

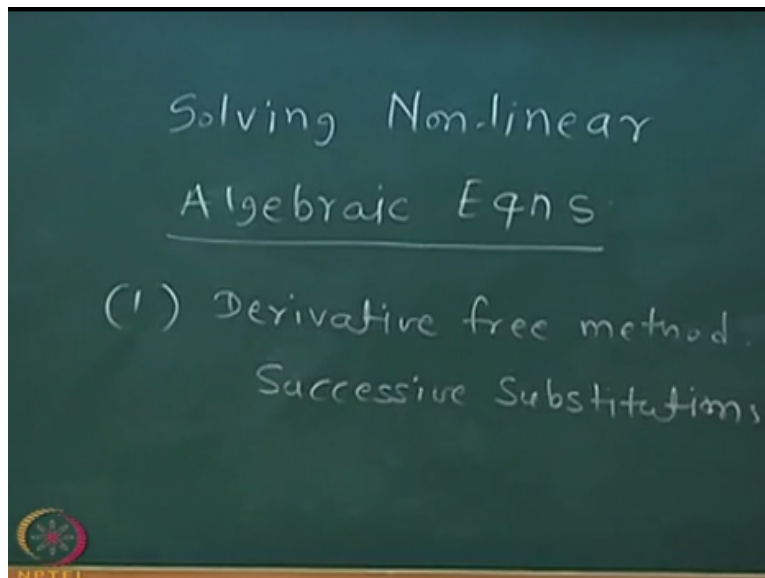
**Advanced Numerical Analysis**  
**Prof. Sachin Patwardhan**  
**Department of Chemical Engineering**  
**Indian Institute of Technology – Bombay**

**Lecture – 37**

**Solving Nonlinear Algebraic Equations: Optimization Based Methods**

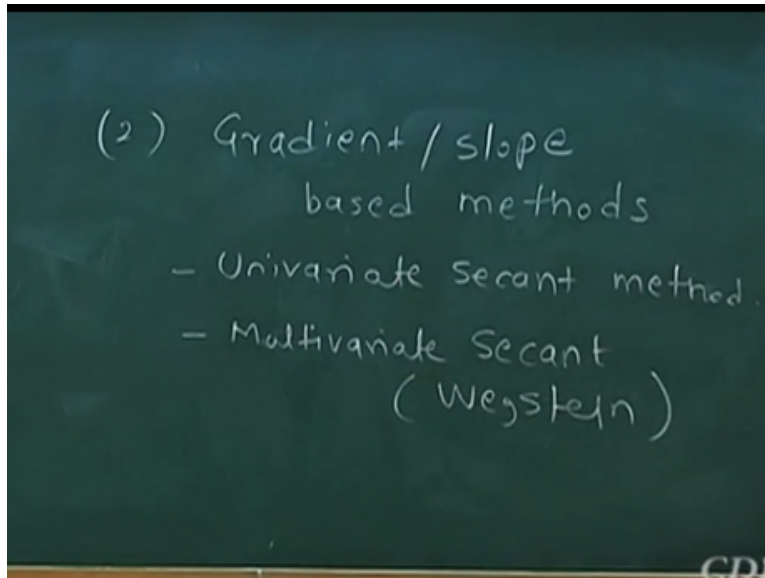
We have been looking at methods for solving nonlinear algebraic equations and in that till now we have covered gradient-free methods or successive substitutions, then gradient based methods and now I am going to move onto a third category which is optimization based methods. So, if I just collate what are the methods that we have seen.

**(Refer Slide Time: 00:51)**



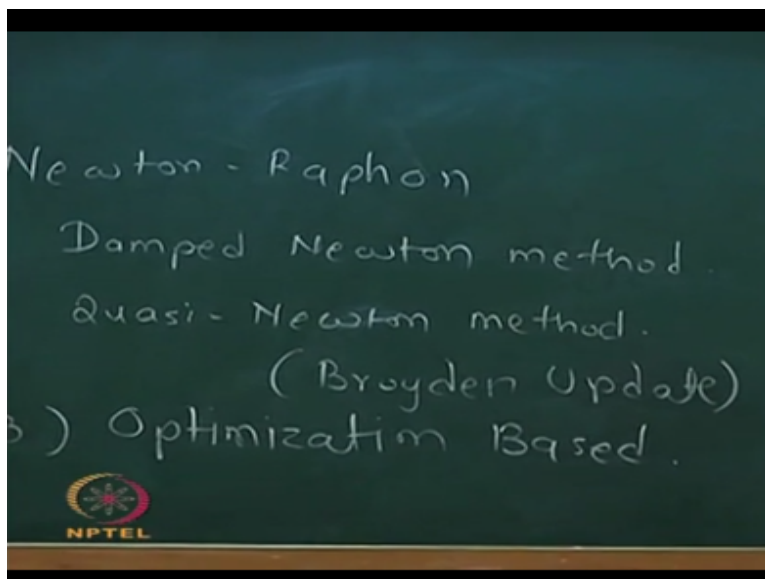
So for solving nonlinear algebraic equations, the first method that we saw was successive substitutions or here we had different variants like Jacobi iterations, Gauss-Seidel iterations and so on. Then, the second class, I would say is gradient or slope based methods and here, we have looked at univariate secant method, multivariate secant or which is popularly known as wegstien iterations.

**(Refer Slide Time: 01:20)**



So, we looked at univariate secant method which are slope based method, I would instead of saying gradient based method, I would say slope based method. It will use two consecutive iterates and then find a slope and use that slope to find out the next step. Then, we looked at multivariate secant method. Under the same category, we moved to Newton-Raphson or Newton's method, sometimes known as Newton-Raphson method.

**(Refer Slide Time: 02:29)**

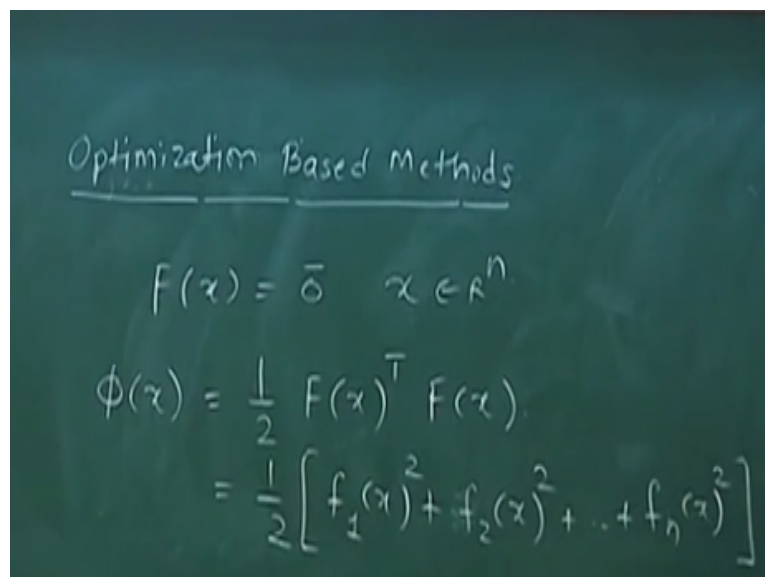


So here, we had some modifications, so damped Newton method and we looked at quasi Newton method. Damped Newton method was adjusting the step length. Quasi Newton method was Jacobi and matrix update using rank-one matrices or Broyden's update. So, this was using Broyden's update. Well, of course you can combine these two and have a Newton's method which is damped Newton method with or quasi Newton, the damped method and so on.

So, you can have all the combinations. So these are the categories on which we have looked at solving nonlinear algebraic equations. The third category that I want to look at is optimization based, okay. So first two categories we have looked at in detail, the third one the optimization based, it has its parallel in solving linear algebraic equations, linear algebraic equations we solved using gradient method, conjugate gradient method.

Likewise, here too we can form solving nonlinear algebraic equations as an optimization problem and it relatively solve the optimization problem till you reach the solution, okay. So, what are this optimization based methods, so here we form an optimization problem. See, I want to solve for  $f(x) = 0$  and  $x$  belongs to  $\mathbb{R}^n$ .  $x$  is an  $n$  dimensional vector and  $f(x) = 0$  is what I want to solve for.

**(Refer Slide Time: 04:21)**



Optimization Based Methods

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

$$\phi(x) = \frac{1}{2} F(x)^T F(x)$$

$$= \frac{1}{2} \left[ f_1(x)^2 + f_2(x)^2 + \dots + f_n(x)^2 \right]$$

I formulate an objective function  $\phi$  of  $x$  which is  $1/2 f(x)^T f(x)$ , which is nothing but  $1/2 f_1(x)^2 + \dots$ , so I have this function vector  $f$  which has components  $f_1$  to  $f_n$ . So, we want to minimize this with respect to  $x$ . So we solve the problem as minimize with respect to  $x$ ,  $\phi$  of  $x$ , okay. What is the optimum? What is the necessary condition for optimality? The gradient = 0, so  $\frac{d\phi}{dx}$  that is  $\frac{d}{dx} f(x)^T f(x) = 0$ .

**(Refer Slide Time: 05:33)**

$$\text{Min}_x \phi(x)$$
$$\frac{\partial \phi}{\partial x} = \left[ \frac{\partial F}{\partial x} \right]^T F(x) = \bar{0}$$

NPTEL

This is the optimality criteria, okay and you can see that when the Jacobian is nonsingular at the optimum, only way you can get the necessary condition satisfied is when  $f$  of  $x = 0$ . If this is Jacobian is nonsingular, then only way you can get the solution are to this problem of you know the necessary condition being satisfied, when  $f$  of  $x = 0$ , so this is when you reached the optimum and when you reach a stationary point where the Jacobian is nonsingular, then here is the optimum

Then, here is the solution of  $f$  of  $x = 0$ , that is the idea. Now, how do you solve this? How do you, well we do it iteratively using different numerical optimization methods. The first one of course the simplest one we used would be gradient method, but as I told you the gradient method has a problem. So, the simplest would be gradient method. This is as I said gradient method is more useful for deriving more complex method as forms the basis.

**(Refer Slide Time: 07:19)**

Gradient method

$$x^{(k+1)} = x^{(k)} + \lambda_k g^{(k)}$$

$$g^{(k)} = -\left[\nabla \phi(x)\right]_{x=x^{(k)}}$$

$$= -J^{(k)T} f^{(k)}$$

Probably, we would not use gradient method. We use conjugate gradient method, but just to state the algorithm, gradient method would start with the initial guess  $x$  not and then you will get  $x_{k+1} = x_k + \lambda_k g_k$  and  $g_k$  is  $-\text{grad } \phi(x)$  that is evaluated at  $x = x_k$  which is nothing but if we develop a notation, I just do it here. So, if I develop a notation  $J_k$  which is  $\text{dof by dof } x$  evaluated at  $x = x_k$ .

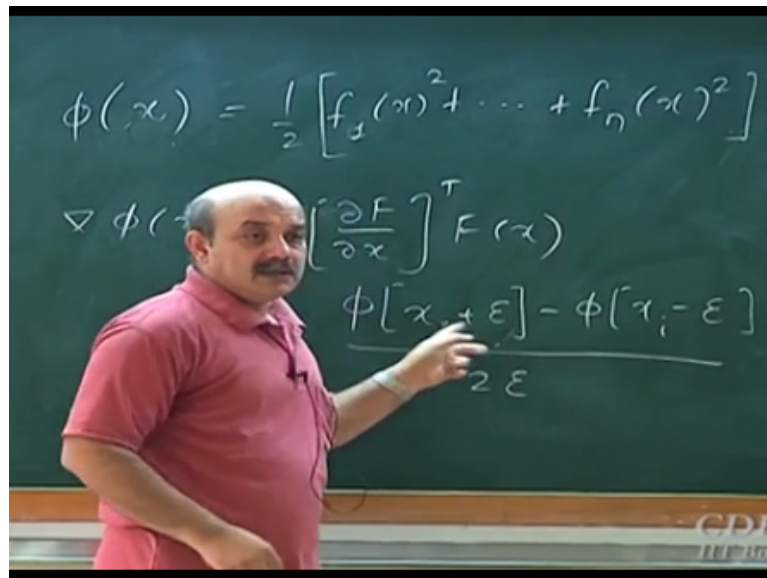
If I do this notation and then of course  $f$  at  $x_k = f$  superscript  $k$ , if I use this notation, we have use this notation earlier, the same notation I am using, okay. Then, this would be  $-j_k \text{ transpose } f_k$ .  $J_k$  would be how do you compute  $\lambda_k$ , this is a scalar parameter, step line parameter and this is found by one dimensional minimization, okay. So, this is the iterative numerical procedure by which you start from  $x$  not.

You generated new guess in the direction of negative of gradient direction and at each time, how much to move is decided by minimizing with respect to  $\lambda$ , okay. So  $\lambda_k$  is min with respect to  $\lambda \phi(x_k + \lambda g_k)$ , okay. So,  $\lambda_k$  is just one dimensional minimization,  $g_k$  is the gradient direction which you are computed, okay,  $x_k$  is known to you.  $G_k$  is known to you.  $\lambda$  is not known which is found by this.

So, of course more popular method are more better way, more computationally efficient way is to use conjugate gradients, okay. So in fact in matlab, when you are solving simultaneous nonlinear algebraic equations, there is a subroutine called `fsolve`. I think `fsolve` is not Newton-Raphson, it is optimization based solver. It actually forms an optimization problem and minimizes by iterative search, okay.

Let me confirmed, but I think fsolve is not a Newton-Raphson solver, it is a optimization based solver. So, the reason is that even though we have written here in terms of, the gradient direction is in written in terms of Jacobian and multiplied by  $f_k$ , actually if you are numerically compute gradient of  $\phi(x)$ , this numerical gradient computation does not involve explicitly computing Jacobian.

**(Refer Slide Time: 11:32)**



See this is a function  $\phi(x)$  is  $1/2 f(x)$  square to, right. So, if I want to compute this is  $\phi(x)$ . If I want to compute the numerical Jacobian of this, okay, I need to compute function vector and take its norm, okay. I do not have to explicitly compute Jacobian, okay. See computing the numerical or computing gradient of this is equivalent to, you understand what I am saying.

See computing  $\text{grad } \phi(x)$  is equivalent to  $\text{dof by do } x \text{ transpose } f(x)$ , these two are equivalent, but numerical computation of this, if you part of this function vector. See what I do for, how do you compute numerical Jacobian or numerical gradient? We have a subroutine which we have written, right in the programming tutorial. We perturb each element and by  $+\epsilon$ ,  $-\epsilon$  and then find out.

So, see basically what I do is to compute, if I want to compute  $\text{dof } \phi / \text{dof } x_i$ , okay. What we do is, we approximate this as  $\phi(x_i + \epsilon) - \phi(x_i - \epsilon)$ . I am just perturbing the  $i$ th element, remaining elements are same, okay divided by  $2\epsilon$ , right. This is how we find out and of course, here I have not written remaining elements of  $x$ , this  $\phi$  is a function of remaining elements of  $x$ .

Just to show you that, so doing this you know function calculation, does not involve anywhere explicitly computing Jacobian. You see the advantage, it only that this is equivalent to this, okay. So, actually numerically when you are computing, you do not have to go through this root. You do not want to compute Jacobian, okay. So, there are advantages in using optimization based search because you do not have to compute Jacobian explicitly.

For a large scale problem, okay you are not required to compute 1000 cross 1000 matrix that is not required, okay. Is this clear? Okay. So, what about conjugate gradient, okay. So, of course we compute the conjugate gradient of course requires the current gradient information. So, we need this  $g_k$ , which is negative of  $\text{grad } x$ , gradient of  $\phi$  with respect to  $x$ . This is the direction, so this we have to compute at  $x = x_k$ .

**(Refer Slide Time: 14:55)**

The image shows a chalkboard with the following handwritten equations:

$$S^{(k)} A S^{(k-1)} = 0$$

$$A = I$$

$$S^{(k)} = \beta_k S^{(k-1)} + g^{(k)}$$

$$\beta_k = - \frac{[g^{(k)}]^T S^{(k-1)}}{S^{(k-1)T} S^{(k-1)}}$$

And what we do next of course is find the conjugate search direction. Now, what was the idea in generating conjugate directions. The idea in conjugate direction was given some matrix  $A$ , we said that two search directions are  $A$  conjugate, okay. When  $S_k^T S_{k-1} = 0$ , even though  $A$  matrix which is positive definite matrix, okay. We call search direction  $S_k$  and  $S_{k-1}$  as  $A$  conjugate.

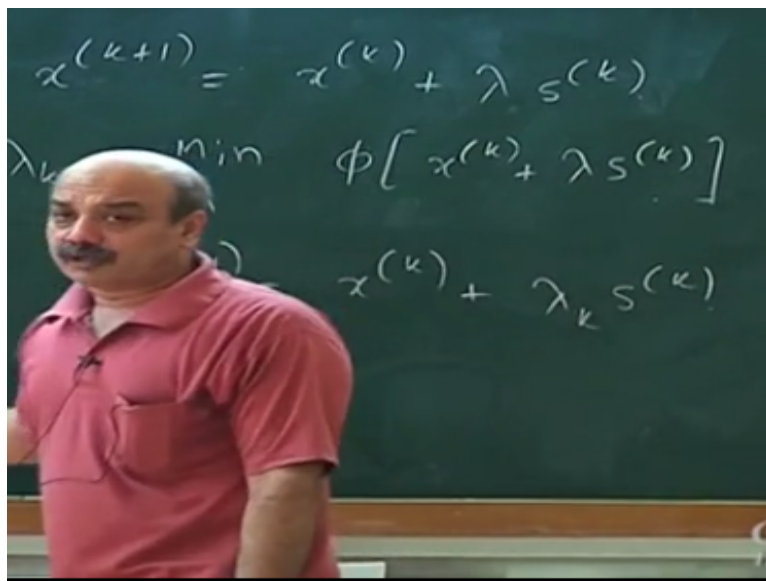
When you take any two successive directions, you should have this property, okay. So, what is done in the conjugate gradient method here is you find out the search directions such that  $A = I$ , okay. So, we find out the search directions such that the alternate directions are



perpendicular to each other, okay. So, we find out  $S_k$ , this = beta k times  $S_{k-1}$  +  $g_k$ ,  $g_k$  is the gradient which we are calculated currently and beta k is given by  $-g_k$  transpose, okay.

And the remaining part is similar to the gradient method that is once you know the search direction, then you find out  $x_{k+1} = x_k + \lambda_k s_k$ , okay. Find out  $\lambda_k$  which is minimization with respect to  $\lambda$  phi. Do this one dimensional minimization, okay. So, just to find out the step length, see this beta k which is computed here. This is only for finding out the new search direction, which is conjugate with respect to identity matrix, okay.

**(Refer Slide Time: 17:46)**



So, this is direction computation. Once you compute the direction  $s_k$ , how much to move in that direction, see one interpretation of this conjugate directions is that it is linear combination of previous gradients. Because you pick of this calculation using, so you at 0, you set that is  $g$ , so your  $S_0$  corresponds to  $g_0$ . The first direction is same as the gradient direction.

Next time, okay  $S_1$  is beta 1  $g_0$  +  $g_1$ , okay. So linear combination of  $g_0$  and  $g_1$ , okay. So linear combination of  $g_0$  and  $g_1$ , when you go to  $g_2$ , it will be linear combination of  $g_0$ ,  $g_1$ ,  $g_2$ , see because  $S_1$  is linear combination of  $g_0$ ,  $g_1$ , okay. So,  $S_2$  turns out to be linear combination of  $g_0$ ,  $g_1$ ,  $g_2$ . So, it is like saying instead of using current, so once I get this direction here, okay, then I know that I move in this direction.

This direction is linear combination of past all gradients, see it is like just imagine you know you are going down the valley, okay. Now, what you mean by using gradient step, you are

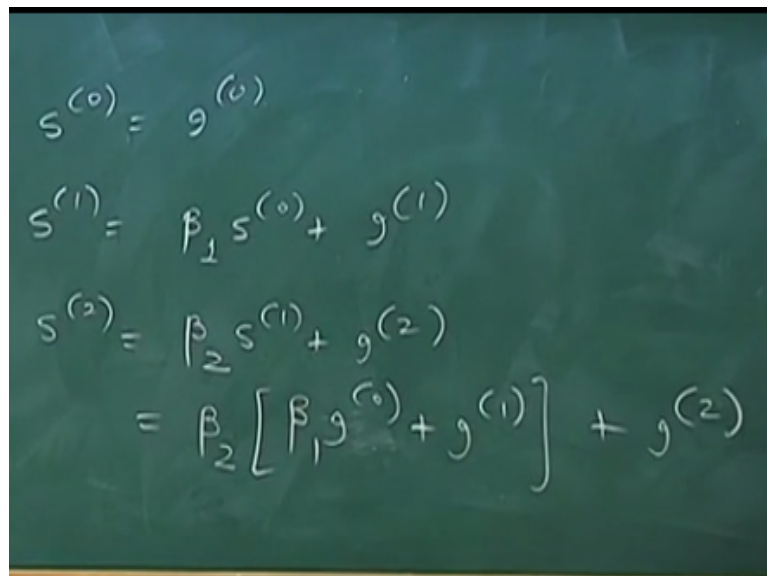


looking at the current slope, okay. Now, you just looking at the current slope could be you know too local, you do not have information about what does happen in the past. If you take it to consideration somehow, okay.

If you make your next move based on the information about the past slopes, okay, then your move will be much better than just basing your decision on the current slope. Because if you take it to consideration past slopes, in some way you are taking into consideration the curvature, right. You are taking consideration the curvature along the path and then making your decision.

See here if you look at how this conjugate gradient method proceeds, so the step here would be you know  $x_{k+1}$  will be  $x_k + \lambda_k S_k$ . So, if I look at the progression of how the direction changes, see this will be  $S_0 = g_0$ , the first time whatever the gradient that you get, negative of the gradient direction, okay. What is next time?  $S_1 = \beta_1 S_0 + g_1$ , negative of the gradient direction at a new point, okay.

**(Refer Slide Time: 21:20)**


$$\begin{aligned}S^{(0)} &= g^{(0)} \\S^{(1)} &= \beta_1 S^{(0)} + g^{(1)} \\S^{(2)} &= \beta_2 S^{(1)} + g^{(2)} \\&= \beta_2 [\beta_1 g^{(0)} + g^{(1)}] + g^{(2)}\end{aligned}$$

What is  $S_2$ ,  $\beta_2 S_1 + g_2$  which is same as  $\beta_2$ , you see this. So, as you progress you are actually basing your move direction, based on what just one gradient, but a history of gradients, okay. That is why conjugate gradient direction you know tends out to be more powerful, it moves much faster even when you go close to the optimum as compared to the, what is the problem with the gradient method when you go close to optimum, it tends to become slow, okay.

Whereas here, the move is based on past history of gradients, okay. So, it is like when you go down a hill, you are trying to make use on the curvature other than trying to make use on the local slope. The gradient method will just look at the local slopes, okay. Gradient method will only look at negative of gradient or  $g_0, g_1, g_2, g_3, g_4$ , okay. So that is why conjugate gradient method is more powerful.

Now, there is one more variant. Here in the gradient method, you are only using the local gradient information, right. Now, if you can generate something more, okay. If you can generate Hessian, what is Hessian the second derivative of the objective function. If you generate Hessian, okay then your moves are much better because you are using more information about the local curvature than just the gradient, okay.

Hessian computation would require second derivatives to be calculated and then you know the optimization methods that you get which use Hessian to generate the search direction, these are called as Newton's optimization method, okay. So, we will just briefly look at this Newton's method. Now, do not confuse the Newton-Raphson method and Newton method here. This is optimization based method you know that is direct substitution kind of method.

So this name appears similar, but, so under the category of optimization, we have again Newton's method and quasi Newton method. So, the Newton and quasi Newton methods, so now, okay, so what did we start with, we started by saying that to solve  $f$  of  $x = 0$ , we form this objective function  $\phi$  which is  $\frac{1}{2} f^T x$  transpose  $f x$ , right and what is the necessary condition for optimality?

**(Refer Slide Time: 24:37)**

Newton + Quasi Newton

$$F(x) = \bar{0}$$

$$\phi = \frac{1}{2} F(x)^T F(x) = \frac{1}{2} \|F(x)\|_2^2$$

$$\nabla_x \phi = \left[ \frac{\partial \phi}{\partial x_1} \quad \frac{\partial \phi}{\partial x_2} \quad \dots \quad \frac{\partial \phi}{\partial x_n} \right]^T = \bar{0}$$

Necessary condition for optimality is  $\text{grad } x \phi = 0$ , right. This should be  $= 0$  vector, the gradient at the stationary point, the gradient of this objective function should be  $= 0$  vector, okay. This is the nonlinear algebraic equation, okay. Now, this nonlinear algebraic, because what is  $\phi$  is the scalar function,  $\phi$  is nothing but  $1/2$  norm  $f(x)$ ,  $2$  norm square. This is nothing but  $2$  norm square, norm is a scalar, okay.

Objective function is a scalar objective function. This is the scalar objective function. So, what is this  $\text{grad } \phi$ , is it a vector or a matrix. It is a vector; it is not a matrix. This  $= \text{d} \phi / \text{d} x_1 \text{ d} \phi / \text{d} x_2 \text{ d} \phi / \text{d} x_n$  transpose, this  $= 0$  vector. So, this is actually because  $\phi$  is a scalar, it is gradient with respect to  $x$ , okay is a vector. This vector, I want to said  $= 0$ , okay.

If I solve this equation exactly, then I will get the solution, okay, but my problem is nonlinear, I cannot solve this exactly, okay. What I decide to do is, instead of solving this equation exactly, I decide to solve this using Newton's step. So, to use Newton's step what I am going to do is, I am going to linearize this equation, okay. **"Professor - student conversation starts"** exactly compute.

How will I get  $f(x) = 0$ ? (()) (27:25). No, no, no, if analytically this  $\text{grad } x \phi$  is  $\text{d} f / \text{d} x$  transpose  $f(x)$ , okay. So, if Jacobian matrix is nonsingular, then this  $= 0$  means, this will happen only  $f(x) = 0$ , okay. So, when you said this  $= 0$ , if this is nonzero or this is nonsingular, only way you will get  $0$  here is when  $f(x) = 0$ , this vector  $= 0$ , that all you get

the solution. Is the same? Is getting the solution is same, okay? **“Professor - student conversation ends”**.

So, now I want to come up with Newton step here. So what I am going to do here is something like this, I will continue here on this side, okay. So what I need to do here is grad phi x, okay. I am going to write this as grad phi x<sub>k</sub> +, okay and use our good old tell us its expansion, okay. So, this is approximately = grad phi x<sub>k</sub>, okay + del square phi, if I do realize why this is called Newton’s method.

**(Refer Slide Time: 28:31)**

$$\begin{aligned} \nabla \phi(x) &= \nabla \phi [x^{(k)} + x - x^{(k)}] \\ &\approx \underbrace{\nabla \phi [x^{(k)}]}_{\substack{J^{(k)T} \\ f^{(k)} \\ = 0}} + \underbrace{[\nabla^2 \phi(x^{(k)})]}_{\substack{H^{(k)} \\ (n \times n)}} \Delta x^{(k)} \end{aligned}$$

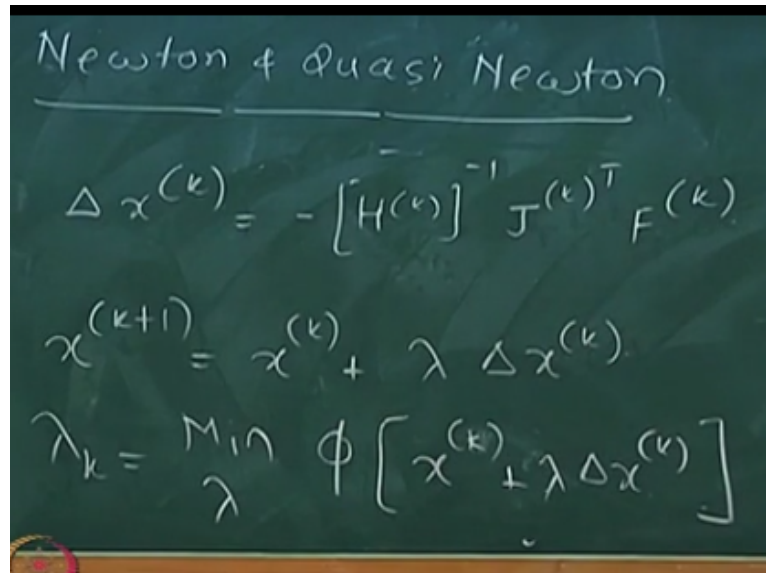
Because this is exactly what you do in Newton’s method for solving algebraic equation. You have nonlinear algebraic equation, you linearize and then instead of solving the original problem, you solved the linearize problem, right. So this we call as H<sub>k</sub>, H<sub>k</sub> is the Hessian and this is the gradient in which we have shown that this is nothing but j<sub>k</sub> transpose f<sub>k</sub>. I have shown this earlier; this is j<sub>k</sub> transpose f<sub>k</sub>.

And now instead of solving for grad phi x = 0, I am going to solve for this approximation = 0 vector, okay. H<sub>k</sub> is the square matrix, so this is the n cross n matrix, phi is the scalar objective function. Its second derivative with respect to x is the Hessian matrix. We are look at Hessian matrix earlier when we talked about the conditions for optimality, necessary and sufficient conditions for optimality.

This is the Hessian matrix, okay. This is the Jacobian transpose f<sub>k</sub>. Well again, I have written here Jacobian transpose f<sub>k</sub>, but you do not have to explicitly compute Jacobian, it is more for

getting end sides. So now, I am going to solve for this, okay. If I solve for this, I get  $\Delta x^k = -H^k$  inverse, I am just solving for that approximate Taylor series expansion, okay, and what I get here is the step, Newton like step, okay.

**(Refer Slide Time: 31:10)**



The chalkboard contains the following equations:

$$\text{Newton \& Quasi-Newton}$$


---


$$\Delta x^{(k)} = -[H^{(k)}]^{-1} J^{(k)T} F^{(k)}$$

$$x^{(k+1)} = x^{(k)} + \lambda \Delta x^{(k)}$$

$$\lambda_k = \underset{\lambda}{\text{Min}} \phi [x^{(k)} + \lambda \Delta x^{(k)}]$$

So this is my search direction, okay. So, the way I construct my next  $x_{k+1}$  is  $x_k +$ , then what you do is the same thing you know, you take lambda times delta  $x_k$ , okay and then you do a search with respect to lambda, okay. So that part remains same, so  $\lambda_k = \min$  with respect to lambda  $\phi$  of  $x_k + \lambda$ , okay. So this part remains same, one dimensional minimization.

But the steps are the direction for moving is obtained using the second order derivative, okay. So actually if you look at optimization based method, let us say just the raw gradient method, then conjugate gradient method and then Newton's method. Of course, Newton method is faster in converging that is because you are using second order derivative information, okay. You are using second order derivative information, okay.

So this is more powerful method, but a price to pay computing Hessian, Hessian is a  $n$  cross  $n$  matrix, if your number of variables is large, Hessian matrix is large and again the same problem, so you know first computations are less number of steps, but large number of computation at each time step or large number of iterations and less computations at each time step, not each time step, iteration, okay.

So it is a balance. In some cases, you know it's worth computing the Hessian and doing quick steps in some cases. Hessian computation can be complex and so you might want to just use the conjugate gradient method and search for the optimum, okay. Now of course, the gradient direction or this direction which you get should be a decent direction and that requires that Hessian should be positive definite and so on.

So, the convergence condition will depend up on the nature of  $H$  are the local Hessian, okay. So what is the trouble with, what is the advantage of Hessian, you are second order information, okay. Convergence can be much better, okay. What is the trouble you have to compute the  $n$  cross  $n$  matrix, okay? In Newton's method, how did we get over this problem, we got over this problem in Newton's method using Broyden update, right.

We use rank one updates and then you know we had a way of updating Jacobian, okay, just like a difference equation and then we use the updated value of the Jacobian and other than actually computing the Jacobian, so same thing is done here. The quasi Newton methods, okay actually use a rank-one update of the Hessian inverse. What you need to compute here is the inverse of hessian matrix, okay.

So what is done is in quasi Newton methods, let us define this matrix  $L$  to be  $H$  inverse. See what is the trouble step in doing this, see this is the gradient calculation, gradient calculation is just one vector calculation by numerical perturbation, not a big issue, but calculating Hessian, see gradient you can do relatively easily because there are only  $n$  components in the gradient of scalar function  $\phi$ , okay.

**(Refer Slide Time: 35:09)**

$$L = H^{-1}$$

Variable metric  
(Davidon-Fletcher-Powell)

$$L^{(0)} = [H^{(0)}]^{-1}$$

$$L^{(k+1)} = L^{(k)} + M^{(k)} - N^{(k)}$$

Much less computations than computing the Hessian, okay. Hessian would require lot more computations, okay. Now to avoid Hessian computations we do gradient computations by numerical perturbation, but Hessian we do an update. So, let us define this  $L = H$  inverse, then we have this update, this is called as variable metric called Davidon-Fletcher-Powell method, in which you update the inverse of Hessian iteratively.

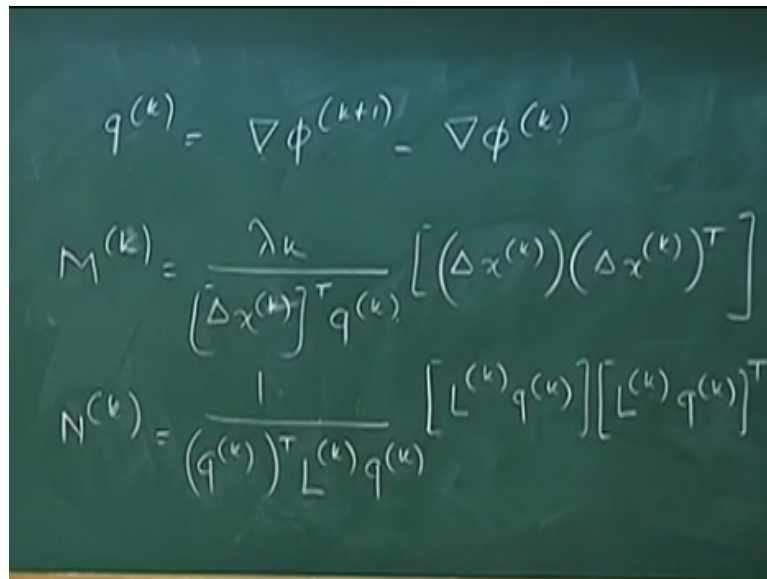
So, only once in the beginning you compute the inverse and after than you just updated without actually having to compute it explicitly, okay. So, this is the quasi Newton idea where you do not have to explicitly keep computing the Hessian. So here, you have this update  $L_{k+1}$ , so we compute  $L_0 = H_0$  inverse. You compute this once in the beginning, okay and after that what we do is, we have this  $L_{k+1} = L_k + M_k - N_k$ , okay.

In quasi Newton method, this is the philosophy that is you define a matrix  $L = H$  inverse, okay you only once compute this and then every time what you do is, the new inverse is old inverse + some correction, okay. This correction does not involve explicit Hessian computation. What is this correction? okay. Let us move to, now the derivation of this is you can find in any book on optimization.

So, derivation of this particular approach, I am going to just leave it to your curiosity, you can go back to a book by S. S. Rao or any other book on optimization, then you will find a derivation, okay. So, you define a vector  $q_k$ , I am just giving you the final formula, which is  $\phi$  evaluated at  $k+1$  - grad  $\phi$  evaluated at iteration  $k$ , okay.  $M_k$ , the correction one is  $\lambda_k \Delta x_k^T q_k$ .



(Refer Slide Time: 38:29)


$$q^{(k)} = \nabla \phi^{(k+1)} - \nabla \phi^{(k)}$$
$$M^{(k)} = \frac{\lambda_k}{[\Delta x^{(k)}]^T q^{(k)}} \left[ (\Delta x^{(k)}) (\Delta x^{(k)})^T \right]$$
$$N^{(k)} = \frac{1}{(q^{(k)})^T L^{(k)} q^{(k)}} \left[ L^{(k)} q^{(k)} \right] \left[ L^{(k)} q^{(k)} \right]^T$$

So this is the rank one matrix,  $\Delta x^k$  is the previous move, okay and this is the rank one matrix,  $\Delta x^k$ ,  $\Delta x^k$  transpose will be a rank one matrix, okay and  $M_k$  and  $N_k$  are two corrections which are rank one corrections. **“Professor - student conversation starts”** (()) (39:37). No, no, no, this is after, see the sequence is like this that, okay, the sequence is that you already have computed  $\Delta x^k$ , okay.

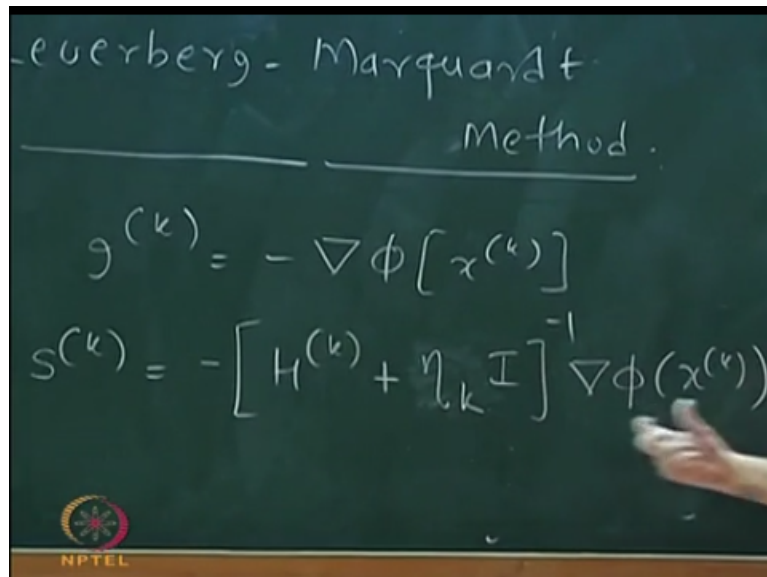
You have then minimize and found out  $\lambda_k$ , okay. Then, you have found out  $x_{k+1}$ , okay. Once you have found out  $x_{k+1}$ , now for the next iteration, you are preparing, okay. So, once I have found out  $x_{k+1}$ , I actually go back here and find this new gradient at  $k+1$ , take the difference between the new gradient and the old gradient, okay and use that and  $\Delta x^k$  and previous inverse, okay.

Previous inverse, okay difference between the old gradient and new gradient and  $\Delta x^k$  move and the  $\lambda_k$  which you have found out, all of them are used to come up with the approximate you know inverse of the Hessian at the next time step. So that is how this quasi Newton method proceed. Well, we do not have this is not a full course optimization, I am just trying to give you the idea that nonlinear algebraic equations can be very efficiently solved using optimization based methods, okay. **“Professor - student conversation ends”**.

So, we are just as I said or I keep saying throughout this course, we are just touching the tip of the iceberg, we are not really getting into the deep, okay Where the last thing which I want to talk here is a method which is very, very popular in solving nonlinear algebraic equations.

This is called Levenberg-Marquardt method is a combination of gradient method and Hessian method, okay.

**(Refer Slide Time: 41:31)**



Levenberg-Marquardt  
Method.

$$g^{(k)} = -\nabla\phi[x^{(k)}]$$
$$s^{(k)} = -[H^{(k)} + \eta_k \mathbf{I}]^{-1} \nabla\phi(x^{(k)})$$

NPTEL

See what is the nice thing about gradient method, gradient method when you are far away from the optimum, it makes very long straights, you know it tries to move towards the optimum very quickly, okay. But once it come close optimum, you know there is a problem. It becomes very, very slow. The optimization, the Hessian based method on the other hand is very fast when you come close to the optimum, okay.

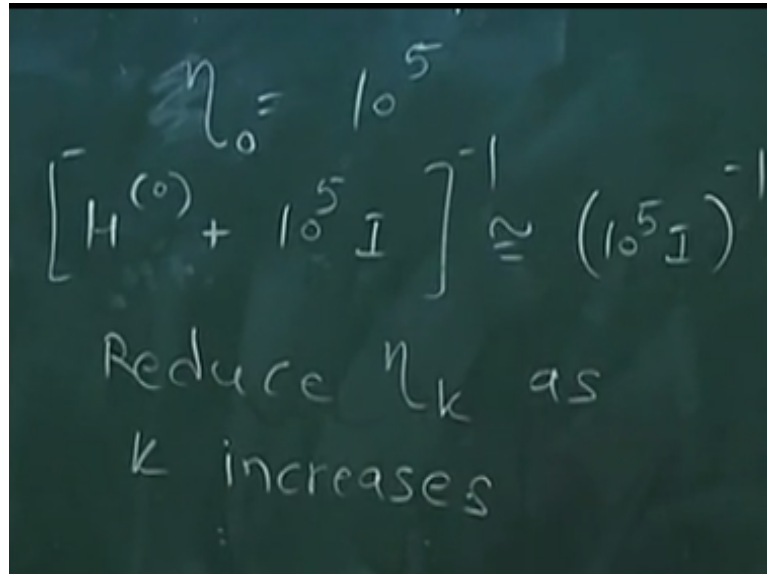
So, there is a meriting having a method which is mixture of the two, which initially starts like a gradient method and later on becomes like a Hessian method. So, this or in the model parlance, it is like multiple agent you know optimization method there are two agents, one is the gradient method, other is a Hessian method and when you tried to mix them, in such a way that one is dominant when it is useful.

The other is dominant when that becomes useful, okay. So this is just a small modification, I am just going to give the philosophy, I am not going to get into details. So, what you do here is that you have this gradient direction which is -, I have this gradient direction, okay. What I do is my search direction  $s_k$  is found by - of this is if I put this beta  $k = 0$ , it is nothing but the Newton step.

If you go back and check this should be nothing but a Newton step, okay. The Newton step actually does involve negative of the gradient direction. It does involve except it is

premultiplied by H inverse, okay. It is premultiplied by H inverse. Otherwise, it involves negative of the gradient direction. So, what we do is okay we start with this eta to be very large, okay.

**(Refer Slide Time: 44:05)**


$$\eta_0 = 10^5$$
$$\left[ H^{(0)} + 10^5 I \right]^{-1} \approx (10^5 I)^{-1}$$

Reduce  $\eta_k$  as  
k increases

So, you start with large value of eta, say 10 to the power 5 or something, you take a large value of eta. So actually your  $H_0 + 10$  to the power 5 times I, okay. See, Hessian elements will not be typically very large, okay. So, you want to take this number sufficiently large compared to elements of the Hessian. So, this inverse when this is a large number is approximately like 10 to the power 5 I inverse.

This term dominates over this, okay. So, initially you would start with very large data, okay. So eta 0 is chosen to be 10 to the power 5 and then what you do is you go on reducing eta, okay as k increases. So, initially since this is like I inverse, this direction  $S_k$  direction is along the negative of the gradient direction. So, initially you are moving along the negative of the gradient direction, okay.

And then you go on, okay, so we reduce eta k as k increases by some logic, okay., As eta k reduces, this terms becomes smaller and smaller and this starts dominating, the Hessian starts dominating, okay. So, initially the method behaves like gradient method. Later on, the method behaves like the Hessian based method. So, Levenberg-Marquardt I think you have programming matlab tool box or any other tool box, scilab toolbox you will find this.

So, this is one of the popular methods which is used for solving nonlinear algebraic equations. So with this we come to an end of algorithmic part of solving nonlinear algebraic equations. We have looked at different methods; we have looked at gradient free or successive substitution methods. We have looked at gradient based or sloped based methods. So in that was wegstien method, Newton method, damped Newton method, okay and so on.

Then, Broyden update for Newton's method, we moved onto optimization based methods, we looked at to gradient method again, okay. Then we looked at Newton's method, quasi Newton method, I have just quickly looked at these things and Levenberg-Marquardt which tries to merge the two gradient and Newton's method.

So, to solve nonlinear algebraic equations, you know it's a complex problem and more and more, we became advanced in computing, we want to solve the larger and larger problems. So, it is an ever challenging problem how to solve nonlinear algebraic equations and there are many, many methods, okay. So, a question that would naturally arise is that which is dou method, there is dou method.

If there are dou method, we would not be required, right. We would be out of business. So, you need an expert, you need a person who understands the physics of the problem, you should know how to concoct the solution, how to concoct a recipe, just like you cook and you will do know the recipe or you will do form a recipe, so here also you have to form a solution for a particular problem and sometimes Newton's method will work.

Sometimes optimization method will work, sometimes wegstien method will work. So, you have to develop an expertise beyond the point as to how to go about solving these problems, okay. So, it does require this human element, otherwise matlab would do everything. It just gives a problem to matlab, but it does not happen that way that is good, that is why we get jobs, okay. So we will now continue with, I will say a few things about the convergence of this nonlinear algebraic equation.

We cannot give justice to that in this course, it is a very, very advance topic. But at least you should be sensitized to what is involve when you talk about convergence, okay. So, early more complex than I talking about convergence of iterative schemes for linear algebraic equations, because here you have nonlinearity and things do not work out as nicely as linear

algebraic equations, but we will have a peak at that and then move on to ODE initial value problems, okay.