

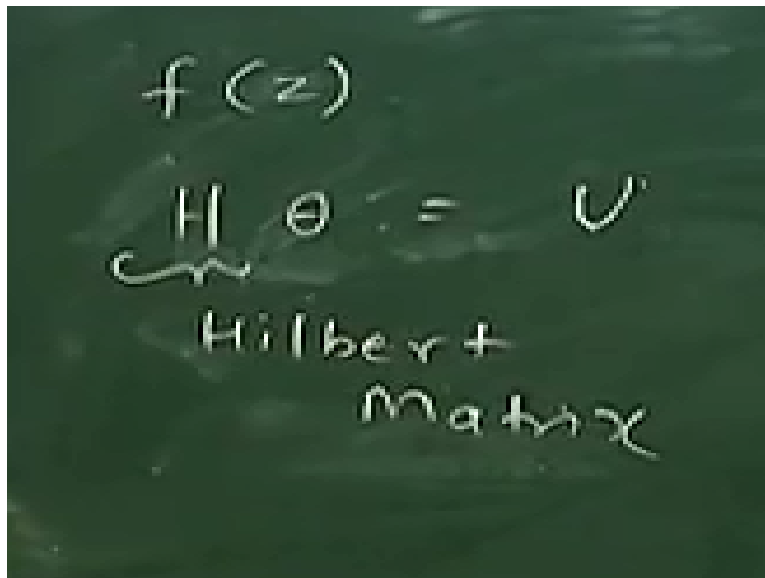
Advanced Numerical Analysis
Prof. Sachin Patwardhan
Department of Chemical Engineering
Indian Institute of Technology – Bombay

Lecture - 35

Matrix Conditioning (Cont) and Solving Non-Linear Algebraic Equations

In our last lecture we were looking at matrix conditioning. So matrix conditioning allows us to separate bad matrices from good matrices and we know when the calculations can go wrong because the matrix is bad and we are able to make a judgment on the quality of solution for linear algebraic equations.

(Refer Slide Time: 00:49)



The image shows a chalkboard with the following handwritten text:

$$f(z)$$
$$H \theta = U$$

Hilbert
Matrix

So I gave a very simple example polynomial approximations. Some continuous function we are trying to approximate of f of x . F of z is of some function which you are trying to approximate and then if you develop an approximation which is polynomial approximation, but in polynomial approximations we have shown that you get equations of the type $H \theta = U$.

So h is the Hilbert matrix. θ is the parameter vector parameters of your polynomial coefficient and U is the depending upon how you have formulated the problem. U will be a vector which is a finite dimensional vector are and this is exactly like solving $Ax=b$. X is θ , b is u and A is matrix h and what I showed you last line was that H_3 . H_3 is Hilbert matrix which is $1, 1/2, 1/3, 1/2, 1/3, 1/4$ and where I will just write it here.

(Refer Slide Time: 02:59)

$$H_3 = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}$$

$$U = \begin{bmatrix} 11/6 & 13/12 & 47/60 \end{bmatrix}^T$$

$$\theta = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$$

So this is my H matrix and then I took a solution here my right hand side was 11/6, 13/12 and 47/60 when theta=, 1, 1, 1 transpose. When theta=1, 1, 1 vector containing three 1, 1, 1 your right hand side will be this and this is the exact solution. I just showed you how things can go wrong even for this matrix for which the condition number is not so bad. The condition number we found here was 748.

So for this particular matrix condition number C infinity or C infinity of H3 was 748. I just showed you how things can go wrong.

(Refer Slide Time: 04:36)

$$H_3 = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0.5 & 0.333 \\ 0.5 & 0.333 & 0.25 \\ 0.333 & 0.25 & 0.2 \end{bmatrix}$$

$$U = \begin{bmatrix} 11/6 & 13/12 & 47/60 \end{bmatrix}^T \rightarrow \begin{bmatrix} 1.83 & 1.08 & 0.783 \end{bmatrix}^T$$

$$\theta = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \rightarrow \begin{bmatrix} 1.09 & 0.488 & 1.491 \end{bmatrix}^T$$

Instead of solving the original problem we solved slightly modified version of this problem which is 1, 0.5, 0.333, 0.5, 0.333 and 0.25. This matrix I would say is my H3+delta H3. This matrix has a slight error as compared to the original matrix H3. So if I start doing

computations with this matrix and instead of this U if I take slightly perturbed U which is 1.83 actually I have done is I have just truncated the fractions.

1.08 and 0.783. So this is my $U + \delta U$. I am just calling this $U + \delta U$ because I have truncated it I decided to truncate and write the same equation. My solution changed so drastically 1.09. So this is my $\theta + \delta \theta$. This is for a matrix for which you have condition number of 700. So actually if I try to estimate what is the fractional change that we have made on in H matrix or in U matrix.

It is of the order of order of 0.3% change, 0.3% error. I can find this error using norms. Norm of δH by norm of H or norm of δU /norm of U. I can find out what is the percentage change it is of the order of 0.3%, but the solution changes by almost 50% drastic change in the solution just because and here whatever you try to do you do maximal pivoting you do whatever reordering of the calculations you will get into trouble.

That is because this matrix is ill-condition. Now the example that I gave you earlier (08:04) might lead you to believe that it is something to do with singular matrices this is nothing to do with singularity. What matters here is the condition number. Condition number if you take in terms of 2 norm. Condition number is ratio or square root of ratio of the singular value of the matrix.

Largest singular value by smaller singular value finite angle values of $A^T A$ and take ratio of maximum eigenvalue of $A^T A$ by minimum eigenvalue of $A^T A$ find the square root that is what matters. If that ratio is small then the matrix is well conditioned that ratio is large individual eigenvalue may not be=0, but if that ratio of the largest eigenvalue of $A^T A$ to smallest eigenvalue of $A^T A$ if that ratio is large matrix is ill-conditional.

That can create problem for you using any sophisticated program or any sophisticated computer it will create a problem for you. So that is inherent problem with the matrix not the problem with the computer or the program. So there is one more example which I have given here I will not write the numbers because numbers are very small.

(Refer Slide Time: 09:37)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\kappa(A) = \sqrt{\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}} = 3.81 \times 10^{16}$$

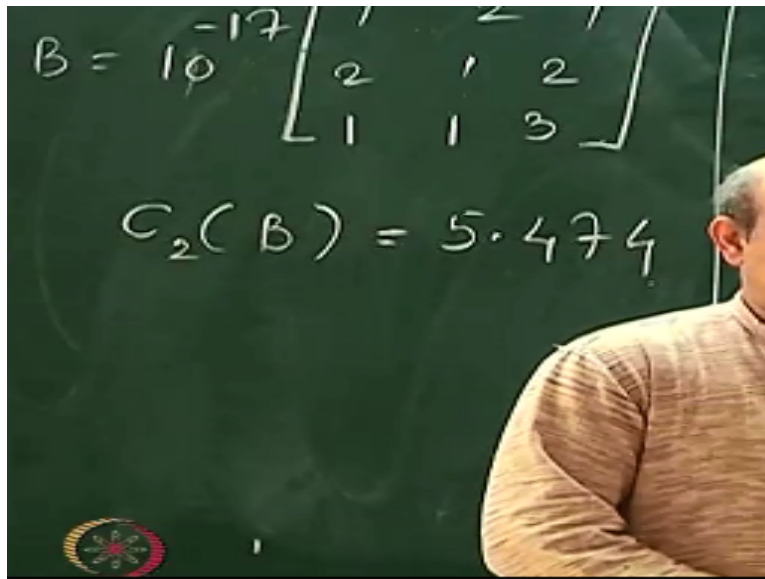
But I have tried to show here in this example is that very, very simple example. I think I demonstrated this to you this A matrix this is simple A matrix 1, 2, 3, 4, 5, 6, 7, 8, 9. This is not called by any name. This particular matrix is highly ill-condition. Condition number of this matrix it appears what is there you have just written 1 to 9 numbers at the particular sequence. The condition number of this matrix if you ask matrix the condition number that is κ_2 of A.

So which is square root of lambda max of A transpose A by lambda min. This turns out to be 3.81×10^{16} . Condition number of this matrix is very, very large. You can do a simple experiment in MATLAB or Scilab or any software take this matrix find its inverse. Well MATLAB will give you a warning. This matrix is highly ill-conditioned the result may not be reliable and you can check that.

If you find inverse of this matrix and what should happen if you find inverse of this matrix and multiply with it the matrix itself, you should get identity matrix. If you do that experiment and numerical experiment in MATLAB you will get matrix which has nothing to do with identity matrix. You will get some other matrix. You get numbers like 2, 8, 18 when you multiply $A \cdot A^{-1}$ for this matrix.

Because this is highly ill-conditioned matrix and then I have given one more example. So what I want to stress here is that inherently a given matrix is every matrix will come with its own characteristics and then that will dictate how the calculations proceed and should be able to recognize bad matrices or ill-conditioned matrices.

(Refer Slide Time: 12:38)



The image shows a chalkboard with the following content:

$$B = 10^{-17} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$
$$C_2(B) = 5.474$$

A person's shoulder and arm are visible on the right side of the chalkboard.

There is one more matrix I have shown here. Okay for this matrix if you do A inverse you can do that experiment in MATLAB you will never get identity matrix, but I will give you another matrix in 10^{-17} , 1, 2, 1, 2, 1, 2. I have taken this matrix. You might say that is it like a null matrix $10^{-17} \cdot 1$. $10^{-17} \cdot 2$ it looks like a null matrix. All the elements of this matrix are close to 0 though I have written here 1, 2, 1.

It is multiplied by 10^{-17} . Now if I do inversion of this matrix and then multiply inverse of this matrix * B matrix MATLAB will give you perfect identity matrix back. Why condition number of this matrix even though this is like a null matrix all elements are close to 0. The condition number of this matrix $C_2(B)$ it turns out to be 5.474 very well conditioned matrix no problems in the calculations.

That is because if you take B transpose B find out its maximum eigenvalue minimum eigenvalue take a ratio that will come out to be this and square root of that it will come out to be this. So this means inherently you are not going to get into any trouble when you do calculations with this matrix which is close to null matrix. Its eigenvalues are very close to 0, but that does not matter.

What matters is the condition number, what matter is the ratio of maximum to minimum singular value square root of that is what dictates the calculations. So with this we come to an end of discussion on linear algebraic equations. We looked at many things, we looked at I suppose you have learned much more than what you know about solving linear algebraic

equations as compared to your undergraduate courses on $Ax=b$.

You probably thought you knew everything about $Ax=b$ you probably thought you knew everything about $Ax=b$ right. Gaussian elimination and then you are done, but what you see here is far, far more than what you know. We looked at sparse matrix methods, efficient way of calculating. That too I just could cover a few of them just to give you a taste of what it is it is much more to it.

There is a sparse matrix toolbox in MATLAB or Scilab you know there are many routines which exploits special structure of a matrix and do fast computations. The reason for introducing sparse matrix was to sensitize you that there exist something called sparse matrix computations. So in your problem in your M. Tech problem or PhD problem when you hit upon large scale matrices try to look for sparse matrix, try to exploit sparsity if you can.

You can make your computation very, very fast. Then next thing we looked at was iterative methods like Jacobi method, Gauss-Seidel method and so on. Iterative methods in general it is difficult to prove, but in general they work faster than this Gaussian elimination particularly for large scale matrices. We looked at 2 classes of iterative methods one was Gauss-Seidel those kind of method the other one was optimization based, gradient method, conjugate gradient method and so on. So we left here 2 classes.

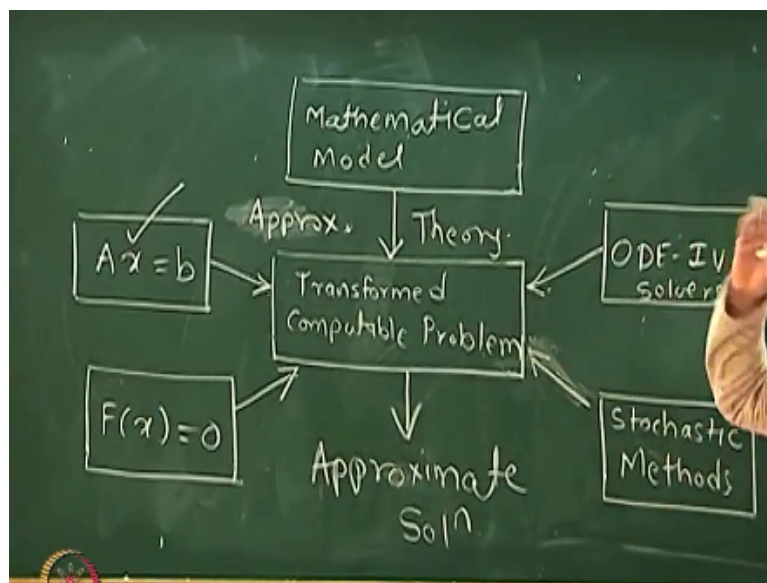
In particular, for Jacobi method and Gauss-Seidel method those kind of methods we also analyzed the convergence behavior. When are you guaranteed to converge to a solution of $Ax=b$. So we looked at convergence properties, we also looked at how to tweak my problem to ensure convergence. So now we have broadened our toolkit for solving $Ax=b$ we have many more methods now for solving $Ax=b$.

Moreover, we know what really matters in iterative methods eigenvalues. Eigenvalue you know probably unexpectedly prompt up when you try to analyze this. It is not really unexpected eigenvalues problems have come when you try to analyze linear difference equations and then we related spectral radius. We related maximum magnitude of eigenvalue of certain matrix. If it is inside unit circle we said that we are guaranteed convergence and so on.

We also looked at some theorems to ensure convergence and then finally we move to this matrix conditioning try to weed out good matrices and bad matrices. Weed out bad matrices from the set of matrices. So we know now how do judge whether a matrix is bad and that is why you are getting wrong answers or your problem formulation the strategy for computing is bad and matrix is good, but you have made mistakes.

So you know where how to distinguish between these 2. So with this you have a good idea of how to deal with $Ax=b$. Now let us move on to solving non linear algebraic equations so that is what I am going to do next. I update my notes. For this so we now start with the next tool.

(Refer Slide Time: 19:08)



Let me draw the diagram again that just to bring you back to the entire theme of this course. We have this original problem. We have this mathematical model and some problem which we cannot solve directly. So we use this. We had a mathematical model and some original problem that we wanted to solve. So using approximation theory we transform this problem to a computable form and then we said we are going to look at 4 different tools or there are 4 different approaches typically to solve this problem.

I am going to cover 3 of them. So one is solving $Ax=b$. We could be using this tool to solve this problem or I could be using $F(x)=0$. It is quite likely that to do this I might be using this Newton method I am using $Ax=b$ to solve. So this could be directly being used or it can be indirectly being used we do not know. The third tool is ODE initial value problem solver IVP solvers all this Euler method, Runge–Kutta methods.

So the third one which we are going to look at is that IVP solver and of course the fourth tool is the Stochastic tool. I am not going to look at the Stochastic tool. So this one we are done with. I am moving to this tool now and towards the end of the course we would be covering this. This will be left untouched because it probably would need one more course to cover Stochastic tools and what do you get here is the approximate solution.

This is the approximate solution for the original problem that you get. So this is done we are moving to this. Eventually we will move to this and that is end of the course. So this is the overall structure just to give you a global picture of what has been happening. So now let us move on F of $x=0$ solving non linear algebraic equations. We have already done something about this.

We have already derived Newton method starting from Taylor series approximations. You might wonder where (()) (22:13) Newton method what is there, why do I need many more things, but just like Gaussian elimination is one way of solving linear algebraic equations you have realized that Newton method is just one approach there are many ways of doing it and the reason why there are many ways of doing it is because there is no method which is NaCl. one method which works for everything.

Sometimes one approach works better sometimes the other approach works better. So you have to be ready with multiple tools and you know use appropriate tool whenever required. In some cases, you do not require Newton method. Some cases it is not possible to apply Newton method because Newton method requires Jacobian calculation. If I have 100 equations in 100 unknowns, you have seen that kind of scenario in solving partial differential equation.

Developing a matrix even if it is numerically developing a matrix 100 **cross** 100 at each iteration it is painful. It is computationally intensive and just imagine if you are trying to solve steady state stimulation of a complete chemical plant thousands of equations to be solved simultaneously. If you are trying to simulate a section of a plant many, many thousand equations non-linear algebraic equations to be solved simultaneously.

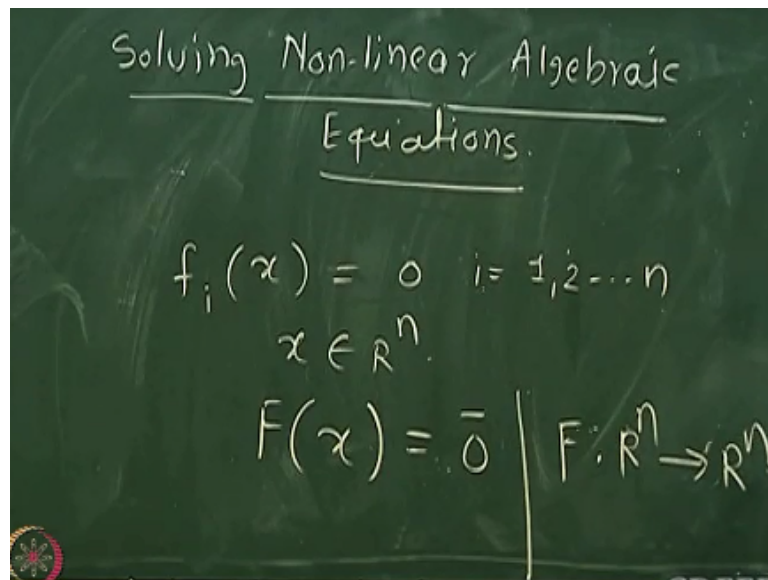
If you have to compute Jacobian even numerically it is not an easy task. So what has happened is as computers have become more and more powerful we are also trying to solve

problems which are larger and larger problem. Okay 25 years, 30 years back probably nobody thought of solving 1000 equations of 1000 unknown in the classroom now you can do it as a part of your assignment which would be probably an M Tech thesis sometime back.

So things have changed because of what we want to solve with growing power of computers also has changed. So there are problems which earlier with slow computers would take days to solve. Well, now also there are problems which takes days to solve except what you are trying now is different from what you are trying earlier. So even with very, very fast computers and very fast good software.

You still have problem which are and there is no end to this. This will just keep on growing.

(Refer Slide Time: 25:08)



Solving Non-linear Algebraic Equations.

$$f_i(x) = 0 \quad i = 1, 2, \dots, n$$
$$x \in \mathbb{R}^n$$
$$F(x) = \bar{0} \quad | \quad F: \mathbb{R}^n \rightarrow \mathbb{R}^n$$

So now let us look at different methods for solving non-linear algebraic equations. So I can now just work with abstract form because you have seen many, many example where you have to solve non-algebraic equations. So my intention is to solve $f_i(x)=0$ where i goes from 1, 2 to n x belongs to \mathbb{R}^n or now we are comfortable with a notion of a function vector or I can write this into function vector $Fx=0$ same problem $Fx=0$ where F is a map from \mathbb{R}^n to \mathbb{R}^n .

More sophisticated way of writing the same thing is that f is as function vector you are trying to look for that value of x where F of x will give you 0 vector this is 0 vector. F of $x=0$. F is a map from \mathbb{R}^n to \mathbb{R}^n N dimensions to N dimensions. So these are the kind of equations that we are interested in solving. What would be the simplest method? So first of all for solving non-linear algebraic equations except for some very, very special cases where you can solve them

magnetically.

If you remove those small set of problems where for example you can solve multiple dimensional quadric equation simultaneously. You can construct just like you can solve one-dimension quantification simultaneously. You can solve multivariable quadric equations simultaneously, but these kind of analytical solutions are very, very few. In general, even if you have a polynomial of (n) (27:25) in one variable you cannot solve it analytically.

It is very difficult to construct solutions or routes of that equation. So we need methods that can solve non-linear algebraic equations. Well one thing I would say is that if there are methods which require less computations better it is. Now first of all let us look at methods which do not require derivatives calculations. I want to solve $F(x)=0$ without having to compute derivatives or even if I have to compute derivatives I can do it in some simple way.

Rather than computing entire Jacobian. So I am going to give you up gradation of methods. Finally, we will of course move to Newton methods, but in Newton methods the problem step in terms of large computations is Jacobian calculations. So there are methods which can do what is called as a Jacobian update. Jacobian updates do not require explicit differentiation. They try to construct an approximate Jacobian by using last value of the Jacobian and adding some correction because you have moved to a new point.

These methods broadly called as Broyden's Updates or Quasi Newton method is also what we look at.

(Refer Slide Time: 29:05)

Successive Substitution

$$x^{(0)}$$

$$F(x) = 0 \longrightarrow Bx = \underbrace{Bx + F(x)}_{g(x)}$$

$$Ax = g(x)$$

$$F(x) = Ax - g(x)$$

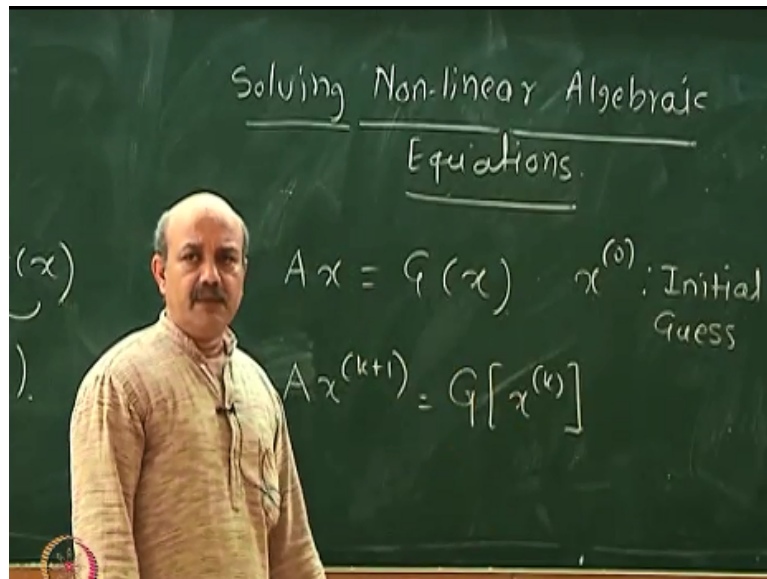
So the first method class of methods I am going to call these are known as well in iterative scheme everything is successive substitutions, but this class is also specially known successive substitution methods. What I mean here by successive substitution method is this sub class of methods by which you do not have to compute any derivative that is what I mean right now.

In general, every method that we are looking at iterative method is Successive substitution. So the question is can I arrange my calculations in such a way that you know I start with some initial guess x_0 and then I generate a new guess from the old guess. I want to solve for F of $x=0$. In some situations, in some problems for example tubular reactor with axial mixing that problem which we have been taking as a theme example throughout the course.

You can rearrange these equations f of $x=0$ into a spectral form $Ax = G$ of x where A is a constant matrix and G is some non linear function. So actually if you want to what is F of x . F of x is nothing, but $Ax - Gx$. I am trying to solve f of $x=0$ in this particular case reduces to this problem. In some cases, like tubular reactor axial mixing or some other things you might get naturally this kind of a form.

Another way of creating this form is just added x on both sides. If I add x on both sides and call this as G of x . So this is $x = G$ of x or I could do in general more some matrix here $Bx =$ so the form is same. So I can do either do this transformation or in some cases the problem discretization will yield this kind of a form depending upon what kind of structure the problem has. So you get this special form now what can I do with this.

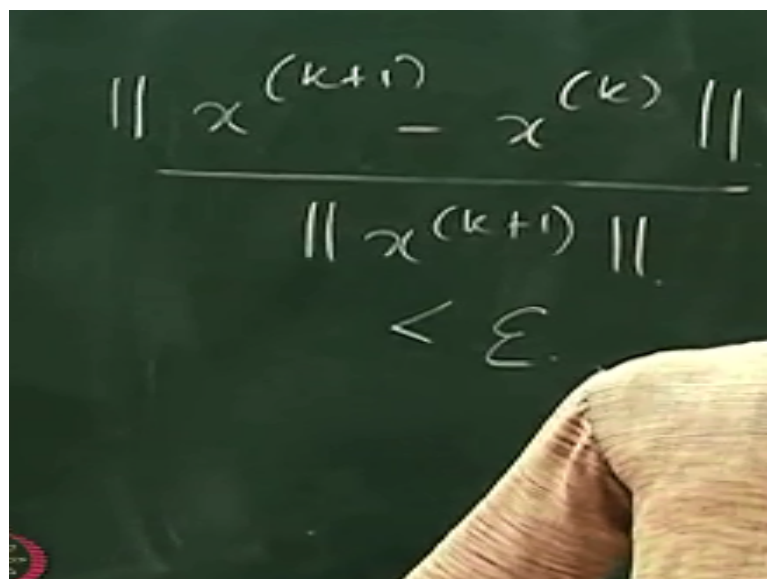
(Refer Slide Time: 32:20)



So if I have the special form $Ax=G$ of x I could convert this into solving linear algebraic equations by very, very simple trick. So if I start with some initial guess let x_0 is my initial guess then what I am going to do I am going to solve for $Ax_{k+1}=G$ of. Is everyone with me on this. See I start with x_0 if I substitute x_0 here I can compute this G of x this is known to me. What is not known to me x_{k+1} x_1 is not known to me.

But then it becomes a linear algebraic equation. This is b this is A this is $Ax=b$. I can solve for x_{k+1} then you know I can solve for x_{k+1} using any method of $Ax=b$ Gaussian elimination or Gauss Seidel method or whatever. So this will generate an iteration and when do you terminate what is the advantage of doing this. I am not computing Jacobian.

(Refer Slide Time: 34:08)



So I will terminate my iterations when $\|x^{k+1} - x^k\| < \epsilon$. So this is less than some epsilon. I will terminate when this becomes $< \epsilon$ I will terminate my iterations. This method in general it looks very simple to formulate no Jacobian you can compute. Well this method in some cases does work and when we move on to implicit methods for solving ODE initial value problems we will see merit in using this method.

What is very, very critical here is that this method will converge if you give a initial guess which is very close to the solution. When will this method converge, how will it converge we will be postponed that discussion to a later part. I will discuss about that towards the end at least I have mentioned about it. Though we cannot go too much into detail. This particular method if you give a good initial reasonable initial guess. This method will converge to the solution.

Okay generating good initial guess may not be always possible particularly for large problems. If you are solving simulation for (()) (35:33) simulation of an entire section of a plant generating initial guess is not joke it is quite difficult. So it might be difficult to use it there, but in some small problem where you can generate initial guess quite well. For example, implicit Newton method or trapezoidal rule where you can use explicit method to create a good guess for the implicit method this method will work quite well.

Now while implementing these kinds of methods I can also have variations which are similar to Jacobi method which are similar to Gauss-Seidel method which are similar to relaxation method. So I am now going to talk about variants of successive substitution method which are like Jacobi method or which are like Gauss-Seidel method. So when I do that I cannot of course use this vector matrix notation.

What we did in Jacobi we went equation by equation. So the same thing I am going to do here.

(Refer Slide Time: 36:45)

Successive Substitution

$$x_i = g_i(x) \quad i = 1, 2, \dots, n$$

$$G(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_n(x) \end{bmatrix}$$

So I go back to my original form. So instead of writing if the equation that I want to solve I am going to rearrange into this form. $x_i = g_i(x)$ for $i = 1, 2, \dots, n$ where G of x is nothing, but $g_1(x), g_2(x), \dots, g_n(x)$ a small g is nothing but one element in the function vector. I am looking at element by element converting into this form is not difficult. I can pre multiply both the sides by A^{-1} . So it will be $x = A^{-1}b$ removing A matrix is not a big deal. So now suppose I have this equation.

(Refer Slide Time: 37:55)

Jacobi-iterations

$$x_i^{(k+1)} = g_i[x^{(k)}]$$

$$i = 1, 2, \dots, n$$

Gauss-Seidel iteration

$$x_1^{(k+1)} = g_1[x^{(k)}]$$

And then how will you form Jacobi like iterations. My Jacobi iterations will be $x_i^{(k+1)} = g_i(x^{(k)})$. I am going to use the old value and create a new value for $i = 1, 2, \dots, n$. How will you create Gauss-Seidel like iterations use the new value as it gets created. So Gauss-Seidel iterations is as concept you can use it in context of linear algebraic equations, you can use in the context of non linear algebraic equations to understand the concept you can do relaxation iterations

the same ideas.

So my first equation will be $x_1^{k+1} = g_1 x_k$ is my first equation.

(Refer Slide Time: 39:11)

The image shows two equations written on a chalkboard. The first equation is $x_2^{(k+1)} = g_2 [x_1^{(k+1)}, x_2^{(k)}, x_3^{(k)}, \dots, x_n^{(k)}]$. The second equation is $x_3^{(k+1)} = g_3 [x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k)}, \dots, x_n^{(k)}]$. The equations illustrate how the value of x_2 is updated using the new value of x_1 and old values of other variables, and how x_3 is updated using the new values of x_1 and x_2 .

My second equation that is x_2^{k+1} will be g_2 . Now here I will use x_1^{k+1} . I will use $x_2^k, x_3^k, \dots, x_n^k$. Well unlike the linear algebraic equations x_2 will appear on both sides because these are non linear equations you may not be able to separate them. What will my x_3^{k+1} this will use $g_3 x_1^{k+1}, x_2^{k+1}, x_3^k$. Is this clear? See as and when the new value gets created I am using it in the next equation.

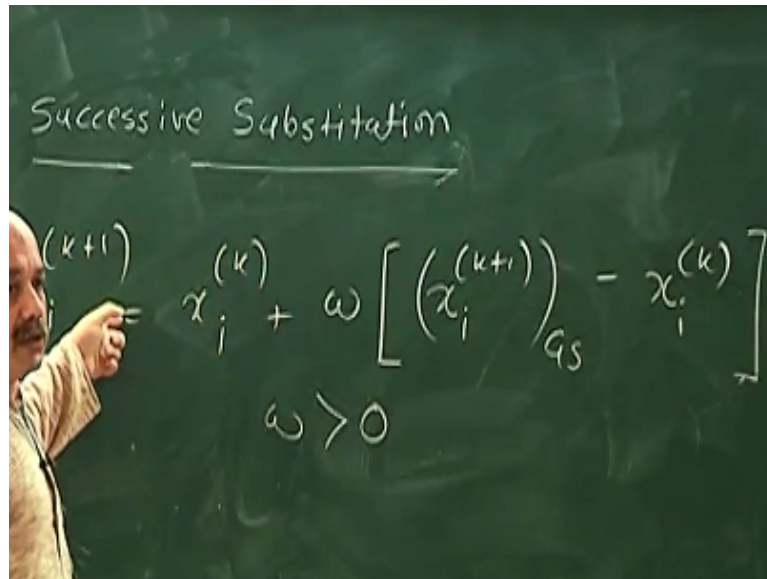
I am solving n equation, equation by equation. I am solving equation by equation one equation at a time. This would be Gauss-Seidel iteration and I can write a generic form for this for i th case to use new values up to $i-1$ and old values for i to n . So you write a generic form for this. How will you create the iteration for relaxation method? X_{nu} will be $x_k + \omega$ times the Gauss Seidel steps where $\omega > 1$ or < 1 depending upon.

Here it is difficult to say whether the convergence will occur between 0 to 2 and all that it is not possible to say here. In linear case we could say that we could give necessary sufficient conditions for convergence it is not possible to do that here. So inherently because you are using the new value every time it generated one would expect that this Gauss-Seidel iteration will converge faster than the Jacobi iterations and so on.

So these iterations would be better in terms of convergence properties. This cannot be proved,

but at least we can hope that this correction.

(Refer Slide Time: 41:55)



Successive Substitution

$$x_i^{(k+1)} = x_i^{(k)} + \omega \left[\left(x_i^{(k+1)} \right)_{GS} - x_i^{(k)} \right]$$

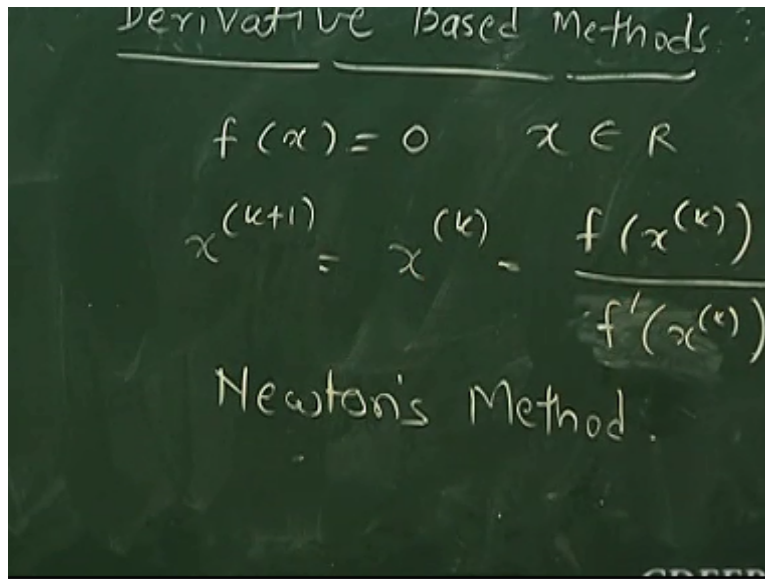
$\omega > 0$

So one can devise relaxation iterations here by saying $x_i^{k+1} = x_i^k + \omega$ times. So you make one Gauss-Seidel iteration choose that ω which is positive $\omega > 0$ and create a new guess which amplifies the change predicted by the Gauss-Seidel step and so on. So one can have all these kinds of variation here. So advantage of these methods is that no gradient evaluation no Jacobian calculations.

The flip side is that they will converge if you have a good initial guess. If without gradient calculation if you are good initial guess and if it works great. You are able to save on computation you can do solve the equations very fast. If not, you have to go for gradient based calculations. Now I want to talk about one method which is in between. This method will use gradient evaluations, but will not do the full Jacobian.

It only calculates some gradient and so this particular method is called as Wegstein method or multivariate Secant method. So let us move on to now gradient based method.

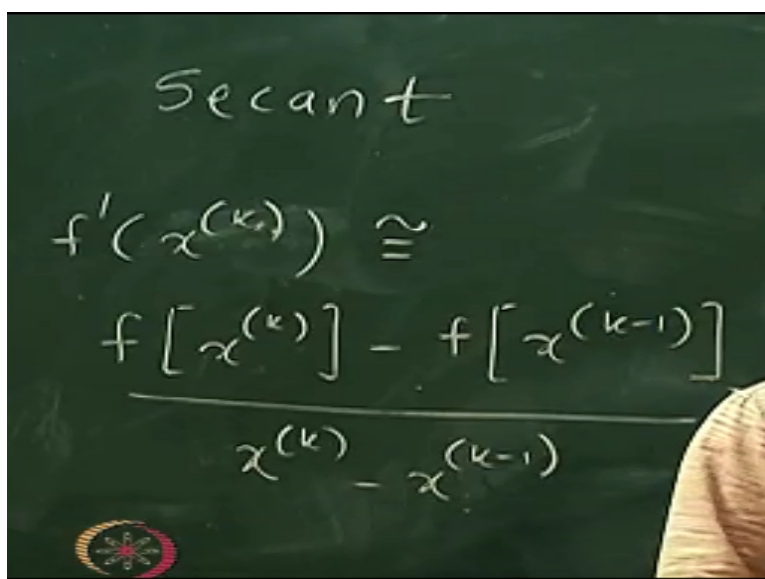
(Refer Slide Time: 44:01)



So in the class of derivative based methods we already looked at Newton method. Now I want to revisit Newton method for the univariate case. Why I am looking at this will become clear soon because I want to talk about this intermediate method called Wegstein method. So the motivation comes from univariate methods. So univariate method Newton method if I have $f(x) = 0$ where x belongs to \mathbb{R} .

I want to solve one variable equation $f(x) = 0$. Newton method of course if this function is differentiable you can write $x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$. I can write this as $x^k - \frac{f(x^k)}{f'(x^k)}$. We can have a slight variation of this method. This is the classical Newton method. In a slight variation of this method called Secant method.

(Refer Slide Time: 45:31)



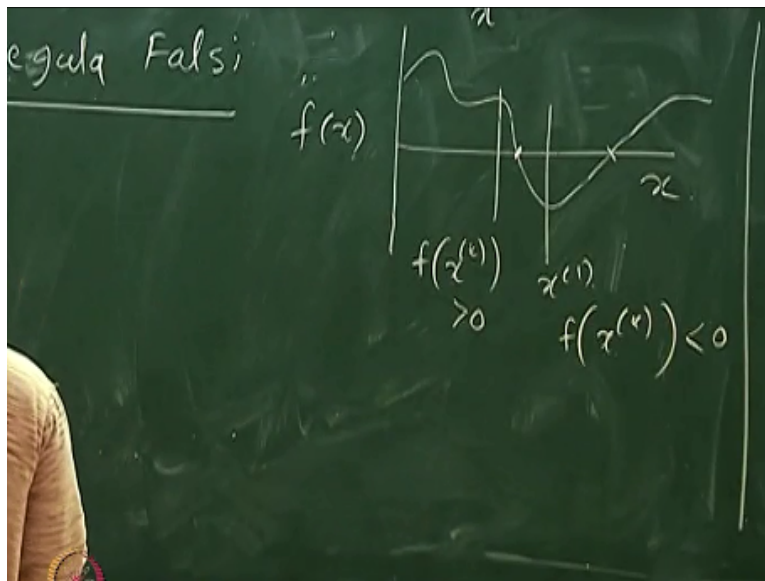
In secant method what we do is this $f'(x^k)$ we approximate this derivative using last 2

iterates. So we approximate as $f(x_k) - f(x_{k-1}) / (x_k - x_{k-1})$ this small variation is called as secant method where this $f'(x_k)$ is replaced by an approximation of the derivative. Here it is not in a true sense a may not be a good approximation because $\Delta x_k - x_{k-1}$ need not be small. So this may not be a good approximation, but this method works quite well for many simple problems.

So this variation is called a Secant method where now to kickoff the Secant method you need 2 initial guesses not 1 initial guess. x_0, x_1 then you can create the next (x_2) (46:47) x_2 starting from x_0, x_1 because this gradient calculation will require x_0, x_1 then you compute the gradient from 2 initial guesses. And then you can move on to the x_2 then x_2, x_1 you can create x_3 from x_3, x_2 you can create x_4 and so on.

So you start with 2 initial guesses x_0, x_1 create x_2 using x_2, x_1 create x_3 and so on.

(Refer Slide Time: 47:30)



There is one more variation of this method is called as Regula Falsi probably all these methods names which I am talking about right now might be familiar to you from your B Tech background because one variable Newton method, secant method and Regula Falsi are typically taught in undergraduate curriculum. The slight variation this is based on observation that if you have this is my x and I am plotting f of x .

f of x has some behavior like this I am looking for this point where f of $x=0$. I am looking for this point or I am looking for this point where f of $x=0$. I am looking for routes of the equation f of $x=0$ which means I am looking for point where f of x $(=)$ (48:23) of x becomes

0. Now one observation is that whenever there is an interval in which f of x has positive sign on one side and f of x has negative sign on the other side f of x crosses 0.

F of x is a continuous function it will cross 0 somewhere at least once it may be multiple times we did not know but at least once it crosses 0. So this Regula Falsi method actually tries to use this idea and make some modification to Secant method. So it starts with 2 initial guesses. So it will start with x_0 and x_1 such that function evaluated x_0 and function evaluated as x_1 have opposite sign.

And then as it does proceed in the calculations it tries to maintain this. It tries to maintain the fact that there are 2 successive guesses should always have function values which are opposite sign if that is maintained then convergence to the solution can be faster. So this is where f of x^k is >0 and this is where f of $x^k < 0$. If you have a scenario where you know function value changes sign from positive to negative.

This is only true for a one variable function. It is difficult to say something like this for a multivariable function. For one variable function multivariable function vector. I am talking of 1 variable scalar function changes sign at 2 different points then there is route somewhere in between that is the idea. So the modification here is that

(Refer Slide Time: 50:41)

Derivative Based Methods

Case $f[x^{(k+1)}] < 0$:

$$x^{(k+2)} = x^{(k+1)} - \frac{x^{(k+1)} - x^{(k)}}{f[x^{(k+1)}] - f[x^{(k)}]} \cdot f(x^{(k+1)})$$

I will just write down this modification here. You carry out the iterations by this formula if it is less than 0.

(Refer Slide Time: 51:06)

$$\text{Case } f[x^{(k+1)}] > 0$$

$$x^{(k+2)} = x^{(k+1)} - \frac{(x^{(k+1)} - x^{(k-1)}) f(x^{(k+1)})}{f[x^{(k+1)}] - f[x^{(k-1)}]}$$

And the second case is. So whether you use x_k or whether you use x_{k-1} then you move forward to compute the derivative approximations that will be based on the sign of f of x_{k+1} which you get. So when you go for the new iteration calculations you keep checking the sign and based on the sign you make a judgment as to how to proceed further. So this is Regula Falsi approximation.

Now what I am going to do next is use this univariate method. I am going to use univariate Secant method and create a multivariate Secant method. This multivariate Secant method is called as Wegstein method. The advantage of multivariate Secant method is that number of derivative calculation is very, very small equal to number of equations whereas in Newton method the classical Newton method you have to compute the full Jacobian and N cross and elements which can be quite large.

So if you see this software like Aspen plus they seem to be preferring this Wegstein method which works quite well which is multivariate Secant method. So we will look at it in our next lecture.