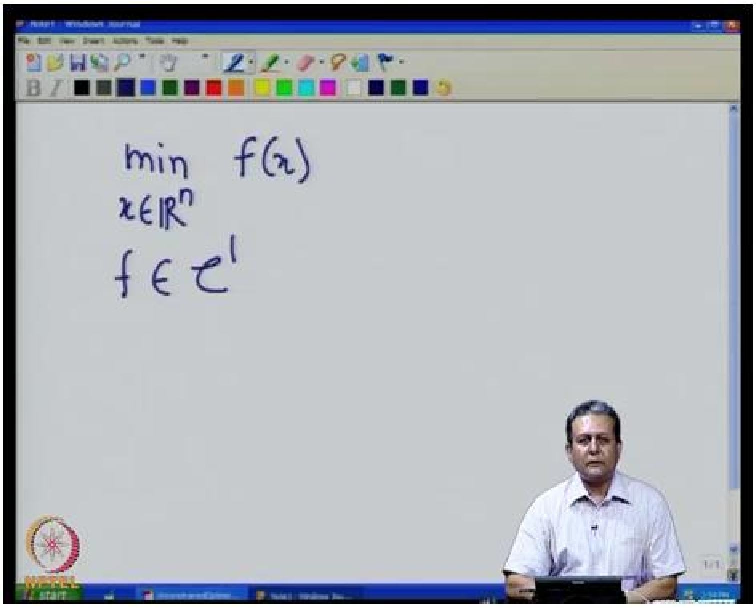
Numerical Optimization
Prof. Shirish K. Shevade
Department of Computer Science and Automation
Indian Institute of Science, Bangalore

Lecture - 12
Global Convergence Theorem

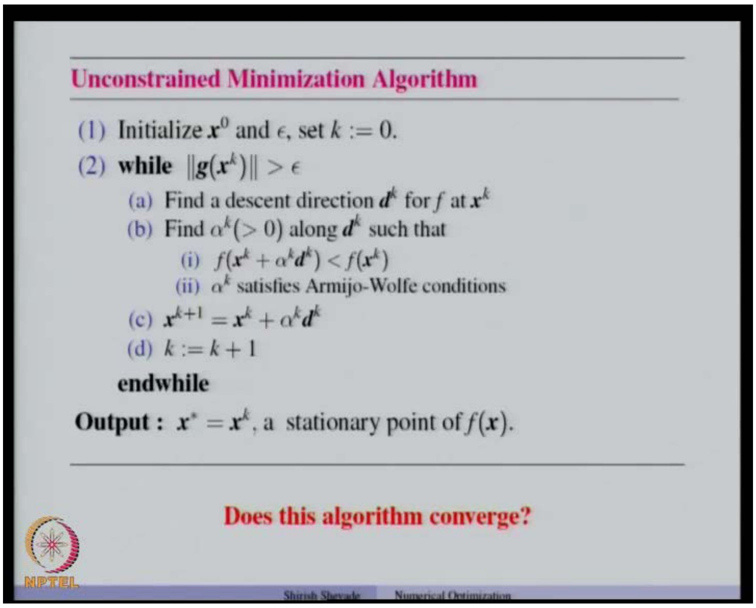
(Refer Slide Time: 00:52)



So, welcome back to the series of lectures on numerical optimization. In the last class, we started discussing about conceptual algorithm for solving an optimization problem especially a minimization problem. So, the problem that we were looking at is minimize f of x, x belongs to R n. This is an unconstrained optimization problem and we assume that f belongs to the class of continuously differentiable functions. Then, in the last class, we looked at different ways to ensure that there is a sufficient decrease in the objective function and also, the step lengths are not small.

So, initially we gave an example which showed that the sufficient decrease in the objective function as well as the step length are important issues. And if they are not addressed properly, an algorithm can converge to a point, which is not a local minimum or even in some cases, it may not converge. So, afterwards we saw the Armijo-Goldstein conditions or Armijo-Wolfe conditions to take care of sufficient decrease as well as the sufficient step length. Now, if we ensure that those conditions are satisfied, what is the guarantee that a typical optimization algorithm will converge and that we will study that in today's class.

(Refer Slide Time: 02:18)

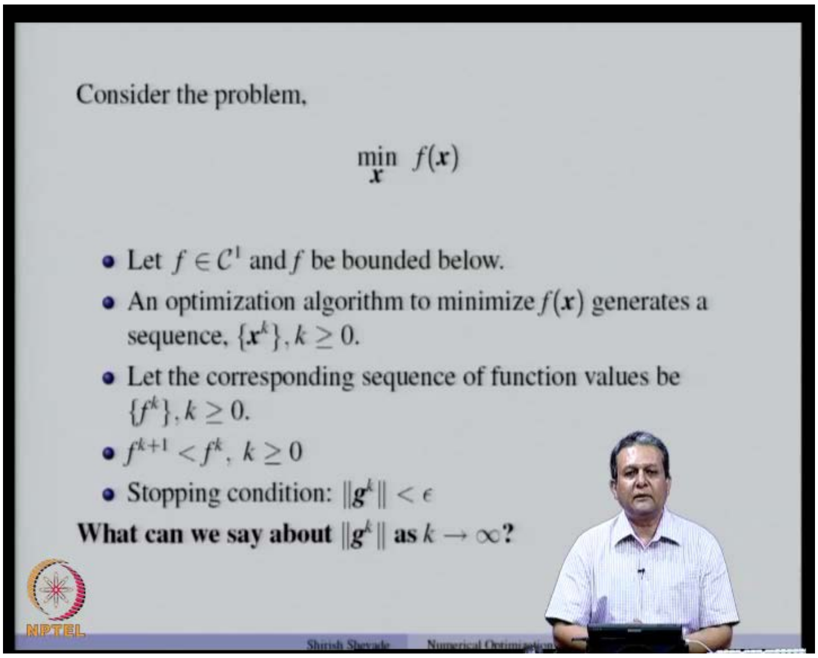


So, this was our optimization algorithm. So, we initialized there point x naught and we had epsilon which is a tolerance parameter for the norm of the gradient said the iteration count to 0 and while the norm of the gradient at a given point x k is greater than epsilon. The first step that we do is to find a descent direction d k for f at x k and the second step is to find alpha cases that there is a decrease in the objective function, but what is alpha k as we saw last time should be chosen, such that either Armijo-Goldstein conditions or Armijo-Wolfe conditions are satisfied.

So, let us assume that alpha k chosen, so that it satisfies Armijo-Wolfe conditions. Now, remember that this alpha k is greater than 0. Now, after having found value of alpha k and these two conditions are satisfied, then we go to the next point. So, the new point is nothing, but x k plus alpha k d k. The iteration count is increased by 1 and the procedure is repeated till the norm of the gradient is less than or equal to epsilon. Now, remember that this is just one of the conditions that one can use for stopping an optimization algorithm. We saw some more conditions that could be used for stopping an optimization algorithm depending upon the application. Now, as an output of this algorithm, what we get is x k which is nothing, but a stationary point x star of given function f of x.

Note also that we are not checking any second order information related conditions in this algorithm. That is why we end up in a stationary point, but which is not always guaranteed to be a local mean. Some more checks need to be done to ensure that this stationary point is indeed a local menu.

(Refer Slide Time: 04:49)

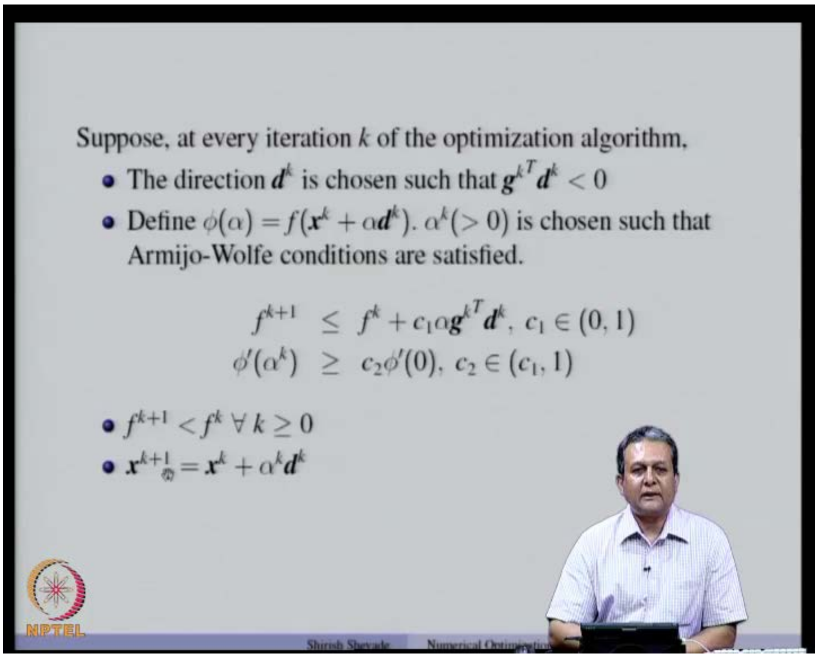


Now, the important question is that does this algorithm converge. So, we will answer this question in today's class. So, this is the problem that we are looking at to minimize a function f of x object to x belongs to r n. Now, let us assume that f belongs to the class of continuously differentiable functions and also, f is bounded below. Now, these are reasonable assumptions because in practice become across lots of functions which are continuously differentiable and also bounded below. So, it makes sense to minimize a function which is bounded below.

Now, as we saw earlier an optimization algorithm to minimize the function f of x generates a sequence x k, where k goes from 0 to infinity. So, that denote to corresponding sequence of function values by f k. So, f k will be a short and notation for f of x k. Here, k is again going from 0 to infinity. So, an optimization algorithm generates x k and corresponding function values of k. Now, we do not know anything about whether x k converges or not, but the function f is bounded below. So, what can we say about the sequence f k. Now, one thing one has to note is that in every iteration, the function value decreases. So, that means that f of k plus 1 f of the x k plus 1 is less than f of x k for all k. So, that means we have got sequence f k which is decreasing sequence. So, it is not only decreasing, but it is monotonically decreasing sequence and further the sequence is bounded below. So, these two properties of this sequence are very important.

Now, the stopping condition for the algorithm is that norm of g k less than or equal to epsilon. Now, as k goes to infinity, what can we say about norm g k? Ideally, what we expect is that we expect the algorithm to terminate at a point where the norm of g k is 0 or less than or equal to epsilon in the practical case for some finite k. This is what we expect ideally, but does that always happen. Let us see.

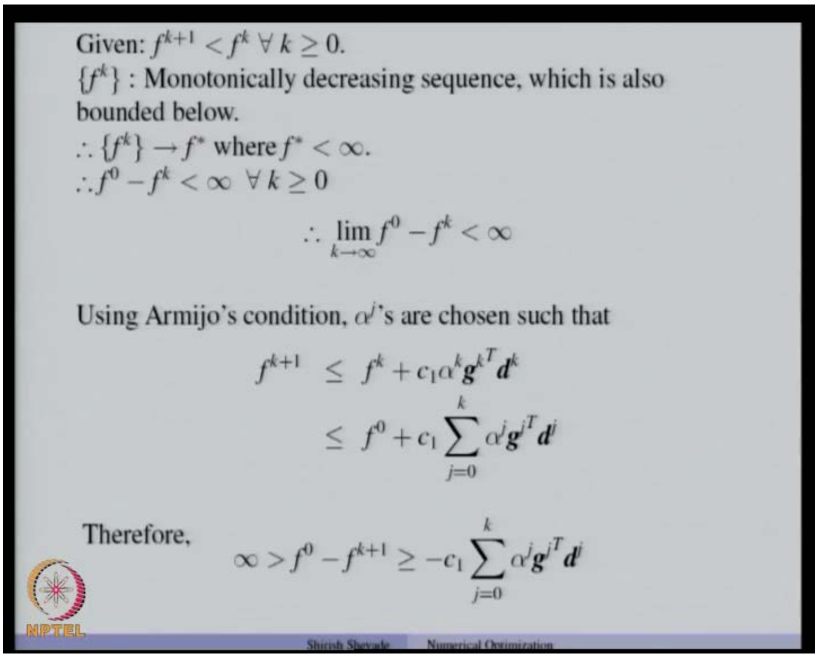
(Refer Slide Time: 07:32)



Now, let us assume that at every iteration of the optimization algorithm, the following conditions hold. So, the direction d k which is chosen as part of the optimization algorithm is such that g k transverse d k is less than 0 which guarantees that d k is descent direction. So, the first thing that we have to ensure is that the d k chosen, the direction d k chosen is always a descent direction and that will be guaranteed if you ensure that g k transpose d k is less than 0.

Now, let us define a function phi alpha to be f of x k plus alpha d k and let us also assume that alpha k which is a positive quantity is chosen, such that Armijo-Wolfe conditions are satisfied. So, f k plus 1 is less than or equal to f k plus c 1 into alpha into g k transpose d k, where c 1 is number in the open interval 0 to 1. So, this is Armijo's condition and Wolfe conditions says that phi prime alpha k is greater than or equal to c 2 phi prime 0. So, the first condition ensures that there is a sufficient decrease and the second condition ensures that the step length is not small. Note also that the c 2 lies in the range c 1 2 1. So, both c 1 and c 2 are positive fractions and c 1 is less than c 2. Now, given that these conditions are satisfied at every iteration of the algorithm. So, this will automatically guarantee that the value of the function decreases in every iteration. So, after finding alpha k, we do the update x k plus 1 to be x k plus alpha k d k.

(Refer Slide Time: 09:52)



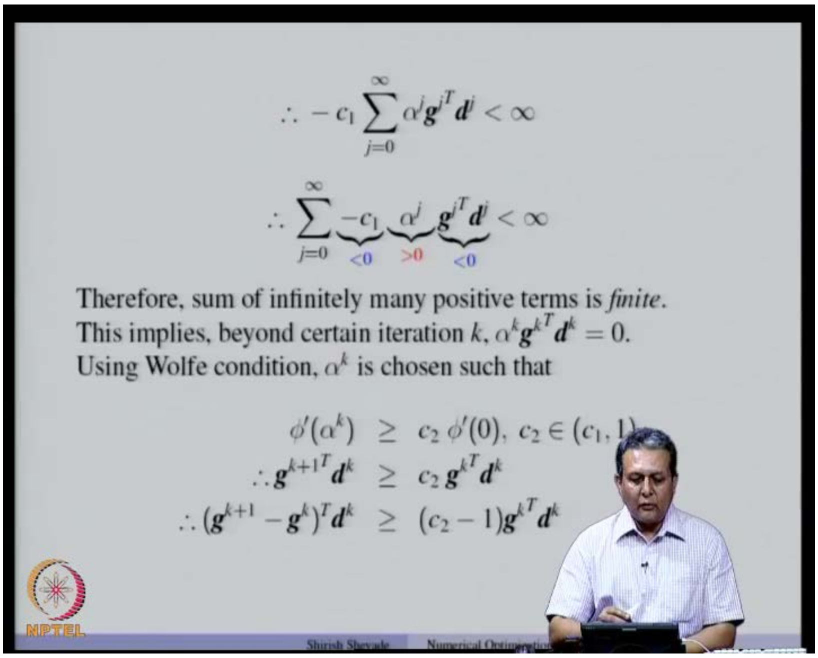
Now, we are given that there is a decreasing sequence of function values that is f of x k plus 1 is less than f of x k for all k greater than or equal to 0, which means that we have monotonically decreasing sequence of function values and f is bounded below. So, we have monotonically decreasing sequence of function values and the sequences also bounded below by some quantity, which means that this sequence will converge to some quantity. So, let us assume that the sequence converges to f star. So, remember that we still have not talked about the convergence of x k 2 x star, but we are just talking about the convergence of f of f k to some quantity f star, where f star is an finite quantity.

Now, we have that f 0 minus f k is less than infinity because every time we are going to reduce the function value. So, the function value at the k-th iteration will be certainly less than f 0 and therefore, f 0 minus f k will be less than infinity and therefore, we can say that k tends to infinity f 0 minus f k is less than infinity because this f 0 minus f k less than infinity whole for all k greater than or equal to 0. So, certainly this limit is going to be a finite limit. Now, given this fact, let us look at Armijo's condition. So, Armijo's condition chooses some alpha j's, such that f k plus 1 is less than or equal to f k plus c 1 alpha k g k transpose d k, where c 1 is the constant in the range 0 to 1.

Now, if we write f k in terms of alpha k minus 1 and g k minus 1, d k minus 1 and f k minus 1 in terms of alpha k minus 2, g k minus 2 and d k minus 2, finally we can write f k in terms of f 0 and all the alpha j's, g j's and d j's going from 0 to k. Therefore, f of x k plus 1 is nothing less than or equal to f of x 0 plus c 1 into sum over alpha j g j transpose d j j going from 0 to k.

Now, we know that f 0 minus f k is less than infinity. So, f 0 minus f k plus 1 is also less than infinity. Therefore, f 0 minus f k plus 1 which is less than infinity, but then f 0 minus f k plus 1 is greater than or equal to minus of the second quantity which is given here, which means that f 0 minus f k plus 1 is greater than or equal to minus c 1 into sum over alpha j g j transpose d j, where j is going from 0 to k.

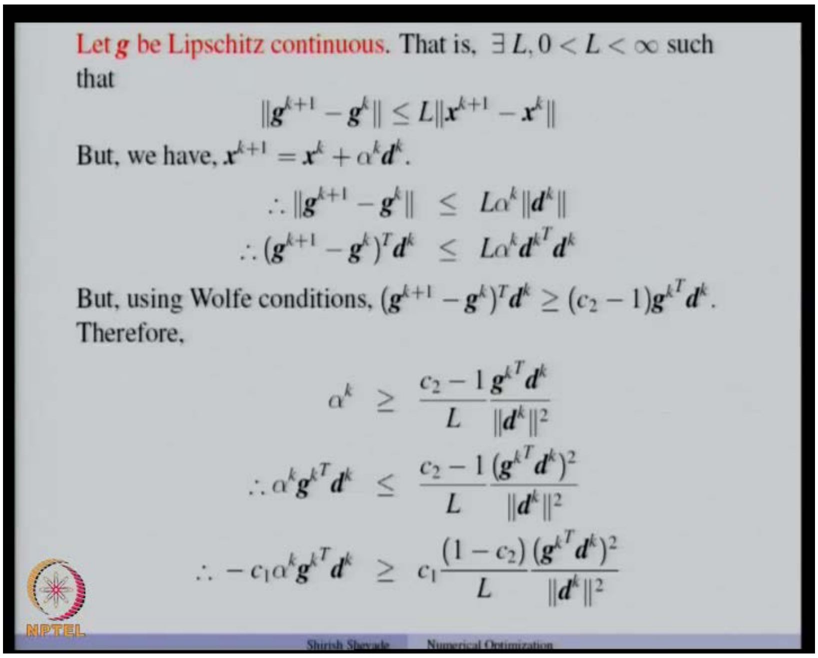
(Refer Slide Time: 13:09)



Now, let us look at this quantity in detail. So, from the previous expression what we have is minus c 1 into alpha j g j transpose d j summed over j going from 0 to infinity is less than infinity. Now, what about these quantities? Now, remember that c 1 is a positive fraction. So, minus c 1 is less than 0. Our algorithm ensures that alpha j is always greater than 0 and also, d j is a descent direction. So, g j transpose d j is less than 0. So, we have a quantity minus c 1 which is less than 0 alpha j which is greater than 0 and g j transpose d j less than 0. So, this entire quantity here is positive quantity and what this expression says is that we have sum of infinitely many positive quantities which is less than infinity. That means that the sum is finite. Now, if the sum of infinitely many positive quantities is finite, so that means that beyond certain k, certain index k, each of this entire expression becomes 0. So, that is beyond certain iteration k alpha k g k transpose d k is 0 because 7 is a constant. So, that cannot become 0. So, the only possibility is that alpha k g k transpose d k becomes 0 beyond certain iteration k, otherwise this condition that the sum of infinitely many positive terms is finite may not hold.

So, now let us see how does this happen. Now, let us try to get a lower bound for the quantity minus c 1 into sigma j going from 0 to infinity alpha j g j transpose d j and suppose, if we get that lower bound independent of d j, then we have both upper bound and lower bound for this quantity and then, we will show that this indeed is true, the alpha k g k transpose d k 0 beyond certain iteration number k. So, for that purpose, let us look at Wolfe conditions. So, according to Wolfe condition, the step length alpha k is chosen such that phi prime alpha k is greater or equal to c 2 into phi prime 0. Here, c 2 is constant in the open intervals c 1 to 1.

(Refer Slide Time: 16:38)



Now, phi prime alpha k, if you recall the definition of phi alpha phi alpha is nothing, but f of x k plus alpha d k. So, phi prime alpha k is nothing, but g k plus 1 transpose d k and that will be greater than or equal to c 2 into phi prime 0 which is nothing, but g k transpose d k. Now, this can be written as so if you subtract g k transpose d k from both sides, so what we can write is g k plus 1 minus g k transpose d k is greater than equal to c 2 minus 1 g k transpose d k. Now, how do we control this g k plus 1 minus g k transpose d k? For that we need some assumption and that assumption is that the function g is Lipschitz continuous.

Now, by Lipschitz continuity what we mean is that there exist some finite positive constant l, such that norm of g k plus 1 minus g k is less than or equal to l into norm of x k plus 1 minus x k. So, what it means is thatif we move from x k to x k plus one the change in the gradients from g kx plus from g k to g k plus 1. Now, if we take the difference of these two gradients and take the norm that norm is always bounded above by this quantity, then note that l is finite positive constant. So, for a given function f, it is reasonable to assume that the gradient of the function does not shoot out arbitrarily. The difference between the two successive gradients is always bounded above by some quantity.

Now, if we make this assumption, then we can use the fact that x k plus 1 is nothing, but x k plus alpha k d k and therefore, x k plus 1 minus x k is nothing, but alpha k d k. So, we plug that alpha k d k. Here, alpha k is a positive constant, alpha k is a positive parameter and d k is a vector. So, what we have is norm of g k plus 1 minus g k is less than or equal to alpha into l into alpha k into norm d k. So, this was obtained by substituting the previous expression. Therefore, what we can write is that g k plus 1 minus g k transpose d k is always less than or equal to l alpha k d k transpose d k. Remember that we are trying to bound this g k plus 1 minus g k transpose d k in the earlier expression. Therefore, using Wolfe condition, where we have g k plus 1 minus g k transpose d k is greater than or equal to c 2 minus 1 g k transpose d k. So, we use this and this together to write a relation between l alpha kd k transpose d k and c 2 minus 1 into g k transpose d k. Therefore, using these two quantities, what we have is alpha k is greater than or equal to c 2 minus 1 by l into g k transpose d k by norm d k square.

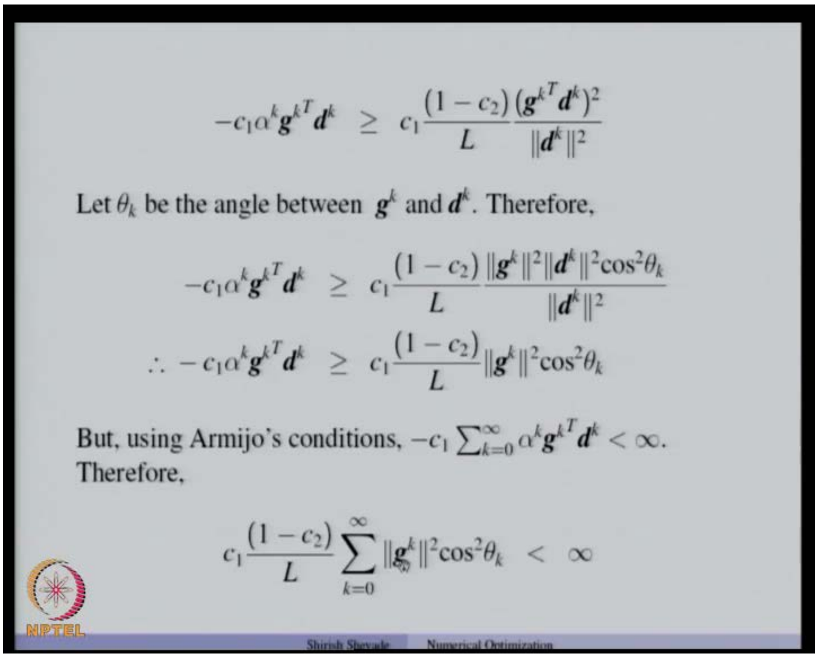
Now, if you multiply throughout by g k transpose d k, remember that the direction d k is chosen such that g k transpose d k is always less than 0. So, the quantity g k transpose d k is less than 0. So, you multiply this inequality in negative quantity. The inequality reverses is direction and therefore, what we get is alpha k g k transpose d k is less than or equal to the first time remains as it is and g k transpose d k is multiplied by g k transposed k. So, we get a square of this quantity and divided by the norm of d k square which remains as it is.

Now, remember that we were trying to get a bound 1 minus c 1 into alpha k g k transpose d k. Now, if we multiply throughout by minus c 1, so again the inequality reverses its direction to minus c 1, but in this case what we can do is that the negative sign will merge with the expression c 2 minus 1. Therefore, what we have on the right side is c 1 into 1 minus c 2 by l into g k transpose d k square your return norm g k d k square. So, we have got a bound, lower bound on minus c 1 alpha k g k transpose d k, that is minus c 1 alpha k g k transpose d k is greater than or equal to the quantity which is there on the right side.

Now, we have to get rid of the term d k here because every time the direction changes, this quantity is going to change. So, we get the bound on minus c 1 alpha k g k transpose d k which is independent of d k. We can use g k in this expression, but we do not want d k in this expression. Now, to get rid of d k, we have to make use of between the two (())

a and b. So, if a and b are two (()), then a transpose b is nothing, but norm a to nom b into cos of the angle between the two (()). So, we make use of that sort.

(Refer Slide Time: 22:00)

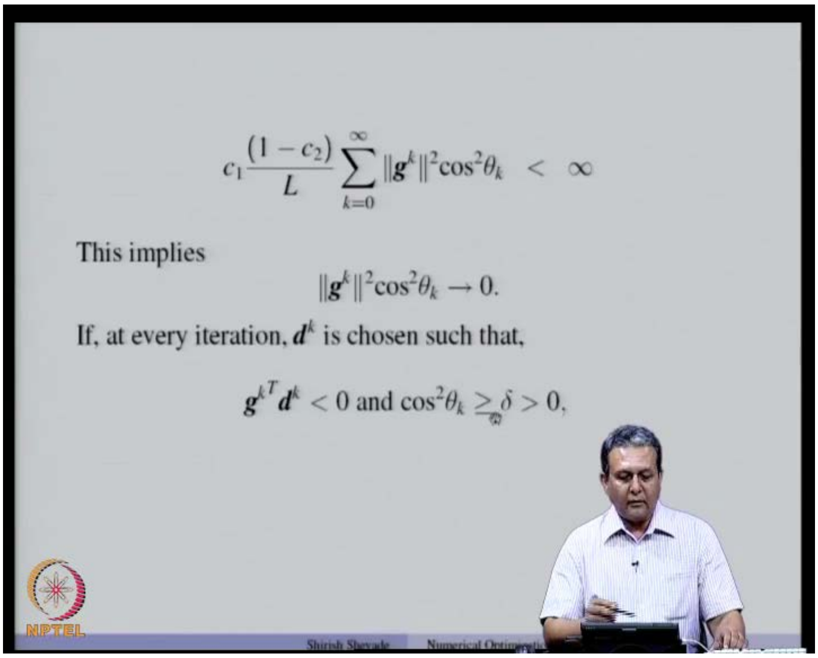


So, let us define theta k to be the angle between g k and d k. Now, if theta k is this angle, then we know that g k transpose d k is nothing, but norm g k into norm d k into cos theta. Now, the square of that will give us norm g k square into norm d k square into cos theta k. The denominator remains as it is. Now, we will see that the norm d k square get cancelled here and therefore, we will get a bound on minus c 1 alpha k g k transpose d k intense of norm g k and cos square theta k. Remember that c 1, c 2 l are all constants. So, we do not have to worry about them. Therefore, we can write this as minus c 1 into alpha k into g k transpose d k and that quantity is greater than equal to c 1 into 1 minus c 2 by l norm g k square cos square theta k. So, this is a layer bound on minus c 1 alpha k g k transpose d k which is independent of d k, but it does use the quantity theta k which depends on d k. So, we can replace cos square theta k by from constant.

Now, if you use a Armijo condition, Armijo's condition says that minus c 1 sigma k going from 0 to infinity alpha k g k transpose d k is less than infinity. So, if we take a summation over k going from 0 to infinity, in this case that will hold provided that sum is greater than or equal to c 1 into 1 minus c 2 by l sigma k going from 0 to infinity norm g k square cos square theta k. Therefore, what we have is c 1 into 1 minus c 2 by l which is a constant. So, we have taken it out of the summation same and then, some over k from 0 to infinity norm g k square cos square theta k, which is less than or equal to minus c 1 into some k going from 0 to infinity alpha k g k transpose d k and using Armijo's conditions, we already know that this one is less than infinity. So, we have this sum less than infinity.

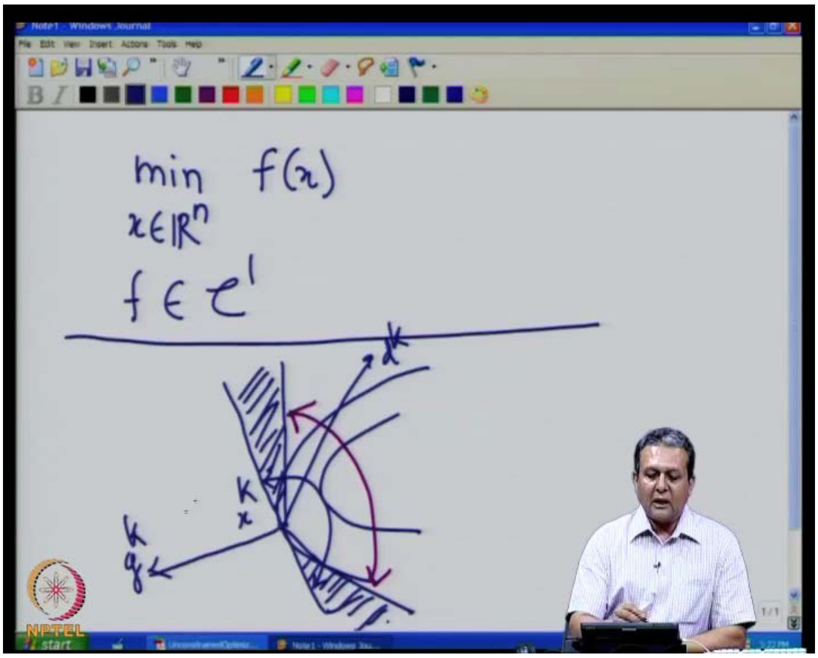
Now, c 1 is a positive quantity, c 2 is a positive fraction, so 1 minus c 2 is always greater than 0. l is also a finite positive number. So, all these quantities which are finite positive numbers, now if you look at the expression which is in the summation sign, so we have norm g k square cos square theta k and that is less than infinity. Now, there are infinitely many quantities which are positive. So, norm g k square is a non negative quantity cos square theta k is a non negative quantity. So, we have infinitely many positive quantities which is finite. So, that means that at some point of iteration, one of these terms could be going to 0. Now, so let us assume that suppose we force cos square theta k to be not 0, so suppose cos square theta k is always bounded below by certain quantity, constant quantity, then the only way that this expression is finite is that the norm g k square tends to 0 and we will see how to do that.

(Refer Slide Time: 26:08)



So, we have some more k going from 0 to infinity norm g k square cos square theta k less than infinity and that means that since some of infinitely many positive terms is finite. That means that some k norm g k square cos square theta k tends to be 0. Now, let us try to get some bound for cos square theta k. So, if the direction d k which is chosen at every iteration is such thatg k transpose d k is less than 0 and cos square theta k is a, then that are equal to delta which is a positive quantity. Then, this quantity cannot become 0 at any particular iteration and therefore, the only way this can happen is that norm g k square tends to be 0 or in other words, norm g k tends to be 0.

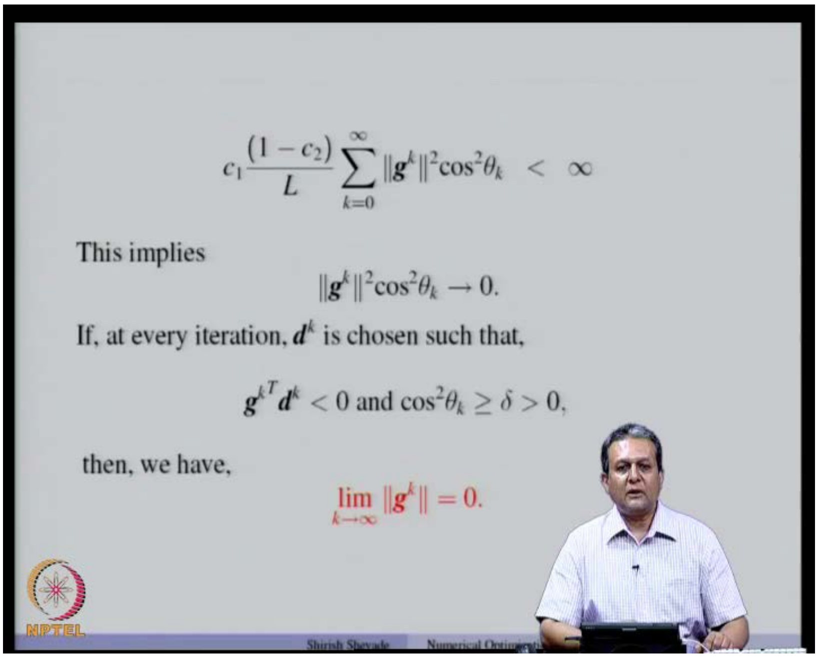
(Refer Slide Time: 27:26)



Now, how do we get this delta? So, the procedure is very simple. Suppose we have the connectives and this is our current point x k and this is the direction d kg k. So, that means that along these directions, the function value is going to increase. Then, we saw in the last class that it is this cone that we are interested in. So, if we are direction d k happens to be in this cone, open cone, then certainly g k transpose d k will be less than 0.

Now, what the previous condition assumes is that will chose the direction d k. So, we leave out some part of the cone. So, this part of the cone is left out and then, we will only take this cone. So, while taking this cone, but we are ensuring that g k transpose d k does not go close to 0 because g k transpose d k will be 0 when d k is either on this line or on this line. Now, by leaving out some part of the original open cone which is shown here and only considering this cone, we are making sure that g k transpose d k will be the angle between the g k and d k, that is theta k. Now, by choosing d k in this cone, we ensure that cos square theta k becomes greater than delta and delta is a positive quantity and therefore, we avoid g k transpose d k going close to 0.

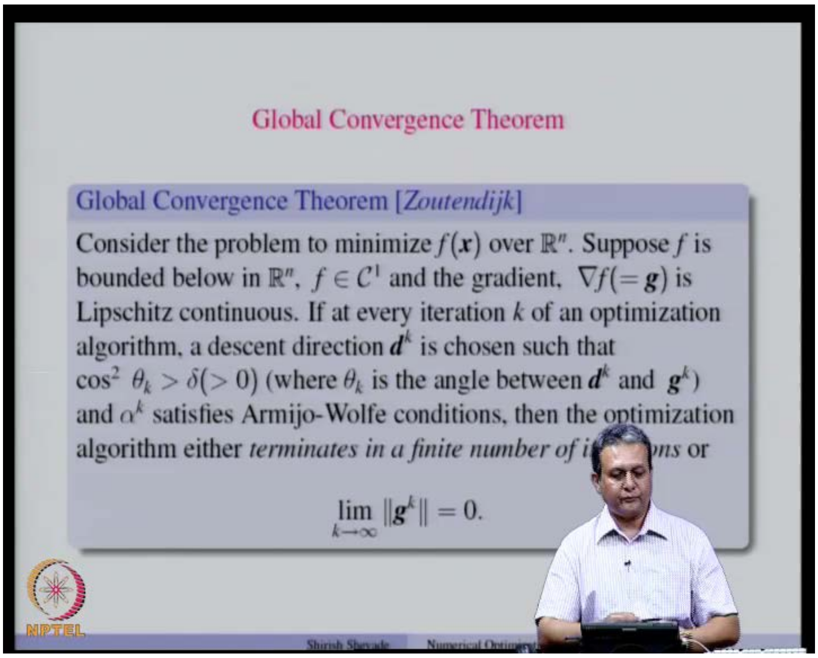
(Refer Slide Time: 29:52)



If we look at that then we have limiters k tense to infinity norm g k goes to 0. So, the important point is that at every iteration k, we get a descent direction d k which is g k transpose d k less than 0 which is given by or which is ensured by g k transpose d k less than 0, but not only that, we also make sure that the angle that d k makes with g k which is the angle theta k, so cos square theta k is greater than equal to some quantity delta which is positive quantity. So, we ensure that this cos square theta k does not go to 0 at any point of time and since, norm g k square cos square theta k tense to 0 is the only way this can happen is when norm g k square tense with 0 or in other words, norm g k a k tense to infinity goes to 0.

So, we saw that whenever we use this optimization algorithm, there are two possibilities. One is the possibility that there exist some finite k, where when the algorithm terminates that is there exist some finite k where nom of g of x k is less than or equal to epsilon. If that does not happen, then if we ensure that the angle that there descent direction makes with g k is such that cos square k theta is greater than or equal to delta and this step size which is chosen is such that it satisfies Armijo-Wolfe condition, then it is guaranteed that a synthetically norm of g k tense to 0. So, this is a very important theorem and remember that in this theorem, we did not use x 0 the initial point at any point of time. So, this result was derived irrespective of the initial point and this powerful result s called global conversions theorem. So, the reason for calling it global conversions theorem is that we can start from any x any initial point x 0 and if we follow certain conditions at every iteration, then the algorithm either terminates infinite number of iterations or limiters k tense to infinity norm g k goes to 0.

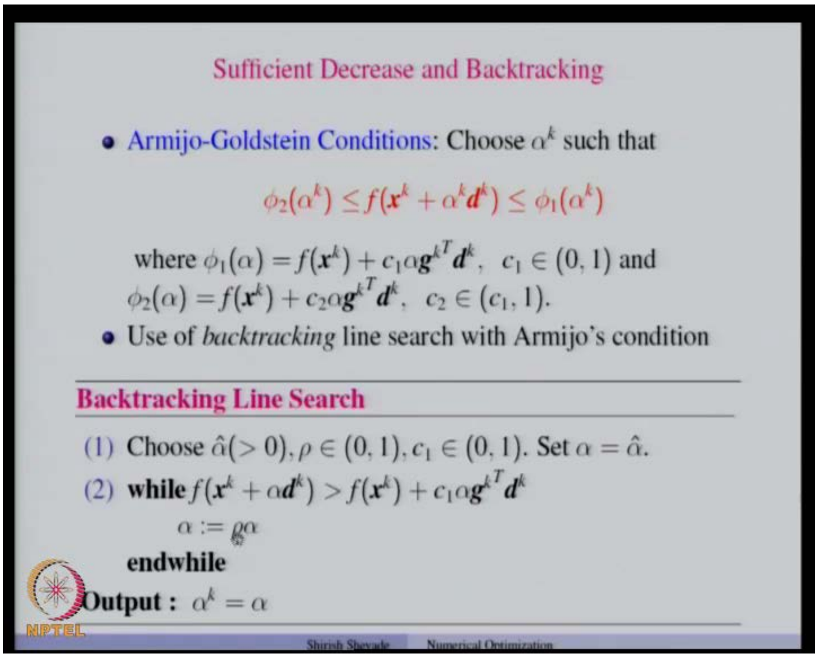
(Refer Slide Time: 32:50)



So, the optimization algorithm that we saw it does converge if all these conditions are ensured. So, this theorem is called global convergence theorem and this is due to Zoutendijk. So, let us look at those statement of the theorem. So, consider the problem to minimize f of x over R n. Now, suppose that f is bounded below in r n and f is continuously differentiable and gradient of f which you we have denoted gradient of f by g and we assume that the gradient of f is Lipschitz continuous. Then, if at every iteration k of an optimization algorithm, if we make sure that a descent direction d k is chosen such that if theta k is the angle between d k and g k, then cos square theta k is greater than some all positive quantity delta and this step length alpha k satisfies Armijo-Wolfe conditions. Then, the optimization algorithm either terminates in a finite number of iterations or as k tense to infinity limit of norm g k goes to 0. So, that means that we will reach a stationary point either in a finite number of iterations or as k tense to infinity will reach the stationary point.

So, this is the very important result in the theory of optimization and note that this is also independent of the initial point x 0. So, the only two conditions that mainly satisfies that the descent direction d k should make an angle with g k, such that cos square theta k is greater than delta and in every iteration, Armijo-Wolfe conditions are satisfied because these conditions are used to prove the convergence of the optimization algorithm, ok.

(Refer Slide Time: 35:01)

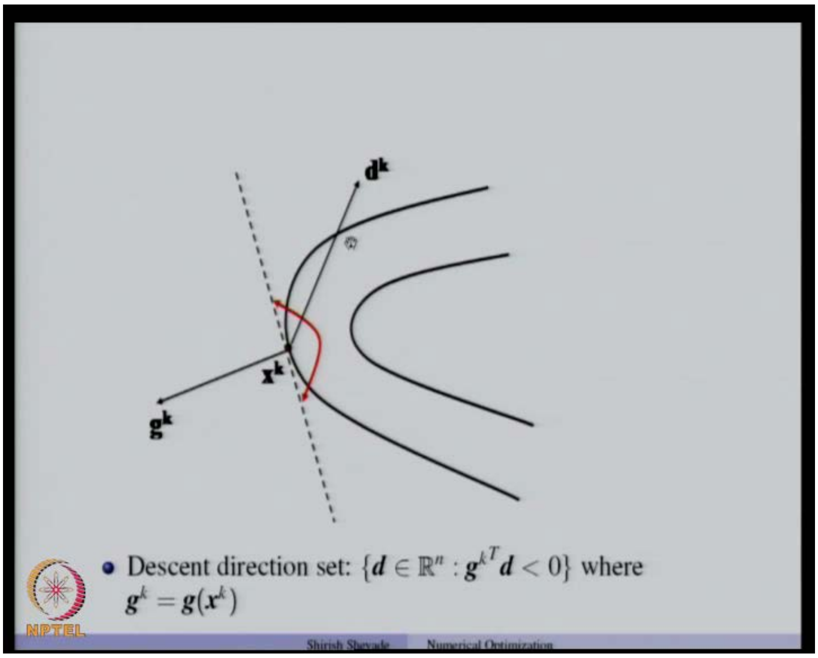


Now, many times while dealing with practical problems, it might be difficult to ensure that Armijo-Wolfe conditions or Armijo-Goldstein condition are satisfied. So, in such cases, it is proposed to use backtracking line search in combination with Armijo's condition. So, let us see how to do that. Note that Armijo-Goldstein conditions which choose alpha k, such that f of x k plus alpha k d k is less than or equal to phi 1 alpha k and f of x k plus alpha k d k is greater than or equal to phi 2 alpha k. So, phi 1 alpha k is a function corresponding to Armijo's condition and phi 2 alpha k is a function corresponding to Goldstein condition. We saw these conditions in last class.

Now, instead of checking whether Goldstein conditions are satisfied, one idea is to be backtracking line search with Armijo's condition. So, it is very simple to implement this idea. So, let us see how this algorithm works. So, the backtracking line search algorithm initially chooses some value of alpha act which is positive quantity. Those quantity in the range 0 to 1 c 1 is a positive fraction. So, initially alpha is alpha act. Now, while f of x k plus alpha d k is greater than f of x k plus c 1 alpha g k transpose d k, which means that when the Armijo's condition is not satisfied at a given alpha, reduce the alpha by multiplying with row. Row is a positive fraction, so the alpha gets reduced.

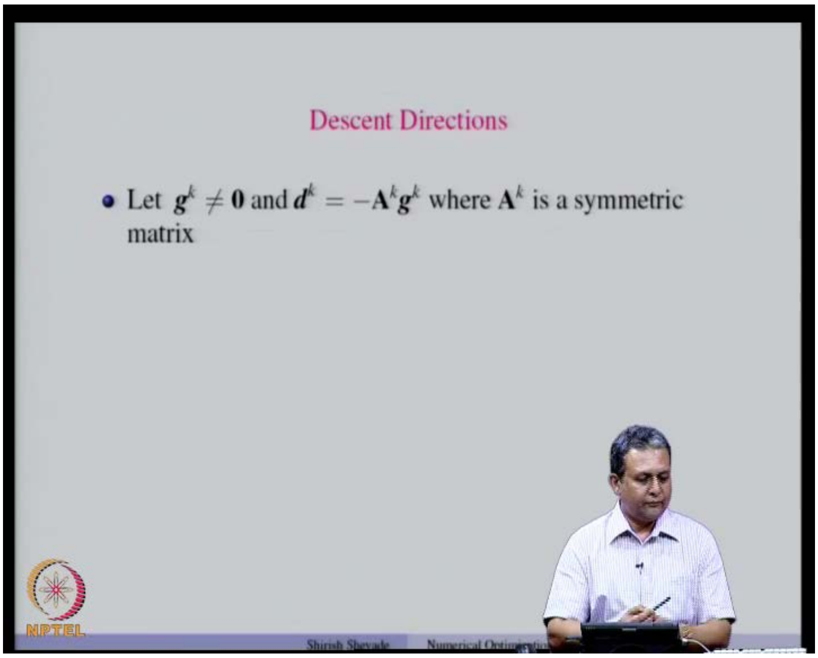
See, if given initial value of alpha which was nothing, but alpha act if Armijo's condition is not satisfied at that point reduce alpha and if at that point, the condition is not satisfied reduce alpha for the, so the process is repeated till Armijo's condition are satisfied. So, this will automatically ensures that you are certain from large step length and coming back to the smaller step length. So, it will automatically ensure that the smaller step length are avoided. So, finally, when the algorithm terminates, we get alpha k which is nothing, but the current value of alpha which satisfies this condition. So, many times this simple procedure of backtracking line search is used which will ensure sufficient decrease as well as it will avoid smaller step length. A good choose of alpha had for many of the algorithms is 1, but for some cases you have to reduce the initial value of alpha act.

(Refer Slide Time: 38:13)



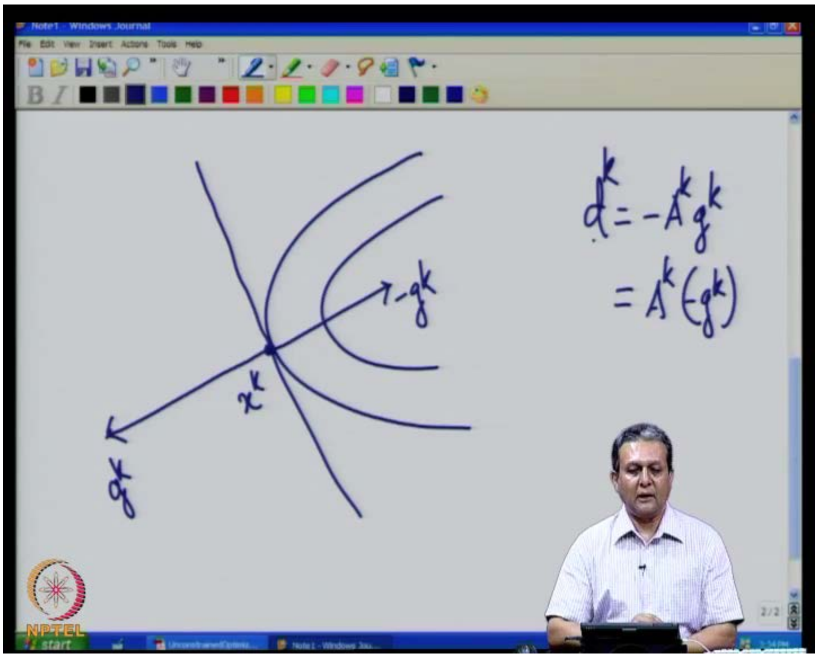
Now, let us look at the procedure to get different descent directions. Now, different optimization algorithms use different ways to determine the different direction. As saw in the last class that any direction d k, such that g k transpose d k is less than 0 will give a different direction. So, any direction lies in the open cone by this red arc is tended for descent direction. So, all these directions in this open cone from a descent direction set that is the set of all d's, such that g k transpose d is less than 0 and these continued to use the shortened notation g k for g of x k.

(Refer Slide Time: 39:23)



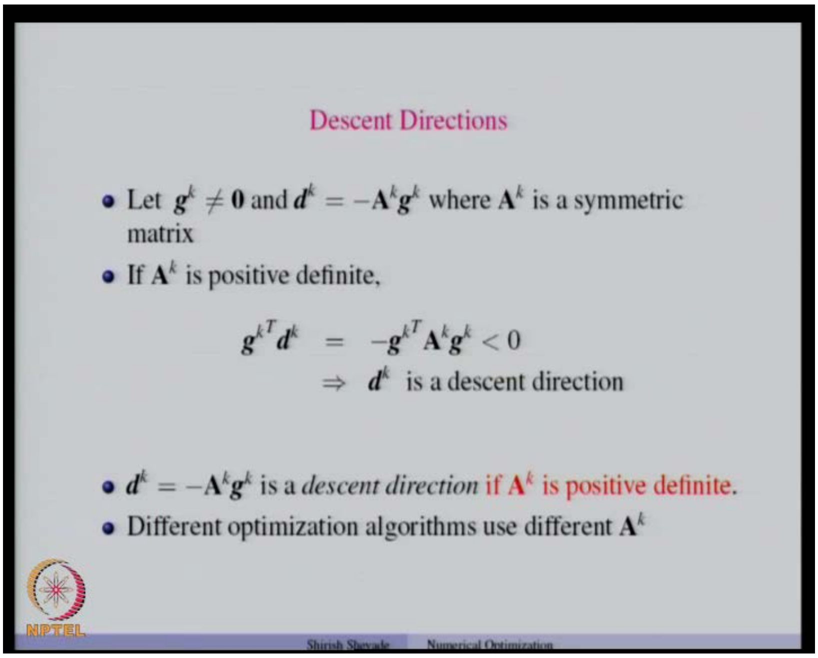
Now, we will look at different optimization algorithms which using some approximation of given function determine the descent direction d k, given descent direction d k. Let us assume that gradient of the current iteration k is not 0 and let us assume that d k is nothing, but minus a k into g k where a k is a symmetric matrix.

(Refer Slide Time: 39:52)



Now, let us see this is that suppose we have the controls and this is the direction g k which means that the function increases along this direction and this is the point x k. Now, we have said d k is nothing, but minus A k into g k or we can think of it as A k into minus g k. So, this is the direction minus g k and A k. Let us assume that A k is symmetric matrix. So, one can think of d k to be the rotation of the direction minus g k using the matrix A k. So, the matrix A k rotates the direction minus g k and the one, but d k such that g k transpose d k is less than 0. So, it all depends on how the rotation takes place using a matrix A k and it is at this place where different optimization algorithms of different direction finding strategies for optimization algorithms differ the way they choose A k, the sides, the rotation of minus g k and let us see what are the extra conditions that are needed to neither on A k which will ensure that d k is in this direction.

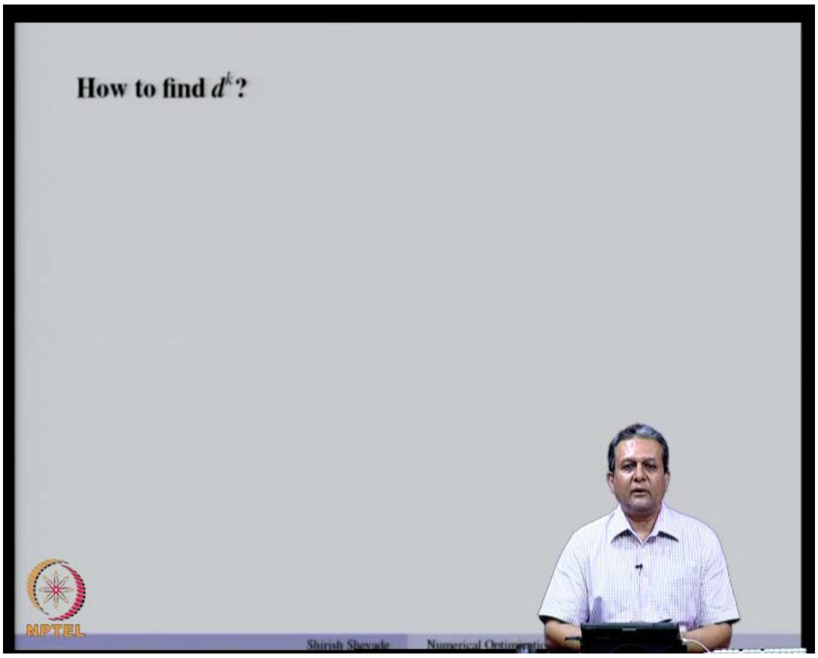
(Refer Slide Time: 41:51)



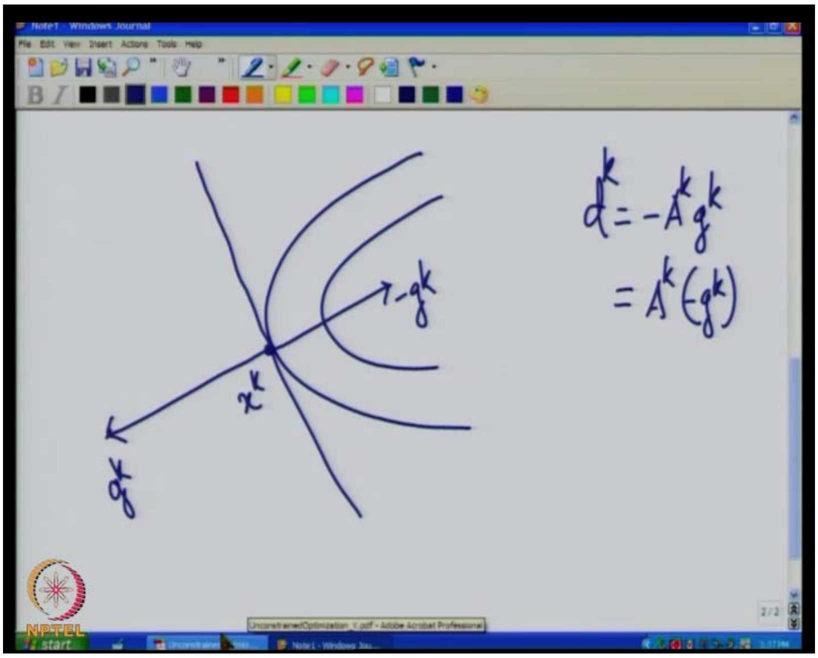
So, we have d k to be minus A k g k, where A k is a symmetric matrix. Now, let us write down what is g k transpose d. Now, g k transpose d k is nothing, but minus g k transpose A k g k. Now, if A k is positive definite quantity definite matrix, then minus g k transpose A k g k is a negative quantity. Now, we have g k tranpose d k to be less than 0. If A k is positive definite and g k transpose d k less than 0, means that d k is a descent direction. So, as long as A k is a symmetric positive definite matrix, d k equal to minus A k g k is descent direction and one can think of A k as matrix which will rotate the direction in g k suitably. So, d k is nothing, but minus A k g k is a descent direction if A k is a positive definite matrix. So, we have to keep this in mind.

Now, this implies positive definite matrix that one can think of is a identity matrix and this case, d k happens to be minus g k. Such directions are called steepest descent directions which see more about steepest direct, a descent directions soon. So, as I mentioned earlier, the different optimization algorithms use different A k's and therefore, these results in different descent directions and we will see some of those methods in the next two classes.

(Refer Slide Time: 43:47)

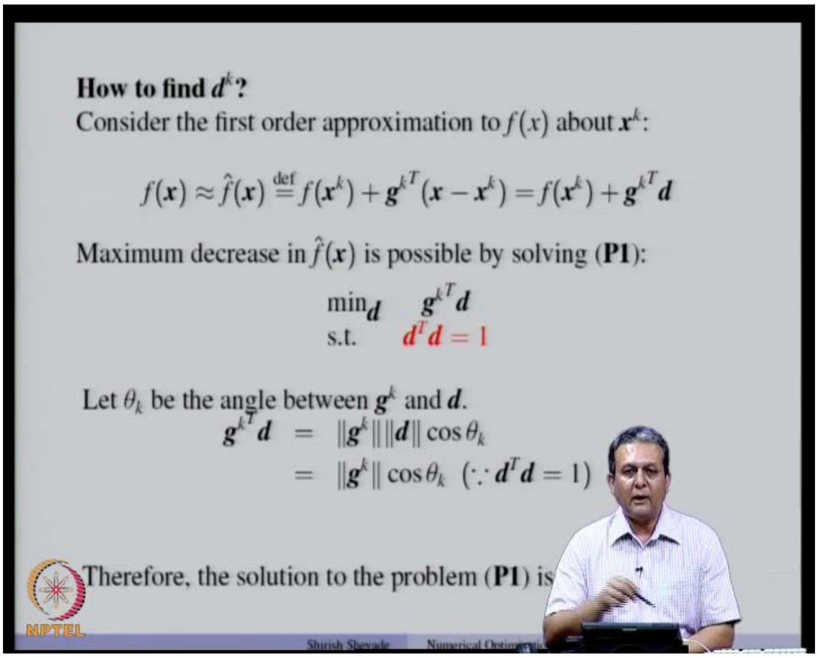


(Refer Slide Time: 44:15)



Now, the question is how to find d k, a descent direction. Now, how to function f x and one simple way to find the descent direction is to approximate the function bind of an function. So, this is an affine approximation of a given function. Now, given this affine approximation, you want to find out which is the direction which gives maximum decrease in the objective function with respect to the affine approximation of the objective function. So, we will see a method which does this.

(Refer Slide Time: 44:48)



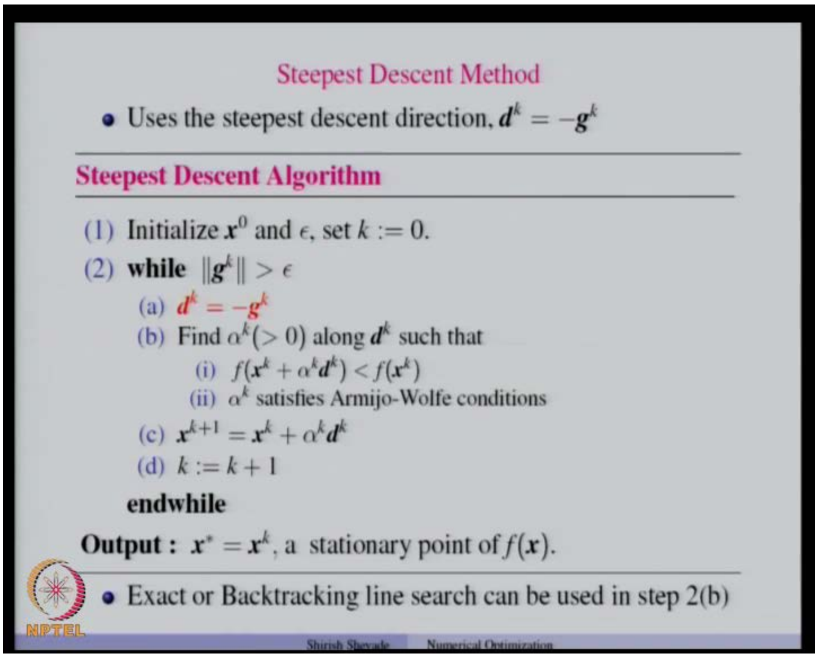
So, let us look at the first order approximation of f about x k. Now, using Taylors series, first order Taylor series, we can write f of x to be approximately equal f act x f act x is defined as f of x k. So, this is the approximation about x k of f about x k. So, f of x k plus a gradient of f act x k transpose x minus x k. Now, x is any point in the inputs place. So, x minus x k, let us call it as d and therefore, let us write this as f of x k plus g k transpose d. Now, this is the first order approximation of f about x k. Now, x k is known, so f of x k is in fix quantity. Then, gradient of f of x k is nothing, but g k that is also a fix quantity. So, the only unknown quantity here is d and what we were interested in that with respect to this first order approximation, what is the best direction d that one can get. So, since we are trying to minimize the function f of x, the best direction with respect to this first order approximation will be the direction which will minimize g k transpose d.

So, the maximum decrease in f act x, it should be the first order approximation of f of x is possible while solving the following problem with respect to d. So, we have to minimize g k transpose d. Now, d is any auditory vector in the inputs place. So, d can take value such that this quantity can be made arbitrarily small. So, to avoid that will enclose one constraint of d which is that the norm of d is 1 or norm d square is 1. So, this will ensure that we will not get any arbitrary vector d which will minimize g k transpose d.

Now, g k is a known quantity. So, again we will make use of the doubt product of two vectors here to split g k and d k to write the g k and d k transpose d in terms of the norms and the angle between them and then, see how to get d. So, let theta k be the angle between g k and d. Therefore, we can write g k transpose d is nothing, but norm g k into norm d cos theta k. We have seen this formula earlier and since norm d is norm d square is 1, norm d is also 1. Therefore, g k transpose is nothing, but norm g k into cos of theta k. Now, this is a fix quantity which is known to us. So, the only way to minimize g k transpose d is by minimizing cos theta k or choose d, such that g k transpose d is minimized when cos theta k is minimized.

Now, the main value of cos theta k is minus 1. Therefore, that occurs when d is equal to minus g k by norm g k because norm of d is square is 1. Therefore, the solution to this problem is nothing, but minus g k by norm d k. So, if you look at this direction, so this is the direction in which with respect to the first order approximation, there will be a maximum decrease in the objective function. So, such a direction is called steepest descent direction. So, in other words, steepest descent direction is the direction when the matrix A k is identity matrix in this case.

(Refer Slide Time: 49:36)

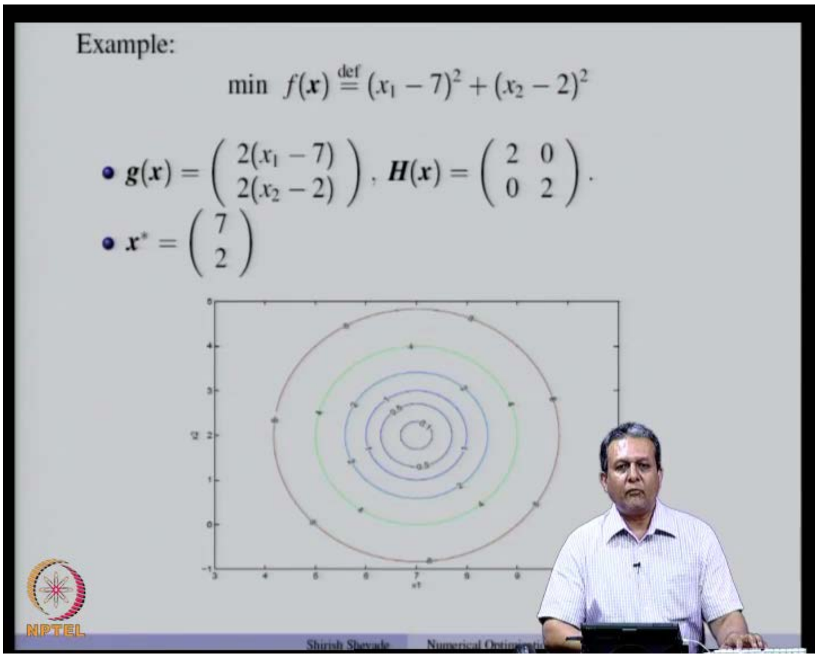


So, this direction is a steepest descent direction where we have d k, so that the direction d k which is equal to minus g k is called the steepest descent directions. So, it is this direction along which the function, there would be a maximum decrease in the objective function with respect to the first order approximation of the function vector given point.

So, an algorithm which uses this steepest descent direction is called steepest descent algorithm. So, the initial part of the optimization algorithm that we saw earlier that remains the same. Now, the first step was to get a descent direction d k and steepest descent direction algorithm uses d k to be minus g k. Now, the other step length determination procedure that is same for all the algorithms. So, we find a positive step length alpha k along direction d k, such that f of x k plus alpha k d k less than f of x k and alpha k satisfies Armijo-Wolfe conditions. So, this will guarantee that there will be a sufficient decrease and the step lengths are not small and the x k plus 1 is said to be x k plus alpha k d k. The iteration counter increase by 1 and the whole procedure is repeated till norm of g k becomes less than or equal to x epsilon and as a output, we get a stationary point x star which is nothing, but x k.

Now, instead of this Armijo-Wolfe condition, one can also use Armijo-Goldstein’s conditions or Armijo’s conditions coupled with backtracking line search or exact line search. So, any of the methods can be used to ensure that there is a sufficient decrease in the objective function and step lengths are not too small. So, one can use either exact line search or backtracking line search in step to be of this algorithm.

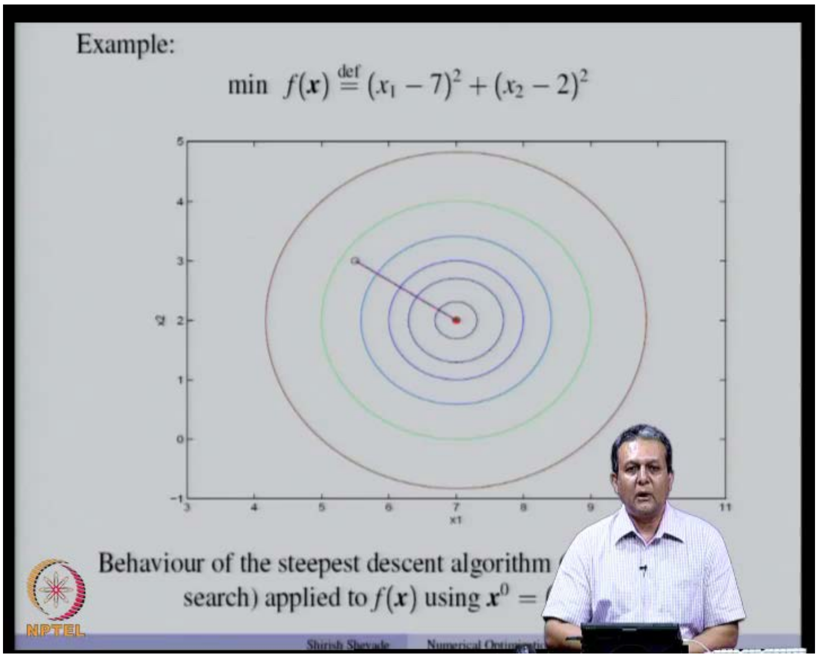
(Refer Slide Time: 52:00)



Now, we will see how this algorithm works on different data sets. So, let us take a simple example. So, the function which we want to optimize is x 1 minus 7 square plus x 2 minus 2 square. Now, this is a function with circular controls. Now, if you write the gradient of the function and since, it is a quadratic function, the (( )) is independent of the x 1 and x 2. Now, if we set the gradients to 0, what we get is x 1 equal to 7 and x 2 equal to 2 and that is and the (( )) is positive definite. So, 7 and 2 is a local minimum of this problem.

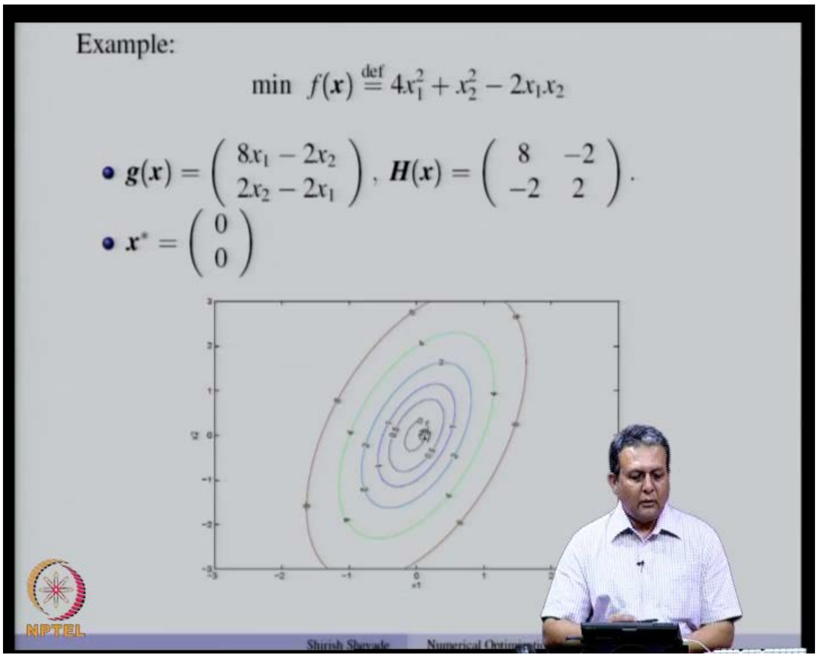
Now, let us see how the controls of this function looks like. So, the controls are showed here. So, you will see that this is the x 1 axis and this is the x 2 axis and these are the circular controls. So, the function value here is 8, then in the function value here is 4, then 2, 1.5, 0.1 and at 7 2 7 come over in the x quantities 7 and while x 2 x 1 quantities 7 and x 2 quantities 2. This is the minimum of this function. Now, suppose given to apply steepest descent algorithm which mentioned earlier to solve this problem iteratively.

(Refer Slide Time: 53:34)



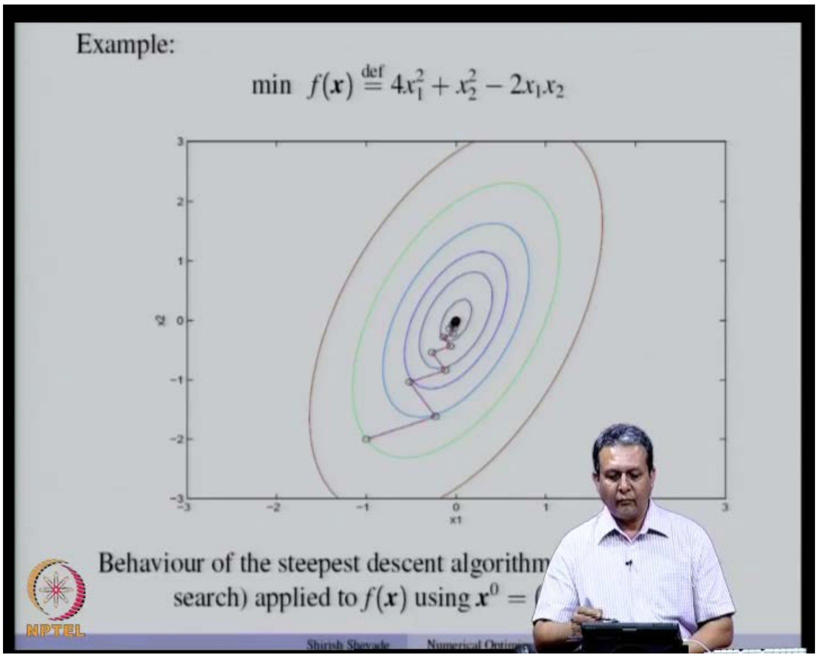
Now, let us start with some initial point. So, we started with some initial point and in one step, the steepest descent algorithm with exact line search 2 cos to the solution. So, since this was the quadratic function because it is very easy to use the exact line search. So, we have to use exact line search here and demonstrate that for the function which circular controls. If we start from this point, we go to the solution in exactly one step. So, the initial point is 9 come of 4 which is here, where we are lucky to get this initial point, so that the solution was in exactly one iteration. Let us see so let us along this take this same problem, but we start with different initial point. So, assume that we start with this point, then even the steepest descent method with exact line search, reach the solution in one step. So, for circular quadratic controls the steepest descent method with exact line search would take us to the solution in exactly one step. Now, what happens when the controls are quadratic, but not circular, but elastic.

(Refer Slide Time: 55:02).



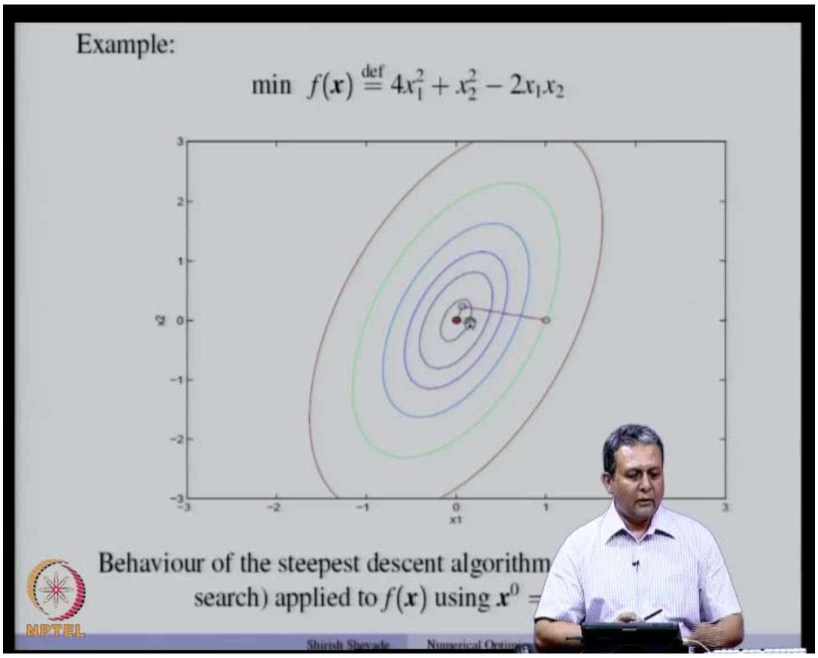
So, let us consider a problem here 1 to minimize 4 x 1 square plus x 2 square minus 2 x 1 x 2. The gradient is given here and (( )) given here. So, clearly 0 is the solution of this problem. So, control of this function are shown here. So, these are elastical controls and so the value at x. So, this is the functional value for corresponding to this control function value corresponding to this control is 4, then 2 and finally at the origin, we have the minimum function value. Now, let us apply steepest descent method with exact line search to this problem.

(Refer Slide Time: 56:00)



So, if we start from the point minus 1 and minus 2, you see that there is a lot of zigzagging kind of directions that you get before one converges to the solution. In fact, in this case, the number of iterations required were about 26.

(Refer Slide Time: 56:27)



Now, further same function if we start from a different point, so if we start from a 0.10, it required about four steps or four iterations to converge to the minimum. So, a lot difference on the initial point in this case. In this case, very few iterations were required. While in the previous case, lot of iterations are required before the method would converge to the minimum. And if you look at the previous case, the circular controls were there, the convergence to place in exactly one iteration irrespective of the initial point. So, why there is such a big difference in the number of iterations for quadratic control? So, we will study those things with respect to steepest descent method in the next class.

Thank you.