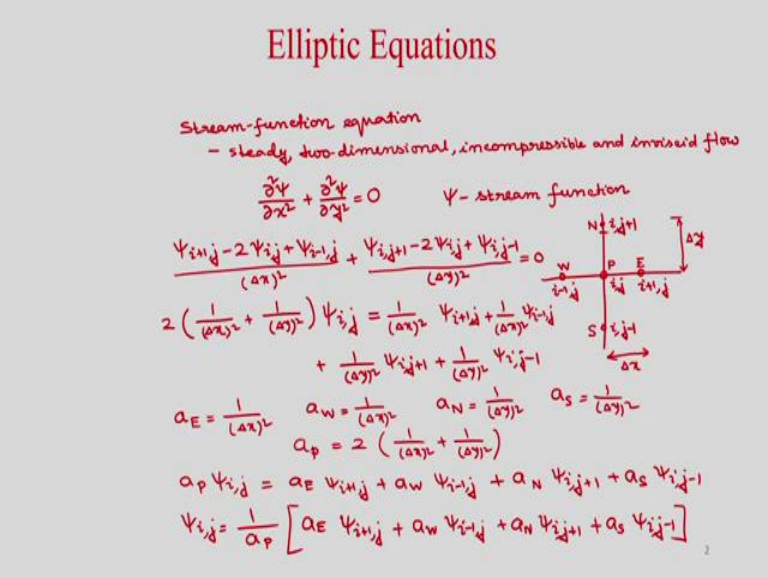
Computational Fluid Dynamics for Incompressible Flows
Prof. Amaresh Dalal
Department of Mechanical Engineering, IIT
Indian Institute of Technology, Guwahati
Lecture 3
Applications

Hello everyone, so, in last lecture, we have learned different iterative methods, Point Gauss Seidel method, Line Gauss Seidel method and also with Relaxation Factor we have learned Successive, Point Successive Gauss Seidel method and Line Successive Gauss Seidel method and also Alternating Direction Implicit method.

Now, today we will solve some application problems and here also the C program for solving this algebraic equation. So, let us consider the same equation, the Laplace equation, but let us write for a stream function.

(Refer Slide Time: 01:22)



So, we are writing the stream function equation for steady, two dimensional, incompressible and inviscid flow, steady, two dimensional, incompressible and inviscid flow. So, you will get Laplace equation where we can write del 2 psi by del x square plus del 2 psi by del y square is equal to 0 where size is the stream function.

So, with constant step size delta x and delta y, if you discretize this equation with central difference method, then we will get the algebraic equation like. So, you can see, so these

are ij, this is your i plus 1j, this is your i minus 1j and this is your ij plus 1 and this is your ij minus 1 and delta x and delta y are constant, this is your delta x and this is your delta y.

So, if you discretize this equation obviously you are going to get psi i plus 1j minus twice psi ij plus i i minus 1j divided by delta x square plus psi ij plus 1 minus 2 psi ij and plus psi ij minus 1 divided by del y square is equal to 0. So, now if you rewrite this equation in this way. Say we are writing this psi ij we are taking in the right hand side, so we will get 2 into 1 by delta x square plus 1 by delta y square.

So, these are the coefficient, diagonal coefficient for psi ij equal to, so you can write 1 by delta x square psi i plus 1j plus 1 by del x square psi i minus 1j plus 1 by del y square psi ij plus 1 and plus 1 by del y square psi ij minus 1 and if you write in the coefficient way.

Let us write that, this is your i plus 1j. So, if we write this point as P, this is your east E, it is west W, it is north and south and the coefficient for these points if we write then we can write aE. So, aE is the coefficient of phi i plus 1j, So, that is 1 by delta x square, aW is the west at this point, so what is the coefficient of psi i minus 1j? So, it is 1 by delta x square.

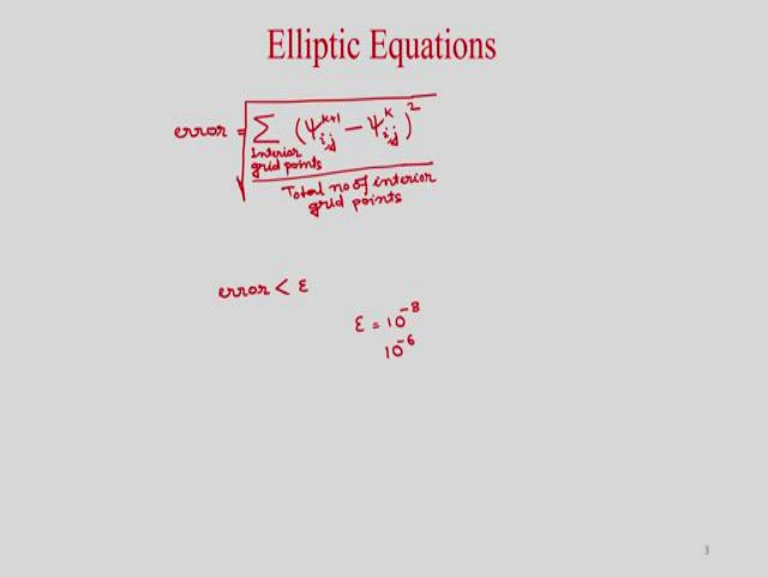
Similarly, a north so, at this point ij plus 1 the coefficients of psi ij plus 1 is 1 by delta y square and a south at this point what is the coefficient? So, we are denoting the coefficient, 1 by delta y square and we are writing the diagonal coefficient aP as 2 into 1 by delta x square plus 1 by the delta y square.

So, this was just we were denoting the coefficient, so you can write here actually aP psi ij is equal to aE psi i plus 1j plus aW psi i minus 1j plus a north psi ij plus 1 and a south psi ij minus 1. So, we are solving for psi ij, so aP you just divide in the right hand side. So, you will get psi ij is equal to 1 by aP, aE psi i plus 1j plus aW psi i minus 1j plus aN psi ij plus 1 plus aS psi ij minus 1.

So, when you are going to write the program, so, you can write in any language whatever you know. Say, you may write in Fortran or C or C++. So, when you are going to write you first calculate all this coefficient aE, aW, aN, aS and aP, So, that it will be easy while solving this algebraic equation. For convergence, we are telling that you have to meet a

convergence criteria. So, to give the condition for the convergence we will calculate the error and will calculate error in this way.

(Refer Slide Time: 07:39)



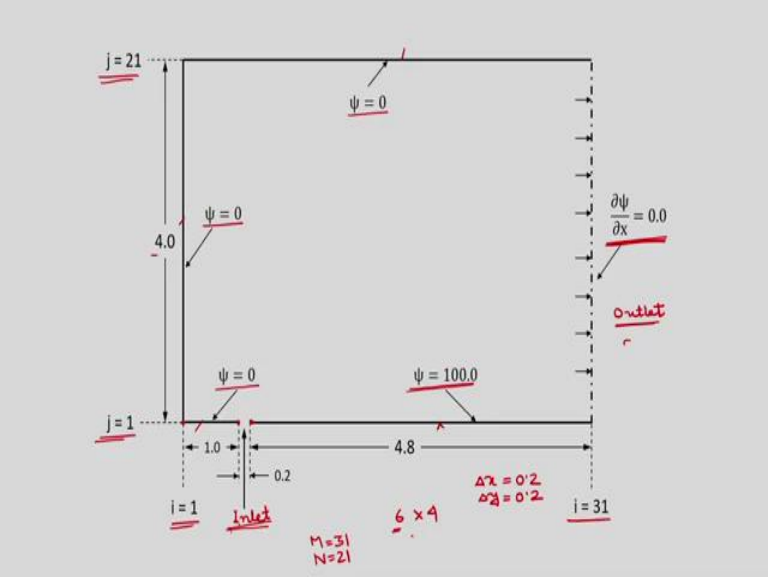
Error so, at each grid point will find the difference with the new value and the old value. Obviously, your psi new value is at psi K plus 1 and psi old value at psi K. So, we will right psi K plus 1 of point ij minus psi K at ij and we will just square it so that whether it is plus or minus it will give a positive and for all the interior grid points we will just sum it up, interior grid points.

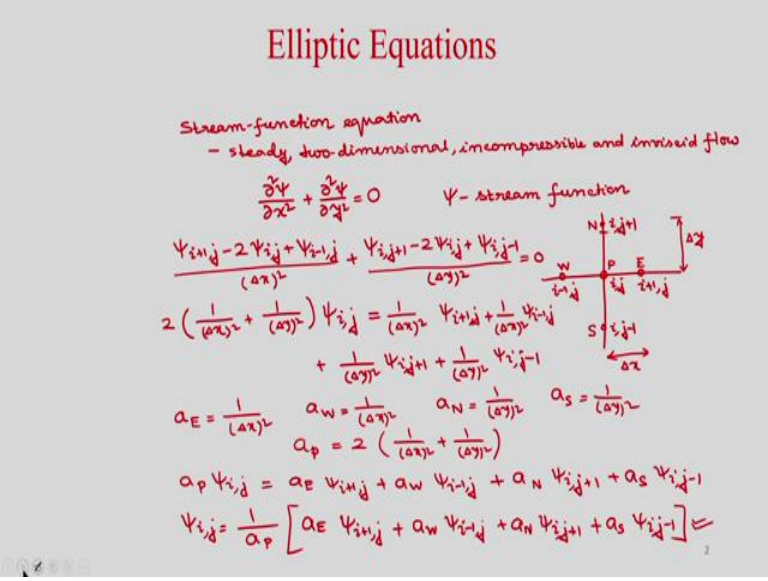
So, you sum it up and once you come out of the loop then you make it as summation. You first calculate this then you divide by total number of grid points, total number of interior grid points. So, first what we will do? We will calculate the difference at each grid points psi new minus psi old, then we will make it square, then we will sum it up in all the interior grid points, then we will just divide it by total number of interior grid points and square root of that will give the error.

So, after calculating this error, you just compare with a small value epsilon, if it is less than epsilon, then you just come out of the loop otherwise loop will continue. So, this error, if a less than some small value. So, small value let us say 10 to the power minus 8 or 10 to the power minus 6, will give a small value and if this error goes below to that

given small error value, which is 10 to the power minus 8 or minus 6, then you come out of the loop. So, you write the code in that way.

(Refer Slide Time: 09:58)





So, now let us define this problem. So, we are solving a stream function equation, here we have taken a domain of you can see 6 by 4. So, 6 by 4 domain we have taken and in x direction the length is 6 and y direction length is 4. Here you can see that it is origin, then from the origin at a distance 1 you have a inlet.

So, this is your inlet and here in the right boundary you have outlet. So, here in the left boundary. So, in the left boundary, top boundary and this boundary, we are specifying the stream function at psi is equal to 0 and here in the x equal to 1.2 to x equal to 6 in this valve we are specifying psi is equal to 100.

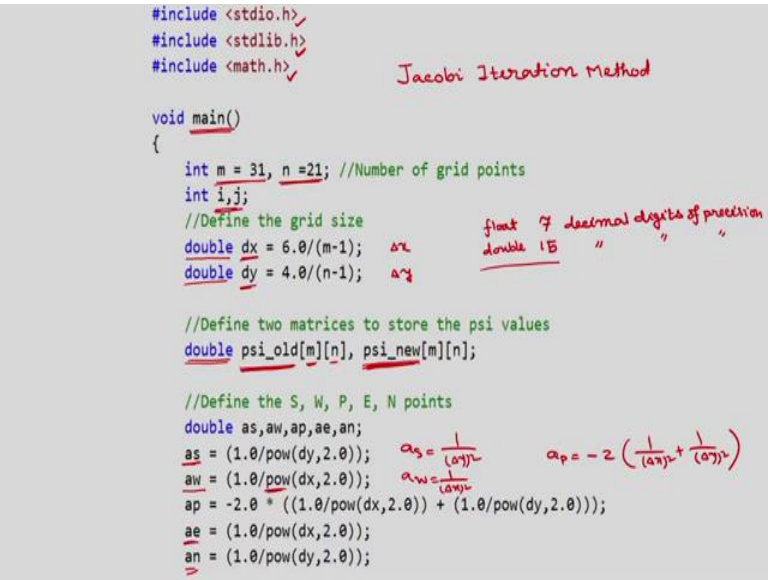
So, these are the boundary conditions and in the right hand side so, in the right boundary, in the outlet you give the boundary condition, the normal gradient to this boundary is 0 so, that means del psi by del x is 0. So, in all the boundaries we have given the boundary conditions and here at the you can specify the psi as linearly varying from 0 to 100 because at this point you have psi 100, at this point you have psi 0. So, linearly you can vary 0 to 100.

In this case, we are considering total number of grid points from 1 to 31 in x direction. So, your M is 31, total number of grid points and in the y direction we are varying j is equal to 1 to j is equal to 21 so j is equal to, N is equal to 21. So, if you see you are getting here delta x of distance 0.2 and similarly delta y you can find that it will be 0.2.

So, we are using constant step size delta x as 0.2 and delta y as 0.2. So, obviously two grid points will be there at here and here so, you are specifying the boundary condition of psi here 0 and psi 100 here, in the interior points now you need to solve this discretized equation, whatever we have derived in previous slide. So, you can see.

So, this is the equation we need to solve at all interior points and you apply the boundary conditions as specified here. So, what is happening? You just imagine that flow is coming through this inlet, these are walls were psi is equal to 0 and psi is equal 100 or specified and from the right boundary your flow is going out parallelly. So, if it is going parallelly than the normal gradient of psi will be 0. So, with these boundary conditions and your discretize equation, you just write the program.

(Refer Slide Time: 13:45)



So, I am not going to describe the details how to write the program, you should learn some basic programming language, Fortran or C or C++ and write the code for the given problem, so you practice it, you write in using any programming language, here I am going to show the Jacobi iteration method and Point Gauss Seidel method using C programming language.

So, you can see here, so this is just we are adding the header files so, you know that it is stdio dot h, that means standard input output, library you are using for printing and taking the value, stdlib, so standard library. So, this is for allocating the memory. So, if you are using some array or dynamic memory so, for that you need to include this library and math dot h. So, mathematical operation you need to include this header file.

So, this is the main program. So, we are specifying total number of points m as 31, n as 21. So, these are defining as integer, the index i and index j we are specifying at i integer ij. Now, we are finding the delta x and delta y, delta x and delta y here. So, length in x direction is 6 divided by m minus 1. So, 6 by 30 so, obviously it will 0.2 and delta y is 4 by n minus 1. So, it will be 4 by 20 so, it is also 0.2. So, we have constant steps size.

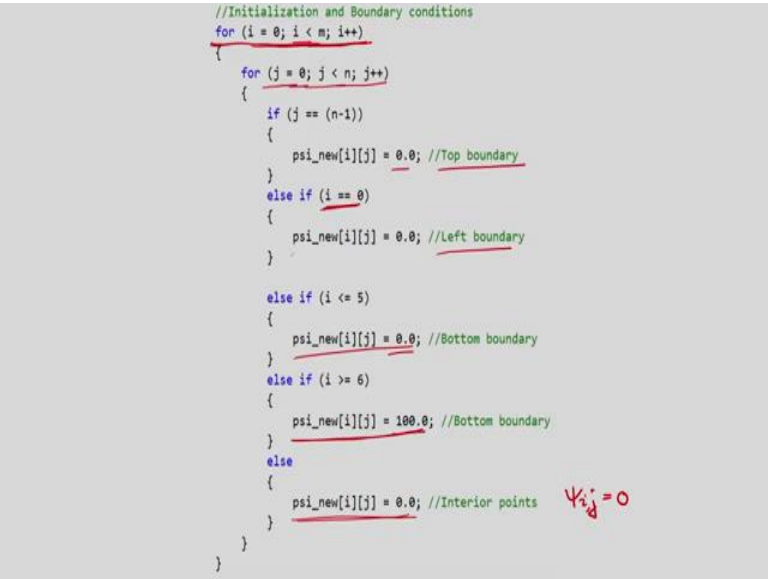
Now, we are defining the arrays, one is storing the new value psi new and storing the old value psi old at first we are showing the Jacobi iteration method, we are using Jacobi

iteration method to solve these algebraic equation. So, we need to store the old value as well because while solving the equation, the algebraic equation, we have seen that all the neighbor points will be at the old value.

So, for that reason we are writing these arrays psi new and psi old of size m and n. So, m in the i direction n in the j direction and you can see all these were specifying double. So, you can also use float but double you know that it is having 14 decimal digits of precision. So, in float, you have 7 decimal digits of precision and in double you have 15 decimal digits of precision. So, for getting more decimal digits we are using double and double we have specified delta x, delta y, psi old and psi new.

Now, you calculate all the coefficients. So, we have defined you see aS 1 by delta x square, aW as is 1 by delta y square, aW is 1 by delta x square, so that we have written, so this is the power delta x2. So aP is obviously minus 2 into 1 by delta x square plus 1 by delta y square. So, aP you have written minus 2 1 by delta x square plus 1 by delta y square, aR and aN similarly 1 by delta x square and an is 1 by delta y square. So, here we have calculated all the coefficient because these are constant it is not going to change while executing the code because delta x and delta y are constant.

(Refer Slide Time: 18:25)



So, now, you specify the boundary conditions and also you have to initialize the values because you have to guess a value. So, it is at when you are going from k to k plus 1. So, while starting the code at k you have to specify some values and that is known as initialization. So, at all interior points, you have to initialize some value of psi as well as you have to apply the boundary conditions.

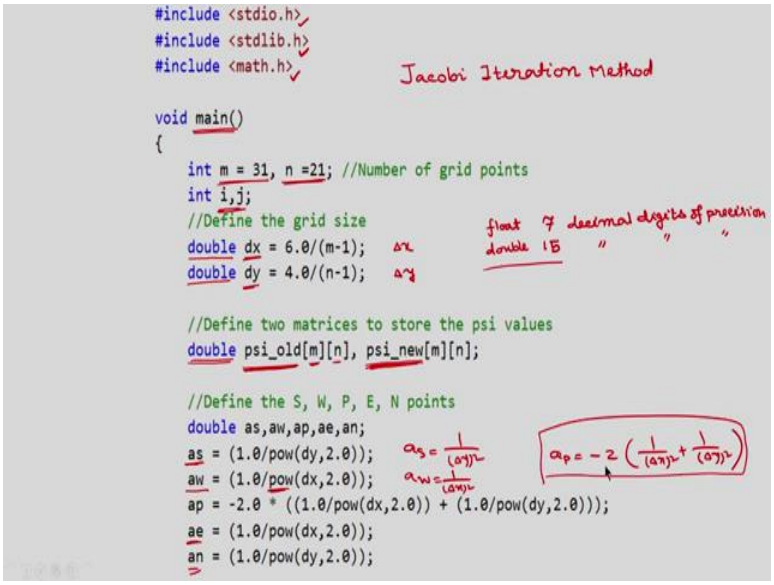
Here, only on the walls you have Dirichlet boundary conditions and on the right wall or right boundary you have Neumann boundary conditions. So, Neumann boundary conditions if you apply first order accurate scheme or secondary order accurate scheme depending on that one or two points or three points will be involved and you can see that you have to repeatedly calculate the value at the right boundary, because it involves the interior points and interior points value will change with iteration.

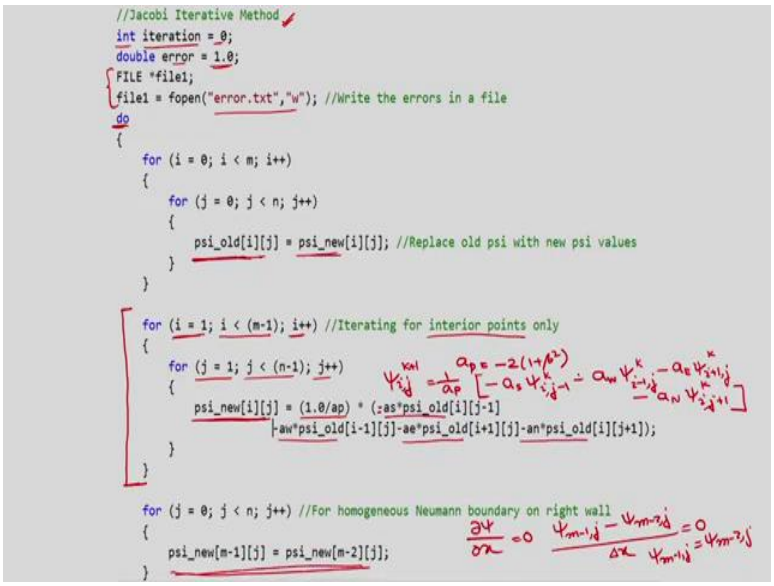
So, you can see here we are starting from i is equal to 0 to. So, we are using loop so, you can see this is your for loop. So, this is your for loop. So, you can see that it is your initial value, you are starting from i is equal to 0. Then you are actually seeing i is less than m. So, this is your condition you are applying and i plus plus you are so incrementing, so i plus plus. So, here in the x direction you are just used for loop and in the j direction also it is varying j is equal to 0 to j is less than n and j plus plus. So, here you are including the boundary points as well as interior points.

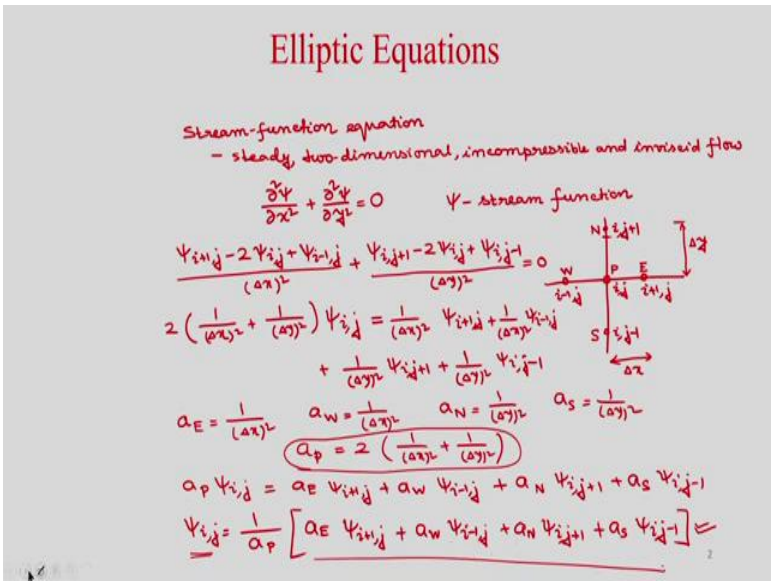
Now, for the boundaries now, you are applying, so top boundaries. So, psi new is 0, we have seen, the left boundaries so where I is equal to 0. So, if I is equal to 0 then you apply this left boundary condition it is 0, i less than equal to 5. So, bottom near to the origin that portion is having again 0.

So, psi new is equal to 0 and in the right side of this inlet, you can see you have 100 value. So, i greater than equal to 6 then you will get the boundary condition as 100 so, that you are applying and else so, if you have all the interior points, you are specifying at 0 value. So, you are putting psi ij is equal to 0. So, boundary conditions we have applied and in all other points we have applied psi as 0 value, you can start with some other values as well, you can take the average value of minimum and maximum that also you can assign. So, here we have assigned psi ij is equal to 0.

(Refer Slide Time: 21:35)







So now, we are starting the Jacobi iteration method. So, to know how much iteration it took just we are defining integer iteration. So, it is just defining 0 and double error, we are giving you a high value so that you are in this loop, it will go inside and this is just to write some values in a file. So, this is the syntax so, we have opened a file with error dot text where it will with the iteration what is the error so, it will print in a file.

Now we are starting the do while loop so, do while loop, so we are starting with do and the condition we are giving at the last so while writing the while, so this is do while loop and you can see that it is a exit control loop so, it will enter here, execute the statement then it will check at the end so, it is exit control loop.

Now, here first we are whatever values are there of psi new we are storing at the psi old. So, what we are doing? Psi k whatever the old values we are storing from the new value, because now we will calculate the interior points where psi new will calculate and it will have a new value. So, now to calculate the error we need the old values. So, we are storing this value as old so, that we can calculate the error.

Now, you write the main program. So, whatever Jacobi method you know, so that we are applying here. So, you have the algebraic equation there only one unknown is there that is your psi ij at k plus 1 and in the right hand side if you use Jacobi method in all the neighbor points i plus 1j, i minus 1j, ij plus 1 and ij minus 1, psi value you have to take

from the previous iteration value, that means k that means here we have to take the value from the old.

So, you can see here now, psi new ij so, here you can see the loop so now, we are looping over the interior points, all interior points we are looping. So, you can see we are starting from i is equal to 1 because boundary value we have given i is equal to 0 and i is equal to m so, interior point will be i is equal to 0 and i is equal to m.

So, interior point will be i I is equal to 1 to i less than m minus 1 and i plus plus so, here similarly j is equal to 1, j less than n minus 1, j plus plus now, we are calculating whatever governing equation you have discretized and written the final algebraic equation of that Laplace equation that we are writing here in Jacobi iteration method.

So, now psi new ij you are calculating 1 by p, this is your diagonal term, ap is nothing but minus 2 into 1 plus beta square into now you are writing all the coefficient. So, you can see that your as psi old, aw psi old, ae psi old and an psi old. So, this we are calculating the value of psi new. So, you can see this is the equation so, psi ij is equal to all these values and here you see ap we written plus, here ap we have written plus and that is why it is right hand side all are positive.

But while doing the coding I have taken ap as minus 2 1 plus delta x square plus delta y square and for that reason you are in governing equation, we are writing minus here so, because this ap is your minus so, it should be actually your positive so this minus we have written.

So, it is ap is minus 2 into 1 plus beta square. So, if it is so, so then your psi ij at K plus 1 is 1 by ap we have written as minus as psi K ij minus 1 minus aw psi K i minus 1j minus ae psi i plus 1j and minus an psi ij plus 1 and all at old iteration level.

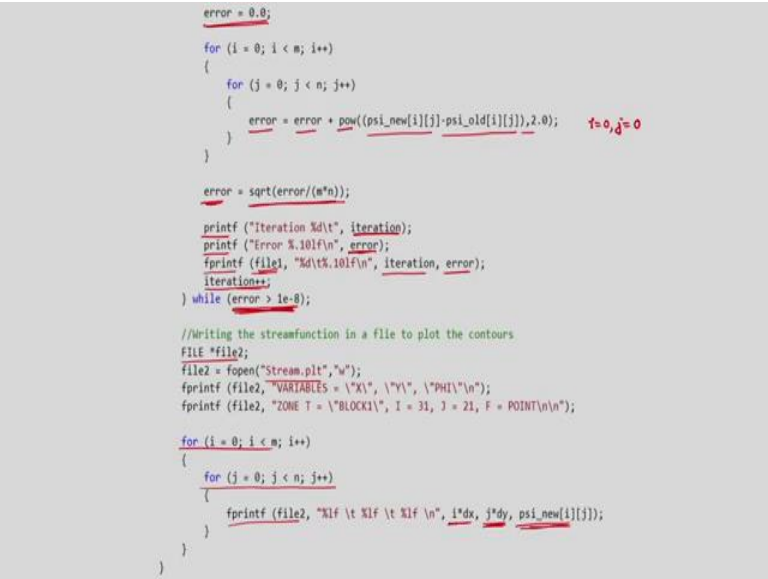
So, minus is coming for these notationally. So, this we have used the Jacobi iteration method and now, we need to apply the homogeneous Neumann boundary condition because here you have updated the interior points. So, at the right boundary value, you can immediately update using this first order discretization we have used. So, del psi by

del x is 0. So, you have used psi m minus 1j minus psi m minus 2j divided by delta x is equal to 0. So, this is your backward difference approximation we have used.

So, psi m minus 1j is equal to psi m minus 2j. So, we have just updated the value at the right boundary. Other places we have Dirichlet boundary condition. So, we do not need to apply the boundary condition but these inside this loop we have to apply because when you are solving for the interior points, you need the value of the boundary. So, right boundary is a Neumann boundary condition. So, you have to update this boundary value so, that at the interior points it will get the updated value. So, you can see we have updated here so, the first order accurate scheme we have used here.

Now, we are calculating the error. So, error first we are applying 0 because error is 0 now, here we have to sum it up. So, here we are doing just summing it up at all the interior points. So, i is equal to 0, i less than m, i plus plus and so all the points including boundary because right boundary also your values getting changed. So, you are writing error is equal to error plus that difference of psi new minus psi old its square so, that we are doing.

(Refer Slide Time: 28:49)



So, now you can see that you have 0 first then you got some value at first grid point at i is equal to 0 and j is equal to 0. Then it will go to i is equal to 2 then again it will loop over

all. So, you can see that this error will be summing up at all the grid points so that we are doing here, then when you are coming out of this for loop, then we are doing the square root of error, square root of error divided by total number of points.

So, total number points is m into n so that we are dividing. So, error is equal to square root of error divided by total number of points which is m into n. So, these error now you check whether it is smaller than the specified value of epsilon which we have specified maybe 10 to the power minus 8 here. So, if it is greater than 10 to the power minus 8 then it will continue once it becomes less than 10 to the power minus 8. So, it will come out of the loop and you can see now, we are in a screen we are printing what is the iteration and corresponding error.

So, this is iteration is integer, error is double and in the file whatever we have open that, in that file we are printing iteration versus error. So, each iteration how the error is decreasing that we are printing in a file. Then we are incrementing dilatation because we started with iteration 0, now we are incrementing. So, this loop will continue till this condition is satisfied. So, this is the convergence criteria and the error we have calculated you can see using this way, so to plot the stream function, you need to write in a specified format.

Here after solving this equation what you will get? You will get only data. So, at each grid point you will get the value of psi. So, seeing the data you will not visualize anything. So to visualize you need to post process it using some post processing software. So, you can use MATLAB or you can use Tecplot or any other post processing software to visualize how your psi looks.

So, you have given boundary conditions you have solved the governing equations. Now, after solution whatever data you are getting, how it looks so, for that in a Tecplot format we have actually written you can see for all the points boundary as well as interior in a file 2 which is your stream dot plt, and we are plotting the x, this is your y and this is the psi value.

So, it is a Tecplot format where you can write in some other format to visualize the data point. So, these are we have described how to write a C program for the stream function equation using Jacobi iteration method.

Now, let us see if you use Gauss Seidel method, Point Gauss Seidel method, then how will you write. So, in Point Gauss Seidel method, what is the difference with the Jacobi iteration method? The difference is that whatever updated values are available that you use, because some point already you have solved at k plus 1 level. So, that is available at k plus 1 level so that value you just use for the computation of psi at ij.

(Refer Slide Time: 32:43)

```
//Gauss Seidel Iterative Method
int iteration = 0;
double temp, error = 1.0;
FILE *file1;
file1 = fopen("error.txt","w"); //Write the errors in a file
do
{   error = 0.0;
    for (i = 1; i < (m-1); i++) //Iterating for interior points only
    {
        for (j = 1; j < (n-1); j++)

        {
            temp = psi[i][j];
            psi[i][j] = (1.0/ap) * (-as*psi[i][j-1]-aw*psi[i-1][j]-ae*psi[i+1][j]-an*psi[i][j+1]);
            error = error + pow((psi[i][j]-temp),2.0);

        }
    }
    for (j = 0; j < n; j++) //For homogeneous Neumann boundary on right wall
    {
        psi[m-1][j] = psi[m-2][j];
    }
    error = sqrt(error/(m*n));
    printf ("Iteration %d\t", iteration);
    printf ("Error %.10lf\n", error);
    fprintf (file1, "%d\t%.10lf\n", iteration, error);
    iteration++;
} while (error > 1e-8);
```

So, for that similar way now, in this case we do not need to write old values. We are not using the psi old value because whatever updated that anyway we will use. So, you do not need any array for psi old. So, that you have to remember, so here you see what we are doing the rest of the things you have to anyway define, only thing is that here you see, here, when we are going in the do while loop, do while loop rest of the things you have to do whatever way Jacobi method we have written the coefficients all these you have to calculate I am showing only the difference where in the loop we have.

So, now do while loop error you define then you loop over the interior points. So, all the interior points you are looping. Now, to calculate the error of the interior points we are

defining temp is equal to psi ij. So, you can see that psi ij is calculated from the next equation, before calculating we are storing in a temporary variable and that we have defined as a double here.

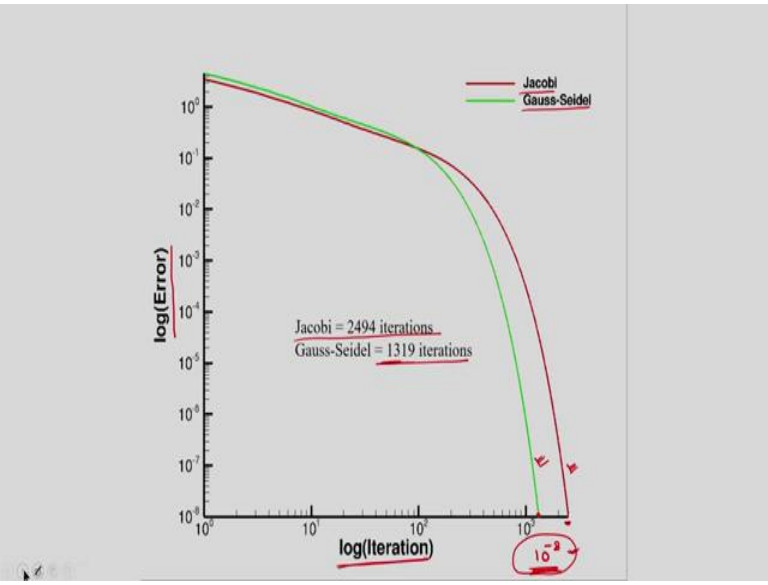
So, here we are storing this psi value first in a temporary variable, it is not array. So, it is just double. So, this you just store it then you calculate the psi ij, psi ij now you calculate whatever way we discussed the Gauss Seidel iteration method.

So, in Gauss Seidel iteration method in the right hand side whatever updated values are available that you will use. So, that is why you are using, 1 by p as psi ij minus 1, aw psi i minus 1 j, ae psi i plus 1jand an psi ij plus 1. So, you can see here we are calculating psi ij and wherever it is already calculated, the updated values will be used. If it is not calculated, then the old iteration value it will take, so the previous iteration value automatically it will take, so you do not need to define a different variable for psi old.

So, here that is why you have used, so whatever updated values of psi are available that will be used otherwise the old iteration value will be used. So, that is a difference and immediately we are calculating the error because error plus your psi new minus psi old. So, it is temp where we have stored the psi value, so that we can now old, because we have calculated the new value of psi.

So, psi minus temp whole square that we have calculated and that we are summing it up and after coming out the loop you have to just calculate square root of error, divided by total number of points m into n. So, if you do then that you just compare with the specified value whether it is less than it or not, then you just continue the loop till it converges. So, this is the difference.

(Refer Slide Time: 36:03)

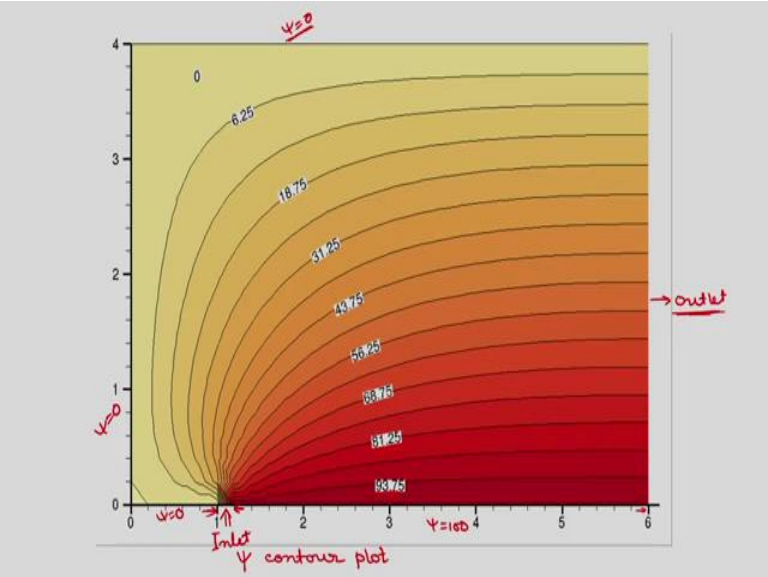


So, you see the error we have printed in a file with different iteration and that we have plotted in each x direction log iteration, x axis log iteration, y axis log of error we have plotted, with Jacobi iteration method and Gauss Seidel iteration method. So, you can see your Jacobi iteration method is the red color continuous, this line. So, you can see these line obviously it is taking more iteration and to converge up to 10 to the power minus 8 it took 2494 iterations.

So, Jacobi iteration method took 2494 iteration but when we use Gauss Seidel method, in Gauss Seidel method the updated value already we have used and with that the error, you can see the error plot. So, here to converge up to 10 to the power minus 8 it took 1319 iteration.

So, you can see obviously your Gauss Seidel is converged faster than the Jacobi iteration method. So, for the same level of convergence criteria. So, 10 to the power minus 8 is your convergence criteria and Gauss Seidel. So, this green color line we have shown the log error versus log iteration for Gauss Seidel method and it took less number of iteration to converge then the Jacobi iteration. Now, let us plot the psi now, visualize the data point.

(Refer Slide Time: 37:48)



So, you can see this is the visualization of psi so, this is your psi contour plot. So, each line is showing the constant psi line because you can see this is your boundary. So, this is the boundary where psi is 0 is specified, here. So, from here these top and left and up to this point, you have specified psi is equal to 0 it is a constant psi line and from 0.2, 1.2 x equal to 1.2 to 6, you have specified 100, psi is equal to 100 and here psi is equal to 0, here psi is equal to 0 and psi is equal to 0.

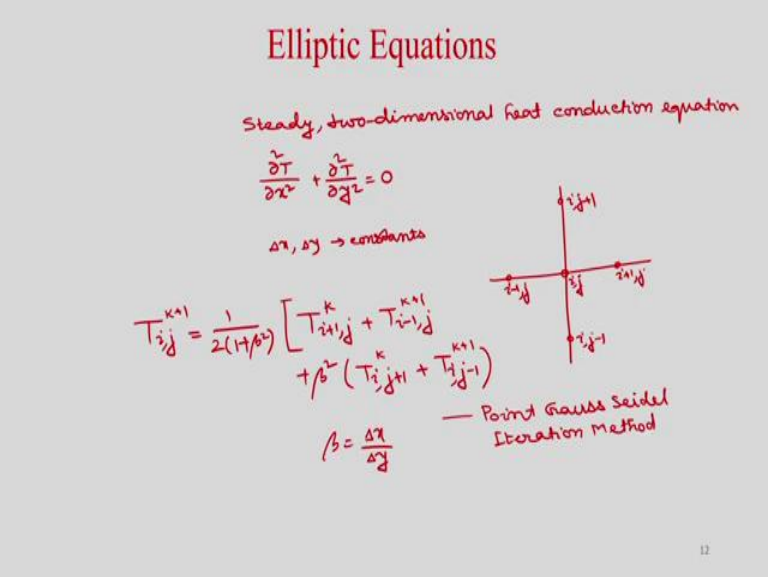
So, now, this is your inlet, this is your inlet and this is your outlet. So, flow is coming here and it is flowing this way and it is going out parallely through the outlet. So, each constant line is showing constant stream function. So, you can see obviously, this is your psi is equal to 100 then gradually it is decreasing. So, we have shown 93.75 and 81.25 and gradually it is decreasing and it is becoming psi is equal to 0 on this wall.

And you can see that from the inlet it is coming in and through the outlet it is going out and whatever data points you generated by solving the Laplace equation at all interior points as well as the boundary points that we have plotted using some post processing software and we are visualizing the contour up stream function.

Now, so, you can see this is some application problem and you should solve this equation for different problem and try to write the code. Similarly, if you solve the steady state,

two dimensional heat conduction equation, then you will get similar equation and you can solve similarly using some iterative techniques.

(Refer Slide Time: 40:12)



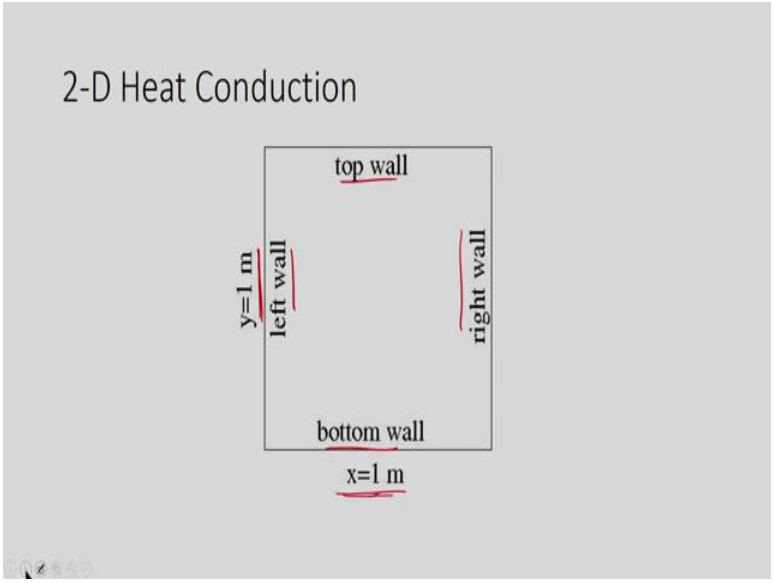
So, if you consider steady, two dimensional heat conduction equation. So, what is the equation? del 2T by del x square plus delta 2T by del y square is equal to 0. So, now you discretize using central difference and write the final algebraic equation using phi points formula. So, if we use phi point formula it is ij, this is your i plus 1j, this is your i minus 1j, this is your ij plus 1 and this is your ij minus 1 and with constant step size, delta x and delta y are constant.

So, you can write the algebraic equation, so if you compare it whatever we have done for the phi. So, you can write T i minus 1j plus or you can write Tij is equal to 1 by 2 into 1 plus beta square. So, T i plus 1j plus Ti minus 1j plus beta square Tij plus 1 and plus Tij minus 1.

So, beta is the ratio of step size delta x by delta y and depending on which type iteration method you are using you just put the superscript K. So, if you are solving this your K plus 1, if you are solving using Gauss Seidel, Point Gauss Seidel method then you can use i plus 1. So, you have not solved yet it is K, i minus 1 you have already solved, Tij plus 1 you have not solved so it is value available K and Tij minus 1 already you have

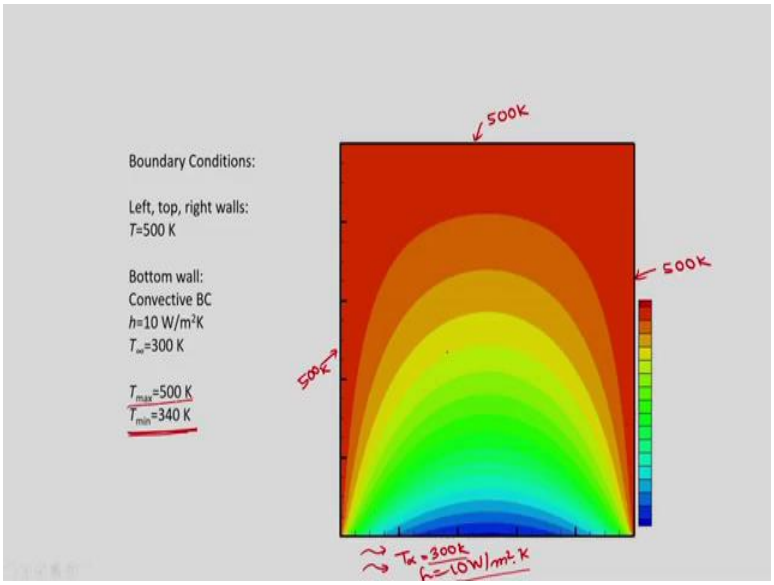
solved so, it is available at k plus 1. So, this is your Point Gauss Seidel method, iteration method. So, you just write the program and solve this equation just I will show the contour plots of the temperature for a given boundary conditions.

(Refer Slide Time: 42:52)



So, we are considering this square domain where this is your bottom wall, left wall, top wall and right wall. So, with x equal to you do not need to give the dimension because in non dimensional form also you can solve.

(Refer Slide Time: 43:10)

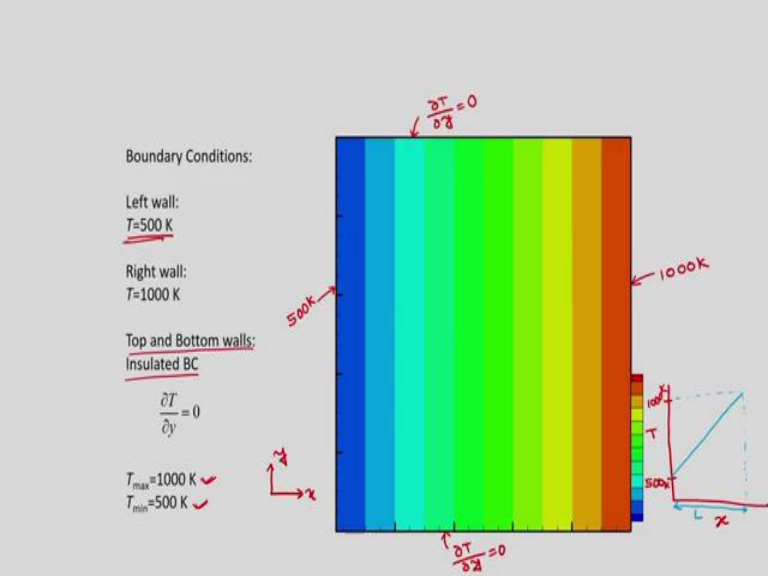


Now, for this problem we have given the boundary conditions left, top and right walls at 500 Kelvin, this is your 500 Kelvin and bottom wall is having the convective boundary condition. So, with now convective boundary condition means you have mixed type boundary condition.

So, here you have T infinity as 300 Kelvin and H is 10 watt per meter square Kelvin. So, it is cooled, because 300 Kelvin is the ambient temperature and ambient fluid heat transfer coefficient is 10 watt per meters per Kelvin so, by convection, this bottom wall is cold. So, you can see after solving this Laplace equation of temperature you can get the temperature profile like this.

So, these are contour lines. So, you can see this is your 500 and as your heat is convicted here. So, gradually there will be change and the minimum temperature we got here 340 Kelvin and maximum obviously it is 500 Kelvin and it varies from 340 to 500 Kelvin. So, these are all isotherms, isotherms means, it is a constant temperature line.

(Refer Slide Time: 44:43)



Another boundary condition if you take say you have left wall 500 Kelvin. So, this is your 500 Kelvin. So, this is your Dirichlet boundary condition, right wall is your 1000 Kelvin, so this is Dirichlet boundary condition but top and bottom walls are insulated that means adiabatic boundary conditions. So, these are adiabatic boundary conditions that

means del T by del y is 0. So, there is no heat loss through this boundary. So del T by del y is 0. So, no heat loss from this boundary.

So, with this condition if you solve it obviously you can see that your temperature will lie between maximum value of 1000 Kelvin and minimum value 500 Kelvin and it is varying from 1000 to 500 linearly along the x direction because there is no heat loss. So, obviously your isotherm is cutting these top and bottom boundary perpendicularly because that is your del T by delta y is 0, it has to be satisfied.

So, to satisfy this condition your isotherms, that means constant temperature line cuts the top and bottom boundary perpendicularly. So, for that essentially it becomes a one dimensional head conduction you can see that there is no variation of temperature in the y direction.

So, if it is x direction, if it is y direction, then you can see there is no variation of temperature in the y direction because it is a constant temperature line, these are straight line, only variation is taking place from 500 to 1000 linearly in the x direction. So, if you plot. So, this is your x and this is your temperature.

So, you can see if this is your 500 Kelvin and this is your 1000 Kelvin. So, it varies linearly, this is your just length of the geometry. So, this is the length of the geometry. So, it varies linearly, so you have seen that this type of boundary condition, now this is your insulated boundary condition. So, this is a Neumann boundary condition, in earlier problem we have used conductive boundary conditions, so your gradient is involved and you can discretize using either first order scheme.

So, where two points will be involved, if you use one sided differencing, so using three points then you will get a second order accurate scheme. So, that also you can use, so you should try writing the program for different type of boundary conditions as well as with different order of accuracy. So, we will stop here today, thank you.