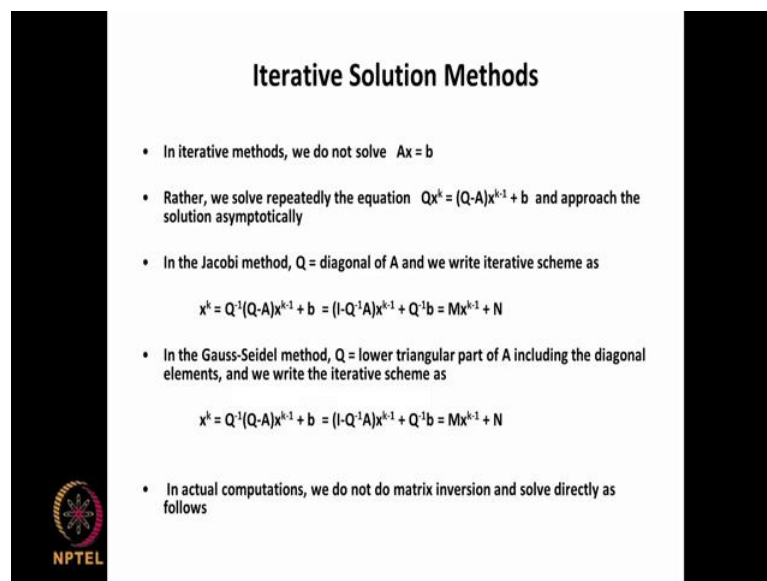


Computational Fluid Dynamics
Prof. Sreenivas Jayanti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 10
Lecture – 49
Convergence analysis of basic iterative methods


Today we are going to start the second part of module 5, which is on the subject of solution of linear algebraic equations that using advanced methods. We have seen a briefly recap that we have a standard is scalar transport equation, and if we discretize it, we finally, get a set of linear algebraic equations or linearized algebraic equations. And it is a solution of this that we seek because the solution of this will give us the values for the variable, in this case five at the grid point i comma j .

(Refer Slide Time: 00:58)



Iterative Solution Methods

- In iterative methods, we do not solve $Ax = b$
- Rather, we solve repeatedly the equation $Qx^k = (Q-A)x^{k-1} + b$ and approach the solution asymptotically
- In the Jacobi method, $Q =$ diagonal of A and we write iterative scheme as
$$x^k = Q^{-1}(Q-A)x^{k-1} + b = (I-Q^{-1}A)x^{k-1} + Q^{-1}b = Mx^{k-1} + N$$
- In the Gauss-Seidel method, $Q =$ lower triangular part of A including the diagonal elements, and we write the iterative scheme as
$$x^k = Q^{-1}(Q-A)x^{k-1} + b = (I-Q^{-1}A)x^{k-1} + Q^{-1}b = Mx^{k-1} + N$$
- In actual computations, we do not do matrix inversion and solve directly as follows



So, we have seen that the solution of $Ax = b$ or if $A = B$ is not a trivial thing for CFD types of problems because we would like to have and given the computational resources at our disposal. We tend to have larger number of grid points; and large number of grid points means large system of equations to be solved. So, we need to be very careful in terms of what kind of methods we choose to solve this.

We made the point that the Cramer's rule cannot be some more than the 30 equations. And 30 grid points and methods established methods for the solution of $Ax = b$

like Gaussian elimination, and LU decomposition, although they are very general, they are not so good as some other methods like iterative methods that we have also seen in the first part of this module.

We saw two basic iterative methods, the Jacobi method and Gaussian, Gauss-Seidel method; and we have seen that these are fairly easy to implement and write programs for. And they tend to have an arithmetic operation count, which varies as n^2 , where n is the number of equations to be solved compared to n^3 for Gaussian elimination or LU decomposition method.

We have also seen one very good direct method, the tri diagonal algorithm in which the operation count varies as n , but that is not applicable for the general case, it is applicable only for one-dimensional problem, one-dimensional problem with three diagonals. And so when you look at the more generic method then those are n^3 , whereas, the basic iterative methods that we have seen are vary as n^2 for a class of problems related to like those have happening in CFD type of problems.

And we are also seen that that in these iterative methods, we do not solve x equal to b directly, and we solve it in an indirect way like $q \times k$ equal to q minus a times x k minus b . And approach the solution asymptotically. And in the Jacobi method Q is the diagonal all the diagonal elements of a , and therefore, this iterations becomes like x k equal to $M \times k$ minus 1 plus N . And similarly, in the Gauss-Seidel method, we choose Q to be the lower triangular part of a including the diagonal elements and that again can be put in the form of x k equal to $M \times k$ minus 1 plus N .

Once you have this kind of formulation in the problem, then we solved with an initial guess, which can be arbitrarily in the case of convergence schemes. We start with any initially guess and then we get x_1 from you put x_1 here, you get x_2 and so on, and then we keep on doing it. And eventually the solution converges is the (Refer Time: 04:28) converges scheme. If we also saw that in actual computation of $A \times x$ equal to b using the Jacobi method and Gauss-Seidel method, we do not do matrix inversion.


(Refer Slide Time: 04:38)

Basic Iterative Schemes: Jacobi Method

- Consider the set of equations given by

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \dots &= c_1 \\ a_{21}u_1 + a_{22}u_2 + \dots &= c_2 \\ \dots & \\ a_{n1}u_1 + a_{n2}u_2 + \dots &= c_n \end{aligned} \quad (1)$$
- In Jacobi method, we rewrite these as

$$\begin{aligned} a_{11}x_1^{k+1} &= -a_{12}x_2^k - a_{13}x_3^k \dots - a_{1n}x_n^k + b_1 \\ a_{22}x_2^{k+1} &= -a_{21}x_1^k - a_{23}x_3^k \dots - a_{2n}x_n^k + b_2 \\ \dots & \\ a_{n-1,n}x_{n-1}^{k+1} &= -a_{n-1,1}x_1^k - a_{n-1,2}x_2^k - a_{n-1,3}x_3^k \dots - a_{n-1,n}x_n^k + b_{n-1} \\ a_{nn}x_n^{k+1} &= -a_{n1}x_1^k - a_{n2}x_2^k - a_{n3}x_3^k \dots - a_{n,n-1}x_{n-1}^k + b_n \end{aligned} \quad (2)$$
- Start with initial guess $\{x_1, x_2, \dots, x_n\}^0$, solve the n equations of eqn (2) for $\{x_1, x_2, x_3, \dots, x_n\}^1$
- From $\{x\}^1$, now solve n equations of (2) again for $\{x\}^2$, and from these for $\{x\}^3$ and so on.
- Note that it would take $(n-1)$ multiplications and one division to solve one equation and there are n such equations to be solved to advance from step k to step $(k+1)$.
- But if $[A]$ is sparse and has only 5 (or 7) non-zero coefficients, then it takes only $5n$ (or $7n$) multiplications to advance from k to $k+1$!




But we write them in this simple way we reposition them in the simple way so as to get to an estimated value of x_1^{k+1} plus x_2^{k+1} plus x_3^{k+1} plus 1, all the way up to x_{n-1}^{k+1} plus and x_n^{k+1} , in a sequential way, using all the guess value of the last previous iterations value of all other variables.

(Refer Slide Time: 05:10)

Basic Iterative Schemes: Gauss-Seidel Method

- For the same set of equations, in the Gauss-Seidel method, we solve as follows:

$$\begin{aligned} a_{11}x_1^{k+1} &= -a_{12}x_2^k - a_{13}x_3^k \dots - a_{1n}x_n^k + b_1 \\ a_{22}x_2^{k+1} &= -a_{21}x_1^{k+1} - a_{23}x_3^k \dots - a_{2n}x_n^k + b_2 \\ \dots & \\ a_{n-1,n}x_{n-1}^{k+1} &= -a_{n-1,1}x_1^{k+1} - a_{n-1,2}x_2^{k+1} - a_{n-1,3}x_3^{k+1} \dots - a_{n-1,n}x_n^k + b_{n-1} \\ a_{nn}x_n^{k+1} &= -a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - a_{n3}x_3^{k+1} \dots - a_{n,n-1}x_{n-1}^{k+1} + b_n \end{aligned} \quad (3)$$
- Start with initial guess $\{x\}^0$, solve the n equations of eqn (3) for $\{x\}^1$, from these for $\{x\}^2$, and from these for $\{x\}^3$, and so on.
- Note that if $[A]$ is sparse and has only 5 (or 7) non-zero coefficients, then it takes only $5n$ (or $7n$) multiplications to advance from k to $k+1$.
- Thus, the number of multiplications (or divisions) to go from step k to step $(k+1)$ is the same as that for the Jacobi method.
- The actual number of arithmetic operations depends on how many times we have solve to eqn (2) (or (3) for GS) to get a "converged" solution, i.e., one in which $\{x\}^{k+1}$ is almost equal to $\{x\}^k$.

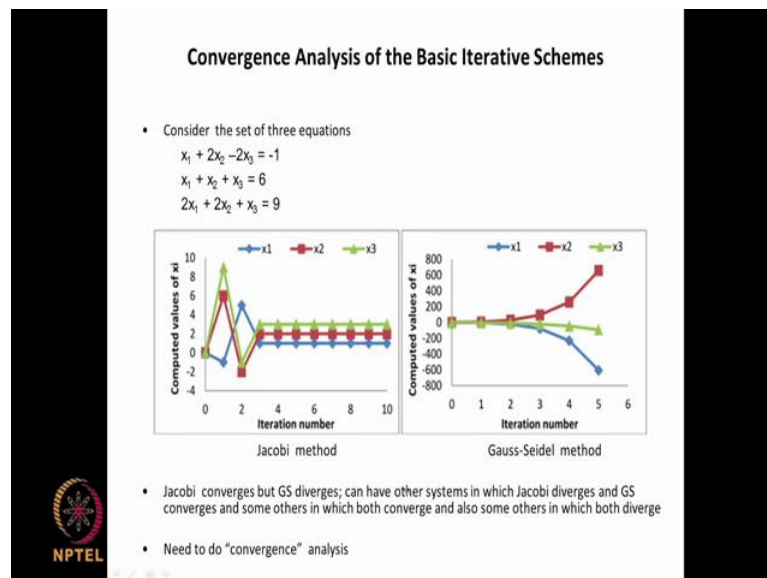


And we have also seen that the way that we solve the Gauss-Seidel method. They are slightly different; in the Gauss-Seidel method, we make use of the updated solution of sum of the variables wherever they become available to make use of the updated one.

Where the updated one is not available, we make use to the old solution. But and we have also seen that the number of multiplications or divisions, varies as N square and we still have to do a lot of iterations.

But because we have 'a' as a very sparse matrix, but Gauss-Seidel method and Jacobi method can take advantage of this sparseness. And do this multiplication operations only on those nonzero coefficient so that the actual number of arithmetic operations to go from step k to step k plus 1 is only of the order of 5 n for 2-D and 7 n for 3-D compact molecules. But we also saw that the actual number of arithmetic operations depends on how many times we have to solve this x k plus equal to M x k minus 1 plus N in order to get the solution that is fairly close to the true solution.

(Refer Slide Time: 06:39)



So, and we also saw one of the characteristics that these methods do not work in all cases. We have taken this simple three unknown equation problem with three questions and the solution can be seen to be 1, 2, 3 for x 1, x 2, x 3. And for this thing, where we know a solution we know the solution and we can show that the solution is unique. We find that the Jacobi methods works very quickly to it gets into that within three four five iteration, whereas, the Gauss-Seidel method diverges.


So, to that extent, although this method for simple, they do not always work, and Gauss-Seidel method even though it is updating making use of the latest value is not necessarily faster or better than Jacobi method. So, we need to do a proper convergence analysis.

(Refer Slide Time: 07:37)

Convergence Analysis of the Basic Iterative Schemes

- In iterative methods, $Ax = b$ is solved as $x^{k+1} = Px^k + q$ $k \geq 0$
- Define error, ϵ , as $\epsilon = \bar{x} - x$ where \bar{x} is the exact solution, i.e., $A\bar{x} = b$ and $\bar{x} = P\bar{x} + q$
- At the k^{th} iteration, we have, $e^k = \bar{x} - x^k$ and $Ax^k = b^k$ and $x^{k+1} = Px^k + q$
- The residual at the k^{th} iteration can be evaluated as $\delta^k = b - b^k = b - Ax^k$: if residual is zero, the solution has converged
- For error reduction, through subtraction, we get $e^{k+1} = Pe^k$
- Thus, at the end of m iterations, we have, $\epsilon_m = P^m \epsilon^0$
- For convergence, $\rho(P) < 1$ where $\rho(P)$ is the spectral radius of iteration matrix P and is given by $\rho(P) = \max \{ |\lambda_i| \}, 1 \leq i \leq n$ where λ_i are the eigenvalues of $P_{n \times n}$.
- For central differencing of the Laplace equation in 2-D with Dirichlet boundary conditions, with uniform spacing in x and y and having $M \times M = M^2$ no of grid points, the spectral radius of the iteration matrix with the Jacobi scheme is given by

$$\rho(P_j) = \cos(\pi/M) \approx 1 - \pi^2/2M^2 \quad \text{for large } M$$
- For the GS scheme, $\rho(P_{GS}) = \cos^2(\pi/M) \approx 1 - \pi^2/M^2 \quad \text{for large } M$



And we start with that convergence analysis in the second part of this. And we look at how we can we establish that there are certain conditions in which these iterative methods can be convergent, and then we will look at what we call as asymptotic rate of convergence, and we characterize an iterative methods by this rate of convergence also among other things.

And then we show that there are certain methods which are better than this basic method in terms of the convergence rate and we discuss a few things, so this is what we are going to do in part b. And we start with the convergence analysis. So, iterative methods we have $Ax = b$ solved as $x_{k+1} = Px_k + q$, for k greater than or equal to 0. So, k equal to 0 is the initial guess; from that to we get x_1, x_2, x_3 and so on.

If we error epsilon here as $\bar{x} - x_k$, where \bar{x} is the exact solution. The double dot is just an easy way of writing it using the script language that is available. So, it does not have any, it is not like exhalation of x , it is not like d square x by d T square, it is just a symbol. It is just to distinguish between the exact solution which is this \bar{x} and the current solution which is x_k . So, the exact solution is such that when we substitute \bar{x} (Refer Time: 09:20) into this, we get $\bar{x} = A\bar{x} + q$. And this satisfies this equation, it also satisfies this equation. So, we also have $\bar{x} = P\bar{x} + q$. So, this is the exact solution.

And if you know the exact solution, and if you know the current solution x , then you can define error as the exact solution minus the current solution. So, you can say that the error at the k th iteration is the exact relation which is always the same for a given A x equal to b minus x^k . And we also have x^k such that once you put this is the one by which we are getting x^k or x^{k+1} . And if you have a x^k , you can put that into this equation and find out what is $A x^k$. And x^k need not be equal to b , so it is equal to b^k .

So, the difference between b^k can b is what is known as the residual; and this residual means that there is a residual amount of dissatisfaction or unsatisfaction of the given equation $A x$ equal to b by the current value x^k . So, if you substitute x^k into this equation, and then you get b^k . And if the b^k equal to b , then x^k is the exact solution, because it is satisfying this equation. So, the difference between b and b^k is therefore, the amount of non satisfaction of the equation $A x$ equal to b by our current guess our, current estimated value of x^k . So, this is the residual amount that still needs to be satisfied.

So, how can we get that we can put this δ^k as b minus $A x^k$. Although, we cannot compute error, because we do not know the true solution, we can readily compute the residual. Because in order to complete the residual, we just have to b which is known because the problem is given and then x^k is the current estimate value, so we know this, and A is also known.

So, b minus $A x^k$ is the residual. If the residual is zero, the solution has coverage; if it is not zero, it has not converged, and we need to do more. If the residual is very, very small then we can say it is pretty close to the solution. If it is satisfactory to me then I will take it if it satisfactory me in the sense that it is not just nothing personal here if I see that previous iteration value and the current iteration value and the two differ by only in the fifth decimal place or the eight decimal place, then maybe I say that it is accurate enough for my purposes and I can stop.

So, we would like to decrease the residual. And when you look at it closely, this δ^k , k is the superscript telling us how many iterations we have done starting from the initial guess, but this delta here is not for a single equation, it is for the entire set of equations. And since we are getting delta equal to b minus b^k , b and b^k are both column vectors, so δ^k is also a column vector.

So, that means, that it indicates the amount of non-satisfaction of $Ax = b$, for each equation, so that means, that when you say Δ_k residual is 0; that means, that each value of this column vector must be 0; that means, that each of the n equations of $Ax = b$ must be satisfied or they should be so small. Now, how do we say that this Δ_k small, because Δ_k is set of numbers, so the first equation is 0.0015, second equation may be 0.3025 may be for third equation 0.20117 something like that.

So, what is small and which one of this is small. We bring then the conflict of norm of this vector. So, the norm can be the largest of these n elements is one measure of the norm is a measure of the bigness of the magnitude of the elements that are there in that particular vector. So, it can be the maximum value of a our Δ_k of any element of Δ_k must be less than a certain value is one particular we are saying that every equations is to be satisfied to within this much tolerance.

Or we can say that some root means square value of all the elements must be so much, so that that is equivalent to L_2 norm kind of thing. So, we can define those kinds of norms of the vector, and in this case, the vector is the residual and then we can look for the norm of this residual vector to be smaller than a certain value. And if you say that if it is smaller than that the value, then you can say that I got a solution which I feel it is close enough. And typically what kind of values to be say we would like to have a Δ_k , which is the small fraction of the Δ_0 .

You have x_0 to start with and then you put that x is 0 here, and then you get b_0 . So, b_0 minus b , $b - b_0$ will give you Δ_0 . And we would like Δ_k to be reduced by orders of magnitude compare to the original value of Δ_0 and that seems to be a good way of measuring what is small, because the small is with respect to what is started out with. If you put the problem in kilometres, you will get small values; but if you express the same thing in meters, it will become 1200; and if you express the same thing millimetres then it will become 12 million or 1.2 million.

And so in that sense, it is not the number, because the number changes with the units. We would like to make sure that the residual is changing relatively, so that is why we can say that Δ_k / Δ_0 it should be about greater than say 10^{-3} or 10^{-4} . We know that Δ_k is decreasing as k is increasing because it is converging scheme so we are slowly, slowly approaching that.

In that sense, we would like the δ_k to be at least say 1000 times smaller or 10,000 times smaller than the beginning value. So, we look for orders of magnitude reduction of the residuals, so how many iterations does it take to reduce the order of magnitude of the residual by a factor of 10 or a factor of e is known as asymptotic convergence rate, we will come to that shortly.

So, we have this one here, and we have this, if you subtract this from this, you get $x^{k+1} - x^k = P(x^k - x^{k-1})$. So, the error at $k+1$ is P times $x^k - x^{k-1}$, so that is error at k , so this; and q and q will cancel out. So, by subtracting this one from this, you can say that error at $k+1$ is P times error at k . So, in the simplest sense, if errors were not vectors, if P is not a matrix, so then we could say P is a multiplication factor, so the error rate k is being multiplied by a factor given by P and that is error $k+1$.

So, this multiplication factor or amplification factor is how the error is changing, and it is not cause tends the multiplication factor, but it is an indication of whether the scheme is convergent and how fast it is convergent. If obviously, if P is less than one, if it is a scalar, then you could say you have a convergence scheme, because if P is point nine, next time it is only point nine of the current value, next time it is 0.9 times 0.9, 0.81, so in that sense it is gradually decreasing. So, at the end of n iterations, we can say that error at m , this should have been a superscript is P raise power m times error at 0. So, if P is less than one in magnitude then you can say the error is decreasing at this rate, a rate which is a function of this magnitude of P .

It is not the magnitude of the, it is not the norm of P or anything like that. What we are looking at is an error, which is governed by its Eigen values, and so that we have something called the largest the spectral radius of the iteration matrix P and spectral radius is the largest Eigen value. Given that we are dealing with A which has real coefficients, and no imaginary coefficients.

We expect the Eigen values of P to be real. And we say that the end of m iterations the error at after m iterations that this m should be superscript here is equal to P raise power m times error at the first iteration. And if P is less than one, then we can see that the error at n step will be less than the error at this. And each time it is being it is being reduced by

a factor, which is given by the magnitude of P . Now, in this particular case, P is not some simple factor.

(Refer Slide Time: 20:59)

Residual Reduction

- At the k^{th} iteration, we have $\{x\}^k$ and we can compute $Ax^k = b^k$
- The residual at the k^{th} iteration can be evaluated as $\delta^k = b - b^k = b - Ax^k$: if residual is zero, the solution has converged
- Residual reduction is not uniform and it does not decrease at the same rate. For large k , an asymptotic rate may set in and this needs to be increased to reduce the overall time of solution of $Ax=b$

NPTEL

The residual usually varies in a non-monotonic way like this. And in some cases, it can even increase and then decrease. So, here we have at k^{th} iteration, we have x^k , and we can compute $Ax^k = b^k$, and then we define this residual δ^k . And residual reduction is not uniform; it does not decrease at the same rate along with iterations numbers. But one can do some mathematics, and say that it is governed by the Eigen values of the iteration matrix. And these Eigen values are going to be some are going to be large and some are going to be small, there are n number of Eigen values for an N system of equations.

So, if you have a million grid points, you will have million Eigen values. So, all of those Eigen values will be contributing to the reduction of the error at all the time, but those Eigen values which are small will die out first, their influence die out first. And the largest Eigen value is the one which is going to ultimately determine the rate at which how fast the residual is decreasing. So, as you go through this iteration number, initially you have usually a steep variation, because the small Eigen value contributions are decreasing pretty fast. And eventually when you put this on a log scale and this on a log scale log-scale and this is linear, you will see something like a logarithmic decreased gradual decrease with a slope which is no longer changing.

So, in this part, there are different Eigen values, and all of them are contributing here. And sometimes it contributing such a way that initially the error does not decrease in especially in practical CFD computation. But then it eventually starts decreasing and as it gets closer to the solution then it will be decreasing at a constant rate at a slope which is essentially the slope of this residual versus iteration will tell us at what rate it is decreasing. So, this the constant slope at which it is decreasing for large number of iterations, for large number of n here is known as the asymptotic rate of convergence.

(Refer Slide Time: 23:42)

Convergence Analysis of the Basic Iterative Schemes

- In iterative methods, $Ax = b$ is solved as $x^{k+1} = Px^k + q$ $k \geq 0$
- Define error, ϵ , as $\epsilon = \bar{x} - x$ where \bar{x} is the exact solution, i.e., $A\bar{x} = b$ and $\bar{x} = P\bar{x} + q$
- At the k^{th} iteration, we have, $\epsilon^k = \bar{x} - x^k$ and $Ax^k = b^k$ and $x^{k+1} = Px^k + q$
- The residual at the k^{th} iteration can be evaluated as $\delta^k = b - b^k = b - Ax^k$: if residual is zero, the solution has converged
- For error reduction, through subtraction, we get $\epsilon^{k+1} = P\epsilon^k$
- Thus, at the end of m iterations, we have, $\epsilon_m = P^m \epsilon^0$
- For convergence, $\rho(P) < 1$ where $\rho(P)$ is the spectral radius of iteration matrix P and is given by $\rho(P) = \max \{ |\lambda_i| \}, 1 \leq i \leq n$ where λ_i are the eigenvalues of $P_{n \times n}$.
- For central differencing of the Laplace equation in 2-D with Dirichlet boundary conditions, with uniform spacing in x and y and having $M \times M = M^2$ no of grid points, the spectral radius of the iteration matrix with the Jacobi scheme is given by

$$\rho(P_J) = \cos(\pi/M) \approx 1 - \pi^2/2M^2 \quad \text{for large } M$$
- For the GS scheme, $\rho(P_{GS}) = \cos^2(\pi/M) \approx 1 - \pi^2/M^2 \quad \text{for large } M$

So, that if in asymptotic rate of convergence is governed by the largest of the Eigen values of the iteration matrix, and that is known as the spectral radius of the iteration matrix. So, we need to know the spectral radius we do not need to know all the Eigen values, all the million Eigen values, we need to know what is the largest Eigen values. If the largest Eigen value is less than one, then we have a converging method; and the rate at which it converges depends on well the spectral radius of this.

So, we will come back to this. As I mentioned you have n-by-n matrix here, and if you have a million grid points, million unknowns, we will have million Eigen values. And there is no easy way of finding all the Eigen values, but for some class of problems, it is possible to get the theoretical expression for Eigen value.

For example, for central differencing of the Laplace scheme, Laplace equation in 2-D with Dirichlet boundary conditions, we have done this. So, with uniforms spacing with x

and y and having M by M that is M square number of grids point. So, we have in the x -direction n grid points, and in the y -direction, we have again M grid points, so that we have a total number of M square grid points.

The spectral radius of iteration matrix with the Jacobi scheme is given by $\cos(\pi/M)$, so that is roughly equal to $1 - \frac{\pi^2}{2M^2}$ for large values of M . So, that is if you have a few division like you have 4 by 4 matrix then this may not be applicable; but if you have 40 by 40 or 100 by 100, for large value of divisions M here then the largest Eigen value is given by $1 - \frac{\pi^2}{2M^2}$.

Now, what is this means, this means that if you are looking at M equal to 50, then this equal to $1 - \frac{\pi^2}{2M^2}$, π^2 is about 10 by 2 M^2 squared, this M is also in the denominators, so that means, that 2 by 50 square 2 by 2500, so the total will be 10 by 5000, so that is 1 by 1000, so this will be equal to 0.999. This whole thing will be equal to 0.001, so $1 - 0.001$ is the largest Eigen value. If you take M to be 100, then this is $1 - \frac{10}{2 \times 100^2}$, so that is 2 times 10 to the power 4 1 by 10 by 20,000. So, this Eigen value, largest Eigen value will be 0.9998. So, it has increased from already large value of 0.999 to even larger value of 0.9998.

Now, what is the difference? The rate of convergence is determined by how far away from one the spectral radius is. So, in one case in the case of M equal to 50, it is away by 0.001. In the case of M equal to 100, it is away by 0.0002, so it that makes it away by factor of 5, and so that means, that the convergence rate will be lower will be less for larger value M . So, as M increases, then the spectral radius approaches one and the closer it is to 1, the slower rate is to converge.

Now, for this Gauss-Seidel scheme for the same Laplace equation 2-D with the same kind of number of divisions here the spectral radius is known theoretically to be $\cos^2(\pi/M)$. Here it is $\cos(\pi/M)$, and so that is one minus π^2 by M , whereas, here it is $1 - \frac{\pi^2}{2M^2}$, this two is the this whole thing is in the denominator.

And what does this mean here, if you have M equal to 50 and π^2 is ten, so by 10 by 250, 2500, so that is 0.996. So, instead of 0.999, it is 0.996. And for M equal to hundred, this will be this will be 0.999 and not what are the values that we had earlier. You know it is 0.999 and we had 0.9998, so; that means that the spectral radius for a

Gauss-Seidel method for this problem is always less than the spectral radius for the Jacobi method.

And since the spectral radius determines the asymptotic rate of convergence for large values, and from this we can say that the Gauss-Seidel method for this particular problem where we know the spectral areas analytically, it is faster than the Jacobi method. And you can also show that it is faster by a factor of two for large values of M . And we would also like to mention that in these methods, it is not all the Eigen values that are contributing to the rate of convergence, it is only the largest Eigen value that is contributing to the rated convergence.

There are other methods, where even these values intermediate values of Eigen values will also contributed. So, other thing we would like to mention here is that as M increases for the same method the spectral radius approach is closer to the value of one and therefore, it becomes slower and slower.

So, what we would like to take from this first lesson of part two of the advanced method through some convergence analysis, we have essentially shown that for a typical iterative method, the residual the error residual decrease as a function of the rate of decrease is determined by the Eigen values of the iteration matrix. And if the iteration matrix changes then the convergence behaviour also changes.

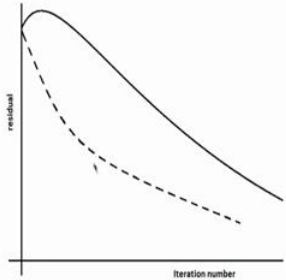
We have also seen the condition that for convergence the spectral radius of the iteration matrix should be less than 1. And the rated which it converges under asymptotic conditions that is for large values of M , in which the influence of the smaller Eigen values die out is given by spectral radius, the closer it is to one, the slower it is to converge. We do not know this Eigen values for the general case. So, in the specific case of Laplace equation 2-D Dirichlet boundary condition, Jacobi methods, uniform spacing and all that then there are certain mathematical formulas, expressions we can drive.

And these expressions show that the Jacobi method has a higher spectral radius than the Gauss-Seidel method for the same grids spacing and all that. And therefore, it converges slowly, and it is the rate of convergence asymptotic convergence is twice is half as much as the Jacobi method.

(Refer Slide Time: 32:20)

Residual Reduction

- At the k^{th} iteration, we have $\{x\}^k$ and we can compute $Ax^k = b^k$
- The residual at the k^{th} iteration can be evaluated as $\delta^k = b - b^k = b - Ax^k$; if residual is zero, the solution has converged
- Residual reduction is not uniform and it does not decrease at the same rate. For large k , an asymptotic rate may set in and this needs to be increased to reduce the overall time of solution of $Ax=b$



NPTEL

So, this is the kind of error reduction that we can expect and for large values of M the iteration number, the residual decrease at a constant rate; and that constant rate is expressed like this. For the error to decrease by a constant factor it is not a summative factor, it is a multiplicative or divisible divisor factors.

So, it is by a constant factor multiplicative factor then it takes the same number of iterations, so that is if you are looking specifically at error reduction by a factor of 10, then it is given the number of iterations is given $1 / \ln(1 - \rho)$ here. So, we have a ρ is the spectral radius. So, $\ln(1 - \rho)$ is the $1 / \ln(1 - \rho)$ is the rate at which is the number of iterations, which are required to reduce errors by a factor of E , which is two point seven one or something like that.

So, if you want to reduce error by a factor of 10, then it will take you $2.3 / \ln(1 - \rho)$ there. And so if you put this on a log scale here, and then if you put the iteration number on a linear scale, then to go from say fourth decimal to fifth decimal, it takes so many number of iterations. And then from fifth decimal to sixth decimal, then it takes again the same number of iterations.

And 6th decimal to 7th decimal, again it takes same number of iterations. So, you can in a way, you can anticipate how many more iterations you have to do in order to get convergence to within the 5th decimal or sixth decimal, seventh decimal like that. And we can stop at some point. So, in usual conventional practice for the usual solutions,

where you are not demanding huge amount of accuracy, you could say that I would like to stop my iteration, once my residual decreases by a factor of 10 to the power 4 or 10 to the power 5 from the initial residual.

And so if you take that kind of thing and given that there is a faster reduction here and given that this is slower reduction, typically we would like to bring this down to very low value and this determines the total number of arithmetic operations iteration numbers that are needed for this. And that number varies as it is estimated to be about $0.466n$, and where n is the number of equations and a number of unknowns.

So, it is about $0.5n$ for every can fold decrease in the residual reduction. So, if you are looking at five orders of magnitude reduction then it will take $2.5n$ number of iterations in order to reduce the residual by factor of 5. And get to our closure stoppage of iteration, that means that you start with some guess value, and you carry out two and a half times n number of iterations to get to the final value. In Gauss-Seidel method or Jacobi method, say it is something proportional to n . And each step takes about $5n$ or $7n$ number of arithmetic operations, so the total number of arithmetic operations is the number of operations per step times the total number of steps that you want to take.

So, the first step is $5n$ for a 2-D problem and that is number of arithmetic operations first step, and we have to take something like $2.5n$ number of steps. So, the total number of arithmetic operations is $5n$ times $2.5n$, so that is about $12.5n^2$. So, that meant that in the Jacobi method or the Gauss-Seidel method in for this Laplace problem, you can say that it takes about 10 to 15 n^2 number of arithmetic operations, where n is the number of unknowns, number of grid points, and this needs to be compared with and n^3 by three number of arithmetic operations, which are required for the Gaussian elimination method. So, if your n is order of hundred then maybe there is not that much of a difference.

But if you have a million grids point, which is not unusual, then Gaussian elimination would take million cubed so that is 10^6 to the power 3 number of arithmetic operations. And Gauss-Seidel method will take only 15 times n^2 , so that is 15 times 10^6 to the power 2, so it is 10^6 to the power 13 number of arithmetic operation. So, there is a huge difference between Gaussian elimination and Gauss-Seidel method for such kind of problems, which is why typically Jacobi method and Gauss-Seidel method can be used

for most of the problems where you have diagonal dominance. And in such case, this is it takes n^2 number of operations, and therefore, these are considered superior to other to the Gaussian elimination or those types direct methods.

In the next lecture, we will see how we can improve on this convergence rate of n^2 number of operations.