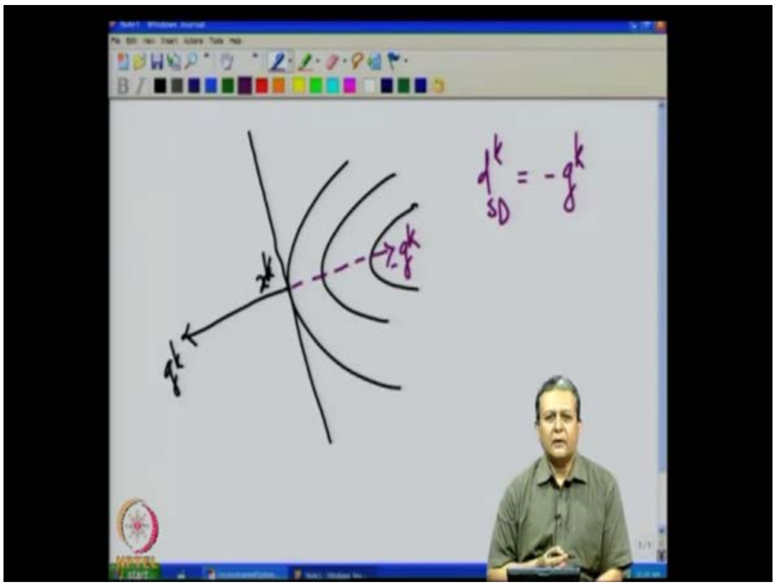
Numerical Optimization

Prof. Shirish K. Shevade
Department of Computer Science and Automation
Indian Institute of Science, Bangalore

Lecture - 14
Classical Newton Method

Welcome back to this series of lectures on numerical optimization. So, in the last class we saw steepest descent algorithm. So, the idea of steepest descent algorithm is to approximate the given function when affine function at a given point and find out the direction in which there is a maximum decrease and that direction is called the steepest descent direction.

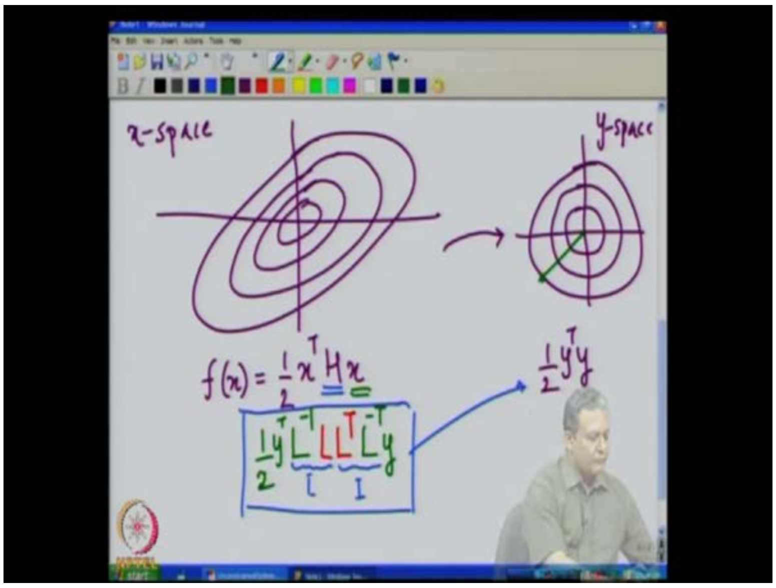
(Refer Slide Time: 00:58)



So, we saw that if we have the contours which are like this, then and this is a current point x k and this is the gradient direction. So, that means in that direction, the function value increases. Now, this is the affine approximation of the function at x k and we use the direction which is the negative of g k, which will give the maximum decrease with respect to the approximation, so this is the negative gradient direction. And so we call this as steepest descent direction and that is nothing but, negative of the gradient direction.

So, we saw about this steepest descent algorithm in the last class where we use in the every iteration of the optimization algorithm, we use this steepest descent direction.

(Refer Slide Time: 02:14)


x-space
y-space
$f(x) = \frac{1}{2}x^T H x$
$\frac{1}{2}y^T y$
$\frac{1}{2}y^T \underbrace{L^{-1} L}_{I} \underbrace{L^T L^{-T}}_{I} y$

Now, we also saw that if we had circular contours, the steepest descent algorithm text one step to reach the solution with respective to the starting point. But if we have elliptical contours. So, for example, suppose the contours are like this then the steepest descent algorithm exhibits zigzagging behaviour. So, lot more iterations are needed for achieving converges, especially when the starting points are not along the Eigen vectors.

So, the question is that can we convert this function into functions. I mean into a function where the contours look circular. So, if you can do that, then we can use a steepest descent method in this new space. So, let us call this as x-space and this as y-space. So, by transforming the original functional into the new space and we can use the steepest descent method in new space to achieve the faster converges.

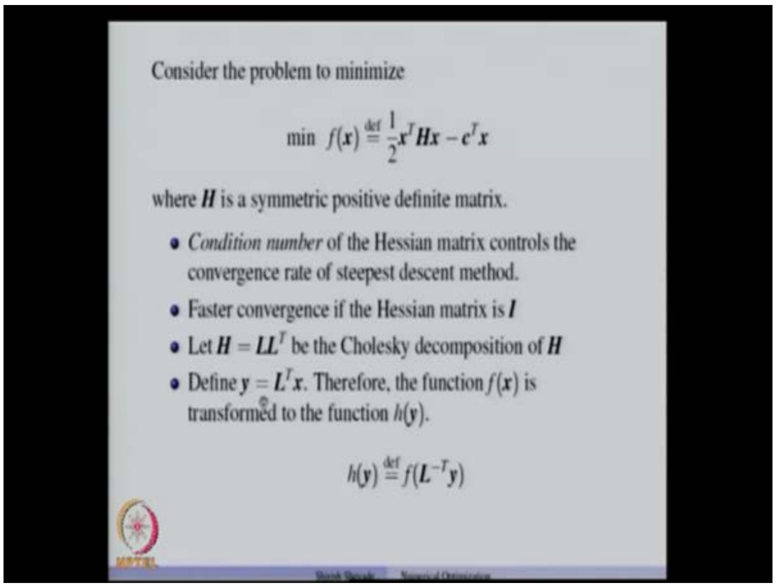
Now, suppose the objective function here is of the type f of x is equal to half of x transpose h x, where h is a symmetric positive definite matrix. Now so, we want to get circular contours here. So, typically this function will be of the form half of y transpose y, where the hessian matrix either is a identity matrix or is a multiple of identity matrix.

So, let us assume that the hessian matrix is a identity matrix. So, how do we convert this original function to this form? So, the idea is to split the matrix H into the matrices L and

L transpose, such that H is equal to L L transpose. We know that this is the Cholesky decomposition of in the matrix H and that is possible because H is a positive definite matrix. Now, once we do that then how do we transform the axis?

So, remember that finally we have to reach a stage where we get the function of the form y transpose y. So, if you substitute x by L minus transpose y and so, then this function becomes half y transpose L inverse L, L transpose, L transpose inverse y. Now, if we consider this is a identity matrix. This is also a identity matrix and therefore, this function that we have got here, that function is same as half of y transpose y and so, if you do this transformation then and then if we use this steepest descent method in the y-space. We know that from any point that we start, suppose we start from this point you will reach the solution in one step using steepest descent method. So, we started looking at this method and so, let us continue our discussion on this.

(Refer Slide Time: 06:34)


Consider the problem to minimize

$$\min \; f(x) \overset{\text{def}}{=} \frac{1}{2}x^T H x - c^T x$$

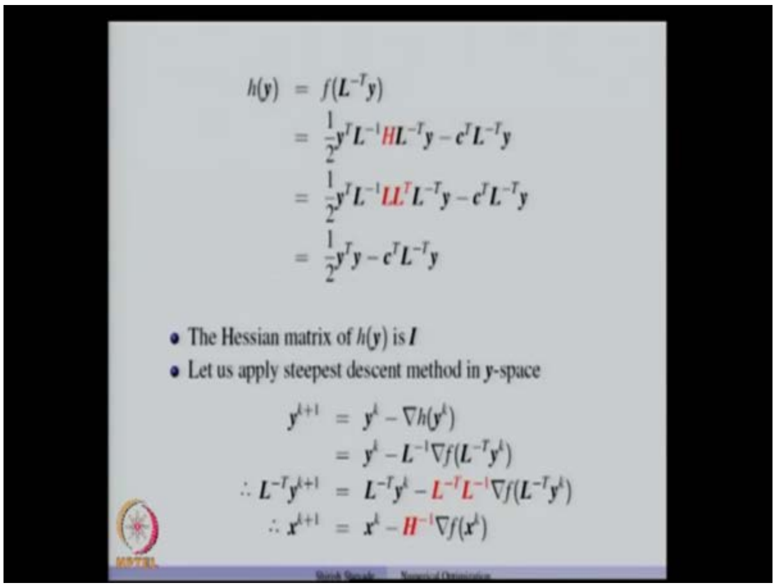
where H is a symmetric positive definite matrix.

- *Condition number* of the Hessian matrix controls the convergence rate of steepest descent method.
- Faster convergence if the Hessian matrix is I
- Let $H = LL^T$ be the Cholesky decomposition of H
- Define $y = L^T x$. Therefore, the function $f(x)$ is transformed to the function $h(y)$.

$$h(y) \overset{\text{def}}{=} f(L^{-T}y)$$

So, if you are given the function f of x to be half x transform H x minus c transform x and if you define y to be L transpose x or x to be L transpose inverse y. Then we have a new function in the y a transform function in the y-space which is h of y which is nothing but, f of L transpose inverse y.

(Refer Slide Time: 06:53)



$$
\begin{aligned}
h(y) &= f(L^{-T}y)\\
&= \frac{1}{2}y^T L^{-1} H L^{-T} y - c^T L^{-T} y\\
&= \frac{1}{2}y^T L^{-1} L L^T L^{-T} y - c^T L^{-T} y\\
&= \frac{1}{2}y^T y - c^T L^{-T} y
\end{aligned}
$$

- The Hessian matrix of $h(y)$ is I
- Let us apply steepest descent method in y-space

$$
\begin{aligned}
y^{k+1} &= y^k - \nabla h(y^k)\\
&= y^k - L^{-1}\nabla f(L^{-T}y^k)\\
\therefore L^{-T}y^{k+1} &= L^{-T}y^k - L^{-T}L^{-1}\nabla f(L^{-T}y^k)\\
\therefore x^{k+1} &= x^k - H^{-1}\nabla f(x^k)
\end{aligned}
$$

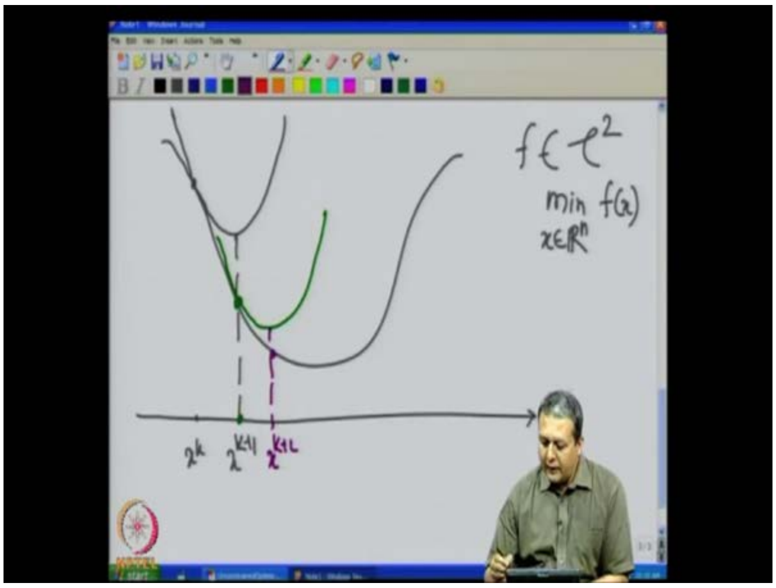
Shirish Shevade Numerical Optimization

And when we expand that what we get is h of y is something like this and which will result in hessian matrix which is a identity matrix. So, this function h y, which is the function defined in the y-space has a hessian, which is a identity matrix. Now, this is very important and then if you apply the steepest descent method in y-space. So, what we get is the new point in the y-space, y k plus 1 is nothing but, y k minus gradient of h y. So, for quadratic functions, convex quadratic functions we can achieve the minimum in one step, if you apply the steepest descent algorithm into y-space.

So, the resulting direction in the x-space, if you look at it that direction is obtained by deflecting the negative gradient direction which is minus gradient f of x k by the matrix H inverse. Now, H is the hessian of the original matrix and so, if the H is the hessian of the original function f and by taking its inverse assuming that H is invertible for general function that amounts to deflecting the negative gradient by H inverse.

So, the Newton method; so, this idea of deflecting the negative gradient by the hessian inverse is called the Newton method. Now, interestingly the newton method was developed sometime in 17th century and the steepest descent method was developed sometime in 19th century by quasi. So, this is not truly the motivation for a newton method. So, the motivation for Newton method is that.

(Refer Slide Time: 09:25)

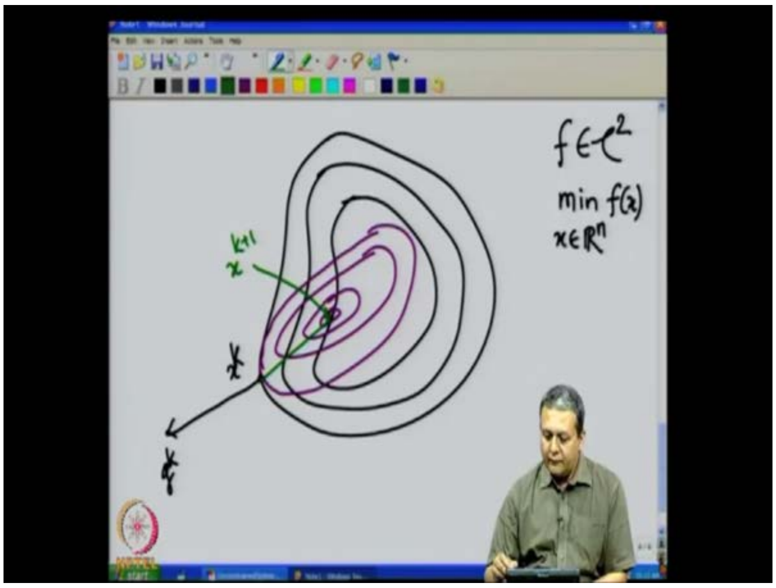


So, let us look at any function which we want to minimize. So, let us assume that this function is twice differentiable. So, we are considering only the one-dimensional functions. Now, the idea behind newton method is that suppose this is the current point x k. So, this is the function value at x k. So, the idea is to approximate the given function by a quadratic function, based on the first order and second order derivatives. And then the newton method typically tries to find the minimum of this quadratic objective function, which is very easy to do. So, if you are given a quadratic function which is convex, it is very easy to find the minimum of quadratic function and in this case this minimum terms out to be this point.

Now, let us look at this point. So, this point becomes x k plus 1. Remember that we are trying to minimize f of x, where x is unconstraint. So, typically x is in the n-dimensional space. Now, we look at this point x k plus 1 and look at the function value at this point and then again we follow the same procedure.

We approximate the given function by a quadratic at this point so that could be something like this and then the newton method finds the minimum of this so that, turns out to be this point and now, that point becomes x k plus 2 and the procedure is repeated. So, every time the newton method approximates a given function by a quadratic function at a given point. So, remember that once we approximate a given function by a quadratic function using Taylor series, we need both first order and second order derivatives that is why we need the function to be twice continuously differentiable. Now, in this case the case which I have shown here is for a simple one-dimensional real valued function.

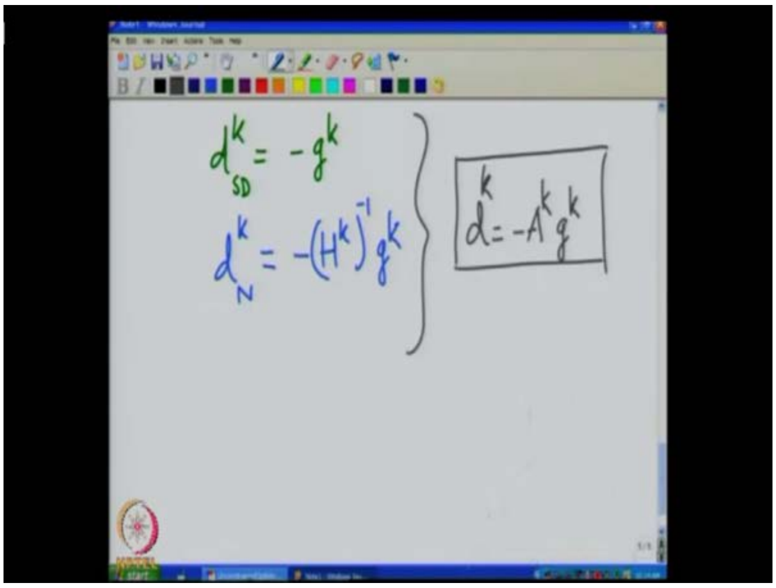
(Refer Slide Time: 12:17)



Now, if you look at a function in n-dimensional space it could have contours of this type and so, if we consider a point suppose, we consider a point here and let us call this as x k, this is the gradient directions. So, the direction along with the function increases. So, if you assume that f is belonging to c 2 and we are trying to minimize f of x, x belongs to r n. So, the newton method what it does is that approximates the given function by a quadratic function.

So, the contours of that quadratic could look like this and we know that we can easily find out the minimum of a quadratic function. So, from x k if you find out the minimum of a quadratic function, the quadratic approximation we can move to this point. So, from x k, one can move to this point. So, this point is going to be x k plus 1 and again at this point 1 starts approximating the given function by a quadratic function. Find the minimum of that quadratic function and the procedure is repeated till the stationary point is reached.

(Refer Slide Time: 14:13)

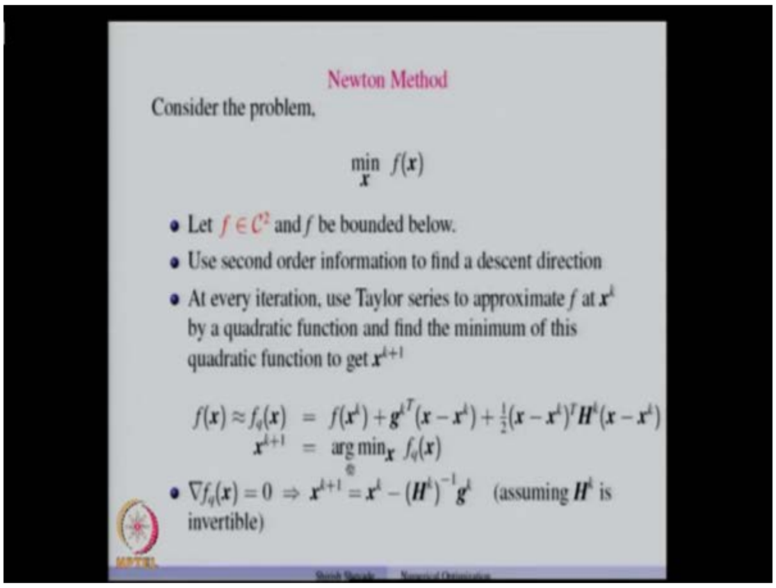


$$\left.\begin{aligned} d_{SD}^{k} &= -g^{k} \\ d_{N}^{k} &= -(H^{k})^{-1} g^{k} \end{aligned}\right\} \boxed{d^{k} = -A^{k} g^{k}}$$

Now, as I mentioned earlier we have the steepest descent direction to be minus g k and we have the newton direction that we are going to see now that is minus H k inverse g k. So, if you see the 2 directions, the newton direction tries to deflect the negative gradient direction by the inverse of the Hessian matrix.

Now, both these directions can be written in the form d k equal to minus A k g k. So, in the steepest descent method, the matrix A k is nothing but, identity matrix while in the Newton method the matrix A k is nothing but, the hessian inverse matrix. So, as I mentioned earlier that lot of directions, descent directions can be generated using this formula. So, different methods use different matrices A k to generate different directions which are used for optimization algorithms. So, let us now look at the newton method.

(Refer Slide Time: 15:50)



Newton Method

Consider the problem,

$$\min_{x} f(x)$$

- Let $f \in C^2$ and f be bounded below.
- Use second order information to find a descent direction
- At every iteration, use Taylor series to approximate f at x^k by a quadratic function and find the minimum of this quadratic function to get x^{k+1}

$$\begin{aligned} f(x) \approx f_q(x) &= f(x^k) + g^{k^T}(x - x^k) + \tfrac{1}{2}(x - x^k)^T H^k (x - x^k) \\ x^{k+1} &= \arg\min_x f_q(x) \end{aligned}$$

- $\nabla f_q(x) = 0 \Rightarrow x^{k+1} = x^k - (H^k)^{-1} g^k$ (assuming H^k is invertible)

Now, let us consider the unconstrained optimization problem to minimize f of x and let us assume that f is twice continuously differentiable. So, this is the difference between the earlier approach and the current approach. In the steepest descent method, we just wanted continuously differentiable functions. But, whenever we want to apply newton method, we need to ensure that the function is twice continuously differentiable and we also assume that f is bounded below, that is the reasonable assumption because we want to minimize a problem. So, we can assume that f is bounded below.

Now, the idea behind newton method is to use second order information also to find the descent direction. And at every iteration of newton method, Taylor series approximation of the function of at a given point x k is used and since the function f is assumed to be twice continuously differentiable. We can approximate f at x k by a quadratic function. So, let us see how to do that.

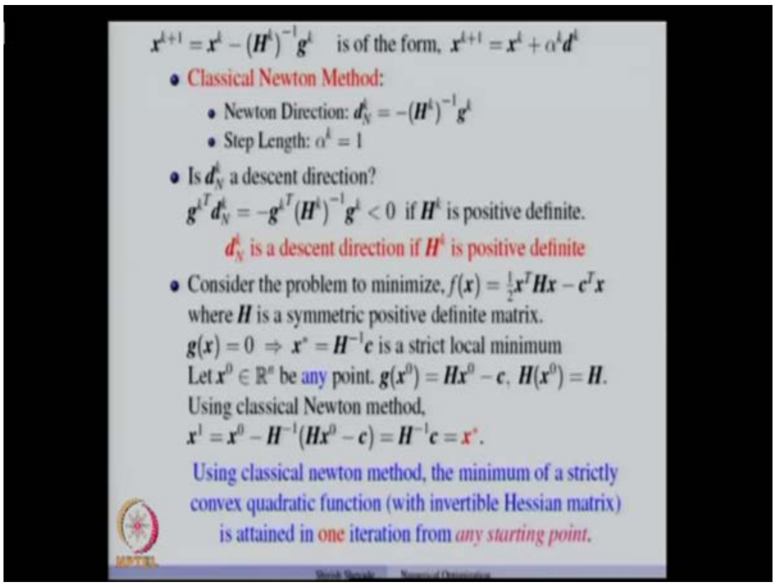
And once we find the quadratic approximation, one can go to the minimum of that quadratic function to find the new point x k plus 1 and this procedure is repeated till some stopping criteria is satisfied.

So, the given function f can be approximated by f q x, those are the subscript q here stands for the quadratic approximation. And the quadratic approximation using second order Taylor series is f q x is nothing but, f of x k. Remember that we are trying to approximate; we are approximating the function f by quadratic function at x k.

So, f q x is nothing but, f of x k plus g k transpose x minus x k plus half x minus x k transpose H k into x minus x k, where H k is the hessian at the current point x k. So, we will continue to use this shortened notations throughout this course. The g k stands for the gradient at the current point x k and H k stand for the hessian at the current point x k. So, you will continue to use these notations throughout this course.

Now, x k plus 1 is obtained by minimizing the quadratic function f q x. Now, finding the minimum of this unconstraint problem f q x is very easy. We have to set take the gradient equated to 0 and then that will give us x k plus 1. So, the gradient of the f q x is equal to 0 gives us the x k plus 1 to be x k minus H k inverse into g k. So, here we have assumed that the hessian matrix H k is invertible, only then this step makes sensible. So, let us assume that the hessian is invertible at x k. Now, once you get a new point then again the procedure is to approximate f of x at new point x k plus 1 and then find the minimum of that new quadratic approximation to get x k plus 2 and so on. So, this results in simple newton algorithm. Now, if you look at the update that we have seen here. x k plus 1 is equal to x k minus H k inverse g k.

(Refer Slide Time: 19:44)



$x^{k+1} = x^k - (H^k)^{-1} g^k$ is of the form, $x^{k+1} = x^k + \alpha^k d^k$

- Classical Newton Method:
  - Newton Direction: $d_N^k = -(H^k)^{-1} g^k$
  - Step Length: $\alpha^k = 1$
- Is d_N^k a descent direction?
  $g^{k^T} d_N^k = -g^{k^T} (H^k)^{-1} g^k < 0$ if H^k is positive definite.
  d_N^k is a descent direction if H^k is positive definite
- Consider the problem to minimize, $f(x) = \frac{1}{2} x^T H x - c^T x$
  where H is a symmetric positive definite matrix.
  $g(x) = 0 \Rightarrow x^* = H^{-1} c$ is a strict local minimum
  Let $x^0 \in \mathbb{R}^n$ be any point. $g(x^0) = Hx^0 - c$, $H(x^0) = H$.
  Using classical Newton method,
  $x^1 = x^0 - H^{-1}(Hx^0 - c) = H^{-1} c = x^*$.
  Using classical newton method, the minimum of a strictly convex quadratic function (with invertible Hessian matrix) is attained in one iteration from *any starting point*.

If you write it in our visual form x k plus 1 is equal to x k plus alpha k d k, then what we get is alpha k to be 1 and d k 2 be minus H k inverse g k and that is called the classical Newton method.

So, the classical Newton method, you just the newton direction which will be denoted by d k n. So, d and the n stands for the newton method and k stands for the k-th iteration and the newton direction at the k-th iteration is nothing but, minus hessian inverse into the gradient at the current point. In addition to this direction, we will also say that alpha k is equal to 1, for the classical newton method. Now, the question that we need to answer is the Newton direction a descent direction or under what conditions this newton direction will be a descent direction?

Now, we know that if g k transpose d k is less than 0, then d k is a descent direction. So, we apply the same idea here. So, g k transpose newton direction is nothing but, if you use this newton direction. So, is g k transpose inverse hessian into g k. Now, this quantity is less than 0, if H k is a positive definite matrix.

So, newton direction is a descent direction, if H k is a positive definite matrix. Now, let us consider the problem to minimize f of x to be half x transpose H x minus c transpose x, where H is a symmetric positive definite matrix. So, clearly H is invertible.

Now, if we set the gradient of this f to 0, what we get is x star to be H inverse c and that is the strict local minimum. Now, if we apply newton method instead of finding the x star in this way, if you decide to apply newton method to solves, minimize this function f of x. So, suppose we start from any point x 0 in the n-dimensional space and the new point is found by approximating the given function by quadratic function. In this case, this function indeed is a quadratic function. So, the approximation is not necessary, we can straight away go to the minimum of this function. So, if you use the Newton method.

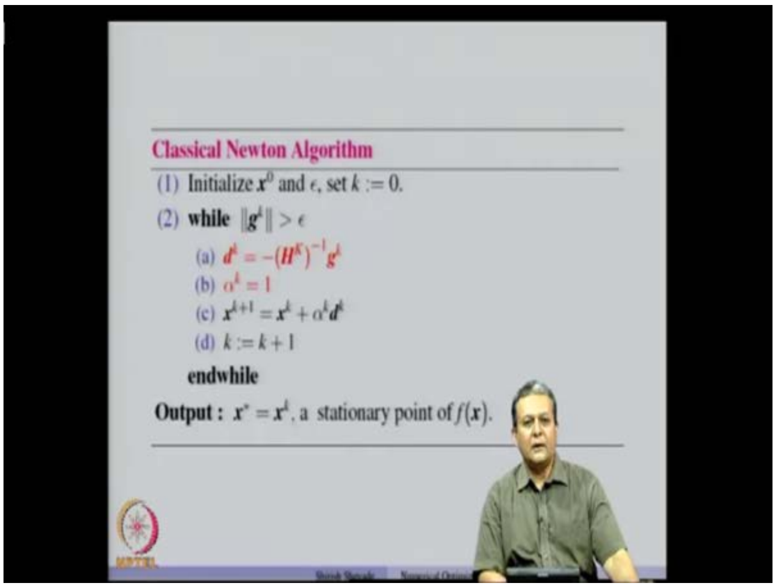
Now, if you use first, if you find the newton direction which is nothing but, the H in hessian inverse into the gradient at the current point. So, the gradient at the current point g x naught is nothing but, H x naught minus c and the hessian at the current point is H. So, if you use classical newton method what we get is x 1 is equal to x 0 plus d d 0 n and d 0 is n is nothing but, minus H inverse into g 0 and g 0 is nothing but, H x 0 minus c. So, H inverse into H is identity matrix. Therefore, this will be x 0 minus x 0 plus H inverse c. So, what we get is x 1 is equal to H inverse c and this is nothing but, x star.

So, you see that if for a quadratic function, if you apply classical newton method starting from any point we can reach the solution in exactly one step assuming that the initial point is not the optimal point.

Now, compare this with the behaviour of the steepest descent method that we saw early this steepest descent method depends the behaviour of the steepest descent method depends a lot on the condition number of the hessian matrix H. So, if the condition number of this hessian matrix is 1, then the steepest descent method converges in one step starting from any time, any point on the other hand, if the condition number is much greater than 1, then we saw that the zigzagging takes place and the steepest descent method requires many more iterations for a typical starting point. Now, this is different from what we see here in the newton method for a quadratic convex, quadratic function the newton method, classical newton method gives us a solution in exactly one step.

So, this is the important result that using classical Newton, method the minimum of a strictly convex quadratic function where the hessian matrix is invertible that minimum is attained in 1 iteration from any starting point. Here, I am assuming that the starting point is not the optimal point.

(Refer Slide Time: 25:45)



Classical Newton Algorithm

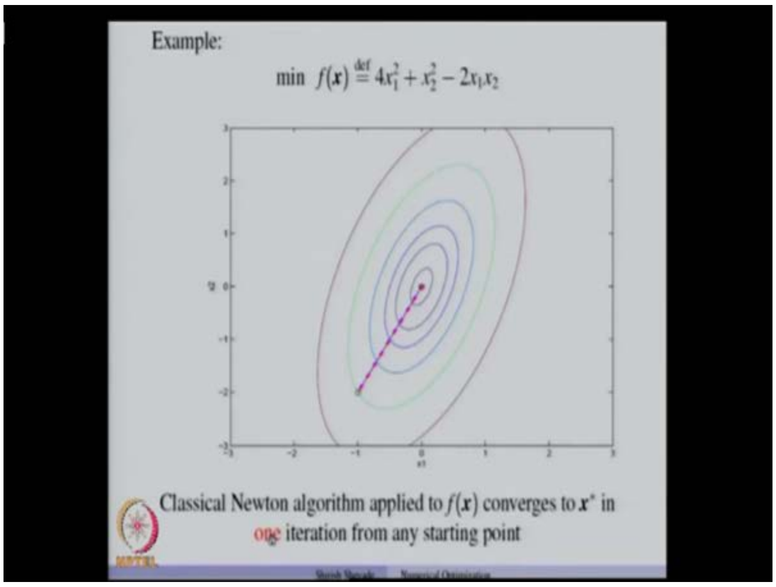
```
(1) Initialize $x^0$ and $\epsilon$, set $k := 0$.
(2) while $\|g^k\| > \epsilon$
        (a) $d^k = -(H^k)^{-1} g^k$
        (b) $\alpha^k = 1$
        (c) $x^{k+1} = x^k + \alpha^k d^k$
        (d) $k := k + 1$
    endwhile
Output : $x^* = x^k$, a stationary point of $f(x)$.
```

Now, let us look at the classical newton algorithm or the algorithm which uses classical newton method to minimize a general objective function not necessarily a quadratic objective function. So, as is the usual case we initialize x 0, then the tolerance parameter for the gradient said the iteration count to 0.

Now, while the norm of the gradient is greater than epsilon, we find the direction which is nothing but, negative of the hessian inverse into the gradient at the current point. We set alpha k to 1 and then the new point is nothing but, x k plus alpha k d k k equal to k plus 1 and the whole procedure is repeated till the norm of the gradient is less than or equal to epsilon. And when the algorithm terminates, we get x start to be x k a stationary point of f of x.

Now, let us see some examples where this Newton method is applied to different kinds of functions.

(Refer Slide Time: 27:00)



Example:

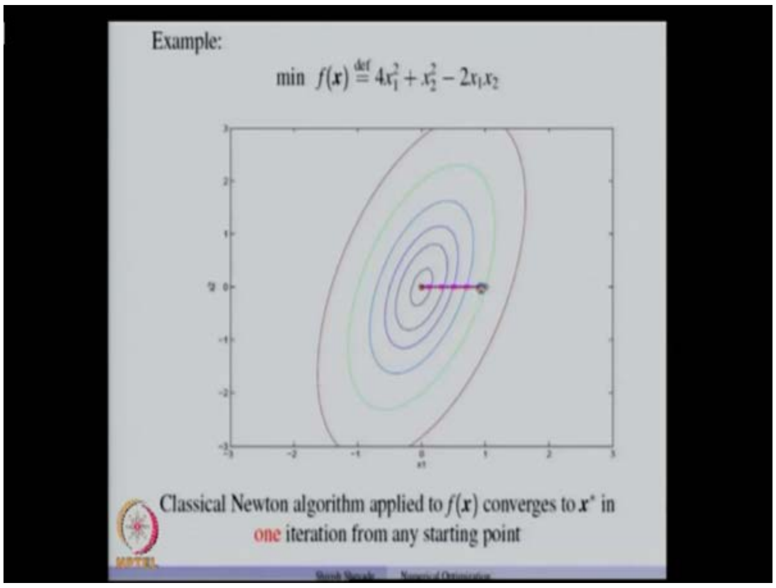
$$\min \; f(\boldsymbol{x}) \overset{\text{def}}{=} 4x_1^2 + x_2^2 - 2x_1x_2$$

Classical Newton algorithm applied to $f(\boldsymbol{x})$ converges to $\boldsymbol{x}^*$ in one iteration from any starting point

So, we already know that if we have circular contours then even the steepest descent method finds a minimum in one step. So, let us consider the quadratic function where the contours are elliptical not the circular ones, because for circular contours both steepest descent newton method would converge in one step, for a quadratic functions.

So, this is the contour plot of the function and suppose this is the initial point and if you apply classical newton algorithm to minimise f of x, then we will see that in one step it has found out the solutions. So, in one step if you start from this point in one step, we reach the solution x star in 1 iteration. Now, if you change the initial point what happens?

(Refer Slide Time: 28:08)



Example:

$$\min\ f(\boldsymbol{x}) \overset{\text{def}}{=} 4x_1^2 + x_2^2 - 2x_1x_2$$

Classical Newton algorithm applied to $f(\boldsymbol{x})$ converges to $\boldsymbol{x}^*$ in one iteration from any starting point

So, let us take the same function but, change the initial point. So, even in this case you will see that the classical newton algorithm converges in exactly one step.

(Refer Slide Time: 27:00)

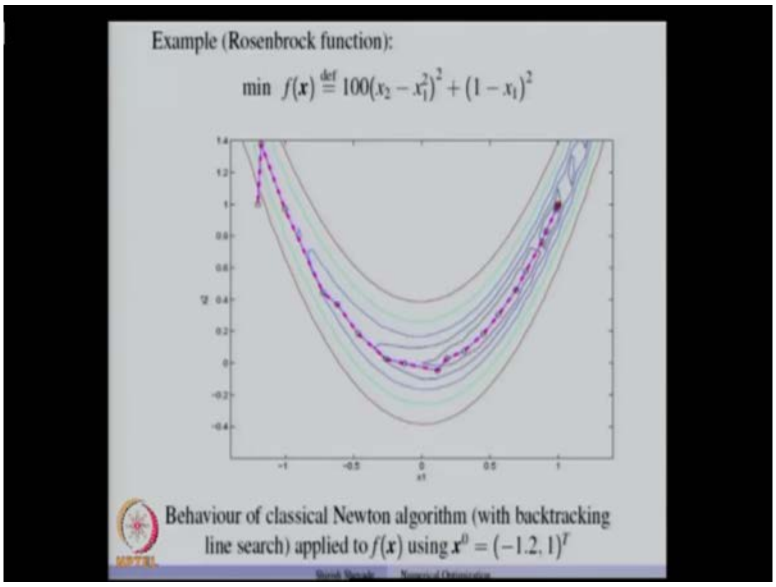
Now, if you remember for this function with the same starting point if you had used steepest descent method, we saw that there was some zigzagging behaviour before the convergence took place. And as I mentioned earlier in this application of the steepest descent method to this function to minimize this function starting from this point required more than 20 iterations to reach the solution x star and compare that with the Newton method.

So, you will see that the newton method is faster than the steepest descent method. But, then if one looks at the algorithm, one realizes that one needs the second order information when one applies classical newton algorithm and that could be in many cases and expensive, computationally expensive operation. Not only do we need the second order information but, also we need the inversion of n by n matrix and as we know the competition complexity of inverting a matrix is order n cube, inverting n by n matrix is order n cube.

Now, typically this step is not carried out by finding the hessian inverse but, typically it is obtained the d k is obtained by solving the system of equations H k D k equal to minus g k and as we have seen in one of the earlier classes on mathematical background, it may be good to do the Cholesky decomposition of this matrix H k, if the matrix is positive definite and then finding the direction d k becomes numerical stable operation.

So, we see that newton method is faster than the steepest descent method but, it requires second order information unlike the steepest descent method.

(Refer Slide Time: 30:40)



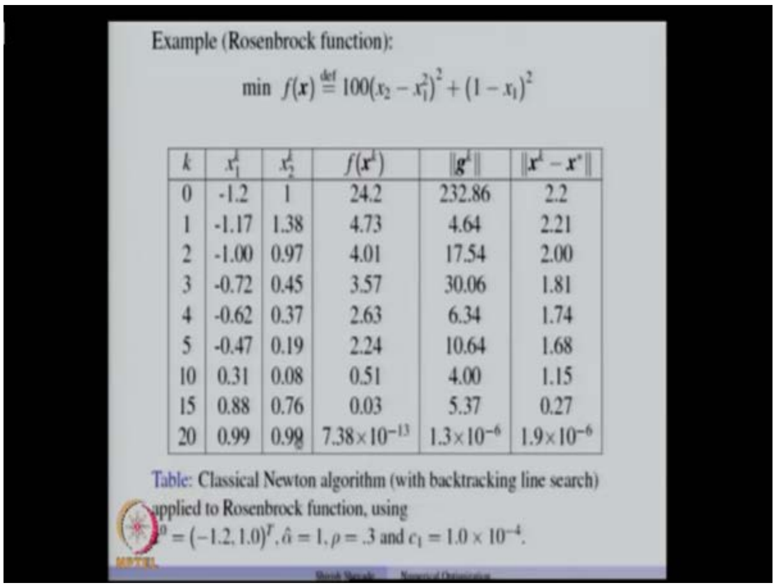
Example (Rosenbrock function):

$$\min\ f(\boldsymbol{x}) \overset{\text{def}}{=} 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Behaviour of classical Newton algorithm (with backtracking line search) applied to $f(\boldsymbol{x})$ using $\boldsymbol{x}^0 = (-1.2, 1)^T$

Now, let us apply newton method, classical newton algorithm to the standard Rosenbrock function which is mentioned here. Now, these are contours of this Rosenbrock function and we know that minimum rise at this point 1 1. Now, if you start from this point, this is the path which is followed by the classical newton algorithm before it converges to x star. In this algorithm, I have used backtracking line search with the Newton direction.

(Refer Slide Time: 31:21)



Example (Rosenbrock function):

$$\min \ f(\boldsymbol{x}) \overset{\text{def}}{=} 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

k	x_1^k	x_2^k	$f(\boldsymbol{x}^k)$	$\lVert \boldsymbol{g}^k \rVert$	$\lVert \boldsymbol{x}^k - \boldsymbol{x}^* \rVert$
0	-1.2	1	24.2	232.86	2.2
1	-1.17	1.38	4.73	4.64	2.21
2	-1.00	0.97	4.01	17.54	2.00
3	-0.72	0.45	3.57	30.06	1.81
4	-0.62	0.37	2.63	6.34	1.74
5	-0.47	0.19	2.24	10.64	1.68
10	0.31	0.08	0.51	4.00	1.15
15	0.88	0.76	0.03	5.37	0.27
20	0.99	0.99	7.38×10^{-13}	1.3×10^{-6}	1.9×10^{-6}

Table: Classical Newton algorithm (with backtracking line search) applied to Rosenbrock function, using $\boldsymbol{x}^0 = (-1.2, 1.0)^T, \hat{\alpha} = 1, \rho = .3$ and $c_1 = 1.0 \times 10^{-4}$.

If you see the performance of this we will see that, here is a table which describes the performance of classical newton algorithm with backtracking line search applied to Rosenbrock function.

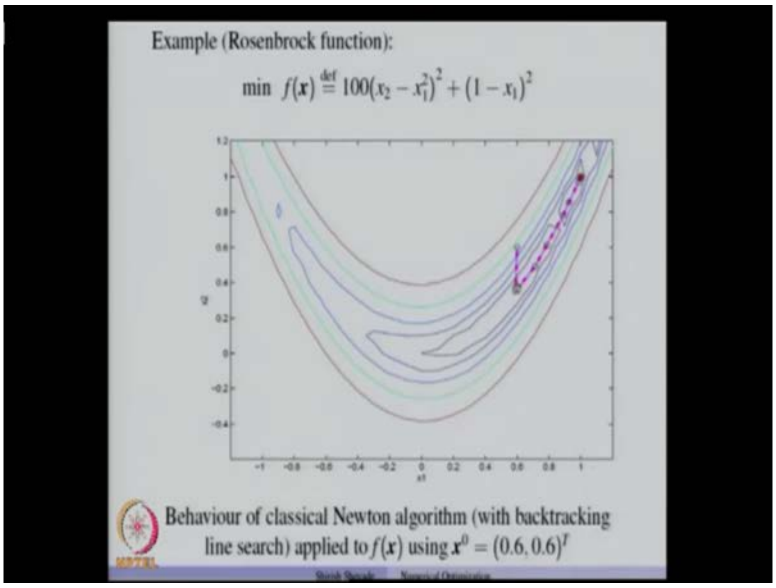
So, the first column denotes the iteration number, the next 2 columns denote the first and the second coordinate of the iterates given by the newton algorithm, that this column gives the function value at a current point x k. This gives a norm of the gradient and this gives the distance between the current point and the solution point and all these norms are Euclidean norms. So, we start with the point minus 1.2 and 1. So, value of the function is 24.2 and the norm of the gradient is 232.86.

And as the iterations progress, you will see that in every iteration the value of the function decreases and finally, when the algorithm terminates, we get a point which is very close to the optimal point which is, whose x and y coordinates or whose both the coordinates are 1 and 1. And you will see that the norm of the gradient is 1.3 into 10 to the power minus 6. The value of epsilon that I have chosen here is 10 to the power minus 3. So, from iteration 19 to 20 so, at the end of iteration 19 the norm of the gradient was greater than epsilon and the 20th iteration became less than epsilon and then the algorithm terminate and we would see that the distance between the point x k and x star is very close to 0.

So, when I use backtracking line search typically for newton algorithms, it is a good idea to set alpha had to be 1. The other parameters related to the backtracking line search can be obtained by using some, by running the algorithm many times and trying to find out what are the best possible parameters.

Now, compare this with the application of the steepest descent algorithm to Rosenbrock function with the same initial point. You recall that, that steepest descent algorithm required more than 2000 iterations to achieve the convergence starting from the same point. So, there is a lot of time saving when one uses Newton method especially, in terms of number of iterations. But, every iteration of Newton method requires the inversion of a hessian matrix to be computed and that could to be time consuming, if the dimension n is very large.

(Refer Slide Time: 34:43)



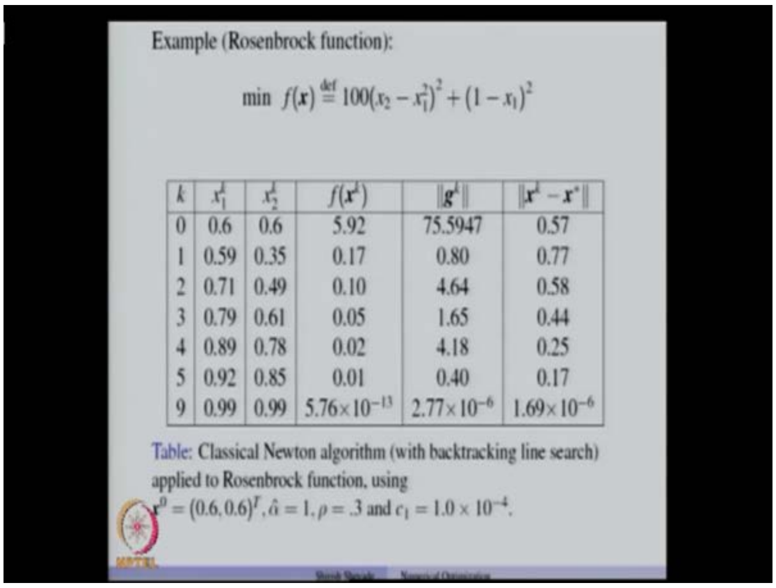
Example (Rosenbrock function):

$$\min \; f(\boldsymbol{x}) \overset{\text{def}}{=} 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Behaviour of classical Newton algorithm (with backtracking line search) applied to $f(\boldsymbol{x})$ using $\boldsymbol{x}^0 = (0.6, 0.6)^T$

Now, we take the same function but, with the different initial point and this point is 0.6, 0.6 and if you apply a classical newton algorithm with backtracking line search. We see that from this the path followed is this before it converges to x star.

(Refer Slide Time: 35:02)



Example (Rosenbrock function):

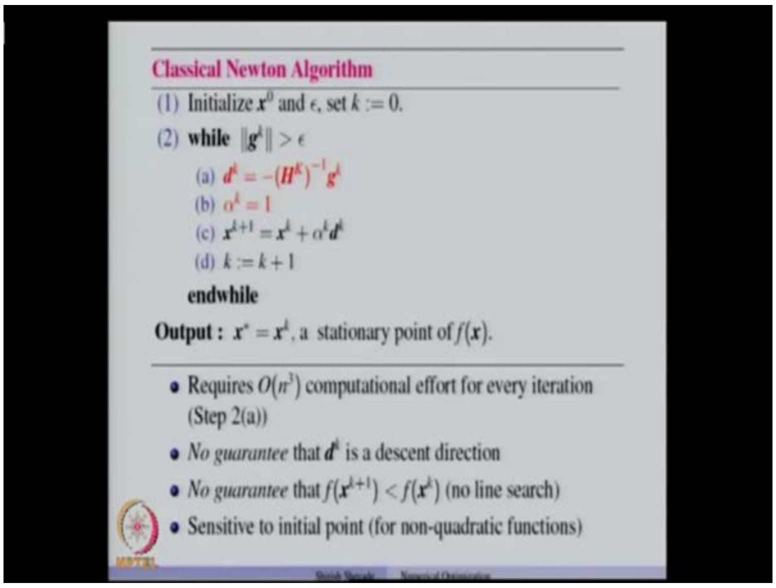
$$\min \ f(\boldsymbol{x}) \overset{\text{def}}{=} 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

k	x_1^k	x_2^k	$f(\boldsymbol{x}^k)$	$\lVert \boldsymbol{g}^k \rVert$	$\lVert \boldsymbol{x}^k - \boldsymbol{x}^* \rVert$
0	0.6	0.6	5.92	75.5947	0.57
1	0.59	0.35	0.17	0.80	0.77
2	0.71	0.49	0.10	4.64	0.58
3	0.79	0.61	0.05	1.65	0.44
4	0.89	0.78	0.02	4.18	0.25
5	0.92	0.85	0.01	0.40	0.17
9	0.99	0.99	5.76×10^{-13}	2.77×10^{-6}	1.69×10^{-6}

Table: Classical Newton algorithm (with backtracking line search) applied to Rosenbrock function, using $\boldsymbol{x}^0 = (0.6, 0.6)^T, \hat{\alpha} = 1, \rho = .3$ and $c_1 = 1.0 \times 10^{-4}$.

And if you look at the table so, here the algorithm terminated in 9 iterations and compare this with the termination achieved by the steepest descent method in more than 2000 iterations. Now, if you start from 0.6, 0.6, the initial norm of the gradient is 75.59 and in 9 iterations, the gradient reduce to 2.7 into 10 to the power minus 6 and value the function is very small. The distance between the x k and x star is almost close to 0 and the point that we get is very close to the optimal point which is 1 comma 1. So, this table and the figures illustrate the application of newton method, 2 different kinds of functions.

(Refer Slide Time: 36:10)



Classical Newton Algorithm

```
(1) Initialize $\boldsymbol{x}^0$ and $\epsilon$, set $k := 0$.
(2) while $\lVert \boldsymbol{g}^k \rVert > \epsilon$
      (a) $\boldsymbol{d}^k = -(H^k)^{-1}\boldsymbol{g}^k$
      (b) $\alpha^k = 1$
      (c) $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$
      (d) $k := k + 1$
    endwhile
Output : $\boldsymbol{x}^* = \boldsymbol{x}^k$, a stationary point of $f(\boldsymbol{x})$.
```

- Requires $O(n^3)$ computational effort for every iteration (Step 2(a))
- *No guarantee* that $\boldsymbol{d}^k$ is a descent direction
- *No guarantee* that $f(\boldsymbol{x}^{k+1}) < f(\boldsymbol{x}^k)$ (no line search)
- Sensitive to initial point (for non-quadratic functions)

Now, let us revisit the classical newton algorithm that we saw earlier. So, first step which is a direction finding step finds the direction d k to be the negative of the hessian inverse into g k the step length is always set to 1 in classical newton algorithm and the new point is obtained by x k plus 1 is equal to x k plus alpha k d k and k equal to k plus 1 and the process is repeated till norm of g k less than or equal to epsilon and as a output we get a stationary point x k.

Now, let us see some of the important observations related to the classical newton algorithm, which is shown here. As I mentioned earlier at every iteration there is a order n cube computational effort needed and that will be expensive if the number of iterations goes up.

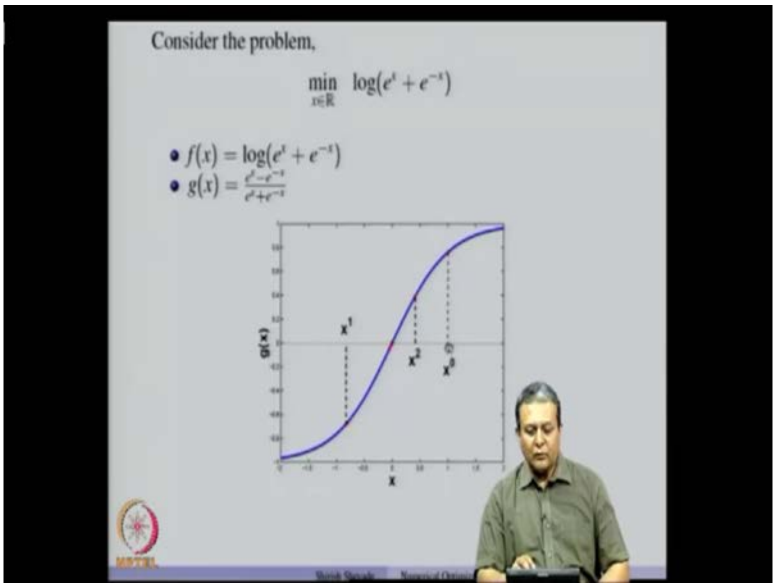
Now, not only that the order n cube computational effort is needed but, the newton method also requires order n square computational storage because we need to store the hessian matrix at every iteration and that requires order n square storage. So, both the storage wise as well as the computational effort wise, newton method is expensive compared to steepest descent method. Steepest descent method does not use any second order information. So, there is no need to store any matrix and there is also no need to invert any matrix in this steepest descent method, while here we need to store this H k as well as inverted in every iteration and that is going to be every expensive computationally for last dimensional problems.

Now, what is the guarantee that the direction d k that we choose here in step 2 is a descent direction? So, in this algorithm you will see that there is no check that d k is a descent direction. So, if H k is a negative definite, then d k will turn out to be in ascent direction rather than the descent direction that we are looking for. But, the algorithm does not check the positive definiteness of the hessian matrix at any iteration k, not only that sometimes the matrix H k also could be close to a singular matrix and therefore, inverting a matrix which is close to a singular matrix will give rise to some numerical difficulties and those are not handled in the classical newton algorithm. So, there is no guarantee that d k is a descent direction, we can get d k to be ascent direction also and not only that, the algorithm also could numerically unstable, if h k at some iteration becomes a matrix which is close to a singular matrix.

Now, another drawback of classical newton algorithm is that there is no guarantee that the objective function value is decreases at every iteration and this is because there is no line search which is done here, note that we are taking alpha k to be always 1 at every iteration. So, when the line search is not done we cannot ensure that the objective function value decreases at every iteration.

In fact, in some cases with alpha k equal to 1 the objective function value could increase and therefore, the sufficient decrease conditions of armijo goldstein or armijo wolfe, in some they are not satisfied. And another important point that needs to be noted is that the algorithm, the classical newton algorithm is very sensitive to the initial initial point to x naught. So, let us see 1 example. We had seen this example in the one-dimensional case but, I just want to repeat that.

(Refer Slide Time: 40:55)



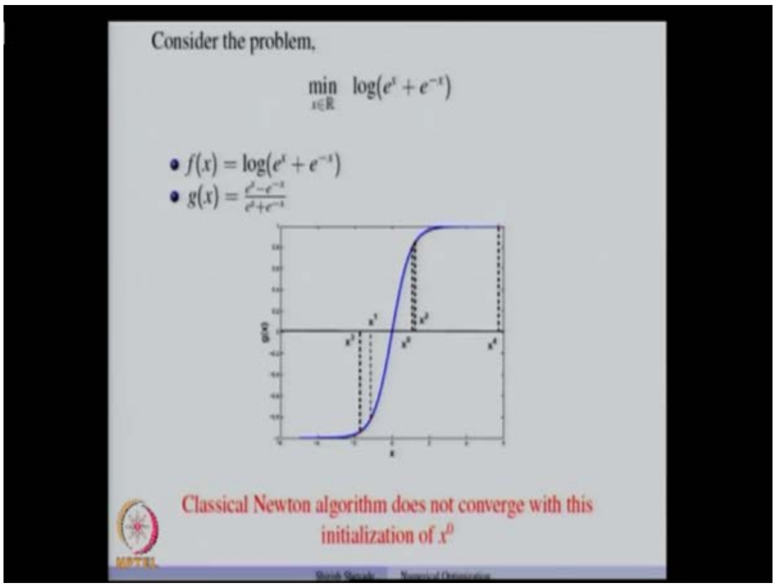
Consider the problem,

$$\min_{x \in \mathbb{R}} \log(e^x + e^{-x})$$

- $f(x) = \log(e^x + e^{-x})$
- $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

So, let us consider or real valid functions defined on R and let us look at the derivative of that function. Now, if you want to minimize this function we want to find out the root of g x. Now, the plot of g x is shown here. Now, if you start from x 0 and use newton algorithm we go to the point x 1 and again when we use newton algorithm, we go to point x 2, then x 3 is very close to this and finally, the algorithm converges to this point which is a root of g x. Now, this was with respect to this initial point.

(Refer Slide Time: 41:44)



Consider the problem,

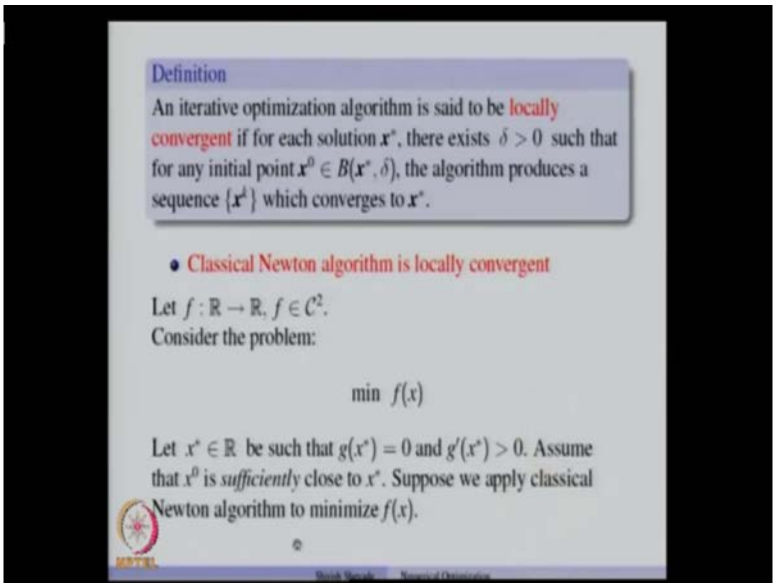
$$\min_{x \in \mathbb{R}} \log(e^x + e^{-x})$$

- $f(x) = \log(e^x + e^{-x})$
- $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Classical Newton algorithm does not converge with this initialization of x^0

Now, suppose if you take another initial point, this was the initial point is x 0 here and then, we move on to x 1, x 2, x 3 and x 4. So, every time you would see that we are moving away from the root of g x because x 1, the distance between x 1 and x star is less than the distance between x 2 and x star and distance between x 2 and x star is less than distance between x 3 and x star and finally, x 4 is here and you will see that the algorithm diverges. So, newton method used in the classical sense may not guarantee convergence always. So, this is a very important point about the Newton method.

(Refer Slide Time: 42:47)



Definition

An iterative optimization algorithm is said to be locally convergent if for each solution x^*, there exists $\delta > 0$ such that for any initial point $x^0 \in B(x^*, \delta)$, the algorithm produces a sequence $\{x^k\}$ which converges to x^*.

- Classical Newton algorithm is locally convergent

Let $f : \mathbb{R} \to \mathbb{R}$, $f \in \mathcal{C}^2$.
Consider the problem:

$$\min f(x)$$

Let $x^* \in \mathbb{R}$ be such that $g(x^*) = 0$ and $g'(x^*) > 0$. Assume that x^0 is *sufficiently* close to x^*. Suppose we apply classical Newton algorithm to minimize $f(x)$.

So, let us look the definition of locally convergent optimization algorithm. So, in iterative optimization algorithm is said to be locally convergent, if for each solution x star there exist some delta which is greater than 0. Now, this delta is a function of x star so, different for different solutions, there would exists different deltas but, to avoid notational clutter, I have a written it as only delta but, keep in mind that the delta is a function of x star. So, there exists some delta which is greater than 0 such that if the initial point is in a ball of radius delta around x star then the algorithm produces sequence x k which converges to x star.

So, such algorithms are called locally convergent algorithms. So, that means that if we are given x star and if the initial point we chosen around x star in the delta neighbourhood of x star or if one happens to choose initial point in the delta neighbourhood of the solution point, then the algorithm and if the algorithm converges to x star then the algorithm is said to be locally convergent.

Now, the claim is that the classical newton algorithm that we saw today is a locally convergent algorithm. So, that means that for each solution x star, there exist some neighbourhood such that if we start within that neighbourhood and if the algorithm generates a sequence which converges to x star, then we will definitely get a locally convergent algorithm.

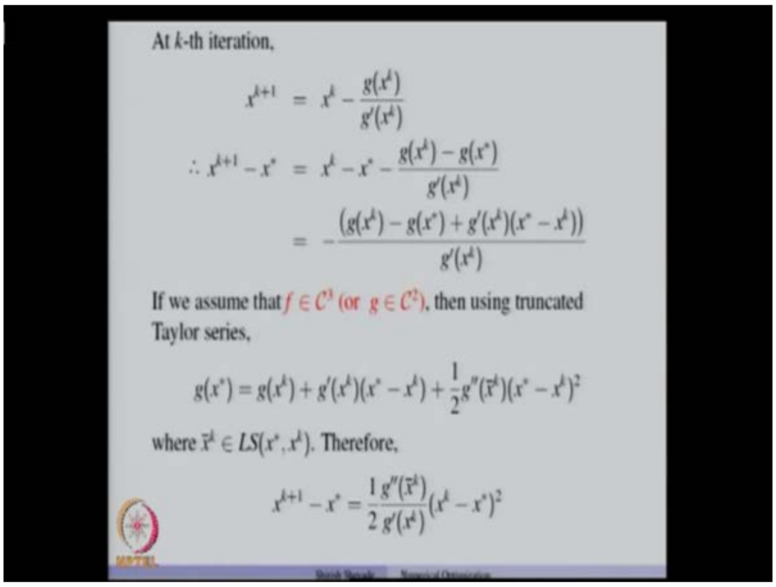
So, let f b from r to r f belongs to c 2 because we are going to use newton algorithm. So, these are reasonable things. So, we will show this result that the classical newton algorithm is locally convergent for a one-dimensional case. The extension to higher dimensions can be worked out.

So, what we want to show is that if we start from a point which is close to x star and use newton method. Then we are guaranteed that it will converge to x star but, if we or in the other words if we start from a point which is not in the delta neighbourhood of x star, there is no guarantee that the newton algorithm will convergent. We saw 1 example just now that for different initializations, the newton algorithm either converges or diverges or sometimes it might oscillate also.

So, let us consider the problem to minimize f of x and if suppose we want to use newton method. Now suppose x star is a point in R such that, the gradient of the function vanishes at that point and then the second derivative of the function at that point is greater than 0, remember that f 2 dash x star is nothing but, g dash x star.

So, we want to now apply the newton algorithm for this function and we will see how to get the delta neighbourhood in around x star so that we can start from that point and then the algorithm will if you use classical newton algorithm, it will definitely converge. So, let us assume initially that x 0 is sufficiently close to x star but, as this movement we do not know, what is meant by sufficiently close but, let us assume that x 0 is sufficiently close to x star. Now, suppose we apply classical newton algorithm to minimize f of x.

(Refer Slide Time: 47:25)



At k-th iteration,

$$\begin{aligned} x^{k+1} &= x^k - \frac{g(x^k)}{g'(x^k)} \\ \therefore x^{k+1} - x^* &= x^k - x^* - \frac{g(x^k) - g(x^*)}{g'(x^k)} \\ &= -\frac{(g(x^k) - g(x^*) + g'(x^k)(x^* - x^k))}{g'(x^k)} \end{aligned}$$

If we assume that $f \in C^3$ (or $g \in C^2$), then using truncated Taylor series,

$$g(x^*) = g(x^k) + g'(x^k)(x^* - x^k) + \frac{1}{2}g''(\bar{x}^k)(x^* - x^k)^2$$

where $\bar{x}^k \in LS(x^*, x^k)$. Therefore,

$$x^{k+1} - x^* = \frac{1}{2}\frac{g''(\bar{x}^k)}{g'(x^k)}(x^k - x^*)^2$$

Now, at the k-th iteration of the newton algorithm, what we have is a x k plus 1 is equal x k minus g of x k by g dash x k. Note that, we are talking about one-dimensional optimization problem. So, this is the equation that we have for the k-th iteration of the Newton algorithm. Now, we are interested in finding out the relationship between x k plus 1 minus x star and x k minus x star and what kind of convergence that the Newton algorithm gives if it converges to x star. So, we subtract x star from both the sides and also subtract g of x star from the numerator. Now, this does not make a difference because we have already assumed that g of x star is 0, x star is a solution point where the gradient vanishes.

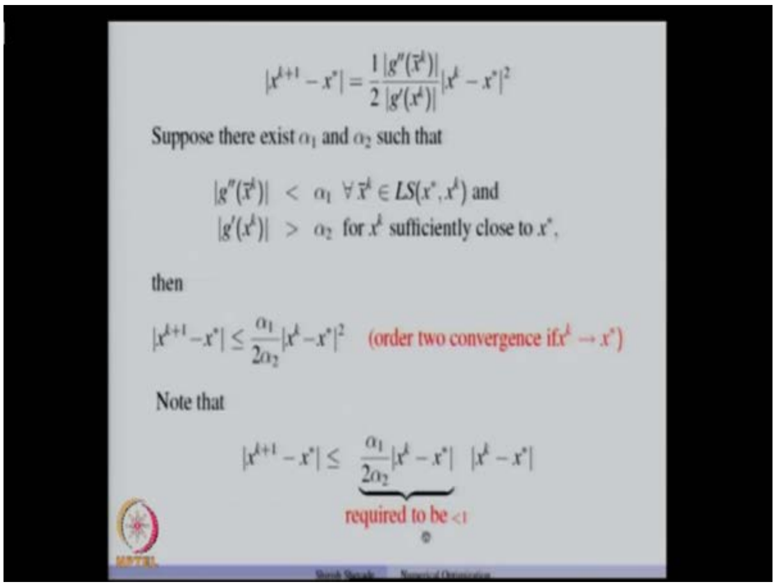
Now, this can be rewritten in this form and the numerator here is in terms of g of x k and g of x star. So, suppose we want to approximate the function g by a quadratic function at x k we need to assume that f is a thrice continuously differential function or g is a twice continuously differentiation and differentiable function and then we use truncated Taylor series to write g x star as a function approximated at x k. So, g of x star is nothing but, g x k plus g dash x k into x star minus x k plus half g 2 dash x bar k into x star minus x k square, where x bar k is point on the open line segment joining x star and x k.

Now, we can use this in this previous equation and therefore, what we get is x k plus 1 minus x star is nothing but, half of g 2 dash x bar k divided by g dash x k into x k minus x star square. So, we have use this x approximation of g x star at x k in this equation to get this equation.

Now, we will see that we have got some relationship between x k plus 1 minus x star and x k minus x star square. Now, if you recall the definition of convergence of a algorithm, this would turn out to be order 2 convergence, provided we make this quantity which is independent of x k because if you look at this quantity, this quantity is still dependent on x k and x star.

So, if you can make it independent of x k then or get a bound on this quantity, then what will have is x k plus 1 minus x star equal to some constant into x k minus x star square and that will tell us that its Newton method is, if it gives a convergent sequence it converges at a rate 2, it has a convergence of order 2 and the rate will be denoted by this quantity. So, let us look at this quantity further.

(Refer Slide Time: 51:08)



$$|x^{k+1} - x^*| = \frac{1}{2}\frac{|g''(\bar{x}^k)|}{|g'(x^k)|}|x^k - x^*|^2$$

Suppose there exist α_1 and α_2 such that

$$\begin{aligned} |g''(\bar{x}^k)| &< \alpha_1 \ \forall \bar{x}^k \in LS(x^*, x^k) \text{ and} \\ |g'(x^k)| &> \alpha_2 \text{ for } x^k \text{ sufficiently close to } x^*, \end{aligned}$$

then

$$|x^{k+1} - x^*| \le \frac{\alpha_1}{2\alpha_2}|x^k - x^*|^2 \quad \text{(order two convergence if } x^k \to x^*\text{)}$$

Note that

$$|x^{k+1} - x^*| \le \underbrace{\frac{\alpha_1}{2\alpha_2}|x^k - x^*|}_{\text{required to be} <1} \ |x^k - x^*|$$

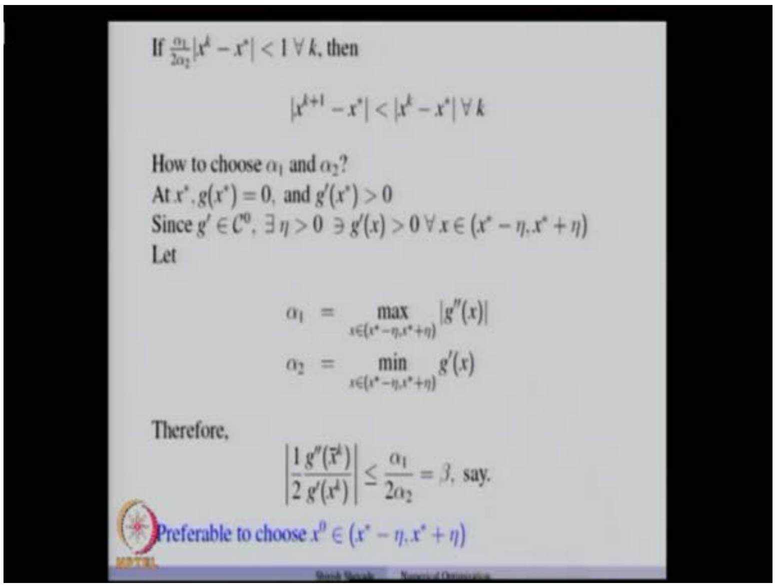
Now, taking the absolute value on both sides, we get this equation. Now, suppose we try to bound this numerator by some quantity alpha 1 and denominator by a quantity alpha 2 and note that alpha 1, alpha 2 both are greater than 0.

So, suppose there exist alpha 1 and alpha 2, both greater than 0. Such that this quantity in the numerator is less than alpha 1 for all x bar k because this quantity depends on x bar k. So, for all x bar k in the on the open line segment joining x star and x k and the quantity in the denominator is greater than alpha 2, where alpha 2 is greater than 0, for x k sufficiently close to x star, then we can write this as x k plus 1 minus x star is less than or equal to alpha 1 into y 2, alpha 2 into x k minus x star square. So, this quantity alpha 1 and alpha by 2 alpha 2 is a constant which is positive and therefore, what we get is that the newton algorithm if it converges to x star has a order of convergence 2. So, if x k converges to x star then this has order of convergence 2.

Now, if we rewrite this equation so, what we get is mod of x k plus 1 minus x star is less than or equal to, suppose if you split this second term into 2 terms x k mod of x k minus x star into mod of x k minus x star then, when we use newton algorithm what we want is that the distance of x k from x star should be or the distance of x k plus 1 from x star should be less than the distance of x k from x star, or in other words x the distance towards the optimal point should reduce after every iteration.

So, which means that this quantity is required to be less than 1, only then we can ensure that the distance from the new point x k plus 1 minus x star will be less than the distance from the distance of the current point from x star.

(Refer Slide Time: 53:51)



If $\frac{\alpha_1}{2\alpha_2}|x^k - x^*| < 1 \; \forall \, k$, then

$$|x^{k+1} - x^*| < |x^k - x^*| \; \forall \, k$$

How to choose α_1 and α_2?
At $x^*, g(x^*) = 0$, and $g'(x^*) > 0$
Since $g' \in C^0$, $\exists \, \eta > 0 \ni g'(x) > 0 \; \forall \, x \in (x^* - \eta, x^* + \eta)$
Let

$$\alpha_1 = \max_{x \in (x^* - \eta, x^* + \eta)} |g''(x)|$$
$$\alpha_2 = \min_{x \in (x^* - \eta, x^* + \eta)} g'(x)$$

Therefore,

$$\left| \frac{1}{2} \frac{g''(\bar{x}^k)}{g'(x^k)} \right| \leq \frac{\alpha_1}{2\alpha_2} = \beta, \text{ say.}$$

Preferable to choose $x^0 \in (x^* - \eta, x^* + \eta)$

Now, if this has to be less than 1, for all k then what we have is mod of x k plus 1 minus x star less then mod x k minus x star, for all k, if this alpha 1 by 2 alpha 2 into mod x k minus x star is less than 1.

Now, this requires the following question that needs to be answered that, how do we choose alpha 1 and alpha 1 so, such that this holds? And if this holds, then we know that the distance from the new point from x star, distance of the new point from x star from is less than the distance of the current point from x star. So, how do choose this alpha 1 and alpha 2 is the question. And so, let us see how to do that? So, at x star we know that the gradient vanishes and then the second derivative is greater than 0. Second derivative of f is greater than 0.

Now, since g dash is a continuous function or f 2 dash is a continuous function, there exist some eta which is greater than 0, such that g dash x is greater than 0 in the interval x star minus eta to x star plus eta. So, if you take any x in this interval x star minus eta to x star plus eta then for that x g dash is greater than 0 because of the continuity of g dash and therefore, if you choose alpha 1 to be max of mod of g 2 dash x and alpha 2 to be min of g dash x in this neighbourhood.

Now, in this neighbourhood g dash x is greater than 0. So, min of g dash x will always be greater than 0. So, that means that alpha 2 will always be greater than 0 in the eta neighbourhood of x star. So, if you choose alpha 1 and alpha 2 in this way, what we have is half of g 2 dash x bar k by H dash x k will be less than or equal to alpha 1 by alpha 2.

And let us call that quantity as beta. Now, the next step is to choose x naught. So, it is preferable to choose x naught to be in this interval, x star minus eta to x star plus eta because in this interval, we know that g dash x is greater than 0 and alpha 1 is a max of this quantity and alpha 2 is max of this quantity. So, if x 0 is chosen in this interval, we expect that all the other iterations and if this holds for at every iterations, all the other iterations also will be in that interval and therefore, the algorithm is expected to converge to x star. So, let us see whether the algorithm does converge to x star. Now, we will do that in the next class.

Thank you.