

**Finite Element Method & Computational Structural Dynamics**  
**Prof. Manish Shrikhande**  
**Department of Earthquake Engineering**  
**Indian Institute of Technology, Roorkee**

**Lecture - 45**  
**Solution of Linear Simultaneous Equations - III**

Hello friends. So, we have seen the Solution of Simultaneous Equations by using direct solvers. We looked at the naive school book approach of Gaussian elimination for elimination of variables, sequential elimination of variables from each equation and followed by backward substitution to solve the unknown solve for the unknowns.

The other one we discussed about the factorization of the matrix coefficient matrix into LU factors lower triangular and upper triangular and in case of special case of structural mechanics or finite element general finite element problems, most of the case times we are dealing with positive definite and symmetric matrices. So, in that case both the factors turn out to be transpose of each other, they can be made to be transpose of each other.

So, eventually that LU decomposition reduces to a simpler Cholesky decomposition in the form of  $L L^T$ . And the coefficients of this factors of the matrix can be obtained from one to one correspondence and those there is a straight forward algorithm which allows us to compute the factors of the matrix and once these factors are available, then the system of equations can be solved by a two stage process and that results in the solution of unknown variables of the system of equations.

So, this is in general very straightforward procedure, the only problem with respect to finite element method use of direct solvers in finite element method is that the factors that we have the either Gaussian elimination or even the LU decomposition or  $LL^T$  transpose decomposition Cholesky decomposition, the factors may actually generate terms which were zero earlier.

Because the coefficient matrix in finite element analysis that we generate that we get from finite element analysis is a strongly banded matrix and a sparse matrix, lots of positions are zeros I mean there is not much there is no information there it is all zeros.

So, the point is those zeros if we have to store, then we have to allocate that floating point storage for assigning that zero which does not actually contribute to computation much or that does not lead to the characterization of the equations governing equations, but during the process of solution by this Gaussian elimination or by Cholesky decomposition, those zeros might be filled in with non zero entities during the process of elimination.

So, the fallout is, we cannot take advantage of neglecting zero terms or not assigning storage space for zero terms, we have to assign the storage for at least the triangular half. If it is an symmetric matrix coefficient matrix is symmetric then at least one half has to be stored including all zeros and that is all though one half I mean below diagonal can be eliminated I mean that storage can be avoided because it can work from the principles of symmetry symmetric structure, but still there might be still a large number of zeros which might have to be provided or which might have to be accounted for and that leads to a huge storage requirement.

So, iterative solvers which actually require only a series of matrix and vector multiplication. Essentially, a series of matrix vector multiplication the coefficient matrix itself is not modified at all. So, in that situation in this particular case it becomes very advantageous proposition that we store only non zero elements of the matrix. So, a large matrix with I mean  $n$  by  $n$ ,  $n$  can be a very large number I mean number of degrees of freedom can be can run into easily run into millions for a typical finite element run.

So, for a such a large matrix instead of storing most of them as zeros, we can just store a few non zero elements and with a mechanism with little book keeping to keep track of which row and which column what is the address of each of these non zero elements and then we can design an algorithm for matrix vector multiplication and that is all that is required for iterative schemes. And that leads to considerable saving in storage space and that allows the computations to be performed in using high speed memory of the computer.

So, iterative schemes are actually very much preferred in a finite element solution or even for computational fluid dynamics approach, I mean all those dealing with very large system of equations and very fast and very efficient solvers iterative solvers are now available which can actually give a good competition to the direct solvers in terms of

accuracy and efficiency and they are and with the added advantage of saving the storage space, they are ah gaining quite a lot of acceptance in scientific computing.

(Refer Slide Time: 07:15)

The slide is titled "Solution of linear simultaneous equations" and "Iterative solvers". It contains three bullet points:

- ▶ The coefficient matrix  $A$ , for large-scale systems, is often sparse and banded with very few non-zero elements
- ▶ Efficient data structures store only non-zero elements to facilitate faster processing using high-speed memory
- ▶ Problem of **fill-ins**: appearance of non-zero elements in places with zero entries during the process of Gaussian elimination or LU factorization.

The slide footer includes the name "Manish Shrikhande", email "manish.shrikhande@eeq.iitr.ac.in", and the department "Department of Earthquake Engineering, Indian Institute of Technology Roorkee". It also mentions "Finite Element Method and Computational Structural Dynamics".

So, essentially iterative solvers, as I mentioned here coefficient matrix for large scale systems is often sparse and banded with very few non zero elements. So, efficient data structures we can design efficient data structures to store only non zero elements and that will allow us to faster processing using high speed memory and the problem of fill- ins that is encountered during direct solver by using Gaussian elimination or LU factorization is a troublesome and that may not be encountered in the process of iterative solvers.

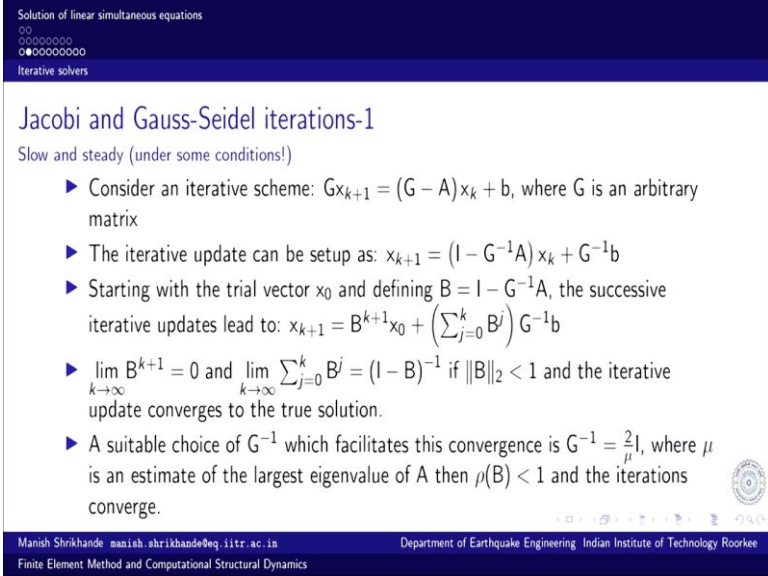
So, that is what we arrive at possible solution to avoid this problem of fill in is a way to arrive at the solution of equations by using only matrix vector multiplications. So, we do only simple basic linear algebra system operation matrix and vector multiplication series of them one after the other. And that leads to what we call as iterative solver we keep on improving upon the initial guess; initial guess of the solution.

We start with trial guess of the solution it can be anything, it can be a very bad guess whatsoever and we continuously improve upon the guess and we can nudge the solution towards the true solution as accurately as possible within the machine tolerance floating point arithmetic tolerance.

So, typical examples I mean these are get Jacobi and Gauss Seidel iterations, these were the very early propositions for iterative solvers and these are still taught in classroom example for iterative solvers although these are only good for that purpose. Nobody uses them for serious computation because the rate of convergence is too slow there are way better algorithms available for iterative solvers, but still they provide us very good vehicle to understand the basic issues involved in iterative solvers and they are very easy to implement as well.

Then next in degree of sophistication is what we call as steepest descent algorithm. This is derived from the optimization technique I mean the minimization problem. So, this is one of the algorithms for very quickly achieving the minimization of a function. So, one of those gradient based approaches. So, steepest descent algorithm and then we come to a very powerful technique which has its own several variants, but we will study only the simple standard format which is applicable for symmetric matrices conjugate gradient algorithm.

(Refer Slide Time: 10:36)



**Solution of linear simultaneous equations**  
 Iterative solvers

### Jacobi and Gauss-Seidel iterations-1

Slow and steady (under some conditions!)

- ▶ Consider an iterative scheme:  $Gx_{k+1} = (G - A)x_k + b$ , where  $G$  is an arbitrary matrix
- ▶ The iterative update can be setup as:  $x_{k+1} = (I - G^{-1}A)x_k + G^{-1}b$
- ▶ Starting with the trial vector  $x_0$  and defining  $B = I - G^{-1}A$ , the successive iterative updates lead to:  $x_{k+1} = B^{k+1}x_0 + \left(\sum_{j=0}^k B^j\right) G^{-1}b$
- ▶  $\lim_{k \rightarrow \infty} B^{k+1} = 0$  and  $\lim_{k \rightarrow \infty} \sum_{j=0}^k B^j = (I - B)^{-1}$  if  $\|B\|_2 < 1$  and the iterative update converges to the true solution.
- ▶ A suitable choice of  $G^{-1}$  which facilitates this convergence is  $G^{-1} = \frac{2}{\mu}I$ , where  $\mu$  is an estimate of the largest eigenvalue of  $A$  then  $\rho(B) < 1$  and the iterations converge.

Manish Shrikhande manish.shrikhande@eeq.iitr.ac.in  
 Finite Element Method and Computational Structural Dynamics  
 Department of Earthquake Engineering Indian Institute of Technology Roorkee

So, let us start with Jacobi and Gauss Seidel iterations. So, we can as I said these are only good for classroom instruction purposes because they are very slow to converge. So, the idea is to replace to consider the iterative scheme  $G$  is a matrix some matrix multiplied by  $X_k + 1^{th}$  that is the improved version improved estimate of the unknown and that is given by  $G - A$  multiplied by  $X_k$ . So, that is the previous estimate of the solution or if we

are starting then it will be the initial estimate of the solution anything any guess estimate and  $b$  is of course, the vector right hand side vector.

So, if you look at it this is a modification of original system of equations  $Ax = b$ . So, we take this simple distinction that  $X$  iterative scheme. So,  $X_k$  is being modified to  $X_{k+1}$  and using this matrix  $G$  suitable matrix. So, what is this matrix  $G$ ? So, matrix  $G$  I mean we can always refer to this as if I consider this simple mathematical operation. So,  $k+1^{\text{th}}$  iteration of the unknown vector is given in the slide.

So, again as I said this is only for mathematical notation we are not inverting matrix  $G$  here. So, if we start with some trial vector  $X_0$  and we define this matrix  $I - G^{-1}A$  as matrix  $B$ , then the successive iterative steps lead to let us go to starting from  $X_0$ .  $X_1 = B$  times  $X_0 + G^{-1}b$ . So, this will lead to if we keep on doing it again and again and substituting recursively  $X_2$  will be in terms of  $X_1$  and so, on.

So, this eventually leads to  $k+1^{\text{th}} = B^{k+1}$ . So, that is matrix  $B^{k+1}$  matrix  $B$  keeps on multiplying  $k+1^{\text{th}}$  times and this is again going to be because this is going to be recursively added up. So, this is again going to be a series of powers of  $B$ .

So,  $B$  raised to the power  $0 + B + B^2 + B^3$  and so, on until the previous iterate multiplied by  $G$  inverse  $b$ . So, if matrix  $B$  is somehow size of the matrix  $B$  if it is less than 1, then a large power of  $B$  would approach a null matrix because I mean anything less than 1 if we raise it to sufficiently large positive power that will diminish in size and eventually it will approach 0.

And this series solution I mean one identity matrix  $+ B + B^2 + B^3$  and so, on. So, this actually converges to  $I$  minus  $B$  inverse. If the size of the  $B$  matrix the norm of the  $B$  matrix that is second norm. So, that is the eigen value largest eigen value of  $B$  matrix is less than 1. And then if this happens then what does it say? So, this term vanishes because  $B$  raised to the power  $k+1$  this approaches a null matrix.

So, null matrix multiplied by  $X_0$ . So, that is going to be a null vector and this reduces to  $I - B^{-1}$ .  $I$  minus  $B$  inverse. So, what is that?  $I - B^{-1}$  inverse is going to be what is  $I - B$ ?  $I - B$  is going to be  $G^{-1}A$ .  $I - B$  is going to be  $G^{-1}A$  and inverse of that is going to be  $A^{-1}G$ . So, what we have here is  $A^{-1}G$  multiplied by  $G^{-1}b$ .

So, eventually what we will have is  $A^{-1}b$  and that is the solution true solution that we were looking at. So, this is how it converges to the true solution. So, what should be the choice of matrix  $G$  which facilitates this convergence to the true solution and that choice is  $G^{-1}$  should be equal to  $2/\mu$  times identity matrix where  $\mu$  is the estimate of largest eigen value of  $A$ .

So, if that condition is satisfied if  $G^{-1}$  is considered to be this or for that matter if I can put it other way around. If  $G$  is equal to  $\mu/2$  times identity matrix, then this convergence criteria would hold or this procedure would converge and then the spectral radius or the norm spectral norm of matrix  $B$  would be less than 1 and the iterations would converge.

Now, the whole problem is now how do we estimate this largest eigen value of matrix  $A$ ? It is not a very difficult proposition we will discuss in our after few lectures this eigen value problem. So, we will get to know how to estimate the largest eigen value and please note that we do not need the exact eigen value, we just need the estimate of largest eigen value.

So, it may be an upper bound estimate and we can find upper bound estimate just by observation by just looking at the rows of matrix  $A$  we can find out the upper bound estimate of the eigen values for matrix  $A$  that is by using the Gershgorin theorem. So, more on that later when we discuss eigen value problems.

So, the point is knowing a matrix  $A$  it is possible to identify a suitable matrix  $G$  which when used in this particular way, this leads to true solution in a recurrence form. So, this actually  $G^{-1}$  we choose this because it is easy to invert I mean this is essentially a diagonal matrix. So, this product  $G^{-1}A$  is very easy to compute and  $G^{-1}b$  is also very easy to compute right.

So, no further no inversion explicit inversion is required it is already a diagonal matrix. So, that is how the calculations proceed and the iterative solutions they gradually move towards the true solution. So, for Gauss Seidel iterations we take advantage. So, this is what the what we discussed here.

This is the Jacobi scheme where the entire vector is updated in one go we can slightly improve upon that by considering making use of whatever I mean term by term if we

improve then whatever  $k + 1^{\text{th}}$  iterate we have already estimated some elements of  $X$ , we can take advantage of that in this iterative scheme and speed up the convergence a little bit and that is what is referred to as Gauss Seidel iterations.

(Refer Slide Time: 19:47)

The slide is titled "Jacobi and Gauss-Seidel iterations-2" and is part of a presentation on "Solution of linear simultaneous equations" and "Iterative solvers". It contains the following bullet points:

- ▶ For Gauss-Seidel iterations, we take advantage of the currently available updates of unknowns in current iterations.
- ▶ The iterations are stopped when a suitable norm of residual vector  $r_k = b - Ax_k$  is less than the acceptable tolerance.
- ▶ A small residual implies a small error in solution only if the coefficient number  $\kappa(A) (= \|A\| \cdot \|A^{-1}\|)$  is small.
- ▶ Convergence is very slow even with the use of over-relaxation.
- ▶ Successive over-relaxation technique:  $x_{k+1} = \omega \hat{x}_{k+1} + (1 - \omega)x_k$ , where  $\hat{x}_{k+1}$  is the Jacobi or Gauss-Seidel estimate and  $\omega$  is the relaxation parameter.
- ▶ For symmetric, positive definite systems,  $\omega \in (0, 2)$  ensures convergence.

The slide footer includes the name "Manish Shrikhande", email "manish.shrikhande@eeq.iitr.ac.in", and affiliation "Department of Earthquake Engineering Indian Institute of Technology Roorkee, Finite Element Method and Computational Structural Dynamics".

So, for Gauss Seidel iterations we take advantage of the currently available updates of unknowns in current iterations and that is very easily done by again rearranging the operations computations within the loop of computation and that is easily arranged and the iterations are stopped when a suitable norm of residual vector.

So, this is the error in solution  $b - A$  times  $X_k$  the current iterate if this residual. So, if this is not equal then please look at the let us refer back to our original proposition original concept of definition of solution of simultaneous equation as the right hand side being resultant of the vector sum of column spaces.

So, this is column space sum of column space and  $b$  should be the resultant of this column sum linear combination of these columns of matrix  $A$  and this is the error in that computation.

So, if it is true solution then obviously, residual vector would be null vector, but it is quite possible or it is almost always there that for engineering analysis, we may not be interested in having a very precise solution to the last point or 14th place of decimal. Generally we are ok we are happy enough with the solution if it is accurate up to two or

three places of decimal or whatever may be the engineering tolerance; tolerance for computation.

So, when we do that; that is there will be some error of approximation and we compute this size of error what is the norm, what is the magnitude of this error and if that is less than a suitable acceptable tolerance level then the iterations are stopped or it continues in the recursive manner. A small residual implies a small error in solution only if the condition number of the matrix that is  $\kappa$  of matrix  $A$  is small.

So, now this is condition number which measures whether the system of equations  $AX = b$  is ill conditioned or not. So, that depends on the condition number of matrix  $A$  and essentially this condition number measures how far the column spaces are independent or the columns of matrix  $A$  they provide independent basis to represent the right hand side vector.

So, if the condition number is small, then this residual small residual that we have computed. If the residual error is very small, then that also implies that this solution computed solution is also very close to true solution. If the condition number is very large then a small residual need not mean or need not imply closeness of the computed solution with true solution. A reason in visual example I mean you can see if the two examples two lines are almost parallel to each other they intersect over a large area.

So, the solution can lie anywhere in this point in this position. So, there is a large amount large possibility which can qualify as a solution. So, the error would be small over a large range and any point in that range could qualify as a small error in this equilibrium or satisfying the equation, but the computed solution may be not very close to the true solution and that correspondence I mean small residual corresponding to true solution close to true solution is like this.

So, two lines very sharp intersection. So, then if there is only one solution one point within even within the limits of floating point round off error etcetera. So, vagaries of taking care of all vagaries of floating point arithmetic even then the computed solution can be very accurate and that is why this condition number is a very important criteria for specially for the iterative solutions or iterative solvers.



For direct solvers condition number does affect the computation and round off error etcetera and that will have its own effect on the computed solution. Computed solution may not be very close to true solution, but there is no mechanism to improve upon or even test whether the computed solution is good enough or not.

Iterative schemes at least you have one has that possibility to keep on iterating keep on refining the solution and see if we can get anything better. So, Jacobi and Gauss Seidel iterations the convergence is very slow even with the use of one of the standard techniques in numerical methods over relaxation. So, we slacken the system of equations and that may accelerate the convergence to some extent.

So, successive over relaxation technique. So,  $X_{k+1}$  is given by  $\omega$  times  $\hat{x}_{k+1} + 1 - \omega X_k$  where  $\hat{x}_{k+1}$  is the Jacobi or Gauss Seidel estimate. So, it is a linear combination of what we compute using Jacobi Gauss Seidel estimate and some weightage is given to the previous iterate and that weighted sum is taken as the new solution and that is put in the recurrence relation for next iteration and this generally accelerates the process.

The relaxation parameter  $\omega$  has to be a value between 0 to 2 and that will ensure convergence for positive definite systems and symmetric systems and for the finite element structural mechanics, solid mechanics problems, we are dealing with symmetric positive definite systems. So, this is not an issue at all.

So, choice of  $\omega$  between some number between 0 to 2 would ensure convergence and little faster convergence than what we normally achieve by using standard Jacobi and Gauss Seidel iterations. So, this is the basic introduction to iterative solvers, in our next lecture we will discuss the more powerful techniques that is steepest descent gradient based approach and of course, the workhorse that is almost a standard issue in all iterative schemes the conjugate gradient method very powerful and very fast iterative scheme.

Thank you.