High Performance Computing for Scientists and Engineers
Prof. Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

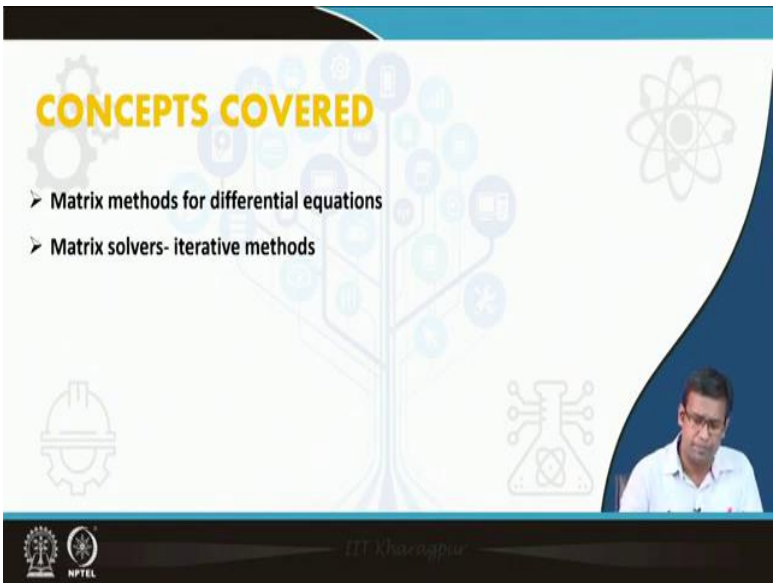
Module – 03
Message Passing Interface (MPI)
Lecture – 24
Matrix Representation of Physical Systems - Matrix Solvers

Hello, welcome to the 24th lecture of the course High Performance Computing for Scientists and Engineers and this is the 3rd module which is Message Passing Interface, and we will discuss Matrix Representation of Physical System and Matrix Solvers.

In scientific computing, a great interest is always involved with matrix solutions. We can see that many complex physical problems can be represented as matrix problems and we need to find a solution to matrix equations. These problems are very large in nature, they are computationally quite complex and therefore, we need to deploy scientific high-performance computing techniques.

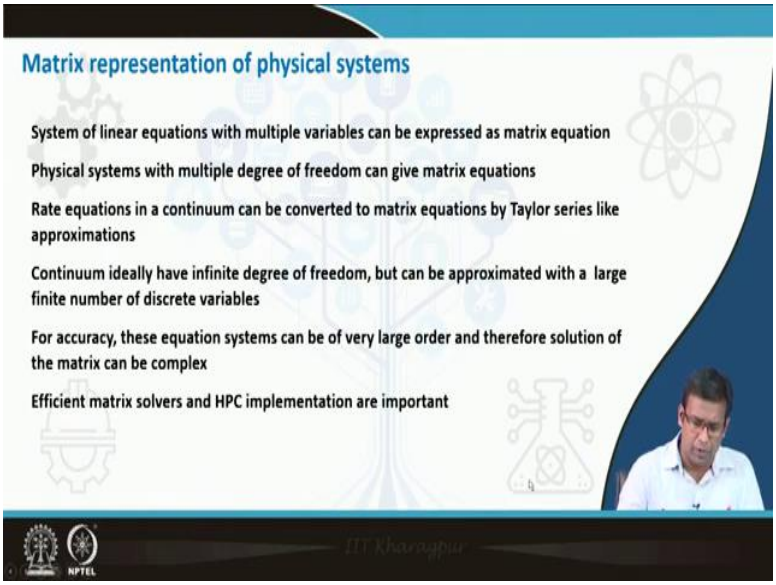
So, this particular class, we will discuss some fundamentals of matrix solvers, and some fundamentals of how we can get matrix equations from physical problems. In the subsequent classes we will see how MPI can be efficiently utilized to parallelize these problems and solve large problems in realizable time.

(Refer Slide Time: 01:35)



In today's lecture, we will discuss matrix methods for differential equations and matrix solvers especially iterative methods.

(Refer Slide Time: 01:45)



Why do we need to represent physical systems in the form of matrix equations? First is that, if we get a system of linear equations with multiple variables, they can be expressed as matrix equations, and the advantage of expressing them as matrix equations is that many common linear algebra techniques can be operated over matrix and we can easily get solutions. We can quantify the nature of solutions. We can predict certain things about the behavior of the system, if we can represent them by matrix equation.

If we have a large number of linear equations, a system of equations we can represent them as a matrix equation. Physical systems with multiple degrees of freedom; therefore, can give us matrix equations, because if we have multiple degrees of freedom, each degree of freedom will give us 1 equation, and if these equations are coupled, and if they are linear equations then we can represent them as matrix equations.

Rate equations in continuum can be converted into matrix equations by Taylor series like approximation. Continuum means, it's not discrete molecules or district elements it is a continuous medium over which some physical laws like conservation of mass, conservation of momentum, conservation of electricity, law of electromagnetism, some of the physical laws are active . Using these physical laws; we can get some rate equations, rate equations means,

they are differential equations; however, they can be converted into linear matrix equations by certain mathematical techniques

As we talk about continuum, this is a continuous media over which we are considering certain behavior, but as this is a continuous medium there are actually infinite degrees of freedom. If we think of something in space, at any point in space we can get say temperature, we can get a different temperature at any point inside a room. So, space is a continuous medium and there are infinite degrees of freedom in that.

However, we can approximate this infinite degree of freedom by a large finite number of discrete variables, because if we think about infinite dimensional space it is difficult to express it as in form of matrix equation or in form of discrete difference equations. So, instead of really going into infinity, we take a large number of points in the space, and, for getting an accurate representation, these numbers must be very large. Therefore, the equation systems which will get while trying to solve a rate equation in a continuum will be a large system of equations and therefore, the solution of the matrix will be computationally very complex.

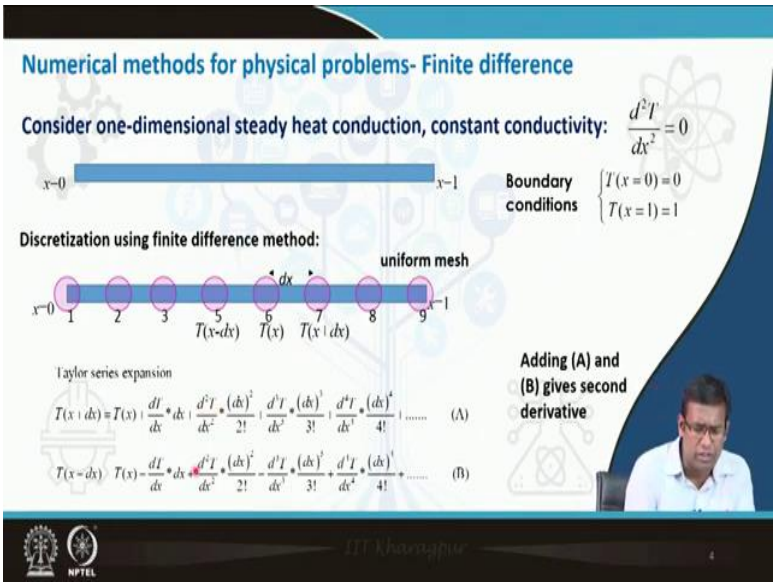
We need efficient matrix solvers for them. Matrix solvers means that given $Ax = b$, A is a matrix, b is a vector, x is a vector, we have to find a solution vector x.

So, you need efficient matrix solvers for doing that because we have a very large number of equations. In many cases, the matrix size can go up to billion by billion in certain cases. Even using a very good matrix solver may not be sufficient and we need to do high performance computing or parallel computing for that. The great interest in parallel computing has been developed within the scientific computing community, due to the fact that this matrix equation can be very efficiently and optimized to be solved early using HPC techniques.

So, in the subsequent classes we will see how matrix equations over a large problem, where the matrix size is very big ,can be solved using parallel computing. Right now, in this particular lecture we will see some basics about matrix equations and some basics about the solution of this matrix equation.

Based on this foundation , in the subsequent classes we will try to see parallelization of the matrix equations.

(Refer Slide Time: 06:03)



So, let us consider a one-dimensional steady heat conduction equation with constant conductivity. We will use a method called finite difference where a physical problem will be solved numerically and will get a matrix out of a rate equation or a differential equation.

The governing equation for one-dimensional steady heat conduction without any heat generation $\frac{d^2T}{dx^2}=0$. We have a 1 d rod which starts from x = 0 ends at x = 1 and we need to solve the governing equation here.

We understand that this is a very simple problem given the right boundary conditions. So, we really do not need any numerical method, we can forget about high performance computing. We can very well integrate this equation and say that T is a linear function of x.

However, I am trying to demonstrate a simple technique of getting a matrix equation out of this differential equation. So, this is a differential equation which is given to us. I am trying to demonstrate a simple technique by which we will get a matrix equation out of this.

So, this is a one-dimensional rod. So, x = 0 to 1 and we have to find out solution of $\frac{d^2T}{dx^2}=0$ here, with the boundary conditions aT (x) = 0 T = 0, at x = 1 T = 1. We know that T = x is the solution of the equation, but finding a solution is not the purpose at this stage, we have to see how this governing equation can be converted into a matrix equation.

So, this is a continuous description in space. Every point will have some value of T. However, once we try to use a numerical method and why do we try to use a numerical method, because we want our computer to solve it. Computers do not understand derivatives, computers do not understand continuity, computers understand about addition, subtraction, multiplication, division and other logical operations.

So, we need to express this problem into addition, subtraction type of problem and therefore, we need to find out discrete points over which discrete addition, subtraction equations are valid.

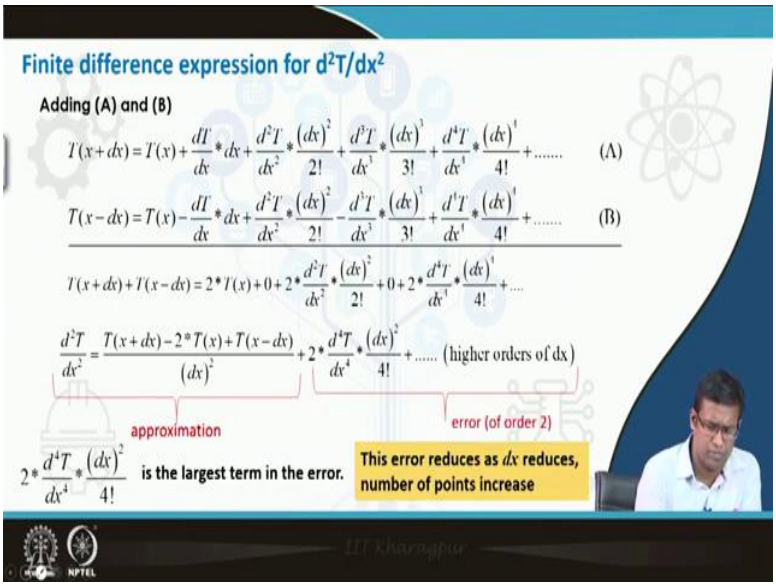
What will we do? We will use something called the finite difference method, where we will fix discrete points on the continuous rod, and we will try to seek a solution of the equation on these points only. If distance between the points; are constant we call this to be a uniform mesh. This is the mesh or grid system. If the distance is varying, we call it non-uniform. For simplicity, I am focusing only on uniform mesh problems. So, at any point (x) temperature is a function of x T(x).At the next point which is dx away from x temperature is T(x+dx) at the previous point which is dx in the negative side, dx away from x temperature is T(x – dx).

Now we use what we popularly known as Taylor series expansion. So, $T\ (x\ +\ dx) = T(x) + \frac{dT}{dx} * dx + \frac{d^2T}{dx^2} * \frac{(dx)^2}{2!} + \cdots$ and higher order terms.

This is the expression of T (x + dx). So, what we can see is that if we know T (x) we can get T(x + dx) provided the derivatives are also known to us and there is an infinite series. Now we write T (x – dx) similarly which is $T\ (x -\ dx) = T(x) - \frac{dT}{dx} * dx + \frac{d^2T}{dx^2} * \frac{(dx)^2}{2!} - \frac{d^3T}{dx^3} * \frac{(dx)^3}{3!} + \cdots$. Now, this rod has a size 0 to 1 therefore, dx is always less than 1. So, dx square is further smaller, dx whole cube is much smaller dx whole to the power 4 is also very small. So, if we can add these two expressions, this $\frac{dT}{dx}$ term will be cancelled out , $\frac{d^3T}{dx^3}$ term will be cancelled out.

We will be left with $\frac{d^2T}{dx^2}$ and the T (x + dx),T(x – dx), T(x) and something which is multiplied by dx to the power 4, then dx to the power 6 etcetera. These terms will be anyway small terms. So, what we will do we will add A and B and try to get an approximate expression of $\frac{d^2T}{dx^2}$.

(Refer Slide Time: 10:44)



We add A and B and get T (x + dx) + T (x - dx) = 2* T (x) +2* $\frac{d^2T}{dx^2} * \frac{(dx)^2}{2!}$ + 2*$\frac{d^4T}{dx^4} * \frac{(dx)^4}{4!}$ + ⋯ so on.

After these terms, all terms will be further smaller because dx to the power 6 is smaller than dx 4, dx to the power 8 is also smaller than dx whole to the power 4, so on. So, these are for smaller terms. This is probably the largest order of the terms remaining in the series.

So, we can rearrange it and write $\frac{d^2T}{dx^2}$= $\frac{T\,(x+dx) - 2\,T\,(x) + T\,(x-dx)}{(dx)^2}$ +2*$\frac{d^4T}{dx^4} * \frac{(dx)^4}{4!}$ +…..(higher order terms)

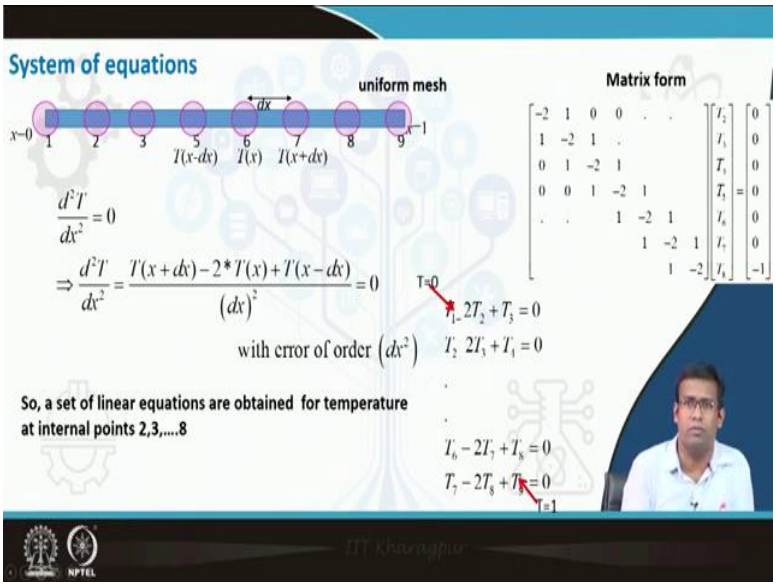
What is this? This is the difference of temperature at 3 neighboring points, some sort of addition subtraction , some support difference of temperatures divided by dx whole square. If we write that, we make an error, we neglect these terms and we say that we are doing an error which is of order 2; that means, the largest term of this error is dx whole square, there are some other smaller terms, but the largest term is dx whole square.

So, this is a second order accurate expression, the error is of the order 2 and this is the largest term in the error. So, if we substitute $\frac{d^2T}{dx^2}$ by this expression, the error which we are making will reduce, because dx is always less than 1. So, if dx is 0.1 the error is of the order of 0.1 square, 0.01 ,if dx is 0.01 the error is 10 to the power - 4. So, as the dx will reduce, this error

will reduce and how can dx reduce? if we go back to the previous slide yeah dx is the distance between 2 consecutive points.

If we have a greater number of points this distance will reduce. So, as we increase the number of points dx reduces, and we get more accurate expressions.

(Refer Slide Time: 13:39)



Well, so starting from a differential equation $\frac{d^2T}{dx^2} = 0$, we get a difference equation T (x + dx) - 2 T (x) + T (x - dx) = 0. So, this is a method in which a differential equation can be converted into a difference equation.

But this difference equation obtained for this point, it can be obtained for other points also. So, the equation for this point will involve the local temperature and the neighboring temperature therefore; we will get a system of coupled linear equations.

So, if we write down the equations, we will get point 2, we will get $T_1 - 2\,T_2 + T_3 = 0$, for point 3 we will get $T_2 - 2\,T_3 + T_4 = 0 = 0$ so on. Well, we are not finding equations for points 1 and 9 because they are boundary points and we will get boundary conditions for them.

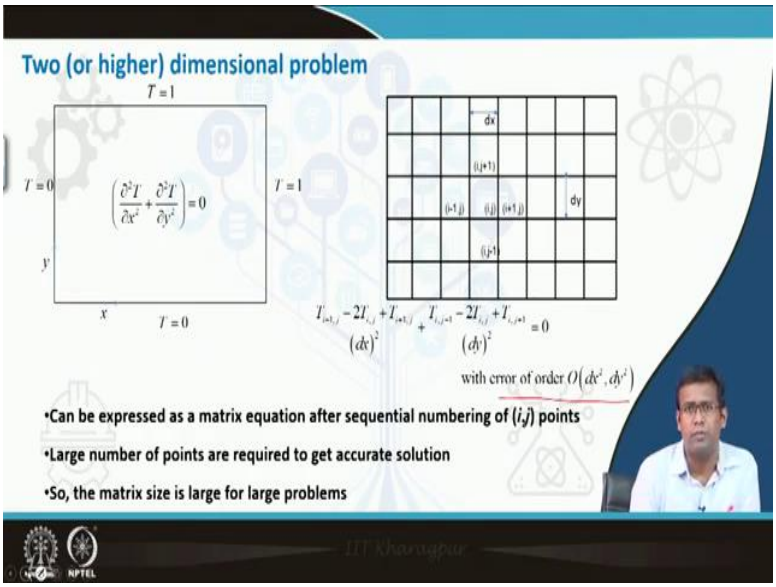
So, we get a set of linear equations for temperature at the internal points and the boundary points can be substituted as boundary equations. So, now, what do we have? We started with a differential equation; we ended with a system of linear equations and which can be represented as a matrix form, so you get a matrix equation out of it.

These are the nonzero numbers in the matrix .This is called a tridiagonal matrix and is a sparse matrix so whatever is not written in this matrix is number 0.

So, nevertheless starting from $\frac{d^2T}{dx^2} = 0$ we get AT = b ,a matrix equation that is why matrices are very important in scientific computing. In scientific computing you do computing to understand science, and science in many cases comes in terms of natural laws or some constitutive equations which are differential equations ;we can tell them as rate equations.

We can mention that this is the rate of certain things, rate of change of temperature, rate of change of velocity, something like that. Here it is the rate of heat moving through the unit area. These rate equations or differential equations can be converted into matrix equations. When do we do scientific computing? We have a differential equation we want to convert it into a system of linear equations and to solve it.Therefore, the next step which will be important for us in this discussion is to how to solve the system of equations.

(Refer Slide Time: 16:39)



Well, in some cases, like it may not be a simple 1-D problem. In fact, nobody will do scientific computing, forget about parallel computing for solving 1 D equation. They can be 3-dimensional geometry. There can be complex geometry. There can be many other complexities involved, for which we need to use this discretization approach, we need to get the difference equation.

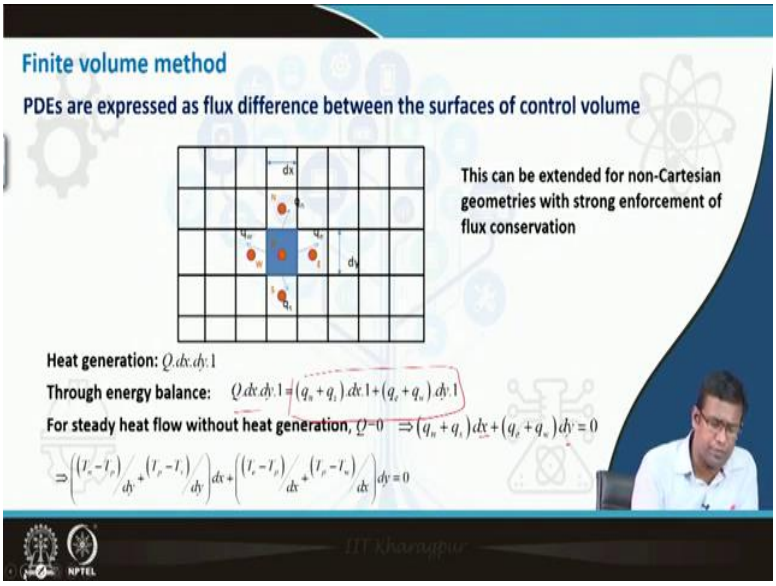
See if we have a 2-D problem, we have to use a 2-D mesh system, and the equation we define each point by their index i j, but for each point we will get a linear equation. So, for all the points we will get a set of linear equations and by efficient mapping of $T_{i,j}$ into a one-dimensional vector we can get a matrix equation out of this also.

So, in all cases, where we start with a differential equation we can end up with a system of linear equations or a matrix equation. We saw that the error in this matrix representation, is a function of $(dx)^2$ and $(dy)^2$. The distance between the points as it will reduce, we will get less error and we will get more accurate representation.

So, in order to get a solution which will resemble very well with the physical behavior we need to take many points, because the distance between the points will then reduce and therefore, as many points as we will take our matrix will be larger.

So, we will end up in a large matrix system if we need to get accurate solutions and therefore, the issue will not be only solving the matrix equation, but the issue will be solving a large matrix equation. So, the matrix size is large for large problems and we need to find out how to solve the large matrices.

(Refer Slide Time: 18:35)



Taylor series-based methods which are known as finite difference methods. There are methods like finite volumes where instead of representing the physical law as a partial differential equation, we express it as a flux difference between the surfaces of a control volume.

Say we have to similarly solve the conduction equation over this 2D geometry, we consider a control volume and write that the equation is nothing, but conservation of heat flux across different cells. So, we can assume that there is some heat generation per unit volume. So, this cell has a volume dxdy and the unit depth 1 total heat generation is Qdxdy and these are the fluxes coming out of this surface.

So, if we have heat generation Q that should be if the flow is steady, if that heat transfer problem is steady there is no change in temperature. Whatever is heat generated within the control volume is the same as the amount of heat coming out of the control volume. So, the net heat q_s q_e q_w q_n are the heat fluxes to each surface multiplied by the area will give us the net heat transfer across the surface.

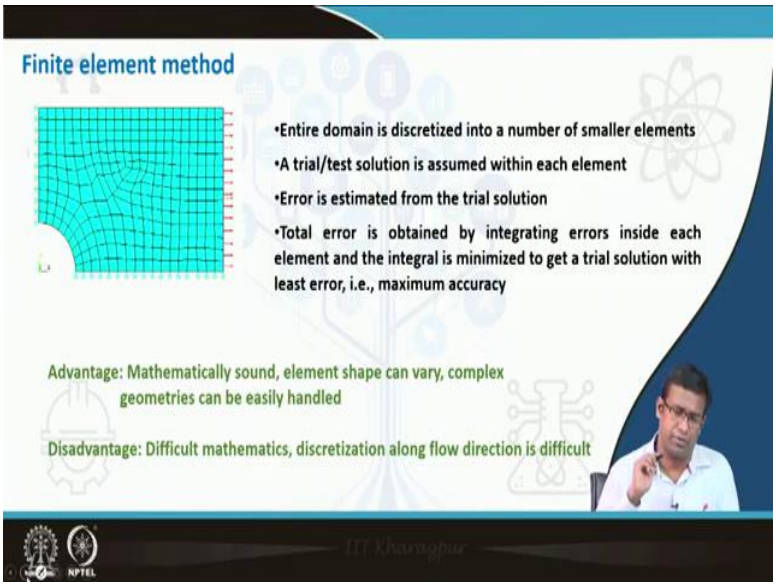
Net heat transfer; from the control volume that should be equal to the heat generation. In case steady flow without heat generation Q = 0. So, we get that summation of the fluxes multiplied by the respective area = 0, and now these fluxes can be obtained using Fourier's law of conductivity and we can replace the that by the difference of temperature, and we end up in getting an equation involving temperature of the central point and temperature of all the neighboring points.

So, you get a linear equation for one particular point. For the entire control volume ,for each centroid of the control volume we get one linear equation and therefore, we get a system of linear equations. So again, it can be shown that this is giving us a matrix equation.

The advantage of going through this particular technique is that if we have a non-Cartesian geometry, we can still get divided into multiple control volumes and write flux conservation equations and get a set of linear equations like that. The previous method or finite difference method is restricted for Cartesian or strictly polar type of geometry, the sides of the geometry must be aligned with the axis of the coordinate system.

But here we can go. We can calculate it for any complex geometry we have only to discretize the geometry into multiple control volumes and write the flux balance equation. This is heavily used in computational fluid dynamics this finite volume method. Nevertheless, we end up in getting a system of linear equations.

(Refer Slide Time: 21:42)



The other method is the finite element method, which is also applicable for complex geometries. In the volume method we saw some idea of extending the numerical method for non-Cartesian complex geometries. Finite element method is mathematically very sound and due to its strong basics, it can be extended to many complex geometries.

The idea of finite element method is that; discretize the domain into smaller elements. Assume a trial or test solution within each element. So, you do not know; what the solution of the differential equation, but assume some function phi which is the solution to the equation and assume a behavior of that function within each element. Now the assumed solution is not the actual solution, so, there will be some error. You substitute the solution into the governing equation, you will get some error. Calculate what is the error from the trial solution and then integrate the error over all the elements. Only for one element you are calculating the error here, but now calculate it for all the elements and sum them up and integrate it.Then, use something like a least square method or some optimization method; so that you can say that you put an objective function that the error should be least. In order to get the least error, you will get how your trial solution will be valid. So, total error is obtained using integrating errors inside each element and the integral is minimized to get a trial solution with least error and by that method you get the least error within and you get the best approximate solution.

This also end up in giving you a matrix equation the advantage is that this mathematically sound the element shape which you have considering here can vary ;in finite volume you have

to calculate flux, we have to see that the neighboring points are perpendicular to the to the inter control volume boundary, but here you do not need to look into that it is not based on flux calculation and complex geometry can be easily handled.

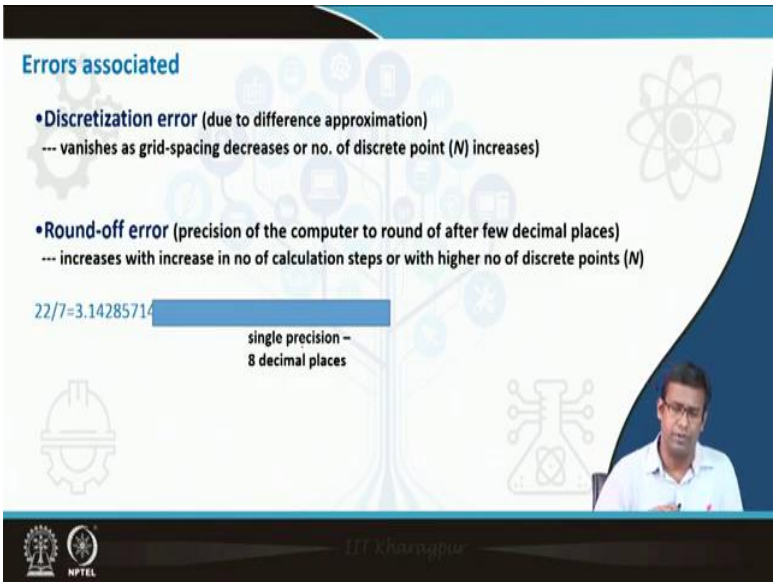
Disadvantage is mathematics is difficult, I understand that if you are a beginner finite difference you probably made some ideas. Finite volume you probably got some faint idea, but if you are a beginner within this one slide discussion you probably did not get could not make a good meaning out of what is a finite element method.

So, mathematics is difficult if I have to explain the mathematics that will be in the 20 lecture class itself. So, that is one disadvantage of finite elements, that the mathematics is difficult and therefore, the algorithms as well as the coding exercise is more difficult than the previous methods.

In certain cases, we need to know the flow direction. We need to do some mathematical modeling using discretization along the flow direction that is difficult here, but for the previous methods it is much simpler.

So, there are some complexities associated with this method; however, this also ends up in giving us a matrix equation. So, all the methods we are discussing will give us a matrix equation for a physical problem and that is one big group of scientific computing problems that if you start with a physical problem you get a matrix equation out of it, then you have to solve the matrix equation.

(Refer Slide Time: 25:28)



Well, one important factor which is associated with converting differential equations into difference equations or matrix equations is error. We have seen that in finite difference methods there are certain terms which we neglected. We told that this is an error and as the number of points will increase this error will reduce.

So, this is known as discretization error. The error between the differential equation and the difference equation is known as discretization error. A large part of the discretization error is due to the truncation or omitting the higher order terms in the Taylor series expansion.

Now, this error will reduce as the distance between the points will reduce, we have seen that this error is of the order of dx whole square dx whole 4 etcetera dx is less than 1. So as dx will reduce, this error will reduce. Therefore, as we increase the number of points the errors will reduce. There is another error which is called round off error which is due to the precision of the computer. Say you are calculating pi. If you see the actual value of pi it is a large number. It is an infinitely long number.

There are infinite decimal places in pi. It is because it is an irrational number. But you if you are using a single precision computer if the machine it can calculate or it can hold only up to 8 decimal places. So, pi will be well approximated as this number and if you do your calculation you will get nearly the right results , instead of pi if you write 3.1428514 even, we know that by just using 3.14 we get good results.

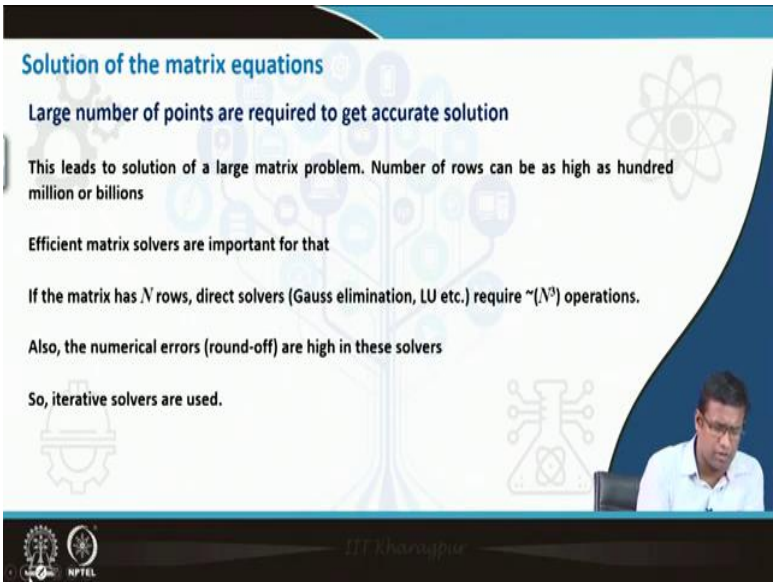
But we are making some errors, because we are not considering all the digits after the decimal place. We are drowning in numbers after certain decimal places and we are making some errors. In each calculation, wherever there is something like an irrational number or something which is not an integer or does not have finite decimal places, we are doing this error.

Now, as large as the problem is, as many points we are having in the geometry as the matrix size is large, many rows are there in the matrix, these errors will increase. Because for each point I am making an error as we have a greater number of points, we will make more errors.

So, this is another error which increases with increasing number of points. So, there is a tradeoff, we have to find out what is the optimum number of points in which we get least error and we try to use that. Fortunately, or unfortunately this number the optimum number of points is in practical problems is also a large number. So, we end up in getting a large matrix equation.

As this is a large number, we I mean we have to be very careful, so that the round off errors do not compile up and we get we kill our solution .Also as the equation system will be very large we have to use an efficient matrix solver also many cases we have to use parallel computing or high performance computing techniques.

(Refer Slide Time: 28:58)

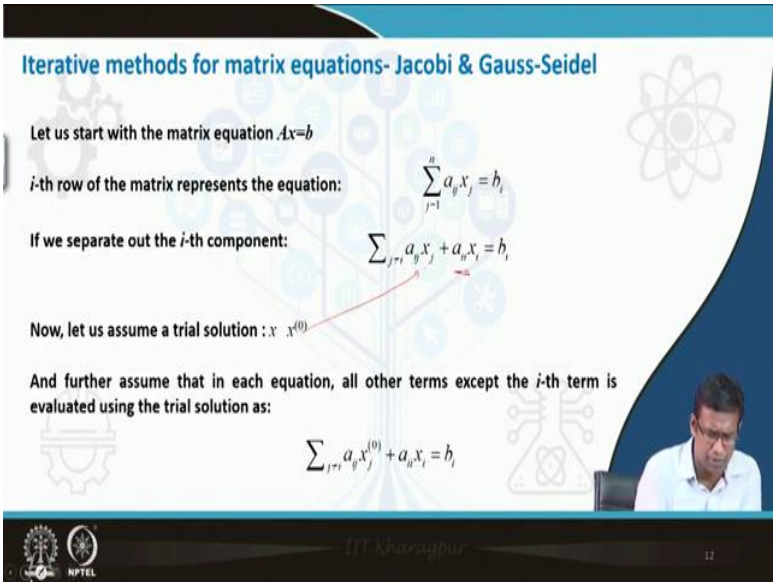


So, we will come to a solution of the matrix equation. Large numbers of points are required to get an accurate solution. This leads to a large matrix problem ;the number of rows can be as high as 100's of millions or billions in the matrix equation.

Now, we need efficient matrix solvers for that if the matrix has N rows the direct solvers like Gauss elimination or LU will require N cube operation. And if there are this is a million by million matrix; this N cube operation means 10 to the power 9 operations and in each operation if we make an error of 10 to the power – 8, the errors pile up of the order of 10; you will be calculating temperature in between 0 to 1 and you have a large error of the order of 10 which will kill the problem.

So, we understand that if we have to solve a practical problem, we have to use a large number of points, if we have a large number of points, using direct solvers we will have large errors. So, what we do instead , we use iterative solvers iterative, solvers mean we try with a trial solution to iterate for a better solution. Each time we had an older solution which is overwritten by a newer solution, so the round off errors are neglected in each iteration only we are having the final round off error. So, the round off errors are; usually less in the iterative solvers.

(Refer Slide Time: 30:28)

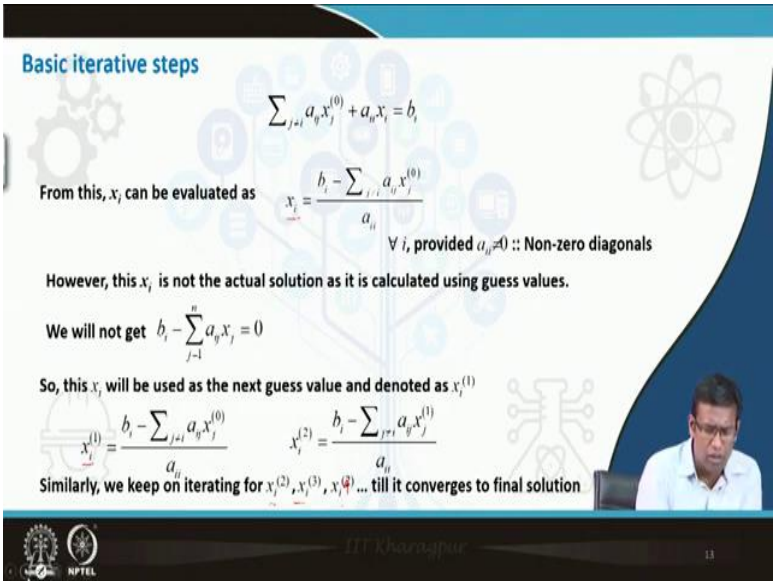


We will see some of the iterative solvers Jacobi and Gauss Seidel. Let us start with the matrix equation Ax = b, i-th row of the equation will be $\sum_{j=1}^{n} \quad a_{i,j}\, x_j = b_i$

If we separate out the i-th component, $\sum_{j=1\ j\neq i}^{n} \quad a_{i,j}\, x_j + a_{i,i}\, x_i = b_i$. Now, let us assume a trial solution x = $x^{(0)}$ and we further assume that in each equation all other terms except i-th term is evaluated using the trial solution.

So, what will I do? We will consider each equation in the matrix system and for each equation we will substitute the trial values in the off diagonal terms and the diagonal term or the pivotal term of the equation will be calculated based on these values. So, we find out that $a_{i,j}x_j$ with guess solution is $a_{i,i}x_i = b_i$.

(Refer Slide Time: 31:32)

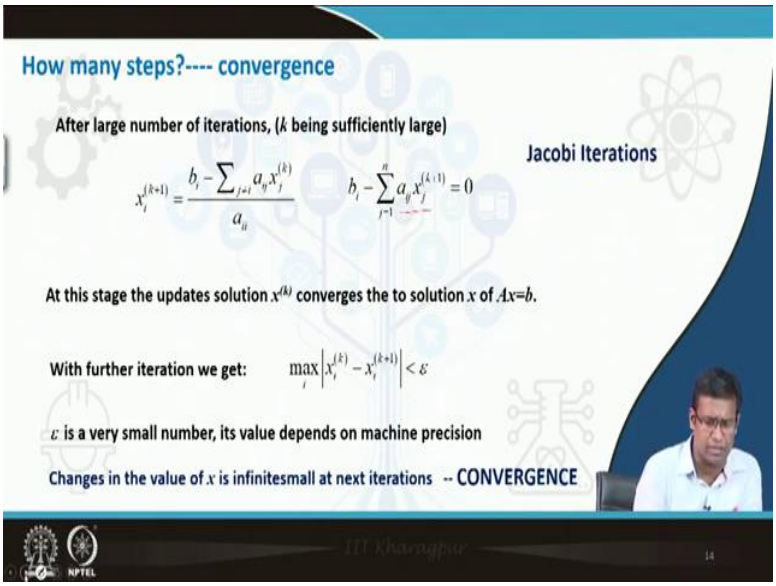


Using this we can get a value of x.

Now, we will do it for all the rows since from x_1 to x_n all the variables will be calculated provided a $_{i,i}$ is not 0 that is one important parameter here that the diagonal term of the matrix is non 0.However, this x $_i$ is not actual solution because these are guess values. So, this will not satisfy the actual equation. We will not get the actual equation to be satisfied for all the equations. So, what we guessed as this x $_i$, this is not the actual solution, but this will be my next case. I will do another iteration using x $_i$. So, all these x is calculated in through this exercise as my next case.

So,$x_i^{(1)}$ will be $\frac{b_i - \sum_{j\neq i} a_{i,j}x_j^{(0)}}{a_{i,i}}$ .We will calculate for ,$x_i^{(2)}, x_i^{(3)}, x_i^{(4)}$So , on till it converges to the first final solution.

(Refer Slide Time: 32:43)

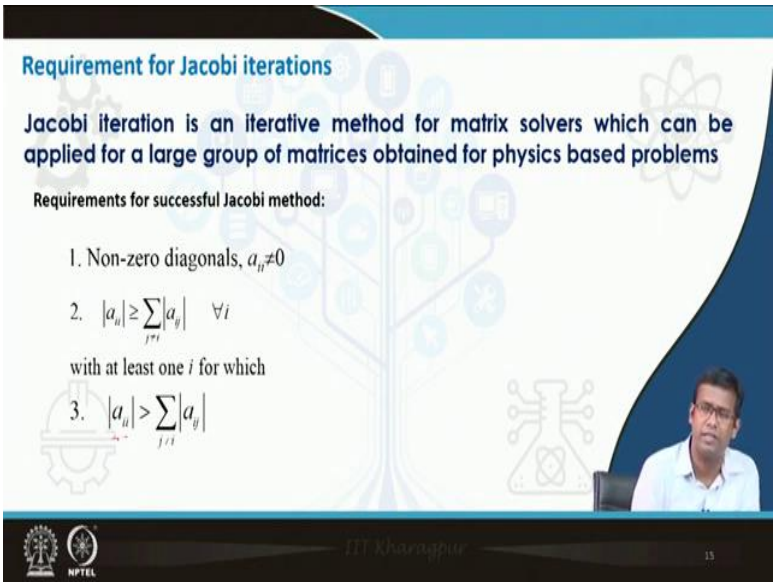


So, when we call that this is converged, when we will see that the guess solutions now are satisfying the equations and it is usually done after a large number of iterations.

So, k is the number of iterations, k being sufficiently large we calculate $x^{(k+1)}$ and we substitute it into the main equation and we will see that each row is giving me 0. There are round off errors, this is the convergence problem which has some asymptotic behavior, so it will not give me exactly 0, but it will give me a very small number, something like a machine precision number.

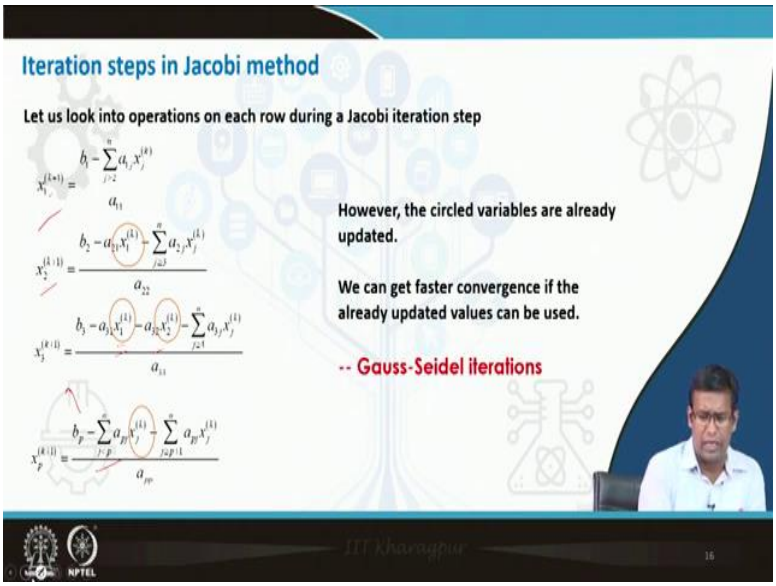
At this stage, we will see that $x^{(k)}$ has converged to the solution x and we will get that the maximum difference of $x^{(k)}$ and $x^{(k+1)}$ for any of the rows is less than epsilon where epsilon is a small number close to the machine precision. Changes in the value of x will be infinitely small if we continue the iteration and this is what we know as convergence.

(Refer Slide Time: 33:46)



Now, this is known as Jacobi iterations. Jacobi iteration is an iterative method which is applicable for a large number of matrices which comes from physics-based problems like finite difference or finite volume or finite element method. The requirement for the Jacobi method is that the diagonal should be nonzero , sum of absolute value of the diagonal term is greater than equal to sum of absolute value of off diagonals for each row. There should be at least one row where the absolute value of the diagonal term is greater than sum of absolute values of the off diagonals and this is usually satisfied for most of the discretized forms or difference forms of the governing equations using finite difference or finite element method or finite volume method for physical problems.

(Refer Slide Time: 34:38)

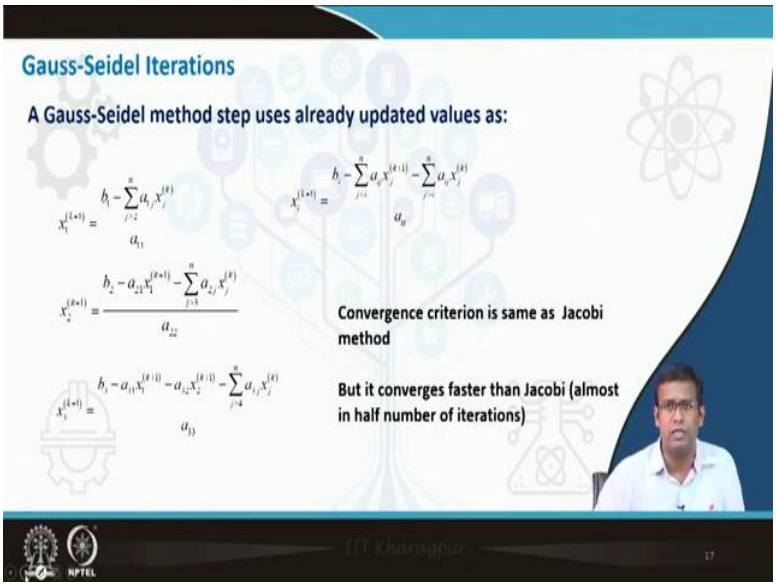


So, we look into each of these rows and we can find out that this is a general iteration loop in the Jacobi method, that while calculating x_1 we are using guess values of $x_j^{(k)}$. While calculating x_2 we are using guess values of x_1 and guess values of all other terms greater than 2, x_3 x_4 and so on.

But x_1 has been already calculated there. While calculating x_3 we are using $x_1^{(k)}$ and $x_2^{(k)}$ which is already updated, because we are trying to converge it, if instead of using the older values we use the already updated values we get a faster convergence .

So, it is observed that while solving for this row already a number of x s are available up till the previous one. These circle variables are already updated, so we can get faster convergence if we use the already updated values, and this is known as Gauss Seidel iteration.

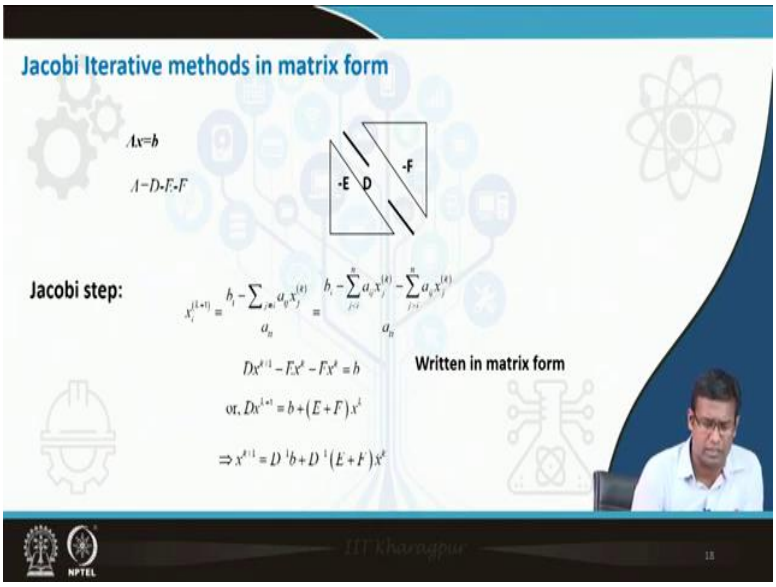
(Refer Slide Time: 35:54)



Instead of using the previous guess values you use the already available updated values, and it is shown that we use the same convergence criteria as Jacobi, but the convergence is faster than Jacobi, the convergence is done in almost half the number of iterations.

So, if Jacobi takes 100 iterations Gauss Seidel will take 50 iterations, this is a better matrix solver. We are trying to get a more efficient matrix solver, because we try to cut down the number of iterations ,and Gauss Seidel can be a way out for that.

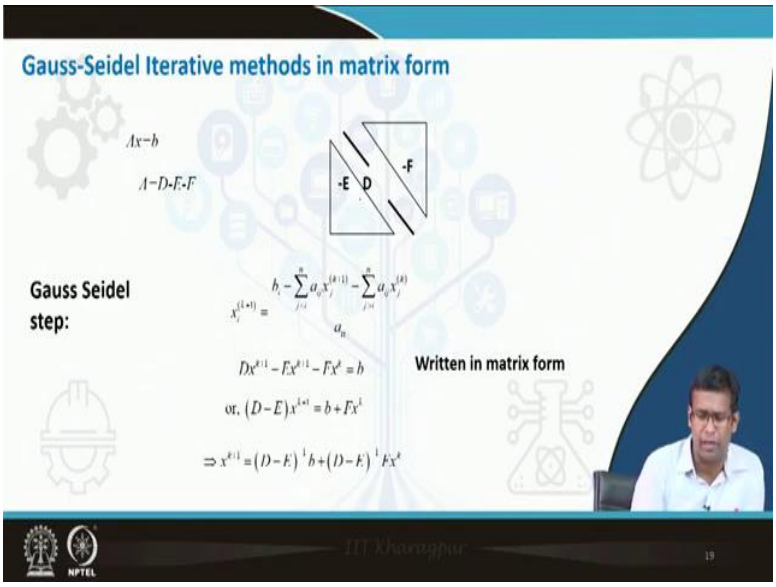
(Refer Slide Time: 36:32)



So, again we can see that the Jacobi iteration method can be represented here that the diagonal term will be multiplied with the updated values and the off diagonals are multiplied with the guess values.

So, this can be written in the matrix form that, $x^{k+1} = D^{-1}b + D^{-1}(E + F)x^{k}$.

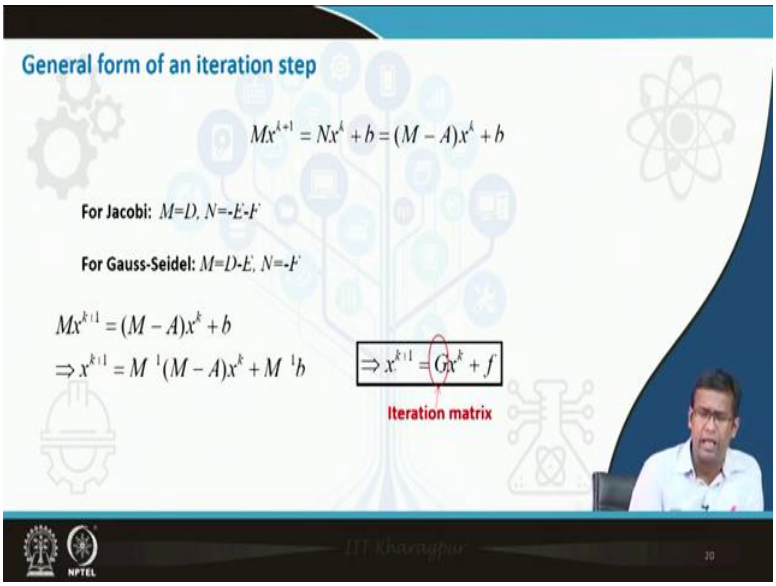
(Refer Slide Time: 37:04)



Similarly, we can write a matrix expression for the Seidel also, $x^{k+1} = (D - E)^{-1}b + (D - E)^{-1}Fx^{k}$.

So, whatever we are doing in Jacobi and Gauss Seidel they are basically matrix operations. We are taking a guess matrix guess vector and multiplying with something from the splitting of the original matrix A, which we are solving and getting the updated values.

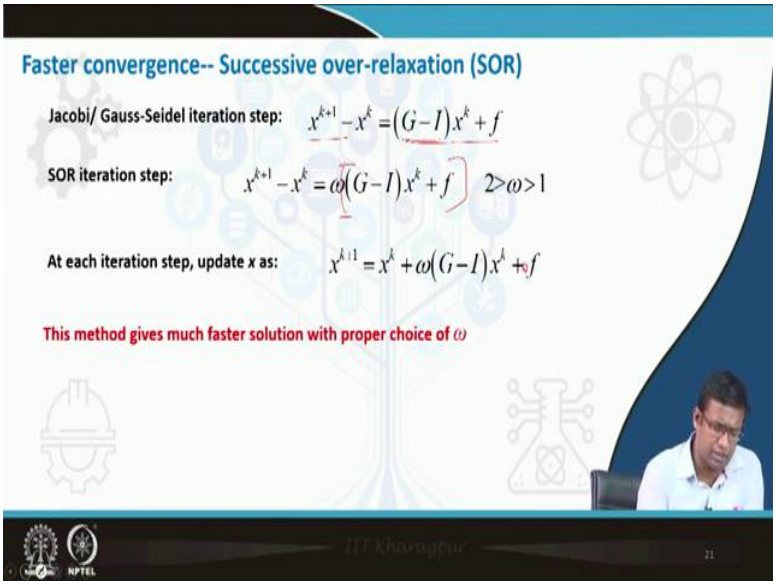
(Refer Slide Time: 37:40)



So, this is like M $x^{k+1} = Nx^k + b$,and we can write that for Jacobi and Gauss Seidel M and N has different meaning $x^{k+1} = M^{-1}(M - A)x^k + M^{-1}b$ or $x^{k+1} = Gx^k + f$, G is called the iteration matrix.

So, you have a guess solution vector, you multiply it with an iteration matrix. What is an iteration matrix? Iteration matrix comes simply from the splitting of the main matrix and add with an f which again comes from the right-hand side vector and some splitting of the main matrix A and get the updated matrix.

(Refer Slide Time: 38:29)



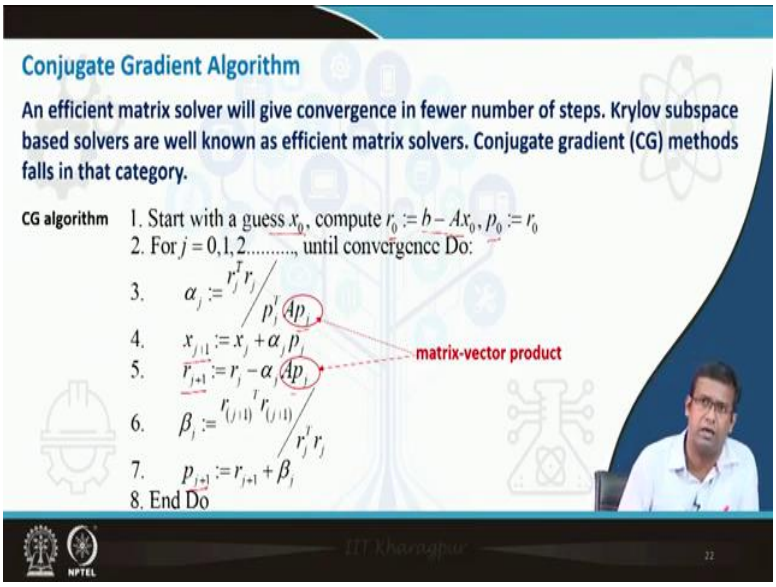
So, using this concept, we can get a faster solution. We can see that in each iteration we are improving our solution by this amount $x^{k+1} - x^k$ which is G - I the iteration matrix minus identity into the older vector plus x^k, but we are improving the solution in each iteration.

So, in case we improve, we increase this amount, we get that this is increment in each iteration, but we will multiply it with some factor greater than 1 so the increments increase, so that we can reduce the number of steps. This is called successive over relaxation.

That we write $x^{k+1} - x^k = \omega\left((G - I)x^k + f\right)$ where $1 < \omega < 2$. $\omega < 2$ for stability $\omega > 1$ for faster convergence. This is known as the successive over relaxation step.

These methods give a much faster solution with the proper choice of omega, so this is a more efficient method of solving the equations.

(Refer Slide Time: 39:44)



These are basic iterative techniques which you have discussed here; they are very simple, but they are little more involved iterative techniques based on Krylov subspace methods or projection methods which can give us a much faster solution. We need faster solutions because we need a smaller number of iterations in which we can solve the equation system.

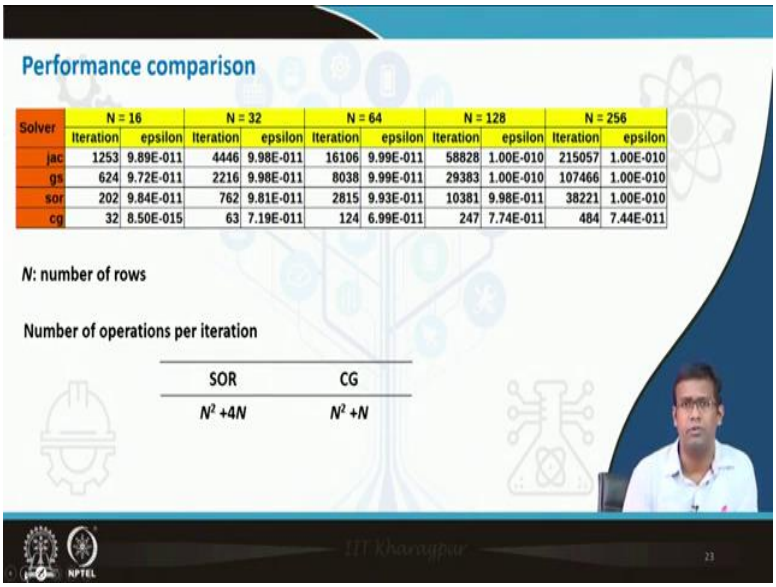
One of them is the conjugate gradient method, we can see the conjugate gradient method, here it also starts with a guess solution and calculates a guess residual. What is the error based on the guess solution b - Ax_0 if x_0 satisfies the equation this will be 0, but as it's a guess solution

it will probably not satisfy the equation. So, there will be some error and sets an auxiliary vector and then follows a certain procedure by which it can update the guess solution, it can update the error and it can update the auxiliary vector. Over the iterations there is an instance in which you see that x^{j+1} has converged to the final solution. This is the very first solution. I will show you some test results that in a much smaller number of iterations you get the solutions. But interestingly, this method requires a matrix vector product. If we think of the matrix operations which I have shown in the last few slides of Gauss Seidel and Jacobi there are also matrix vector products involved.

So, all the matrix solvers, especially the iterative matrix solvers, require matrix vector products, and if it is an N by N matrix and the vector is N by 1 the matrix vector product will take N square operations. This is a large number of operations and it is important to parallelize this part if you are thinking about using HPC techniques. If you remember in openMP discussions we looked into parallelization of matrix vector products.

This is important because these are usually the most time consuming and most costly steps in the matrix solvers or in the scientific computing algorithms. So, it becomes important to parallelize. We will see some of the techniques in the subsequent classes.

(Refer Slide Time: 41:53)



Well, I just give you an update that if we have a 16 by 16 matrix Jacobi took 1,253 iterations ,Gauss Seidel took almost half 620 iterations, SOR took 200 iterations and conjugate gradient took 32 iterations.

If I have a large matrix 256 by 256, this is large, but this is not as large as the largest matrices we usually deal with practical problems. However, Jacobi took 2,15,000 iterations, Gauss Seidel took almost half of that and conjugated gradient took only 484 iterations.

So, conjugate gradient is a very fast way of solution. SOR is in between conjugate gradient and Jacobi .In conjugate the number of iterations is of the order of the number of rows of the matrix. That is one important observation. Now you have operations per iteration. Within each of iteration you have to find out matrix vector, products etcetera, so, you have to do many operations

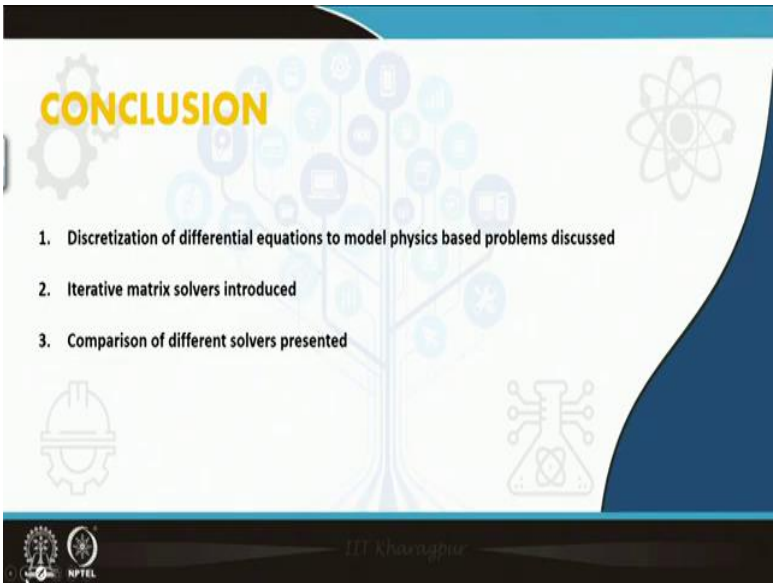
In SOR this operation is $N^2 + 4 N$, in conjugate gradient this operation is N^2+N. So, you do order of N of iterations and N square operations in each iteration you end up in having order of N cube operations.

So, the number of operations in any matrix solver is very high and therefore, while doing HPC ,any scientific computing problem which involves matrix solver large class of scientific computing problem involve matrix solver; it is extremely important that we look into the matrix solver and parallelize the matrix solver part and that is what we will try to do in the subsequent classes.

(Refer Slide Time: 43:30)

For the reference you can look into Yousef Saad, book iterative methods for sparse linear systems from SIAM and I have another NPTEL course called matrix solvers you can access that and can find more details about the matrix solvers.

(Refer Slide Time: 43:45)



So, come to the conclusion, discretization of differential equations for physics-based problems are discussed, some iterative solvers are introduced and we have compared different solvers based on their convergence and the number of operations required for them.