

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 1)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Introduction  
Taylor's Theorem  
Optimality conditions  
Strategies of optimization  
Step length conditions  
Convergence



Санкт-Петербургский  
государственный  
университет



36 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

This is the first lecture of the first part of the our course, which is called "**Optimization: classical approaches**". We will look at the basic concepts and results, as well as the key principles of approaches to solving unconstrained optimization problems.

## Standard Formulation

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{or} \quad f(x) \rightarrow \min_{x \in \mathbb{R}^n}$$

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$  — smooth (continuously differentiable) scalar function.
- No constraints on the variable  $x$ : the feasible set is  $\mathbb{R}^n$ .
- Such problems arise in physics (energy minimization), economics (utility optimization), machine learning (loss minimization), etc.

## Objective

Find a point  $x^*$  such that  $f(x^*) \leq f(x)$  for all  $x$  in  $\mathbb{R}^n$ .



## Comments

Let's begin with the most basic formulation of an optimization problem.

On the slide, you see the standard mathematical model for an unconstrained optimization problem. We are looking to minimize a scalar function of several variables, which we denote as  $f(x)$ , where  $x$  is a vector in  $\mathbb{R}^n$ .

The key point here is that there are no constraints on the variable  $x$ . That means we are free to search for the minimum of the function over the entire real space, without any additional conditions or restrictions.

The function  $f$  is assumed to be continuously differentiable — in other words, it has derivatives that are continuous — because these derivatives will be crucial when we discuss optimality conditions and develop numerical algorithms.

Problems of this type appear frequently in applications. For example, in physics, we often want to minimize the potential energy of a system; in economics, we may want to maximize utility, which we can convert to a minimization problem; and in machine learning, we typically minimize a loss function.

Our goal is to find a point — let's call it  $x^*$  — such that  $f(x^*) \leq f(x)$  for all  $x$  in  $\mathbb{R}^n$ .



## Global Minimum

A point  $x^* \in \mathbb{R}^n$  is a global minimum of  $f$  if

$$f(x^*) \leq f(x) \quad \text{for all } x \in \mathbb{R}^n$$

## Local Minimum

A point  $x^* \in \mathbb{R}^n$  is a local minimum of  $f$  if there exists  $\varepsilon > 0$  such that

$$f(x^*) \leq f(x) \quad \text{for all } x \text{ with } \|x - x^*\| < \varepsilon$$

- Every global minimum is local, but not vice versa.
- Global minimization is significantly more challenging in general.

## Comments

For the sake of completeness we will give a rigorous mathematical definition of global and local minimums and begin discussing the problems associated with finding them.

A point  $x^*$  is called a global minimum of the function  $f$  if  $f(x^*) \leq f(x)$  for all  $x \in \mathbb{R}^n$ . By contrast, a local minimum requires that this condition holds only within some neighborhood of  $x^*$ , i.e. for all  $x$  with  $\|x - x^*\| < \varepsilon$ .

Every global minimum is also a local minimum, but not every local minimum is global — especially in the case of non-convex functions. In the general case, the task of finding the global extremum turns out to be significantly more difficult than the task of finding the local one. Global optimization methods, as a rule, require additional assumptions or the presence of structural properties of the function under study, such as convexity, for example.



## Theorem 1 (Taylor's Theorem)

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be continuously differentiable and  $p \in \mathbb{R}^n$ . Then:

$$f(x + p) = f(x) + \nabla f(x + tp)^T p \quad \text{for some } t \in (0, 1).$$

If  $f$  is twice continuously differentiable, then:

$$\nabla f(x + tp) = \nabla f(x) + \int_0^1 \nabla^2 f(x + tp)p dt$$

and then

$$f(x + p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp) p \quad \text{for some } t \in (0, 1).$$

- $\nabla f$  is the gradient;  $\nabla^2 f$  is the Hessian matrix.
- The remainder is expressed via evaluation at an intermediate point.

## Comments

Let us now recall Taylor's theorem in a form convenient for analyzing optimality conditions. Let the function  $f$  be continuously differentiable, and let  $p$  be an arbitrary vector in  $\mathbb{R}^n$ .

Then, for some  $t \in (0, 1)$ , we can write:  $f(x + p) = f(x) + \nabla f(x + tp)^T p$ .

If the function  $f$  is twice continuously differentiable, this expansion can be extended by adding a second-order term. Specifically, for some  $t \in (0, 1)$ :  $f(x + p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp) p$ .

This is a form of Taylor's theorem with remainder in Lagrange form. It is this form that we will use in several upcoming proofs — in particular when establishing first and second-order conditions for optimality. The proof of this theorem can be found in any good textbook on mathematical analysis.



## Theorem 2 (Necessary condition for a local minimum)

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be continuously differentiable in an open neighborhood of  $x^*$ . If  $x^*$  is a local minimum of  $f$ , then

$$\nabla f(x^*) = 0$$

- $\nabla f(x^*)$  is the gradient (vector of partial derivatives) at  $x^*$ .
- This condition is necessary, but not sufficient.
- All candidate minimizers must satisfy this condition.

## Comments

We now proceed to one of the key analytical tools in optimization — the first-order necessary condition for a local minimum.

Suppose the function  $f$  is continuously differentiable. Then, if a point  $x^*$  is a local minimum, the gradient of  $f$  at that point must be equal to zero:  $\nabla f(x^*) = 0$ . That is, all partial derivatives of  $f$  vanish at  $x^*$ .

Intuitively, this means the following: when we expand the function in a neighborhood of  $x^*$ , using a first-order approximation, there is no direction in which the value of the function decreases. More precisely, the directional derivative of  $f$  at  $x^*$  in any direction is zero.

This condition is called necessary because any local minimum must satisfy it. However, it is not sufficient: there may be points where the gradient is zero, but which are not minima — for example, maxima or saddle points. In practice, this condition allows us to identify candidate points for minima, which are then subject to further analysis — for instance, using second-order conditions.

## Proof of Theorem 2 (First-Order Necessary Condition)

**Proof:** Assume  $\nabla f(x^*) \neq 0$  and define  $p = -\nabla f(x^*)$ . Then

$$p^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$$

Since  $\nabla f$  is continuous, there exists  $T > 0$  such that

$$p^T \nabla f(x^* + tp) < 0 \quad \text{for all } t \in [0, T]$$

Now fix  $\bar{t} \in (0, T]$ . By Taylor's theorem, for some  $t \in (0, \bar{t})$ ,

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t} \cdot p^T \nabla f(x^* + tp)$$

Hence  $f(x^* + \bar{t}p) < f(x^*)$ , which contradicts the minimality of  $x^*$ . Therefore,  $\nabla f(x^*) = 0$ .  $\square$

### Stationary point

We call  $x^*$  a stationary point if  $\nabla f(x^*) = 0$

5/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Introduction

Taylor's Theorem

Optimality conditions

Strategies of optimization

Step length conditions

Convergence



### Comments

Suppose the function  $f$  is continuously differentiable in an open neighborhood of a point  $x^*$ , and that  $x^*$  is a local minimizer.

We argue by contradiction. Assume that  $\nabla f(x^*) \neq 0$ . Then we define the vector  $p = -\nabla f(x^*)$ . The scalar product is then  $p^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$ .

Since the gradient is continuous, we can choose a small enough  $T > 0$  such that for all  $t \in [0, T]$ , the scalar product  $p^T \nabla f(x^* + tp) < 0$ .

Now, take any  $\bar{t} \in (0, T]$ . Applying Taylor's theorem we can write  $f(x^* + \bar{t}p) = f(x^*) + \bar{t} \cdot p^T \nabla f(x^* + tp)$  for some  $t \in (0, \bar{t})$ .

Since  $p^T \nabla f(x^* + tp)$  is negative, we get  $f(x^* + \bar{t}p) < f(x^*)$  — contradicting the assumption that  $x^*$  is a local minimum. Hence, the assumption was false, and the gradient at  $x^*$  must be zero:  $\nabla f(x^*) = 0$ .

We call  $x^*$  a stationary point if  $\nabla f(x^*) = 0$ .



### Definitions

A symmetric matrix  $A \in \mathbb{R}^{n \times n}$  is:

- positive semidefinite if

$$p^T A p \geq 0 \quad \text{for all } p \in \mathbb{R}^n$$

- positive definite if

$$p^T A p > 0 \quad \text{for all } p \neq 0$$

### Theorem 3 (Second-Order Necessary Condition)

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be twice continuously differentiable in an open neighborhood of  $x^*$ . If  $x^*$  is a local minimum of  $f$ , then

$$\nabla f(x^*) = 0 \quad \text{and} \quad \nabla^2 f(x^*) \text{ is positive semidefinite.}$$

### Comments

Before stating the theorem, let us recall the standard definitions of positive definite and positive semidefinite matrices. A symmetric matrix  $A$  is called positive semidefinite if  $p^T A p \geq 0$  for all  $p \in \mathbb{R}^n$ . The matrix is positive definite if  $p^T A p > 0$  for all  $p \neq 0$ .

Now we can formulate the second-order necessary condition for a local minimum. Suppose that the function  $f$  is twice continuously differentiable in an open neighborhood of a point  $x^*$ , and that  $x^*$  is a local minimum. Then two conditions must hold:  $\nabla f(x^*) = 0$  and the Hessian matrix  $\nabla^2 f(x^*)$  is positive semidefinite.

These two conditions — vanishing of the gradient and nonnegativity of the second-order term — together form the second-order necessary condition.

## Proof of Theorem 3 (Second-Order Necessary Condition)

Proof. Let  $x^*$  be a local minimum, and let  $f$  be twice continuously differentiable in an open neighborhood of  $x^*$ . Then  $\nabla f(x^*) = 0$  (by the first-order condition). Fix any  $p \in \mathbb{R}^n$  and define  $\varphi(\alpha) = f(x^* + \alpha p)$ . From Taylor's theorem:

$$\varphi(\alpha) = f(x^*) + \alpha \nabla f(x^*)^T p + \frac{\alpha^2}{2} p^T \nabla^2 f(x^* + \theta \alpha p) p$$

for some  $\theta \in (0, 1)$ . Since  $\nabla f(x^*) = 0$  and  $x^*$  is a minimum, we have

$$\varphi(\alpha) \geq \varphi(0) \quad \text{for small } \alpha > 0$$

This implies:

$$p^T \nabla^2 f(x^* + \theta \alpha p) p \geq 0$$

Taking the limit as  $\alpha \rightarrow 0$ , continuity of  $\nabla^2 f$  gives:

$$p^T \nabla^2 f(x^*) p \geq 0 \quad \text{for all } p \in \mathbb{R}^n$$

i.e. the Hessian at  $x^*$  ( $\nabla^2 f(x^*)$ ) is positive semidefinite. □

7/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Introduction**

**Taylor's Theorem**

**Optimality conditions**

**Strategies of optimization**

**Step length conditions**

**Convergence**



### Comments

Let us now prove the second-order necessary condition. Suppose that  $x^*$  is a local minimum, and that the function  $f$  is twice continuously differentiable in an open neighborhood of  $x^*$ .

From the first-order condition, we already know that  $\nabla f(x^*) = 0$ . Now fix an arbitrary direction  $p$ , and define an auxiliary function  $\varphi(\alpha) = f(x^* + \alpha p)$ .

From Taylor's theorem we have  $\varphi(\alpha) = f(x^*) + \alpha \nabla f(x^*)^T p + \frac{\alpha^2}{2} p^T \nabla^2 f(x^* + \theta \alpha p) p$ .

Since  $\nabla f(x^*) = 0$  and  $x^*$  is a minimum, we must have  $\varphi(\alpha) \geq \varphi(0)$  for small positive  $\alpha$ . This implies that the quadratic term  $p^T \nabla^2 f(x^* + \theta \alpha p) p \geq 0$ . Passing to the limit as  $\alpha \rightarrow 0$  and using the continuity of the Hessian, we conclude that  $p^T \nabla^2 f(x^*) p \geq 0$  for all  $p \in \mathbb{R}^n$ .

This proves that the Hessian  $\nabla^2 f(x^*)$  is positive semidefinite at the local minimum.



## Theorem 4 (Second-Order Sufficient Condition)

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be twice continuously differentiable in an open neighborhood of  $x^*$ . Suppose  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*)$  is positive definite. Then  $x^*$  is a strict local minimum of  $f$ .

- ▶ Positive definiteness of the Hessian strengthens the second-order condition.
- ▶ A strict minimum means:  $f(x) > f(x^*)$  for all  $x$  sufficiently close to  $x^*$  with  $x \neq x^*$ .

## Comments

We now formulate the second-order sufficient condition for a strict local minimum. Let the function  $f$  be twice continuously differentiable in an open neighborhood of a point  $x^*$ .

Suppose that two conditions hold:  $\nabla f(x^*) = 0$  and the Hessian matrix  $\nabla^2 f(x^*)$  is positive definite. Then  $x^*$  is a strict local minimum of the function  $f$ .

In other words,  $f(x) > f(x^*)$  for all points  $x$  sufficiently close to  $x^*$  (but not equal to it). This result strengthens the necessary condition from the previous slide. Positive definiteness of the Hessian guarantees that the second-order term in the Taylor expansion is strictly positive in all nonzero directions.

<b>Introduction</b>
<b>Taylor's Theorem</b>
<b>Optimality conditions</b>
<b>Strategies of optimization</b>
<b>Step length conditions</b>
<b>Convergence</b>



Proof. Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be twice continuously differentiable in an open neighborhood of  $x^*$ . Assume  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*)$  is positive definite. Fix any  $p \in \mathbb{R}^n$  and consider  $\varphi(\alpha) = f(x^* + \alpha p)$ . By Taylor's theorem:

$$f(x^* + \alpha p) = f(x^*) + \frac{\alpha^2}{2} p^T \nabla^2 f(x^* + \theta \alpha p) p$$

for some  $\theta \in (0, 1)$ . Since  $\nabla^2 f$  is continuous and positive definite at  $x^*$ , we have:

$$p^T \nabla^2 f(x^* + \theta \alpha p) p > 0 \quad \text{for all small } \alpha > 0$$

Hence,

$$f(x^* + \alpha p) > f(x^*) \quad \text{for all small } \alpha > 0$$

Thus  $x^*$  is a strict local minimum.  $\square$

### Comments

Let us now prove the second-order sufficient condition for a strict local minimum. Suppose that the function  $f$  is twice continuously differentiable in an open neighborhood of a point  $x^*$ , and that  $\nabla f(x^*) = 0$ . Suppose also that the Hessian matrix  $\nabla^2 f(x^*)$  is positive definite.

Fix any unit vector  $p$ , and define the function  $\varphi(\alpha) = f(x^* + \alpha p)$ . By Taylor's theorem, we have the expansion:  $f(x^* + \alpha p) = f(x^*) + \frac{\alpha^2}{2} p^T \nabla^2 f(x^* + \theta \alpha p) p$ .

Since the Hessian is continuous and positive definite at  $x^*$ , the quadratic form  $p^T \nabla^2 f(x^* + \theta \alpha p) p > 0$  for all small positive  $\alpha$ . Therefore,  $f(x^* + \alpha p) > f(x^*)$  for all small positive  $\alpha$ . This proves that  $x^*$  is a strict local minimum.



## Theorem 5 (Global Minimum for Convex Functions)

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex. Then any local minimizer  $x^*$  is also a global minimizer. If in addition  $f$  is differentiable, then any stationary point  $x^*$  with

$$\nabla f(x^*) = 0$$

is a global minimizer.

Proof. Suppose  $x^*$  is a local but not global minimizer. Then there exists  $z \in \mathbb{R}^n$  such that

$$f(z) < f(x^*)$$

Define  $x = \lambda z + (1 - \lambda)x^*$  for  $\lambda \in (0, 1]$ .

By convexity of  $f$ :

$$f(x) \leq \lambda f(z) + (1 - \lambda)f(x^*) < f(x^*)$$

But every neighborhood of  $x^*$  contains such  $x$ , contradicting the local minimality of  $x^*$ .

10/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Let us now state an important result connecting convexity and global optimality. Suppose the function  $f$  is convex. Then any local minimizer of  $f$  is automatically a global minimizer.

Moreover, if  $f$  is differentiable, then any stationary point — that is, any point where  $\nabla f(x^*) = 0$  — is a global minimizer. This result shows how convexity allows us to go beyond local analysis. The proof is by contradiction. Suppose that  $x^*$  is a local minimum but not a global one. Then there exists a point  $z$  such that  $f(z) < f(x^*)$ .

Now consider the convex combination  $x = \lambda z + (1 - \lambda)x^*$ , for some  $\lambda \in (0, 1]$ . By convexity of the function  $f$ , we have  $f(x) \leq \lambda f(z) + (1 - \lambda)f(x^*) < f(x^*)$ . But such a point  $x$  lies arbitrarily close to  $x^*$  — contradicting the assumption that  $x^*$  is a local minimum.

Therefore,  $x^*$  must be a global minimizer.



Proof. For the second part of the theorem, suppose that  $x^*$  is not a global minimizer and choose  $z$  as above. From convexity,

$$\begin{aligned}\nabla f(x^*)^T(z - x^*) &= \frac{d}{d\lambda} f(x^* + \lambda(z - x^*)) \Big|_{\lambda=0} = \lim_{\lambda \downarrow 0} \frac{f(x^* + \lambda(z - x^*)) - f(x^*)}{\lambda} \\ &\leq \lim_{\lambda \downarrow 0} \frac{\lambda f(z) + (1 - \lambda)f(x^*) - f(x^*)}{\lambda} = f(z) - f(x^*) < 0\end{aligned}$$

Hence,  $\nabla f(x^*) \neq 0$ , and so  $x^*$  is not a stationary point.  $\square$

## Comments

Let us now prove the second part of the theorem: that is, we want to show that if the function  $f$  is convex and differentiable, then any stationary point is a global minimizer.

We proceed by contradiction. Suppose that  $x^*$  is not a global minimizer. Consider the point  $z$  introduced in the first part of the proof, such that  $f(z) < f(x^*)$ .

Now take the directional derivative of  $f$  at  $x^*$  in the direction of  $(z - x^*)$ . Since  $f$  is convex and differentiable, this directional derivative,  $\nabla f(x^*)^T(z - x^*) \leq f(z) - f(x^*)$ .

And since  $f(z) - f(x^*) < 0$ , the inner product  $\nabla f(x^*)^T(z - x^*)$  is strictly less than zero. Therefore, the gradient  $\nabla f(x^*) \neq 0$ .

This contradicts the assumption that  $x^*$  is a stationary point, and the statement is proved. This result has a fundamental implication for optimization methods. In the convex and differentiable case, we do not need to distinguish between local and global minimizers — any point where the gradient vanishes is guaranteed to be a global minimum. This is why many algorithms for unconstrained optimization are built around the search for stationary points: under the right structural assumptions, such as convexity, they correspond to global solutions.



- ▶ All unconstrained optimization algorithms generate a sequence  $\{x_k\}$  starting from an initial guess  $x_0$  (or some set of initial points).
- ▶ At each iteration, they attempt to reduce the objective function  $f(x)$  by using information at  $x_k$  and possibly also information from earlier iterates  $x_0, \dots, x_{k-1}$ .
- ▶ Most classical algorithms follow one of two main strategies:
  - ▶ Line Search: Choose a direction  $p_k$  and a step length  $\alpha_k$  to move along  $p_k$ .
  - ▶ Trust Region: Build a local model of  $f(x)$  near  $x_k$  and minimize it within a restricted region.
- ▶ The search direction  $p_k$  may depend only on the current point or on previous iterates as well.

## Comments

In the future, we will consider various algorithmic approaches to finding the minimum (or maximum) of a function. Let us briefly summarize how unconstrained optimization algorithms work in general. Every algorithm starts from an initial point (or a population of initial points), usually called  $x_0$ , and generates a sequence of iterates  $x_k$ . At each step, the algorithm uses information about the function at the current point — such as the value of  $f$  and possibly the gradient or Hessian — to decide where to move next. Most classical algorithms are based on one of two general ideas. The first is line search: we pick a direction  $p_k$  and move in that direction by choosing a step length  $\alpha_k$ . The second is the trust-region approach: here, we build a local approximation of the function near the current point, and try to minimize this model inside a region — typically a ball or an ellipsoid — centered at  $x_k$ . The way we choose the direction  $p_k$  can depend either only on the current point, or also on earlier iterates. We'll look at examples of both in upcoming lectures.

Introduction

Taylor's Theorem

Optimality conditions

Strategies of optimization

Step length conditions

Convergence



## Line Search Methods

- ▶ Choose a search direction  $p_k$ .
- ▶ Select a step length  $\alpha_k > 0$  to reduce  $f(x)$  along  $p_k$ :

$$\min_{\alpha > 0} f(x_k + \alpha p_k)$$

- ▶ Simpler step computation, but step size must be carefully chosen.

## Trust Region Methods

- ▶ Construct a model  $m_k(p)$  approximating  $f(x_k + p)$ .
- ▶ Solve a constrained subproblem:

$$\min_{p: \|p\| \leq \Delta_k} m_k(p)$$

- ▶ Step is accepted if it gives sufficient reduction; otherwise shrink  $\Delta_k$ .

## Comments

Let us now compare the two fundamental strategies for unconstrained optimization: line search methods and trust region methods. In line search methods, we first fix a direction — typically a descent direction — and then look for a suitable step length that decreases the function along this direction. The step length is found by solving, either exactly or approximately, a one-dimensional minimization problem. Then we select a new direction and repeat the process. Trust region methods, on the other hand, follow a different logic. We first define a neighborhood — the trust region — around the current point, and within this region we minimize a model function  $m_k(p)$  that approximates  $f$ . The model is usually quadratic and based on local information such as the gradient and Hessian or their approximations. If the step computed from the model does not reduce the original function sufficiently, we shrink the trust region and try again. These two strategies differ not only in mechanics, but also in how they balance local approximation and global behavior.



## Comments

Let us now look at different ways to choose the search direction  $p_k$  in line search methods. The most basic choice is the steepest descent direction,  $p_k = -\nabla f_k$ . It is easy to compute but usually converges slowly, especially in ill-conditioned problems. A more powerful alternative is the Newton direction, which uses the inverse of the Hessian matrix. This can lead to rapid convergence, especially near the solution, provided the Hessian  $\nabla^2 f_k$  is positive definite. To avoid computing the full Hessian, quasi-Newton methods approximate it with a matrix  $B_k$  that is updated at each step. These methods offer a good compromise between speed and computational cost. Finally, the nonlinear conjugate gradient direction is often used in large-scale optimization. It combines the current gradient with the previous direction and avoids storing or inverting matrices altogether. All of these directions are used within the same basic line search framework, where the next iterate is computed by moving from  $x_k$  along  $p_k$  with some step length  $\alpha_k$ .



Trust-region methods solve:

$$\min_p m_k(p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T B_k p \quad \text{s.t. } \|p\| \leq \Delta_k$$

Typical model choices:

- Steepest descent model:  $B_k = 0$ , gives

$$p_k = -\Delta_k \frac{\nabla f_k}{\|\nabla f_k\|}$$

- Newton model:  $B_k = \nabla^2 f_k$  (exact Hessian)
- Quasi-Newton model:  $B_k$  approximates  $\nabla^2 f_k$

The trust-region constraint ensures existence of a solution, even if  $B_k$  is not positive definite.

## Comments

Let us now take a closer look at the types of models used in trust-region methods. At each iteration, the algorithm builds a local quadratic model  $m_k(p)$  of the objective function near the current point. This model includes a constant term  $f_k$ , a linear term  $p^T \nabla f_k$ , and a quadratic term  $\frac{1}{2} p^T B_k p$  defined by a symmetric matrix  $B_k$ . We then minimize this model subject to a constraint: the step must remain within a region of radius  $\Delta_k$ , called the trust region. When  $B_k = 0$ , we obtain a step that points in the direction of steepest descent, scaled to match the trust-region radius. More powerful models use second-order information. If  $B_k$  is the exact Hessian  $\nabla^2 f_k$ , we get the trust-region Newton. If  $B_k$  is built via quasi-Newton updates, we get the trust-region quasi-Newton method. Unlike line search, the trust-region formulation guarantees that a solution to the subproblem always exists (due to the constraint), even if the Hessian is not positive definite.



Optimization performance can be sensitive to variable scaling.

- A poorly scaled problem has variables with very different magnitudes.
- This can distort level sets and hinder convergence of gradient-based methods.

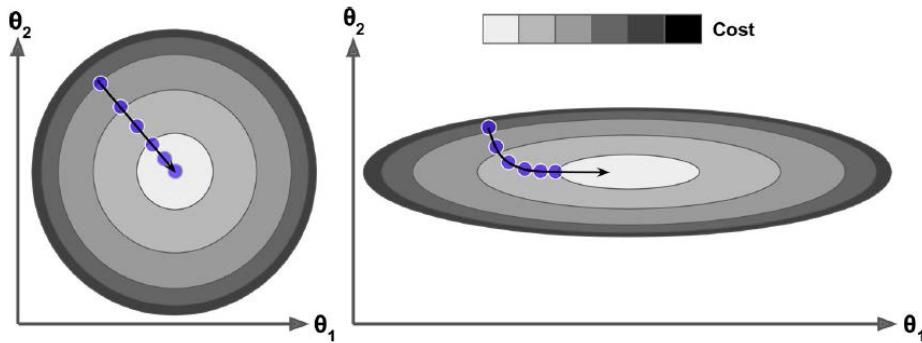
### Example

Minimize  $f(x) = 100x_1^2 + x_2^2$ . Level sets are highly elongated ellipses. Steepest descent zigzags and converges slowly.

Solution: Rescale variables so that typical values are of similar order. This can dramatically improve algorithm performance.

### Comments

Let us now briefly discuss the scaling problem. Optimization algorithms can behave very differently depending on how the variables are scaled. In poorly scaled problems, some variables may take values that are orders of magnitude larger than others. This stretches and distorts the level sets of the function, turning circles into narrow ellipses. In such cases, gradient-based methods like steepest descent may perform very poorly. The algorithm tends to zigzag and make slow progress toward the solution. A typical example is the function  $f(x) = 100x_1^2 + x_2^2$ . Its level sets are narrow ellipses, and steepest descent struggles with it. The solution is to rescale the variables so that their magnitudes are more uniform. This simple change can make a big difference in convergence speed and reliability.



- ▶ Balanced scaling yields nearly spherical level sets.
- ▶ Poor scaling results in elongated level sets.
- ▶ Gradient methods are sensitive to this distortion.

Proper scaling improves conditioning and accelerates convergence.

17/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

<b>Introduction</b>
<b>Taylor's Theorem</b>
<b>Optimality conditions</b>
<b>Strategies of optimization</b>
<b>Step length conditions</b>
<b>Convergence</b>



## Comments

This diagram illustrates how variable scaling affects the geometry of the objective function. On the left, we see a well-scaled problem: the level sets are nearly circular, and the gradient consistently points toward the minimum. On the right, one variable is scaled much more than the other, causing the level sets to stretch into narrow ellipses. In such cases, gradient-based methods tend to make slow progress or zigzag across the valley. Intuitively, the objective function becomes distorted — instead of a well-shaped bowl, we get a flat dish with a shallow bottom. This effect is especially pronounced in methods like steepest descent and nonlinear conjugate gradient, which rely solely on the gradient direction. Newton-type methods, on the other hand, use the Hessian matrix and are generally more tolerant to poor scaling, since the curvature information helps adjust the step direction and length. Overall, optimization methods that are less sensitive to variable scaling tend to be more reliable in practice, as they can handle poorly scaled problems more robustly. When designing algorithms, one should aim to preserve scale invariance not just in the search direction, but also in components like the line search or trust-region strategy and stopping criteria. In general, achieving scale invariance is easier in line search methods than in trust-region methods.



Basic update:

$$x_{k+1} = x_k + \alpha_k p_k$$

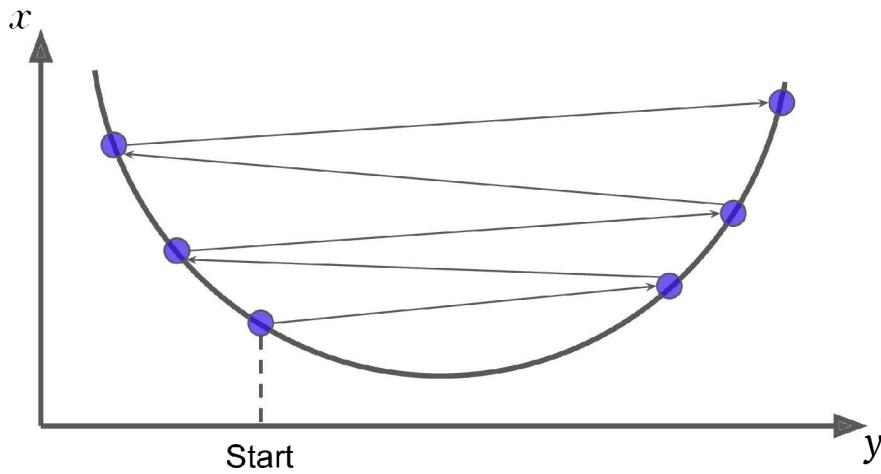
Direction  $p_k$  is usually fixed (e.g.,  $-\nabla f_k$ ), but step length  $\alpha_k$  must be carefully chosen:

- ▶ Too large: the algorithm may overshoot the minimum or diverge.
- ▶ Too small: the progress becomes negligible and convergence slows down.
- ▶ Inappropriate steps can lead to instability, even with a good search direction.

A good step length balances sufficient decrease with stability.

### Comments

Let us now focus on the role of the step length in line search methods. The search direction  $p_k$  is often determined by a specific rule, for example  $p_k = -\nabla f_k$ . But the choice of step length — that is, how far to move in that direction — is just as important. If the step is too large, the method may overshoot the region where the function decreases and fail to converge. If the step is too small, we make very little progress and waste many iterations. Even if the direction is well chosen, poor step sizes can lead to instability or stagnation. The goal is to strike a balance: to take steps that are not too small and not too aggressive, ensuring stable and consistent progress toward the solution.



Introduction
Taylor's Theorem
Optimality conditions
Strategies of optimization
Step length conditions
Convergence



Even when a descent direction is used, a poorly chosen step length may lead to divergence.

19/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

This figure illustrates a case where the optimization algorithm diverges. Although the search direction is a descent direction — for example, the negative gradient — the step length is chosen poorly: it is too large to stay within the region where the function decreases. As a result, the iterates move away from the solution. The function values may even start increasing. In some cases, this behavior leads to complete divergence — the algorithm never returns to the vicinity of the minimizer. This example emphasizes that it is not enough to choose a good direction — the step length must also be carefully controlled to ensure convergence.



We aim to choose  $\alpha_k$  such that:

- $f(x_k + \alpha_k p_k) < f(x_k)$  (descent)
- The decrease is not too small
- The search direction remains effective
- Convergence is ensured

However: Simple decrease is not sufficient for convergence. Counterexample:

Suppose the minimum value of  $f(x)$  is  $-1$ , and define  $f(x_k) = \frac{1}{k}$ . Then  $f(x_{k+1}) < f(x_k)$  for all  $k$ , but  $f(x_k) \rightarrow 0 > -1$  as  $k \rightarrow \infty$ . Conclusion: To avoid this

behavior, we must require that the decrease in  $f$  is substantial enough, i.e. we need to enforce a sufficient decrease condition, which we will discuss next.

## Comments

Let us now clarify what we expect from a good step length in line search methods. First and foremost, the step must produce a decrease in the objective function — that is,  $f(x_k + \alpha_k p_k) < f(x_k)$ . But decrease alone is not enough. We must also make sure that the decrease is not too small. Otherwise, we might waste many iterations making almost no progress. Furthermore, the choice of step length should support convergence in the long run. In particular, it must allow the iterates to approach an actual minimizer, not just wander around points with slightly decreasing values. These principles guide the design of practical step length rules. To illustrate why such care is necessary, let us look at a simple example. Suppose the minimum value of  $f$  is  $-1$ , and we define the value of  $f$  at each step as  $f(x_k) = 1/k$ . Then  $f(x_{k+1}) < f(x_k)$  — the function decreases at every step. But the values converge to zero, which is still strictly greater than the minimum. So, the method never reaches the minimizer, even though the function seems to decrease. To avoid this behavior, we must require that the decrease in  $f$  is substantial enough, i.e., we need to enforce a sufficient decrease condition, which we will discuss next.



We say that a step length  $\alpha > 0$  satisfies the sufficient decrease condition if

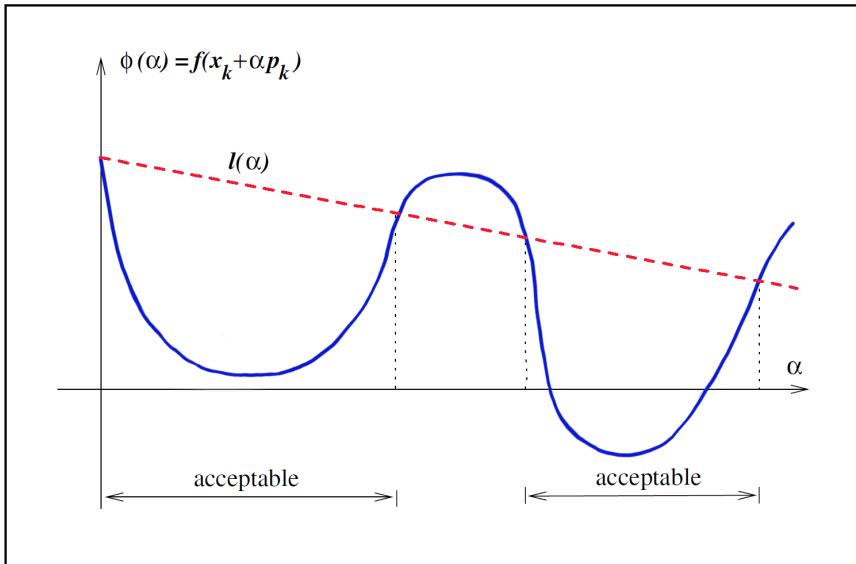
$$f(x + \alpha p) \leq f(x) + c_1 \alpha \nabla f(x)^T p, \quad c_1 \in (0, 1)$$

Interpretation:

- ▶ Ensures  $f$  decreases by at least a fixed fraction of the directional derivative.
- ▶ Prevents steps that produce only negligible improvement.
- ▶ Also known as the Armijo condition.

### Comments

We now formally introduce the sufficient decrease condition, which ensures that the step we take leads to a meaningful reduction in the objective function. A step length  $\alpha$  is said to satisfy this condition if  $f(x + \alpha p) \leq f(x) + c_1 \alpha \nabla f(x)^T p$ . The constant  $c_1$  is fixed in the interval  $(0, 1)$  and is typically chosen close to zero in practice — for example,  $10^{-4}$ . This means that the function must decrease by at least a small but fixed portion of how much we expect it to decrease in the direction  $p$ . This condition is often referred to as the Armijo condition, and it provides the foundation for practical step length selection.



<b>Introduction</b>
Taylor's Theorem
Optimality conditions
Strategies of optimization
<b>Step length conditions</b>
Convergence



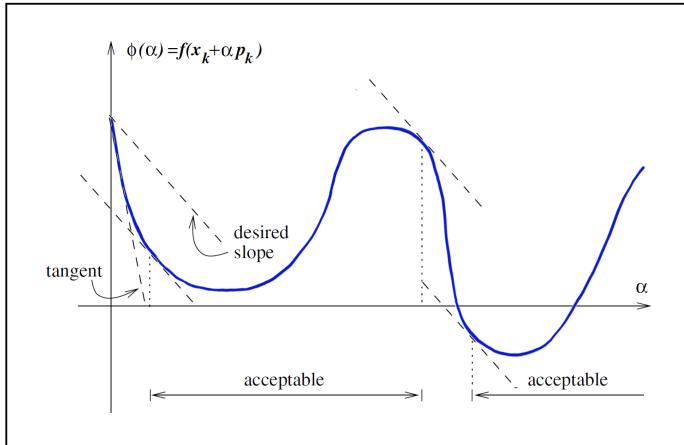
## Comments

Let us now interpret the sufficient decrease condition geometrically. We consider the function  $\phi(\alpha) = f(x_k + \alpha p_k)$ . It is shown in blue on the graph. The dashed line represents the value  $f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k$ . For small values of  $c_1$  — which are typically chosen in practice — this line decreases slowly and remains close to horizontal. Thus, in the neighborhood of  $\alpha = 0$ , the function  $\phi$  decreases more rapidly than this line. Therefore, the dashed line lies above the graph of  $\phi$ , which corresponds to the requirement of the sufficient decrease condition. In essence, the condition means that the actual reduction in the function must be no less than a fixed fraction of the linear approximation to  $f$  along the direction  $p_k$ . In the figure, the regions of acceptable values of  $\alpha$  — where this inequality is satisfied — are marked explicitly.

## The Curvature Condition

- To prevent overly small steps, we impose an additional constraint: the curvature condition:

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k, \text{ where } c_2 \in (c_1, 1).$$



Typical values:  $c_2 \approx 0.9$  for Newton-type methods,  $c_2 \approx 0.1$  for conjugate gradient.

23/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

<b>Introduction</b>
<b>Taylor's Theorem</b>
<b>Optimality conditions</b>
<b>Strategies of optimization</b>
<b>Step length conditions</b>
<b>Convergence</b>



### Comments

Let us now introduce the second condition used in line search algorithms — the curvature condition. As we have seen, the sufficient decrease condition alone is not enough. It may allow very small steps that formally reduce the function value but do not ensure real progress. The curvature condition aims to exclude such steps.

It requires that the directional derivative of the function  $f$  at the new point — that is, at  $x_k + \alpha p_k$  — is greater than or equal to  $c_2$  times the directional derivative of  $f$  at  $x_k$ . In other words, the slope of the function at the new point must not be too steep — it must be at least a fixed fraction of the initial slope. This ensures that the method does not stop while the function is still decreasing rapidly. Note that the left-hand side of the curvature condition is simply the derivative of the function  $\phi$  at the current step length  $\alpha$  — that is,  $\phi'(\alpha)$ . So this condition ensures that the slope of  $\phi$  at  $\alpha$  is no more negative than a fixed fraction of the initial slope at zero. This makes sense: if the slope is still sharply negative, the function may continue to decrease and a longer step would be preferable. On the other hand, if the slope is only mildly negative or has become positive, it indicates that no substantial decrease remains in that direction, and the line search should terminate. As we can see in the figure, the tangent at zero has a greater slope than the dashed line. Typical values for the constant  $c_2$  are close to one in Newton-type methods, and much smaller — around 0.1 — in conjugate gradient methods.



The **Wolfe conditions** for a step length  $\alpha > 0$ :

1. Sufficient decrease:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k \quad (1)$$

2. Curvature condition:

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k \quad (2a)$$

The strong Wolfe conditions replace the curvature condition by:

$$|\nabla f(x_k + \alpha p_k)^T p_k| \leq c_2 |\nabla f(x_k)^T p_k| \quad (2b)$$

Constants:  $0 < c_1 < c_2 < 1$

## Comments

The sufficient decrease and curvature conditions are known collectively as the Wolfe conditions. A step length  $\alpha$  satisfies the Wolfe conditions if, first, the value of the function at  $x_k + \alpha p_k$  is less than or equal to the value at  $x_k$  plus  $c_1 \alpha$  times the directional derivative at  $x_k$ . This ensures a meaningful decrease in the objective function. Second, the directional derivative at the new point is greater than or equal to  $c_2$  times the directional derivative at the current point. This prevents the step from being too short.

In many algorithms, especially those that do not assume convexity, it is more appropriate to use the strong Wolfe conditions. In that case, the curvature condition is replaced by requiring that the absolute value of the directional derivative at the new point is less than or equal to  $c_2$  times the absolute value of the initial directional derivative. This allows the derivative to change sign near a local minimizer or stationary point, while still limiting how steep it can be. Hence, we exclude points that are far from stationary points of  $\phi$ .



Exact line search is rarely used in practice:

- ▶ Too expensive to minimize  $f(x_k + \alpha p_k)$  exactly.
- ▶ Noisy or costly function evaluations in real-world problems.

Instead, inexact line search is used:

- ▶ Satisfy Wolfe or strong Wolfe conditions.
- ▶ Provide balance between decrease and progress.

Typical implementation:

- ▶ Start with  $\alpha = 1$ .
- ▶ Use backtracking or bracketing strategy.
- ▶ Adjust until Wolfe conditions are met.

## Comments

In theory, one could choose the step length  $\alpha$  by exactly minimizing the function  $f(x_k + \alpha p_k)$ . But in practice, this is almost never done. Exact line search is too computationally expensive, and in real-world applications, function evaluations can be noisy or expensive. Instead, most practical algorithms rely on inexact line search strategies. These strategies look for a step length  $\alpha$  that satisfies the Wolfe or strong Wolfe conditions. This approach is more efficient and still guarantees convergence under reasonable assumptions. A typical implementation works as follows: we start with  $\alpha = 1$ , then reduce it using backtracking or expand and shrink a bracketed interval, adjusting  $\alpha$  until both the sufficient decrease and the curvature conditions are satisfied. In the backtracking strategy, we repeatedly multiply  $\alpha$  by a factor less than one —for example, one-half — until the conditions are met. This ensures that we quickly reduce the step size if the function is not decreasing enough. The result is a compromise: we don't insist on finding the best possible  $\alpha$ , but we ensure that the function is decreasing enough, and that the step is not too short. This method is particularly effective in Newton-type methods, where the initial step is often close to acceptable. It is less suited for quasi-Newton and conjugate gradient methods, where step control requires more subtle handling.



### Lemma 1 (Step lengths Existence)

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be continuously differentiable. Suppose  $p_k$  is a descent direction at  $x_k$ , and  $f$  is bounded below along the ray  $\{x_k + \alpha p_k \mid \alpha > 0\}$ . Then, for any  $0 < c_1 < c_2 < 1$ , there exists an interval of step lengths  $\alpha$  such that the Wolfe and strong Wolfe conditions are satisfied.

**Proof.** Define the function  $\phi(\alpha) = f(x_k + \alpha p_k)$ . Then  $\phi(0) = f(x_k)$  and  $\phi'(0) = \nabla f(x_k)^T p_k < 0$ . We seek a value of  $\alpha > 0$  that satisfies:

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0), \quad \phi'(\alpha) \geq c_2 \phi'(0)$$

Such  $\alpha$  exists by analysis of the function  $\phi$  along the descent direction.

### Comments

Let us now justify the use of Wolfe conditions in line search algorithms by proving that suitable step lengths actually exist under mild assumptions. The lemma on this slide guarantees that for any constants  $c_1$  and  $c_2$  between zero and one, we can always find a value of  $\alpha$  that satisfies the Wolfe or strong Wolfe conditions — provided that the search direction is indeed a descent direction, and the function  $f$  is continuously differentiable and bounded below along the chosen ray. This result is important: it shows that the criteria we impose during line search are not overly restrictive and do not rule out the existence of a valid step. Now we begin the proof. We define a one-dimensional function  $\phi(\alpha)$  as  $f(x_k + \alpha p_k)$ . Since  $p_k$  is a descent direction  $\phi'(0) = \nabla f(x_k)^T p_k < 0$ . This tells us that the function  $\phi$  initially decreases, and we now seek a value of  $\alpha > 0$  such that  $\phi(\alpha)$  satisfies the sufficient decrease and curvature conditions. These two conditions can be rewritten in terms of  $\phi$ , and we will now study the behavior of this function to show that they can be simultaneously satisfied.

## Proof of Lemma 1 (Existence of Step Lengths)

Since  $\phi$  is bounded below by assumption, while  $\ell(\alpha) \rightarrow -\infty$  as  $\alpha \rightarrow \infty$  (because  $\phi'(0) < 0$  and  $c_1 > 0$ ), the graphs of  $\phi$  and  $\ell$  must intersect: there exists  $\alpha^* > 0$  such that

$$\phi(\alpha^*) = \ell(\alpha^*) \quad \text{and} \quad \phi(\alpha) < \ell(\alpha) \quad \text{for all } \alpha < \alpha^*.$$

By the mean value theorem, there exists  $\alpha' \in (0, \alpha^*)$  such that

$$\phi(\alpha^*) = \phi(0) + \alpha^* \phi'(\alpha').$$

Combining this with  $\ell(\alpha^*) = \phi(0) + c_1 \alpha^* \phi'(0)$ , we get:

$$\phi'(\alpha') = c_1 \phi'(0).$$

Since  $\phi'(0) < 0$  and  $c_1 < c_2$ , it follows that:

$$\phi'(\alpha') > c_2 \phi'(0).$$

Thus,  $\alpha'$  satisfies both Wolfe conditions.

27/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Introduction

Taylor's Theorem

Optimality conditions

Strategies of optimization

Step length conditions

Convergence



### Comments

Now we will complete the proof of the lemma using the mean value theorem.

Since the function  $\phi$  is bounded below, but the line  $\ell$  tends to minus infinity due to the negative slope  $\phi'(0)$ , the two graphs must intersect at some point  $\alpha^*$ . For all smaller  $\alpha$ ,  $\phi$  lies strictly below  $\ell$ . Applying the mean value theorem to the function  $\phi$  on the interval from zero to  $\alpha^*$ , we obtain some intermediate point  $\alpha'$  where the derivative matches the slope of the chord. Because the line  $\ell$  intersects  $\phi$  at  $\alpha^*$ , we equate the two expressions and find that the directional derivative of  $\phi$  at  $\alpha'$  is equal to  $c_1 \phi'(0)$ . Since  $c_1 < c_2$ , and  $\phi'(0)$  is negative, this derivative is strictly greater than  $c_2 \phi'(0)$ . Therefore,  $\alpha'$  satisfies both Wolfe conditions: sufficient decrease and the curvature condition. Since the derivative  $\phi$  at  $\alpha'$  is negative and  $c_1 < c_2$ , its absolute value is strictly less than  $c_2$  times the absolute value of the initial derivative. Therefore, the strong Wolfe condition holds in the same interval as well.



A step length  $\alpha > 0$  satisfies the Goldstein conditions if:

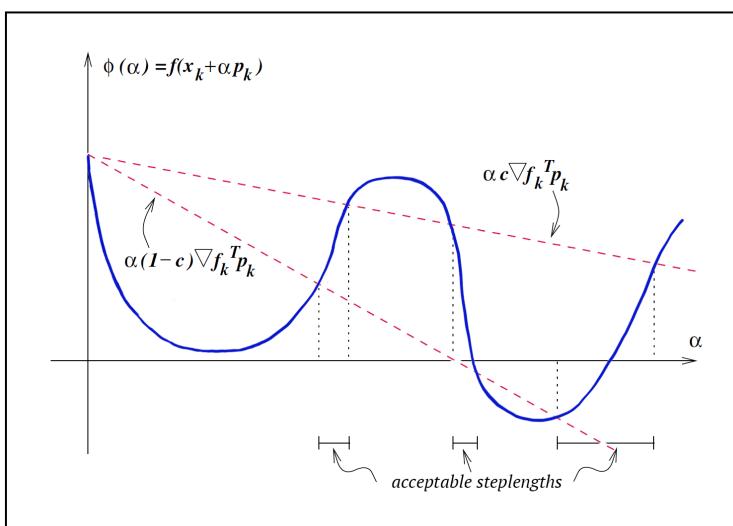
$$f(x_k) + (1 - c)\alpha \nabla f(x_k)^T p_k \leq f(x_k + \alpha p_k)$$

$$f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f(x_k)^T p_k$$

with  $c \in (0, 0.5)$ . These conditions ensure that the step is neither too short nor too long.

### Comments

The Goldstein conditions define another pair of inequalities used to guide the choice of step lengths in line search. A step  $\alpha$  satisfies these conditions if the value of the function at the trial point lies between two linear boundaries based on the initial value of the function and the directional derivative. The constant  $c$  must lie strictly between zero and one-half. The first inequality ensures that the step is not too short — that is, the function has decreased enough. The second inequality ensures that the step is not too long — meaning we have not overshot the region of useful decrease. Together, these bounds define an interval of acceptable step lengths centered around the optimal value predicted by linear approximation.



<b>Introduction</b>
<b>Taylor's Theorem</b>
<b>Optimality conditions</b>
<b>Strategies of optimization</b>
<b>Step length conditions</b>
<b>Convergence</b>



The Goldstein conditions define an interval that avoids too-short and too-long steps, but may exclude points where the directional derivative vanishes.

29/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

This figure illustrates the idea behind the Goldstein conditions. The step length  $\alpha$  must be chosen so that the function value lies between two lines defined by the initial value and the directional derivative.

The resulting interval excludes steps that are either too short or too long. However, a known drawback is that this condition can eliminate points where the directional derivative is zero — that is, local minimizers of the function  $\phi$ .

In contrast, the Wolfe conditions explicitly allow such points and are therefore more suitable for general-purpose optimization, especially in quasi-Newton methods where maintaining a positive definite Hessian approximation is essential.

Nonetheless, both Goldstein and Wolfe conditions share similar theoretical properties, and their convergence analysis is largely parallel.

The Goldstein conditions are often preferred in Newton-type methods where the curvature of the function is accurately captured by the Hessian.

## Theorem 6 (Global Convergence)

Let  $\{x_k\}$  be a sequence generated by the iteration  $x_{k+1} = x_k + \alpha_k p_k$ , where

- $p_k$  is a descent direction,
- the step length  $\alpha_k$  satisfies Wolfe conditions ((1) and (2a)).

Assume:

- (i) The function  $f$  is bounded below on  $\mathbb{R}^n$ .
- (ii)  $f$  is continuously differentiable in an open set  $\mathcal{N}$  containing the level set  $\mathcal{L} = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$ , where  $x_0$  is the starting point of the iteration.
- (iii) The gradient  $\nabla f$  is Lipschitz continuous on  $\mathcal{N}$ :

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| \quad \forall x, y \in \mathcal{N}.$$

$$\text{Then: } \sum_{k \geq 0} (\cos \theta_k)^2 \|\nabla f(x_k)\|^2 < \infty, \quad \text{where } \cos \theta_k = \frac{-\nabla f(x_k)^T p_k}{\|\nabla f(x_k)\| \|p_k\|}.$$

30/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

<b>Introduction</b>
<b>Taylor's Theorem</b>
<b>Optimality conditions</b>
<b>Strategies of optimization</b>
<b>Step length conditions</b>
<b>Convergence</b>



## Comments

We now state a result that concerns the convergence of line search methods in a global sense. That is, it guarantees that, regardless of the starting point  $x_0$ , and under certain general conditions on the function, its gradient, and the step lengths, the method will converge toward a stationary point.

More precisely, let's consider a sequence of iterates defined by the standard scheme: each new point is obtained by moving from the previous one along some search direction, scaled by a step length.

We assume that every direction is a descent direction, and that the step length satisfies Wolfe conditions, i.e., conditions (1) and (2a) from earlier.

In addition, the function is assumed to be bounded below on the whole space and continuously differentiable in an open set that contains the lower-level set of points where the function value is less than or equal to the initial value.

We also require that the gradient be Lipschitz continuous in that region. Under these assumptions, the theorem states that the sum of the squared gradients, weighted by the square of the cosine of the angle between the gradient and the search direction, is finite.

This result does not immediately imply that the gradient tends to zero, but it provides a crucial foundation for proving convergence under additional assumptions — for example, when the angle between the directions and gradients stays away from ninety degrees.

## Proof of Theorem 6 (part 1)

**Proof.** From the curvature condition (Wolfe's second condition), we have

$$(\nabla f(x_{k+1}) - \nabla f(x_k))^T p_k \geq (c_2 - 1) \nabla f(x_k)^T p_k$$

From Lipschitz continuity of  $\nabla f$ , we obtain

$$(\nabla f(x_{k+1}) - \nabla f(x_k))^T p_k \leq \|\nabla f(x_{k+1}) - \nabla f(x_k)\| \|p_k\| \leq \alpha_k L \|p_k\|^2$$

Combining both estimates:

$$\alpha_k \geq \frac{(c_2 - 1) \nabla f(x_k)^T p_k}{L \|p_k\|^2}$$

Substituting into the sufficient decrease condition (Armijo rule):

$$f(x_{k+1}) \leq f(x_k) - \frac{c_1(1 - c_2)}{L} \cdot \frac{(\nabla f(x_k)^T p_k)^2}{\|p_k\|^2}$$

Introduction
Taylor's Theorem
Optimality conditions
Strategies of optimization
Step length conditions
Convergence



### Comments

We begin the proof. According to the curvature condition, which is Wolfe's second condition, the scalar product of  $\nabla f(x_{k+1}) - \nabla f(x_k)$  with the direction  $p_k$  is greater than or equal to  $(c_2 - 1)$  times the scalar product of  $\nabla f(x_k)$  with the same direction  $p_k$ .

On the other hand, because the gradient is Lipschitz continuous, the same scalar product is less than or equal to  $\alpha_k$  times the Lipschitz constant  $L$  times the square of the norm of  $p_k$ .

Combining the two gives a lower bound for  $\alpha_k$ . It must be greater than or equal to the product of  $(c_2 - 1)$  and the scalar product of the gradient and  $p_k$ , divided by  $L$  times the squared norm of  $p_k$ .

Now we substitute this into the first Wolfe condition, also known as the Armijo condition. It states that the value of the function at  $x_{k+1}$  is less than or equal to the value at  $x_k$  plus  $\alpha_k$  times  $c_1$  times the directional derivative.

Substituting our estimate for  $\alpha_k$ , we get that the function decreases by at least a constant times the square of the scalar product of the gradient and  $p_k$ , divided by the squared norm of  $p_k$ .

## Proof of Theorem 6 (part 2)

**Proof (continued).** Using

$$\cos \theta_k = \frac{-\nabla f(x_k)^T p_k}{\|\nabla f(x_k)\| \cdot \|p_k\|}$$

we get

$$f(x_{k+1}) \leq f(x_k) - c (\cos \theta_k)^2 \|\nabla f(x_k)\|^2$$

where  $c = \frac{c_1(1-c_2)}{L} > 0$ . Summing over k:

$$c \sum_{k=0}^N (\cos \theta_k)^2 \|\nabla f(x_k)\|^2 \leq f(x_0) - f(x_{N+1})$$

Since  $f$  is bounded below by assumption, the sum must converge:

$$\sum_{k \geq 0} (\cos \theta_k)^2 \|\nabla f(x_k)\|^2 < \infty \quad \square$$

Introduction
Taylor's Theorem
Optimality conditions
Strategies of optimization
Step length conditions
<b>Convergence</b>



### Comments

We now continue the proof. To express everything in terms of the angle between the gradient and the direction, we use the definition of the cosine.

The cosine of the angle  $\theta_k$  is equal to minus the scalar product of  $\nabla f(x_k)$  and  $p_k$ , divided by the product of their norms.

Substituting the expression for cosine into the inequality, we get that  $f(x_{k+1}) \leq f(x_k) - c (\cos \theta_k)^2 \|\nabla f(x_k)\|^2$

Now we sum over all k from zero to N. On the left, we have a constant times the sum of  $(\cos \theta_k)^2$  times the squared gradient norm.

On the right, we have the total decrease in the function value, from  $x_0$  to  $x_{N+1}$ . Since the function is bounded below, the decrease is bounded above. Therefore, the sum must be finite.

This completes the proof.

It is worth noting that similar results can be obtained when different line search conditions are used.

In particular, if we replace the Wolfe conditions with the Goldstein conditions or the strong Wolfe conditions, the conclusion remains valid.

This shows that the convergence mechanism is robust with respect to the specific form of the step-size rules.

- Zoutendijk condition:

$$\sum_{k \geq 0} (\cos \theta_k)^2 \|\nabla f(x_k)\|^2 < \infty$$

- If  $(\cos \theta_k)^2$  is bounded away from zero, then  $\|\nabla f(x_k)\| \rightarrow 0$ .
- For Newton-type steps (when  $p_k = -B_k^{-1} \nabla f_k$ ), we have the estimate:

$$\cos \theta_k \geq \frac{1}{\kappa(B_k)}, \quad \text{where } \kappa(B_k) \text{ is the condition number of } B_k.$$

This follows from the inequality (if we substitute  $B = B_k$ ,  $a = \nabla f_k$ ):

$$\frac{a^T B a}{\|a\| \|B a\|} \geq \frac{a^T B a}{\|a\|^2 \|B\|} \geq \frac{\lambda_{\min}}{\lambda_{\max}} = \frac{1}{\kappa(B)} \quad \forall a \in \mathbb{R}^n, \text{ if } B \succ 0,$$

where  $\lambda_{\max}, \lambda_{\min}$  are the minimum and maximum eigenvalues of matrix  $B$ .

- Similar convergence holds for:
  - Newton and quasi-Newton methods with globalization
  - Conjugate gradient methods (under assumptions)
- Global convergence requires: descent direction + proper line search.

33/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Introduction

Taylor's Theorem

Optimality conditions

Strategies of optimization

Step length conditions

Convergence



## Comments

We have completed the discussion of global convergence for optimization methods using line search.

The main theoretical result in this part is the Zoutendijk condition, which holds for a wide class of methods that satisfy two key requirements: each step must use a descent direction, and the step size must satisfy Wolfe-type conditions.

The Zoutendijk condition states that the sum over all iterations of the squared cosine of the angle between the search direction and the negative gradient, multiplied by the squared norm of the gradient, is finite.

This is crucial, because if the angle between the direction and the gradient remains uniformly bounded away from ninety degrees, then it follows that the gradient norm must go to zero.

In other words, the iterates converge to a point satisfying the first-order optimality condition.

It is important to understand that global convergence means convergence from an arbitrary starting point to a stationary point.

This cannot be guaranteed by the core method alone — for example, Newton's method may diverge if used naively.

Therefore, additional mechanisms are introduced: the direction must be a descent direction, and the step size must ensure sufficient decrease. These elements are implemented through line search strategies satisfying conditions such as Armijo, Wolfe, or Goldstein.

In Newton or quasi-Newton methods, such as BFGS, we can estimate the angle between the gradient and the search direction: if the matrix  $B_k$  in Newton-type methods is symmetric and positive definite, then the cosine of this angle at iteration  $k$  is at least  $1/\kappa(B_k)$ , where  $\kappa(B_k)$  is the condition number of  $B_k$ .

This follows directly from the well-known inequality for a symmetric positive definite matrix.



- Zoutendijk condition:

$$\sum_{k \geq 0} (\cos \theta_k)^2 \|\nabla f(x_k)\|^2 < \infty$$

- If  $(\cos \theta_k)^2$  is bounded away from zero, then  $\|\nabla f(x_k)\| \rightarrow 0$ .
- For Newton-type steps (when  $p_k = -B_k^{-1} \nabla f_k$ ), we have the estimate:

$$\cos \theta_k \geq \frac{1}{\kappa(B_k)}, \quad \text{where } \kappa(B_k) \text{ is the condition number of } B_k.$$

This follows from the inequality (if we substitute  $B = B_k$ ,  $a = \nabla f_k$ ):

$$\frac{a^T B a}{\|a\| \|B a\|} \geq \frac{a^T B a}{\|a\|^2 \|B\|} \geq \frac{\lambda_{\min}}{\lambda_{\max}} = \frac{1}{\kappa(B_k)} \quad \forall a \in \mathbb{R}^n, \text{ if } B \succ 0,$$

where  $\lambda_{\max}, \lambda_{\min}$  are the minimum and maximum eigenvalues of matrix  $B$ .

- Similar convergence holds for:
  - Newton and quasi-Newton methods with globalization
  - Conjugate gradient methods (under assumptions)
- Global convergence requires: descent direction + proper line search.

34/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

This means that if  $B_k$  is well-conditioned, the direction cannot be nearly orthogonal to the gradient, and the Zoutendijk condition implies convergence. Besides gradient-based and quasi-Newton methods, global convergence theory also applies to conjugate gradient methods, which we will discuss further. These are especially effective for solving large-scale quadratic problems with sparse structure.

Under certain assumptions, such as convexity and exact line search, one can prove global convergence results for these methods as well.

Although more general settings require stronger assumptions, the overall logic remains similar: descent plus control over the direction leads to convergence. In conclusion, we now have a complete view of the conditions that guarantee convergence to stationary points with minimal assumptions.

The next step in our analysis is to study the rate of convergence, where we shift from the question "Will it converge?" to "How fast will it converge?" — a key concern for algorithmic efficiency.

## Convergence Rate of Steepest Descent

Consider minimizing a strictly convex quadratic:

$$f(x) = \frac{1}{2}x^T Qx - b^T x, \text{ where } Q \text{ is symmetric positive definite.}$$

Steepest descent step:  $p_k = -\nabla f(x_k) = -Qx_k + b$

The  $Q$ -norm of a vector  $v$  is defined as:  $\|v\|_Q = \sqrt{v^T Q v}$

By using the relation  $Qx^* = b$ , we can show that:  $\frac{1}{2}\|x_k - x^*\|_Q^2 = f(x_k) - f(x^*)$

### Theorem 7

Let  $f(x) = \frac{1}{2}x^T Qx - b^T x$ , where  $Q$  is symmetric positive definite with eigenvalues  $0 < \lambda_1 \leq \dots \leq \lambda_n$ . If the steepest descent method with exact line searches is applied to  $f$  starting from any  $x_0$ , then

$$f(x_{k+1}) - f^* \leq \left( \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right)^2 (f(x_k) - f^*), \text{ where } f^* \text{ is the minimum value of } f.$$

Proof in: David G. Luenberger, *Linear and Nonlinear Programming*, 4th ed., 2016 (see Theorem 2, p. 235).

35/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Introduction

Taylor's Theorem

Optimality conditions

Strategies of optimization

Step length conditions

Convergence



### Comments

Before proceeding to the analysis of the convergence rate of the steepest descent method, let us highlight some important conceptual points.

Convergence rate is not merely a technical concern. Algorithmic strategies often have to balance between achieving fast local convergence and maintaining global reliability. For example, the steepest descent method is the quintessential example of a globally convergent algorithm: it reliably finds a stationary point when proper step sizes are used. However, its practical rate of convergence is often quite slow.

Conversely, the pure Newton iteration achieves rapid convergence near a solution, but away from the solution its steps may not even be descent directions, which endangers global convergence.

In the following, we will examine the convergence rate for different line search strategies. We begin with the simplest case — the steepest descent method.

To better understand the nature of the steepest descent method, we consider the ideal case — minimizing a strictly convex quadratic function.

It has the form:  $f(x) = \frac{1}{2}x^T Qx - b^T x$ . Here, the matrix  $Q$  is symmetric and positive definite.

At each iteration, the steepest descent method selects the search direction as minus the gradient of the function at the current point.

Since the function is quadratic, the gradient is  $\nabla f(x_k) = Qx_k - b$ . Therefore, the search direction is  $p_k = -Qx_k + b$ .

To estimate the convergence rate for a quadratic function, we first need to introduce the concept of the  $Q$ -norm.

For a vector  $v$ , the  $Q$ -norm, denoted as  $\|v\|_Q$ , is the square root of  $v^T Q v$ . Here, the matrix  $Q$  is symmetric and positive definite, and it reflects the curvature of the quadratic function.

For a quadratic function  $f$ , there is a useful formula:  $\frac{1}{2}\|x_k - x^*\|_Q^2 = f(x_k) - f(x^*)$ . This relation is obtained by using the optimality condition that  $Qx^* = b$ .

## Convergence Rate of Steepest Descent

Consider minimizing a strictly convex quadratic:

$$f(x) = \frac{1}{2}x^T Qx - b^T x, \text{ where } Q \text{ is symmetric positive definite.}$$

Steepest descent step:  $p_k = -\nabla f(x_k) = -Qx_k + b$

The  $Q$ -norm of a vector  $v$  is defined as:  $\|v\|_Q = \sqrt{v^T Q v}$

By using the relation  $Qx^* = b$ , we can show that:  $\frac{1}{2}\|x_k - x^*\|_Q^2 = f(x_k) - f(x^*)$

### Theorem 7

Let  $f(x) = \frac{1}{2}x^T Qx - b^T x$ , where  $Q$  is symmetric positive definite with eigenvalues  $0 < \lambda_1 \leq \dots \leq \lambda_n$ . If the steepest descent method with exact line searches is applied to  $f$  starting from any  $x_0$ , then

$$f(x_{k+1}) - f^* \leq \left( \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right)^2 (f(x_k) - f^*), \text{ where } f^* \text{ is the minimum value of } f.$$

Proof in: David G. Luenberger, *Linear and Nonlinear Programming*, 4th ed., 2016 (see Theorem 2, p. 235).

36/36 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Introduction

Taylor's Theorem

Optimality conditions

Strategies of optimization

Step length conditions

Convergence



### Comments

Now, let us state a theorem that describes the convergence rate of the steepest descent method with exact line searches when optimizing a quadratic function.

Suppose the function  $f$  has the form:  $f(x) = \frac{1}{2}x^T Qx - b^T x$ .

The matrix  $Q$  is symmetric and positive definite, and its eigenvalues are ordered as follows:  $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ . Then, for any initial point  $x_0$ , the steepest descent method with exact line searches satisfies the following bound. The difference  $f(x_{k+1}) - f^*$  is at most  $\left( \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right)^2$  multiplied by the difference  $f(x_k) - f^*$ , where  $f^*$  is the minimum value of  $f$ .

Thus, the convergence rate is determined by the condition number of the matrix  $Q$ , which is the ratio of its largest eigenvalue  $\lambda_n$  to its smallest eigenvalue  $\lambda_1$ . The smaller the condition number, the faster the convergence.

In the extreme case where all eigenvalues are equal, the method achieves the minimum in a single iteration.

The proof of this theorem can be found in Luenberger's book *Linear and Nonlinear Programming*, fourth edition, 2016, Theorem 2 on page 235.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 2)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Newton's Method  
Superlinear convergence  
Global Convergence  
Cholesky Factorization  
Interpolation



Санкт-Петербургский  
государственный  
университет



30 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In today's lecture, we will explore the issue of convergence in Newton-type methods. We will also consider approaches to constructing Hessian approximations in quasi-Newton methods, specifically, algorithms based on Cholesky factorization.



Newton iteration defines the search direction as

$$p_k = -(\nabla^2 f_k)^{-1} \nabla f_k$$

where  $\nabla^2 f_k$  is the Hessian matrix at  $x_k$ .

### Theorem 8

Suppose that  $f$  is twice differentiable and that  $\nabla^2 f(x)$  is Lipschitz continuous near a solution  $x^*$  satisfying the second-order sufficient conditions (Theorem 4).

Consider the iteration  $x_{k+1} = x_k + p_k$ , where  $p_k$  is the Newton step.

Then:

- (i) If the starting point  $x_0$  is sufficiently close to  $x^*$ , the sequence of iterates converges to  $x^*$ .
- (ii) The convergence rate of  $\{x_k\}$  is quadratic.
- (iii) The sequence  $\{\|\nabla f_k\|\}$  converges quadratically to zero.

### Comments

We now consider the Newton iteration, for which the search direction is defined as  $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$ .

As discussed previously, Newton's method can be extremely effective — it converges rapidly under favorable conditions. However, such conditions are typically satisfied only near the solution. Away from the solution, the Newton step may not even be a descent direction, which raises issues with global convergence. It is therefore essential to distinguish between two situations: the behavior far from the minimizer (which may require modifications), and the behavior close to the minimizer. In the future, we will look at the approaches used in the first situation, and now we will discuss the second in detail.

The following theorem shows that if the initial point is sufficiently close to the solution, and the function is well-behaved (in particular, the Hessian is positive definite and Lipschitz continuous), then Newton's method performs extremely well: the iterates converge quadratically, and the gradient norms go to zero at a quadratic rate. This is the key advantage of Newton's method compared to first-order methods like steepest descent.

The following theorem describes the behavior near the solution. We assume that the function  $f$  is twice differentiable, and that the Hessian of  $f$  is Lipschitz continuous in a neighborhood of a solution  $x^*$ . We also assume that  $\nabla f(x^*) = 0$  and that  $\nabla^2 f(x^*)$  is positive definite. Under these assumptions, consider the Newton iteration:  $x_{k+1} = x_k + p_k$ . Then the following conclusions hold:

First, if the initial point  $x_0$  is sufficiently close to  $x^*$ , then the sequence  $\{x_k\}$  converges to  $x^*$ . Second, the convergence is quadratic. Third, the sequence  $\{\|\nabla f(x_k)\|\}$  converges to zero quadratically.

## Proof of Theorem 8 (I)

**Proof:** From the Newton step and the optimality condition  $\nabla f(x^*) = 0$ , we have

$$\begin{aligned} x_{k+1} - x^* &= x_k + p_k - x^* = x_k - x^* - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \\ &= (\nabla^2 f(x_k))^{-1} [\nabla^2 f(x_k)(x_k - x^*) - (\nabla f(x_k) - \nabla f(x^*))] \end{aligned}$$

By Taylor's theorem (Theorem 1):

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x_k + t(x^* - x_k))(x_k - x^*) dt$$

Hence,  $\nabla^2 f(x_k)(x_k - x^*) - (\nabla f(x_k) - \nabla f(x^*)) =$

$$\int_0^1 [\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))] (x_k - x^*) dt \Rightarrow$$

$$\|x_{k+1} - x^*\| \leq \|\nabla^2 f(x_k)^{-1}\| \cdot \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))\| \cdot \|x_k - x^*\| dt$$

2/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



### Comments

From the definition of the Newton step and the optimality condition  $\nabla f(x^*) = 0$ , we can write the Newton update as follows:

$$x_{k+1} - x^* = x_k - x^* - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

We now factor out the inverse Hessian. This gives:

$$x_{k+1} - x^* = (\nabla^2 f(x_k))^{-1} [\nabla^2 f(x_k)(x_k - x^*) - (\nabla f(x_k) - \nabla f(x^*))].$$

Now we apply Taylor's theorem to the gradient:

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x_k + t(x^* - x_k))(x_k - x^*) dt$$

Substituting this into the previous formula, we obtain an upper bound for the norm of the difference between the Newton iteration and the solution. This gives us the basic inequality we will now use to derive a quadratic bound on the distance to the solution.

## Proof of Theorem 8 (II)

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



Since  $\nabla^2 f$  is Lipschitz continuous near  $x^*$ , there exists  $L > 0$  such that

$$\|\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))\| \leq L \cdot t \cdot \|x_k - x^*\|$$

and we obtain:

$$\begin{aligned} \|x_{k+1} - x^*\| &\leq \|\nabla^2 f(x_k)^{-1}\| \cdot \int_0^1 Lt \|x_k - x^*\|^2 dt \\ &= \|\nabla^2 f(x_k)^{-1}\| \cdot L \cdot \|x_k - x^*\|^2 \cdot \int_0^1 t dt = \frac{L}{2} \cdot \|\nabla^2 f(x_k)^{-1}\| \cdot \|x_k - x^*\|^2 \end{aligned}$$

Since  $\nabla^2 f(x^*)$  is nonsingular there exist a constant  $r > 0$  such that for all  $x \in B_r(x^*)$ :

$$\|\nabla^2 f(x)^{-1}\| \leq 2 \|\nabla^2 f(x^*)^{-1}\|$$

3/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Since the Hessian of the function  $f$  is Lipschitz continuous near the point  $x^*$ , there exists a positive constant  $L$  such that  $\|\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))\| \leq L \cdot t \cdot \|x_k - x^*\|$ . Substituting this bound into the previous estimate for the Newton update, we obtain:  $\|x_{k+1} - x^*\| \leq \|\nabla^2 f(x_k)^{-1}\| \cdot \int_0^1 Lt \|x_k - x^*\|^2 dt$ .

Since  $\|x_k - x^*\|$  does not depend on the integration variable, we factor it out. Then the result becomes:  $\|x_{k+1} - x^*\| \leq \|\nabla^2 f(x_k)^{-1}\| \cdot L \cdot \|x_k - x^*\|^2 \cdot \int_0^1 t dt$ .

The value of that integral is  $\frac{1}{2}$ . So we arrive at the final bound:  $\|x_{k+1} - x^*\| \leq \frac{L}{2} \cdot \|\nabla^2 f(x_k)^{-1}\| \cdot \|x_k - x^*\|^2$ . Next, since  $\nabla^2 f(x^*)$  is nonsingular (by the condition of the theorem), and the function  $f$  is twice continuously differentiable, by continuity there exists constants  $r > 0$  such that for all  $x \in B_r(x^*)$ :  $\|\nabla^2 f(x)^{-1}\| \leq 2 \|\nabla^2 f(x^*)^{-1}\|$ .

## Proof of Theorem 8 (conclusion)

Thus, for all  $x_k \in B_r(x^*)$ :

$$\|x_{k+1} - x^*\| \leq \bar{L} \|x_k - x^*\|^2, \text{ where } \bar{L} = L \|\nabla^2 f(x^*)^{-1}\|$$

Choosing  $x_0$  so that  $\|x_0 - x^*\| \leq \min(r, 1/(2\bar{L}))$  we can use this inequality inductively to deduce that the sequence converges to  $x^*$ , and the rate of convergence is quadratic.

By using the relations  $x_{k+1} - x_k = p_k$  with  $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$  (i.e.  $\nabla f(x_k) + \nabla^2 f(x_k)p_k = 0$ ), we obtain that

$$\|\nabla f(x_{k+1})\| = \|\nabla f(x_{k+1}) - \nabla f(x_k) - \nabla^2 f(x_k)p_k\| =$$

$$= \left\| \int_0^1 \nabla^2 f(x_k + tp_k)(x_{k+1} - x_k) dt - \nabla^2 f(x_k)p_k \right\| \leq$$

$$\leq \int_0^1 \|\nabla^2 f(x_k + tp_k) - \nabla^2 f(x_k)\| \cdot \|p_k\| dt \leq$$

$$\leq \frac{1}{2} L \|p_k\|^2 \leq \frac{1}{2} L \|\nabla^2 f(x_k)^{-1}\|^2 \|\nabla f(x_k)\|^2 \leq 2L \|\nabla^2 f(x^*)^{-1}\|^2 \|\nabla f(x_k)\|^2$$

proving that the gradient norms converge to zero quadratically.  $\square$

4/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Newton's Method**

**Superlinear convergence**

**Global Convergence**

**Cholesky Factorization**

**Interpolation**



### Comments

Thus, for all  $x_k \in B_r(x^*)$ ,  $\|x_{k+1} - x^*\| \leq \bar{L} \|x_k - x^*\|^2$ , where  $\bar{L} = L \|\nabla^2 f(x^*)^{-1}\|$ . Choosing  $x_0$  so that  $\|x_0 - x^*\| \leq \min(r, 1/(2\bar{L}))$ , we inductively deduce that  $\{x_k\}$  converges to  $x^*$  quadratically.

It remains for us to prove the third part of the theorem. Using the Newton step  $x_{k+1} - x_k = p_k$ , where  $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$ , we bound  $\|\nabla f(x_{k+1})\|$ . This norm equals  $\left\| \int_0^1 \nabla^2 f(x_k + tp_k)p_k dt - \nabla^2 f(x_k)p_k \right\|$ . Since the Hessian is Lipschitz continuous with constant  $L$ , this is at most  $\int_0^1 Lt \|p_k\|^2 dt = \frac{1}{2} L \|p_k\|^2$ . Substituting  $p_k$ , we get  $\|p_k\|^2 \leq \|\nabla^2 f(x_k)^{-1}\|^2 \|\nabla f(x_k)\|^2$ . Since for  $x_k$  from the corresponding neighborhood of  $x^*$  we have  $\|\nabla^2 f(x_k)^{-1}\| \leq 2\|\nabla^2 f(x^*)^{-1}\|$ , it follows that  $\|\nabla f(x_{k+1})\| \leq 2L \|\nabla^2 f(x^*)^{-1}\|^2 \|\nabla f(x_k)\|^2$ , proving the gradient norms converge quadratically.

## Quasi-Newton Methods

In Quasi-Newton methods instead of the true Hessian  $\nabla^2 f_k$ , a matrix  $B_k$  is used and updated after each step to reflect new curvature information.

### The idea:

By Taylor's theorem we have:

$$\nabla f(x + p) = \nabla f(x) + \nabla^2 f(x)p + \int_0^1 [\nabla^2 f(x + tp) - \nabla^2 f(x)]p dt$$

Since  $\nabla^2 f(x)$  is continuous, the integral term is  $o(\|p\|)$ . Letting  $x = x_k$ ,  $p = x_{k+1} - x_k$ , we get:

$$\nabla f(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) + o(\|x_{k+1} - x_k\|)$$

Thus, when  $x_k$  and  $x_{k+1}$  lie in a region near the solution  $x^*$

$$\nabla^2 f(x_k)(x_{k+1} - x_k) \approx \nabla f(x_{k+1}) - \nabla f(x_k)$$

### Secant equation:

$$B_{k+1}s_k = y_k, \quad s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k$$

5/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Newton's Method**

**Superlinear convergence**

**Global Convergence**

**Cholesky Factorization**

**Interpolation**



### Comments

The main drawback of the Newton direction is the need for the Hessian. Explicit computation of this matrix of second derivatives can sometimes be a cumbersome, error-prone, and expensive process. When Hessian is not positive definite, the Newton direction may not even be defined, since the inverse Hessian may not exist. Even when it is defined, it may not satisfy the descent property  $\nabla f(x_k)^T p_k < 0$ , in which case it is unsuitable as a search direction. In these situations, line search methods modify the definition of  $p_k$  to make it satisfy the descent condition while retaining the benefit of the second-order information contained in Hessian.

Quasi-Newton methods provide an attractive alternative to Newton's method, as they avoid computing the Hessian while still achieving superlinear convergence. Instead of using the true Hessian, these methods use an approximation  $B_k$ , which is updated after each iteration based on observed gradient changes. The key idea is based on Taylor's theorem. Assume that the function  $f$  satisfies its conditions. Then  $\nabla f(x + p) = \nabla f(x) + \nabla^2 f(x)p + \int_0^1 [\nabla^2 f(x + tp) - \nabla^2 f(x)]p dt$ . Since the function is twice continuously differentiable, the integral is  $o(\|p\|)$ , so we approximate  $\nabla f(x_{k+1}) - \nabla f(x_k) \approx \nabla^2 f(x_k)(x_{k+1} - x_k)$ . This leads to the secant equation, which the updated matrix  $B_{k+1}$  is required to satisfy.

We choose  $B_{k+1}$  so that it satisfies the secant equation

$$B_{k+1}s_k = y_k, \quad s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

Additional conditions are typically imposed:

- $B_{k+1}$  should be symmetric (like the true Hessian),
- $B_{k+1} - B_k$  should be of low rank.

### SR1 update:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$

### BFGS update:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

SR1: rank-one update. BFGS: rank-two update, preserves positive definiteness if  $B_0 \succ 0$  and  $s_k^T y_k > 0$ .



## Comments

To construct effective Quasi-Newton methods, we choose the Hessian approximation matrix  $B_{k+1}$  to satisfy the secant equation  $B_{k+1}s_k = y_k$ , which ensures it mimics the Hessian's behavior along the step direction. We impose additional requirements on  $B_{k+1}$ : it should be symmetric, like the true Hessian of a twice-differentiable function, and the difference  $B_{k+1} - B_k$  should be low-rank to keep updates computationally efficient. Two widely used Quasi-Newton updates are the Symmetric Rank-One, or SR1, and the Broyden-Fletcher-Goldfarb-Shanno, or BFGS, formulas.

The SR1 update is defined as:  $B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$ . This is a rank-one update, meaning the change  $B_{k+1} - B_k$  has rank one, but it does not guarantee that  $B_{k+1}$  remains positive definite, which can affect convergence.

The BFGS update is:  $B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$ . This is a rank-two update, as the change has rank two. BFGS preserves positive definiteness of  $B_{k+1}$  if the initial matrix  $B_0$  is positive definite and  $s_k^T y_k > 0$ , ensuring robust convergence for convex problems.

Assume the search direction is given by  $p_k = -B_k^{-1}\nabla f(x_k)$  where  $B_k$  is symmetric positive definite and updated at each iteration by a quasi-Newton formula.

## Theorem 9

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice continuously differentiable. Let the iteration be:  $x_{k+1} = x_k + \alpha_k p_k$  with  $p_k$  a descent direction and  $\alpha_k$  satisfying Wolfe conditions with  $c_1 \leq \frac{1}{2}$ . If the sequence  $\{x_k\}$  converges to a point  $x^*$  such that  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*)$  is positive definite, and if the search direction satisfies

$$\lim_{k \rightarrow \infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k)p_k\|}{\|p_k\|} = 0$$

then:

- (i) the step length  $\alpha_k = 1$  is admissible for all  $k$  greater than a certain index  $k_0$ .
- (ii) if  $\alpha_k = 1$  for all  $k > k_0$ , then  $\{x_k\}$  converges to  $x^*$  superlinearly.

Proof in: Dennis, J. E., Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, 1996 (see Theorem 6.3.4, p. 123).

7/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Newton's Method**

**Superlinear convergence**

**Global Convergence**

**Cholesky Factorization**

**Interpolation**



## Comments

We consider a search direction defined as  $p_k = -B_k^{-1}\nabla f(x_k)$ . The matrix  $B_k$  is assumed to be symmetric and positive definite, and it is updated at each iteration using a quasi-Newton formula.

Theorem 9 states the following. Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be twice continuously differentiable. Suppose the sequence  $\{x_k\}$  is generated by  $x_{k+1} = x_k + \alpha_k p_k$ , where  $p_k$  is a descent direction. The step  $\alpha_k$  satisfies the Wolfe conditions with parameter  $c_1 \leq \frac{1}{2}$ .

Suppose further that  $\{x_k\}$  converges to a point  $x^*$  such that  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*)$  is positive definite. Assume also that  $\lim_{k \rightarrow \infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k)p_k\|}{\|p_k\|} = 0$ .

Then two conclusions follow. First: the step length  $\alpha_k = 1$  is admissible for all  $k$  greater than a certain index  $k_0$ . Second: if  $\alpha_k = 1$  for all  $k > k_0$ , then  $\{x_k\}$  converges to  $x^*$  superlinearly.

We will not prove this theorem. Its proof is similar to the proof of proposition 1.3.2 from Dimitri Bersekas' book "Nonlinear Programming".

Theorem 9 provides sufficient conditions for superlinear convergence of an iterative method in which the search direction is based on a quasi-Newton approximation. If the function is twice continuously differentiable, the search direction is a descent direction, and the step size satisfies the Wolfe conditions with a parameter less than or equal to one-half, and if the iterates converge to a point at which the gradient vanishes and the Hessian is positive definite, then two conclusions hold. First, a unit step is admissible for all sufficiently large iterations. Second, if a unit step is used at all sufficiently large iterations, the convergence is superlinear. By definition that means the ratio of the error at the next step to the current error tends to zero.

The condition in the theorem — that  $\lim_{k \rightarrow \infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k)p_k\|}{\|p_k\|} = 0$  — is automatically satisfied by Newton's method. In that case,  $\nabla f(x_k) + \nabla^2 f(x_k)p_k = 0$ , so the ratio is zero for all  $k$ . As a result, implementations of Newton's method with Wolfe or Goldstein line search conditions, and which always try a unit step first, will accept that step for all large  $k$  and achieve quadratic convergence locally.

This observation is also important in the analysis of quasi-Newton methods. Hence, we have the beautiful result that a superlinear convergence rate can be attained even if the sequence of quasi-Newton matrices  $\{B_k\}$  does not converge to  $\nabla^2 f(x^*)$ ; it suffices that the matrices  $B_k$  become increasingly accurate approximations to the Hessian along the search directions  $p_k$ .

## Theorem 10

Suppose  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice continuously differentiable. Consider the iteration  $x_{k+1} = x_k + p_k$  (i.e. the step length  $\alpha_k$  is uniformly 1), where  $p_k = -B_k^{-1}\nabla f(x_k)$  and  $B_k$  is symmetric and positive definite. Assume that  $\{x_k\}$  converges to a point  $x^*$  such that  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x)$  is positive definite at  $x^*$  and Lipschitz continuous near  $x^*$ . Then  $\{x_k\}$  converges to  $x^*$  superlinearly if and only if

$$\lim_{k \rightarrow \infty} \frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} = 0.$$

**Proof:** Assume that

$$\frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} \rightarrow 0.$$

This condition is equivalent to  $p_k - p_k^N = o(\|p_k\|)$ , where  $p_k^N = -\nabla^2 f(x_k)^{-1}\nabla f(x_k)$  is the Newton step. Indeed, since  $\nabla^2 f(x_k)^{-1}$  is bounded above for  $x_k$  sufficiently close to  $x^*$  we have

$$p_k - p_k^N = \nabla^2 f(x_k)^{-1}(\nabla^2 f(x_k) - B_k)p_k = O(\|(\nabla^2 f(x_k) - B_k)p_k\|) = o(\|p_k\|).$$

8/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



## Comments

In the previous lecture, we discussed Theorem 9. That result provided sufficient conditions for superlinear convergence of line search methods. It stated that, if the step satisfies Wolfe conditions and the search direction approximates the Newton direction well enough, then a unit step becomes admissible after some point, and using unit steps leads to superlinear convergence. Now we formulate a sharper result — Theorem 10 — which provides necessary and sufficient conditions for superlinear convergence in the case where a quasi-Newton approximation to the Hessian is used.

Assume that  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice continuously differentiable, and the sequence  $\{x_k\}$  converges to a point  $x^*$  such that  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*)$  is positive definite and Lipschitz continuous near  $x^*$ . Suppose also that the search direction is given by  $p_k = -B_k^{-1}\nabla f(x_k)$ . Then,  $\{x_k\}$  converges superlinearly if and only if  $\lim_{k \rightarrow \infty} \frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} = 0$ .

This means that the quasi-Newton approximation does not have to converge to the full Hessian. It is enough that the approximation becomes increasingly accurate in the directions of the iterates.

We now prove Theorem 10. Assume that  $\frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} \rightarrow 0$ .

The proof is based on showing that this condition is equivalent to  $p_k - p_k^N = o(\|p_k\|)$ , where  $p_k^N = -\nabla^2 f(x_k)^{-1}\nabla f(x_k)$  is the Newton step. Indeed,  $p_k - p_k^N = \nabla^2 f(x_k)^{-1}(\nabla^2 f(x_k) - B_k)p_k$ . Since  $\nabla^2 f(x_k)^{-1}$  is bounded above for  $x_k$  sufficiently close to  $x^*$ , we have  $p_k - p_k^N = O(\|(\nabla^2 f(x_k) - B_k)p_k\|) = o(\|p_k\|)$ .

## Superlinear Convergence and Quasi-Newton Approximations

Since the Newton step converges quadratically (see Theorem 8) and by previous estimation, we get

$$\|x_k + p_k - x^*\| \leq \|x_k + p_k^N - x^*\| + \|p_k - p_k^N\| = O(\|x_k - x^*\|^2) + o(\|p_k\|).$$

On the other hand, by Taylor's theorem (as we applied on slide 5) we have

$$\nabla f(x_k) = \nabla^2 f(x^*)(x_k - x^*) + o(\|x_k - x^*\|),$$

thus (since  $\nabla^2 f(x^*)$  and  $B_k$  are positive define)

$$\begin{aligned} \|p_k\| &= \| -B_k^{-1} \nabla f(x_k) \| \leq \| -B_k^{-1} \| (\|\nabla^2 f(x^*)\| \cdot \|x_k - x^*\| + o(\|x_k - x^*\|)) = \\ &= O(\|(x_k - x^*)\|) + o(\|x_k - x^*\|) = O(\|(x_k - x^*)\|) \end{aligned}$$

hence:

$$\|x_{k+1} - x^*\| \leq O(\|x_k - x^*\|^2) + o(O(\|(x_k - x^*)\|)) = o(\|x_k - x^*\|),$$

which completes the proof.  $\square$

9/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



### Comments

Next, we estimate  $\|x_{k+1} - x^*\|$ . This distance is bounded by  $\|x_k + p_k^N - x^*\| + \|p_k - p_k^N\|$ . The first term decreases quadratically — it is  $O(\|x_k - x^*\|^2)$  — and the second term is  $o(\|p_k\|)$ .

On the other hand, by Taylor's theorem,  $\nabla f(x_k) = \nabla^2 f(x^*)(x_k - x^*) + o(\|x_k - x^*\|)$ . Therefore,  $\|p_k\| = \| -B_k^{-1} \nabla f(x_k) \| \leq \|B_k^{-1}\| (\|\nabla^2 f(x^*)\| \cdot \|x_k - x^*\| + o(\|x_k - x^*\|)) = O(\|x_k - x^*\|)$ .

Substituting this back into our previous bound, we conclude that  $\|x_{k+1} - x^*\| \leq O(\|x_k - x^*\|^2) + o(\|x_k - x^*\|) = o(\|x_k - x^*\|)$ . This is exactly the definition of superlinear convergence. The theorem is proved.

When  $\nabla^2 f(x_k)$  is not positive definite, the Newton step

$$\nabla^2 f(x_k) p_k = -\nabla f(x_k)$$

may not define a descent direction. To ensure positive definiteness, we replace the Hessian by a modified matrix:

$$B_k = \nabla^2 f(x_k) + E_k,$$

where  $E_k = 0$  if the Hessian is sufficiently positive definite, and otherwise chosen to ensure  $B_k \succ 0$ .

### Algorithm 1: Line Search Newton with Modification

```

1: Given: initial point  $x_0$ 
2: for  $k = 0, 1, 2, \dots$  do
3:   Compute  $B_k = \nabla^2 f(x_k) + E_k$ 
4:   Solve  $B_k p_k = -\nabla f(x_k)$ 
5:   Update  $x_{k+1} = x_k + \alpha_k p_k$ , where  $\alpha_k$  satisfies Wolfe or Armijo conditions
6: end for

```

**Newton's Method**

**Superlinear convergence**

**Global Convergence**

**Cholesky Factorization**

**Interpolation**



### Comments

Away from the solution,  $\nabla^2 f(x_k)$  may not be positive definite. In this situation, the classic Newton step  $\nabla^2 f(x_k) p_k = -\nabla f(x_k)$  may not produce a descent direction. This is problematic because descent directions are essential for line search strategies to work.

To address this, we replace the Hessian by a modified matrix  $B_k = \nabla^2 f(x_k) + E_k$ . If the Hessian is already sufficiently positive definite,  $E_k = 0$ . Otherwise,  $E_k$  is chosen to ensure  $B_k \succ 0$ .

This leads to Algorithm 1. The method begins with an initial point  $x_0$ . At each iteration, we compute the modified matrix  $B_k$ , solve the linear system  $B_k p_k = -\nabla f(x_k)$  to obtain the step direction, and update the iterate  $x_{k+1} = x_k + \alpha_k p_k$ . The step length  $\alpha_k$  is determined by line search conditions such as Wolfe or Armijo. This modified Newton method can be applied from any starting point. It ensures that the search direction is always a descent direction and allows global convergence, provided the modification ensures that the matrices  $B_k$  remain well-conditioned throughout the process.

## Theorem 11

Let  $f$  be twice continuously differentiable on an open set  $D \subset \mathbb{R}^n$ , and let  $x_0 \in D$  be such that

the level set  $L = \{x \in D : f(x) \leq f(x_0)\}$  is compact.

If the sequence  $\{B_k\}$  generated by Algorithm 1 satisfies the bounded modified factorization property:

$$\kappa(B_k) = \|B_k\| \cdot \|B_k^{-1}\| \leq C \text{ for all } k,$$

then

$$\lim_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0.$$

**Proof in:** J. J. Moré, D. Sorensen, *Newton's Method*, Argonne National Laboratory, Argonne, Illinois, 1982 (see Theorem 4.15, p. 23).

**Interpretation:** If the step directions remain well-conditioned, the algorithm converges globally.

11/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



## Comments

We now state Theorem 11, which is a key result guaranteeing global convergence of the modified Newton method. Assume the function is twice continuously differentiable on an open set. The starting point is chosen so that the level set — that is, the set of points where the function value is less than or equal to its value at the starting point — is compact. This level set is denoted  $L$ , and being compact means, it is closed and bounded.

Next, assume that the matrices  $B_k$ , constructed by the algorithm, satisfy the bounded modified factorization property. That is, the condition number of each matrix  $B_k$ , defined as  $\|B_k\| \cdot \|B_k^{-1}\|$ , is uniformly bounded above by a constant  $C$ , for all iterations. Then the theorem asserts that  $\lim_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0$  — in other words, the method converges to a stationary point.

This result is crucial because it ensures that the modified Newton method converges globally, even when the Hessian is not positive definite at early steps. As long as the modifications maintain good numerical properties — especially well-conditioned search directions — the algorithm is guaranteed to work. Hence, Algorithm 1 is globally reliable under the assumptions of this theorem.

Newton's Method
Superlinear convergence
Global Convergence
Cholesky Factorization
Interpolation

Assume that the iterates  $\{x_k\}$  generated by Algorithm 1 converge to a point  $x^*$ , and that the Hessian  $\nabla^2 f(x^*)$  is positive definite for all sufficiently large  $k$ . Then, for all sufficiently large  $k$ , the correction vanishes:

$$E_k = 0 \Rightarrow B_k = \nabla^2 f(x_k).$$

By Theorem 10, assuming  $\alpha_k = 1$ , the method converges quadratically:

$$\|x_{k+1} - x^*\| = O(\|x_k - x^*\|^2).$$

**Caveat:** If  $\nabla^2 f(x^*)$  is nearly singular,  $E_k$  may not vanish. Then the convergence rate may be only linear.



## Comments

Let us now examine the convergence rate of the modified Newton method. Suppose that the sequence of iterates converges to a point  $x^*$ , and the Hessian at that point is sufficiently positive definite. Then, after some point, the modification term  $E_k$  becomes zero, and  $B_k$  equals the actual Hessian. The method thus becomes the pure Newton method.

Under these conditions, and assuming that unit steps are accepted, the convergence becomes quadratic. That is,  $\|x_{k+1} - x^*\| = O(\|x_k - x^*\|^2)$ .

However, this favorable outcome does not always occur. If the Hessian at the solution is nearly singular — for example, poorly conditioned or close to losing positive definiteness — then the modification  $E_k$  may never vanish. In such cases, the method does not revert to the classical Newton regime. The convergence may then be only linear. This limitation is essential to keep in mind in practical implementations.

In addition to ensuring that the modified matrix  $B_k$  has a bounded condition number (to satisfy the requirements of Theorem 11), we aim to minimize the magnitude of the modification to preserve the second-order information encoded in the Hessian as much as possible. Furthermore, we seek a modified factorization that can be computed efficiently with reasonable computational cost.

Newton's Method
Superlinear convergence
Global Convergence
Cholesky Factorization
Interpolation

**Goal:** Replace  $\nabla^2 f(x_k)$  by a positive definite  $B_k$  that is close to it, well-conditioned, and cheap to compute.

**Step 1:** Compute symmetric factorization

$$\nabla^2 f(x_k) = Q \Lambda Q^T,$$

where  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$

**Step 2:** Modify negative or small eigenvalues

$$\hat{\lambda}_i = \max\{\lambda_i, \delta\} \quad \text{with small } \delta > 0$$

**Step 3:** Reconstruct

$$B_k = Q \hat{\Lambda} Q^T$$

**Guarantees:**  $B_k$  is symmetric, positive definite and preserves curvature.



## Comments

We now introduce the first practical technique for modifying the Hessian — based on eigenvalue correction. The goal of this approach is to construct a positive definite matrix  $B_k$  that:

- (1) preserves as much second-order information as possible, (2) remains well-conditioned (so Theorem 11 applies), and is computationally feasible.

The method proceeds in three steps. In the first step, we compute a symmetric factorization of the Hessian:  $\nabla^2 f(x_k) = Q \Lambda Q^T$ , where  $Q$  is orthogonal and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$  contains the eigenvalues. In the second step, we correct the spectrum: any negative or very small eigenvalue is replaced by a small positive threshold  $\delta$ . This guarantees positive definiteness. In the third step, we reconstruct the matrix using the same orthogonal transformation and the modified eigenvalues:  $B_k = Q \hat{\Lambda} Q^T$ , where  $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_n)$  and  $\hat{\lambda}_i = \max\{\lambda_i, \delta\}$ . The resulting matrix  $B_k$  is symmetric, positive definite, and stays close to the original Hessian.

This method is attractive because it maintains curvature information, ensures theoretical convergence guarantees, and introduces minimal perturbation to the problem. Moreover, it can be implemented efficiently if the eigen structure is already available or cheaply approximated.

**Example:** Let

$$\nabla f(x_k) = \begin{pmatrix} 1 \\ -4 \\ 4 \end{pmatrix}, \quad \nabla^2 f(x_k) = \text{diag}(8, 1, -2)$$

Then Newton step solves  $p_k^N = -\nabla^2 f(x_k)^{-1} \nabla f(x_k) = (-0.125, 4, 2)^T$  and  $\nabla f(x_k)^T p_k^N = 7.975 > 0$ .

By the spectral decomposition theorem we have

$$\nabla^2 f(x_k) = Q \Lambda Q^T = \sum_{i=1}^n \lambda_i q_i q_i^T$$

where  $Q = I$  and  $\lambda_1 = 8, \lambda_2 = 1, \lambda_3 = -2$ . Modify eigenvalues: set  $\hat{\lambda}_3 = \delta = 10^{-8}$

$$B_k = \sum_{i=1}^2 \lambda_i q_i q_i^T + \frac{1}{\delta} q_3 q_3^T = \text{diag}(8, 1, 10^{-8})$$

Then the modified Newton step is

$$p_k = -B_k^{-1} \nabla f_k = -\sum_{i=1}^2 \frac{1}{\lambda_i} q_i (q_i^T \nabla f_k) - \frac{1}{\delta} q_3 (q_3^T \nabla f_k) \approx -(4 \times 10^8) q_3$$

14/30 || SPbU & HIT 2025 || Shpilov P.V. || Classical optimization approaches

Newton's Method
Superlinear convergence
Global Convergence
Cholesky Factorization
Interpolation



## Comments

Now we will consider a specific example that illustrates how a poor choice of  $\delta$  can adversely affect the efficiency of the method. Let the gradient at the current point be  $\nabla f(x_k) = (1, -4, 4)^T$ , and let the Hessian be  $\nabla^2 f(x_k) = \text{diag}(8, 1, -2)$ , which is clearly indefinite. The Newton step is given by  $p_k^N = -\nabla^2 f(x_k)^{-1} \nabla f(x_k) = (-0.125, 4, 2)^T$ . The inner product  $\nabla f(x_k)^T p_k^N = 7.975 > 0$ . Therefore, the Newton direction is not a descent direction.

To construct a suitable approximation, we apply the spectral decomposition theorem. The Hessian is equal to  $\nabla^2 f(x_k) = Q \Lambda Q^T = \sum_{i=1}^n \lambda_i q_i q_i^T$ . In this case,  $Q = I$  and the eigenvalues are  $\lambda_1 = 8, \lambda_2 = 1, \lambda_3 = -2$ . We now modify the spectrum by replacing the negative eigenvalue with a small positive number  $\delta = 10^{-8}$ . The resulting modified matrix is  $B_k = \text{diag}(8, 1, 10^{-8})$ .

The modified Newton step is given by  $p_k = -B_k^{-1} \nabla f_k = -\sum_{i=1}^2 \frac{1}{\lambda_i} q_i (q_i^T \nabla f_k) - \frac{1}{\delta} q_3 (q_3^T \nabla f_k) \approx -(4 \times 10^8) q_3$ . Although  $f$  decreases along the direction  $p_k$ , its extreme length violates the spirit of Newton's method, which relies on a quadratic approximation of the objective function that is valid in a neighborhood of the current iterate  $x_k$ . It is therefore not clear that this search direction is effective.

Newton's Method
Superlinear convergence
Global Convergence
Cholesky Factorization
Interpolation

Let  $A$  be symmetric with spectral decomposition  $A = Q\Lambda Q^T$ , where  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ .

#### Frobenius-norm-optimal correction:

$$\Delta A = Q \text{diag}(\tau_i) Q^T, \quad \tau_i = \begin{cases} 0, & \lambda_i \geq \delta \\ \delta - \lambda_i, & \lambda_i < \delta \end{cases}$$

$$A + \Delta A = Q(\Lambda + \text{diag}(\tau_i))Q^T$$

#### Euclidean-norm-optimal diagonal correction:

$$\tau = \max(0, \delta - \lambda_{\min}(A)), \quad \Delta A = \tau I$$

$$A + \Delta A = A + \tau I$$



### Comments

We now consider different strategies for modifying a Hessian matrix so that it becomes positive definite. A number of alternatives to the standard eigenvalue thresholding are possible. For example, we may flip the signs of negative eigenvalues; in the previous example, this would correspond to setting  $\delta = 1$ . Another option is to completely remove the component of the search direction along the negative curvature directions, which amounts to dropping the last term in the modified Newton step formula. Yet another approach is to choose  $\delta$  adaptively, so that the resulting step does not become excessively large. This approach has similarities with trust-region methods.

These examples illustrate that there is a wide variety of possible modification strategies, and there is currently no agreement in the field about which is best. Some strategies prioritize preserving curvature, others focus on controlling the step norm, and others prioritize computational simplicity. Setting aside the question of how to choose  $\delta$  for now, we examine how to construct a modification that is in a certain sense optimal.

Suppose that the matrix  $A$  is symmetric and has a spectral decomposition:  $A = Q\Lambda Q^T$ , where  $Q$  is an orthogonal matrix, and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ .

To ensure that all eigenvalues of the modified matrix are greater than or equal to a small positive threshold  $\delta$ , we define a correction matrix  $\Delta A$  as follows. It is given by  $\Delta A = Q \text{diag}(\tau_i) Q^T$ , where each  $\tau_i = 0$  if  $\lambda_i \geq \delta$ , and  $\tau_i = \delta - \lambda_i$  otherwise. The resulting modified matrix is  $A + \Delta A = Q(\Lambda + \text{diag}(\tau_i))Q^T$ . This correction has the smallest Frobenius norm among all possible corrections that guarantee the minimum eigenvalue is at least  $\delta$ .

Alternatively, we can apply a uniform diagonal shift. We compute a scalar  $\tau = \max(0, \delta - \lambda_{\min}(A))$ . The correction matrix is then  $\Delta A = \tau I$ , and the modified matrix becomes  $A + \tau I$ . This modification has the smallest Euclidean norm among all diagonal corrections that achieve the same eigenvalue bound.

The Frobenius-optimal correction changes only the problematic eigenvalues and leaves the large eigenvalues untouched. The diagonal shift is simpler and affects the entire spectrum equally. Both strategies are useful, and the choice between them depends on computational considerations and desired accuracy.

## Adding a Multiple of the Identity

**Goal:** Make  $\nabla^2 f(x_k) + \tau I$  sufficiently positive definite by choosing  $\tau > 0$

**Algorithm 2** (Cholesky with Added Multiple of the Identity):

```
Choose parameter  $\beta > 0$ 
2: if all diagonal elements of  $\nabla^2 f(x_k)$  are positive then
    $\tau_0 \leftarrow 0$ 
4: else
    $\tau_0 \leftarrow -\min a_{ii} + \beta$ 
6: end if
   for  $j = 0, 1, 2, \dots$  do
8:   if Cholesky factorization of  $A + \tau_k I$  succeeds (i.e.  $A + \tau_k I \succ 0$ ) then
      stop and return factor  $L$  ( $LL^T = A + \tau_k I$ )
10:  else
       $\tau_{j+1} \leftarrow \max(2\tau_j, \beta)$ 
12:  end if
   end for
```

**Note:** The choice of  $\beta$  is heuristic; a typical value is  $\beta = 10^{-3}$

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



### Comments

This slide presents Algorithm 2, which modifies the Hessian in Newton's method to ensure it is positive definite, as part of Algorithm 1. The goal is to find a scalar  $\tau > 0$  such that the matrix  $\nabla^2 f(x_k) + \tau I$  is positive definite, ensuring the Newton step is a descent direction.

The algorithm starts by selecting a positive parameter  $\beta$ , typically  $10^{-3}$ , a heuristic choice. If all diagonal elements of the Hessian are positive, we set  $\tau_0 = 0$ , as the Hessian may already be positive definite. Otherwise,  $\tau_0$  is set to  $-\min a_{ii} + \beta$ , providing an initial shift. The algorithm then loops over indices  $j$ , attempting the Cholesky factorization of the Hessian plus  $\tau_j I$ .

Why Cholesky? Computing  $A + \tau I$  is simple, but we need to confirm the matrix is positive definite, meaning all eigenvalues are positive. Cholesky factorization tests this efficiently: it succeeds only if the matrix is positive definite. It also provides the factor  $L$  to solve the Newton system,  $(\nabla^2 f(x_k) + \tau I)p_k = -\nabla f(x_k)$ , without additional cost.

If the factorization succeeds, we return the factor, as the matrix is positive definite. If it fails,  $\tau_j$  is too small, so we set  $\tau_{j+1} = \max(2\tau_j, \beta)$  and try again. The heuristic  $\beta$  balances making  $\tau$  large enough for positive definiteness and small enough to preserve the Hessian's information. This method is simple but may require multiple factorizations, which can be costly, as each trial of  $\tau$  needs a new factorization.



## Definition

For a symmetric positive definite matrix  $A \in \mathbb{R}^{n \times n}$ , the Cholesky factorization is:

$$A = LL^T,$$

where  $L$  is lower triangular with positive diagonal elements.

## Application to Newton System:

$$\nabla^2 f(x_k) p_k = -\nabla f(x_k).$$

Using  $\nabla^2 f(x_k) = LL^T$ :

1. Solve  $Lz = -\nabla f(x_k)$  (forward substitution).
2. Solve  $L^T p_k = z$  (backward substitution).

## Why Cholesky?

- Tests positive definiteness: Factorization exists only if  $A$  is positive definite.
- Efficient: Costs  $\frac{1}{3}n^3$  flops, versus  $\frac{2}{3}n^3$  for LU factorization.
- Solves system directly: Triangular systems are fast ( $O(n^2)$  flops).

17/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The Cholesky factorization applies to a symmetric positive definite matrix  $A$ , such as the Hessian or a modified Hessian in Newton's method. It decomposes  $A$  into  $LL^T$ , where  $L$  is a lower triangular matrix with positive diagonal elements. This factorization is crucial for solving the Newton system,  $\nabla^2 f(x_k) p_k = -\nabla f(x_k)$ , which determines the search direction in Algorithm 1. Using the factorization, we replace the Hessian with  $LL^T$ . This transforms the system into two easier steps.

First, we solve  $Lz = -\nabla f(x_k)$ , a forward substitution since  $L$  is lower triangular, which is computationally simple. Second, we solve  $L^T p_k = z$ , a backward substitution, also straightforward due to the triangular structure.

Why use Cholesky factorization? It serves two purposes.

First, it tests whether the matrix is positive definite. The factorization exists only if all eigenvalues of  $A$  are positive, confirming that the Newton direction is a descent direction. This is why Algorithm 2 uses Cholesky to check if  $\nabla^2 f(x_k) + \tau I$  is positive definite.

Second, it's efficient. Computing the Cholesky factorization costs about  $\frac{1}{3}n^3$  flops, half the cost of a general LU factorization. Solving the two triangular systems costs  $O(n^2)$  flops, much faster than direct methods.

In Algorithm 2, Cholesky ensures the modified Hessian,  $\nabla^2 f(x_k) + \tau I$ , is positive definite and provides the factors to solve the Newton system efficiently. This method avoids expensive eigenvalue computations while ensuring a robust and fast solution process. Next, we'll explore the modified Cholesky factorization, which adjusts the Hessian during factorization to handle non-positive definite cases.

## Modified Cholesky Factorization

**Goal:** Modify  $\nabla^2 f(x_k)$  during factorization to ensure positive definiteness, avoiding multiple factorizations (see Algorithm 2).

**Algorithm 3** (Modified Cholesky,  $LDL^T$  Form):

```
1: for j = 1, 2, ..., n do
2:    $c_{jj} \leftarrow a_{jj} - \sum_{x=1}^{j-1} d_x l_{jx}^2$ 
3:    $\theta_j \leftarrow \max_{j < i \leq n} |c_{ij}|$ , where  $c_{ij} \leftarrow a_{ij} - \sum_{x=1}^{j-1} d_x l_{ix} l_{jx}$ 
4:    $d_j \leftarrow \max \left( |c_{jj}|, \left( \frac{\theta_j}{\beta} \right)^2, \delta \right)$ 
5:   for i = j + 1, ..., n do
6:      $c_{ij} \leftarrow a_{ij} - \sum_{x=1}^{j-1} d_x l_{ix} l_{jx}$ 
7:      $l_{ij} \leftarrow c_{ij}/d_j$ 
8:   end for
9: end for
```

**Note:** Parameters  $\delta, \beta > 0$  ensure  $d_j \geq \delta$  and bounded factors. Produces  $PAP^T + E = LDL^T$ , where  $E$  is diagonal.

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



### Comments

Now let's consider the Modified Cholesky Factorization as a more efficient alternative to Algorithm 2 for ensuring a positive definite Hessian in Newton's method. The goal is to modify the Hessian,  $\nabla^2 f(x_k)$ , during its factorization to make it positive definite, avoiding the repeated factorizations required in Algorithm 2.

Algorithm 3 computes an  $LDL^T$  factorization, where  $L$  is lower triangular with unit diagonal, and  $D$  is diagonal with positive elements. For each column  $j$ , it calculates  $c_{jj}$  as the diagonal element adjusted by prior terms, and  $\theta_j$  as the maximum absolute off-diagonal element in the remaining columns. The key modification is in setting  $d_j$ , the diagonal element of  $D$ , to the maximum of  $|c_{jj}|$ ,  $(\theta_j/\beta)^2$ , and a positive parameter  $\delta$ . This ensures  $d_j$  is at least  $\delta$ , guaranteeing positive definiteness, and controls the size of the factors to keep them bounded. For each row  $i > j$ , it computes  $c_{ij}$  and sets  $l_{ij} = c_{ij}/d_j$ , forming the off-diagonal elements of  $L$ . Parameters  $\delta$  and  $\beta$ , both positive, ensure  $d_j$  is sufficiently large and the factors  $L$  and  $D$  are not too large, maintaining numerical stability.

The algorithm may include row and column permutations, represented by a permutation matrix  $P$ , producing  $PAP^T + E = LDL^T$ , where  $E$  is a nonnegative diagonal matrix that's zero if the Hessian is already positive definite. This approach modifies the Hessian only when necessary, preserving its second-order information while ensuring a descent direction for Algorithm 1.

Next, we'll illustrate this with an example to show how the modification works in practice.

## Example of Modified Cholesky Factorization

**Example 1:** Let  $A = \begin{bmatrix} 4 & 2 & 1 \\ 2 & -1 & 0 \\ 1 & 0 & 2 \end{bmatrix}$ ,  $\text{EV} = 5, \sqrt{3}, -\sqrt{3}$ . Set  $\delta = 0.1, \beta = 1$ .

**Steps:**

- $j = 1$ :  $c_{11} = a_{11} = 4, \theta_1 = \max(|a_{21}|, |a_{31}|) = 2, d_1 = \max(4, 4, 0.1) = 4$ .

$$l_{21} = a_{21}/d_1 = 0.5, \quad l_{31} = a_{31}/d_1 = 0.25.$$

- $j = 2$ :  $c_{22} = a_{22} - d_1 l_{21}^2 = -1 - 4 \cdot 0.5^2 = -2, \theta_2 = |a_{32} - d_1 l_{31} l_{21}| = 0.5$ ,

$$d_2 = \max(2, 0.25, 0.1) = 2, \quad l_{32} = (a_{32} - d_1 l_{31} l_{21})/d_2 = -0.25.$$

- $j = 3$ :  $c_{33} = a_{33} - d_1 l_{31}^2 - d_2 l_{32}^2 = 2 - 4 \cdot 0.25^2 - 2 \cdot (-0.25)^2 = 1.625$ ,

$$d_3 = \max(1.625, 0, 0.1) = 1.625.$$

**Result:**  $A + E = LDL^T$ , with  $E = \text{diag}(0, 4, 0)$ , where  $E_{jj} = d_j - c_{jj}$ .

Newton's Method  
Superlinear convergence  
Global Convergence  
**Cholesky Factorization**  
Interpolation



19/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Let's consider a three-by-three symmetric matrix  $A = \begin{bmatrix} 4 & 2 & 1 \\ 2 & -1 & 0 \\ 1 & 0 & 2 \end{bmatrix}$ . The eigenvalues

of this matrix are  $5, \sqrt{3}$ , and  $-\sqrt{3}$ , indicating that  $A$  is indefinite and needs modification before it can be used in Newton's method.

We now apply the Modified Cholesky Factorization. We choose  $\delta = 0.1$  and  $\beta = 1$ . Then, step by step, we compute the values used in the factorization. For each index  $j$  from 1 to 3, we calculate the corrected diagonal entry  $d_j$  and the multipliers  $l_{ij}$  according to the algorithm.

Once all steps are completed, we can compute the correction matrix  $E$  as the difference between  $d_j$  and the elements  $c_{jj}$ . This gives a diagonal matrix  $E = \text{diag}(0, 4, 0)$ . Thus, we obtain that  $A + E = LDL^T$ , where  $L$  is unit lower triangular and  $D$  is diagonal with strictly positive entries.

This example shows how, using Algorithm 3, we can modify the matrix  $A$  to ensure positive definiteness with minimal changes. In the next part, we will explain why the specific formula for  $d_j$  used in step 4 is chosen to balance the need for positive definiteness and numerical stability.

## Cholesky Factorization, $LDL^T$ Form

Every symmetric positive definite matrix  $A$  can be written as  $A = LDL^T$  where  $L$  is a lower triangular matrix with unit diagonal elements and  $D$  is a diagonal matrix with positive elements on the diagonal.

**Example 2:** Consider the case  $n = 3$ . The equation  $A = LDL^T$  is given by

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{bmatrix}.$$

$$a_{11} = d_1,$$

$$a_{21} = d_1 l_{21} \implies l_{21} = \frac{a_{21}}{d_1},$$

$$a_{31} = d_1 l_{31} \implies l_{31} = \frac{a_{31}}{d_1},$$

$$a_{22} = d_1 l_{21}^2 + d_2 \implies d_2 = a_{22} - d_1 l_{21}^2,$$

$$a_{32} = d_1 l_{31} l_{21} + d_2 l_{32} \implies l_{32} = \frac{a_{32} - d_1 l_{31} l_{21}}{d_2},$$

$$a_{33} = d_1 l_{31}^2 + d_2 l_{32}^2 + d_3 \implies d_3 = a_{33} - d_1 l_{31}^2 - d_2 l_{32}^2.$$

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



20/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

It is well-known from algebra, that every symmetric positive definite matrix  $A$  can be written as  $A = LDL^T$  where  $L$  is a lower triangular matrix with unit diagonal elements and  $D$  is a diagonal matrix with positive elements on the diagonal.

Let us illustrates the Cholesky factorization in  $LDL^T$  form for a three-by-three symmetric positive definite matrix  $A$ . We express  $A$  as  $LDL^T$ . The matrices  $L$  and  $D$  are shown on the slide.

By equating the elements of  $A$  with those of  $LDL^T$  column by column, we derive formulas for the elements of  $L$  and  $D$ . For the first column,  $a_{11} = d_1$ , and the subdiagonal elements yield  $l_{21} = a_{21}/d_1$  and  $l_{31} = a_{31}/d_1$ .

For the second column,  $a_{22} = d_1 l_{21}^2 + d_2$ , giving  $d_2 = a_{22} - d_1 l_{21}^2$ , and  $a_{32}$  yields  $l_{32} = (a_{32} - d_1 l_{31} l_{21})/d_2$ .

The third column gives  $d_3 = a_{33} - d_1 l_{31}^2 - d_2 l_{32}^2$ . For a positive definite matrix  $A$ , the values  $d_j$  computed in these formulas are equivalent to  $c_{jj}$  in Algorithm 3, ensuring positive diagonal elements.

Further, we will discuss why Algorithm 3 uses a modified formula for  $d_j$  to handle cases where  $A$  is not positive definite, ensuring numerical stability and positive definiteness.

## Problem with standard LDL<sup>T</sup>:

- For indefinite  $\nabla^2 f(x_k)$ ,  $A = LDL^T$  may not exist (e.g.,  $c_{jj} \leq 0$ ).
- If it exists, L, D elements can grow arbitrarily large, causing numerical instability.
- Post-factorization fix (e.g., forcing  $d_j > 0$ ) may drastically alter A.

## Algorithm 3 solution:

$$d_j = \max \left( |c_{jj}|, \left( \frac{\theta_j}{\beta} \right)^2, \delta \right),$$

ensures:

- $d_j \geq \delta > 0$ : Positive definiteness of  $A + E$ .
- $d_j \geq \left( \frac{\theta_j}{\beta} \right)^2$ : Bounds  $|l_{ij}| \leq \beta$ , controlling L.
- $d_j \geq |c_{jj}|$ : Minimizes  $E_{jj} = d_j - c_{jj}$ .

**Result:**  $A + E = LDL^T$ , with bounded L, D and small E.

**Note:** Parameters  $\delta, \beta > 0$  balance stability and fidelity to  $\nabla^2 f(x_k)$ .

Newton's Method
Superlinear convergence
Global Convergence
Cholesky Factorization
Interpolation



## Comments

As we just saw in a standard LDL<sup>T</sup> factorization, we compute A as LDL<sup>T</sup>, setting  $d_j$  equal to  $c_{jj}$  at each step. For the Hessian,  $\nabla^2 f(x_k)$ , this assumes positive definiteness. If the Hessian is indefinite, with negative or zero eigenvalues,  $c_{jj}$  may be negative or zero, causing the factorization to fail. Even if the factorization exists for an indefinite matrix, the elements of L and D can become arbitrarily large, leading to numerical instability. For example, small diagonal elements amplify off-diagonal elements in L, making the factorization unreliable.

One might consider computing the standard factorization and then adjusting D afterward to make its elements positive. However, this post-hoc modification can significantly change the matrix, producing a result far from the original Hessian, which loses valuable second-order information needed for Algorithm 1.

Instead, Algorithm 3 modifies the Hessian during factorization by setting  $d_j$  to the maximum of  $|c_{jj}|$ ,  $(\theta_j/\beta)^2$ , and  $\delta$ , where  $\theta_j$  is the largest absolute off-diagonal element in the remaining columns, and  $\delta$  and  $\beta$  are positive parameters. This formula achieves three goals.

First,  $d_j \geq \delta$ , ensuring all diagonal elements of D are positive, so  $A + E$ , where E is the modification matrix, is positive definite, suitable for solving the Newton system.

Second,  $d_j \geq (\theta_j/\beta)^2$ , which bounds the off-diagonal elements  $l_{ij}$ , computed as  $c_{ij}/d_j$ . Since  $c_{ij}$  is at most  $\theta_j$ , this ensures  $|l_{ij}| \leq \beta$ , keeping L well-conditioned and preventing large factors that cause instability.

Third,  $d_j = |c_{jj}|$  when  $c_{jj}$  is sufficiently large and positive, minimizing  $E_{jj} = d_j - c_{jj}$ . This keeps A + E close to the original Hessian, preserving its curvature information for effective Newton steps.

The result is  $A + E = LDL^T$ , where E is a diagonal matrix with entries  $d_j - c_{jj}$ , and both L and D have bounded elements, ensuring numerical stability and a positive definite matrix. Parameters  $\delta$  and  $\beta$  balance the trade-off between making E large enough for positive definiteness and small enough to stay close to the Hessian.

This approach improves on Algorithm 2 by performing a single, stable factorization, as seen in our first example. Next, we'll discuss the modified symmetric indefinite factorization, another method for handling indefinite Hessians.

## Example: Symmetric Indefinite Factorization

Any symmetric matrix  $A$ , whether positive definite or not, can be written as  $PAP^T =LBL^T$  (where  $P$ : permutation,  $L$ : unit lower triangular,  $B$ : block diagonal with blocks of dimension 1 or 2).

### Definition

Inertia of a matrix is the number of positive, negative, and zero eigenvalues.

**Factorization Example:** Let's  $P = [e_1 \ e_4 \ e_3 \ e_2]$ ,

$$A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & -1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & \frac{1}{3} & 1 & 0 \\ 2 & \frac{2}{3} & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & -\frac{5}{3} \\ 0 & 0 & -\frac{5}{3} & -\frac{19}{3} \end{bmatrix}.$$

**Inertia:**  $B$  has one  $1 \times 1$  positive block, one  $2 \times 2$  block (three positive, zero zero, and one negative eigenvalue).

Newton's Method  
Superlinear convergence  
Global Convergence  
Cholesky Factorization

Interpolation



### Comments

We mentioned earlier that attempting to compute the  $LDL^T$  factorization of an indefinite matrix, where  $D$  is a diagonal matrix, is inadvisable because even if the factors  $L$  and  $D$  are well defined, they may contain entries that are larger than the original elements of  $A$ , thus amplifying rounding errors that arise during the computation.

However, by using the block diagonal matrix  $B$ , which allows  $2 \times 2$  blocks as well as  $1 \times 1$  blocks on the diagonal, we can guarantee that the  $LBL^T$  factorization always exists and can be computed by a numerically stable process. The symmetric indefinite factorization allows us to determine the inertia of a matrix, that is, the number of positive, zero, and negative eigenvalues.

One can show that the inertia of  $B$  equals the inertia of  $A$ . Moreover, the  $2 \times 2$  blocks in  $B$  are always constructed to have one positive and one negative eigenvalue. Thus, the number of positive eigenvalues in  $A$  equals the number of positive  $1 \times 1$  blocks plus the number of  $2 \times 2$  blocks.

Here is an example of such a factorization for an indefinite  $4 \times 4$  symmetric matrix  $A$ . The matrices  $P$ ,  $L$ , and  $B$  are computed to satisfy  $PAP^T =LBL^T$ , revealing the inertia of  $A$ : three positive, zero zero, and one negative eigenvalue.

## Modified Symmetric Indefinite Factorization

We now consider techniques for finding a minimum of the one-dimensional function

### Modification Steps:

1. Compute  $B = Q\Lambda Q^T$ , define  $F = Q \text{diag}(\tau_i)Q^T$ , where

$$\tau_i = \begin{cases} 0, & \lambda_i \geq \delta, \\ \delta - \lambda_i, & \lambda_i < \delta, \end{cases}$$

for a positive  $\delta$ .

2. Form modified factorization:  $P(A + E)P^T = L(B + F)L^T$ , where  $E = P^T L F L^T P$ .

**Exercise:** For  $A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & -1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 3 \end{bmatrix}$ , modify  $B$  to ensure  $\lambda_{\min}(B + F) \geq \delta$ .

**Note:**  $F$  ensures positive definiteness with minimal Frobenius norm.

23/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



### Comments

An indefinite symmetric factorization algorithm can be modified to ensure that the modified factors are the factors of a positive definite matrix. The idea is to modify the matrix  $B$  so that as a result of this modification, all the eigenvalues become positive. To do this, first we need to calculate the spectral decomposition of this matrix. The spectral decomposition of  $B$  is computed as  $B = Q\Lambda Q^T$ , where  $\Lambda$  contains the eigenvalues  $\lambda_i$ .

A modification matrix  $F$  is defined as  $F = Q \text{diag}(\tau_i)Q^T$ , where  $\tau_i = 0$  if  $\lambda_i \geq \delta$ , and  $\tau_i = \delta - \lambda_i$  otherwise, for a positive  $\delta$ . This ensures that  $B + F$  has a minimum eigenvalue of at least  $\delta$ . The modification  $F$  is designed to minimize the Frobenius norm while ensuring positive definiteness. The modified factorization becomes  $P(A + E)P^T = L(B + F)L^T$ , where  $E = P^T L F L^T P$ . (Note that  $E$  will not be diagonal, in general.)

Hence, in contrast to the modified Cholesky approach, this modification strategy changes the entire matrix  $A$ , not just its diagonal. The goal is for the modified matrix to satisfy that the smallest eigenvalue of  $A + E$  is approximately equal to  $\delta$ , whenever the smallest eigenvalue of the original matrix  $A$  is less than  $\delta$ . However, it is unclear whether this goal is always achieved.

As a small exercise, you can try to modify matrix  $B$  from the previous example and construct a modified factorization for matrix  $A$ .

Minimize one-dimensional function:

$$\phi(\alpha) = f(x_k + \alpha p_k). \quad (*)$$

Assume  $p_k$  is a descent direction ( $\phi'(0) < 0$ ), so  $\alpha > 0$ .

**Convex Quadratic:**

$$f(x) = \frac{1}{2}x^T Qx - b^T x, \quad \alpha_k = -\frac{\nabla f_k^T p_k}{p_k^T Q p_k}. \quad (**)$$

**Procedure:**

- ▶ Bracketing phase: Find interval  $[\bar{a}, \bar{b}]$  with acceptable step lengths.
- ▶ Selection phase: Zoom in to locate final step length.

**Note:** Derivative-based algorithms are efficient, often terminating after one evaluation near solutions.

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



## Comments

Let us consider techniques for finding a minimum of the one-dimensional function  $\phi(\alpha) = f(x_k + \alpha p_k)$ , as given in equation (\*). We assume that  $p_k$  is a descent direction—that is,  $\phi'(0) < 0$ —so that our search can be confined to positive values of  $\alpha$ . This ensures that moving along  $p_k$  will reduce the function value for some positive step length. For a convex quadratic function, defined as  $f(x) = \frac{1}{2}x^T Qx - b^T x$ , the one-dimensional minimizer along the ray  $x_k + \alpha p_k$  can be computed analytically. This minimizer is given by  $\alpha_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T Q p_k}$ , as shown in equation (\*\*).

For general nonlinear functions, it is necessary to use an iterative procedure, as finding the exact minimizer is too costly. Line search procedures consist of two phases: a bracketing phase that finds an interval containing acceptable step lengths, and a selection phase that zooms in to locate the final step length. Algorithms that use derivative information are more efficient, as they can often determine whether a suitable step length has been located after just one evaluation, particularly when the current iterate is close to the solution. This efficiency is critical for the robustness and performance of nonlinear optimization methods.

The aim is to find a value of  $\alpha$  that satisfies the sufficient decrease condition (1), without being “too small.” The sufficient decrease condition is:

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0).$$

If the initial trial value  $\alpha_0 > 0$  does not satisfy this condition, an improved estimate can be computed using interpolation.

### Quadratic interpolation:

Constructed using function values at  $\alpha_0 > 0$  and 0, and the derivative at 0. Then  $m(\alpha)$  has the form:

$$m(\alpha) = \phi(0) + \phi'(0)\alpha + \left[ \frac{\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0}{\alpha_0^2} \right] \alpha^2,$$

and its minimizer is:

$$\alpha = -\frac{\phi'(0)\alpha_0^2}{2[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0]}.$$

**Newton's Method**

**Superlinear convergence**

**Global Convergence**

**Cholesky Factorization**

**Interpolation**



## Comments

The purpose of using interpolation in the line search procedure is to find a value of  $\alpha$  that satisfies the sufficient decrease condition without being too small. Recall that the sufficient decrease condition is:  $\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0)$ . The constant  $c_1$  is usually chosen to be small in practice (say  $c_1 = 10^{-4}$ ). We typically start with a positive initial guess  $\alpha_0$ . If this initial trial step does not satisfy the condition, then a refined estimate can be generated using interpolation based on previously computed function and derivative values.

This slide describes the quadratic interpolation strategy. A quadratic model  $m(\alpha)$  is constructed using the value of the function at 0, its derivative at 0, and the value of the function at  $\alpha_0$ . This model approximates  $\phi$  locally and provides an analytical formula for the minimum of the quadratic, which serves as a new candidate for the step length. It's important to be cautious when using such formulas. In some cases, the denominator might be close to zero or even negative, leading to unreliable or invalid step sizes. Therefore, the result of interpolation must always be checked before being accepted as the next trial step.

## Cubic Interpolation: Four-Point Model

If the sufficient decrease condition is satisfied at  $\alpha_1$ , we terminate the search. Otherwise, we build a cubic function  $\phi_c(\alpha)$  interpolating:

$$\phi(0), \quad \phi'(0), \quad \phi(\alpha_0), \quad \phi(\alpha_1).$$

The cubic model takes the form:

$$\phi_c(\alpha) = a\alpha^3 + b\alpha^2 + \alpha\phi'(0) + \phi(0),$$

where coefficients  $a$  and  $b$  are given by:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_0^2 \alpha_1^2 (\alpha_1 - \alpha_0)} \begin{bmatrix} \alpha_0^2 & -\alpha_1^2 \\ -\alpha_0^3 & \alpha_1^3 \end{bmatrix} \begin{bmatrix} \phi(\alpha_1) - \phi(0) - \phi'(0)\alpha_1 \\ \phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0 \end{bmatrix}.$$

The minimizer of  $\phi_c(\alpha)$  is:

$$\alpha_2 = \frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a}.$$

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



### Comments

If the sufficient decrease condition is satisfied at  $\alpha_1$ , we terminate the search. Otherwise, we construct a cubic model that interpolates four known values:  $\phi(0)$ ,  $\phi'(0)$ , and  $\phi(\alpha_0)$  and  $\phi(\alpha_1)$ . That is, the model must match the values of the function  $\phi$  and its derivative at these points. The cubic model has an explicit form with coefficients  $a$  and  $b$ , which are determined by solving a linear system.

These coefficients depend on the differences between the function values and their linear approximations at  $\alpha_0$  and  $\alpha_1$ . In addition, the model provides an analytic formula for its minimizer. If the discriminant under the square root in this formula is negative, it indicates that the cubic model is unreliable. In such cases, we discard the interpolation result and fall back to a safer choice - for example, halving the previous step length. This safeguard ensures steady progress and prevents the algorithm from stalling. If, at the  $i$ -th iteration, the computed  $\alpha$  is either too close to the previous value or significantly smaller, we do the same: we reset  $\alpha_i$  to be equal to half of  $\alpha_{i-1}$ , where  $\alpha_i$  and  $\alpha_{i-1}$  denote the step lengths used at iterations  $i$  and  $i - 1$  of the optimization algorithm, respectively.

Why not just always halve the step size instead of using interpolation? Because interpolation uses available information - function and derivative values - to more intelligently predict a suitable step. This often leads to finding an acceptable step with fewer function evaluations. Halving remains a fallback strategy when interpolation fails.

When both function and directional derivative values are available at little cost, we can interpolate  $\phi$  and  $\phi'$  at two previous points  $\alpha_{i-1}$  and  $\alpha_i$ :

- A unique cubic interpolant always exists;
- Its minimizer  $\alpha_{i+1}$  lies in  $[\bar{a}, \bar{b}]$  and is given by:

$$\alpha_{i+1} = \alpha_i - (\alpha_i - \alpha_{i-1}) \left[ \frac{\phi'(\alpha_i) + d_2 - d_1}{\phi'(\alpha_i) - \phi'(\alpha_{i-1}) + 2d_2} \right],$$

where

$$d_1 = \phi'(\alpha_{i-1}) + \phi'(\alpha_i) - 3 \frac{\phi(\alpha_{i-1}) - \phi(\alpha_i)}{\alpha_{i-1} - \alpha_i},$$

$$d_2 = \text{sign}(\alpha_i - \alpha_{i-1}) [d_1^2 - \phi'(\alpha_{i-1})\phi'(\alpha_i)]^{1/2}.$$



## Comments

This slide presents an alternative interpolation strategy that uses both function and derivative values at two points. The key idea is that if evaluating the derivative is not much more expensive than evaluating the function itself, we can leverage more information to build a better model. In particular, we use a cubic polynomial that interpolates both function and derivative values at  $\alpha_{i-1}$  and  $\alpha_i$ . This interpolation always yields a unique cubic, whose minimizer is given by the formula at the top. Specifically, the new trial point  $\alpha_{i+1}$  is equal to  $\alpha_i - (\alpha_i - \alpha_{i-1}) \left[ \frac{\phi'(\alpha_i) + d_2 - d_1}{\phi'(\alpha_i) - \phi'(\alpha_{i-1}) + 2d_2} \right]$ .

The quantity  $d_1$  is the sum of the derivatives at  $\alpha_{i-1}$  and  $\alpha_i$ , minus three times the difference in function values divided by the difference in step lengths. The value  $d_2$  incorporates curvature information and includes a square root, which can be complex. In such case, the interpolation is deemed unreliable, and the algorithm discards it, reverting to a safer strategy like halving the previous step length to maintain steady progress. If the interpolation is successful, the process can be repeated by replacing either the data at  $\alpha_{i-1}$  or  $\alpha_i$  with the new function and derivative values at  $\alpha_{i+1}$ , depending on the line search termination conditions. The decision on which of  $\alpha_{i-1}$  and  $\alpha_i$  should be kept and which discarded depends on the specific conditions used to terminate the line search; we discuss this issue next in the context of the Wolfe conditions.

In general, cubic interpolation is a powerful strategy because it accounts for changes in the function's curvature, often leading to a quadratic rate of convergence, which means the algorithm can find an optimal step length more quickly than simpler methods.

$$\alpha_k^{(0)} = \alpha_{k-1} \frac{\nabla f_{k-1}^T p_{k-1}}{\nabla f_k^T p_k}.$$

**Note:** For the first iteration ( $k = 0$ ) set  $\alpha_0^{(0)} = 1$  for Newton methods; for others, use  $\alpha_0^{(0)} = \min\left(1, \frac{c}{\|\nabla f(x_0)\|}\right)$  or a task-dependent value or a task-dependent value.



## Comments

Denote by  $\alpha_k^{(0)}$  the initial approximation of the step length at the  $k$ -th iteration of the linear search. Selecting an effective initial step length is crucial for optimization efficiency. For Newton and quasi-Newton methods,  $\alpha_k^{(0)}$  is set to 1 on every iteration to prioritize unit steps, leveraging their fast convergence when conditions like sufficient decrease are satisfied. For methods with poorly scaled directions, such as steepest descent a popular strategy is to assume that the first-order change in the function at iterate  $x_k$  will be the same as that obtained at the previous step. This causes  $\alpha_k^{(0)}$  to be  $\alpha_{k-1}$  times the ratio of the dot product of the gradient at  $x_{k-1}$  with the previous direction to the dot product at  $x_k$  with the current direction.

For the first iteration ( $k = 0$ ), where prior data is unavailable, Newton and quasi-Newton methods use  $\alpha_0^{(0)} = 1$  to exploit their scaling properties. For poorly scaled methods,  $\alpha_0^{(0)}$  is set to  $\min\left(1, \frac{c}{\|\nabla f(x_0)\|}\right)$ , where  $c$  is a positive constant scale the step to fit the task, or a task-dependent value like a small constant, providing a robust starting point.

## Algorithm 4: Line Search

- ▶ Finds  $\alpha_k$  satisfying strong Wolfe conditions for direction  $p_k$ .
- ▶ Parameters:  $0 < c_1 < c_2 < 1$ ,  $\alpha_{\max} > 0$ , e.g.,  $c_1 = 10^{-4}$ ,  $c_2 = 0.9$ .

**Algorithm 4** (Line Search Algorithm):

```
1: Set:  $\alpha_0 \leftarrow 0$ , choose  $\alpha_{\max} > 0$  and  $\alpha_1 \in (0, \alpha_{\max})$ ,  $i \leftarrow 1$ 
2: repeat
3:   if  $\phi(\alpha_i) > \phi(0) + c_1 \alpha_i \phi'(0)$  or  $(\phi(\alpha_i) \geq \phi(\alpha_{i-1})$  and  $i > 1)$  then
4:      $\alpha_k \leftarrow \text{zoom}(\alpha_{i-1}, \alpha_i)$ 
5:     return  $\alpha_k$ 
6:   end if
7:   if  $|\phi'(\alpha_i)| \leq -c_2 \phi'(0)$  then  $\alpha_k \leftarrow \alpha_i$ 
8:     return  $\alpha_k$ 
9:   end if
10:  if  $\phi'(\alpha_i) \geq 0$  then  $\alpha_k \leftarrow \text{zoom}(\alpha_i, \alpha_{i-1})$ 
11:    return  $\alpha_k$ 
12:  end if
13:  Choose  $\alpha_{i+1} \in (\alpha_i, \alpha_{\max})$ 
14:   $i \leftarrow i + 1$ 
15: until false
```

Newton's Method

Superlinear convergence

Global Convergence

Cholesky Factorization

Interpolation



## Comments

Algorithm 4 searches for a step length  $\alpha_k$  satisfying strong Wolfe conditions. It starts with  $\alpha_0 = 0$  and a trial  $\alpha_1$ , iteratively increasing  $\alpha_i$  while checking sufficient decrease and curvature conditions. If conditions fail, it calls a function called zoom to refine the step within an interval. Parameters  $c_1$  and  $c_2$  (e.g.,  $10^{-4}$ , 0.9) ensure robust step selection.

## Algorithm 5: Zoom

- Refines step length  $\alpha_k \in (\alpha_{lo}, \alpha_{hi})$  satisfying strong Wolfe conditions.
- Parameters:  $0 < c_1 < c_2 < 1$ , e.g.,  $c_1 = 10^{-4}$ ,  $c_2 = 0.9$ .

**Algorithm 5** (Zoom):

```
1: loop
2:   Interpolate to find  $\alpha_j \in (\alpha_{lo}, \alpha_{hi})$ 
3:   if  $\phi(\alpha_j) > \phi(0) + c_1 \alpha_j \phi'(0)$  or  $\phi(\alpha_j) \geq \phi(\alpha_{lo})$  then
4:      $\alpha_{hi} \leftarrow \alpha_j$ 
5:   else
6:     if  $|\phi'(\alpha_j)| \leq -c_2 \phi'(0)$  then  $\alpha_k \leftarrow \alpha_j$ 
7:       return  $\alpha_k$ 
8:     end if
9:     if  $\phi'(\alpha_j)(\alpha_{hi} - \alpha_{lo}) \geq 0$  then  $\alpha_{hi} \leftarrow \alpha_{lo}$ 
10:    end if
11:     $\alpha_{lo} \leftarrow \alpha_j$ 
12:  end if
13: end loop
```

Newton's Method  
Superlinear convergence  
Global Convergence  
Cholesky Factorization  
**Interpolation**



30/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Algorithm 5 refines a step length  $\alpha_k$  within an interval  $(\alpha_{lo}, \alpha_{hi})$  containing steps satisfying strong Wolfe conditions. It interpolates to find a trial step  $\alpha_j$ , evaluates  $\phi$  and  $\phi'$ , and updates the interval endpoints to maintain the smallest function value and derivative conditions. It stops when  $\alpha_j$  satisfies both Wolfe conditions.

As we mentioned earlier, the interpolation step for selecting  $\alpha_j$  must be safeguarded to prevent the new step from being too close to the interval endpoints. Practical line search algorithms leverage properties of interpolating polynomials to make informed predictions about the next step length. Near the solution, function values  $f(x_k)$  and  $f(x_{k-1})$  may become indistinguishable due to finite-precision arithmetic, so the line search should include a stopping criterion if no lower function value is found after a set number of trials (typically ten) or if the change in  $x$  approaches machine precision or a user-defined threshold.

Strong Wolfe conditions with parameters like  $c_1 = 10^{-4}$  and  $c_2 = 0.9$  have similar computational costs to regular Wolfe conditions but offer better control over search quality because by decreasing  $c_2$ , we can bring the assumed value of  $\alpha$  closer to a local minimum, which is crucial for steepest descent and nonlinear conjugate gradient methods.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 3)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Trust-Region  
Methods

Global  
solution

Cauchy Point

Dogleg  
Method

Global  
Convergence

TR  
Subproblem  
Algorithm



Санкт-Петербургский  
государственный  
университет



41 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

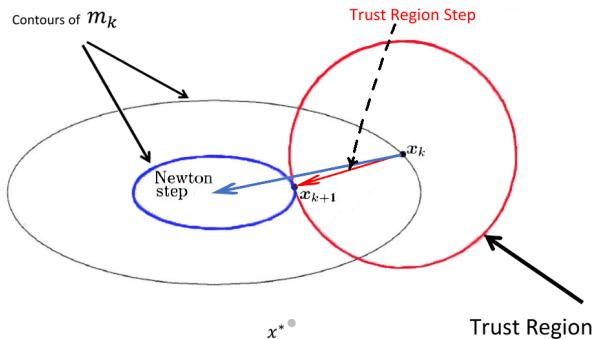
### Comments

In this lecture, we will consider a different approach to finding the minimum: an approach based on the use of a trust region. The idea is that instead of searching for the minimum of the objective function itself, we search for the minimum of a model—a quadratic approximation of the objective function. We will explore the question of the existence of a solution and general fundamental approaches such as the Cauchy point and the dogleg method.

## Trust-Region Methods

- Minimizes quadratic model  $m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$  within trust region  $\|p\| \leq \Delta_k$ .
- Chooses step direction and length together, unlike line search.

### Trust region methods



**Figure:** Trust-region step (red arrow within circle) vs. line search direction (blue arrow) for  $x_k$  to  $x^*$ .

1/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Trust-region methods minimize a quadratic model  $m_k$  within a trust region of radius  $\Delta_k$ , determining the step's direction and length simultaneously. This differs from line search methods, which first choose a direction and then adjust the step length  $\alpha$ . In the figure, the blue arrow represents the Newton step, the unconstrained minimizer of  $m_k$ , which exceeds  $\Delta_k$ .

The red arrow shows the Trust-Region Step, the minimizer of  $m_k$  within the trust region, staying inside the red circle. These steps are distinct: the Newton step ignores the trust region constraint, while the Trust-Region Step adheres to it. The size of  $\Delta_k$  is key—too small slows progress toward  $x^*$ , too large risks model inaccuracy, prompting adjustments based on the step's success.

Basically, the core concept of trust-region methods lies in the fact that the quadratic model approximating the objective function within the trust region does not diverge significantly from the actual function, thereby allowing the iterative step to closely approach the true minimum of the function within that region.



- Quadratic model  $m_k$  at  $x_k$  based on Taylor expansion:

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla^2 f(x_k + tp)p, \text{ where } f_k = f(x_k), g_k = \nabla f(x_k), t \in (0, 1)$$

- Approximated model using  $B_k \approx \nabla^2 f(x_k)$ , symmetric:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$$

- Subproblem: Minimize  $m_k$  within trust region:

$$\min_{p \in \mathbb{R}^n} m_k(p) \quad \text{s.t. } \|p\| \leq \Delta_k \quad (3a)$$

**Note:** Error  $m_k(p) - f(x_k + p) = O(\|p\|^2)$ , or  $O(\|p\|^3)$  if  $B_k = \nabla^2 f(x_k)$ . If  $B_k$  positive definite and  $\|B_k^{-1} g_k\| \leq \Delta_k$ , solution is  $p_k^B = -B_k^{-1} g_k$ . In this case, we call  $p_k^B$  the full step.

## Comments

Consider the Taylor expansion of the objective function around the point  $x_k$  up to the second-order term included. Let  $f_k$  be the value of the function  $f$  at the point  $x_k$ , and let  $g_k$  be the gradient of  $f$  at that point. The expansion takes the form  $f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla^2 f(x_k + tp)p$ , where  $t$  is a scalar between zero and one.

Approximating the Hessian at  $x_k + tp$  with a symmetric matrix  $B_k$ , the quadratic model is defined as  $m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$ . The error between  $m_k(p)$  and  $f(x_k + p)$  is of the order of the square of the norm of  $p$ , or of the order of the cube of the norm of  $p$  if  $B_k = \nabla^2 f(x_k)$ .

The trust-region subproblem minimizes  $m_k(p)$  subject to the norm of  $p$  less than or equal to  $\Delta_k$ , using the Euclidean norm. When  $B_k$  is positive definite and the norm of the inverse of  $B_k$  times  $g_k$  is less than or equal to  $\Delta_k$ , the unconstrained minimizer  $p_k^B = -B_k^{-1} g_k$  becomes the solution, known as the full step. Otherwise, an approximate solution ensures convergence.



**Strategy:** Choose trust-region radius  $\Delta_k$  based on agreement between model  $m_k$  and objective  $f$ , using ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)},$$

where the numerator is called the actual reduction, and the denominator is the predicted reduction.

**Principle:**

- ▶ Predicted reduction is always nonnegative since  $p_k$  minimizes  $m_k$ .
- ▶ If  $\rho_k \approx 1$ , model  $m_k$  aligns well with  $f$ , so expand  $\Delta_k$ .
- ▶ If  $\rho_k$  is small or negative, shrink  $\Delta_k$  to improve model accuracy.

### Comments

Consider the strategy for selecting the trust-region radius at each iteration, denoted as  $\Delta_k$ , which depends on the agreement between the model function  $m_k$  and the objective function  $f$  from previous steps. Define the ratio  $\rho_k$  as the actual reduction in the function value divided by the reduction predicted by the model.

The actual reduction is the difference between the function value at  $x_k$  and at  $x_k + p_k$ , while the predicted reduction is the difference between  $m_k(0)$  and  $m_k(p_k)$ . Since  $p_k$  minimizes  $m_k$  over a region including zero, the predicted reduction is always nonnegative. When  $\rho_k$  is close to one, the model  $m_k$  aligns well with the function  $f$ , indicating that the model is a good approximation, so the radius  $\Delta_k$  can be safely expanded for the next iteration to allow for larger steps. Conversely, if  $\rho_k$  is small or negative, the model does not accurately represent the function, so the radius  $\Delta_k$  should be reduced to improve the model's accuracy and ensure better progress.

## Trust-Region Algorithm

### Algorithm 6 (Trust Region):

```
Require:  $\hat{\Delta} > 0$ ,  $\Delta_0 \in (0, \hat{\Delta})$ , and  $\eta \in [0, \frac{1}{4}]$ 
1: for  $k = 0, 1, 2, \dots$  do
2:   Obtain  $p_k$  by (approximately) solving the subproblem, evaluate  $\rho_k$ .
3:   if  $\rho_k < \frac{1}{4}$  then  $\Delta_{k+1} = \frac{1}{4}\Delta_k$ 
4:   else
5:     if  $\rho_k > \frac{3}{4}$  and  $\|p_k\| = \Delta_k$  then  $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$ 
6:     else
7:        $\Delta_{k+1} = \Delta_k$ 
8:     end if
9:   end if
10:  if  $\rho_k > \eta$  then  $x_{k+1} = x_k + p_k$ 
11:  else
12:     $x_{k+1} = x_k$ 
13:  end if
14: end for
```

Note:  $\hat{\Delta}$  is the upper bound on step length. Radius increases only if step reaches region boundary; otherwise,  $\Delta_k$  remains unchanged.

4/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Consider the process of iteratively adjusting the trust-region radius to balance the accuracy of the model with the progress of the optimization. The algorithm evaluates how well the model predicts the function's behavior by comparing actual and predicted reductions, using this to decide whether to accept a step or adjust the radius. If the model's prediction is poor, the radius shrinks to ensure the model becomes more accurate locally, preventing steps that might worsen the objective function. If the model performs well and the step fully utilizes the current radius, the radius expands to allow larger steps, accelerating convergence. Steps are accepted only when they provide sufficient improvement, ensuring steady progress toward the minimum.

**Theorem 12 (Moré-Sorensen)**

The vector  $p^*$  is a global solution of the trust-region subproblem

$$\min_{p \in \mathbb{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p \quad \text{s.t. } \|p\| \leq \Delta,$$

if and only if  $p^*$  is feasible and there exists a scalar  $\lambda \geq 0$  such that:

►  $(B + \lambda I)p^* = -g,$  (4a)

►  $\lambda(\Delta - \|p^*\|) = 0,$  (4b)

►  $B + \lambda I$  is positive semidefinite. (4c)

**Comments**

To make the algorithm we discussed practical, we need to address the subproblem of finding the global minimum of the model  $m$  within the trust region defined by  $\Delta$ . The following theorem provides the necessary and sufficient conditions for a given vector  $p^*$  to be the global minimizer of this subproblem. To achieve this, an auxiliary parameter  $\lambda \geq 0$  is introduced. The theorem establishes that  $p^*$  is the global minimizer if and only if it satisfies a specific system of equations involving  $\lambda$ . This approach not only guarantees the optimality of the solution but also offers a computationally efficient method to determine  $p^*$ , thereby making the trust-region framework both theoretically robust and practically applicable.

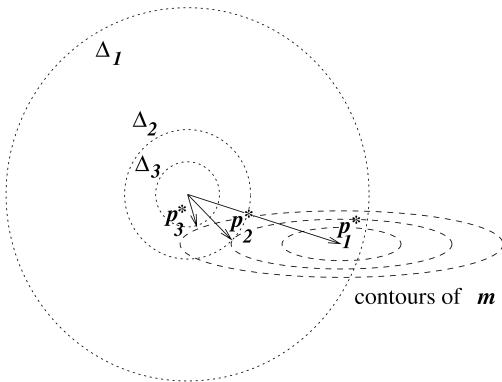


Figure: Solution of trust-region subproblem for different radii  $\Delta_1, \Delta_2, \Delta_3$ .

<b>Trust-Region Methods</b>
<b>Global solution</b>
<b>Cauchy Point</b>
<b>Dogleg Method</b>
<b>Global Convergence</b>
<b>TR Subproblem Algorithm</b>



## Key Features:

- ▶ Complementarity condition:  $\lambda(\Delta - \|p^*\|) = 0$ , where  $\lambda \geq 0$ .
- ▶ For  $\Delta = \Delta_1$ :  $p_1^*$  inside trust region,  $\lambda = 0$ ,  $Bp^* = -g$ .
- ▶ For  $\Delta = \Delta_2, \Delta_3$ :  $p_2^*, p_3^*$  on boundary,  $\lambda > 0$ ,  $p^*$  collinear with  $-\nabla m(p^*)$ .

6/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The trust-region subproblem focuses on minimizing a quadratic model subject to the constraint that the norm of the solution step is less than or equal to the radius  $\Delta$ . The image shows solutions, labeled as  $p_1^*$ ,  $p_2^*$ , and  $p_3^*$ , for three different radii, called  $\Delta_1$ ,  $\Delta_2$ , and  $\Delta_3$ .

The complementarity condition states that the product of a scalar parameter  $\lambda$  and the difference between the radius and the norm of the solution step must equal zero, meaning either  $\lambda$  is zero or the norm of the solution step equals the radius.

For a large radius,  $\Delta_1$ , the first optimal step lies strictly inside the trust region, with its norm less than  $\Delta_1$ , so  $\lambda$  is zero, and the optimality condition—where the matrix  $B$  times the solution step equals the negative gradient—implies the first optimal step is the unconstrained minimizer, provided  $B$  allows it.

For smaller radii,  $\Delta_2$  and  $\Delta_3$ , the second and third optimal steps lie on the boundaries, with their norms equal to  $\Delta_2$  and  $\Delta_3$ , respectively, so  $\lambda$  is allowed to take a positive value. Here, the condition that  $\lambda$  times the solution step equals the negative of  $B$  times the solution step minus the gradient, which is also the negative gradient of the model at the solution step, implies that the solution step is collinear with the negative gradient of the model and normal to its contours, as seen in the alignment of the second and third optimal steps with the gradient direction in the figure.

## Lemma 2

Let  $m(p) = g^T p + \frac{1}{2} p^T B p$ , where  $B$  is any symmetric matrix. Then the following statements are true:

- (i)  $m$  attains a minimum if and only if  $B$  is positive semidefinite and  $g$  is in the range of  $B$ .  
If  $B$  is positive semidefinite, then every  $p$  satisfying  $Bp = -g$  is a global minimizer of  $m$ .
- (ii)  $m$  has a unique minimizer if and only if  $B$  is positive definite.

**Proof:** (i) We start by proving the “if” part. Let  $B$  be positive semidefinite, then, since  $g$  is in the range of  $B$ , there is a  $p$  with  $Bp = -g$ . For all  $w \in \mathbb{R}^n$ , we have:

$$\begin{aligned} m(p + w) &= g^T(p + w) + \frac{1}{2}(p + w)^T B(p + w) \\ &= \left(g^T p + \frac{1}{2} p^T B p\right) + g^T w + (Bp)^T w + \frac{1}{2} w^T B w = \end{aligned}$$

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Now let's move on to the proof of the Theorem 12, which provides the conditions for identifying the exact solution to the trust-region subproblem, a critical step in optimizing a function within a constrained region. The proof builds on Lemma 2, which addresses the minimization of a quadratic function without constraints. The lemma establishes that a minimum exists when the matrix representing the function's curvature is positive semidefinite and the gradient lies within its range, allowing multiple solutions satisfying the gradient equation to be global minimizers. If the matrix is positive definite, the minimum is unique, ensuring a single optimal point. This foundation is essential for understanding how the trust-region constraint modifies the solution process.

We prove each of the claims in turn. We start by proving the “if” direction of part (i). Let  $B$  be positive semidefinite, then, since  $g$  is in the range of  $B$ , there is a  $p$  such that  $Bp = -g$ . For all  $w \in \mathbb{R}^n$ , we have the value of  $m$  at  $p + w$  equals the inner product of  $g$  with  $p + w$  plus one-half times the inner product of  $p + w$  with  $B$  times  $p + w$ , which expands to the inner product of  $g$  with  $p$  plus one-half times the inner product of  $p$  with  $B$  times  $p$  plus the inner product of  $g$  with  $w$  plus the inner product of  $Bp$  with  $w$  plus one-half times the inner product of  $w$  with  $B$  times  $w$  is equal to

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



$$= m(p) + g^T w - g^T w + \frac{1}{2} w^T B w = m(p) + \frac{1}{2} w^T B w.$$

As  $B$  is positive semidefinite,  $w^T B w \geq 0$ , so  $m(p + w) \geq m(p)$ , proving  $p$  is a global minimizer.

- (i) For the “only if” direction we have If  $p$  minimizes  $m$ , then  $\nabla m(p) = Bp + g = 0$ , so  $g$  is in  $B$ ’s range. The Hessian  $\nabla^2 m(p) = B$  is positive semidefinite (by Theorem 3 (Second-Order Necessary Condition)).
- (ii) For the “if” part, the same argument as in (i) suffices with the additional point that  $w^T B w > 0$  whenever  $w \neq 0$ .
- (iii) For the “only if” part, we proceed as in (i) to deduce that  $B$  is positive definite. If  $B$  is not positive definite, there is a vector  $w \neq 0$  such that  $Bw = 0$ . Hence, we have  $m(p + w) = m(p)$ , so the minimizer is not unique, giving a contradiction.  $\square$

### Comments

Continuing the proof, we simplify the expression to show that  $f(p + w) = f(p) + \frac{1}{2} w^T B w$ . Since  $B$  is positive semidefinite, this additional term is non-negative, ensuring  $f(p + w) \geq f(p)$ , which confirms  $p$  as a global minimizer.

For the "only if" direction: if  $p$  is a minimizer, the gradient at  $p$ ,  $\nabla f(p) = Bp + g$ , must be zero, meaning  $g \in \text{col}(B)$ , and the Hessian  $\nabla^2 f(p) = B$  must be positive semidefinite.

For the second claim: if  $B$  is positive definite, then  $w^T B w > 0$  for all  $w \neq 0$ , so  $f(p+w) > f(p)$  for any perturbation, ensuring uniqueness of the minimizer. Conversely, if  $B$  is positive semidefinite but not positive definite, there exists some  $w \neq 0$  such that  $Bw = 0$ , leading to  $f(p+w) = f(p)$ , which implies multiple minimizers and contradicts uniqueness.

## Example for Lemma 2

Consider  $B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ , with eigenvalues 2, 0, 3, so  $B$  is positive semidefinite but singular. Let  $m(p) = g^T p + \frac{1}{2} p^T B p$ .

**Case 1:**  $g = (4, 0, -6)^T$ , second component zero, so  $g$  is in  $B$ 's column space. Solve  $Bp = -g$ :

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} -4 \\ 0 \\ 6 \end{bmatrix} \implies p_1 = -2, p_3 = 2, p_2 \text{ arbitrary.}$$

Thus,  $p = (-2, p_2, 2)^T$  minimizes  $m$ . For  $p = (-2, 0, 2)^T$ ,  $m(p) = -8 - 12 + 10 = -10$ .

**Case 2:**  $g = (4, 1, -6)^T$ , second component nonzero, so  $g$  is not in  $B$ 's column space. Along direction  $d = (0, -1, 0)^T$ , compute the directional derivative:

$\nabla m(p)^T d = g^T d = 1(-1) = -1 < 0$ , and  $d^T B d = 0$ , so

$m(p + td) = m(p) + t(-1) \rightarrow -\infty$  as  $t \rightarrow \infty$ , meaning no minimum exists.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

The example illustrates Lemma 2 by showing how the existence of a minimum for a quadratic function depends on the gradient's position relative to the matrix's column space. In the first case, the gradient has a zero second component, so it lies in the matrix's column space, allowing a solution to the gradient equation and yielding a minimum, as the function value remains constant along certain directions. In the second case, the gradient's second component is nonzero, placing it outside the matrix's column space, so the function decreases indefinitely along a specific direction where the matrix has no effect, demonstrating the absence of a minimum.

## Proof of Theorem 12

**Proof of Theorem 12:** Assume  $\lambda \geq 0$  satisfies conditions (4, i.e. (4a-4c)). By Lemma 2,  $p^*$  is a global minimizer of the modified quadratic:

$$\hat{m}(p) = g^T p + \frac{1}{2} p^T (B + \lambda I)p = m(p) + \frac{\lambda}{2} p^T p.$$

Since  $\hat{m}(p) \geq \hat{m}(p^*)$ , we have:

$$m(p) \geq m(p^*) + \frac{\lambda}{2} ((p^*)^T p^* - p^T p).$$

Given  $\lambda(\Delta - \|p^*\|) = 0$ , so  $\lambda(\Delta^2 - (p^*)^T p^*) = 0$ , this becomes:

$$m(p) \geq m(p^*) + \frac{\lambda}{2} (\Delta^2 - p^T p).$$

Since  $\lambda \geq 0$ ,  $m(p) \geq m(p^*)$  for all  $p$  with  $\|p\| \leq \Delta$ , so  $p^*$  is a global minimizer of (3b).

For the converse, assume  $p^*$  is a global solution of (3b) and show that there is a  $\lambda \geq 0$  that satisfies (4). If  $\|p^*\| < \Delta$ , then  $p^*$  is an unconstrained minimizer of  $m$ , so the gradient  $Bp^* + g = 0$ , and the Hessian  $B$  is positive semidefinite, satisfying (4) with  $\lambda = 0$ .

10/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

We begin by assuming there exists a non-negative scalar  $\lambda$  satisfying the given conditions. Using Lemma 2, we show that the candidate solution is a global minimizer of a modified quadratic function, which includes an additional term involving  $\lambda$  times the squared norm of the step. This implies the original function's value at any point is at least the value at the candidate plus  $\lambda$  times half the difference of the squared norms of the candidate and the point. The complementarity condition allows us to adjust this inequality, and since  $\lambda$  is non-negative, we conclude the candidate minimizes the original function within the trust region.

For the converse, if the candidate is a global minimizer and its norm is less than the trust region radius, it must be an unconstrained minimizer, meaning the gradient at that point is zero and the Hessian is positive semidefinite, satisfying the conditions with  $\lambda = 0$ .

## Proof of Theorem 12 (Continued)

Assume for the remainder of the proof that  $\|p^*\| = \Delta$ . Then condition (4b) holds, and  $p^*$  solves the constrained problem: minimize  $m(p)$  subject to  $\|p\| = \Delta$ . Using constrained optimization conditions (we'll discuss it later), there exists  $\lambda$  such that the Lagrangian:

$$\mathcal{L}(p, \lambda) = m(p) + \frac{\lambda}{2}(p^T p - \Delta^2)$$

is stationary at  $p^*$ . Setting the gradient of the Lagrangian with respect to  $p$  to zero we obtain:

$$Bp^* + g + \lambda p^* = 0 \implies (B + \lambda I)p^* = -g,$$

so condition (4a) holds. Since  $m(p) \geq m(p^*)$  for any  $p$  with  $p^T p = (p^*)^T p^* = \Delta^2$ , we have:

$$m(p) \geq m(p^*) + \frac{\lambda}{2} ((p^*)^T p^* - p^T p).$$

Substituting  $g$  from the previous equation, we get after rearrangement:

$$\frac{1}{2}(p - p^*)^T (B + \lambda I)(p - p^*) \geq 0.$$

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Assuming the norm of the candidate solution equals the trust region radius, the complementarity condition is immediately satisfied, and the candidate solves a constrained minimization problem where the norm of the step equals the radius. We introduce a Lagrangian function, which combines the original function with a penalty term involving the squared norm of the step minus the squared radius, scaled by half of  $\lambda$ .

Setting the gradient of this Lagrangian with respect to the step to zero, we derive that the matrix  $B$  times the candidate plus the gradient plus  $\lambda$  times the candidate equals zero, which rearranges to show the candidate satisfies the modified linear system, confirming one of the theorem's conditions. Since the function value at any point on the boundary is at least the value at the candidate, we adjust this inequality using the Lagrangian multiplier, and after substituting the gradient expression, we obtain a quadratic form involving the modified matrix.

## Proof of Theorem 12 (Completed)

The set of directions  $w = \pm \frac{p - p^*}{\|p - p^*\|}$ , for  $\|p\| = \Delta$ , is dense on the unit sphere suffices to prove

$(B + \lambda I)$  is positive semidefinite.

To show  $\lambda \geq 0$ : Since:  $(B + \lambda I)p^* = -g$ , and  $(B + \lambda I)$  is positive semidefinite, Lemma 2 implies  $p^*$  minimizes:

$$\hat{m}(p) = g^T p + \frac{1}{2} p^T (B + \lambda I) p.$$

Suppose only negative  $\lambda$  satisfy the conditions. Then:

$$m(p) \geq m(p^*) + \frac{\lambda}{2} ((p^*)^T p^* - p^T p)$$

holds for  $\|p\| \geq \Delta = \|p^*\|$ . Since  $p^*$  minimizes  $m$  for  $\|p\| \leq \Delta$ ,  $p^*$  is an unconstrained minimizer. By Lemma 2(i),  $Bp^* = -g$  and  $B$  is positive semidefinite, so:

$$Bp^* + g = 0,$$

and  $(B + 0 \cdot I)$  is positive semidefinite, satisfying the conditions with  $\lambda = 0$ , contradicting negative  $\lambda$ . Thus,  $\lambda \geq 0$ .  $\square$

12/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Since the set of directions  $w$  is dense on the unit sphere, fulfilling the previous inequality suffices to prove the modified matrix is positive semidefiniteness . To prove  $\lambda$  is non-negative, we take advantage of the fact that the modified matrix satisfies the conditions of Lemma 2 and, therefore, by Lemma 2 the candidate minimizes a modified quadratic. Assuming only negative  $\lambda$  values work, the inequality implies the original function's value at points outside the trust region is at least the value at the candidate, and since the candidate minimizes within the region, it becomes an unconstrained minimizer. This leads to the gradient equation and positive semidefiniteness with  $\lambda = 0$ , contradicting the negative  $\lambda$  assumption, so  $\lambda$  must be non-negative. The theorem is proved.

## Trust-Region Methods: The Cauchy Point

Recall that we are looking for the optimal solution to the subproblem:

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \quad \text{s.t. } \|p\| \leq \Delta_k$$

**Key Idea** Global convergence of trust-region methods does not require exact minimization of the subproblem. It suffices to compute an approximate solution  $p_k$  inside the trust region that yields a sufficient reduction of the model.

### Cauchy Point Algorithm:

Step 1. Solve the linearized subproblem:

$$p_k^s = \arg \min_{p \in \mathbb{R}^n} f_k + g_k^T p \quad \text{s.t. } \|p\| \leq \Delta_k$$

Step 2. Find scalar  $\tau_k > 0$  that minimizes  $m_k(\tau p_k^s)$  subject to  $\|\tau p_k^s\| \leq \Delta_k$ :

$$\tau_k = \arg \min_{\tau \geq 0} m_k(\tau p_k^s) \quad \text{s.t. } \|\tau p_k^s\| \leq \Delta_k$$

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



13/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Although the trust-region subproblem is defined as a quadratic minimization, exact minimization is not required for global convergence. It is sufficient to compute an approximate solution that lies within the trust region and ensures a sufficient decrease in the model function. Such an approximation is provided by the so-called Cauchy point.

The first step of the algorithm is to solve a linearized minimization problem. Specifically, we minimize a linear model consisting of the current function value plus the directional derivative in the direction of the step, under the constraint that the step lies within the trust region, meaning its norm does not exceed the current trust-region radius. The solution to this constrained linear problem is referred to as the steepest descent step.

In the second step, we determine a positive scalar that minimizes the quadratic model along the direction of the steepest descent step, again under the constraint that the resulting scaled step remains within the trust region. This reduces to a one-dimensional optimization problem over the step length along that direction.

In the third step, we define the Cauchy point as the product of the computed scalar and the steepest descent direction. This point lies within the trust region and ensures sufficient decrease of the model function, which is essential for establishing convergence of the algorithm.

Trust-Region Methods

Global solution

**Cauchy Point**

Dogleg Method

Global Convergence

TR Subproblem Algorithm



Step 3. Define the Cauchy point:

$$p_k^c = \tau_k p_k^s$$

Step 4. The solution of the trust-region subproblem is:

$$p_k^s = -\frac{\Delta_k}{\|g_k\|} g_k$$

Step 5. For computing  $\tau_k$ , consider the two cases:

$$\tau_k = \begin{cases} 1 & \text{if } g_k^T B_k g_k \leq 0; \\ \min\left(\frac{\|g_k\|^3}{\Delta_k g_k^T B_k g_k}, 1\right) & \text{otherwise.} \end{cases}$$

Step 6. The Cauchy point is then:

$$p_k^c = -\tau_k \frac{\Delta_k}{\|g_k\|} g_k$$

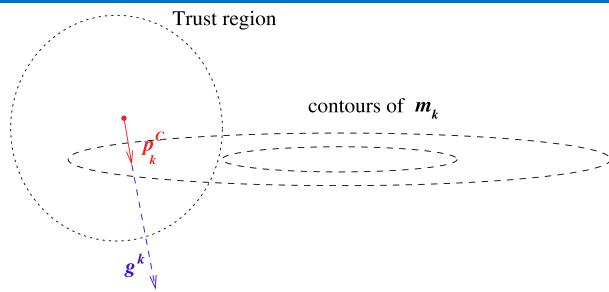
## Comments

It is easy to write down a closed-form definition of the Cauchy point. For a start, the solution of the trust-region subproblem is simply:  $p_k^s = -\frac{\Delta_k}{\|g_k\|} g_k$ .

To obtain the scalar parameter  $\tau_k$  explicitly, we consider two cases. In the first case, when  $g_k^T B_k g_k \leq 0$ , the function  $m_k$  evaluated at  $\tau$  times the search direction decreases monotonically with  $\tau$ , provided the gradient is nonzero. Therefore,  $\tau_k$  is simply 1, the largest value that satisfies the trust-region constraint.

In the second case, when  $g_k^T B_k g_k > 0$ , the function  $m_k(\tau p_k^s)$  is a convex quadratic in  $\tau$ . In that case,  $\tau_k$  is either the unconstrained minimizer of the quadratic, given by  $\frac{\|g_k\|^3}{\Delta_k g_k^T B_k g_k}$ , or the boundary value 1, whichever is smaller.

In summary, the Cauchy point is equal to  $-\tau_k \frac{\Delta_k}{\|g_k\|} g_k$ .



**Figure:** The Cauchy point  $p_k^c$  lies strictly inside the trust region.  $B_k$  is positive definite.

**Key Insight:** The Cauchy point  $p_k^c$  plays a central role in ensuring global convergence of trust-region methods.

**Global Convergence Criterion:** If the actual step  $p_k$  satisfies

$$m_k(0) - m_k(p_k) \geq \gamma(m_k(0) - m_k(p_k^c)) \quad \text{for some fixed } \gamma > 0,$$

then global convergence is guaranteed.

The Cauchy step is inexpensive to compute: it avoids matrix factorizations.

15/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



## Comments

The Cauchy point, denoted  $p_k^c$ , provides a baseline measure of progress for trust-region methods. It is defined without requiring any matrix factorizations, making it computationally inexpensive and robust. In the context of this figure, we consider the case when the matrix  $B_k$  is positive definite. This ensures that the Cauchy point lies strictly inside the trust region, that is,  $\|p_k^c\| < \Delta_k$ .

Crucially, the Cauchy step plays a pivotal role in verifying whether an approximate solution  $p_k$  to the trust-region subproblem is acceptable. Specifically, trust-region algorithms ensure global convergence by requiring that the actual step  $p_k$  achieves at least a constant fraction, denoted  $\gamma > 0$ , of the model decrease attained by the Cauchy point. This criterion enables global convergence without solving the subproblem exactly, emphasizing the importance of the Cauchy point in both theory and practice.

Why not always use the Cauchy point?

- It ensures global convergence and is inexpensive to compute.
- However, it corresponds to the steepest descent method, which may converge slowly even with optimal step lengths.

Limitations:

- The Cauchy point is weakly dependent on  $B_k$ , which affects only the step length.
- Rapid convergence requires that  $B_k$  influence both direction and length, and that it reflect true curvature information.

Improvement strategies:

- Try to improve on the Cauchy point.
- Use  $p_k^B = -B_k^{-1}g_k$  if  $B_k \succ 0$  and  $\|p_k^B\| \leq \Delta_k$ .
- This yields superlinear convergence if  $B_k = \nabla^2 f(x_k)$  or quasi-Newton.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Although the Cauchy point guarantees sufficient decrease in the model and is computationally cheap, always using it is equivalent to applying the steepest descent method with a certain step length. As we noted in the previous lecture, steepest descent converges slowly, even when optimal step lengths are used.

The Cauchy point depends only weakly on the matrix  $B_k$ , which only influences the step length. For rapid convergence, we need  $B_k$  to influence the step direction and to contain accurate curvature information.

Hence, many trust-region methods first compute the Cauchy point and then try to improve upon it. A common strategy is to take the full step  $p_k^B = -B_k^{-1}g_k$  whenever  $B_k$  is positive definite and the norm of this step is less than or equal to  $\Delta_k$ . If  $B_k$  is the true Hessian, or a good quasi-Newton approximation, this strategy typically leads to superlinear convergence.

Recall the trust-region subproblem:

$$\min_{p \in \mathbb{R}^n} m(p) \stackrel{\text{def}}{=} f + g^T p + \frac{1}{2} p^T B p \quad \text{s.t. } \|p\| \leq \Delta$$

Notation:

- Focus on one iteration: drop subscript  $k$ .
- Denote solution by  $p^*(\Delta)$ , emphasizing dependence on  $\Delta$ .

Dogleg method:

- Applicable when  $B$  is positive definite.
- If  $\Delta \geq \|p^B\|$ , then  $p^*(\Delta) = p^B$ , where  $p^B = -B^{-1}g$ .
- If  $\Delta$  is small, quadratic term in  $m$  has little effect:  

$$p^*(\Delta) \approx -\Delta \cdot \frac{g}{\|g\|}.$$
- For intermediate  $\Delta$ , path of  $p^*(\Delta)$  is curved.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

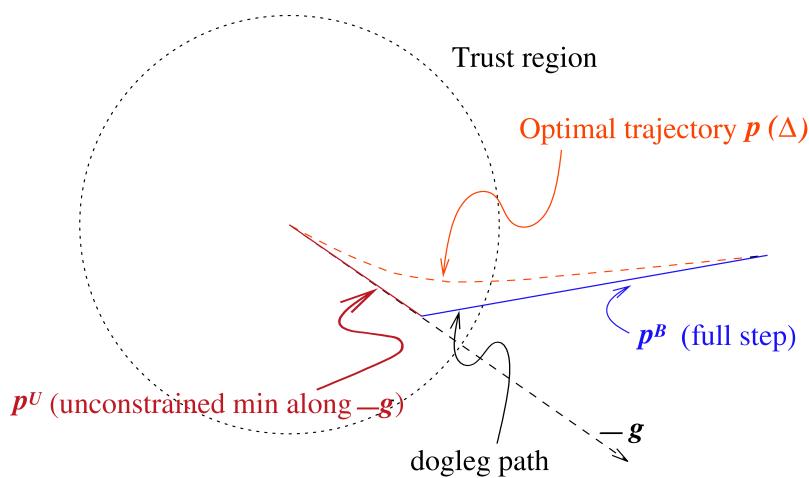
Global Convergence

TR Subproblem Algorithm



## Comments

We begin by introducing the dogleg method — the first of three approaches to approximately solving the trust-region subproblem using curvature information. This method assumes that the symmetric matrix in the model is positive definite. To focus on a single iteration, we drop the iteration subscript and simplify the subproblem: the task is to minimize a quadratic model consisting of the function value, the inner product of the gradient and the step, and a quadratic term, subject to a bound on the step norm. If the trust-region radius exceeds the length of the unconstrained minimizer, then that minimizer becomes the solution. When the radius is very small, the optimal step approximates a scaled version of the negative gradient direction. In between, the solution follows a smooth trajectory between these two extremes, forming the characteristic path that gives the dogleg method its name.



Trust-Region Methods

Global solution

**Cauchy Point**

Dogleg Method

Global Convergence

TR Subproblem Algorithm



**Figure:** The dogleg method constructs a piecewise linear path from the origin to the Cauchy point, then toward the full Newton step.

## Comments

This figure illustrates the geometry of the dogleg method, which is used in trust-region frameworks when the model matrix is positive definite. The method constructs a piecewise linear path starting from the origin in the direction of the negative gradient, leading to the so-called Cauchy point — the minimizer of the model along the steepest descent direction under the trust-region constraint. From there, the path continues toward the full Newton step, which minimizes the quadratic model without constraints. The trust-region boundary intersects this path, and the solution lies somewhere along it. This construction allows the method to account for both the linear and quadratic components of the model effectively.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



Steepest descent point:

$$p^U = -\frac{g^T g}{g^T B g} g$$

Dogleg path: for  $\tau \in [0, 2]$  (and  $p^B = -B^{-1}g$ )

$$\tilde{p}(\tau) = \begin{cases} \tau p^U, & 0 \leq \tau \leq 1, \\ p^U + (\tau - 1)(p^B - p^U), & 1 \leq \tau \leq 2. \end{cases}$$

The dogleg method minimizes the model  $m$  over this trajectory, subject to the trust-region constraint.

## Comments

So, as we were saying, the dogleg method constructs an approximate solution to the trust-region subproblem using a piecewise linear trajectory. Instead of following the true curved path of the solution, it builds a path consisting of two straight-line segments. The first segment goes from the origin in the direction of steepest descent, ending at the point that minimizes the model along this direction. This point is calculated by taking the gradient, computing its squared norm, and dividing by the curvature along the gradient direction. The second segment continues from this steepest descent point toward the full Newton step, which is the unconstrained minimizer of the quadratic model.

The entire path is parametrized by a scalar variable ranging from zero to two. For parameter values between zero and one, the point lies along the steepest descent segment. For values between one and two, the point is on the segment connecting the steepest descent step to the Newton step.

The dogleg method selects the point along this path that minimizes the quadratic model, while also remaining within the trust region, which means its norm cannot exceed the current trust-region radius. In other words, the value of the parameter  $\tau$  is chosen so that the norm of the vector  $\tilde{p}(\tau)$  is equal to  $\Delta$ .

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm

**Lemma 3**

Let  $B$  be positive definite. Then

- (i)  $\|\tilde{p}(\tau)\|$  is an increasing function of  $\tau$ , and
- (ii)  $m(\tilde{p}(\tau))$  is a decreasing function of  $\tau$ .

**Proof:**

It is easy to show that (i) and (ii) both hold for  $\tau \in [0, 1]$ , so we restrict our attention to the case of  $\tau \in [1, 2]$ . For (i), define  $h(\alpha)$  by

$$\begin{aligned} h(\alpha) &= \frac{1}{2} \|\tilde{p}(1 + \alpha)\|^2 \\ &= \frac{1}{2} \|p^U + \alpha(p^B - p^U)\|^2 \\ &= \frac{1}{2} \|p^U\|^2 + \alpha(p^U)^T(p^B - p^U) + \frac{1}{2} \alpha^2 \|p^B - p^U\|^2. \end{aligned}$$

Our result is proved if we can show that  $h'(\alpha) \geq 0$  for  $\alpha \in (0, 1)$ .

20/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Comments**

The following lemma shows that the minimum along the dogleg path can be found easily.

There are two key monotonicity properties of the dogleg path. First, the norm of  $\tilde{p}(\tau)$  grows as  $\tau$  increases, and second, the model function decreases along this path.

Let us prove this. Notice that both statements are straightforward when  $\tau$  is between zero and one, because in this case the step simply moves in the steepest descent direction. Therefore, we only need to consider  $\tau$  between one and two.

To handle this case, define the function  $h(\alpha)$  as one half the squared norm of  $\tilde{p}(1+\alpha)$ . Substituting the definition of the dogleg step, we obtain  $h(\alpha) = \frac{1}{2} \|p^U + \alpha(p^B - p^U)\|^2$ . Expanding gives a quadratic function in  $\alpha$ .

If the derivative of this function,  $h'(\alpha)$ , is nonnegative for  $\alpha$  in the interval zero to one, then the squared norm, and therefore the norm itself, is increasing. This proves part (i).

### Proof of Lemma 3

Now, compute the derivative  $h'(\alpha)$ :

$$h'(\alpha) = -(p^U)^T(p^U - p^B) + \alpha\|p^U - p^B\|^2 \geq -(p^U)^T(p^U - p^B) = \\ \frac{g^T g}{g^T B g} g^T \left( -\frac{g^T g}{g^T B g} g + B^{-1} g \right) = g^T g \frac{g^T B^{-1} g}{g^T B g} \left[ 1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} \right] \geq 0,$$

where the final inequality follows from the Cauchy-Schwarz inequality.

For (ii), we define  $\hat{h}(\alpha) = m(\tilde{p}(1 + \alpha)) =$

$$= f + g^T(p^U + \alpha(p^B - p^U)) + \frac{1}{2}(p^U + \alpha(p^B - p^U))^T B(p^U + \alpha(p^B - p^U))$$

and show that  $\hat{h}'(\alpha) \leq 0$  for  $\alpha \in (0, 1)$ :

$$\begin{aligned} \hat{h}'(\alpha) &= (p^B - p^U)^T(g + Bp^U) + \alpha(p^B - p^U)^T B(p^B - p^U) \\ &\leq (p^B - p^U)^T(g + Bp^U + B(p^B - p^U)) = (p^B - p^U)^T(g + Bp^B) = 0, \end{aligned}$$

yielding the result.  $\square$

21/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

We now compute the derivative of  $h(\alpha)$ . Expanding gives  $h'(\alpha) = -(p^U)^T(p^U - p^B) + \alpha\|p^U - p^B\|^2$ . This is bounded below by the first term.

Substituting the definitions of  $p^U$  and  $p^B$ , we rewrite the expression in terms of the gradient  $g$  and the matrix  $B$ . After simplification, we obtain  $\frac{g^T g}{g^T B g} g^T \left[ 1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} \right]$ . By the Cauchy-Schwarz inequality, this last factor is nonnegative. Therefore,  $h'(\alpha) \geq 0$ , which proves monotonicity of the norm. Thus the part (i) is proved.

For part (ii), we define  $\hat{h}(\alpha) = m(\tilde{p}(1 + \alpha))$ . Expanding gives a quadratic expression in  $\alpha$ . Its derivative is  $\hat{h}'(\alpha) = (p^B - p^U)^T(g + Bp^U) + \alpha(p^B - p^U)^T B(p^B - p^U)$ . Using the positive definiteness of the Hessian and the fact that  $\alpha$  is at most 1, we bound this term above by the expression with  $g + Bp^B$ , which equals zero because  $p^B$  is the minimizer of the quadratic model. Hence,  $\hat{h}'(\alpha) \leq 0$ , and the model function decreases as  $\tau$  increases. The Lemma is proved.

From Lemma 3, the path  $\tilde{p}(\tau)$  intersects the trust-region boundary  $\|p\| = \Delta$  at one point if  $\|p^B\| \geq \Delta$ , and nowhere otherwise. Since  $m$  decreases along the path, choose:

$$p = \begin{cases} p^B, & \text{if } \|p^B\| \leq \Delta, \\ \tilde{p}(\tau), & \text{at intersection } \|p^U + (\tau - 1)(p^B - p^U)\|^2 = \Delta^2. \end{cases}$$

Solve for  $\tau$ :  $\|p^U + (\tau - 1)(p^B - p^U)\|^2 = \Delta^2$ .

For the model  $m(p) = f + g^T p + \frac{1}{2} p^T B p$ ,  $\|p\| \leq \Delta$ :

- If  $\nabla^2 f(x_k)$  is positive definite, set  $B = \nabla^2 f(x_k)$ , so  $p^B = -(\nabla^2 f(x_k))^{-1} g_k$ .
- Otherwise, use a positive definite modified Hessian (see the previous lecture) for  $B$ .

Near a solution satisfying second-order sufficient conditions,  $p^B$  is the Newton step, enabling rapid convergence.

**Note:** Modified Hessians perturb  $\nabla^2 f(x_k)$  arbitrarily, and trust-region solves with  $B_k = \nabla^2 f(x_k)$  yield  $-(\nabla^2 f(x_k) + \lambda I)^{-1} g_k$ , where  $\lambda$  ensures positive definiteness.

22/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

From the lemma on dogleg path properties, it follows that if the Newton step has a norm greater than the trust-region radius, then the path  $\tilde{p}(\tau)$  intersects the boundary exactly once. Since the model function decreases along the path, the selected step will either be the full Newton step if it lies within the trust region, or the intersection point of the path with the boundary. In the latter case,  $\tau$  is found by solving a scalar quadratic equation ensuring the step has norm equal to the trust-region radius.

When the exact Hessian is available and is positive definite, it can be directly used as  $B$  in the model, and the Newton step becomes  $-(\nabla^2 f(x_k))^{-1} g_k$ . Then the above procedure is followed: if this step is feasible, it is accepted; otherwise,  $\tau$  is determined as described. If the Hessian is not positive definite, a positive definite modification of it is used instead, as we discussed earlier, and the same procedure is applied.

Near a solution that satisfies second-order sufficient conditions, the Hessian becomes positive definite, and the Newton step is used without modification, leading to fast local convergence.

Nevertheless, using a modified Hessian in the Newton-dogleg method is unsatisfying from an intuitive perspective. The modification perturbs the diagonals arbitrarily, which may undermine the trust-region strategy. Moreover, this modification is somewhat redundant because the trust-region approach already replaces the Hessian by the Hessian at the current iterate plus a scalar times the identity matrix, with the scalar chosen to make the matrix positive definite and depending on the trust-region radius.

Thus, the Newton-dogleg method is best suited for convex problems, where the Hessian is always positive semidefinite. In general nonconvex problems, it is preferable to use strategies that explicitly incorporate directions of negative curvature, meaning directions in which the quadratic form defined by the Hessian is negative.

For global convergence, the step  $p_k$  must achieve at least a fixed fraction of the Cauchy decrease:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right), \quad (5)$$

for some constant  $c_1 \in (0, 1]$ .

This is satisfied by Dogleg methods.

When  $\Delta_k$  is minimal in the expression, this condition resembles the first Wolfe condition.

**Trust-Region Methods**

**Global solution**

**Cauchy Point**

**Dogleg Method**

**Global Convergence**

**TR Subproblem Algorithm**



### Lemma 4 (Cauchy Decrease)

The Cauchy point  $p_k^c$  satisfies the bound with  $c_1 = \frac{1}{2}$ :

$$m_k(0) - m_k(p_k^c) \geq \frac{1}{2} \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right).$$

### Comments

For global convergence of trust-region methods, each step  $p_k$  must guarantee a sufficient decrease of the quadratic model. This requirement is expressed as inequality (5), where the decrease must be at least a fixed fraction of the so-called Cauchy decrease. Explicitly, the model decrease is bounded below by a constant  $c_1$  times the norm of the gradient, multiplied by the minimum of two terms: the trust-region radius  $\Delta_k$  and the ratio of the gradient norm to the operator norm of  $B_k$ .

Intuitively, this condition means that the step cannot be too small in terms of model reduction. The dogleg method always satisfies this requirement, which ensures that it produces globally convergent iterations.

Lemma 4 states that the Cauchy point  $p_k^c$ , defined as the minimizer of the model along the steepest descent direction truncated by the trust region, achieves this inequality with constant  $c_1 = \frac{1}{2}$ . In other words, the Cauchy point guarantees at least half of the maximal possible Cauchy decrease. This result is fundamental, because it provides a uniform lower bound that can be used to prove convergence of trust-region algorithms.

## Proof of Lemma 4

**Proof:** Recall, that the Cauchy point is defined as:

$$p_k^c = -\tau_k \frac{\Delta_k}{\|g_k\|} g_k, \quad \text{where } \tau = \begin{cases} 1 & \text{if } g_k^T B_k g_k < \frac{\|g_k\|^3}{\Delta}; \\ \frac{\|g_k\|^3}{\Delta g_k^T B_k g_k} & \text{otherwise.} \end{cases}$$

For simplicity, we omit the iteration index  $k$  during the proof.

Case 1:  $g^T B g < \frac{\|g\|^3}{\Delta}$  then  $\tau = 1$  and we have  $m(p^c) - m(0) =$

$$m\left(-\Delta \frac{g}{\|g\|}\right) - f = -\Delta \|g\| + \frac{1}{2} \frac{\Delta^2}{\|g\|^2} g^T B g < -\frac{1}{2} \Delta \|g\| \leq -\frac{1}{2} \|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right)$$

Case 2:  $g^T B g \geq \frac{\|g\|^3}{\Delta}$  then  $\tau = \frac{\|g\|^3}{\Delta g^T B g}$  and we get

$$\begin{aligned} m(p^c) - m(0) &= -\frac{\|g\|^4}{g^T B g} + \frac{1}{2} \frac{\|g\|^4}{g^T B g} = -\frac{1}{2} \frac{\|g\|^4}{g^T B g} \leq \\ &- \frac{1}{2} \frac{\|g\|^4}{\|B\| \|g\|^2} = -\frac{1}{2} \frac{\|g\|^2}{\|B\|} \leq -\frac{1}{2} \|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right) \end{aligned}$$

□

24/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

Let's walk through this streamlined proof of Lemma 4, which demonstrates the effectiveness of the Cauchy point in trust-region optimization. The lemma shows that the Cauchy point, a strategic step we compute, reduces the model function enough to satisfy a key inequality. Recall that the Cauchy point is defined using a step size parameter that depends on the gradient, the trust-region radius, and the model's curvature, represented by a matrix  $B$ . The proof is carefully split into two cases based on the curvature condition, making it easier to follow, and we'll go through each to see why the Cauchy point is so reliable.

For simplicity, we omit the iteration index  $k$  during the proof. In the first case, when  $g^T B g < \frac{\|g\|^3}{\Delta}$ , the step size parameter  $\tau$  equals one, meaning the Cauchy point takes the full trust-region radius in the steepest descent direction. The proof shows that this reduces the model function by greater than a half norm of the gradient times the radius, which satisfies the inequality.

In Case 2, when  $g^T B g \geq \frac{\|g\|^3}{\Delta}$ ,  $\tau$  is chosen so that the minimizer lies inside the trust region. Substituting this expression gives the decrease equal to  $-\frac{1}{2} \frac{\|g\|^4}{g^T B g}$ . This is then bounded by  $-\frac{1}{2} \frac{\|g\|^2}{\|B\|}$ . Again, this is less than or equal to  $-\frac{1}{2} \|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right)$ .

Therefore, in both cases the Cauchy point satisfies the desired decrease with constant one half.

To satisfy (5), it suffices that the approximate solution  $p_k$  achieves at least a fixed fraction  $c_2$  of the reduction obtained by the Cauchy point:

$$m_k(0) - m_k(p_k) \geq c_2(m_k(0) - m_k(p_k^c)).$$

## Theorem 13

Let  $p_k$  satisfy  $\|p_k\| \leq \Delta_k$  and  $m_k(0) - m_k(p_k) \geq c_2(m_k(0) - m_k(p_k^c))$ .

Then  $p_k$  satisfies (5) with  $c_1 = \frac{1}{2}c_2$ . In particular, if  $p_k$  is the exact solution  $p_k^*$  of the trust-region subproblem, then  $c_1 = \frac{1}{2}$ .

**Proof:** Since  $\|p_k\| \leq \Delta_k$ , the previous Lemma implies

$$m_k(0) - m_k(p_k) \geq c_2(m_k(0) - m_k(p_k^c)) \geq \frac{1}{2}c_2\|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right).$$

□

Note: The dogleg algorithm satisfies (5) with  $c_1 = \frac{1}{2}$ , because it produces approximate solution  $p_k$  for which  $m_k(p_k) \leq m_k(p_k^c)$ .

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



## Comments

Let's explore the use of the Cauchy point to ensure global convergence in trust-region optimization. The focus here is on a condition that guarantees our approximate solution is good enough to drive the optimization process toward a solution. The key idea is that any approximate step we take should reduce the model function by at least a fixed fraction of the reduction achieved by the Cauchy point, which we know is effective from our previous lemma. This slide presents a theorem that formalizes this idea, followed by a brief proof and a note about the dogleg algorithm.

The theorem states that if our step stays within the trust-region radius and achieves at least a fraction of the Cauchy point's reduction, it satisfies the required condition for convergence, with a constant that's half the fraction we choose. The proof is straightforward: it uses the lemma we proved earlier, which guarantees the Cauchy point's reduction, and scales it by the chosen fraction to show the step meets the convergence criterion.

Note that the dogleg algorithm, which constructs a path to approximate the trust-region solution, always does at least as well as the Cauchy point, ensuring the same convergence constant. This ties together the Cauchy point's reliability with practical algorithms, showing how they work together to ensure steady progress in optimization.

Global convergence results depend on the choice of the acceptance parameter  $\eta$  in Algorithm 6:

- ▶ For  $\eta = 0$ , the step is accepted if it reduces  $f$ , and the gradient sequence has a limit point at zero.
- ▶ For  $\eta > 0$ , actual reduction must be at least some small fraction of the predicted decrease we obtain the stronger result that  $g_k \rightarrow 0$ .

Assumptions for global convergence:

- ▶ The approximate Hessians  $B_k$  are uniformly bounded in norm.
- ▶ The objective function  $f$  is bounded below on the level set

$$S \stackrel{\text{def}}{=} \{x \mid f(x) \leq f(x_0)\}.$$

- ▶ We define a neighborhood of  $S$  by

$$S(R_0) \stackrel{\text{def}}{=} \{x \mid \|x - y\| < R_0 \text{ for some } y \in S\},$$

where  $R_0 > 0$  is fixed.

- ▶ The step length may exceed the trust-region radius, as long as

$$\|p_k\| \leq \gamma \Delta_k, \quad \text{for some constant } \gamma \geq 1.$$

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



## Comments

Let's now explore the global convergence properties of trust-region methods, which tell us how the algorithm behaves over many iterations. Recall that we are talking about an algorithm that determines how the radius of the trust region changes depending on the ratio of the actual reduction to the predicted reduction.

Let's consider two scenarios based on a parameter that controls how strictly we accept a step. Next, we will prove that if we accept a step as soon as it reduces the objective function (the case  $\eta = 0$ , we can show that the gradients of the function at our iterates will have at least one point where they approach zero i.e. The sequence of gradients  $g_k$  has a subsequence converging to zero. However, if we require the step to achieve a small fraction of the predicted reduction, we get a stronger result: the gradients converge to zero, indicating we're approaching a stationary point.

To set the stage for the proofs, some key assumptions are introduced. The matrices approximating the curvature of the function are assumed to have bounded norms, ensuring they don't grow uncontrollably. The objective function is also assumed to be bounded below within a specific set of points (the level set) where the function value is no higher than at the starting point. Additionally, a neighborhood around this set is defined for flexibility in the analysis, and the step size is allowed to slightly exceed the trust-region radius, as long as it stays within a fixed multiple of it. These assumptions provide a solid foundation for proving that trust-region methods reliably drive the optimization process toward a solution.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm

**Theorem 14**

Let  $\eta = 0$  in Algorithm 6. Suppose the following hold:

- $\|B_k\| \leq \beta$  for some constant  $\beta$ ;
- $f$  is bounded below on the level set  $S$  and is Lipschitz continuously differentiable in a neighborhood  $S(R_0)$  for some  $R_0 > 0$ ;
- all approximate solutions  $p_k$  of the trust-region subproblem (3a) satisfy the conditions

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min \left( \Delta_k, \frac{\|g_k\|}{\|B_k\|} \right), \quad \|p_k\| \leq \gamma \Delta_k,$$

for some constants  $c_1 \in (0, 1]$ ,  $\gamma \geq 1$ .

Then we have  $\liminf_{k \rightarrow \infty} \|g_k\| = 0$ .

**Comments**

Let's dive into the global convergence theorem for trust-region methods when the step acceptance parameter is set to zero, meaning we accept any step that reduces the objective function. This theorem, labeled as Theorem 14, outlines conditions under which the algorithm guarantees that the norm of the gradient at some points in the iteration sequence approaches zero, a sign that we're nearing a stationary point. The theorem sets up a framework with specific assumptions to ensure this behavior.

The theorem assumes that the matrices approximating the function's curvature are uniformly bounded in norm, the objective function is bounded below on the level set defined by the initial function value and is Lipschitz continuously differentiable in an open neighborhood around this set, and the approximate solutions to the trust-region subproblem achieve a reduction in the model function at least proportional to the norm of the gradient and the trust-region radius (or a related term), while remaining within a fixed multiple of the radius.

Let me remind you that Lipschitz continuous differentiability means the norm of the difference between gradients at any two points in the neighborhood is bounded by a constant times the distance between those points, ensuring smooth behavior.

Trust-Region Methods
Global solution
Cauchy Point
Dogleg Method
Global Convergence
TR Subproblem Algorithm



**Proof:** From Taylor's theorem (Theorem 1) we have:

$$f(x_k + p_k) = f(x_k) + g(x_k)^T p_k + \int_0^1 [g(x_k + tp_k) - g(x_k)]^T p_k dt.$$

Using the definition of  $m_k(p) = f(x_k) + g(x_k)^T p + \frac{1}{2} p^T B_k p$ , it follows that

$$|m_k(p_k) - f(x_k + p_k)| = \left| \frac{1}{2} p_k^T B_k p_k - \int_0^1 [g(x_k + tp_k) - g(x_k)]^T p_k dt \right|.$$

This gives the bound:

$$|m_k(p_k) - f(x_k + p_k)| \leq \frac{\beta}{2} \|p_k\|^2 + \beta_1 \|p_k\|^2,$$

where  $\beta_1$  is the Lipschitz constant of  $g$  on  $S(R_0)$ . We assumed that  $\|p_k\| \leq R_0$  to ensure that  $x_k$  and  $x_k + tp_k$  lie in  $S(R_0)$ .

### Comments

By Taylor's theorem, we express the value of the true objective function at the new point  $x_k + p_k$  as the function value at  $x_k$  plus the scalar product of the gradient at  $x_k$  with the step  $p_k$ , plus an integral term that accounts for the change in the gradient along the step from  $x_k$  to  $x_k + p_k$ . This is compared to the model function, defined earlier, which includes the function value at  $x_k$ , the same gradient term, and a quadratic term involving the curvature matrix.

We then compute the absolute difference between the model function and the true function at  $p_k$ , which reduces to the difference between the quadratic term and the integral term. This difference is bounded using the norm of the curvature matrix and the Lipschitz constant of the gradient, assuming the step  $p_k$  stays within a neighborhood where the gradient is Lipschitz continuous.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Assumption for contradiction.

Suppose that there exist  $\epsilon > 0$  and an index  $K > 0$  such that

$$\|g_k\| \geq \epsilon, \quad \text{for all } k \geq K.$$

From the assumptions of the theorem, for all  $k \geq K$  we have:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right) \geq c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right).$$

We analyze the acceptance ratio  $\rho_k$ :

$$|\rho_k - 1| = \left| \frac{(f(x_k) - f(x_k + p_k)) - (m_k(0) - m_k(p_k))}{m_k(0) - m_k(p_k)} \right| = \left| \frac{m_k(p_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \right|.$$

Using the bound on the model accuracy and the step length, we get:

$$|\rho_k - 1| \leq \frac{\gamma^2 \Delta_k^2 (\beta/2 + \beta_1)}{c_1 \epsilon \min(\Delta_k, \epsilon/\beta)}.$$

### Comments

Next, to prove the theorem, we assume for contradiction that the norm of the gradient remains bounded below by a positive constant for all iterations beyond some point. By the theorem's assumptions, for iterations  $k \geq K$ , we use the bound on the model function reduction, which is at least a constant times the norm of the gradient multiplied by the minimum of the trust-region radius and a term involving the gradient norm and the curvature matrix norm. Given the contradiction assumption that the gradient norm is bounded below by a positive constant, this reduction is further bounded by a term proportional to the constant, the positive threshold  $\epsilon$ , and the minimum of the trust-region radius and a scaled threshold  $\epsilon$ .

We then evaluate the acceptance ratio  $\rho_k$ , which measures how close the actual reduction in the objective function is to the predicted reduction in the model. The absolute difference between this ratio and one is expressed as the ratio of the model-true function discrepancy to the model reduction. Using the earlier bounds on the model-true function difference, the model reduction, and the step size, we derive an upper bound on this difference, which depends on the square of the trust-region radius and constants from the curvature and Lipschitz properties.

## Proof of Theorem 14 (continued)

Define the threshold that holds for all sufficiently small values of  $\Delta_k$ , that is, for all  $\Delta_k \leq \bar{\Delta}$ :

$$\bar{\Delta} = \min\left(\frac{1}{2} \cdot \frac{c_1 \epsilon}{\gamma^2(\beta/2 + \beta_1)}, \frac{R_0}{\gamma}\right).$$

The term  $R_0/\gamma$  in  $\bar{\Delta}$  ensures that  $\|p_k\| \leq R_0$ . Since  $c_1 \leq 1$  and  $\gamma \geq 1$ , we have  $\bar{\Delta} \leq \epsilon/\beta$ , so

$$\min(\Delta_k, \epsilon/\beta) = \Delta_k \quad \text{for all } \Delta_k \leq \bar{\Delta}.$$

Hence,

$$|\rho_k - 1| \leq \frac{\gamma^2 \Delta_k^2 (\beta/2 + \beta_1)}{c_1 \epsilon \Delta_k} = \frac{\gamma^2 \Delta_k (\beta/2 + \beta_1)}{c_1 \epsilon} \leq \frac{\gamma^2 \bar{\Delta} (\beta/2 + \beta_1)}{c_1 \epsilon} \leq \frac{1}{2}.$$

Therefore,  $\rho_k \geq 1/2$ , and Algorithm 6 guarantees that  $\Delta_{k+1} \geq \Delta_k$  for all  $\Delta_k \leq \bar{\Delta}$ . Thus,  $\Delta_k$  is reduced (by a factor of  $\frac{1}{4}$ ) only when

$$\Delta_k > \bar{\Delta},$$

which implies

$$\Delta_k \geq \min(\Delta_K, \bar{\Delta}/4) \quad \text{for all } k \geq K.$$

30/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

We now derive a bound on the right-hand side of the expression for the absolute difference between  $\rho_k$  and one, valid for all sufficiently small values of the trust-region radius,  $\Delta_k$ , specifically those less than or equal to a threshold,  $\bar{\Delta}$ . This threshold is defined as the minimum of two terms: one proportional to the constant  $c_1$  times the positive gradient threshold  $\epsilon$ , divided by the square of  $\gamma$  times the sum of half the curvature bound  $\beta$  and the Lipschitz constant  $\beta_1$ , and another equal to the neighborhood radius  $R_0$  divided by  $\gamma$ . The second term ensures the step's norm, at most  $\gamma$  times  $\Delta_k$ , remains at most  $R_0$ , satisfying the step size bound.

From the definition of  $\bar{\Delta}$  and the fact that  $c_1 \leq 1$ , and  $\gamma \geq 1$ , it follows that  $\bar{\Delta} \leq \epsilon/\beta$ , we simplify the denominator of the estimate for the absolute difference between  $\rho_k$  and one taking  $\Delta_k$  as the smaller value for  $\Delta_k$  up to  $\bar{\Delta}$ . Using this, we bound the absolute difference between  $\rho_k$  and one, reducing it to a term proportional to  $\Delta_k$  times the sum of half  $\beta$  and  $\beta_1$ , divided by  $c_1$  times  $\epsilon$ . Substituting  $\bar{\Delta}$ , this difference is at most one-half, ensuring  $\rho_k$  is at least half. Thus, Algorithm 6 does not reduce  $\Delta_k$  when  $\Delta_k$  is at most  $\bar{\Delta}$ , as the step is sufficiently successful. Hence, a reduction of  $\Delta_k$  by a factor of one-fourth occurs only when  $\Delta_k$  is greater than  $\bar{\Delta}$ . Consequently,  $\Delta_k$  remains at least the minimum of the initial radius at iteration  $K$  and one-fourth of  $\bar{\Delta}$ , ensuring the algorithm maintains a sufficiently large radius to advance the contradiction argument.

## Proof of Theorem 14 (continued)

Suppose there is an infinite subsequence  $\mathcal{K}$  such that  $\rho_k \geq 1/4$  for  $k \in \mathcal{K}$ . For  $k \in \mathcal{K}$  and  $k \geq K$ , we get

$$f(x_k) - f(x_{k+1}) = f(x_k) - f(x_k + p_k) \geq \frac{1}{4} [m_k(0) - m_k(p_k)] \geq \frac{1}{4} c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right).$$

Since  $f$  is bounded below, this implies

$$\lim_{k \in \mathcal{K}, k \rightarrow \infty} \Delta_k = 0,$$

which contradicts the result that  $\Delta_k$  is bounded from below. Hence, no such infinite subsequence  $\mathcal{K}$  exists, and we must have  $\rho_k < 1/4$  for all sufficiently large  $k$ . This, in turn, contradicts the result that  $\rho_k \geq 1/2$  for all  $\Delta_k \leq \bar{\Delta}$ . Therefore, our assumption is false, and we conclude that

$$\liminf_{k \rightarrow \infty} \|g_k\| = 0.$$

□

31/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

We now suppose there exists an infinite subsequence of iterations, denoted by  $\mathcal{K}$ , where  $\rho_k$  is at least one-fourth for each iteration  $k$  in  $\mathcal{K}$ . For these iterations, when  $k \geq K$ , the decrease in the objective function from  $x_k$  to  $x_k + p_k$  is at least one-fourth of the model function reduction, which itself is at least a constant  $c_1$  times  $\epsilon$  times the minimum of the trust-region radius  $\Delta_k$  and  $\epsilon$  divided by the curvature bound  $\beta$ . Since the objective function is bounded below, this positive decrease over an infinite subsequence implies that  $\Delta_k$  must approach zero for  $k$  in  $\mathcal{K}$  as  $k$  approaches infinity.

However, this contradicts the earlier result that  $\Delta_k$  is bounded below by the minimum of the initial radius at iteration  $K$  and one-fourth of  $\bar{\Delta}$ . Thus, no such infinite subsequence  $\mathcal{K}$  can exist, meaning  $\rho_k$  must be less than one-fourth for all sufficiently large  $k$ . This, in turn, contradicts the finding that  $\rho_k$  is at least one-half whenever  $\Delta_k$  is at most  $\bar{\Delta}$ . Therefore, the assumption that the gradient norm remains bounded below by  $\epsilon$  is false, leading to the conclusion that the limit inferior of the gradient norm as  $k$  approaches infinity is zero, completing the proof.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



**Proof:** Let  $m$  be an index such that  $g_m \neq 0$ . Denote by  $\beta_1$  the Lipschitz constant for  $g$  on  $S(R_0)$  (i.e.  $\|g(x) - g_m\| \leq \beta_1 \|x - x_m\|$ ), and define

$$\epsilon = \frac{1}{2} \|g_m\|, \quad R = \min \left( \frac{\epsilon}{\beta_1}, R_0 \right).$$

Then the ball  $B(x_m, R) = \{x \mid \|x - x_m\| \leq R\}$  lies in  $S(R_0)$ , and Lipschitz continuity of  $g$  holds for all  $x \in B(x_m, R)$ , thus we have

$$\|g(x)\| \geq \|g_m\| - \|g(x) - g_m\| \geq \frac{1}{2} \|g_m\| = \epsilon.$$

32/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

We now extend our convergence analysis to the case where the acceptance threshold  $\eta$  is strictly positive, with  $\eta$  chosen between zero and one-half. The assumptions are similar to the previous result. We suppose that the sequence of Hessian approximations, denoted by  $B_k$ , has bounded norm, specifically less than or equal to a constant  $\beta$ . We also assume that the objective function  $f$  is bounded below on the level set, and Lipschitz continuously differentiable on a slightly larger set with Lipschitz constant  $\beta_1$ . As before, every approximate solution  $p_k$  of the trust-region subproblem must satisfy two conditions: first, that the model decrease, written as  $m_k(0) - m_k(p_k)$ , is at least a constant  $c_1$  times the gradient norm multiplied by the minimum of the trust-region radius  $\Delta_k$  and the gradient norm divided by the norm of  $B_k$ ; and second, that the step norm, the norm of  $p_k$ , does not exceed  $\gamma$  times  $\Delta_k$ .

To start the proof, we fix an index  $m$  such that the gradient at iteration  $m$  is not zero. We then define  $\epsilon$  to be one-half the gradient norm at this point, and we set a radius  $R$  equal to the minimum of  $\epsilon$  divided by  $\beta_1$  and  $R_0$ . The closed ball centered at  $x_m$  with radius  $R$  is fully contained in the region where Lipschitz continuity holds. Inside this ball, every point  $x$  satisfies that the gradient norm at  $x$  is at least  $\epsilon$ . This construction will be used to develop the contradiction argument.

If the entire sequence  $\{x_k\}_{k \geq m}$  stays inside the ball  $\mathcal{B}(x_m, R)$ , we would have  $\|g_k\| \geq \epsilon > 0$  for all  $k \geq m$ . The reasoning in the proof of Theorem 14 can be used to show that this scenario does not occur. Therefore, the sequence  $\{x_k\}_{k \geq m}$  eventually leaves  $\mathcal{B}(x_m, R)$ .

- Let  $l \geq m$  be such that  $x_{l+1}$  is the first iterate after  $x_m$  outside  $\mathcal{B}(x_m, R)$ .
- Since  $\|g_k\| \geq \epsilon$  for  $k = m, \dots, l$ , we use  $\left( m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min \left( \Delta_k, \frac{\|g_k\|}{\|B_k\|} \right) \right)$  to write:

$$\begin{aligned} f(x_m) - f(x_{l+1}) &= \sum_{k=m}^l [f(x_k) - f(x_{k+1})] \geq \sum_{\substack{k=m \\ x_k \neq x_{k+1}}}^l \eta [m_k(0) - m_k(p_k)] \\ &\geq \sum_{\substack{k=m \\ x_k \neq x_{k+1}}}^l \eta c_1 \epsilon \min \left( \Delta_k, \frac{\epsilon}{\beta} \right) \end{aligned}$$

where we have limited the sum to the iterations  $k$  for which  $x_k \neq x_{k+1}$ , that is, those iterations on which a step was actually taken.

33/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Trust-Region Methods**

**Global solution**

**Cauchy Point**

**Dogleg Method**

**Global Convergence**

**TR Subproblem Algorithm**



### Comments

Suppose for contradiction that the sequence of iterates remains entirely within this ball around  $x_m$ . In that case, the gradient norms would remain at least  $\epsilon$  for all subsequent iterations. However, reasoning similar to that in the earlier proof shows that this cannot happen, and therefore the sequence must eventually leave the ball. Let  $l$  be the index such that the iterate  $x_{l+1}$  is the first point outside this ball. For every index from  $m$  through  $l$ , the gradient norm remains at least  $\epsilon$ .

We now consider the decrease in the objective function over these iterations. By summing the reductions from  $x_m$  through  $x_{l+1}$ , and restricting attention only to those iterations where a step was actually taken, we find that the total reduction is at least  $\eta$  times the sum of the model decreases. Using the lower bound on the model decrease, this sum is bounded below by  $\eta$  times  $c_1$  times  $\epsilon$  times the minimum of  $\Delta_k$  and  $\epsilon$  divided by  $\beta$ , aggregated over the relevant iterations.

This inequality quantifies the guaranteed progress made whenever the gradient is bounded away from zero. It shows that the function value must decrease by a fixed positive amount whenever the algorithm is forced to exit the ball. This observation is a central step in reaching a contradiction, since the function is bounded below and cannot decrease indefinitely.

## Global Convergence (Case $\eta > 0$ , conclusion)

- If  $\Delta_k \leq \epsilon/\beta$  for all  $k = m, \dots, l$ , then

$$f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \sum_{\substack{k=m \\ x_k \neq x_{k+1}}}^l \Delta_k \geq \eta c_1 \epsilon R = \eta c_1 \epsilon \min \left( \frac{\epsilon}{\beta_1}, R_0 \right)$$

- Otherwise, if  $\Delta_k > \epsilon/\beta$  for some  $k = m, \dots, l$ , then

$$f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \cdot \frac{\epsilon}{\beta}$$

- Since the sequence  $\{f(x_k)\}_{k=0}^\infty$  is decreasing and bounded below, we have

$$f(x_k) \downarrow f^* \text{ for some } f^* > -\infty$$

- Therefore,

$$\begin{aligned} f(x_m) - f^* &\geq f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \min \left( \frac{\epsilon}{\beta}, \frac{\epsilon}{\beta_1}, R_0 \right) \\ &= \frac{1}{2} \eta c_1 \|g_m\| \min \left( \frac{\|g_m\|}{2\beta}, \frac{\|g_m\|}{2\beta_1}, R_0 \right) > 0 \end{aligned}$$

- Since  $f(x_m) - f^* \rightarrow 0$ , it must be that  $\|g_m\| \rightarrow 0$ .

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

To complete the argument, we separate the analysis into two cases. First, suppose that for every relevant iteration, the trust-region radius  $\Delta_k$  is less than or equal to  $\epsilon$  divided by  $\beta$ . In this situation, the cumulative decrease in the function value from  $x_m$  to  $x_{l+1}$  is bounded below by  $\eta$  times  $c_1$  times  $\epsilon$  times the sum of  $\Delta_k$ . Since the total step length needed to exit the ball is at least  $R$ , this bound implies that the function decreases by at least  $\eta$  times  $c_1$  times  $\epsilon$  times  $R$ . Substituting the definition of  $R$ , this becomes  $\eta$  times  $c_1$  times  $\epsilon$  times the minimum of  $\epsilon$  divided by  $\beta_1$  and  $R_0$ .

In the second case, suppose that for some iteration the trust-region radius  $\Delta_k$  exceeds  $\epsilon$  divided by  $\beta$ . Then the decrease in the function value from  $x_m$  to  $x_{l+1}$  is bounded below by  $\eta$  times  $c_1$  times  $\epsilon$  times  $\epsilon$  divided by  $\beta$ .

Since the sequence of function values  $f(x_k)$  is monotonically decreasing and bounded below, it must converge to some finite limit  $f^*$ . Combining this convergence with the inequalities we just derived, we see that  $f(x_m) - f^*$  is greater than or equal to a strictly positive quantity that depends on the gradient norm at iteration  $m$ . But since  $f(x_m)$  approaches  $f^*$ , this can only be true if the gradient norm itself approaches zero. This establishes the global convergence result. The theorem is proved.

## Characterization of the Trust-Region Solution

The characterization of Theorem 12 suggests an algorithm for finding the solution  $p$  of the subproblem (3b):

$$\min_{p \in \mathbb{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p \quad \text{s.t. } \|p\| \leq \Delta.$$

Either  $\lambda = 0$  satisfies the following conditions of Theorem 12:

- (4a):  $(B + \lambda I)p^* = -g$
- (4c):  $B + \lambda I$  is positive semidefinite, with  $\|p\| \leq \Delta$

or else for  $\lambda$  sufficiently large that  $B + \lambda I$  is positive definite we define:

$$p(\lambda) = -(B + \lambda I)^{-1}g \quad \text{and find } \lambda > 0 \text{ such that } \|p(\lambda)\| = \Delta$$

This is a one-dimensional root-finding problem in  $\lambda$ .

To see that a value of  $\lambda$  with all the desired properties exists, we appeal to the eigendecomposition of  $B$  and use it to study the properties of  $\|p(\lambda)\|$ .

Since  $B$  is symmetric, there is an orthogonal matrix  $Q$  and a diagonal matrix  $\Lambda$  such that

$$B = Q\Lambda Q^T, \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n),$$

where  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$  are the eigenvalues of  $B$ .

35/41 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Trust-Region Methods**

**Global solution**

**Cauchy Point**

**Dogleg Method**

**Global Convergence**

**TR Subproblem Algorithm**



### Comments

We now turn to the characterization of the trust-region solution. The trust-region subproblem consists of minimizing the quadratic model  $m(p) = f + g^T p + \frac{1}{2} p^T B p$ , subject to the constraint that the norm of  $p$  does not exceed the trust-region radius  $\Delta$ . The Moré-Sorensen theorem states that there exists a scalar  $\lambda$  such that the step  $p^*$  satisfies the equation  $(B + \lambda I)p^* = -g$ , and that the matrix  $B + \lambda I$  is positive semidefinite. Furthermore, either  $\lambda = 0$  and the step norm is within  $\Delta$ , or else we can define  $p(\lambda) = -(B + \lambda I)^{-1}g$ . Then we seek a strictly positive value of  $\lambda$  for which the norm of  $p(\lambda)$  equals  $\Delta$ .

This reduces the subproblem to a one-dimensional root-finding task in the scalar variable  $\lambda$ . To ensure that such a  $\lambda$  exists, we analyze the eigenstructure of the Hessian approximation  $B$ . Since  $B$  is symmetric, it can be decomposed as  $Q\Lambda Q^T$ , where  $Q$  is an orthogonal matrix and  $\Lambda$  is diagonal with entries  $\lambda_1$  through  $\lambda_n$ .

Since  $B + \lambda I = Q(\Lambda + \lambda I)Q^T$ , we obtain for  $\lambda \neq \lambda_j$ :

$$p(\lambda) = -Q(\Lambda + \lambda I)^{-1}Q^T g = -\sum_{j=1}^n \frac{q_j^T g}{\lambda_j + \lambda} q_j,$$

where  $q_j$  is the  $j$ th column of  $Q$ .

By orthonormality of the columns of  $Q$ , the squared norm is

$$\|p(\lambda)\|^2 = \sum_{j=1}^n \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2}.$$

If  $\lambda > -\lambda_1$ , then  $\lambda_j + \lambda > 0$  for all  $j = 1, 2, \dots, n$ , so  $\|p(\lambda)\|$  is a continuous, nonincreasing function of  $\lambda$  on  $(-\lambda_1, \infty)$ .

We have:

- ▶  $\lim_{\lambda \rightarrow \infty} \|p(\lambda)\| = 0$ ,
- ▶ moreover, if  $q_j^T g \neq 0$  then  $\lim_{\lambda \rightarrow -\lambda_j} \|p(\lambda)\| = \infty$ .

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



### Comments

This decomposition allows us to express  $p(\lambda)$  explicitly as a weighted sum of eigenvectors  $q_j$ , with coefficients depending on the inner products  $q_j^T g$  and the shifted eigenvalues  $\lambda_j + \lambda$ . As a result, the squared norm of  $p(\lambda)$  can be written as a sum over  $j$  of the squared inner product divided by the squared shifted eigenvalue.

This formula has three immediate consequences. First, continuity and monotonicity: for  $\lambda > -\lambda_1$ , all denominators are positive; increasing  $\lambda$  increases every denominator, so the norm of  $p(\lambda)$  is a continuous, nonincreasing function of  $\lambda$  on the interval  $(-\lambda_1, \infty)$ . Second, limits: as  $\lambda$  tends to infinity, every term vanishes and the norm goes to zero—large  $\lambda$  suppresses all directions uniformly. If for some index  $j$  the projection  $q_j^T g$  is nonzero, then as  $\lambda$  approaches  $-\lambda_j$  from the right, the corresponding denominator tends to zero and the norm blows up to infinity. Third, existence and typically uniqueness of the trust-region multiplier: because the norm of  $p(\lambda)$  decreases continuously from infinity down to zero on that interval, there exists a value of  $\lambda \geq 0$  such that the norm of  $p(\lambda)$  equals the trust-region radius  $\Delta$ . When the projections are not all zero, this solution is unique.

Intuitively,  $\lambda$  acts as a global damping knob: increasing  $\lambda$  shrinks components along “soft” eigen-directions—those with small or negative  $\lambda_j$ —more aggressively, yielding a step that neatly fits the prescribed radius.

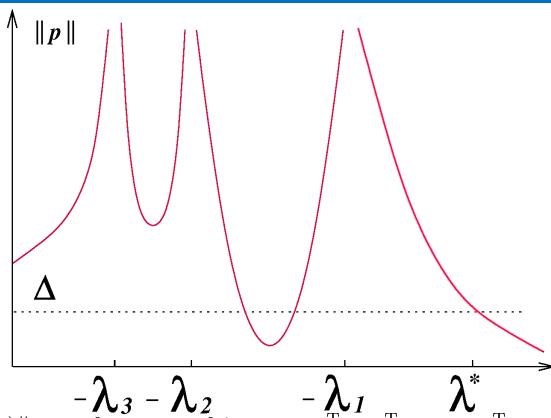


Figure:  $\|p(\lambda)\|$  as a function of  $\lambda$  when  $q_1^T g$ ,  $q_2^T g$ , and  $q_3^T g$  are all nonzero.

**Note:** If the properties of  $\|p(\lambda)\|$  derived on the previous slide hold and  $\|p(\lambda)\|$  is a continuous, nonincreasing function on  $(-\lambda_1, \infty)$ ; If  $q_1^T g \neq 0$ , then there exists a unique  $\lambda^* \in (-\lambda_1, \infty)$  such that  $\|p(\lambda^*)\| = \Delta$  (other smaller values of  $\lambda$  might satisfy this equation, but they violate condition (4c)).

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



## Comments

Let us now take a closer look at the function that defines the step length as a function of the parameter  $\lambda$ . The key property is monotonicity. As  $\lambda$  increases beyond  $-\lambda_1$ , the smallest eigenvalue, each denominator becomes larger, and therefore the step norm decreases. Hence, the function  $\|p(\lambda)\|$  is continuous and strictly decreasing on the interval from  $-\lambda_1$  to infinity. Its limits are also clear: as  $\lambda$  tends to  $-\lambda_1$  from above, the step length tends to infinity; as  $\lambda$  tends to infinity, the step length approaches zero.

From this, we conclude that for any trust-region radius  $\Delta$ , there must exist a unique value of  $\lambda^* > -\lambda_1$ , such that  $\|p(\lambda^*)\| = \Delta$ . Graphically, we see a curve descending smoothly from infinity to zero, and the horizontal line at  $\Delta$  will intersect this curve exactly once. This intersection defines  $\lambda^*$ . The uniqueness of this solution is crucial, because it ensures that our trust-region subproblem always has a well-defined and consistent answer. This forms the basis for the numerical procedures we develop next.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



**Key Idea** Identify the value of  $\lambda^* \in (-\lambda_1, \infty)$  for which the norm of the step equals the trust-region radius, that is,  $\|p(\lambda^*)\| = \Delta$ , which works when  $q_1^T g \neq 0$ . Use a root-finding procedure based on properties of  $\|p(\lambda)\|$ .

If  $B$  is positive definite and  $\|B^{-1}g\| \leq \Delta$ , then  $\lambda^* = 0$  satisfies the conditions (4a)-(4c) of Theorem 12, so the procedure can be terminated immediately with  $\lambda^* = 0$ .

Otherwise (in the case  $q_1^T g \neq 0$ , we solve the nonlinear equation

$$\phi_1(\lambda) = \|p(\lambda)\| - \Delta = 0$$

for  $\lambda > -\lambda_1$  using Newton's method.

Near the asymptote at  $\lambda = -\lambda_1$ , the function  $\phi_1(\lambda)$  behaves like

$$\phi_1(\lambda) \approx \frac{C_1}{\lambda + \lambda_1} + C_2 \quad \text{with } C_1 > 0.$$

This approximation highlights potential instability of Newton's method near  $-\lambda_1$  (i.e. it may be unreliable or slow).

## Comments

The next step is to discuss how  $\lambda^*$  can actually be computed. The key idea is to identify the value of  $\lambda^*$  within the interval from  $-\lambda_1$  to infinity for which the norm of  $p(\lambda)$  equals the trust-region radius  $\Delta$ . If the matrix  $B$  is positive definite and the unconstrained minimizer, which is  $-B^{-1}g$ , already lies inside the trust region, then the trust-region constraint is inactive, and the correct value is simply  $\lambda^* = 0$ . In this case, no iteration is necessary. However, in the more general situation, when  $q_1^T g \neq 0$  and the unconstrained minimizer lies outside the trust region, we must solve the nonlinear equation  $\phi_1(\lambda) = \|p(\lambda)\| - \Delta = 0$ . This is precisely a root-finding problem. Newton's method is the natural choice here because it can take advantage of the smoothness of the function. However, a difficulty arises near the singularity at  $\lambda = -\lambda_1$ . In this region,  $\phi_1(\lambda)$  behaves approximately like  $C_1/(\lambda + \lambda_1) + C_2$ , with  $C_1 > 0$ . This means that near the singularity, Newton's method can become unstable or converge slowly. Thus, although Newton's method is powerful, care must be taken when  $\lambda$  is close to the negative of the smallest eigenvalue.

Trust-Region Methods

Global solution

Cauchy Point

Dogleg Method

Global Convergence

TR Subproblem Algorithm



**Key Idea** Improve convergence and numerical stability by reformulating the root-finding problem using an alternative function that behaves nearly linearly near the singularity at  $-\lambda_1$ .

To avoid instability near  $-\lambda_1$ , define:

$$\phi_2(\lambda) = \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}$$

so that  $\phi_2$  is nearly linear near  $-\lambda_1$ . Using the expansion of  $\|p(\lambda)\|$ , we get:

$$\phi_2(\lambda) \approx \frac{1}{\Delta} - \frac{\lambda + \lambda_1}{C_3}, \quad C_3 > 0$$

The root-finding Newton's method will perform well and it iteration becomes:

$$\lambda^{(\ell+1)} = \lambda^{(\ell)} - \frac{\phi_2(\lambda^{(\ell)})}{\phi'_2(\lambda^{(\ell)})}$$

## Comments

To overcome the difficulty near the singularity, the root-finding problem is reformulated in terms of an alternative function. Instead of directly working with  $\phi_1$ , we introduce  $\phi_2(\lambda) = 1/\Delta - 1/\|p(\lambda)\|$ . The advantage of this formulation is that  $\phi_2$  behaves nearly linearly near the critical point at  $-\lambda_1$ . Using the expansion of the step norm, one can show that  $\phi_2(\lambda) \approx 1/\Delta - (\lambda + \lambda_1)/C_3$ , with  $C_3 > 0$ . This linear-like behavior makes Newton's method far more reliable. The Newton iteration is then expressed as  $\lambda^{(\ell+1)} = \lambda^{(\ell)} - \phi_2(\lambda^{(\ell)})/\phi'_2(\lambda^{(\ell)})$ . Thanks to the improved conditioning of the problem, convergence becomes faster and more stable, especially when  $\lambda$  approaches values close to  $-\lambda_1$ . This reformulation illustrates a general principle in numerical optimization: sometimes, by changing the function under consideration, we can transform a numerically ill-conditioned problem into one that is far more stable and efficient to solve.

## Graph of Reformulated Function $\phi_2(\lambda)$

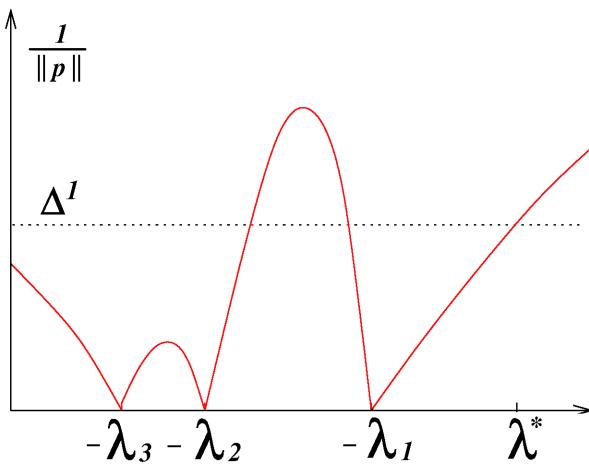


Figure:  $1/\|p(\lambda)\|$  as a function of  $\lambda$ .

The shape of  $1/\|p(\lambda)\|$  shows that  $\phi_2(\lambda)$  behaves nearly linearly near  $-\lambda_1$ , improving stability of Newton iterations.

- Trust-Region Methods
- Global solution
- Cauchy Point
- Dogleg Method
- Global Convergence**
- TR Subproblem Algorithm



### Comments

Finally, let us interpret the graph of the reformulated function. Instead of plotting the step norm directly, we plot its reciprocal,  $1/\|p(\lambda)\|$ . This transformation makes the behavior near the singularity transparent. Specifically, as  $\lambda$  approaches  $-\lambda_1$ , the reciprocal of the norm grows approximately linearly, which confirms that  $\phi_2(\lambda)$  indeed behaves like a straight line in this region. As a result, when Newton's method is applied to  $\phi_2$ , the iterations are not slowed down by the presence of the singularity but instead proceed smoothly. The graphical evidence clearly demonstrates that the reformulation is not only theoretically justified but also practically effective. Thus, by introducing the reciprocal function, we obtain a numerically stable and efficient root-finding process for determining the trust-region parameter  $\lambda^*$ . This completes the analysis of existence, uniqueness, and computation of  $\lambda^*$ , and provides the foundation for implementing robust trust-region algorithms.

**Key Idea** Use Newton's method on a transformed root-finding function to find  $\lambda^* > -\lambda_1$  such that  $\|p(\lambda^*)\| = \Delta$ , computing  $p(\lambda)$  via Cholesky factorization.

It is easy to show by direct calculations that

$$\frac{\phi_2(\lambda^{(\ell)})}{\phi'_2(\lambda^{(\ell)})} = \frac{\frac{1}{\|p\|} - \frac{1}{\Delta}}{\frac{p^T(B + \lambda I)^{-1}p}{\|p\|^3}} = \frac{\|p\|^2}{p^T(B + \lambda I)^{-1}p} \left( \frac{\Delta - \|p_\ell\|}{\Delta} \right)$$

**Algorithm 7** (Trust Region Subproblem):

Require: Initial guess  $\lambda^{(0)}$ , trust region radius  $\Delta > 0$

1: for  $\ell = 0, 1, 2, \dots$  do

2: Factor  $B + \lambda^{(\ell)}I = R^T R$  (Cholesky)

3: Solve  $R^T R p_\ell = -g$  and then  $R^T q_\ell = p_\ell$

4: Update

$$\lambda^{(\ell+1)} = \lambda^{(\ell)} - \left( \frac{\|p_\ell\|}{\|q_\ell\|} \right)^2 \left( \frac{\Delta - \|p_\ell\|}{\Delta} \right)$$

5: end for

Note: When  $\lambda^{(\ell)} < -\lambda_1$ , Cholesky factorization fails. In practice, the algorithm with safeguards usually converges in a few steps.



## Comments

The presented algorithm for solving the trust-region subproblem is a practical implementation of the ideas discussed earlier. Its main goal is to find the value of the parameter  $\lambda$  that ensures the fulfillment of the condition  $\|p(\lambda)\| = \Delta$ . To achieve this, Newton's method is applied to a specially transformed function. At each step, the algorithm performs a decomposition of the matrix  $B + \lambda I$  using the Cholesky factorization, which guarantees the numerical stability of the computations provided the corresponding matrix is positive definite. Next, a system of linear equations is solved to find the current approximation  $p_\ell$  and an auxiliary vector  $q_\ell$ . The iterative process involves adjusting the parameter  $\lambda$ , and the update step is designed to bring the norm of the found vector  $p_\ell$  closer to the given trust-region radius. It is important to understand that practical implementations of this method usually do not require driving  $\lambda$  to high precision; a few iterations are sufficient to achieve an acceptable approximation. However, the method has a limitation: if  $\lambda$  becomes less than  $-\lambda_1$ , where  $\lambda_1$  is the smallest eigenvalue of the matrix  $B$ , then the Cholesky factorization becomes impossible, and it becomes necessary to introduce special "safeguards." These protective measures prevent failures and make the algorithm applicable in practice even for ill-conditioned problems.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 4)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025



Санкт-Петербургский  
государственный  
университет



Conjugate  
Methods  
Expanding  
Subspace  
Minimization

CG —  
Algorithm

CG Method

Finite-  
Termination  
Property



34 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In this lecture, we will consider the so-called hard case arising from the application of the More-Sorensen's Theorem. The rest of the lecture is devoted to the consideration of conjugate gradient method and its practical implementations.

## The Hard Case in Trust-Region Subproblem

Assumption Relaxed: If  $q_1^T g = 0$  or eigenvalue  $\lambda_1$  is multiple with  $Q_1^T g = 0$  (where  $Q_1$  is the matrix whose columns span the subspace corresponding to the eigenvalue  $\lambda_1$ ), the root-finding fails: there may not be a value  $\lambda \in (-\lambda_1, \infty)$  s.t.  $\|p(\lambda)\| = \Delta$ . This is called the hard case.

Solution: Take  $\lambda = -\lambda_1$ , as guaranteed by Theorem 12. Use:

$$p = \sum_{\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j + \tau z$$

Where:  $(B - \lambda_1 I)z = 0$ ,  $\|z\| = 1$ ,  $z$  orthogonal to all  $q_j$  with  $\lambda_j \neq \lambda_1$ .

Then:

$$\|p\|^2 = \sum_{\lambda_j \neq \lambda_1} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2} + \tau^2$$

Choose  $\tau$  such that  $\|p\| = \Delta$ . Then  $p$  and  $\lambda = -\lambda_1$  satisfy the optimality conditions (4a)-(4c) of Theorem 12.

1/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate  
Methods  
Expanding  
Subspace  
Minimization

CG –  
Algorithm

CG Method

Finite-  
Termination  
Property



### Comments

Up to now we assumed that the condition  $q_1^T g \neq 0$ , which ensures that the root-finding equation has a solution inside the interval from  $-\lambda_1$  to infinity. However, what happens if this assumption fails? Suppose that  $q_1^T g = 0$ , or more generally, that  $\lambda_1$  is a multiple eigenvalue and the entire eigenspace associated with  $\lambda_1$  is orthogonal to  $g$ . In this situation, the root-finding approach no longer works, because there may be no value of  $\lambda$  in the open interval from  $-\lambda_1$  to infinity that makes the norm of  $p(\lambda)$  equal to  $\Delta$ .

This pathological scenario is called the hard case. At first, it might seem that no solution exists, but Moré-Sorensen's theorem guarantees that a correct solution still lies within the closed interval from  $-\lambda_1$  to infinity. In fact, there is only one possibility:  $\lambda$  must be equal to  $-\lambda_1$ .

Once  $\lambda$  is fixed at  $-\lambda_1$ , we must carefully construct the step  $p$ . We cannot simply omit the components corresponding to eigenvalue  $\lambda_1$ . Instead, we recognize that the matrix  $B - \lambda_1 I$  is singular. Therefore, there exists a vector  $z$ , of unit norm, such that  $(B - \lambda_1 I)z = 0$ . This vector  $z$  is precisely an eigenvector corresponding to the eigenvalue  $\lambda_1$ . Moreover, it is orthogonal to all eigenvectors associated with other eigenvalues.

Using this property, we can express  $p$  as the sum of two terms. The first term is the usual series over all eigenvalues not equal to  $\lambda_1$ : for each such index  $j$ , we take  $q_j^T g / (\lambda_j + \lambda)$  multiplied by  $q_j$ . The second term is  $\tau z$ , where  $\tau$  is a scalar parameter. The squared norm of  $p$  is then the sum of the squared contributions of the first part plus  $\tau^2$ . By choosing  $\tau$  appropriately, we can always ensure that the norm of  $p$  equals  $\Delta$ . This construction gives us both  $p$  and  $\lambda = -\lambda_1$  that satisfy the optimality conditions of the trust-region subproblem.

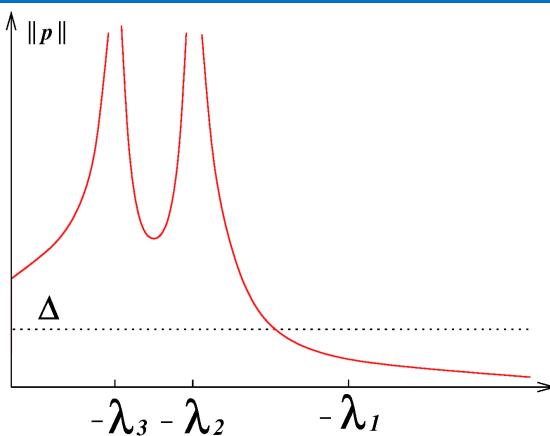


Figure: The hard case:  $\|p(\lambda)\| < \Delta$  for all  $\lambda \in (-\lambda_1, \infty)$ .

- ▶  $\|p(\lambda)\| < \Delta$  for all  $\lambda \in (-\lambda_1, \infty)$ .
- ▶ No  $\lambda$  solves  $\|p(\lambda)\| = \Delta$  in this interval.
- ▶ Occurs when  $Q_1^T g = 0$  for eigenspace  $Q_1$  of  $\lambda_1$ .

Conjugate  
Methods  
  
 Expanding  
Subspace  
Minimization  
  
 CG –  
Algorithm  
  
 CG Method  
  
 Finite-  
Termination  
Property



### Comments

The hard case can also be visualized graphically. Recall that in the standard situation, as  $\lambda$  increases from values close to  $-\lambda_1$ , the norm of  $p(\lambda)$  eventually decreases and crosses the trust-region radius  $\Delta$ . That crossing point defines  $\lambda^*$ . But in the hard case, this crossing never happens.

Specifically, for every  $\lambda$  strictly greater than  $-\lambda_1$ , the computed step norm remains strictly less than  $\Delta$ . In other words, the curve representing the norm of  $p(\lambda)$  lies entirely below the horizontal line at height  $\Delta$ , throughout the entire interval from  $-\lambda_1$  to infinity. As a consequence, there is no root of the equation  $\|p(\lambda)\| = \Delta$  within this range.

This behavior occurs exactly when the projection of  $g$  onto the eigenspace corresponding to the smallest eigenvalue  $\lambda_1$  vanishes. Equivalently, the subspace spanned by the eigenvectors for  $\lambda_1$  is orthogonal to  $g$ . In this case, the search direction never accumulates enough length to reach the boundary of the trust region, no matter how large  $\lambda$  becomes.

Nevertheless, the theory assures us that the trust-region problem still has a valid solution. The solution is obtained by setting  $\lambda = -\lambda_1$  and then augmenting the step with a suitable component along the eigenvector  $z$  of that eigenvalue. By adjusting the scalar  $\tau$ , as discussed earlier, we can inflate the norm of  $p$  to exactly  $\Delta$ .

This picture underscores the importance of theoretical guarantees in numerical optimization. Even when the intuitive root-finding approach fails completely, the structure of the problem ensures that a solution always exists. The so-called hard case is therefore not a breakdown of the trust-region method, but rather a reminder that careful handling of eigenvalue structure is essential in practical algorithms.



We derive the conjugate gradient method and discuss its convergence. For simplicity, we drop the qualifier “linear” throughout.

The conjugate gradient method is an iterative method for solving the linear system:

$$Ax = b,$$

where  $A$  is a  $n \times n$  symmetric positive definite.

The system is equivalent to the minimization problem

$$\min \phi(x) \stackrel{\text{def}}{=} \frac{1}{2}x^T Ax - b^T x,$$

which has the same unique solution.

The gradient of  $\phi$  equals the residual of the linear system:

$$\nabla \phi(x) = Ax - b \stackrel{\text{def}}{=} r(x),$$

so at  $x = x_k$  we have:

$$r_k = Ax_k - b.$$

## Comments

The conjugate gradient method is one of the most important iterative algorithms for solving systems of linear equations of the form  $Ax = b$ , where  $A$  is a symmetric positive definite matrix of size  $n \times n$ . Such matrices arise frequently in optimization, differential equations, and engineering problems. Instead of attempting to solve this system directly by factorization, the conjugate gradient method reformulates the task as a minimization problem. Specifically, we define a quadratic function  $\phi(x) = \frac{1}{2}x^T Ax - b^T x$ . This quadratic function is strictly convex because the matrix  $A$  is positive definite, and therefore it possesses a unique minimizer. The remarkable fact is that the minimizer of  $\phi$  coincides exactly with the solution of the linear system  $Ax = b$ .

To advance further, we note the connection between the gradient of  $\phi$  and the residual of the linear system. Computing the derivative gives  $\nabla \phi(x) = Ax - b$ , which we denote as  $r(x)$ . Thus, the residual vector  $r$  indicates how far our current iterate  $x$  is from solving the system. In the iterative process, at each step  $k$ , the residual is defined as  $r_k = Ax_k - b$ . This close relationship between optimization and linear algebra is the foundation of the conjugate gradient method. The algorithm will use properties of residuals and special search directions to reach the exact solution in at most  $n$  steps.



Any such set is linearly independent.

**Key Property:** A quadratic function  $\phi(x)$  can be minimized in at most  $n$  steps along conjugate directions.

Given an initial point  $x_0 \in \mathbb{R}^n$  and a set of conjugate directions  $\{p_0, \dots, p_{n-1}\}$ , define the iterates

$$x_{k+1} = x_k + \alpha_k p_k,$$

where the step length  $\alpha_k$  is the one-dimensional minimizer of the quadratic function  $\phi(\cdot)$  along  $x_k + \alpha p_k$ , given by

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}, \quad r_k = Ax_k - b.$$

4/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

To design the method, we must introduce the concept of conjugate directions. A set of nonzero vectors  $\{p_0, p_1, \dots, p_l\}$  is said to be conjugate with respect to the matrix  $A$  if  $p_i^T A p_j = 0$  whenever  $i \neq j$ . This condition generalizes the notion of orthogonality. While ordinary orthogonality uses the Euclidean inner product, conjugacy uses the inner product defined by the positive definite matrix  $A$ . Importantly, any conjugate set of vectors is automatically linearly independent, and therefore it can span the entire space when its size reaches  $n$ .

The key property is striking: if we minimize a quadratic function  $\phi$  along conjugate directions, we obtain the exact solution in at most  $n$  steps. The procedure begins from an initial guess  $x_0 \in \mathbb{R}^n$ , together with a chosen sequence of conjugate directions  $\{p_0, \dots, p_{n-1}\}$ . At each iteration, we update the iterate according to  $x_{k+1} = x_k + \alpha_k p_k$ . The step length  $\alpha_k$  is determined by exact minimization of the quadratic function along this one-dimensional line. A straightforward calculation shows that  $\alpha_k = -(r_k^T p_k) / (p_k^T A p_k)$ , where  $r_k = Ax_k - b$ . This formula guarantees that each step makes the residual orthogonal to the chosen direction, ensuring progress toward the minimizer.

Conjugate Methods
Expanding Subspace Minimization
CG – Algorithm
CG Method
Finite-Termination Property



### Theorem 16

For any  $x_0 \in \mathbb{R}^n$ , the sequence  $\{x_k\}$  generated by the conjugate direction algorithm  $x_{k+1} = x_k + \alpha_k p_k$ , with  $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$ ,  $r_k = Ax_k - b$ , converges to the solution  $x^*$  of the linear system  $Ax = b$  in at most  $n$  steps.

**Proof:** Since the directions  $\{p_i\}$  are linearly independent, they span  $\mathbb{R}^n$ . Therefore,

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \cdots + \sigma_{n-1} p_{n-1}$$

for some scalars  $\sigma_k$ . Premultiplying by  $p_k^T A$  and using conjugacy yields

$$\sigma_k = \frac{p_k^T A (x^* - x_0)}{p_k^T A p_k}.$$

We now show that these coefficients  $\sigma_k$  equal the step sizes  $\alpha_k$  from the algorithm.

### Comments

The convergence of the conjugate direction method can now be established formally. The following theorem holds. Under our assumptions, for any initial approximation, the conjugate directions method converges to the solution in no more than  $n$  steps. To be precise, suppose we begin from any initial point  $x_0$ . Because the set of directions  $\{p_0, \dots, p_{n-1}\}$  is linearly independent, it spans the entire space  $\mathbb{R}^n$ . Consequently, the difference between the exact solution  $x^*$  and the initial guess  $x_0$  can be expressed as a linear combination:  $x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \cdots + \sigma_{n-1} p_{n-1}$ . Here,  $\sigma_k$  are certain scalar coefficients.

To determine these coefficients, we premultiply both sides of the equation by  $p_k^T A$ . By conjugacy, all terms vanish except the  $k$ -th, which yields  $\sigma_k = \frac{p_k^T A (x^* - x_0)}{p_k^T A p_k}$ . This expression shows that the exact displacement from the initial point toward the solution is uniquely determined by the geometry of the conjugate directions. The next step is to prove that these coefficients  $\sigma_k$  are precisely the step lengths  $\alpha_k$  chosen by the algorithm. If this identity holds, then the sequence of iterates constructed by the algorithm coincides with the linear decomposition of the exact solution, and hence the method must terminate in at most  $n$  steps.

Conjugate Methods
Expanding Subspace Minimization
CG – Algorithm
CG Method

**Finite-Termination Property**



Since the algorithm generates

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1},$$

premultiplying by  $p_k^T A$  and using conjugacy yields

$$p_k^T A(x_k - x_0) = 0.$$

Thus,

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = -p_k^T r_k.$$

Comparing this with  $\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$  and  $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$ , we obtain  $\sigma_k = \alpha_k$ , completing the proof. □

## Comments

After  $k$  iterations, the current point has the form  $x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_{k-1} p_{k-1}$ . Premultiplying by  $p_k^T A$  and using conjugacy shows that  $p_k^T A(x_k - x_0) = 0$ . This implies  $p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k)$ . Rewriting the last term gives  $p_k^T (b - Ax_k) = -p_k^T r_k$ .

Comparing this identity with the earlier expression  $\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$ , and with the definition  $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$ , we see that  $\sigma_k = \alpha_k$ . Therefore, the coefficients in the linear representation of the exact solution coincide with the actual step lengths computed by the method. This equality completes the proof of convergence: the conjugate direction method always produces the exact solution in at most  $n$  iterations. This result is remarkable because it provides both an algorithm and a theoretical guarantee of finite termination.

Conjugate Methods
Expanding Subspace Minimization
CG – Algorithm
CG Method
Finite-Termination Property

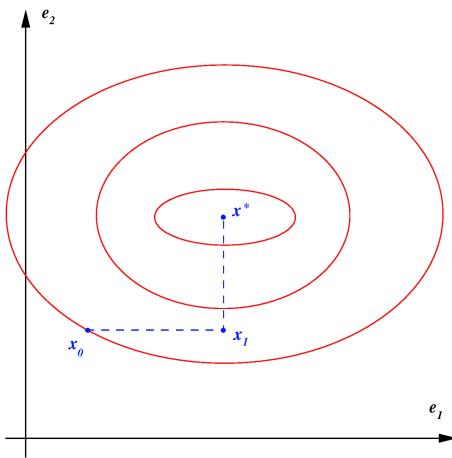


Figure: Successive minimizations along coordinate directions solve the problem in n steps.

If  $A$  is diagonal, the level sets of  $\phi(\cdot)$  are axis-aligned ellipses. Minimization along coordinate directions yields the solution in n iterations.

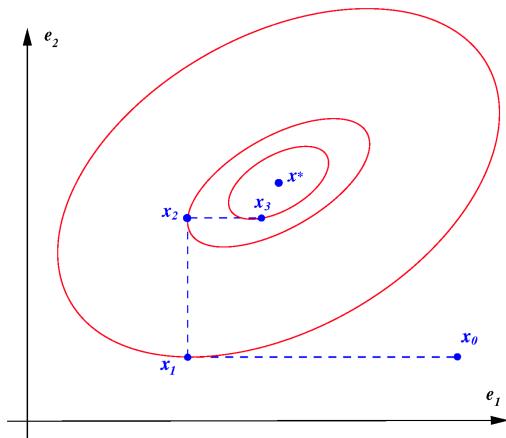
7/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches



### Comments

There is a simple interpretation of the properties of conjugate directions when the matrix  $A$  is diagonal. In this situation, the quadratic function  $\phi(x) = \frac{1}{2}x^T Ax - b^T x$  has level sets that are ellipses aligned with the coordinate axes. This alignment means that if we minimize the function one coordinate at a time, we move directly toward the exact minimizer. More precisely, in each step, we perform a line minimization along a coordinate direction, such as  $e_1, e_2, \dots, e_n$ . Each such step recovers the exact value of the corresponding component of the solution vector  $x^*$ . After  $n$  steps, the full minimizer is obtained. This property reflects the separability of the problem: since  $A$  is diagonal, there are no cross terms linking different variables, and each coordinate can be optimized independently. Thus, in the diagonal case, a naive coordinate descent method solves the problem in exactly  $n$  iterations. While this may appear trivial, it provides the motivation for constructing methods that preserve this efficiency even when  $A$  is not diagonal.

Conjugate Methods
Expanding Subspace Minimization
CG — Algorithm
CG Method
Finite-Termination Property



**Figure:** Successive minimizations along coordinate directions fail for nondiagonal Hessians.

When  $A$  is not diagonal, the level sets of  $\phi(\cdot)$  are not aligned with coordinate directions. Coordinate-wise minimization does not reach the solution in  $n$  steps.

8/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches



## Comments

When the matrix  $A$  is not diagonal, the picture changes drastically. The quadratic function  $\phi$  still has elliptical contours, but these ellipses are now tilted with respect to the coordinate axes. The axes of the ellipses are determined by the eigenvectors of  $A$ , not by the coordinate directions. If we insist on minimizing along coordinate directions one after another, the procedure no longer converges in  $n$  steps. In fact, the method may zigzag endlessly, never hitting the solution exactly. This happens because the coordinates are no longer independent; cross terms in the quadratic couple the variables together. As a result, optimizing one coordinate while ignoring the others disturbs the progress we have already made. Therefore, the naive coordinate descent method loses its remarkable efficiency as soon as  $A$  contains off-diagonal elements. This observation motivates us to look for new directions, ones that are not necessarily aligned with the coordinate axes but still allow us to systematically eliminate the error component by component. This is the starting point for the theory of conjugate directions.



We can diagonalize A by transforming variables using the conjugate directions:

$$\hat{x} = S^{-1}x, \quad S = [p_0 \ p_1 \ \dots \ p_{n-1}]$$

$$\text{Then } \hat{\phi}(\hat{x}) = \phi(S\hat{x}) = \frac{1}{2}\hat{x}^T(S^TAS)\hat{x} - (S^Tb)^T\hat{x}$$

- $S^TAS$  is diagonal by A-conjugacy  $\Rightarrow$  we minimize  $\hat{\phi}$  via n univariate steps.
- The ith direction in  $\hat{x}$ -space corresponds to  $p_i$  in  $x$ -space.
- Hence, this coordinate search is equivalent to the conjugate direction algorithm.

**Conclusion:** Conjugate direction method solves the problem in at most n steps.

## Comments

The key idea to overcome this difficulty is to construct a new set of directions, called conjugate directions, with respect to the matrix A. Suppose we gather such directions  $p_0, p_1, \dots, p_{n-1}$ , and arrange them as the columns of a matrix S. Then we define a change of variables by setting  $\hat{x} = S^{-1}x$ . In this new variable space, the quadratic function  $\phi(S\hat{x})$  becomes  $\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T(S^TAS)\hat{x} - (S^Tb)^T\hat{x}$ . By construction, the matrix  $S^TAS$  is diagonal, because the directions  $p_i$  are A-conjugate. Therefore, in the transformed space, the minimization problem reduces exactly to the simple diagonal case discussed earlier: we need only n one-dimensional minimizations along the coordinate directions of  $\hat{x}$ . Each coordinate direction in the transformed space corresponds to a conjugate direction in the original space. Thus, the procedure of coordinate minimization in the transformed coordinates is equivalent to moving along conjugate directions in the original coordinates. Once again, we conclude that the conjugate direction algorithm finds the exact minimizer in no more than n steps.

## Expanding Subspace Minimization

**Note:** If  $A$  is diagonal, each coordinate step recovers one component of the minimizer. After  $k$  steps, the quadratic is minimized over  $\text{span}\{e_1, \dots, e_k\}$ .

We will use that the residuals satisfy the recurrence:

$$r_{k+1} = b - Ax_{k+1} = \underbrace{b - A(x_k + \alpha_k p_k)}_{r_k} = r_k + \alpha_k A p_k.$$

### Theorem 17(Expanding Subspace Minimization)

Let  $x_0 \in \mathbb{R}^n$  be any starting point, and suppose that the sequence  $\{x_k\}$  is generated by the algorithm  $x_{k+1} = x_k + \alpha_k p_k$ , where  $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$ .

Then the residual  $r_k = b - Ax_k$  satisfies  $r_k^T p_i = 0$ , for  $i = 0, 1, \dots, k-1$ , and  $x_k$  is the minimizer of the quadratic function

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x \text{ over the affine subspace}$$

$$\{x \mid x = x_0 + \text{span}\{p_0, p_1, \dots, p_{k-1}\}\}.$$

Conjugate Methods

Expanding Subspace Minimization

CG – Algorithm

CG Method

Finite-Termination Property



### Comments

There is another perspective that further clarifies the efficiency of conjugate direction methods. In the diagonal case, each coordinate step correctly recovers one component of the solution. After  $k$  steps, the quadratic has been minimized over the span of the first  $k$  coordinate directions. A similar property holds in the general case. Suppose we start from an arbitrary point  $x_0$  and generate iterates by the rule  $x_{k+1} = x_k + \alpha_k p_k$ , where  $\alpha_k$  is given by  $-\frac{r_k^T p_k}{p_k^T A p_k}$ . Here  $r_k$  denotes the residual, defined as  $b - Ax_k$ . Then we can verify a recurrence for the residuals:  $r_{k+1} = r_k + \alpha_k A p_k$ . Theorem 17 states that the residual  $r_k$  is orthogonal to all previous search directions, that is,  $r_k^T p_i = 0$  for all  $i = 0, 1, \dots, k-1$ . Moreover, the current iterate  $x_k$  is the minimizer of  $\phi$  over the affine subspace consisting of  $x_0 + \text{span}\{p_0, \dots, p_{k-1}\}$ . In other words, after  $k$  steps, we have minimized  $\phi$  over a  $k$ -dimensional subspace generated by the search directions. Each new step expands this subspace, and after at most  $n$  steps, the entire space is covered and the exact minimizer is obtained. This expanding subspace property provides the geometric intuition behind the finite termination of conjugate direction methods.

## Proof of Theorem 17

**Proof:** We begin by showing that a point  $\tilde{x}$  minimizes  $\phi$  over the set

$$\{x \mid x = x_0 + \text{span}\{p_0, p_1, \dots, p_{k-1}\}\}$$

if and only if  $r(\tilde{x})^T p_i = 0$  for all  $i = 0, 1, \dots, k - 1$ .

Let

$$h(\sigma) = \phi\left(x_0 + \sum_{i=0}^{k-1} \sigma_i p_i\right), \quad \sigma \in \mathbb{R}^k.$$

This function is a strictly convex quadratic and hence has a unique minimizer  $\sigma^*$  such that

$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, \quad i = 0, 1, \dots, k - 1.$$

Using the chain rule, this yields

$$\nabla \phi(x_0 + \sum_{i=0}^{k-1} \sigma_i^* p_i)^T p_i = 0, \quad i = 0, 1, \dots, k - 1,$$

and hence

$$r(\tilde{x})^T p_i = 0, \quad \text{where } \tilde{x} = x_0 + \sum_{i=0}^{k-1} \sigma_i^* p_i.$$

Conjugate Methods

Expanding Subspace Minimization

CG – Algorithm

CG Method  
Finite-Termination Property



### Comments

Let us carefully examine the reasoning behind the proof. The idea is that if we want to minimize our quadratic function, denoted as  $\phi$ , over a subspace spanned by certain search directions, the necessary and sufficient condition for optimality is that the residual at the minimizer must be orthogonal to all those search directions. In more detail, consider any point that can be expressed as the initial vector  $x_0$  plus a linear combination of  $p_0$  through  $p_{k-1}$ . We introduce a new function, called  $h(\sigma)$ , which takes the coefficients  $\sigma$  and evaluates  $\phi$  at this linear combination. Since  $\phi$  is a strictly convex quadratic, the function  $h$  is also strictly convex, which ensures the existence of a unique minimizer. The unique minimizer  $\sigma^*$  satisfies the condition that the partial derivative of  $h$  with respect to each  $\sigma_i$  equals zero. Applying the chain rule, this requirement translates into the gradient of  $\phi$  at the minimizing point being orthogonal to every search direction  $p_i$ . By the definition of the residual, which is  $\nabla \phi(x)$ , this condition is exactly that  $r(\tilde{x})^T p_i = 0$  for all indices  $i$  between 0 and  $k - 1$ . This is a very important property: the residual is not arbitrary, but constrained to be orthogonal to the entire span of previous directions. Thus, the geometry of the problem links minimization along subspaces with orthogonality relations between residuals and search directions. This reasoning forms the foundation of the conjugate gradient method, because it ensures that by constructing the sequence of points in this way, we always maintain orthogonality, which is essential for convergence.

## Proof of Theorem 17 (continued)

We now use induction to show that  $x_k$  satisfies

$$r_k^T p_i = 0, \quad \text{for all } i = 0, 1, \dots, k-1.$$

For the case  $k = 1$ , since  $x_1 = x_0 + \alpha_0 p_0$  minimizes  $\phi$  along  $p_0$ , we have

$$r_1^T p_0 = 0.$$

Assume inductively that

$$r_{k-1}^T p_i = 0 \quad \text{for } i = 0, 1, \dots, k-2.$$

Using

$$r_k = r_{k-1} + \alpha_{k-1} A p_{k-1},$$

we obtain

$$p_{k-1}^T r_k = p_{k-1}^T r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0,$$

by the definition of  $\alpha_{k-1}$ .

For  $i = 0, 1, \dots, k-2$ , we have

$$p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1} = 0,$$

since the first term vanishes by the induction hypothesis and the second by conjugacy of  $\{p_i\}$ . We have shown that  $r_k^T p_i = 0$ , for  $i = 0, 1, \dots, k-1$ , thus the theorem is proved.  $\square$

12/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate Methods

Expanding Subspace Minimization

CG – Algorithm

CG Method

Finite-Termination Property



### Comments

Now we extend the previous result by proving that this orthogonality is preserved at every iteration. We use mathematical induction. For the base case, when  $k = 1$ , the update  $x_1 = x_0 + \alpha_0 p_0$  minimizes  $\phi$  along direction  $p_0$ . This immediately implies that the residual  $r_1$  is orthogonal to  $p_0$ . Next, assume that for some step  $k - 1$ , the residual  $r_{k-1}$  is orthogonal to all earlier directions  $p_0$  through  $p_{k-2}$ . This is our induction hypothesis. Then, using the recurrence relation for residuals, namely  $r_k = r_{k-1} + \alpha_{k-1} A p_{k-1}$ , we analyze the orthogonality conditions. First, when we take the dot product of  $r_k$  with  $p_{k-1}$ , the formula simplifies to zero because of the definition of the step length  $\alpha_{k-1}$ . Second, when we consider  $p_i$  with index less than  $k - 1$ , the inner product  $p_i^T r_k$  splits into two parts:  $p_i^T r_{k-1}$ , which vanishes by the induction assumption, and  $\alpha_{k-1} p_i^T A p_{k-1}$ , which vanishes because the directions are conjugate. Therefore,  $r_k$  is orthogonal to all  $p_i$  for  $i = 0, 1, \dots, k-1$ . This completes the induction and confirms that every new residual maintains orthogonality with all previous search directions. This recursive orthogonality is not just a neat property—it is the backbone of why the method works. It ensures that each step eliminates error components in new independent directions, so after a finite number of steps, the method can reach the exact solution in exact arithmetic.

Conjugate Methods

Expanding Subspace Minimization

CG – Algorithm

CG Method

Finite-Termination Property



- ▶ Each new direction  $p_k$  is built from only  $p_{k-1}$  and  $r_k$ .
- ▶ Conjugacy to all previous directions is automatic.
- ▶ Requires low memory and computation.

Direction update:

$$p_k = -r_k + \beta_k p_{k-1},$$

where the scalar  $\beta_k$  is determined by requiring  $p_{k-1}^T A p_k = 0$ . Premultiplying the update by  $p_{k-1}^T A$  yields:

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}.$$

Initial direction:

$$p_0 = -r_0.$$

## Comments

We now shift from general properties to a very practical insight of the conjugate gradient method: it does not need to store or recompute all the previous directions. Instead, each new direction  $p_k$  is built from only two ingredients: the current residual  $r_k$  and the immediately preceding direction  $p_{k-1}$ . This is known as a two-term recurrence. The advantage is striking: the new direction automatically remains conjugate to all earlier directions, even though we never explicitly refer to them. This fact keeps memory requirements extremely low and computational effort minimal. To be specific, the update formula is  $p_k = -r_k + \beta_k p_{k-1}$ . Here,  $\beta_k$  is a scalar that ensures conjugacy with the previous direction. We find  $\beta_k$  by requiring that the inner product of  $p_{k-1}^T A p_k$  equals zero. Solving this condition gives  $\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$ . Notice the simplicity: all the information comes from the current residual and the last search direction. The very first direction is chosen as  $-r_0$ , which is the steepest descent direction at the starting point. This two-term recurrence is the key that makes the conjugate gradient method both elegant and efficient. Without it, we would have to store a growing set of vectors and perform costly orthogonalizations, but with it the method requires only minimal updates and yet guarantees the full conjugacy property.

## Conjugate Gradient: Preliminary Version

**Goal:** Perform 1D minimizations along A-conjugate directions defined via recurrence using residuals.

**Algorithm 8** (CG — Preliminary Version):

```
1: given  $x_0$ 
2:  $r_0 \leftarrow Ax_0 - b$ ,  $p_0 \leftarrow -r_0$ 
3:  $k \leftarrow 0$ 
4: while  $r_k \neq 0$  do
5:    $\alpha_k \leftarrow -(r_k^T p_k) / (p_k^T A p_k)$ 
6:    $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
7:    $r_{k+1} \leftarrow Ax_{k+1} - b$ 
8:    $\beta_{k+1} \leftarrow (r_{k+1}^T A p_k) / (p_k^T A p_k)$ 
9:    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$ 
10:   $k \leftarrow k + 1$ 
11: end while
```

Note: This version is useful for studying the essential properties of the conjugate gradient method; efficiency improvements follow in next version.

Conjugate Methods

Expanding Subspace Minimization

CG — Algorithm

CG Method

Finite-Termination Property



### Comments

We now have all the ingredients to describe the conjugate gradient method as an explicit algorithm. The procedure starts from an initial guess  $x_0$ . We compute the initial residual  $r_0 = Ax_0 - b$ , and set the first search direction  $p_0 = -r_0$ . Then, iteration begins. At each step, we compute a step length  $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$ . This formula guarantees minimization of  $\phi$  along the direction  $p_k$ . Next, we update the solution:  $x_{k+1} = x_k + \alpha_k p_k$ . We also update the residual:  $r_{k+1} = Ax_{k+1} - b$ . Then, to build the new search direction, we compute  $\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$ , and define  $p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$ . Finally, we increment  $k$  and repeat the process while the residual is nonzero. This structure embodies everything we have established: orthogonality of residuals, conjugacy of directions, and the use of the two-term recurrence. It is worth emphasizing that this preliminary version of the algorithm is primarily pedagogical: it reveals the essential mechanics of conjugate gradients. Later refinements make the method more efficient and numerically stable. But even at this stage, the method is already extremely powerful, since in exact arithmetic it can find the exact solution in at most  $n$  steps for an  $n$ -dimensional system.



Note:

- ▶ The directions  $p_0, p_1, \dots, p_{n-1}$  are conjugate.
- ▶ The residuals  $r_i$  are mutually orthogonal.
- ▶ Each residual  $r_k$  and direction  $p_k$  lies in the Krylov subspace of degree  $k$  for  $r_0$ , defined as.  $\mathcal{K}(r_0; k) \stackrel{\text{def}}{=} \text{span}\{r_0, Ar_0, \dots, A^k r_0\}$ .

## Theorem 18

Suppose that the  $k$ -th iterate generated by the conjugate gradient method is not the solution point  $x^*$ . The following four properties hold:

$$\begin{aligned} r_k^T r_i &= 0, \quad \text{for } i = 0, 1, \dots, k-1; \\ \text{span}\{r_0, r_1, \dots, r_k\} &= \text{span}\{r_0, Ar_0, \dots, A^k r_0\}; \\ \text{span}\{p_0, p_1, \dots, p_k\} &= \text{span}\{r_0, Ar_0, \dots, A^k r_0\}; \\ p_k^T A p_i &= 0, \quad \text{for } i = 0, 1, \dots, k-1. \end{aligned}$$

Therefore, the sequence  $\{x_k\}$  converges to  $x^*$  in at most  $n$  steps.

## Comments

At this stage, we can summarize the central structural properties of the conjugate gradient method. The method simultaneously produces two sequences of vectors: the residuals and the search directions. The residuals, denoted by  $r_k$ , represent the current gradient information, and a remarkable fact is that these residuals are mutually orthogonal. This means that each residual points in a direction completely independent of all the previous ones. In parallel, the search directions, denoted by  $p_k$ , are conjugate with respect to the matrix  $A$ , meaning that the inner product of  $p_i^T A p_j$  equals zero whenever  $i \neq j$ . These properties guarantee that the algorithm never revisits the same error component twice. Another fundamental idea is that both the residuals and the search directions lie in Krylov subspaces. More precisely, each search direction  $p_k$  and residual  $r_k$  belong to the Krylov subspace generated by the initial residual  $r_0$ , which is the span of  $r_0, Ar_0, A^2 r_0, \dots, A^k r_0$ . This nested sequence of subspaces provides the geometric framework for the method. Because the dimension of the Krylov subspaces can increase at most by one at each iteration, and because the search directions are linearly independent, the method must terminate in at most  $n$  steps for an  $n$ -dimensional system. The orthogonality of residuals, the conjugacy of directions, and the structure of Krylov subspaces together explain the efficiency of the conjugate gradient method: it is both memory-efficient and theoretically optimal in exact arithmetic.

## Proof of Theorem 18: Base Case and Induction Hypothesis

**Proof:** The proof proceeds by induction.

For  $k = 0$ , the relations

$$\text{span}\{r_0\} = \text{span}\{r_0\}, \quad \text{span}\{p_0\} = \text{span}\{r_0\}$$

hold trivially.

The conjugacy condition (for  $k = 1$ )

$$p_1^T A p_0 = -r_1^T A p_0 + \beta_1 p_0^T A p_0 = 0$$

holds by construction of  $\beta_1 = \frac{r_1^T A p_0}{p_0^T A p_0}$ .

Assume the following for some  $k$ :

$$r_k \in \text{span}\{r_0, Ar_0, \dots, A^k r_0\} \text{ and } p_k \in \text{span}\{r_0, Ar_0, \dots, A^k r_0\}$$

Then (by multiplying the second of these expressions by  $A$ ) we obtain:

$$Ap_k \in \text{span}\{Ar_0, \dots, A^{k+1} r_0\}, \quad r_{k+1} = r_k + \alpha_k Ap_k \in \text{span}\{r_0, Ar_0, \dots, A^{k+1} r_0\}$$

Hence,

$$\text{span}\{r_0, \dots, r_{k+1}\} \subset \text{span}\{r_0, Ar_0, \dots, A^{k+1} r_0\}$$

Conjugate Methods  
Expanding Subspace Minimization

CG — Algorithm

CG Method  
Finite-Termination Property



### Comments

The proof of these properties is naturally based on induction. We begin with the base case. For  $k = 0$ , the span of the residual set is simply the span of  $r_0$ , and similarly, the span of the initial search direction is also the span of  $r_0$ . These relations are trivial but essential as a starting point. For  $k = 1$ , the conjugacy condition between  $p_0$  and  $p_1$  follows directly from the construction of  $\beta_1$ . Recall that  $\beta_1$  is defined as the ratio of  $r_1^T A p_0$  divided by  $p_0^T A p_0$ . Substituting this into the recurrence relation for the search direction  $p_1$ , we find that the inner product of  $p_1^T A p_0$  equals zero, which establishes conjugacy at the first step. With this foundation, the induction hypothesis assumes that for some index  $k$ , both  $r_k$  and  $p_k$  belong to the Krylov subspace of degree  $k$ . That is, they can be written as linear combinations of  $r_0, Ar_0, \dots, A^k r_0$ . Multiplying the second inclusion by the matrix  $A$ , we see that  $Ap_k$  lies in the Krylov subspace of degree  $k + 1$ . Using the recurrence relation for residuals, namely  $r_{k+1} = r_k + \alpha_k Ap_k$ , we then conclude that  $r_{k+1}$  also belongs to the Krylov subspace of degree  $k + 1$ . This establishes the forward inclusion and prepares the ground for the reverse inclusion that completes the induction argument.

## Proof of Theorem 18 (continued)

To prove the reverse inclusion

$$\text{span}\{r_0, Ar_0, \dots, A^{k+1}r_0\} \subset \text{span}\{r_0, r_1, \dots, r_{k+1}\}$$

we use the induction assumption to deduce that:

$$A^{k+1}r_0 = A(A^k r_0) \in \text{span}\{Ap_0, Ap_1, \dots, Ap_k\}$$

From the update rule ( $r_{k+1} = r_k + \alpha_k Ap_k$ ) we have

$$Ap_i = \frac{r_{i+1} - r_i}{\alpha_i}, \quad i = 0, 1, \dots, k$$

and we conclude:

$$A^{k+1}r_0 \in \text{span}\{r_0, r_1, \dots, r_{k+1}\}$$

By combining this expression with the induction hypothesis for second conditions we obtain

$$\text{span}\{r_0, Ar_0, \dots, A^{k+1}r_0\} \subset \text{span}\{r_0, r_1, \dots, r_{k+1}\}$$

and equality holds (when  $k$  is replaced by  $k + 1$ ).

17/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate  
Methods

Expanding  
Subspace  
Minimization

CG —  
Algorithm

CG Method

Finite-  
Termination  
Property



### Comments

The next step in the induction is to prove the reverse inclusion: that the Krylov subspace of degree  $k + 1$  is contained in the span of the first  $k + 1$  residuals. This is the crucial part because it establishes equality between the two spans. To achieve this, we use the induction assumption that  $A^k r_0$  belongs to the span of  $Ap_0, Ap_1, \dots, Ap_k$ . Multiplying by  $A$ , we obtain that  $A^{k+1}r_0$  lies in the span of  $Ap_0, Ap_1, \dots, Ap_k$ . The update formula for residuals, which is  $r_{i+1} = r_i + \alpha_i Ap_i$ , can be rearranged to give  $Ap_i = \frac{r_{i+1} - r_i}{\alpha_i}$ . This shows that each  $Ap_i$  lies in the span of residuals. Therefore,  $A^{k+1}r_0$  is also a linear combination of residuals  $r_0, r_1, \dots, r_{k+1}$ . By combining this result with the earlier inclusion, we establish that the span of the first  $k + 1$  residuals is equal to the Krylov subspace of degree  $k + 1$ . This completes the induction step for the residuals.

## Proof of Theorem 18 (continued)

To show the relation

$$\text{span}\{p_0, \dots, p_{k+1}\} = \text{span}\{r_0, Ar_0, \dots, A^{k+1}r_0\}$$

we use:

$$\begin{aligned} & \text{span}\{p_0, \dots, p_k, p_{k+1}\} \\ &= \text{span}\{p_0, \dots, p_k, r_{k+1}\} \quad (\text{since } p_{k+1}^T = -r_{k+1}^T + \beta_{k+1}p_k^T) \\ &= \text{span}\{r_0, Ar_0, \dots, A^k r_0, r_{k+1}\} \quad (\text{induction}) \\ &= \text{span}\{r_0, \dots, r_{k+1}\} \\ &= \text{span}\{r_0, Ar_0, \dots, A^{k+1}r_0\} \quad (\text{from previous step}) \end{aligned}$$

To prove conjugacy for  $p_{k+1}$  we multiply  $p_{k+1}^T = -r_{k+1}^T + \beta_{k+1}p_k^T$  by  $A p_i$ :

$$p_{k+1}^T A p_i = -r_{k+1}^T A p_i + \beta_{k+1} p_k^T A p_i, \quad i = 0, \dots, k$$

When  $i = k$ , the right-hand side is zero by the definition of  $\beta_{k+1}$ .

For  $i < k$ , by induction hypothesis ( $A p_i = \frac{r_{i+1} - r_i}{\alpha_i}$ ) and by Theorem 17 we get:

$$r_{k+1}^T p_i = 0, \quad i = 0, \dots, k.$$

Conjugate Methods  
Expanding Subspace Minimization

CG — Algorithm

CG Method

Finite-Termination Property



### Comments

Once the relationship between residuals and Krylov subspaces is established, we can turn to the search directions. The goal is to show that the span of the first  $k+1$  search directions equals the Krylov subspace of degree  $k+1$ . This follows by analyzing the recurrence relation for directions. Recall that each  $p_{k+1}$  is defined as  $-r_{k+1} + \beta_{k+1}p_k$ . Hence, the span of  $p_0, \dots, p_{k+1}$  is equivalent to the span of  $p_0, \dots, p_k$  together with  $r_{k+1}$ . Using the induction assumption, we know that the span of  $p_0, \dots, p_k$  equals the span of  $r_0, \dots, r_k$ , which itself equals the Krylov subspace of degree  $k$ . Adding  $r_{k+1}$  extends this to the Krylov subspace of degree  $k+1$ . Therefore, the equality of spans holds also for the directions. The second property to establish is conjugacy. To verify that  $p_{k+1}$  is conjugate to all previous directions, we compute the inner product of  $p_{k+1}^T A p_i$  for  $i \leq k$ . Expanding this using the recurrence relation, we obtain two terms:  $-r_{k+1}^T A p_i + \beta_{k+1} p_k^T A p_i$ . For  $i = k$ , this vanishes by the definition of  $\beta_{k+1}$ . For  $i < k$ , both terms vanish by the induction hypothesis, since the residuals are orthogonal and the previous directions are already conjugate.

## Proof of Theorem 18 (continued)

For  $i = 0, \dots, k-1$ , the induction gives:

$$\begin{aligned} Ap_i &\in A \text{ span}\{r_0, Ar_0, \dots, A^i r_0\} \\ &= \text{span}\{Ar_0, \dots, A^{i+1} r_0\} \subset \text{span}\{r_0, Ar_0, \dots, A^{i+1} r_0\} \\ &= \text{span}\{p_0, \dots, p_{i+1}\} \end{aligned}$$

Thus,

$$r_{k+1}^T A p_i = 0, \quad i = 0, \dots, k-1$$

Together with

$$p_k^T A p_i = 0 \text{ for } i < k \text{ (because of the induction hypothesis)}$$

this implies that

$$p_{k+1}^T A p_i = -r_{k+1}^T A p_i + \beta_{k+1} p_k^T A p_i = 0 \text{ for } i = 0, \dots, k-1.$$

Hence,

$$p_{k+1}^T A p_i = 0, \quad i = 0, \dots, k.$$

Conjugate Methods  
Expanding Subspace Minimization

CG — Algorithm

CG Method

Finite-Termination Property



### Comments

The final step in the induction concerns verifying the remaining conjugacy conditions. For indices  $i$  strictly less than  $k$ , we must show that the inner product of  $r_{k+1}^T A p_i$  vanishes. From the induction hypothesis, we know that  $A p_i$  belongs to the span of search directions, specifically the span of  $p_0, \dots, p_{i+1}$ . Since  $r_{k+1}$  is orthogonal to all these earlier search directions, it follows that  $r_{k+1}^T A p_i = 0$ . Noting that by virtue of the induction assumption  $p_k^T A p_i = 0$  for  $i < k$ , and combining this with the last conclusions we obtain that the new direction  $p_{k+1}$  is conjugate to every previous direction (let me remind you that for  $i = k$ , this vanishes by the definition of  $\beta_{k+1}$ ). This means that the whole sequence of search directions retains pairwise  $A$ -conjugacy as the algorithm proceeds. Importantly, this guarantees that each step eliminates the error component along a new independent direction, making the method fundamentally different from simple gradient descent. While steepest descent tends to zigzag and revisit directions, the conjugate gradient method systematically constructs independent search directions, ensuring finite termination in exact arithmetic.

## Proof of Theorem 18 (final)

It follows that the directions generated by the conjugate gradient method is indeed a conjugate direction. Therefore the algorithm terminates in at most  $n$  steps.

To prove the orthogonality of the gradients, observe that

$$r_k^T p_i = 0, \quad i = 0, \dots, k-1 \text{ and any } k = 1, 2, \dots, n-1$$

and from

$$p_i = -r_i + \beta_i p_{i-1}$$

we conclude:

$$r_i \in \text{span}\{p_i, p_{i-1}\}, \quad i = 1, \dots, k-1$$

so

$$r_k^T r_i = 0, \quad i = 1, \dots, k-1$$

Finally,  $r_k^T r_0 = -r_k^T p_0 = 0$ . □

Note: This result relies on the choice  $p_0 = -r_0$ ; in fact, the result does not hold for other choices of  $p_0$ . Since the gradients  $r_k$  are mutually orthogonal, the term “conjugate gradient method” is somewhat misleading. It is the search directions, not the gradients, that are conjugate with respect to  $A$ .

**Conjugate Methods**  
**Expanding Subspace Minimization**

**CG — Algorithm**

**CG Method**

**Finite-Termination Property**



### Comments

We can now summarize the complete result. The induction proves that every new residual belongs to the growing Krylov subspaces, that every new search direction is also within these subspaces, and that the directions are pairwise conjugate with respect to the matrix  $A$ . Therefore, the conjugate gradient method generates a conjugate sequence of directions. Since the Krylov subspaces expand by at most one dimension per iteration, and since the directions remain linearly independent, the method must terminate in at most  $n$  steps in an  $n$ -dimensional space. An additional property is that the residuals themselves are mutually orthogonal. This follows because each residual lies in the span of two successive directions, specifically  $p_i$  and  $p_{i-1}$ , and because conjugacy implies orthogonality of residuals. Thus, the residuals form an orthogonal sequence while the search directions form a conjugate sequence. It is important to notice a nuance here: the orthogonality of residuals relies crucially on the initial choice  $p_0 = -r_0$ . If we had chosen a different initial direction, the entire orthogonality result might not hold. This explains why the name “conjugate gradient method” can be slightly misleading. The gradients themselves are not conjugate; rather, they are orthogonal. It is the search directions that are conjugate with respect to the matrix  $A$ .

**Key Idea:** Using Theorems 17 and 18 we can eliminate matrix-vector products in the step size formulas using orthogonality and recurrence properties.

## Orthogonality Condition:

$$r_k^T p_i = 0, \quad \text{for } i = 0, 1, \dots, k-1$$

## Conjugate Direction Recurrence:

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$$

## Updated Step Length:

$$\text{we can replace } \alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k} \text{ by } \alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$

**From**  $\alpha_k A p_k = r_{k+1} - r_k$  and  $\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$  we obtain:

$$\beta_{k+1} = \frac{r_{k+1}^T \alpha_k A p_k}{\alpha_k p_k^T A p_k} = \frac{r_{k+1}^T (r_{k+1} - r_k)}{\alpha_k p_k^T A p_k} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

Conjugate Methods  
Expanding Subspace Minimization

CG – Algorithm

CG Method

Finite-Termination Property



## Comments

In practice, the conjugate gradient method can be expressed in a more economical form, which reduces computational effort while preserving its theoretical properties. The key idea is that many of the quantities in the original algorithm can be simplified by exploiting orthogonality of residuals and recurrence relations of search directions. For example, instead of writing the step length  $\alpha_k$  as  $-\frac{r_k^T p_k}{p_k^T A p_k}$ , we can show that it equals  $\frac{r_k^T r_k}{p_k^T A p_k}$ . This avoids extra inner products involving residuals and directions. Similarly, the recurrence for  $\beta_{k+1}$  can be simplified. Starting from the relation that  $\alpha_k A p_k = r_{k+1} - r_k$ , one can eliminate the matrix-vector product and show that  $\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ . Both simplifications make the algorithm more efficient, as they require only inner products and vector updates, which are cheap compared to matrix-vector multiplications. At the same time, the recurrence for the search directions remains unchanged: each new direction  $p_{k+1}$  is equal to  $-r_{k+1} + \beta_{k+1} p_k$ . Altogether, these formulas provide a practical and streamlined version of conjugate gradients that avoids redundant computations.

### Algorithm 9 (Conjugate Gradient Method):

```

1: given  $x_0$ 
2:  $r_0 \leftarrow Ax_0 - b$ ,  $p_0 \leftarrow -r_0$ ,  $k \leftarrow 0$ 
3: while  $r_k \neq 0$  do
4:    $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$ 
5:    $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
6:    $r_{k+1} \leftarrow r_k + \alpha_k A p_k$ 
7:    $\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ 
8:    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$ 
9:    $k \leftarrow k + 1$ 
10: end while

```

Note: At each iteration, only the last two values of the vectors  $x_k$ ,  $r_k$ , and  $p_k$  are needed; older values can be overwritten to save memory. The main cost is the matrix-vector product  $A p_k$ ; inner products and vector updates require only a small multiple of  $n$  operations. CG is suitable for large problems, as it does not modify the matrix and avoids fill-in. For small problems, direct methods are preferable due to better numerical stability.

22/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate Methods  
Expanding Subspace Minimization  
CG – Algorithm  
CG Method  
Finite-Termination Property



### Comments

This practical form of the algorithm can be summarized in a compact iterative scheme. We begin with an initial guess  $x_0$ , compute the initial residual  $r_0 = Ax_0 - b$ , and set the first search direction  $p_0 = -r_0$ . Then, at each iteration, the algorithm updates four quantities. First, the step length  $\alpha_k$  is computed as  $\frac{r_k^T r_k}{p_k^T A p_k}$ . Second, the new iterate  $x_{k+1}$  is obtained as  $x_k + \alpha_k p_k$ . Third, the residual is updated as  $r_{k+1} = r_k + \alpha_k A p_k$ . Fourth, the coefficient  $\beta_{k+1}$  is determined as  $\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ . Finally, the new search direction is defined as  $-r_{k+1} + \beta_{k+1} p_k$ . The loop continues until the residual vanishes or falls below a prescribed tolerance. An important practical remark is that at any point, only the most recent values of  $x$ ,  $r$ , and  $p$  need to be stored; earlier ones can be overwritten, which greatly reduces memory requirements. The dominant cost per iteration is the matrix-vector product of  $A$  and  $p_k$ , while inner products and vector operations are relatively inexpensive. This makes conjugate gradients particularly attractive for large-scale problems, where factorization methods would require excessive storage and introduce fill-in. For small systems, however, direct solvers may remain preferable, because they are numerically more robust.



**Key Observation:** While CG terminates in at most  $n$  iterations in exact arithmetic, it often finds the solution in many fewer iterations when  $A$ 's eigenvalues have favorable distributions.

### Polynomial Interpretation:

From the update  $x_{k+1} = x_k + \alpha_k p_k$  and Krylov subspace property:

$$x_{k+1} = x_0 + \sum_{i=0}^k \alpha_i p_i = x_0 + \sum_{j=0}^k \gamma_j A^j r_0$$

Let's define a polynomial  $P_k^*(A) = \sum_{j=0}^k \gamma_j A^j$  such that:

$$x_{k+1} = x_0 + P_k^*(A)r_0$$

### Comments

A remarkable property of the conjugate gradient method is that, although in exact arithmetic it must converge in at most  $n$  steps for an  $n$ -dimensional system, in practice it often achieves a very accurate solution much earlier. The reason lies in the distribution of eigenvalues of the system matrix  $A$ . When the eigenvalues are clustered or have certain favorable patterns, the convergence can be dramatically faster. This phenomenon can be explained through a polynomial interpretation. Each iterate can be expressed as the initial guess  $x_0$  plus a linear combination of powers of  $A$  applied to the initial residual. In other words, after  $k$  steps, the solution can be written as  $x_0 + P_k^*(A)r_0$ , where  $P_k^*$  is a polynomial of degree  $k$  in  $A$ . This viewpoint reveals that the conjugate gradient method constructs, at each iteration, the best possible polynomial transformation of the residual within the Krylov subspace. Thus, instead of simply marching forward step by step, the algorithm is implicitly filtering the spectrum of  $A$  through these polynomials, which explains why the method adapts to the eigenvalue distribution and often converges much faster than the theoretical bound suggests.

Conjugate Methods

Expanding Subspace Minimization

CG – Algorithm

CG Method

Finite-Termination Property



## Key Idea

Among all methods restricted to the Krylov subspace  $\mathcal{K}(r_0; k)$ , Algorithm 9 minimizes the A-norm distance to the solution  $x^*$  after  $k$  steps.

$$\textbf{A-norm: } \|z\|_A^2 = z^T Az$$

Using this norm and the definition of  $\phi$ , and the fact that  $x^*$  minimizes  $\phi$  it is easy to show that:

$$\frac{1}{2} \|x - x^*\|_A^2 = \frac{1}{2} (x - x^*)^T A (x - x^*) = \phi(x) - \phi(x^*)$$

## CG Optimality:

Algorithm 9 produces  $x_{k+1}$  minimizes  $\phi(x)$  and hence  $\|x - x^*\|_A^2$  over  $x_0 + \text{span}\{p_0, p_1, \dots, p_k\} = x_0 + \text{span}\{r_0, Ar_0, \dots, A^k r_0\}$

**Polynomial Characterization:**  $P_k^*$  solves the following problem:

$$\min_{P_k} \|x_0 + P_k(A)r_0 - x^*\|_A$$

where  $P_k$  ranges over all polynomials of degree  $k$ .

## Comments

The deeper sense of optimality in conjugate gradients is revealed when we measure error in the so-called A-norm. For a vector  $z$ , this norm is defined as  $\sqrt{z^T Az}$ . It arises naturally because the method minimizes the quadratic functional  $\phi(x) = \frac{1}{2}x^T Ax - b^T x$ . The exact solution  $x^*$  minimizes  $\phi$ , and the difference  $\phi(x) - \phi(x^*)$  equals  $\frac{1}{2} \|x - x^*\|_A^2$ . Therefore, minimizing  $\phi$  is equivalent to minimizing the A-norm of the error. The key result states that after  $k$  steps, the iterate  $x_{k+1}$  is the unique vector within the affine space  $x_0 + \text{span}\{r_0, Ar_0, \dots, A^k r_0\}$  that minimizes this A-norm error. Equivalently, there exists a polynomial  $P_k^*$  of degree  $k$  such that  $x_{k+1} = x_0 + P_k^*(A)r_0$ , and this polynomial is chosen to minimize the A-norm distance to  $x^*$  among all degree  $k$  polynomials. This explains why conjugate gradients are not just efficient but also optimal within the Krylov subspace framework: each step delivers the best possible approximation according to the geometry induced by the matrix  $A$ .

Conjugate Methods

Expanding Subspace Minimization

CG – Algorithm

CG Method

Finite-Termination Property



Since

$$r_0 = Ax_0 - b = Ax_0 - Ax^* = A(x_0 - x^*)$$

we have

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*) \quad (*)$$

Let  $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$  be the eigenvalues of  $A$ , and let  $v_1, v_2, \dots, v_n$  be the corresponding orthonormal eigenvectors, so that

$$A = \sum_{i=1}^n \lambda_i v_i v_i^T$$

Since the eigenvectors span  $\mathbb{R}^n$ , we can write

$$x_0 - x^* = \sum_{i=1}^n \xi_i v_i \quad (**)$$

for some coefficients  $\xi_i$ .

## Comments

A central idea in the analysis of iterative methods such as Conjugate Gradient is to express the error in terms of the eigenstructure of the matrix. The error after  $k$  iterations, relative to the true solution, can be written as a transformed version of the initial error. This transformation involves a polynomial of the matrix  $A$ . The beauty of spectral decomposition is that it allows us to diagonalize the problem conceptually: instead of thinking about the full vector space at once, we decompose the initial error into contributions along the eigenvectors of  $A$ . Each eigencomponent evolves independently, scaled by the eigenvalues and the chosen polynomial. Thus, the problem of understanding error propagation reduces to studying scalar effects of polynomials on eigenvalues. If we express the initial error as a linear combination of eigenvectors, then the coefficients in this expansion describe how strongly each eigenmode is present. The subsequent iterations dampen these modes differently depending on the spectrum of  $A$ . This perspective highlights why convergence depends not only on the condition number but also on how the initial error aligns with the eigenstructure. Some components may be eliminated faster, while others persist longer, depending on the polynomial chosen by the method. This structural insight provides the foundation for explaining the optimality of the Conjugate Gradient method.

It is easy to show that any eigenvector of  $A$  is also an eigenvector of  $P_k(A)$  for any polynomial  $P_k$ . For our particular matrix  $A$  and its eigenvalues  $\lambda_i$  and eigenvectors  $v_i$ , we have

$$P_k(A)v_i = P_k(\lambda_i)v_i, \quad i = 1, 2, \dots, n.$$

By substituting (\*\*) into (\*) we have

$$x_{k+1} - x^* = \sum_{i=1}^n [1 + \lambda_i P_k^*(\lambda_i)] \xi_i v_i.$$

By using the fact that  $\|z\|_A^2 = z^T A z = \sum_{i=1}^n \lambda_i (v_i^T z)^2$ , we have

$$\|x_{k+1} - x^*\|_A^2 = \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k^*(\lambda_i)]^2 \xi_i^2.$$

Since the polynomial  $P_k^*$  generated by the CG method is optimal with respect to this norm, we have

$$\|x_{k+1} - x^*\|_A^2 = \min_{P_k} \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k(\lambda_i)]^2 \xi_i^2.$$

Conjugate Methods  
Expanding Subspace Minimization  
CG – Algorithm  
CG Method

Finite-Termination Property



### Comments

Once the error has been decomposed along eigenvectors, an important property emerges: polynomials of the matrix act diagonally on this decomposition. That is, applying  $P_k(A)$  to an eigenvector simply scales it by  $P_k(\lambda_i)$ . This simplification means that the evolution of each error component depends only on the corresponding eigenvalue. In the context of the Conjugate Gradient method, this leads to an elegant interpretation: at each iteration, the algorithm chooses a polynomial that minimizes the  $A$ -norm of the error. The minimization is performed across all possible polynomials of a given degree, which makes the CG polynomial  $P_k^*$  optimal. The resulting expression shows that the error at iteration  $k+1$  is a weighted sum of scaled eigencomponents, where the weights depend on both the initial error and the polynomial evaluated at the eigenvalues. This framework highlights that CG does not act blindly—it systematically constructs polynomials that suppress certain eigenvectors as effectively as possible. The implication is profound: convergence is not random but guided by an optimal balance over the spectrum of the matrix. This explains why CG often converges much faster than simple bounds suggest, particularly when eigenvalues are clustered.

By extracting the largest of the terms  $[1 + \lambda_i P_k(\lambda_i)]^2$ , we obtain

$$\|x_{k+1} - x^*\|_A^2 \leq \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \left( \sum_{j=1}^n \lambda_j \xi_j^2 \right) \\ = \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \|x_0 - x^*\|_A^2,$$

where we used the fact that

$$\|x_0 - x^*\|_A^2 = \sum_{j=1}^n \lambda_j \xi_j^2.$$

The expression above allows us to quantify the convergence rate of the CG method by estimating the nonnegative scalar

$$\min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2. \quad (***)$$

In other words, we search for a polynomial  $P_k$  that minimizes this quantity. In some practical cases, we can find this polynomial explicitly and draw interesting conclusions about CG behavior.

27/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate Methods  
Expanding Subspace Minimization  
CG – Algorithm  
CG Method

Finite-Termination Property



### Comments

The next step is to move from exact expressions to bounds on the error. By extracting the worst-case factor from the sum, we obtain an inequality that ties the error norm to a minimization problem over polynomials. Specifically, the convergence rate depends on minimizing the maximum value of  $[1 + \lambda_i P_k(\lambda_i)]^2$  across all eigenvalues. This formulation isolates a single scalar quantity that encapsulates the challenge of convergence. The question is no longer how each individual eigencomponent behaves, but rather what is the best polynomial that simultaneously controls all components. The beauty of this approach is that it connects numerical linear algebra with approximation theory: the problem of finding such polynomials is directly related to Chebyshev polynomials, which are known for minimizing the maximum deviation on an interval. This perspective allows us to derive sharp convergence bounds that depend on the spectral condition number. More importantly, it shows that convergence is governed not by the dimension of the system but by how well polynomials can approximate the inverse function  $1/\lambda$  over the spectrum. This explains why CG is often so efficient even for very large systems, provided the eigenvalues are not too widely spread.

**Theorem 19**

If A has only r distinct eigenvalues, then the Conjugate Gradient (CG) iteration will terminate at the solution in at most r iterations.

**Proof:** Suppose that the eigenvalues  $\lambda_1, \dots, \lambda_n$  take on the r distinct values  $\tau_1 < \tau_2 < \dots < \tau_r$ .

We define a polynomial  $Q_r(\lambda)$  by

$$Q_r(\lambda) = \frac{(-1)^r}{\tau_1 \cdots \tau_r} (\lambda - \tau_1) \cdots (\lambda - \tau_r)$$

so that  $Q_r(\lambda_i) = 0$  for all  $i = 1, \dots, n$ , and  $Q_r(0) = 1$ .

It follows that  $Q_r(\lambda) - 1$  has a root at  $\lambda = 0$ , so we define a function  $\tilde{P}_{r-1}$  by

$$\tilde{P}_{r-1}(\lambda) = \frac{Q_r(\lambda) - 1}{\lambda}$$

which is a polynomial of degree  $r - 1$ .

28/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Comments**

One of the most striking theoretical results about the Conjugate Gradient method is its finite-termination property. If the matrix has only r distinct eigenvalues, the algorithm is guaranteed to find the exact solution in at most r steps. This is remarkable because it links algebraic structure directly to computational behavior. The proof relies on constructing a special polynomial that vanishes at all the eigenvalues of the matrix. Since the CG error after k steps involves such polynomials applied to the spectrum, having a polynomial that annihilates all eigencomponents means that the error becomes zero. The explicit construction shows that if we multiply linear factors corresponding to each distinct eigenvalue, we obtain a polynomial of degree r. Adjusting it properly, one can derive a degree-(r - 1) polynomial that fits into the CG framework. This observation demonstrates that the method is not only asymptotically convergent but, in principle, exact after finitely many steps. In practice, floating-point errors prevent perfect termination, yet the result still has deep implications: it explains why clustered eigenvalues lead to rapid convergence, and why preconditioning strategies aim to reduce the number of distinct effective eigenvalues. Thus, the finite-termination theorem serves both as a theoretical cornerstone and as guidance for practical algorithm design.



From (\*\*), by setting  $k = r - 1$ , we obtain

$$0 \leq \min_{P_{r-1}} \max_{1 \leq i \leq n} [1 + \lambda_i P_{r-1}(\lambda_i)]^2 \leq \max_{1 \leq i \leq n} [1 + \lambda_i \tilde{P}_{r-1}(\lambda_i)]^2 = \max_{1 \leq i \leq n} Q_r^2(\lambda_i) = 0.$$

Hence, the minimal value is zero, and substituting into the estimation for  $\|x_r - x^*\|_A^2$ , we find:

$$\|x_r - x^*\|_A^2 = 0 \Rightarrow x_r = x^*. \quad \square$$

Building on similar reasoning, Luenberger derives the following estimate, which provides a useful characterization of the CG method's behavior.

## Theorem 20 (Luenberger)

If  $A$  has eigenvalues  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ , then the CG method satisfies the estimate:

$$\|x_{k+1} - x^*\|_A^2 \leq \left( \frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x_0 - x^*\|_A^2.$$

Proof in: David G. Luenberger, Yinyu Ye *Linear and Nonlinear Programming*, 2016 (see Theorem (Partial Conjugate Gradient Method), p. 275).

29/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

At this point, let us carefully examine how the finite-termination result is established. The essential construction is to define a polynomial that vanishes at all distinct eigenvalues of the matrix. Suppose the eigenvalues take the values  $\tau_1, \tau_2, \dots, \tau_r$ . We build a polynomial  $Q_r(\lambda)$  as the product of terms, each of the form  $(\lambda - \tau_j)$ , divided by the product of  $\tau_j$ , multiplied by the factor  $(-1)^r$ . By design, this polynomial evaluates to zero at every eigenvalue and equals one at  $\lambda = 0$ . Now, subtracting one gives us a polynomial with a root at zero. Dividing by  $\lambda$  then produces a polynomial of degree  $r - 1$ , which we call  $\tilde{P}_{r-1}$ . This polynomial lies in the admissible set considered in the convergence analysis. The importance of this step is that it provides an explicit candidate showing that the minimization problem over all such polynomials has value zero. Therefore, when we substitute into the bound for the  $A$  norm of the error, the result vanishes. In other words, the error vector after  $r$  steps becomes exactly zero, which implies that the method produces the exact solution. Thus, the termination property follows directly from polynomial approximation arguments.

The polynomial viewpoint offers a powerful way of interpreting conjugate gradient convergence. Each iteration of the method can be seen as applying a polynomial transformation to the eigenvalues of the matrix. The residual error after  $k$  steps is essentially the initial error multiplied by a polynomial of degree  $k$  that vanishes at selected eigenvalues. This is why the annihilating polynomial plays such an important role: if such a polynomial can eliminate all eigenvalues simultaneously, then the error must vanish. From this perspective, one can also derive precise estimates of the convergence rate, even when the number of distinct eigenvalues is large. Luenberger's estimate provides a useful bound in terms of the extreme eigenvalues. The bound shows that the error after  $k + 1$  steps can be reduced relative to the initial error by a factor that depends on the ratio of the largest and smallest eigenvalues among those still influencing the residual. This estimate is not only a theoretical curiosity but also a practical tool, because it demonstrates that eigenvalue clustering strongly improves performance. When most eigenvalues are located in a narrow interval, the polynomial can approximate zero much more effectively on that interval, leading to rapid error reduction. On the other hand, when eigenvalues are widely spread, the polynomial must stretch across a larger range, which weakens its effectiveness and slows convergence. Thus, the polynomial interpretation provides both a proof of finite termination and an explanation for varying rates of practical convergence.

Conjugate Methods
Expanding Subspace Minimization
CG – Algorithm
CG Method
Finite-Termination Property



Suppose the eigenvalues of  $A$  consist of  $m$  large values and the remaining  $n - m$  eigenvalues clustered near 1. Let

$$\epsilon = \lambda_{n-m} - \lambda_1$$

Then by Theorem 20, after  $m + 1$  steps of CG we have:

$$\|x_{m+1} - x^*\|_A \approx \epsilon \|x_0 - x^*\|_A$$

For small  $\epsilon$ , this implies that CG gives a good approximation in just  $m + 1$  steps.

The next figure compares two scenarios:

- (i) One with 5 large eigenvalues, and the rest clustered in  $[0.95, 1.05]$
- (ii) One with randomly distributed eigenvalues

In both cases, we plot  $\log(\|x_{m+1} - x^*\|_A^2)$  versus iteration number to visualize convergence.

### Comments

Let us illustrate how the convergence estimate becomes especially insightful in cases where eigenvalues have particular structure. Suppose the spectrum of the matrix consists of a few large eigenvalues, say  $m$  of them, while the remaining  $n - m$  eigenvalues are tightly clustered around one. Define  $\epsilon = \lambda_{n-m} - \lambda_1$ . According to the inequality of Luenberger, after  $m + 1$  iterations the error in the  $A$  norm is approximately  $\epsilon$  times the initial error. This means that once the large eigenvalues have been handled, the method effectively suppresses the remaining clustered part extremely quickly. In practical terms, when eigenvalues form tight clusters, the conjugate gradient method can achieve a highly accurate solution in a surprisingly small number of iterations. On the contrary, when eigenvalues are widely spread and lack clustering, convergence slows down significantly. This observation explains the different shapes of convergence curves that we often see in numerical experiments. It also shows why preconditioning methods, which aim to cluster eigenvalues, can be so powerful. The structure of the spectrum thus directly dictates the practical efficiency of the method, reinforcing the deep connection between algebraic properties and computational performance.

## Performance of the conjugate gradient method

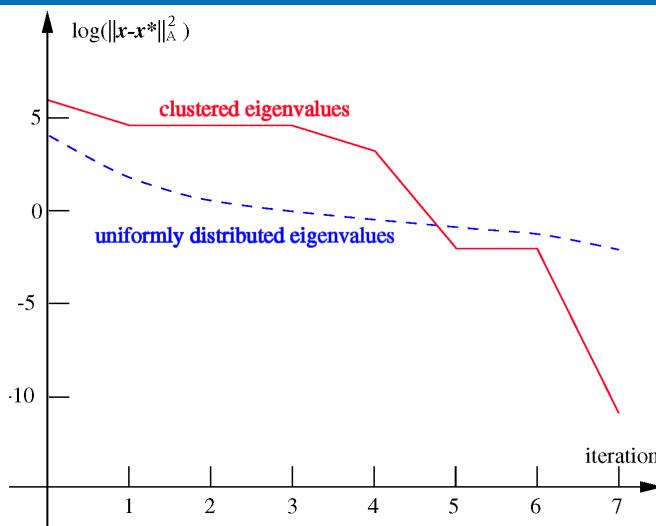


Figure: Performance of the conjugate gradient method: (i) Five large eigenvalues, remaining clustered near 1; (ii) Uniformly distributed eigenvalues.

Conjugate Methods  
 Expanding Subspace Minimization  
 CG — Algorithm  
 CG Method  
 Finite-Termination Property



### Comments

The performance plot should be read through the lens of spectral geometry and polynomial damping. The vertical axis is “logarithm of the A-norm squared of the error”, and the horizontal axis is the iteration count. A straight line with a steep negative slope corresponds to geometric decay per iteration. In the clustered case, with five outliers and the rest in a tight interval around one, the curve falls very rapidly over the first few steps: the CG polynomials quickly place near-zeros at or near the outlying eigenvalues, neutralizing those components. After this early phase, the remaining spectrum is short, and near-Chebyshev polynomials achieve uniformly small values on that short interval, so the slope remains favorable. By contrast, when eigenvalues are spread more uniformly across a wide interval, no low-degree polynomial can be simultaneously small everywhere; the best minimax factor stays comparatively large, and the slope is much gentler. You may also notice a “knee”: once the dominant outliers are handled, the rate transitions to that controlled by the residual cluster width, consistent with the “epsilon” discussion. Thus, the figure is not just empirical; it visualizes exactly how polynomial annihilation and uniform approximation govern convergence. The key takeaway is that the number of iterations tracks spectral features—outliers and cluster width—rather than the dimension of the system. This is why deflation and preconditioning, which mimic the spectral clustering effect, so dramatically improve practical performance.



To improve the convergence rate of the CG method, we apply a change of variables:

$$\hat{x} = Cx,$$

where  $C$  is a nonsingular matrix. Such procedure is known as *preconditioning*. This transforms the original quadratic:

$$\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T(C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T\hat{x}.$$

- Solve the equivalent system:

$$(C^{-T}AC^{-1})\hat{x} = C^{-T}b$$

using CG on the transformed system.

- Convergence depends on the eigenvalues of  $C^{-T}AC^{-1}$ .
- **Goal:** Choose  $C$  such that
  - $\kappa(C^{-T}AC^{-1}) \ll \kappa(A)$ , or
  - the eigenvalues of  $C^{-T}AC^{-1}$  are clustered.

### Comments

The discussion naturally brings us to the idea of preconditioning. We have seen that the efficiency of the conjugate gradient method depends crucially on the distribution of eigenvalues. If they are tightly clustered, convergence is rapid, whereas widely spread eigenvalues slow the method down. The purpose of preconditioning is to artificially transform the system into one with a more favorable spectrum. Concretely, we apply a change of variables by setting  $\hat{x} = Cx$ , where  $C$  is a nonsingular matrix. Substituting into the quadratic form shows that the new system matrix becomes  $C^{-T}AC^{-1}$ . The right-hand side becomes  $C^{-T}b$ . Solving this equivalent system with conjugate gradients is mathematically identical to solving the original one, but the convergence behavior is governed by the eigenvalues of the transformed matrix. The goal is therefore to choose the preconditioner  $C$  such that the condition number of  $C^{-T}AC^{-1}$  is much smaller than that of  $A$ , or equivalently, such that its eigenvalues are well clustered. Preconditioning is thus not an optional enhancement but rather a central technique in modern iterative linear algebra. It is the main tool to bridge the gap between theoretical properties and practical efficiency when solving very large linear systems.

## Preconditioned Conjugate Gradient Algorithm

**Algorithm 10** (Preconditioned CG):

```

1: Given  $x_0$ , preconditioner  $M$ .
2:  $r_0 \leftarrow Ax_0 - b$ 
3: Solve  $My_0 = r_0$ 
4:  $p_0 \leftarrow -y_0$ ,  $k \leftarrow 0$ 
5: while  $r_k \neq 0$  do
6:    $\alpha_k \leftarrow \frac{r_k^T y_k}{p_k^T A p_k}$ 
7:    $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
8:    $r_{k+1} \leftarrow r_k + \alpha_k A p_k$ 
9:   Solve  $My_{k+1} = r_{k+1}$ 
10:   $\beta_{k+1} \leftarrow \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$ 
11:   $p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1} p_k$ 
12:   $k \leftarrow k + 1$ 
13: end while

```

Note: If  $M = I$ , we recover the standard CG method. Algorithm works via  $M = C^T C$  (no explicit use of  $C$ ). In terms of computational effort, the main difference between the preconditioned and unpreconditioned CG methods is the need to solve systems of the form  $My = r$ .

33/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate Methods
Expanding Subspace Minimization
CG – Algorithm
CG Method
Finite-Termination Property



### Comments

The preconditioned conjugate gradient method extends the classical conjugate gradient idea by inserting a preconditioner matrix that modifies the geometry of the problem. We begin with an initial guess, denoted as  $x_0$ , and define the residual  $r_0 = Ax_0 - b$ . Instead of working with this residual directly, we solve the auxiliary system  $My_0 = r_0$ , where  $M$  is the preconditioner. This produces a vector  $y_0$  that acts as a transformed residual. The initial search direction is then chosen as  $-y_0$ .

At each iteration, the algorithm computes a step length  $\alpha_k = \frac{r_k^T y_k}{p_k^T A p_k}$ . The new approximation is updated as  $x_{k+1} = x_k + \alpha_k p_k$ . The residual is also updated, and we again solve a preconditioning system to obtain the new vector  $y_{k+1}$ . A correction factor  $\beta_{k+1}$  is formed as  $\frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$ . Finally, the new search direction is built as  $-y_{k+1} + \beta_{k+1} p_k$ .

This modification preserves the essential structure of conjugate gradients while accelerating convergence. If we set  $M = I$ , the procedure reduces exactly to the classical method. The only real extra cost is solving the small system involving  $M$  at each step. Thus, the efficiency of the whole method hinges on choosing  $M$  wisely: it must be easy to invert but powerful enough to cluster eigenvalues effectively.

## Preconditioning Strategies

There is no universally best preconditioner: trade-offs depend on the problem.

Key considerations:

- ▶ Effectiveness of  $M$ ,
- ▶ Cost of computing/storing  $M$ ,
- ▶ Cost of solving  $My = r$ .

For structured problems (e.g., PDE discretizations),  $My = r$  may correspond to a simplified or coarser version of  $Ax = b$ .

General-purpose preconditioners:

- ▶ SSOR, banded, incomplete Cholesky.
- ▶ Incomplete Cholesky: compute sparse  $\tilde{L} \approx L$  so that

$$A \approx \tilde{L}\tilde{L}^T, \quad M = \tilde{L}\tilde{L}^T, \quad C = \tilde{L}^T$$

$$C^{-T}AC^{-1} = \tilde{L}^{-1}A\tilde{L}^{-T} \approx I$$

- ▶ Solve  $My = r$  via two triangular solves with  $\tilde{L}$ ; cost is comparable to matrix-vector product  $Ap$ .

Challenges:

- ▶  $\tilde{L}$  may not exist or be unstable  $\Rightarrow$  modify diagonals or allow more fill-in (more costly).

34/34 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Conjugate  
Methods

Expanding  
Subspace  
Minimization

CG –  
Algorithm

CG Method

Finite-  
Termination  
Property



### Comments

The effectiveness of preconditioning rests on the choice of the matrix  $M$ . There is no universal best option; instead, the trade-offs depend on the problem structure. The preconditioner must satisfy three competing demands: it should substantially improve convergence, it should not be too expensive to compute or store, and solving the system  $My = r$  should itself be inexpensive. In practice, one often exploits knowledge about the underlying application. For instance, in partial differential equation discretizations, the preconditioner is frequently built to mimic a coarser or simplified version of the original operator. Solving  $My = r$  is then akin to solving a cheaper approximation of the full problem.

Several general-purpose strategies have also been proposed. Symmetric successive over-relaxation, banded preconditioners, and especially incomplete Cholesky factorizations are widely used. The incomplete Cholesky method works by approximating the Cholesky factorization of  $A = LL^T$ . Instead of computing the full factor  $L$ , we compute a sparse matrix  $\tilde{L}$ , chosen to remain close to  $L$  but much cheaper to store. The preconditioner is then  $M = \tilde{L}\tilde{L}^T$ , and one can view the transformed system as  $\tilde{L}^{-1}A\tilde{L}^{-T}$ , which ideally resembles the identity matrix. This means the eigenvalues of the preconditioned system are well clustered, ensuring rapid convergence.

In terms of cost, solving  $My = r$  amounts to two triangular substitutions with  $\tilde{L}$ , which is comparable to performing one matrix-vector multiplication with  $A$ . Challenges remain, however:  $\tilde{L}$  may not exist, or numerical instability may occur under strict sparsity constraints. Remedies include adjusting diagonal elements or allowing more fill-in, but these increase computational burden. Thus, the art of preconditioning is in balancing accuracy and efficiency, tailoring  $M$  to the specific problem.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 5)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Constrained Optimization  
Lagrangian and First-Order Condition  
Tangent Cone  
Tangent Cone  
LICQ



Санкт-Петербургский  
государственный  
университет



30 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In this lecture, we will study the theory of constrained optimization, focusing on how restrictions shape the search for optimal solutions. We begin by formulating constrained problems and examining the nature of local solutions through illustrative examples. The lecture introduces the Lagrangian method and first-order optimality conditions for both equality and inequality constraints, highlighting the role of active sets. We then explore geometric intuition via tangent cones, linearized feasible directions, and constraint qualifications such as LICQ. Through a sequence of progressively complex examples, we demonstrate optimal and nonoptimal cases, emphasizing how geometry and constraint structure determine solution behavior.

Minimize  $f(x)$  over  $x \in \mathbb{R}^n$  subject to

$$\begin{cases} c_i(x) = 0, & i \in \mathcal{E}, \\ c_i(x) \geq 0, & i \in \mathcal{I}, \end{cases}$$

where  $f$  and  $c_i$  are smooth, real-valued functions on a subset of  $\mathbb{R}^n$ ;  $\mathcal{E}$  and  $\mathcal{I}$  are finite index sets. As before,  $f$  is the objective function;  $c_i, i \in \mathcal{E}$  are equality constraints;  $c_i, i \in \mathcal{I}$  are inequality constraints.

The feasible set is defined as  $\Omega = \{x \in \mathbb{R}^n \mid c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\}$  and we can rewrite the problem as

$$\min_{x \in \Omega} f(x). \quad (6)$$

For unconstrained problems, optimality conditions are:

- Necessary: Any local minimizer  $x^*$  must satisfy  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*) \succeq 0$  (positive semidefinite Hessian).
- Sufficient: If  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*) \succ 0$  (positive definite), then  $x^*$  is a strict local minimizer.

1/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

Now we are starting a new fascinating topic: “Constrained Optimization”. When moving from unconstrained optimization to constrained optimization, we significantly change the nature of the problem. In unconstrained optimization, our only goal is to minimize the function  $f(x)$  over all possible values of  $x$  in  $\mathbb{R}^n$ . Now, however, we impose additional restrictions, expressed as equations or inequalities. These restrictions define what we call the feasible set, denoted  $\Omega$ . More formally,  $\Omega$  is the set of all vectors  $x$  such that each equality constraint  $c_i(x) = 0$  for indices in  $\mathcal{E}$  is satisfied, and each inequality constraint  $c_i(x) \geq 0$  for indices in  $\mathcal{I}$  is also satisfied. The optimization problem then becomes minimizing  $f(x)$  only over this feasible set.

It is important to notice how constraints modify the geometry of the search space. Instead of being free to move in every direction, we are restricted to remain within, or on the boundary of, the feasible set. This makes the analysis both richer and more complicated. Just as in the unconstrained case, we will later derive necessary and sufficient conditions for optimality, but these conditions will have to take into account the presence of the constraints. Recall that in the unconstrained setting, a necessary condition for a local minimizer was that the gradient at the point vanished, and the Hessian matrix was positive semidefinite. For sufficiency, the Hessian had to be positive definite. In constrained problems, the situation is similar in spirit but requires entirely new tools, since feasible solutions are no longer determined solely by the landscape of  $f(x)$ , but also by the geometry of the constraints.



Adding constraints can simplify or complicate the search for global minima:

- ▶ Constraints may eliminate many local minima, making the global minimum easier to identify.
- ▶ But constraints can also introduce new local minima and complicate the landscape.

### Example

$$\min(x_2 + 100)^2 + x_1^2 \quad \text{s.t.} \quad x_2 - \cos x_1 \geq 0$$

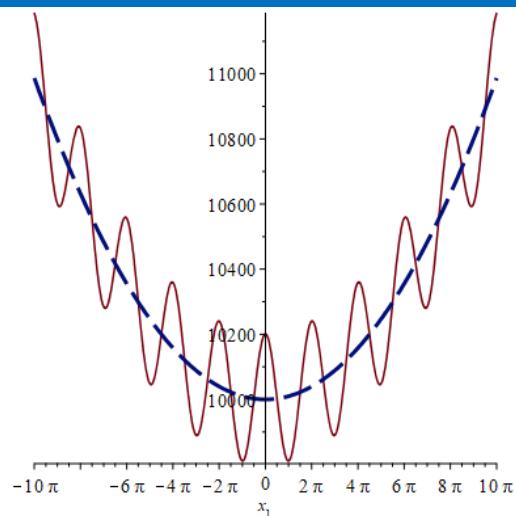
- ▶ Without constraint: Unique solution at  $(0, -100)^T$
- ▶ With constraint: Many local solutions near

$$x^{(k)} = (k\pi, -1)^T, \quad k = \pm 1, \pm 3, \dots$$

### Comments

Adding constraints has a dual effect on optimization problems. On one hand, constraints can simplify the problem. By eliminating infeasible regions, they may also eliminate extraneous local minima, leaving fewer candidate points to consider, and sometimes making it easier to locate the global minimizer. On the other hand, constraints may create additional complexity. They can introduce new local minimizers that would not exist in the unconstrained case, and these may be scattered throughout the feasible region.

The example illustrates this phenomenon clearly. Consider the function  $(x_2 + 100)^2 + x_1^2$ . Without constraints, the global minimizer is unique and easy to identify: the point  $(0, -100)$ . However, when we impose the inequality constraint  $x_2 - \cos x_1 \geq 0$ , the feasible region changes drastically. The unconstrained minimizer no longer belongs to the feasible set, and instead, new local minimizers emerge near the points  $(k\pi, -1)$ , where  $k$  is any odd integer. This situation shows how constraints can fundamentally alter the optimization landscape. What was once a single smooth valley with one bottom point becomes a landscape with multiple valleys and many feasible local solutions. From a practical point of view, this means constrained optimization can sometimes be more challenging, since one must distinguish between multiple local solutions and identify which, if any, is globally optimal.



Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



**Figure:** Comparison of constrained and unconstrained minimization: for each  $x_1$ , the unconstrained minimizer lies at  $x_2^* = -100$  (blue dash line is shifted by  $100^2$ ), while the constraint  $x_2 \geq \cos x_1$  shifts the minimizer to the boundary  $x_2^* = \cos x_1$  (brown solid line).

3/30 || SPbU &amp; HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

The visual comparison between constrained and unconstrained cases highlights the role of feasibility. In the unconstrained problem, for any value of  $x_1$ , the minimizer in the vertical direction occurs at  $x_2 = -100$ . This corresponds to a flat horizontal line of minima in the  $(x_1, x_2)$ -plane. Once the constraint  $x_2 \geq \cos x_1$  is introduced, the picture changes completely. The feasible minimizer for each value of  $x_1$  can no longer lie at negative one hundred, because that would violate the constraint. Instead, the minimizer is forced onto the boundary curve  $x_2 = \cos x_1$ .

This example illustrates an essential principle of constrained optimization: feasible solutions are determined not just by the objective function, but by the interaction between the objective and the feasible set. Often, constraints “push” the minimizer to lie on the boundary rather than inside the region. Thus, constrained optimization problems are frequently about finding the correct balance between the tendency of the objective function to pull the solution toward its unconstrained minimizer, and the restriction of the feasible set, which may prevent this point from being attainable. This boundary interaction is central to the development of optimality conditions, and later we will see how concepts like Lagrange multipliers and Karush-Kuhn-Tucker conditions formalize this balance.

**Defenition: local solution**

A vector  $x^* \in \Omega$  is a local solution of the problem (6) if there exists a neighborhood  $\mathcal{N}$  of  $x^*$  such that

$$f(x) \geq f(x^*) \quad \text{for all } x \in \mathcal{N} \cap \Omega.$$

**Defenition: strict local solution**

A vector  $x^* \in \Omega$  is a strict (or strong) local solution if there exists a neighborhood  $\mathcal{N}$  of  $x^*$  such that

$$f(x) > f(x^*) \quad \text{for all } x \in \mathcal{N} \cap \Omega, \quad x \neq x^*.$$

**Defenition: isolated local solution**

A point  $x^* \in \Omega$  is an isolated local solution if there exists a neighborhood  $\mathcal{N}$  of  $x^*$  such that  $x^*$  is the only local solution in  $\mathcal{N} \cap \Omega$ .

Note: Isolated local solutions are always strict, but not every strict local solution is isolated.

**Comments**

To proceed with constrained optimization, we need precise definitions of what constitutes a local solution. A point  $x^*$  in the feasible set  $\Omega$  is called a local solution if there exists a neighborhood around it in which the objective function does not decrease. More formally, for all feasible points near  $x^*$ , the value of  $f(x)$  is greater than or equal to  $f(x^*)$ . If the inequality is strict for all neighboring feasible points distinct from  $x^*$ , then  $x^*$  is called a strict local solution, or sometimes a strong local solution. The strictness ensures that the minimizer is truly unique in its neighborhood in terms of objective value, and not just one of many points with the same value.

Another refinement is the notion of an isolated local solution. This is a stricter concept: not only is the point a minimizer in its neighborhood, but it is the only such minimizer there. It is important to note that every isolated local solution is necessarily strict, but the reverse is not true. There may be strict local solutions that are not isolated—for instance, when there exists a continuum of minimizers forming a curve or a surface. These distinctions will be crucial later when we discuss stability, uniqueness, and algorithms designed to identify or approximate such solutions.



- Smoothness of  $f$  and constraints is essential: it ensures predictable behavior and enables effective optimization algorithms.
- Feasible regions often have nonsmooth boundaries (e.g., “kinks” or “jumps”), but can still be described using smooth constraint functions.
- Example:  $\|x\|_1 = |x_1| + |x_2| \leq 1$  is nonsmooth, but equivalent to four linear constraints:  $x_1 + x_2 \leq 1$ ,  $x_1 - x_2 \leq 1$ ,  $-x_1 + x_2 \leq 1$ ,  $-x_1 - x_2 \leq 1$ .
- Nonsmooth unconstrained problems can sometimes be reformulated as smooth constrained ones.

### Example

Minimize  $f(x) = \max(x^2, x)$  Reformulated as:

$$\min t \quad \text{s.t.} \quad t \geq x, \quad t \geq x^2$$

Note: Any inequality constraints can be rewritten in the standard form  $c_i(x) \geq 0$  by rearranging terms.

### Comments

In optimization, smoothness of both the objective function and the constraints plays a crucial role. A smooth function is differentiable, and this property enables us to rely on gradients and higher-order information to guide optimization algorithms. Without smoothness, the behavior of the function can be unpredictable, making convergence difficult or even impossible for many standard methods. At the same time, it is important to understand that the feasible region itself does not necessarily need to look smooth. For instance, the boundary of a set may contain corners, edges, or discontinuities. Nevertheless, by using suitable reformulations, we can often describe these regions with smooth functions. This ability to transform nonsmooth structures into smooth ones is one of the cornerstones of constrained optimization.

A clear example is the  $\ell_1$ -ball, defined by the condition that the sum of absolute values of coordinates is less than or equal to one. This condition is nonsmooth because the absolute value function is not differentiable at zero. However, the same region can be equivalently represented by a finite set of linear inequalities, each of which is smooth. This reformulation opens the door to using powerful smooth optimization techniques. Another instructive example is a minimization problem involving the maximum of two functions. Such a problem is naturally nonsmooth, because the maximum introduces a “kink.” But by introducing an auxiliary variable and adding constraints, we can reformulate the task into a smooth constrained problem. This strategy highlights the flexibility of optimization: even when the original problem looks irregular, with a careful reformulation we can bring it into a setting where smooth mathematical tools apply effectively.

**Definition: Active Set**

The active set  $\mathcal{A}(x)$  at any feasible  $x$  consists of the equality constraint indices from  $\mathcal{E}$  together with the indices of the inequality constraints  $i$  for which  $c_i(x) = 0$ ; that is,

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\}.$$

**Definition: Active and Inactive Constraints**

At a feasible point  $x$ , the inequality constraint  $i \in \mathcal{I}$  is said to be active if  $c_i(x) = 0$  and inactive if the strict inequality  $c_i(x) > 0$  is satisfied.

**Example 1: A Single Equality Constraint**

$$\min x_1 + x_2 \quad \text{s.t.} \quad x_1^2 + x_2^2 - 2 = 0$$

In the language of the original task we have  $f(x) = x_1 + x_2$ ,  $\mathcal{I} = \emptyset$ ,  $\mathcal{E} = \{1\}$ , and  $c_1(x) = x_1^2 + x_2^2 - 2$ .

**Comments**

When analyzing constrained optimization problems, a central concept is the so-called active set. At any feasible point, the active set collects all constraints that are currently binding. Binding means that the constraint holds exactly as an equality and leaves no “slack.” For equality constraints, this is always the case, since they must hold exactly. For inequality constraints, only those that are tight, meaning satisfied as equalities, are considered active. The others, which are satisfied with strict inequality, are inactive and do not directly affect the local behavior of the solution.

This distinction between active and inactive constraints is fundamental because optimization near the boundary of feasibility is governed by the active set. From a geometric point of view, the active constraints describe the local shape of the feasible region. They can be seen as the walls that confine possible movements. Understanding which constraints are active is therefore equivalent to identifying which walls are currently blocking further progress.

To illustrate, consider a simple problem: minimize the sum of two variables subject to the condition that their squared lengths add up to two. This condition describes a circle of radius  $\sqrt{2}$ . The feasible set is thus exactly the points lying on this circle. Since this is an equality constraint, it is active everywhere along the boundary. In this case, there are no inequality constraints, so the active set coincides with the set of equality indices. This example shows how the concept of the active set gives us a precise language to describe which restrictions matter at a given point and sets the stage for developing optimality conditions.

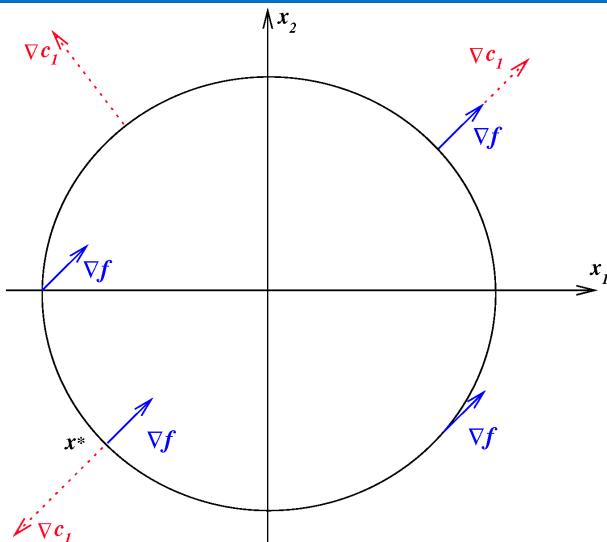


Figure: Problem from Example 1, showing constraint and function gradients at various feasible points.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

The geometric interpretation of constrained optimization provides powerful intuition. Consider again the problem of minimizing the sum of two variables subject to a circular constraint. In this setup, the objective function has gradient vectors that point in the direction of increasing values of the sum. These gradient vectors are constant, since the objective is linear. On the other hand, the constraint corresponds to a circle, and its gradient vectors point radially outward, perpendicular to the boundary.

At feasible points along the circle, we can compare the direction of the objective's gradient with that of the constraint's gradient. If the gradient of the objective points outward the feasible region, then movement against that direction would increase the objective, but it is blocked by the constraint boundary. Conversely, if the gradient points into the feasible region, then movement against it would leave the feasible set. The interesting case arises when the gradient of the target is antiparallel to the gradient of the constraint. At such points, any attempt to move in a direction that maintains feasibility fails to produce descent in the objective. This signals that an optimal solution has been reached.

This simple picture captures a general principle: at an optimum under constraints, the direction of steepest descent cannot be realized because it conflicts with the geometry of feasibility. Instead, the best the optimizer can do is to stop at a point where the objective gradient is balanced against the constraint gradient. This equilibrium is the essence of optimality under equality constraints.

## Example 1: Geometry of Optimality

We observe from the figure that at the solution  $x^*$  there exists a scalar  $\lambda_1^*$  (in this case,  $\lambda_1^* = -\frac{1}{2}$ ) such that

$$\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*).$$

We derive the optimality condition via first-order Taylor expansions:

- Feasibility condition: To stay feasible under  $c_1(x) = 0$ , we require a small (but nonzero) step  $s$  must satisfy that  $c_1(x + s) = 0$ ; that is,

$$c_1(x + s) \approx c_1(x) + \nabla c_1(x)^T s = \nabla c_1(x)^T s = 0.$$

- Descent condition: To ensure decrease in  $f(x)$ , we require

$$f(x + s) - f(x) \approx \nabla f(x)^T s < 0.$$

These two conditions cannot be fulfilled simultaneously, only if  $\nabla f(x)$  is collinear to  $\nabla c_1(x)$ .

Note: No step  $s$  can satisfy both conditions if  $\nabla f(x)$  and  $\nabla c_1(x)$  are collinear to each other, that is, if there exists a scalar  $\lambda$  such that  $\nabla f(x) = \lambda \nabla c_1(x)$ .

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

The key geometric fact is that, at the constrained minimizer, the objective gradient and the constraint normal are parallel. Precisely:  $\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*)$ , where  $\lambda_1^*$  is strictly negative; in this example it equals  $-\frac{1}{2}$ . This statement is the first-order optimality balance for a single equality constraint.

To see why, consider small feasible steps. Feasibility with respect to  $c_1(x) = 0$  requires that any infinitesimal step  $s$  lie in the tangent space of the constraint. In first-order terms, this is  $\nabla c_1(x)^T s = 0$ . A step that decreases the objective must also satisfy the descent inequality  $\nabla f(x)^T s < 0$ . If there exists a direction  $d$  with  $\nabla c_1(x)^T d = 0$  and  $\nabla f(x)^T d < 0$ , then we can reduce the objective while staying feasible.

Now, such a feasible descent direction exists unless the two gradients are collinear. Equivalently, if the projection of the negative objective gradient onto the constraint's tangent space is nonzero, we can move along that projection and decrease the objective. The only way this procedure fails is when the projection vanishes, which happens exactly when  $\nabla f(x) = \lambda \nabla c_1(x)$  for some negative scalar  $\lambda$ . In that collinear case, no first-order feasible descent is possible.



## Comments

The method of Lagrange multipliers is one of the central tools in constrained optimization. The idea is to embed the constraint into the objective function by introducing an auxiliary variable, the so-called Lagrange multiplier. For a problem with a single constraint  $c_1(x) = 0$ , we define the Lagrangian function as  $\mathcal{L}(x, \lambda_1) = f(x) - \lambda_1 c_1(x)$ . Taking the gradient with respect to  $x$ , we obtain  $\nabla_x \mathcal{L}(x, \lambda_1) = \nabla f(x) - \lambda_1 \nabla c_1(x)$ .

At an optimal solution  $x^*$ , there must exist some multiplier  $\lambda_1^*$  such that this gradient vanishes, meaning the condition  $\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0$  is satisfied. This expresses the fact that at the optimum, the gradient of the objective is aligned with the gradient of the constraint surface. The multiplier itself has an important interpretation: it represents the shadow price or marginal cost of relaxing the constraint.

However, this first-order condition is necessary but not sufficient. In other words, the condition may hold at points that are not actual optima—for instance, at saddle points or at local maxima. Moreover, the sign of the multiplier depends on the way the constraint is written. If we multiply the constraint by minus one, the feasible set does not change, but the multiplier flips sign.



## Example 2: A Single Unequality Constraint

$$\min x_1 + x_2 \quad \text{s.t.} \quad 2 - x_1^2 - x_2^2 \geq 0.$$

- Feasible region: disk defined by the constraint.
- Solution remains  $x^* = (-1, -1)^T$ .
- At the boundary, the condition

$$\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*), \quad \lambda_1^* = \frac{1}{2}$$

holds, as in the equality case.

- Key difference: For inequality constraints, the sign of  $\lambda_1^*$  matters.

First-order conditions:

- Descent:  $\nabla f(x)^T s < 0$ .
- Feasibility (to first order):  $c_1(x) + \nabla c_1(x)^T s \geq 0$ .

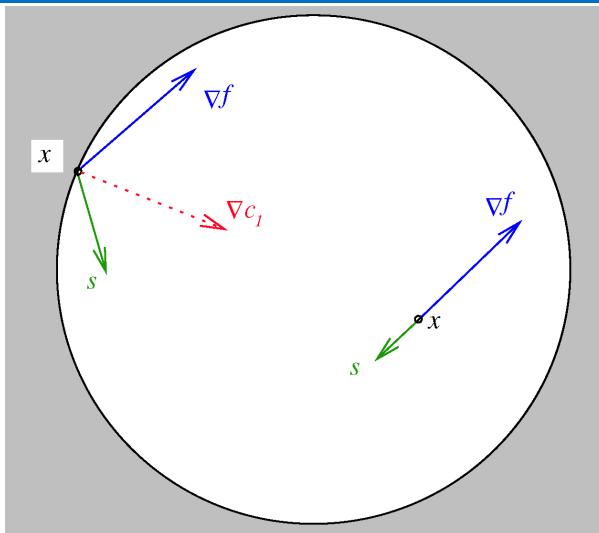
10/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

We now extend the framework to problems with inequality constraints. Consider the example of minimizing  $x_1 + x_2$  subject to the condition  $2 - x_1^2 - x_2^2 \geq 0$ . The feasible region is the closed disk defined by the inequality, which includes both the circle and its interior. The optimal solution remains the point  $(-1, -1)^T$ , exactly as in the equality-constrained case.

At this solution, the gradient of the objective is equal to  $\frac{1}{2}$  times the gradient of the constraint, that is,  $\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*)$  with  $\lambda_1^* = 1/2$ . This is structurally the same as in the equality case. The critical difference, however, is that for inequality constraints the multiplier must satisfy a sign condition. Specifically, it must be nonnegative if the constraint is written in the form  $c_1(x) \geq 0$ . This requirement prevents us from obtaining spurious solutions that would be inconsistent with feasibility.

To test whether a point is optimal, we use two first-order criteria. First, a descent condition: for a candidate step  $s$ , the inner product  $\nabla f(x)^T s$  must be strictly negative to reduce the objective. Second, a feasibility condition: moving along  $s$  must not violate the inequality, which to first order requires  $c_1(x) + \nabla c_1(x)^T s \geq 0$ . Thus, a feasible direction must simultaneously decrease the function and preserve the constraint. The interplay between these two requirements is at the heart of the Karush–Kuhn–Tucker conditions that generalize the Lagrangian method to inequality constraints.



**Figure:** Improvement directions  $s$  from two feasible points  $x$  for the problem from Example 2 at which the constraint is active and inactive, respectively.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

The geometry of inequality constraints is well illustrated by considering feasible directions at different types of points. Inside the disk defined by our inequality, the constraint is inactive: small perturbations in any direction remain feasible. At the boundary, however, the constraint becomes active, and movement must be carefully aligned with the feasible region.

To visualize this, imagine two feasible points. At the interior point, every sufficiently small step is allowed, so the optimizer is free to follow the negative gradient direction and reduce the objective. At the boundary point, in contrast, only certain directions preserve feasibility. Specifically, those that do not cross outside the circle. This restriction is equivalent to requiring that the step  $s$  satisfies  $\nabla c_1(x)^T s \geq 0$ .

Therefore, the interaction between the gradient of the objective and the constraint's normal vector determines whether descent is possible. If they are not aligned, then there exist feasible descent directions. If they are aligned, the feasible set blocks further improvement. This intuition explains why the Lagrange multiplier condition emerges naturally: when the gradients coincide up to a nonnegative factor, no descent direction exists, and the point is stationary with respect to the constrained problem.

This picture makes clear the fundamental difference between equality and inequality constraints. Equalities always restrict us to a surface, while inequalities can either be inactive, allowing free movement, or active, restricting feasible directions. This dichotomy lies at the core of constrained optimization theory.



Case I: Interior Point ( $c_1(x) > 0$ )

- ▶ Any sufficiently small step  $s$  preserves feasibility:

$$c_1(x + s) \approx c_1(x) + \nabla c_1(x)^T s > 0.$$

- ▶ If  $\nabla f(x) \neq 0$ , we can take  $s = -\alpha \nabla f(x)$  for small  $\alpha > 0$ .
- ▶ This step yields a first-order decrease in  $f$  ( $\nabla f(x)^T s < 0$ ) and keeps  $x + s$  feasible.
- ▶ If  $\nabla f(x) = 0$ , no first-order improvement is possible.

Case II: Boundary Point ( $c_1(x) = 0$ )

- ▶ Conditions for improvement and feasibility:

$$\nabla f(x)^T s < 0, \quad \nabla c_1(x)^T s \geq 0.$$

- ▶ These define an open and a closed half-space.
- ▶ Their intersection is empty if and only if the gradients are positively aligned:

$$\nabla f(x) = \lambda_1 \nabla c_1(x), \quad \lambda_1 \geq 0.$$

## Comments

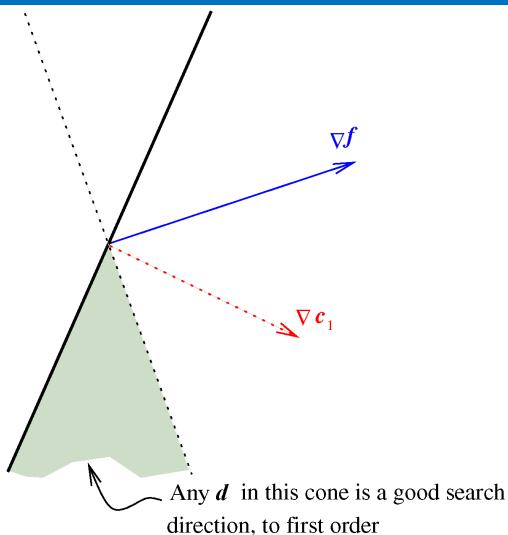
Let us formalize the two possible situations.

Case I: Interior point, where  $c_1(x) > 0$ . In this case, feasibility is easy to preserve. For any sufficiently small step  $s$ , the first-order approximation tells us that  $c_1(x + s) \approx c_1(x) + \nabla c_1(x)^T s$  remains strictly positive. Thus, the inequality is satisfied automatically. If the gradient of the objective is nonzero, we can safely take  $s = -\alpha \nabla f(x)$  for some small positive  $\alpha$ . This yields a strict decrease in the function since  $\nabla f(x)^T s < 0$ . If instead the gradient vanishes, no first-order improvement is possible, and the point is already stationary.

Case II: Boundary point, where  $c_1(x) = 0$ . Here, the situation is subtler. Any candidate step must both decrease the objective and avoid violating the constraint. This means that  $\nabla f(x)^T s < 0$  and  $\nabla c_1(x)^T s \geq 0$  must hold simultaneously. Geometrically, the first condition defines an open half-space, while the second defines a closed half-space. Feasible descent directions exist if and only if these two regions overlap.

The intersection becomes empty precisely when the two gradients point in the same direction, that is, when  $\nabla f(x) = \lambda_1 \nabla c_1(x)$  for some nonnegative  $\lambda_1$ . In this case, no feasible descent step exists, and the point is a candidate optimum under inequality constraints.

This characterization captures the essential difference from equality-constrained problems. For inequalities, the multiplier is constrained to be nonnegative, ensuring consistency with the feasible region. These ideas generalize naturally to multiple constraints, forming the basis of the Karush–Kuhn–Tucker conditions.



**Figure:** A direction  $d$  that satisfies both descent and feasibility conditions lies in the intersection of a closed half-plane and an open half-plane (light green area).

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

When dealing with inequality constraints, the concept of a good search direction becomes more subtle than in unconstrained optimization. A candidate direction must satisfy two simultaneous requirements. First, it should decrease the objective function, which means that the inner product of the gradient of the objective with the step direction must be strictly negative. Second, the direction must preserve feasibility, which requires that the step does not violate the inequality constraint when approximated to first order. Geometrically, these two conditions define two half-planes: one open and one closed. The feasible descent directions are exactly those that belong to the intersection of these two regions. This intersection can be a wedge-shaped sector that narrows or widens depending on how the constraint's gradient aligns with the objective's gradient. If the wedge is nonempty, a descent step exists that simultaneously reduces the function and respects the constraint. If the wedge is empty, then no such direction exists, and we are forced to conclude that the current point is stationary with respect to the constrained problem. This geometric perspective is helpful because it illustrates how feasibility interacts with optimality. In particular, it shows that unlike the unconstrained case, where any negative gradient is a valid descent direction, inequality constraints cut down the available choices. Thus, a "good" search direction is not only about improving the function but also about respecting the boundary geometry of the feasible set.

- The sign of the multiplier matters: if  $\lambda_1 < 0$ , then  $\nabla f(x)$  and  $\nabla c_1(x)$  point in opposite directions (if the equality  $\nabla f(x) = \lambda_1 \nabla c_1(x)$  holds).
- In this case, descent directions satisfying both

$$\nabla f(x)^T s < 0, \quad \nabla c_1(x)^T s \geq 0$$

would occupy an entire open half-plane.

- When no feasible descent direction exists, optimality is characterized by:

$$\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0, \quad \lambda_1^* \geq 0,$$

$$\lambda_1^* c_1(x^*) = 0.$$

- The last condition is the complementarity condition, requiring the Lagrange multiplier  $\lambda_1^* > 0$  only if  $c_1(x^*) = 0$  (constraint is active).
- Case I (interior):  $c_1(x^*) > 0 \Rightarrow \lambda_1^* = 0 \Rightarrow \nabla f(x^*) = 0$ .
- Case II (boundary):  $c_1(x^*) = 0 \Rightarrow \lambda_1^* \geq 0$  allowed, so

$$\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*).$$

**Constrained Optimization**

**Lagrangian and First-Order Condition**

**Tangent Cone**

**Tangent Cone**

**LICQ**



### Comments

The framework can be formalized through the Lagrangian approach, which elegantly incorporates inequality constraints. Here the sign of the multiplier plays a decisive role. If the multiplier were negative, then the gradient of the objective and the gradient of the constraint would point in opposite directions. In that case, feasible descent directions would fill an entire half-plane, contradicting the absence of improvement. Therefore, only nonnegative multipliers are admissible. Optimality is then characterized by three conditions. First, the gradient of the Lagrangian with respect to the decision variables vanishes at the candidate point. Second, the multiplier itself must be nonnegative. Third, we impose the complementarity condition: the product of the multiplier and the constraint value equals zero. This last requirement is crucial, as it enforces that the multiplier can be strictly positive only when the constraint is active, that is, exactly satisfied. Two distinct situations follow. In the interior case, where the inequality is strictly satisfied, the multiplier must vanish, and thus the usual condition that the gradient of the objective is zero is recovered. At the boundary, where the inequality holds with equality, the multiplier may take a nonnegative value, and the stationarity condition requires the gradient of the objective to be a multiple of the gradient of the active constraint. These rules together provide a unified and precise characterization of optimality under inequality constraints.



## Example 3: Two Active Constraints

$$\min x_1 + x_2 \quad \text{s.t.} \quad 2 - x_1^2 - x_2^2 \geq 0, \quad x_2 \geq 0$$

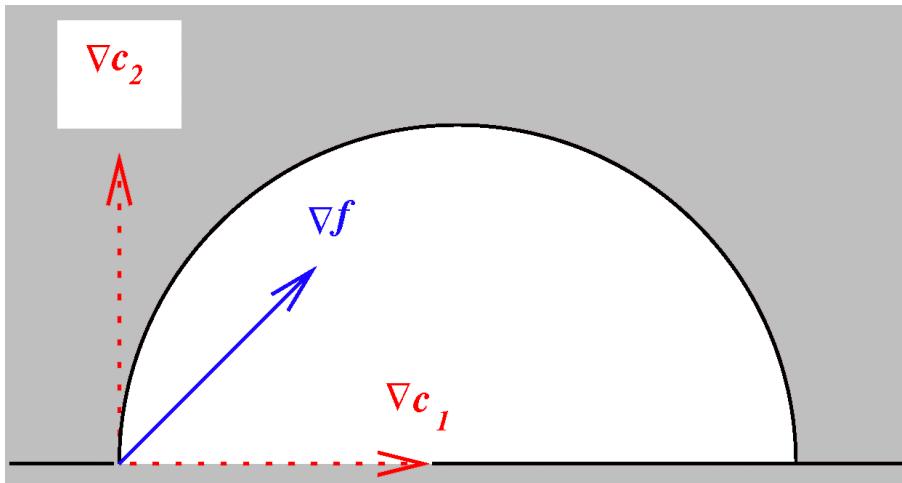
- The feasible region is a half-disk. The solution is at  $x^* = (-\sqrt{2}, 0)^T$ , where both constraints are active.
- As in earlier examples, we look for a direction  $d$  such that:

$$\nabla c_i(x)^T d \geq 0, \quad i \in \mathcal{I} = \{1, 2\}, \quad \nabla f(x)^T d < 0$$

- However, no such direction exists at  $x^*$ . The conditions  $\nabla c_i(x^*)^T d \geq 0$ ,  $i = 1, 2$ , are both satisfied only if  $d$  lies in the quadrant defined by  $\nabla c_1(x^*)$  and  $\nabla c_2(x^*)$ .
- All directions  $d$  in this quadrant satisfy  $\nabla f(x^*)^T d \geq 0$ , so no first-order descent is possible.

## Comments

Let us now examine the case with more than one active constraint. Consider the problem of minimizing the sum of the first variable and the second variable, subject to two conditions: first, that the point lies within the disk of radius  $\sqrt{2}$ , and second, that the second variable is nonnegative. The feasible region is therefore a half-disk. The optimal solution is at the boundary point where the first variable equals  $-\sqrt{2}$  and the second variable equals zero. At this point, both constraints are active simultaneously. To check optimality, we consider possible descent directions. A valid direction must make a nonnegative inner product with the gradient of each constraint, while making a strictly negative inner product with the gradient of the objective. However, these requirements are mutually incompatible at the solution. The gradients of the two constraints define a quadrant, and only directions lying in this quadrant satisfy feasibility. Yet, in this entire quadrant, the inner product with the objective's gradient is nonnegative, meaning that no feasible descent is possible. This situation illustrates the power of the Lagrangian framework with multiple constraints: the conditions exclude any descent direction and thereby certify optimality. It also highlights how the presence of several active constraints can restrict feasible motion much more severely than a single constraint.



**Figure:** The problem in Example 3, illustrating the gradients of the active constraints and objective at the solution.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

The geometric picture of this example helps consolidate the key ideas. At the solution point, we can visualize the gradients of both active constraints and of the objective. The two constraint gradients span a cone that restricts all feasible directions. Because the objective's gradient lies within this cone, every feasible step fails to reduce the objective value. This observation confirms stationarity under the Karush–Kuhn–Tucker framework. More importantly, it provides an intuitive understanding of why the complementarity condition and nonnegativity of multipliers are necessary. Each active constraint introduces a boundary, and the associated multiplier adjusts the balance between the objective and the constraint's influence. When multiple constraints are active, their gradients combine to form a feasible cone, and the multipliers weigh how these constraints prevent descent. If the objective gradient can be written as a nonnegative linear combination of the constraint gradients, then no feasible descent exists, and the point is optimal. This geometric interpretation shows that constrained optimization is not only about algebraic conditions but also about the geometry of feasible sets. By visualizing how gradients and feasible directions interact, one gains deeper insight into why the Karush–Kuhn–Tucker conditions provide a complete description of optimality in inequality-constrained problems.

## Lagrangian and Optimality for Example 3

The Lagrangian for the problem in Example 3 is:

$$L(x, \lambda) = f(x) - \lambda_1 c_1(x) - \lambda_2 c_2(x),$$

where  $\lambda = (\lambda_1, \lambda_2)^T$  is the vector of Lagrange multipliers.

First-order optimality condition:

$$\nabla_x L(x^*, \lambda^*) = 0, \quad \text{for some } \lambda^* \geq 0.$$

Complementarity conditions:

$$\lambda_1^* c_1(x^*) = 0, \quad \lambda_2^* c_2(x^*) = 0.$$

At the point  $x^* = (-\sqrt{2}, 0)^T$ :

$$\nabla f(x^*) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \nabla c_1(x^*) = \begin{bmatrix} 2\sqrt{2} \\ 0 \end{bmatrix}, \quad \nabla c_2(x^*) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Choosing:

$$\lambda^* = \begin{bmatrix} \lambda_1^* \\ \lambda_2^* \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} \\ 1 \end{bmatrix}$$

satisfies the optimality and complementarity conditions.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

At this stage, we extend the method of Lagrange multipliers to the case of two inequality constraints, as posed in Example 3. The Lagrangian is constructed by subtracting each constraint function, multiplied by its corresponding multiplier, from the objective. Thus, the multipliers act as weights that penalize infeasibility, while also shaping the first-order conditions for optimality. Importantly, the multipliers are not arbitrary; they must be nonnegative. This reflects the intuition that constraints only restrict, they never provide "negative pressure."

The first-order stationarity condition states that the gradient of the Lagrangian with respect to the decision variables must vanish at the solution. In other words, the direction of steepest increase of the objective is exactly balanced by a combination of the constraint gradients, scaled by the multipliers. This balancing ensures that no feasible direction exists that can further improve the objective.

Additionally, the complementarity conditions are essential. For each constraint, either the multiplier is zero, meaning the constraint is inactive, or the constraint is active, meaning it binds at the solution and its multiplier can be positive. At the candidate point  $(-\sqrt{2}, 0)^T$ , the gradients of the objective and the two constraints can be explicitly computed. Choosing multipliers equal to  $\frac{1}{2\sqrt{2}}$  and 1, respectively, satisfies both the stationarity and complementarity conditions. Since both multipliers are nonnegative, all conditions for optimality are met.

This point illustrates the fundamental mechanism of constrained optimization: optimality is achieved not by the objective alone, but by a delicate equilibrium between the objective gradient and the constraint gradients.

## Nonoptimal Point: Two Active Constraints

We now consider feasible points that are *not* optimal for problem in Example 3, and examine the behavior of the Lagrangian and its gradient.

Point:  $x = (\sqrt{2}, 0)^T$

- Both constraints are active at this point.
- A feasible descent direction exists:

$$d = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

- For this  $x$ , the Lagrangian gradient condition is satisfied only if

$$\lambda = \begin{bmatrix} -\frac{1}{2\sqrt{2}} \\ 1 \end{bmatrix}$$

- But this violates the condition  $\lambda \geq 0$ .

Therefore, the Lagrangian gradient condition is not satisfied at  $x = (\sqrt{2}, 0)^T$ , even though both constraints are active.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



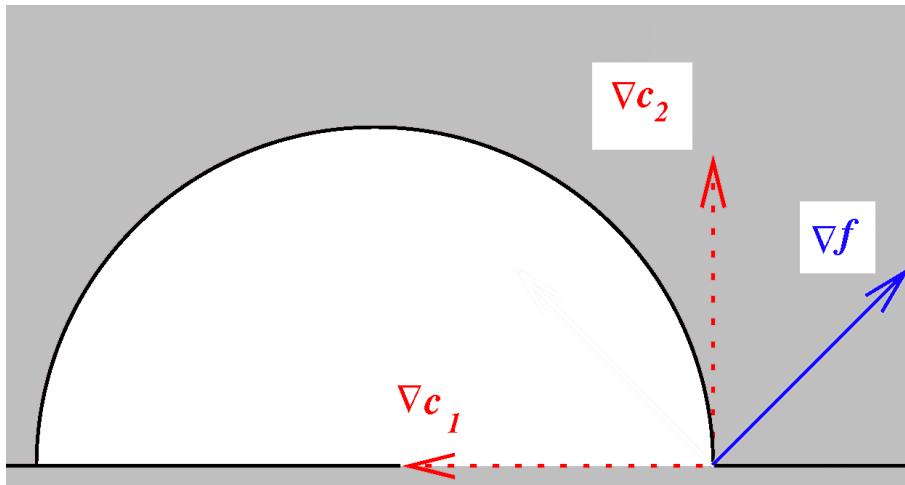
### Comments

Having established the optimal point, we now examine a different feasible point to see why it fails. Consider the point  $(\sqrt{2}, 0)^T$ . At this location, both constraints are active, meaning they hold with equality. At first glance, one might expect this to indicate a potential solution, since active constraints often characterize optimal boundaries. However, the analysis shows otherwise.

The existence of a feasible descent direction is crucial. At this point, the vector  $(-1, 0)^T$  serves as such a direction: moving in that direction maintains feasibility while strictly reducing the objective. This fact alone suggests that the point cannot be optimal, because improvement is still possible.

From the Lagrangian perspective, the gradient condition can only be satisfied if the multipliers take specific values:  $-\frac{1}{2\sqrt{2}}$  and 1. While this combination balances the gradients, the first multiplier is negative. This violates the nonnegativity requirement of the Karush–Kuhn–Tucker conditions. Therefore, the formal optimality conditions are not met.

The lesson here is that merely having multiple active constraints does not guarantee optimality. The key lies in whether the objective gradient can be expressed as a non-negative combination of the active constraint gradients. If not, the geometry indicates that a feasible descent direction must exist, allowing us to move toward better values.



**Figure:** The problem in Example 3, illustrating the gradients of the active constraints and objective at a nonoptimal point.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

The geometric picture provides additional insight into why the nonoptimal point fails. At the point with coordinates  $(\sqrt{2}, 0)^T$ , the gradients of the two active constraints are orthogonal, forming directions that restrict movement into the feasible region.

At the point under discussion, the two constraint gradients span a cone of feasible directions. The question is whether the negative gradient of the objective, representing steepest descent, lies inside this cone. If it does, no feasible improvement is possible, and the point could be optimal. If it does not, then a feasible descent direction exists.

Here, the illustration makes clear that the objective gradient cannot be written as a nonnegative combination of the two constraint gradients. This mismatch means that the descent direction avoids the infeasible region, and hence, improvement is possible. Visually, we see that the optimality conditions are geometric constraints on how the objective aligns with the active constraint surfaces.

This graphical interpretation reinforces the algebraic conclusion: although both constraints bind, the point cannot be a solution because the geometry still allows descent. The figure thus bridges the intuition between the Lagrangian equations and the underlying feasible set. It highlights the importance of checking not only which constraints are active, but also how their gradients combine with the objective. With this understanding, we can proceed to examine situations where only one constraint is active.

## Nonoptimal Point: One Active Constraint

Point:  $x = (1, 0)^T$ , where only constraint  $c_2$  is active.

- Since any small step  $s$  away from this point will continue to satisfy  $c_1(x + s) > 0$ , only  $c_2$  affects feasible descent directions.
- These directions must satisfy:

$$\nabla c_2(x)^T d \geq 0, \quad \nabla f(x)^T d < 0$$

- With:

$$\nabla f(x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \nabla c_2(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- A feasible descent direction is:

$$d = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{4} \end{bmatrix}$$

To satisfy  $\nabla_x L(x, \lambda) = 0$ , we must set  $\lambda_1 = 0$  (since  $c_1(x) > 0$ ). But no value of  $\lambda_2$  can satisfy:

$$\nabla f(x) - \lambda_2 \nabla c_2(x) = 0$$

So this point fails to satisfy the optimality conditions.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

At this stage we examine a boundary point where only one of the two inequality constraints is active. The chosen point has the first constraint strictly satisfied, so it plays no role locally. The second constraint, however, lies exactly on the boundary, which means it alone governs the feasible directions. Geometrically, the feasible region at this point resembles a half-plane bounded by a straight line determined by the active inequality. Any admissible step must respect this line and remain on the feasible side.

The gradient of the objective at the point is a vector pointing diagonally upward. The gradient of the active constraint points vertically upward. To determine if improvement is possible, we check whether there exists a direction that both respects the constraint and decreases the objective. By constructing such a vector explicitly, one observes that it indeed exists: a small movement with a negative horizontal component and a slightly positive vertical component keeps feasibility intact while reducing the objective. This demonstrates that the point cannot be optimal, because one can still descend along this feasible direction.

A complementary way to confirm nonoptimality is to attempt balancing the gradients through the method of multipliers. If the point were optimal, one could assign a nonnegative multiplier to the active constraint such that the gradient of the objective becomes exactly offset by this weighted constraint gradient. But here this balancing fails: no value of the multiplier can make the two vectors align. This mismatch underlines that the point lacks the necessary conditions for optimality.

Thus, from both the geometric picture and the algebraic check, we conclude that this boundary point is not optimal.

We examine whether a linearized model gives meaningful information near a feasible point  $x^* \in \Omega$ .

- ▶ The linearized problem is only helpful if it accurately reflects the local geometry of the feasible region near  $x^*$ .
- ▶ This requires that the feasible set and its linear approximation be "similar" near  $x^*$ .
- ▶ This similarity is ensured by constraint qualifications.

## Definition: feasible sequence

Given a feasible point  $x$ , we call  $\{z_k\}$  a *feasible sequence* approaching  $x$  if

$$z_k \in \Omega \text{ for all large } k, z_k \rightarrow x^*.$$

A tangent direction is a limiting direction of a feasible sequence.

We define the tangent cone  $T_\Omega(x^*)$  as the set of all such directions.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

When analyzing constrained optimization problems, we often ask whether the linearized model around a feasible point gives reliable information about the original nonlinear system. This question is subtle, because linearization is useful only if it preserves the essential local geometry of the feasible region. If the linearized approximation is too different from the actual feasible set, then it may predict directions of movement that are impossible in reality. To address this, we introduce the notion of constraint qualifications. These are conditions that guarantee a meaningful relationship between the feasible set and its linearized counterpart.

To formalize the discussion, we define what it means for a sequence to be feasible. A feasible sequence is simply a sequence of points, all lying within the feasible set, that converges to the point under consideration. From this perspective, tangent directions can be understood as limiting directions of such feasible sequences. If you imagine zooming in on the feasible region around the point, tangent directions describe all possible directions in which you can infinitesimally move while remaining feasible.

The set of all these tangent directions is called the tangent cone. The terminology reflects the fact that the collection of tangent directions often resembles a cone emanating from the point. This cone provides the geometric foundation for analyzing optimality conditions. If the tangent cone is aligned in a way that prevents descent of the objective function, the point may be optimal. If instead there exist tangent directions that allow descent, the point cannot be optimal.

Constraint qualifications are thus crucial because they ensure that the tangent cone, which is defined geometrically, corresponds well to the algebraic description provided by the constraint functions. Without such assumptions, the linearized feasible directions may fail to represent the true geometry of the feasible set, leading to misleading conclusions about optimality.

**Definition: tangent cone**

The vector  $d$  is said to be a *tangent* (or tangent vector) to  $\Omega$  at a point  $x$  if there are a feasible sequence  $\{z_k\}$  approaching  $x$  and a sequence of positive scalars  $\{t_k\}$  with  $t_k \rightarrow 0$  such that

$$\lim_{k \rightarrow \infty} \frac{z_k - x}{t_k} = d.$$

The set of all tangents to  $\Omega$  at  $x$  is called the *tangent cone* and is denoted by  $T_\Omega(x)$ .

Note: the tangent cone  $T_\Omega(x)$  is indeed a cone:  $d \in T_\Omega(x) \Rightarrow \alpha d \in T_\Omega(x), \forall \alpha > 0$  and  $0 \in T_\Omega(x)$  trivially, using  $z_k \equiv x$ .

**Definition: set of linearized feasible directions**

Given a feasible point  $x$  and the active constraint set  $\mathcal{A}(x)$ , *the set of linearized feasible directions*  $\mathcal{F}(x)$  is

$$\mathcal{F}(x) = \{d \mid d^T \nabla c_i(x) = 0, \text{ for all } i \in \mathcal{E}, \quad d^T \nabla c_i(x) \geq 0, \text{ for all } i \in \mathcal{A}(x) \cap \mathcal{I}\}.$$

**Comments**

To give a more rigorous description of tangent directions, let us define them formally. A direction vector is called tangent at a feasible point if there exists a feasible sequence approaching that point, together with a sequence of positive step lengths tending to zero, such that the difference between each sequence element and the point, scaled by the step length, converges to the direction. In other words, if you zoom in infinitely close to the point, the sequence of steps approaches a straight-line movement along this direction.

The set of all such tangent vectors is called the tangent cone. This terminology emphasizes two important properties. First, if a direction belongs to the tangent cone, then any positive multiple of that direction also belongs to it. This reflects the conic nature of the set. Second, the zero vector is trivially included, since a constant sequence equal to the point itself defines a tangent of zero.

Alongside this purely geometric construction, we introduce the concept of linearized feasible directions. Here, we return to the algebraic formulation of the problem, where constraints are described by equality and inequality functions. Given a feasible point, the active constraints are those that are exactly satisfied at that point. The set of linearized feasible directions consists of all directions that satisfy certain linear conditions: they must be orthogonal to the gradients of equality constraints, and they must not point into the infeasible side of the active inequalities.

This distinction is essential. While the tangent cone depends only on the geometry of the feasible set, the linearized feasible directions depend on the algebraic representation of the constraints. When constraint qualifications hold, these two sets coincide locally. When they do not, the linearized feasible directions may give an inaccurate picture, highlighting why constraint qualifications are fundamental in optimization theory.

We return to the first example:

$$\min x_1 + x_2 \quad \text{s.t.} \quad x_1^2 + x_2^2 - 2 = 0$$

Point:  $x = (-\sqrt{2}, 0)^T$

- Consider a feasible sequence approaching  $x$ :

$$z_k = \begin{bmatrix} -\sqrt{2 - 1/k^2} \\ -1/k \end{bmatrix}$$

- Let  $t_k = \|z_k - x\|$ . Then:

$$d = \lim_{k \rightarrow \infty} \frac{z_k - x}{t_k} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \Rightarrow \text{a tangent direction}$$

- The objective function increases along the sequence:

$$f(z_{k+1}) > f(z_k), \quad \text{for } k = 2, 3, \dots$$

- Thus,  $f(z_k) < f(x)$  for all large  $k$

Since  $f(z_k) < f(x)$  for all  $k$ , the point  $x = (-\sqrt{2}, 0)^T$  cannot be optimal.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

Let us now consider an explicit example that illustrates these definitions. Suppose the objective is to minimize the sum of the two variables, subject to the equality constraint that the sum of their squares equals two. This feasible set is a circle of radius  $\sqrt{2}$ . Focus on the point with coordinates  $(-\sqrt{2}, 0)^T$ . This point lies on the circle and is therefore feasible.

To analyze whether it could be optimal, we construct a feasible sequence converging to it. One such sequence is defined by setting the first coordinate equal to  $-\sqrt{2 - 1/k^2}$ , and the second coordinate equal to  $-1/k$ . As  $k$  increases, these points remain feasible, and they converge to our chosen point. We then normalize the step lengths by choosing  $t_k$  equal to the distance between  $z_k$  and  $x$ . In the limit, the normalized difference yields a tangent direction pointing straight downward.

Examining the objective along this sequence, we find that it strictly increases: each successive point yields a higher value than the previous one, and all values are below the objective at the original point. This observation immediately shows that the point cannot be optimal, since we have exhibited feasible perturbations that reduce the objective.

The geometric lesson is that although the point lies on the feasible set, the tangent cone at this point admits directions of descent.

Now consider a different feasible sequence approaching the same point  $x = (-\sqrt{2}, 0)^T$  from the opposite direction:

$$z_k = \begin{bmatrix} -\sqrt{2 - 1/k^2} \\ 1/k \end{bmatrix}$$

- This sequence also lies on the constraint set  $x_1^2 + x_2^2 = 2$
- As  $k \rightarrow \infty$ ,  $z_k \rightarrow x$  and the direction of approach is:

$$d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- The objective  $f(x) = x_1 + x_2$  *decreases* along this sequence
- Tangents from both sequences are of the form  $d = (0, d_2)^T$

The tangent cone at  $x = (-\sqrt{2}, 0)^T$  is given by

$$T_{\Omega}(x) = \left\{ \begin{bmatrix} 0 \\ d_2 \end{bmatrix} \mid d_2 \in \mathbb{R} \right\}$$

24/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



### Comments

The picture becomes even clearer when we consider an alternative feasible sequence approaching the same point from the opposite side. Again, we keep the first coordinate equal to  $-\sqrt{2 - 1/k^2}$ , but now set the second coordinate equal to  $1/k$ . These points also lie on the circle defined by the equality constraint and converge to the same boundary point.

The limiting tangent direction obtained from this sequence points upward along the vertical axis. Thus, when taken together, the two feasible sequences demonstrate that the tangent cone at the point is the entire vertical line through it. More formally, the tangent cone consists of all vectors with zero in the first coordinate and any real value in the second coordinate.

From the perspective of the objective, this second sequence has the opposite effect: along it, the value of the objective decreases rather than increases. In fact, the objective strictly improves as  $k$  grows. The coexistence of both directions — one yielding improvement and the other worsening — shows that the tangent cone captures all possible infinitesimal approaches.

This example reinforces the power of the tangent cone concept. By characterizing all potential limiting directions, it provides a complete geometric picture of feasible movements. Whether a point is optimal depends on how the objective aligns with these directions. If every tangent direction fails to reduce the objective, the point may be optimal. But if even one direction allows descent, optimality is ruled out.

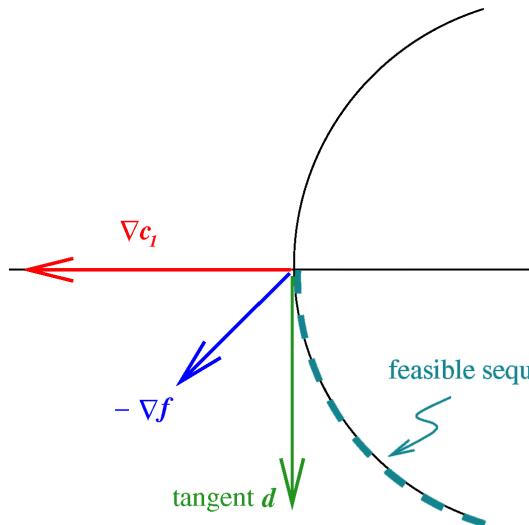


Figure: Constraint normal, objective gradient, and feasible sequence for Example 4.



### Comments

We now examine a new example that further illustrates the relationship between constraint geometry and optimality. Consider the circle defined by the equation  $x_1^2 + x_2^2 = 2$ . The figure shows the boundary of this constraint together with the negative gradient of the objective function and a feasible sequence approaching the boundary point. At the chosen point, with coordinates  $(-\sqrt{2}, 0)^T$ , the constraint surface is smooth and its normal vector is horizontal, pointing leftward. This normal represents the gradient of the constraint function, and it defines the directions that are infeasible.

The negative objective gradient, on the other hand, points diagonally downward, forming a nonzero angle with the constraint normal. The important observation is that feasible directions lie tangentially along the circle at this point. One can move vertically up or down while remaining feasible. The figure highlights this by sketching a possible feasible sequence, which traces the curve of the circle while respecting the constraint.

This example shows that geometric intuition plays a key role in constrained optimization. The feasible region is not simply any set of directions, but rather those aligned with the tangent space of the constraint surface. The gradient of the objective must be considered relative to this tangent space: if there exists a feasible direction along which the objective decreases, then the point cannot be optimal. In this case, such directions clearly exist, as illustrated by the vertical movement along the circle. This confirms that geometry provides both intuition and rigorous guidance when analyzing constrained problems.

## Example 4

We return to the original problem:  $x_1^2 + x_2^2 - 2 = 0$   
and evaluate the linearized feasible directions at  $x = (-\sqrt{2}, 0)^T$ :

$$\nabla c_1(x)^T d = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}^T \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = -2\sqrt{2}d_1 = 0$$

- Hence,  $\mathcal{F}(x) = \left\{ \begin{bmatrix} 0 \\ d_2 \end{bmatrix} \mid d_2 \in \mathbb{R} \right\}$
- In this case:  $\mathcal{F}(x) = T_{\Omega}(x)$

Now suppose that the feasible set is defined instead by the formula

$$\Omega = \{x \mid c_1(x) = 0\}, \quad \text{where } c_1(x) = (x_1^2 + x_2^2 - 2)^2 = 0.$$

Then:

$$\nabla c_1(x) = \begin{bmatrix} 4(x_1^2 + x_2^2 - 2)x_1 \\ 4(x_1^2 + x_2^2 - 2)x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- This implies  $\nabla c_1(x)^T d = 0$  for all  $d$
- So  $\mathcal{F}(x) = \mathbb{R}^2$ , while  $T_{\Omega}(x)$  remains unchanged

Even though the set  $\Omega$  is the same, its algebraic form affects  $\mathcal{F}(x)$ .

In this case:  $\mathcal{F}(x) \neq T_{\Omega}(x)$ .

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

Let us now analyze the same problem from the perspective of linearized feasible directions. The constraint is given by the equation  $x_1^2 + x_2^2 - 2 = 0$ . At the point with coordinates  $(-\sqrt{2}, 0)^T$ , the gradient of the constraint is the vector with components  $-2\sqrt{2}$  and 0. The condition for a feasible direction is that the gradient of the constraint, transposed and multiplied by the direction vector, equals zero. Substituting, this becomes  $-2\sqrt{2} \cdot d_1 = 0$ . Therefore,  $d_1$  must be zero, while  $d_2$  is arbitrary. Hence, the linearized feasible set consists of all vertical directions, which coincides with the tangent cone at this point.

Now suppose we represent the same feasible set in an alternative algebraic form. Instead of writing the constraint as  $x_1^2 + x_2^2 - 2 = 0$ , we square this expression and write  $c_1(x) = (x_1^2 + x_2^2 - 2)^2 = 0$ . Although the underlying set is identical, its gradient behaves differently. Computing the gradient at the same point gives the zero vector. This means that the linearized condition reduces to  $0 = 0$  for all directions, implying that every direction is feasible in the linearized sense. Thus, the linearized feasible set becomes the entire plane, even though the tangent cone remains vertical.

This example is crucial because it shows that the algebraic representation of constraints influences the linearized feasible set. The geometry of the feasible region is unchanged, but its linearized approximation depends on how the constraint is written. In this modified form, the linearized feasible set no longer matches the tangent cone, demonstrating that consistency between algebraic form and geometric reality is essential in optimization theory.

### Example 5

We reconsider the problem (from Example 2):

$$\min x_1 + x_2 \quad \text{s.t.} \quad 2 - x_1^2 - x_2^2 \geq 0$$

The solution  $x = (-1, -1)^T$  is the same as in the equality-constrained version.

- There are infinitely many feasible sequences converging to a boundary point  $x = (-\sqrt{2}, 0)^T$  along a straight line from the interior of the circle.
- These sequences have the form:

$$z_k = (-\sqrt{2}, 0)^T + \frac{1}{k}w,$$

where  $w$  is any vector whose first component is positive  $w_1 > 0$ .

- The point  $z_k$  is feasible provided that  $\|z_k\| \leq \sqrt{2}$ , that is:

$$\left(-\sqrt{2} + \frac{w_1}{k}\right)^2 + \left(\frac{w_2}{k}\right)^2 \leq 2$$

This inequality holds for  $k \geq \frac{w_1^2 + w_2^2}{2\sqrt{2}w_1}$



### Comments

When we move from equality-constrained optimization problems to inequality-constrained ones, the geometry of feasible sets becomes richer and more complex. Consider the problem of minimizing the sum of two variables, subject to the constraint that the point lies inside or on the boundary of a circle of radius  $\sqrt{2}$ . At first glance, this problem looks similar to its equality-constrained counterpart, but the difference lies in the nature of the feasible region. Instead of being restricted to the curve of the circle, we now have access to the entire disk.

The point of interest,  $(-\sqrt{2}, 0)^T$ , lies on the boundary of this disk. To study tangent cones, we examine how feasible sequences approach this boundary point. Importantly, there are infinitely many ways to approach it. One family of sequences takes the form of straight lines from the interior, written as  $(-\sqrt{2}, 0)^T + \frac{1}{k}w$ , where the first component of  $w$  is strictly positive. These sequences remain feasible as long as their norm is less than or equal to  $\sqrt{2}$ . By analyzing the inequality that ensures feasibility, we find a lower bound on  $k$  depending on the components of  $w$ .

The key lesson here is that the inequality-constrained case permits a larger set of feasible directions than the equality case. This abundance of feasible sequences provides more candidate tangent directions, enriching the tangent cone. Thus, the transition from equality to inequality not only changes the algebraic description but also deepens the geometric picture of how solutions can be approached.

## Tangent Cone: Inequality-Constrained Case (continued)

In addition to straight-line feasible sequences, we can construct infinitely many curved sequences approaching  $x = (-\sqrt{2}, 0)^T$  from the interior of the circle.

- These sequences also contribute to the tangent cone
- Therefore, the tangent cone at  $x = (-\sqrt{2}, 0)^T$  is:

$$T_{\Omega}(x) = \{(w_1, w_2)^T \mid w_1 \geq 0\}$$

- For the inequality-constrained definition:

$$\Omega = \{x \mid 2 - x_1^2 - x_2^2 \geq 0\}$$

we have from the definition of the set of linearized feasible directions:

$$d \in \mathcal{F}(x) \quad \text{if} \quad \nabla c_1(x)^T d \geq 0$$

- At  $x = (-\sqrt{2}, 0)^T$ , this becomes:

$$\nabla c_1(x)^T d = \begin{bmatrix} -2x_1 \\ -2x_2 \end{bmatrix}^T \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = 2\sqrt{2}d_1 \geq 0$$

We conclude that  $\mathcal{F}(x) = T_{\Omega}(x)$  for this formulation of the feasible set.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



28/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Building on this, we recognize that straight-line sequences are not the only paths to the boundary point. In fact, curved paths that remain entirely within the disk also converge to the same boundary location. This observation significantly expands the collection of feasible sequences. The consequence is that the tangent cone at  $(-\sqrt{2}, 0)^T$  is not restricted to a narrow set of directions but instead includes all vectors whose first component is nonnegative. Put differently, the tangent cone consists of every direction that points outward or along the boundary without moving back into the infeasible region.

To connect this geometric description with the algebraic definition, we recall that feasible directions can also be characterized by linearization. For inequality constraints, a vector  $d$  is feasible if the gradient of the constraint at the point, transposed and multiplied by  $d$ , is greater than or equal to zero. For our example, the gradient of  $c_1$  at  $(-\sqrt{2}, 0)^T$  simplifies to the vector  $(2\sqrt{2}, 0)^T$ . The feasibility condition then reduces to  $2\sqrt{2} \cdot d_1 \geq 0$ , which simply means that the first component of  $d$  must be nonnegative.

This algebraic condition perfectly matches the earlier geometric interpretation. Therefore, in this case, the set of linearized feasible directions coincides exactly with the tangent cone. This alignment emphasizes that the tangent cone is not an abstract construct but one that faithfully represents both geometry and algebra when the constraints are well-behaved.

*Constraint qualifications* are conditions under which the linearized feasible set  $\mathcal{F}(x)$  accurately reflects the tangent cone  $T_\Omega(x)$ .

- Typically, constraint qualifications ensure that  $\mathcal{F}(x) = T_\Omega(x)$ .
- They guarantee that the linearization captures the essential local geometry of  $\Omega$  near  $x$ .

**Example:** Constraints

$$c_1(x) = 1 - x_1^2 - (x_2 - 1)^2 \geq 0, \quad c_2(x) = -x_2 \geq 0$$

define the feasible set  $\Omega = \{(0, 0)^T\}$ .

- At  $x = (0, 0)^T$ , all feasible sequences approaching  $x$  must have  $z_k = x = (0, 0)^T$  for all  $k$  sufficiently large

$$\Rightarrow T_\Omega(x) = \{(0, 0)^T\}$$

- However, linearizing gives:

$$\mathcal{F}(x) = \{(d_1, 0)^T \mid d_1 \in \mathbb{R}\}$$

In this case,  $\mathcal{F}(x) \neq T_\Omega(x)$  — constraint qualifications are not satisfied.

Constrained Optimization

Lagrangian and First-Order Condition

Tangent Cone

Tangent Cone

LICQ



## Comments

We have just considered the example in which the tangent cone and the linearized feasible set agree. However, this is not always guaranteed. To formalize the relationship, we introduce the concept of constraint qualifications. These are conditions that ensure the linearized feasible directions truly capture the local geometry of the feasible region. In many practical problems, constraint qualifications guarantee that the linearized feasible set equals the tangent cone, preserving consistency between algebraic approximation and geometric reality.

To see why these conditions are important, consider an example in which the feasible region collapses to a single point, the origin. This occurs when we impose two inequalities simultaneously: the first describes the interior of a shifted circle, while the second restricts us to nonnegative vertical coordinates. Their intersection reduces the feasible set to the single point  $(0, 0)^T$ . In this setting, the tangent cone is trivial; it contains only the zero vector, since there is no way to move infinitesimally within the feasible region.

Yet, when we attempt to construct the linearized feasible set, the outcome is quite different. Linearization suggests that all horizontal directions are allowed, yielding an entire axis of feasible directions. This mismatch between geometry and algebra reveals that constraint qualifications are not satisfied. The linearized set no longer represents the true local behavior of the feasible region.

Thus, constraint qualifications serve as safeguards. They ensure that the simplified linear models we use in optimization remain faithful to the underlying geometry. Without them, algorithms risk following directions that do not actually preserve feasibility, potentially undermining convergence and accuracy.



Given the point  $x$  and the active set  $\mathcal{A}(x)$ , we say that the *linear independence constraint qualification* (LICQ) holds if the set of active constraint gradients

$$\{\nabla c_i(x), i \in \mathcal{A}(x)\}$$

is linearly independent.

- LICQ *fails* for:
  - Example with  $\Omega = \{(0, 0)^T\}$
  - The alternative formulation in Example 4
- If LICQ holds, then *none* of the active constraint gradients can be zero.

## Comments

Among the various constraint qualifications, one of the most fundamental and widely used is the linear independence constraint qualification, often abbreviated LICQ. The principle behind LICQ is straightforward but powerful. At any feasible point, we consider the active constraints—those inequalities that are satisfied exactly at equality. Each of these constraints contributes a gradient vector, and LICQ requires that this collection of gradients be linearly independent.

The intuition is that if the active constraints are independent, they intersect cleanly and define a well-structured feasible region locally. In contrast, if some of these gradients are redundant or linearly dependent, the linearized feasible set may misrepresent the true tangent cone, leading to difficulties in optimization. For example, in the case where the feasible set reduces to a single point, the active constraints do not form an independent set. Consequently, LICQ fails, and the linearized model diverges from geometric reality.

Another important implication is that none of the active gradients can be the zero vector. A zero gradient would provide no directional information, violating the independence requirement. Ensuring LICQ holds is particularly significant in the design of optimization algorithms, because many theoretical guarantees—such as the existence of Lagrange multipliers and the validity of optimality conditions—rely on it.

In summary, LICQ acts as a cornerstone condition in constrained optimization. By demanding linear independence among active constraint gradients, it ensures that the interplay between algebraic linearization and geometric structure remains intact, paving the way for robust and reliable analysis.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 6)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025



Санкт-Петербургский  
государственный  
университет



F-O Necessary  
condition

KKT

Tangent Cone  
& Feasible  
Directions

Necessary  
condition

Second-Order  
Necessary  
condition

Sufficient  
Conditions



35 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In this lecture, we will delve into first- and second-order necessary conditions for constrained optimization, with a focus on the well-known Karush–Kuhn–Tucker (KKT) conditions. We will begin by examining the KKT conditions for a box constraint and explore the geometry behind feasible directions and tangent cones. The lecture includes a detailed proof of Lemma 5, illustrating its relevance to optimality conditions. We will also discuss Farkas' Lemma, a classical result in the theory of alternatives, and use it to establish the existence of Lagrange multipliers and verify the KKT conditions. Building on these foundations, we will investigate second-order optimality conditions, both necessary and sufficient, and their application to characterize the behavior of solutions at optimal points. The lecture concludes with a deeper exploration of the critical cone and its role in second-order conditions.

## First-Order Necessary Conditions

Let's consider the general problem

$$\min_{x \in \Omega} f(x), \text{ where } \Omega = \{x \in \mathbb{R}^n \mid c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\} \quad (*)$$

We define the Lagrangian function for this problem as follows:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x).$$

### Theorem 21 (First-Order Necessary Conditions)

Suppose that  $x^*$  is a local solution of (\*), that the functions  $f$  and  $c_i$  in (\*) are continuously differentiable, and that the LICQ holds at  $x^*$ . Then there is a Lagrange multiplier vector  $\lambda^*$ , with components  $\lambda_i^*, i \in \mathcal{E} \cup \mathcal{I}$ , such that the following conditions are satisfied at  $(x^*, \lambda^*)$

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0, \quad (7a)$$

$$c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E}, \quad (7b)$$

$$c_i(x^*) \geq 0, \quad \text{for all } i \in \mathcal{I}, \quad (7c)$$

$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{I}, \quad (7d)$$

$$\lambda_i^* c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}. \quad (7e)$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

When studying constrained optimization, one of the most fundamental tools is the Lagrangian function. It allows us to combine the objective function with the constraints into a single mathematical object. For a problem where we minimize the function  $f(x)$ , subject to both equality and inequality constraints, the Lagrangian is defined as  $f(x)$  minus the sum over all multipliers  $\lambda_i$  times the constraint  $c_i(x)$ . This construction captures how constraints influence the optimal point.

The central result is the first-order necessary conditions, which describe what must hold at any local solution. If a point  $x^*$  is indeed a local minimizer, and if the objective and constraint functions are continuously differentiable, and if the linear independence constraint qualification, or LICQ, is satisfied at  $x^*$ , then there exists a multiplier vector  $\lambda^*$  with remarkable properties. Together, the pair  $x^*$  and  $\lambda^*$  must satisfy several conditions. First, the gradient of the Lagrangian with respect to  $x$  must vanish at  $x^*$ , which expresses stationarity. Second, all equality constraints must hold exactly. Third, all inequality constraints must be satisfied. Fourth, the multipliers associated with inequalities must be nonnegative. Finally, the complementarity condition must hold: for every constraint, either the multiplier or the constraint value is zero.

These conditions together are powerful because they transform a constrained problem into algebraic relationships between gradients and multipliers. They give us the necessary structure to analyze and eventually compute optimal solutions. They are called “first-order” because they rely only on first derivatives, making them both general and computationally useful.

## Karush–Kuhn–Tucker (KKT) Conditions

The conditions (7a)–(7d) are often known as the *Karush–Kuhn–Tucker conditions*, or KKT conditions for short.

The conditions (7d) are called *complementarity conditions*. They imply that either constraint  $i$  is active or  $\lambda_i^* = 0$ , or both.

In particular, for inequality constraints:

- If constraint  $i$  is inactive at  $x^*$  (i.e.  $i \notin \mathcal{A}(x^*)$ ), then  $\lambda_i^* = 0$ .
- Thus, we can rewrite (7a) as:

$$0 = \nabla_x \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*).$$

### Definition: Strict Complementarity

Given a local solution  $x^*$  of (\*) and a vector  $\lambda^*$  satisfying (7a)–(7d), we say that *strict complementarity conditions* holds if:

- For each index  $i \in \mathcal{I}$ , exactly one of  $\lambda_i^*$  and  $c_i(x^*)$  is zero.
- That is,  $\lambda_i^* > 0$  for each  $i \in \mathcal{I} \cap \mathcal{A}(x^*)$ .

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

The collection of first-order necessary conditions is more commonly known as the Karush–Kuhn–Tucker, or KKT, conditions. These conditions are central to nonlinear optimization. They refine the idea of optimality by combining feasibility, stationarity, and complementarity in one framework.

The stationarity condition states that the gradient of the Lagrangian vanishes when evaluated at the solution. Feasibility requires that all equality constraints are exactly satisfied and all inequality constraints are respected. But the most distinctive part is complementarity. This condition links multipliers and constraints: for each inequality constraint, either the multiplier is strictly positive and the constraint is active, or the multiplier is zero when the constraint is inactive. Both being positive at the same time is impossible.

A stronger version is known as strict complementarity. It asserts that for every active inequality constraint, the corresponding multiplier is not only nonnegative but strictly positive. In other words, if a constraint touches the solution boundary, it contributes actively to defining the geometry, and the multiplier reflects that influence. Strict complementarity is not guaranteed in all problems, but when it holds, algorithms often perform more reliably, because it clarifies which constraints are truly active.

Another important refinement is that in the stationarity condition, the sum over constraints can be restricted to the active set. This means only those constraints that “bite” at the solution matter in forming the balance of gradients. This interpretation reveals the beauty of the KKT system: the optimizer balances the gradient of the objective function against a weighted combination of gradients of only the active constraints.

## Example 6: KKT Conditions for a Box Constraint

### Example 6

Consider the feasible region  $\Omega$  defined by the four constraints:

$$x_1 + x_2 \leq 1, \quad x_1 - x_2 \leq 1, \quad -x_1 + x_2 \leq 1, \quad -x_1 - x_2 \leq 1.$$

Restating the problem in standard form with an objective:

$$\min_{\mathbf{x}} \left( x_1 - \frac{3}{2} \right)^2 + \left( x_2 - \frac{1}{2} \right)^4 \quad \text{s.t.} \quad \begin{bmatrix} 1 - x_1 - x_2 \\ 1 - x_1 + x_2 \\ 1 + x_1 - x_2 \\ 1 + x_1 + x_2 \end{bmatrix} \geq 0.$$

The solution is  $\mathbf{x}^* = (1, 0)^T$ , where the first two constraints are active. Denoting active ones as  $c_1, c_2$ , and inactive as  $c_3, c_4$ , we compute:

$$\nabla f(\mathbf{x}^*) = \begin{bmatrix} -1 \\ -\frac{1}{2} \end{bmatrix}, \quad \nabla c_1(\mathbf{x}^*) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \nabla c_2(\mathbf{x}^*) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

$$\lambda^* = \left[ \frac{3}{4} \quad \frac{1}{4} \quad 0 \quad 0 \right]^T \Rightarrow \text{KKT conditions are satisfied.}$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



3/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

To see how these abstract conditions work in practice, let us consider a concrete example. Imagine a feasible region defined by four linear inequalities:  $x_1 + x_2 \leq 1$ ,  $x_1 - x_2 \leq 1$ ,  $-x_1 + x_2 \leq 1$ , and  $-x_1 - x_2 \leq 1$ . Together, these inequalities form a diamond-shaped region centered at the origin.

We now minimize a nonlinear objective: the squared term of  $x_1 - \frac{3}{2}$  plus the quartic term of  $x_2 - \frac{1}{2}$ . Without constraints, the unconstrained minimum would be at  $x_1 = \frac{3}{2}$  and  $x_2 = \frac{1}{2}$ . However, this point lies outside the diamond. Hence, the true solution must occur on the boundary. Careful examination shows that the minimizing point is  $\mathbf{x}^* = (1, 0)^T$ . At this point, the first two constraints are active, while the remaining two are inactive.

We compute the gradient of the objective at this solution, which turns out to be the vector with components  $-1$  and  $-\frac{1}{2}$ . The gradients of the active constraints are vectors with components  $(-1, -1)$  and  $(-1, 1)$ . The KKT conditions can now be checked directly. By solving for the multipliers, we find  $\lambda_1^* = \frac{3}{4}$  and  $\lambda_2^* = \frac{1}{4}$ , while the remaining multipliers are zero. All conditions are satisfied: stationarity holds, constraints are feasible, nonnegativity of multipliers is respected, and complementarity is fulfilled.

This simple but rich example shows that the KKT framework gives a systematic way to verify optimality and to compute supporting multipliers.

F-O Necessary condition

KKT

Tangent Cone &amp; Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



We now relate the tangent cone  $T_\Omega(x^*)$  to the set  $\mathcal{F}(x^*)$  of first-order feasible directions. This relation requires a constraint qualification (LICQ).

### Lemma 5: Tangent Cone and First-Order Feasible Directions

Let  $x^*$  be a feasible point. Then:

- (i)  $T_\Omega(x^*) \subset \mathcal{F}(x^*)$ ,
- (ii) If LICQ holds at  $x^*$ , then  $T_\Omega(x^*) = \mathcal{F}(x^*)$ .

In the proof below we will use the notation

$$A(x^*)^T = [\nabla c_i(x^*)]_{i \in \mathcal{A}(x^*)} \quad (\text{matrix of active constraint gradients}).$$

### Comments

Beyond specific examples, we can deepen our understanding by connecting KKT conditions to geometric concepts. One of the most important is the relationship between the tangent cone at a point and the set of first-order feasible directions. The tangent cone describes all directions in which we can move infinitesimally while staying inside the feasible region. The set of first-order feasible directions, on the other hand, is defined through linear approximations of the constraints.

A general result shows that the tangent cone is always contained within the set of first-order feasible directions. Intuitively, any truly feasible infinitesimal motion must also satisfy the linearized conditions. However, the reverse inclusion does not always hold. In order for the two sets to coincide, a constraint qualification is required. Specifically, when the linear independence constraint qualification is satisfied at the point, the tangent cone and the set of first-order feasible directions are identical.

This equality is crucial because it guarantees that linearization provides a faithful representation of the local geometry. In optimization algorithms, we rely on linearized models to propose steps and directions. If those models overestimate feasible directions, we may encounter false paths. But if LICQ holds, the geometry and algebra align perfectly, making the algorithms reliable.

Formally, the gradients of all active constraints are gathered into a matrix, often denoted  $A(x^*)$ . This matrix provides a compact way to represent the linearized restrictions, and it becomes a central object in both theory and computation.

## Proof of Lemma 5 (Part i)

**Proof:** Without loss of generality, assume all constraints  $c_i(\cdot)$ ,  $i = 1, 2, \dots, m$ , are active at  $x^*$ . (This can be arranged by removing all inactive constraints near  $x^*$  and renumbering the active ones.)

To prove (i), let  $\{z_k\}$  and  $\{t_k\}$  be the sequences for which the following equality holds:

$$\lim_{k \rightarrow \infty} \frac{z_k - x^*}{t_k} = d. \quad (**)$$

Note that  $t_k > 0$  for all  $k$ . This gives:

$$z_k = x^* + t_k d + o(t_k).$$

Now for any  $i \in \mathcal{E}$ , applying Taylor's theorem:

$$\begin{aligned} 0 &= \frac{1}{t_k} c_i(z_k) = \frac{1}{t_k} [c_i(x^*) + t_k \nabla c_i(x^*)^T d + o(t_k)] = \\ &= \nabla c_i(x^*)^T d + \frac{o(t_k)}{t_k}. \end{aligned}$$

Taking the limit as  $k \rightarrow \infty$ , the final term vanishes. Thus, we obtain:

$$\nabla c_i(x^*)^T d = 0.$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

At this point, we begin the detailed proof of the lemma. The first step is to simplify the situation without losing generality. If we focus only on those constraints that are active at the candidate solution, then the analysis becomes cleaner. Inactive constraints do not affect the local geometry, so they can be safely removed, and the active ones can be relabeled. Having done this, we consider sequences that approach the point of interest, together with a sequence of positive scalars that tend to zero.

The condition under study is that the normalized difference between the sequence points and the reference point converges to a certain direction. Intuitively, this means we are zooming in closer and closer to the solution and watching how feasible points approach it. When we write the points in the form of the reference point plus the small scalar times the direction plus higher-order terms, we are effectively applying the definition of differentiability.

Next, we expand each active constraint function around the solution using Taylor's theorem. Because each active constraint vanishes at the reference point, the first nonzero contribution comes from its gradient evaluated at the solution, multiplied by the candidate direction. Higher-order terms become negligible when divided by the small scalar.

As a result, the limit condition enforces that the gradient of each equality constraint at the point must be orthogonal to the direction under consideration. This step confirms that every tangent direction must lie in the null space of the active constraint gradients. This characterization will be essential for completing the proof in later steps.

## Proof of Lemma 5 (cont.)

For active inequality constraints  $i \in \mathcal{A}(x^*) \cap \mathcal{I}$ , we similarly obtain:

$$0 \leq \frac{1}{t_k} c_i(z_k) = \frac{1}{t_k} [c_i(x^*) + t_k \nabla c_i(x^*)^T d + o(t_k)] = \nabla c_i(x^*)^T d + \frac{o(t_k)}{t_k}.$$

Thus, taking the limit as  $k \rightarrow \infty$ , we conclude:

$$\nabla c_i(x^*)^T d \geq 0.$$

To prove (ii), we use the implicit function theorem. Since LICQ holds, the matrix  $A(x^*)$  of active constraint gradients has full row rank  $m$ . Let  $Z \in \mathbb{R}^{n \times (n-m)}$  be a matrix whose columns form a basis for the null space of  $A(x^*)$ :

$$Z \text{ has full column rank, } A(x^*)Z = 0.$$

Pick any  $d \in \mathcal{F}(x^*)$ , and let  $\{t_k\}_{k=0}^{\infty}$  be any sequence of positive scalars with  $\lim_{k \rightarrow \infty} t_k = 0$ .

Define the system of equations  $R : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$  as:

$$R(z, t) = \begin{bmatrix} c(z) - tA(x^*)d \\ Z^T(z - x^* - td) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (***)$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

After establishing the condition for equality constraints, we now turn to inequality constraints that are active at the solution. The logic is similar, but there is a subtle difference. For an inequality constraint, feasibility only requires that the function value remain nonnegative, not necessarily equal to zero once we move away. By expanding the inequality constraint around the solution using Taylor's theorem, we see that its gradient dotted with the direction cannot be negative; otherwise, the constraint would be violated for small perturbations.

Taking the limit, the higher-order terms vanish, and we obtain a nonnegativity condition for the directional derivative. This shows that feasible tangent directions must not decrease active inequalities at the point. Having handled both equality and inequality constraints, the proof advances to its second part. Here, we use a powerful tool from analysis, the implicit function theorem. Because the linear independence constraint qualification holds, the active constraint gradients are linearly independent, which ensures that the associated matrix has full row rank.

This property is crucial: it allows us to define a complementary basis for the space of directions, represented by a matrix whose columns span the null space of the constraints. With this construction, we design a system of equations that couples the original constraints with additional linear conditions involving the null-space basis.

The idea is that solving this system will generate feasible points that move away from the solution in a controlled way along the chosen direction. This prepares the ground for the final argument, where we verify that such constructed sequences indeed realize the definition of tangent vectors.

## Proof of Lemma 5 (cont.)

We claim that the solutions  $z = z_k$  of this system for small  $t = t_k > 0$  give a feasible sequence that approaches  $x^*$  and satisfies the definition of tangent vector.

At  $t = 0$ ,  $z = x^*$ , and the Jacobian of  $R$  at this point is

$$\nabla_z R(x^*, 0) = \begin{bmatrix} A(x^*) \\ Z^T \end{bmatrix},$$

which is nonsingular by construction of  $Z$ .

Thus, by the implicit function theorem, the system  $(***)$  has a unique solution  $z_k$  for small  $t_k > 0$ , and from  $(***)$  and the definition of the set of linearized feasible directions  $(\mathcal{F}(x))$ :

$$\begin{aligned} i \in \mathcal{E} \Rightarrow c_i(z_k) = t_k \nabla c_i(x^*)^T d = 0, \\ i \in \mathcal{A}(x^*) \cap \mathcal{I} \Rightarrow c_i(z_k) = t_k \nabla c_i(x^*)^T d \geq 0, \end{aligned}$$

so  $z_k$  is feasible.

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

The next step is to argue why the constructed system of equations produces feasible points close to the solution. At the specific case where the perturbation parameter equals zero, the only solution is the reference point itself.

To study what happens for small positive values of the parameter, we examine the Jacobian of the system with respect to the variable of interest. This Jacobian turns out to be a block matrix consisting of the active constraint gradients and the transpose of the null-space basis. By design, this matrix is nonsingular, which means we can apply the implicit function theorem. The theorem guarantees that for sufficiently small parameter values, there exists a unique smooth solution to the system.

This solution defines a sequence of points converging back to the original point as the parameter tends to zero. What remains is to check that these points indeed satisfy feasibility conditions. By inspecting the definitions in the system, we see that equality constraints are preserved exactly, while inequality constraints are satisfied in a nondecreasing manner.

This ensures that the constructed points are not only mathematically consistent but also feasible with respect to the optimization problem. Conceptually, this confirms that we can approximate feasible paths by smooth curves that start at the solution and extend in the direction of interest. This bridge between algebraic conditions and geometric intuition is the key contribution of this part of the proof.

## Proof of Lemma 5 (conclusion)

It remains to verify that  $(**)$  holds for this choice of  $\{z_k\}$ . Since  $R(z_k, t_k) = 0$  and by Taylor's theorem:

$$\begin{aligned} 0 = R(z_k, t_k) &= \begin{bmatrix} c(z_k) - t_k A(x^*) d \\ Z^T(z_k - x^* - t_k d) \end{bmatrix} \\ &= \begin{bmatrix} A(x^*)(z_k - x^*) + o(\|z_k - x^*\|) - t_k A(x^*) d \\ Z^T(z_k - x^* - t_k d) \end{bmatrix} \\ &= \begin{bmatrix} A(x^*) \\ Z^T \end{bmatrix} (z_k - x^* - t_k d) + o(\|z_k - x^*\|). \end{aligned}$$

Dividing by  $t_k$  and using nonsingularity of the coefficient matrix, we get:

$$\frac{z_k - x^*}{t_k} = d + o\left(\frac{\|z_k - x^*\|}{t_k}\right).$$

Thus,  $(**)$  is satisfied (for  $x = x^*$ ), and  $d \in T_\Omega(x^*)$  for arbitrary  $d \in \mathcal{F}(x^*)$ , completing the proof.  $\square$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

The final part of the proof focuses on verifying the convergence property of the constructed sequence. We return to the condition that defines tangent vectors, which requires that the normalized difference between sequence points and the reference point converges to the candidate direction.

By substituting the system representation and applying Taylor expansion, we can express the residual as a combination of the active constraint gradients and the null-space basis applied to the deviation from the reference point. The higher-order terms again vanish in the limit. The coefficient matrix in front of the deviation is nonsingular, and this allows us to conclude that the deviation, once scaled by the parameter, must approach the chosen direction.

In other words, the constructed sequence of feasible points not only stays within the constraint set but also exhibits the precise directional behavior required by the definition of tangent vectors. This completes the argument by showing that any feasible direction indeed corresponds to a valid tangent direction at the solution. The lemma thus establishes an equivalence between linearized feasible directions and the tangent cone.

This result plays a central role in optimization theory, as it provides the rigorous link between local linear approximations and actual feasible displacements. By completing this proof, we confirm that the geometric concept of tangent cones is perfectly aligned with the analytic characterization obtained through constraint gradients.

$$\min_{x \in \Omega} f(x), \text{ where } \Omega = \{x \in \mathbb{R}^n \mid c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\} \quad (*)$$

### Theorem 22

If  $x^*$  is a local solution of (\*), then we have

$$\nabla f(x^*)^T d \geq 0, \text{ for all } d \in T_\Omega(x^*).$$

**Proof:** Suppose for contradiction there exists  $d$  with  $\nabla f(x^*)^T d < 0$ . Let  $\{z_k\}$  and  $\{t_k\}$  be the sequences satisfying the Definition of tangent to  $\Omega$  for  $d$ . Then (since  $z_k = x^* + t_k d + o(t_k)$ ) we have

$$\begin{aligned} f(z_k) &= f(x^*) + (z_k - x^*)^T \nabla f(x^*) + o(\|z_k - x^*\|) \\ &= f(x^*) + t_k d^T \nabla f(x^*) + o(t_k). \end{aligned}$$

Since  $d^T \nabla f(x^*) < 0$ , for sufficiently large  $k$

$$f(z_k) < f(x^*) + \frac{1}{2} t_k d^T \nabla f(x^*) < f(x^*),$$

contradicting local optimality of  $x^*$ .  $\square$



### Comments

The necessary condition for local optimality can be expressed in terms of the gradient of the objective function and the feasible directions at a point. Suppose we are analyzing a candidate point that might be a local minimizer. The key observation is that if this point is indeed locally optimal, then moving in any feasible direction cannot decrease the value of the function, at least to first order. Formally, this means that the inner product between the gradient at the point and any tangent direction must be nonnegative.

The gradient represents the direction of steepest increase, and its inner product with a feasible direction measures how the function changes along that direction. If a feasible direction produced a strictly negative value for this product, it would indicate the existence of a descent direction, contradicting local minimality. The proof relies on constructing sequences of feasible points that approximate the tangent vector.

Proof by contradiction. Suppose that there is a direction whose inner product with the gradient is strictly negative. By expanding the objective function around the candidate point using Taylor's theorem, we see that the first-order term dominates small perturbations. If the inner product is negative, we can always find feasible perturbations that reduce the function value, no matter how close we stay to the candidate point.

This logical contradiction shows that for a true local solution, every feasible direction must satisfy the nonnegativity condition. This result forms the foundation for developing stronger optimality conditions and motivates the search for examples where the condition holds but local minimality fails.



The converse of Theorem 22 does not hold: it is possible that

$$\nabla f(x^*)^T d \geq 0 \quad \text{for all } d \in T_\Omega(x^*),$$

yet  $x^*$  is not a local minimizer.

Consider the problem:

$$\min x_2 \text{ subject to } x_2 \geq -x_1^2.$$

This problem is unbounded, but at  $x^* = (0, 0)^T$ , we find that all feasible directions satisfy  $d_2 \geq 0$ , so  $\nabla f(x^*)^T d = d_2 \geq 0$ .

However,  $x^*$  is not a local minimizer, since  $(\alpha, -\alpha^2)^T$  has lower  $x_2$ -value and approaches  $x^*$  as  $\alpha \rightarrow 0$ .

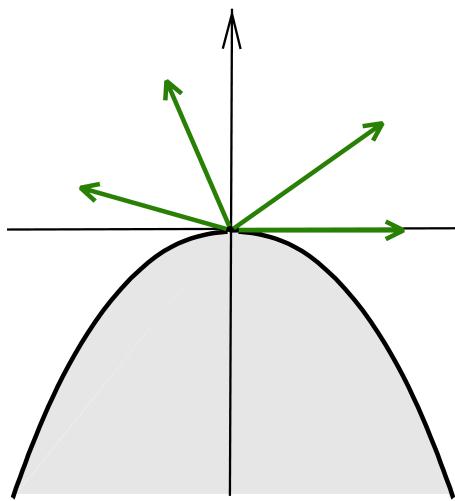
## Comments

Although the nonnegativity condition involving the gradient and feasible directions is essential, it does not fully characterize local minimizers. There exist cases where this condition is satisfied, yet the candidate point is not actually optimal.

A simple yet instructive example is given by minimizing the second coordinate of a vector subject to the constraint that the second coordinate is greater than or equal to the negative square of the first coordinate. Geometrically, this feasible region is the set of points lying above a parabola opening downward. Consider the origin as a candidate solution. At this point, the feasible tangent cone consists of all directions where the second component is nonnegative.

The gradient of the objective function is simply the vector pointing upward, and its inner product with any feasible direction is indeed nonnegative. This satisfies the necessary condition. However, the origin is not a minimizer. One can approach it along points lying on the parabola, which have strictly smaller values of the second coordinate. By choosing a small parameter, these points come arbitrarily close to the origin while providing better function values.

This counterexample illustrates a crucial point: the necessary condition is not sufficient. In practice, this means that checking the gradient condition alone is not enough; further analysis is required to distinguish genuine local solutions from deceptive candidates. The example also highlights the subtle role of curvature in the feasible set. Even when directions seem harmless at first order, second-order geometry can reveal directions that lead to a decrease in the objective, disqualifying the point from being a local solution. This insight motivates the development of second-order optimality conditions.



F-O Necessary condition

KKT

Tangent Cone &amp; Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



Figure: Problem  $\min x_2$  subject to  $x_2 \geq -x_1^2$ . showing various limiting directions of feasible sequences at the point  $(0,0)^T$

### Comments

The graphical interpretation of this example offers additional clarity. At the origin, the set of feasible directions is restricted to those pointing upward or horizontally, since moving downward would immediately violate the inequality constraint. This explains why the gradient condition appears to be satisfied: the gradient points upward, and so its inner product with any feasible direction is nonnegative. However, the picture also reveals the weakness of relying solely on first-order analysis. The feasible set has a curved boundary, given by the parabola, and along this boundary one can find sequences of feasible points that approach the origin from below.

These sequences effectively demonstrate that the origin is not a minimizer, despite satisfying the necessary condition. The limiting directions observed in the figure emphasize the subtle interplay between tangent cones and feasible sequences. While the tangent cone captures only the immediate, first-order directions, it does not fully describe how the feasible set behaves nearby.

In our example, the tangent cone at the origin misses the fact that the boundary curves downward, permitting feasible sequences that dip below the candidate point. This mismatch between first-order information and actual local geometry is the heart of why the necessary condition fails to guarantee optimality. By studying these visualizations, one gains intuition about why stronger conditions, such as those involving second derivatives or curvature information, are needed. They provide a more complete characterization of how the objective interacts with the feasible set, ensuring that candidate solutions are not undermined by subtle geometric features invisible to first-order analysis.

Consider the cone

$$K = \{By + Cw \mid y \geq 0\},$$

where

- $B \in \mathbb{R}^{n \times m}$ ,  $C \in \mathbb{R}^{n \times p}$ ,
- $y \in \mathbb{R}^m$ ,  $w \in \mathbb{R}^p$ ,

and  $y \geq 0$  componentwise.

Farkas' Lemma states exactly one of the following holds for a given vector  $g \in \mathbb{R}^n$ :

either  $g \in K$ , or there exists  $d \in \mathbb{R}^n$  such that

$$g^T d < 0, \quad B^T d \geq 0, \quad C^T d = 0.$$

The two cases are illustrated in the next Figure for the case of  $B$  with three columns,  $C$  null, and  $n = 2$ .

**Note:** In the second case,  $d$  defines a hyperplane separating  $g$  from the cone  $K$ .

12/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



## Comments

To move beyond these limitations, optimization theory employs powerful results from convex analysis. One of the most fundamental is Farkas' Lemma, a classical result of the alternative. This lemma deals with cones generated by linear combinations of certain columns, where some coefficients must be nonnegative. The cone thus represents a structured subset of space that reflects feasible directions in linear constraint systems.

The lemma asserts a striking dichotomy: given a vector, either it belongs to the cone, or there exists a separating vector that certifies its exclusion. This separating vector has three defining properties: its inner product with the given vector is strictly negative, its inner product with the generating columns subject to nonnegativity is nonnegative, and it is orthogonal to the unconstrained components. Geometrically, this separating vector defines a hyperplane that strictly divides the candidate vector from the cone.

The power of Farkas' Lemma lies in providing a certificate of infeasibility in linear systems. If a candidate vector does not lie in the cone, the lemma guarantees the existence of a simple, linear witness to this fact. This principle becomes the backbone of many optimality conditions in constrained optimization. By reformulating feasibility and optimality questions in terms of cones and separating hyperplanes, one can translate nonlinear optimization problems into settings where classical linear results apply.

In particular, the lemma plays a central role in deriving first-order necessary conditions for constrained problems, bridging the gap between geometric intuition and rigorous analytic criteria. It illustrates the deep connection between linear algebra, convex geometry, and optimization theory.

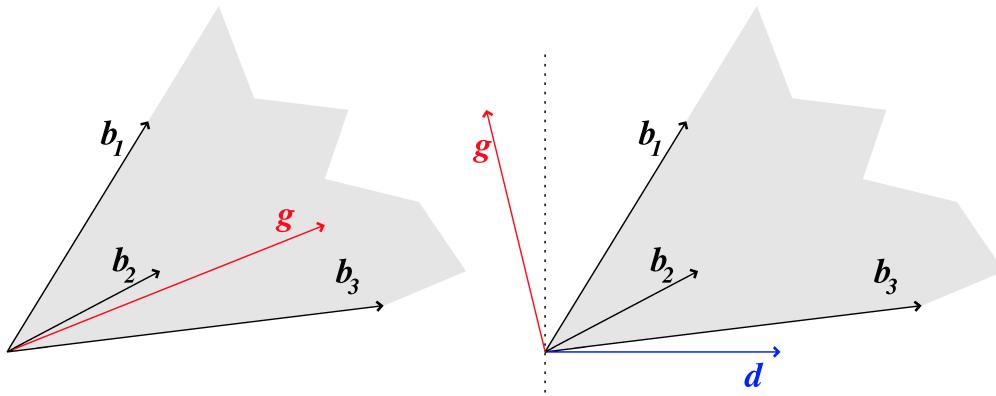


Figure: Farkas' Lemma: Either  $g \in K$  (left) or there is a separating hyperplane (right).



## Comments

This figure provides a simple two-dimensional illustration of the general problem of separating a vector from a cone. The cone is constructed by taking several generating rays and closing them under nonnegative linear combinations. In two dimensions, this cone looks like a wedge spanned by two or more rays. Every vector lying inside this wedge can be represented as a nonnegative combination of the generators. For example, the vector  $g$  in the left panel belongs to the cone, so it can be expressed through those generating rays with nonnegative coefficients.

The situation changes when the vector does not lie inside the cone. In the right panel,  $g$  points outside the wedge. In this case, it becomes possible to draw a separating line, sometimes called a separating hyperplane in higher dimensions. This line divides the plane into two half-spaces: all points of the cone remain on one side, while the vector  $g$  lies strictly on the other side. Geometrically, the line touches or supports the cone without cutting into it, and at the same time it excludes the vector  $g$ .

This very simple two-dimensional picture already captures the essential dichotomy: either the vector lies inside the cone and is representable as a nonnegative combination of the generators, or it lies outside and can be separated from the cone by a line. In the next step, this geometric idea will be extended far beyond the plane. What we will see is that the same reasoning works in any dimension, and the formal statement of this fact is known as Farkas' Lemma.

**Lemma 6 (Farkas')**

Let the cone  $K = \{By + Cw \mid y \geq 0\}$  be defined as before. Given any vector  $g \in \mathbb{R}^n$ , we have either that  $g \in K$  or that there exists  $d \in \mathbb{R}^n$  satisfying the conditions:

$$g^T d < 0, \quad B^T d \geq 0, \quad C^T d = 0. \quad (i)$$

**Proof:** We first show that the two alternatives cannot hold simultaneously.

If  $g \in K$ , then there exist vectors  $y \geq 0$  and  $w$  such that

$$g = By + Cw.$$

If there also exists  $d$  such that

$$g^T d < 0, \quad B^T d \geq 0, \quad C^T d = 0,$$

then taking inner products yields

$$0 > d^T g = d^T By + d^T Cw = (B^T d)^T y + (C^T d)^T w \geq 0,$$

where the last inequality follows from  $B^T d \geq 0$ ,  $y \geq 0$ , and  $C^T d = 0$ .

Thus, both alternatives cannot hold at the same time.

14/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions

**Comments**

Farkas' Lemma is one of the most fundamental results in optimization and convex analysis, because it describes a precise dichotomy about the relationship between a vector and a cone. To understand its meaning, recall that a cone is a set closed under multiplication by positive scalars. In our case, the cone is defined as all combinations of the form  $By + Cw$ , where  $y$  is restricted to be nonnegative, while  $w$  is unrestricted. Now, given any vector  $g$ , the lemma states that one and only one of two situations must occur. Either the vector  $g$  belongs to the cone, or there exists a vector  $d$  that separates  $g$  from the cone by means of inequalities.

This second possibility is expressed by three properties: the inner product of  $g$  and  $d$  is strictly negative; the transpose of  $B$  times  $d$  is nonnegative; and the transpose of  $C$  times  $d$  equals zero. The key intuition is that  $g$  either lies inside the cone, in which case no separating vector can exist, or it lies outside, in which case we can construct such a separator.

The proof starts by showing that both alternatives cannot happen at the same time. If  $g$  is inside the cone, then it can be written as  $By + Cw$ , with  $y$  nonnegative. But then, if there also existed a separating vector  $d$ , we would derive a contradiction: the inner product would be strictly negative and simultaneously greater than or equal to zero. This is impossible. Hence, the two cases are mutually exclusive. The next step will be to show that at least one of them always holds.

## Proof of Farkas' Lemma (continued)

We now show that one of the alternatives holds. To be precise, we show how to construct  $d$  with the properties

$$g^T d < 0, \quad B^T d \geq 0, \quad C^T d = 0$$

in the case that  $g \notin K$ .

It can be shown that  $K$  is a closed set. Let  $\hat{s}$  be the vector in  $K$  that is closest to  $g$  in the sense of the Euclidean norm. Then  $\hat{s}$  is well defined and is given by the solution of the following optimization problem:

$$\min \|s - g\|_2^2 \quad \text{subject to } s \in K.$$

Since  $\hat{s} \in K$ , we have from the fact that  $K$  is a cone that  $\alpha\hat{s} \in K$  for all scalars  $\alpha \geq 0$ . Since  $\|\alpha\hat{s} - g\|_2^2$  is minimized by  $\alpha = 1$ , we have by simple calculus that

$$\frac{d}{d\alpha} \|\alpha\hat{s} - g\|_2^2 \Big|_{\alpha=1} = 0 \Rightarrow (-2\hat{s}^T g + 2\alpha\hat{s}^T \hat{s}) \Big|_{\alpha=1} = 0 \Rightarrow \hat{s}^T (\hat{s} - g) = 0.$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



15/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

We now turn to the constructive part of the proof, where we must show that whenever the vector  $g$  does not belong to the cone, we can explicitly build a vector  $d$  with the desired separating properties. The strategy is geometric: we look for the point in the cone that is closest to  $g$  in terms of Euclidean distance. This point, denoted by  $\hat{s}$ , always exists because the cone is a closed set, meaning that it contains its boundary and has no “holes.”

By definition,  $\hat{s}$  solves an optimization problem: minimize the squared distance between  $s$  and  $g$  subject to  $s$  being inside the cone. Since the cone is a cone in the mathematical sense, any positive multiple of  $\hat{s}$  also belongs to the cone. Now, calculus provides an important condition. If we vary the scalar multiplier  $\alpha$  and minimize the distance between  $\alpha\hat{s}$  and  $g$ , the minimizing value occurs at  $\alpha = 1$ . Differentiating the squared norm with respect to  $\alpha$  and evaluating at one gives a simple equation: the inner product of  $\hat{s}$  with the difference between  $\hat{s}$  and  $g$  equals zero.

This orthogonality condition is crucial. It says that the line connecting  $g$  and  $\hat{s}$  is perpendicular to  $\hat{s}$  itself. Intuitively, this reflects the fact that  $\hat{s}$  is the closest point in the cone to  $g$ . In geometry, whenever we project a point onto a convex set, the vector connecting the point to its projection is orthogonal to the set at that projection point. This observation will serve as the foundation for constructing the separating vector  $d$ .

## Proof of Farkas' Lemma (continued)

Now, let  $s$  be any other vector in  $K$ . Since  $K$  is convex, we have by the minimizing property of  $\hat{s}$  that

$$\|\hat{s} + \theta(s - \hat{s}) - g\|_2^2 \geq \|\hat{s} - g\|_2^2 \quad \text{for all } \theta \in [0, 1],$$

and hence

$$2\theta(s - \hat{s})^\top(\hat{s} - g) + \theta^2\|s - \hat{s}\|_2^2 \geq 0.$$

By dividing this expression by  $\theta$  and taking the limit as  $\theta \downarrow 0$ , we have

$$(s - \hat{s})^\top(\hat{s} - g) \geq 0.$$

Therefore, because of  $\hat{s}^\top(\hat{s} - g) = 0$  we have

$$s^\top(\hat{s} - g) \geq 0, \quad \text{for all } s \in K.$$

We claim now that the vector

$$d = \hat{s} - g$$

satisfies the conditions (i):

$$g^\top d < 0, \quad B^\top d \geq 0, \quad C^\top d = 0.$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



16/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Having established the orthogonality relation, we continue by exploring how this property extends to all other points in the cone. Take any vector  $s$  that lies in the cone. Because the cone is convex, we can form convex combinations between  $\hat{s}$  and any other  $s$ . In other words, the point  $\hat{s} + \theta(s - \hat{s})$  remains in the cone for all  $\theta$  between zero and one. By the minimizing property of  $\hat{s}$ , the distance from this new point to  $g$  cannot be smaller than the distance from  $\hat{s}$  to  $g$ .

Expanding this inequality and performing a simple limit argument shows that the inner product of the difference  $s - \hat{s}$  with the vector  $\hat{s} - g$  is nonnegative. Combining this with the orthogonality condition, we arrive at a general inequality: the inner product of any  $s$  in the cone with the vector  $\hat{s} - g$  is nonnegative. This is an important result, because it suggests a natural candidate for the separating vector. We can now define  $d$  as  $\hat{s} - g$ .

By construction,  $d$  is nonzero whenever  $g$  does not belong to the cone. Furthermore, this inequality shows that  $d$  forms a nonnegative inner product with every element of the cone. Thus,  $d$  satisfies one part of the alternative conditions we are seeking. The remaining step is to check directly that  $d$  also satisfies the other two requirements involving  $g$  and  $C$ , namely that the transpose of  $g$  times  $d$  is negative and the transpose of  $C$  times  $d$  equals zero. These verifications will confirm that we have indeed constructed the separating hyperplane.

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



Note that  $d \neq 0$  because  $g \notin K$ . From the equality  $\hat{s}^T(\hat{s} - g) = 0$  we have

$$d^T g = d^T(\hat{s} - d) = (\hat{s} - g)^T \hat{s} - d^T d = -\|d\|_2^2 < 0,$$

so that  $d$  satisfies the first property in

$$g^T d < 0, \quad B^T d \geq 0, \quad C^T d = 0.$$

Since  $d^T s \geq 0$  for all  $s \in K$ , we have

$$d^T(By + Cw) \geq 0 \quad \text{for all } y \geq 0, w.$$

- Fix  $y = 0$ : then  $(C^T d)^T w \geq 0$  for all  $w$  implies  $C^T d = 0$ .
- Fix  $w = 0$ : then  $(B^T d)^T y \geq 0$  for all  $y \geq 0$  implies  $B^T d \geq 0$ .

Thus,  $d$  satisfies all properties in the alternative and the proof is complete.  $\square$

### Comments

To complete the argument, let us carefully verify the properties of the vector  $d$ . Recall that  $d$  is defined as  $\hat{s} - g$ , and  $g$  is not in the cone, so  $d$  cannot be the zero vector. We can easily deduce from the orthogonality condition that the inner product of  $d$  with  $g$  is strictly negative.

More precisely, substituting back, we obtain that this inner product equals the negative of the squared norm of  $d$ , which is strictly less than zero. Thus, the first required property is satisfied. Next, because  $d$  has a nonnegative inner product with every element of the cone, we can test this condition on particular elements. If we take  $s$  to be equal to  $Cw$ , where  $y = 0$ , we conclude that the transpose of  $C$  times  $d$  must vanish; otherwise, the inequality would not hold for all possible  $w$ .

Similarly, if we take  $s$  equal to  $By$  with  $w = 0$ , the condition reduces to the requirement that the transpose of  $B$  times  $d$  is nonnegative for all  $y \geq 0$ . This is only possible if  $B^T d$  itself is a nonnegative vector. Therefore,  $d$  satisfies all three properties simultaneously: the inner product with  $g$  is negative,  $B^T d$  is nonnegative, and  $C^T d$  equals zero. Together, these facts complete the proof of Farkas' Lemma.

The lemma guarantees that for any given vector, either it is in the cone or there is a separating hyperplane defined by  $d$ . This result is elegant because it transforms a membership problem into a dual characterization involving linear inequalities, forming a cornerstone of duality theory in optimization.

Let's now return to our claim that the set

$$K = \{By + Cw \mid y \geq 0\},$$

(where  $B$  and  $C$  are matrices of dimensions  $n \times m$  and  $n \times p$ , respectively, and  $y, w$  are vectors of appropriate dimensions) is a closed set. This fact was used in the proof of Farkas' Lemma.

## Lemma 7

The set  $K$  is closed.

**Proof:** By splitting  $w$  into positive and negative parts, we write

$$K = \left\{ [B \quad C \quad -C] \begin{bmatrix} y \\ w^+ \\ w^- \end{bmatrix} \mid \begin{bmatrix} y \\ w^+ \\ w^- \end{bmatrix} \geq 0 \right\}.$$

Hence, we may assume w.l.o.g. that  $K = \{By \mid y \geq 0\}$ . Suppose  $B \in \mathbb{R}^{n \times m}$ .

18/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



## Comments

When studying convex cones in linear algebra, one important property is whether such a set is closed. Recall that the cone we are working with is defined as all vectors of the form  $By$ , where  $y$  is constrained to be nonnegative. This structure is essential, because it appears directly in the reasoning behind separation results. To check closedness, we begin by reformulating the set so that all variables are explicitly nonnegative.

This is done by decomposing the vector  $w$  into its positive and negative components, which ensures that the cone is expressed only through nonnegative combinations of columns. After this transformation, the cone can be written simply as all vectors  $By$  with  $y \geq 0$ . This reduction is helpful because it removes unnecessary complications and allows us to focus on the basic mechanism: a cone generated by a finite set of vectors.

The question now is: if we take a convergent sequence of points inside this cone, does the limit point also belong to the cone? Intuitively, this means we want to know whether the cone is robust under limits, or whether points can “slip out” as we pass to the boundary. Proving that the cone is closed is important, because when proving Farkas' lemma we rely on the fact that limits of feasible points remain feasible. Thus, our immediate goal is to build the argument step by step, first establishing a minimal representation of elements in the cone, and then using that to demonstrate closedness rigorously.

F-O Necessary condition

KKT

Tangent Cone &amp; Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



- First, we show that for any  $s \in K$ , we can write  $s = B_I y_I$  with  $y_I > 0$ , where  $I \subset \{1, \dots, m\}$ ,  $B_I$  is the submatrix of  $B$  with columns indexed by  $I$  having full column rank, and  $I$  has minimal cardinality.
- Assume for contradiction that  $\mathcal{N} \subset \{1, \dots, m\}$  is a minimal index set such that  $s = B_{\mathcal{N}} y_{\mathcal{N}}$ ,  $y_{\mathcal{N}} > 0$  (since  $\mathcal{N}$  is minimal, all components of  $y_{\mathcal{N}}$  are strictly positive.), but the columns of  $B_{\mathcal{N}}$  are linearly dependent.
- Then, there exists a nonzero vector  $t$  with  $B_{\mathcal{N}} t = 0$ . For any scalar  $\tau$ , we have:

$$s = B_{\mathcal{N}}(y_{\mathcal{N}} + \tau t).$$

- We adjust  $\tau$  so that some components of  $y_{\mathcal{N}} + \tau t$  become zero, while others remain positive. Let  $\hat{\mathcal{N}}$  be the indices with positive components, and let  $\bar{y}_{\hat{\mathcal{N}}}$  be the corresponding vector.
- Then  $s = B_{\hat{\mathcal{N}}} \bar{y}_{\hat{\mathcal{N}}}$  with  $\bar{y}_{\hat{\mathcal{N}}} > 0$ , contradicting the minimality of  $\mathcal{N}$ .

## Comments

The first step toward proving closedness is to understand how an arbitrary element of the cone can be represented. Take any vector  $s$  belonging to the cone. By construction, it can be written as  $B$  times some nonnegative vector  $y$ . But among all possible such representations, we want one that uses the fewest columns of  $B$ . That means choosing an index set  $I$  with minimal cardinality such that  $s = B_I y_I$ , with  $y_I > 0$ , and with the chosen columns linearly independent.

Why is this reduction important? Because if the chosen columns were linearly dependent, then we could adjust the coefficients and remove one of them, producing a more economical representation. The key contradiction argument works like this: assume we have already chosen a minimal set of indices, yet the corresponding columns are linearly dependent. Then there exists a nonzero vector  $t$  such that multiplying  $B_{\mathcal{N}}$  by  $t$  gives zero. If we add any scalar multiple of  $t$  to our coefficient vector, the product with  $B$  remains unchanged. This allows us to adjust coefficients so that at least one component becomes zero while the others stay positive.

In this way, we construct a new representation of  $s$  using fewer columns, contradicting the assumption that the index set was minimal. Therefore, in a true minimal representation, the selected columns must necessarily be linearly independent. This structural property guarantees that every element of the cone can be written in a way that avoids redundancy. Later, this will play a central role in handling sequences of points, since linear independence allows us to manipulate limits more effectively.

## Step 2: Closedness of Cone K

- ▶ Let  $\{s^j\}$  be a sequence with  $s^j \in K$  for all  $j$  and  $s^j \rightarrow s$ . We aim to prove that  $s \in K$ .
- ▶ By the previous claim, for each  $j$  we can write  $s^j = B_{I_j} y_{I_j}^j$  with  $y_{I_j}^j > 0$ ,  $I_j$  minimal, and  $B_{I_j}$  has linearly independent columns.
- ▶ Since there are only finitely many possible choices of index set  $I_k$ , at least one index set occurs infinitely often in the sequence. By choosing such an index set  $I$ , we can take a subsequence if necessary and assume without loss of generality that  $I_k = I$  for all  $k$ .
- ▶ Then  $s^j = B_I y_I^j$  with  $y_I^j > 0$  and  $B_I$  of full column rank. So  $B_I^T B_I$  is invertible, and
$$y_I^j = (B_I^T B_I)^{-1} B_I^T s^j.$$
- ▶ Taking the limit as  $j \rightarrow \infty$  and using  $s^j \rightarrow s$ , we obtain
$$y_I^j \rightarrow y_I := (B_I^T B_I)^{-1} B_I^T s \geq 0.$$
- ▶ Thus,  $s = B_I y_I$  with  $y_I \geq 0$ , so  $s \in K$ . □

20/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Now we are ready to establish closedness formally. Consider a sequence of points inside the cone that converges to some vector  $s$ . The question is whether  $s$  also lies in the cone. Thanks to the minimal representation result, each point in the sequence can be expressed using a linearly independent set of columns. Since the total number of columns in  $B$  is finite, there are only finitely many possible index sets.

By the pigeonhole principle, at least one index set must appear infinitely often. That means we can extract a subsequence where every element uses the same set of linearly independent columns. For this fixed index set  $I$ , we then write each element of the subsequence as  $B_I$  times a nonnegative vector  $y_I$ . Because the columns of  $B_I$  are independent, the matrix transpose of  $B_I$  times  $B_I$  is invertible. This allows us to solve uniquely for  $y_I$  in terms of the point  $s$ . Now, since our subsequence converges to  $s$ , the corresponding sequence of  $y$  vectors also converges, and importantly, the limit remains nonnegative.

This is crucial: nonnegativity is preserved under limits when the coefficients are obtained through a continuous linear transformation. Thus, the limit vector  $s$  can still be written as  $B_I$  times a nonnegative vector, meaning it belongs to the cone. This proves that the cone is closed. Conceptually, what we have shown is that the cone is stable under convergence: if you approach a boundary point by feasible points, that boundary point is still feasible. This stability under limits ensures the reliability of the cone in optimization and separation theorems.



By applying Farkas' Lemma to the cone

$$N = \left\{ \sum_{i \in \mathcal{A}(x^*)} \lambda_i \nabla c_i(x^*) \mid \lambda_i \geq 0 \text{ for } i \in \mathcal{A}(x^*) \cap \mathcal{I} \right\},$$

with  $g = \nabla f(x^*)$ , we get the alternative:

$$\nabla f(x^*) = \sum_{i \in \mathcal{A}(x^*)} \lambda_i \nabla c_i(x^*) = A(x^*)^\top \lambda^*, \quad \lambda_i \geq 0 \text{ for } i \in \mathcal{A}(x^*) \cap \mathcal{I},$$

or there exists  $d$  with  $d^\top \nabla f(x^*) < 0$  and  $d \in \mathcal{F}(x^*)$ .

**Proof of Theorem 21** (First-Order Necessary Conditions):

Combining Lemmas 5 and 6, we derive the KKT conditions. Suppose  $x^* \in \mathbb{R}^n$  is a feasible point at which the LICQ holds. Then if  $x^*$  is a local solution for (\*), there exists  $\lambda^* \in \mathbb{R}^m$  such that the KKT conditions are satisfied.

## Comments

We are now ready to begin the proof of the theorem 21, which postulates the necessary conditions for a point to be a local minimum. The crucial tool that allows us to proceed is Farkas' Lemma. Recall that this lemma provides a sharp alternative: either a given vector belongs to a certain cone, or there exists a separating direction that violates the nonnegativity property. In our context, the cone is formed by linear combinations of the gradients of the active constraints at the candidate point, with nonnegative coefficients for the inequality constraints.

The vector of interest is the gradient of the objective function at  $x^*$ . If this gradient belongs to the cone, then we can represent it as a weighted combination of the active gradients, exactly the structure required for the KKT system. On the other hand, if the gradient does not belong to this cone, then Farkas' Lemma guarantees the existence of a feasible direction  $d$  such that the inner product of  $d$  with the gradient is strictly negative. But such a direction would allow us to decrease the objective function while remaining feasible, contradicting the assumption that  $x^*$  is a local minimizer.

Therefore, the second alternative is impossible, and the gradient must lie in the cone. This establishes the foundation of the proof: we have shown that if  $x^*$  is indeed a local solution, then the gradient of the objective can be expressed as a combination of active gradients with nonnegative coefficients. This is the central step that connects the geometry of feasible directions with the algebraic representation required for the multipliers.

We first show that there exist multipliers  $\lambda_i$ ,  $i \in \mathcal{A}(x^*)$ , satisfying

$$\nabla f(x^*) = \sum_{i \in \mathcal{A}(x^*)} \lambda_i \nabla c_i(x^*) = A(x^*)^\top \lambda^*, \quad \lambda_i \geq 0 \text{ for } i \in \mathcal{A}(x^*) \cap \mathcal{I}.$$

From Theorem 22,  $d^\top \nabla f(x^*) \geq 0$  for all  $d \in \mathcal{T}_\Omega(x^*)$ . Since LICQ holds, Lemma 5 gives  $\mathcal{T}_\Omega(x^*) = \mathcal{F}(x^*)$ , so we have

$$d^\top \nabla f(x^*) \geq 0 \quad \forall d \in \mathcal{F}(x^*).$$

Then by Farkas' Lemma, the desired multipliers exist.

We now define:

$$\lambda_i^* = \begin{cases} \lambda_i, & i \in \mathcal{A}(x^*), \\ 0, & i \in \mathcal{I} \setminus \mathcal{A}(x^*), \end{cases}$$

and show that  $\lambda^*$  and  $x^*$  satisfy the KKT conditions.

We check these conditions in turn.

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Having established that the gradient lies within the cone of active constraints, the next step is to demonstrate the existence and explicit construction of the multipliers. We start from the fact that for any feasible direction  $d$ , the inner product of  $d$  with the gradient of the objective at  $x^*$  is greater than or equal to zero. This follows from Theorem 22: a local minimizer cannot admit a feasible descent direction. Under the Linear Independence Constraint Qualification, the tangent cone coincides with the feasible direction cone. Therefore, the condition of nonnegativity holds for all feasible directions.

By applying Farkas' Lemma once again, we deduce that the gradient can indeed be represented as a linear combination of the active gradients with nonnegative coefficients. This provides us with the set of multipliers associated with active constraints. To extend this into a complete multiplier vector, we simply assign zero to the components corresponding to inactive inequality constraints. Thus, we obtain a full multiplier vector, which we denote  $\lambda^*$ .

This construction is essential, because it ensures that every constraint is accounted for consistently: active constraints receive the multipliers provided by the cone representation, while inactive constraints are given zero weight. At this point, we have built the mathematical object that should satisfy the KKT conditions, and our task reduces to verifying that each condition in the theorem holds for the pair consisting of  $x^*$  and  $\lambda^*$ .

## Verification of KKT Conditions

- Condition (7a) ( $\nabla_x \mathcal{L}(x^*, \lambda^*) = 0$ ) follows immediately from

$$\nabla f(x^*) = \sum_{i \in \mathcal{A}(x^*)} \lambda_i \nabla c_i(x^*) = A(x^*)^\top \lambda^*, \quad \lambda_i \geq 0 \text{ for } i \in \mathcal{A}(x^*) \cap \mathcal{I},$$

and from the definition of the Lagrangian function,

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x),$$

as well as the previous definition of the multipliers,  $\lambda_i^* = \begin{cases} \lambda_i, & i \in \mathcal{A}(x^*), \\ 0, & i \in \mathcal{I} \setminus \mathcal{A}(x^*). \end{cases}$

- Since  $x^*$  is feasible, the conditions (7b) and (7c) are satisfied:

$$c_i(x^*) = 0 \quad \text{for all } i \in \mathcal{E}, \quad c_i(x^*) \geq 0 \quad \text{for all } i \in \mathcal{I}.$$

- From the active combination condition,  $\lambda_i^* \geq 0$  for  $i \in \mathcal{A}(x^*) \cap \mathcal{I}$ , and from the definition of  $\lambda^*$ ,  $\lambda_i^* = 0$  for  $i \in \mathcal{I} \setminus \mathcal{A}(x^*)$ . Hence,  $\lambda_i^* \geq 0$  for all  $i \in \mathcal{I}$ , so that (7d) holds.

- For  $i \in \mathcal{A}(x^*) \cap \mathcal{I}$ , we have  $c_i(x^*) = 0$ , while for  $i \in \mathcal{I} \setminus \mathcal{A}(x^*)$ , we have  $\lambda_i^* = 0$ . Therefore,  $\lambda_i^* c_i(x^*) = 0$  for all  $i \in \mathcal{I}$ . so that (7e) is satisfied as well.

And the proof is complete.  $\square$

23/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

The final step of the proof is to check each of the KKT conditions systematically. We begin with the stationarity condition, which requires that the gradient of the Lagrangian function at  $x^*$  equals the transpose of the Jacobian of the constraints at  $x^*$  multiplied by  $\lambda^*$ . This holds directly by construction, since the multipliers were chosen to represent the gradient in this precise form. Next, the feasibility conditions must be verified.

Because  $x^*$  is assumed to be a feasible point, all equality constraints vanish and all inequality constraints are satisfied with nonnegative values. The nonnegativity condition on the multipliers follows immediately: multipliers for active inequalities are nonnegative, and those for inactive inequalities were set to zero. Finally, complementary slackness must be checked.

For every inequality constraint, either it is active, in which case its multiplier may be positive but the constraint itself is zero, or it is inactive, in which case the multiplier is zero while the constraint is strictly positive. In both cases, the product of the multiplier with the constraint value is zero, as required. Having confirmed stationarity, feasibility, nonnegativity, and complementary slackness, all conditions of the theorem are satisfied.

This completes the proof of the KKT theorem: any local solution under the Linear Independence Constraint Qualification admits a corresponding multiplier vector such that the KKT system holds. Thus, the argument is closed, and the result stands as a cornerstone of constrained optimization.

First-order (KKT) conditions describe how  $\nabla f$  and gradients of active  $c_i$  behave at a solution  $x^*$ .

- When these conditions hold, a move along any  $w \in \mathcal{F}(x^*)$  either:
  - increases  $f$  to first order:  $w^\top \nabla f(x^*) > 0$ , or
  - leaves  $f$  unchanged to first order:  $w^\top \nabla f(x^*) = 0$ .
- Second-order conditions analyze such “undecided” directions.

**Key Idea:** In directions where  $w^\top \nabla f(x^*) = 0$ , the second-order terms in the Taylor expansions of  $f$  and  $c_i$  are examined to determine whether  $f$  increases or decreases. This analysis concerns the curvature of the Lagrangian in those directions.

**Assumption:** Further, until it is said otherwise, all  $f$  and  $c_i$ ,  $i \in \mathcal{E} \cup \mathcal{I}$ , are assumed to be twice continuously differentiable.

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Up to now, we have mainly discussed first-order conditions, also known as the Karush–Kuhn–Tucker conditions. These describe how the gradient of the objective function relates to the gradients of the active constraints at a candidate solution. They allow us to understand whether moving in a certain feasible direction will tend to increase the objective function or leave it unchanged, at least to first order.

However, there are directions where first-order information is silent: directions in which the inner product between the feasible direction and the gradient of the objective function is zero. Along such directions, the first-order expansion provides no clear guidance. This is precisely where second-order conditions become essential. By examining the second derivatives of both the objective and the constraints, we are able to capture the curvature properties of the problem.

In other words, second-order terms act as a kind of “tiebreaker,” deciding whether a feasible move will ultimately increase or decrease the objective. The curvature is analyzed through the Hessian of the Lagrangian function, which integrates the objective and constraints into a single mathematical object. Since curvature considerations are more delicate, we impose stronger smoothness assumptions, requiring both the objective and the constraints to be twice continuously differentiable.

This guarantees that the Taylor expansions we rely on behave properly. Altogether, the step from first- to second-order analysis represents a refinement: it moves from directional slopes to local surface shapes, providing deeper insight into the structure of optimal solutions.

## Critical Cone: Definition

Given  $\mathcal{F}(x^*)$  (the set of linearized feasible directions) and a Lagrange multiplier vector  $\lambda^*$  satisfying the KKT conditions (7), we define the *critical cone*  $\mathcal{C}(x^*, \lambda^*)$  as:

$$\mathcal{C}(x^*, \lambda^*) = \{w \in \mathcal{F}(x^*) \mid \nabla c_i(x^*)^T w = 0, \text{ for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* > 0\}$$

Equivalently:

$$w \in \mathcal{C}(x^*, \lambda^*) \Leftrightarrow \begin{cases} \nabla c_i(x^*)^T w = 0, & \text{for all } i \in \mathcal{E}, \\ \nabla c_i(x^*)^T w = 0, & \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* > 0, \\ \nabla c_i(x^*)^T w \geq 0, & \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* = 0. \end{cases} \quad (8)$$

Key property: From the definition (8) and the fact that  $\lambda_i^* = 0$  for all inactive constraints  $i \in \mathcal{I} \setminus \mathcal{A}(x^*)$ , it follows that

$$w \in \mathcal{C}(x^*, \lambda^*) \Rightarrow \lambda_i^* \nabla c_i(x^*)^T w = 0 \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}.$$

From the first KKT condition (7a) and the definition of the Lagrangian function, we have that

$$w \in \mathcal{C}(x^*, \lambda^*) \Rightarrow w^T \nabla f(x^*) = \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^* w^T \nabla c_i(x^*) = 0.$$

Hence the critical cone  $\mathcal{C}(x^*, \lambda^*)$  contains directions from  $\mathcal{F}(x^*)$  for which it is not clear from first derivative information alone whether  $f$  will increase or decrease.

25/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



## Comments

To deepen our understanding of second-order conditions, we need a precise way of describing the feasible directions that are truly “critical.” These are not just any directions that satisfy the linearized constraints; instead, they are directions for which the first-order information fails to distinguish whether the objective will improve or worsen. The mathematical object that captures this subtle set of directions is called the critical cone.

Formally, the critical cone consists of feasible directions that are tangent to all equality constraints and to those active inequality constraints with strictly positive Lagrange multipliers. For constraints with zero multipliers, the feasible directions may allow movement that does not immediately violate feasibility, so a weak inequality suffices. This careful definition ensures that the cone isolates exactly the cases where the first-order expansion leaves us undecided.

An important property follows immediately: for any direction in the critical cone, the linear term of the Lagrangian expansion vanishes, meaning that the directional derivative of the objective is zero. Consequently, if we want to decide about local optimality, we must look at second-order terms in precisely these directions. The introduction of the critical cone thus provides the right framework: it trims away irrelevant directions, leaving us with the delicate set where higher-order analysis is truly necessary.

## Example: Critical Cone Illustration

### Example 7

Consider the problem:

$$\min x_1 \quad \text{subject to } x_2 \geq 0, \quad 1 - (x_1 - 1)^2 - x_2^2 \geq 0.$$

The solution is  $x^* = (0, 0)^\top$  with active set  $\mathcal{A}(x^*) = \{1, 2\}$  and a unique optimal Lagrange multiplier:

$$\lambda^* = (0, 0.5)^\top.$$

Since the gradients of active constraints at  $x^*$  are  $(0, 1)^\top$  and  $(2, 0)^\top$ , the LICQ holds, ensuring uniqueness of  $\lambda^*$ .

Linearized feasible set:

$$\mathcal{F}(x^*) = \{d \mid d \geq 0\}.$$

Critical cone:

$$\mathcal{C}(x^*, \lambda^*) = \{(0, w_2)^\top \mid w_2 \geq 0\}.$$

Key observation: The only direction in the critical cone is vertical and nonnegative, due to the fact that only the second constraint has a positive multiplier.

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Let us now turn to an example that illustrates the definition of the critical cone. Consider an optimization problem with two inequality constraints: one requires a variable to be nonnegative, while the other describes the interior of a circle. At the solution, both constraints are active. Computing the gradients of the constraints shows that the linear independence constraint qualification is satisfied, ensuring a unique multiplier vector.

Interestingly, only one of the active constraints has a strictly positive multiplier. This detail is decisive, because it dictates which directions belong to the critical cone. Specifically, the linearized feasible set allows only nonnegative movements in both coordinates. But once we impose the definition of the critical cone, we see that only vertical, nonnegative directions remain. In other words, the problem structure and the positivity of the multiplier eliminate all other feasible tangents from critical consideration.

This example demonstrates how the cone is typically much smaller than the entire feasible set: it isolates the directions along which first-order conditions provide no information. Through this concrete case, we see the importance of multipliers in shaping the geometry of the critical cone. The example also hints at the deeper role of second-order analysis: once we know which directions are critical, we can check whether the curvature of the Lagrangian guarantees nonnegative behavior there, a condition that is central to second-order optimality.

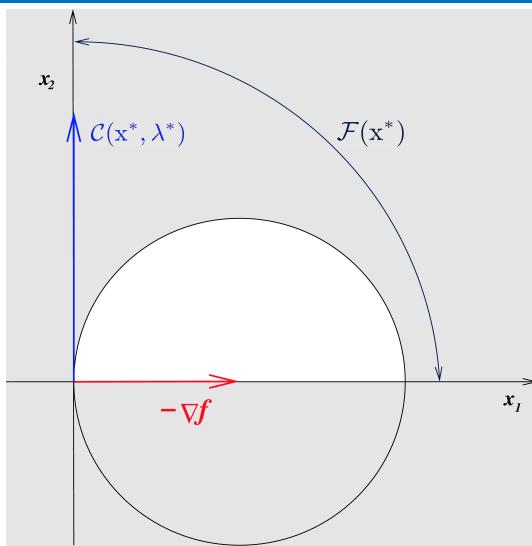
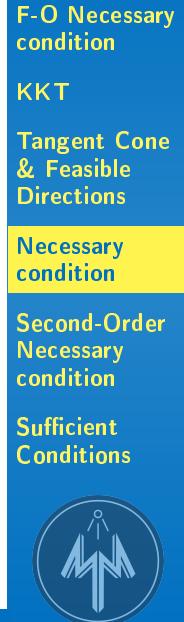


Figure: Problem from Example 7, showing  $\mathcal{F}(x^*)$  and  $\mathcal{C}(x^*, \lambda^*)$ .



## Comments

A geometric picture makes these concepts far clearer. In the example just discussed, the feasible set is the intersection of a half-space and a disk. At the solution point, located on the boundary of both sets, we can visualize the directions allowed by the linearized constraints. These directions form a cone in the first quadrant. Within this cone, however, the critical cone is only a narrow slice: the vertical ray extending upward. The visualization thus makes concrete the distinction between feasible directions and critical directions.

The feasible cone tells us where we can move infinitesimally without immediately violating constraints, but the critical cone further filters this set to the directions where the first-order change in the objective vanishes. The picture shows this filtering vividly: the wide region of feasible movements collapses to a single line of critical interest.

Such illustrations are extremely helpful for building intuition about the geometry of optimality conditions. They emphasize that the interplay between constraints and multipliers is not abstract but manifests directly in the shape of allowable movement near the solution. This geometric perspective provides a bridge between algebraic definitions and practical understanding, preparing us for the more general second-order theory.

### Theorem 23: Second-Order Necessary Conditions

Suppose  $x^*$  is a local solution of the problem

$$\min_{x \in \Omega} f(x), \quad \Omega = \{x \in \mathbb{R}^n \mid c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\} \quad (*)$$

and that LICQ holds. Let  $\lambda^*$  be a Lagrange multiplier satisfying the KKT conditions (7). Then:

$$w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w \geq 0 \quad \text{for all } w \in \mathcal{C}(x^*, \lambda^*).$$

**Key idea:** At a local solution  $x^*$ , the Hessian of the Lagrangian must have nonnegative curvature along all critical directions (that is, the directions in  $\mathcal{C}(x^*, \lambda^*)$ ).

**Proof:** Note that for a local solution  $x^*$ , any feasible sequence  $\{z_k\} \rightarrow x^*$  must satisfy  $f(z_k) \geq f(x^*)$  for all  $k$  sufficiently large.

Since  $w \in \mathcal{C}(x^*, \lambda^*) \subset \mathcal{F}(x^*)$ , we can use the technique in the proof of Lemma 5, to choose a feasible sequence  $\{z_k\} \rightarrow x^*$  and positive scalars  $\{t_k\}$  such that

$$\lim_{k \rightarrow \infty} \frac{z_k - x^*}{t_k} = w, \quad \text{or equivalently, } z_k - x^* = t_k w + o(t_k).$$

28/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

We now turn to the second-order necessary condition. Recall that we already have a candidate point, which we call  $x^*$ , and a multiplier vector, which we call  $\lambda^*$ , that satisfy the Karush–Kuhn–Tucker conditions. Our goal is to show that at a local minimizer the Lagrangian must have nonnegative curvature along every critical direction. Concretely, for any direction  $w$  in the critical cone, the quadratic form given by  $w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w$  must be greater than or equal to zero.

The proof is based on building feasible approximations along  $w$ : starting from a critical direction  $w$ , we build a feasible sequence of points that approach  $x^*$  with steps proportional to  $w$ . To do this we use the tangent/feasible construction: choose positive scalars  $t_k$  that go to zero and define points  $z_k = x^* + t_k w$  plus higher-order terms.

By the definition of critical directions, these  $z_k$  can be chosen feasible, so the objective values at  $z_k$  cannot be less than  $f(x^*)$  for large  $k$ . This feasibility and limiting construction are the heart of the argument: they let us compare the objective evaluated at  $z_k$  with the Taylor expansion of the Lagrangian around  $x^*$ . The rest of the proof quantifies the second-order term in that expansion and shows it cannot be negative without contradicting local optimality. In short, the plan is: 1) build feasible approximations along  $w$ , 2) expand the Lagrangian to second order, and 3) use optimality to force the quadratic form to be nonnegative.

## Second-Order Necessary Conditions (cont.)

Because of the construction of the feasible sequence  $\{z_k\}$ , we have:

$$c_i(z_k) = t_k \nabla c_i(x^*)^T w, \quad \text{for all } i \in \mathcal{A}(x^*).$$

From this equality and the definition of critical cone (see (8)), it follows that:

$$\begin{aligned} \mathcal{L}(z_k, \lambda^*) &= f(z_k) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^* c_i(z_k) = \\ &= f(z_k) - t_k \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*)^T w = f(z_k). \end{aligned}$$

On the other hand, from Taylor's theorem (Theorem 1) and continuity of the Hessians  $\nabla^2 f$  and  $\nabla^2 c_i$ ,  $i \in \mathcal{E} \cup \mathcal{I}$ , we obtain the expansion:

$$\begin{aligned} \mathcal{L}(z_k, \lambda^*) &= \mathcal{L}(x^*, \lambda^*) + (z_k - x^*)^T \nabla_x \mathcal{L}(x^*, \lambda^*) = \\ &\quad + \frac{1}{2} (z_k - x^*)^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) (z_k - x^*) + o(\|z_k - x^*\|^2). \end{aligned}$$

29/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Continuing the argument, we exploit two linked observations. First, because the constructed sequence  $z_k$  is feasible and because of the linearization of constraints, as we showed in the proof of Lemma 5 the value of each active constraint at  $z_k$  equals  $t_k$  times the inner product of the corresponding active gradient with  $w$ , up to higher orders. Second, by the definition of the critical cone and complementarity, the sum over active indices of  $\lambda^*$  times those linearized constraint values vanishes.

Combining these facts yields an important identity: the Lagrangian evaluated at  $z_k$  with multipliers  $\lambda^*$  equals simply  $f(z_k)$ , because the constraint terms cancel at the relevant order. This equality is extremely useful because it lets us work with the Lagrangian expansion instead of the raw objective. We next apply Taylor's theorem to the Lagrangian about  $x^*$ : the Lagrangian at  $z_k$  equals the Lagrangian at  $x^*$  plus the linear term, which vanishes by stationarity, plus one-half times  $(z_k - x^*)^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) (z_k - x^*)$ , plus little-o of the squared norm.

## Second-Order Necessary Conditions (cont.)

By the complementarity condition ( $\lambda_i^* c_i(x^*) = 0$ , for all  $i \in \mathcal{E} \cup \mathcal{I}$ ), we have:

$$\mathcal{L}(x^*, \lambda^*) = f(x^*).$$

From the stationarity condition, the linear term in the Taylor expansion vanishes:

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0.$$

Using the expansion  $z_k - x^* = t_k w + o(t_k)$ , we obtain:

$$\mathcal{L}(z_k, \lambda^*) = f(x^*) + \frac{1}{2} t_k^2 w^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w + o(t_k^2).$$

Since  $\mathcal{L}(z_k, \lambda^*) = f(z_k)$ , this yields:

$$f(z_k) = f(x^*) + \frac{1}{2} t_k^2 w^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w + o(t_k^2).$$

If  $w^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w < 0$ , then  $f(z_k) < f(x^*)$  for all  $k$  sufficiently large, contradicting local optimality of  $x^*$ . Hence, the condition

$$w^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w \geq 0$$

must hold for all  $w \in \mathcal{C}(x^*, \lambda^*)$ . □

30/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Replacing  $z_k - x^*$  by  $t_k w +$  smaller terms produces a second-order term proportional to  $t_k^2$  times the quadratic form  $w^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w$ , plus higher-order remainders. Because Lagrangian at  $z_k$  equals  $f(z_k)$ , we now have a precise asymptotic expansion of  $f(z_k)$  in powers of  $t_k$ , with the coefficient of  $t^2$  given by one-half of that quadratic form.

We are now in a position to finish the necessary condition. Because  $x^*$  is a local minimizer,  $f(z_k)$  is at least  $f(x^*)$  for all sufficiently large  $k$ . Substituting the Taylor expansion derived previously, we obtain that  $f(x^*) + \frac{1}{2} t_k^2 w^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w +$  smaller order terms is greater than or equal to  $f(x^*)$  for small  $t_k$ . Dividing by  $t_k^2$  and letting  $k \rightarrow \infty$  forces the quadratic form to be nonnegative. If instead that quadratic form were strictly negative, then for sufficiently small  $t_k$  the objective at  $z_k$  would be strictly less than  $f(x^*)$ , contradicting local optimality.

Hence the quadratic form must be greater than or equal to zero for every critical direction  $w$ . Conceptually, this result says that along directions where the first derivative gives no information, the second derivative of the Lagrangian must curve upwards or stay flat — it cannot bend downward. This requirement is a natural higher-order analogue of the stationarity condition: stationarity kills linear terms, and the second-order necessary condition controls the sign of the remaining quadratic terms. We will later contrast this necessary statement with a sufficient condition that strengthens “greater than or equal to” to “strictly greater than” and, in return, guarantees strict local optimality.

Key idea: A feasible point satisfying KKT conditions is a strict local solution if the Lagrangian is strictly convex along all nonzero critical directions.

### Theorem 24: Second-Order Sufficient Conditions

Suppose that for some feasible point  $x^* \in \mathbb{R}^n$ , there exists a multiplier  $\lambda^*$  such that the KKT conditions are satisfied. Suppose also that

$$w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w > 0 \quad \text{for all } w \in \mathcal{C}(x^*, \lambda^*), \quad w \neq 0.$$

Then  $x^*$  is a strict local solution of problem (\*):

$$\min_{x \in \Omega} f(x), \quad \Omega = \{x \in \mathbb{R}^n \mid c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\} \quad (*)$$

- ▶ Sufficient conditions guarantee that  $x^*$  is a local solution to (\*).
- ▶ Unlike necessary conditions, they do not require a constraint qualification.
- ▶ The inequality is strict: " $>$ " instead of " $\geq$ ".

31/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

We now state and interpret a second-order sufficient condition. Suppose we have a feasible point  $x^*$  and a multiplier vector  $\lambda^*$  satisfying the KKT conditions. If the Hessian of the Lagrangian at  $x^*, \lambda^*$  is strictly positive along every nonzero vector in the critical cone — that is, for every nonzero  $w$  in the critical cone the quadratic form  $w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w$  is strictly greater than zero — then  $x^*$  is a strict local minimizer.

Intuitively, this condition says that not only are the linear terms neutralized by stationarity, but the curvature in all previously undecided directions is strictly upward. Any small feasible perturbation then increases the Lagrangian — and, because the constraint linearization and complementarity arguments carry through, increases the objective.

Two useful comments follow. First, unlike the necessary condition, the sufficient condition does not require a constraint qualification: we only need a KKT pair to exist. Second, the inequality is strict: “greater than” rather than “greater than or equal to”. That strictness is what converts nondecrease into a genuine local increase, producing a neighborhood where  $x^*$  yields a strictly smaller objective than any other feasible point.

In practice, checking the sufficient condition provides a certificate of strict local optimality and is widely used in nonlinear programming algorithms to verify convergence to a true minimizer.

**Proof:** Let  $\tilde{\mathcal{C}} = \{d \in \mathcal{C}(x^*, \lambda^*) \mid \|d\| = 1\}$ .

This set is compact, and by the condition

$$d^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) d > 0 \quad \forall d \in \tilde{\mathcal{C}},$$

the minimizer of  $d^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) d$  over  $\tilde{\mathcal{C}}$  is a strictly positive number, say  $\sigma$ .

Since  $\mathcal{C}(x^*, \lambda^*)$  is a cone, we have that  $(w/\|w\|) \in \tilde{\mathcal{C}}$  if and only if  $w \in \mathcal{C}(x^*, \lambda^*)$ ,  $w \neq 0$ . Therefore:

$$w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w \geq \sigma \|w\|^2 \quad \forall w \in \mathcal{C}(x^*, \lambda^*).$$

Now suppose there exists a feasible sequence  $\{z_k\} \rightarrow x^*$  such that

$$f(z_k) < f(x^*) + \frac{\sigma}{2} \|z_k - x^*\|^2 \quad \text{for all } k \text{ sufficiently large.}$$

Let's show that this assumption leads to a contradiction.

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

To establish the second-order sufficient condition, we begin by focusing on the geometry of the critical cone. Consider the normalized cone, denoted  $\tilde{\mathcal{C}}$ , which consists of all unit-length directions that lie inside the critical cone. Because this set is compact, and because the quadratic form defined by  $d^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) d$  is strictly positive for all such directions, we can conclude that this quadratic form has a minimum value on the set.

Importantly, this minimum value, which we call  $\sigma$ , is strictly greater than zero. The interpretation of this fact is straightforward: the Lagrangian does not merely have positive curvature in critical directions, but in fact has uniformly positive curvature bounded away from zero. Extending from the normalized cone to the full cone, we obtain the inequality that for every direction  $w$  in the critical cone, the quadratic form  $w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w \geq \sigma \|w\|^2$ .

This inequality is crucial because it provides a uniform lower bound on curvature in all feasible critical directions. The proof then proceeds by contradiction. Suppose there exists a feasible sequence of points approaching  $x^*$  whose function values fall strictly below  $f(x^*) + \frac{\sigma}{2} \|z_k - x^*\|^2$ . If such a sequence existed, it would mean that the objective could decrease faster than the quadratic lower bound established by the Hessian. Our task is to show that such a situation is impossible, because it would contradict the uniform curvature guaranteed by the second-order sufficient condition.

## Second-Order Sufficient Conditions (cont.)

By taking a subsequence, we get a limit direction:

$$\lim_{k \rightarrow \infty} \frac{z_k - x^*}{\|z_k - x^*\|} = d.$$

Then  $d \in \mathcal{F}(x^*)$  by properties of feasible directions (Lemma 5).

From KKT conditions and feasibility:

$$\mathcal{L}(z_k, \lambda^*) = f(z_k) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* c_i(z_k) \leq f(z_k).$$

Taylor expansion of  $\mathcal{L}(z_k, \lambda^*)$  from Theorem 23 still holds.

If  $d \notin \mathcal{C}(x^*, \lambda^*)$ , then there exists some  $j \in \mathcal{A}(x^*) \cap \mathcal{I}$  such that

$$\lambda_j^* \nabla c_j(x^*)^T d > 0,$$

and for the remaining indices  $i \in \mathcal{A}(x^*)$ , we have

$$\lambda_i^* \nabla c_i(x^*)^T d \geq 0.$$

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

Let us assume that such a contradictory sequence does exist and examine its limiting behavior. Any sequence approaching  $x^*$  can be normalized by dividing by its distance from  $x^*$ , yielding a limiting direction, which we call  $d$ . By construction,  $d$  belongs to the set of feasible directions.

The Karush–Kuhn–Tucker conditions now enter the argument. Since the sequence is feasible, we can express the Lagrangian at each point  $z_k$  as  $f(z_k) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* c_i(z_k)$ . Because the multipliers are nonnegative and constraints are satisfied, this Lagrangian value is always less than or equal to the raw objective. At the same time, the Taylor expansion of the Lagrangian around  $x^*$  still applies.

Now we distinguish two cases: if the limiting direction  $d$  is not in the critical cone, then there must exist an active inequality constraint with positive multiplier such that the gradient of that constraint at  $x^*$ , dotted with  $d$ , is strictly positive. Meanwhile, for the other active constraints, the corresponding dot products are nonnegative. This separation tells us that the candidate direction  $d$  cannot escape the cone without activating a constraint in a way that contributes positively, which becomes incompatible with the assumed inequality for the sequence. In other words, directions outside the cone are eliminated by the structure of the KKT conditions combined with feasibility.

## Second-Order Sufficient Conditions (cont.)

From Taylor's theorem and the limiting direction  $d$ :

$$\begin{aligned}\lambda_j^* c_j(z_k) &= \lambda_j^* c_j(x^*) + \lambda_j^* \nabla c_j(x^*)^\top (z_k - x^*) + o(\|z_k - x^*\|) = \\ &= \|z_k - x^*\| \lambda_j^* \nabla c_j(x^*)^\top d + o(\|z_k - x^*\|).\end{aligned}$$

Then,

$$\begin{aligned}\mathcal{L}(z_k, \lambda^*) &= f(z_k) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* c_i(z_k) \\ &\leq f(z_k) - \lambda_j^* c_j(z_k) \\ &\leq f(z_k) - \|z_k - x^*\| \lambda_j^* \nabla c_j(x^*)^\top d + o(\|z_k - x^*\|).\end{aligned}$$

But Taylor expansion also gives:

$$\mathcal{L}(z_k, \lambda^*) = f(x^*) + O(\|z_k - x^*\|^2),$$

so combining:

$$f(z_k) \geq f(x^*) + \|z_k - x^*\| \lambda_j^* \nabla c_j(x^*)^\top d + o(\|z_k - x^*\|).$$

34/35 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

F-O Necessary condition

KKT

Tangent Cone & Feasible Directions

Necessary condition

Second-Order Necessary condition

Sufficient Conditions



### Comments

To make this exclusion precise, we apply Taylor's theorem to the constraint functions themselves. For the particular active constraint identified, say index  $j$ , the value of  $\lambda_j^* c_j(z_k)$  expands as the distance from  $z_k$  to  $x^*$  multiplied by  $\lambda_j^*$  times the gradient dotted with direction  $d$ , plus smaller terms. Since  $z_k - x^*$  has length equal to the distance from  $z_k$  to  $x^*$ , this expansion simplifies to that distance multiplied by  $\lambda_j^*$  times the gradient dotted with direction  $d$ , plus smaller terms.

Now recall that we previously assumed  $\lambda_j^* \nabla c_j(x^*)^\top d$  is strictly positive. Therefore the contribution of this term grows linearly with the distance from  $x^*$ . Plugging this estimate back into the Lagrangian expression yields an inequality: the Lagrangian at  $z_k$  is bounded above by  $f(z_k)$  – a positive multiple of the distance from  $x^*$ , up to smaller terms. But on the other hand, Taylor expansion of the Lagrangian about  $x^*$  guarantees it equals  $f(x^*)$  + terms of order squared distance. Combining both perspectives, we deduce that  $f(z_k)$  must be at least  $f(x^*)$  + a positive multiple of the distance.

From  $\lambda_j^* \nabla c_j(x^*)^\top d > 0$ , we get a contradiction with our assumption (that is, that  $f(z_k) < f(x^*) + (\sigma/2)\|z_k - x^*\|^2$ ), so  $d \in \mathcal{C}(x^*, \lambda^*)$ , and

$$d^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) d \geq \sigma.$$

Thus, using Taylor expansion and the expression for the limiting direction:

$$\begin{aligned} f(z_k) &\geq f(x^*) + \frac{1}{2}(z_k - x^*)^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*)(z_k - x^*) + o(\|z_k - x^*\|^2) \\ &= f(x^*) + \frac{1}{2}d^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) d \|z_k - x^*\|^2 + o(\|z_k - x^*\|^2) \\ &\geq f(x^*) + (\sigma/2)\|z_k - x^*\|^2 + o(\|z_k - x^*\|^2). \end{aligned}$$

This contradicts our assumption, thus we conclude that every feasible sequence  $\{z_k\}$  approaching  $x^*$  must satisfy  $f(z_k) \geq f(x^*) + (\sigma/2)\|z_k - x^*\|^2$ , for all  $k$  sufficiently large, so  $x^*$  is a strict local solution. □



### Comments

But this contradicts to our earlier assumption that  $f(z_k)$  was strictly less than  $f(x^*) + \frac{\sigma}{2}\|z_k - x^*\|^2$ . Hence directions  $d$  belongs to the critical cone.

In this case, the uniform curvature bound applies: the quadratic form  $d^\top \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) d$  is greater than or equal to  $\sigma$ . Substituting into the Taylor expansion of the Lagrangian, we find that  $f(z_k)$  is bounded below by  $f(x^*) + \frac{1}{2}\sigma$  times this quadratic form times the squared distance, up to higher-order terms.

Since the quadratic form is at least  $\sigma$ , this inequality reduces to  $f(z_k)$  being greater than or equal to  $f(x^*) + \frac{1}{2}\sigma\|z_k - x^*\|^2$ , plus smaller corrections. This clearly contradicts the original assumption that  $f(z_k)$  was strictly less than  $f(x^*) + \frac{1}{2}\sigma\|z_k - x^*\|^2$ . The contradiction shows that no such sequence exists. Therefore, every feasible sequence approaching  $x^*$  must eventually satisfy the inequality  $f(z_k) \geq f(x^*) + \frac{1}{2}\sigma\|z_k - x^*\|^2$ .

This conclusion precisely establishes that  $x^*$  is a strict local minimizer: in a neighborhood around  $x^*$ , no feasible point yields a smaller objective value, and in fact, the objective increases quadratically away from  $x^*$ . This completes the proof of the second-order sufficient condition, which complements the necessary condition studied earlier and provides a robust certificate of strict local optimality.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 7)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025



Санкт-Петербургский  
государственный  
университет



31 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem



### Comments

In this lecture, we continue the study of optimality conditions and move toward duality theory in constrained optimization. We start with examples illustrating second-order sufficient conditions, unbounded objectives, and the role of constraint qualifications such as LICQ and MFCQ in guaranteeing validity of optimality results. Geometric perspectives involving normal cones and necessary conditions are introduced to deepen intuition. We then examine sensitivity analysis through Lagrange multipliers and distinguish between strongly and weakly active constraints. The second part of the lecture develops duality theory, beginning with the Lagrangian dual, weak duality, and the link to KKT conditions, before exploring primal-dual relationships and the Wolfe dual. Finally, we apply these ideas to linear and convex quadratic programming, covering standard form transformations, KKT conditions, dual formulation, and the proof of strong duality, thereby connecting theory with fundamental optimization models.

## Example: Second-Order Sufficient Conditions

### Example 8 (revisiting Example 2)

Objective:  $f(x) = x_1 + x_2$ ,

Constraint:  $c_1(x) = 2 - x_1^2 - x_2^2$

Index sets:  $\mathcal{E} = \emptyset$ ,  $\mathcal{I} = \{1\}$

Lagrangian:  $\mathcal{L}(x, \lambda) = (x_1 + x_2) - \lambda_1(2 - x_1^2 - x_2^2)$

At  $x^* = (-1, -1)^T$ ,  $\lambda_1^* = \frac{1}{2}$  satisfies KKT.

Lagrangian Hessian:

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) = \begin{bmatrix} 2\lambda_1^* & 0 \\ 0 & 2\lambda_1^* \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ is positive definite,}$$

so it certainly satisfies the conditions of Theorem Second-Order Sufficient Conditions (Theorem 24)  $\Rightarrow x^*$  is a strict local solution (and in fact global).

### Example 9

Now let's consider a more complex problem

$$\min -0.1(x_1 - 4)^2 + x_2^2 \quad \text{s.t. } x_1^2 + x_2^2 - 1 \geq 0,$$

in which we seek to minimize a nonconvex function over the exterior of the unit circle.

1/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem



### Comments

Having established the second-order sufficient conditions for optimality, it is natural to verify how they operate in practice. The theory by itself is abstract, but the true power of such results becomes visible when we apply them to explicit problems. Let us therefore revisit a familiar optimization setting and check whether the conditions can indeed confirm the optimality of a candidate solution. This process not only consolidates our understanding of the theorem but also illustrates how local analysis based on derivatives can provide global insights.

In the first case, we minimize a simple linear objective subject to a quadratic inequality. The candidate solution at hand satisfies the Karush–Kuhn–Tucker conditions with an associated multiplier, and when we form the Hessian of the Lagrangian, it turns out to be positive definite. This immediately triggers the sufficient conditions: the point must be a strict local minimizer. In fact, because of the geometry of the feasible set and the simplicity of the linear objective, the solution is not just local but also global.

With this straightforward example confirming the theorem, we turn next to a richer problem where the objective is nonconvex. Specifically, the function contains a negative quadratic term, making it unbounded below if unconstrained. The feasible set is also unusual: optimization takes place not inside but outside the unit circle. This setting forces us to rely on the machinery of optimality conditions, as intuition alone may not be enough to determine whether local solutions exist and how they behave.

## Example: Unbounded Objective and Local Solution

The objective function is not bounded below on the feasible region: e.g., consider the feasible sequence

$$\begin{bmatrix} 10 \\ 0 \end{bmatrix}, \begin{bmatrix} 20 \\ 0 \end{bmatrix}, \begin{bmatrix} 30 \\ 0 \end{bmatrix}, \begin{bmatrix} 40 \\ 0 \end{bmatrix}, \dots \Rightarrow f(x) \rightarrow -\infty$$

⇒ No global solution exists.

Still: A strict local solution on the constraint boundary may exist.

Approach: Use KKT conditions and second-order conditions (Theorem 24).  
Lagrangian derivatives:

$$\nabla_x \mathcal{L}(x, \lambda) = \begin{bmatrix} -0.2(x_1 - 4) - 2\lambda_1 x_1 \\ 2x_2 - 2\lambda_1 x_2 \end{bmatrix},$$
$$\nabla_{xx}^2 \mathcal{L}(x, \lambda) = \begin{bmatrix} -0.2 - 2\lambda_1 & 0 \\ 0 & 2 - 2\lambda_1 \end{bmatrix}$$

2/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem



### Comments

The new example presents an interesting challenge: the objective function is non-convex and, in fact, not bounded below over the feasible region. If we move far enough along the horizontal axis while remaining feasible, the function tends to minus infinity. This observation immediately tells us that a global solution cannot exist. Yet, the absence of a global minimizer does not exclude the possibility of meaningful local solutions. Indeed, optimization theory teaches us that local minima may persist even in nonconvex and unbounded settings.

To identify such candidates, we turn again to the Karush–Kuhn–Tucker framework. By forming the Lagrangian and computing its first and second derivatives, we prepare to test both first- and second-order conditions. The gradient of the Lagrangian gives necessary equations linking the primal variables with the multiplier, while the Hessian describes the curvature of the Lagrangian in the neighborhood of feasible points. At this stage, the analysis is less about guessing and more about systematically applying the theory to filter out points that can legitimately serve as local minimizers.

This example is particularly instructive because it shows that even when the objective “escapes” to negative infinity, local minima constrained to the boundary of the feasible set may still appear. Such solutions are inherently different from global ones, but they are crucial in applications: algorithms for nonlinear optimization typically converge to local solutions, and the theory of second-order conditions gives us tools to certify them.



Point  $x^* = (1, 0)^T$  satisfies KKT conditions with  $\lambda_1^* = 0.3$ , active set  $\mathcal{A}(x^*) = \{1\}$ .

Compute critical cone:

$$\nabla c_1(x^*) = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \Rightarrow \mathcal{C}(x^*, \lambda^*) = \{(0, w_2)^T \mid w_2 \in \mathbb{R}\}$$

Substitute into Hessian, we have for any  $w \in \mathcal{C}(x^*, \lambda^*)$  with  $w_2 \neq 0$ :

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) = \begin{bmatrix} -0.4 & 0 \\ 0 & 1.4 \end{bmatrix} \Rightarrow w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) w = 1.4w_2^2 > 0$$

$\Rightarrow$  Second-order sufficient conditions hold  $\Rightarrow$  by Theorem 24  $x^* = (1, 0)^T$  is a strict local solution.

### Comments

Carrying out the analysis, we arrive at a specific candidate point on the boundary of the feasible set. This point satisfies the Karush–Kuhn–Tucker conditions with a positive multiplier, which already signals that it deserves closer attention. To proceed, we construct the critical cone, representing the feasible directions that remain tangent to the active constraint. This step is essential because second-order sufficient conditions require positivity of the Lagrangian Hessian only along those directions, not in the entire space.

When we substitute the cone directions into the Hessian, the quadratic form turns out to be strictly positive for all nonzero feasible directions. This observation confirms the sufficient condition: the candidate point is indeed a strict local minimizer. Although no global minimum exists, this result demonstrates that the theory successfully identifies local structure even in an unbounded and nonconvex problem.

From a conceptual perspective, the lesson here is significant. Optimization is not always about global best solutions; often it is about characterizing and certifying local optima that arise naturally from constraints and curvature. The second-order sufficient conditions serve exactly this role: they bridge first-order stationarity with genuine optimality. By applying them, we gain both a rigorous test and a deeper understanding of how local minima emerge in constrained optimization.

*Constraint qualifications* ensure that the linearized approximation to the feasible set  $\Omega$  captures its essential shape near  $x^*$ .

- A key case occurs when all *active constraints* are linear:

$$c_i(x) = a_i^T x + b_i, \quad \text{for some } a_i \in \mathbb{R}^n, b_i \in \mathbb{R}.$$

- In this case, the linearized feasible direction set  $\mathcal{F}(x^*)$  accurately represents the feasible set  $\Omega$ .

### Lemma 8: Linear Constraints

Suppose that at some  $x^* \in \Omega$ , all active constraints  $c_i(\cdot)$ ,  $i \in \mathcal{A}(x^*)$ , are linear functions. Then:

$$\mathcal{F}(x^*) = \mathcal{T}_\Omega(x^*).$$

**Proof:** From a previous result (Lemma 5),  $\mathcal{T}_\Omega(x^*) \subset \mathcal{F}(x^*)$ . To show  $\mathcal{F}(x^*) \subset \mathcal{T}_\Omega(x^*)$ , take any  $w \in \mathcal{F}(x^*)$  and prove  $w \in \mathcal{T}_\Omega(x^*)$ . By the definition of  $\mathcal{F}(x^*)$  and the linear constraint form  $c_i(x) = a_i^T x + b_i$ , we have:

$$\mathcal{F}(x^*) = \left\{ d \mid \begin{array}{l} a_i^T d = 0, \quad \text{for all } i \in \mathcal{E}, \\ a_i^T d \geq 0, \quad \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \end{array} \right\}.$$

4/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem



### Comments

When studying constraint qualifications, an especially transparent situation arises when all active constraints at the candidate point are linear. In this case, the linearized feasible direction set is not merely an approximation but in fact coincides exactly with the tangent cone of the feasible set. To see why, recall that each active constraint has the form  $a_i^T x + b_i$ . Because these are linear expressions, the first-order expansion does not introduce any curvature. Therefore, the feasible directions obtained by linearization are already the true feasible directions of the original problem. This observation gives us a powerful simplification: the feasible cone defined by conditions  $a_i^T d = 0$  for equality constraints and  $a_i^T d \geq 0$  for active inequalities is precisely the tangent cone at  $x^*$ . The proof of this result relies on constructing feasible sequences that approach  $x^*$  in directions belonging to the linearized cone. Intuitively, this means that if we take any direction consistent with the linear active constraints, then by moving a small positive distance along this direction we stay inside the feasible region. Thus, the linearized model loses no information about the true geometry of the set. This lemma is an important foundation, because in many optimization problems active constraints do turn out to be linear, such as in linear programming. In those cases, verifying the condition is immediate, and the tangent cone characterization becomes exact. Hence, the gap between approximation and reality disappears, making analysis considerably easier.

## Constraint Qualifications: Linear Constraints (continued)

There exists a positive scalar  $\bar{t}$  such that inactive constraints remain inactive at  $x^* + tw$  for  $t \in [0, \bar{t}]$ :

$$c_i(x^* + tw) > 0, \quad \text{for all } i \in \mathcal{I} \setminus \mathcal{A}(x^*), t \in [0, \bar{t}].$$

Define the feasible sequence:

$$z_k = x^* + \frac{\bar{t}}{k}w, \quad k = 1, 2, \dots$$

For active inequality constraints  $i \in \mathcal{I} \cap \mathcal{A}(x^*)$ , since  $a_i^T w \geq 0$ :

$$c_i(z_k) = c_i(z_k) - \underbrace{c_i(x^*)}_{=0} = a_i^T(z_k - x^*) = \frac{\bar{t}}{k}a_i^T w \geq 0.$$

Thus,  $z_k$  is feasible for active inequality constraints. By choice of  $\bar{t}$ ,  $z_k$  is feasible for inactive inequality constraints  $i \in \mathcal{I} \setminus \mathcal{A}(x^*)$ . For equality constraints  $i \in \mathcal{E}$ :

$$c_i(z_k) = a_i^T z_k + b_i = a_i^T \left( x^* + \frac{\bar{t}}{k}w \right) + b_i = a_i^T x^* + b_i = c_i(x^*) = 0.$$

Hence,  $z_k$  is feasible for each  $k = 1, 2, \dots$ . In addition, we have that

$$\frac{z_k - x^*}{\bar{t}/k} = \frac{(\bar{t}/k)w}{\bar{t}/k} = w,$$

so that indeed  $w$  is the limiting direction of  $\{z_k\}$ . Hence,  $w \in T_{\Omega}(x^*)$ .  $\square$

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



### Comments

Let's prove this lemma. The key idea is to show that any direction allowed by the linearized system can be realized as an actual limiting feasible direction. Continuity yields that for each inactive constraint we can find a small positive scalar, let us call it  $\bar{t}$ , such that moving from  $x^*$  by any step  $t$  in the range between zero and  $\bar{t}$  keeps those constraints inactive. We then define a sequence  $z_k = x^* + \frac{\bar{t}}{k}w$ . As  $k$  grows, these points move closer and closer to  $x^*$ , but they always remain feasible. For the active inequality constraints, feasibility is preserved because the expression  $a_i^T w$  is nonnegative, and hence  $c_i(z_k)$  remains nonnegative. For the equality constraints, feasibility holds exactly, because the linear structure ensures that  $c_i(z_k) = c_i(x^*)$ , which is zero. Consequently, every element of the sequence  $z_k$  lies inside the feasible set. Finally, we compute the limit of the normalized difference between  $z_k$  and  $x^*$ , and we recover exactly the direction  $w$ . This shows that  $w$  is indeed a tangent direction, so it belongs to the tangent cone. The conclusion is that the linearized cone and the tangent cone coincide. This constructive approach highlights not just the logic of the proof but also its geometric meaning: feasible directions can be followed by feasible curves, so the geometry of the feasible set is faithfully captured by the linear approximation whenever constraints are linear.

## Constraint Qualifications: MFCQ

Linear constraints and LICQ are constraint qualifications ensuring  $\mathcal{F}(x^*)$  captures the geometry of  $\Omega$  near  $x^*$ .

- Linear active constraints imply  $\mathcal{F}(x^*) = \mathcal{T}_\Omega(x^*)$ , a constraint qualification neither weaker nor stronger than LICQ.

### Definition: MFCQ

Mangasarian-Fromovitz Constraint Qualification (MFCQ) holds if there exists  $w \in \mathbb{R}^n$  such that:

$$\begin{aligned}\nabla c_i(x^*)^T w &> 0, \quad \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I}, \\ \nabla c_i(x^*)^T w &= 0, \quad \text{for all } i \in \mathcal{E},\end{aligned}$$

and the set  $\{\nabla c_i(x^*), i \in \mathcal{E}\}$  is linearly independent.

- Note the *strict inequality* for active inequality constraints.
- MFCQ is weaker than LICQ:
  - If LICQ holds, the system

$$\begin{aligned}\nabla c_i(x^*)^T w &= 1, \quad \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I}, \\ \nabla c_i(x^*)^T w &= 0, \quad \text{for all } i \in \mathcal{E},\end{aligned}$$

has a solution  $w$ , satisfying MFCQ.

- MFCQ can hold when LICQ does not.

6/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem



### Comments

Beyond the special case of linear constraints, optimization theory introduces broader conditions known as constraint qualifications. These conditions ensure that the linearized approximation reflects the true geometry of the feasible set, even when nonlinear constraints are present. Among the most important is the Mangasarian–Fromovitz Constraint Qualification, or MFCQ. It requires the existence of a vector  $w$  such that for every active inequality constraint, the gradient of the constraint at  $x^*$ , transposed and multiplied by  $w$ , is strictly positive. For equality constraints, the same expression must be zero. Additionally, the set of gradients of equality constraints must be linearly independent. The strict inequality condition is crucial: it demands that we can find a feasible direction pointing strictly into the interior of the feasible set, thereby preventing degeneracy at the boundary. Compared with the Linear Independence Constraint Qualification, or LICQ, MFCQ is weaker. If LICQ holds, then MFCQ automatically follows, because one can solve a simple linear system to find a suitable  $w$ . However, the converse is not true: MFCQ may hold even when LICQ fails, for example when gradients of active inequalities are linearly dependent. This flexibility makes MFCQ especially useful in nonlinear problems, where strict linear independence may be too demanding. In practice, MFCQ guarantees that first-order optimality conditions, such as the Karush–Kuhn–Tucker system, remain valid and meaningful. Thus, it forms a cornerstone of constrained optimization theory.

MFCQ can hold when LICQ fails if active constraint gradients are linearly dependent but a vector  $w$  satisfies MFCQ conditions.

### Example

Consider the feasible set  $\Omega = \{x \in \mathbb{R}^2 \mid c_1(x) = -x_1 \geq 0, c_2(x) = -x_1 - x_2^2 \geq 0\}$  at  $x^* = (0, 0)^T$ .

- ▶ Active constraints:  $\mathcal{A}(x^*) = \{1, 2\}$ , with  $c_1(x^*) = 0, c_2(x^*) = 0$ .
- ▶ Gradients:  $\nabla c_1(x^*) = [-1, 0]^T, \nabla c_2(x^*) = [-1, 0]^T$ .
- ▶ LICQ fails:  $\{\nabla c_1(x^*), \nabla c_2(x^*)\}$  is linearly dependent (identical vectors).
- ▶ MFCQ holds: Choose  $w = [1, 0]^T$ . Then:

$$\begin{aligned}\nabla c_1(x^*)^T w &= [-1, 0][1, 0]^T = -1 < 0, \\ \nabla c_2(x^*)^T w &= [-1, 0][1, 0]^T = -1 < 0.\end{aligned}$$

Adjust  $w = [-1, 0]^T$ :

$$\nabla c_1(x^*)^T w = [-1, 0][-1, 0]^T = 1 > 0, \quad \nabla c_2(x^*)^T w = 1 > 0.$$

No equality constraints ( $\mathcal{E} = \emptyset$ ), so MFCQ is satisfied.

7/31 || SPbU & HIT 2025 || Shpilley P.V. || Classical optimization approaches

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



### Comments

An instructive illustration shows that MFCQ may hold even when LICQ fails. Consider a feasible set in two dimensions defined by two inequality constraints:  $c_1(x) = -x_1 \geq 0$ , and  $c_2(x) = -x_1 - x_2^2 \geq 0$ . At the point  $x^* = (0, 0)^T$ , both constraints are active. The gradients at this point are identical: both equal to the vector  $(-1, 0)^T$ . Because of this, the set of active gradients is linearly dependent, and hence LICQ is violated. Nevertheless, we can still satisfy MFCQ. By choosing the direction  $w = (-1, 0)^T$ , we compute the gradient transpose times  $w$  for both constraints and obtain the value 1, which is strictly positive. Thus, all active inequalities admit a feasible inward direction. Since no equality constraints are present, the remaining conditions of MFCQ are trivially satisfied. This example makes clear that MFCQ is more flexible than LICQ: linear dependence among gradients of active inequalities does not necessarily preclude the existence of a vector that strictly points inward. Geometrically, what matters is not independence but the ability to find a genuine feasible direction that pushes away from the boundary. Therefore, MFCQ provides a robust foundation for analyzing nonlinear constrained problems, ensuring that optimality conditions can still be applied in the cases when LICQ fails.

Constraint qualifications are sufficient but not necessary for  $\mathcal{F}(x^*)$  to capture the geometry of  $\Omega$  near  $x^*$ .

### Example

Consider the feasible set  $\Omega = \{x \in \mathbb{R}^2 \mid x_2 \geq -x_1^2, x_2 \leq x_1^2\}$  at  $x^* = (0, 0)^T$ .

- ▶ No constraint qualifications (e.g., LICQ, MFCQ) are satisfied.
- ▶ Linear approximation:  $\mathcal{F}(x^*) = \{(w_1, 0)^T \mid w_1 \in \mathbb{R}\}$ .
- ▶ This accurately reflects the geometry of  $\Omega$  near  $x^*$ .
- ▶ Geometric optimality condition for:

$$\min f(x) \quad \text{subject to } x \in \Omega.$$

Depends only on the geometry of  $\Omega$ , not its algebraic description.

### Definition: Normal Cone

The normal cone to  $\Omega$  at  $x \in \Omega$  is:

$$\mathcal{N}_\Omega(x) = \{v \mid v^T w \leq 0 \text{ for all } w \in \mathcal{T}_\Omega(x)\},$$

where  $\mathcal{T}_\Omega(x)$  is the tangent cone. Each  $v \in \mathcal{N}_\Omega(x)$  is a normal vector.

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



### Comments

When we talk about optimality under constraints, an important subtlety arises. The conditions we often impose, such as the Linear Independence Constraint Qualification or the Mangasarian–Fromovitz Constraint Qualification, are designed to guarantee that the tangent cone captures the geometry of the feasible region near a point. But in fact, these conditions are not always necessary. Sometimes, even when no constraint qualification is satisfied, the tangent cone still gives an accurate description of the local geometry.

Consider the example of the feasible region defined by the inequalities  $x_2 \geq -x_1^2$  and  $x_2 \leq x_1^2$ . At the origin, both constraints “touch” in such a way that no standard qualification holds. Yet, if we approximate the feasible set linearly, we obtain the set of all vectors of the form  $(w_1, 0)^T$ . This simple one-dimensional subspace perfectly matches the geometry of the feasible set at that point. This teaches us that the essential feature is not whether the algebraic conditions are satisfied, but whether the tangent cone aligns with the true local geometry.

This brings us naturally to the notion of the normal cone. Just as the tangent cone describes directions that remain feasible, the normal cone describes directions that are “orthogonal” in a generalized sense. Formally, a vector belongs to the normal cone at a point if its inner product with every tangent direction is nonpositive. Intuitively, we can think of these vectors as those that “push against” the feasible set, restricting motion. They represent generalized outward normals, even in nonsmooth or nonconvex settings.

This geometric perspective shifts our attention away from the algebraic form of constraints toward the underlying shape of the feasible region. Optimality conditions then become conditions on the relationship between the gradient of the objective and the normal cone, rather than on the specific constraint equations.

## Geometric Necessary Condition

A geometric first-order condition relies on the normal cone  $\mathcal{N}_\Omega(x^*)$  to characterize local minimizers.

### Theorem 25: Geometric Necessary Condition

Suppose  $x^*$  is a local minimizer of  $f$  in  $\Omega$ . Then:

$$-\nabla f(x^*) \in \mathcal{N}_\Omega(x^*).$$

#### Proof:

- For any  $d \in \mathcal{T}_\Omega(x^*)$ , there exist  $\{t_k\}$ ,  $\{z_k\}$  (by the definition) such that:

$$z_k \in \Omega, \quad z_k = x^* + t_k d + o(t_k).$$

- Since  $x^*$  is a local minimizer,  $f(z_k) \geq f(x^*)$  for all  $k$  sufficiently large.

- By Taylor's theorem, with  $f$  continuously differentiable:

$$f(z_k) - f(x^*) = t_k \nabla f(x^*)^T d + o(t_k) \geq 0.$$

- Divide by  $t_k$  and taking limits as  $k \rightarrow \infty$ , we have:

$$\nabla f(x^*)^T d \geq 0.$$

- Thus,  $-\nabla f(x^*)^T d \leq 0$  for all  $d \in \mathcal{T}_\Omega(x^*)$ .

- By the normal cone definition,  $-\nabla f(x^*) \in \mathcal{N}_\Omega(x^*)$ . □

9/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem



## Comments

Once we have the normal cone, we can state the fundamental geometric necessary condition for constrained optimization. Suppose we are given a differentiable objective function and a feasible region. If a point is a local minimizer of the function over the feasible set, then its gradient must interact with the geometry of the set in a very specific way. Concretely, the negative gradient at the minimizer must belong to the normal cone of the feasible region at that point. To understand this condition, recall that the gradient of the function indicates the steepest ascent direction. Its negative, therefore, points toward steepest descent. For the point to be a minimizer subject to constraints, there can be no feasible descent direction available. This is precisely captured by the inclusion of the negative gradient in the normal cone: it ensures that the inner product between the gradient and any tangent direction is nonnegative.

The proof of this condition relies on the definition of the tangent cone. If we take any feasible tangent direction, we can approximate feasible points along that direction by constructing sequences that remain inside the feasible set. Because the candidate point is assumed to be a minimizer, the function values at those nearby feasible points cannot fall below the value at the minimizer. By Taylor expansion, the first-order term governing this change is the step size multiplied by the dot product of the gradient with the tangent direction. Since the difference in function values is nonnegative, this dot product must also be nonnegative for all feasible directions. Rearranging the inequality, we obtain that the negative gradient produces nonpositive dot products with every tangent direction. This is exactly the definition of membership in the normal cone. Thus, the geometric condition concisely encodes the idea that at a constrained minimizer, descent directions are blocked by the geometry of the feasible region.

Under LICQ, the normal cone  $\mathcal{N}_\Omega(x^*)$  is directly related to the cone of active constraint gradients.

### Lemma 9: Normal Cone under LICQ

Suppose the LICQ holds at  $x^*$ . Then the normal cone is:

$$\mathcal{N}_\Omega(x^*) = -\mathcal{N},$$

where  $\mathcal{N}$  is the cone of active constraint gradients.

#### Proof:

- From Farkas' Lemma, for  $g \in \mathcal{N}$ :

$$g^T d \geq 0 \text{ for all } d \in \mathcal{F}(x^*).$$

- Since LICQ holds,  $\mathcal{F}(x^*) = \mathcal{T}_\Omega(x^*)$  (by Lemma 5).

- Thus, for  $g \in -\mathcal{N}$ :

$$g^T d \leq 0 \text{ for all } d \in \mathcal{T}_\Omega(x^*).$$

- By the normal cone definition,  $\mathcal{N}_\Omega(x^*) = \{v \mid v^T d \leq 0 \text{ for all } d \in \mathcal{T}_\Omega(x^*)\}$ .

- Hence,  $\mathcal{N}_\Omega(x^*) = -\mathcal{N}$ . □



### Comments

The concept of the normal cone becomes especially powerful when combined with constraint qualifications. Under the Linear Independence Constraint Qualification, or LICQ, the geometry of the feasible set aligns cleanly with the gradients of the active constraints. In this setting, the normal cone at a point can be described directly in terms of those gradients. Specifically, it is equal to the negative of the cone generated by the active constraint gradients. Intuitively, each active constraint has a gradient vector pointing into the feasible region. Taking nonnegative combinations of these gradients describes all directions that push against the feasible set. By including the negative sign, we obtain the set of vectors that oppose feasible tangent directions, which is exactly what the normal cone represents.

The formal justification uses Farkas' Lemma. This result connects feasibility of linear inequalities with the existence of certain nonnegative multipliers. Applying it here, one shows that if a vector belongs to the cone generated by active gradients, then its dot product with every feasible direction is nonnegative. Under LICQ, the linearized feasible cone coincides with the tangent cone, so this reasoning extends precisely to the geometry of the set. Reversing the sign, the negative of that cone characterizes all vectors that satisfy the defining inequality of the normal cone. The result is both elegant and practical: instead of reasoning directly with tangent cones, which can be abstract, one can rely on a concrete description in terms of constraint gradients.

Lagrange multipliers  $\lambda_i^*$  measure the sensitivity of  $f(x^*)$  to perturbations in constraint  $c_i$ .

- For inactive constraints ( $c_i(x^*) > 0, i \notin \mathcal{A}(x^*)$ ):
  - $x^*$  and  $f(x^*)$  are unaffected by small perturbations.
  - $\lambda_i^* = 0$  (KKT conditions) indicates no significance.

- For active constraint  $i \in \mathcal{A}(x^*)$ , perturb  $c_i(x) \geq 0$  to  $c_i(x) \geq -\epsilon \|\nabla c_i(x^*)\|$ :

$$-\epsilon \|\nabla c_i(x^*)\| \approx (x^*(\epsilon) - x^*)^T \nabla c_i(x^*),$$

$$0 \approx (x^*(\epsilon) - x^*)^T \nabla c_j(x^*), \quad j \in \mathcal{A}(x^*), \forall j \neq i.$$

- Objective change:

$$\begin{aligned} f(x^*(\epsilon)) - f(x^*) &\approx (x^*(\epsilon) - x^*)^T \nabla f(x^*) = \\ &= \sum_{j \in \mathcal{A}(x^*)} \lambda_j^* (x^*(\epsilon) - x^*)^T \nabla c_j(x^*) \approx -\epsilon \|\nabla c_i(x^*)\| \lambda_i^*. \end{aligned}$$

- Sensitivity:

$$\frac{df(x^*(\epsilon))}{d\epsilon} = -\lambda_i^* \|\nabla c_i(x^*)\|.$$

- Large  $\lambda_i^* \|\nabla c_i(x^*)\|$ : High sensitivity to  $c_i$ .
- $\lambda_i^* = 0$  for active constraint: negligible first-order effect.



## Comments

Having established the link between the normal cone and active constraint gradients, we now turn to the interpretation of Lagrange multipliers. Beyond their role in optimality conditions, multipliers have a powerful sensitivity interpretation. Each multiplier associated with an active constraint measures how strongly the objective value responds to perturbations of that constraint. If a constraint is inactive at the solution, then loosening or tightening it slightly has no effect on the optimal point or its function value. In this case, the multiplier is zero, which aligns with the Karush–Kuhn–Tucker conditions.

For an active constraint, the situation is more subtle. Imagine perturbing the constraint by shifting its right-hand side outward by a small parameter  $\epsilon$ , scaled by the norm of its gradient. This perturbation moves the feasible boundary slightly. The corresponding optimal point shifts accordingly, and the first-order change in the objective function can be computed. It turns out that this change is approximately  $-\epsilon$  times the multiplier associated with that constraint, multiplied by the gradient's norm. Thus, the multiplier quantifies how sensitive the optimal value is to small relaxations of the constraint. A large multiplier signals that even a tiny relaxation of the constraint yields a significant improvement in the objective. Conversely, a zero multiplier for an active constraint indicates that relaxing it has negligible effect at first order.

This interpretation is crucial in applications. In economics, multipliers are often called shadow prices, because they measure the marginal value of resources represented by constraints. In engineering, they indicate which constraints are truly binding and which can be relaxed without much impact. In both settings, sensitivity analysis guided by multipliers provides insight far beyond the feasibility of optimality conditions—it highlights the practical importance of individual constraints.

Lagrange multipliers distinguish strongly and weakly active constraints and maintain sensitivity under constraint scaling.

## Definition: Strongly and Weakly Active Constraints

For a solution  $x^*$  of  $\min f(x)$  s.t.  $c_i(x) = 0, i \in \mathcal{E}, c_i(x) \geq 0, i \in \mathcal{I}$ , with KKT conditions satisfied we say that an inequality constraint  $c_i$  is

- ▶ **strongly active** (or *binding*) if  $i \in \mathcal{A}(x^*)$ ,  $\lambda_i^* > 0$  for some  $\lambda^*$  satisfying KKT conditions;
- ▶ **weakly active**:  $i \in \mathcal{A}(x^*)$ ,  $\lambda_i^* = 0$  for all  $\lambda^*$  satisfying KKT conditions.

- ▶ Sensitivity analysis is invariant to constraint scaling:

- ▶ Replace  $c_i$  by  $10c_i$ : Same feasible set,  $\lambda_i^* \rightarrow \lambda_i^*/10$ ,  $\|\nabla c_i(x^*)\| \rightarrow 10\|\nabla c_i(x^*)\|$ .
- ▶ Product  $\lambda_i^* \|\nabla c_i(x^*)\|$  unchanged.
- ▶ Scaling objective  $f$  by 10f:
  - ▶  $\lambda_i^* \rightarrow 10\lambda_i^*$ , increasing sensitivity  $\frac{df(x^*(\epsilon))}{d\epsilon} = -\lambda_i^* \|\nabla c_i(x^*)\|$  by 10.

12/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Example: S-O  
Suf. Cond.**

**Constraint Qualifications**

**Sensitivity Analysis**

**Duality Theory**

**Wolfe Dual**

**LP**

**KKT Conditions**

**Dual Problem**



## Comments

When we examine inequality constraints in optimization, it is important to distinguish between those that truly shape the optimal solution and those that, although satisfied with equality, exert no real “force” on the problem. This leads us to the notions of strongly and weakly active constraints. A constraint is called strongly active (or binding) if, at the optimal point, its corresponding Lagrange multiplier is strictly positive. This means that the constraint plays a crucial role: relaxing it would improve the objective, while tightening it would make the problem infeasible. In contrast, a constraint is weakly active if its multiplier is zero for all multiplier choices that satisfy the Karush–Kuhn–Tucker conditions. In such a case, the constraint touches the solution without actually influencing it. An important property here is the invariance of sensitivity analysis under scaling of constraints. If we multiply a constraint function by a constant factor, the feasible region remains unchanged, and the multiplier rescales accordingly, while the product of the multiplier and the gradient norm stays the same. This shows that the dual information is well-behaved and does not depend on arbitrary choices of constraint scaling. On the other hand, scaling the objective function by a factor directly scales the multipliers and therefore alters sensitivity results. This distinction highlights that multipliers reflect relative importance with respect to the objective, not absolute magnitudes of the constraints themselves.

## Duality Theory: Introduction

Duality theory constructs an alternative problem to provide insights, bounds, or algorithms for the original problem.

- Duality motivates algorithms (e.g., augmented Lagrangian) and applies to convex and discrete optimization.
- Dual problem may be easier to solve or provide a lower bound on the primal objective.

Consider the problem with no equality constraints, convex  $f$ , and  $-c_i$ :

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad c_i(x) \geq 0, \quad i = 1, 2, \dots, m.$$

Constraint vector:

$$c(x) = \begin{bmatrix} c_1(x) \\ c_2(x) \\ \vdots \\ c_m(x) \end{bmatrix}.$$

Simplified form:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad c(x) \geq 0.$$

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem



### Comments

The analysis of constraint qualifications naturally leads us toward a deeper structural concept in optimization — the theory of duality. Duality theory is one of the central ideas in optimization. The main idea is that instead of solving the original, or primal, problem directly, one can formulate a related but different problem called the dual problem. The dual problem provides valuable insights, such as lower bounds on the primal objective, conditions for optimality, and in some cases computational simplification. In convex optimization, the duality framework is especially powerful, since strong theoretical guarantees often hold. For instance, under certain regularity conditions, the optimal values of the primal and dual problems coincide. But duality is not limited to convex optimization: ideas of dual formulations appear even in discrete and combinatorial problems, where they serve as a basis for relaxation techniques and approximation methods. A key motivation for developing duality theory comes from algorithms. Many modern optimization methods, such as the augmented Lagrangian method or interior-point methods, make essential use of dual variables and dual functions. To illustrate the setup, let us focus on inequality-constrained convex problems. We can write such a problem in compact form by collecting all constraints into a single vector inequality. This representation simplifies notation and sets the stage for constructing the Lagrangian, which will serve as the bridge between the primal and dual perspectives. In doing so, we begin to see how constraints can be encoded into the objective function with multipliers, thereby opening a path toward the dual problem.

The dual problem maximizes a convex Lagrangian-based objective, often simplifying computation.

- Lagrangian for  $\lambda \in \mathbb{R}^m$ :

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x).$$

- Dual objective:

$$q(\lambda) = \inf_x \mathcal{L}(x, \lambda).$$

- Computing  $\inf_x \mathcal{L}(x, \lambda)$  is often difficult.

- If  $f, -c_i$  convex and  $\lambda \geq 0$ ,  $\mathcal{L}(\cdot, \lambda)$  is convex, so local minimizers are global, making  $q(\lambda)$  practical.

- Domain of  $q$ :

$$\mathcal{D} = \{\lambda \mid q(\lambda) > -\infty\}.$$

- Dual problem:

$$\max_{\lambda \in \mathbb{R}^m} q(\lambda) \quad \text{s.t.} \quad \lambda \geq 0.$$

**Example: S-O**

**Suf. Cond.**

**Constraint Qualifications**

**Sensitivity Analysis**

**Duality Theory**

**Wolfe Dual**

**LP**

**KKT Conditions**

**Dual Problem**



## Comments

The core tool for moving from the primal problem to the dual is the Lagrangian function. By combining the objective function with the constraints weighted by non-negative multipliers, we construct a function that encodes both feasibility and optimality information. For a given multiplier vector, minimizing the Lagrangian over the decision variables defines the dual function. This dual function gives, for each multiplier choice, a bound on the primal objective value. Because it is formed as an infimum of affine functions in the multipliers, the dual function is always concave, even when the primal problem is not convex. This universal concavity makes the dual problem a maximization problem: we seek the best lower bound on the primal by maximizing the dual function over nonnegative multipliers. An essential practical consideration is the domain of the dual function. If the Lagrangian can be driven to negative infinity, then the corresponding multipliers are not meaningful and must be excluded. Therefore, the effective domain of the dual function is defined by all multipliers for which the infimum is finite. When both the objective and constraints are convex, minimizing the Lagrangian with respect to the primal variables is tractable, and the dual function can often be evaluated effectively. The resulting dual problem then provides a powerful tool: it is convex by construction, and its solution supplies both a bound on the primal and insights into the role of constraints through the values of the multipliers.

The dual problem simplifies optimization by maximizing a derived objective.

### Example

Consider the problem:

$$\min_{x_1, x_2} 0.5(x_1^2 + x_2^2) \quad \text{s.t.} \quad x_1 - 1 \geq 0.$$

- Lagrangian:  $\mathcal{L}(x_1, x_2, \lambda_1) = 0.5(x_1^2 + x_2^2) - \lambda_1(x_1 - 1)$ .
- Convex in  $(x_1, x_2)$ , so infimum at  $\partial\mathcal{L}/\partial x_1 = 0, \partial\mathcal{L}/\partial x_2 = 0$ :

$$x_1 - \lambda_1 = 0, \quad x_2 = 0.$$

- Dual objective:  $q(\lambda_1) = -0.5\lambda_1^2 + \lambda_1$ .

- Dual problem:

$$\max_{\lambda_1 \geq 0} -0.5\lambda_1^2 + \lambda_1.$$

- Solution:  $\lambda_1 = 1$ .

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



### Comments

To clarify the ideas considered, let us examine a simple quadratic programming example with one inequality constraint. The primal problem asks us to minimize a quadratic objective subject to a linear inequality. The Lagrangian combines the quadratic term with the constraint, scaled by a nonnegative multiplier. Because the quadratic objective is convex and the constraint linear, the Lagrangian is convex in the primal variables, making the search for its infimum straightforward. By setting the derivatives with respect to the primal variables equal to zero, we obtain explicit expressions for the minimizers as functions of the multiplier. Substituting these back into the Lagrangian yields the dual function, which in this case is a concave quadratic in the multiplier. The dual problem then becomes a one-dimensional concave maximization subject to nonnegativity. Solving it produces an explicit optimal multiplier. This example is illuminating because it demonstrates all the key steps of duality in a transparent way: construction of the Lagrangian, computation of the dual function, and formulation of the dual maximization problem. Moreover, it highlights the power of duality: although the primal is already simple, the dual provides an alternative lens and shows how constraints influence the solution through multipliers. In more complex problems, this same process allows us to transform a difficult constrained minimization into a potentially simpler maximization, often revealing structural properties that are otherwise hidden in the primal form.

The dual objective  $q$  is concave, and its optimal value bounds the primal objective.

### Theorem 26: Concavity of Dual Objective

The dual objective  $q(\lambda) = \inf_x \mathcal{L}(x, \lambda) = \inf_x [f(x) - \lambda^T c(x)]$  is concave, and its domain  $\mathcal{D} = \{\lambda \mid q(\lambda) > -\infty\}$  is convex.

**Proof:** For any  $\lambda^0, \lambda^1$  in  $\mathbb{R}^m$ , any  $x \in \mathbb{R}^n$ , and any  $\alpha \in [0, 1]$ , we have:

$$\mathcal{L}(x, (1-\alpha)\lambda^0 + \alpha\lambda^1) = (1-\alpha)\mathcal{L}(x, \lambda^0) + \alpha\mathcal{L}(x, \lambda^1).$$

Taking infimum of both sides in this expression over  $x$ :

$$q((1-\alpha)\lambda^0 + \alpha\lambda^1) \geq (1-\alpha)q(\lambda^0) + \alpha q(\lambda^1).$$

Confirms  $q$  is concave. If both  $\lambda^0$  and  $\lambda^1$  belong to  $\mathcal{D}$ , this inequality implies that  $q((1-\alpha)\lambda^0 + \alpha\lambda^1) \geq -\infty$  also, and therefore  $(1-\alpha)\lambda^0 + \alpha\lambda^1 \in \mathcal{D}$ , verifying convexity of  $\mathcal{D}$ .  $\square$



### Comments

An essential property of the dual function is its concavity. To understand why this holds, recall that the dual function is defined as the infimum over the decision variables of the Lagrangian. Explicitly, the dual objective, written as  $q(\lambda) = \inf_x [f(x) - \lambda^T c(x)]$ , takes the lowest possible value of the Lagrangian for a fixed multiplier vector. Because the Lagrangian is affine in the multiplier variable, the resulting function is concave in  $\lambda$ . The proof uses a simple convex combination argument: if we take any two multipliers, call them  $\lambda^0$  and  $\lambda^1$ , and combine them with a weight  $\alpha$  between zero and one, then the value of the Lagrangian at this mixture is the same mixture of the two original Lagrangians. When we then take the infimum over all  $x$ , the inequality reverses into the definition of concavity. This establishes that  $q$  is concave. Furthermore, the effective domain of  $q$ , meaning the set of multipliers for which the infimum is not negative infinity, is a convex set. This is because the infimum respects convex combinations in exactly the same way. Concavity of the dual is a universal result: it does not require convexity of the primal problem, and it guarantees that the dual problem always takes the form of a maximization over a concave function. This structural fact is one of the reasons duality theory is so robust. Even in problems where the primal objective is nonconvex and difficult to analyze, the dual problem inherits a convex structure automatically. Therefore, the dual provides not only theoretical insights but also computational leverage, since optimization of a concave function is well-posed and tractable.

**Theorem 27: Weak Duality**

For any feasible  $\bar{x}$  in  $\min_x f(x)$  s.t.  $c(x) \geq 0$  and  $\bar{\lambda} \geq 0$ ,  $q(\bar{\lambda}) \leq f(\bar{x})$ .

**Proof:**  $q(\bar{\lambda}) = \inf_x [f(x) - \bar{\lambda}^T c(x)] \leq f(\bar{x}) - \bar{\lambda}^T c(\bar{x})$  where the final inequality follows from  $\bar{\lambda} \geq 0$  and  $c(\bar{x}) \geq 0$ .  $\square$

► KKT conditions for the primal problem:

$$\nabla f(\bar{x}) - \nabla c(\bar{x})\bar{\lambda} = 0, \quad (9a)$$

$$c(\bar{x}) \geq 0, \quad (9b)$$

$$\bar{\lambda} \geq 0, \quad (9c)$$

$$\bar{\lambda}_i c_i(\bar{x}) = 0, \quad i = 1, 2, \dots, m, \quad (9d)$$

where  $\nabla c(x)$  is the  $n \times m$  matrix defined by  $\nabla c(x) = [\nabla c_1(x), \dots, \nabla c_m(x)]$ .

**Theorem 28: Solutions of the Dual Problem**

Suppose that  $\bar{x}$  is a solution of  $\min_{x \in \mathbb{R}^n} f(x)$  s.t.  $c(x) \geq 0$  and that  $f$  and  $-c_i$ ,  $i = 1, 2, \dots, m$  are convex functions on  $\mathbb{R}^n$  that are differentiable at  $\bar{x}$ . Then any  $\bar{\lambda}$  for which  $(\bar{x}, \bar{\lambda})$  satisfies the KKT conditions (9) is a solution of the dual problem  $\max_{\lambda \in \mathbb{R}^m} q(\lambda)$  s.t.  $\lambda \geq 0$ .

17/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem

**Comments**

The next central result is the principle of weak duality. According to this principle for any feasible point of the primal problem and any nonnegative multiplier vector, the value of the dual function does not exceed the value of the primal objective. The proof is straightforward: substituting  $\bar{x}$  into the definition of  $q$  yields  $f(\bar{x}) - \bar{\lambda}^T c(\bar{x})$ . Because feasibility requires that  $c(\bar{x})$  is greater than or equal to zero, and because  $\bar{\lambda}$  is also nonnegative, the term  $\bar{\lambda}^T c(\bar{x})$  is nonnegative. Thus,  $q(\bar{\lambda})$  is bounded above by  $f(\bar{x})$ . Weak duality has profound consequences: it guarantees that the dual problem always provides a lower bound on the primal objective. This bound may or may not be tight, but it gives us a systematic way to certify optimality. The link between primal and dual solutions is formalized by the Karush–Kuhn–Tucker conditions. These conditions include stationarity, which requires that the gradient of  $f$  at  $\bar{x}$  equals the gradient of  $c$  at  $\bar{x}$  multiplied by  $\bar{\lambda}$ ; primal feasibility, which requires  $c(\bar{x})$  to be nonnegative; dual feasibility, which requires  $\bar{\lambda}$  to be nonnegative; and complementary slackness, which states that  $\bar{\lambda}_i$  times  $c_i(\bar{x})$  equals zero for every constraint index  $i$ . Together these four conditions provide a necessary system that any primal–dual optimal pair must satisfy. When convexity holds, they are also sufficient. This interplay of weak duality and KKT conditions is the foundation of modern optimization, guiding both theoretical understanding and algorithm design.

The Karush–Kuhn–Tucker conditions not only characterize optimal solutions but also reveal the deep link between primal and dual problems. Suppose a pair consisting of  $\bar{x}$  and  $\bar{\lambda}$  satisfies the KKT system. Then, under convexity assumptions, we can show that the value of the primal objective at  $\bar{x}$  equals the value of the dual function at  $\bar{\lambda}$ . This equality, often called strong duality, means that both problems achieve the same optimal value.

KKT conditions and strict convexity link primal and dual problem solutions.

**Proof:**

Suppose  $(\bar{x}, \bar{\lambda})$  satisfies KKT conditions:

$$\nabla f(\bar{x}) - \nabla c(\bar{x})\bar{\lambda} = 0, \quad c(\bar{x}) \geq 0, \quad \bar{\lambda} \geq 0, \quad \bar{\lambda}_i c_i(\bar{x}) = 0, \quad i = 1, 2, \dots, m.$$

- Since  $\bar{\lambda} \geq 0$ ,  $\mathcal{L}(\cdot, \bar{\lambda})$  is convex, so for any  $x$ ,  $\mathcal{L}(x, \bar{\lambda}) \geq \mathcal{L}(\bar{x}, \bar{\lambda}) + \nabla_x \mathcal{L}(\bar{x}, \bar{\lambda})^T(x - \bar{x}) = \mathcal{L}(\bar{x}, \bar{\lambda}) + (\nabla f(\bar{x}) - \nabla c(\bar{x})\bar{\lambda})^T(x - \bar{x}) = \mathcal{L}(x, \bar{\lambda}).$
- Thus,  $q(\bar{\lambda}) = \inf_x \mathcal{L}(x, \bar{\lambda}) = \mathcal{L}(\bar{x}, \bar{\lambda}) = f(\bar{x}) - \bar{\lambda}^T c(\bar{x}) = f(\bar{x}).$
- By weak duality (Theorem 27),  $q(\lambda) \leq f(\bar{x})$  for all  $\lambda \geq 0$ , so  $q(\bar{\lambda}) = f(\bar{x})$  implies  $\bar{\lambda}$  solves  $\max_{\lambda \geq 0} q(\lambda)$ .  $\square$

If  $f, c_i$  are continuously differentiable and a constraint qualification such as LICQ holds at  $\bar{x}$ , then an optimal Lagrange multiplier is guaranteed to exist.

Dual-to-primal: If  $\mathcal{L}(\cdot, \bar{\lambda})$  is strictly convex (e.g.,  $f$  strictly convex or  $c_i$  strictly convex with  $\bar{\lambda}_i > 0$ ), dual solutions can yield primal solutions.

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



## Comments

The proof proceeds as follows. Since  $\bar{\lambda}$  is nonnegative, the Lagrangian with multiplier  $\bar{\lambda}$  is convex in the primal variables. By the definition of convexity, the Lagrangian at any  $x$  is greater than or equal to the Lagrangian at  $\bar{x}$  plus the gradient term. But the stationarity condition ensures that this gradient term vanishes, leaving us with the inequality that the Lagrangian at any  $x$  is at least as large as the Lagrangian at  $\bar{x}$ . Thus, the infimum over  $x$  is achieved at  $\bar{x}$ , and  $q(\bar{\lambda})$  equals the Lagrangian of  $\bar{x}, \bar{\lambda}$ . By complementary slackness, this equals  $f(\bar{x})$ . Therefore,  $q(\bar{\lambda}) = f(\bar{x})$ . Weak duality then guarantees that  $\bar{\lambda}$  is optimal for the dual. Importantly, if the functions are continuously differentiable and a regularity condition such as the linear independence constraint qualification holds, then an optimal multiplier is guaranteed to exist. Moreover, if the Lagrangian is strictly convex in the primal variables, for example if the objective is strictly convex or if certain constraints with positive multipliers are strictly convex, then the dual solution can be used to recover the primal solution uniquely. This dual-to-primal recovery is central in optimization algorithms, where solving the dual efficiently can directly yield the primal solution.

**Theorem 29: The uniqueness of a Solution**

Suppose that  $f$  and  $-c_i$ ,  $i = 1, 2, \dots, m$  are convex and continuously differentiable on  $\mathbb{R}^n$ . Suppose that  $\bar{x}$  is a solution of  $\min_x f(x)$  s.t.  $c(x) \geq 0$  at which LICQ holds.

Suppose that  $\hat{\lambda}$  solves the dual problem  $\max_{\lambda \geq 0} q(\lambda)$ ,  $\inf_x \mathcal{L}(x, \hat{\lambda})$  and that the infimum in  $\inf_x \mathcal{L}(x, \hat{\lambda})$  is attained at  $\hat{x}$ . Assume further that  $\mathcal{L}(\cdot, \hat{\lambda})$  is a strictly convex function. Then  $\bar{x} = \hat{x}$  (that is,  $\hat{x}$  is the unique solution), and  $f(\bar{x}) = \mathcal{L}(\hat{x}, \hat{\lambda})$ .

**Proof:** Assume  $\bar{x} \neq \hat{x}$ . By Theorem 21 (because of the LICQ assumption), there exists  $\bar{\lambda}$  satisfying KKT conditions.

- Theorem 28 and KKT conditions implies  $\mathcal{L}(\bar{x}, \bar{\lambda}) = q(\bar{\lambda}) = q(\hat{\lambda}) = \mathcal{L}(\hat{x}, \hat{\lambda})$ .
- Since  $\hat{x} = \arg \min_x \mathcal{L}(x, \hat{\lambda})$ , we have that  $\nabla_x \mathcal{L}(\hat{x}, \hat{\lambda}) = 0$ . By strict convexity:

$$\mathcal{L}(\bar{x}, \hat{\lambda}) > \mathcal{L}(\hat{x}, \hat{\lambda}) + \nabla_x \mathcal{L}(\hat{x}, \hat{\lambda})^T (\bar{x} - \hat{x}) = \mathcal{L}(\hat{x}, \hat{\lambda}) = \mathcal{L}(\bar{x}, \bar{\lambda}).$$

- Thus,  $-\hat{\lambda}^T c(\bar{x}) > -\bar{\lambda}^T c(\bar{x}) = 0$ , contradicting  $\hat{\lambda} \geq 0$ ,  $c(\bar{x}) \geq 0$ . Hence,  $\bar{x} = \hat{x}$ . □

Example: S-O  
Suf. Cond.

Constraint  
Qualifications

Sensitivity  
Analysis

Duality  
Theory

Wolfe Dual

LP

KKT  
Conditions

Dual Problem

**Comments**

A natural question arises: when is the primal solution uniquely determined by the dual? The answer is given by a converse result under conditions of strict convexity. Suppose the primal problem has a solution  $\bar{x}$  that satisfies the linear independence constraint qualification, and suppose the dual problem has a solution  $\hat{\lambda}$  for which the Lagrangian is strictly convex in the primal variables. Then the point that minimizes the Lagrangian for  $\hat{\lambda}$ , call it  $\hat{x}$ , must coincide with the primal solution  $\bar{x}$ . To prove it, assume the opposite, that  $\bar{x}$  and  $\hat{x}$  are different. The KKT theorem ensures that there exists a multiplier  $\bar{\lambda}$  such that the pair  $\bar{x}$  and  $\bar{\lambda}$  satisfy the conditions. By dual optimality, we then have that the Lagrangian evaluated at  $\bar{x}$ ,  $\bar{\lambda}$  equals the dual value  $q(\bar{\lambda})$ , which in turn equals  $q(\hat{\lambda})$ , which equals the Lagrangian at  $\hat{x}$ ,  $\hat{\lambda}$ . But since  $\hat{x}$  is the unique minimizer of the strictly convex Lagrangian for  $\hat{\lambda}$ , any other point such as  $\bar{x}$  must give a strictly larger value. This creates a contradiction, because the equalities we obtained earlier show that both values should be the same. The contradiction arises from the assumption that  $\bar{x}$  and  $\hat{x}$  differ. Therefore, they must in fact be equal, and uniqueness is established. This result is powerful: it tells us that under convexity and strict convexity assumptions, solving the dual not only provides the optimal value but actually identifies the unique primal solution. Thus, duality does not just bound the problem but can, in favorable cases, completely resolve it.

**Theorem 30: Wolfe Dual**

If  $f, -c_i$  ( $i = 1, \dots, m$ ) are convex, continuously differentiable on  $\mathbb{R}^n$ , and  $(\bar{x}, \bar{\lambda})$  solves  $\min_x f(x)$  s.t.  $c(x) \geq 0$  with LICQ, then  $(\bar{x}, \bar{\lambda})$  solves

$$\max_{x, \lambda} \mathcal{L}(x, \lambda) \text{ s.t. } \nabla_x \mathcal{L}(x, \lambda) = 0, \lambda \geq 0.$$

**Proof:** KKT conditions give  $\nabla_x \mathcal{L}(\bar{x}, \bar{\lambda}) = 0, \lambda \geq 0, \mathcal{L}(\bar{x}, \bar{\lambda}) = f(\bar{x})$ .

- For any  $(x, \lambda)$  with  $\nabla_x \mathcal{L}(x, \lambda) = 0, \lambda \geq 0$ :

$$\mathcal{L}(\bar{x}, \bar{\lambda}) = f(\bar{x}) \geq f(\bar{x}) - \lambda^T c(\bar{x}) = \mathcal{L}(x, \lambda).$$

- Convexity of  $\mathcal{L}(\cdot, \lambda)$  implies:

$$\mathcal{L}(\bar{x}, \bar{\lambda}) \geq \mathcal{L}(x, \lambda) + \nabla_x \mathcal{L}(x, \lambda)^T (\bar{x} - x) = \mathcal{L}(x, \lambda).$$

- Thus,  $\mathcal{L}(\bar{x}, \bar{\lambda}) \geq \mathcal{L}(x, \lambda)$ , so  $(\bar{x}, \bar{\lambda})$  solves the Wolfe dual.  $\square$

Example: For  $\min_{x_1, x_2} 0.5(x_1^2 + x_2^2)$  s.t.  $x_1 - 1 \geq 0$ , at  $(x_1, x_2, \lambda_1) = (1, 0, 1)$ ,  $\nabla_x \mathcal{L} = (x_1 - 1, x_2) = 0$ ,  $\mathcal{L} = 0.5$ . For  $\nabla_x \mathcal{L} = (x_1 - \lambda_1, x_2) = 0$ ,  $\lambda_1 \geq 0$ ,  $\mathcal{L} = -\lambda_1^2/2 + \lambda_1 \leq 0.5$ .

20/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem

**Comments**

The Wolfe dual formulation provides a powerful way to connect the primal problem with its dual counterpart by reformulating the optimization task in terms of the Lagrangian function. When the objective function and the inequality constraints are convex and differentiable, and the Linear Independence Constraint Qualification is satisfied, the KKT conditions not only characterize optimality in the primal problem but also guarantee equivalence with a specific maximization problem involving the Lagrangian. This is known as the Wolfe dual.

The essence of the Wolfe dual lies in requiring two conditions simultaneously: stationarity of the Lagrangian with respect to the primal variables, and nonnegativity of the multipliers. Stationarity enforces that the primal solution is “balanced” with respect to the constraints, while the multipliers represent the strength of each inequality constraint at the solution. Together, they ensure that the pair of primal and dual solutions mutually reinforce one another.

Convexity plays a central role here. Because the Lagrangian is convex in the primal variable, we can compare the value at the candidate solution with any other feasible point. The inequalities show that no other feasible primal-dual pair achieves a higher value of the Lagrangian, thereby confirming that the solution is indeed optimal for the dual problem. This symmetry between minimization of the primal objective and maximization of the Lagrangian demonstrates the unifying role of duality theory.

A simple quadratic example illustrates the idea: minimizing a convex quadratic function subject to a linear inequality constraint leads to a dual problem in which the multiplier’s optimization reproduces the same optimal value. This provides intuition for why dual problems can sometimes be easier to analyze than their primal counterparts.

Linear programming duals simplify computations in certain cases.

## Example: Linear Programming

Primal problem:

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} - \mathbf{b} \geq 0.$$

Dual objective:  $q(\lambda) = \inf_{\mathbf{x}} [\mathbf{c}^T \mathbf{x} - \lambda^T (\mathbf{A}\mathbf{x} - \mathbf{b})] = \inf_{\mathbf{x}} [(\mathbf{c} - \mathbf{A}^T \lambda)^T \mathbf{x} + \mathbf{b}^T \lambda]$ .

- If  $\mathbf{c} - \mathbf{A}^T \lambda \neq 0$ ,  $q(\lambda) = -\infty$ .
- If  $\mathbf{c} - \mathbf{A}^T \lambda = 0$ ,  $q(\lambda) = \mathbf{b}^T \lambda$ .
- Dual problem:  $\max_{\lambda} \mathbf{b}^T \lambda$  s.t.  $\mathbf{A}^T \lambda = \mathbf{c}$ ,  $\lambda \geq 0$ .

Wolfe dual:  $\max_{\mathbf{x}, \lambda} \mathbf{c}^T \mathbf{x} - \lambda^T (\mathbf{A}\mathbf{x} - \mathbf{b})$  s.t.  $\mathbf{A}^T \lambda = \mathbf{c}$ ,  $\lambda \geq 0$ , reduces to the dual problem.

For some matrices  $\mathbf{A}$ , the dual problem is computationally easier to solve than the primal.

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



## Comments

Linear programming provides one of the most direct illustrations of duality. In its standard form, the primal problem is to minimize a linear cost subject to linear inequalities. The dual problem emerges naturally from the Lagrangian relaxation, where multipliers are introduced to enforce the inequalities. By carefully analyzing the infimum of the Lagrangian over the primal variables, we can distinguish two possibilities: if the linearity conditions are not satisfied, the Lagrangian can be driven to negative infinity; but if the compatibility condition holds, the infimum collapses into a finite linear function in the multipliers.

This reasoning gives the dual problem: maximize a linear functional of the multipliers subject to an equality constraint that reflects the compatibility condition, and nonnegativity constraints on the multipliers. Importantly, the structure of the dual problem mirrors that of the primal: the objective is linear, and the constraints are also linear. In fact, the dual of a linear program is another linear program.

One striking feature of linear programming duality is computational efficiency. For certain structures of the constraint matrix, solving the dual is substantially easier than solving the primal. For example, when the number of constraints is small compared to the number of variables, the dual may have fewer variables and hence be more efficient to optimize. Conversely, in other situations, the primal may be easier. Thus, duality not only provides theoretical insights into optimality and sensitivity analysis but also has direct implications for numerical computation.

Note, that in this case the Wolfe dual formulation applied to linear programs recovers exactly the standard dual problem.

Quadratic programming duals leverage convexity for explicit solutions.

### Example: Convex Quadratic Programming

Primal problem (G symmetric, positive definite):

$$\min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{c}^T \mathbf{x} \quad \text{s.t.} \quad \mathbf{A} \mathbf{x} - \mathbf{b} \geq 0.$$

Dual objective:  $q(\lambda) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \inf_{\mathbf{x}} \left[ \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{c}^T \mathbf{x} - \lambda^T (\mathbf{A} \mathbf{x} - \mathbf{b}) \right].$

- Since G is positive definite,  $\mathcal{L}(\cdot, \lambda)$  is strictly convex.
- Infimum at  $\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{G} \mathbf{x} + \mathbf{c} - \mathbf{A}^T \lambda = 0$ .
- Substituting  $\mathbf{x} = \mathbf{G}^{-1}(\mathbf{A}^T \lambda - \mathbf{c})$  gives:

$$q(\lambda) = -\frac{1}{2} (\mathbf{A}^T \lambda - \mathbf{c})^T \mathbf{G}^{-1} (\mathbf{A}^T \lambda - \mathbf{c}) + \mathbf{b}^T \lambda \quad \text{s.t.} \quad \lambda \geq 0$$

22/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Example: S-O  
Suf. Cond.**

**Constraint  
Qualifications**

**Sensitivity  
Analysis**

**Duality  
Theory**

**Wolfe Dual**

**LP**

**KKT  
Conditions**

**Dual Problem**



### Comments

Quadratic programming extends linear programming by allowing a quadratic objective while keeping linear inequality constraints. When the quadratic term is defined by a symmetric positive definite matrix, the primal problem remains convex, which ensures a unique minimizer. Constructing the dual proceeds by evaluating the infimum of the Lagrangian with respect to the primal variables. Because the quadratic form is strictly convex, this infimum exists at a unique point determined by the stationarity condition.

Specifically, setting the gradient of the Lagrangian equal to zero gives an explicit expression for the primal variable in terms of the multipliers. Substituting this expression back into the Lagrangian produces a closed-form dual objective. The resulting dual problem has a concave quadratic objective in the multipliers, along with nonnegativity constraints. Thus, the dual transforms a constrained quadratic minimization into an unconstrained quadratic maximization, up to the inequality conditions on the multipliers.

This reformulation highlights an important computational advantage: sometimes optimizing over the dual variables is easier than directly solving the primal. The dual function automatically incorporates the quadratic curvature of the primal, reducing the complexity of the optimization landscape. Moreover, because the dual objective explicitly involves the inverse of the quadratic matrix, it emphasizes how primal curvature shapes dual feasibility.

Example: S-O

Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem



The Wolfe dual reformulates the quadratic program with explicit constraints.

### Example: Convex Quadratic Programming (continued)

Wolfe dual:

$$\max_{x, \lambda} \frac{1}{2} x^T G x + c^T x - \lambda^T (Ax - b) \quad \text{s.t.} \quad Gx + c - A^T \lambda = 0, \quad \lambda \geq 0.$$

To make it clearer that the objective is concave we can rewrite the dual formulation (using  $x^T G x = -(c - A^T \lambda)^T x$ ) as follows:

$$\max_{x, \lambda} -\frac{1}{2} x^T G x + \lambda^T b \quad \text{s.t.} \quad Gx + c - A^T \lambda = 0, \quad \lambda \geq 0.$$

The Wolfe dual requires only positive semidefiniteness of  $G$ .

### Comments

The Wolfe dual for convex quadratic programming presents an alternative but equivalent way to express the dual problem. Instead of eliminating the primal variable explicitly, we retain both the primal variable and the multipliers, subject to the stationarity condition. This gives a maximization problem with quadratic terms but linear equality constraints linking the variables.

One of the advantages of the Wolfe dual formulation is that it exposes the concavity of the objective function more directly. By rewriting the quadratic expression, we make it clear that the Wolfe dual is a concave maximization problem, which guarantees the existence of an optimal solution provided the feasible region is nonempty. Furthermore, the Wolfe dual requires only that the quadratic matrix be positive semidefinite, not strictly positive definite. This broadens the range of applicability, since many practical problems involve semidefinite matrices.

From a computational perspective, the Wolfe dual can sometimes be solved more efficiently than the standard dual, depending on the problem structure. The explicit presence of both primal and dual variables can be exploited by specialized algorithms, such as interior-point or decomposition methods. This flexibility makes the Wolfe dual a valuable tool in both theory and practice.

Conceptually, the Wolfe dual reinforces the central theme of duality theory: optimization problems can often be reformulated in multiple equivalent ways, each highlighting different aspects of structure and computational tractability.

## Introduction to Linear Programs

Linear programs have a linear objective function and linear constraints (equalities/inequalities). The feasible set is a polytope (convex, connected, with flat polygonal faces).

Note: Solutions may be non-unique (e.g., same  $c^T x$  over an edge or face). No solution exists if the feasible set is empty (the infeasible case) or the objective is unbounded below (the unbounded case).

### Standard Form of Linear Programs:

$$\min c^T x, \text{ subject to } Ax = b, x \geq 0 \quad (10)$$

where  $c, x \in \mathbb{R}^n$ ,  $b \in \mathbb{R}^m$ , and  $A$  is an  $m \times n$  matrix.

### Conversion to Standard Form:

For a problem with inequality constraints:

$$\min c^T x, \text{ subject to } Ax \leq b$$

introduce slack variables  $z$  to obtain:

$$\min c^T x, \text{ subject to } Ax + z = b, z \geq 0$$

Example: S-O  
Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem



### Comments

Up to this point, we have studied general approaches to optimization with constraints, often involving nonlinear functions and more complex feasible sets. Now we turn to a very special but extremely important class of problems: linear programming. These are optimization problems where both the objective function and the constraints are linear. Despite their apparent simplicity, linear programs form the backbone of many applications in economics, engineering, operations research, and computer science.

A linear program seeks to minimize a linear objective, written as  $c^T x$ , subject to linear equality and inequality constraints. The set of feasible solutions is always a convex polytope. A polytope can be thought of as a geometric object with flat polygonal faces, and this convexity ensures that if two points are feasible, then every point on the line between them is also feasible. This property is crucial because it implies that local optima are always global optima.

However, several issues may arise. The solution may not be unique: for example, an entire edge or face of the polytope might yield the same objective value. In some cases, the problem has no solution at all. This happens either because the feasible set is empty, meaning the constraints are inconsistent, or because the objective is unbounded. An unbounded case occurs when the feasible region extends infinitely in a direction that keeps decreasing the objective value.

To study these problems systematically, it is convenient to use a standard form. In this form, we minimize  $c^T x$ , subject to  $Ax = b$  and  $x \geq 0$ . Any linear program with inequalities can be converted into this format by introducing auxiliary variables. For instance, inequality constraints can be transformed into equalities by adding slack variables, which are required to remain nonnegative. This process allows us to work within a unified framework for analysis and algorithms.

**Variable Splitting:**

Split  $x = x^+ - x^-$ , where  $x^+ = \max(x, 0) \geq 0$ ,  $x^- = \max(-x, 0) \geq 0$ . Then:

$$\min \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^T \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix}, \text{ s.t. } [A \quad -A \quad I] \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} = b, \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} \geq 0$$

**Handling Inequalities:**

- $x \leq u \Leftrightarrow x + w = u$ ,  $w \geq 0$  (slack variables).
- $Ax \geq b \Leftrightarrow Ax - y = b$ ,  $y \geq 0$  (surplus variables).

Maximization  $\max c^T x$  becomes  $\min(-c)^T x$ .

Note: The linear program (10) is *infeasible* if the feasible set is empty; *unbounded* if there exists a sequence  $x^k$  such that  $c^T x^k \rightarrow -\infty$ .

**Comments**

Once the standard form has been introduced, the natural question is how to convert an arbitrary linear program into this representation. There are several important steps in this process.

First, we must ensure that all variables are nonnegative. If a variable is unrestricted in sign, we split it into two components: a nonnegative part and a nonnegative counterpart. Specifically, a variable  $x$  is written as  $x^+ - x^-$ , where  $x^+ = \max(x, 0)$ , and  $x^- = \max(-x, 0)$ . Both of these are greater than or equal to zero. In effect, this substitution replaces a single unrestricted variable with two nonnegative variables.

Second, inequality constraints must be rewritten as equalities. For example, an upper bound constraint such as  $x \leq u$  can be converted by adding a nonnegative slack variable  $w$ , so that  $x + w = u$ . Similarly, a lower bound inequality such as  $Ax \geq b$  can be expressed as  $Ax - y = b$ , where  $y$  is a nonnegative surplus variable. These manipulations bring every inequality into an equality format, which matches the requirements of the standard form.

It is also conventional to write maximization problems as minimization problems. For instance, maximizing  $c^T x$  is equivalent to minimizing  $-c^T x$ . This simple transformation ensures that all linear programs can be expressed consistently.

Finally, we must consider the possibility that a program is infeasible or unbounded. If no feasible solution satisfies the constraints, then the problem is infeasible. On the other hand, if there exists a sequence of feasible solutions such that the objective decreases without limit, then the program is unbounded. These two situations represent fundamental difficulties that algorithms must be able to detect.

Example: S-O  
Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem



## Lagrangian:

For  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ , partition multipliers into  $\lambda \in \mathbb{R}^m$  (for the equality constraints  $Ax = b$ ) and  $s \in \mathbb{R}^n$  (for the bound constraints  $x \geq 0$ ).

Lagrangian:

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x$$

## KKT Conditions:

$x^*$  is a solution if  $\exists \lambda, s$  s.t.:

$$A^T \lambda + s = c, \quad (11a)$$

$$Ax = b, \quad (11b)$$

$$x \geq 0, \quad (11c)$$

$$s \geq 0, \quad (11d)$$

$$x_i s_i = 0, \quad i = 1, 2, \dots, n. \quad (11e)$$

Complementarity :  $x_i s_i = 0$  (or  $x^T s = 0$ ) means  $x_i = 0$  or  $s_i = 0$ .

## Comments

To analyze optimality in linear programming, we rely on the Karush–Kuhn–Tucker conditions. For a linear program in standard form, the Lagrangian plays a central role. Recall that the objective is to minimize  $c^T x$ , subject to  $Ax = b$  and  $x \geq 0$ . We introduce multipliers for the constraints. For the equality constraints, we associate a vector of multipliers, usually denoted by  $\lambda$ . For the nonnegativity constraints, we introduce a vector of multipliers  $s$ , sometimes called dual variables.

The Lagrangian is then defined as  $c^T x - \lambda^T (Ax - b) - s^T x$ . The KKT conditions emerge by requiring stationarity, primal feasibility, dual feasibility, and complementarity. More explicitly: first, the gradient condition implies that  $A^T \lambda + s = c$ . Second, primal feasibility enforces  $Ax = b$  and  $x \geq 0$ . Third, dual feasibility requires  $s \geq 0$ . Finally, complementarity states that  $x_i s_i = 0$  for each index  $i$ . In words, for each component, either the variable is zero or the corresponding dual variable is zero, but not both positive simultaneously.

In linear programming, these conditions are not only necessary but also sufficient for optimality. That is, if we can find  $x$ ,  $\lambda$ , and  $s$  satisfying all these equations and inequalities, then  $x$  is guaranteed to be a global solution. This strong property is one of the reasons why linear programming is so tractable compared to nonlinear optimization.

**Example: S-O  
Suf. Cond.**

**Constraint Qualifications**

**Sensitivity Analysis**

**Duality Theory**

**Wolfe Dual**

**LP**

**KKT Conditions**

**Dual Problem**



Let  $\langle x^*, \lambda^*, s^* \rangle$  satisfy KKT conditions:

$$A^T \lambda^* + s^* = c, \quad Ax^* = b, \quad x^* \geq 0, \quad s^* \geq 0, \quad x^{*T} s^* = 0.$$

Then:

$$c^T x^* = (A^T \lambda^* + s^*)^T x^* = (Ax^*)^T \lambda^* = b^T \lambda^*,$$

indicating that the primal and dual objectives are equal (since  $b^T \lambda$  is the objective function for the dual problem).

- The KKT conditions (11) are sufficient for  $x^*$  to be a global solution of (10). Let  $\bar{x}$  be any other feasible point (with  $A\bar{x} = b$ ,  $\bar{x} \geq 0$ ):

$$c^T \bar{x} = (A^T \lambda^* + s^*)^T \bar{x} = b^T \lambda^* + \bar{x}^T s^* \geq b^T \lambda^* = c^T x^*, \text{ since } \bar{x} \geq 0, s^* \geq 0.$$

Thus, no feasible point has a lower objective value than  $c^T x^*$ .

- $\bar{x}$  is optimal if and only if:

$$\bar{x}^T s^* = 0,$$

so when  $s_i^* > 0$ , then  $\bar{x}_i = 0$  for all solutions  $\bar{x}$ .

## Comments

An essential feature of linear programming is the deep connection between the primal and dual problems. If we have a solution triple  $x^*, \lambda^*, s^*$  that satisfies the KKT conditions, then strong duality holds: the objective values of the primal and the dual problems are equal. In particular, we can write  $c^T x^* = b^T \lambda^*$ . This relationship reveals that solving the primal problem automatically provides a solution to the dual, and vice versa.

The equality of objectives allows us to establish optimality. Suppose  $x^*$  is feasible and paired with multipliers  $\lambda^*$  and  $s^*$  satisfying the KKT system. Then for any other feasible point  $\bar{x}$ , the objective value  $c^T \bar{x}$  can be expressed as  $b^T \lambda^* + \bar{x}^T s^*$ . Since both  $\bar{x}$  and  $s^*$  are nonnegative, this expression is always greater than or equal to  $b^T \lambda^*$ . But that equals  $c^T x^*$ , the objective at the candidate solution. Therefore, no feasible point can achieve a smaller value. This proves that  $x^*$  is indeed optimal.

Moreover, the complementarity condition offers a finer characterization of the solution. If  $s_i^*$  is strictly positive, then the corresponding primal variable  $x_i$  must be zero in every optimal solution. Conversely, if  $x_i$  is strictly positive, then the associated  $s_i^*$  must vanish. This interplay provides insight into which constraints are active at optimality and which variables are essential in the solution.

Thus, the primal-dual structure of linear programming not only ensures optimality but also provides powerful tools for interpretation and sensitivity analysis.

Let's consider the problem (10) with the given data  $c$ ,  $b$ , and  $A$ . The *dual problem* to the primal problem (10) is:

$$\max b^T \lambda, \text{ subject to } A^T \lambda \leq c. \quad (12)$$

- **Alternative Formulation:** Introduce a vector of dual slack variables  $s$  to rewrite the dual problem as:

$$\max b^T \lambda, \text{ subject to } A^T \lambda + s = c, \quad s \geq 0.$$

The variables  $(\lambda, s)$  are sometimes collectively called dual variables.

- **Relationship:** The primal and dual problems provide two perspectives on the same data.

- **Standard Form:** To align with standard minimization form, restate the dual problem as:

$$\min -b^T \lambda, \text{ subject to } c - A^T \lambda \geq 0.$$

- **Lagrangian:** Using  $x \in \mathbb{R}^n$  as Lagrange multipliers for  $A^T \lambda \leq c$ , the Lagrangian is:

$$\tilde{\mathcal{L}}(\lambda, x) = -b^T \lambda - x^T (c - A^T \lambda).$$

**Example: S-O  
Suf. Cond.**

**Constraint Qualifications**

**Sensitivity Analysis**

**Duality Theory**

**Wolfe Dual**

**LP**

**KKT Conditions**

**Dual Problem**



### Comments

The construction of the dual problem is one of the most fundamental ideas in linear programming. Given a primal problem with cost vector  $c$ , constraint matrix  $A$ , and right-hand side  $b$ , the dual problem is defined as maximizing  $b^T \lambda$ , subject to the inequality  $A^T \lambda \leq c$ . Here,  $\lambda$  is the vector of dual variables, sometimes interpreted as shadow prices, because each component measures the marginal value of relaxing the corresponding primal constraint.

Just as in the case of a primal problem, slack variables are introduced to bring the dual problem to a standard form. Setting  $s = c - A^T \lambda$ , the dual inequalities are replaced by equalities  $A^T \lambda + s = c$ , with  $s \geq 0$ . The pair of variables  $(\lambda, s)$  together represent the complete set of dual unknowns. This form is particularly useful because it mirrors the structure of the primal, where constraints are also written in equality form with nonnegative slack variables.

Another useful reformulation rewrites the dual as a minimization problem: minimize  $-b^T \lambda$ , subject to  $c - A^T \lambda \geq 0$ . This way, both primal and dual are cast into consistent minimization frameworks, facilitating comparisons and theoretical developments.

Finally, the Lagrangian perspective offers another layer of interpretation. By treating the primal variables  $x$  as Lagrange multipliers for the dual constraints  $A^T \lambda \leq c$ , we can define the dual Lagrangian as  $-b^T \lambda - x^T (c - A^T \lambda)$ . This reveals a striking symmetry: primal variables act as multipliers for dual constraints, just as dual variables act as multipliers for primal constraints. The two problems are, in essence, different but equivalent views of the same optimization structure.

## First-Order Necessary Conditions for the dual problem

For  $\lambda$  to be optimal for the dual problem  $\max b^T \lambda$ , s.t.  $A^T \lambda \leq c$ , there exists  $x$  such that:

$$Ax = b,$$

$$A^T \lambda \leq c,$$

$$x \geq 0,$$

$$x_i(c - A^T \lambda)_i = 0, \quad i = 1, 2, \dots, n.$$

**Example: S-O  
Suf. Cond.**

**Constraint Qualifications**

**Sensitivity Analysis**

**Duality Theory**

**Wolfe Dual**

**LP**

**KKT Conditions**

**Dual Problem**



- Defining  $s = c - A^T \lambda$  we obtain that these conditions are identical to (11):

$$A^T \lambda + s = c, \quad Ax = b, \quad x \geq 0, \quad s \geq 0, \quad x^T s = 0.$$

- The optimal  $\lambda$  in the primal problem are the optimal variables in the dual, and the optimal  $x$  in the dual are the optimal variables in the primal.
- Sufficiency: For  $x^*$ ,  $\lambda^*$  satisfying these conditions (with  $s = c - A^T \lambda^*$ ), and any dual feasible  $\tilde{\lambda}$  ( $A^T \tilde{\lambda} \leq c$ ):

$$b^T \tilde{\lambda} = (x^*)^T A^T \tilde{\lambda} = (x^*)^T (A^T \tilde{\lambda} - c) + c^T x^* \leq c^T x^* = b^T \lambda^*,$$

since  $A^T \tilde{\lambda} - c \leq 0$ ,  $x^* \geq 0$ . Thus,  $\lambda^*$  is optimal for the dual problem.

29/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The first-order conditions for the dual problem play a central role in linking primal and dual optimality. Recall that the dual is to maximize  $b^T \lambda$  subject to  $A^T \lambda \leq c$ . For a vector  $\lambda$  to be optimal, there must exist a vector  $x$  satisfying several conditions simultaneously. These conditions mirror the Karush–Kuhn–Tucker system we have already encountered.

The requirements are: first, the primal feasibility condition  $Ax = b$ ; second, the dual feasibility condition  $A^T \lambda \leq c$ ; third, nonnegativity of  $x$ ; and fourth, complementarity, meaning that for each component  $i$ , the product  $x_i(c - A^T \lambda)_i$  equals zero. In other words, either the primal variable is strictly positive and the corresponding constraint in the dual is tight, or the dual inequality is slack and the primal variable is zero.

Defining  $s = c - A^T \lambda$ , the system can be written more symmetrically as  $A^T \lambda + s = c$ ,  $Ax = b$ ,  $x \geq 0$ ,  $s \geq 0$ , and  $x^T s = 0$ . This is exactly the KKT system for linear programming.

Why do these conditions guarantee optimality? Suppose  $x^*$  and  $\lambda^*$  satisfy them. For any dual feasible vector  $\tilde{\lambda}$ , we can compare  $b^T \tilde{\lambda}$  with  $c^T x^*$ . Using feasibility and complementarity, it follows that  $b^T \tilde{\lambda}$  is less than or equal to  $b^T \lambda^*$ , which equals  $c^T x^*$ . Hence  $\lambda^*$  is indeed an optimal solution of the dual.

The primal-dual relationship is symmetric: the dual of the dual problem  $\max b^T \lambda$ , subject to  $A^T \lambda \leq c$ , is the primal problem.

- Weak Duality: For a primal feasible  $x$  ( $Ax = b$ ,  $x \geq 0$ ) and dual feasible  $(\lambda, s)$  ( $A^T \lambda + s = c$ ,  $s \geq 0$ ):

$$c^T x - b^T \lambda = (c - A^T \lambda)^T x = s^T x \geq 0.$$

Thus,  $c^T x \geq b^T \lambda$ , i.e., the dual objective is a lower bound on the primal objective.

### Theorem 31: Strong Duality

- (i) If either the primal problem ( $\min c^T x$ , s. t.  $Ax = b, x \geq 0$ ) or the dual problem ( $\max b^T \lambda$ , s. t.  $A^T \lambda \leq c$ ), has a finite solution, then so does the other, and their objective values are equal.
- (ii) If either problem is unbounded, then the other is infeasible.

30/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Example: S-O**

**Suf. Cond.**

**Constraint Qualifications**

**Sensitivity Analysis**

**Duality Theory**

**Wolfe Dual**

**LP**

**KKT Conditions**

**Dual Problem**



### Comments

The relationship between primal and dual problems is not merely one-sided but deeply symmetric. In fact, the dual of the dual problem is the primal problem itself. This fundamental symmetry underlies the power of linear programming duality.

The first observation is the principle of weak duality. For any feasible primal solution  $x$  satisfying  $Ax = b$  and  $x \geq 0$ , and any feasible dual pair  $(\lambda, s)$  satisfying  $A^T \lambda + s = c$  with  $s \geq 0$ , we have the inequality  $c^T x - b^T \lambda = s^T x \geq 0$ . This shows that the primal objective value is always greater than or equal to the dual objective value. In other words, the dual provides a guaranteed lower bound on the primal.

The deeper result is strong duality. It asserts two things. First, if either the primal minimization problem or the dual maximization problem has a finite optimal solution, then so does the other, and their optimal objective values coincide exactly. Second, if one problem is unbounded, the other must be infeasible. Thus, in the context of linear programming, the primal and dual problems are inseparably linked through the strong duality theorem: in fact, they represent equivalent formulations of the same optimization problem, with their optimal solutions interconnected via the KKT conditions. That's why KKT conditions are necessary and sufficient for optimality of linear program's problems.

This interrelation provides the theoretical basis for efficient algorithms, such as the simplex method and interior-point methods, because checking dual feasibility and complementary slackness is often computationally advantageous. It also gives interpretive power: dual solutions identify shadow prices that quantify the value of constraints, offering direct economic and engineering insights. It also demonstrates the power of formalization, which we discussed in the first lesson.

## Proof of Strong Duality

**Proof of (i):** Assume the primal  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , has a finite optimal solution  $x^*$ .

- By Theorem 21 (with a caveat, that for LP  $T_\Omega(x) = \mathcal{F}(x)$ ) there exist  $\lambda^*$ ,  $s^*$  such that  $\langle x^*, \lambda^*, s^* \rangle$  satisfies:

$$A^T \lambda^* + s^* = c, \quad Ax^* = b, \quad x^* \geq 0, \quad s^* \geq 0, \quad x^{*T} s^* = 0.$$

- These conditions are equivalent to:

$Ax^* = b$ ,  $A^T \lambda^* \leq c$ ,  $x^* \geq 0$ ,  $x_i^*(c - A^T \lambda^*)_i = 0$ ,  $i = 1, \dots, n$ , which are sufficient for  $\lambda^*$  to solve the dual  $\max b^T \lambda$ , subject to  $A^T \lambda \leq c$ .

- Then,  $c^T x^* = b^T \lambda^*$ . A symmetric argument holds if the dual has a solution.

**Proof of (ii):** Assume the primal is unbounded, i.e., there exists a sequence  $x^k$  such that:

$$c^T x^k \rightarrow -\infty, \quad Ax^k = b, \quad x^k \geq 0.$$

- If the dual is feasible, i.e.,  $\exists \bar{\lambda}$  such that  $A^T \bar{\lambda} \leq c$ , then:

$$\bar{\lambda}^T b = \bar{\lambda}^T Ax^k \leq c^T x^k \rightarrow -\infty,$$

leading to a contradiction. Thus, the dual is infeasible.

- Similarly, unboundedness of the dual implies infeasibility of the primal.  $\square$

31/31 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Example: S-O  
Suf. Cond.

Constraint Qualifications

Sensitivity Analysis

Duality Theory

Wolfe Dual

LP

KKT Conditions

Dual Problem



### Comments

The proof of strong duality demonstrates beautifully the unity of primal and dual problems. Suppose the primal problem of minimizing  $c^T x$ , subject to  $Ax = b$  and  $x \geq 0$ , has a finite optimal solution  $x^*$ . Then, by the KKT theorem, there exist multipliers  $\lambda^*$  and  $s^*$  satisfying the conditions  $A^T \lambda^* + s^* = c$ ,  $Ax^* = b$ ,  $x^* \geq 0$ ,  $s^* \geq 0$ , and  $x^{*T} s^* = 0$ .

These conditions can be restated as primal feasibility, dual feasibility, and complementarity, showing that the pair  $(x^*, \lambda^*)$  satisfies both systems simultaneously. This suffices to prove that  $\lambda^*$  is an optimal solution of the dual problem maximizing  $b^T \lambda$ . Moreover, equality of objectives follows:  $c^T x^* = b^T \lambda^*$ . By symmetry, if we start with an optimal dual solution, we can construct a corresponding primal solution with the same value.

The second part of the theorem deals with unboundedness. If the primal is unbounded below, there exists a sequence  $x^k$  with  $c^T x^k$  tending to minus infinity, while satisfying feasibility. If the dual were feasible, then for some feasible multiplier  $\bar{\lambda}$ , we would have  $\bar{\lambda}^T b = \bar{\lambda}^T Ax^k$ , which is less than or equal to  $c^T x^k$ . But as the latter tends to minus infinity, this produces a contradiction. Hence, the dual must be infeasible. By symmetry, the same reasoning applies in reverse.

Thus, the proof of strong duality is not only rigorous but also elegantly demonstrates the inseparable balance between primal and dual optimization problems.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 8)

Shpilev Petr Valerievich  
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



Санкт-Петербургский  
государственный  
университет



29 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In this lecture, we turn to linear programming and develop the simplex method as a central algorithm for solving such problems. We begin with sensitivity analysis, the role of full row rank, and the link between basic feasible points and polytope vertices, leading to the proof of the fundamental theorem of linear programming. The revised simplex method is then introduced, with emphasis on its KKT interpretation, pivoting mechanics, algebraic structure, and termination properties. Through examples and geometric illustrations, we trace the simplex iterates and examine practical implementation aspects, including LU updates, index selection, and the steepest-edge rule. Finally, we address initialization via the two-phase simplex method, detailing the process of finding a feasible starting point and carrying out Phase II optimization.

## Sensitivity Analysis in Linear Programming

The multipliers  $\lambda, s$  for the primal problem (??), indicate the sensitivity of the optimal objective to perturbations in the constraints.

- For small perturbations  $\Delta b, \Delta x, \Delta \lambda, \Delta s$  (with  $\Delta x, \Delta s$  having zeros where  $x, s$  do, and  $x^T s = 0$ ):  
$$0 = x^T s = x^T \Delta s = (\Delta x)^T s = (\Delta x)^T \Delta s.$$
- By strong duality, for original and perturbed problems:  
$$c^T x = b^T \lambda, \quad c^T(x + \Delta x) = (b + \Delta b)^T(\lambda + \Delta \lambda).$$
- Perturbed feasibility:  $A(x + \Delta x) = b + \Delta b, A^T \Delta \lambda = -\Delta s.$
- Change in optimal objective:

$$\begin{aligned} c^T \Delta x &= (b + \Delta b)^T(\lambda + \Delta \lambda) - b^T \lambda \\ &= (b + \Delta b)^T \Delta \lambda + (\Delta b)^T \lambda \\ &= (x + \Delta x)^T A^T \Delta \lambda + (\Delta b)^T \lambda \\ &= (x + \Delta x)^T \Delta s + (\Delta b)^T \lambda = (\Delta b)^T \lambda. \end{aligned}$$

- For  $\Delta b = \epsilon e_j$  (unit vector in  $\mathbb{R}^n$ ), we have for all  $\epsilon$  sufficiently small that:

$$c^T \Delta x = \epsilon \lambda_j,$$

i.e., the change in objective is  $\lambda_j$  times the size of the perturbation to  $b_j$ .

1/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

Sensitivity analysis in linear programming provides a framework for understanding how small perturbations in the input data affect the optimal solution. The central objects here are the Lagrange multipliers, usually denoted by  $\lambda$ , and the slack variables, denoted by  $s$ . These multipliers measure how sensitive the optimal objective value is to changes in the right-hand side of the constraints. To make this precise, one considers infinitesimal perturbations  $\Delta b$  in the constraints and studies the induced variations in the primal and dual variables,  $\Delta x, \Delta \lambda$ , and  $\Delta s$ . An important property of complementary slackness ensures that perturbations preserve orthogonality: the product of  $x$  and  $s$  remains zero, as do the mixed terms.

Strong duality tells us that the primal and dual objective values are always equal at optimality. When the system is perturbed, the balance between  $c^T x$  and  $b^T \lambda$  remains consistent. Careful expansion of this relation reveals that the change in the objective function depends solely on the perturbation to  $b$  multiplied by the multiplier  $\lambda$ . This shows that the dual variable  $\lambda$  acts as a shadow price: it measures the marginal impact on the objective of increasing a constraint by one unit. For example, if we perturb a single entry  $b_j$  by  $\epsilon$ , then the change in the objective is exactly  $\epsilon \lambda_j$ , provided  $\epsilon$  is small enough. This result justifies interpreting the multipliers as rates of change of the optimal value with respect to the constraints. In practice, this interpretation is fundamental in economics and operations research, where one is often interested not only in an optimal plan but also in its stability under fluctuations of resources or demand.

Assume the  $m \times n$  matrix  $A$  in the primal problem  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , has full row rank.

- In practice, preprocessing removes redundancies and reformulates constraints (using slack, surplus, or artificial variables) to ensure this property.

## Definition: Basic Feasible Point

A vector  $x$  is a *basic feasible point* for the primal problem if it is feasible ( $Ax = b$ ,  $x \geq 0$ ) and there exists a subset  $\mathcal{B} \subseteq \{1, 2, \dots, n\}$  such that:

- $\mathcal{B}$  contains exactly  $m$  indices.
- $i \notin \mathcal{B} \Rightarrow x_i = 0$  (that is, the bound  $x_i \geq 0$  can be inactive only if  $i \in \mathcal{B}$ ).
- The  $m \times m$  matrix  $B = [A_i]_{i \in \mathcal{B}}$ , where  $A_i$  is the  $i$ th column of  $A$ , is nonsingular.
- Such a set  $\mathcal{B}$  is called a *basis* for the problem (??), and the corresponding matrix  $B$  is the *basis matrix*.

2/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



## Comments

A central structural property of linear programs is the rank of the constraint matrix  $A$ . By assuming that  $A$  has full row rank, we eliminate redundancies among the equations and ensure that the system is well-posed. In practice, preprocessing techniques identify and remove such redundancies or reformulate constraints by introducing slack, surplus, or artificial variables so that the model achieves this rank condition. With this assumption, the concept of a basic feasible point becomes meaningful.

A feasible solution is any nonnegative vector  $x$  that satisfies  $Ax = b$ . Among these solutions, one calls  $x$  a basic feasible point if the support of  $x$ , that is, the set of indices of its positive components, has no more than  $m$  elements, where  $m$  is the number of constraints. Moreover, the corresponding  $m$  columns of  $A$ , collected into the basis matrix  $B$ , must be linearly independent, so that  $B$  is invertible. In this setting, the values of the basic variables are uniquely determined by solving the corresponding system ( $Bx_{\mathcal{B}} = b$ ).

This construction is not arbitrary. Each basic feasible point corresponds to a vertex, or corner point, of the polyhedron defined by the constraints. The nonsingularity of  $B$  guarantees that exactly  $n$  hyperplanes intersect at this point, producing a vertex. Conversely, any vertex of the feasible region can be represented as a basic feasible point relative to some choice of basis. This equivalence, which we will formally prove a little bit later, establishes a bridge between algebraic descriptions of solutions and geometric intuition. It also provides the foundation for algorithms such as the simplex method, which explore feasible solutions by moving from one vertex to another. Understanding basic feasible points is therefore essential both for theory and computation in linear optimization.

The simplex method examines only basic feasible points and converges to a solution of the primal problem  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , if:

- (a) The problem has basic feasible points.
- (b) At least one is a basic optimal point (a solution that is also a basic feasible point).

### Theorem 32: Fundamental Theorem of Linear Programming

- (i) If the primal problem has a nonempty feasible region, then there is at least one basic feasible point.
- (ii) If the primal problem has solutions, then at least one solution is a basic optimal point.
- (iii) If the nonempty feasible region is bounded, then the primal problem has an optimal solution.

**Proof:** Among all feasible vectors  $x$  ( $Ax = b$ ,  $x \geq 0$ ), choose one with the minimal number of nonzero components, say  $p$ , with nonzeros  $x_1, x_2, \dots, x_p$ , satisfying:

$$\sum_{i=1}^p A_i x_i = b.$$

3/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

The simplex method is one of the most celebrated algorithms in optimization. Its core idea is that instead of searching through all feasible solutions, which may be infinitely many, one only needs to examine the basic feasible points. Because each such point corresponds to a vertex of the feasible region, the algorithm effectively navigates along the edges of the polyhedron from one vertex to another. It starts at some initial vertex and moves step by step to adjacent vertices, improving the objective function value at each move, until it reaches a point where no further improvement is possible. That point is then optimal.

The legitimacy of this approach rests on the Fundamental Theorem of Linear Programming. This theorem has three parts. First, whenever the feasible region is nonempty, there exists at least one basic feasible point. This assures that the simplex method always has a starting point. Second, if there exists an optimal solution, then one can be found among the basic feasible points. In other words, the search for an optimal solution can be restricted to vertices without loss of generality. Third, if the feasible region is both nonempty and bounded, then there necessarily exists an optimal solution.

These statements ensure that the simplex method is not only heuristic but theoretically justified. It is guaranteed to terminate with an optimal solution whenever one exists and the feasible set is bounded. The proof of the theorem relies on selecting feasible points with the minimal number of nonzero variables denoted by  $p$ , and then analyzing the linear independence of the corresponding columns of  $A$ .

## Proof of Fundamental Theorem (i)

Suppose columns  $A_1, \dots, A_p$  are linearly dependent, so:

$$A_p = \sum_{i=1}^{p-1} A_i z_i,$$

for scalars  $z_1, \dots, z_{p-1}$ . Define:

$$x(\epsilon) = x + \epsilon [z_1, \dots, z_{p-1}, -1, 0, \dots, 0]^T.$$

- $Ax(\epsilon) = b$  for any  $\epsilon$ , and  $x_i(\epsilon) > 0$  for  $i = 1, \dots, p$  if  $|\epsilon|$  is sufficiently small.
- There exists  $\bar{\epsilon} \in (0, x_p]$  such that  $x_i(\bar{\epsilon}) = 0$  for some  $i = 1, \dots, p$ , so  $x(\bar{\epsilon})$  is feasible with at most  $p - 1$  nonzeros, contradicting the minimality of  $p$ .

Hence,  $A_1, \dots, A_p$  are linearly independent, so  $p \leq m$ .

- If  $p = m$ , then  $x$  is a basic feasible point with  $\mathcal{B} = \{1, \dots, m\}$ .
- If  $p < m$ , since  $A$  has full row rank, choose  $m - p$  columns from  $A_{p+1}, \dots, A_n$  to form  $m$  linearly independent vectors, and construct  $\mathcal{B}$  by adding their indices to  $\{1, \dots, p\}$ . The proof of (i) is complete.

4/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

If the associated columns of  $A$  are linearly dependent, then one can form a nontrivial linear combination in which one column is expressed in terms of the others. Using this relation, it is possible to perturb the solution slightly, redistributing weight among the variables while preserving feasibility. The construction  $x(\epsilon)$  illustrates this idea: the vector is adjusted by a small multiple of the dependence relation.

This perturbed solution remains feasible for sufficiently small  $\epsilon$ , and its nonzero components remain positive. By choosing an appropriate value of  $\epsilon$ , one can drive at least one variable to zero while keeping the others nonnegative. The resulting solution then has fewer than  $p$  nonzeros, contradicting the minimality assumption. The contradiction shows that the columns must in fact be linearly independent. Hence, the number of nonzero variables cannot exceed the number of constraints, so  $p$  is at most  $m$ .

If  $p$  equals  $m$ , then the solution is already a basic feasible point, since exactly  $m$  linearly independent columns correspond to the nonzero entries. If  $p$  is less than  $m$ , the full row rank assumption ensures that one can add extra columns from  $A$  to form a complete basis of  $m$  independent vectors. This allows extending the support of the solution to form a basis set, and the solution is then recognized as a basic feasible point. Thus, in every case, a feasible region that is nonempty contains at least one basic feasible point, completing the proof of the first statement.

## Proof of Fundamental Theorem (ii)

Let  $x^*$  be a solution to  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , with minimal nonzero components  $p$ , say  $x_1^*, \dots, x_p^*$ .

- If columns  $A_1, \dots, A_p$  are linearly dependent, define:

$$x^*(\epsilon) = x^* + \epsilon z,$$

where  $z = [z_1, \dots, z_{p-1}, -1, 0, \dots, 0]^T$  as in the proof of (i).

- $x^*(\epsilon)$  is feasible for small  $|\epsilon|$ . Since  $x^*$  is optimal:

$$c^T(x^* + \epsilon z) \geq c^T x^* \Rightarrow \epsilon c^T z \geq 0,$$

for small  $\epsilon$  (positive and negative), so  $c^T z = 0$  and  $c^T x^*(\epsilon) = c^T x^*$ .

- As in (i), there exists  $\bar{\epsilon} > 0$  such that  $x^*(\bar{\epsilon})$  is feasible and optimal with at most  $p - 1$  nonzeros, contradicting the minimality of  $p$ .
- Thus,  $A_1, \dots, A_p$  are linearly independent. Using the same reasoning as in (i),  $x^*$  is a basic feasible point, hence a basic optimal point.

The final statement (iii) is a consequence of finite termination of the simplex method. We comment on the latter property after consideration of this method.

□

5/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

The second part of the Fundamental Theorem concerns optimality. Consider an optimal solution  $x^*$  with the minimal number of nonzero variables, again denoted by  $p$ . If the corresponding columns of  $A$  are linearly dependent, then as before one can construct a perturbation by adding  $\epsilon$  times a suitable vector  $z$ . The perturbed vector remains feasible for small values of  $\epsilon$ , because the linear dependence guarantees that the equality constraints remain satisfied.

Now, since  $x^*$  is optimal, any perturbation cannot strictly improve the objective. This leads to the inequality  $c^T(x^* + \epsilon z) \geq c^T x^*$ . Simplifying, one deduces that  $\epsilon c^T z$  must be nonnegative for both positive and negative values of  $\epsilon$ . The only possibility is that  $c^T z = 0$ . Therefore, the perturbation does not change the objective value.

Using the same construction as before, one can choose  $\epsilon$  so that at least one variable becomes zero, reducing the number of nonzero entries while preserving feasibility and optimality. This contradicts the assumption that  $p$  was minimal. Hence the columns must be linearly independent. Consequently, the optimal solution is a basic feasible point. Because it is also optimal, it qualifies as a basic optimal point.

This argument confirms that if an optimal solution exists, then one can always find an optimal solution among the basic feasible points. Together with the first part, this establishes the validity of the simplex method: one needs to examine only the vertices of the feasible region, and one of them will be optimal. The final part of the theorem, concerning boundedness, follows from the fact that the simplex method terminates after finitely many steps. So, we'll return to the proof of this part after reviewing this method.

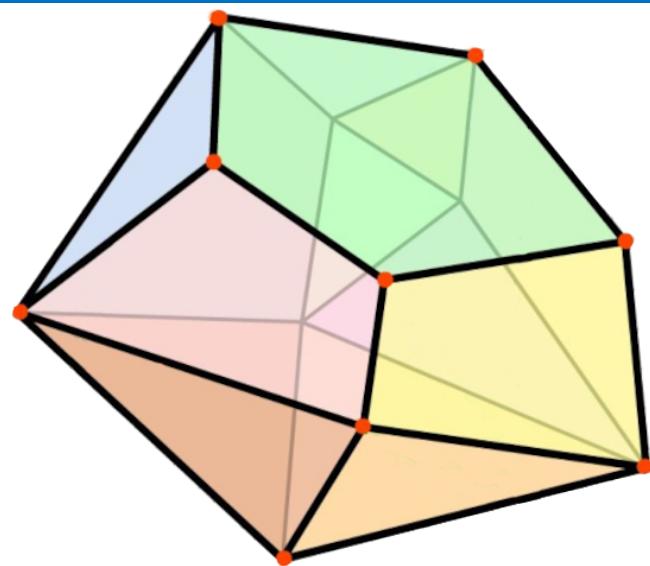


Figure: Vertices of a three-dimensional polytope

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



## Comments

The feasible region determined by a finite system of linear equations and inequalities is always a convex polyhedron. In geometric terms, such an object is called a polytope whenever it is bounded. Each polytope is composed of faces of various dimensions: facets, edges, and vertices. The most important elements for linear programming are the vertices, since they encode the structure of possible solutions. A vertex of the feasible polytope is a point that cannot be expressed as a convex combination of two distinct feasible points; in other words, it does not lie strictly between any other feasible solutions. These special points are geometrically easy to recognize, and they play a central role in the simplex method.

From an algebraic perspective, the vertices of the feasible region coincide precisely with the basic feasible points introduced earlier. Each such point corresponds to a choice of a basis, that is, a set of linearly independent constraint columns, which uniquely determine the nonzero coordinates of the solution. This means that solving a linear program essentially reduces to moving from one vertex of the polytope to another, in search of the one that yields the smallest objective value. The entire power of the simplex algorithm relies on this fact: by focusing only on vertices, it can ignore the infinitely many points lying inside the polytope, while still guaranteeing that an optimal solution will be found among the finite set of basic feasible points. Thus, the geometry and algebra align perfectly: the corner points of the polytope are the basic feasible solutions, and they are the only candidates that need to be examined to locate an optimum.

## Vertices and Basic Feasible Points

The feasible set  $\{x \mid Ax = b, x \geq 0\}$  is a polytope, with vertices being points not on a straight line between two other points in the set.

- ▶ Geometrically, vertices are easily recognizable.
- ▶ Algebraically, vertices are exactly the basic feasible points.
- ▶ This links algebraic and geometric views, aiding understanding of the simplex method.

### Theorem 33: Vertices and Basic Feasible Points

All basic feasible points for  $\min c^T x$ , subject to  $Ax = b, x \geq 0$ , are vertices of the feasible polytope  $\{x \mid Ax = b, x \geq 0\}$ , and vice versa.

**Proof:** Let  $x$  be a basic feasible point with  $B = \{1, 2, \dots, m\}$ . Then,  $B = [A_i]_{i=1,\dots,m}$  is nonsingular, and:

$$x_{m+1} = x_{m+2} = \dots = x_n = 0.$$

If  $x = \alpha y + (1 - \alpha)z$  for feasible  $y, z$  and  $\alpha \in (0, 1)$ , then  $y_i = z_i = 0$  for  $i = m + 1, \dots, n$ .

7/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

At this stage, we can formalize the deep connection between geometry and algebra in linear programming. Recall that the feasible region defined by a system of linear equations and nonnegativity constraints is a convex polyhedron, and its corner points are what we call vertices. Geometrically, a vertex is a point that cannot be represented as a convex combination of two other feasible points. In algebraic terms, however, we have already introduced the notion of a basic feasible point: a solution where at most  $m$  variables are nonzero, with  $m$  being the number of constraints, and the associated columns of matrix  $A$  forming a linearly independent set.

Theorem thirty-three tells us that these two concepts are not just related but in fact equivalent. Every basic feasible point corresponds exactly to a vertex of the feasible polyhedron, and conversely, every vertex corresponds to some basic feasible point. This equivalence closes the gap between intuition and formal proof: the geometry of polytopes aligns perfectly with the algebra of bases and feasible solutions.

This result is crucial because it guarantees that the simplex method, which moves from one basis to another, is in fact moving across the vertices of the feasible region, and thus no relevant candidates for optimality are ever missed.

The proof begins by assuming we have a basic feasible point with basis set  $B$ . The corresponding basis matrix,  $B$ , is nonsingular, and all nonbasic variables are zero. Suppose we could write this point as a convex combination of two different feasible points. Then, because the nonbasic variables are all zero, the other points must also have zeros in these positions. Consequently, all three feasible solutions would agree on the same coordinates.

## Proof: Vertices and Basic Feasible Points (Continued)

Define  $x_b = (x_1, \dots, x_m)^T$ , similarly  $y_b, z_b$ . Since  $Ax = Ay = Az = b$ :

$$Bx_b = By_b = Bz_b = b.$$

As  $B$  is nonsingular,  $x_b = y_b = z_b$ , so  $x = y = z$ , contradicting  $y, z \neq x$ . Thus,  $x$  is a vertex.

- Conversely, let  $x$  be a vertex with nonzeros  $x_1, \dots, x_p$ . If  $A_1, \dots, A_p$  are linearly dependent, construct  $x(\epsilon) = x + \epsilon z$  as in prior proofs. For small  $|\epsilon|$ ,  $x(\hat{\epsilon})$  and  $x(-\hat{\epsilon})$  are feasible for some  $\hat{\epsilon} > 0$ . Since  $x = x(0)$  lies between them,  $x$  is not a vertex, a contradiction.
- Thus,  $A_1, \dots, A_p$  are linearly independent,  $p \leq m$ . If  $p < m$ , add  $m - p$  indices to  $\{1, \dots, p\}$  to form a basis  $\mathcal{B}$ , making  $x$  a basic feasible point.

□

### Definition: Degeneracy

A basis  $\mathcal{B}$  is *degenerate* if  $x_i = 0$  for some  $i \in \mathcal{B}$ , where  $x$  is the basic feasible solution for  $\mathcal{B}$ . The linear program  $\min c^T x$ , subject to  $Ax = b, x \geq 0$ , is degenerate if it has at least one degenerate basis and nondegenerate, otherwise.

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



8/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

On the other hand since the basis matrix is invertible, this implies that the basic components also coincide, making all points identical. Therefore, the supposed convex combination reduces to trivial equality, showing that our basic feasible point is indeed a vertex.

To complete the proof of equivalence, we now consider the reverse direction: suppose we begin with a vertex of the feasible polyhedron. We ask whether this point can be described as a basic feasible solution. Assume the vertex has  $p$  nonzero coordinates, and let the corresponding columns of  $A$  be denoted  $A_1$  through  $A_p$ . If these columns were linearly dependent, then we could construct a perturbation by adding  $\epsilon$  times some vector  $z$ , as in earlier arguments. For small positive or negative values of  $\epsilon$ , the perturbed solutions would remain feasible. But then our vertex would lie strictly between two distinct feasible points, which contradicts the definition of a vertex. Therefore, the associated columns must be linearly independent.

Since we have linear independence, the number  $p$  of nonzero components cannot exceed  $m$ , the number of constraints. If  $p$  equals  $m$ , then the vertex is already a basic feasible point. If  $p$  is strictly less than  $m$ , then we can extend the set of columns by adding additional independent ones, completing a full basis of size  $m$ . This construction proves that every vertex corresponds to some basic feasible point, establishing the equivalence in both directions.

This brings us naturally to the notion of degeneracy. A basis is called degenerate if the associated basic feasible solution has one or more basic variables equal to zero. In other words, although the column is part of the basis, the corresponding coordinate vanishes. A linear program is called degenerate if it admits at least one degenerate basis. Otherwise, it is nondegenerate. Degeneracy plays an important role in the analysis of the simplex method, because it can lead to situations where the algorithm cycles without making progress in the objective value. Later, we will examine strategies developed to overcome this issue, ensuring that the simplex method remains efficient and reliable.

## Introduction to the Revised Simplex Method

There are a number of variants the simplex method; we start with the *revised simplex method*, a variant of the simplex method for solving  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$  and further consider some other versions.

- All iterates are basic feasible points, i.e., vertices of the feasible polytope  $\{x \mid Ax = b, x \geq 0\}$ .
- Most steps move to an adjacent vertex, changing one index in the basis  $\mathcal{B}$ , typically reducing  $c^T x$ .
- Unbounded case: Move along an edge reducing  $c^T x$ , without reaching a vertex.
- Key task: Choose which index to remove from  $\mathcal{B}$ , replacing it with one from outside  $\mathcal{B}$ .

Define the nonbasic index set  $\mathcal{N} = \{1, \dots, n\} \setminus \mathcal{B}$ , and matrices:

$$B = [A_i]_{i \in \mathcal{B}}, \quad N = [A_i]_{i \in \mathcal{N}}.$$

Partition vectors:  $x_{\mathcal{B}} = [x_i]_{i \in \mathcal{B}}$ ,  $x_{\mathcal{N}} = [x_i]_{i \in \mathcal{N}}$ , similarly for  $s_{\mathcal{B}}$ ,  $s_{\mathcal{N}}$ ,  $c_{\mathcal{B}}$ ,  $c_{\mathcal{N}}$ .

From  $Ax = b$  (??):

$$Bx_{\mathcal{B}} + Nx_{\mathcal{N}} = b.$$

The primal iterate is:

$$x_{\mathcal{B}} = B^{-1}b, \quad x_{\mathcal{N}} = 0.$$

9/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

We now move to a computational perspective. The simplex method in its original form requires explicit manipulation of the full constraint matrix, which is often inefficient for large-scale problems. To address this, a refined approach known as the revised simplex method was developed. The revised version retains the essential structure of moving from one basic feasible point to another, but it does so in a more economical way by focusing on the basis matrix and related quantities, rather than handling all variables simultaneously.

At each iteration, the algorithm maintains a current basis  $B$ , and the set of basic and nonbasic variables is determined accordingly. The basic variables are obtained by solving the system  $Bx_{\mathcal{B}} = b$ , while all nonbasic variables are set to zero. In this way, each iterate is guaranteed to be a basic feasible solution, that is, a vertex of the feasible region. The algorithm then decides whether it can reduce the objective function by exchanging one index of the basis with one outside it. Such an exchange corresponds geometrically to moving along an edge of the polyhedron to an adjacent vertex.

In most cases, the algorithm makes progress toward improving the objective function. However, if the problem is unbounded, the algorithm detects a direction along which the objective can be decreased indefinitely, without ever reaching another vertex. This situation indicates that no finite optimal solution exists.

The central computational task of the revised simplex method is determining which variable should leave the basis and which should enter. These decisions are crucial for efficiency, and we will later explore the rules used for pivot selection.

## Revised Simplex Method: KKT Conditions

For a basic feasible point in  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , with basis  $\mathcal{B}$ , we derive primal and dual variables satisfying KKT conditions.

- Since  $B$  is nonsingular and  $x_{\mathcal{B}} = B^{-1}b \geq 0$ ,  $x$  satisfies  $Ax = b$  and  $x \geq 0$ .
- Set  $s_{\mathcal{B}} = 0$  to satisfy complementarity ( $x_i s_i = 0$ ).

From  $A^T \lambda + s = c$ , partition into:

$$B^T \lambda = c_{\mathcal{B}}, \quad N^T \lambda + s_{\mathcal{N}} = c_{\mathcal{N}}.$$

Solve for  $\lambda$ :

$$\lambda = B^{-T} c_{\mathcal{B}}.$$

Compute  $s_{\mathcal{N}}$  (reduced costs of nonbasic variables  $x_{\mathcal{N}}$ ):

$$s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda = c_{\mathcal{N}} - (B^{-1}N)^T c_{\mathcal{B}}.$$

Computing  $s_{\mathcal{N}}$  is called *pricing*. The components of  $s_{\mathcal{N}}$  are often called the *reduced costs* of the nonbasic variables  $x_{\mathcal{N}}$ .

10/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

To fully justify the revised simplex method, we recall that linear programming problems can be expressed as primal–dual pairs, with optimality characterized by the Karush–Kuhn–Tucker, or KKT, conditions. For a given basis  $B$ , we can explicitly construct both the primal and dual variables satisfying these conditions.

On the primal side, since the basis matrix  $B$  is invertible, we can solve for the basic variables as  $x_{\mathcal{B}} = B^{-1}b$ . By definition, the nonbasic variables are zero. This ensures feasibility: the equality constraints are satisfied, and the variables remain nonnegative. Furthermore, the complementarity condition requires that each basic variable is paired with a dual slack variable equal to zero. Thus, for all indices in the basis, the slack variables vanish.

On the dual side, the relation  $A^T \lambda + s = c$  can be partitioned into two systems:  $B^T \lambda = c_{\mathcal{B}}$ , and  $N^T \lambda + s_{\mathcal{N}} = c_{\mathcal{N}}$ . Solving the first system gives the dual multipliers  $\lambda$  as  $B^{-T} c_{\mathcal{B}}$ . Substituting into the second expression yields the reduced costs:  $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$ . This can also be written as  $c_{\mathcal{N}} - (B^{-1}N)^T c_{\mathcal{B}}$ .

The only KKT condition that remains to be satisfied is the nonnegativity condition:  $s \geq 0$ . Since  $s_B = 0$ , we need  $s_N \geq 0$ , where  $s_N = c_N - (B^{-1}N)^T c_B$ .

- If  $s_N \geq 0$ , then  $\langle x, \lambda, s \rangle$  is optimal, and the algorithm stops.
- If some  $s_q < 0$  for  $q \in N$ , choose  $q$  as the *entering index*. Increasing  $x_q > 0$  reduces  $c^T x$  if  $x$  remains feasible.

Pivoting Process:

- Increase  $x_q$  from zero, keeping other  $x_N$  at zero.
- Adjust  $x_B$  to maintain  $Ax = b$ .
- Increase  $x_q$  until some  $x_p$  ( $p \in B$ ) reaches zero (the *leaving index*), or find no such  $p$  (unbounded case).
- Replace  $p$  with  $q$  in  $B$ .

This process, updating  $B$  and tracking  $x, \lambda, s$ , is called *pivoting*.

For a new iterate  $x^+$  in  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , with  $x_i^+ = 0$  for  $i \in N \setminus \{q\}$ , we maintain  $Ax^+ = b$ .

11/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

At this stage of the algorithm, all conditions except nonnegativity have already been satisfied. The central question becomes whether the reduced costs, denoted by the vector  $s_N$ , are nonnegative. If they are, we have reached optimality. If, however, some component is negative, that variable offers a direction in which the objective can be decreased. The corresponding index is chosen as the entering variable, and we attempt to increase it from zero while preserving feasibility.

Maintaining feasibility requires adjustments of the basic variables. Specifically, when we raise the entering variable, the equality constraint  $Ax = b$  must still hold. This forces a compensating decrease in some basic variables. If one of these basic variables is driven to zero, it can no longer remain in the basis and must leave. This is the essence of pivoting: exchanging one variable from the basis with one from the nonbasis.

The procedure has two possible outcomes. In the generic case, some basic variable hits zero, and we update the basis accordingly, moving to a new vertex of the feasible polytope. In the exceptional case, no basic variable is forced to zero, which indicates that the objective can be improved indefinitely. This is the unbounded situation, where the problem has no finite solution.

Pivoting is not only a computational mechanism but also a geometric one: it corresponds to moving along an edge of the feasible polyhedron. Each pivot replaces one defining hyperplane of the vertex with another, ensuring that we always remain on a vertex while searching for improvement. This edge-following nature explains the efficiency of the simplex method in practice.

## Revised Simplex Method: Pivoting Algebra

► Since  $Ax^+ = Bx_B^+ + A_q x_q^+ = Bx_B = Ax$ , we derive:

$$x_B^+ = x_B - B^{-1}A_q x_q^+. \quad (i)$$

- Geometrically, this is a move along a polytope edge, reducing  $c^T x$ , until a new vertex where  $x_p = 0$  for some  $p \in \mathcal{B}$ .
- Update  $\mathcal{B}$ : Remove p, add q.

Effect on Objective:

$$c^T x^+ = c_B^T x_B^+ + c_q x_q^+ = c_B^T x_B - c_B^T B^{-1} A_q x_q^+ + c_q x_q^+.$$

Since  $\lambda = B^{-T} c_B$ , we have  $c_B^T B^{-1} = \lambda^T$ , and  $A_q^T \lambda = c_q - s_q$ , so:

$$c_B^T B^{-1} A_q x_q^+ = (c_q - s_q) x_q^+.$$

Thus:

$$c^T x^+ = c_B^T x_B - (c_q - s_q) x_q^+ + c_q x_q^+ = c^T x + s_q x_q^+.$$

Since  $s_q < 0$ , it follows that the step (i) produces a decrease in the primal objective function  $c^T x$  whenever  $x_q^+ > 0$ .

12/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

To formalize the pivot operation, we consider the structure of the constraints. When the entering variable is increased, the basic variables must adjust to keep the equality system valid. Algebraically, this relation is expressed as  $x_B^+ = x_B - B^{-1}A_q x_q^+$ .

This equation captures the trade-off: raising the entering variable reduces some of the basic ones proportionally. The vector  $B^{-1}A_q$  specifies exactly how the adjustment is distributed among the basic components.

Geometrically, the formula describes motion along an edge of the feasible polytope. The path is linear until a boundary is hit, meaning one of the basic variables becomes zero. At that point, the vertex changes, and the new basis reflects the exchange between leaving and entering variables.

The impact on the objective can also be made precise. Substituting into the cost function gives  $c^T x^+ = c^T x + s_q x_q^+$ .

Here,  $s_q$  is the reduced cost associated with the entering variable. If this reduced cost is negative, then increasing the corresponding variable decreases the objective function. The magnitude of improvement is proportional to the step length. This shows clearly why negative reduced costs signal opportunities for descent.

Thus, the algebra provides both a computational rule and an intuitive explanation: pivoting moves us to another vertex while ensuring that each step strictly reduces the objective, provided the reduced cost is negative. This mechanism guarantees systematic progress toward optimality or unboundedness, leaving no ambiguity in the direction of search.

If  $x_{\mathcal{B}}^+ = x_{\mathcal{B}} - B^{-1}A_q x_q^+ \geq 0$  for all  $x_q^+ > 0$ , the linear program  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , is unbounded. The simplex method identifies a ray in the feasible polytope where  $c^T x \rightarrow -\infty$ .

- If the basis  $\mathcal{B}$  is nondegenerate (i.e.,  $x_i > 0$  for all  $i \in \mathcal{B}$ ), then  $x_q^+ > 0$ , ensuring a strict decrease in  $c^T x$ .
- For a nondegenerate problem, every step reduces  $c^T x$ .

### Theorem 34: Termination of the simplex method

If the linear program  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ , is nondegenerate and bounded, the simplex method terminates at a basic optimal point.

Proof:

- Each step strictly decreases  $c^T x$ , so the same basic feasible point is never revisited.
- There are finitely many bases  $\mathcal{B}$  (subsets of  $m$  indices from  $\{1, \dots, n\}$ ), hence finitely many basic feasible points.
- Since the problem is bounded, the method terminates at a basic optimal point.  $\square$

13/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

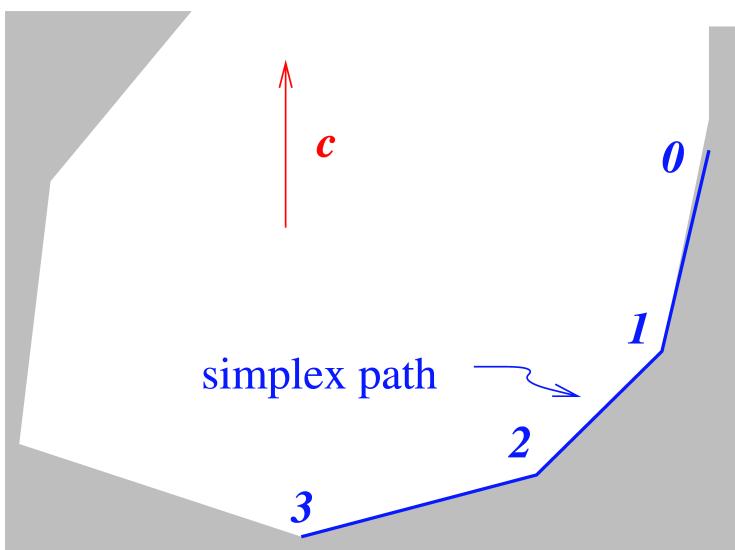
The simplex method continues this process of exchanging variables until one of two situations arises. The first possibility is that no further negative reduced costs exist, which certifies optimality. The second is that the problem is unbounded. This occurs when, after selecting an entering variable, the corresponding direction allows all basic variables to remain nonnegative, even as the step length grows indefinitely. In such a case, the objective decreases without bound, and no finite solution exists.

For bounded problems, progress is guaranteed under the assumption of nondegeneracy. Nondegeneracy means that every basic variable is strictly positive, so each pivot produces an actual change in the objective value. Because the cost decreases strictly at every step, the method can never revisit the same basic feasible point.

The finite termination argument rests on combinatorial grounds. There are only finitely many possible bases, since a basis is determined by selecting  $m$  indices out of  $n$ . Each corresponds to a potential basic feasible solution. With every pivot leading to a new and distinct solution, the method cannot continue indefinitely without reaching either an optimal vertex or detecting unboundedness.

Thus, under the assumptions of boundedness and nondegeneracy, the simplex method is guaranteed to terminate with an optimal solution. The result is significant: despite the fact that the algorithm moves locally from vertex to vertex, it cannot cycle or diverge. The combination of finite basis count and strictly improving steps ensures convergence in a finite number of iterations.

This result gives us a proof of part (iii) of Theorem 32 in the case in which the linear program is nondegenerate. The proof of finite termination is considerably more complex when nondegeneracy of the primal problem is not assumed, as we discuss a little bit further.



Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



Figure: Simplex iterates for a two-dimensional problem.

14/29 || SPbU &amp; HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

To gain intuition, it is helpful to visualize the method in two dimensions. In this case, the feasible region is a polygon defined by the intersection of linear inequalities. Each vertex corresponds to a basic feasible solution. The simplex method traces a path along the edges of this polygon, moving systematically from one vertex to another in search of improvement.

The entering and leaving variable rules translate geometrically into edge-following rules. When a negative reduced cost is detected, the method chooses an edge of the polygon that leads downhill in terms of the objective function. Movement continues along this edge until reaching the next vertex, which represents the exchange of one constraint for another. The new vertex then serves as the starting point for the next iteration.

This visualization explains why the simplex method often performs so efficiently in practice. Even though the number of potential vertices can be very large in higher dimensions, the method typically explores only a small fraction. By always following an improving edge, it avoids wandering through the interior and instead exploits the polyhedral structure directly.

In two dimensions, the process is particularly transparent: we can literally see the objective function decreasing as the algorithm marches along polygon edges. Each pivot corresponds to sliding down to a lower-cost vertex, step by step, until the minimum is reached or unboundedness is revealed. This geometric picture builds intuition that extends to higher dimensions, even though direct visualization is no longer possible.

## Revised Simplex Method: One Step

We summarize the mechanics of a single step in the revised simplex method for  $\min c^T x$ , subject to  $Ax = b$ ,  $x \geq 0$ .

### Procedure: One Step of Simplex

- 1: Input:  $\mathcal{B}$ ,  $\mathcal{N}$ ,  $x_{\mathcal{B}} = B^{-1}b \geq 0$ ,  $x_{\mathcal{N}} = 0$
- 2: Solve  $B^T \lambda = c_{\mathcal{B}}$  for  $\lambda$
- 3: Compute  $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$  ▷ Pricing
- 4: if  $s_{\mathcal{N}} \geq 0$  then
- 5:     Stop ▷ Optimal point found
- 6: end if
- 7: Select  $q \in \mathcal{N}$  with  $s_q < 0$  as the entering index
- 8: Solve  $Bd = A_q$  for  $d$
- 9: if  $d \leq 0$  then
- 10:     Stop ▷ Problem is unbounded
- 11: end if
- 12: Compute  $x_q^+ = \min_{i|d_i > 0} (x_{\mathcal{B}})_i / d_i$ , denote minimizing  $i$  as  $p$
- 13: Update  $x_{\mathcal{B}}^+ = x_{\mathcal{B}} - dx_q^+$ ,  $x_{\mathcal{N}}^+ = (0, \dots, x_q^+, \dots, 0)^T$
- 14: Update  $\mathcal{B}$ : Add  $q$ , remove index  $p$

Sensitivity Analysis

Fundamental Theorem

**SIMPLEX METHOD**

Steepest-Edge Rule

Two-Phase Simplex



15/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

The revised simplex method executes each iteration through a precise sequence of linear algebraic operations. We begin with a chosen basis, denoted by  $\mathcal{B}$ , and its complement, the nonbasis  $\mathcal{N}$ . The basic variables are obtained as  $B^{-1}b$ , ensuring feasibility, while all nonbasic variables remain equal to zero. From this point, dual multipliers are recovered by solving the system  $B^T \lambda = c_{\mathcal{B}}$ , where  $c_{\mathcal{B}}$  denotes the cost coefficients corresponding to the basis. Once  $\lambda$  is available, we compute the reduced costs for the nonbasic variables as  $c_{\mathcal{N}} - N^T \lambda$ .

If every reduced cost is nonnegative, then no improvement is possible, and we have reached an optimal solution. Otherwise, some component is strictly negative, signaling a direction of descent. The corresponding index, denoted  $q$ , is chosen as the entering variable. To maintain feasibility when  $x_q$  increases, we must determine how the basic variables adjust. This requires solving  $Bd = A_q$ , where  $A_q$  is the  $q$ -th column of the matrix  $A$ . If the direction vector  $d$  has only nonpositive entries, then no basic variable is ever forced to zero, and the problem is unbounded. Otherwise, the ratio test identifies the maximum permissible step length: we take the minimum of the quotients between the current basic values and the positive entries of  $d$ . The minimizing index indicates the leaving variable, denoted  $p$ .

With these updates, we modify both the basis and the solution vector, ensuring feasibility and reducing the objective. This compact procedure guarantees that each step either moves us closer to optimality or certifies unboundedness, fully capturing the mechanics of a simplex pivot in algebraic form.



## Example

Consider:  $\min -4x_1 - 2x_2$  s.t.  $x_1 + x_2 + x_3 = 5$ ,  $2x_1 + \frac{1}{2}x_2 + x_4 = 8$ ,  $x \geq 0$ , or in matrix form:  $Ax = b$ , with  $A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & \frac{1}{2} & 0 & 1 \end{bmatrix}$  and  $b = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$ .

- Basis  $\mathcal{B} = \{3, 4\}$ , so  $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  (columns of A for  $x_3, x_4$ ). Set  $x_{\mathcal{B}} = B^{-1}b = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$ ,  $x_{\mathcal{N}} = (x_1, x_2) = (0, 0)$ ,  $c^T x = 0$ .
- Solve  $B^T \lambda = c_{\mathcal{B}} = (0, 0)^T$ :  $\lambda = (0, 0)^T$ . Compute  $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$ , where  $c_{\mathcal{N}} = (-4, -2)^T$ ,  $N = \begin{bmatrix} 1 & 1 \\ 2 & \frac{1}{2} \end{bmatrix}$ :  $s_{\mathcal{N}} = (-4, -2)^T - \begin{bmatrix} 1 & 2 \\ 1 & \frac{1}{2} \end{bmatrix} (0, 0)^T = (-4, -2)^T$ .
- Since  $s_1, s_2 < 0$ , choose  $q = 1$ . Solve  $Bd = A_1 = (1, 2)^T$ :  $d = (1, 2)^T$ . As  $d \not\leq 0$ , not unbounded.
- Ratio test:  $\min \left\{ \frac{(x_{\mathcal{B}})_i}{d_i} \mid d_i > 0 \right\} = \min \left\{ \frac{5}{1}, \frac{8}{2} \right\} = 4$ , so  $p = 2$  (corresponding to the index 4),  $x_1^+ = 4$ . Update  $\mathcal{B} = \{3, 1\}$ ,  $\mathcal{N} = \{4, 2\}$ .

16/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

To deepen our understanding, let us analyze a concrete numerical example. Consider the objective function to minimize  $-4x_1 - 2x_2$ , subject to the constraints:  $x_1 + x_2 + x_3 = 5$ , and  $2x_1 + \frac{1}{2}x_2 + x_4 = 8$ , with all variables nonnegative. In matrix notation, the constraint matrix A is two by four, with rows  $[1 \ 1 \ 1 \ 0]$  and  $[2 \ \frac{1}{2} \ 0 \ 1]$ , while the right-hand side vector b is  $\begin{bmatrix} 5 \\ 8 \end{bmatrix}$ .

We initially select the basis indices three and four, corresponding to the slack variables. The basis matrix in this case is the identity matrix, so the basic variables equal five and eight, respectively, while the nonbasic decision variables  $x_1$  and  $x_2$  are zero. The initial objective value is zero. Next, we solve  $B^T \lambda = 0$ , which gives  $\lambda$  equal to the zero vector. Then we compute the reduced costs:  $c_{\mathcal{N}}$  is the vector  $(-4, -2)$ . Subtracting  $N^T \lambda$ , which remains zero, we obtain reduced costs of  $-4$  and  $-2$ . Both are negative, so improvement is possible.

Following the rule, we select  $x_1$  to enter the basis. We then solve  $Bd =$  column one of A, which is the vector  $(1, 2)$ . The result is  $d = (1, 2)$ . Because some components of d are positive, the problem is not unbounded. The ratio test is then applied: we take the minimum between  $5/1$  and  $8/2$ , which are five and four. The minimum is four, corresponding to variable  $x_4$ , which therefore leaves the basis. The updated basis now contains indices three and one.

Continued

Continuing: with  $A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & \frac{1}{2} & 0 & 1 \end{bmatrix}$ ,  $b = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$ ,  $c = [-4 \ -2 \ 0 \ 0]^T$ .

► Iteration 2:  $\mathcal{B} = \{3, 1\}$ ,  $B = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$ ,  $x_{\mathcal{B}} = \begin{bmatrix} x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 & -1/2 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$ ,  $c^T x = -16$ .

Solve  $\lambda = (B^T)^{-1} c_{\mathcal{B}} = \begin{bmatrix} 1 & 0 \\ -1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 0 \\ -4 \end{bmatrix} = (0, -2)^T$ .

$\mathcal{N} = \{4, 2\}$ ,  $c_{\mathcal{N}} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$ ,  $N = \begin{bmatrix} 0 & 1 \\ 1 & \frac{1}{2} \end{bmatrix}$ . Compute:  $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$ .

Choose  $q = 2$  ( $s_2 < 0$ ), solve:  $Bd = A_2 \Rightarrow d = B^{-1}A_2 = (\frac{3}{4}, \frac{1}{4})^T$ .

Ratio test:  $\min \left\{ \frac{4}{3}, 16 \right\} = \frac{4}{3}$ ,  $p = 1$  (corresponding to the index 3),  $x_2^+ = \frac{4}{3}$ .

Update:  $\mathcal{B} = \{2, 1\}$ ,  $\mathcal{N} = \{4, 3\}$ .

► Iteration 3:  $\mathcal{B} = \{2, 1\}$ ,  $B = \begin{bmatrix} 1 & 1 \\ \frac{1}{2} & 2 \end{bmatrix}$ ,  $x_{\mathcal{B}} = \begin{bmatrix} \frac{4}{3} \\ \frac{11}{3} \end{bmatrix}$ ,  $c^T x = -\frac{52}{3}$ ,  $s_{\mathcal{N}} = (\frac{4}{3}, \frac{4}{3})^T$ .

Since  $s_{\mathcal{N}} \geq 0$ , stop: optimal.

Sensitivity Analysis

Fundamental Theorem

**SIMPLEX METHOD**

Steepest-Edge Rule

Two-Phase Simplex



### Comments

Continuing with the example, we now examine subsequent iterations. With the updated basis containing indices three and one, the corresponding basis matrix is two by two with columns  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$  and  $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ . Its inverse, when multiplied by the right-hand side vector, yields new basic values:  $x_3 = 1$ , and  $x_1 = 4$ . The objective function evaluates to  $-16$ . Next, solving  $B^T \lambda = c_{\mathcal{B}}$  which is  $(0, -4)^T$  produces  $\lambda = (0, -2)^T$ .

The reduced costs are then computed. For the nonbasic variables, namely  $x_4$  and  $x_2$ , we have  $c_{\mathcal{N}} = (0, -2)$ . Subtracting  $N^T \lambda$ , we obtain reduced costs of 2 and  $-1$ . Since  $x_2$  has a negative reduced cost, it becomes the entering variable. To preserve feasibility, we solve  $Bd =$  column two of  $A$ , which yields  $d = (3/4, 1/4)$ . The ratio test is applied: the minimum of  $1/(3/4)$ , and  $4/(1/4)$ , gives  $4/3$ . Therefore,  $x_3$  is the leaving variable, and  $x_2$  enters with value  $4/3$ .

The next basis contains indices two and one. Solving again, we obtain basic values  $x_2 = 4/3$  and  $x_1 = 11/3$ . The objective function equals  $-52/3$ . At this stage, the reduced costs for the nonbasic variables are both positive  $4/3$ . Since no negative reduced costs remain, optimality is reached.

This sequence of calculations demonstrates the complete cycle of the revised simplex method: from an initial feasible vertex, through systematic pivots, to an optimal solution. Each iteration carefully balances feasibility and optimality conditions, guiding the process to termination.

Enhancing the simplex method for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$  requires addressing:

- ▶ Linear algebra: Maintain an LU factorization of basis matrix  $B$  to solve for  $\lambda$  and  $d$ .
- ▶ Entering index: Select  $q$  from negative components of  $s_N$ .
- ▶ Degeneracy: Handle degenerate bases and steps where  $x_q^+ \leq 0$  violates feasibility.

Linear Algebra: Solve  $B^T \lambda = c_B$ ,  $Bd = A_q$  using LU factorization  $LU = B$  ( $L$  unit lower triangular,  $U$  upper triangular).

- ▶ Avoid computing  $B^{-1}$  explicitly; use triangular substitutions.
- ▶ Update factorization per iteration (single column change in  $B$ ).
- ▶ For sparse  $B$ , reorder rows/columns for stability and sparsity (e.g., Markowitz pivot strategy).
- ▶ Solve  $Bd = A_q$  via:  $Lz = A_q$ ,  $Ud = z$ .

Sensitivity Analysis

Fundamental Theorem

**SIMPLEX METHOD**

Steepest-Edge Rule

Two-Phase Simplex



### Comments

Beyond the mechanics of individual pivots, efficient implementation of the revised simplex method requires attention to numerical linear algebra. Each iteration involves solving two types of systems:  $B^T \lambda = c_B$ , and  $Bd = A_q$ . Although one might be tempted to compute  $B^{-1}$  explicitly, this is computationally wasteful and numerically unstable. Instead, we maintain a factorization of  $B$ , typically in the form of an LU decomposition, where  $L$  is unit lower triangular and  $U$  is upper triangular. This structure allows forward and backward substitutions, yielding solutions at far lower cost than recomputing inverses.

Because each pivot modifies only one column of the basis matrix, the factorization can be updated efficiently without starting from scratch. Specialized update formulas ensure that computational effort remains proportional to the matrix dimension, rather than its square or cube. Moreover, in large-scale problems, the basis matrix is often sparse. Preserving sparsity is crucial, since fill-in during factorization can destroy efficiency. Strategies such as the Markowitz pivot rule reorder rows and columns to minimize fill-in while maintaining numerical stability.

In practice, this careful treatment of linear algebra makes the revised simplex method highly scalable. The ability to exploit sparsity and incremental updates is what enables modern implementations to handle problems with tens or even hundreds of thousands of constraints and variables. Without these optimizations, the theoretical mechanics of the method would remain impractical at large scales. Thus, efficient factorization, stability control, and update strategies are as essential to the success of the simplex method as the underlying mathematics of pivoting.



Solving  $B^T \lambda = c_B$ : Use LU factorization  $LU = B$  via:

$$U^T \tilde{\lambda} = c_B, \quad L^T \lambda = \tilde{\lambda}.$$

- Updating LU Factors: After replacing index  $p$  with  $q$  in basis  $\mathcal{B}$  after one step of the simplex method, update  $B$  to  $B^+$  (replace column  $B_p$  with  $A_q$ ).
- $L^{-1}B^+$  is upper triangular except in column  $p$ .
- Apply cyclic permutation  $P_{\downarrow}$ : Move column  $p$  to last position  $m$ , shift columns  $p+1, \dots, m$  left, and permute rows  $p$  to  $m$ . Result:  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T$  shifts non-triangular part to last row.
- Perform sparse Gaussian elimination on  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T = L_{\downarrow}U_{\downarrow}$ :
  - $L_{\downarrow}$ : Identity except last row.
  - $U_{\downarrow}$ : Matches  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T$  except modified  $(m, m)$  element and zeroed off-diagonal last row elements.

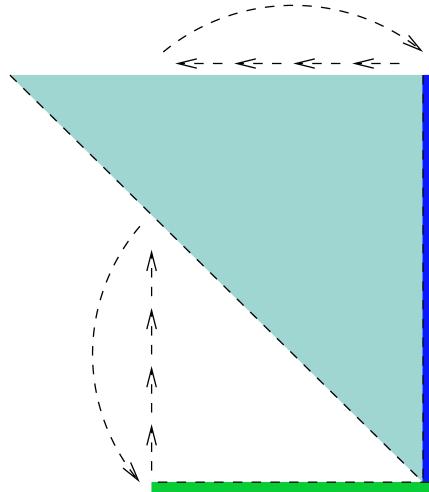
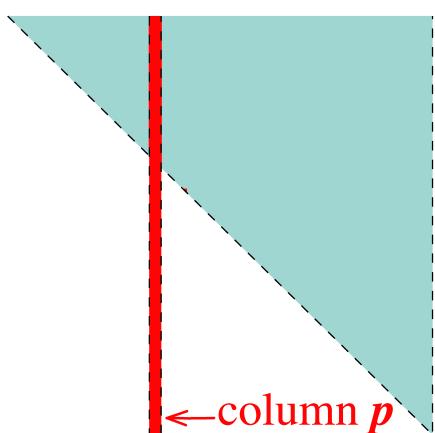
19/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

When solving the dual system given by  $B^T \lambda = c_B$ , the LU factorization of the basis matrix provides a computationally efficient route. Recall that LU factorization decomposes the basis matrix  $B$  into the product of a lower triangular matrix  $L$  and an upper triangular matrix  $U$ . To solve for  $\lambda$ , we instead consider the system  $U^T \tilde{\lambda} = c_B$ , followed by  $L^T \lambda = \tilde{\lambda}$ . This two-step triangular system avoids directly computing the inverse of  $B$ , which would be numerically unstable and computationally expensive.

After a pivot step, the basis matrix changes: one column, say column  $p$ , is replaced with the entering column  $A_q$ . This produces the new basis matrix  $B^+$ . Rather than recomputing the full LU factorization from scratch, which would cost cubic time in the dimension  $m$ , we instead update the factorization incrementally. It can be noted that the matrix  $L^{-1}B^+$  is nearly upper triangular: all columns except column  $p$  retain their triangular structure. Thus, the only disturbance is localized.

To systematically restore triangular form, we apply a cyclic permutation. This permutation, denoted  $P_{\downarrow}$ , moves column  $p$  to the last column and permutes rows accordingly. After this rearrangement, the non-triangular structure is confined to the last row, making the system well suited for a sparse Gaussian elimination step. Performing elimination yields new triangular factors  $L_{\downarrow}$  and  $U_{\downarrow}$ , which differ only minimally from the original ones. In this way, the revised simplex method maintains an efficient updating mechanism without the prohibitive cost of refactorization.



**Figure:** Left:  $L^{-1}B^+$ , which is upper triangular except for the column occupied by  $A_p$ . Right: After cyclic row and column permutation  $P_{\downarrow}$ , the non-upper triangular part of  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T$  appears in the last row.

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



## Comments

The role of cyclic permutation in the updating procedure is subtle but crucial. When the entering column replaces column  $p$  of the basis, the structure  $L^{-1}B^+$  remains upper triangular except at that single column. If we were to attempt Gaussian elimination immediately, the disturbance would spread through the matrix, destroying sparsity and efficiency. Instead, we strategically reorder rows and columns so that the problematic column is shifted to the last position.

The permutation operator  $P_{\downarrow}$  performs this rearrangement. Specifically, column  $p$  is moved to column  $m$ , while columns from  $p + 1$  through  $m$  are shifted one position to the left. At the same time, rows  $p$  through  $m$  are cyclically permuted, preserving consistency. The effect of this operation is that the non-triangular part, originally scattered, becomes localized entirely in the final row. This containment is what enables efficient elimination.

After applying  $P_{\downarrow}$ , the new matrix  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T$  can be expressed as a nearly upper triangular form. Only the last row contains off-diagonal elements. Sparse Gaussian elimination then operates exclusively on this row, transforming it into a standard triangular shape. The resulting matrices, denoted  $L_{\downarrow}$  and  $U_{\downarrow}$ , preserve triangularity while modifying only the necessary entries. Importantly,  $L_{\downarrow}$  is equal to the identity matrix except in the last row, while  $U_{\downarrow}$  matches the permuted structure except with a modified bottom-right element. This process highlights a fundamental principle of the revised simplex method: each update is localized, efficient, and exploits matrix structure to minimize computational effort.

## Revised Simplex Method: LU Update Case (1)

Updating LU factorization for  $m = 5$ ,  $p = 2$  (replace column 2 with  $L^{-1}A_q$ ).

► Given:  $L^{-1}B = U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ & u_{22} & u_{23} & u_{24} & u_{25} \\ & & u_{33} & u_{34} & u_{35} \\ & & & u_{44} & u_{45} \\ & & & & u_{55} \end{bmatrix}$ ,  $L^{-1}A_q = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix}$ .

► Form:  $L^{-1}B^+ = \begin{bmatrix} u_{11} & w_1 & u_{13} & u_{14} & u_{15} \\ & w_2 & u_{23} & u_{24} & u_{25} \\ & w_3 & u_{33} & u_{34} & u_{35} \\ & w_4 & & u_{44} & u_{45} \\ & w_5 & & & u_{55} \end{bmatrix}$ .

► Apply permutation  $P_\downarrow$ : Move column 2 to position 5, shift columns 3–5 left,

permute rows 2–5. Result:  $P_\downarrow L^{-1}B^+P_\downarrow^T = \begin{bmatrix} u_{11} & u_{13} & u_{14} & u_{15} & w_1 \\ & u_{33} & u_{34} & u_{35} & w_3 \\ & & u_{44} & u_{45} & w_4 \\ & & & u_{55} & w_5 \\ u_{23} & u_{24} & u_{25} & & w_2 \end{bmatrix}$ .

21/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

**SIMPLEX METHOD**

Steepest-Edge Rule

Two-Phase Simplex



### Comments

To illustrate the mechanics of the LU update, let us consider a concrete case where the dimension  $m$  equals five and the replaced column index  $p$  equals two. Initially, we have the factorization  $L^{-1}B = U$ , where  $U$  is a five-by-five upper triangular matrix with diagonal entries  $u_{11}$  through  $u_{55}$ . The entering column  $A_q$ , after being premultiplied by  $L^{-1}$ , yields the vector  $w$  with components  $w_1$  through  $w_5$ .

Forming the updated product  $L^{-1}B^+$  amounts to substituting the second column of  $U$  with this vector  $w$ . As a result, the new matrix preserves triangular structure everywhere except in column two, which now contains the entries  $w_1$  through  $w_5$ . This breaks the upper triangularity because entries below the diagonal in that column are generally nonzero.

To prepare for elimination, we apply the cyclic permutation operator  $P_\downarrow$ . In this case, column two is moved to position five, and columns three through five shift one place to the left. Simultaneously, rows two through five undergo the same cyclic shift. The transformed matrix  $P_\downarrow L^{-1}B^+P_\downarrow^T$  now exhibits the desired structure: the disturbance is entirely relocated to the last row. The upper left block, consisting of the first four rows and columns, remains perfectly triangular. This sets the stage for applying sparse elimination to restore full triangularity with minimal work, highlighting how theory translates into efficient computational practice.



Continuing LU factorization update for  $m = 5, p = 2$ .

► Gaussian elimination on  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T$  gives:  $L_1 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ 0 & l_{52} & l_{53} & l_{54} & 1 \end{bmatrix}$ ,

$$U_1 = \begin{bmatrix} u_{11} & u_{13} & u_{14} & u_{15} & w_1 \\ & u_{33} & u_{34} & u_{35} & w_3 \\ & & u_{44} & u_{45} & w_4 \\ & & & u_{55} & w_5 \\ & & & & \hat{w}_2 \end{bmatrix}, \text{ for certain values of } l_{52}, l_{53}, l_{54} \text{ and } \hat{w}_2.$$

- $L_1$ : Identity except last row;  $U_1$ : Matches  $P_{\downarrow}L^{-1}B^+P_{\downarrow}^T$  except modified (5,5) element and zeroed off-diagonal last row elements.
- Result:  $B^+ = L^+U^+$ , where  $L^+ = LP_{\downarrow}^T L_1$ ,  $U^+ = U_1 P_{\downarrow}$ .

22/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Continuing with the same example, we now perform Gaussian elimination on the permuted matrix. The result is a factorization into  $L_1$  and  $U_1$ . The matrix  $L_1$  is identical to the identity except for its last row, where entries  $l_{52}$ ,  $l_{53}$ , and  $l_{54}$  appear. These coefficients record the multipliers used in the elimination process. The matrix  $U_1$  retains the triangular form: its upper left block matches the permuted structure, while the last row has been cleared of off-diagonal elements, leaving only the modified entry  $\hat{w}_2$  in the bottom-right corner.

This operation produces a valid LU decomposition of the updated basis matrix  $B^+$ . Explicitly, we can write  $B^+ = L^+U^+$ , where  $L^+ = LP_{\downarrow}^T L_1$ , and  $U^+ = U_1 P_{\downarrow}$ . Thus, both factors are obtained without recomputing the entire decomposition from scratch. The efficiency gain is significant: each pivot requires only localized row operations, rather than a global recomputation.

From a computational perspective, this method ensures that the revised simplex algorithm scales gracefully to large problems. Because only a single column update is needed at each iteration, the work per pivot remains manageable. At the same time, numerical stability is maintained by careful use of triangular structure. This combination of localized updates, sparsity preservation, and stability makes the LU updating procedure a cornerstone of modern simplex implementations.



Selecting entering index  $q$  from negative  $s_{\mathcal{N}}$  components in simplex method for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ .

- ▶ **Goal:** Choose  $q$  to reduce  $c^T x$ , ideally minimizing steps to optimal  $x^*$ . Practical strategies focus on current iteration.
- ▶ **Dantzig's Rule:** Select  $q$  with most negative  $s_q$  in  $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$ . Maximizes  $c^T x$  decrease per unit  $x_q$ , but  $x_q^+$  may be small.
- ▶ **Partial Pricing:** Compute subvector of  $s_{\mathcal{N}}$  (less costly than full  $N^T$  multiplication). Choose  $q$  from negative entries, cycling nonbasic indices.
- ▶ **Multiple Pricing:** For a subset of indices, compute  $s_q$ ,  $d = B^{-1} A_q$ ,  $\max x_q^+$  (feasible), and  $s_q x_q^+$ . Reuse subset until  $s_q \geq 0$ , then update  $s_{\mathcal{N}}$ . Reduces memory access.
- ▶ **Heuristics:** Combine partial and multiple pricing for efficiency.

23/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

When applying the revised simplex method, the choice of the entering index plays a central role. Recall that our aim is to reduce the objective value, namely the inner product of  $c$  and  $x$ , while maintaining feasibility. Among all nonbasic variables, we look at their reduced costs. Only those with negative reduced costs are eligible to enter, since moving them from zero upward can potentially reduce the objective. The most straightforward approach is Dantzig's rule: simply pick the index with the most negative reduced cost. This maximizes the decrease in the objective per unit increase of that particular nonbasic variable. However, the step length for that variable may be very small, which limits the actual improvement achieved in one iteration.

Because evaluating all reduced costs can be computationally demanding, especially for large-scale problems, several variants were proposed. Partial pricing considers only a subset of the nonbasic variables at each iteration, rotating through different subsets over time. This lowers the cost of the reduced-cost computation. Multiple pricing takes a different approach: it evaluates reduced costs and candidate directions for a chosen set of variables, and reuses that information for multiple iterations until it becomes outdated. This reduces memory traffic, which is often a bottleneck on modern machines.

Finally, heuristics often combine partial and multiple pricing. They do not guarantee the mathematically best choice at each step, but they balance progress toward optimality with computational efficiency. The critical message is that, in practice, rules for entering variables are shaped not only by theoretical optimality but also by considerations of speed and scalability.

## Revised Simplex Method: Steepest-Edge Rule

Selecting entering index  $q$  using steepest-edge rule for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ .

- **Steepest-Edge Rule:** Chooses  $q \in \mathcal{N}$  for largest decrease in  $c^T x$  per unit distance along edge (most downhill), unlike Dantzig's rule, which maximizes decrease per unit  $x_q^+$ . A small  $x_q^+$  change can mean a large edge distance.
- **Pivot Step:** Change in  $x$  is

$$x^+ = \begin{bmatrix} x_B^+ \\ x_N^+ \end{bmatrix} = \begin{bmatrix} x_B \\ x_N \end{bmatrix} + \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix} x_q^+ = x + \eta_q x_q^+, \text{ where}$$

$$\eta_q = \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix} = \begin{bmatrix} -d \\ e_q \end{bmatrix}, e_q \text{ is unit vector with 1 at } q.$$

- **Decrease Measure:** Change in  $c^T x$  per unit step along  $\eta_q$  is  $\frac{c^T \eta_q}{\|\eta_q\|}$ . Choose  $q$  to minimize this.
- **Efficient Update:** Computing  $\eta_i$  for all  $i \in \mathcal{N}$  (via  $Bd = A_i$ ) is costly. Goldfarb and Reid's method updates  $c^T \eta_i$  and  $\|\eta_i\|$  economically each iteration.

24/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

**Steepest-Edge Rule**

Two-Phase Simplex



### Comments

Dantzig's rule focuses on the decrease per unit of the entering variable, but that can be misleading. Imagine a situation where the variable's allowable increase is tiny, but the reduced cost is very negative. The predicted gain is large per unit of that variable, yet the total step is negligible. The steepest-edge rule addresses this by measuring progress differently. Instead of considering a unit increase of the variable, it evaluates the decrease in the objective per unit length of movement along the actual edge of the feasible polytope. In other words, it asks: if we step in this direction, how steep is the descent of the objective function?

Formally, the update vector is written as  $\eta_q = \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix}$ . The change in the basic variables is  $-B^{-1}A_q$ , while the entering variable increases in the unit direction. Thus, the entire step is represented as  $x^+ = x + \eta_q x_q^+$ . The rate of decrease per unit distance in this direction is given by the ratio  $(c^T \eta_q) / \|\eta_q\|$ . Choosing the entering index now means selecting the variable  $q$  that minimizes this ratio.

The challenge is computational. For each candidate, one needs both the numerator and the norm of  $\eta_q$ . Computing these from scratch is costly. This is where the work of Goldfarb and Reid becomes essential: they developed efficient recursive formulas to update the required quantities after each iteration. Thanks to this, the steepest-edge rule became practical and has shown strong empirical performance, often outperforming Dantzig's rule on difficult problems.

Updating  $c^T \eta_i$  and  $\|\eta_i\|$  for steepest-edge rule, where  $\eta_i = \begin{bmatrix} -B^{-1}A_i \\ e_i \end{bmatrix}$  picks direction for index i.

- **Numerator:**  $c^T \eta_i = s_i$  (from  $c^T x^+ = c^T x + s_q x_q^+$  (was shown earlier) and  $c^T x^+ = c^T x + c^T \eta_q x_q^+$ ). No need to compute  $\eta_i$ ;  $s_i$  measures objective change rate.
- **Denominator** (we define  $\gamma_i = \|\eta_i\|^2$ ):

$$\gamma_i = \|\eta_i\|^2 = \|B^{-1}A_i\|^2 + 1 \text{ (current step);}$$

$$\gamma_i^+ = \|\eta_i^+\|^2 = \|(B^+)^{-1}A_i\|^2 + 1 \text{ (next step).}$$

Norms adjust steepness.

- **Basis Update:** Assume, e.g., that we replace column  $A_t$  (corresponds to column p in B) with  $A_q$ :

$$B^+ = B + (A_q - Be_p)e_p^T, \quad e_p = (0, \dots, 0, p, 0, \dots, 0)^T. \text{ Updates basis matrix.}$$

- **Applying Sherman-Morrison formula obtain:**  $(B^+)^{-1} = B^{-1} - \frac{(d - e_p)e_p^T B^{-1}}{e_p^T d}$ , where  $d = B^{-1}A_q$ .

$$\text{Then, } (B^+)^{-1}A_i = B^{-1}A_i - \frac{e_p^T B^{-1} A_i}{e_p^T d}(d - e_p). \text{ Adjusts inverse efficiently.}$$

25/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Sensitivity Analysis**

**Fundamental Theorem**

**SIMPLEX METHOD**

**Steepest-Edge Rule**

**Two-Phase Simplex**



### Comments

To apply the steepest-edge rule efficiently, we need a systematic way to maintain the quantities that determine the choice of entering index. Recall that the numerator in our ratio is simply the reduced cost  $s_i$ . This is convenient: it means we do not need to explicitly compute  $\eta_i$ . The reduced cost already encodes the rate of change of the objective in the direction associated with variable i.

The denominator is more challenging. We define  $\gamma_i$  as the squared norm of the step direction, i.e. as the squared norm of  $\eta_i$ . For the current basis, this is the squared norm of  $B^{-1}A_i$  plus one. After a pivot, the basis changes, and thus  $\gamma_i$  must be updated. If we had to recompute these norms from scratch every iteration, the computational cost would be prohibitive.

Here enters the Sherman–Morrison formula. When the basis matrix is updated by replacing one column with a new one, the inverse can be updated cheaply without recomputation. Specifically, if  $B^+$  is the new basis matrix, then its inverse is given by  $B^{-1}$  minus (the quantity  $d - e_p$ ) times  $e_p^T B^{-1}$ , all over  $e_p^T d$ , where  $d = B^{-1}A_q$ . This allows us to update  $(B^+)^{-1}A_i$  efficiently. Once we have this, we can update the squared norms  $\gamma_i^+$ .

This mechanism is what makes the steepest-edge method competitive. Instead of recalculating large matrix inverses, we adjust only the necessary vectors. In practice, this reduces the computational load drastically, while keeping the selection criterion accurate.

## Revised Simplex Method: Steepest-Edge Update (Cont)

Continuing update of  $\gamma_i = \|\eta_i\|^2$  for steepest-edge rule, where  $\eta_i$  defines direction for index  $i \in \mathcal{N}$ .

- **Update Formula:** Substitute  $(B^+)^{-1}A_i$  into  $\gamma_i^+ = \|(B^+)^{-1}A_i\|^2 + 1$  to get:

$$\gamma_i^+ = \gamma_i - 2 \left( \frac{e_p^T B^{-1} A_i}{e_p^T d} \right) A_i^T (B^{-1})^T d + \left( \frac{e_p^T B^{-1} A_i}{e_p^T d} \right)^2 \gamma_q, \text{ where } d = B^{-1} A_q.$$

- **Simplified Form:** Using  $r, \hat{d}$  (solving  $B^T r = e_p, B^T \hat{d} = d$ ),

$$\gamma_i^+ = \gamma_i - 2 \left( \frac{r^T A_i}{r^T A_q} \right) \hat{d}^T A_i + \left( \frac{r^T A_i}{r^T A_q} \right)^2 \gamma_q.$$

Computes new norms efficiently.

- **Computation:** Solve  $B^T \hat{d} = d, B^T r = e_p$ , then evaluate  $r^T A_i, \hat{d}^T A_i$  for  $i \in \mathcal{N}, i \neq q$ , to update all  $\gamma_i^+$ .
- **Practical Note:** Steepest-edge doesn't ensure long steps to next vertex but is highly effective in practice.

26/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

**Sensitivity Analysis**

**Fundamental Theorem**

**SIMPLEX METHOD**

**Steepest-Edge Rule**

**Two-Phase Simplex**



### Comments

Let us now complete the update for the squared norms. Substituting the expression for  $(B^+)^{-1}A_i$  into the formula for  $\gamma_i^+$ , we obtain an explicit update. The key observation is that the new norm can be expressed in terms of the old norm, together with inner products involving a small number of auxiliary vectors. Specifically, one introduces two vectors, denoted  $r$  and  $\hat{d}$ , which solve the linear systems  $B^T r = e_p$  and  $B^T \hat{d} = d$ , respectively. Using these, the update formula becomes much more compact:

$$\gamma_i^+ = \gamma_i - 2 \left( \frac{r^T A_i}{r^T A_q} \right) \hat{d}^T A_i + \left( \frac{r^T A_i}{r^T A_q} \right)^2 \gamma_q.$$

This formula highlights the efficiency of the approach: rather than recomputing norms from scratch, one solves two auxiliary systems once per iteration, and then applies inexpensive vector operations to update all  $\gamma_i$ .

From a practical perspective, this makes the steepest-edge rule computationally viable. Importantly, while the method aims to choose directions that correspond to steepest descent, it does not necessarily ensure long steps in the primal space. Nevertheless, empirical studies have consistently shown that it tends to accelerate convergence and to avoid the stalling that can occur under Dantzig's rule. In fact, many state-of-the-art simplex implementations rely on steepest-edge or variants of it as their default pricing strategy.

## Two-Phase Simplex: Finding a Starting Point

**Key Idea:** Simplex method for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$  needs a feasible starting point  $x$  with basis  $\mathcal{B}$ . Use two-phase approach to find it.

**Problem Context:** Finding initial  $x$  with basis  $\mathcal{B}$  ( $m$  indices), where basis matrix  $B$  is nonsingular and  $x_{\mathcal{B}} = B^{-1}b \geq 0$ ,  $x_{\mathcal{N}} = 0$ , is as hard as solving the linear program.

### Two-Phase Strategy:

- ▶ **Phase I:** Solve auxiliary problem to get feasible point.
- ▶ **Phase II:** Use Phase I solution to solve original problem.

**Why It Works:** The point  $(x = 0, z_j = |b_j|)$  satisfies  $Ax + Ez = b$ , and  $E$ 's diagonal structure guarantees an invertible basis for **Phase I**, enabling **Phase II** to extract the original solution.

**Phase I** designs an easy-to-start problem:

$$\min \sum z_j, \text{ s.t. } Ax + Ez = b, (x, z) \geq 0,$$

with  $z \in \mathbb{R}^m$ ,  $E$  is a diagonal matrix:

$$E_{jj} = \begin{cases} 1, & \text{if } b_j \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

**Starting Point:** Set  $x = 0, z_j = |b_j|$ .

Basis  $B = E$  is invertible, ensuring a simple feasible solution to launch **Phase I**

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

In order for the simplex method can work, it is necessary to select an initial feasible point with a valid basis. Without such a point, the algorithm cannot even begin, because every iteration of the simplex relies on moving between adjacent basic feasible solutions. The challenge is that finding this first feasible solution can be just as difficult as solving the entire problem itself. To overcome this, the two-phase strategy was developed.

The essence of Phase One is to construct an auxiliary problem that is guaranteed to have a simple starting solution. Artificial variables are introduced to absorb infeasibilities in the original system. Suppose we want to solve the system  $Ax = b$  with  $x \geq 0$ . For each constraint, we add an artificial variable  $z_j$ . Then we minimize the sum of all artificial variables. In words, the auxiliary problem becomes: minimize the sum over all  $z_j$ , subject to  $Ax + Ez = b$ , with nonnegativity on both  $x$  and  $z$ . The matrix  $E$  is chosen as a diagonal matrix where each diagonal entry is either plus one or minus one, depending on the sign of the corresponding right-hand side  $b_j$ . This construction guarantees feasibility.

The starting point is straightforward: set all  $x$  variables to zero, and assign each artificial variable  $z_j$  to be the absolute value of  $b_j$ . This ensures the equation is satisfied, because then  $Ax + Ez = b$ . At the same time, the artificial basis matrix  $B=E$  is invertible by design. Thus, Phase One always begins with a valid basis and a feasible point. Once Phase One concludes, if the artificial variables can be driven to zero, then the solution obtained will also satisfy the original system and will provide the required feasible starting point for Phase Two.

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

Phase I minimizes constraint violations to find a feasible point for the original problem  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ , solved in Phase II.

Phase I Role: For

$$\min \sum z_j, \text{ s.t. } Ax + Ez = b, (x, z) \geq 0,$$

$z$  measures violations of  $Ax = b$ .

- ▶ Minimizing  $\sum z_j$  seeks  $x$  feasible for original constraints.
- ▶  $\sum z_j = 0$  if and only if original problem is feasible (then  $\tilde{z} = 0$ ,  $A\tilde{x} = b$ ,  $\tilde{x} \geq 0$ ). If  $\sum z_j > 0$ , problem is infeasible.

Start simplex at  $x = 0$ ,  $z_j = |b_j|$ .

Bounded below by 0, so terminates.

Phase II Role: If  $\sum \tilde{z}_j = 0$ , use  $(\tilde{x}, \tilde{z})$  to solve:

$$\min c^T x, \text{ s.t. } Ax + z = b, x \geq 0, 0 \geq z \geq 0.$$

- ▶ Matches original problem, as  $z = 0$  for feasibility.
- ▶ Retains  $z$  in basis from Phase I (with  $\tilde{z}_j = 0$ ).

Solution gives original  $x$ .

28/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches



### Comments

The crucial purpose of Phase One is to determine whether the original linear program is feasible. The artificial problem introduces slack in the form of variables  $z_j$  that capture violations of the equality constraints. If we denote the auxiliary objective as minimizing the sum of these violations, the interpretation becomes clear: if the minimum sum equals zero, then we have found an exact feasible point for the original system. If the minimum remains strictly positive, then no feasible solution exists.

To start Phase One, the simplex algorithm uses the artificially constructed feasible solution, where all original variables are set to zero and the artificial variables equal the absolute values of the right-hand sides. Since the objective is nonnegative and bounded below by zero, the procedure is guaranteed to terminate.

When the outcome of Phase One yields a zero objective value, the artificial variables vanish, and the corresponding vector of original variables satisfies  $Ax = b$  with  $x \geq 0$ . At that point, Phase Two can begin. In Phase Two, the artificial variables are formally kept in the system, but they are forced to equal zero. Their role is essentially transitional: they remain in the basis initially, ensuring continuity of the simplex process, but they do not affect the optimization since their coefficients vanish.

If, instead, the Phase One optimization concludes with a strictly positive objective value, then it is impossible to satisfy the original equalities with nonnegative variables. In that case, we must declare the original problem infeasible, and no further optimization is possible. Thus, Phase One not only generates an initial feasible basis when possible, but it also acts as a test of feasibility for the entire model.

Finalizing Phase II for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ , and optimizing Phase I.

- **Phase II Adjustments:** For  $\min c^T x$ , s.t.  $Ax + z = b$ ,  $x \geq 0$ ,  $0 \geq z \geq 0$ , modify simplex to handle  $z$ 's two-sided bounds. Remove  $z_j$  from problem once it leaves basis to avoid redundant iterations.
- **Solution Properties:** Basic solution  $(x^*, z^*)$  has  $z^* = 0$ , so  $x^*$  solves original problem. But basis  $\mathcal{B}$  may include  $z^*$ -components, not ideal for original problem.

**Postprocessing:** Extract  $z$ -components from  $\mathcal{B}$ , replace with nonbasic  $x$ -components to keep basis matrix invertible, yielding optimal basis for original problem.

- **Phase I Optimization:** No need for  $m$  artificial variables if slack/surplus variables (from inequality constraints) can act as artificial ones, reducing complexity.

29/29 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex



### Comments

Once Phase One provides a feasible point for the original system, the algorithm transitions into Phase Two, where the true objective function is optimized. The formulation at this stage can include artificial variables constrained between zero and zero, but these variables are superfluous. As soon as any artificial variable leaves the basis, it is eliminated from further consideration, since it no longer plays any role in the optimization.

The key property of the Phase Two solution is that the final basic feasible solution will have all artificial variables equal to zero. The remaining vector of decision variables then constitutes a feasible solution to the original problem. However, the basis at termination may still formally include some artificial variables. In such cases, a postprocessing step is necessary: one replaces artificial components in the basis with genuine decision variables that are nonbasic but can enter without losing invertibility. This yields a proper basis for the original problem, ensuring that the final tableau fully corresponds to the actual decision space.

Another practical refinement concerns the construction of Phase One itself. In many problems that originate with inequality constraints, slack or surplus variables already exist. These can often serve as artificial variables, reducing the number of additional variables that must be introduced. This significantly lowers the size of the auxiliary problem and simplifies computations. Thus, the two-phase method is not only theoretically sound but also efficient in practice, as it exploits the problem's structure to minimize unnecessary complexity.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 9)

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution

Shpilev Petr Valerievich

Faculty of Mathematics and Mechanics, SPbU

September, 2025



Санкт-Петербургский  
государственный  
университет



30 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In this lecture, we extend our study of linear and quadratic programming algorithms. We begin with the two-phase simplex method, discussing challenges such as degeneracy, cycling, and perturbation strategies to ensure progress. We then introduce the dual simplex method, exploring its step-by-step procedure and illustrating its application through examples. The lecture also covers presolving techniques in linear programming, including detection of redundant or dominated constraints and problem simplification. Building on this foundation, we transition to quadratic programming, with portfolio optimization as a motivating example. We analyze equality-constrained QPs, properties of the KKT matrix such as nonsingularity and inertia, and solution methods including Schur-complement and null-space approaches. Finally, we discuss iterative methods for large-scale problems, focusing on preconditioned conjugate gradients for reduced systems and the projected CG method.

### Example: Inequality-Constrained Linear Program

Consider:  $\min 3x_1 + x_2 + x_3$ , s.t.  $2x_1 + x_2 + x_3 \leq 2$ ,  $x_1 - x_2 - x_3 \leq -1$ ,  $x \geq 0$ .

Convert to standard form and set up **Phase I** to find a feasible starting point.

**Standard Form:** Add slack variables  $x_4$ ,  $x_5$ :

$$\min 3x_1 + x_2 + x_3, \text{ s.t.}$$

$$2x_1 + x_2 + x_3 + x_4 = 2,$$

$$x_1 - x_2 - x_3 + x_5 = -1,$$

$$x \geq 0.$$

Point  $x = (0, 0, 0, 2, 0)$  is feasible for first constraint, not second.

**Phase I Setup:** Add one artificial variable  $z_2$  to second constraint:

$$\min z_2, \text{ s.t.}$$

$$2x_1 + x_2 + x_3 + x_4 = 2,$$

$$x_1 - x_2 - x_3 + x_5 - z_2 = -1,$$

$$(x, z_2) \geq 0.$$

Feasible point:  $(x, z_2) = ((0, 0, 0, 2, 0), 1)$ ,

with basis matrix  $B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

(invertible).

**Note:**  $B$  is formed from columns of  $x_4$  and  $z_2$  (basic variables);  $x_4$  acts as artificial variable for first constraint, so no  $z_1$  needed.

1/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches



### Comments

To illustrate the mechanics of the two-phase simplex method, consider a small linear program with inequalities. The objective is to minimize  $3x_1 + x_2 + x_3$ , subject to two constraints: first,  $2x_1 + x_2 + x_3 \leq 2$ ; second,  $x_1 - x_2 - x_3 \leq -1$ . All variables are required to be nonnegative.

The first step is to rewrite the inequalities in standard form by introducing slack variables. Adding slack variable  $x_4$  to the first inequality converts it into an equality. Adding slack variable  $x_5$  to the second inequality does the same. However, the second equality is problematic: it reads  $x_1 - x_2 - x_3 + x_5 = -1$ . This right-hand side is negative, which immediately makes the naive choice of slacks infeasible. To resolve this, an artificial variable  $z_2$  is added to absorb the infeasibility. The modified system then includes  $z_2$  in the second constraint.

The auxiliary Phase One objective is simply to minimize  $z_2$ . The starting feasible point assigns all decision variables to zero, sets  $x_4 = 2$ , and sets  $z_2 = 1$ . At this point, the system is consistent, and the basis matrix  $B$  is invertible, constructed from the columns corresponding to  $x_4$  and  $z_2$ . This demonstrates how slack variables sometimes automatically act as artificial variables, and how artificial variables are introduced only where truly necessary. The procedure guarantees a feasible starting point for Phase One, after which the artificial objective will be minimized to check whether feasibility of the original problem is achievable.

Challenges in simplex method when updating basis for  
 $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ .

## Degenerate Steps:

- ▶ Occur when entering variable  $x_q$  cannot increase without violating  $x \geq 0$ , if some basic variable  $x_i = 0$  and  $d_i > 0$  ( $d = B^{-1}A_q$ ).
- ▶ No change in  $x$  or objective  $c^T x$ , but basis changes, potentially nearing optimal basis.

## Cycling Issue:

- ▶ Repeated degenerate steps may return to a previous basis, causing infinite loop.
- ▶ Common in large integer program relaxations; requires avoidance strategies.

## Perturbation Strategy:

- ▶ For degenerate basis  $\hat{B}$  with matrix  $\hat{B}$ , modify constraints:

$$b(\epsilon) = b + \hat{B} \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^m \end{bmatrix}, \quad \epsilon > 0 \text{ small.}$$

- ▶ A very small positive number  $\epsilon$  shifts right-hand side to avoid degenerate cycling.



2/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

One of the subtle difficulties in the simplex method arises from the phenomenon of degeneracy. Degeneracy occurs when, during a pivot operation, the entering variable cannot increase because one or more basic variables are already equal to zero. In this case, although the basis changes, the vector of decision variables and the value of the objective function remain unchanged. In words, the algorithm performs a step that does not advance the solution in the objective space.

While this may sound harmless, repeated degenerate steps create the risk of cycling. Cycling refers to the possibility that the simplex algorithm revisits the same basis multiple times, which can theoretically lead to an infinite loop without ever reaching an optimal solution.

To mitigate this issue, a perturbation strategy is introduced. The idea is to slightly shift the right-hand side vector of the system of equations in a carefully controlled manner. If the current degenerate basis is denoted by  $\hat{B}$ , with basis matrix  $\hat{B}$ , then we define a perturbed right-hand side as vector  $b(\epsilon)$  equals

$$b + \hat{B} \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^m \end{bmatrix}.$$

Here  $\epsilon$  is a small positive scalar. This construction slightly perturbs the problem so that every basic variable is strictly positive, avoiding degeneracy. Although  $\epsilon$  is taken arbitrarily small, the modification is sufficient to guarantee that each basis step produces an actual improvement or movement, thereby preventing endless cycling.

Intuitively, the perturbation acts like nudging the system away from exact degeneracy, ensuring that pivots lead to meaningful changes. Importantly, once the algorithm terminates under the perturbed system, the solution can be mapped back to the original problem, preserving correctness while ensuring finite termination.

Prevent cycling in degenerate simplex steps for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ .

## Perturbation Effect:

- Perturbed constraints

$$Ax = b + \hat{B} \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^m \end{bmatrix} \text{ shift basic solution:}$$

$$x_B(\epsilon) = x_B + \sum_{k=1}^m (B^{-1}\hat{B})_{:k} \epsilon^k,$$

$(B^{-1}\hat{B})_{:k}$  - the kth column of  $B^{-1}\hat{B}$  and  $x_B$  - the basic solution.

Ensures  $x_B(\epsilon)_i > 0$  for all  $\epsilon$  sufficiently small, making bases nondegenerate (all basic variables positive).

## Nondegeneracy and Termination:

- Nondegeneracy proven: If  $x_B(\epsilon)_i = 0$ , then  $(x_B)_i = 0$  and i-th row of  $B^{-1}\hat{B} = 0$ , impossible since  $B$ ,  $\hat{B}$  invertible.
- Prevents revisiting bases, ensures finite termination; reset  $x_B = B^{-1}b$  for original solution.

## Lexicographic Strategy:

- Tracks coefficients of  $\epsilon, \epsilon^2, \dots, \epsilon^m$  in  $x_B(\epsilon)$ , selects leaving variable by lexicographically minimizing  $x_B(\epsilon)_i/d_i$ , updates pivot to maintain  $\epsilon$ -dependence.

3/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



## Comments

The perturbation strategy becomes more precise when we analyze its algebraic effect. Suppose the current basis matrix is  $B$  and the degenerate basis we perturb is  $\hat{B}$ . Then the new right-hand side is vector  $b$  plus matrix  $\hat{B}$  times the vector  $\epsilon, \epsilon^2, \dots, \epsilon^m$ , and so on.

Solving for the basic variables gives the perturbed basic solution. Concretely, the basic solution  $x_B(\epsilon)$  equals the unperturbed solution  $x_B$  plus the sum from  $k = 1$  to  $m$  of the k-th column of the product  $B^{-1}\hat{B}$ , multiplied by  $\epsilon^k$ . This expansion shows that each basic variable is shifted slightly upward by a positive power of  $\epsilon$ .

The crucial property is that every basic variable becomes strictly positive for sufficiently small  $\epsilon$ . To see this, note that if a perturbed variable were zero, then both its unperturbed value and all coefficients multiplying  $\epsilon$  terms would have to vanish. But since  $B$  and  $\hat{B}$  are invertible matrices, such a simultaneous vanishing is impossible. This argument establishes nondegeneracy rigorously.

Once nondegeneracy is guaranteed, the simplex algorithm can no longer revisit the same basis, and termination in finitely many steps is assured. After finishing under the perturbed system, we simply set  $\epsilon$  back to zero, recovering the original solution.

In practice, instead of explicitly adding symbolic perturbations, the algorithm can adopt a lexicographic pivoting rule. This rule compares ratios in a lexicographic manner by tracking the coefficients of  $\epsilon, \epsilon^2, \dots, \epsilon^m$  in the expansion of the basic variables. The leaving variable is chosen by minimizing the ratio in lexicographic order, ensuring consistency. This approach emulates perturbation implicitly, without requiring explicit small numbers, and is the standard way to enforce progress in degenerate situations.

The dual simplex method is a faster variant (in many cases) for  $\min c^T x$ , s.t.  $Ax = b$ ,  $x \geq 0$ , using basis updates like primal simplex.

## Overview:

- ▶ Starts with dual feasible point  $(\lambda, s)$ :  $s_B = 0$ ,  $s_N \geq 0$ , but primal  $x_B = B^{-1}b$  may have negative components.
- ▶ Updates basis  $B, N$  to reach primal feasibility ( $x_B \geq 0$ ), achieving optimality.
- ▶ Unlike primal,  $B$  is nonsingular but not always a basis matrix (may not satisfy  $x_B = B^{-1}b \geq 0$ ).

## Single Step:

- ▶ Define variables:

$$x_B = B^{-1}b, \quad x_N = 0, \quad \lambda = B^{-T}c_B, \quad s_B = 0, \quad s_N = c_N - N^T\lambda \geq 0.$$

- ▶ If  $x_B \geq 0$ , the current point  $(x, \lambda, s)$  satisfies the optimality (KKT) conditions, and we are done.

4/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



## Comments

The dual simplex method provides an alternative pivoting scheme that often proves faster than the primal simplex in applications.

The primal simplex begins with a feasible primal solution and maintains feasibility while driving the objective function toward optimality.

The dual simplex takes the opposite perspective: it begins from a dual feasible solution and then works toward primal feasibility, all while maintaining dual feasibility. Ultimately, the method converges to a solution that satisfies both primal and dual conditions, which is equivalent to optimality.

To be more precise, suppose we partition the variables into basic and nonbasic sets corresponding to matrices  $B$  and  $N$ . Then we can write the primal and dual variables explicitly.

The primal basic variables equal  $B^{-1}b$ . The primal nonbasic variables are zero. The dual vector  $\lambda$  is defined as  $B^{-T}c_B$ . The dual slacks are zero for the basic set, and equal the cost vector for nonbasic variables minus the transpose of  $N$  times  $\lambda$ .

If the primal basic variables are nonnegative, then the point defined by  $x, \lambda$ , and  $s$  satisfies all Karush-Kuhn-Tucker conditions, meaning we have reached an optimal solution. The distinguishing feature of the dual simplex method is that it can start from a situation where primal feasibility is violated—some basic variables are negative—but dual feasibility still holds.

The algorithm then corrects these violations step by step, making the basic variables nonnegative, while preserving the dual conditions.

This makes the dual simplex particularly efficient in reoptimization problems, such as when new constraints are added to a model or in large-scale branch-and-bound procedures for integer programming.



## Index Selection:

- ▶ Select a leaving index  $q \in \mathcal{B}$  such that  $x_q < 0$  to move  $x_q$  to zero (ensuring nonnegativity);
- ▶ Identify an entering index  $r \in \mathcal{N}$  such that  $s_r$  becomes zero, while  $x_r$  increases from zero, moving  $q$  from  $\mathcal{B}$  to  $\mathcal{N}$ ,  $r$  from  $\mathcal{N}$  to  $\mathcal{B}$ .

## Dual Variable Updates:

- ▶ Update  $s_{\mathcal{B}}^+$ :  $s_{\mathcal{B}}^+ = s_{\mathcal{B}} + \alpha e_q$ , where  $e_q$  is a vector of length  $m$  with 1 at the position of  $q$  in  $\mathcal{B}$ , zeros elsewhere, for some positive scalar  $\alpha$  to be determined update  $\lambda^+$  and  $s_{\mathcal{B}}^+$  for some vector  $v$ :

$$s_{\mathcal{B}}^+ = s_{\mathcal{B}} + \alpha e_q, \quad \lambda^+ = \lambda + \alpha v.$$

## Deriving Update Vector:

- ▶ Since  $s_{\mathcal{B}}^+ = c_{\mathcal{B}} - B^T \lambda^+$  must hold, we derive the system of equations that we have to solve to obtain  $v$ :

$$\begin{aligned} s_{\mathcal{B}} + \alpha e_q &= c_{\mathcal{B}} - B^T(\lambda + \alpha v) \implies \alpha e_q = -B^T \alpha v \implies \\ e_q &= -B^T v \implies v = -B^{-T} e_q \end{aligned}$$

5/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Let us now examine the mechanics of a single pivot step in the dual simplex method. The first task is to identify a leaving variable among the basic set. Since primal feasibility is violated, there must exist a basic variable with a negative value. Denote its index as  $q$ .

This variable will be forced to zero in the next step, which moves it out of the basis. The second task is to identify an entering variable from the nonbasic set. Denote its index as  $r$ .

The entering variable increases from zero while its corresponding dual slack decreases to zero. In this way, variables exchange roles between the basis and the nonbasis.

To maintain dual feasibility, the update of dual variables must be carefully orchestrated. We adjust the dual slacks for the basic set by adding  $\alpha$  times the unit vector  $e_q$ , and simultaneously update the dual vector  $\lambda$  by adding  $\alpha$  times another vector  $v$ . The scalar  $\alpha$  is chosen so that one nonbasic slack becomes zero exactly at the right moment, enabling variable  $r$  to enter the basis.

The structure of the update ensures consistency with complementary slackness. To determine the vector  $v$ , we impose the defining relation between slacks and costs. The updated slacks for the basis must equal the cost vector of the basis minus the transpose of matrix  $B$  times the updated  $\lambda$ .

Substituting the proposed form yields an equation that simplifies to  $e_q = -B^T v$ . Solving this system gives  $v = -B^{-T} e_q$ . This explicit formula for  $v$  completes the update rule. Thus, each pivot step in the dual simplex adjusts both primal and dual variables in tandem, steadily reducing infeasibility and progressing toward optimality.



## Dual Objective Change:

- Using  $\lambda^+ = \lambda + \alpha v$ ,  $v = -B^{-T}e_q$ ,  $x_B = B^{-1}b$ , and  $x_q = x_B^T e_q$ , derive the dual objective change:

$$b^T \lambda^+ = b^T \lambda + \alpha b^T v = b^T \lambda - \alpha b^T B^{-T} e_q = b^T \lambda - \alpha x_B^T e_q = b^T \lambda - \alpha x_q.$$

## Maximizing the Dual Objective:

- Since  $x_q < 0$ , choose  $\alpha$  as large as possible to maximize the dual objective  $b^T \lambda$ , subject to the constraint  $s_N^+ \geq 0$ .

## Computing $\alpha$

- Update  $s_N^+ = c_N - N^T \lambda^+ = s_N - \alpha w$ , where  $w = N^T v = -N^T B^{-T} e_q$ . The largest  $\alpha$  for which  $s_N^+ \geq 0$  is:

$$\alpha = \min_{j \in N, w_j > 0} \frac{s_j}{w_j}.$$

## Comments

When analyzing the dual simplex method at this stage, the focus is on how the dual objective changes as the iteration progresses. The dual variable is updated according to the formula  $\lambda^+ = \lambda + \alpha v$ , with  $v = -B^{-T}e_q$ .

Using the relationship  $x_B = B^{-1}b$  and  $x_q = x_B^T e_q$ , we can derive how the dual objective  $b^T \lambda$  changes. Substituting, we obtain that  $b^T \lambda^+ = b^T \lambda - \alpha x_q$ .

This simple but crucial identity tells us that the effect of increasing  $\alpha$  is directly tied to the current value of  $x_q$ . Now, since  $x_q$  is negative, subtracting  $\alpha$  times  $x_q$  actually increases the dual objective value. In other words, the negative component in the primal basic solution drives improvement in the dual objective. Therefore, the strategy is to choose  $\alpha$  as large as possible while still respecting feasibility in the dual slack variables.

This feasibility constraint appears through the update  $s_N^+ = s_N - \alpha w$ , where  $w = -N^T B^{-T} e_q$ . To preserve nonnegativity of the dual slack variables, we must bound  $\alpha$  so that  $s_N^+$  remains nonnegative. This gives the condition that  $\alpha$  is less than or equal to the minimum ratio of  $\frac{s_j}{w_j}$  across all nonbasic indices  $j$  with  $w_j > 0$ .

The minimum ratio rule here mirrors the logic of the standard simplex method but in the dual setting: it guarantees that one of the dual constraints becomes tight exactly at the point where  $\alpha$  reaches its maximum permissible value.

Thus, the step simultaneously improves the dual objective while preserving feasibility, establishing the foundation for moving towards optimality.



## Entering Index:

- ▶ Define the entering index  $r$  as the index at which the minimum in  $\alpha = \min_{j \in N, w_j > 0} \frac{s_j}{w_j}$  is achieved, so:

$$s_r^+ = 0 \quad \text{and} \quad w_r = A_r^T v > 0,$$

- ▶ where  $A_r$  is the  $r$ -th column of  $A$ .

## Updating Primal Variables:

- ▶ Set  $x_q^+ = 0$  for the leaving index  $q$ , allow  $x_r^+$  nonzero for the entering index  $r$ . Define the direction  $d$  for  $x_B$ :

$$Bd = \sum_{i \in B} A_i d_i = A_r.$$

$$\sum_{i \in B} A_i x_i = b \implies \sum_{i \in B} A_i (x_i - \gamma d_i) + A_r \gamma = b, \quad \gamma = \frac{x_q}{d_q}.$$

## Deriving Direction:

- ▶ Since  $d_q$  must be nonzero, we have  $d_q < 0$ , since:

$$d_q = d^T e_q = A_r^T B^{-T} e_q = -A_r^T v = -w_r < 0,$$

using  $Bd = A_r$ ,  $e_q = -B^T v$ , and  $w_r = A_r^T v > 0$ .

7/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

At this stage, we determine the entering index in the dual simplex iteration. The entering index  $r$  is defined as the position at which the minimum ratio  $\frac{s_j}{w_j}$  is achieved among all nonbasic indices with  $w_j > 0$ .

By this choice,  $s_r^+$  becomes zero, and the corresponding  $w_r$  is strictly positive. This identifies the variable that should enter the basis. Importantly, the column  $A_r$  associated with this index provides the structure needed to compute the direction of movement in the primal space.

Once the entering variable is identified, we must update the primal solution to reflect the change in basis. The leaving variable  $x_q$  is set to zero, while  $x_r$  becomes nonzero. To maintain primal feasibility, the basic variables adjust along a direction vector  $d$ , defined by the equation  $Bd = A_r$ .

This equation ensures that the linear constraints remain satisfied after the exchange of basis elements. Next, to maintain consistency, the adjustment is parameterized by  $\gamma$ , defined as  $\frac{x_q}{d_q}$ .

This ratio captures how far we can move in the computed direction before the leaving variable becomes zero, which matches the rule that enforces primal feasibility. At the same time, we recognize that  $d_q$  cannot be zero, since otherwise the exchange would not reduce infeasibility.

Further analysis shows that  $d_q$  must in fact be negative. Specifically,  $d_q = -w_r$ , which is less than zero because  $w_r$  is positive. This condition confirms that  $\gamma$  is positive, ensuring that the update preserves feasibility and progresses towards optimality. Thus, the entering and leaving indices together guide the iteration toward an improved basis.



## Updating Primal Variables:

- ▶ Since  $x_q < 0$ , it follows from  $\gamma = \frac{x_q}{d_q}$  that  $\gamma > 0$ . Following  $\sum_{i \in \mathcal{B}} A_i(x_i - \gamma d_i) + A_r \gamma = b$ , define the updated vector  $x^+$ :

$$x_i^+ = \begin{cases} x_i - \gamma d_i, & \text{for } i \in \mathcal{B} \text{ with } i \neq q, \\ 0, & \text{for } i = q, \\ 0, & \text{for } i \in \mathcal{N} \text{ with } i \neq r, \\ \gamma, & \text{for } i = r. \end{cases}$$

## Example: Dual Simplex Method

Consider the problem:

$$\min -4x_1 - 2x_2, \text{ s.t. } x_1 + x_2 + x_3 = 5, \quad 2x_1 + \frac{1}{2}x_2 + x_4 = 8, \quad x \geq 0.$$

- ▶ Basis:  $\mathcal{B} = \{2, 3\}$ ,  $\mathcal{N} = \{1, 4\}$ . Compute  $x_{\mathcal{B}} = B^{-1}b$ ,  $B = \begin{bmatrix} 1 & 1 \\ \frac{1}{2} & 0 \end{bmatrix}$ :

$$x_{\mathcal{B}} = \begin{bmatrix} 0 & 2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \end{bmatrix} = \begin{bmatrix} 16 \\ -11 \end{bmatrix}, \quad x_2 = 16, \quad x_3 = -11, \quad x_1 = x_4 = 0.$$

8/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The update of the primal variables now becomes explicit. Since  $x_q$  is negative and  $d_q$  is also negative, then, as we mentioned earlier, their ratio  $\gamma = \frac{x_q}{d_q}$  is strictly positive.

This scalar determines the amount by which the new entering variable increases. The updated primal solution  $x^+$  is then defined piecewise: each remaining basic variable is adjusted by subtracting  $\gamma$  times the corresponding component of  $d$ , the leaving variable  $x_q$  is set to zero, all other nonbasic variables remain zero except for the new entering variable  $x_r$ , which is assigned the value  $\gamma$ .

This update guarantees that the new solution continues to satisfy all primal constraints while eliminating the infeasibility caused by the negative basic variable. The result is a feasible basis, at least with respect to the variable that previously violated nonnegativity.

Importantly, the update preserves the balance between primal and dual feasibility conditions, ensuring consistency of the simplex method framework.

To illustrate the procedure concretely, consider the example problem where the objective is to minimize  $-4x_1 - 2x_2$ , subject to two equality constraints with nonnegativity conditions. The chosen basis consists of variables 2 and 3, while variables 1 and 4 are nonbasic. By computing the basic solution  $x_{\mathcal{B}} = B^{-1}b$ , we obtain values sixteen and negative eleven for  $x_2$  and  $x_3$ , respectively.

The negative value of  $x_3$  signals the necessity of applying the dual simplex method. This example provides a concrete demonstration of the abstract derivations, showing how a negative component in the primal solution triggers a corrective iteration that not only restores feasibility but also improves the dual objective simultaneously.



- Since  $x_3 < 0$ , choose  $q = 2$  (corresponding to the second component). Dual:

Solve  $B^T \lambda = c_B = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$ , get  $\lambda = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$ . Compute  $s_N = c_N - N^T \lambda$ :

$$s_N = \begin{bmatrix} -4 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -4 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}.$$

$$v = -B^{-T} e_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \quad w = N^T v = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad \alpha = \min \left\{ \frac{4}{3}, \frac{4}{2} \right\} = 4/3, \quad r = 1.$$

- Compute direction: Solve  $Bd = A_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ :

$$d = \begin{bmatrix} 0 & 2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \end{bmatrix}, \quad \gamma = \frac{x_q}{d_q} = \frac{-11}{-3} = \frac{11}{3}.$$

- Update:  $x_2^+ = 16 - \frac{11}{3} \cdot 4 = \frac{4}{3}$ ,  $x_3^+ = 0$ ,  $x_1^+ = \frac{11}{3}$ ,  $x_4^+ = 0$ . New basis:  $\mathcal{B} = \{2, 1\}$ . And since all  $x_i > 0$  the problem is solved with  $z = -\frac{52}{3}$ .

9/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Continuing with the example, the negative value of  $x_3$  identifies the leaving index  $q$ . Solving the dual system  $B^T \lambda = c_B$  yields the dual variable values, which in this case are zero and negative four.

Using these dual multipliers, we compute the reduced costs for the nonbasic variables as  $s_N = c_N - N^T \lambda$ , which gives positive values. This confirms dual feasibility while the primal is still infeasible. Next, the search direction is determined.

The vector  $v$  is computed as  $-B^{-T} e_2$ , resulting in entries negative one and two. From this, the vector  $w = N^T v$  is obtained as entries three and two. The maximum permissible step length  $\alpha$  is the minimum of  $\frac{s_j}{w_j}$  for  $w_j > 0$ , giving four thirds in this case.

The corresponding entering index  $r$  is one, associated with the first variable. With this choice, we then compute the direction  $d$  by solving  $Bd = A_1$ , leading to the vector four, negative three.

Using the ratio test,  $\gamma$  equals negative eleven divided by negative three equals eleven thirds.

The update then produces new values for the primal variables:  $x_2$  becomes four thirds,  $x_3$  becomes zero,  $x_1$  becomes eleven thirds, and  $x_4$  remains zero. The new basis is thus formed by variables two and one.

Since all variables are now nonnegative, feasibility is restored, and the problem is solved with the optimal objective value negative fifty-two thirds.

This sequence of calculations demonstrates how the dual simplex method systematically removes infeasibilities while moving towards optimality in a finite number of steps.

Presolving (preprocessing) reduces the size of an LP problem before passing it to the solver. It's used in both simplex and interior-point methods.

**LP Formulation for Discussion:**  $\min c^T x$ , subject to  $Ax = b$ ,  $l \leq x \leq u$ .

Some components  $l_i (u_i)$  of the lower(upper) bound vector may be  $-\infty$  ( $\infty$ ).

## Key Presolving Techniques:

### Row Singleton:

- ▶ If constraint  $k$  involves only variable  $j$  ( $A_{kj} \neq 0$ ,  $A_{ki} = 0$  for  $i \neq j$ ).
- ▶ Set  $x_j = b_k / A_{kj}$  and eliminate  $x_j$ .
- ▶ If  $x_j$  violates bounds ( $x_j < l_j$  or  $x_j > u_j$ ), the problem is **infeasible**.

### Free Column Singleton:

- ▶ Variable  $x_j$  appears in only one equality constraint  $k$  ( $A_{kj} \neq 0$ ,  $A_{lj} = 0$  for all  $l \neq k$ ) and is free ( $l_j = -\infty$ ,  $u_j = +\infty$ ).
- ▶ Use constraint  $k$  to eliminate  $x_j$ :  $x_j = \frac{b_k - \sum_{p \neq j} A_{kp} x_p}{A_{kj}}$ .
- ▶ Update cost vector:  $c_p \leftarrow c_p - c_j A_{kp} / A_{kj}$  for all  $p \neq j$ .
- ▶ Dual variable for constraint  $k$ :  $\sum_{i=1}^m A_{ij} \lambda_i = c_j \Rightarrow \lambda_k = c_j / A_{kj}$ .

10/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



## Comments

In large-scale linear programming, presolving plays a crucial role in reducing the complexity of the problem before it is passed to the main solver. The central idea is to exploit the structure of the model, eliminate redundancies, and simplify constraints in ways that preserve equivalence of the feasible set and the optimal solution. Consider the general form of a linear program: minimize  $c^T x$ , subject to  $Ax = b$ , with bounds  $l \leq x \leq u$ . Here, the bound vectors  $l$  and  $u$  may contain infinite components, reflecting variables that are unbounded in one direction.

One fundamental presolving rule is the row singleton case. If a constraint involves only a single variable, then that variable is uniquely determined: we set  $x_j = b_k$  divided by  $A_{kj}$ . Once substituted, the variable and the associated row can be removed from the system. If this substitution forces  $x_j$  outside its lower or upper bounds, then the entire problem is immediately infeasible. This simple detection can save the solver from wasting effort on an impossible problem.

Another powerful rule is the free column singleton. Suppose a variable appears in only one equation and has no finite bounds, meaning  $l_j = -\infty$  and  $u_j = +\infty$ . In this case, the variable can be eliminated directly by rewriting the equation to solve for  $x_j$  in terms of the remaining variables. This elimination also requires updating the objective function coefficients, ensuring equivalence in the reduced model. Moreover, the dual variable corresponding to the eliminated constraint can be derived explicitly as  $\lambda_k = c_j$  divided by  $A_{kj}$ .

Through such transformations, presolving reduces problem size, accelerates convergence, and often determines feasibility early. In practice, these seemingly small reductions accumulate to substantial efficiency gains in large optimization tasks.



## Zero Rows and Columns in A:

### Zero Row ( $A_{ki} = 0$ for all i):

- ▶ If  $b_k = 0$ , delete the row.  $\lambda_k$  can be arbitrary.
- ▶ If  $b_k \neq 0$ , the problem is **infeasible**.

### Zero Column ( $A_{kj} = 0$ for all k):

- ▶ Determine  $x_j$  based on  $c_j$  and bounds  $[l_j, u_j]$ :
  - ▶ If  $c_j < 0$ : set  $x_j = u_j$ . If  $u_j = +\infty$ , problem is **unbounded**.
  - ▶ If  $c_j > 0$ : set  $x_j = l_j$ . If  $l_j = -\infty$ , problem is **unbounded**.
  - ▶ If  $c_j = 0$ :  $x_j$  can be any value in  $[l_j, u_j]$ .

**Forcing/Dominated Constraints:** Some constraints can only be satisfied if certain variables are at their bounds. **Example:** Consider  $5x_1 - x_4 + 2x_5 = 10$  with bounds:

$$0 \leq x_1 \leq 1, \quad -1 \leq x_4 \leq 5, \quad 0 \leq x_5 \leq 2.$$

To satisfy the equality, we must have:

- ▶  $x_1 = 1$  (to maximize  $5x_1$ )
- ▶  $x_4 = -1$  (to minimize  $-x_4$ , effectively maximizing it as  $+1$ )
- ▶  $x_5 = 2$  (to maximize  $2x_5$ )

Thus, set  $x_1 = 1, x_4 = -1, x_5 = 2$ , and eliminate these variables and the constraint.

## Comments

Beyond singletons, presolving also addresses cases where entire rows or columns in the coefficient matrix vanish. A zero row corresponds to an equation where all coefficients are zero. If the right-hand side  $b_k$  is also zero, then the constraint is redundant and can be deleted without consequence. However, if  $b_k$  is nonzero, then no assignment of variables can satisfy the equality, and the problem is infeasible. This quick detection prevents unnecessary processing of inconsistent models.

A zero column indicates a variable that never appears in any constraint. In this case, the variable influences only the objective function through its cost coefficient  $c_j$  and its bounds. If  $c_j$  is negative, the optimal strategy is to maximize the variable, pushing it to its upper bound  $u_j$ . If that bound is infinite, the objective becomes unbounded, and the problem has no finite solution. Conversely, if  $c_j$  is positive, the optimizer drives the variable to its lower bound  $l_j$ , unless that bound is  $-\infty$ , again leading to unboundedness. Finally, if  $c_j$  is zero, the variable is irrelevant to the objective and can take any feasible value within its bounds.

Another critical category consists of forcing or dominated constraints, where the structure of the equality combined with variable bounds forces specific variables to their extremal values. For instance, an equation such as  $5x_1 - x_4 + 2x_5 = 10$  with restricted ranges can only be satisfied if  $x_1 = 1, x_4 = -1$ , and  $x_5 = 2$ . Thus, the constraint forces variables to particular values, enabling their elimination.

By exploiting such structural properties, presolving strips away redundancy, sharpens problem formulation, and simplifies the path to solution.



## 1. Dominated Constraints:

- ▶ Implicitly tightened bounds for a variable due to other constraints.
- ▶ Example: Consider  $2x_2 + x_6 - 3x_7 = 8$  with bounds:

$$-10 \leq x_2 \leq 10, \quad 0 \leq x_6 \leq 1, \quad 0 \leq x_7 \leq 2.$$

- ▶ Rearranging and using bounds on  $x_6, x_7$ :

$$\begin{aligned} x_2 &= 4 - (1/2)x_6 + (3/2)x_7 \\ \Rightarrow \quad 4 - (1/2)(1) + (3/2)(0) &\leq x_2 \leq 4 - (1/2)(0) + (3/2)(2) \\ \Rightarrow \quad 7/2 &\leq x_2 \leq 7. \end{aligned}$$

- ▶ Original bounds  $[-10, 10]$  on  $x_2$  are redundant;  $x_2$  is implicitly confined to  $[7/2, 7]$ .
- ▶ We can drop the original bounds and treat  $x_2$  as a free variable (within its new implicit bounds).

12/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Presolving also leverages the concept of dominated constraints, which serve to tighten variable bounds implicitly. This process refines feasible regions without altering the actual problem. Consider the equality  $2x_2 + x_6 - 3x_7 = 8$ . With the original bounds  $-10 \leq x_2 \leq 10$ ,  $0 \leq x_6 \leq 1$ , and  $0 \leq x_7 \leq 2$ , one might initially think that  $x_2$  can vary widely. However, rearranging the equation yields  $x_2 = 4 - \frac{1}{2}x_6 + \frac{3}{2}x_7$ . Substituting the bounds of  $x_6$  and  $x_7$  produces implicit limits:  $\frac{7}{2} \leq x_2 \leq 7$ . These are far narrower than the original interval, rendering the earlier bounds redundant.

This example illustrates how presolving captures hidden relationships between variables. Instead of treating the constraints and bounds as separate entities, presolving integrates them to uncover more precise restrictions. The benefit is twofold: solvers work with tighter feasible regions, reducing exploration of infeasible areas, and redundant inequalities are discarded, simplifying the optimization.

Such reasoning is particularly impactful in large-scale systems, where thousands of constraints interact. By uncovering dominated constraints, presolving reduces dimensionality not by eliminating variables outright but by sharpening their permissible ranges. This yields a model that is leaner, yet faithfully equivalent to the original.

Ultimately, dominated constraints exemplify how structural reasoning turns a superficially loose formulation into a tightly defined optimization task, ensuring solvers converge more quickly and robustly.



## 2. Recursive Application of Presolving:

- ▶ Eliminating variables/constraints can create new opportunities for further eliminations.
- ▶ Example:

$$3x_2 = 6 \quad (\text{Row Singleton})$$

$$x_2 + 4x_5 = 10$$

- ▶ First constraint implies  $x_2 = 2$ . Eliminate  $x_2$  and first constraint.
- ▶ Second constraint becomes  $4x_5 = 10 - 2 = 8$ , which is now a Row Singleton.
- ▶ This leads to  $x_5 = 2$ , eliminating  $x_5$  and the second constraint.

Note: Detailed information on presolving is scarce due to its commercial value in LP software.

13/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

An important observation is that presolving operates not just once, but recursively. Eliminating a variable or a constraint can expose new opportunities for further simplification.

For example, suppose the system contains the equations  $3x_2 = 6$  and  $x_2 + 4x_5 = 10$ . The first relation is a row singleton, immediately yielding  $x_2 = 2$ . Substituting this value eliminates both the variable and the equation. The second relation then reduces to  $4x_5 = 8$ , which itself is a row singleton, leading directly to  $x_5 = 2$ . Through this chain, two variables and both constraints vanish entirely.

This recursive nature is what makes presolving extremely powerful. A single reduction may propagate through the model, triggering cascades of simplifications. The end result is a much leaner system, one that preserves the same optimal solution but requires far less computational effort to solve.

However, despite its importance, detailed methodologies of presolving are not widely published. The reason lies in their commercial value: state-of-the-art optimization solvers often rely heavily on proprietary presolving routines that distinguish their performance from competitors. This scarcity of open literature highlights how central presolving has become in practical optimization.

In essence, presolving embodies the principle of problem transformation. Rather than tackling a complex formulation head-on, it reshapes the problem into a smaller, sharper, and more efficient form. Recursive application ensures no opportunity for reduction is missed, making presolving one of the cornerstones of modern large-scale linear programming.



## Definition: Quadratic Program

A *quadratic program* (QP) is an optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} q(\mathbf{x}) &= \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \text{ s.t.} \\ \mathbf{a}_i^T \mathbf{x} &= b_i, i \in \mathcal{E}, \\ \mathbf{a}_i^T \mathbf{x} &\geq b_i, i \in \mathcal{I}, \end{aligned}$$

$\mathbf{G}$ : symmetric  $n \times n$  matrix;  $\mathbf{c}, \mathbf{x}, \mathbf{a}_i$ : vectors in  $\mathbb{R}^n$ ;  $\mathcal{E}, \mathcal{I}$ : finite index sets.

- ▶ QPs arise as subproblems in sequential quadratic programming, augmented Lagrangian, and interior-point methods.
- ▶ Always solvable (or infeasible) in finite computation; effort depends on objective function and number of inequality constraints.
- ▶ *Convex QP*:  $\mathbf{G}$  positive semidefinite, similar in difficulty to linear programs.
- ▶ *Strictly convex QP*:  $\mathbf{G}$  positive definite.
- ▶ *Nonconvex QP*:  $\mathbf{G}$  indefinite, may have multiple stationary points and local minima.

Convex QPs are computationally tractable; nonconvex QPs are more challenging.

14/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

Up to this point, we have focused on linear programming and related optimization frameworks. These methods allowed us to describe and solve a wide range of problems where both the objective function and the constraints were linear. However, many important applications in science, engineering, and economics involve objectives that are not simply linear but quadratic in nature. To handle such cases, we turn to the theory of quadratic programming, which generalizes linear programming by incorporating quadratic terms in the objective function while preserving linear constraints. This extension significantly broadens the modeling power of optimization.

A quadratic program is defined by an objective function that combines both quadratic and linear terms. Concretely, the goal is to minimize the function  $\frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c}$ , subject to linear constraints. Here,  $\mathbf{G}$  is a symmetric matrix,  $\mathbf{c}$  is a vector of coefficients, and the feasible set is determined by equalities and inequalities of the form  $\mathbf{a}_i^T \mathbf{x}$  equal to or greater than  $b_i$ . This structure retains the clarity of linear programming but introduces curvature into the optimization landscape.

The nature of the problem depends heavily on the matrix  $\mathbf{G}$ . If  $\mathbf{G}$  is positive semidefinite, the objective function is convex, and the problem behaves much like a linear program in terms of difficulty. If  $\mathbf{G}$  is strictly positive definite, we obtain a strictly convex optimization problem, which guarantees a unique solution. By contrast, if  $\mathbf{G}$  is indefinite, the problem becomes nonconvex, leading to multiple stationary points, including possibly many local minima. This distinction is central in practice because convex quadratic programs can be solved reliably and efficiently, whereas nonconvex ones are substantially more challenging.

Quadratic programming plays an essential role not only as a standalone tool but also as a subproblem within more advanced methods. For example, sequential quadratic programming and interior-point algorithms rely on solving quadratic subproblems as intermediate steps toward tackling more complex nonlinear optimization tasks. The appeal of quadratic programming is that, while the problem may appear more difficult than linear programming, in its convex form it is computationally tractable and enjoys strong theoretical guarantees.

## Example: Portfolio Optimization

Portfolio theory models the tradeoff between *risk* and *return*: higher expected return requires tolerating greater risks.

- ▶ Consider  $n$  investments with returns  $r_i$ ,  $i = 1, \dots, n$ , assumed to be random variables (often normal).
- ▶ Characterized by:
  - ▶ Expected value:  $\mu_i = E[r_i]$ ,
  - ▶ Variance:  $\sigma_i^2 = E[(r_i - \mu_i)^2]$  (larger  $\sigma_i$   $\Rightarrow$  riskier).
- ▶ Returns not independent; correlation between  $r_i$ ,  $r_j$ :

$$\rho_{ij} = \frac{E[(r_i - \mu_i)(r_j - \mu_j)]}{\sigma_i \sigma_j}, \quad i, j = 1, \dots, n.$$

- ▶  $\rho_{ij}$  measures co-movement:  $\rho_{ij} \approx 1 \Rightarrow$  returns track closely; negative  $\rho_{ij} \Rightarrow$  opposite movements.
- ▶ Portfolio: fraction  $x_i$  of funds in investment  $i$ ,  $i = 1, \dots, n$ .
- ▶ Constraints (full investment, no short-selling):  $\sum_{i=1}^n x_i = 1, \quad x \geq 0$ .

T-P Simplex:  
Example  
Dual Simplex  
Presolving  
Quadratic Programming

KKT  
Iterative Solution



15/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

One of the most celebrated applications of quadratic programming is found in portfolio optimization, a foundational concept in modern finance. The central idea is to model the tradeoff between risk and return: higher expected returns typically come with higher risk, and investors must balance these two objectives when allocating their wealth across different assets.

Consider a situation with  $n$  possible investments, each associated with a random return denoted by  $r_i$ . For each return, the expected value  $\mu_i$  represents the average gain, while the variance  $\sigma_i^2$  measures the volatility, or riskiness, of the investment. A higher variance indicates that returns are more uncertain. However, investments are rarely independent. The correlation coefficient  $\rho_{ij}$  between two returns  $r_i$  and  $r_j$  captures how their outcomes move together. A correlation close to one means the assets behave almost identically, while a negative correlation suggests that gains in one tend to be offset by losses in the other.

A portfolio is represented by a vector  $x$ , where each component  $x_i$  specifies the fraction of wealth allocated to asset  $i$ . Standard constraints ensure that all available capital is invested — meaning the sum of  $x_i$  equals one — and that allocations are nonnegative, ruling out short selling. This simple framework already demonstrates the richness of the model, as the introduction of correlation among assets provides investors with the possibility of diversification: spreading investments across imperfectly correlated assets reduces overall risk without necessarily lowering expected return.

## Portfolio Optimization (continued)

- ▶ Portfolio return:  $R = \sum_{i=1}^n x_i r_i$ .
- ▶ Expected return:  $E[R] = E \left[ \sum_{i=1}^n x_i r_i \right] = \sum_{i=1}^n x_i E[r_i] = \sum_{i=1}^n x_i \mu_i = x^T \mu$ .
- ▶ Variance:  

$$\text{Var}[R] = \sum_{i=1}^n \sum_{j=1}^n x_i x_j \sigma_i \sigma_j \rho_{ij} = x^T G x,$$

where  $G_{ij} = \rho_{ij} \sigma_i \sigma_j$  is the symmetric positive semidefinite *covariance matrix*.

Goal: Maximize expected return  $x^T \mu$  while minimizing variance  $x^T G x$ .

- ▶ Markowitz model combines objectives using risk tolerance  $\kappa \geq 0$ :

$$\max x^T \mu - \kappa x^T G x, \quad \text{s.t. } \sum_{i=1}^n x_i = 1, x \geq 0.$$

- ▶  $\kappa$  reflects investor preference:

- ▶ Large  $\kappa$ : Conservative, emphasizes low variance (risk).
- ▶ Small  $\kappa$ : Daring, prioritizes high return.

- ▶ Challenge: Estimating  $\mu_i$ ,  $\sigma_i^2$ ,  $\rho_{ij}$  using historical data and financial insights.

Markowitz model balances return and risk via  $\kappa$ .

16/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



### Comments

The mathematical formulation of portfolio return builds directly on these definitions. The portfolio return  $R$  is a weighted sum of the individual asset returns, where the weights are the allocations  $x_i$ . Its expected value,  $E$ , is simply the dot product  $x^T \mu$ , reflecting the linear combination of expected asset returns. The variance of the portfolio, however, captures interactions between all pairs of assets.

Specifically, it can be written as  $x^T G x$ , where  $G$  is the covariance matrix, built from the correlations and standard deviations of individual assets. This quadratic structure makes portfolio optimization a natural quadratic programming problem.

The classical Markowitz model seeks to balance these competing objectives. It introduces a parameter  $\kappa$ , called risk tolerance, that determines the tradeoff. The optimization goal is to maximize expected return minus  $\kappa$  times the variance. A large value of  $\kappa$  corresponds to a conservative investor who places high emphasis on reducing risk. A small value of  $\kappa$  corresponds to a more aggressive investor willing to tolerate volatility in pursuit of higher returns.

A crucial challenge lies in estimating the inputs: the expected returns  $\mu_i$ , the variances  $\sigma_i^2$ , and the correlations  $\rho_{ij}$ . In practice, these are derived from historical data and financial modeling, but they are subject to uncertainty and estimation error. Nevertheless, the Markowitz framework has had enormous influence because it formalizes the intuition that diversification reduces risk and because it provides a systematic method for designing portfolios that reflect different levels of risk aversion.

Equality-constrained QPs, critical for general QP algorithms, are formulated as:

$$\min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \text{ s.t. } \mathbf{A} \mathbf{x} = \mathbf{b},$$

where  $\mathbf{G}$  is a symmetric  $n \times n$  matrix,  $\mathbf{A}$  is an  $m \times n$  Jacobian ( $m \leq n$ , full row rank), and  $\mathbf{b} \in \mathbb{R}^m$ . The constraint  $\mathbf{A} \mathbf{x} = \mathbf{b}$  ensures feasibility, with full rank guaranteeing consistency.

- ▶ **First-order optimality (KKT system):** Solution  $\mathbf{x}^*$  requires a Lagrange multiplier vector  $\lambda^*$  satisfying the KKT system:

$$\begin{bmatrix} \mathbf{G} & -\mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -\mathbf{c} \\ \mathbf{b} \end{bmatrix}.$$

This system balances the gradient  $\nabla q(\mathbf{x}^*) = \mathbf{G}\mathbf{x}^* + \mathbf{c}$  with constraint forces  $\mathbf{A}^T\lambda^*$ .

- ▶ Computational approach: Express  $\mathbf{x}^* = \mathbf{x} + \mathbf{p}$ , where  $\mathbf{p}$  is the step from estimate  $\mathbf{x}$ . Solve:

$$\begin{bmatrix} \mathbf{G} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} -\mathbf{p} \\ \lambda^* \end{bmatrix} = \begin{bmatrix} \mathbf{c} + \mathbf{G}\mathbf{x} \\ \mathbf{Ax} - \mathbf{b} \end{bmatrix}.$$

Here,  $\mathbf{c} + \mathbf{G}\mathbf{x}$  is the gradient at  $\mathbf{x}$ , and  $\mathbf{Ax} - \mathbf{b}$  measures constraint violation and the matrix is called the Karush-Kuhn-Tucker (KKT) matrix.

17/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



## Comments

Equality-constrained quadratic programs represent a very important subclass of quadratic optimization problems. They consist of minimizing a quadratic function subject only to equality constraints. Concretely, the objective has the form  $\frac{1}{2}\mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c}$ , where  $\mathbf{G}$  is a symmetric matrix and  $\mathbf{c}$  is a vector. The constraints are linear equalities written as  $\mathbf{A}\mathbf{x} = \mathbf{b}$ . Here  $\mathbf{A}$  is an  $m \times n$  matrix of full row rank, and  $\mathbf{b}$  is a vector in  $m$  dimensions. The full rank assumption ensures that the equalities are consistent and that the feasible region is well-defined.

To find an optimal solution, one uses the first-order optimality conditions, often called the Karush-Kuhn-Tucker or KKT system. The idea is that at a solution  $\mathbf{x}^*$ , there exists a multiplier vector  $\lambda^*$  such that the gradient of the objective, which is  $\mathbf{G}\mathbf{x}^* + \mathbf{c}$ , is exactly balanced by the constraint forces  $\mathbf{A}^T\lambda^*$ . While keeping the restrictions on the  $\mathbf{x}^*$ :  $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ .

From a computational perspective, one often updates an approximate solution  $\mathbf{x}$  by considering a correction step  $\mathbf{p}$ , with the new solution being  $\mathbf{x} + \mathbf{p}$ . The corresponding linear system can again be written in block form, now with  $\mathbf{G}$  in the top-left and  $\mathbf{A}^T$  in the top-right, and  $\mathbf{A}$  and zero on the bottom row.

On the right-hand side, the top block is the gradient at the current  $\mathbf{x}$ , namely  $\mathbf{c} + \mathbf{G}\mathbf{x}$ , and the bottom block is the constraint residual, namely  $\mathbf{Ax} - \mathbf{b}$ . Solving this system produces both the step direction  $\mathbf{p}$  and the updated multipliers. This KKT matrix, while structured, is often large and indefinite, making its numerical solution a central challenge in quadratic programming.

The KKT matrix arises in equality-constrained QPs. Let  $Z$  be an  $n \times (n - m)$  matrix with full rank, forming a basis for the null space of  $A$  (i.e.,  $AZ = 0$ ).

## Lemma 10: KKT Nonsingularity

If  $A$  has full row rank and  $Z^T G Z$  is positive definite, then the KKT matrix

$$K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix}$$

is nonsingular, ensuring a unique solution  $(x^*, \lambda^*)$  to the KKT system.

**Proof:** Suppose  $\exists w, v$  such that

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

We have from this equation (since  $Aw = 0$ ), that

$$0 = \begin{bmatrix} w \\ v \end{bmatrix}^T \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = w^T G w.$$

Since  $w$  lies in the null space of  $A$ , it can be written as  $w = Zu$  for some vector  $u \in \mathbb{R}^{n-m}$ .

18/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



### Comments

When analyzing equality-constrained quadratic programs, one of the central questions is whether the Karush-Kuhn-Tucker, or KKT, system admits a unique solution. The KKT matrix in this setting is built as a block structure with the Hessian matrix, denoted by capital  $G$ , in the upper-left block, the transpose of capital  $A$  in the upper-right block, capital  $A$  itself in the lower-left block, and a block of zeros in the lower-right. The uniqueness of the solution depends critically on two conditions.

First, the matrix capital  $A$  must have full row rank, meaning its rows are linearly independent.

Second, the reduced Hessian, which takes the form  $Z^T G Z$ , must be positive definite. Here,  $Z$  is a basis matrix for the null space of  $A$ , that is, any vector  $w$  satisfying  $Aw = 0$  can be written as  $Z$  times some vector  $u$ .

The proof of nonsingularity is straightforward but important. Suppose the KKT system has a nontrivial solution vector, written as  $w$  stacked with  $v$ . Multiplying through shows that  $w^T G w = 0$ . Since  $w$  lies in the null space of  $A$ , we can express it as  $Z$  times  $u$ .

From  $w = Zu$ , we have

$$0 = w^T G w = u^T Z^T G Z u.$$

Since  $Z^T G Z$  is positive definite,  $u = 0$ , so  $w = 0$ . Then,  $A^T v = 0$ , and full row rank of  $A$  implies  $v = 0$ . Thus, only  $w = 0$ ,  $v = 0$  satisfies the KKT null space, proving nonsingularity.  $\square$

## Example: Equality-Constrained QP

Consider:

$$\min 3x_1^2 + 2x_1x_2 + x_1x_3 + 2.5x_2^2 + 2x_2x_3 + 2x_3^2 - 8x_1 - 3x_2 - 3x_3 \text{ s.t. } x_1 + x_3 = 3, x_2 + x_3 = 0.$$

Form:  $G = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{bmatrix}$ ,  $c = \begin{bmatrix} -8 \\ -3 \\ -3 \end{bmatrix}$ ,  $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ ,  $b = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ . Solution:

$$x^* = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \lambda^* = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, \text{ with } G \text{ positive definite, } Z = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}.$$

Positive definite  $G$  and full rank  $A$  ensure unique QP solution.

19/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example  
Dual Simplex  
Presolving  
Quadratic Programming

KKT

Iterative Solution



## Comments

Substituting gives  $u^T Z^T G Z u = 0$ . But if the reduced Hessian is positive definite, the only solution is  $u = 0$ , hence  $w = 0$ .

Then from  $A^T v = 0$ , and the full row rank of  $A$ , it follows that  $v$  must also vanish.

The only solution is the trivial one, proving that the KKT matrix is nonsingular. This establishes the foundation for the existence of a unique solution pair, denoted by  $x^*$  and  $\lambda^*$ .

To see how this theory works in practice, consider an equality-constrained quadratic program in three variables.

The objective is a quadratic function:  $3x_1^2 + 2x_1x_2 + x_1x_3 + 2.5x_2^2 + 2x_2x_3 + 2x_3^2 - 8x_1 - 3x_2 - 3x_3$ .

The constraints are linear equalities:  $x_1 + x_3 = 3$ , and  $x_2 + x_3 = 0$ . In matrix notation, the Hessian  $G$  is a three-by-three symmetric matrix with entries six, two, one in the first row, two, five, two in the second, and one, two, four in the third.

The vector  $c$  is minus eight, minus three, minus three.

The constraint matrix  $A$  is two by three with first row one, zero, one and second row zero, one, one.

The right-hand side vector  $b$  is three and zero.

Solving this quadratic program yields the optimal point  $x^*$  equal to the vector two, minus one, one, and the corresponding multipliers  $\lambda^* = (3, -2)^T$ .

The Hessian is positive definite, and the null-space basis matrix  $Z$  can be taken as the vector  $(-1, -1, 1)^T$ . This example illustrates that with a positive definite  $G$  and a full-rank  $A$ , the problem admits a unique solution.



## Theorem 35: Global Solution

If  $A$  has full row rank and assume that the reduced-Hessian matrix  $Z^T G Z$  is positive definite, then  $x^*$  satisfying the KKT system is the unique global solution of the equality-constrained QP.

**Proof:** For feasible  $x$  ( $Ax = b$ ), define  $p = x^* - x$ , so  $Ap = 0$ . The objective expands as:

$$q(x) = \frac{1}{2}(x^* - p)^T G(x^* - p) + c^T(x^* - p) = \frac{1}{2}p^T Gp - p^T Gx^* - c^T p + q(x^*).$$

Since  $Gx^* = -c + A^T \lambda^*$  (satisfying the KKT system) and  $Ap = 0$ , we get:

$$q(x) = \frac{1}{2}p^T Gp - p^T(-c + A^T \lambda^*) - c^T p + q(x^*) = \frac{1}{2}p^T Gp + q(x^*).$$

As  $p = Zu$  we have  $q(x) = \frac{1}{2}u^T Z^T G Z u + q(x^*)$  and positive definiteness of  $Z^T G Z$  ensures  $q(x) > q(x^*)$  unless  $p = 0$  (i.e.,  $x = x^*$ ).  $\square$

- Positive semidefinite  $Z^T G Z$  with zero eigenvalues:  $x^*$  is a local minimizer.
- Negative eigenvalues in  $Z^T G Z$ :  $x^*$  is a stationary point.

Positive definite  $Z^T G Z$  ensures a unique global minimum.

20/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

The discussion of nonsingularity sets the stage for a deeper result: the global solution theorem for equality-constrained quadratic programs.

It asserts that if the constraint matrix  $A$  has full row rank and the reduced Hessian  $Z^T G Z$  is positive definite, then any point  $x^*$  that satisfies the KKT system is not only a stationary point but the unique global minimizer of the problem. The proof relies on examining any feasible vector  $x$ , that is, any  $x$  with  $Ax = b$ . Let  $p$  be defined as  $x^* - x$ , so that  $Ap = 0$ .

Expanding the quadratic objective function  $q(x)$  leads to a decomposition. Specifically,  $q(x)$  equals  $\frac{1}{2}p^T Gp - p^T Gx^* - c^T p + q(x^*)$ . From the KKT system we know that  $Gx^* = -c + A^T \lambda^*$ .

Since  $Ap = 0$ , the cross terms vanish, and we are left with  $q(x) = \frac{1}{2}p^T Gp + q(x^*)$ .

By expressing  $p$  as  $Zu$ , the expression becomes  $\frac{1}{2}u^T Z^T G Z u + q(x^*)$ .

If the reduced Hessian is positive definite, this expression is strictly greater than  $q(x^*)$  unless  $u = 0$ .

Thus, no feasible point has a lower objective value than  $x^*$ , and the minimizer is unique.

If  $Z^T G Z$  is only positive semidefinite, the situation weakens:  $x^*$  remains a local minimizer but not necessarily unique.

If  $Z^T G Z$  has negative eigenvalues, then  $x^*$  is merely a stationary point without optimality guarantees.

This reasoning emphasizes how definiteness conditions translate into global solution properties.



The KKT system for equality-constrained QPs is indefinite for  $m \geq 1$ .

The inertia of a symmetric matrix  $K$  is defined as  $\text{inertia}(K) = (n_+, n_-, n_0)$ , where  $n_+$ ,  $n_-$ ,  $n_0$  are the numbers of positive, negative, and zero eigenvalues.

## Theorem 36: KKT Inertia

For  $K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix}$  with  $A$  of rank  $m$ , the inertia is:

$$\text{inertia}(K) = \text{inertia}(Z^T G Z) + (m, m, 0).$$

If  $Z^T G Z$  is positive definite,  $\text{inertia}(K) = (n, m, 0)$ .

**Proof in:** N. I. M. Gould, *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem*, Math. Program., 32 (1985), pp. 90–99 (see Lemma 3.4, p. 96).

Inertia informs efficient KKT system solution strategies.

21/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

The final step is to connect nonsingularity and optimality with the spectral structure of the KKT system itself. Because the KKT matrix combines both the Hessian and the constraint structure, it is not positive definite in general. For any problem with at least one equality constraint, that is, when  $m \geq 1$ , the KKT system is indefinite.

This means its spectrum contains both positive and negative eigenvalues. To analyze this systematically, we use the concept of inertia of a symmetric matrix. The inertia is a triplet consisting of the numbers of positive, negative, and zero eigenvalues.

For the KKT matrix, defined as the block matrix with  $G$  and  $A^T$  in the top row, and  $A$  and zero in the bottom row, the inertia can be expressed in terms of the reduced Hessian.

Specifically, the inertia of the KKT matrix equals the inertia of  $Z^T G Z$  plus the triplet  $(m, m, 0)$ . An important corollary arises when the reduced Hessian is positive definite. In that case,  $Z^T G Z$  contributes only positive eigenvalues, and the inertia of the full KKT matrix becomes  $(n, m, 0)$ .

Here  $n - m$  counts the positive eigenvalues corresponding to the reduced Hessian, while the remaining  $m$  of positive eigenvalues and  $m$  of negative eigenvalues counts both positive and negative directions associated with the constraints. This result, due to Nicholas Gould in the mid-1980s, is crucial for practical algorithms, because inertia reveals whether a factorization of the KKT system is numerically stable and whether the resulting search directions are well-defined. Understanding inertia therefore provides both theoretical insight and practical guidance for solving quadratic programs efficiently.

## Solving the KKT System

The KKT matrix is indefinite for  $m \geq 1$ , ruling out Cholesky factorization. Gaussian elimination with partial pivoting ignores symmetry, so symmetric indefinite factorization is preferred:

$$P^T K P = L B L^T,$$

where  $P$  is a permutation matrix,  $L$  is unit lower triangular, and  $B$  is block-diagonal ( $1 \times 1$  or  $2 \times 2$  blocks).

- Solution steps for KKT system:

$$\text{solve } Lz = P^T \begin{bmatrix} c + Gx \\ Ax - b \end{bmatrix} \text{ for } z;$$

$$\text{solve } B\hat{z} = z \text{ for } \hat{z};$$

$$\text{solve } L^T \bar{z} = \hat{z} \text{ for } \bar{z};$$

$$\text{set } \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = P\bar{z}, \text{ (recall: } x^* = x + p)$$

- Permutations ( $P, P^T$ ) are inexpensive;  $B$  systems are small; triangular solves with  $L, L^T$  depend on sparsity.

Symmetric factorization is efficient but costly if  $L$  loses sparsity.

22/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



### Comments

The Karush–Kuhn–Tucker system involves an indefinite matrix, which immediately eliminates the possibility of Cholesky factorization. Instead, symmetric indefinite factorization is the natural choice. This approach rearranges the system using permutation matrices and reduces it to triangular solves with a block-diagonal middle matrix.

The procedure unfolds in three simple steps: forward substitution, block solve, and backward substitution. Each of these respects the sparsity structure, making the method efficient when the underlying problem is sparse.

The final step reconstructs both the primal direction and the Lagrange multipliers. What is remarkable here is that the expensive work is confined to factorizations, while the application of permutations and triangular solves is relatively cheap. Nonetheless, if the triangular factor loses too much sparsity during factorization, the overall method can become costly.

Hence, there is a delicate balance: symmetry is preserved, stability is ensured, but computational effort depends strongly on how much structure the factorization destroys.

Assuming  $G$  is positive definite, multiply the first KKT equation by  $AG^{-1}$  and subtract the second to get:

$$(AG^{-1}A^T)\lambda^* = AG^{-1}(c + Gx) - (Ax - b).$$

Solve this for  $\lambda^*$ , then recover  $p$  from:

$$Gp = A^T\lambda^* - (c + Gx).$$

Block Gaussian elimination on the KKT matrix yields:

$$\begin{bmatrix} G & A^T \\ 0 & -AG^{-1}A^T \end{bmatrix}.$$

- ▶ Effective when:  $G$  is well-conditioned (e.g., diagonal);  $G^{-1}$  is known; or  $m$  is small (so that the number of backsolves needed to form the matrix  $AG^{-1}A^T$  is not too large).

Schur-Complement method leverages  $G^{-1}$  for efficient KKT solutions.

23/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



## Comments

When the Hessian matrix  $G$  is positive definite, the KKT system can be represented as two subsystems separated through the Schur complement. Multiplying the first KKT equation by the  $AG^{-1}$ , followed by subtraction, produces a reduced system in terms of the multipliers  $\lambda^*$ . The central object here is the matrix  $AG^{-1}A^T$  which captures the constraint interactions.

Once the multipliers are determined from this reduced system, the search direction for the primal variables is recovered by solving a linear system with  $G$ . Conceptually, this approach transforms the saddle-point structure into two coupled problems: one of size equal to the number of constraints, and another of size equal to the number of primal variables.

Efficiency arises when  $G$  is simple, such as diagonal, or when its inverse is easily available. In such situations, forming and solving with the Schur complement is highly attractive. However, when  $G$  is large and dense, explicitly building  $AG^{-1}A^T$  may be too expensive.

Therefore, the method shines when the number of constraints is modest and the Hessian has a structure that makes inversion cheap.



The matrix  $AG^{-1}A^T$  is the Schur complement of  $G$  in the KKT matrix  $K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix}$ . Block elimination on the KKT system yields the Schur-Complement method equations.

Explicit inverse of  $K$ :

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix}^{-1} = \begin{bmatrix} C & E \\ E^T & F \end{bmatrix},$$

- ▶  $C = G^{-1} - G^{-1}A^T(AG^{-1}A^T)^{-1}AG^{-1}$ ,
- ▶  $E = G^{-1}A^T(AG^{-1}A^T)^{-1}$ ,
- ▶  $F = -(AG^{-1}A^T)^{-1}$ .

Multiplying the KKT system's right-hand side by this inverse, grouping terms, recovers the Schur-Complement method.

Explicit inverse links directly to Schur-Complement solution.

24/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The Schur complement provides not only a computational strategy but also a clean algebraic perspective.

If we consider the KKT matrix as a block matrix, its inverse can be written explicitly in terms of  $G^{-1}$  and the inverse of the Schur complement  $AG^{-1}A^T$ .

Each block of this inverse has an interpretation: the  $C$  block corresponds to adjustments in the primal variables, the  $E$  block links constraints to variables, and the  $F$  block represents interactions between constraints themselves. By multiplying the right-hand side of the KKT equations with this inverse, we recover exactly the Schur-complement equations previously derived.

Thus, the inverse form and the elimination approach are simply two views of the same underlying mechanism.

The explicit inverse is rarely formed in practice, because it is dense and costly. Yet, it provides valuable theoretical insight, clarifying how constraints project onto the variable space and how feasibility and optimality conditions are enforced simultaneously.



The null-space method, unlike Schur-Complement, does not require  $G$  nonsingularity, only  $A$  full row rank and  $Z^T G Z$  positive definite. It uses a null-space basis  $Z$  to decouple the KKT system into smaller systems.

Decompose  $p$  as:

$$p = Yp_Y + Zp_Z,$$

where  $Z$  is  $n \times (n - m)$  null-space matrix ( $AZ = 0$ ),  $Y$  is  $n \times m$  such that  $[Y | Z]$  is nonsingular,  $p_Y \in \mathbb{R}^m$ ,  $p_Z \in \mathbb{R}^{n-m}$ . Substituting into the KKT system's second equation gives:

$$(AY)p_Y = -(Ax - b).$$

- $Yp_Y$  is a particular solution to  $A(x + p) = b$ ;  $Zp_Z$  moves along constraints.

Null-space method leverages  $Z$  for broader applicability.

25/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The null-space method offers an alternative to the Schur complement approach for solving the Karush–Kuhn–Tucker, or KKT, system. Unlike the Schur complement, it does not require the Hessian matrix, denoted as  $G$ , to be invertible. Instead, the only requirements are that the constraint matrix  $A$  has full row rank, and that the matrix  $Z^T G Z$  is positive definite. Here  $Z$  is a basis for the null space of  $A$ , meaning that  $AZ = 0$ .

The key idea is to split the step vector  $p$  into two parts: a particular part that ensures feasibility, and a null-space part that moves within the feasible region. Formally, we write  $p = Yp_Y + Zp_Z$ . The columns of  $Y$  are chosen so that together with the columns of  $Z$  they form a full basis of the entire space. Substituting this decomposition into the KKT equations shows that the component  $Yp_Y$  ensures that  $A(x + p) = b$  is satisfied. Meanwhile, the component  $Zp_Z$  represents directions that stay within the feasible set, since  $AZ = 0$ .

Thus, the null-space method eliminates the need to solve for all variables simultaneously. Instead, it first enforces feasibility through  $Yp_Y$ , and then optimizes within the feasible set by solving for  $p_Z$ . This makes the method broadly applicable even when  $G$  is singular or poorly conditioned.



Given  $A$  rank  $m$  and nonsingular  $[Y \mid Z]$ ,  $AY$  is an  $m \times m$  nonsingular matrix, as  $A[Y \mid Z] = [AY \mid 0]$  has rank  $m$ . Solve  $(AY)p_Y = -(Ax - b)$  for  $p_Y$ , the component of  $p$  in the range of  $Y$ .

Substitute  $p = Yp_Y + Zp_Z$  into the KKT system's first equation:

$$-GYp_Y - GZp_Z + A^T \lambda^* = c + Gx.$$

Multiply by  $Z^T$  (since  $AZ = 0$ ) to isolate  $p_Z$ :

$$(Z^T GZ)p_Z = -Z^T GYp_Y - Z^T(c + Gx).$$

Solve for  $p_Z$  using Cholesky factorization of positive definite  $Z^T GZ$ , then compute total step  $p = Yp_Y + Zp_Z$ .

- ▶ For  $\lambda^*$ , solve  $(AY)^T \lambda^* = Y^T(c + Gx + Gp)$ , using the computed  $p$ .

Null-space method efficiently decouples KKT system via  $Y$  and  $Z$ .

26/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The construction of the null-space method becomes clearer once we look at how the system of equations decouples.

Recall that we decomposed the step vector  $p$  into  $Yp_Y + Zp_Z$ . Because the block matrix formed by concatenating  $Y$  and  $Z$  is nonsingular, the submatrix  $AY$  is square and invertible. This allows us to compute the component  $p_Y$  uniquely by solving the linear system  $(AY)p_Y = -(Ax - b)$ . In other words,  $p_Y$  is the correction that ensures feasibility of the equality constraints.

Next, we substitute the full decomposition  $p = Yp_Y + Zp_Z$  into the first block of the KKT equations, which originally reads  $-Gp + A^T \lambda^* = c + Gx$ . After substitution, this becomes  $-GYp_Y - GZp_Z + A^T \lambda^* = c + Gx$ . To separate the null-space component, we multiply the equation by  $Z^T$ . Since  $AZ = 0$ , this operation eliminates the dependence on  $\lambda^*$  and isolates  $p_Z$ .

The resulting equation is  $(Z^T GZ)p_Z = -Z^T GYp_Y - Z^T(c + Gx)$ . The matrix  $Z^T GZ$  is symmetric and positive definite, and therefore well-suited for numerical solution using Cholesky factorization.

Once  $p_Z$  is obtained, the complete step is reconstructed as  $Yp_Y + Zp_Z$ . Finally, with  $p$  known, the multipliers  $\lambda^*$  can be recovered by solving the smaller system involving the transpose of  $AY$ . Altogether, the method achieves an elegant decoupling: feasibility is handled through  $p_Y$ , optimality is enforced through  $p_Z$ , and the multipliers are obtained as a final adjustment.



## Example: Applying Null-Space Method

For the QP with  $G$ ,  $c$ ,  $A$ ,  $b$  as defined in previous Example:  $G = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{bmatrix}$ ,

$$c = \begin{bmatrix} -8 \\ -3 \\ -3 \end{bmatrix}, A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \text{choose: } Y = \begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \\ 1/3 & 1/3 \end{bmatrix}, Z = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix},$$

$AY = I$ . At  $x = (0, 0, 0)^T$  compute  $Ax - b = -b$ ,  $c + Gx = (-8, -3, -3)^T$ . Solve

$(AY)p_Y = b$  to get  $p_Y = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ , from  $(Z^T G Z)p_Z = -Z^T G Y p_Y - Z^T(c + Gx)$  get  $p_Z = [0]$ . Total step:  $p = Y p_Y + Z p_Z = (2, -1, 1)^T$ .

- ▶ Solve  $(AY)^T \lambda^* = Y^T(c + Gx + Gp)$  to obtain  $\lambda^* = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$ .
- ▶ Solution:  $x^* = x + p = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$ ,  $\lambda^* = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$ .

Null-space method computes  $x^*$  and  $\lambda^*$  efficiently.

## Comments

To illustrate the mechanics of the null-space method, let us work through a concrete quadratic programming problem. The Hessian matrix  $G$  is given as the three-by-three matrix with rows: six, two, one; two, five, two; one, two, four. The cost vector  $c$  is negative eight, negative three, negative three.

The constraint matrix  $A$  has two rows: the first row is one, zero, one; the second row is zero, one, one. The right-hand side vector  $b$  is three, zero. A suitable choice of bases is  $Y$  equal to the three-by-two matrix with columns two-thirds, minus one-third, one-third and minus one-third, two-thirds, one-third; and  $Z$  equal to the three-by-one vector minus one, minus one, one. With this choice,  $AY$  equals the identity matrix. We begin at the point  $x = (0, 0, 0)^T$ .

Then  $Ax - b$  equals minus  $b$ , so the feasibility correction is determined by solving  $(AY)p_Y = b$ . Since  $AY$  is the identity, this immediately gives  $p_Y$  equal to the vector three, zero.

Next, we compute the null-space component by solving the reduced system  $(Z^T G Z)p_Z = -Z^T G Y p_Y - Z^T(c + Gx)$ . Substitution shows that the right-hand side vanishes, so  $p_Z = 0$ . Therefore the full step is  $Y p_Y + Z p_Z$ , which evaluates to the vector two, negative one, one.

Finally, to compute the multipliers, we solve the system  $(AY)^T \lambda^* = Y^T(c + Gx + Gp)$ . This yields  $\lambda^*$  equal to the vector three, negative two. Thus, the optimal solution of this quadratic program is  $x^*$  equal to two, negative one, one, with associated multipliers  $\lambda^*$  equal to three and negative two. The example demonstrates how the method enforces feasibility via  $p_Y$ , finds the optimal feasible step using  $p_Z$ , and then recovers the multipliers in a clean sequence of linear solves, each of relatively small dimension.

## Iterative Null-Space Method

Iterative methods suit large KKT systems, leveraging parallelization. CG is unstable for indefinite systems, so Krylov methods (GMRES, QMR, LSQR) or CG on the reduced system are preferred, assuming  $Z^T G Z$  positive definite.

Decompose solution:  $x^* = Yx_Y + Zx_Z$ , where  $x_Y \in \mathbb{R}^m$ ,  $x_Z \in \mathbb{R}^{n-m}$ ,  $Z$  is  $n \times (n - m)$  null-space matrix ( $AZ = 0$ ),  $Y$  is  $n \times m$  such that  $[Y \mid Z]$  is nonsingular. Constraints  $Ax = b$  give:  $AYx_Y = b$ .

Substitute into the QP to get the reduced problem:

$$\min_{x_Z} \frac{1}{2} x_Z^T Z^T G Z x_Z + x_Z^T c_z, \text{ where } c_z = (2Z^T G Y x_Y + Z^T c).$$

Solve  $(Z^T G Z)x_Z = -c_z$  using CG, then compute  $x^*$ .

- ▶ Preconditioner  $W_{zz}$  enhances CG convergence.
- ▶ Assumes  $A$  full rank,  $Z^T G Z$  positive definite.

CG on reduced system efficiently solves equality-constrained QP.

28/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

T-P Simplex:  
Example

Dual Simplex

Presolving

Quadratic  
Programming

KKT

Iterative  
Solution



### Comments

When the number of variables is very large, direct methods for solving the Karush–Kuhn–Tucker system become prohibitively expensive. In such cases, iterative approaches are preferable. Among the iterative methods, the conjugate gradient method is widely used, but it is only reliable when applied to positive definite systems.

Since the KKT system is indefinite, we cannot apply conjugate gradient directly. Instead, we work with the reduced null-space formulation. The idea is to decompose the solution into two components: a feasible part that enforces the constraints and a null-space part that optimizes within the feasible set. More precisely, the solution vector  $x^*$  is written as  $Yx_Y + Zx_Z$ , where the matrix  $Z$  has full rank, and  $AZ = 0$ .

The equality constraints  $Ax = b$  then reduce to  $(AY)x_Y = b$ . Once this feasible component is determined, the optimization reduces to minimizing a quadratic function of the null-space variables. The reduced objective function is  $\frac{1}{2} x_Z^T Z^T G Z x_Z + x_Z^T c_z$ . The vector  $c_z$  equals  $2Z^T G Y x_Y + Z^T c$ . The resulting linear system is  $(Z^T G Z)x_Z = -c_z$ .

This system is symmetric and positive definite under standard assumptions that  $A$  has a full rank and  $Z^T G Z$  is positive definite, which makes it suitable for solution by conjugate gradient.

Once  $x_Z$  is found, the full solution  $x^*$  is reconstructed as  $Yx_Y + Zx_Z$ . Preconditioning plays a crucial role: a carefully chosen preconditioner matrix  $W_{zz}$  can dramatically accelerate convergence (recall that the preconditioner is a special matrix used in the conjugate gradient method to improve the convergence rate. We discussed this earlier.).

**Goal:** Solve  $(Z^T G Z)x_Z = -c_z$  iteratively using CG with preconditioner  $W_{zz}$  (which is assumed to be given) for efficiency. **Algorithm: Preconditioned CG**

- 1: Choose initial  $x_Z$ ;
- 2: Compute  $r_z = Z^T G Z x_Z + c_z$ ,  $g_z = W_{zz}^{-1} r_z$ ,  $d_z = -g_z$ ;
- 3: for iterations until  $r_z^T W_{zz}^{-1} r_z$  is sufficiently small do
  - 4:  $\alpha \leftarrow r_z^T g_z / d_z^T Z^T G Z d_z$ ;
  - 5:  $x_Z \leftarrow x_Z + \alpha d_z$ ;
  - 6:  $r_z^+ \leftarrow r_z + \alpha Z^T G Z d_z$ ;
  - 7:  $g_z^+ \leftarrow W_{zz}^{-1} r_z^+$ ;
  - 8:  $\beta \leftarrow (r_z^+)^T g_z^+ / r_z^T g_z$ ;
  - 9:  $d_z \leftarrow -g_z^+ + \beta d_z$ ;
  - 10:  $g_z \leftarrow g_z^+$ ,  $r_z \leftarrow r_z^+$ ;
- 11: end for

Note: No explicit  $Z^T G Z$  or  $Z$  needed as long as we are able to compute products of  $Z$  and  $Z^T$  with arbitrary vectors. These products are often cheaper to compute than  $Z$  itself (for sparse  $Z$ ).

T-P Simplex:  
Example  
Dual Simplex  
Presolving  
Quadratic Programming  
KKT  
Iterative Solution



29/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

At this stage, we arrive at the reduced system  $(Z^T G Z)x_Z = -c_z$ . This is exactly the type of system for which the conjugate gradient method is designed: it is symmetric and positive definite.

Instead of solving it by direct factorization, which would be too expensive for large-scale problems, we employ the preconditioned conjugate gradient algorithm that we already studied earlier in the course.

So, conceptually, nothing new is happening here—we are simply applying Algorithm 10, the preconditioned conjugate gradient method, but now in the special setting of quadratic programming with equality constraints. Let me briefly recall the main idea behind conjugate gradients. Ordinary gradient descent follows the steepest descent direction at each step, but this often leads to slow zig-zagging.

The conjugate gradient method improves on this by building a sequence of search directions that are conjugate with respect to the matrix of the system. In other words, each new direction is chosen not only to reduce the error but also to avoid undoing progress made in earlier steps. This property guarantees convergence to the solution in at most  $n$  steps, and often much faster in practice.

The role of the preconditioner  $W_{zz}$  is to accelerate this process. If  $Z^T G Z$  has eigenvalues spread across a wide range, the convergence of conjugate gradients can be painfully slow. The preconditioner reshapes the system so that the eigenvalues are better clustered, which leads to faster convergence.

Importantly, we do not need to build  $Z^T G Z$  explicitly. It is sufficient to compute products of  $Z$  and  $Z^T$  with vectors, which can be done efficiently in structured or sparse cases. Thus, the message here is simple: we reduce the KKT problem to a symmetric positive definite system in the null space, and then solve it using a preconditioned conjugate gradient method that we already understand well.



- ▶ Preconditioner  $W_{zz} = Z^T H Z$ , with  $H$  symmetric and  $Z^T H Z$  positive definite, clusters eigenvalues of  $W_{zz}^{-1/2} (Z^T G Z) W_{zz}^{-1/2}$  to accelerate CG convergence.
- ▶ Ideal case:  $W_{zz} = Z^T G Z$ , making the system trivial.
- ▶ Projected CG solves  $A Y x_Y = b$  and updates  $x_Z$  in  $x = Z x_Z + Y x_Y$  implicitly in  $n$ -dimensional space, using projection matrix:

$$P = Z(Z^T H Z)^{-1} Z^T.$$

Operates with  $n$ -vectors:  $x = Z x_Z + Y x_Y$ ,  $Z^T r = r_z$ ,  $g = Z g_z$ ,  $d = Z d_z$ . This avoids explicit  $Z$ , reducing computational cost.

- ▶ Robust to ill-conditioned  $A$  or poor  $Z$ , as  $P$  projects onto null space.
- ▶ Solves reduced system  $(Z^T G Z)x_Z = -c_z$  using matrix-vector products.
- ▶ Ensures numerical stability via orthogonal  $Z$  assumption.

Projected CG efficiently handles large systems implicitly.

30/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Another approach provided by the projected conjugate gradient method is to work entirely in the original  $n$ -dimensional space and avoid building an explicit null-space basis  $Z$ . The problem is that the reduced system  $(Z^T G Z)x_Z = -c_z$ , although smaller in dimension, can still be very costly to solve if we form it explicitly.

To overcome this, one can rearrange the previous algorithm so that all computations are carried out in the original  $n$ -dimensional space, but implicitly within the null space of the constraints.

This is the essence of the projected conjugate gradient approach. The idea is to define a projection operator  $P = Z(Z^T H Z)^{-1} Z^T$ , where  $H$  is some symmetric positive definite matrix.

This projection maps arbitrary vectors into the null space of the constraint matrix  $A$ . By doing so, we ensure that all search directions generated by the conjugate gradient iterations remain feasible with respect to the constraints. In practice, this means that feasibility is preserved automatically at each step, without requiring explicit enforcement.

Choosing an effective preconditioner is crucial. The matrix  $W_{zz} = Z^T H Z$  serves this role. Its effect is to cluster the eigenvalues of the transformed operator  $W_{zz}^{-1/2} (Z^T G Z) W_{zz}^{-1/2}$ . When eigenvalues are tightly clustered, conjugate gradients converge in far fewer iterations.

The ideal case, though usually impractical, would be to take  $W_{zz}$  exactly equal to  $Z^T G Z$ , which would reduce the problem to a trivial one-step solution. A key advantage of the projected approach is efficiency. We avoid forming  $Z$  explicitly, which can be extremely expensive for large systems. Instead, everything is expressed in terms of  $n$ -vectors such as  $x = Z x_Z + Y x_Y$ . The updates are done directly in the full space, but in such a way that the iterations live in the null space. This ensures both numerical stability and robustness, even when the constraint matrix  $A$  is ill-conditioned or the choice of basis  $Z$  is less than ideal.

# PART I. OPTIMIZATION: CLASSICAL APPROACHES

## (LECTURE 10)

Projected CG  
Method

Convex QPs

Active-Set  
Methods

Big M  
Method

Properties of  
Active-Set M.

Shpilev Petr Valerievich

Faculty of Mathematics and Mechanics, SPbU

September, 2025



Санкт-Петербургский  
государственный  
университет



30 || SPbU & HIT, 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

In this lecture, we explore advanced algorithms for solving convex and nonconvex quadratic programming (QP) problems. We begin with the projected conjugate gradient (CG) method, discussing its projection computation and the use of general preconditioners to enhance convergence. We then shift focus to optimality conditions for convex QPs and address challenges in nonconvex settings, including degeneracy. The lecture introduces active-set methods for convex QPs, with a detailed examination of primal active-set algorithms, including step computation, iteration, and termination criteria. We explore the concept of feasible and descent directions, as well as optimality checks and convergence analysis. Through practical examples, we demonstrate the application of primal active-set methods, discussing how to initialize feasible points and apply the Big M method. Finally, we cover the QR update for adding constraints, its relationship to the Hessian, and key properties of the active-set method to ensure reliable convergence.

**Algorithm: Projected CG**

```

1: Choose initial  $x$  with  $Ax = b$ ;
2: Compute  $r = Gx + c$ ,  $g = Pr$ ,  $d = -g$ ;
3: for iterations until  $r^TPr$  is smaller than a prescribed tolerance do
4:    $\alpha \leftarrow r^Tg/d^Tg$ ;
5:    $x \leftarrow x + \alpha d$ ;
6:    $r^+ \leftarrow r + \alpha Gd$ ;
7:    $g^+ \leftarrow Pr^+$ ;
8:    $\beta \leftarrow (r^+)^Tg^+/r^Tg$ ;
9:    $d \leftarrow -g^+ + \beta d$ ;
10:   $g \leftarrow g^+$ ,  $r \leftarrow r^+$ ;

```

- Preconditioned residual  $g^+ = Pr^+$  lies in null space of  $A$ , ensuring  $Ad = 0$  and  $Ax = b$ .
- Requires  $Z^TGZ$  and  $Z^THZ$  positive definite for well-defined iterations.
- Preconditioner  $H$ : e.g.,  $\text{diag}(|G_{ii}|)$ ,  $I$ , or block diagonal submatrix of  $G$ .

Projected CG iterates in null space for efficient QP solution.

**Projected CG Method**

**Convex QPs**

**Active-Set Methods**

**Big M Method**

**Properties of Active-Set M.**

**Comments**

The projected conjugate gradient algorithm is essentially a direct translation of the standard preconditioned conjugate gradient method into the setting where feasibility with respect to the constraints must always be maintained. The structure of the algorithm reflects this requirement. We begin with an initial point  $x$  that already satisfies the linear constraints  $Ax = b$ . This is crucial, because all subsequent iterates must remain feasible.

At each step, the residual  $r$  is computed as  $Gx + c$ , which represents the violation of the optimality conditions. However, rather than working with  $r$  directly, we project it into the null space using  $g = Pr$ . This projected residual serves as the effective search gradient, ensuring that all directions are feasible. The initial search direction is then taken as  $-g$ .

The iteration follows the usual conjugate gradient structure: a step size  $\alpha$  is chosen based on the ratio of  $r^Tg$  to  $d^Tg$ , the solution is updated by  $x + \alpha d$ , and the residual is correspondingly updated. The crucial difference lies in the computation of the new preconditioned residual  $g^+$ , which is again obtained by projecting the new residual  $r^+$  into the null space. The coefficient  $\beta$  is then computed in the familiar way, and the new search direction is formed so as to preserve conjugacy.

The important theoretical property here is that  $g^+$  always lies in the null space of  $A$ . This guarantees that both the search direction  $d$  and the iterates  $x$  remain feasible throughout the process. For well-defined behavior, we require that both  $Z^TGZ$  and  $Z^THZ$  are positive definite, ensuring that all denominators and projections are meaningful.

In practice, the choice of preconditioner  $H$  has a strong influence on performance. Simple options such as the identity matrix, a diagonal scaling with absolute values of  $G$ 's diagonal entries, or block diagonal approximations of  $G$  can already yield significant acceleration. Thus, the projected conjugate gradient method provides an efficient and robust way to solve quadratic programs with equality constraints by embedding feasibility directly into the iterative process.

Projected CG uses  $P = Z(Z^T H Z)^{-1} Z^T$ . For  $H = I$ , orthogonal projection is:

$$P_I = Z(Z^T Z)^{-1} Z^T = I - A^T(AA^T)^{-1}A.$$

Compute  $g^+ = P_I r^+$  in two ways:

- ▶ Normal equations: Solve  $AA^T v^+ = Ar^+$ , then  $g^+ = r^+ - A^T v^+$ , using Cholesky factorization of  $AA^T$ .
- ▶ Augmented system: Solve

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}$$

via symmetric indefinite factorization.

- ▶ No explicit  $Z$  needed; only matrix-vector products with  $P_I$ .
- ▶ Normal equations: efficient for well-conditioned  $AA^T$ .
- ▶ Augmented system: robust for large, sparse systems.

Projection computed efficiently without null-space basis  $Z$ .

2/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Projected CG Method

Convex QPs

Active-Set Methods

Big M Method

Properties of Active-Set M.



## Comments

The computation of the projection in the projected conjugate gradient method can be achieved without explicitly constructing the null-space basis. In the simplest case, when the preconditioning matrix  $H$  is equal to the identity, the projection operator reduces to an orthogonal projection. This projection can be written in two equivalent forms. The first uses the null-space matrix, but a more practical expression avoids it entirely, namely  $I - A^T(AA^T)^{-1}A$ . This formula shows that the projection can be realized using only operations involving  $A$  and its transpose.

There are two standard procedures to apply the projection to a residual vector. The first relies on solving the so-called normal equations. One computes  $Ar^+$ , solves the system  $AA^T v = Ar^+$ , and then recovers the projection as  $r^+ - A^T v$ . Since the matrix  $AA^T$  is symmetric and positive definite, a Cholesky factorization can be precomputed once and reused in each iteration, making this route efficient when the conditioning is reasonable.

The second approach is to work with the augmented system. In this case, one solves a larger symmetric indefinite system involving identity,  $A$ , and its transpose. This approach is more robust, especially when dealing with large sparse problems where direct use of the normal equations may lead to numerical instability.

The key message is that the projection can be computed entirely implicitly. There is no need to form the basis matrix  $Z$ , nor to handle it explicitly. Only matrix–vector products with  $A$  and the corresponding factorizations are required, which makes the method suitable for practical large-scale constrained optimization.



Compute  $g^+ = \Pr^+$  with  $P = H^{-1}(I - A^T(AH^{-1}A^T)^{-1}AH^{-1})$  when  $H$  is nonsingular, or solve (when  $z^T H z \neq 0$  for all nonzero  $z$  with  $Az = 0$ ):

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}.$$

- ▶ Constraint preconditioner ( $H \neq G$ ) requires  $Z^T H Z$  positive definite.
- ▶ Ideal:  $H = G$ .
- ▶ Initial point  $x$  with  $Ax = b$  via:

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \text{ or } \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

- ▶ No explicit  $Z$ ; uses factorizations of  $AA^T$  or system matrices.
- ▶ Backsolve computes  $x$  efficiently from projection factorizations.
- ▶ Iterative refinement mitigates round-off errors in  $g^+$ .

General preconditioner enables robust, implicit projection.

3/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

The use of a general preconditioner greatly expands the flexibility of the projected conjugate gradient framework. In this setting, the projection operator is defined with the preconditioner matrix  $H$ , which is assumed to be nonsingular. The expression for the projection then becomes  $H^{-1}(I - A^T(AH^{-1}A^T)^{-1}AH^{-1})$ . Although this formula appears complicated, in practice the projection is again never formed explicitly. Instead, one applies it by solving linear systems.

A robust way to implement the projection is to solve the augmented saddle-point system consisting of the block matrix with  $H$  and  $A^T$  in the first row and  $A$  with zero in the second row. The unknowns are the projected vector and an auxiliary multiplier. This formulation is attractive because it naturally enforces the constraints and incorporates the preconditioner in a symmetric and stable way.

It is important that the reduced matrix  $Z^T H Z$  be positive definite, which ensures that the method works correctly. The ideal choice is to set  $H = G$ , the Hessian of the quadratic objective. In that case, the projection aligns perfectly with the natural metric of the problem, and convergence is typically accelerated.

In addition, the same factorization used for projection can be exploited to compute a feasible initial point. One can solve either the identity-based augmented system or the  $H$ -based version with the right-hand side containing the constraint vector  $b$ . This approach allows one to find a starting  $x$  that already satisfies the equality constraints.

Finally, iterative refinement can be applied to improve numerical stability, correcting small errors that accumulate in the projection step and ensuring accurate progress of the algorithm.



For QPs with equality and inequality constraints:

- ▶ Active-set (1970s): Suits small/medium problems, detects unboundedness/infeasibility, estimates optimal active set.
- ▶ Interior-point (1990s): Ideal for large problems, less suited for related QPs.
- ▶ Gradient projection: Efficient for bound-constrained QPs.

## Definition: Active Set

The active set at a solution  $x^*$  is the following set:  $\mathcal{A}(x^*) = \{i \in \mathcal{E} \cup \mathcal{I} \mid a_i^T x^* = b_i\}$ .

Lagrangian:  $\mathcal{L}(x, \lambda) = \frac{1}{2}x^T Gx + x^T c - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i (a_i^T x - b_i)$ .

KKT conditions at  $x^*$  with  $\lambda_i^*$ :  $Gx^* + c - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i^* a_i = 0$ ,

$$a_i^T x^* = b_i, \text{ for } i \in \mathcal{A}(x^*),$$

$$a_i^T x^* > b_i, \text{ for } i \in \mathcal{I} \setminus \mathcal{A}(x^*),$$

$$\lambda_i^* \geq 0, \quad \text{for } i \in \mathcal{I} \cap \mathcal{A}(x^*),$$

$$\lambda_i^* (a_i^T x^* - b_i) = 0, \text{ for } i \in \mathcal{I} \cup \mathcal{E}.$$

## Comments

Quadratic programming with both equality and inequality constraints has motivated the development of several algorithmic families, each with strengths in particular regimes. One of the earliest approaches is the active-set method, originating in the 1970s. This strategy is well suited for small to medium sized problems. Its main advantage lies in its ability to detect infeasibility and unboundedness while systematically building an estimate of the active set of constraints at the solution. By maintaining and updating this active set, the method effectively reduces the problem to a sequence of equality-constrained quadratic programs.

In contrast, interior-point methods became dominant in the 1990s. They are particularly effective for very large and sparse quadratic programs. Their power comes from a polynomial complexity bound and robustness in treating large-scale linear constraints. However, their drawback is that they are less convenient when many related quadratic programs must be solved, since each new instance may require a fresh factorization.

Another important class is gradient projection methods, which are especially efficient for bound-constrained problems. In these cases, the constraints simply restrict variables to lie within intervals, and projection onto the feasible set can be done by simple clipping operations. This makes such methods lightweight and scalable.

The concept of the active set plays a central role. At an optimal solution, the active set is defined as the set of all equality constraints together with those inequality constraints that are satisfied at equality. With this definition, the Karush–Kuhn–Tucker conditions can be stated compactly. The Lagrangian is formed from the quadratic objective and a weighted combination of all constraints. The KKT conditions then characterize stationarity, primal feasibility, and dual feasibility, and for inequality constraints they include the nonnegativity of the associated multipliers and complementarity.



KKT conditions for QP hold without LICQ, e.g., under linear constraints, satisfied in quadratic programming. For convex QP ( $G$  positive semidefinite), KKT conditions are sufficient for global optimality.

## Theorem 37 (Sufficiency for Convex QP)

If  $x^*$  satisfies KKT conditions with  $\lambda_i^*$ ,  $i \in \mathcal{A}(x^*)$ , and  $G$  is positive semidefinite, then  $x^*$  is a global solution of the QP.

**Proof:** For any feasible  $x$ ,  $a_i^T(x - x^*) = 0$  for  $i \in \mathcal{E}$ ,  $a_i^T(x - x^*) \geq 0$  for  $i \in \mathcal{A}(x^*) \cap \mathcal{I}$ . Thus:

$$(x - x^*)^T(Gx^* + c) = \sum_{i \in \mathcal{E}} \lambda_i^* a_i^T(x - x^*) + \sum_{i \in \mathcal{A}(x^*) \cap \mathcal{I}} \lambda_i^* a_i^T(x - x^*) \geq 0.$$

Hence:

$$q(x) = q(x^* + (x - x^*)) = q(x^*) + (x - x^*)^T(Gx^* + c) + \frac{1}{2}(x - x^*)^T G(x - x^*) \geq q(x^*) + \frac{1}{2}(x - x^*)^T G(x - x^*) \geq q(x^*),$$

since  $G$  is positive semidefinite, proving  $x^*$  is a global solution.  $\square$

**Note:** KKT conditions ensure global optimality for convex QPs.

5/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

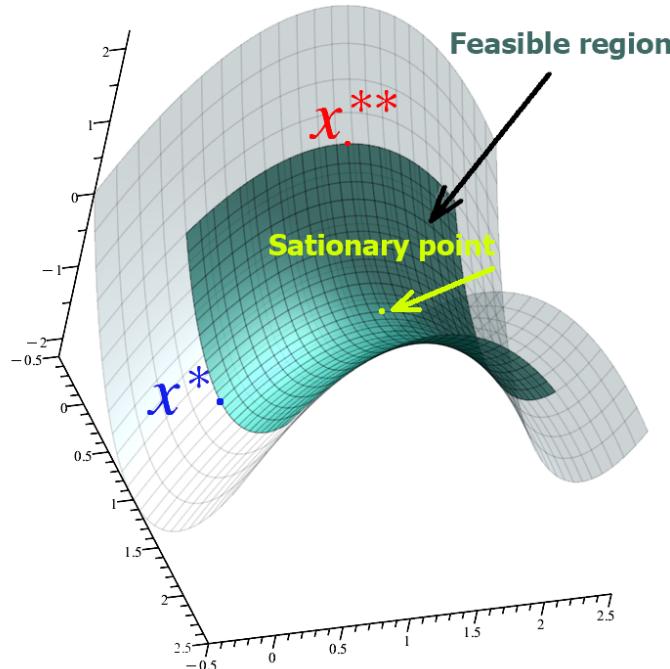
### Comments

The Karush–Kuhn–Tucker conditions serve as the fundamental optimality criterion for quadratic programming. Remarkably, in the quadratic case these conditions hold under very mild assumptions, even without requiring the linear independence constraint qualification. For convex quadratic programs, where the Hessian matrix  $G$  is positive semidefinite, the KKT conditions are not only necessary but also sufficient for global optimality.

Let's prove this theorem. Suppose a candidate point  $x^*$  and multipliers  $\lambda^*$  satisfy the KKT system. Consider any other feasible point  $x$ . For each equality constraint, the inner product of its coefficient vector with the difference  $x - x^*$  vanishes. For each inequality that is active at  $x^*$ , the corresponding multiplier is nonnegative, and the inner product with  $x - x^*$  is also nonnegative. Combining these relations shows that the product of  $x - x^*$  with the gradient of the objective at  $x^*$  is nonnegative.

Next, expand the quadratic objective at  $x^* + (x - x^*)$ . By direct substitution we immediately obtain that the value  $q(x)$  equals  $q(x^*)$  plus the inner product of  $x - x^*$  with the gradient at  $x^*$  plus  $\frac{1}{2}(x - x^*)^T G(x - x^*)$ . Because the Hessian is positive semidefinite, the last term is always nonnegative. Since the linear term was also shown to be nonnegative, the entire expression is greater than or equal to  $q(x^*)$ . This inequality demonstrates that  $x^*$  achieves the minimum among all feasible points, and therefore is a global solution. The theorem is proved.

Thus, in convex quadratic programming the KKT conditions not only describe optimality but actually guarantee it, providing a clean and powerful characterization of solutions.



**Figure:** Here we have plotted the feasible region and the graph of a quadratic objective  $q(x)$  in which  $G$  has one positive and one negative eigenvalue. Note that  $x^{**}$  is a local maximizer,  $x^*$  a local minimizer, and the center of the region is a stationary point.



## Comments

Quadratic programming becomes much more intricate once convexity is lost. In a convex problem, the Hessian matrix  $G$  is positive semidefinite, guaranteeing that every local minimizer is also a global minimizer. But in the nonconvex case,  $G$  has at least one negative eigenvalue, and the landscape of the objective function becomes significantly more complicated. A single stationary point is no longer sufficient to characterize the solution, because stationary points may correspond to local minima, local maxima, or even saddle points.

To illustrate, imagine a quadratic function of two variables where the Hessian has one positive and one negative eigenvalue. In this case the graph of the function forms a saddle-shaped surface. At certain feasible points, such as  $x^*$ , the point is indeed a local minimizer, while at another feasible point, say  $x^{**}$ , the point turns out to be a local maximizer. The very center of the feasible region may even be a stationary point without being an extremum.

This lack of convexity raises several challenges. First, optimality conditions such as the Karush–Kuhn–Tucker system no longer guarantee global solutions. They can only identify stationary points, and further analysis is required to classify their nature. Second, algorithmic strategies like active-set or interior-point methods may converge to local minima that are not globally optimal. Finally, the geometry of feasible regions interacts with the curvature of the quadratic objective in subtle ways, often making global optimization extremely difficult.

For these reasons, nonconvex quadratic programming is generally considered computationally hard. Specialized algorithms exist, for example branch-and-bound or cutting-plane techniques, but they are far less efficient than methods available in the convex case. This contrast underlines why convexity is such a central property in optimization theory.



Degeneracy occurs when:

- ▶ Active constraint gradients  $a_i$ ,  $i \in \mathcal{A}(x^*)$ , are linearly dependent.
- ▶ Strict complementarity fails:  $\exists i \in \mathcal{A}(x^*) \cap \mathcal{I}$  with  $\lambda_i^* = 0$  (weakly active constraint).

Example:  $\min x_1^2 + (x_2 + 1)^2$  s.t.  $x \geq 0$ , solution  $x^* = 0$ , both constraints active,  $\lambda_1^* = 0$ , violating strict complementarity.

- ▶ Linear dependence causes rank-deficient matrices, complicating step computation.
- ▶ Weakly active constraints lead to zigzagging in active-set/gradient projection methods.
- ▶ Safeguards prevent algorithmic oscillation near weakly active constraints.

Degeneracy challenges numerical stability and constraint identification.

7/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Another important phenomenon in quadratic programming is degeneracy. Degeneracy occurs when the structure of the active set at a solution point prevents algorithms from behaving in a straightforward way. There are two primary causes of degeneracy.

The first arises when the gradients of the active constraints are linearly dependent. At the solution, the active set is defined by equality constraints and those inequalities that are tight. If the associated gradient vectors are linearly dependent, then the matrix used in computing search directions becomes rank-deficient, and algorithms may struggle to define unique steps.

The second source of degeneracy is the failure of strict complementarity. Strict complementarity means that every active inequality constraint at the solution has a strictly positive multiplier. If instead some active inequality has multiplier equal to zero, the constraint is only weakly active. This situation leads to ambiguous behavior: algorithms such as active-set or gradient projection methods may oscillate near weakly active constraints, producing a zigzagging path instead of steady convergence.

A concrete example makes this clear. Consider minimizing  $x_1^2 + (x_2 + 1)^2$  subject to nonnegativity of both variables. The solution occurs at the origin, where both constraints are active. However, the multiplier for the first constraint is zero, violating strict complementarity.

Degeneracy has practical consequences. It complicates the identification of the correct active set, and it challenges numerical stability, especially when iterative updates require solving ill-conditioned systems. Modern implementations therefore incorporate safeguards to detect and control oscillations. Although degeneracy cannot be avoided in principle, careful algorithmic design ensures that optimization methods remain robust in its presence.



For convex QPs (G positive semidefinite), solve:

$$\min_{\mathbf{x}} q(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{G} \mathbf{x} + \mathbf{x}^T \mathbf{c} \quad \text{s.t.} \quad \mathbf{a}_i^T \mathbf{x} = b_i, i \in \mathcal{E}, \quad \mathbf{a}_i^T \mathbf{x} \geq b_i, i \in \mathcal{I}.$$

Active-set methods iteratively estimate optimal active set  $\mathcal{A}(\mathbf{x}^*)$ . Primal methods:

- ▶ Maintain feasible iterates, decrease  $q(\mathbf{x})$ .
- ▶ Solve subproblems with working set  $\mathcal{W}_k \subseteq \mathcal{E} \cup \mathcal{I}$  as equalities.
- ▶ Ensure  $\mathbf{a}_i, i \in \mathcal{W}_k$ , are linearly independent.
- ▶ Analogous to simplex method: Adjust  $\mathcal{W}_k$  using gradients and multipliers.
- ▶ Challenge: Identify  $\mathcal{A}(\mathbf{x}^*)$  without prior knowledge.
- ▶ Iterates may not be vertices, unlike linear programming.

Primal active-set methods iteratively refine  $\mathcal{W}_k$  for QP solutions.

8/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

In convex quadratic programming, where the Hessian matrix G is positive semidefinite, active-set methods form one of the most classical algorithmic approaches. The idea is to iteratively build an approximation of the optimal active set, denoted by  $\mathcal{A}(\mathbf{x}^*)$ . At each iteration, the algorithm selects a working set, denoted  $\mathcal{W}_k$ , which is a subset of the equality and inequality constraints. These constraints are treated as equalities, and the resulting subproblem is solved.

The working set must be chosen carefully. The gradients of its constraints, denoted by vectors  $\mathbf{a}_i$ , must be linearly independent to avoid rank deficiencies. Within this reduced system, the algorithm computes a feasible step that decreases the quadratic objective function. In this sense, the procedure resembles the simplex method for linear programming, where the basis of constraints is modified step by step to move closer to the optimum.

However, there are important differences compared to linear programming. In quadratic programming, iterates generated by active-set methods are not necessarily vertices of the feasible polyhedron. Instead, they may lie along edges or faces, reflecting the curved nature of the quadratic objective. The main challenge is that the optimal active set is unknown in advance, so the algorithm must discover it dynamically, adjusting the working set based on gradients and multipliers.

Despite these complications, active-set methods are particularly effective for small to medium-sized convex quadratic programs. They maintain feasibility of all iterates, provide systematic progress toward the solution, and offer a clear geometric interpretation. For these reasons, active-set strategies remain widely studied and applied, especially when reliability and interpretability are valued.



**Key Idea** Primal active-set methods iteratively refine working set  $\mathcal{W}_k$  to approximate optimal active set  $\mathcal{A}(x^*)$ .

Use  $\mathcal{W}_k \subseteq \mathcal{E} \cup \mathcal{I}$  at iterate  $x_k$ , with linearly independent gradients  $a_i$ ,  $i \in \mathcal{W}_k$ . If  $x_k$  is not optimal in  $\mathcal{W}_k$ , solve subproblem:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G x_k + c)^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k, \end{aligned}$$

where  $p = x - x_k$ , derived from  $q(x_k + p) = \frac{1}{2} p^T G p + (G x_k + c)^T p + \rho_k$ , with  $\rho_k = \frac{1}{2} x_k^T G x_k + c^T x_k$  which is independent of  $p$ .

- ▶ Subproblem imposes  $\mathcal{W}_k$  as equalities, ignores other constraints.
- ▶ Linear independence of  $a_i$  ensures solvable subproblem.
- ▶ Step  $p$  reduces  $q(x)$  while maintaining feasibility.

Working set  $\mathcal{W}_k$  guides feasible steps in QP optimization.

9/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The primal active-set method refines the working set in a systematic way, making it a powerful tool for solving convex quadratic programs. At a given iterate, denoted  $x_k$ , the algorithm maintains a working set  $\mathcal{W}_k$  consisting of linearly independent constraints. If  $x_k$  is not yet optimal with respect to this working set, a subproblem is solved to find a feasible step.

This subproblem is defined in terms of the search direction  $p$ , where  $p = x - x_k$ . The objective of the subproblem is  $\frac{1}{2} p^T G p + (G x_k + c)^T p$ . The constraints of the subproblem enforce that  $a_i^T p = 0$  for every constraint in the working set. This construction ensures that the step  $p$  respects the currently assumed active constraints.

By definition, the quadratic objective function at  $x_k + p$  equals  $\frac{1}{2} p^T G p + (G x_k + c)^T p + \rho_k$ , where  $\rho_k$  represents the part of the objective that depends only on  $x_k$ . Since  $\rho_k$  is independent of  $p$ , the subproblem can be solved by focusing only on the first two terms.

The step  $p$  computed in this manner reduces the value of the objective function while maintaining feasibility with respect to the active set. If the step remains feasible for all constraints, it is accepted; otherwise, the working set is updated by adding or removing constraints. Through repeated iterations, the algorithm converges toward the true optimal active set and thereby finds the solution of the quadratic program.



Solve QP subproblem for step  $p_k$ :

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (Gx_k + c)^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

For  $i \in \mathcal{W}_k$ , constraints remain satisfied:  $a_i^T(x_k + \alpha_k p_k) = a_i^T x_k = b_i$ .

Update:

$$x_{k+1} = x_k + \alpha_k p_k, \quad \alpha_k \in [0, 1],$$

where  $\alpha_k \stackrel{\text{def}}{=} \min \left( 1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right)$ .

- ▶  $p_k \neq 0$  solved using equality-constrained QP techniques ( $G$  positive definite).
- ▶  $\alpha_k$  ensures feasibility for  $i \notin \mathcal{W}_k$  when  $a_i^T p_k < 0$  (since the inequality holds:  $a_i^T(x_k + \alpha_k p_k) \geq b_i$ ).
- ▶ Constraints with  $a_i^T p_k \geq 0$  for some  $i \notin \mathcal{W}_k$  remain satisfied for  $\alpha_k \geq 0$ .

Step  $p_k$  and  $\alpha_k$  maintain feasibility and reduce  $q(x)$ .

10/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

In primal active-set methods, each iteration begins by solving a quadratic programming subproblem to compute a search direction, denoted  $p_k$ . The subproblem minimizes  $\frac{1}{2} p^T G p + (Gx_k + c)^T p$ , subject to the constraints in the current working set  $\mathcal{W}_k$  treated as equalities. This guarantees that all constraints currently assumed active remain satisfied along the step. Specifically, for each  $i \in \mathcal{W}_k$ ,  $a_i^T(x_k + \alpha_k p_k) = b_i$ .

The computed step  $p_k$  is then scaled by a step length  $\alpha_k$ . If taking the full step preserves feasibility for all constraints,  $\alpha_k$  is set to 1. Otherwise,  $\alpha_k$  is determined by the most restrictive constraint outside the working set. In particular, for each  $i \notin \mathcal{W}_k$  where  $a_i^T p_k$  is negative,  $\alpha_k$  is taken as the minimum of the ratio of  $b_i - a_i^T x_k$  over  $a_i^T p_k$ . This ensures that no constraint is violated along the step.

This mechanism guarantees that every iterate remains feasible while ensuring progress in reducing the objective function  $q(x)$ . If a constraint is already satisfied and its  $a_i^T p_k$  is nonnegative, it remains satisfied for any nonnegative  $\alpha_k$ . By combining  $p_k$  and  $\alpha_k$  in this way, the algorithm moves along a feasible path in the search space, systematically decreasing the quadratic objective while maintaining all required equalities and inequalities.



Constraints  $i \notin \mathcal{W}_k$  for which step-length  $\alpha_k$  is minimized are called *blocking constraints*.

If  $\alpha_k = 1$  and no new constraints active at  $x_k + \alpha_k p_k$ , there are no blocking constraints.

If  $\alpha_k < 1$ , add one of the blocking constraints to  $\mathcal{W}_{k+1}$ .

Note:  $\alpha_k = 0$  possible if a constraint  $i$  is active at  $x_k$ ,  $i \notin \mathcal{W}_k$ .

Iterate by adding constraints to  $\mathcal{W}_k$  until a point  $\hat{x}$  is reached where  $p = 0$  is the solution to the QP subproblem.

- This means  $\hat{x}$  minimizes  $q(x)$  over  $\hat{\mathcal{W}}$  and the first KKT optimality condition for this subproblem is satisfied:

$$G\hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i \quad (\text{for some Lagrange multipliers } \hat{\lambda}_i)$$

since we set Lagrange multipliers for constraints not in  $\hat{\mathcal{W}}$  to zero.

- The step length  $\alpha_k$  ensures feasibility, so  $\hat{x}$  also satisfies the second and the third KKT conditions as well.

11/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Once a candidate solution  $\hat{x}$  is obtained, its optimality is verified by examining the signs of the Lagrange multipliers associated with inequality constraints in the working set, denoted  $\hat{\lambda}_i$  for  $i \in \hat{\mathcal{W}} \cap \mathcal{I}$ . If all of these multipliers are nonnegative, the fourth KKT condition is satisfied, indicating that  $\hat{x}$  is a KKT point. Because  $G$  is positive semidefinite,  $\hat{x}$  is then a global solution. If  $G$  is strictly positive definite, the global solution is unique.

If, however, any  $\hat{\lambda}_j$  is negative for  $j \in \hat{\mathcal{W}} \cap \mathcal{I}$ , the fourth KKT condition fails. In this case, the algorithm removes one of these constraints from the working set and resolves the QP subproblem for a new search direction  $p$ . This new step is feasible with respect to the dropped constraint, ensuring progress while maintaining feasibility for all remaining constraints.

Throughout the iterations, linear independence of the gradients  $a_i$  for  $i \in \mathcal{W}_k$  is maintained. This is critical because it guarantees that the equality-constrained subproblem remains solvable. By continually adjusting  $\mathcal{W}_k$  based on the signs of Lagrange multipliers, the algorithm navigates toward a globally optimal solution. The process balances between expanding the active set when a blocking constraint appears and contracting it when a negative multiplier indicates the corresponding constraint is unnecessarily restrictive.

In this way, the primal active-set method converges to a feasible point where all KKT conditions are satisfied. It provides a clear and constructive approach to convex quadratic programming, combining step computation, feasibility enforcement, and multiplier-based checks into a coherent algorithmic framework.

Check signs of Lagrange multipliers  $\hat{\lambda}_i$  for inequality constraints  $i \in \hat{\mathcal{W}} \cap \mathcal{I}$  at  $\hat{x}$ .

- ▶ If all  $\hat{\lambda}_i \geq 0$ , the fourth KKT condition is satisfied, so  $\hat{x}$  is a KKT point.
- ▶ Since  $G$  is positive semidefinite,  $\hat{x}$  is a global solution. If  $G$  is positive definite,  $\hat{x}$  is the unique global solution (by Theorem 37).

If any  $\hat{\lambda}_j < 0$  for  $j \in \hat{\mathcal{W}} \cap \mathcal{I}$ , the fourth KKT condition fails.

- ▶ Remove one such  $j$  from  $\mathcal{W}_{k+1}$  and solve a new QP subproblem for the next step  $p$ .
- ▶ The new step  $p$  is feasible for the dropped constraint.

Constraint gradients  $a_i$ ,  $i \in \mathcal{W}_k$ , are linearly independent (maintained throughout iterations).

Adjust  $\mathcal{W}_k$  based on  $\hat{\lambda}_i$  signs to find global solution  $\hat{x}$ .



## Comments

Once a candidate solution  $\hat{x}$  is obtained, its optimality is verified by examining the signs of the Lagrange multipliers associated with inequality constraints in the working set, denoted  $\hat{\lambda}_i$  for  $i \in \hat{\mathcal{W}} \cap \mathcal{I}$ . If all of these multipliers are nonnegative, the fourth KKT condition is satisfied, indicating that  $\hat{x}$  is a KKT point. Because  $G$  is positive semidefinite,  $\hat{x}$  is then a global solution. If  $G$  is strictly positive definite, the global solution is unique.

If, however, any  $\hat{\lambda}_j < 0$  for  $j \in \hat{\mathcal{W}} \cap \mathcal{I}$ , the fourth KKT condition fails. In this case, the algorithm removes one of these constraints from the working set and resolves the QP subproblem for a new search direction  $p$ . This new step is feasible with respect to the dropped constraint, ensuring progress while maintaining feasibility for all remaining constraints.

Throughout the iterations, linear independence of the gradients  $a_i$  for  $i \in \mathcal{W}_k$  is maintained. This is critical because it guarantees that the equality-constrained subproblem remains solvable. By continually adjusting  $\mathcal{W}_k$  based on the signs of Lagrange multipliers, the algorithm navigates toward a globally optimal solution. The process balances between expanding the active set when a blocking constraint appears and contracting it when a negative multiplier indicates the corresponding constraint is unnecessarily restrictive.

In this way, the primal active-set method converges to a feasible point where all KKT conditions are satisfied. It provides a clear and constructive approach to convex quadratic programming, combining step computation, feasibility enforcement, and multiplier-based checks into a coherent algorithmic framework.



### Theorem 38: Feasible direction

Suppose that the point  $\hat{x}$  satisfies first-order conditions for the equality-constrained subproblem with working set  $\hat{\mathcal{W}}$ ; that is,

$$G\hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i$$

is satisfied along with  $a_i^T \hat{x} = b_i$  for all  $i \in \hat{\mathcal{W}}$ . Suppose, too, that the constraint gradients  $a_i$ ,  $i \in \hat{\mathcal{W}}$ , are linearly independent and that there is an index  $j \in \hat{\mathcal{W}}$  such that  $\hat{\lambda}_j < 0$ . Let  $p$  be the solution obtained by dropping the constraint  $j$  and solving the following subproblem:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G\hat{x} + c)^T p, \\ \text{subject to} \quad & a_i^T p = 0, \quad \text{for all } i \in \hat{\mathcal{W}} \text{ with } i \neq j. \end{aligned}$$

Then  $p$  is a feasible direction for constraint  $j$ , that is,  $a_j^T p \geq 0$ . Moreover, if  $p$  satisfies second-order sufficient conditions for the subproblem, then we have that  $a_j^T p > 0$ , and that  $p$  is a descent direction for  $q(\cdot)$ .

13/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

The concept of feasible directions plays an important role in refining the working set. Suppose a candidate point  $\hat{x}$  satisfies the first-order conditions for the equality-constrained subproblem defined by the working set  $\hat{\mathcal{W}}$ , and that the constraint gradients  $a_i$  for  $i \in \hat{\mathcal{W}}$  are linearly independent. If a multiplier  $\hat{\lambda}_j$  associated with some constraint  $j \in \hat{\mathcal{W}}$  is negative, the algorithm can generate a feasible direction by temporarily dropping constraint  $j$  and solving a modified subproblem.

The subproblem minimizes  $\frac{1}{2} p^T G p + (G\hat{x} + c)^T p$ , subject to  $a_i^T p = 0$  for all  $i \in \hat{\mathcal{W}}$  excluding  $j$ . The resulting vector  $p$  satisfies  $a_j^T p \geq 0$ , making it a feasible direction for the previously active constraint. If second-order sufficient conditions hold, then  $a_j^T p > 0$ , meaning that moving along  $p$  increases satisfaction of the previously restrictive constraint and reduces the quadratic objective.

This property guarantees that the working set can be adjusted systematically. Negative multipliers indicate which constraints are too restrictive, while feasible directions allow the algorithm to make progress without violating any other active constraints. The step along  $p$  is both feasible and a descent direction, ensuring that the objective function decreases. This mechanism forms the foundation for the removal of constraints with negative multipliers and helps drive the iteration toward the global optimum.

## Proof:

Since  $p$  solves the subproblem:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G \hat{x} + c)^T p, \\ \text{s.t.} \quad & a_i^T p = 0, \quad \forall i \in \hat{\mathcal{W}}, i \neq j, \end{aligned}$$

there exist multipliers  $\tilde{\lambda}_i$ , for all  $i \in \hat{\mathcal{W}}$ ,  $i \neq j$ , such that:

$$\sum_{i \in \hat{\mathcal{W}}, i \neq j} \tilde{\lambda}_i a_i = Gp + (G \hat{x} + c).$$

- ▶ By second-order necessary conditions, if  $Z$  is a null-space basis for the matrix  $[a_i^T]_{i \in \hat{\mathcal{W}}, i \neq j}$ , then  $Z^T G Z$  is positive semidefinite.
- ▶ Since  $p = Zp_z$  for some  $p_z$ , it follows that  $p^T G p \geq 0$ .
- ▶ Subtracting  $G \hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i$  from the above, we get:

$$\sum_{i \in \hat{\mathcal{W}}, i \neq j} (\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = Gp.$$

Projected CG Method
Convex QPs
Active-Set Methods
Big M Method
Properties of Active-Set M.



## Comments

To justify that dropping a constraint with a negative multiplier produces a feasible direction, consider the subproblem solved for  $p$ . This subproblem minimizes  $\frac{1}{2} p^T G p + (G \hat{x} + c)^T p$ , subject to all constraints in  $\hat{\mathcal{W}}$  except the dropped index  $j$  treated as equalities. By optimality of this subproblem, there exist Lagrange multipliers, denoted  $\tilde{\lambda}_i$  for  $i \in \hat{\mathcal{W}}$  excluding  $j$ , such that the sum over  $i$  of  $\tilde{\lambda}_i a_i$  equals  $Gp + G \hat{x} + c$ .

Applying second-order necessary conditions, let  $Z$  be a basis for the null space of the matrix formed by  $a_i^T$  for  $i \in \hat{\mathcal{W}}$  excluding  $j$ . Then  $Z^T G Z$  is positive semidefinite. Since  $p$  lies in the column space of  $Z$ , it follows that  $p^T G p \geq 0$ . Subtracting the first-order condition at  $\hat{x}$ , namely  $G \hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i$ , we obtain the equation: the sum over  $i \in \hat{\mathcal{W}}$  excluding  $j$  of  $(\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = Gp$ . This establishes a foundation for showing that  $a_j^T p \geq 0$ , proving that  $p$  is a feasible direction for constraint  $j$ .

## Primal Active-Set Methods: Proof of Feasible Direction (Cont.)

By taking inner products of both sides with  $p$  and using  $a_i^T p = 0$  for all  $i \in \hat{\mathcal{W}}$ ,  $i \neq j$ , we have:

$$-\hat{\lambda}_j a_j^T p = p^T G p.$$

Since  $p^T G p \geq 0$  and  $\hat{\lambda}_j < 0$ , it follows that  $a_j^T p \geq 0$ .

- ▶ If second-order sufficient conditions hold,  $Z^T G Z$  is positive definite. Then  $a_j^T p = 0$  only if  $p^T G p = p_z^T Z^T G Z p_z = 0$ , so  $p_z = 0$  and  $p = 0$ .
- ▶ If  $p = 0$ , then  $\sum_{i \in \hat{\mathcal{W}}, i \neq j} (\tilde{\lambda}_i - \hat{\lambda}_j) a_i - \hat{\lambda}_j a_j = 0$ . By linear independence of  $a_i$ ,  $i \in \hat{\mathcal{W}}$ , we get  $\hat{\lambda}_j = 0$ , a contradiction.
- ▶ Thus,  $p^T G p > 0$ , so  $a_j^T p > 0$  when second-order sufficient conditions hold.

The claim that  $p$  is a descent direction for  $q(\cdot)$  will be proved further.  $\square$

While any  $j$  with  $\hat{\lambda}_j < 0$  yields a direction  $p$  for progress, we often choose the most negative  $\hat{\lambda}_j$ .

- ▶ Sensitivity analysis shows the decrease in  $q$  is proportional to  $|\hat{\lambda}_j|$ .
- ▶ The step along  $p$  may be short if blocked by a new constraint, so the decrease in  $q$  is not guaranteed to be maximal.

15/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Projected CG Method
Convex QPs
Active-Set Methods
Big M Method
Properties of Active-Set M.



### Comments

Taking the inner product of both sides of the equation with  $p$  and using the fact that  $a_i^T p = 0$  for  $i \in \hat{\mathcal{W}}$  excluding  $j$  gives  $-\hat{\lambda}_j a_j^T p = p^T G p$ . Because  $p^T G p$  is nonnegative and  $\hat{\lambda}_j$  is negative, it follows that  $a_j^T p \geq 0$ . This confirms that  $p$  points in a feasible direction for constraint  $j$ .

If second-order sufficient conditions hold,  $Z^T G Z$  is positive definite. Then  $a_j^T p = 0$  only if  $p^T G p = 0$ , which implies that  $p$  itself is zero. But if  $p = 0$ , linear independence of the  $a_i$  implies  $\hat{\lambda}_j$  must be zero, contradicting the assumption that it is negative. Hence,  $p^T G p > 0$ , and  $a_j^T p > 0$  as well.

In practice, when several multipliers are negative, the most negative  $\hat{\lambda}_j$  is often chosen. Sensitivity analysis shows that the decrease in the quadratic objective  $q$  is approximately proportional to the magnitude of  $\hat{\lambda}_j$ . The step along  $p$  may be shortened if blocked by other constraints, so the actual decrease in  $q$  may not reach its maximum possible value. Nevertheless, the direction is guaranteed to be feasible and provides systematic progress toward the optimum.



Consider the subproblem:

$$\min_p \frac{1}{2} p^T Gp + (Gx_k + c)^T p, \text{ s.t. } a_i^T p = 0, \forall i \in \mathcal{W}_k, \quad (*)$$

## Theorem 39

Suppose that the solution  $p_k$  of the subproblem (\*) is nonzero and satisfies the second-order sufficient conditions. Then  $q(\cdot)$  is strictly decreasing along  $p_k$ .

### Proof:

Since  $p_k$  satisfies second-order conditions,  $Z^T GZ$  is positive definite, where  $Z$  is a basis of the null space of  $[a_i^T]_{i \in \mathcal{W}_k}$ .

- ▶ Thus,  $p_k$  is the unique global solution of the subproblem.
- ▶ Since  $p = 0$  is feasible, its objective value  $\frac{1}{2} p^T Gp + (Gx_k + c)^T p$  at  $p = 0$  is larger than at  $p_k$ , so:

$$\frac{1}{2} p_k^T Gp_k + (Gx_k + c)^T p_k < 0.$$

- ▶ Since  $p_k^T Gp_k \geq 0$ , we have  $(Gx_k + c)^T p_k < 0$ , so:

$$q(x_k + \alpha_k p_k) = q(x_k) + \alpha_k (Gx_k + c)^T p_k + \frac{1}{2} \alpha_k^2 p_k^T Gp_k < q(x_k),$$

for all  $\alpha_k > 0$  sufficiently small. □

## Comments

A key property of the subproblem solution  $p_k$  is that it produces a descent direction for the quadratic objective  $q$  when it is nonzero and satisfies second-order sufficient conditions. The subproblem minimizes  $\frac{1}{2} p^T Gp + (Gx_k + c)^T p$ , subject to  $a_i^T p = 0$  for all  $i$  in the working set  $\mathcal{W}_k$ . Let  $Z$  denote a basis for the null space of the matrix formed by  $a_i^T$ . Second-order sufficient conditions ensure that  $Z^T GZ$  is positive definite.

Since  $p_k$  lies in the column space of  $Z$ , it is the unique minimizer of the subproblem. Comparing the objective at  $p = 0$ , which is feasible, we see that the value at  $p_k$  is strictly smaller. This implies that the inner product of  $Gx_k + c$  with  $p_k$  is negative. Therefore, for any sufficiently small positive  $\alpha_k$ ,  $q(x_k + \alpha_k p_k) = q(x_k) + \alpha_k (Gx_k + c)^T p_k + \frac{1}{2} \alpha_k^2 p_k^T Gp_k$ , which is strictly less than  $q(x_k)$ .

This proves that  $p_k$  is a descent direction for the quadratic function. The combination of feasible steps and guaranteed descent is crucial for the convergence of the active-set method. It ensures that each iteration either reduces the objective or adjusts the working set in a well-defined manner, systematically moving toward the optimal solution while maintaining feasibility.

# Primal Active-Set Methods: Algorithm

## Algorithm: Active-Set Method for Convex QP

```
1: Compute a feasible starting point  $x_0$ ;  
2: Set  $\mathcal{W}_0$  to a subset of active constraints at  $x_0$ ;  
3: for  $k = 0, 1, 2, \dots$  do  
4:   Solve (*) to find  $p_k$ ;  
5:   if  $p_k = 0$  then compute  $\hat{\lambda}_i$  satisfying  $Gx_k + c = \sum_{i \in \mathcal{W}_k} \hat{\lambda}_i a_i$ ;  
6:   if  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{I}$  then stop with solution  $x^* = x_k$ ;  
7:   else  
8:      $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$ ;  
9:      $x_{k+1} \leftarrow x_k$ ;  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;  
10:  else ( $p_k \neq 0$ )  
11:    Compute step length  $\alpha_k$ ;  
12:     $x_{k+1} \leftarrow x_k + \alpha_k p_k$ ;  
13:    if there are blocking constraints then  
14:       $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{\text{one blocking constraint}\}$ ;  
15:    else  
16:       $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$ ;  
17:    end (for)
```

Projected CG Method
Convex QPs
Active-Set Methods
Big M Method
Properties of Active-Set M.



17/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The primal active-set algorithm for convex quadratic programming begins by computing a feasible starting point  $x_0$  and defining an initial working set  $\mathcal{W}_0$ , typically a subset of active constraints at  $x_0$ . At each iteration, the subproblem is solved to find a step  $p_k$ . If  $p_k = 0$ , the Lagrange multipliers  $\hat{\lambda}_i$  corresponding to constraints in  $\mathcal{W}_k$  are computed from  $Gx_k + c = \sum_{i \in \mathcal{W}_k} \hat{\lambda}_i a_i$ .

If all  $\hat{\lambda}_i$  for  $i \in \mathcal{W}_k \cap \mathcal{I}$  are nonnegative, the current point satisfies all KKT conditions and the algorithm terminates with  $x^* = x_k$ . If some  $\hat{\lambda}_j$  is negative, the corresponding constraint is removed from  $\mathcal{W}_k$ , and the next iteration begins with the same  $x_k$  but a reduced working set.

If  $p_k$  is nonzero, a step length  $\alpha_k$  is computed to maintain feasibility, and the next iterate  $x_{k+1}$  is  $x_k + \alpha_k p_k$ . If any new blocking constraints are encountered along this step, one of them is added to  $\mathcal{W}_{k+1}$ ; otherwise, the working set remains unchanged. Iterations continue in this manner until convergence.

This algorithm systematically alternates between adjusting the working set and computing feasible steps. The combination of step computation, Lagrange multiplier checks, and working set updates ensures that the method progresses toward a globally optimal solution, while maintaining feasibility and satisfying all KKT conditions. Its design guarantees convergence for convex quadratic programs, and it provides a clear operational framework for implementation.



## Phase 1: Initial Feasible Point

Given estimate  $\tilde{x}$ , solve the feasibility LP:

$$\begin{aligned} \min_{\{x,z\}} \quad & e^T z, \\ \text{s.t.} \quad & a_i^T x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \\ & a_i^T x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{I}, \\ & z \geq 0, \end{aligned}$$

where  $e = (1, 1, \dots, 1)^T$ ,  $\gamma_i = -\text{sign}(a_i^T \tilde{x} - b_i)$  for  $i \in \mathcal{E}$ ,  $\gamma_i = 1$  for  $i \in \mathcal{I}$ .

Initial point:

$$x = \tilde{x}, \quad z_i = |a_i^T \tilde{x} - b_i| \quad (i \in \mathcal{E}), \quad z_i = \max(b_i - a_i^T \tilde{x}, 0) \quad (i \in \mathcal{I}).$$

- If  $\tilde{x}$  is feasible for the QP  $\min_x \frac{1}{2} x^T G x + c^T x$  s.t.  $a_i^T x = b_i$ ,  $i \in \mathcal{E}$ ,  $a_i^T x \geq b_i$ ,  $i \in \mathcal{I}$ , then  $(\tilde{x}, 0)$  is optimal.
- If the QP has feasible points, the LP's optimal value is zero, yielding a feasible  $x$ .
- $\mathcal{W}_0$  for Active-Set Method: linearly independent subset of active constraints at the LP solution.

## Comments

When applying the primal active-set method, the first step is always to ensure that we have a feasible starting point. This is handled through what is called a phase one procedure. The idea is to take an initial estimate, denoted  $\tilde{x}$ , and to construct an auxiliary linear program that searches for feasibility. In this program we introduce slack variables, denoted  $z$ , which measure constraint violation. The linear objective minimizes the sum of all these violations, written as  $e^T z$ , where  $e$  is the vector of all ones. Each equality constraint is expressed as  $a_i^T x + \gamma_i z_i = b_i$ , and each inequality as  $a_i^T x + \gamma_i z_i \geq b_i$ , with  $z \geq 0$ . The coefficients  $\gamma_i$  are chosen to adjust the direction in which violation is measured: for equality constraints,  $\gamma_i = -\text{sign}(a_i^T \tilde{x} - b_i)$ , while for inequalities  $\gamma_i = 1$ .

If the initial estimate  $\tilde{x}$  already satisfies the quadratic program, then  $(\tilde{x}, 0)$  is optimal for this feasibility problem. If feasible points exist at all, the optimal value of this linear program will be zero, giving us a feasible  $x$  to start the quadratic method. The first working set  $\mathcal{W}_0$  is then chosen as a linearly independent subset of the constraints active at this feasible solution. This systematic approach guarantees that our algorithm always begins from a valid point.



"**Big M**" **Penalty Method** includes infeasibility measure  $\eta$  in the QP:

$$\begin{aligned} \min_{\{x, \eta\}} \quad & \frac{1}{2} x^T G x + x^T c + M\eta, \\ \text{s.t.} \quad & |(a_i^T x - b_i)| \leq \eta, \quad i \in \mathcal{E}, \\ & b_i - a_i^T x \leq \eta, \quad i \in \mathcal{I}, \\ & 0 \leq \eta, \end{aligned}$$

for some large  $M > 0$ . As before,  $\tilde{x}$  is the user-supplied initial guess.

- ▶ For feasible QP  $\min_x \frac{1}{2} x^T G x + c^T x$  s.t.  $a_i^T x = b_i$ ,  $i \in \mathcal{E}$ ,  $a_i^T x \geq b_i$ ,  $i \in \mathcal{I}$ , and large  $M$ , the solution has  $\eta = 0$ , with  $x$  solving the QP.
- ▶ Strategy: Choose  $M$ , solve the problem; if  $\eta > 0$ , increase  $M$  and retry.
- ▶  $\ell_1$ -norm variant:  $\min_{\{x, s, t, v\}} \frac{1}{2} x^T G x + x^T c + M e_{\mathcal{E}}^T (s + t) + M e_{\mathcal{I}}^T v$ ,

$$\begin{aligned} \text{s.t.} \quad & a_i^T x - b_i + s_i - t_i = 0, \quad i \in \mathcal{E}, \\ & a_i^T x - b_i + v_i \geq 0, \quad i \in \mathcal{I}, \\ & s \geq 0, t \geq 0, v \geq 0, \end{aligned}$$

$e_{\mathcal{E}}$  is the vector  $\langle 1, 1, \dots, 1 \rangle^T$  of length  $|\mathcal{E}|$ ; similarly for  $e_{\mathcal{I}}$ .

19/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

An alternative approach is the so-called big M penalty method. The principle here is to modify the quadratic program itself so that infeasibility is penalized directly in the objective. We introduce a nonnegative variable  $\eta$ , which measures the maximum violation across all constraints. The modified problem minimizes  $\frac{1}{2} x^T G x + c^T x + M\eta$ , subject to the condition that for each equality constraint the absolute value of  $a_i^T x - b_i$  is less or equal to  $\eta$ , and for each inequality  $b_i - a_i^T x$  is less or equal to  $\eta$ . The parameter  $M$  is a large positive constant, chosen so that feasibility is heavily favored. If the original quadratic program is feasible, the optimal solution of this penalized problem has  $\eta = 0$  and gives exactly the same  $x$  as the feasible quadratic program. If  $\eta > 0$ , one can increase the value of  $M$  and resolve until feasibility dominates.

A common variant is based on the one-norm of the violations, which replaces the single  $\eta$  with separate nonnegative variables. For equalities, violations are split into positive and negative parts, denoted  $s$  and  $t$ , so that  $a_i^T x - b_i + s_i - t_i = 0$ . For inequalities, slack variables  $v$  are added. This approach provides finer control and avoids masking different sources of infeasibility under a single  $\eta$ .

## Primal Active-Set Methods: Example

In the following example subscripts on  $x$  and  $p$  denote components (e.g.,  $x_1$ ), superscripts denote iteration (e.g.,  $x^4$ ).

### Example

Apply Active-Set Method to the problem:

$$\begin{aligned} \min_x \quad & q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2, \\ \text{s.t.} \quad & x_1 - 2x_2 + 2 \geq 0, \\ & -x_1 - 2x_2 + 6 \geq 0, \\ & -x_1 + 2x_2 + 2 \geq 0, \\ & x_1 \geq 0, \\ & x_2 \geq 0. \end{aligned}$$

Constraints are indexed 1 to 5.

- ▶ Feasible initial point:  $x^0 = (2, 0)^T$ .
- ▶ Constraints 3 and 5 are active at  $x^0$ , set  $\mathcal{W}_0 = \{3, 5\}$ .
- ▶ Other choices (e.g.,  $\mathcal{W}_0 = \{5\}$ ,  $\mathcal{W}_0 = \{3\}$ , or  $\mathcal{W}_0 = \emptyset$ ) are valid, affecting algorithm behavior.

20/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Projected CG Method
Convex QPs
Active-Set Methods
Big M Method
Properties of Active-Set M.



### Comments

Let's consider a simple quadratic program to see how the primal active-set method works in practice. The objective function is  $(x_1 - 1)^2 + (x_2 - 2.5)^2$ . Without constraints, the minimum would be at  $x = (1, 2.5)^T$ . But we impose five inequalities that form a polygonal feasible region in the first quadrant.

As the starting point, we take  $x^0 = (2, 0)^T$ . This point satisfies all constraints, so it is feasible. At this point, the third and the fifth constraints are active: the third because  $-x_1 + 2x_2 + 2 = 0$ , and the fifth because  $x_2 = 0$ . So, the initial working set is chosen as constraints three and five.

It is important to note that this choice is not unique. We could start with only one of these active, or even with none. But whichever option we choose, the algorithm will eventually find the same solution. What changes is only the path — the sequence of working sets and iterates the method goes through. This illustrates one of the strengths of the active-set framework: flexibility in the starting configuration while preserving global convergence.



At  $x^0 = (2, 0)^T$ ,  $\mathcal{W}_0 = \{3, 5\}$ ,  $x^0$  is a vertex, so  $p^0 = 0$  for:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (Gx^0 + c)^T p, \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_0. \end{aligned}$$

Multipliers from  $Gx^0 + c = \sum_{i \in \mathcal{W}_0} \hat{\lambda}_i a_i$  (here  $G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ ,  $c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$ ):

$$\begin{bmatrix} -1 \\ 2 \end{bmatrix} \hat{\lambda}_3 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \hat{\lambda}_5 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}, \quad \langle \hat{\lambda}_3, \hat{\lambda}_5 \rangle = \langle -2, -1 \rangle.$$

- ▶ Remove constraint 3 ( $\hat{\lambda}_3 = -2$ ), set  $\mathcal{W}_1 = \{5\}$ ,  $x^1 = (2, 0)^T$ .
- ▶ Iteration 1: Solve QP for  $\mathcal{W}_1$ , get  $p^1 = (-1, 0)^T$ , step length  $\alpha_1 = 1$ , so  $x^2 = (1, 0)^T$ . No blocking constraints, so  $\mathcal{W}_2 = \mathcal{W}_1 = \{5\}$ .
- ▶ Iteration 2: Solve QP, get  $p^2 = 0$ . Multiplier:  $\hat{\lambda}_5 = -5$ . Drop constraint 5, set  $\mathcal{W}_3 = \emptyset$ .

21/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

At the starting point  $x^0$ , we solve the quadratic subproblem restricted by the active constraints three and five. Because this point is already a vertex, and therefore the corresponding columns of matrix A are linearly independent, the step  $p^0$ , satisfying the constraints, is zero. To test whether the current point is optimal, we compute the Lagrange multipliers. These multipliers measure whether the active constraints support an optimal solution. If they are all nonnegative, we have optimality. But here both multipliers turn out negative. That means the current point cannot be optimal with both constraints active.

According to the rules of the active-set method, we remove one of the constraints with a negative multiplier. Let's drop the third constraint corresponding to the smallest multiplier. The working set now contains only constraint five, while the iterate itself remains unchanged.

With this new set, solving the subproblem gives a nonzero step:  $p^1 = (-1, 0)^T$ . Taking this step moves us to  $x^2 = (1, 0)^T$ . No new constraints become active, so the working set remains  $\{5\}$ . In the next subproblem, however, we again get zero step, but the multiplier for constraint five is negative. Therefore, we must drop constraint five as well. The working set becomes empty, which sets the stage for solving the unconstrained problem in the next iteration.



Objective:  $q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2$ ,  $G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ ,  $c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$ .

► Iteration 3:  $\mathcal{W}_3 = \emptyset$ , solve unconstrained QP:  $\min_p \frac{1}{2} p^T G p + (Gx^3 + c)^T p$ ,  $x^3 = (1, 0)^T$ .

$$\blacktriangleright Gx^3 + c = \begin{bmatrix} 2 \cdot 1 - 2 \\ 2 \cdot 0 - 5 \end{bmatrix} = \begin{bmatrix} 0 \\ -5 \end{bmatrix}, \text{ solution } p^3 = (0, 2.5)^T.$$

► For  $x^4 = x^3 + \alpha p^3 = (1, \alpha \cdot 2.5)$ , substitute into constraints:

$$1: x_1 - 2x_2 + 2 \geq 0 \Rightarrow 1 - 2 \cdot (\alpha \cdot 2.5) + 2 = 3 - 5\alpha \geq 0 \Rightarrow \alpha \leq 0.6,$$

$$2: -x_1 - 2x_2 + 6 \geq 0 \Rightarrow -1 - 2 \cdot (\alpha \cdot 2.5) + 6 = 5 - 5\alpha \geq 0 \Rightarrow \alpha \leq 1.0,$$

$$3: -x_1 + 2x_2 + 2 \geq 0 \Rightarrow -1 + 2 \cdot (\alpha \cdot 2.5) + 2 = 1 + 5\alpha \geq 0 \Rightarrow \alpha \geq -0.2,$$

$$4: x_1 \geq 0 \Rightarrow x_1 = 1 \geq 0, \text{ satisfied},$$

$$5: x_2 \geq 0 \Rightarrow x_2 = \alpha \cdot 2.5 \geq 0, \text{ satisfied for } \alpha \geq 0.$$

►  $\alpha_3 = \min(0.6, 1.0) = 0.6$  (constraint 1 active), so  $x^4 = (1, 1.5)^T$ ,  $\mathcal{W}_4 = \{1\}$ .

22/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

With no active constraints, the method reduces to solving the unconstrained quadratic problem with respect to  $p$ . The step we obtain is  $p^3 = (0, 2.5)^T$ . So the algorithm wants to move straight upward.

Before taking this step fully, we must check feasibility. Substituting the candidate  $x = (1, \alpha \cdot 2.5)$  into the constraints, we find the tightest restriction comes from the first inequality, which requires  $\alpha \leq 0.6$ . That means we cannot move all the way to the unconstrained minimizer; instead, we stop earlier, exactly when constraint one becomes active.

Thus, the step length is  $\alpha_3 = 0.6$ . The new point is  $x^4 = (1, 1.5)^T$ . At this location, constraint one is active, so the working set for the next iteration is  $\{1\}$ . This shows how the algorithm balances between moving in a descent direction and respecting feasibility by stopping at the boundary of the feasible set.



Objective:  $q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2$ ,  $G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ ,  $c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$ .

Iteration 4: Solve QP for  $\mathcal{W}_4 = \{1\}$ ,  $a_1 = (1, -2)^T$ :  $p_1 - 2p_2 = 0$ .

$$\blacktriangleright x^4 = (1, 1.5)^T, Gx^4 + c = \begin{bmatrix} 2 \cdot 1 - 2 \\ 2 \cdot 1.5 - 5 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}.$$

$\blacktriangleright$  QP:  $\min_{p_1} 5p_2^2 - 2p_2$  s.t.  $p_1 = 2p_2$ , solution  $p^4 = (0.4, 0.2)^T$ , step length  $\alpha_4 = 1$ , so  $x^5 = (1 + 0.4, 1.5 + 0.2) = (1.4, 1.7)^T$ .

$\blacktriangleright$  No blocking constraints, so  $\mathcal{W}_5 = \mathcal{W}_4 = \{1\}$ .

Iteration 5: Solve QP for  $\mathcal{W}_5 = \{1\}$ , get  $p^5 = (0, 0)^T$ .

$$\blacktriangleright$$
 Multiplier:  $Gx^5 + c = \begin{bmatrix} 2 \cdot 1.4 - 2 \\ 2 \cdot 1.7 - 5 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -1.6 \end{bmatrix} = \hat{\lambda}_1 \begin{bmatrix} 1 \\ -2 \end{bmatrix}$ , so  $\hat{\lambda}_1 = 0.8$ .

$\blacktriangleright$  Since  $\hat{\lambda}_1 \geq 0$ , stop with  $x^* = (1.4, 1.7)^T$ .

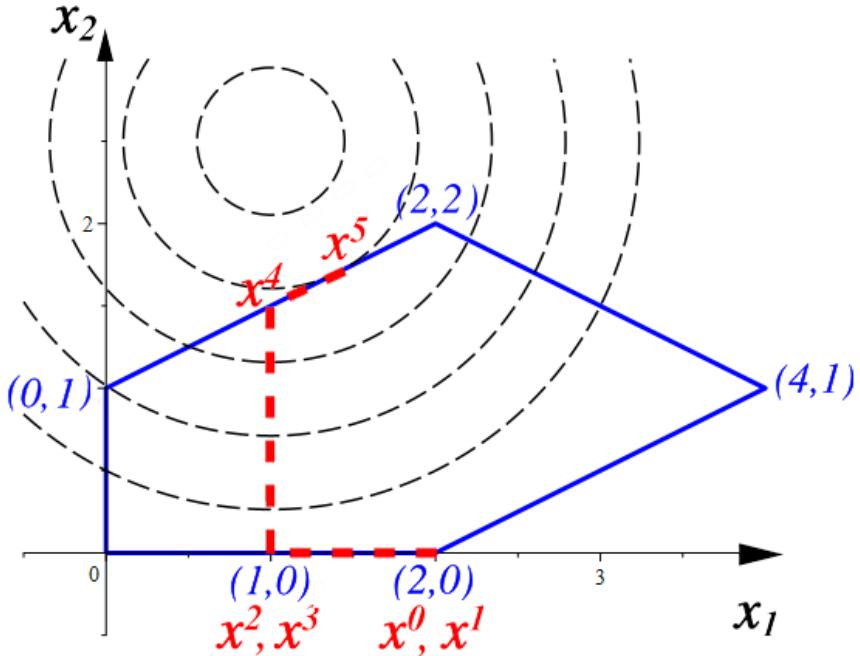
23/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

At iteration four, the working set contains only constraint one. This means our step must remain tangent to the boundary defined by  $x_1 - 2x_2 + 2 = 0$ . Solving the subproblem gives a direction  $p^4 = (0.4, 0.2)^T$ . Taking a full step moves us to  $x^5 = (1.4, 1.7)^T$ . No new constraints become active, so the working set stays the same.

In iteration five, solving again produces zero step. That is a signal that we might have reached optimality. To confirm, we compute the multiplier associated with the active constraint. It comes out positive. This tells us the Karush-Kuhn-Tucker conditions are satisfied: the point is feasible, the gradient is balanced by the active constraint, and the multiplier is nonnegative. Therefore, the algorithm stops here.

The final solution is  $x^* = (1.4, 1.7)^T$ . The example shows clearly how the active-set method proceeds: starting from a feasible point, it alternates between moving along descent directions and adjusting the set of active constraints, until the conditions for optimality are fully met.



24/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

### Comments

This figure gives us a geometric view of how the active-set method behaves on the quadratic example we have just solved. The dashed black curves are the level sets of the objective function. They show where the function has the same value, and their centers mark the unconstrained minimizer at point  $(1, 2.5)^T$ . The solid blue curve outlines the feasible region defined by the inequalities.

The red polyline is the trajectory of the algorithm. We start at the point  $(2, 0)^T$ , lying on the boundary. From there, the method takes steps along edges of the feasible set or moves inside, depending on the current working set. Each kink in the red path corresponds to a change in the set of active constraints: either dropping one with a negative multiplier or adding a new one when we hit the boundary.

Notice that the path does not move directly toward the unconstrained minimizer, because feasibility must always be maintained. Instead, the algorithm zigzags between descent directions and boundary adjustments. Finally, it settles at the point  $(1.4, 1.7)^T$ , where the gradient is balanced against the active constraint and all optimality conditions are satisfied.

This picture nicely summarizes the logic of the active-set approach: optimization proceeds as a sequence of constrained steps, adapting the working set until the solution is reached.



### Choice of Initial Working Set

In the previous example for  $x^0 = (2, 0)^T$ , active constraints were  $\mathcal{W}_0 = \{3, 5\}$ .  
 Alternatives:

- ▶  $\mathcal{W}_0 = \{3\}$ :  $p^0 = (0.2, 0.1)^T$ ,  $x^1 = (2.2, 0.1)^T$ .
- ▶  $\mathcal{W}_0 = \{5\}$ :  $p^0 = (-1, 0)^T$ ,  $x^1 = (1, 0)^T$  (skips dropping constraint 3).
- ▶  $\mathcal{W}_0 = \emptyset$ :  $p^1 = (-1, 2.5)^T$ ,  $\alpha_1 = \frac{2}{3}$ ,  $x^1 = (\frac{4}{3}, \frac{5}{3})^T$ ,  $\mathcal{W}_1 = \{1\}$ , solution  $x^*$  on next iteration.

$\mathcal{W}_k$  and  $\mathcal{A}(x^k)$  may differ later:

- ▶ If multiple blocking constraints, only one added to  $\mathcal{W}_k$ , breaking  $\mathcal{W}_k = \mathcal{A}(x^k)$
- ▶ Choice of blocking constraint affects subsequent iterates.

### Properties of Active-Set Method

**Linear Independence:** Constraint gradients in  $\mathcal{W}_0$  are linearly independent.

Strategy ensures this for all  $\mathcal{W}_k$ :

- ▶ Blocking constraint's normal  $a_i$  is not a linear combination of  $\{a_i \mid i \in \mathcal{W}_k\}$  (see properties of blocking constraints).
- ▶ Deletion of constraints preserves linear independence.

### Comments

When we begin the active-set method, the choice of the initial working set plays an important role in determining the sequence of iterates, although it does not affect the final solution. For instance, in the example with starting point  $x^0 = (2, 0)^T$ , we could select constraints three and five as the working set, which was the choice we discussed earlier. But alternative choices are possible. If we begin with only constraint three, then the computed step is  $p^0 = (0.2, 0.1)^T$ , leading to  $x^1 = (2.2, 0.1)^T$ . If instead we select only constraint five, the step is  $p^0 = (-1, 0)^T$ , giving  $x^1 = (1, 0)^T$ , thus skipping the process of dropping constraint three later. Finally, if we start with an empty working set, the first step is  $p^1 = (-1, 2.5)^T$ , with step length  $\alpha = 2/3$ . This produces  $x^1 = (4/3, 5/3)^T$ , and in the next iteration constraint one becomes active.

This example highlights two important facts. First, the working set at iteration  $k$  does not always coincide with the set of constraints active at  $x^k$ . Second, linear independence of constraint gradients is preserved throughout. Each added blocking constraint has a normal vector that is linearly independent of the current set, and deletion of constraints never introduces dependence. These structural properties guarantee that the algorithm remains well-defined at every step.



**Constraint Removal:** Remove constraint with most negative Lagrange multiplier  $\hat{\lambda}_i$ :

- ▶ Effective but sensitive to constraint scaling (scaling constraint by  $\beta > 0$  scales  $\lambda_i$  by  $1/\beta$ ).
- ▶ Analogous to Dantzig's pivot rule in simplex method.

**Iteration Bound:** Adding/removing at most one constraint per iteration implies  $\geq m$  iterations if  $m$  constraints are active at  $x^*$  and  $x^0$  is strictly feasible.

## Convergence for Strictly Convex QPs

Active-Set Method for QP:  $\min_x \frac{1}{2}x^T Gx + c^T x$ , s.t.  $a_i^T x = b_i$ ,  $i \in \mathcal{E}$ ,  $a_i^T x \geq b_i$ ,  $i \in \mathcal{I}$ , converges to  $x^*$  in finite iterations if  $\alpha_k > 0$  when  $p_k \neq 0$ .

If  $p_k = 0$  solves  $\min_p \frac{1}{2}p^T Gp + (Gx_k + c)^T p$ , s.t.  $a_i^T p = 0$ ,  $i \in \mathcal{W}_k$ , then  $x_k$  is the unique global minimizer of  $q(\cdot)$  for  $\mathcal{W}_k$  (see descent direction properties).

- ▶ If  $\hat{\lambda}_i < 0$  for some  $i \in \mathcal{W}_k$ , dropping  $i$  yields  $p_{k+1} \neq 0$ , a strict descent direction for  $q(x)$ .
- ▶ Since  $\alpha_k > 0$ ,  $q(x_{k+1}) < q(x_k)$ , so  $\mathcal{W}_k$  is never revisited.

26/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

Two central properties govern the behavior of the active-set method: the rule for removing constraints and the guarantee of convergence for strictly convex quadratic programs. When we encounter a zero step, we compute the Lagrange multipliers for the current working set. If one of them is negative, we must remove the corresponding constraint. The standard rule is to delete the constraint with the most negative multiplier. This choice is effective but has one drawback: it depends on scaling. If a constraint is multiplied by a positive factor  $\beta$ , then its multiplier scales by  $1/\beta$ , which can change which constraint looks "most negative." This sensitivity is very similar to what happens in the simplex method, where Dantzig's pivot rule selects the variable with the largest reduced cost.

Another property is that at most one constraint is added or removed per iteration. Suppose the optimal point has  $m$  active constraints, and we start from a strictly feasible  $x^0$  with none of them active. Then, in the worst case, we need at least  $m$  iterations, since each step adds at most one constraint. Fortunately, for strictly convex quadratic programs, we can guarantee finite termination. Whenever  $p_k$  is nonzero, the step length  $\alpha_k$  is strictly positive, so the objective decreases. If  $p_k$  is zero, but some multiplier is negative, removing that constraint produces a strict descent direction. Because the objective always decreases and no working set is ever repeated, the algorithm must terminate in finitely many steps at the unique global solution.



If  $p_k \neq 0$ , either  $\alpha_k = 1$  (reaching minimizer for  $\mathcal{W}_k$ , so  $p_{k+1} = 0$ ) or a constraint is added to  $\mathcal{W}_k$ .

- ▶ After  $\leq n$  iterations,  $\mathcal{W}_k$  has  $n$  independent constraints, forcing  $p_k = 0$ .

## Convergence and Cycling

**Finite Termination:** Since  $p_k = 0$  occurs at least every  $n$  iterations and  $\mathcal{W}_k$  is never revisited, the finite number of possible  $\mathcal{W}_k$  ensures termination at  $x^*$  satisfying optimality for the QP.

**Cycling:**  $\alpha_k > 0$  for  $p_k \neq 0$  prevents cycling (where  $x^k = x^{k+s}$ ,  $\mathcal{W}_k = \mathcal{W}_{k+l}$  for some integers  $k$  and for some  $s \geq 1$ ).

- ▶ Cycling occurs if constraints are dropped/added without any movement along the computed direction  $p$ .
- ▶ Handled similarly to linear programming methods; most QP implementations ignore cycling.

## Comments

The convergence of the active-set method relies on two key mechanisms: the behavior of the search direction and the evolution of the working set. Whenever the computed step direction, denoted  $p_k$ , is nonzero, the method always makes progress. Either the step length  $\alpha_k$  equals one, which means we have reached the minimizer relative to the current working set and so the next direction  $p_{k+1}$  becomes zero, or a new constraint becomes active and is added to the working set. Because the dimension of the problem is  $n$ , after at most  $n$  consecutive steps, we accumulate  $n$  linearly independent constraints, which forces the direction  $p_k$  to be zero. This ensures that the algorithm cannot continue indefinitely without reaching a stationary situation.

When  $p_k = 0$ , we check multipliers and either terminate at the optimal solution or drop a constraint. Thus, the situation  $p_k = 0$  occurs at least once every  $n$  iterations, guaranteeing steady progress. Moreover, the working set is never revisited. Since the number of possible working sets is finite, the method must terminate at a point  $x^*$  satisfying the Karush-Kuhn-Tucker conditions.

Cycling, which would mean revisiting the same point with the same working set, is excluded by the fact that  $\alpha_k$  is strictly positive whenever  $p_k$  is nonzero. Cycling could only occur if constraints were added and removed without any actual step. In practice, this is treated similarly to the simplex method, and most quadratic programming implementations safely ignore it.

## Computing Steps in Active-Set Method

Step for QP:  $\min_p \frac{1}{2} p^T G p + (Gx_k + c)^T p$ , s.t.  $a_i^T p = 0$ ,  $i \in \mathcal{W}_k$ , solves KKT system:

$$\begin{bmatrix} G & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} -(Gx_k + c) \\ 0 \end{bmatrix}.$$

- ▶  $\mathcal{W}_k$  changes by  $\leq 1$  index per iteration, so KKT matrix differs by  $\leq 1$  row/column ( $G$  fixed,  $A$  changes).
- ▶ Update matrix factors from previous iteration instead of recomputing.

**Null-space method:**  $A$  has  $m$  independent rows, QR factorization  $A^T \Pi = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$ , where  $\Pi$  is a permutation matrix;  $R$  is square, upper triangular and nonsingular;  $Q = [Q_1 \ Q_2]$  is  $n \times n$  orthogonal; and  $Q_1$  and  $R$  both have  $m$  columns while  $Q_2$  has  $n - m$  columns. We can choose  $Z = Q_2$ .

- ▶ New constraint:  $\tilde{A}^T = [A^T \ a]$ , where  $a$  is a column vector of length  $n$  such that  $\tilde{A}^T$  retains full column rank. Let's update  $Q$ ,  $R$  to get  $\tilde{Z}$  (with  $n - m - 1$  columns) for the expanded matrix  $\tilde{A}$ . Since  $Q_1 Q_1^T + Q_2 Q_2^T = I$  we have:

$$\tilde{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} = [A^T \Pi \ a] = Q \begin{bmatrix} R & Q_1^T a \\ 0 & Q_2^T a \end{bmatrix}.$$

28/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

Projected CG Method  
Convex QPs  
Active-Set Methods  
Big M Method  
Properties of Active-Set M.



## Comments

The core computational task in the active-set method is the determination of the search direction. At iteration  $k$ , we solve a quadratic subproblem in the variable  $p$ : minimize  $\frac{1}{2} p^T G p + (Gx_k + c)^T p$ , subject to the linear equalities  $a_i^T p = 0$  for all  $i$  in the working set. This is equivalent to a Karush-Kuhn-Tucker system, written as a block matrix with  $G$  in the upper left, the active constraint matrix  $A$  in the upper right,  $A^T$  in the lower left, and zeros in the lower right. Solving this system yields both the step  $p$  and the multipliers  $\lambda$ .

Because the working set changes by at most one constraint per iteration, the KKT matrix also changes by at most one row and one column. This structure makes it possible to update matrix factorizations efficiently instead of recomputing them from scratch.

One popular approach is the null-space method. If  $A$  has  $m$  independent rows, we can compute a QR factorization of  $A^T$  with permutation, so that  $A^T \Pi = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$ . Here  $R$  is a nonsingular upper triangular matrix of size  $m \times m$ , and  $Q$  is orthogonal, partitioned as  $Q_1$  and  $Q_2$ . The columns of  $Q_2$  form a basis of the null space, which we denote as  $Z$ . When a new constraint  $a$  is added, we extend  $A^T$  to a new matrix and update the factorization to obtain a new null-space basis. This update is more efficient than starting over, and the orthogonal structure ensures numerical stability.



## QR Update for Adding Constraint

Add constraint to  $\mathcal{W}_k$ , new matrix  $\tilde{A}^T = [A^T \quad a]$ . Update QR factorization to get new null-space basis  $\tilde{Z}$ .

- Define orthogonal  $\hat{Q}$  s.t.:

$$\hat{Q}(Q_2^T a) = \begin{bmatrix} \gamma \\ 0 \end{bmatrix}, \quad \|Q_2^T a\| = |\gamma|, \text{ where } \gamma \text{ is a scalar.}$$

- Update QR:

$$\begin{aligned} \tilde{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} &= Q \begin{bmatrix} R & Q_1^T a \\ 0 & \hat{Q}^T \begin{bmatrix} \gamma \\ 0 \end{bmatrix} \end{bmatrix} = Q \begin{bmatrix} I & 0 \\ 0 & \hat{Q}^T \end{bmatrix} \begin{bmatrix} R & Q_1^T a \\ 0 & \gamma \\ 0 & 0 \end{bmatrix} = \tilde{Q} \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix}, \\ \tilde{\Pi} &= \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix}, \quad \tilde{Q} = [Q_1 \quad Q_2 \hat{Q}^T], \quad \tilde{R} = \begin{bmatrix} R & Q_1^T a \\ 0 & \gamma \end{bmatrix}. \end{aligned}$$

- $\tilde{Z}$  = last  $n - m - 1$  columns of  $Q_2 \hat{Q}^T$ .
- Efficiency: Computing  $\hat{Q}$ ,  $Q_2 \hat{Q}^T$  costs  $\mathcal{O}(n(n - m))$ , vs.  $\mathcal{O}(n^2m)$  for full QR, especially efficient when  $n - m \ll n$ .

## Comments

When a new constraint enters the working set, we must update the null-space basis accordingly. Suppose the new active matrix is  $A^T$  augmented with an additional column  $a$ . To avoid recomputing a full QR factorization, we apply a structured update. First, we compute an orthogonal transformation, denoted  $\hat{Q}$ , such that  $\hat{Q}$  applied to  $Q_2^T a$  produces a vector with  $\gamma$  in the first entry and zeros elsewhere, where  $\gamma$  is the norm of  $Q_2^T a$ . This isolates the new contribution in a single coordinate.

The update proceeds by combining the original  $Q$  with this transformation, forming a new orthogonal matrix  $\tilde{Q}$ , and a new upper triangular matrix  $\tilde{R}$ . From these, the expanded null-space basis  $\tilde{Z}$  is taken as the last  $n - m - 1$  columns of  $Q_2 \hat{Q}^T$ . The essential point is that the update cost is only of order  $n(n - m)$ , significantly cheaper than recomputing a full QR, which would cost on the order of  $n^2m$ . This difference is especially important when  $n - m$ , the dimension of the null space, is small relative to  $n$ .

Thus, by cleverly updating factorizations, the active-set method maintains efficiency across iterations. The algorithm adapts to each new constraint with only incremental work, while preserving the orthogonality and stability properties crucial for accurate computations.

## QR Update (Removing Constraint) and Hessian

- ▶ **Remove constraint:** Delete row from A, column from R. Restore upper triangularity in R via plane rotations.
- ▶ **Update Q** by inexpensive transformation of its first m columns, and the updated null-space matrix is the last  $n - m + 1$  columns from this matrix after the transformations are complete:

$$\tilde{Z} = [\tilde{z} \quad Z].$$

- ▶ **Cost:** Cheaper than  $\mathcal{O}(n^2m)$  for full QR (see numerical linear algebra methods).

- ▶ **Reduced Hessian:** For QP,  $h = 0$ ,  $p_Y = 0$ , solve:

$$(Z^T G Z) p_z = -Z^T (G x_k + c).$$

- ▶ **Update Cholesky:**  $Z^T G Z = LL^T$ . Then for  $\tilde{Z} = [\tilde{z} \quad Z]$ , transform L to  $\tilde{L}$  for  $\tilde{Z}^T G \tilde{Z}$  via elementary operations.
- ▶ **Simplifications:** Update  $Z^T (G x_k + c)$  with Z to  $\tilde{Z}$  (see further simplifications).

Projected CG Method  
Convex QPs  
Active-Set Methods  
Big M Method  
Properties of Active-Set M.



30/30 || SPbU & HIT 2025 || Shpilev P.V. || Classical optimization approaches

## Comments

The reverse operation occurs when a constraint is removed from the working set. In this case, the active matrix loses a row, and correspondingly the triangular factor R loses a column. To restore the triangular form, we apply plane rotations, a standard tool in numerical linear algebra. The orthogonal matrix Q is then updated by inexpensive transformations of its first m columns, and the new null-space basis is obtained as the last  $n - m + 1$  columns of the updated Q. Symbolically, the new  $\tilde{Z}$  is constructed by adding one extra column vector to the previous basis.

The computational cost of this operation is much less than recomputing a full QR factorization, making constraint removal efficient as well. With both adding and dropping constraints handled incrementally, the method adapts dynamically while keeping computations light.

Once the null-space basis is updated, we must also update the reduced Hessian, defined as  $Z^T G Z$ . This reduced matrix governs the quadratic behavior in the feasible subspace. To solve for the search step, we form the equation  $Z^T G Z p_z = -Z^T (G x_k + c)$ . The system is solved efficiently using a Cholesky factorization, denoted  $LL^T$ . When Z is replaced by  $\tilde{Z}$ , the factor L is transformed into a new  $\tilde{L}$  using elementary operations, again without starting over.

This interplay between updating factorizations and maintaining reduced Hessians is essential. It ensures that each iteration of the active-set method remains computationally feasible while rigorously preserving the mathematical structure required for convergence.