

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

**Lecture - 36**

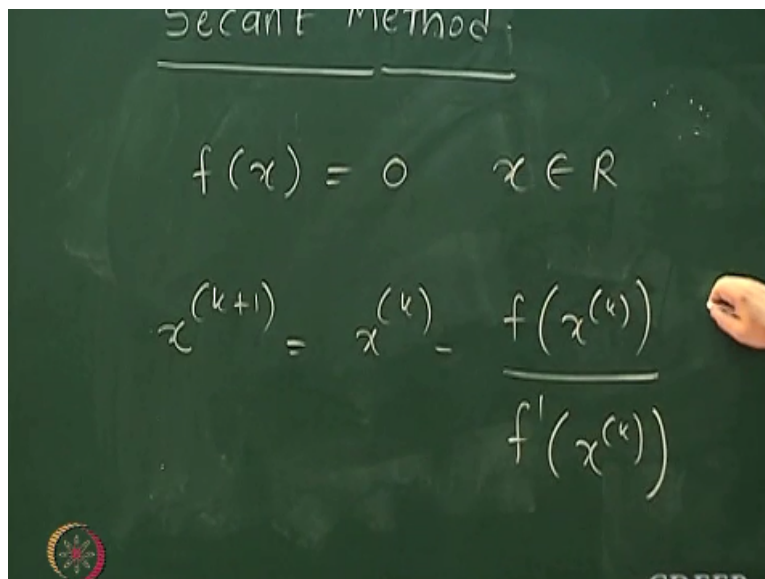
**Solving Non Linear Algebraic Equations: Wegstein Method and Variants of Newton's Method**

So we have been looking at methods for solving non linear algebraic equations and initially we looked at these successive substitutions very, very briefly. So which are derivative free methods no gradient calculation and then I move to this Univariate Newton method basically a variant called Secant method.

And then the motivation for doing this is to look at a method which is intermediate between univariate method and multivariate method. This method is called as Wegstein iterations and then we will move on to modifications of the Newton method. The derivation of Newton method we have already done using Taylor series expansion. So this is application of Taylor series expansion.

But there are many more modifications which are useful for implementing in a practical scenario. So I will discuss about them today.

**(Refer Slide Time: 01:35)**



Secant Method

$$f(x) = 0 \quad x \in \mathbb{R}$$
$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

So first is this Secant method. So Secant method I want to solve for  $f(x)=0$   $x$  belongs to  $\mathbb{R}$  univariate problem I want to solve for  $f(x)=0$  and the way this is done. So this is update rule

for the Newton method.

**(Refer Slide Time: 02:30)**

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

---

Wegstein / Multivariate Secant

$$f(x) = 0 \quad x \in \mathbb{R}^n$$

$n \times 1$   $f^n$  vector

And then in Secant method you just approximate this you replace the derivative is approximated as  $f$  at  $x$ . We approximate this instead of using exact derivative you use last two iterations and then you find the next guess. So to kickoff Secant method, you need 2 initial guesses and now what I want to do is come up with a multivariate analog of Secant method. So now my problem is so this is multivariate Secant method.

So it is known as Wegstein iterations or Wegstein method and what I am going to do now here is I am going to solve for  $f(x)=0$  where  $x$  belongs to  $\mathbb{R}^n$  and  $f$  is a  $n$  cross 1 function vector. So this is  $n$  cross 1 function vector. Let say there is some way by which we can arrange this set of equations.

**(Refer Slide Time: 04:45)**

$$f_i(x) = 0 \quad x \in \mathbb{R}^n$$

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

$$x_i - g_i(x) = f_i(x) = 0$$

$$x_i + f_i(x) = x_i$$

$$\underbrace{\hspace{10em}}_{g_i(x)}$$

So we essentially have  $f_i(x) = 0$   $x$  belongs to  $\mathbb{R}^n$   $i$  going from  $1$  to  $n$  and let say we have some way of arranging these equations as  $x_i - g_i(x) = 0$ . Okay I have some way of arranging this into this kind of a form. The simplest way could be just added if I start from this if I add  $x_i + f_i(x)$  if I rewrite it like this I could call this as  $g_i(x)$ . So basically I want to rearrange it. I want to rearrange it as some  $x_i -$  this is just adding and subtracting  $x_i$  on both sides.

So this why I am putting into this form because a particular way this method is implemented that is the reason why I am putting into this form, but what you will see soon is that this method is nothing but a multivariable analog of the Secant method. So now what I am going to do is something like this.

**(Refer Slide Time: 07:05)**

$$S_i^{(k)} = \frac{g_i(x^{(k)}) - g_i(x^{(k-1)})}{x_i^{(k)} - x_i^{(k-1)}}$$

$$x_i^{(k+1)} = x_i^{(k)} - \frac{f_i(x^{(k)})}{S_i^{(k)}}$$

$$= x_i^{(k)} - \frac{f_i(x^{(k)})}{\frac{g_i(x^{(k)}) - g_i(x^{(k-1)})}{x_i^{(k)} - x_i^{(k-1)}}}$$

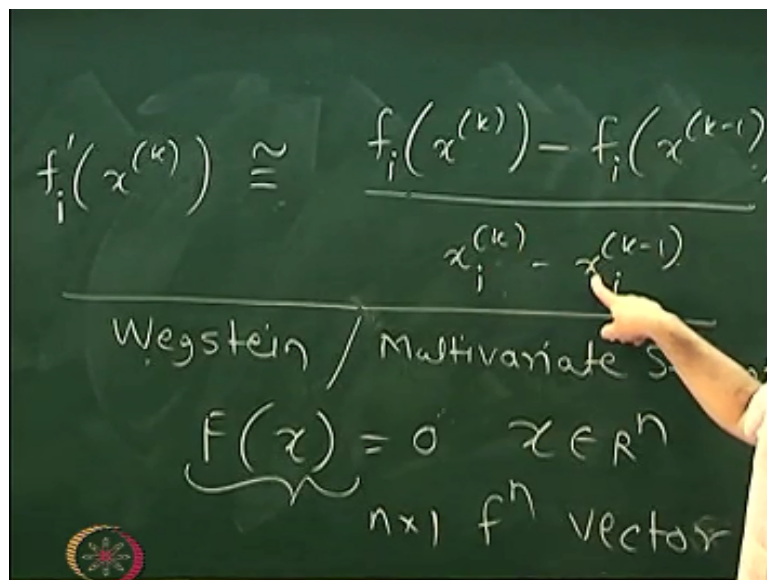
I am going to define this  $S_i$ . I am going to define this **slope** not exactly partial derivative, but

this is some kind of a crude derivative of  $g$   $S_i$  is a crude derivative of  $g$  with respect to  $x_i$  or the rate of change with respect to  $x_i$  that is how you can look at it. Now I am going to apply the Secant method to  $i$ th equation. I have these equations  $f_i(x)=0$  just for the time being keep this  $S_i$  this side.

Why am I defining this  $S_i$  will become clear soon? I am going to apply. Now look at this, this is  $i$ th component of  $x$  vector the new one is old value of the  $i$ th component + a correction which is like Secant method apply it only to the scalar function  $F_i$ .  $F_i$  is a scalar function. Capital  $F$  is the vector function.  $F_i$  is the component of the vector function  $f$ . So I am taking one scalar function  $i$ th scalar function.

And this is if you look carefully this is  $f_i(x^{(k)}) - f_i(x^{(k-1)}) / x_i^{(k)} - x_i^{(k-1)}$  upon that will become right.

**(Refer Slide Time: 09:50)**



So this derivative has been computed for the  $i$ th function with respect to  $i$ th element in  $x$  and that is how I am going to generate. If you do this is nothing but a Wegstein method if you generate iterations like this. See what is the advantage of doing this way over Newton method. First of all, you do not require explicit derivative calculations this is only an approximation. How many such derivatives suppose you call this as a derivative approximation how many such derivative approximation you need = number of equations compared with the Newton method.

You need to compute Jacobian how many elements in the Jacobian  $N$  cross  $n$ . So if you have 100 equations to solve you have to compute derivatives which are 100 cross 100 if your

thousand equation is solved you have to compute numerical derivative even if you compute it numerically this thousand cross thousand huge number of calculations whereas here you have less number of calculations.

So this is somewhere in between. It does use some kind of rate information, but not all possible rates some partial rate information is used, but not all possible rates. So this is computationally more let say attractive because it requires less calculations. So it does use some kind of  $(\cdot)$  (11:39) rate information but not fully some partial rate information is used. Now this is not the way it is normally reported or implemented.

A slight variation is done not in terms of the formula, but in terms of the way it is implemented now which is what I am going.

**(Refer Slide Time: 12:03)**

$$f'_i(x^{(k)}) \approx \frac{f_i(x^{(k)}) - f_i(x^{(k-1)})}{x_i^{(k)} - x_i^{(k-1)}}$$

$$g_i^{(k)} \equiv g_i[x^{(k)}].$$

Now this derivative is same as now  $f_i$  is  $x_i g_i$ . I am going to substitute that here. So if I substitute that here it will be  $x_i^{(k)} - x_i^{(k-1)}$ . Okay let us develop slightly simplified notations then the things will become. So let call  $g_i^{(k)}$  is defined as  $g_i$  of  $x_k$ . So that if  $g_i^{(k-1)}$  which means  $g_i$  evaluated at  $x_{k-1}$  and so on.

**(Refer Slide Time: 13:03)**

$$x_i^{(k+1)} = x_i^{(k)} - \left[ x_i^{(k)} - g_i^{(k)} \right] \cdot \frac{1}{1 - s_i^{(k)}}$$

$$x_i^{(k+1)} = x_i^{(k)} - f_i \left( x_i^{(k)} \right) \cdot \frac{x_i^{(k)} - x_i^{(k-1)}}{f_i \left( x_i^{(k)} \right) - f_i \left( x_i^{(k-1)} \right)}$$

So with this notation this derivative here I am going to write this as  $x_i^{(k)} - x_i^{(k-1)} \cdot g_i^{(k)} / x_i^{(k)}$  which is  $1 - s_i^{(k)}$  how was  $s_i^{(k)}$  defined. See this will give you 1 and this divided by this is  $s_i^{(k)}$ . So this formula which I have here now in terms of  $s_i^{(k)}$  this can be written as  $x_i^{(k+1)} = x_i^{(k)} - \frac{f_i(x_i^{(k)})}{1 - s_i^{(k)}} (x_i^{(k)} - g_i^{(k)})$ . Now what is  $f_i(x_i^{(k)} - g_i^{(k)})$  what is  $f_i(x_i^{(k)})$ . This is  $x_i^{(k)} - x_i^{(k-1)}$  we have defined it like this  $\frac{x_i^{(k)} - x_i^{(k-1)}}{f_i(x_i^{(k)}) - f_i(x_i^{(k-1)})}$  same formula I have just re-written in the different form same formula we have written in different form.

Well if you were to implement if I just go back here if you were to implement this formula as it is it is fine nothing wrong with it. That will be Wegstein method. Why I am doing this is because I want to clamp some values I want to introduce some heuristic into my iterations so that is where I am just doing this rearrangement.

**(Refer Slide Time: 15:30)**

$$= \left[ 1 - \frac{1}{1 - s_i^{(k)}} \right] x_i^{(k)} + \frac{1}{1 - s_i^{(k)}} g_i^{(k)}$$

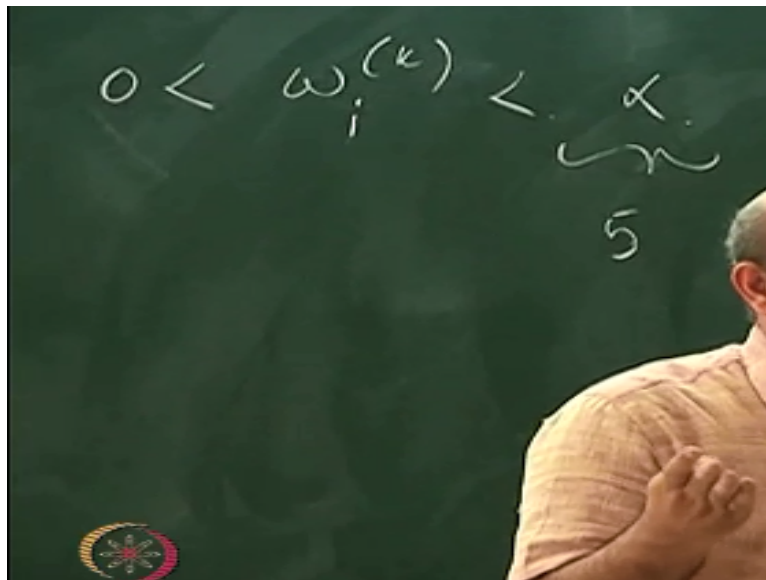
$$= \left[ 1 - \omega_i^{(k)} \right] x_i^{(k)} + \omega_i^{(k)} g_i^{(k)}$$

$$\omega_i^{(k)} = \frac{1}{1 - s_i^{(k)}}$$

So this particular equation now I can rearrange this as follows. I will just rearrange that equation nothing else just put it. Now I am going to define another variable which is  $\omega_i^k$  which is  $1 / (1 - S_i^k)$ . I am going to introduce a new variable  $\omega_i^k$  which is  $1 / (1 - S_i^k)$ . And in terms of  $\omega_i^k$  I am going to rewrite this as  $(1 - \omega_i^k) x_i^k + \omega_i^k$ . The reason why I am going this is draw some parallel with successive substitution with relaxation method this is like relaxation iterations except that  $\omega$  in relaxation is typically fixed.

Here  $\omega$  is not fixed  $\omega$  is changing with the iterations from one iteration to the other iteration  $\omega$  is changing. So time varying  $\omega$  it is like successive substitution with relaxation and now what we are going to do this is some kind of thumb rule to make your computations.

**(Refer Slide Time: 17:36)**



Typically, we tried to restrict  $\omega$  between 0 and some  $\alpha$  suggested  $\alpha$  for this  $\alpha$  is 5 typically you restrict this between 0 to 5. So you are doing something like a relaxation method, successive substitutions with variable  $\omega$ . And  $\omega$  you want to restrict between some number between 0 and 5. So all this jugglery I have done because I wanted to put this limit that is why I am doing all this jugglery otherwise I could have.

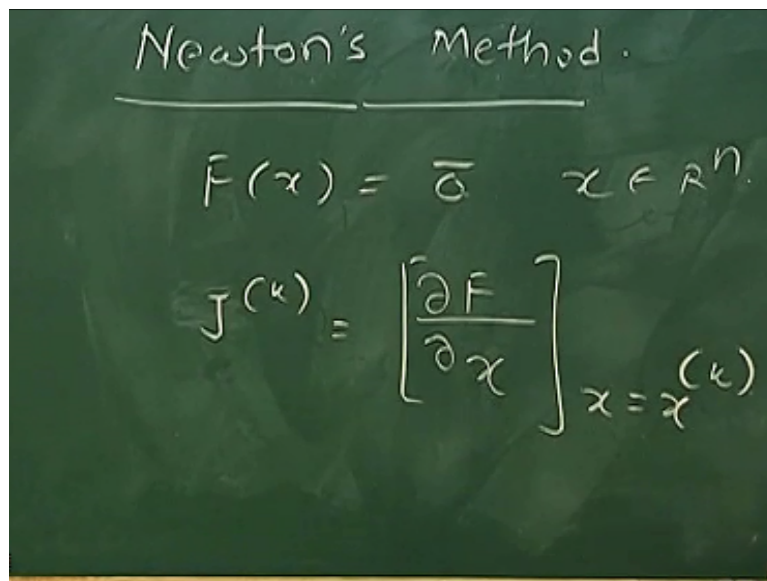
So this multivariate version of Secant method together with this limit imposed on  $\omega$  this is called as Wegstein iterations this is very popular method and if you go to many of these plant wide steady state simulation software like Aspen HYSYS you will one of the options they will give this Wegstein iterations. Wegstein iterations can be performed even if the function  $f_i$  is not differentiable you do not require differentiability here you just require

evaluation of functions at two points.

And divide it by you know you may have discontinuities when you are writing equation for some unit operation. You may have different equations in different regions depending upon the operating region and so on. If flooding occur, you may have some different equation normal operation you may have some different equation. In chemical plant you can have this kinds of situations.

So we actually many times prefer this as in between way of doing calculation completely gradient based and completely gradient free. So this is somewhere in between. So given a large scale problem what I would do is well I would first try Wegstein method if it works great. If it does work probably I should look for something else. So because this is computationally more friendly, but is this clear any doubt about this.

**(Refer Slide Time: 20:25)**



Newton's Method.

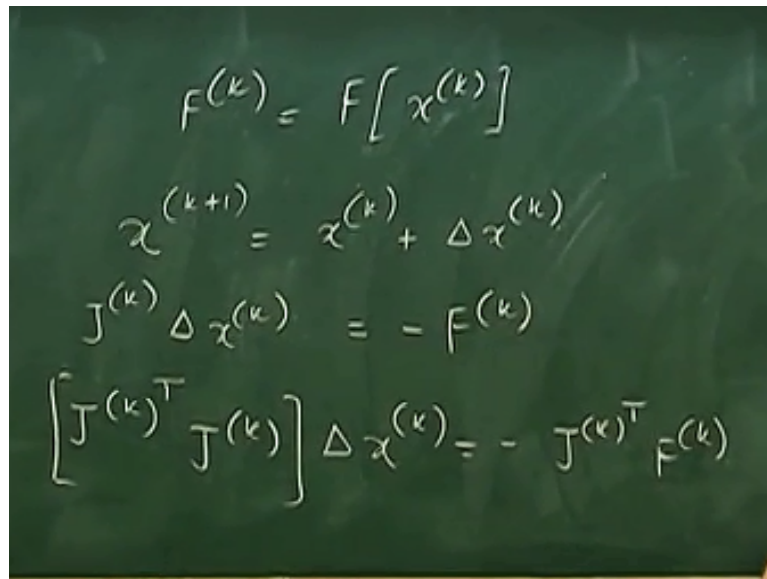
$$F(x) = 0 \quad x \in \mathbb{R}^n$$
$$J^{(k)} = \left[ \frac{\partial F}{\partial x} \right]_{x = x^{(k)}}$$

Let us move on to now the Newton method which we have derived from Taylor series approximation, multivariate Taylor series approximation then we also did some exercises in which we solved some problems using Newton methods. So you already know something about Newton method now what more is there to it. So the next one is on our radar is Newton method and you will say that we already know about Newton method.

I want to solve for  $f$  of  $x=0$   $x$  belongs to  $\mathbb{R}^n$  and then all that you do is at each time period you solve for this let us define this Jacobian at time  $k$  to make my writing simple I am going to use this terminology  $J_k$ .  $J_k$  is  $\text{d}F/\text{d}x$  evaluated at  $x=x_k$ .



(Refer Slide Time: 21:43)



The image shows a chalkboard with four equations written in white chalk. The equations are:

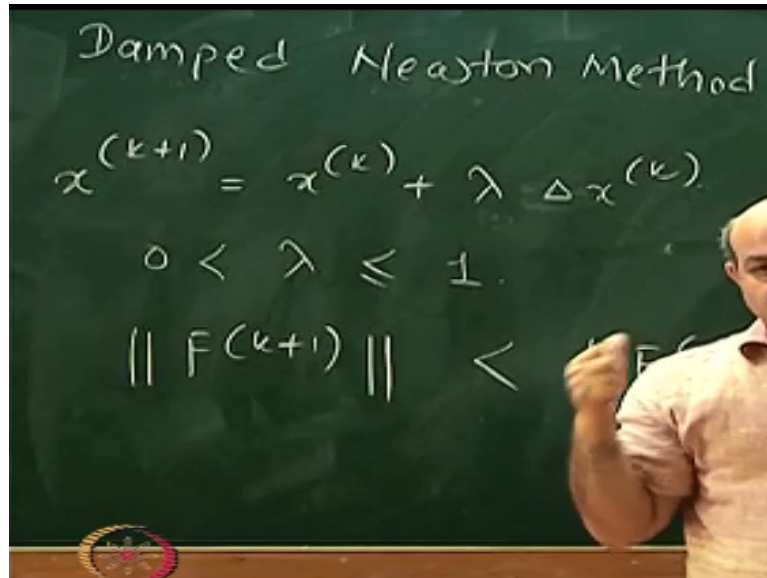
$$f^{(k)} = F[x^{(k)}]$$
$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$
$$J^{(k)} \Delta x^{(k)} = -f^{(k)}$$
$$\left[ J^{(k)T} J^{(k)} \right] \Delta x^{(k)} = -J^{(k)T} f^{(k)}$$

And then one more notation that I am going to introduce here is  $F^k$  this is function vector  $F$  evaluated at  $x^k$ . I am just introducing this notation so that my subsequent derivations become simpler in notation. Just remember that  $f$  superscript  $k$  is nothing but function  $f$  evaluated at  $x^k$ . So my formula which you have implemented is  $x^{k+1} = x^k + \Delta x^k$ . And  $\Delta x^k$  is computed by solving this linear algebraic equation.

This is what you know right now as Newton method. Well one variant which is often implemented rather than doing this is to make this equation well conditioned you pre multiply this by  $J^k$  transpose. So instead of doing this equation often this is done in fact the computing tutorial which I have given you I asked you to modify this step like this and then solve this resulting problem using Gauss-Seidel method.

What is the reason this become positive definite Gauss-Seidel method is guaranteed to converge? In general, when we work with positive definite matrices, matrices are well conditioned easier to work with positive definite matrices. So this is instead of this step we often implement this step one more modification.

(Refer Slide Time: 24:14)



This is called as Damped Newton method. So what we do here is that so one modification I told you is this. Other modification is we change this particular equation we modify it to  $x_{k+1} = x_k + \lambda \Delta x_k$ . So  $\lambda$  is chosen between 0 and 1. Now what is the reason? First of all remember when you took Newton step it was based on local linearization of function vector  $f$  of  $x_k$ .

How did we take this step it was based on the local linearization this  $J_k$  is a local Jacobian local derivative. So actually what should happen is that when you take this step the function vector should reduce because I want to go to  $f(x)=0$ . When I go from  $x_k$  to  $x_{k+1}$  this function vector should reduce. Now a step which is based on linearization may not ensure that at a new point actually the function is reducing.

Because you have done a local approximation of a non linear function make some decision of  $(\lambda)$  (25:59) based on the local approximation. This may not lead to a good value of  $x_{k+1}$ . See what should happen I should move towards the solution. Now what is the guarantee that if I make some decisions based on the local slope alone it will lead to decrease or it will lead to small value of  $f$ .

To put it in little more mathematical words see what should happen I want that  $f$  of norm of  $f_{k+1}$  this should be less than norm of  $f_k$  do you agree with me. When I take a new step if I evaluate the function vector at a new point. See I want to finally go to  $f(x)=0$ . When I make a new step so this function evaluation at a new step that is  $x_{k+1}$  should actually reduce as when compared to this.

Now this may not happen if I set  $\lambda=1$  it may not happen because  $\Delta x$  has been determined using local slope. So what is the way out? So why should we choose  $\lambda$  less than 1 let us look at our rational behind it. So essentially I chose a  $\lambda$  by some means such that this condition that is function evaluated at  $k+1$  is less than function evaluated at  $k$ . You could do this let me go back here before I move on to the rational.

So what I want to do is first I check  $\lambda=1$ . If for  $\lambda=1$  if this condition is satisfied I am happy. I accept  $\Delta x$  chose  $\lambda=1$  and proceed with the next iteration. If it does not happen okay I will reduce  $\lambda$  to say 0.9 for example I will give you a very crude way of doing it. I will reduce it to 0.9. Then for  $\lambda=0.9$   $\Delta x$  is fixed. I am not going to change.

$\Delta x$  has been formed by using the Jacobian so I want to reduce  $\lambda$ , I will reduce  $\lambda$  and check whether this condition is satisfied. If not, I will further reduce  $\lambda$  I go on reducing  $\lambda$  till this condition is satisfied. The moment I get one  $\lambda$  for which this condition is met. I will take that step and so how this is done. One algorithm for doing this is called as Armijo line search.

And I am not going to go into details of that I have given this here in the notes it is in table 1 Damped Newton algorithm with Armijo line search. So those are detailed as to how do you select iteratively  $\lambda$ . What I want to do on the (( )) (29:23) is not the algorithm as to how to select  $\lambda$  such that this is met. So that is matter of implementation I want to give the rational why this is done.

This is done this  $\lambda$  which is less than 1 is done because we are doing Taylor series approximation and Taylor series approximation is actually valid in a small neighborhood and then this  $\lambda$  actually helps us to find out what is that neighborhood where you should apply Taylor series approximation. So what I am going to do now is I am going to look at this function.

**(Refer Slide Time: 30:00)**

$$\begin{aligned}\phi[x^{(k+1)}] &= \frac{1}{2} F^{(k+1)T} F^{(k+1)} \\ &= \frac{1}{2} \|F^{(k+1)}\|_2^2 \\ &= \frac{1}{2} F[x^{(k)} + \lambda \Delta x^{(k)}]^T F[x^{(k)} + \lambda \Delta x^{(k)}]\end{aligned}$$

I am going to look at this function phi which is I am going to look at this function vector. F is my function vector, Fk transpose what is this, this is nothing but  $\frac{1}{2} F^{k+1} 2$  square nothing but norm 2 square. I am going to look at this function. Now this particular function is nothing but  $\frac{1}{2} f$  of  $x_k + \lambda \Delta x_k$  transpose actually the raw method which we have studied in the beginning and which we have implemented for some simple problems works only for simple cases.

To make it work for large size complex problems we have to do all kinds of tricks. So this is a scalar function phi is a scalar function. I am going to now expand phi in the neighborhood of  $x_k$ .

**(Refer Slide Time: 31:30)**

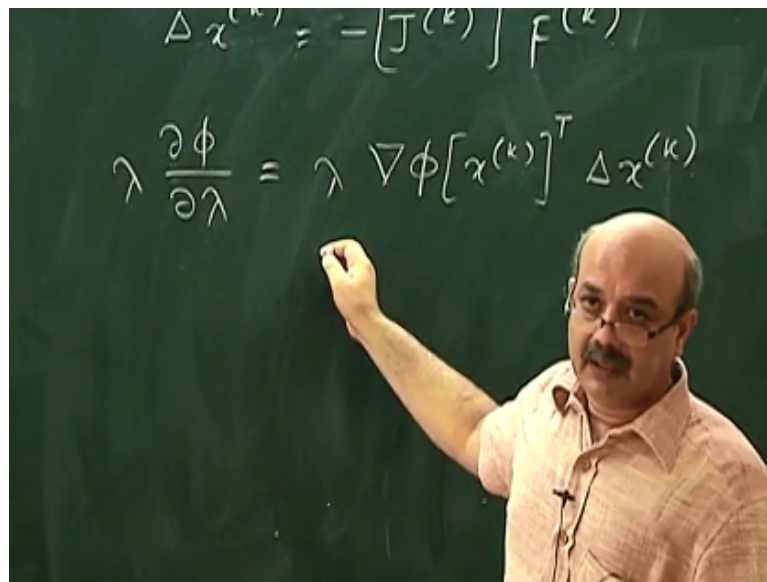
$$\begin{aligned}\phi(x^{(k+1)}) &= \phi(x^{(k)}) + \\ &\lambda \frac{\partial \phi}{\partial \lambda} + \frac{\lambda^2}{2} \frac{\partial^2 \phi}{\partial \lambda^2} + \dots \\ \phi[x^{(k+1)}] - \phi[x^{(k)}]\end{aligned}$$

So my  $\phi(x_{k+1}) = \phi(x_k)$  so this Taylor theorem is like foundation it helps you everywhere

wherever you move in applied engineering mathematics it is one of the cornerstone. Now when you are writing it like this what is unknown to you here only lambda. Delta x we have already calculated xk is known to us. So this vector is known I am just now worried about choosing lambda correctly.

So lambda dou phi/dou lambda + lambda square/2 dou 2 ph/ dou lambda square and so on. I am expanding this as a function of right. So now what should happen is phi xk-1-phi xk what should this quantity be positive or negative? It should be negative this quantity should be negative. This is nothing, but look here. This is if I were to square this and subtract okay I will this quantity just multiplying by 1/2. 1/2 is not going to make much difference 1/2 is a positive quantity. So this quantity is what I am worried about.

**(Refer Slide Time: 33:32)**



Now we know that delta xk=-Jk inverse Fk and lambda times dou phi/ dou lambda this is same as lambda times grad phi x k transpose delta x k. Just check this I am just doing derivatives by in succession. I first differentiate phi with respect to entire quantity and then so this is dou phi/dou x\* dou x/dou lambda I am not writing that just skipping the in between steps okay.