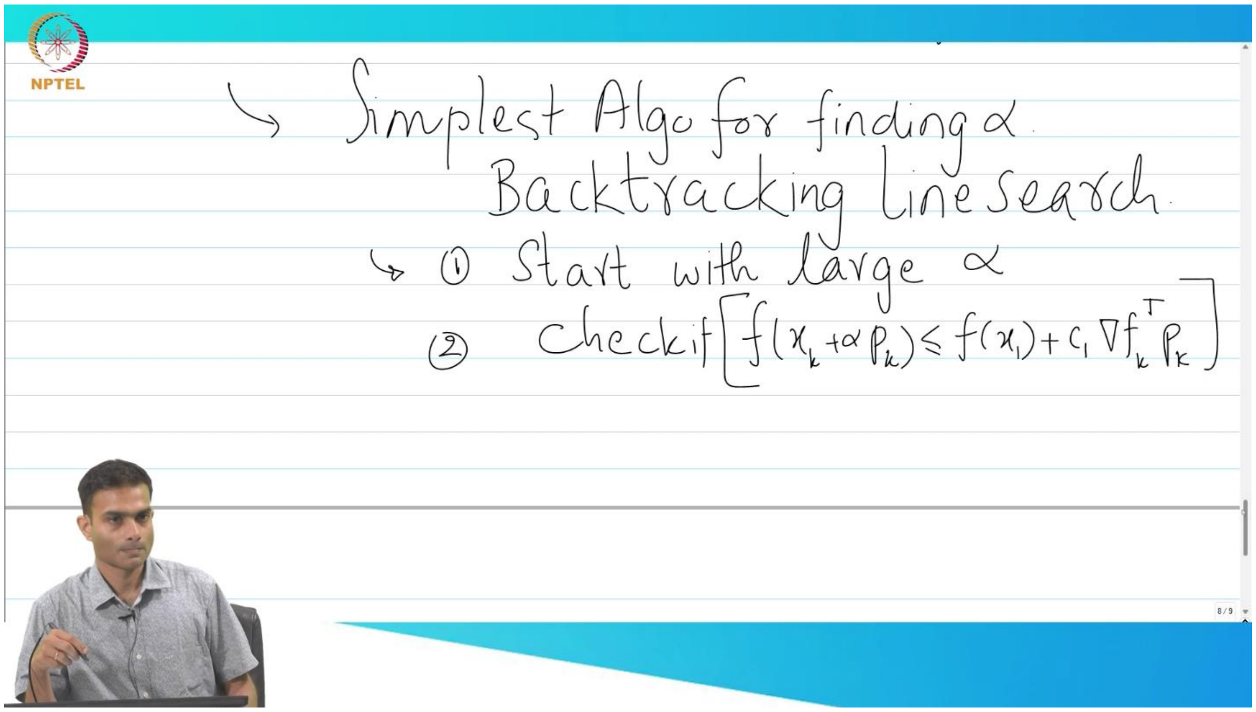
Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 04
Lecture - 26

Backtracking Line Search

We now have our acceptable range for α, but the question remains: how do we actually choose α? Until now, we've only discussed tests for determining whether a given α is good enough, such as the sufficient decrease condition, Wolfe conditions, and the strong Wolfe conditions. But now, let's discuss the simplest algorithm for actually finding α. This algorithm is called the **Backtracking Line Search**.



↳ Simplest Algo for finding α.
Backtracking Line Search.
↳ ① Start with large α
② Check if $\left[f(x_k + \alpha p_k) \leq f(x_1) + c_1 \nabla f_k^T p_k \right]$

1. Starting Point for Alpha

When picking α, should you start with a very small or a very large value of α? The answer is a large α, especially if you're ambitious or impatient. Starting with a large α might get you lucky sooner. So, begin with a large α and check whether the chosen condition is satisfied, for example, the sufficient decrease condition:

$$f(x_k + \alpha p_k) < c_1 \cdot \text{linear approximation.}$$

If this condition holds, accept α. If it doesn't, reduce α using the following rule:

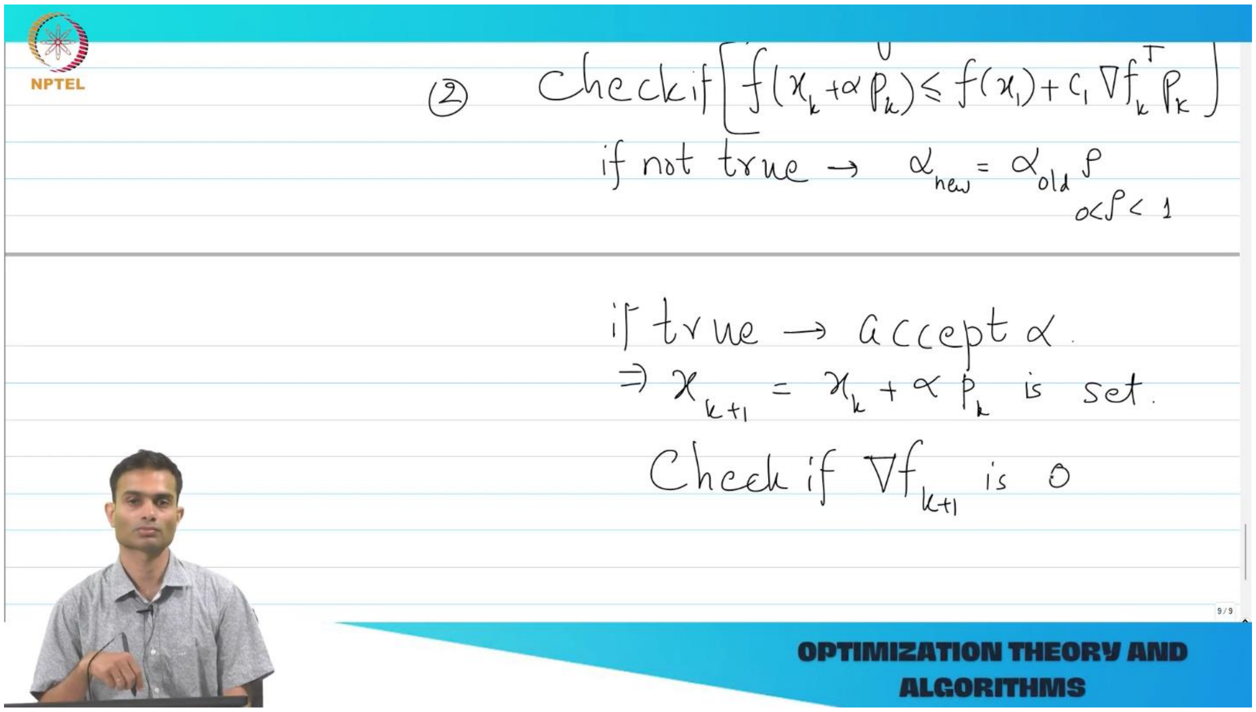
$$\alpha_{\text{new}} = \alpha_{\text{old}} \cdot \rho,$$

where $\rho \in (0,1)$ is a reduction factor.

2. Why Backtracking?

The algorithm gets its name from the fact that we start with a large α and keep backtracking, i.e., reducing α, until the condition is satisfied. In this example, we used the sufficient decrease condition because it is cheaper to compute than the curvature condition (which involves evaluating gradients). However, if evaluating the curvature condition is easier in a particular case, you could choose that instead.

3. Algorithm Summary



② Check if $\left[f(x_k + \alpha p_k) \leq f(x_1) + c_1 \nabla f_k^T p_k \right]$

if not true $\rightarrow$ $\alpha_{new} = \alpha_{old}\, \rho$ $\quad 0 < \rho < 1$

if true $\rightarrow$ accept α.

$\Rightarrow x_{k+1} = x_k + \alpha p_k$ is set.

Check if ∇f_{k+1} is 0

The Backtracking Line Search is one of the most common techniques for determining α. The basic steps are:

- Start with a large value of α.
- Backtrack by reducing α until the chosen condition (e.g., sufficient decrease) is satisfied.
- Update the point:

$$x_{k+1} = x_k + \alpha p_k.$$

- Repeat this process until $\nabla f(x_{k+1})$ is close to zero, indicating convergence.

4. Outer and Inner Loops

There are two loops in this method:

- The **inner loop** checks for a good value of α by backtracking.
- The **outer loop** checks for convergence by evaluating the gradient $\nabla f(x_{k+1})$.

If the gradient at x_{k+1} is small enough, we have reached the solution.

5. Practical Considerations

The algorithm has a heuristic flavor. When we say "large" α, we don't mean arbitrarily large values like 10^6, because those wouldn't be useful. A common heuristic is to normalize the direction p_k to unit length and start with $\alpha = 1$. The reduction factor ρ is typically chosen as something like 0.8, meaning that α is reduced in a geometric progression:

$$\alpha_{\text{new}} = 0.8 \cdot \alpha_{\text{old}}.$$

In about five steps, α can decrease to 10^{-5}, at which point it might be accepted.

6. Adaptive Step Sizes

As you approach the solution, smaller values of α are more likely to work. Therefore, in practice, α may vary from iteration to iteration. We can denote it as α_k to reflect this:

$$\alpha_k \text{ may change in each iteration.}$$

7. Summary of Line Search Algorithms

We've discussed the need for inexact line search because exact line search is computationally expensive. The key conditions we use are:

- **Sufficient Decrease Condition**: Ensures the function value decreases enough, compared to a linear approximation.
- **Curvature Condition**: Ensures the slope is sufficiently small.

To prevent overshooting, we introduced the modulus of the slope, which leads to the strong Wolfe condition. Backtracking Line Search is one of the simplest algorithms for determining α, relying on only first- and second-order Taylor approximations.

8. Challenges in Practical Implementation

Although the algorithm appears simple, practical implementation involves some heuristics, such as choosing the starting value for α and tuning ρ. These parameters affect the rate of convergence, but too large a step may cause you to miss the solution, while too small a step slows down convergence. Furthermore, different iterations may require different values of α_k, especially as you approach the solution, where smaller step sizes are more effective.

Finally, while the algorithm is easy to understand conceptually, it becomes more complex when implemented in high-dimensional spaces, where exact solutions are infeasible and line search algorithms must rely on function and gradient evaluations to navigate towards the optimal solution.