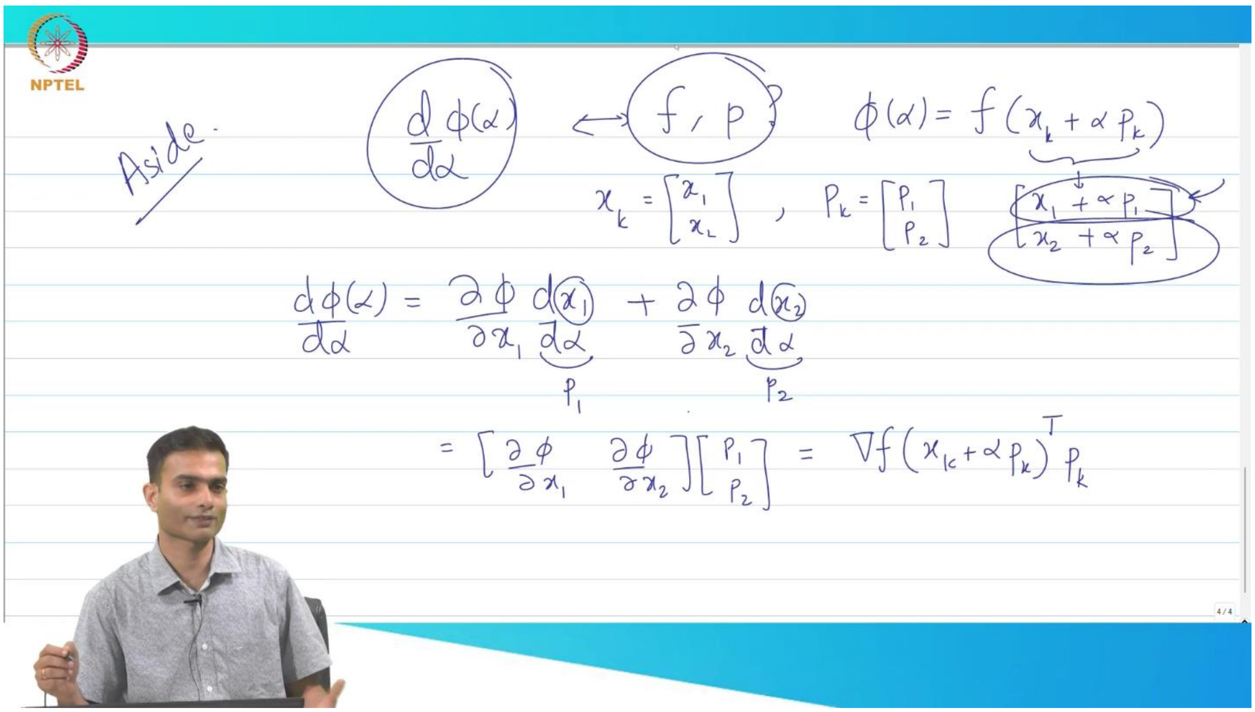
Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 04
Lecture - 24

Wolfe Conditions

Let us take an aside before diving in. Now I want to calculate a quantity that will be needed frequently. How do I write this in terms of f and p? What is the expression for this? Recall the definition of $\phi(\alpha)$: it is simply $f(x_k + \alpha p_k)$. So, which of the theorems of calculus can be used here? We can use the chain rule. Let's derive it explicitly in two dimensions to get a good grasp of it.



Consider x_k as a two-dimensional vector: let $x_k = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Similarly, p_k is also two-dimensional: let $p_k = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$. Therefore, $\phi(\alpha)$ becomes a two-dimensional vector:

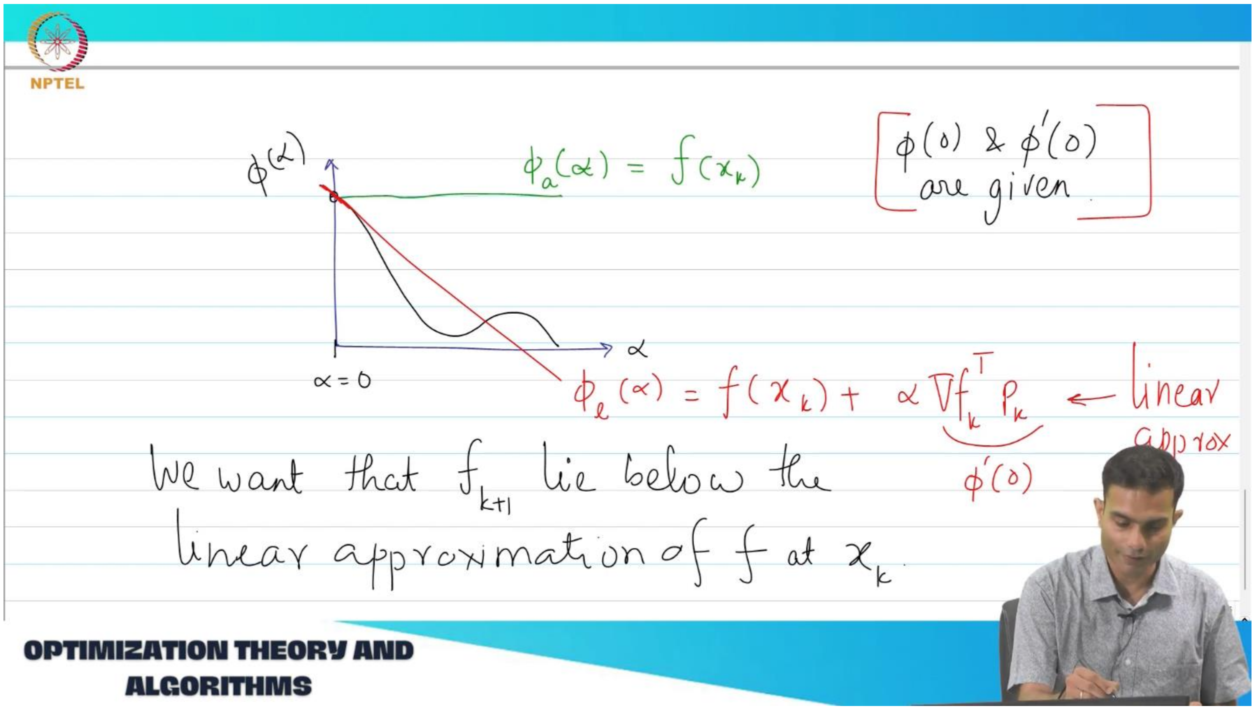
$$x_1 + \alpha p_1, \quad x_2 + \alpha p_2.$$

Now, applying the chain rule to $\frac{d\phi(\alpha)}{d\alpha}$, we get two partial derivatives:

$$\frac{d\phi}{dx_1}\frac{dx_1}{d\alpha} + \frac{d\phi}{dx_2}\frac{dx_2}{d\alpha}.$$

What is $\frac{dx_1}{d\alpha}$? It's simply p_1. Similarly, $\frac{dx_2}{d\alpha} = p_2$. Therefore, the result becomes:

$$p_1 + p_2.$$

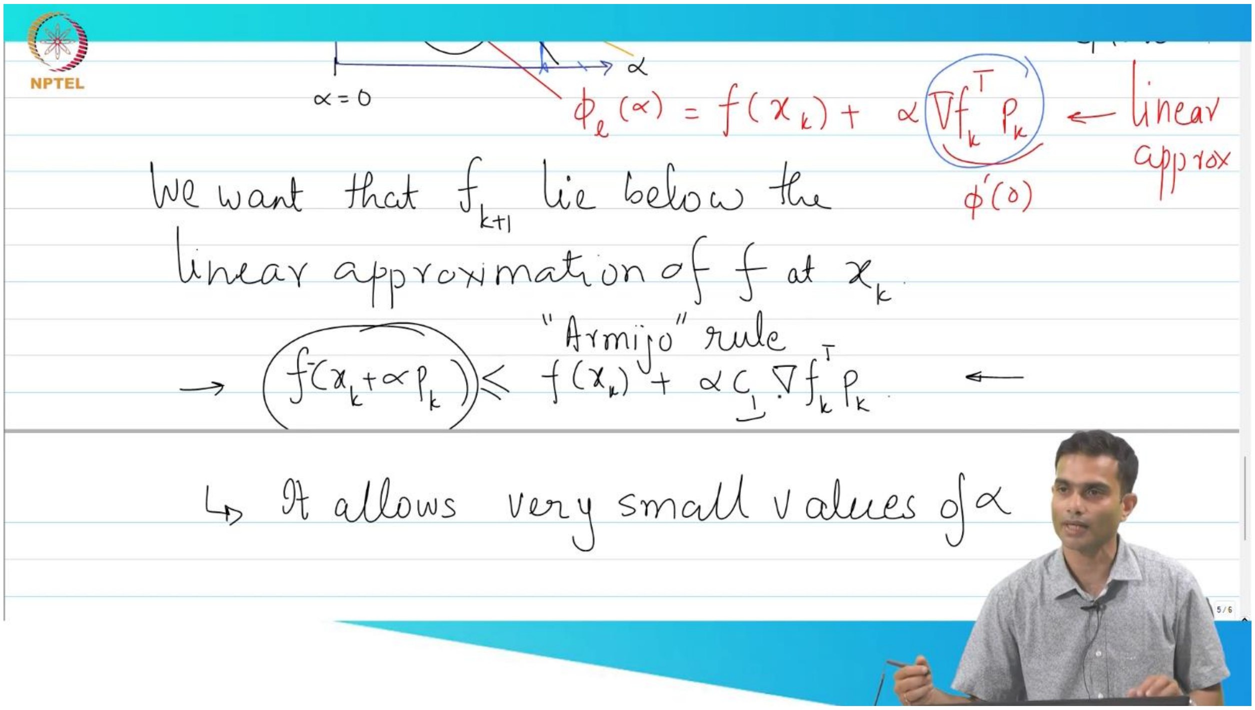


In compact notation, what is the partial derivative of ϕ with respect to x_1 and x_2? It's the gradient of f. So, we have:

$$\frac{d\phi(\alpha)}{d\alpha} = \nabla f(x_k + \alpha p_k)^\top p_k.$$

This small result gives us the relationship between $\frac{d\phi(\alpha)}{d\alpha}$, f, and p, and will be useful later.

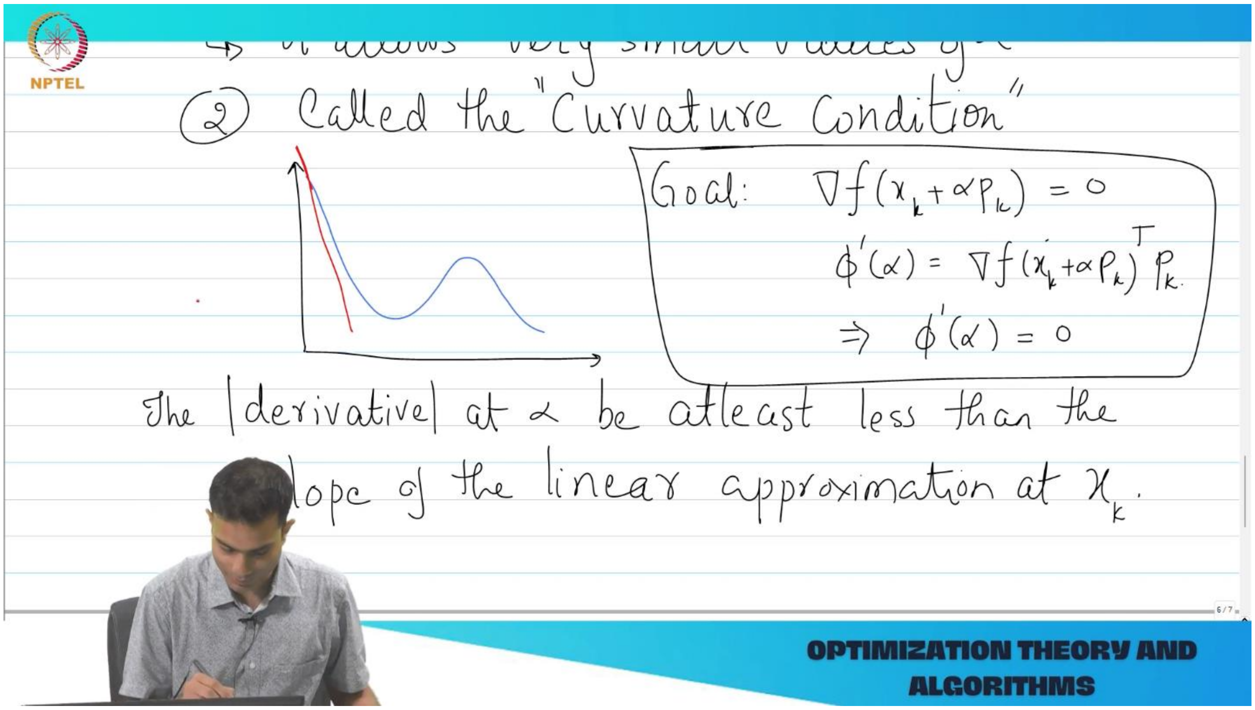
Sufficient Decrease Condition



Now let's discuss the first of the Wolfe conditions, called the condition of *sufficient decrease*. To understand this, imagine we are given two pieces of information at $\alpha = 0$: the function value $\phi(0)$ and the derivative $\phi'(0)$. These are the minimum information needed to proceed. The descent direction is also given (which could be the gradient descent direction, conjugate gradient direction, or Newton method direction).

If we only know $\phi(0)$, the simplest model we can construct is a constant function, i.e., $\phi(\alpha) = f(x_k)$, which is just a zeroth-order approximation. The next step is to construct a more sophisticated model using the derivative information. This leads to a linear approximation, based on the first-order Taylor expansion:

$$\phi(\alpha) = f(x_k) + \alpha \nabla f_k^{\top} p_k.$$



If we set $\alpha = 0$, the expression simplifies to:

$$\phi'(0) = \nabla f_k^\top p_k.$$

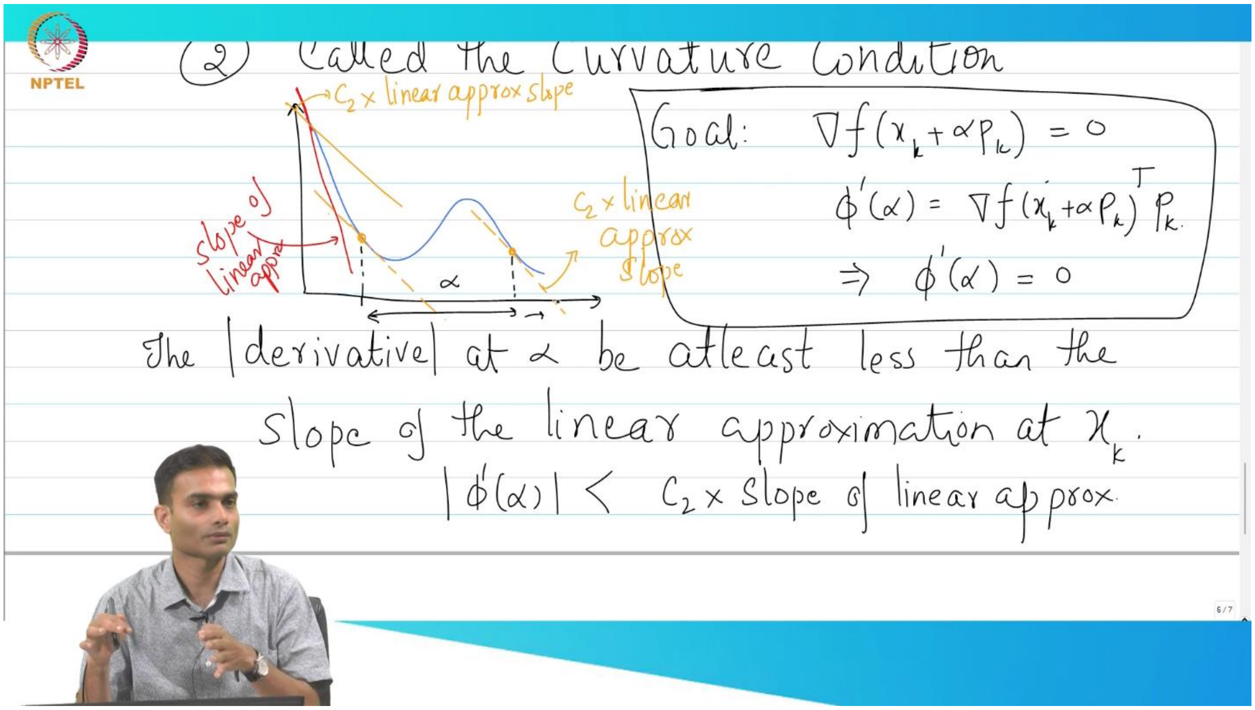
The idea of sufficient decrease is that the function value at the new point, x_{k+1}, should lie below this linear approximation. In other words, the function value at x_{k+1} must decrease more than this linear approximation predicts.

However, in practice, this condition tends to be too strict, so it is relaxed by introducing a small factor $c_1 \in (0,1)$. This gives us the condition:

$$f(x_k + \alpha p_k) \leq f(x_k) + \alpha c_1 \nabla f_k^\top p_k,$$

where c_1 is typically a small number, often around 10^{-4}. This is known as the *Armijo rule*, which relaxes the linear approximation by allowing more step lengths.

Curvature Condition



The first Wolfe condition focuses on function values, while the second, called the *curvature condition*, looks at the derivative values. The goal is to ensure that at the new point x_{k+1}, the gradient should ideally be zero. Mathematically, this means:

$$\phi'(\alpha) = \nabla f(x_k + \alpha p_k)^\top p_k = 0.$$

Since exact line search is often impractical, we relax this by saying that the derivative at α should at least be smaller than the derivative at the current point, but not too small. We introduce another factor, $c_2 \in (0,1)$, to allow this relaxation:

$$\nabla f(x_k + \alpha p_k)^\top p_k \geq c_2 \nabla f_k^\top p_k.$$

This condition ensures that the step size α is not too small, which would lead to very slow progress. By balancing the two Wolfe conditions, we ensure that the step size is neither too small nor too large, allowing the algorithm to progress efficiently.