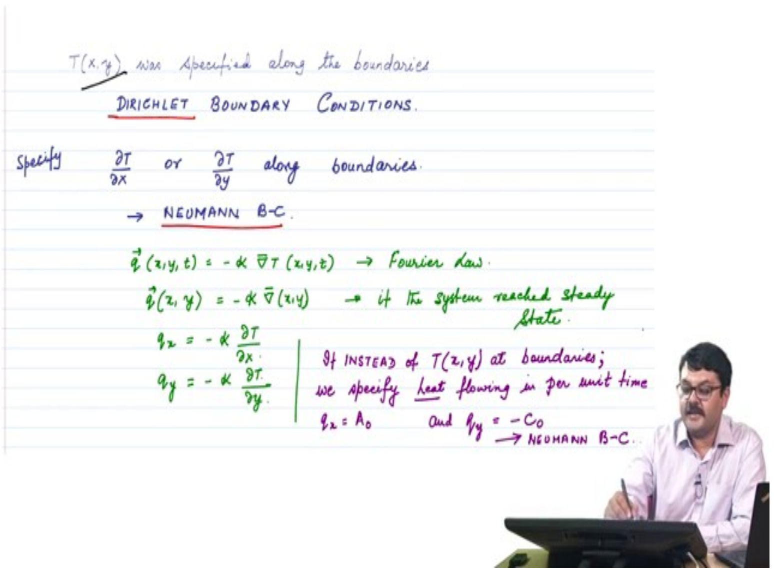
Computational Physics

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

Lecture - 35
Partial Differential Equations Part 03

(Refer Slide Time: 00:16)



$T(x,y)$ was specified along the boundaries

DIRICHLET BOUNDARY CONDITIONS.

Specify $\frac{\partial T}{\partial x}$ or $\frac{\partial T}{\partial y}$ along boundaries.

$\rightarrow$ NEUMANN B-C.

$\vec{q}(x,y,t) = -\kappa \vec{\nabla} T(x,y,t) \rightarrow$ Fourier Law.

$\vec{q}(x,y) = -\kappa \vec{\nabla}(x,y) \rightarrow$ if the system reached steady state.

$q_x = -\kappa \frac{\partial T}{\partial x}$.

$q_y = -\kappa \frac{\partial T}{\partial y}$.

If INSTEAD of $T(x,y)$ at boundaries; we specify heat flowing in per unit time

$q_x = A_0$ and $q_y = -C_0$

$\rightarrow$ NEUMANN B-C.

So, welcome back to the class. So, in the last class what we discussed was solving the Laplacian equation for T x y right and there the temperature was specified along the boundaries, along the four boundaries and on a plate we had specified the temperature. And we solved self consistently for the points inside the plate leaving out the boundaries and we found the final temperature of the system right; there was no time dependence we had said in the diffusion equation del phi del T was put to 0 we are just looking at this steady state solution.

Now, when you specify the boundary conditions and you specify the temperature which is the quantity we are solving for at the boundaries it is called the Dirichlet boundary condition right. So, it is a name when you specify T as a or phi whatever quantity you are solving for at the boundaries. On the other hand you can also specify that d T d x, the d T dx at the boundaries or d T d y along the boundaries, which is the first derivative of the field that were solving for, in our case it was temperature.

And when you specify d T d x or d T d y along the boundaries then you call it the Neumann boundary condition. Now why or what does it mean by specifying d T d x or the gradient the first derivative of the field we are solving for d T d x or d T d y at the boundaries, what does Neumann boundary condition imply; in our case it would imply that we are actually specifying the heat current.

And why is that? Let me remind you about the Furriers law which we had discussed in the last class. And q vector was the heat current and that was proportional to kappa the thermal conductivity, the gradient of temperature and that is the so called furious law from which we obtained the so called diffusion equation for heat transport in a plate, right.

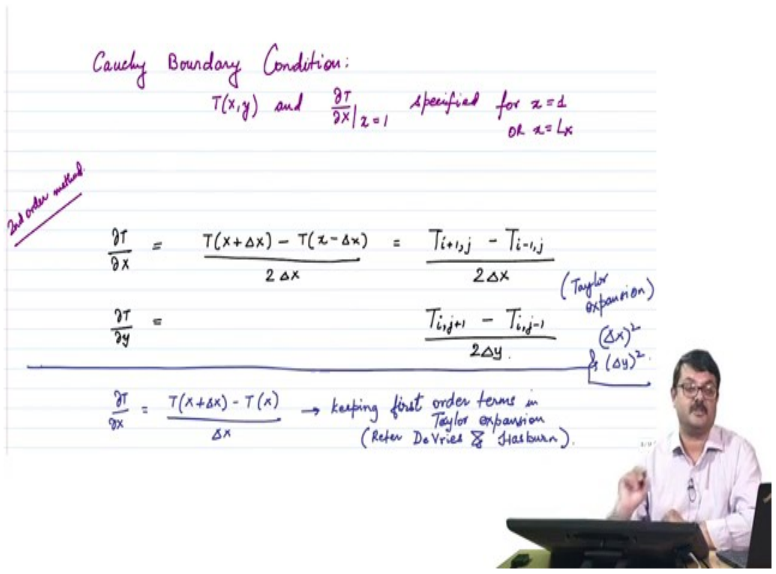
And basically of course, if you specify this and you say that I specify or how much amount of heat is transported at different points in x and y along the boundaries right; then which means, that you are essentially specifying, so there was an error here it should be like this. So, if you specify the basically temperature gradient only at the boundaries, when it is reached steady state you are basically specifying the so called Neumann boundary condition.

So, q x, right, if you specify q x which is basically kappa d T d x, right. So, how much of heat is being transported on the left side of the box along the x direction. So, if you specify this quantity, the heat current along the x direction you can specify that and then you can also specify the heat current along the y direction and the appropriate boundaries.

And then the question is how would you solve for the Laplacian with the appropriate boundary conditions and get your heat or temperature profile as a function of x and y and that is what we are going to discuss today. So, just to summarize if we specify instead of T x y at the boundaries, we specify the heat flowing in per unit time, per unit area at the boundaries.

And you will specify suppose q x equal to A 0 some amount of heat trunk being transported along the boundaries, and q y equal to some value C 0 say along the boundaries; how do we solve the Laplacian and get our temperature profile So, that is what we are going to discuss today.

(Refer Slide Time: 04:50)



Cauchy Boundary Condition:

$T(x,y)$ and $\frac{\partial T}{\partial x}\Big|_{x=1}$ specified for $x=1$ or $x=L_x$

2nd order method

$$\frac{\partial T}{\partial x} = \frac{T(x+\Delta x) - T(x-\Delta x)}{2\Delta x} = \frac{T_{i+1,j} - T_{i-1,j}}{2\Delta x}$$

$$\frac{\partial T}{\partial y} = \frac{T_{i,j+1} - T_{i,j-1}}{2\Delta y}$$

(Taylor expansion) $(\Delta x)^2$ & $(\Delta y)^2$.

$$\frac{\partial T}{\partial x} = \frac{T(x+\Delta x) - T(x)}{\Delta x} \rightarrow$$ keeping first order terms in Taylor expansion (Refer DeVries & Hasburn).

Having said that and before we come down to actually solving a Laplacian with Neumann boundary conditions; let me also tell you that you can also have Cauchy boundary conditions. And in the Cauchy boundary conditions, you specify both T the temperature as a function of x and y only at the boundaries; as well as the gradient d T d x along the boundaries right, and you can specify it along x equal to 1 or x equal to L x and so on so forth.

And you can also specify boundary conditions where the both are specified, this is a more restrictive system, its bit more complicated; we are not going to discuss this, but just for your information you can also specify boundary conditions like this. And you might have when you solve partial differential equations in a real case have to handle such systems and then you have to look up the book and learn how to do it, right ok.

Coming back to our Neumann boundary condition where we are specifying del T del x at the boundaries; just to remind you that as per the second order scheme of Taylor expansion where such a you basically use such expression for d T d x. Just to tell you that though we have not discussed the second order scheme in great detail, we have not discussed it at all for that matter it is there in the books.

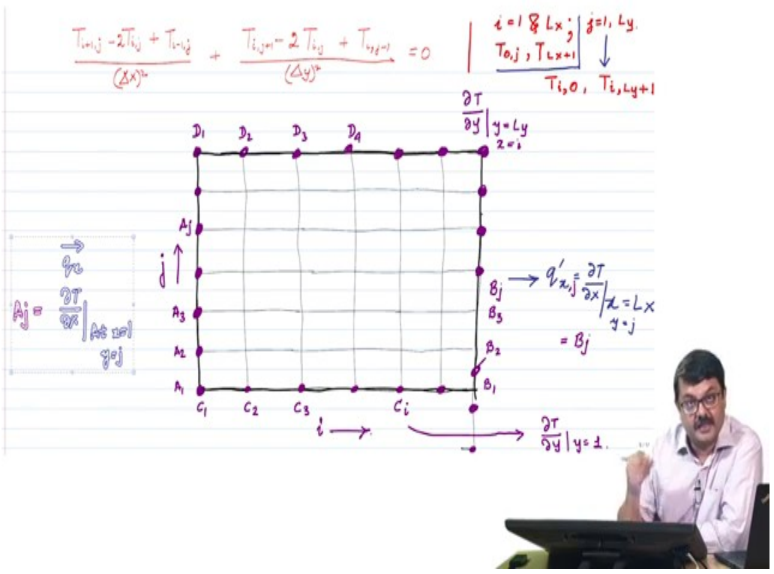
You can have also first order schemes in the Taylor expansion and then you can write d T d x equal to T temperature at x plus delta x minus T at x by delta x and this is the so called first order scheme right. But the expression we are using for del 2 del 2 T del x 2 this is already the second order scheme. So, for consistency I am writing down the expressions of d T d x in this form which is the second order scheme. If you want to know more about it, open a book you will get the derivations and so on so forth and it is obtained through a Taylor expansion, right.

So, you can refer to this book if you want to know more about first order expansion, second order expansions; stability when you mix up terms with first order expansions like this, with second order expansions for the suppose the second derivative d 2 T del x 2 there are certain stability conditions to be maintained, to be checked for.

So, we are not going there, we are just choosing a 2nd order scheme method, the 2nd order method, formula and the d T d x can be written like this; which essentially, when you write it in the discreet manner it is T i plus 1 delta x is i plus 1, j minus T i minus 1 corresponding to x minus delta x j by 2 delta x.

And similarly d T del d T d y can be written as T i j plus 1. So, basically what is the value of the temperature in the y direction, one that is coordinate up and one would lattice coordinate down divided by 2 delta y. So, this is what we have to. So, this is what is essentially specified. So, when you specify Neumann boundary condition, you are specifying del T del x at the boundaries which will have some values and that in turn can be written by these formulas, right in the finite difference method.

(Refer Slide Time: 08:42)



$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta x)^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta y)^2} = 0$
$i = 1 \& L_x;$ $j = 1, L_y$
$T_{0,j}, T_{Lx+1}$
$T_{i,0}, T_{i,Ly+1}$
$\frac{\partial T}{\partial y}\Big|_{y=L_y, x=i}$
D_1 D_2 D_3 D_4
A_j A_3 A_2 A_1
$j \uparrow$
$A_j = \frac{\partial T}{\partial x}\Big|_{At\ x=1, y=j}$
$\vec{q_x}$
$B_j \rightarrow q'_{x,j} = \frac{\partial T}{\partial x}\Big|_{x=L_x, y=j} = B_j$
B_3 B_2 B_1
C_1 C_2 C_3 C_i
$i \rightarrow$
$\frac{\partial T}{\partial y}\Big|_{y=1}$

But now you have a problem and let me tell you what the problem is as well as the solution of course. So, you are Laplacian, right as we discussed in the last class can be written in this expression which stands for basically del 2 T del x 2 and del 2 T del y 2 can be written like this and that is equal to 0; and this is the equation we have to solve for over the entire lattice. Now if this is your lattice right, previously the values of the temperature was specified at the boundaries, at these points say and these points.

So, we were not, they were fixed at the boundaries, the value of temperature was fixed at the boundaries as a consequence we were not changing or updating the value of the temperature at these boundaries because they were considered to remains fixed and we were only updating and finding out the steady state temperature of the points within the lattice.

We were basically solving for those iteratively, but in this case, but in today's class what we are discussing is what is specified is not the temperature, but the heat right. I had suppose this boundary and what is the heat being flowing in or out at this boundary; and what is the heat flowing in or out at this boundary and this is the last boundary, right.

So, the temperatures at these points can change and they can be updated, so that the, so you are specifying the amount of heat input and heat output along the boundaries. And the temperature will evolve as per the solution of the Laplacian equation given the certain boundary conditions and you are solving exactly for that. So, what is specified is, at each point here, here and here and here what are the values of heat current coming in. The simplest case will be the heat coming in will be uniform along this boundary, and the heat going out would be uniform along this boundary.

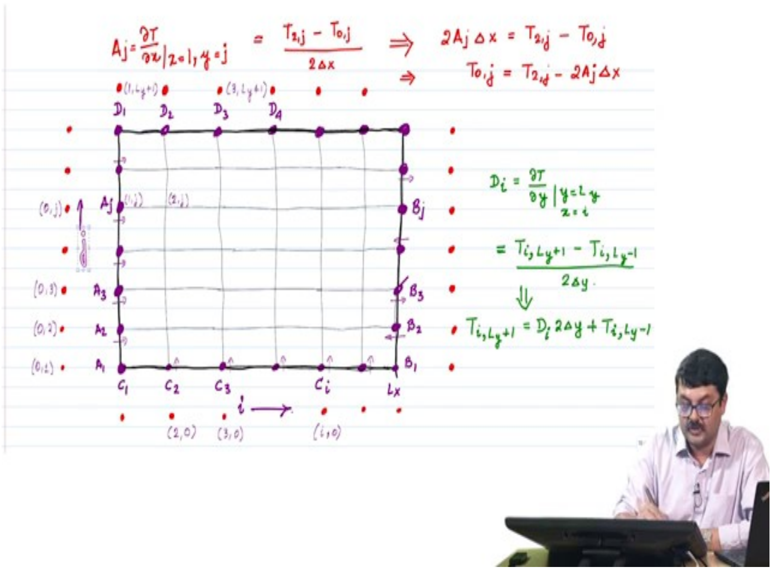
But in principle you can also have different amounts of heat coming in here and here and here. So, basically A which is the measure of the heat coming in the boundary condition right; d T x del T del x it can have different values as a function of y the j coordinate, it can we have different values along this right. Though in our test case we will consider that the values of A are independent of j, but you can have different amount of heat as you go here, you can have different amounts of heat leaking out.

As you go from here to here you can have different amounts of heat leaking out here to here; basically that would correspond to different surface properties, you can treat the surface differently. So, that is what we are specifying and the more important thing is we have to find out the values of the temperature for these points as well.

Now when you write down such an expression, when i equal to 1, right, so if this is your i equal to 1, at this point for a particular value of j suppose for this point, at this point; then you also have i minus 1. So, if i is 1; i minus 1 is 0; but you do not have those points on the lattice right. You have to solve, for to solve you need the values of this point which is outside the lattice, but you do not know what is T i minus 1 j.

Similarly, if you were writing for j equal to 1 say suppose this point or suppose this point right when j equal to L y. So, when j equal to L y you would have j plus 1, which is outside the lattice, right. If j is L y and j plus 1 would be outside the lattice and then how do, the question is how do we solve for this, right. And the solution is. So, you have to write down the expressions. So, to solve for T i j you need to express, you need the values such as you need the values of this and you need the values of this and so on so forth; specially if they are the boundaries and the way you solve it, is discussed in the next slide.

(Refer Slide Time: 13:41)



$$A_j = \frac{\partial T}{\partial x}\Big|_{x=1,\, y=j} = \frac{T_{2,j} - T_{0,j}}{2\Delta x} \Rightarrow 2A_j \Delta x = T_{2,j} - T_{0,j}$$

$$\Rightarrow T_{0,j} = T_{2,j} - 2A_j \Delta x$$

$$D_i = \frac{\partial T}{\partial y}\Big|_{\substack{y=L_y \\ x=i}} = \frac{T_{i,L_y+1} - T_{i,L_y-1}}{2\Delta y}$$

$$\Downarrow$$

$$T_{i,L_y+1} = D_i\, 2\Delta y + T_{i,L_y-1}$$

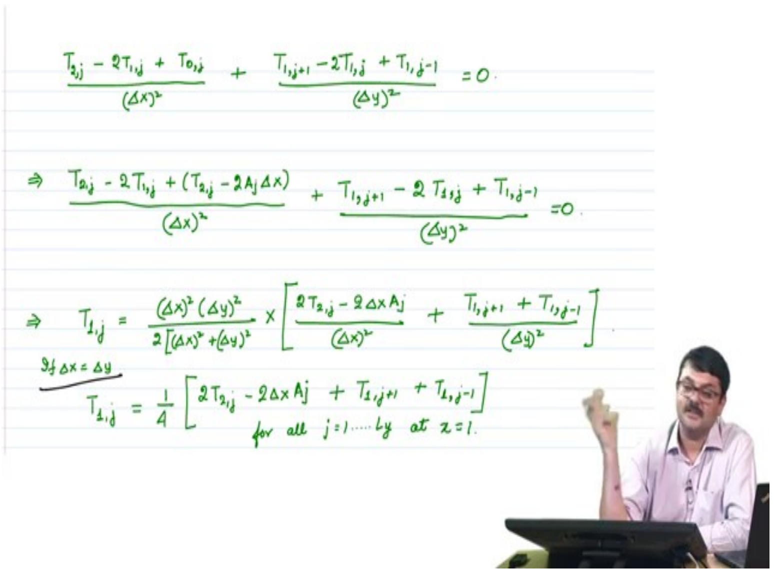
Here we ask you to note that A j; A j is that heat current, right. So, A j is this heat current, that is what you are specifying the del T del x is what you are specifying along the boundaries instead of temperature. So, this A j equal to del T del x at x equal to 1 and y equal to j can be written as T 2 j minus T 0 j by 2 del x. T 0 j is essentially these points outside the lattice which do not exist.

And the question is how do you figure out this value? And the way you figure out this value is by noting that; if A j which is specified then you can write this equality A j equal to this and then what you have is you solve for T 0 j, you basically you solve. So, you have this equation, get delta x on the other side of the equation that is what I have done here and write T 0 j in terms of A j, for each value of j right.

And similarly suppose you have specified the heat current along, suppose y equal to l y right. So, suppose you have for different values of i, you have specified the heat current at basically y equal to L y; that in turn can be written just like this in a similar expression, where D i is nothing but this, right. So, basically this, L y plus 1 here right, L y plus 1 and L y plus 1 is essentially these points which again those exist, there is a ghost points, right.

So, basically what you do is write this quantity, by a similar algebraic thing in terms of D i right. So, you basically write the temperature here in terms of D I, by this algebraic formula. So, you can do similar things for all the ghost points which are here, which will be B j B 1 B 2 B 3 B j. And you will you can similarly write the temperatures for these ghost points which do not exist; but we will be there in the expression for the Laplacian in terms of C 1 and then C 2 and C 3 and in general C i, right.

(Refer Slide Time: 16:59)



$$\frac{T_{2,j} - 2T_{1,j} + T_{0,j}}{(\Delta x)^2} + \frac{T_{1,j+1} - 2T_{1,j} + T_{1,j-1}}{(\Delta y)^2} = 0.$$

$$\Rightarrow \frac{T_{2,j} - 2T_{1,j} + (T_{2,j} - 2A_j \Delta x)}{(\Delta x)^2} + \frac{T_{1,j+1} - 2T_{1,j} + T_{1,j-1}}{(\Delta y)^2} = 0.$$

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

If $\Delta x = \Delta y$

$$T_{1,j} = \frac{1}{4}\left[2T_{2,j} - 2\Delta x A_j + T_{1,j+1} + T_{1,j-1}\right]$$

for all $j = 1 \ldots L_y$ at $x = 1$.

So, finally, let us focus at the Laplacian at this point and on this boundary. The Laplacian which you have the expression is this, right; you have this expression and you have T 0 j here, if you wanted to solve for T 1 j. If you wanted to solve for T 1 j you have T 0 j sitting here, right. And this part is standard, you do not have to worry it is basically the temperatures on the neighboring points along the y direction. But this T 0 j is now substituted by this expression, which we figured out here.

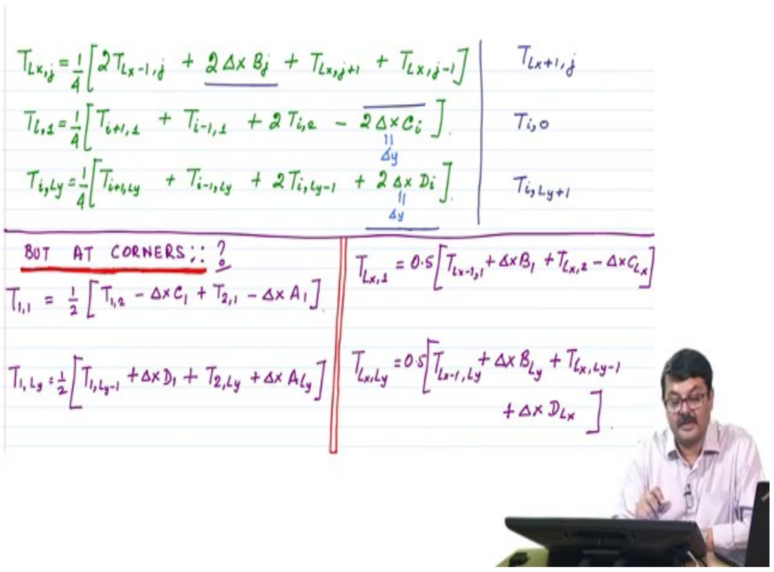
So, I have just substituted that expression here in lieu of this. And now what you have is all the quantities whose values you know; I mean, at least you know at a particular iteration and of course, iteratively the values of each of these points shall evolve and when it is value does not change while maintaining the boundary conditions. Now in terms of d T d x or rather in terms of A j then they basically become stationary; means, they do not evolve within your tolerance factor, you say that you have got your answer.

But at least you at your 0th iteration and your 1th and second and whatever nth iteration for each case, you have the values of all these quantities, right. So, now, you solve for basically T 1 j along the boundaries, T 1 j is the temperature at this point and this point for their different values of j, for different values of j along the y axis, right. So, j is varying for each value of j you find out the temperature at the boundary and the formula by just doing some algebraic manipulation of this you basically get this, right.

So, what I have taken is, done is basically multiplied by del x square whole square and just separation of variables. And basically now T 1 j depends upon all the neighboring points; the neighboring point and the boundary condition is sitting here nicely, right, from here it is sitting nicely here. So, you have applied the boundary condition, right. And if delta x equal to delta y just as previously, the expression reduces to even a simpler expression like this for all j.

And of course, A j has to be specified that is what you are specifying d T d x is what you are specifying for at the boundaries for each value of j and then all that you need is solved such a simple equation.

(Refer Slide Time: 20:07)



$$T_{Lx,j} = \frac{1}{4}\left[2T_{Lx-1,j} + \underline{2\Delta x B_j} + T_{Lx,j+1} + T_{Lx,j-1}\right] \qquad T_{Lx+1,j}$$

$$T_{i,1} = \frac{1}{4}\left[T_{i+1,1} + T_{i-1,1} + 2T_{i,2} - \overline{2\underset{\Delta y}{\overset{\shortparallel}{\Delta x}} C_i}\right]. \qquad T_{i,0}$$

$$T_{i,Ly} = \frac{1}{4}\left[T_{i+1,Ly} + T_{i-1,Ly} + 2T_{i,Ly-1} + \underline{2\underset{\Delta y}{\overset{\shortparallel}{\Delta x}} D_i}\right]. \qquad T_{i,Ly+1}$$

BUT AT CORNERS :: ?

$$T_{1,1} = \frac{1}{2}\left[T_{1,2} - \Delta x C_1 + T_{2,1} - \Delta x A_1\right].$$

$$T_{1,Ly} = \frac{1}{2}\left[T_{1,Ly-1} + \Delta x D_1 + T_{2,Ly} + \Delta x A_{Ly}\right]$$

$$T_{Lx,1} = 0.5\left[T_{Lx-1,1} + \Delta x B_1 + T_{Lx,2} - \Delta x C_{Lx}\right]$$

$$T_{Lx,Ly} = 0.5\left[T_{Lx-1,Ly} + \Delta x B_{Ly} + T_{Lx,Ly-1} + \Delta x D_{Lx}\right].$$

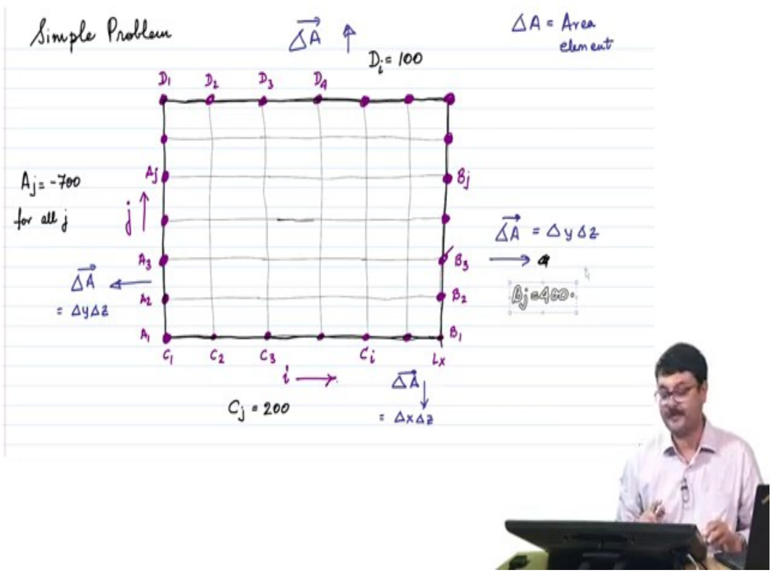
Now, for the other boundaries, for the boundaries along this and this and this; you can similarly write down expressions which are written here for x equal to L x for different values of j, which is essentially this boundary x equal to L x, right. And different values of j, that can be written in terms of doing us very similar procedure as was discussed in the last slide; the expression can be this and note that del T del x at x equal to L x which is B j it is sitting nicely here, right. So, it is basically, so you have use similar expressions, this is sitting there.

Similarly along the other boundaries, what you have specified is D 1 D 2 D 3 along this boundary and C 1 C 2 C 3 along this boundary, as you change x right. And at each point you could have different amounts of heat coming in at different points of i; you have this expression where you see that C i is sitting here and note delta x equal to delta y, the value of delta x is also sitting here basically, right. So, you get these expressions.

And similarly for the other boundary you have this expression, where you have D i and note delta x equal to del y and that is why we have this 1 by 4 factor and these are all terms which you have access to. However, there is still a complication and what is that complication; note at these four corner points, these four corner points heat is coming in, our boundary conditions as specified del T del x, A 1 is also specified del T del y C 1 is also specified.

Similarly for here right C L x is specified as well as B 1 is specified. So, you have basically heating going out from this direction as well as heat going out from this direction and so on so forth. For that you have to work a bit more in terms of working out the algebra, but what I have done here is written down the expressions of how to calculate for T 1 1, T 1 L y, T L x 1 those are the four corners and T L x L x. So, these are of course, for all the cases when delta x equal to delta y it is a bit more of algebra; but these are the expressions which is basically a few steps ahead from doing this right.

(Refer Slide Time: 22:59)


Simple Problem
$\vec{\Delta A}$
$D_i = 100$
ΔA = Area element
D_1 D_2 D_3 D_4
$A_j = -700$ for all j
A_j
j
B_j
$\vec{\Delta A} = \Delta y \Delta z$
A_3 A_2 A_1
B_3 B_2 B_1
$\vec{\Delta A} = \Delta y \Delta z$
$B_j = 400$
C_1 C_2 C_3 C_i L_x
i
$\vec{\Delta A} = \Delta x \Delta z$
$C_j = 200$

So, what do you have? So, just to figure out where we stand, you had to solve the Laplacian at the boundary where boundary conditions were given, where the heat current del t del x was given; that you have used such an expression to get rid of the expressions in the Laplacian which was depending upon these Ghosh positions those you have replaced in terms of A j and D j and C j and B j.

So, basically you have these now in the expression for the update at the boundaries right. And what we shall consider is look at a simple problem where the A j s across for all values of y is minus 700, for all values of along this boundary is 100 and along this boundary is 200. So, basically heat is coming in here and basically the B j is 400. So, heat is going out from here, heat is going out from here; but the amount of heat going out from here and from this boundary C j is equal to 200 which is more than 100.

So, more heat is going out from here compared to this boundary right; heat is entering from this direction, heat is going off from this maximum 400. Then this boundary has more heat going out, this boundary has relatively less heat going out.

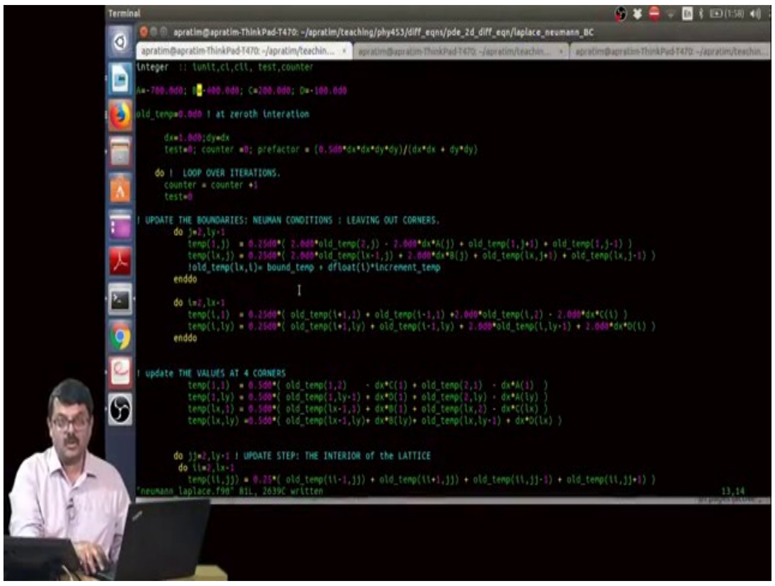
Now in steady state right, once the temperature has finalized it does not change as a function of time, then basically what is happening is there is no heat, net heat inflow into the system right, that is what you have specified this for. And if you remember your Gauss's law, if you take A j dot area; now area here will be in this direction will be in this direction, the area element is normal to the surface is in this direction, the area are normal to the surface is in this direction going pointing out and the area here normal to the surface is in this direction, right.

And if you, so A j is of course, q vector, if you take q A, so basically what I am calculating is amount of heat passing through this each of these four surfaces. And if you take the correct dot product, you will see that the net amount of heat A j into I mean you have to take the sign of delta A as well, if we do it right you will see that there is no net heat input of the system. The amount of heat coming in is equal to the amount of heat going out and only under such a condition would you have a steady state.

If there was a net heat input into the system then the temperature of the system would rise right. But what you have is, there is no del T, del time the time derivative of temperature has been set to 0; that means, you are discussing steady state conditions and temperature is not varying as a function of time. In such a case the net heat input into the system has to be 0, net flux of heat at on each boundaries you have a flux.

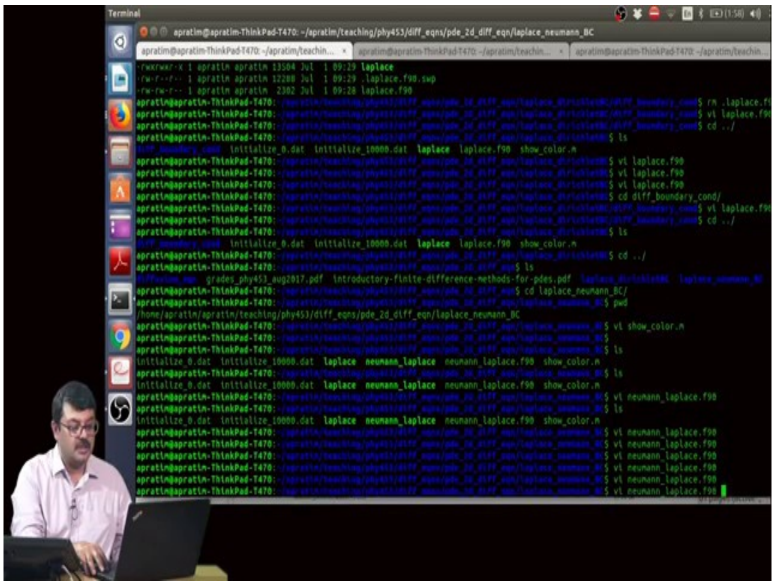
But the no net, the system does not gain any net heat; if it gained it the temperature would raise and you would see that the boundary conditions have been chosen or the appropriate boundary conditions would be such that q dot A across the four surfaces would be 0 and you can check that for yourself, right. With this background, now let us move and show you the code and figure out how would the temperature profiles look like.

(Refer Slide Time: 27:31)



```
integer  :: iunit,ci,cii, test,counter

A=-700.0d0; B=-400.0d0; C=200.0d0; D=-100.0d0

old_temp=0.0d0 ! at zeroth interation

      dx=1.0d0;dy=dx
      test=0; counter =0; prefactor = (0.5d0*dx*dx*dy*dy)/(dx*dx + dy*dy)

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

! UPDATE THE BOUNDARIES: NEUMAN CONDITIONS : LEAVING OUT CORNERS.
        do j=2,ly-1
           temp(1,j)  = 0.25d0*( 2.0d0*old_temp(2,j) - 2.0d0*dx*A(j) + old_temp(1,j+1) + old_temp(1,j-1) )
           temp(lx,j) = 0.25d0*( 2.0d0*old_temp(lx-1,j) + 2.0d0*dx*B(j) + old_temp(lx,j+1) + old_temp(lx,j-1) )
           !old_temp(lx,i)= bound_temp + dfloat(i)*increment_temp
        enddo

        do i=2,lx-1
           temp(i,1)  = 0.25d0*( old_temp(i+1,1) + old_temp(i-1,1) +2.0d0*old_temp(i,2) - 2.0d0*dx*C(i) )
           temp(i,ly) = 0.25d0*( old_temp(i+1,ly) + old_temp(i-1,ly) + 2.0d0*old_temp(i,ly-1) + 2.0d0*dx*D(i) )
        enddo


! update THE VALUES AT 4 CORNERS
           temp(1,1)  = 0.5d0*( old_temp(1,2)    - dx*C(1) + old_temp(2,1)  - dx*A(1)  )
           temp(1,ly) = 0.5d0*( old_temp(1,ly-1) + dx*D(1) + old_temp(2,ly) - dx*A(ly) )
           temp(lx,1) = 0.5d0*( old_temp(lx-1,1) + dx*B(1) + old_temp(lx,2) - dx*C(lx) )
           temp(lx,ly) =0.5d0*( old_temp(lx-1,ly)+ dx*B(ly)+ old_temp(lx,ly-1) + dx*D(lx) )


        do jj=2,ly-1 ! UPDATE STEP: THE INTERIOR of the LATTICE
         do ii=2,lx-1
           temp(ii,jj) = 0.25*( old_temp(ii-1,jj) + old_temp(ii+1,jj) + old_temp(ii,jj-1) + old_temp(ii,jj+1) )
"neumann_laplace.f90" 81L, 2639C written                                                        13,14
```

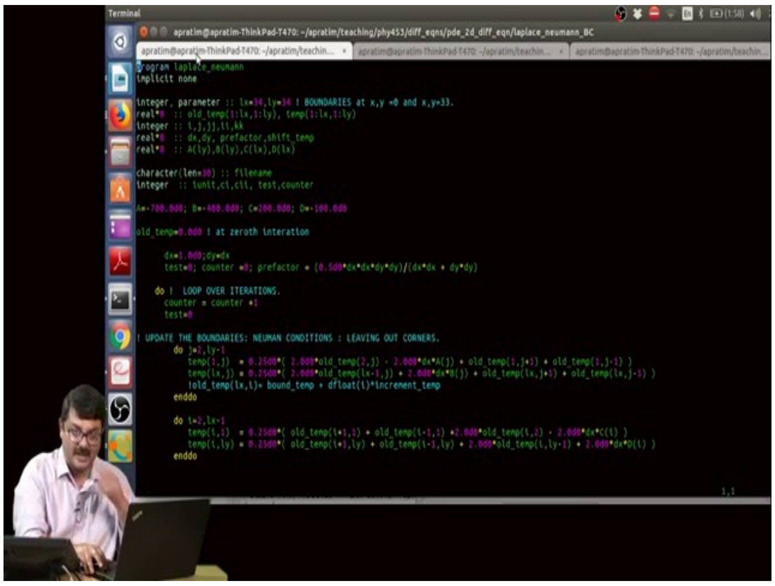
(Refer Slide Time: 27:33)



```
-rwxrwxr-x 1 apratim apratim 13504 Jul  1 09:29 laplace
-rw-r--r-- 1 apratim apratim 12288 Jul  1 09:29 .laplace.f90.swp
-rw-rw-r-- 1 apratim apratim  2302 Jul  1 09:28 laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC/diff_boundary_cond$ rm .laplace.f9
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC/diff_boundary_cond$ vi laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC/diff_boundary_cond$ cd ../
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ ls
diff_boundary_cond  initialize_0.dat  initialize_10000.dat  laplace  laplace.f90  show_color.m
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ vi laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ vi laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ vi laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ cd diff_boundary_cond/
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC/diff_boundary_cond$ vi laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC/diff_boundary_cond$ cd ../
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ ls
diff_boundary_cond  initialize_0.dat  initialize_10000.dat  laplace  laplace.f90  show_color.m
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_dirichletBC$ cd ../
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn$ ls
diffusion_eqn  grades_phy453_aug2017.pdf  introductory-finite-difference-methods-for-pdes.pdf  laplace_dirichletBC  laplace_neumann_BC
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn$ cd laplace_neumann_BC/
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ pwd
/home/apratim/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi show_color.m
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ls
initialize_0.dat  initialize_10000.dat  laplace  neumann_laplace  neumann_laplace.f90  show_color.m
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ls
initialize_0.dat  initialize_10000.dat  laplace  neumann_laplace  neumann_laplace.f90  show_color.m
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ls
initialize_0.dat  initialize_10000.dat  laplace  neumann_laplace  neumann_laplace.f90  show_color.m
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ vi neumann_laplace.f90
```

So, the code is called Neumann dot Laplace dot f 90, right.

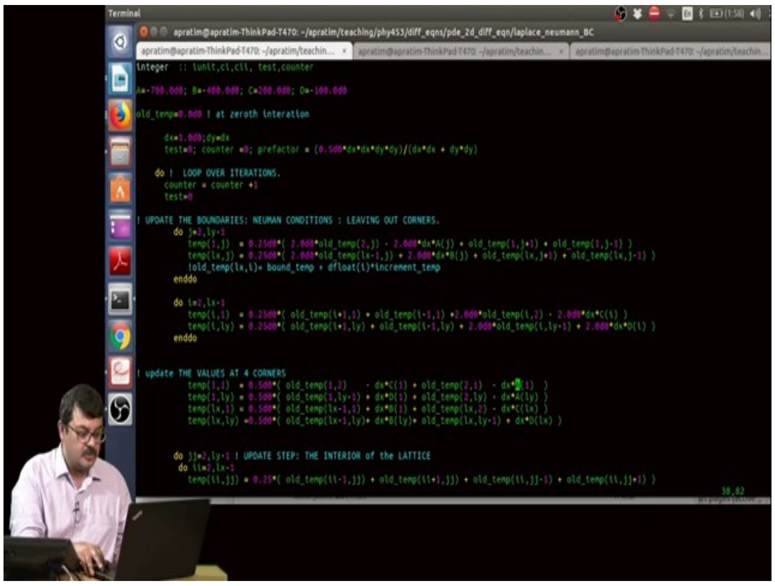
(Refer Slide Time: 27:37)



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

program laplace_neumann
implicit none

integer, parameter :: lx=34,ly=34 ! BOUNDARIES at x,y =0 and x,y=33.
real*8  :: old_temp(1:lx,1:ly), temp(1:lx,1:ly)
integer :: i,j,jj,ii,kk
real*8  :: dx,dy, prefactor,shift_temp
real*8  :: A(ly),B(ly),C(lx),D(lx)

character(len=30) :: filename
integer  :: iunit,ci,cii, test,counter

A=-700.0d0; B=-400.0d0; C=200.0d0; D=-100.0d0

old_temp=0.0d0 ! at zeroth interation

     dx=1.0d0;dy=dx
     test=0; counter =0; prefactor = (0.5d0*dx*dx*dy*dy)/(dx*dx + dy*dy)

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

! UPDATE THE BOUNDARIES: NEUMAN CONDITIONS : LEAVING OUT CORNERS.
       do j=2,ly-1
          temp(1,j)  = 0.25d0*( 2.0d0*old_temp(2,j) - 2.0d0*dx*A(j) + old_temp(1,j+1) + old_temp(1,j-1) )
          temp(lx,j) = 0.25d0*( 2.0d0*old_temp(lx-1,j) + 2.0d0*dx*B(j) + old_temp(lx,j+1) + old_temp(lx,j-1) )
          !old_temp(lx,i)= bound_temp + dfloat(i)*increment_temp
       enddo

       do i=2,lx-1
          temp(i,1)  = 0.25d0*( old_temp(i+1,1) + old_temp(i-1,1) +2.0d0*old_temp(i,2) - 2.0d0*dx*C(i) )
          temp(i,ly) = 0.25d0*( old_temp(i+1,ly) + old_temp(i-1,ly) + 2.0d0*old_temp(i,ly-1) + 2.0d0*dx*D(i) )
       enddo

                                                                                          1,1
```

And this is the program Laplace Neumann and here I specify the lattice sizes L x equal to 34 just as the previous case and L y equal to 34 just as the previous case. Now note I have four arrays called A l y, A l y, C l x and D l x. So, these are those A B C D which I specify at the boundaries right, those are the amount of heat current and that you specify at the boundaries and the rest of it is basically very similar to the previous code.

(Refer Slide Time: 28:19)



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

integer  :: iunit,ci,cii, test,counter

A=-700.0d0; B=-400.0d0; C=200.0d0; D=-100.0d0

old_temp=0.0d0 ! at zeroth interation

     dx=1.0d0;dy=dx
     test=0; counter =0; prefactor = (0.5d0*dx*dx*dy*dy)/(dx*dx + dy*dy)

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

! UPDATE THE BOUNDARIES: NEUMAN CONDITIONS : LEAVING OUT CORNERS.
       do j=2,ly-1
          temp(1,j)  = 0.25d0*( 2.0d0*old_temp(2,j) - 2.0d0*dx*A(j) + old_temp(1,j+1) + old_temp(1,j-1) )
          temp(lx,j) = 0.25d0*( 2.0d0*old_temp(lx-1,j) + 2.0d0*dx*B(j) + old_temp(lx,j+1) + old_temp(lx,j-1) )
          !old_temp(lx,i)= bound_temp + dfloat(i)*increment_temp
       enddo

       do i=2,lx-1
          temp(i,1)  = 0.25d0*( old_temp(i+1,1) + old_temp(i-1,1) +2.0d0*old_temp(i,2) - 2.0d0*dx*C(i) )
          temp(i,ly) = 0.25d0*( old_temp(i+1,ly) + old_temp(i-1,ly) + 2.0d0*old_temp(i,ly-1) + 2.0d0*dx*D(i) )
       enddo


! update THE VALUES AT 4 CORNERS
          temp(1,1)  = 0.5d0*( old_temp(1,2)    - dx*C(1) + old_temp(2,1)  - dx*A(1)  )
          temp(1,ly) = 0.5d0*( old_temp(1,ly-1) + dx*D(1) + old_temp(2,ly) - dx*A(ly) )
          temp(lx,1) = 0.5d0*( old_temp(lx-1,1) + dx*B(1) + old_temp(lx,2) - dx*C(lx) )
          temp(lx,ly) =0.5d0*( old_temp(lx-1,ly)+ dx*B(ly)+ old_temp(lx,ly-1) + dx*D(lx) )


       do jj=2,ly-1 ! UPDATE STEP: THE INTERIOR of the LATTICE
        do ii=2,lx-1
          temp(ii,jj) = 0.25*( old_temp(ii-1,jj) + old_temp(ii+1,jj) + old_temp(ii,jj-1) + old_temp(ii,jj+1) )
                                                                                          38,82
```

Except in my previous case I was specifying the temperature at the boundaries and I was updating the temperature at only the inside points, not at the boundaries here we are updating the temperature at the boundaries as well. And so you see that there differences, where I am not specifying the temperature and the boundaries which I did in the previous case, here the code starts directly.

So, old temperature has been set to 0 just as an initial condition. So, the entire lattice has initial temperature or initial does not mean with respect to time initial here with rest means at the 0th iteration has been set to 0, right. And d x is 1 and d y has been set to d x that is when those formulas are valid. And you would see that, I am basically updating the values of the temperature at the boundaries here.

So, I am updating the boundaries first, because the formula is different; because you will have A j s and B j s sitting here, I am just implemented whatever I wrote in the code here. So, old temperature is the older value of whatever temperature is there, right; which has all been set to 0. But what is nonzero, in the first iteration are these values, right. So, here I have set the values of A, A, so this is Fortran's.

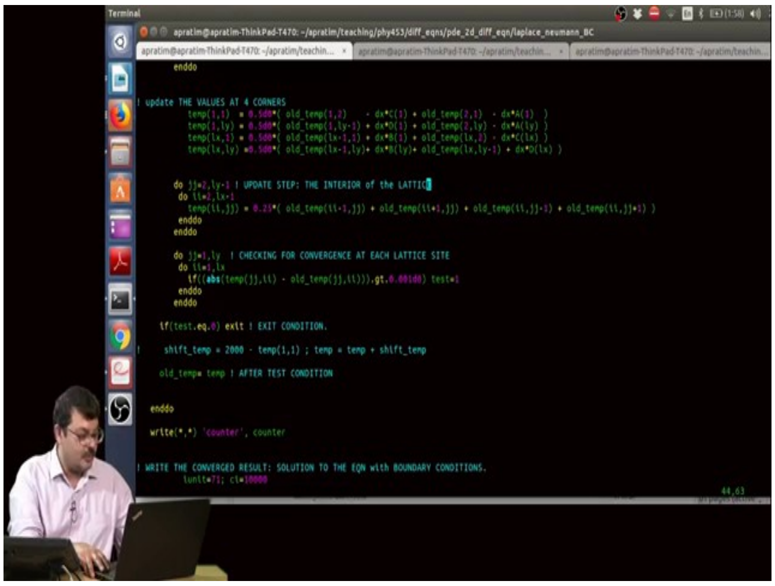
So, basically though I am writing A all the elements in the area being set to minus 700, all the elements in array B are being set to minus 400 and so on so forth. So, these are nonzero values. So, these are the values which are nonzero, this and this at the 0 th iteration. So, all of old temperature has been set to 0 here.

But after the first iteration, because you have nonzero values here, you have nonzero values in temp, right. So, this is you have basically though tempered have been 0 initially; it now has nonzero values. And similarly here for the boundary points you see that, this is basically x remains 1 and the loop is only from the boundary points.

But note I have left out the corner points, the corner point the formula is different; hence they are being updated here, right, where you both in the expression for the temperature updated the corners you both have C 1 and A 1 and at a different point you have D and you have A right and you have B and you have C. So, the corners are being updated separately and they are basically 1, 1; 1, l y; l x 1 and l x l y.

But for all other points on the boundary you have A sitting here, B sitting here; in the other cases you have C sitting here those are the same implementation of the same formulas which I showed and you have D sitting here. Once you have updated the boundaries and the corners, what you do is basically update the interior of the lattice.

(Refer Slide Time: 32:05)



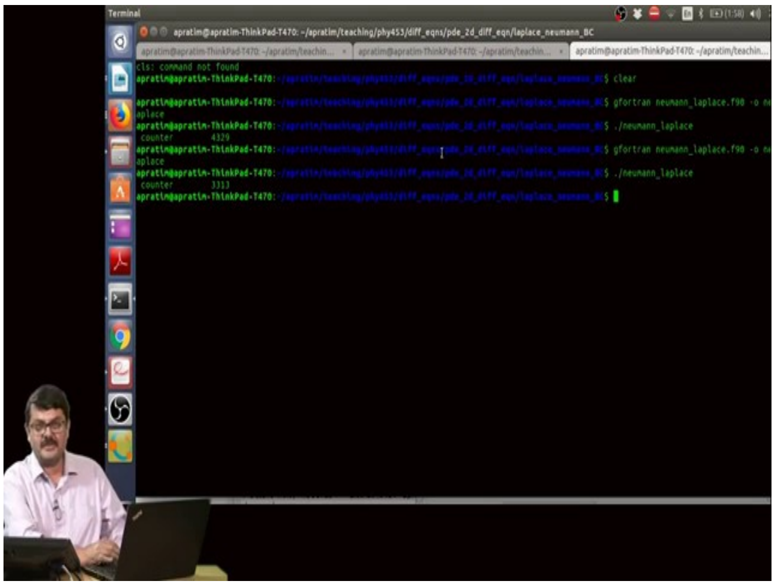
```
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC
        enddo

! update THE VALUES AT 4 CORNERS
          temp(1,1)  = 0.5d0*( old_temp(1,2)    - dx*C(1) + old_temp(2,1)  - dx*A(1)  )
          temp(1,ly) = 0.5d0*( old_temp(1,ly-1) + dx*D(1) + old_temp(2,ly) - dx*A(ly) )
          temp(lx,1) = 0.5d0*( old_temp(lx-1,1) + dx*B(1) + old_temp(lx,2) - dx*C(lx) )
          temp(lx,ly) =0.5d0*( old_temp(lx-1,ly)+ dx*B(ly)+ old_temp(lx,ly-1) + dx*D(lx) )


        do jj=2,ly-1 ! UPDATE STEP: THE INTERIOR of the LATTICE
         do ii=2,lx-1
          temp(ii,jj) = 0.25*( old_temp(ii-1,jj) + old_temp(ii+1,jj) + old_temp(ii,jj-1) + old_temp(ii,jj+1) )
         enddo
        enddo

        do jj=1,ly  ! CHECKING FOR CONVERGENCE AT EACH LATTICE SITE
         do ii=1,lx
          if((abs(temp(jj,ii) - old_temp(jj,ii))).gt.0.001d0) test=1
         enddo
        enddo

      if(test.eq.0) exit ! EXIT CONDITION.

!      shift_temp = 2000 - temp(1,1) ; temp = temp + shift_temp

      old_temp= temp ! AFTER TEST CONDITION


    enddo

    write(*,*) 'counter', counter

! WRITE THE CONVERGED RESULT: SOLUTION TO THE EQN with BOUNDARY CONDITIONS.
          lunit=71; ci=10000
                                                                        44,63
```

And the interior of the lattice is very simple. So, you go to each lattice point in the interior from 2 equal to l y minus 1 and 2 equal to l x minus 1. Write down the expression which you get by solving the finite differences method, which is also discussed last time, you update all the points, the new temperature, right; then you check for convergence at each lattice point, right. So, you go to each lattice point and check that the new value of temperature, what is the difference from the old value of temperature to the new value of temperature, right

And if it is above a certain tolerance level, so 0.01 or even if you put 0.001 right; if the difference is more at any point is more than 0.001 then basically you set dummy variable test equal to 1. And only if test equal to 0 do you exit this number of iterations loop, right; test has been is being set to 0 every time. This is the loop over iterations; test is being set to 0.

Here you are checking, if any of the differences in temperature is above your tolerance level test is set to 1; then it cannot exit the loop and then you basically old temp is set to temp and you want doing this, right. Now, what does one do, well one compiles new Laplace dot f 90 minus o to new Laplace.

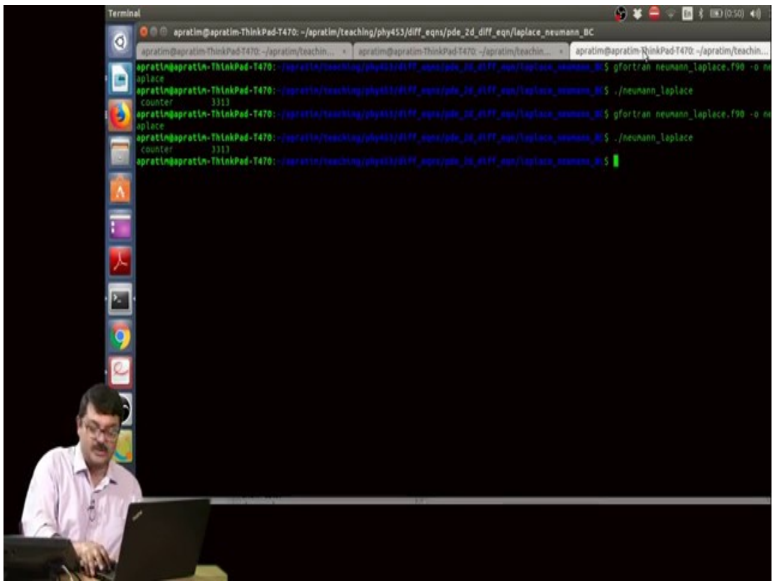
(Refer Slide Time: 33:51)



```
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC
cls: command not found
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ clear

apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ gfortran neumann_laplace.f90 -o ne
aplace
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ./neumann_laplace
 counter        4329
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ gfortran neumann_laplace.f90 -o ne
aplace
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ./neumann_laplace
 counter        3313
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$
```

Then you run it and it you see that it takes 4329 iterations number of iterations for the results to converge the final temperature distribution.

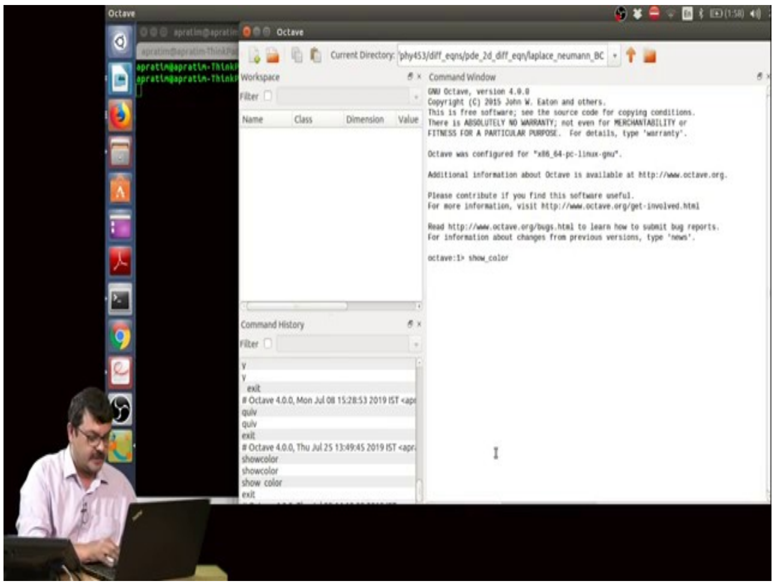
(Refer Slide Time: 34:07)



```
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ gfortran neumann_laplace.f90 -o ne
aplace
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ./neumann_laplace
 counter        3313
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ gfortran neumann_laplace.f90 -o ne
aplace
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$ ./neumann_laplace
 counter        3313
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC$
```

How do we visualize it, well you have again the show color just as previously or the final data is being stored in 10000.

(Refer Slide Time: 34:16)

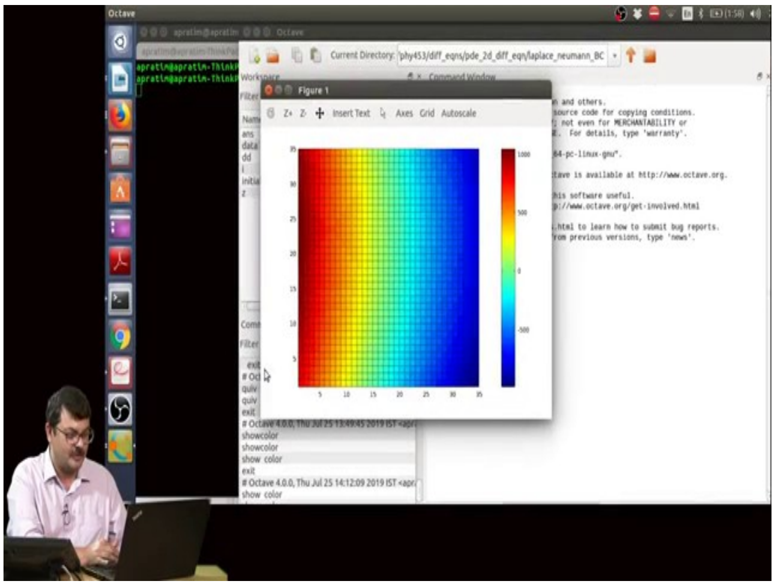


```
GNU Octave, version 4.0.0
Copyright (C) 2015 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> show_color
```

```
y
y
  exit
# Octave 4.0.0, Mon Jul 08 15:28:53 2019 IST <apr
quiv
quiv
exit
# Octave 4.0.0, Thu Jul 25 13:49:45 2019 IST <apr
showcolor
showcolor
show  color
exit
```

So, initialized are 10000.

(Refer Slide Time: 34:23)



```
# Octave 4.0.0, Thu Jul 25 14:12:09 2019 IST <apr
show  color
```

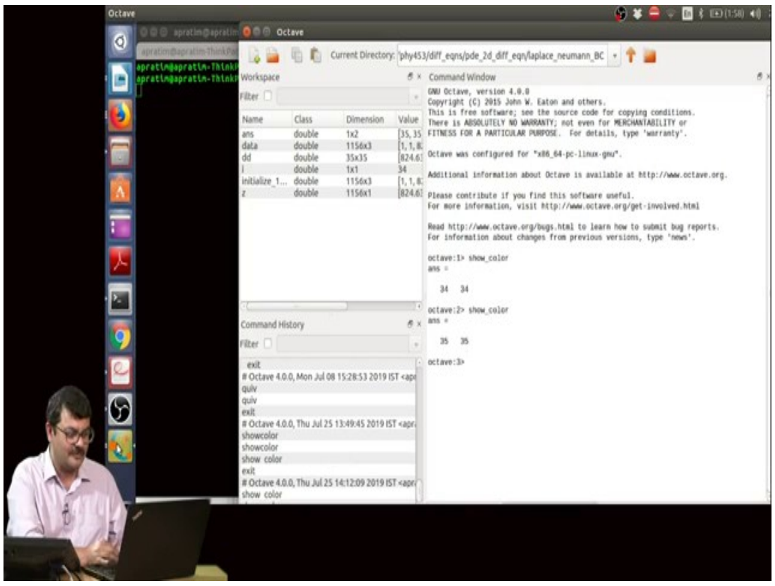
And you can open octave, which I have already opened right and show color. So, this is the temperature distribution, this is a long x, this is along y and that is something which is very fishy and is what is fishy is that the blue color is minus 5000. How can temperature be minus 5000 right; I mean suppose you are discussing it in Kelvin, you cannot have. So, there is something which is fishy, which is not sensible, what is that we will discuss it.

But mean, while if you remember heat was coming in from this direction and you have a very high red color here, showing high temperatures; heat was going out minus 400 the d t d x, the heat going out was minus 400 here. Correspondingly you have relatively low temperatures here. There was more heat going out from this direction, in along this direction of y compared to this.

So, you see that you have a higher value of temperature even here compared to a, not higher value; but higher distribution I mean it is hotter over a larger amount of space here; compared to here because more heat is going out in this direction and this direction compared to this direction. So, the thing seems to make sense, but this temperature range is completely of the mark, it does not make any sense; maybe we should change the temperature a bit, the amount of heat going in or coming in a bit.

So, suppose you decrease the amount of heat going here, heat current and let us see what happens, you will see that the problem will remain. Now you have got to the final value of the temperature distribution in a fewer number of iterations.

(Refer Slide Time: 36:52)


Current Directory: phy453/diff_eqns/pde_2d_diff_eqn/laplace_neumann_BC

Name	Class	Dimension	Value
ans	double	1x2	[35, 35
data	double	1156x3	[1, 1, 8
dd	double	35x35	[824.6
i	double	1x1	34
initialize_1...	double	1156x3	[1, 1, 8
z	double	1156x1	[824.6

```
GNU Octave, version 4.0.0
Copyright (C) 2015 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> show_color
ans =

   34   34

octave:2> show_color
ans =

   35   35

octave:3>
```

```
exit
# Octave 4.0.0, Mon Jul 08 15:28:53 2019 IST <ap
quiv
quiv
exit
# Octave 4.0.0, Thu Jul 25 13:49:45 2019 IST <apr
showcolor
showcolor
show color
exit
# Octave 4.0.0, Thu Jul 25 14:12:09 2019 IST <apr
show color
```

But if you again plot the data, you have a similar distribution of temperature; but the range of temperature is still minus 100 though it has gone from minus 500 or minus 1000 to 1000, temperature cannot be negative. What is going wrong ok, it is very important to realize this. It is intentionally that I showed you this code.

So, what is happening is you have set the temperature arbitrarily to 0, right as the initial condition; that really does not make much sense. And I have also set it to 1000 right and if you are talking in Kelvin setting the temperature arbitrarily the initial condition if you like to 0 is pretty arbitrarily. And from that all that you have is amount the heat current, the amount of heat going in and the amount of heat going out. So, all that it cares about is the gradient in temperature, the changes in temperature about this certain arbitrary value of initial temperature scale that you have set right.

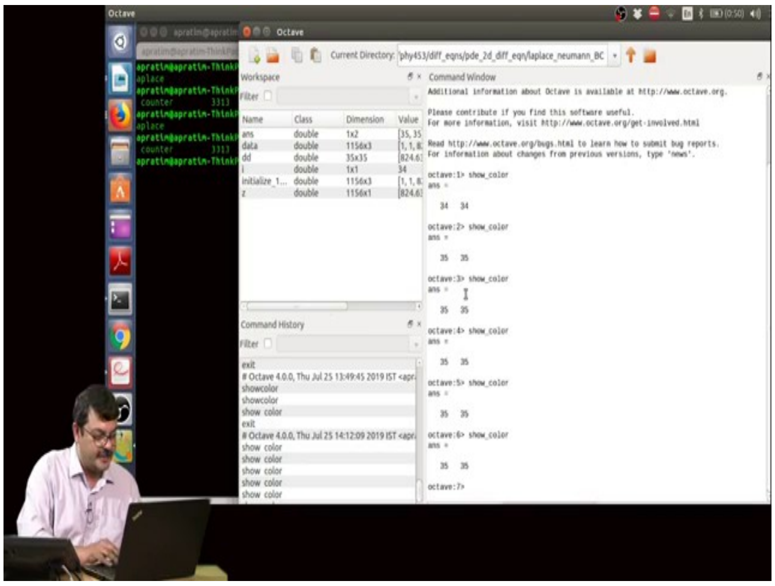
And if you set that correctly you would not have such problems. So, what you have to do is, say that somehow you set the temperature at one corner to be 2000 right; I mean, you set your temperature whatever in your right units, right now all the units all the scales the R 1 right. But you can, I mean not necessarily you will be working with a 34 cross 34 lattice you might be working with a plate of size 0 to 2. And then you need to then discretize the lattice, then you will have finite values of delta x and delta y and so on so forth.

So, putting in units is suppose at some point you say, that at one corner at corner suppose one and one. So, the left hand bottom corner, you set the temperature to be 2000, so that every time you in an iteration you change the temperature; but you want that temperature at this corner one and one to p remain 2000 right. And that is how you implement it, think about it, how you would implement it; I am intentionally not explaining this part.

Now, but suppose you wanted maintain the temperature at this point to be 2000 and accordingly shift the temperatures of all the other points, all that you have is the gradient right, news and the differences in temperature. And now you are saying that one corner shall maintain of temperature whatever 2000, you can choose 1000 does not really matter. And then with respect to this point where you are specifying both the temperature and the gradients is like Cauchy right.

What are the temperatures at the other points given the certain boundary conditions, where you are specifying the heat coming in and the heat going out right? Now if you do that, I compile it again, run it again and the data is being stored in the same file, you should put it if you want you can put it in a different file.

(Refer Slide Time: 40:30)



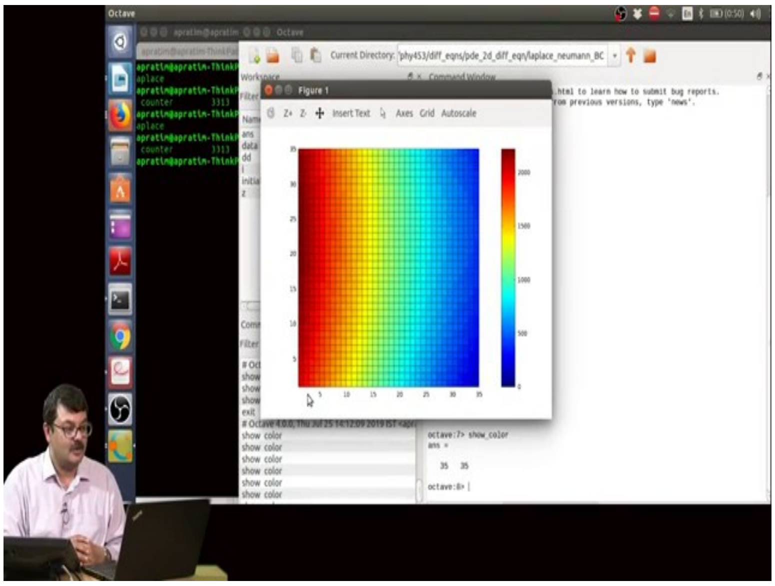
Name	Class	Dimension	Value
ans	double	1x2	[35, 35
data	double	1156x3	[1, 1, 8
dd	double	35x35	[824.61
i	double	1x1	34
initialize_1...	double	1156x3	[1, 1, 8
z	double	1156x1	[824.61

```
exit
# Octave 4.0.0, Thu Jul 25 13:49:45 2019 IST <apr
showcolor
showcolor
show color
exit
# Octave 4.0.0, Thu Jul 25 14:12:09 2019 IST <apr
show color
show color
show color
show color
show color
```

```
Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> show_color
ans =

   34   34

octave:2> show_color
ans =

   35   35

octave:3> show_color
ans =

   35   35

octave:4> show_color
ans =

   35   35

octave:5> show_color
ans =

   35   35

octave:6> show_color
ans =

   35   35

octave:7>
```

And if you plot the data this is what you get.

(Refer Slide Time: 40:34)



```
octave:7> show_color
ans =

   35   35

octave:8> |
```

And this, now makes more sense, right. So, under those conditions, if you are maintaining 1 x 1 1 to be temperature 2000 which is here; then the temperature distribution over the rest of the plate, so here it will be higher than 2000, because you are putting in a certain amount of heat, but less amount of heat is going out from this direction.

So, here you have higher values of temperature than 2000, here you have lower values of temperature, so it would be somewhere here; here your medium values of temperature and this is how the temperature distribution would look like given the boundary conditions. Now of course, you can have put in boundary conditions, so that the amount of heat current is also changing along here.

So, right now it is 100 here and 200 here and it come incoming heat is 700 and outgoing heat is 400 you can make them more complicated depending upon your real engineering problems or whatever problems; when you are doing a thermal conductivity problem you can have different boundary conditions, those have to be put in and you can find out the temperature profile, right.

So, this was essentially a discussion of the Dirichlet boundary condition which we discussed in last class; Neumann boundary condition which we discussed in today's class and we touched upon Cauchy boundary condition. But really you know; what if you are discussing diffusion equation you would also might want to know that, if you set up these boundary conditions at time T equal to 0, how would the temperature profile evolve as a function of time?

Del T temperature del tau, tau being the time. Now if you have to do that you can basically put in a suitable finite differences method for del T temperature versus del tau time; but you can do it, but one has to be a bit more careful, because you can use the first order finite difference formula and then one has to look carefully analyze the stability conditions, the criteria not always the answers are stable or alternatively you have to basically given initial conditions, the initial boundary conditions and then maintain the boundary conditions the rest of the time.

So, this is a bit more involved when you do it, you also have a Laplacian on there on the right side of the equation. And you have also temperature as a function of time, but that would actually need quite some bit of learning stability conditions, what are the different expressions, what order scheme should we use for writing del T del tau and that if and when you need it, we hope that with this at least introduction you can read up the book and learn about it.

You might have realized that in this course we are touching few of the topics Monte Carlo, Ising model and random numbers, random number integration we are doing this in a hands on manner. But if we had to basically do larger number of topics, we would not be have, we would not have so much time to discuss these things in a hands on manner. So, the hope is, we are exploring, we are basically giving you an introduction to various kinds of computational techniques and algorithm building.

And in the next class you will be doing will be starting molecular dynamics. And with this various exposure to different kinds of thoughts and algorithms and implementing different kinds of algorithms in the computer; whenever you need in the future right, which will of course, very likely to be beyond the scope of this course, techniques. You will be able to read up the book and learn from what the algorithm building and implementation and the background physics from the book right.

So, this class is basically the end to the this module and from the next class you will have a different module. And I shall teach you molecular dynamics after a few more lectures, which will be given by Prasenjit Ghosh.

Thanks.