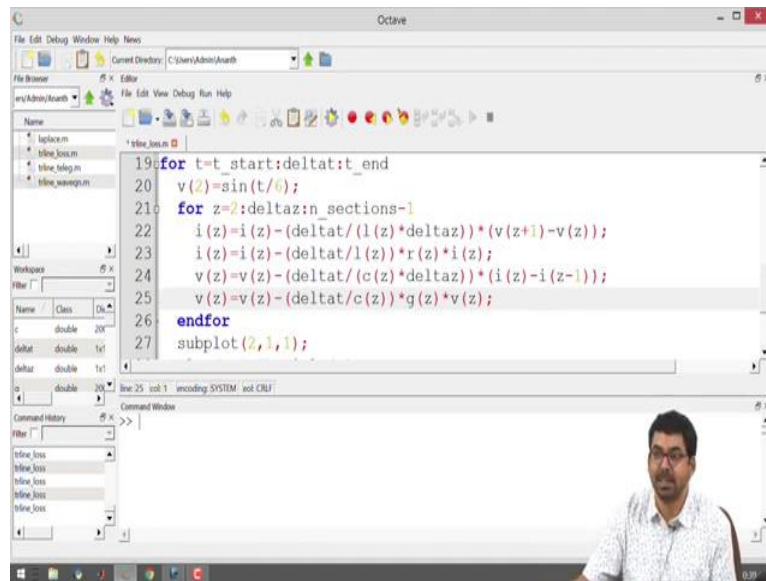


```
19: for t=t_start:deltat:t_end
20:     v(z)=sin(t/6);
21:     for z=2:deltaz:n_sections-1
22:         i(z)=i(z)-(deltat/(l(z)*deltaz))*(v(z+1)-v(z));
23:         i(z)=i(z)-(deltat/l(z))*r(z)*i(z);
24:         v(z)=v(z)-(deltat/(c(z)*deltaz))*(i(z)-i(z-1))-(deltat/c(z))*g(z);
25:     endfor
26:     subplot(2,1,1);
27:     plot(v,'r');title(t);
```

Now, is it correct to write it like this all right or right, it starts to matter a lot, all right. I have taken the term that I have added to a new line, but it is not the same, all right. We know that we have calculated the new value of i , and from that new value of i we are subtracting Δt divided by Δz times $r(i)$.

Previously we were when it was, you are using the old value, this detail also matters ok when you are doing this. So, it may be that sometimes you make changes to the program and you get some unexpected results, it is not because you are doing it incorrectly, it is just because of the way it works, all right. It needs some thought as to when you add some term or subtract some extra term, you need to be very careful where you are doing what, ok. Now, I will do this for the voltage term also, ok.

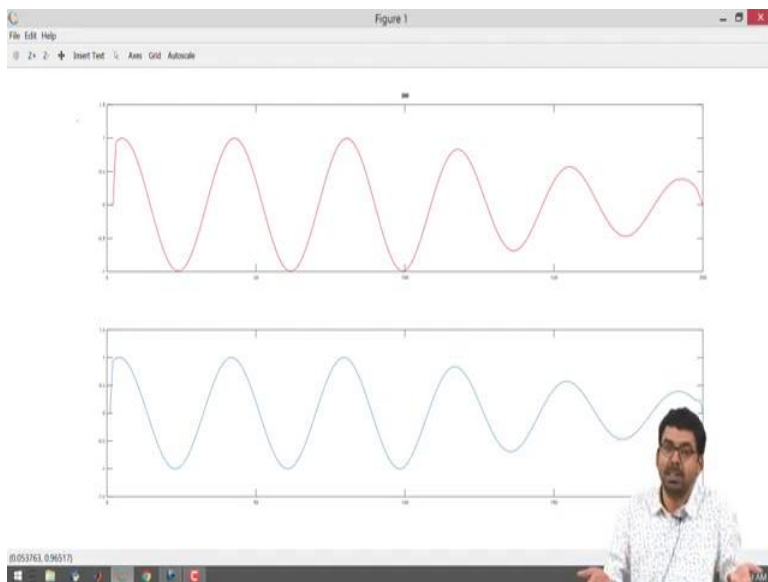
(Refer Slide Time: 35:09)



```
19: for t=t_start:deltat:t_end
20:   v(z)=sin(t/6);
21:   for z=2:deltaz:n_sections-1
22:     i(z)=i(z)-(deltat/l(z)*deltaz)*(v(z+1)-v(z));
23:     i(z)=i(z)-(deltat/l(z))*r(z)*i(z);
24:     v(z)=v(z)-(deltat/c(z)*deltaz)*(i(z)-i(z-1));
25:     v(z)=v(z)-(deltat/c(z))*g(z)*v(z);
26:   endfor
27:   subplot(2,1,1);
```

And then I will run this again, all right. And then we will see if that fixes anything, all right.

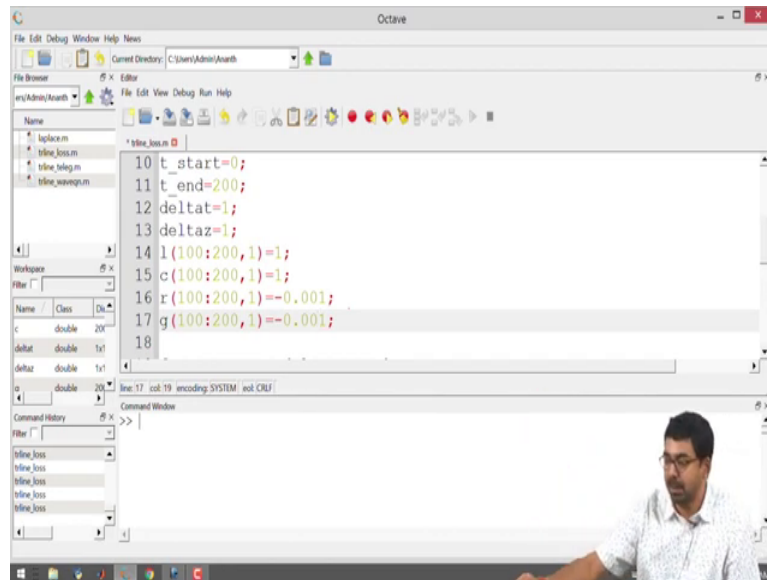
(Refer Slide Time: 35:41)



So, your wave is travelling now in a medium where it is being attenuated, all right. It is very clear that the envelope is shrinking exponentially, all right. So, that seems to have fixed something. So, all I wanted to point out is the way you do add and subtract terms to your update equation needs extreme care, but is this a great program, all right. Let us test the program even a little bit more.

We talked about attenuation constant being positive, we also talked about attenuation constant can be negative in an active system, where you provide external power to amplify the voltage and the current wave. An equivalent in the electrical engineering domain can be an operational amplifier, where you are trying to amplify the voltage or the voltage difference between the pins. Because we are running simulations, we can always plug in different values of r and g , all right.

(Refer Slide Time: 37:09)



```
10 t_start=0;
11 t_end=200;
12 deltat=1;
13 deltax=1;
14 l(100:200,1)=1;
15 c(100:200,1)=1;
16 r(100:200,1)=-0.001;
17 g(100:200,1)=-0.001;
18
```

And one of the ways to do that is by making minus 0 points, because it is going to exponentially go up high. So, I am just going to make it 0.001, 0.001. So, I am just making r and g to be negative, all right. It is just a model, right to tell you that the wave is going to actually gain an amplitude as it travels, I am going to run this.

(Refer Slide Time: 37:29)



But I notice that there is some symptom already, the symptom is that I am noticing these absurd peaks that are coming on top of the sinusoid, even for a tiny bit of gain that I have put in, so how do we go about this? Now we have solved the problem for the positive and negative values of r and g , there seems to be no big issue, all right.

But for the negative values, I still would like to be able to do what I mean to simulate what is going on, all right. So, then we have to look at the program and then actually think ok granted that I have separated the effect of r and g from the l and c , ok. So, I have a one-line corresponding to the effect of l and c and the next line corresponding to the way you are changing for the effect of r .

Questions that can be ask is, when you start and go into the loop you have a value of the voltage that your setting it to be $\text{Sin}\left(\frac{t}{6}\right)$ and the first thing you are doing is calculate the current, ok. And then you are going ahead, correcting the current with some attenuation and then you are calculating the voltage and then you are correcting for the voltage. Now is this the correct way, one can always argue, all right.

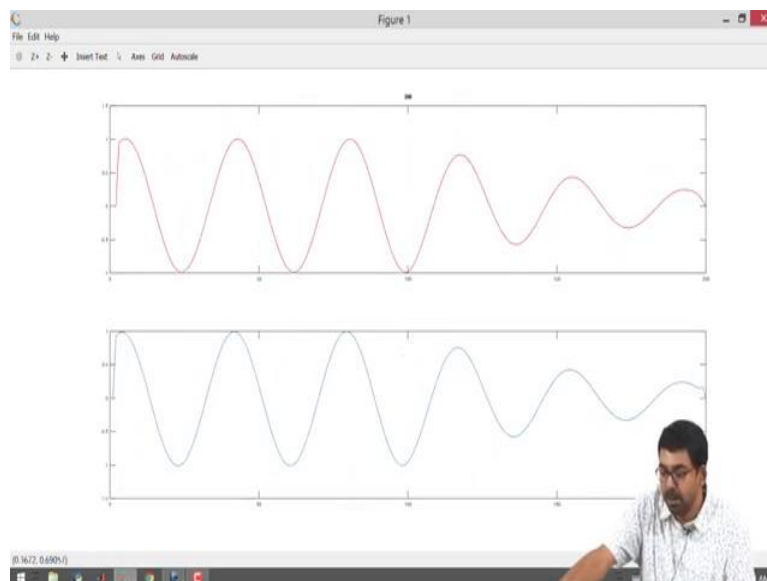
(Refer Slide Time: 39:21)

```
17 g(100:200,1)=0.01;
18
19 for t=t_start:deltat:t_end
20 v(z)=sin(t/6);
21 for z=2:deltaz:n_sections-1
22 v(z)=v(z)-(deltat/c(z))*g(z)*v(z);
23 i(z)=i(z)-(deltat/l(z)*deltaz)*(v(z+1)-v(z));
24 i(z)=i(z)-(deltat/l(z))*r(z)*i(z);
25 v(z)=v(z)-(deltat/(c(z)*deltaz))*(i(z)-i(z-1));
```

We can make some small changes to this, all right. And you can say that, the voltage correction has to happen before you even calculate the current, because you are using the value of the voltage near update equation. It is a very very minor detail that you would not have noticed, if you had not plugged you know negative quantities of r and g, otherwise your code would have worked fine and you would be perfectly fine, right. But now we are going into some small details, all right.

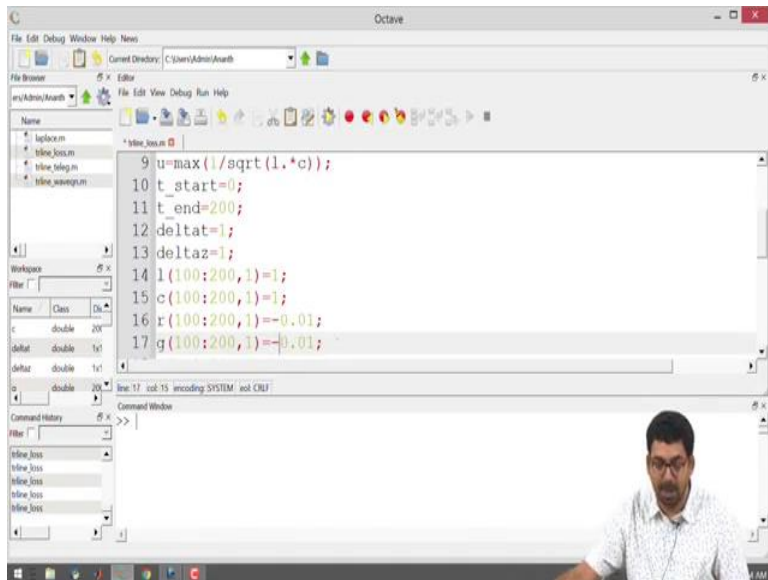
Detail here is I want to be able to just simulate tiny negative values of r and g and find out what is going to happen. Now, I have moved you know one line to another place ok and you know I want to just be able to run this and see what happens.

(Refer Slide Time: 40:11)



You know, but maybe I have negative ok, first it is let see when it is positive if it works or not, ok. So, it seems to be working for positive values of r and g, then let me just quickly switch to a negative value of r and g.

(Refer Slide Time: 40:53)



```
9 u=max(1/sqrt(1.^c));
10 t_start=0;
11 t_end=200;
12 deltat=1;
13 delta_z=1;
14 l(100:200,1)=1;
15 c(100:200,1)=1;
16 r(100:200,1)=-0.01;
17 g(100:200,1)=-0.01;
```

And the symptoms are appearing again, ok.

(Refer Slide Time: 40:57)

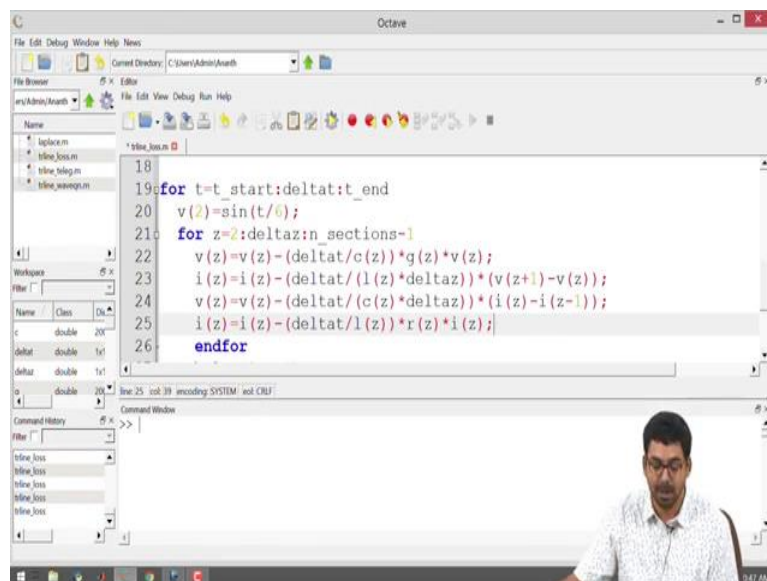


So, I am just telling you that, the way this has to be done has to be really really precise. So, in the previous classes I had given you an impression that, I will just use forward difference

someplace, backward difference someplace, I just plug in something and that is it, it is very simple.

Most of the time the devil is in the details, as we start adding more details, the more carefully you have to think about which line goes where, in what order will you update something all these need some thought. Unfortunately, in this course we do not have time to go into these great details, but I still want you to have a program with which you can verify what is what we are going to be doing with the theory in the class, ok.

(Refer Slide Time: 42:17)



```
18
19: for t=t_start:deltat:t_end
20   v(z)=sin(t/6);
21   for z=z_start:deltaz:n_sections-1
22     v(z)=v(z)-(deltat/c(z))*g(z)*v(z);
23     i(z)=i(z)-(deltat/l(z)*deltaz)*(v(z+1)-v(z));
24     v(z)=v(z)-(deltat/c(z)*deltaz)*(i(z)-i(z-1));
25     i(z)=i(z)-(deltat/l(z))*r(z)*i(z);
26   endfor
```

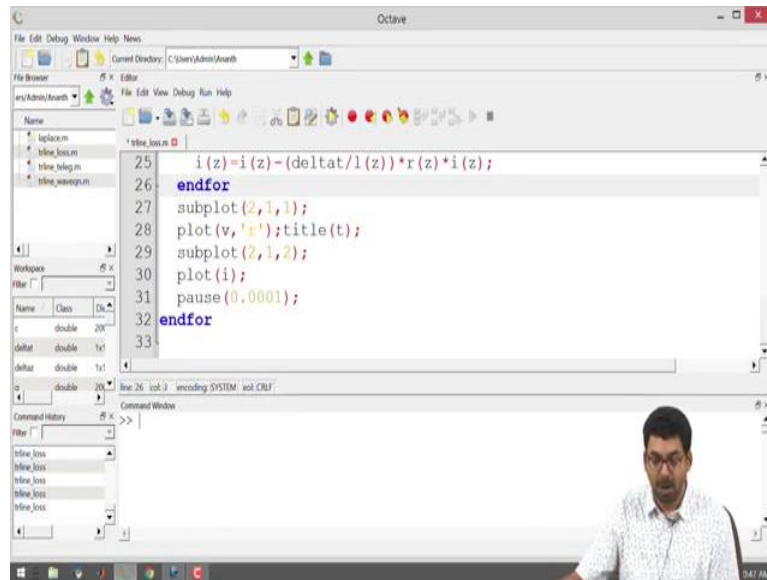
So, the medicine for this is you know take the current update and do it after the voltage, ok. Seems like we are doing things in a very arbitrary manner, we just move one thing from here, one thing from there, and then we are trying to see when, if the simulation works, that is when I will stop. For now, that is ok, as long as a simulation works and you are able to tweak and you are able to verify the analytical results it is ok.

But I still want you to understand that, if you go by simple as you know finite difference and then try to do the code, you should expect a lot of problems to happen like this. And the problem that I am going over is, one your y axis can blow, that means, that you have an issue with delta d and delta z.

And the other is, if you have symptoms which are showing you, you know distortions of the sinusoid, that means, the order in which you are updating things are not correct. These are

the two things, once you know the source, then you can fix it. Even by trial and error by taking different terms and plugging it in different places, you will be able to figure it out within a few you know the trials, all right.

(Refer Slide Time: 43:21)

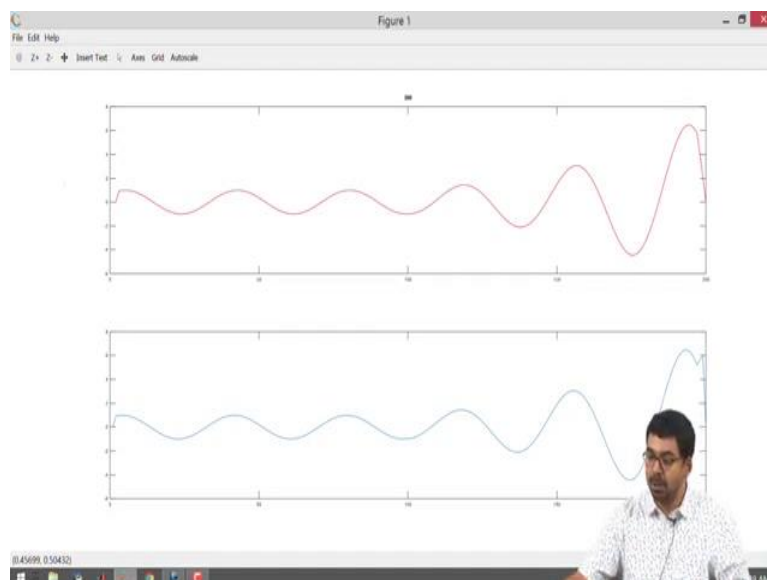


```
25 i(z)=i(z)-(deltat/l(z))*r(z)*i(z);
26
27 endfor
28 subplot(2,1,1);
29 plot(v,'r');title(t);
30 subplot(2,1,2);
31 plot(i);
32 pause(0.0001);
33 endfor
```

The screenshot shows the Octave environment with a workspace containing variables like 'l', 'r', 'deltat', and 'v'. The code in the editor window is a loop that updates the signal $i(z)$ and plots it in two subplots. The top subplot shows the signal v in red, and the bottom subplot shows the signal i . A small video inset of a man is visible in the bottom right corner.

Let us go ahead and see what is happening here.

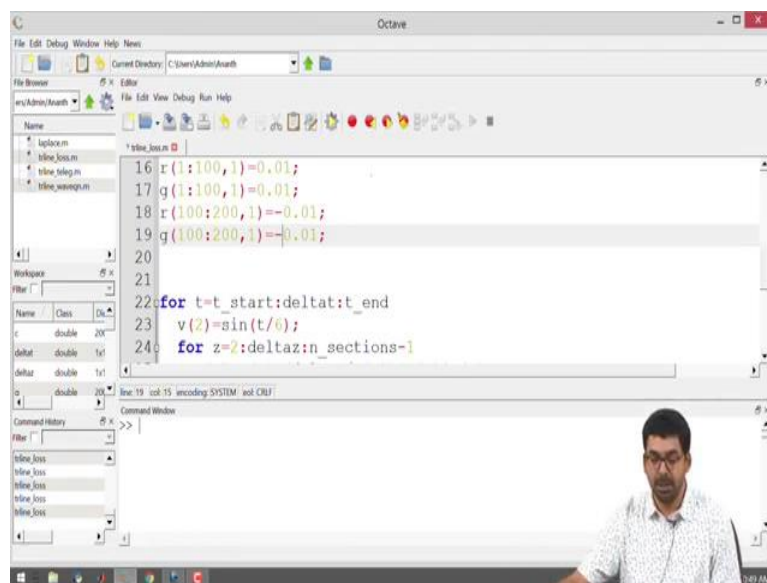
(Refer Slide Time: 43:25)



All I want to do is watch that again to happen, seems to be fine, ok. So, r negative, g negative in a transmission line will give an alpha which is negative all right, and that will mean that, as it travels in the second medium, it is going to gain power, all right.

Now, in the previous class one of the questions that the students had was, why do we even need spatial information, why not just you know make it with respect to time and not worry about it. The reason I did all these things was just to illustrate that particular aspect.

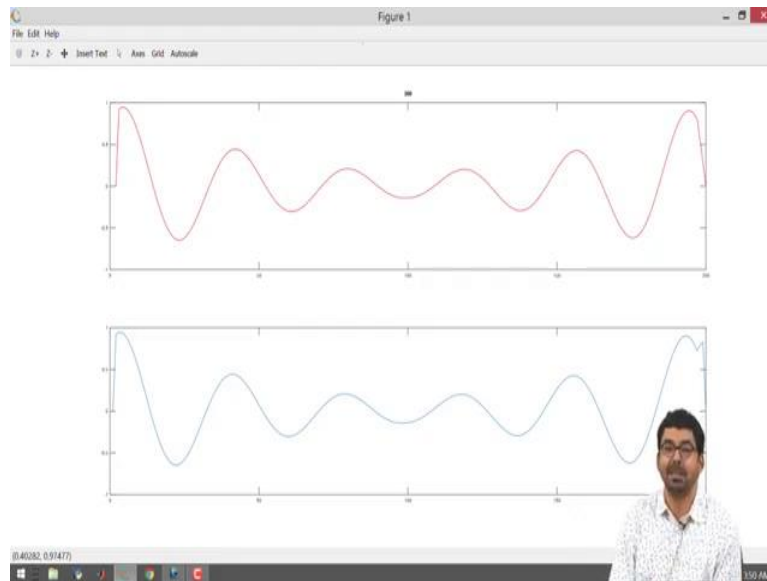
(Refer Slide Time: 44:31)



```
16 r(1:100,1)=0.01;
17 g(1:100,1)=0.01;
18 r(100:200,1)=-0.01;
19 g(100:200,1)=-0.01;
20
21
22 for t=t_start:deltat:t_end
23     v(z)=sin(t/6);
24     for z=2:deltaz:n_sections-1
```

So, what I am going to do now is, I am going to make r from 100 to 200 comma an ok r from 100 to 200 is I am going to make it lossy in the first section. So, I am going to make this from 1 to 100, ok. Going to have an attenuation in the first section and I am going to have a gain in the second section, all right.

(Refer Slide Time: 45:19)



And I am just going to ask, that should answer your question, why we need the spatial information. Suppose I had a transmitter on the left side which is producing the source of this and I had a receiver on the other end. A receiver will say that, yes I have received the information intact, but something has gone into a transmission line in between, there was a decay and then there was a gain to make it look like the information that came from the left side exactly went to the right side.

So, a casual observer on the right side of your simulation will think that I am getting the exact information. There is also another way of looking at it, suppose I just had a loss, I did not have the gain at all and the magnitude goes to zero, a receiver will receive no signal with respect to time. The assumption that you make when you do not receive any signal is, no signal has been sent.

So, you need to know what is happening inside the transmission line and you need to know the spatial information and the temporal information. The spatial information and temporal information together will tell you what is the wavelength in different places, what is the loss in different places, what is the velocity in different places, how much delay should I know to design my circuit to withstand all these things are important.

But one of the more important things that I have given you is a tool where you can plug in some values of r l g c just to understand what is going on. Mind you this program is also not

perfect, this program will also do strange things under different circumstances, but it is a good point, good starting point, all right.

The details of this usually have a higher level graduate course, where you work out the mathematical details of each and everything, all right and that will be appropriate. But for now, you have something that many students do not have, a starting point to just fiddle around with transmission line parameters, calculate reflection coefficients, calculate the attenuation, calculate the gain, as a result do some calculations on the outcome, all right. So, we will stop here.