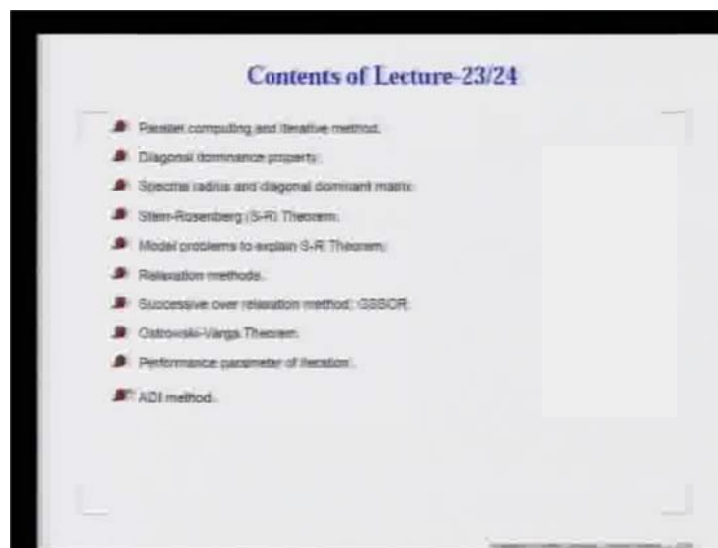


Foundation of Scientific Computing
Prof. T. K. Sengupta
Department of Aerospace Engineering
Indian Institution of Technology, Kanpur

Module No. # 01
Lecture No. # 23-24

(Refer Slide Time: 00:15)



In this combined lecture 23rd and 24th, we continue our discussion on solving elliptic equation once again.

And we try to go back historically to the Jacobi method and point out that some of these point iterative methods are more amenable to parallel computing. Having identified that we also discuss about the convergence properties of these methods, which relate to the diagonal dominance property of this associated A matrix which in turn determines the spectral radius of this diagonal dominant matrix. In this context, we are going to talk about a theorem which is attributed to Stein and Rosenberg. We will pick some model problems and explain this Stein and Rosenberg Theorem.

We notice that there are other possibilities of accelerating the convergence namely by relaxation methods; these are built up on around those basic methods that we already talked about.

In this relaxation method which is also known as successive over relaxation method, we are going to talk about SOR method or successive over relaxation method on the Gauss-Seidel basic methods.

We find out that there are ranges of this relaxation parameters that is given by this Ostrowski-Varga Theorem, which says that this parameter has to be between 0 and 2.

Once we have figured out how to accelerate the convergence, we try to quantify this convergence criteria or the performance parameter of iterative methods - that is what we are going to talk about. And in this context we see that some of these line iterative methods could be further improved.

A new method was **AD** in 1950s which is called the alternative direction implicit method or ADI method. This is how we are going to switch over from classical method to a method which is found to be quite efficient.

(Refer Slide Time: 02:39)

Convergence Issues for Iterative Methods

- We heuristically establish certain rules for the Jacobi method first.
- For the Jacobi iteration for $[A]\{X\} = \{b\}$, we split $[A]$ as:
 $[A] = [L] + [D] + [U]$.
- Where $[L]$ is strictly the lower triangular matrix (without any non-zero diagonal entries); $[D]$ is the diagonal matrix and $[U]$ is strictly the upper triangular matrix.
- For the Jacobi iteration, $[N] = [D]$ and then the error-gain matrix is given by,
$$\begin{aligned}[G] &= [I] - [D]^{-1}[A] \\ &= [D]^{-1}[D] - [D]^{-1}[A] \\ &= -[D]^{-1}([A] - [D])\end{aligned}$$
- Thus, the error gain matrix is given by,
$$[G] = -[D]^{-1}([L] + [U]) \quad (45)$$

© 2002 by Pearson Education, Inc. All rights reserved.

You would find the Jacobi method still used for solving some of these classes of problems, because you do not really need to pass on the information as the iteration progress is from the neighboring nodes.

You know one of the major problems in parallel computing is IO, input output processes. IO processes actually reduce the speed and sometimes it can lead to what is called as a latency problem.

Maybe one node is done, but it is waiting for the neighboring nodes to complete that sequence of work and pass that information; then only you can proceed.

Some nodes will be active and some will be inactive. This is a problem of latency that comes about and that is not visited upon, when you are actually using Jacobi method.

Please do understand that it is not a historic discourse. We still have utility for methods like the Jacobi method.

For the Jacobi iteration for this equation $AX = b$, we split the A into three parts. This is not the same as what we talked about that particular example of $A_1 A_2 A_3$ matrix - that we have talked about. Here it is a much more of a general problem. So this A matrix is split into a strictly lower triangular matrix that we identify as L .

Then we have a D matrix which is the diagonal matrix and U is the strictly upper triangular matrix.

When I write L and U on the diagonal, there are no non-zero entries; those non-zero entries are strictly put in to the D matrix itself.

Now we also know for the Jacobi iteration. We do that iteration where we take that new matrix, which replaces the original A matrix, given by the diagonal matrix itself. So N is equal to D .

And we have seen that G is given by $I - N^{-1}A$; N is D so, this is $D^{-1}A$. The I itself, I could write it as $D^{-1}D$. If I do that then I can factor out D^{-1} and I get this: What is $A - D$? $A - D$ is nothing but $L + U$.

We have an expression for G matrix which looks rather clean as given in the last equation 45. All you need to know is the diagonal split matrix and the sum of lower and upper triangular matrix.

(Refer Slide Time: 05:35)

Convergence Issues for Iterative Methods

- As the error vector iterates are related by,

$$\{e\}^{(k)} = [G]\{e\}^{(k-1)} \quad (46)$$
- One can write this in component form with the help of (45) and using $[A]_{ij} = [\alpha_{ij}]$ as,

$$e_i^{(k)} = \frac{1}{\alpha_{ii}} [-\alpha_{i1}e_1^{(k-1)} - \alpha_{i2}e_2^{(k-1)} - \dots - \alpha_{iN}e_N^{(k-1)}] \quad (47)$$
- We set (for $i = 1, 2, \dots, N$),

$$\theta_i = \sum_{j=1}^{\prime N} \left| \frac{\alpha_{ij}}{\alpha_{ii}} \right|$$
- Primed summation indicates that $j = i$ is omitted in the sum. The parameter θ_i shows the measure of the dominance of coefficients vis-à-vis the diagonal component.

Proceedings of Scientific Computing, Chennai, 1999, p. 2463.

This we have already stated that error goes by multiplication of the G matrix and - what we can say that - e of k would be written in terms of G into e of k minus 1.

Now what we can do in writing out in component form - what we actually do - is basically, we write it like alpha ii into e i. Everything else other than the diagonal entries put on the right hand side - that is what you have it in this square bracket in 47.

Starting from e1 to eN, they are at the previous level and multiplied by the entries of that e matrix set **A r G r** - whatever you say - that is how you get it. This is how we get.

Now what you notice is that we can define a parameter which I call as theta i, which is nothing but these quantities that are inside the coefficients alpha ij. And everything is divided by the diagonal entries so that i write it as alpha i i.

What happens is - the error at the i th point at the k th iterate depends on the product of this theta i times the error at the previous step.

What does this parameter indicate? The parameter actually tells you how each of these quantities - different points and different errors - are contributing to the error at this k th level.

If I have this quantity very small, that quantity that particular component of e at the previous level does not contribute.

While if this point is very close to 1 or more than 1, then I can see that the quantity actually gives more contribution.

(Refer Slide Time: 07:48)

Convergence Issues for Iterative Methods

- If we define $\|e_k\| = \max_i |e_i^{(k)}|$, then using the maximum principle on the absolute values in (47) and triangle inequality, one gets

$$|e_i^{(k)}| \leq \frac{1}{|a_{ii}|} \sum_{j=1}^N |a_{ij}| |e_j^{(k-1)}|$$
 or $|e_i^{(k)}| \leq \|e^{(k-1)}\| \theta_i \quad \text{for } i = 1, 2, \dots, N \quad (48)$
- Hence the above can be written for the maximum norm to provide the estimate:

$$\|e^{(k)}\| \leq \|e^{(k-1)}\| \theta_{max}$$
- Convergence will be faster for smaller θ_{max} . This is alternatively stated as: If $\theta_{max} < 1$, then the Jacobi iteration converges. This is the *sufficient* condition of **Diagonal Dominance** for convergence. It also is the sufficient condition for Gauss-Seidel iteration.

Foundations of Scientific Computing, Chapter 10, p. 1050.

What happens is this ratio theta i, basically tells you the role of the coefficient vi-sa-vi, the corresponding diagonal component - that is what it is.

Given a matrix A, then you should start looking for what is this ratio going to be. If this ratio is less than 1, that means what? It will contribute lesser, and lesser at the subsequent level - that is what it means. This property is called the property of diagonal dominance.

I could basically define based on maximum principle, that some kind of an error norm which I have called here at e of k, that is the maximum where I scan through all the entries of that ek vector, and then we have this triangular inequality.

I could write that the modulus of error at the k th point is less than that quantity that we defined as θ_i .

In 48, you can see this θ_i is here and that basically tells you that - I could say that - the error at k th level is bounded by error at the previous level times the maximum value of θ_i .

This makes at a stronger case. If I scan through all θ_i and pick up the maximum, then this inequality will be even stronger when I replace these θ_i 's by θ_{\max} - so that is what we have done.

What happens is - then we can see - the convergence will be faster and the error will decay faster if I have this θ_{\max} itself as small as possible.

This is what we really want; as we keep on going through this exercise of iteration, we want the successive error norm to keep coming down and that will be ensured if this θ_{\max} is less than 1.

If θ_{\max} is less than 1, then we can see this Jacobi iteration will converge and this is what is called as the property of diagonal dominance.

That means what? The diagonal entry dominates over all the half diagonal terms. Then of course, all the θ_i 's will be less than 1, the maximum value is also less than 1 and so θ_{\max} is less than 1. This is what is called as a diagonal dominance condition for convergence of Jacobi iteration; this is shown to be sufficient and it has been also shown as a sufficient condition for Gauss-Seidel iteration.

(Refer Slide Time: 10:41)

Convergence Issues for Iterative Methods

- Condition of diagonal dominance is equivalent to having spectral radius of the matrix $[G_J]$ to be less than one and both in turn implies convergence.
- Relative magnitude of spectral radii of Jacobi and Gauss-Seidel error gain matrix has been enunciated by Stein-Rosenberg theorem, stated as:
 - If the $[A]$ matrix has the following properties,
 - (i) $\alpha_{ii} > 0, \alpha_{ij} \leq 0, i \neq j$
 - (ii) $\alpha_{ii} \geq \sum_{j=1, j \neq i}^N |\alpha_{ij}|$ with strict inequality for some i
 - (iii) $[A]$ is irreducible
- And $[G_J]$ and $[G_S]$ are the matrices of Jacobi and Gauss-Seidel iterations, respectively, then one and only one of the following mutually exclusive relations holds for the spectral radius, λ :

(49)

Foundations of Scientific Computing, Chapter 10, p. 1050

However, I will come back to you and show you some simple examples where I will investigate that in some cases Jacobi iteration converges, Gauss-Seidel does not, and vice versa.

Basically, what does this diagonal dominance mean? That is also something similar to what we have already said. Theta max would be somewhat related to the spectral radius of the G matrix. We have seen right? In the beginning of today, that the spectral radius - the maximum eigen value determines the convergence rate. And if spectral radius is less than 1 and - you have alternatively - we have a matrix which is diagonally dominant, then both of them actually imply convergence.

There has been some attempt in formalizing some of these results. And this has been stated by one of the theorems that is due to Stein and Rosenberg, which states the following conditions that if I have A matrix with the following property: The properties are written like this - the diagonal quantities are positive and up diagonal quantities are negative. And in addition what we have is the magnitude of the diagonal is greater than the sum of the magnitude of all of diagonal terms.

Theta was measuring individual entities, but this theorem demands something more stronger.

It says, you take the (i,i) diagonal entries, obtain its magnitude, sum it over, excepting of course the diagonal part, and if this inequality is given, then it is strictly diagonally dominant; so the diagonal is really overpowering everything.

Now, it may seem too much of a thing to us - I mean - how often would you get to that kind of a scenario?

Let us take a look at what we actually do. If you recall the A matrix that we have written for the Laplacian, what we found is that the A matrix had minus 4 along the diagonal, then we had plus 1 sub diagonal, super diagonal we had 1 - I am just talking about a simple case of where $x = y$ kind of a problem and that is this.

Now if you look at many problems of computing, you come across the Laplacian operator. And this is the corresponding expression for the A matrix for the Laplacian.

You can see, I could just take a minus sign whole through; I could get the diagonals as all positive and non-diagonal terms as all negative.

In this case, what happens if I look at it - Say some line here - Well, I have drawn it in such a way that I do not have all the five quantities together, but if I look at it, what I see is that the diagonal term in those cases where I have all the five entities becomes equal to - see this sign, there would be equal to sign that is what is satisfied - 4 is the sum of all the (i,i) diagonal term; this is quite often obtained.

In addition, if you look at some of the top lines and the bottom lines where you can actually find that this is missing - so of course - this is a strictly diagonally dominant row in the first line.

In the same way, the last row you can see is strictly diagonally dominant; because this has become 4 and this has become 2. This kind of condition, as demanded in this Stein-Rosenberg theorem is quite often met.

This statement that is strict inequality for sum i is required; it is very often made. And the last condition $3A$ is irreducible.

I suppose you can recall here - earlier exposure on linear algebra. What is irreducible matrix? Well, it is that the rank of the matrix is equal to the size of the matrix.

That means what? You do not have any irradiant equations; all the equations are linearly independent. Then most of the physical problem, that is how it should be; that is not a very strict condition to demand from.

Now, if we have a matrix A with these three properties and we can estimate the gain matrix for the Jacobi iteration GJ and the Gauss-Seidel matrix, that I call it as GS.

(Refer Slide Time: 16:17)

Stein-Rosenberg Theorem

- (i) $\lambda(G_J) = \lambda(G_S) = 0$
- $0 < \lambda(G_S) < \lambda(G_J) < 1$
- $1 = \lambda(G_S) = \lambda(G_J)$
- $1 < \lambda(G_J) < \lambda(G_S)$ (50)

• The essence of this theorem is that: The Jacobi and the Gauss-Seidel methods are either both convergent or both divergent and if both methods converges then Gauss-Seidel method converges faster. Similarly if both are divergent, then Gauss-Seidel method is more divergent.

• If $[A]$ is a symmetric positive definite matrix, then Gauss-Seidel iteration always converges.

• Due to boundary conditions/ closures, above condition of SPD is hardly ever met in practical computations.

Then we could have the following mutually exclusive relationship holding: The first condition which says that may be all the eigen values of a Jacobi and Gauss-Seidel iterations are equal to 0. Either that is true or both the methods are convergent. So, all the eigen values are less than 1.

However, if you look at the sequence that when both the methods are convergent, the Gauss-Seidel is more convergent; because Jacobi lambdas are greater than lambdas for Gauss-Seidel.

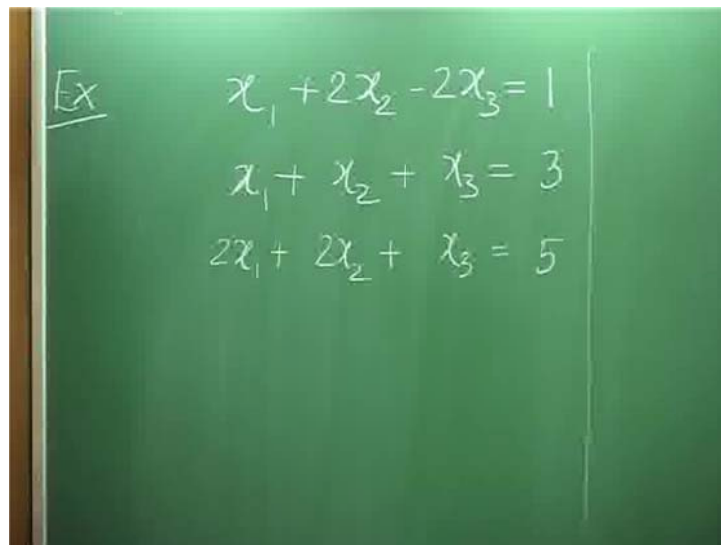
When both the methods converge, the Gauss-Seidel methods seems to perform better and that should be our expectation. That is what I told you in the beginning that when I use

more concurrent information, I should expect some improvement, but that provided that A matrix should have all those three properties that we saw in the previous line.

The second last case is that both the methods are neutrally convergent; means - they do not reduce error, it just simply keeps on retaining it and change. In fact, in many of the practical competitions you will see that is what is happening.

After a few iterations you see, it does not change at all; it just remains like this. And the last condition says, both the methods could be divergent. If they are divergent, you will find the Gauss-Seidel method is more divergent than the Jacobi method.

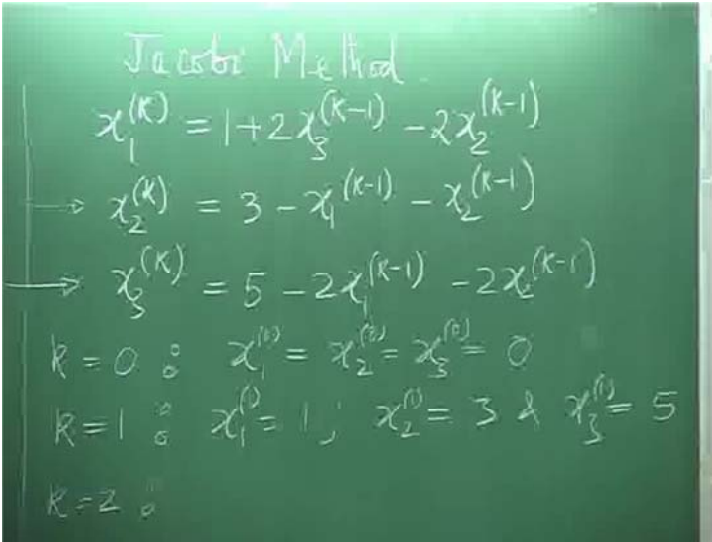
(Refer Slide Time: 17:54)



Ex
$$\begin{aligned}x_1 + 2x_2 - 2x_3 &= 1 \\x_1 + x_2 + x_3 &= 3 \\2x_1 + 2x_2 + x_3 &= 5\end{aligned}$$

It is a simple model problem. We are looking at - it is simple model problem - you should not have to really worry. You could just simply inverse the matrix and get the solution directly, but let us consider this. Why are the Jacobian Gauss-Seidel iteration? How do we follow it?

(Refer Slide Time: 18:30)



Handwritten notes on a green chalkboard showing the Jacobi Method equations and initial values.

Jacobi Method

$$x_1^{(k)} = 1 + 2x_3^{(k-1)} - 2x_2^{(k-1)}$$
$$\rightarrow x_2^{(k)} = 3 - x_1^{(k-1)} - x_3^{(k-1)}$$
$$\rightarrow x_3^{(k)} = 5 - 2x_1^{(k-1)} - 2x_2^{(k-1)}$$

$k=0$: $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$

$k=1$: $x_1^{(1)} = 1$, $x_2^{(1)} = 3$ & $x_3^{(1)} = 5$

$k=2$:

Basically, what we are going to do is - if I am doing Jacobi iteration, what I would do?

If I do this what do I get? Let me start with some initial Guess. And the easiest thing for me to do would be to take all of them equal to 0.

Then what I am going to get from here? The next iterate k equal to 1 - what I would get? x_1 would be equal to 1, x_2 would be equal to 3 and x_3 would be equal to 5. So, these are basically we are $(())$ these are $(())$ initially $(())$ and from the initial guesses we are obtaining this. Now the next step, what would I get?

(Refer Slide Time: 19:50)

$$\begin{aligned}
 x_3 &= 1 & x_1^{(k)} &= 1 + 2x_2^{(k-1)} - 2x_3^{(k-1)} \\
 x_2 &= 3 & x_2^{(k)} &= 3 - x_1^{(k-1)} - x_3^{(k-1)} \\
 x_1 &= 5 & x_3^{(k)} &= 5 - 2x_1^{(k-1)} - 2x_2^{(k-1)}
 \end{aligned}$$

$k=0$: $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$
 $k=1$: $x_1^{(1)} = 1$, $x_2^{(1)} = 3$ & $x_3^{(1)} = 5$
 $k=2$: $x_1^{(2)} = 5$, $x_2^{(2)} = -3$ & $x_3^{(2)} = -3$
 $k=3$: $x_1^{(3)} = 1$, $x_2^{(3)} = 1$ & $x_3^{(3)} = 1$
 $k=4$:

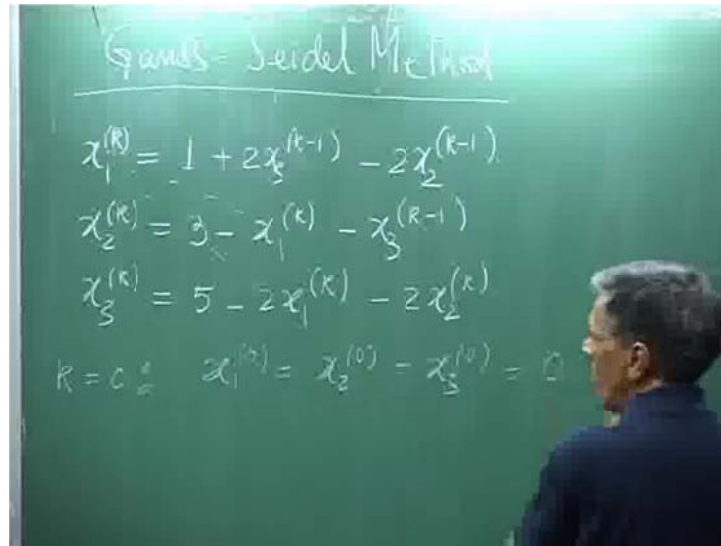
You just substitute these values of x_1 and x_3 . What I am going to get here? This will be, 1 plus 10 minus 6 - that will be 5.

The second equation would be equal to minus 3 and third equation would give us also equal to minus 3. And you look at k equal to 3. What we are going to get? This 2 will cancel out. I will get x_1 of 3 equal to 1. And from the second equation, I will get 3 minus 1 and plus 3; this will also be minus of minus (()) 1 and this also equal to 1.

Of course, there is no price for guessing, that is the solution. We can see that all the equations are satisfied when I take this. So what it actually means is that if you keep doing it further on, they will be the same (())

What it shows is that for this model problem, Jacobi method actually converge in the pre (()) pre step (())

(Refer Slide Time: 21:50)



Gauss-Seidel Method

$$x_1^{(k)} = 1 + 2x_2^{(k-1)} - 2x_3^{(k-1)}$$
$$x_2^{(k)} = 3 - x_1^{(k)} - x_3^{(k-1)}$$
$$x_3^{(k)} = 5 - 2x_1^{(k)} - 2x_2^{(k)}$$

$k=0 \quad x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$

Now if I move over and try to set up an equivalent Gauss-Seidel method, what I would be doing? I would take - let say - the first equation to evaluate x_1 .

If I call this as the k th iterate for x_1 , then I will probably have an identical step, that is what we had done for the Jacobi method.

However, what will happen is when I come to evaluating x_2 , because it is a point of Gauss-Seidel (()) if I already have something (()) I will use that current information.

The x_1 being evaluated here, I will put it like this, and of course x_2 . Did I write something wrong? None of you pointed out. This will be x_3 k minus 1.

And look at the third step that would give us what? 5 minus 2 x_1 . This is available at a current level - so I could basically - so from here, I will use this information, whatever x_1 is made available by the current level of competition - we will use that.

The same way the x_2 is evaluated; I use it here. That is probably one of the ways. You can set it up in various other possible ways (()) struggling. And then what I would find - that k equal to 0. We start off with again the same thing like before.

(Refer Slide Time: 24:13)

$$\begin{aligned}
 x_2^{(k)} &= 3 - x_1^{(k)} - x_3^{(k-1)} \\
 x_3^{(k)} &= 5 - 2x_1^{(k)} - 2x_2^{(k)}
 \end{aligned}$$

$k=0$: $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$
 $k=1$: $x_1^{(1)} = 1$, $x_2^{(1)} = 2$ & $x_3^{(1)} = -1$
 $k=2$: $x_1^{(2)} = -5$, $x_2^{(2)} = 9$ & $x_3^{(2)} = -3$
 $k=3$: $x_1^{(3)} = 23$, $x_2^{(3)} = 29$ & $x_3^{(3)} = -9$
 $k=4$:
 $k=5$:

↓
Diverges!

You can keep doing and you will find that this does not show any sign of convergence; so we draw the conclusion.

What happen is even for such a simple system, for a 3 by 3 system, I have got (()) one of the k per by (()) to show that your Jacobi method converges and the Gauss-Seidel method diverges. So, where do you use again? How did this happen?

(Refer Slide Time: 25:00)

$$\begin{aligned}
 x_3 = 3 &\rightarrow x_2^{(k)} = 3 - x_1^{(k-1)} - x_3^{(k-1)} \\
 x_3 = 5 &\rightarrow x_3^{(k)} = 5 - 2x_1^{(k-1)} - 2x_2^{(k-1)}
 \end{aligned}$$

$k=0$: $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$
 $k=1$: $x_1^{(1)} = 1$, $x_2^{(1)} = 3$ & $x_3^{(1)} = 5$
 $k=2$: $x_1^{(2)} = 5$, $x_2^{(2)} = -3$ & $x_3^{(2)} = -3$
 $k=3$: $x_1^{(3)} = 1$, $x_2^{(3)} = 1$ & $x_3^{(3)} = 1$
 $k=4$:
 $k=5$:

Was the previous theorem of any use? We just enunciated the Stein-Rosenberg Theorem.
What we get? Look at the property of the A matrix (())

(Refer Slide Time: 25:19)

Convergence Issues for Iterative Methods

- Condition of diagonal dominance is equivalent to having spectral radius of the matrix $[G_J]$ to be less than one and both in turn implies convergence.
- Relative magnitude of spectral radii of Jacobi and Gauss-Seidel error gain matrix has been enunciated by Stein-Rosenberg theorem, stated as:
 - If the $[A]$ matrix has the following properties,
 - (i) $\alpha_{ii} > 0, \alpha_{ij} \leq 0, i \neq j$
 - (ii) $\alpha_{ii} \geq \sum_{j=1, j \neq i}^N |\alpha_{ij}|$ with strict inequality for some i
 - (iii) $[A]$ is irreducible
- And $[G_J]$ and $[G_S]$ are the matrices of Jacobi and Gauss-Seidel iterations, respectively, then one and only one of the following mutually exclusive relations holds for the spectral radius, λ :

(49)

This one is of course satisfied. This is irreducible. You actually lose out on the second condition very quickly.

(Refer Slide Time: 25:34)

Example Method

$$\begin{aligned} x_1 + 2x_2 - 2x_3 &= 1 \\ x_1 + x_2 + x_3 &= 3 \\ 2x_1 + 2x_2 + x_3 &= 5 \end{aligned} \rightarrow \begin{aligned} x_1^{(k)} &= 1 + 2x_3^{(k-1)} - x_2^{(k-1)} \\ x_2^{(k)} &= 3 - x_1^{(k-1)} - x_3^{(k-1)} \\ x_3^{(k)} &= 5 - 2x_1^{(k-1)} - 2x_2^{(k-1)} \end{aligned}$$

$k=0 \Rightarrow x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$
 $k=1 \Rightarrow x_1^{(1)} = 1, x_2^{(1)} = 2, x_3^{(1)} = 3$
 $k=2 \Rightarrow x_1^{(2)} = 5, x_2^{(2)} = 1, x_3^{(2)} = 1$
 $k=3 \Rightarrow x_1^{(3)} = 1, x_2^{(3)} = 2, x_3^{(3)} = 3$

The diagonal terms (()) the way we have written down these are the diagonal (()) method (()) diagonal term they have magnitude dominate over the diagonal lengthily. So do not have that diagonal dominance.

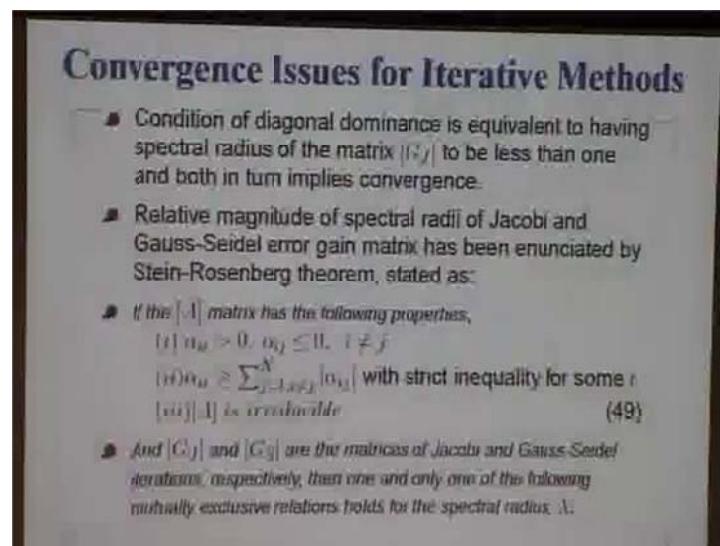
Of course, this theorem does not work. Initial condition, the starting requirement of the property of A matrix are not satisfied.

That is what we are not going to make use of the Stein-Rosenberg Theorem. We have to be aware of the fact - that is what I said that - apart from the property A, even this one will have to check this, A is the symmetric positive (())

How do you do that? If you remember how to check for that - what you do is - you take that A matrix and you work out the various sub matrix - is that you can from that right - you can get this 3 by 3, 2 by 2 and 1 by 1 matrices.

And then when you take the determinant of those matrices, they all have to be quickly has actually (()) positive

(Refer Slide Time: 26:43)



That is when you call them as a positive definite. And symmetries, of course you can (()) - this is a not at all symmetry. Of course, symmetries are also evaluated and diagonal dominances are evaluated - that is what we do not get it.

(Refer Slide Time: 27:19)

Relaxation Methods

- For the linear algebraic equation $Ax = b$, the success of iterative methods depend upon the spectral radius of the error-gain matrix.
- In the iterative methods, one usually solves:

$$N(u^{l+1}) = b + R^l \quad (13)$$
- Spectral radius of G can be altered to advantage by relaxing the right hand side, as given by

$$N(u^{l+1}) = \omega [b + R^l] \quad (51)$$
- Where ω is the relaxation parameter. If $\omega < 1$ then we call it an under-relaxation method. For $1 < \omega < 2$, we have an over-relaxation method. Commonly all these are called Successive Over Relaxation methods.
- The right hand limit of over-relaxation method is determined by Varga-Ostrowski theorem, stated later.

Then we focus our attention to other things, other ways and needs by which we can get the method work for us. And one of the way that we can make even this iterative methods work for us is by these relaxation methods.

If you recall that we started up with iterative methods by solving this alternative equation, remember we changed A to new matrix N , then we work upon the solution increment over iteration and the right hand side that is pushed by the residue.

Basically, residue is like forces, right? So, residue forces are the variation in u^{l+1} from u^l to u^{l+1} , that is driven by this quantity r .

This was observed and we figure out that the spectral radius was given by $I - A^{-1}N$; remember, that is what we got if we adopt the equation 13.

However, we could try to do something different; that is, we do not take the residue at each and every point as we have calculated. We actually - over I mean - give an overestimated role or underestimated role. So what we do is - we take the residue and multiply by parameter θ . And that is what is called as a relaxation parameter.

If θ less than 1 there of course, we are under relaxing.

Whatever the residue we have, we are taking a smaller component and forcing it to convergence, trying to converge. Or, if we take θ between 1 and 2, then we have - what we call as - over relaxation method.

How in the $((\theta))$ both this type of variants are called successive over relaxation method or what we call as the SOR- successive over relaxation method.

Now, why we choose the limit for θ to be between 1 and 2 for the over relaxation method is governed by theorem $((\theta))$ due Varga-Ostrowski which will state later

But I cannot just resist telling you the story that when $((\theta))$ Seidel is to do this kind of work in Cambridge more than 100 years ago, then he use to actually used human operator to do this sort of calculation in a room. I may have narrated this story to you before, that you would make this human operator to represent node and a grid.

And then you go through this kind of exercise what I did just now. Remember, in the Gauss-Seidel methods, I said first equation I use to calculate x_1 . And that information is required for the person who calculates x_2 ; because that was due to that operator, because you have to take the most current value, whatever is available.

What would happen? All these people would keep doing this and $((\theta))$ Seidel noted - that if you actually say - that is why the residue is maximum and in this point on residue is minimum.

I would try to change, give more weightage here and less weightage there. Try to $((\theta))$ equalize the march towards convergence by changing it.

Basically, $((\theta))$ even talking about this θ that we have given, capital data would be position dependent; someone will take more weightage, someone will take less weightage and this is to go on. That is how this method was $((\theta))$ worked out.

But when you actually approach a computer, you do not do such a thing. You try to keep θ the same for all the fact; otherwise, it will become mind bogglingly difficult, because you have to work out some kind of a strategy algorithm or you relate this θ with node location.

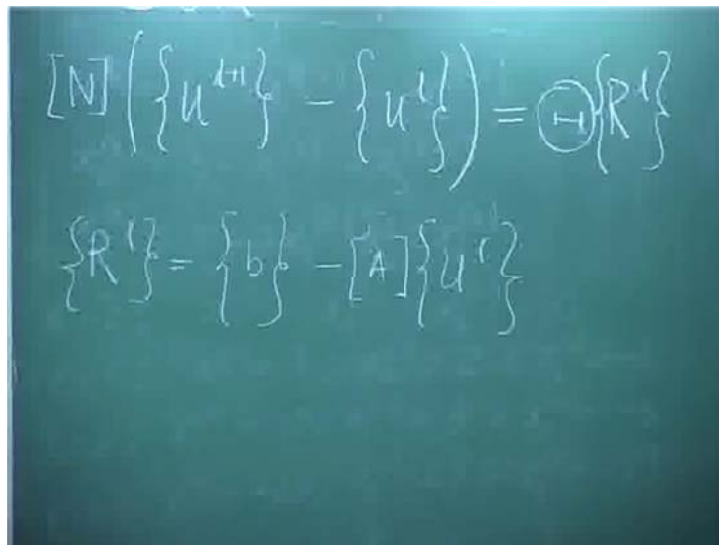
Basically (()) abandoned and this is what is correctly practiced. Well, about that story was - it had a tragic ending of that story, because a (()) very I would just turn task master. So he is to take away the salary for every mistake that someone makes.

What would happen in that room? Everybody would compute and every sort of iteration would be over; then few would oversee and you will be ok. Now, we can go to the next iteration. That is how you use to do it. And in the process you would find out that some operator would have made some mistake.

And of course, that destroys the whole flow of calculation. And so he is to dock salary; he is to take money away for making wrong calculation. And it is - say I am not sure it is just a story, just some of them would went back to the deficit at the end of the day.

That is the sad part. By anyway, this is (())a how the subject is (()) where, what computer does today; once upon a time unfortunate human beings do all that. And of course, what we can do is really fantastic compared to those days.

(Refer Slide Time: 33:39)



$$[N] \left(\{u^{k+1}\} - \{u^k\} \right) = \ominus \{R^k\}$$

$$\{R^k\} = \{b\} - [A] \{u^k\}$$

Now, suppose we adopt that algorithm that we said, that N matrix working on solution residue. And here I have put theta and this is Rl.

If I look at it, I could (()) work it out. How did you define Rl? I Forgot. Tell me how did you do that? A th minus b or b minus A - does anyone remember? I do not know.

(()) how I do it? I can plug this in here and I can simplify. What I find is that I get this new iterate u^{l+1} , should be now I minus θ times N inverse A .

Remember, when we did not adopt relaxation method. θ was 1.

(Refer Slide Time: 35:23)

Successive Over Relaxation Methods

- The linear algebraic equation: $[A]\{x\} = \{b\}$ solved by (51) can be rearranged to get,

$$\{u^{(l+1)}\} = ([I] - \theta[N]^{-1}[A])\{u^{(l)}\} + \theta[N]^{-1}\{b\} \quad (52)$$
- Thus, the error-gain matrix is given by:

$$[G_\theta] = ([I] - \theta[N]^{-1}[A]) \quad (53)$$
- Operationally, the relaxation procedure acts like a corrector stage of a predictor-corrector method as given by

$$\{u^{(l+1)}\} = \{u^{(l)}\} + \theta [\{u^{(l+1)}\} - \{u^{(l)}\}] \quad (54)$$
- Where $\{u^{(l+1)}\}$ is the predicted quantity.
- We display the Gauss-Seidel Successive Over Relaxation (GSSOR) method next.

Then we have the G matrix - I minus N inverse A . Here what has happened? By adding this degree of freedom, by introducing capital data, we have altered the gain matrix, which is now a function of θ .

Now, what we notice is that (()) gain matrix is given by I minus θ (())

This is the way (()) we would like to do the (()) frame work; obtain the G matrix like this. Operationally what we do is that is given in equation 54.

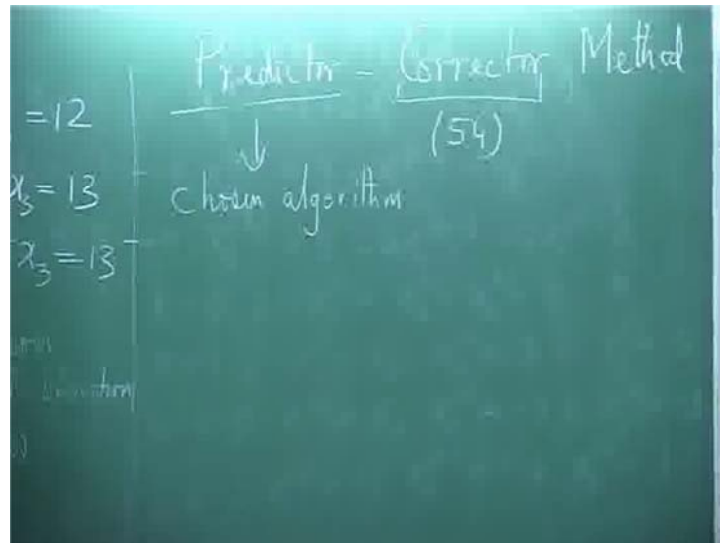
What you do is you have an algorithm. With the algorithm, you calculate \bar{u} at $l+1$. That is why - that over bar tells you that you have worked it out here.

\bar{u} at $l+1$ (()) gives surprise way.

What we notice is that this is the way we would adopt. We have an older iterate u^l . We have just now calculated \bar{u} at $l+1$. And then we look at what the solution is **change**. We multiply by capital θ with that we add it to the prior quantity.

Like what is called a corrector strategy in a predictor corrector approach. Any solution method is broken down into two steps.

(Refer Slide Time: 37:09)



You have a predictor method. You predict it the way any algorithm that you choose. Let say, it could be Jacobi, could be Gauss-Seidel - you would do that and then you obtain that. This \bar{u}^{l+1} is output of the predictor stage.

Successive Over Relaxation Methods

- The linear algebraic equation: $[A]\{x\} = \{b\}$ solved by (51) can be rearranged to get,

$$\{u^{l+1}\} = ([I] - \Theta[N]^{-1}[A])\{u^{(l)}\} + \Theta[N]^{-1}\{b\} \quad (52)$$
- Thus, the error-gain matrix is given by:

$$[G_{\Theta}] = ([I] - \Theta[N]^{-1}[A]) \quad (53)$$
- Operationally, the relaxation procedure acts like a corrector stage of a predictor-corrector method as given by

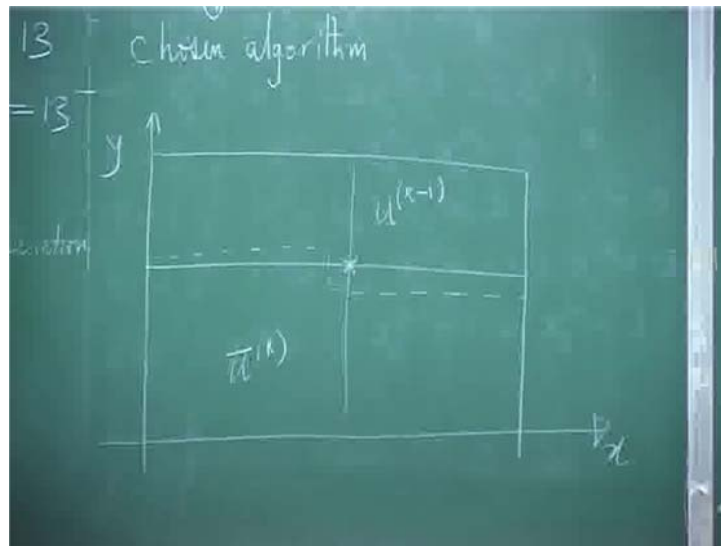
$$\{u^{(l+1)}\} = \{\bar{u}^{(l+1)}\} + \Theta \left[\{u^{(l+1)}\} - \{\bar{u}^{(l+1)}\} \right] \quad (54)$$
- Where $\{\bar{u}^{(l+1)}\}$ is the predicted quantity.
- We display the Gauss-Seidel Successive Over Relaxation (GSSOR) method next.

Now what you are doing? You are over laying an additional correction and that correction is obtained from this equation 54.

Equation 54 is your corrective stage. And this is your chosen algorithm. It is basically the method that you may like to adopt, you will do that.

Now, I have tried to show it to you. If I adopt such a method along with Gauss-Seidel point relaxation method - and I may say, (()) I have a feeling that some of these equations need modification. So please bear with me. I may actually put up the correct slides in a day or two.

(Refer Slide Time: 38:27)



However, what you notice is that suppose we are doing a Gauss-Seidel method, our approach has been like this. If I am solving the problem in a domain like this, (()) arrived at a point here, that may mean that information available (())

This is what I have; I will call that as \bar{u} level. So, (()) at the k th iterate here. Whereas on this side, I would continue to use the previous iterate, that is your Gauss-Seidel strategy.

If I do that I try to write down and I will feel that this equation may not be correct - this 55; I will have to take a look among the correct one for you.

That is - (()) you have a two dimensional array. (()) It is two dimensional but you have stacked it up as a one dimensional array. That is what we do all the time.

When I write u vector here, basically it is a two dimensional array written like a one dimensional form.

(Refer Slide Time: 39:40)

Gauss-Seidel SOR Method

- This is an example of a typical predictor-corrector method. The predictor stage is given by:

$$u_i^{(k)} = \sum_{j=1}^{i-1} a_{ij} u_j^{(k)} + \sum_{j=i+1}^N a_{ij} u_j^{(k-1)} + r_i \quad (55)$$
- The corrector stage evaluates the next iterate:

$$u_i^{(k)} = (1-\Theta)u_i^{(k-1)} + \Theta \left[\sum_{j=1}^{i-1} a_{ij} u_j^{(k)} + \sum_{j=i+1}^N a_{ij} u_j^{(k-1)} + r_i \right] \quad (56)$$
- This can be written down in matrix notation as,

$$[D]u^{(k)} = (1-\Theta)[D]u^{(k-1)} - \Theta \left[[D]^{-1}[L] \{ u^{(k)} \} + [D]^{-1}[U] \{ u^{(k-1)} \} - [D]^{-1}[b] \right] \quad (57)$$
- This can be simplified to,

$$([D] + \Theta[L]) \{ u^{(k)} \} = [(1-\Theta)[D] - \Theta[U]] + \Theta[b]$$

That is what we do. If I have now come to the i th node, the predictor stage - what I am doing when I have come to the i th node? I think that there is g_{ij} missing here, please bear with me.

There is a g_{ij} which is multiplying the use, which are already been calculated - that is what we are doing. $(())$ has been taken to the left hand side and $(())$ below that i, I write it like this. And anything that is above i, I write in the second path.

Now, $(())$ that is the way I calculate, I predict and update for all the points.

That is what we are doing. And in doing so, we are actually adopting those values which are currently available in this first set of term. There is a second set of term belonging to this part of the domain.

Following that predictor stage, we adopt the corrector stage; what we do is we do some simple waiting.

What do we do? Whatever we have predicted, we multiply $(())$ by theta - so this is this part - so this part is nothing but the equation 55 itself $(())$ 1 minus theta times that previous $(())$ accepted solution.

At that end of predictor and corrector stage, whatever I have - I call that the quantity without any over $(())$. This is the previous iterate.

So that I give 1 minus theta weightage and theta weightage to the currently calculate. And of course, you can write it down. What I have written in an algebraic equation form in 56 - you can write it down in a matrix form.

For example: u_i . What is u_i ? u_i would be simply nothing but an identity matrix operating on the u vector.

So that you just have picking the diagonal matrix. That is what you are doing. And of course, 1 minus theta is scalar quantity and u_i minus 1, I have just written it like this and theta times. This operation - now you will understand what we have done.

(Refer Slide Time: 42:27)

$$[N] \left(\{u^{(k)}\} - \{u^{(l)}\} \right) = -\{R^{(l)}\}$$

$$\{R^{(l)}\} = \{b\} - [A] \{u^{(l)}\}$$

$$x_{ij} u_j = \sum_{\substack{j=1 \\ j \neq i}}^N x_{ij} u_j = \sum_{j=1}^{i-1} x_{ij} u_j$$

If you recall what we have done, we have taken this quantity times - remember that is what we did - alpha ii into ui. And this we kept it on the left hand side. And on the right hand side what I do is I write alpha ij uj. This is going to be j equal to 1 to N. And of course, I will exclude j equal to i point - that is what we do.

This comes from where? This gives you something like your D matrix diagonal entry. If I split that A into D, L and U, then this quantity gives you - alpha ii gives you the D matrix. And this part what we have done? We have split into two parts.

(Refer Slide Time: 43:29)

The chalkboard shows the following equations:

$$N \left(\{u_i^{(k)}\} - \{u_i^{(k-1)}\} \right) = -\{R_i^{(k)}\}$$

$$\{R_i^{(k)}\} = \{b_i\} - [A] \{u_i^{(k-1)}\}$$

$$u_i^{(k)} = \sum_{j=1}^N \alpha_{ij} u_j^{(k-1)} + b_i = \sum_{j=1}^{i-1} \alpha_{ij} u_j^{(k)} + \sum_{j=i+1}^N \alpha_{ij} u_j^{(k-1)} + b_i$$

$$= [L] \{u_i^{(k)}\} + [U] \{u_i^{(k-1)}\} + \{b_i\}$$

We have written it down like this: alpha ij and uj; now j - I will go from j - equal to 1 to i minus 1. That is one part which is known. And then that would be at the current level, that is what we are doing. Here I did not write it, because I decided to write here for u in a detailed way.

And then in the second part, I will start from the right of the point in question. If i is the point in question, it will start from i plus 1 and this will take you all the way here.

This part is also done like this, but this part is not known at the current level; that is what we do.

You can see that this part - what is this part? If I write A matrix as L plus D plus U, where would this come from? If you remember, if you take a look at this, this part comes in the lower triangle part of the matrix.

Diagonal defines where I am. Anything below is the lower triangular part. Anything to the right and above belongs to the upper triangular part. That is what we have done. This part is then something like D matrix multiplying with u.

That is that. And this part I will write it as, I matrix operating on u. I will write it as u of k and there also let me write it. And this part would be $(())$.

Basically that is how we have done. D times u_k is equal to 1 of u_k plus u of u_k minus 1. Having done that of course, then what you could do? You can write u_k would be nothing but D inverse I into this, plus D inverse u into this - that is what we have written here.

You can see, there is a D inverse I part; so this bracket has not been closed. And there is a D inverse u part, that is operating at the previous iterate level.

And then we have $(())$ equal to b; we just keep on adding that. There we may actually have added here b vector. That is what we have here b and there I will also $(())$

(Refer Slide Time: 46:22)

Gauss-Seidel SOR Method

- This is an example of a typical predictor-corrector method. The predictor stage is given by:

$$u_j^{(k)} = \sum_{i=1}^{j-1} u_i^{(k)} + \sum_{i=j+1}^N y_i u_j^{(k-1)} + c_j \quad (55)$$
- The corrector stage evaluates the next iterate:

$$u_j^{(k)} = (1-\Theta)u_j^{(k-1)} + \Theta \left[\sum_{i=1}^{j-1} u_i^{(k)} + \sum_{i=j+1}^N y_i u_j^{(k-1)} + c_j \right] \quad (56)$$
- This can be written down in matrix notation as,

$$\begin{aligned} [I] \{u^{(k)}\} &= (1-\Theta)[I] \{u^{(k-1)}\} - \\ &\Theta \left[[D]^{-1} [L] \{u^{(k)}\} + [D]^{-1} [U] \{u^{(k-1)}\} - [D]^{-1} \{b\} \right] \end{aligned} \quad (57)$$
- This can be simplified to,

$$([I] + \Theta[L]) \{u^{(k)}\} = [(1-\Theta)[I] - \Theta[U]] \{u^{(k-1)}\} + \Theta\{b\}$$

It is sort of term 1 minus theta operating on D and theta operating on u matrix, and that the last one is that additive part on the b vector.

I am relaxing; I am not taking exactly the same value as the residue has been obtained - I am not. I have just added the additional degree of freedom. That comes from the previous equation that I wrote, remember.

(Refer Slide Time: 46:57)

Gauss-Seidel SOR Method

- Thus, the GSSOR algorithm is given by,
$$\{u^{(k)}\} = ([D] + \Theta[L])^{-1} \{[(1 - \Theta)D] - \Theta[U]\} \{u^{(k-1)}\} + \Theta\{b\}$$
(58)
- With the error-gain matrix given by,
$$[G_\Theta] = ([D] + \Theta[L])^{-1} [(1 - \Theta)D] - \Theta[U]$$
(59)
- Ostrowski-Varga Theorem: If $[A]$ is symmetric and $\alpha_{ii} > 0$ (for all i 's). Then $\lambda(G_\Theta) < 1$, iff $[A]$ is positive definite and $0 < \Theta < 2$.
- This is the basis for the choice of the upper limit for Θ in SOR methods.

Foundations of Scientific Computing: Chapter 10 © 2005

We have not taken what we have here. We have an algorithm by which we predict. That is what is given in fifty (()) second part. That part - I am not taking what it is calculated as it is. I am taking a fraction of it or little more of it. That is what this theta indicates.

Theta indicates whether I am taking what I have calculated - the whole of it or a part of it or more of it. That is the role of capital theta 1.

Then to that what I do is I already have some estimate of it in the previous level; I take the rest of it. This is - it is your something - like your lever rule, you may have done it in your mechanic score.

Say if I have one value here and another value here about the fulcrum, I take a balance quantity from both the side. Theta basically tells you about this relative size of the distances from the fulcrum to the ends.

Think of your 1 of the left hand point at the bar and 1 plus 1 as the right. The one that you are choosing by this method is where you are actually calculating the moment about the fulcrum.

This is exactly the same procedure. This follow is whenever you adopt some relaxation procedure. You take it in a manner that it has to be consistent. What is the consistency condition here? Consistency condition is the coefficient.

See on the equation 56, what is the coefficient of u is 1. And in the right hand side, the coefficients are 1 minus θ and θ . When I add it up again it has to become 1. So I have no other choice. I cannot just simply take 1 minus θ into θ . That will be inconsistent.

You understand that - that is the consistency condition that you have invoke, that you have taken a part of θ . If it is over estimated, then the other part has to be underestimated. So that the two together would give a quantum which is exactly what you want it to be. Of course, you can read if you find a new variable that is the easy to do.

(Refer Slide Time: 49:36)

Efficiency of Iterative Methods

- The efficiency depends upon:
 - (i) Work required per iteration.
 - (ii) Total number of iterations.
- For large k , it is seen that,

$$\frac{\|e_{k+1}\|}{\|e_k\|} \approx \lambda(G)$$
- Thus, if $\lambda(G) < 1$, then $\log_{10} \lambda(G)$ equals the number of decimal digits of accuracy gained at each step.
- The rate of convergence is thus defined as,

$$R(G) = -\log_{10} \lambda(G)$$

Foundations of Scientific Computing, Chaitin, 1995, p. 200.

That is not difficult. Having obtained the equation that we have in the last part, what I could do is I can multiply both side by D plus θI inverse. That will give me the equation 58.

That is what I have. We have done and we have just simply taken D (()). The way we defined the gain matrix is nothing but the coefficient of u of k minus 1, that is what we are doing.

See I have u of k minus 1. I am multiplying by G to get the new iterate. And that is precisely what we have here. In 58, this whole thing minus (()) gain matrix. With this we have done it.

Now what happens? You can very clearly see how θ comes into play, because by different values of θ , I could alter the Eigen spectrum of the G matrix.

For this Gauss-Seidel method, we can do that. And we can obtain the gain matrix as given by 59. And then we can fall back up on this theorem of Ostrowski and Varga.

And these also require something out of the A matrix that you have. The conditionality being that A has to be symmetric and α is positive - I just forgot to write that.

And then what will happen? This gain matrix - which is now the function of the relaxation parameter θ , so the gain matrix - will have this Eigen spectrum. The maximum of the spectral radius of that G of θ would be less than 1, if this A matrix is positive definite, and θ is bracketed between 0 and 2.

This is the driving observation that why θ has to be upper bounded up to 2 - this comes from this theorem.

Now of course, once again you find that many a times that these expectations from the property of A matrix are not obtained.

We need to look for other iterative methods. We do that. We try to calibrate the iterative methods by defining a quantifying parameter. Here we are calling it as the efficiency of the iterative method.

The efficiency of the iterative method depends upon the following two sub classes: how much work that I have to do in each iteration? That is what we are talking about - work required for iteration. That has to be compounded with how many such iteration are left $(())$ define the total work, for over all converges. And these $(())$ two together should be minimum, that is precisely what we are aiming at.

To understand how to calculate, this will be little beyond the scope of $(())$. But it is shown that if I keep on doing this iteration over many steps then we can find out the error norm at k plus 1 th stage, when divided by the error at the k th stage almost becomes given by the spectral radius.

You have actually seen this equation in some more different way. When we did that - if we recall how the error at the k th stage was related to error at the initial condition - We found out that was related to lambda to the power N.

That was there we had, say that - If it was a stationary linear iteration, we had the same set of lambda, but what happens is if we adopt the point of view that the G matrices are going to be different at different levels of iteration like what you are going to go and give it to me even for that 3 by 3 system.

Then what you find is that this spectrum of the G matrix will keep changing from iteration to iteration. But if you go over a large step, then you would find that it all balances out to the spectral radius of the equivalent stationary linear iteration.

So, basically done. You can see the importance of the spectral radius of this matrix then (()) error you are reducing for time step; that is if lambda of G is less than 1, then if I take a logarithm of that quantity, that will tell you how many decimal places that we have shifted the error tool.

That is why we define this rate of convergence as minus of log lambda to the base 10. So minus is of course, there to give it a positive flavor, because you know - I mean - if it is really convergent method and lambda will be less than 1, and log of that will be minus, so that is what over all you get a positive quantity.

(Refer Slide Time: 55:17)

Alternative Direction Implicit (ADI) Method

This is a method similar in spirit to the line iteration method with relaxation. But, it is employed by switching the direction of sweeping the solution alternately.

Consider the following elliptic PDE in self-adjoint form:

$$G(x, y)u - \frac{\partial}{\partial x} \left[A(x, y) \frac{\partial u}{\partial x} \right] - \frac{\partial}{\partial y} \left[C(x, y) \frac{\partial u}{\partial y} \right] = S(x, y) \quad (60)$$

where $G \geq 0$ and $A, C > 0$.

Most of the times, the main operator originates from diffusion and that is unbiased. Thus, it is discretized in the following manner,

$$\begin{aligned} \frac{\partial}{\partial x} \left(A \frac{\partial u}{\partial x} \right)_{ij} &= \frac{\partial}{\partial x} \left[A_{ij} \frac{u_{i+1/2j} - u_{i-1/2j}}{\Delta x} \right] \\ &= \left[A_{i+1/2j} (u_{i+1j} - u_{ij}) - A_{i-1/2j} (u_{ij} - u_{i-1j}) \right] / \Delta x^2 \end{aligned}$$

$$= [A_{i+1/2j} u_{i+1j} - (A_{i-1/2j} + A_{i+1/2j}) u_{ij} + A_{i-1/2j} u_{i-1j}] / \Delta x^2 \quad (61)$$

Now, this was the kind of scenario that people are looking at for a long time. Then came this new method which is called the alternative direction implicit method, ADI method.

This is for some of you who are from chemical engineering would be happy to note that this was done by some people working in chemical engineering. It came out in mid 50s to late 50s. Peaceman-Rachford(()) - they are the major people developing this method.

See what really happened? Now - if you look at in a - take a historic perspective; first we had this point method. In a point method also we could take a Jacobi method or we could take a Gauss-Seidel method - we have seen that.

Then we have seen that we could also solve it line by line. We have those line method. Then we have this relaxation method. So the subject is progressing. We have gone from point to line and then from point to line to relaxation methods. And then this idea occurred to some people right (()) is the boundary value problems. So all the boundaries (()) could be incorporated as quickly as possible.

See what is happening here. Here the boundary condition is slowly percolating as I am going up. Suppose I do it like this, that in 1 stroke I go from bottom to top and in the next stroke, I go from left to right.

Then what happens? I am going from this way to that way, I am bringing this (()) and(()) at 1 goal. This is being felt only when I almost come there.

Now if I (()) from left to right, this two boundary condition and this boundary condition would come (()) right at one side itself.

In this alternative direction implicit method, ADI method, the name itself suggests that it will switch directions alternatively.

Once I will go from - lets say - bottom to top, next I will go from left to right, and this is what is the basic idea.

Now, let us try to explain it in terms of an equation elliptic PDE, as given by this equation 60. And this is what is called as the elliptic PDE at self-adjoint form. Basically, a self-adjoin form comes the way this A and C appears in this equation.

You can discretize this term in a manner so that the resultant matrix - you can invoke symmetry of that matrix.

The role of self-adjoint form in solving elliptic PDE is quite central and we will start from here in the next class.