Matrix Solvers

Prof. Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 41
Developing Computer Programs for Basic Iterative Methods

Welcome over last few lectures we have looked into the Algorithms for different solution methods both direct and iterative way for matrix equations. However, these algorithms are designed to solve large matrices, which we really cannot handle using simple paper and pencil and as we have mentioned this several times a good solution technique should be also designed, keeping into mind that it has to be translated into a computer program.

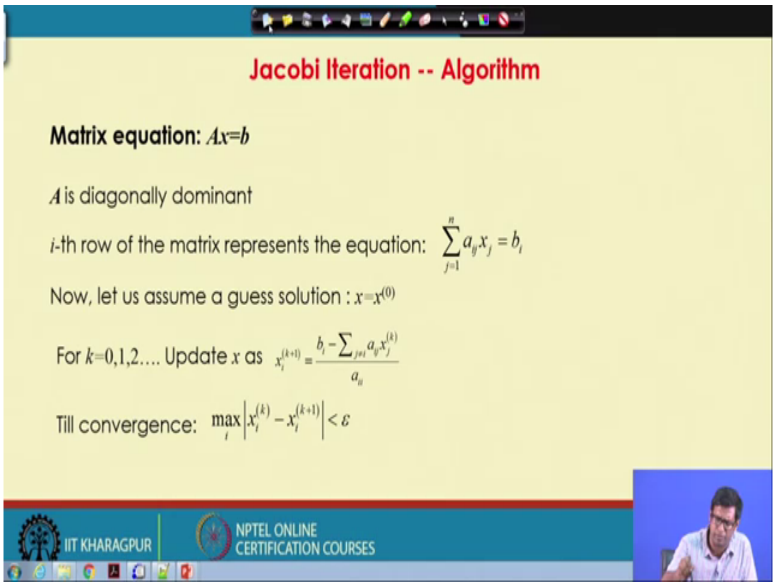
So, in this particular session and also in the subsequent class we will start looking into how we can make computer programs from these solution algorithms. So, mainly look into c programs, but any programming language as well as the programming platforms like matlab can be used for writing these programs and you will see plethora of already writ10 programs in Jacobi algorithm for Gauss Seidel or for the algorithms, well discuss in future like by conjugate stabilized algorithm or generalized minimum ratio algorithms plethora of examples are there of this algorithms translated into computer programs in different C C plus plus Fortran Fortran 90 Python as well as matlab r and these are well available in internet resources.

So, sometimes we pick these algorithms for from open source codes available in internet as internet resources and plug it into the particular application we are trying to develop and sometimes also there in form of some software or some executable built using the source code is not available to us on the executable, we chooses these essentially these algorithms for matrix solutions and we plug call those softwares we plug this executable with our program and (Refer Time: 02:25).

Nevertheless it will be important to look into inside the program and what is inside the code. So, well primarily look into few examples using c and also you look into few Fortran implementations and see how different variants of similar solution technology, like for basic iterative method how different variants Gauss Seidel, Jacobi, Sor can be written as computer programs and later we will see, how the generalized production processes like steepest descent, minimum residual method etcetera also come and when we will discuss about conjugate gradient by conjugate stabilized.

We will also discuss about those algorithms now but based on the understandings we have built and the algorithms we have yet covered we will concentrate on a few of the basic iterative algorithms and well start with Jacobi, Gauss Seidel and SOR in this particular lecture. So, this is on computer programming for basic iterative methods.

(Refer Slide Time: 03:34)



Jacobi Iteration -- Algorithm

Matrix equation: $Ax=b$

A is diagonally dominant

i-th row of the matrix represents the equation: $\sum_{j=1}^{n} a_{ij} x_j = b_i$

Now, let us assume a guess solution : $x=x^{(0)}$

For $k=0,1,2....$ Update x as $x_i^{(k+1)} = \frac{b_i - \sum_{j\neq i} a_{ij} x_j^{(k)}}{a_{ii}}$

Till convergence: $\max_i \left| x_i^{(k)} - x_i^{(k+1)} \right| < \varepsilon$

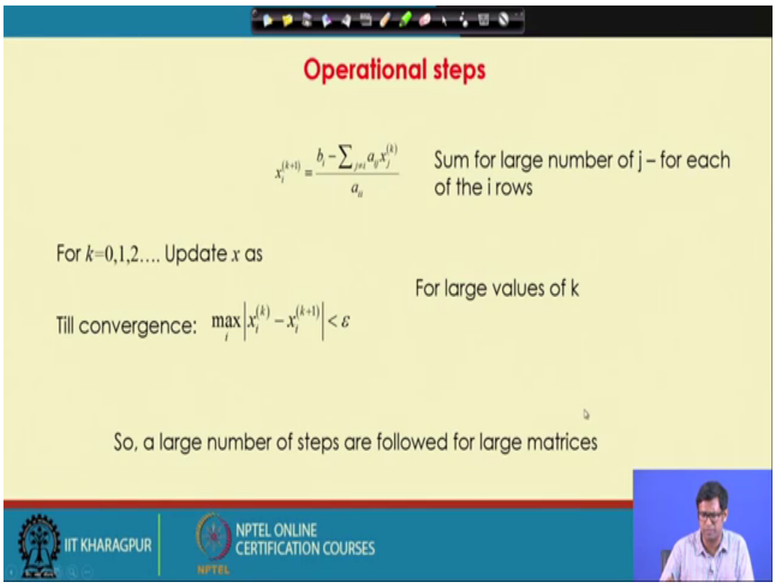
So, if we look into the Jacobi iterations algorithm, we are solving a matrix equation Ax is equal to b and as we know A is diagonally dominant or irreducibly diagonally dominant for implementation of this particular algorithm and in a Jacobi method the ith row of the matrix in general the ith row of the matrix is given as a sum of a ij x i is equal to bi. Now let us assume I will get solution x is equal to x 0 for different values of the different elements of the (Refer Time: 04:09) vector x we start with a guess solution and for the levels of k which are the iteration levels k is equal to 0, 1, 2 we update x as x i k plus 1 is equal to bi minus a ij xjk where j is not is equal to i 4. So, for all of diagonal ai j we multiply it with the corresponding x and divide it by aii.

And we iterate it we will so, we get a k. So, we start with x 0 you got x 1 using x 0 here and then with x 1 we get a using x 1 here we will get x 2 using x 2 here we will get x 3. So, you are updating all the elements of the vector x at different level and we will do this till we get convergence that is x i k minus x ik plus 1 its absolute maximum of the absolute values is less than epsilon and we have seen that once these value convergence to a very small number the equation a ij xj is equal to bi or the residual of the equation that is bi j minus sum of a ij x i for each row of i also converges to 0.

So, converges convergence of the x vector also imply that the residual will converge to 0 or the equation will be approximately satisfied to a to a high accuracy, based on the parameter epsilon and this epsilon, usually depends on machine, machine accuracy is something it is a small value it is a 10 to the power minus 8, 10 to the power minus 12, 10 to the power minus 16 etcetera.

Based on based on what type if we are using single precision variables you usually stick to 10 to the power minus 8, but there are certain cases using double precision and high double precision computer and high accuracy algorithms you can go for higher level of the value of 10 to the power minus 16. So, epsilon is a small number the differences between the present value and the updated value of the expector must be small that is the idea which will carry forward.

(Refer Slide Time: 06:21)



Operational steps

$$x_i^{(k+1)} = \frac{b_i - \sum_{j\neq i} a_{ij} x_j^{(k)}}{a_{ii}}$$

Sum for large number of j – for each of the i rows

For k=0,1,2.... Update x as

For large values of k

Till convergence: $\max_i \left| x_i^{(k)} - x_i^{(k+1)} \right| < \varepsilon$

So, a large number of steps are followed for large matrices

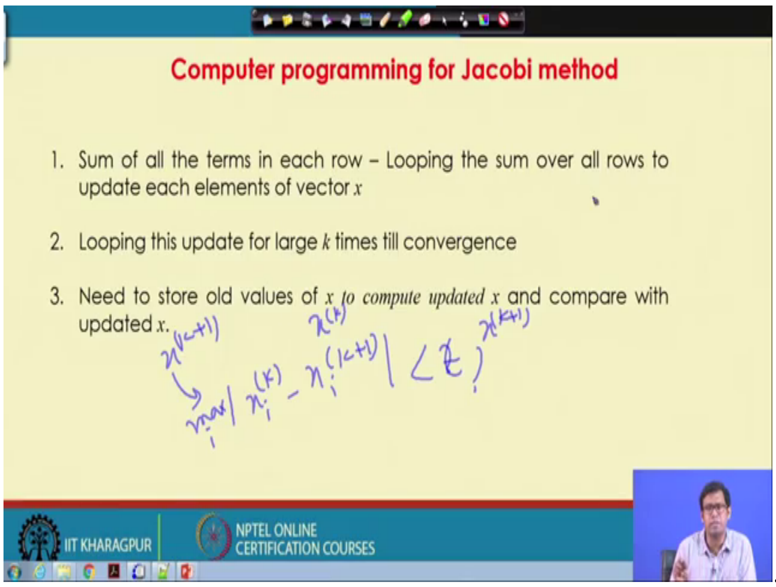
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The Operational steps main operational steps, we will look into this particular case for every at any iteration level for every element of x or every row of the equation we have to find out x ik plus 1 is equal to bi minus sum a ij its xj j is not is equal to i divided by aii; that means, this sum has to done for all the elements in a row or for all the members of the vector x if the x if I I have a 2000 by 2000 matrix x s 2000 element. So, this summation has to be done 1900 and 99 times except the diagonal term and this now this operation has also been repeated for all the rows.

So, sum of four large number of j j is not is equal to i for each of the i rows. So, we have to do it for 2000 i rows and in each row we have to do it for 1900 99 times. So, it said I had a large number of operations if we think of doing it manually and also the update of k x k plus 1 is and xk this k plus the with different values of k, I will find out x a different iteration level, this update has to be found out till this value is less than epsilon which is also a large number of iterations has to be done and we will see some example that; iteration numbers can go up to few 100, few 1000, 10000 also depending on the matrix size and the nature of the matrix.

So, for large values of k this has to be related. So, large number of steps are to be followed for large matrices both in terms of the multiplication and find finding out x ij plus 1 as well as the same operation this operation has to be looped over a number of iterations, till we get a high value of case. So, that the solution converges.

(Refer Slide Time: 08:28)



Computer programming for Jacobi method

1. Sum of all the terms in each row – Looping the sum over all rows to update each elements of vector x
2. Looping this update for large k times till convergence
3. Need to store old values of x *to compute updated* x and compare with updated x.

$x^{(k+1)}$ $x^{(k)}$ $x^{(k+1)}$

$$\max_i |x_i^{(k)} - x_i^{(k+1)}| < \epsilon)$$

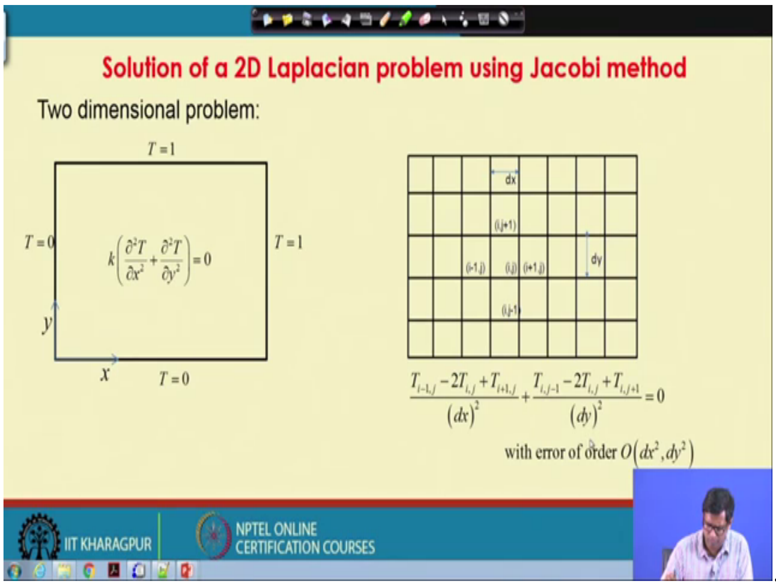
So, sum of computer programming for Jacobi method will essentially have few important step sum of all the terms in each row that is, looping the sum over all rows to update each element of vector x. So, we have to find out for all js in one particular row and then we have to loop it over all rows to find out all image elements of vector x and then looping this update for large k times till convergence. So, once we do this is only one step and it will not converge to the right result. So, you have to loop this for several time till you get a convert solution.

Also you need to store old values of x x k once it is it will be used to compute the updated xk or xk plus 1 and as well as it has to be compared with the updated x. So, what I will say that xk x at kth level has to be stored to calculate x at k plus 1 as well as, but once we calculate this we cannot discard this variable, because this will be further needed to compare with xk plus 1 in in the way we have to see what is the max of i x i k minus x i k plus 1 is this is this less than epsilon or not. So, you have to also compare the variables are different levels.

So, these three are the main steps when you think of a computer program and a computer program essentially does few arithmetic calculations like addition, multiplication, division etcetera and also there are logical loop logical gates like if the value is less than epsilon, then we will say that the solution is converged and will not carry forward the calculations, if it is greater than epsilon then we have to again go back to the previous type set of operations; that means, calculating of xk plus 1 using xk.

So, you have to loop over the rows to calculate xk plus x ik plus 1 for different value of i and again we have to put a loop of calculation of xk plus 1 x ik plus 1 for a large number of values of k, once we get it we will start with k is equal to 0 will calculate x except x 1 wills do then we have k is equal to 1 and we will calculate x 2 and so on. So, you have to loop it over the values of k also. So, there are 2 loops similar operations that has to be done again and again in these 2 loops and there is an important logical step that, whether the value is less than epsilon or not.

(Refer Slide Time: 11:22).



Solution of a 2D Laplacian problem using Jacobi method

Two dimensional problem:

$T=1$
$T=0$
$k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)=0$
$T=1$
y
x
$T=0$
dx
(i,j+1)
(i-1,j)
(i,j)
(i+1,j)
dy
(i,j-1)
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

$$\frac{T_{i-1,j}-2T_{i,j}+T_{i+1,j}}{(dx)^2}+\frac{T_{i,j-1}-2T_{i,j}+T_{i,j+1}}{(dy)^2}=0$$

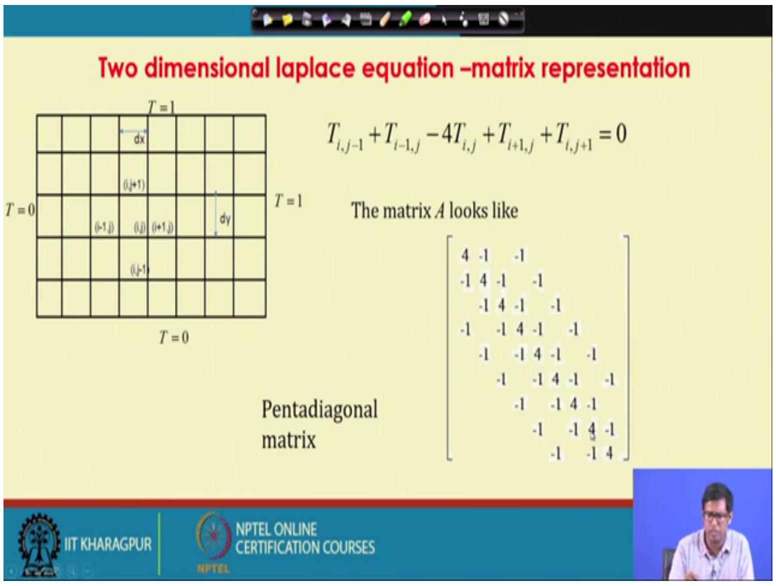
with error of order $O\left(dx^2,dy^2\right)$.

So, as an example we will look into a 2D Laplacian problem using Jacobi method, where solving del 2 T essentially delta 2 T del x square plus del 2 T del y square is equal to 0 and the boundary conditions are given T is equal to 1 T is equal to 1 A is equal to 0 T is equal to 0 as a matter of fact you can give any boundary condition of T here. When you convert it using finite difference when you convert it to a matrix to this difference equation Ti minus 1 j minus 2 T ij plus Ti plus 1 j by dx square plus T ij minus 1 minus 2 T ij plus Tij plus 1 by d y square is equal to 0.

And if dx is equal to d y we can write Ti minus 1 j plus T i j minus 1 minus 4 Ti j plus Ti j plus 1 plus T i plus 1 j is equal to 0. Now what we have to do we have to express this in the matrix form and we have discussed this earlier we renumber the points 1 2 3 4, 5, 6, 7, 8, 9, 10 11, 12, 13, 14, 15 16, 17, 18 so on. So, this particular point which is 19, 20 21, 22, 33 and this will be 32.

So, this particular point will give us a this is 24 matrix equation where there will be T 24, T 22, T 14, T 24, T 22 and we will get a matrix equation a t is equal to 0 for the 23 row all the terms will be 0 except 23, 22, 23, 33, 14, 22, 24, 22, 32, 22. We have discussed it earlier that we can express it now as a matrix equation and we will try to solve this matrix equation using a Jacobi method first and then we will see how this can be modified to Gauss Seidel or SOR.

(Refer Slide Time: 13:41)



Two dimensional laplace equation –matrix representation

$$T_{i,j-1} + T_{i-1,j} - 4T_{i,j} + T_{i+1,j} + T_{i,j+1} = 0$$

The matrix A looks like

$$\begin{bmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & -1 & & -1 & & & \\ -1 & & -1 & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & -1 & & -1 \\ & & & -1 & & -1 & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix}$$

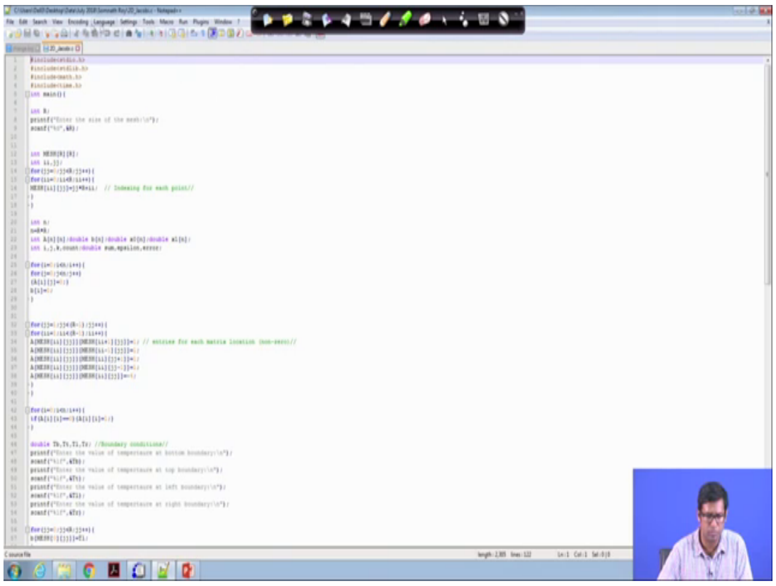
Pentadiagonal matrix

So, this is how the equation for each row and when we will write it in a matrix we will get up into a diagonal matrix few minus 1 1 4 and like multiplying minus 1 with here to get the positive diagonal term, it could have been done similar with the negative diagonal also because the absolute value of the diagonal is important not the value of the diagonal the sign is not important to look into diagonal dominant.

So, these are diagonally dominant system as when we will apply boundary condition the boundary terms we have 4 minus 1 1, so, 4 is greater than some of them. So, this is a diagonally dominant system, so we get a row like this and how can we solve this equation and there that will have a matrix like this and we will put the boundary conditions to make the matrix non singular and diagonally dominant and try to solve it using a Jacobi method.

So, what we will have to do in Jacobi method we have to find it start with an js value of x find the updated value of x for each node for each of the points you have to find out the updated value of x, then we will check whether the updated value and the previous value are very close to each other if not then we have to use take the updated value as the guessed value and then we have to find out the next updated value.

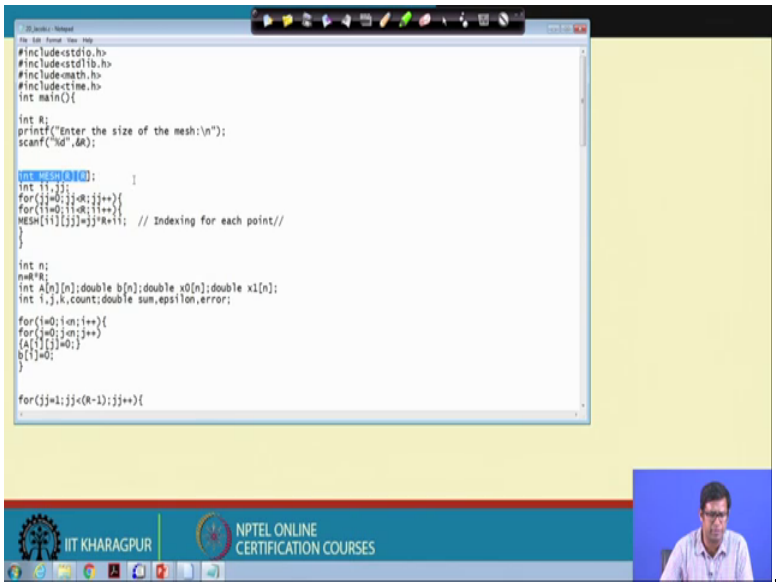
(Refer Slide Time: 15:06).



```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int main(){

int R;
printf("Enter the size of the mesh:\n");
scanf("%d",&R);


int MESH[R][R];
int ii,jj;
for(jj=0;jj<R;jj++){
for(ii=0;ii<R;ii++){
MESH[ii][jj]=jj*R+ii;  // Indexing for each point//
}
}

int n;
n=R*R;
int A[n][n];double b[n];double x0[n];double x1[n];
int i,j,k,count;double sum,epsilon,error;

for(i=0;i<n;i++){
for(j=0;j<n;j++)
{A[i][j]=0;}
b[i]=0;
}


for(jj=1;jj<(R-1);jj++){
for(ii=1;ii<(R-1);ii++){
A[MESH[ii][jj]][MESH[ii+1][jj]]=1; // entries for each matrix location (non-zero)//
A[MESH[ii][jj]][MESH[ii-1][jj]]=1;
A[MESH[ii][jj]][MESH[ii][jj+1]]=1;
A[MESH[ii][jj]][MESH[ii][jj-1]]=1;
A[MESH[ii][jj]][MESH[ii][jj]]=-4;
}
}

for(i=0;i<n;i++){
if(A[i][i]==0){A[i][i]=1;}
}

double Tb,Tt,Tl,Tr; //Boundary conditions//
printf("Enter the value of temperature at bottom boundary:\n");
scanf("%lf",&Tb);
printf("Enter the value of temperature at top boundary:\n");
scanf("%lf",&Tt);
printf("Enter the value of temperature at left boundary:\n");
scanf("%lf",&Tl);
printf("Enter the value of temperature at right boundary:\n");
scanf("%lf",&Tr);

for(jj=0;jj<R;jj++){
b[MESH[0][jj]]=Tl;
```

So, we can look into a Jacobi code maybe I should come out of this and open this in a notepad plus plus which is more real well visible. So, this is the Jacobi code 2 d Jacobi dot c I just have to improve the font size I hope that this is very visible now. So,

(Refer Slide Time: 16:38)



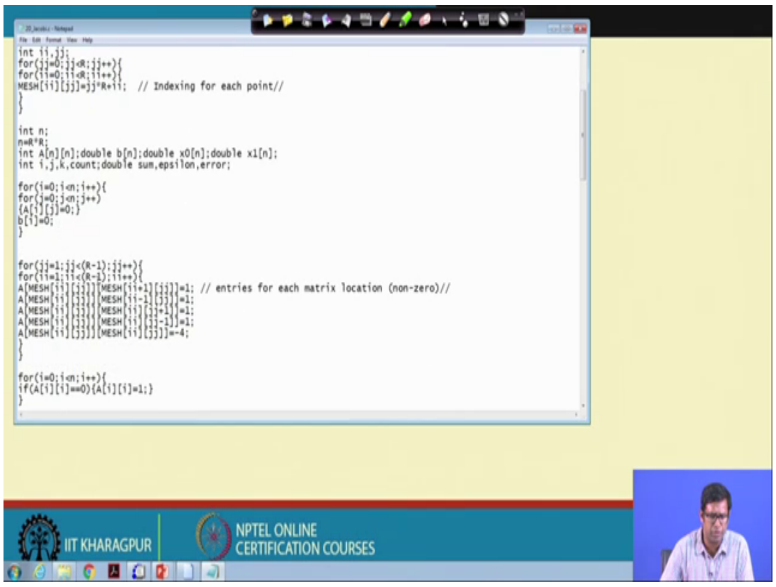
```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int main(){

int R;
printf("Enter the size of the mesh:\n");
scanf("%d",&R);


int MESH[R][R];
int ii,jj;
for(jj=0;jj<R;jj++){
for(ii=0;ii<R;ii++){
MESH[ii][jj]=jj*R+ii;  // Indexing for each point//
}
}

int n;
n=R*R;
int A[n][n];double b[n];double x0[n];double x1[n];
int i,j,k,count;double sum,epsilon,error;

for(i=0;i<n;i++){
for(j=0;j<n;j++)
{A[i][j]=0;}
b[i]=0;
}


for(jj=1;jj<(R-1);jj++){
```

.

So, this is the 2 d Jacobi code and it starts with all the header files you need for calculation for a c program, especially the math and time are important because some mathematical functions will be included and we also we are also trying to measure the performance of the code in a sense how much time it has taken.

So, you have to internalize of the mesh we took a 2 d mesh with same number of points in x and y direction. So, what will be the total number of points and you define the file mesh based on that.

(Refer Slide Time: 17:10)



```
int ii,jj;
for(jj=0;jj<R;jj++){
for(ii=0;ii<R;ii++){
MESH[ii][jj]=jj*R+ii;  // Indexing for each point//
}
}

int n;
n=R*R;
int A[n][n];double b[n];double x0[n];double x1[n];
int i,j,k,count;double sum,epsilon,error;

for(i=0;i<n;i++){
for(j=0;j<n;j++)
{A[i][j]=0;}
b[i]=0;
}


for(jj=1;jj<(R-1);jj++){
for(ii=1;ii<(R-1);ii++){
A[MESH[ii][jj]][MESH[ii+1][jj]]=1; // entries for each matrix location (non-zero)//
A[MESH[ii][jj]][MESH[ii-1][jj]]=1;
A[MESH[ii][jj]][MESH[ii][jj+1]]=1;
A[MESH[ii][jj]][MESH[ii][jj-1]]=1;
A[MESH[ii][jj]][MESH[ii][jj]]=-4;
}
}

for(i=0;i<n;i++){
if(A[i][i]==0){A[i][i]=1;}
}
```

And now you may this mesh is basically mesh will take basically the x and y coordinate x i and j index of 1 particular point and then with mesh ii ij will not locate this point into one particular like from 2 3 that is this point what is its absolute location in this mesh.

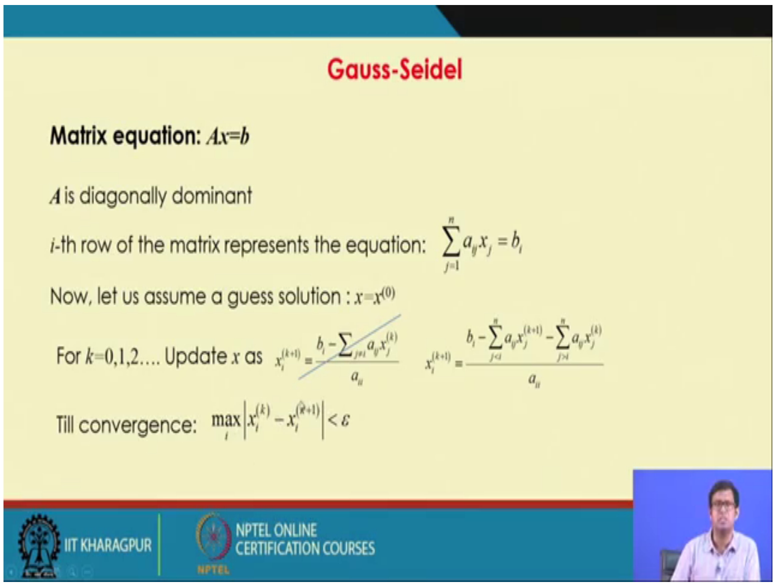
So, based on this we will form from the A matrix A and they this A matrix will have 1 in the off diagonals and minus 4 in the diagonal term and all other terms in the a matrix will be 0. So, we have initialized the matrix as 0 and now this terms are 1 and minus 4 and we also put the boundary conditions we read the value of temperature, which are the boundary conditions at for the sidewalls and the top and bottom wall and substitute this into the equation. So, he up to here we have got the a and b made a matrix and b vector.

Then the next step is that you use initial value of x the initial gauss all the xs are made to be 0 it could have been made anything in a Jacobi program, but here we try to make all the xs to be 0. So, in take the desired accuracy last for what will be the accuracy or what is the value of epsilon specify epsilon, what should be the value of epsilon and it is great as epsilon. So, the user will have an input that whether you will try to get a convergence up to 10 to the power minus four x k plus 1 minus xk is of the order of 10 to the power minus four or 10 to the power minus 8 or 10 to the power minus 12 will specify the value of epsilon like that.

And an error is calculated as twice of the epsilon clock time is started to take care of how much time its taking etcetera. So, and the loops are done where error is greater than epsilon initially we took some error twice epsilon, but later we will find out the error and in calculation error; that means, the difference between 2 values of x at 2 different iteration level, whether this is greater than epsilon or not if this is not greater than epsilon the loop will not be done and iteration count number of iteration is less than 1500 or not if we go for more than 1500 iterations we will say that this is not converging and will come out of it in this particular code.

So, error is xk minus xk plus 1 and the maximum value of because xk is also a vector xk plus 1 is also a vector there difference is a vector maximum value of that vector is a l l infinity, now we will take this as the error and this should be less than epsilon for the convergence. The next step we do for k loops we start with k is equal to 0 and go for k less than n and start adding k and later you can see the variable count is advanced in each iteration; that means, if k is equal to 0 count becomes 1 k is equal to 1 count becomes 2 etcetera count is the counter of the number of iteration its exactly the increases exactly the same way as k increases also. Now, what we have to do in the iterations.

(Refer Slide Time: 20:36)



Gauss-Seidel

Matrix equation: $Ax=b$

A is diagonally dominant

i-th row of the matrix represents the equation: $\sum_{j=1}^{n} a_{ij}x_j = b_i$

Now, let us assume a guess solution : $x=x^{(0)}$

For $k=0,1,2\ldots.$ Update x as $\cancel{x_i^{(k+1)} = \frac{b_i - \sum_{j\neq i} a_{ij}x_j^{(k)}}{a_{ii}}}$ $\quad x_i^{(k+1)} = \frac{b_i - \sum_{j<i}^{n} a_{ij}x_j^{(k+1)} - \sum_{j>i}^{n} a_{ij}x_j^{(k)}}{a_{ii}}$

Till convergence: $\max_i \left| x_i^{(k)} - x_i^{(k+1)} \right| < \varepsilon$

We have to find out this summation p i x i k plus 1 will be bi minus x i j a ij xj for j is not is equal to i. So, we will do these summations a ij xj up to the diagonal term and start with a i 1 x 1 and do up to aii x before ai i x 1 ai i i minus 1 x i minus 1 we will do this sum and then also we will start the sum with a i i i plus 1 x i plus 1 and go up to ai n xn.

So, this sum is broken into two parts one is up before the diagonal another is the times after the diagonal, we can see the program that is sum is equal to is the sum ai j xj into x 0 and this is done for j is less than i and then this is done for j is equal to i plus 1 to j is less than n. So, except i is equal to j that answer some and this sum is subtracted from b i and then is divided by ai i which gives me x 1 x 1 is the updated value of x.

X 0 was the gauss value x 1 is the updated value. So, we will find out what is the absolute value between x 1 and x 0 for 0 at vector and then will find it out for or other vectors and if the for the 0th element we will find it first for 0th element and then well check for which it is maximum, if it is maximum for first element then error is replaced by x 1 minus x 1 0 1 if it is more in second element error will be replaced.

So, that is why we will find out the maximum value of the error which is maximum value of xk plus 1 minus xk or x 1 minus x 0 for, but this look this k is not the iteration level k this k is the element number in the vector x and here our iteration loop ends. So, that the iteration counter will go and check whether count has been added by plus 1 1 where count is less than 1500; that means, 1500 iterations are not are done or not and error is greater than epsilon.

And once it is done it will come out and once the error is less than epsilon then this will not be satisfied; that means, the sub program has converged, the x x 1 and x 0 is minimally far away from each other the difference is less than a number which you have thought already. So, we will write down that this is the count and this is the error and we will write down the results. So, this is what we get made the Jacobi iteration method.

Now, we will look into few other methods, so, this is what we got in Jacobi iteration. Now we look into the Gauss Seidel method a is a diagonally dominant matrix same thing ith row is a ij ai e jxi is equal to b and we will start with x i is equal to x 0, but update is not x i k plus 1 is bi minus j is not equal to i aij xk i i rather the update will be x i k plus 1 is b i minus j is less than i a i j xjk plus 1 because this xs are already available minus a ij xj k.

So, this is done based on the values already known to us and then we will do the same loop it till we get convergence; that means, maximum value of this is less than epsilon. In the next section we will look into Gauss Seidel and Jacobi and successive over relaxation method, which is these codes can be very easily obtained by slight variation of Jacobi method and will also look into a different class of method which are the generalized which are the general projection methods like steepest descent minimum residual etcetera.

Thank you.