Interior-Point Methods in Convex Optimization

Interior-point methods (IPMs) are a class of algorithms used to solve convex optimization problems, particularly linear programming (LP), quadratic programming (QP), and more general convex optimization problems. Unlike the **simplex method**, which moves along the edges of the feasible region, interior-point methods traverse the interior of the feasible set.

Theory of Interior-Point Methods

Interior-point methods solve optimization problems by maintaining strictly feasible iterates within the interior of the feasible region. The general form of a convex optimization problem is:

$$\min f(x) \quad \text{subject to} \quad g_i(x) \leq 0, \quad i = 1, ..., m$$

where $f(x)$ is a convex function, and $g_i(x)$ are convex constraints. Interior-point methods typically rely on **Newton's method** for efficiently solving nonlinear equations that arise in the optimality conditions.

One of the fundamental approaches within interior-point methods is the **barrier method**.

Barrier Method

The barrier method is a foundational technique in interior-point methods. Instead of enforcing constraints explicitly, it incorporates a **barrier function** into the objective, preventing iterates from reaching the boundary of the feasible region.

For a problem of the form:

$$\min f(x) \quad \text{subject to} \quad g_i(x) \leq 0, \quad i = 1, ..., m$$

we introduce a logarithmic barrier function:

$$\phi(x) = -\sum_{i=1}^{m} \ln(-g_i(x))$$

The new unconstrained problem to be solved is:

$$\min \ f(x) + \frac{1}{t}\phi(x)$$

where t is a **barrier parameter**. As t increases, the solution of the modified problem approaches the solution of the original constrained problem. The optimization is typically solved iteratively using Newton's method.

Importance of Interior-Point Methods

Interior-point methods are crucial for large-scale optimization problems due to their **polynomial-time complexity**, making them faster for many practical problems compared to the simplex method. They are widely used in:

- **Linear Programming (LP)**
- **Quadratic Programming (QP)**
- **Semidefinite Programming (SDP)**
- **Nonlinear Programming (NLP)**

Since the 1980s, interior-point methods have been favored for **large** problems because they avoid the worst-case exponential complexity of the simplex method.

Interior-Point Method vs. Simplex Method

Feature	Interior-Point Method	Simplex Method
Approach	Moves through the interior of the feasible region	Moves along the edges of the feasible region
Efficiency	Polynomial-time complexity for LP	Exponential-time complexity in the worst case
Scalability	More efficient for large-scale problems	More efficient for small to medium-sized problems
Path to Solution	Uses Newton's method and barrier functions	Uses pivoting operations between adjacent vertices
Iterates	Produces intermediate infeasible points	Always remains on a feasible vertex

Interior-point methods are more **efficient for large-scale problems**, while the **simplex method** may be better for small problems due to its practical efficiency and ease of implementation.

Semidefinite Programming (SDP)

A **semidefinite programming problem (SDP)** is an optimization problem where the objective function is linear, and the constraints require that a matrix (depending on the decision variables) be **positive semidefinite** (PSD). The general form is:

Minimize:
$\mathrm{Tr}(CX)$

Subject to:
$\mathrm{Tr}(A_i X) = b_i, \quad i = 1, 2, ..., m$

X is a positive semidefinite (PSD) matrix (denoted as **X ≥ 0**)

where:

- X is a symmetric **positive semidefinite (PSD) matrix**.
- C and A_i are given symmetric matrices.
- X must be positive semidefinite (i.e., all its eigenvalues are non-negative).
- The trace function represents a **linear** objective function.

Applications of SDP

SDP problems appear in various fields such as:

- **Control theory** (Lyapunov functions for stability analysis)
- **Robust optimization** (handling uncertainty in constraints)
- **Quantum computing** (density matrix optimization)
- **Machine learning** (kernel methods and dimensionality reduction)