

Computational Physics
Dr. Apratim Chatterji
Dr. Prasenjit Ghosh
Department of Physics
Indian Institutes of Science Education and Research, Pune

Lecture - 34
Partial Differential Equations Part 02

(Refer Slide Time: 00:16)

$T_{i,j} = \frac{(\Delta x)^2 (\Delta y)^2}{2[(\Delta x)^2 + (\Delta y)^2]} \left[\frac{T_{i+1,j} + T_{i-1,j}}{(\Delta x)^2} + \frac{T_{i,j+1} + T_{i,j-1}}{(\Delta y)^2} \right]$

If $\Delta y = \Delta x$

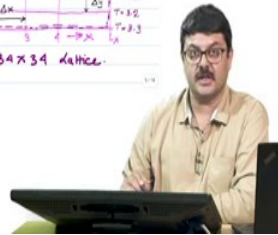
$T_{i,j} = \frac{1}{4} [T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}]$

IN ADDITION: You need Boundary Conditions (B-C). → FINITE DIFFERENCE METHOD.

When no explicit dependence.

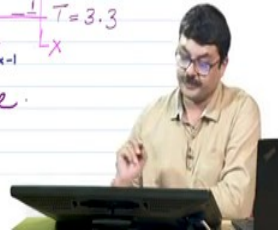
$\nabla^2 T = 0 \Rightarrow \frac{\partial^2 T}{\partial x^2} = 0 \rightarrow$ BUT \rightarrow

$\frac{dT}{dx} = C_1 \Rightarrow \int T = C_1 x + C_2$
 and B-C: $T = T_0$ at $x=0$
 $T = T_1$ at $x=L_x$



(Refer Slide Time: 00:23)

34×34 lattice.
 $L_x \times L_y$.

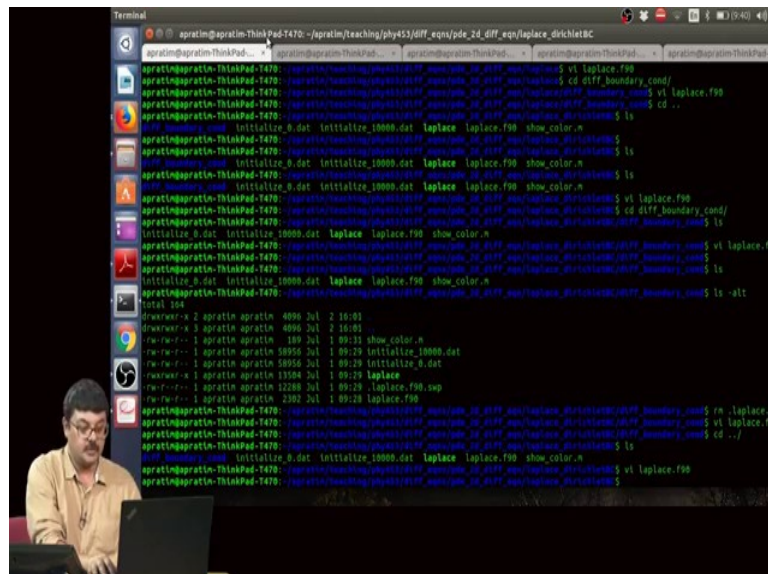


So, before moving to the computer, just let us look at this figure in greater detail. So, what we have is basically the same picture as in the last page, but and here this is basically the lattices drawn and what we shall do while we write the code is that the temperature along these boundaries, they are essentially fixed right. And so, we are not going to update those points.

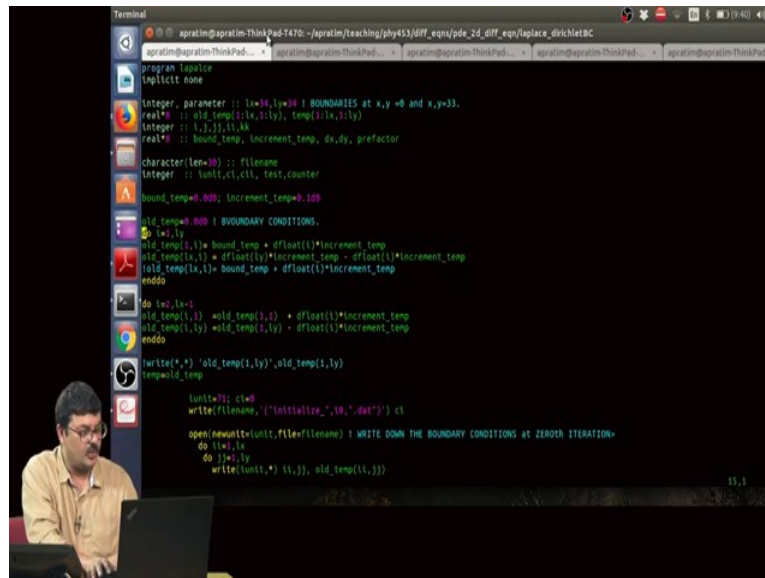
Instead what we are going to update are: this lattice point, this lattice point, like this lattice point, basically whatever is there between this blue line. So, we are going to update the temperature at these at the points within this blue line, because at the boundaries here and here and here and here basically they are fixed the temperatures are fixed they are boundary conditions, they are not going to be updated the temperature are going to remain fixed because their boundary conditions.

What will change are the temperature in these points right, in the middle of the plate and that will evolve with iterations. So, while iterating the loop shall go from this point and basically from this point to this point to this point to this point and not over the entire lattice right and we are basically have a 34 cross 34 lattice. So, the loop shall essentially go from 2 to 33 right or $l \times \text{minus } 1$, again 2 to $l \times \text{minus } 1$ and so on.

(Refer Slide Time: 02:00)



(Refer Slide Time: 02:13)



```
program laplace
implicit none
integer, parameter :: ix=34, iy=34 ! BOUNDARIES at x,y =0 and x,y=33.
real*8 :: old_temp(:,1:ix), temp(:,1:iy)
integer :: i,j,jj,ikk
real*8 :: bound_temp, increment_temp, dx,dy, prefactor
character(len=100) :: filename
integer :: iunit,c1,c2, test, counter
bound_temp=0.00; increment_temp=0.100
old_temp=0.000 ! BOUNDARY CONDITIONS.
do i=1,iy
old_temp(i,1)=bound_temp + dfloat(i)*increment_temp
old_temp(i,ix)= dfloat(i)*increment_temp - dfloat(i)*increment_temp
old_temp(i,1)= bound_temp + dfloat(i)*increment_temp
enddo
do i=1,ix-1
do j=1,iy-1
old_temp(i,j) =old_temp(i,j) + dfloat(i)*increment_temp
old_temp(i,j) =old_temp(i,j) - dfloat(i)*increment_temp
enddo
enddo
write(*,*) 'old_temp(1,ly)',old_temp(1,ly)
temp=old_temp
iunit=70; c1=0
write(filename, '( "install", "10", ".dat" )') c1
open(newunit=iunit, file=filename) ! WRITE DOWN THE BOUNDARY CONDITIONS AT ZEROth ITERATION=
do (i),ix
do (j),iy
write(iunit,*) i,j, old_temp(i,j)

```

And now to look at the code so, the code is called v i Laplace dot f 90, because we are solving a Laplacian essentially with the appropriate boundary conditions right and the program is named Laplace implicit none as you know l x I have said to be 34 l y to be 34 that is a square lattice. We are going to update the temperature and store the temperature in a new array.

So, the so, we are going to store the temperature in this lattice 1 comma l x and 1 comma l y and after an update I am going to store the value of this temperature into the old temperature using the old temperature we are going to calculate new value of temperature over the entire lattice in the iteration right. So, that is what that is why we have two arrays. These are dummy variables i j j j k k and so on so forth.

Now, here basically I am giving the boundary temperature, the increment in temperature that I shall give along the boundaries basically 0.1 right and so, what I am doing I have defined the values of d x and d y and though we have should not be using them, because basically the expressions becomes independent of d x and d y. But if you had chosen finite values of d x and d y or d x and d y was different then you would have to use the more complicated expression which I had shown earlier right.

(Refer Slide Time: 03:40)

```

Integer, parameter :: lx=10,ly=10 ! BOUNDARIES at x,y =0 and x,y=10
real*8 :: old_temp(:,ly), temp(:,ly)
Integer :: i,j,ii,kk
real*8 :: bound_temp, increment_temp, dx,dy, prefactor

character(len=30) :: filename
Integer :: iunit,c1,c11, test_counter

bound_temp=0.000; increment_temp=0.100

!id temp=0.000 ! BOUNDARY CONDITIONS.
do i=1,ly
  old_temp(i,1)= bound_temp + dfloat(i)*increment_temp
  old_temp(i,ly) = dfloat(i)*increment_temp - dfloat(i)*increment_temp
  old_temp(i,1)= bound_temp + dfloat(i)*increment_temp
enddo

do i=1, lx-1
  old_temp(i,1) = old_temp(i,1) + dfloat(i)*increment_temp
  old_temp(i,ly) = old_temp(i,ly) - dfloat(i)*increment_temp
enddo

write(*,*) 'old_temp(i,ly)',old_temp(i,ly)
temp=old_temp

iunit=1; c1=
write(filename, '( "iterating_", i0, ".dat" )') c1

open(newunit=iunit,file=filename) ! WRITE DOWN THE BOUNDARY CONDITIONS AT ZEROth ITERATION=
do i=1,ly
  do j=1,ly
    write(iunit,*) i,j, old_temp(i,j)
  enddo
enddo
close(iunit)
!laplace.F90: 000, 2195C written
10,10

```

So, here I am specifying the boundary conditions and from 1 to 1 y that is along the y direction for each points in the y direction and this loop is over 1 to 1 y. So, for x equal to 1 right that is the left side the left side of the square the left side of the square where x equal to 1 for different values of y, I am saying that the temperature is going to increase from bound temperature bound temperature has been set to 0 and the increment in tough temperature is 0.1.

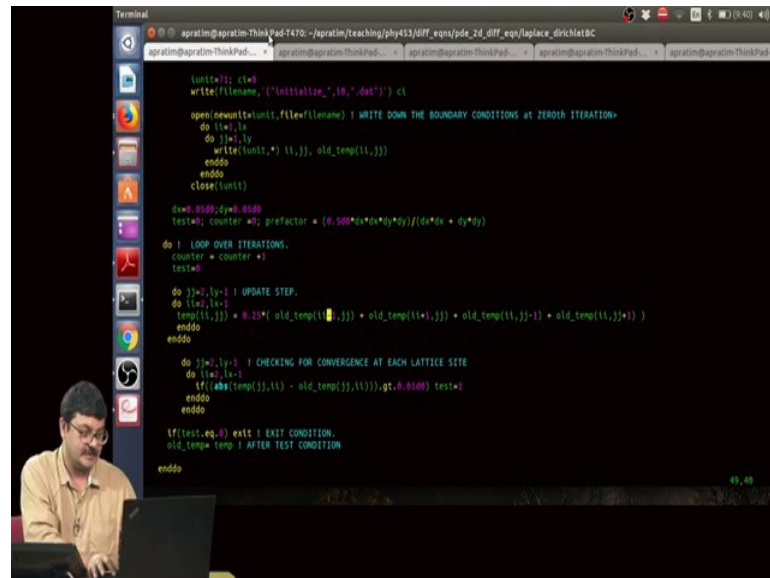
So, as you increase j along y direction at different points along the boundaries, you are going to have essentially 0, 0.1, 0.2, 0.3, 0.4 up till the top at basically 1 to 1 y right. And similarly, on the right side of the box when x equal to 1 x, I am again changing the different y coordinates and there what I am doing is basically I am going to decrease the temperature right if you remember.

So, it is deep float minus increment into 1 y into temperature which is the maximum temperature possible 3.3 and as you increase i right; as you increase i the temperature is constantly decreasing along the y direction. This is at x equal to 1 x. Now in the other two so, when you change x right for y equal to 1 and y equal to 1 y which is the two ends of the box right like will the two ends of the box. So, you have to specify the temperature as you change x and that is exactly what is being done here.

Basically I have already given the temperature remember at the corners in this loop so, that is why this loop is from 2 to 1 x minus 1 and I am suitably incrementing and decrementing decreasing the temperature at x equal to 1, that is the lower plate. I am

increasing the temperature and along at y equal to l_y which is the top at the top of the plate. I am decreasing the temperature just as we discussed in the basically when we were discussing the schematics right.

(Refer Slide Time: 06:34)



```
Terminal
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC
apratim@apratim-ThinkPad: ~
apratim@apratim-ThinkPad: ~
apratim@apratim-ThinkPad: ~
apratim@apratim-ThinkPad: ~
apratim@apratim-ThinkPad: ~

lunit=:; cl=0
write(filename, ("initialize_"//lunit//".dat")) cl

open(newunit=lunit, file=filename) ! WRITE DOWN THE BOUNDARY CONDITIONS AT ZEROth ITERATION>
do (lx), lx
  do (ly), ly
    write(lunit,*) (i, j), old_temp(i, j))
  enddo
enddo
close(lunit)

dx=0.500; dy=0.500
testw; counter =; prefactor = (0.500**dx**dy)/(dx**dx + dy**dy)

do ! LOOP OVER ITERATIONS.
  counter = counter + 1
  testw
  do (j)=1, ly-1 ! UPDATE STEP.
    do (i)=1, lx-1
      temp(i, j) = 0.25* old_temp(i-1, j) + old_temp(i+1, j) + old_temp(i, j-1) + old_temp(i, j+1) )
    enddo
  enddo

  do (j)=1, ly-1 ! CHECKING FOR CONVERGENCE AT EACH LATTICE SITE
    do (i)=1, lx-1
      if(abs(temp(j), i) - old_temp(j), i)) .gt. 0.500) testw=
    enddo
  enddo

  if(test.eq.0) exit ! EXIT CONDITION.
  old_temp= temp ! AFTER TEST CONDITION
enddo

49,40
```

So, this is what is I am setting the boundary conditions right and then I am saying rest of the lattice I have set the temperature to be 0. I am going to set the temperature to be 0 and I think there is set somewhere here well I should give so, I had set old temperature equal to 0 over the entire lattice here boundary conditions and then along the boundaries I specify the temperature.

Now I am storing I am I have copied old temperature to temperature here right and I want to see what I have given as the initial conditions right and that I am storing basically in a file called initialize underscore 0 dot dat c i equal to 0 if you see here right. So, it is initialized 0 that so, I want to see what is the initial condition and well here I have set dx equal to 0 point and dy equal to 0.5, but we are not going to use this because dx equal to dy . So, the final expression for the update of temperature does not have dx and dy right and the actual iterations to find the final temperature is being done here.

So, after writing down the initial condition, basically initial condition before the start of the iterations where everything is 0 except at the boundaries. So, here I am counting just like the previous class when we are doing a 1 d loop, here we are there is a do loop here

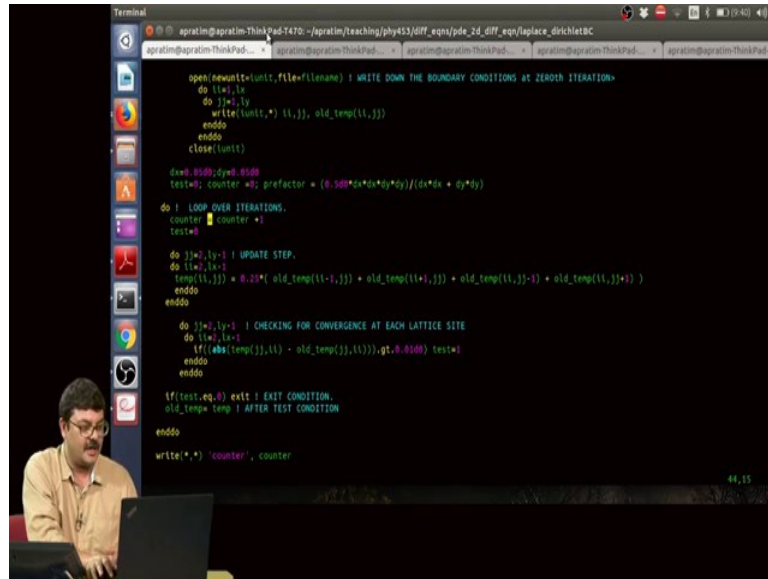
right with no specified number of iterations this do loop ends here. Now here we are going to scan each point in the lattice leaving out the boundaries. So, j goes from 2 to $l_y - 1$ i goes as a dummy variable i goes from 2 to $l_x - 1$ and temperature at i, j at any particular i and j or rather i and j is 1 by 4 into these temperatures the previous temperatures at $i - 1, j$ and here this is $i + 1, j$, the two neighbours on plus x and minus x and here on plus y and minus y right or vice versa this is plus y and this is minus y .

Note here I am doing it old temperature because I am using the previous value of temperature and when i equal to 2 c right. So, $i - 1$ so, i is 2 so, $i - 1$ is 1 which means we are using the boundary temperature right. So, to update suppose i equal to 2 and j equal to 2, you are going to use also the temperature of 1 2 and 2 1 which are at the boundaries and those shall come from $i + 1$ or rather $i - 1$ and $j - 1$ all right. So, we have in this step we have updated in one iteration the temperatures within the lattice within the blue line if you remember the what I showed to us the end of the lecture right and here I am checking for the tolerance.

So, here basically I am going again from each lattice site except the boundaries 2 to $l_y - 1$, 2 to $l_x + 2$ to $l_x - 1$ and if the temperature at each of these points or rather the difference in temperature the updated temperature is temp, the previous temperature is old temp for each value of j and i if this difference is greater than a certain tolerance value 0.01 right I set a dummy variable test equal to 1 this test had the beginning of each loop this is just after the beginning of loop is set equal to 0.

And if the difference in temperature at any point along the lattice is greater than this tolerance value you can set it to a 0.01 or 0.00012 to the minus 4 then test equal to 1 and only if the difference in temperature in one iteration for each of the lattice points is less than 0.01 then test will remain equal to 0 else test will be set to 1. If test equal to 0 exit the loop this is the exit condition right. And however, if you do not exit the loop; that means, test has become equal to 1. Then you set old temp equal to temp and redo this entire calculation, find out the new values of temperature for all the lattice points right.

(Refer Slide Time: 12:13)



```
Terminal
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqs/pde_2d_diff_eqs/laplace_dirichletBC
apratim@apratim-ThinkPad:~$ cat laplace_dirichletBC.f90
open(newunit=lunit, file=filename) ! WRITE DOWN THE BOUNDARY CONDITIONS AT ZEROth ITERATION>
do i=1,ix
do j=1,iy
write(lunit,*) i,j, old_temp(i,j))
enddo
enddo
close(lunit)

dxx=dx/dx;dyy=dy/dy
test=0; counter = 0; prefactor = (0.500*dx*dx*dy*dy)/(dx*dx + dy*dy)

do ! LOOP OVER ITERATIONS.
counter = counter + 1
test=0

do j=1,iy-1 ! UPDATE STEP.
do i=1,ix-1
temp(i,j) = 0.25*( old_temp(i-1,j)) + old_temp(i+1,j)) + old_temp(i,j-1) + old_temp(i,j+1) )
enddo
enddo

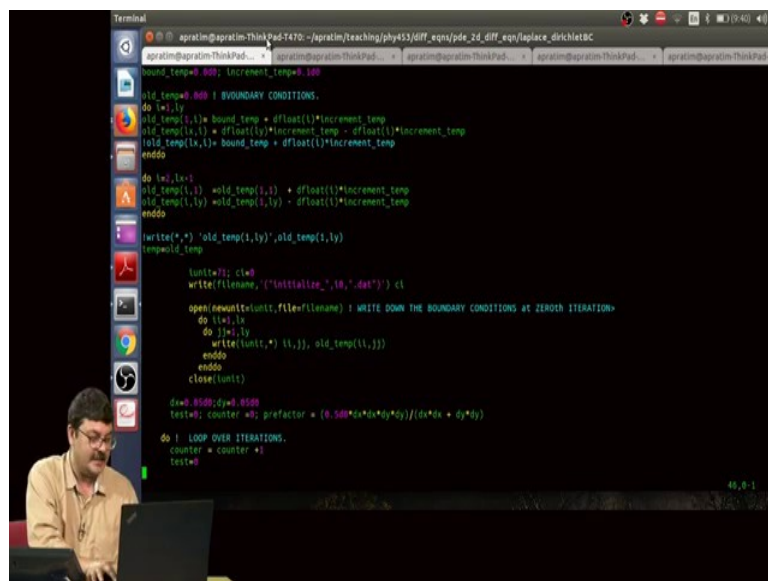
do j=1,iy-1 ! CHECKING FOR CONVERGENCE AT EACH LATTICE SITE
do i=1,ix-1
if (abs(temp(j,i) - old_temp(j,i)))>.01 then test=1
enddo
enddo

if(test.eq.0) exit ! EXIT CONDITION.
old_temp= temp ! AFTER TEST CONDITION

enddo

write(*,*) 'counter', counter
44,15
```

(Refer Slide Time: 12:35)



```
Terminal
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqs/pde_2d_diff_eqs/laplace_dirichletBC
apratim@apratim-ThinkPad:~$ cat laplace_dirichletBC.f90
!BOUNDARY CONDITIONS.
old_temp= 0.0; increment_temp= 1.00

old_temp= 0.0 ! BOUNDARY CONDITIONS.
do i=1,iy
old_temp(i,1)= bound_temp + dfloat(i)*increment_temp
old_temp(i,ix)= dfloat(iy)*increment_temp - dfloat(i)*increment_temp
old_temp(i,1)= bound_temp + dfloat(i)*increment_temp
enddo

do i=1,ix-1
old_temp(i,1) = old_temp(i,1) + dfloat(i)*increment_temp
old_temp(i,iy) = old_temp(i,iy) - dfloat(i)*increment_temp
enddo

write(*,*) 'old_temp(1,iy)',old_temp(1,iy)
temp=old_temp

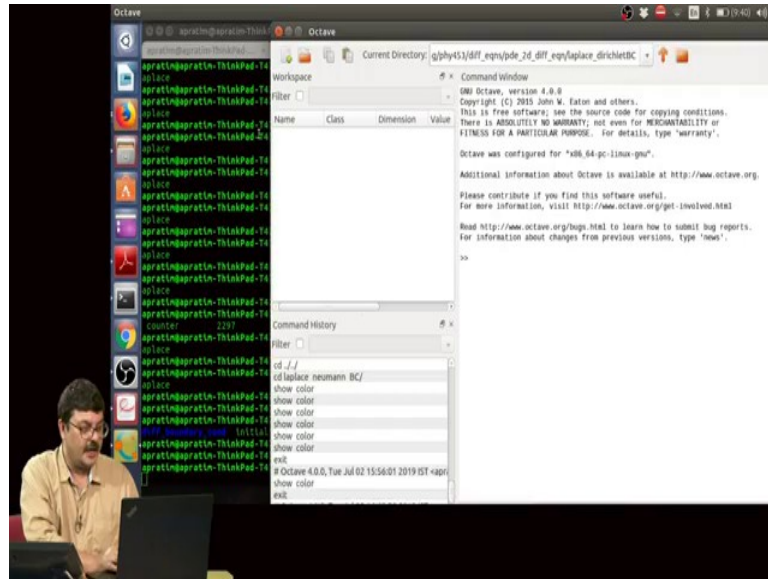
lunit=1; clw=
write(filename, ('lattice_' || i0 || '.dat')) cl
open(newunit=lunit, file=filename) ! WRITE DOWN THE BOUNDARY CONDITIONS AT ZEROth ITERATION>
do i=1,ix
do j=1,iy
write(lunit,*) i,j, old_temp(i,j))
enddo
enddo
close(lunit)

dxx=dx/dx;dyy=dy/dy
test=0; counter = 0; prefactor = (0.500*dx*dx*dy*dy)/(dx*dx + dy*dy)

do ! LOOP OVER ITERATIONS.
counter = counter + 1
test=0
48,8-1
```

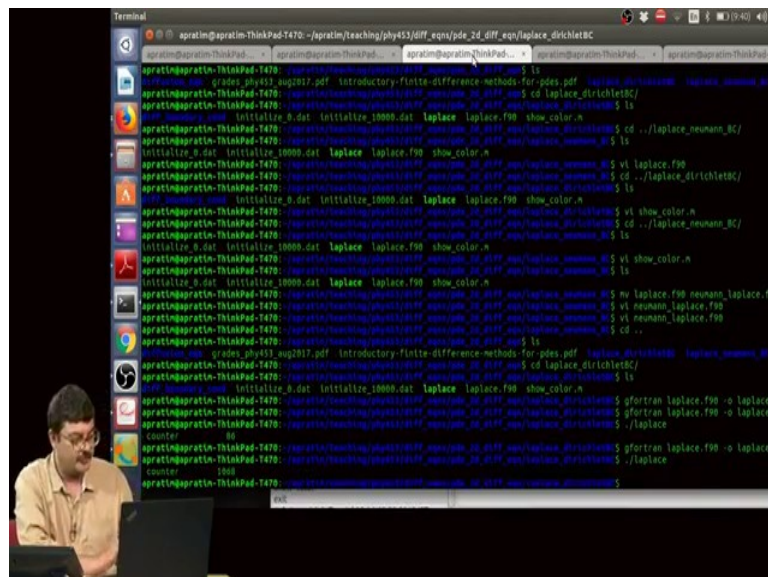
Once you come out of the loop, you have to you just want to know in how many iterations did you get your converged answer. The counter is set equal to 0 somewhere in the beginning of the code and each time the code goes through this do loop counter is incremented by 1. And you make the computer the code write down the value of the counter after you have come out of the loop. At the end what do you want? You want to write down the value of the temperature at each of these lattice points right the value of temperature at each of these lattice points in a file.

(Refer Slide Time: 14:16)



How are you going to visualize it? You can use it using matlab what I have in this computer is octave which is basically an open source version of mat lab. You do not have to buy octave you can just download it and what I do is basically so, I have already octave installed in my computer and so, if I just type octave, I will get; I will get a screen like this right.

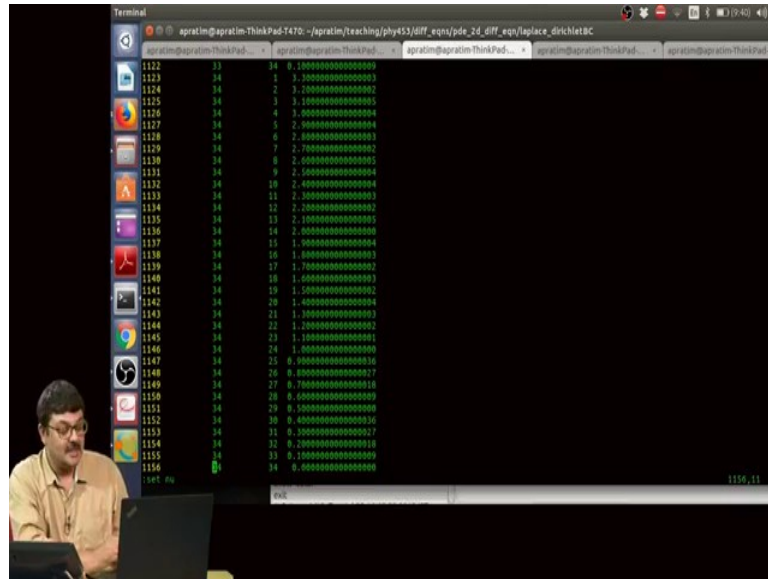
(Refer Slide Time: 14:27)



And just to show you to visualize this data right so, what do I have at the end I have essentially, I will just compile it and show you. So, I have to basically g Fortran and

Laplace dot f 90 minus or Laplace, then I run my code right and in 86 iterations it has converged. But of course, if you increase the tolerance to say sorry decrease the tolerance to this value right, then you can again compile and if you run it you need around 1068. So, 1068 iterations before you get the converged results of the temperature across the lattice right and the data is written as I said in initialize 1000 dot dat.

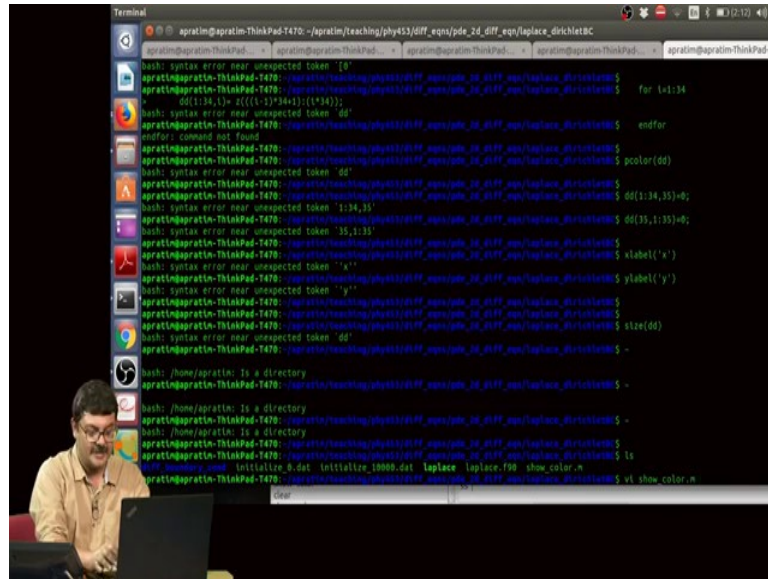
(Refer Slide Time: 15:31)



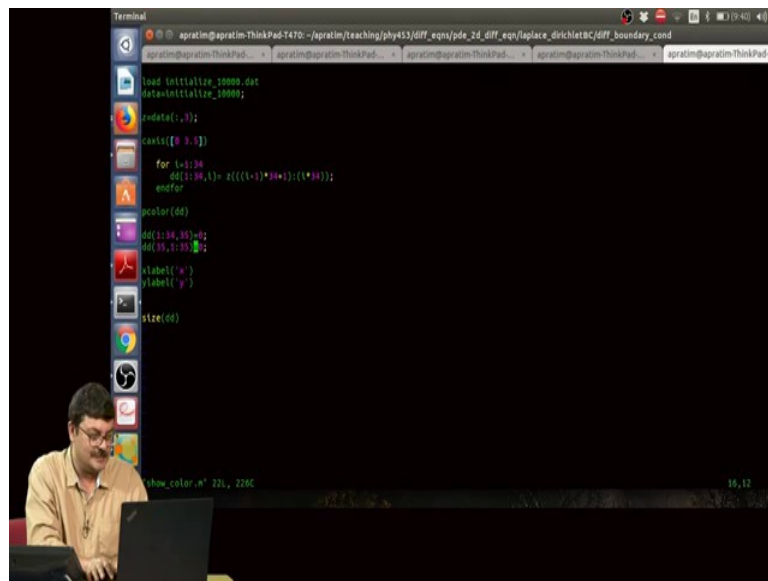
So, these are a basically 1 1 1 2 at each lattice point the lattice goes so, this is basically x coordinate this is y coordinate decreasing from 1 to 34 and you write down the temperature. The temperature as you know you have set from 0.1, 0.2, 0.3, 0.4 up till at the boundary it is 3.4 and this is for x equal to 2 you have changing values of y a j if you like j j if you like, for each value of x.

So, basically for each value of x you are writing down all the values of y and the value of the temperature the converged temperature so, that is what you are writing down so, that is what is there. Now, how do I visualize. So, I mean this is a huge file of line 1156 lines 34 cross 34 I guess is 1156. Now what do we do with this data? We want to know what is the temperature profile. So, we want to visualize the data in a x y lattice.

(Refer Slide Time: 16:53)



(Refer Slide Time: 17:10)



To do that I shall be using octave as I told you right and rather than write down my commands every time I have written down the commands to visualize this data in a file called show colour dot m. The dot m is important for running in matlab or an octave you must name your file which ends with dot m and all that I have done here is basically I have said load initialize 1000 dot dat right, which means the software will upload the data to itself from the hard disk of this laptop.

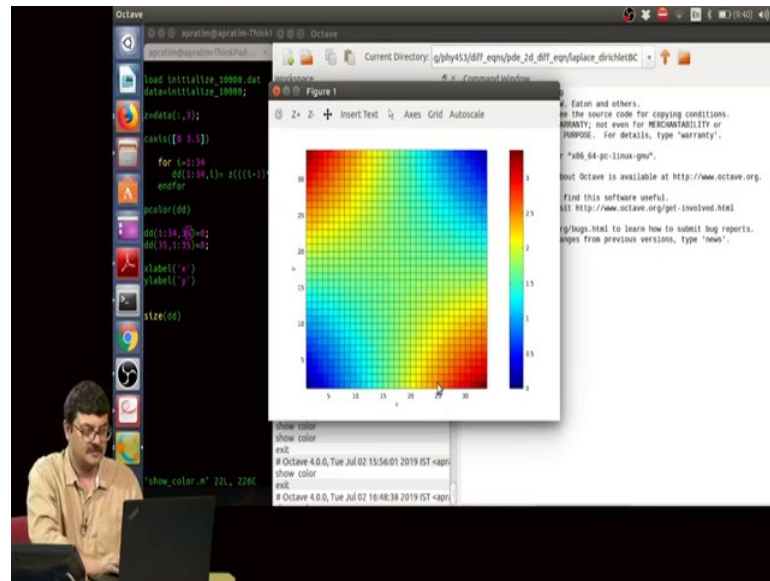
And then I have said store or whatever is there; whatever is there inside initialize dot dat, save it in a array basically 1156 cross 3 times length array and store it in a variable called data. I mean you was just naming that I am going to use data to store the data from here. Then so, all the data which is here has been stored here now. Now you are saying that z, so, let there be a 1 dimensional array called z and the third column the third column and all the lines in the third column, you store it in z. So, third column if you remember was the value of the temperature at each of these lattice points from 0 to 34 for each value of x and y.

I am setting so, I am going to draw a colour map ok. I am going to draw a colour map and the range of the colour map will be from 0 to 3.5 and that is what I am setting in this command called c axis. Now to plot I have to tell octave that for each value of x and y the temperature should also be stored as a xy, I mean it should be stored in a square variable right now from z equal to 3 it is stored in a linear. It is stored in a linear array right because that is how I wrote down the data when I used when I wrote it down in the code right. Like x y temperature at x y and that currently is a linear array to that I must convert it to a square array.

So, that when x and y is specified the temperature is plotted and that is being done here from do i equal to 1 comma 34 d d is some variable. So, basically 1 to 34th line and so, basically d d 1 to 34 for a particular value of i stores the appropriate values from that long column. Just think about it how am I storing it. I am saying from i minus 1 into 34 plus 1 to i into 34, 34 of my columns and I changes from 1 to 34 is being stored in this square array.

This is a linear array of length 1156 and then I do p colour. P colour basically says plot. So, basically plot d d which is now a square array here and stores the value of the temperature for each x and y; x label x x label y and its telling me the size of this array. Here you see that I have intentionally added an extra column and a row to d d even 35 this is basically to help me in visualization.

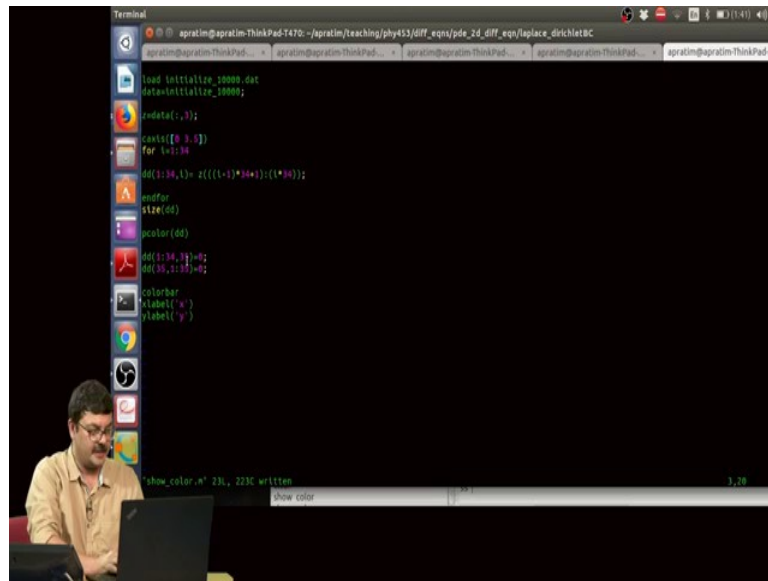
(Refer Slide Time: 20:55)



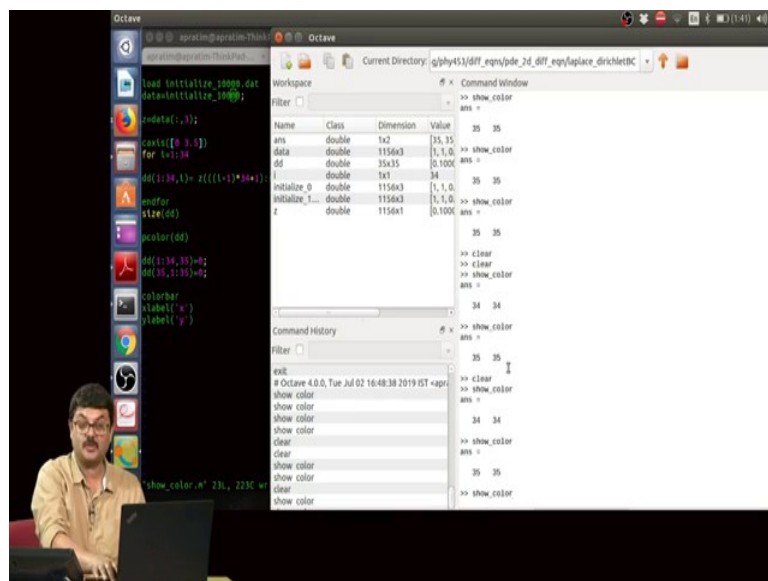
Without much ado let me show you how the colour the temperature profile will look. So, this is your colour bar right, which says that blue is 0 and basically this deep red a corresponds to temperature of 3.5 which I specified is in c axis. Here if you remember the initial condition that we have given is it should increase from 3 to 304. Here the temperature was decreasing from 3.4 to 0. From 0 again it was increasing to 3.4.

So, those are the boundary conditions and we now solve the Laplacian iteratively and now we have the temperature at each point within the lattice. So, basically if you want to know what is the temperature at this point, you have to just look its around 1.5 say right and if its yellow then you know that the temperature is around 2 or slightly more than 2 between around 2.3. And here you know that the temperatures are higher and here the temperatures are cooler because that is how we have set the boundary conditions.

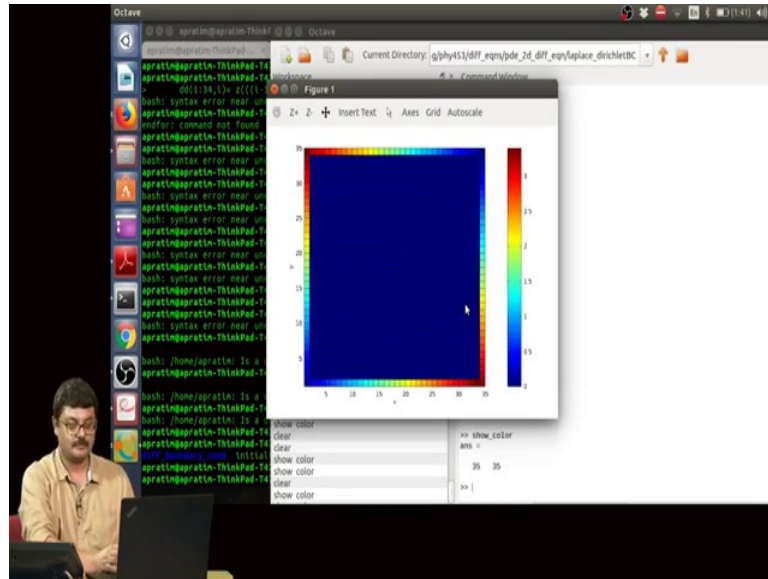
(Refer Slide Time: 22:21)



(Refer Slide Time: 22:35)



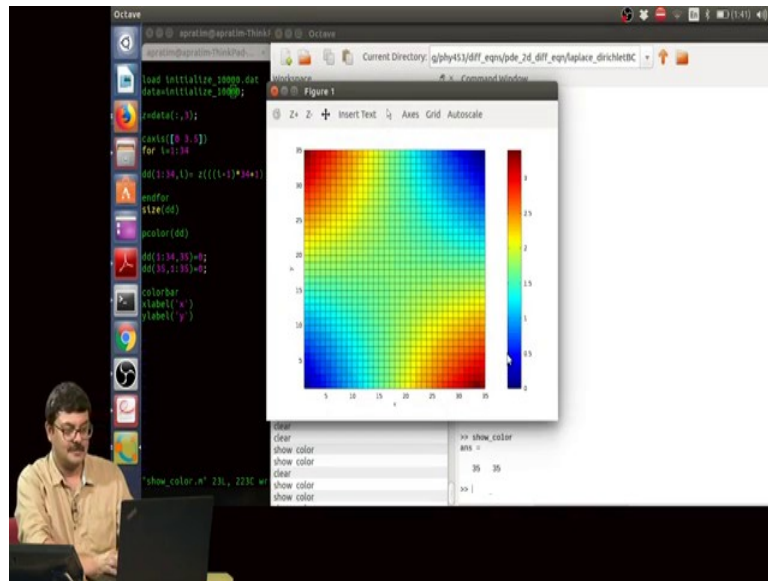
(Refer Slide Time: 22:38)



Now, suppose if you want to see the initial condition right so, which is stored in a file called initialize 0 dot dat right and it is basically this initialize 0 dot dat. Now, if I want to visualize that one the initialize condition so, that is basically here I have to go and change the file name to initialize 0 dot dat. And if I want to visualize it, it is show colour and this was your initial condition.

The temperature here was increasing, decreasing here increasing and here it was increasing right and this was what this was your right at the beginning of the iteration, everything inside the lattice except the boundaries had been set to 0 and after suitable evolution what we had was the data the converged data is stored here right. And if we just want to visualize that this is what you have already seen before this was your final temperature for the boundary conditions which are given.

(Refer Slide Time: 23:23)



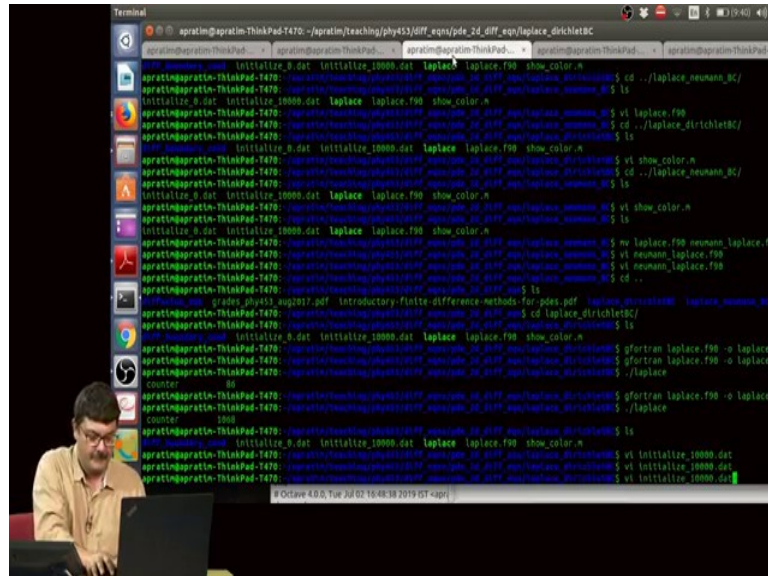
(Refer Slide Time: 23:33)

```
program laplace
implicit none
integer, parameter :: ix=35, iy=35 ! BOUNDARIES at x,y =0 and x,y=35.
real*8 :: old_temp(ix,iy), temp(ix,iy)
integer :: i,j,ii,ik
real*8 :: bound_temp, increment_temp, dx,dy, prefactor
character(len=10) :: filename
integer :: iunit,ci,cil, test_counter
bound_temp=0.00; increment_temp=0.100
old_temp=0.000 ! BOUNDARY CONDITIONS.
do i=1,ix
    do j=1,iy
        old_temp(i,j) = bound_temp + dfloat(i)*increment_temp
        old_temp(i,1) = dfloat(i)*increment_temp - dfloat(i)*increment_temp
        old_temp(i,ix) = bound_temp + dfloat(i)*increment_temp
    enddo
enddo
do i=1,ix-1
    do j=1,iy-1
        old_temp(i,j) = old_temp(i,j) + dfloat(i)*increment_temp
        old_temp(i,j) = old_temp(i,j) - dfloat(i)*increment_temp
    enddo
enddo
write(*,*) 'old_temp(1,iy),old_temp(1,1)'
temp=old_temp
iunit=31; cil=0
write(filename, '( "initialize_",0, ".dat" )' ) cil
open(newunit=iunit,file=filename) ! WRITE DOWN THE BOUNDARY CONDITIONS AT ZEROth ITERATION.
do i=1,ix
    do j=1,iy
        write(iunit,*) i,j, old_temp(i,j)
    enddo
enddo
```

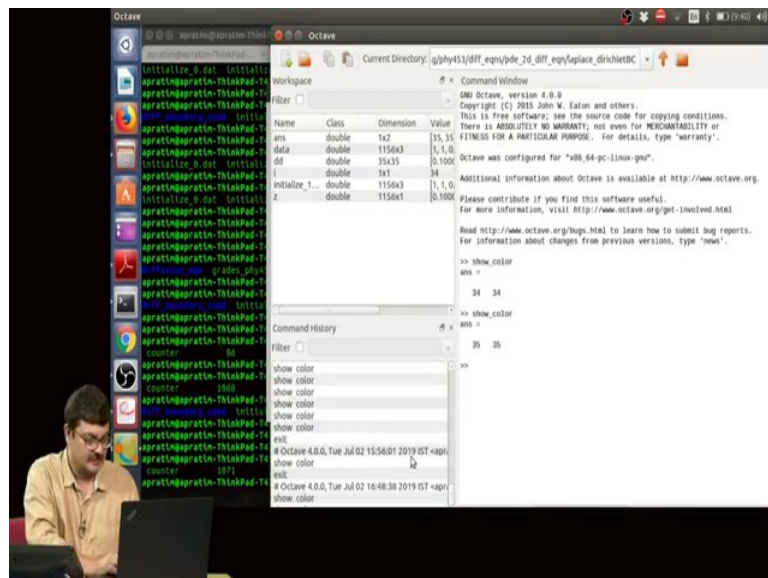
Now suppose we have set the boundary conditions slightly differently and suppose the boundary conditions I have already tried that out where suppose something like this so, basically what I have here is both along x equal to 1 and at x equal to 1 x the temperature is increasing right. And basically along $1 \times$ the top part you again you have the same temperature as previously basically what I have given here is old temperature 1.

Whatever is there on the x axis it should keep on increasing right. So, here I have changed the boundary condition from this line where it was as I specified before where here I am saying it is the same as this rights of the both increasing.

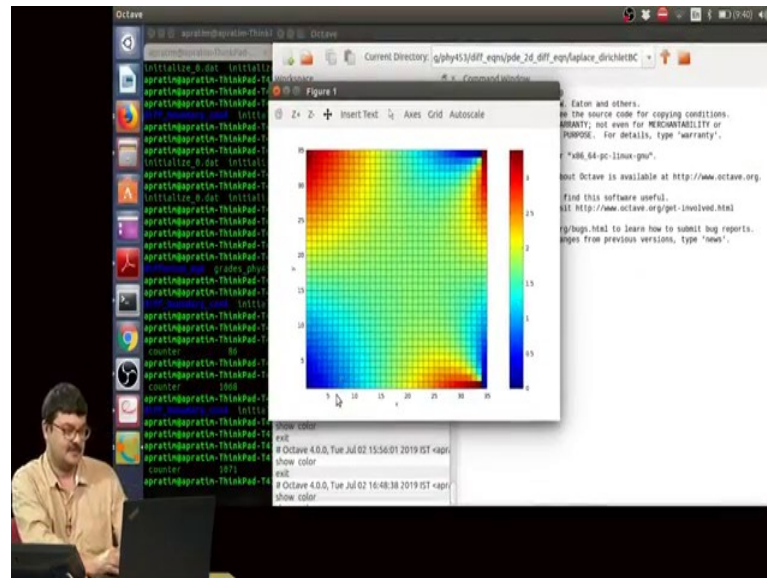
(Refer Slide Time: 24:35)



(Refer Slide Time: 24:52)



(Refer Slide Time: 25:01)

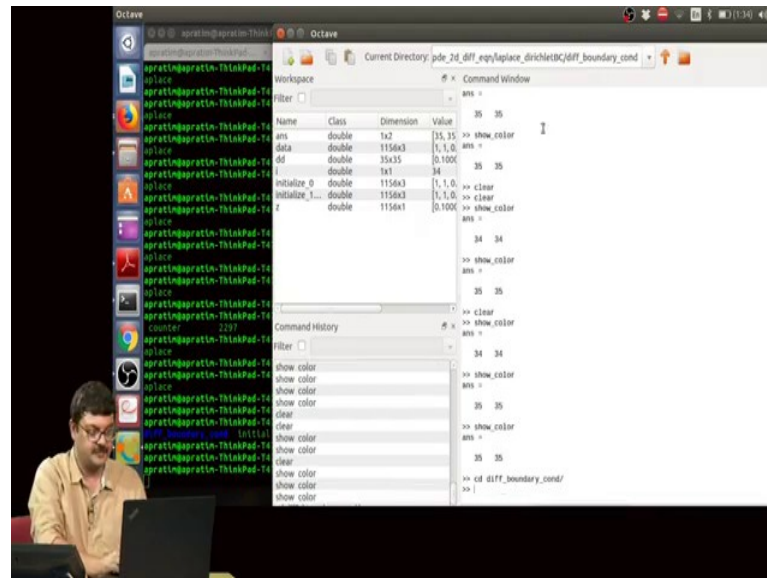


And if you run this now, it converged in 1071 line. And if I just choose octave, I have again save the data and initialize 1000 dot dat what you have is something like this right. So, what I gave is basically the temperature increases from here to here, the temperature increases from here to here in the boundaries, but here I said let the temperature decrease and I forced it by the boundary conditions.

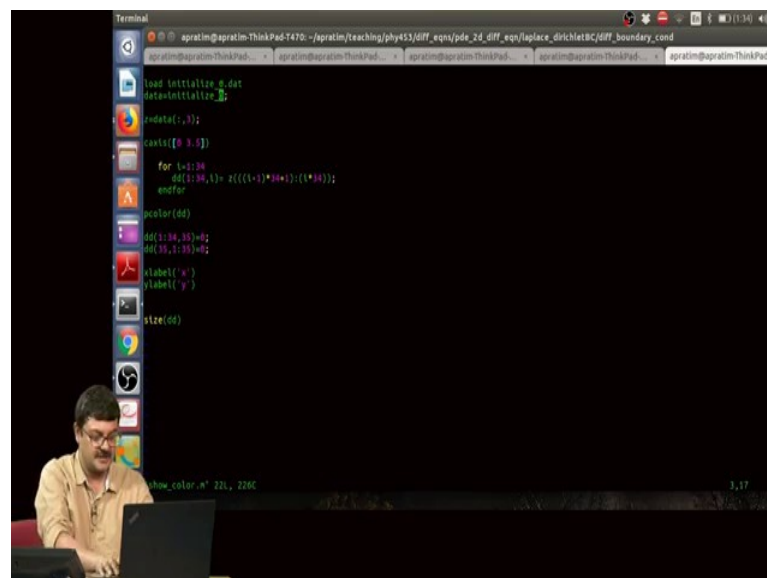
Again here I said that let the temperature increase right. So, I changed only this boundary condition and for this would be the temperature profile and you can read up read out what is the temperature at each of these points from this colour bar right.

Now, if we had slightly more complicated boundary conditions suppose, that along this axis and along this axis I increase the temperature, but along this line and along this line I keep the temperature to be 0 right. And then what would be the temperature profile like? Let us have the look we have to just change the boundary condition I have already done that right.

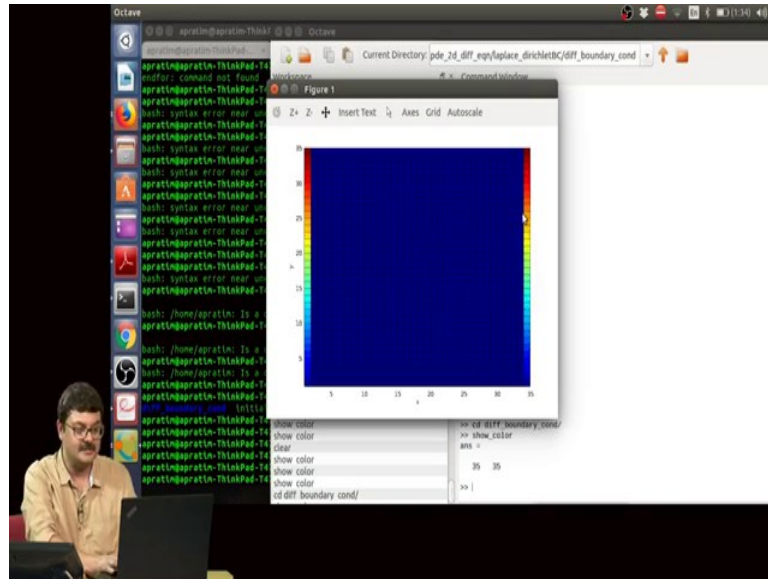
(Refer Slide Time: 27:38)



(Refer Slide Time: 28:02)

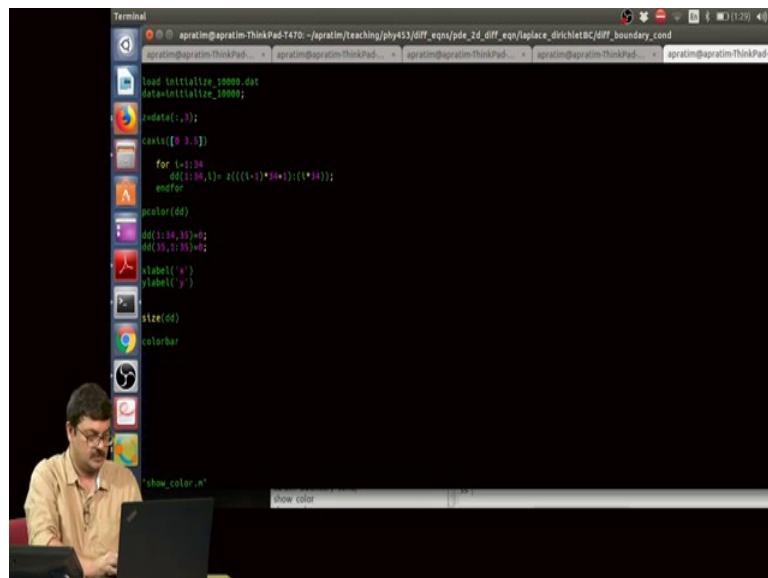


(Refer Slide Time: 28:14)

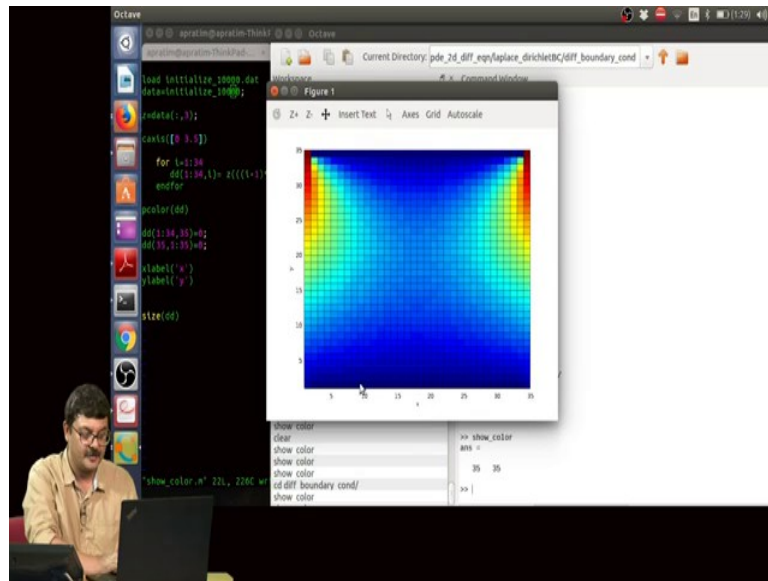


Now, if you run this code and I have to also go here to a different directory if you run this code, it has basically converged in 918 iterations and well let us see what the initial condition is right so, that you know so, this is the initial condition. And this is what I have given, all the temperature inside is 0, here it increases, here it increases along the boundaries I have said them to be 0 and shall remain 0 along this and this they shall remain 0. And after the solution of the Laplacian if you iterate it and the result converges the final temperature profile is going to be something like this right.

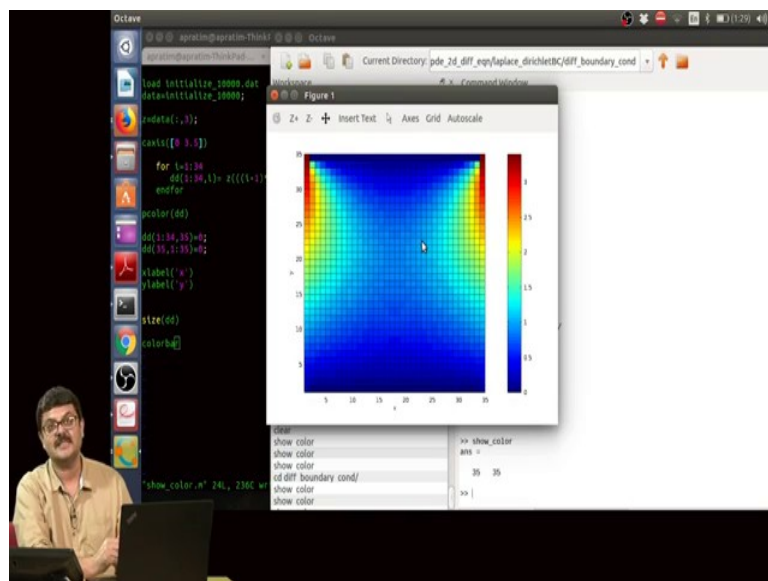
(Refer Slide Time: 28:34)



(Refer Slide Time: 28:51)



(Refer Slide Time: 29:28)



So, here you have high temperatures suitably you have high temperatures here, this along this boundary I had mandated that the temperature should remain 0. Along this boundary I had mandated that the temperature should remain 0 so, here you have a deep blue colours right and corresponding to so, I have put in the colour bar now.

So, the command to put in the colour bar is just colour bar and 0 to 3.5 and now you can see that here you have extremely cold temperatures. Here you have warm temperatures

and the temperature basically goes in up till this point. Here it is again close to 0, but close to these regions with temperatures with very high boundaries you have this profile.

What is the message? You can put in whatever complex boundary conditions you want to fix up the temperature along the boundaries and you can calculate the temperature within the plate right. And I have given three different examples intentionally put to have different boundary conditions and you can get the temperature profile. In the next class we will discuss about some other kinds of boundary conditions.

Thanks.