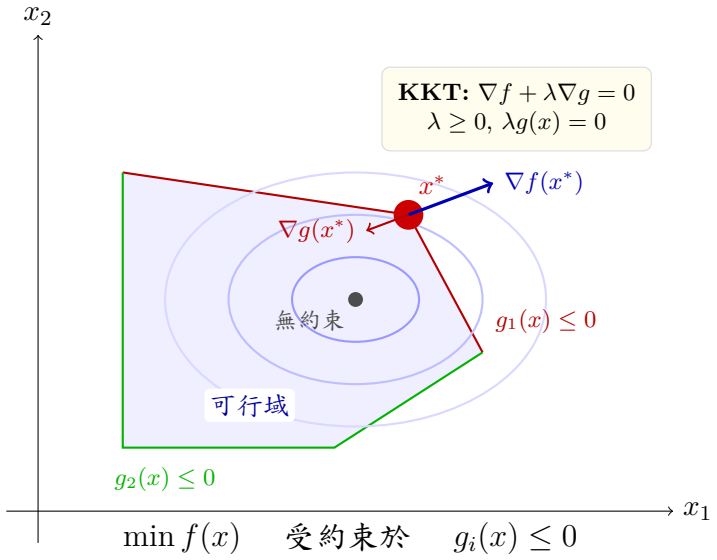
數學建模補充教材

約束最佳化

理論與應用



Kenneth, Sok Kin Cheng

最後更新：July 8, 2025

掌握帶約束與邊界的最佳化

Contents

Chapter 1

約束最佳化的基礎

1.1 引言

約束最佳化處理的是當決策變數必須滿足特定限制時尋找最優解的基本問題。與無約束最佳化不同，約束的存在根本改變了最優性條件和求解方法的性質。

定義 **1.1** (約束最佳化問題)**.** 一般約束最佳化問題可表述為：

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{1.1}$$

$$\text{受約束於} \quad g_i(x) \leq 0, \quad i = 1, 2, \ldots, m \tag{1.2}$$

$$h_j(x) = 0, \quad j = 1, 2, \ldots, p \tag{1.3}$$

其中 $f : \mathbb{R}^n \to \mathbb{R}$ 是目標函數，$g_i : \mathbb{R}^n \to \mathbb{R}$ 是不等式約束，$h_j : \mathbb{R}^n \to \mathbb{R}$ 是等式約束。

1.2 可行集與約束資格

定義 **1.2** (可行集)**.** 可行集 $\mathcal{S}$ 定義為：

$$\mathcal{S} = \{x \in \mathbb{R}^n : g_i(x) \leq 0 \text{ 對所有} i, \text{ 且} h_j(x) = 0 \text{ 對所有} j\}$$

定義 **1.3** (活躍集)**.** 在點 x^* 處，活躍集 $\mathcal{A}(x^*)$ 由以下組成：

$$\mathcal{A}(x^*) = \{i : g_i(x^*) = 0\} \cup \{j : h_j(x^*) = 0\}$$

定義 **1.4** (線性無關約束資格 (LICQ))**.** 在可行點 x^* 處，如果梯度 $\{\nabla g_i(x^*) : i \in \mathcal{A}(x^*)\}$ 和 $\{\nabla h_j(x^*) : j = 1, \ldots, p\}$ 線性無關，則 LICQ 成立。

1.3 拉格朗日乘數法與 KKT 條件

拉格朗日乘數法將駐點概念擴展到約束問題。

定義 **1.5** (拉格朗日函數)**.** 拉格朗日函數定義為：

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x)$$

其中 $\lambda_i \geq 0$ 是不等式約束的乘數，μ_j 是等式約束的乘數。

定理

Karush-Kuhn-Tucker (KKT) 條件
設 x^* 是約束最佳化問題的局部最小值，且在 x^* 處 LICQ 成立。則存在乘數 $\lambda^* \in \mathbb{R}^m$ 和 $\mu^* \in \mathbb{R}^p$ 使得：
駐點條件： $\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$
原始可行性： $g_i(x^*) \leq 0$, $h_j(x^*) = 0$
對偶可行性： $\lambda_i^* \geq 0$
互補鬆弛： $\lambda_i^* g_i(x^*) = 0$

1.3.1 應用與例子

例子 **1.1** (經典拉格朗日乘數法 - 等式約束). 考慮從原點到直線的最短距離問題：

$$\begin{aligned} \min \quad & f(x_1, x_2) = x_1^2 + x_2^2 && (1.4) \\ \text{受約束於} \quad & h(x_1, x_2) = x_1 + 2x_2 - 3 = 0 && (1.5) \end{aligned}$$

求解過程：
步驟 **1**：建立拉格朗日函數：

$$\mathcal{L}(x_1, x_2, \mu) = x_1^2 + x_2^2 + \mu(x_1 + 2x_2 - 3)$$

步驟 **2**：應用一階必要條件：

$$\frac{\partial \mathcal{L}}{\partial x_1} = 2x_1 + \mu = 0 \quad \Rightarrow \quad x_1 = -\frac{\mu}{2} \tag{1.6}$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 2x_2 + 2\mu = 0 \quad \Rightarrow \quad x_2 = -\mu \tag{1.7}$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = x_1 + 2x_2 - 3 = 0 \tag{1.8}$$

步驟 **3**：求解方程組：代入約束條件：$-\frac{\mu}{2} + 2(-\mu) - 3 = 0$

$$-\frac{\mu}{2} - 2\mu = 3 \quad \Rightarrow \quad -\frac{5\mu}{2} = 3 \quad \Rightarrow \quad \mu = -\frac{6}{5}$$

步驟 **4**：求最優點：

$$x_1^* = -\frac{\mu}{2} = \frac{3}{5}, \quad x_2^* = -\mu = \frac{6}{5}$$

驗證： $\frac{3}{5} + 2 \cdot \frac{6}{5} = \frac{15}{5} = 3$
解釋：拉格朗日乘數 $\mu = -\frac{6}{5}$ 代表最優目標值對約束右端項的變化率。

例子 **1.2** (帶不等式約束的 KKT 條件). 求解約束二次規劃問題：

$$\begin{aligned} \min \quad & f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 4x_2 && (1.9) \\ \text{受約束於} \quad & g_1(x_1, x_2) = x_1 + x_2 - 2 \leq 0 && (1.10) \\ & g_2(x_1, x_2) = -x_1 \leq 0 && (1.11) \\ & g_3(x_1, x_2) = -x_2 \leq 0 && (1.12) \end{aligned}$$

求解過程：

步驟 **1**：建立拉格朗日函數：

$$\mathcal{L}(x,\lambda) = x_1^2 + x_2^2 - 2x_1 - 4x_2 + \lambda_1(x_1 + x_2 - 2) + \lambda_2(-x_1) + \lambda_3(-x_2)$$

步驟 **2**：寫出 KKT 條件：

$$\text{駐點條件：} \quad \begin{cases} \frac{\partial \mathcal{L}}{\partial x_1} = 2x_1 - 2 + \lambda_1 - \lambda_2 = 0 \\ \frac{\partial \mathcal{L}}{\partial x_2} = 2x_2 - 4 + \lambda_1 - \lambda_3 = 0 \end{cases} \tag{1.13}$$

$$\text{原始可行性：} \quad x_1 + x_2 \le 2, \quad x_1 \ge 0, \quad x_2 \ge 0 \tag{1.14}$$

$$\text{對偶可行性：} \quad \lambda_i \ge 0, \quad i = 1,2,3 \tag{1.15}$$

$$\text{互補鬆弛：} \quad \begin{cases} \lambda_1(x_1 + x_2 - 2) = 0 \\ \lambda_2 x_1 = 0 \\ \lambda_3 x_2 = 0 \end{cases} \tag{1.16}$$

步驟 **3**：情況分析：

情況 **1**：檢查無約束解（$\lambda_1 = \lambda_2 = \lambda_3 = 0$）由駐點條件：$x_1 = 1$, $x_2 = 2$ 檢查可行性：$1 + 2 = 3 > 2$ （違反約束 1）

情況 **2**：約束 1 活躍（$\lambda_1 > 0$, $x_1 + x_2 = 2$）假設 $\lambda_2 = \lambda_3 = 0$（相對於邊界內點）：

$$2x_1 - 2 + \lambda_1 = 0 \quad \Rightarrow \quad \lambda_1 = 2 - 2x_1 \tag{1.17}$$

$$2x_2 - 4 + \lambda_1 = 0 \quad \Rightarrow \quad \lambda_1 = 4 - 2x_2 \tag{1.18}$$

設為相等：$2 - 2x_1 = 4 - 2x_2$ 由約束：$x_2 = 2 - x_1$ 代入：$2 - 2x_1 = 4 - 2(2 - x_1) = 2x_1$ 求解：$2 = 4x_1 \Rightarrow x_1 = \frac{1}{2}$, $x_2 = \frac{3}{2}$

檢查：$\lambda_1 = 2 - 2 \cdot \frac{1}{2} = 1 > 0$

步驟 **4**：驗證所有 KKT 條件：

- 駐點條件：$2 \cdot \frac{1}{2} - 2 + 1 = 0$, $2 \cdot \frac{3}{2} - 4 + 1 = 0$
- 原始可行性：$\frac{1}{2} + \frac{3}{2} = 2$, $x_1, x_2 \ge 0$
- 對偶可行性：$\lambda = (1,0,0) \ge 0$
- 互補鬆弛：$1 \cdot 0 = 0$, $0 \cdot \frac{1}{2} = 0$, $0 \cdot \frac{3}{2} = 0$

最優解： $x^* = \left(\frac{1}{2}, \frac{3}{2}\right)$，乘數為 $\lambda^* = (1,0,0)$

例子 **1.3** (帶混合約束的投資組合最佳化)**.** 考慮簡化的投資組合最佳化問題：

$$\min \quad \frac{1}{2}(w_1^2 + w_2^2) \quad \text{（最小化變異數/風險）} \tag{1.19}$$

$$\text{受約束於} \quad 0.08w_1 + 0.12w_2 \ge 0.10 \quad \text{（最小預期報酬）} \tag{1.20}$$

$$w_1 + w_2 = 1 \quad \text{（預算約束）} \tag{1.21}$$

$$w_1, w_2 \ge 0 \quad \text{（無放空）} \tag{1.22}$$

求解過程：

步驟 **1**：轉換為標準形式：

$$\min \quad \frac{1}{2}(w_1^2 + w_2^2) \tag{1.23}$$

$$\text{受約束於} \quad g_1(w) = -0.08w_1 - 0.12w_2 + 0.10 \le 0 \tag{1.24}$$

$$h_1(w) = w_1 + w_2 - 1 = 0 \tag{1.25}$$

$$g_2(w) = -w_1 \le 0, \quad g_3(w) = -w_2 \le 0 \tag{1.26}$$

步驟 **2**：建立拉格朗日函數：

$$\mathcal{L} = \frac{1}{2}(w_1^2 + w_2^2) + \lambda_1(-0.08w_1 - 0.12w_2 + 0.10) + \mu(w_1 + w_2 - 1) + \lambda_2(-w_1) + \lambda_3(-w_2)$$

步驟 **3**： KKT 條件：

$$\text{駐點條件：} \quad \begin{cases} w_1 - 0.08\lambda_1 + \mu - \lambda_2 = 0 \\ w_2 - 0.12\lambda_1 + \mu - \lambda_3 = 0 \end{cases} \tag{1.27}$$

$$\text{約束：} \quad w_1 + w_2 = 1, \quad 0.08w_1 + 0.12w_2 \geq 0.10, \quad w_1, w_2 \geq 0 \tag{1.28}$$

步驟 **4**：情況分析：
假設內點解（$\lambda_2 = \lambda_3 = 0$）且報酬約束活躍（$\lambda_1 > 0$）:

$$w_1 = 0.08\lambda_1 - \mu \tag{1.29}$$

$$w_2 = 0.12\lambda_1 - \mu \tag{1.30}$$

由預算約束：$w_1+w_2 = 0.20\lambda_1-2\mu = 1$ 由報酬約束：$0.08w_1+0.12w_2 = 0.0208\lambda_1-0.20\mu = 0.10$
求解方程組：

$$\begin{bmatrix} 0.20 & -2 \\ 0.0208 & -0.20 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \mu \end{bmatrix} = \begin{bmatrix} 1 \\ 0.10 \end{bmatrix}$$

使用克拉默法則：$\det = 0.20 \cdot (-0.20) - (-2) \cdot 0.0208 = -0.04 + 0.0416 = 0.0016$
$\lambda_1 = \dfrac{\begin{vmatrix} 1 & -2 \\ 0.10 & -0.20 \end{vmatrix}}{0.0016} = \dfrac{-0.20+0.20}{0.0016} = 0$
由於 $\lambda_1 = 0$，報酬約束不活躍。檢查無約束最優解是否滿足報酬約束：
當 $\lambda_1 = 0$ 時：$w_1 = w_2 = \mu$ 由預算約束：$2\mu = 1 \Rightarrow \mu = 0.5$ 所以 $w_1 = w_2 = 0.5$
檢查報酬：$0.08 \cdot 0.5 + 0.12 \cdot 0.5 = 0.10$
最優投資組合： $w^* = (0.5, 0.5)$ - 兩種資產等權重。
經濟解釋：當最小風險投資組合正好滿足最小報酬約束時，不需要在風險和報酬之間做權衡。

Python 程式碼

```python
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

def solve_kkt_examples():
    """
    數值求解 KKT 範例進行驗證
    """

    # 範例 1：等式約束問題
    def obj1(x):
        return x[0]**2 + x[1]**2

    def constraint1(x):
        return x[0] + 2*x[1] - 3

    constraints1 = [{'type': 'eq', 'fun': constraint1}]
    x0_1 = np.array([1.0, 1.0])
```

```
result1 = minimize(obj1, x0_1, method='SLSQP', constraints=constraints1)

print("範例 1 - 等式約束：")
print(f"數值解： x = {result1.x}")
print(f"解析解： x = [0.6, 1.2]")
print(f"目標函數值： {result1.fun}")
print(f"約束違反： {abs(constraint1(result1.x))}")
print()

# 範例 2：不等式約束
def obj2(x):
    return x[0]**2 + x[1]**2 - 2*x[0] - 4*x[1]

def grad2(x):
    return np.array([2*x[0] - 2, 2*x[1] - 4])

constraints2 = [
    {'type': 'ineq', 'fun': lambda x: 2 - x[0] - x[1]},  # x1 + x2 <= 2
    {'type': 'ineq', 'fun': lambda x: x[0]},              # x1 >= 0
    {'type': 'ineq', 'fun': lambda x: x[1]}               # x2 >= 0
]

x0_2 = np.array([0.5, 0.5])
result2 = minimize(obj2, x0_2, method='SLSQP', jac=grad2,
                   constraints=constraints2)

print("範例 2 - 不等式約束：")
print(f"數值解： x = {result2.x}")
print(f"解析解： x = [0.5, 1.5]")
print(f"目標函數值： {result2.fun}")

# 檢查約束活躍性
x_opt = result2.x
g1 = x_opt[0] + x_opt[1] - 2
print(f"約束 1 (x1+x2-2)： {g1:.6f} {'(活躍)' if abs(g1) < 1e-6 else '(不活躍)'}")
print()

# 範例 3：投資組合最佳化
def portfolio_obj(w):
    return 0.5 * (w[0]**2 + w[1]**2)

constraints3 = [
    {'type': 'ineq', 'fun': lambda w: 0.08*w[0] + 0.12*w[1] - 0.10},
    {'type': 'eq', 'fun': lambda w: w[0] + w[1] - 1},
    {'type': 'ineq', 'fun': lambda w: w[0]},
    {'type': 'ineq', 'fun': lambda w: w[1]}
]

w0 = np.array([0.5, 0.5])
result3 = minimize(portfolio_obj, w0, method='SLSQP', constraints=constraints3)

print("範例 3 - 投資組合最佳化：")
```

```
    print(f"最優權重： w = {result3.x}")
    print(f"投資組合變異數： {result3.fun:.6f}")

    # 計算預期報酬
    expected_return = 0.08 * result3.x[0] + 0.12 * result3.x[1]
    print(f"預期報酬： {expected_return:.4f} = {expected_return*100:.1f}%")

    return result1, result2, result3

# 可視化函數
def plot_constrained_optimization():
    """
    可視化第二個範例的等高線和約束
    """
    x1 = np.linspace(-0.5, 3, 100)
    x2 = np.linspace(-0.5, 3, 100)
    X1, X2 = np.meshgrid(x1, x2)

    # 目標函數
    F = X1**2 + X2**2 - 2*X1 - 4*X2

    plt.figure(figsize=(10, 8))

    # 等高線
    contours = plt.contour(X1, X2, F, levels=15, alpha=0.7, colors='blue')
    plt.clabel(contours, inline=True, fontsize=8)

    # 約束邊界
    x1_line = np.linspace(0, 2.5, 100)
    x2_line = 2 - x1_line

    # 可行域
    x1_fill = np.linspace(0, 2, 100)
    x2_upper = 2 - x1_fill
    x2_lower = np.zeros_like(x1_fill)

    plt.fill_between(x1_fill, x2_lower, x2_upper, alpha=0.3,
                     color='lightgreen', label='可行域')

    # 邊界
    plt.plot(x1_line, x2_line, 'r-', linewidth=2, label='$x_1 + x_2 = 2$')
    plt.axhline(y=0, color='black', linewidth=1.5, label='$x_2 \\geq 0$')
    plt.axvline(x=0, color='black', linewidth=1.5, label='$x_1 \\geq 0$')

    # 最優點
    plt.plot(0.5, 1.5, 'ro', markersize=12, label='約束最優解')
    plt.plot(1, 2, 'bs', markersize=10, label='無約束最優解')

    # 在最優解處添加箭頭顯示梯度
    plt.arrow(0.5, 1.5, 0.3, 0.6, head_width=0.1, head_length=0.1,
              fc='red', ec='red', alpha=0.7)
    plt.text(0.9, 2.2, '$\\nabla f$', fontsize=12, color='red')
```

```
    plt.xlim(-0.2, 2.8)
    plt.ylim(-0.2, 2.8)
    plt.xlabel('$x_1$', fontsize=14)
    plt.ylabel('$x_2$', fontsize=14)
    plt.title('KKT 範例：約束二次最佳化', fontsize=16)
    plt.legend(fontsize=12)
    plt.grid(True, alpha=0.3)
    plt.show()

# 執行範例
results = solve_kkt_examples()
plot_constrained_optimization()
```

探索

進階 **KKT** 分析
探索這些概念以加深理解：

1. 約束資格違反：構造 LICQ 失效的範例，觀察 KKT 條件可能無法提供最優性必要條件的情況。
2. 多重最優解：尋找 KKT 條件在多個點滿足的問題，研究何時會發生這種情況。
3. 經濟解釋：在投資組合範例中，將拉格朗日乘數解釋為影子價格並分析其對約束變化的敏感性。
4. 活躍集變化：研究當問題參數變化時活躍集如何變化，特別是在不同約束組合之間的轉換。

練習 **1.1.** 使用 KKT 條件求解下列約束最佳化問題：

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 + x_3^2 && (1.31)\\ \text{受約束於} \quad & x_1 + x_2 + x_3 = 3 && (1.32)\\ & x_1^2 + x_2^2 \le 2 && (1.33)\\ & x_1, x_2, x_3 \ge 0 && (1.34) \end{aligned}$$

識別所有可能的活躍集並確定哪一個對應最優解。

練習 **1.2.** 對於一般二次規劃問題：

$$\begin{aligned} \min \quad & \frac{1}{2} x^T Q x + c^T x && (1.35)\\ \text{受約束於} \quad & Ax \le b && (1.36)\\ & Ex = d && (1.37)\\ & x \ge 0 && (1.38) \end{aligned}$$

其中 $Q \succ 0$，推導完整的 KKT 系統並說明如何將其寫成線性互補問題。

1.4 幾何解釋

KKT 條件有清晰的幾何解釋：在最優解處，目標函數的梯度必須是活躍約束梯度的線性組合。

1.4.1 基本幾何原理

定理

KKT 條件的幾何解釋
在約束最優解 x^* 處，目標函數梯度 $\nabla f(x^*)$ 位於活躍約束梯度生成的錐中。具體地：

$$\nabla f(x^*) = -\sum_{i \in \mathcal{A}(x^*)} \lambda_i \nabla g_i(x^*) - \sum_{j=1}^{p} \mu_j \nabla h_j(x^*)$$

其中 $\mathcal{A}(x^*)$ 是活躍集，所有 $\lambda_i \geq 0$。

這個幾何條件確保我們找不到既可行又能降低目標函數值的方向，這正是最優性條件。

例子 **1.4** (簡單約束問題)**.** 考慮問題：

$$\min \quad f(x) = x_1^2 + x_2^2 \tag{1.39}$$

$$\text{受約束於} \quad x_1 + x_2 - 1 = 0 \tag{1.40}$$

拉格朗日函數是：$\mathcal{L}(x, \mu) = x_1^2 + x_2^2 + \mu(x_1 + x_2 - 1)$
KKT 條件給出：

$$\frac{\partial \mathcal{L}}{\partial x_1} = 2x_1 + \mu = 0 \tag{1.41}$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 2x_2 + \mu = 0 \tag{1.42}$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = x_1 + x_2 - 1 = 0 \tag{1.43}$$

解得：$x_1 = x_2 = \frac{1}{2}$, $\mu = -1$。
幾何驗證： - $\nabla f(\frac{1}{2}, \frac{1}{2}) = (1, 1)$ - $\nabla h(\frac{1}{2}, \frac{1}{2}) = (1, 1)$ - 確實：$\nabla f = -\mu \nabla h = -(-1)(1, 1) = (1, 1)$

1.4.2 不同幾何情況的案例研究

例子 **1.5** (不等式約束變為活躍)**.** 考慮問題：

$$\min \quad f(x) = (x_1 - 2)^2 + (x_2 - 2)^2 \tag{1.44}$$

$$\text{受約束於} \quad g(x) = x_1 + x_2 - 2 \leq 0 \tag{1.45}$$

幾何分析：
情況 **1**：無約束最優解在 $(2, 2)$ 檢查約束：$2 + 2 - 2 = 2 > 0$ （不可行）
情況 **2**：約束最優解在邊界 $x_1 + x_2 = 2$ 上約束活躍，所以需要：$\nabla f(x^*) = \lambda \nabla g(x^*)$ 其中 $\lambda \geq 0$
在約束上任意點 (x_1, x_2) 處：- $\nabla f = (2(x_1 - 2), 2(x_2 - 2))$ - $\nabla g = (1, 1)$
對於 KKT 條件：$(2(x_1 - 2), 2(x_2 - 2)) = \lambda(1, 1)$
這給出：$x_1 - 2 = x_2 - 2 = \frac{\lambda}{2}$，所以 $x_1 = x_2$
由約束：$2x_1 = 2$，所以 $x_1 = x_2 = 1$
解： $x^* = (1, 1)$，$\lambda = -2$
由於 $\lambda < 0$，這違反了對偶可行性。約束在最優解處不活躍。
正確分析：無約束最優解不可行，所以我們投影到約束邊界上。最優點是目標函數梯度與約束邊界正交的地方，給出 $x^* = (1, 1)$。

例子 **1.6** (多重活躍約束 - 角點解). 考慮問題：

$$\begin{aligned} \min \quad & f(x) = x_1 + 2x_2 && (1.46)\\ \text{受約束於} \quad & g_1(x) = -x_1 \le 0 && (1.47)\\ & g_2(x) = -x_2 \le 0 && (1.48)\\ & g_3(x) = x_1 + x_2 - 3 \le 0 && (1.49) \end{aligned}$$

幾何分析：
可行域是一個三角形，頂點為 $(0,0)$, $(3,0)$, 和 $(0,3)$。
在原點 $(0,0)$：- $\nabla f = (1,2)$（指向東北）- $\nabla g_1 = (-1,0)$（指向西）- $\nabla g_2 = (0,-1)$（指向南）
對於 KKT：$\nabla f + \lambda_1 \nabla g_1 + \lambda_2 \nabla g_2 = 0$ $(1,2) + \lambda_1(-1,0) + \lambda_2(0,-1) = (0,0)$
這給出：$\lambda_1 = 1 > 0$ 且 $\lambda_2 = 2 > 0$
幾何洞察：目標梯度 $(1,2)$ 位於 $(-1,0)$ 和 $(0,-1)$ 生成的正錐中，確認 $(0,0)$ 是最優的。

1.4.3 幾何關係的可視化

例子 **1.7** (不等式約束的切線條件). 考慮經典問題：

$$\begin{aligned} \min \quad & f(x) = x_1^2 + x_2^2 && (1.50)\\ \text{受約束於} \quad & g(x) = x_1^2 + x_2^2 - 1 \le 0 && (1.51) \end{aligned}$$

幾何分析：
無約束最小值在 $(0,0)$，滿足約束因為 $0 + 0 - 1 = -1 < 0$。
由於約束不活躍（$\lambda = 0$），KKT 條件簡化為 $\nabla f = 0$，在 $(0,0)$ 處滿足。
幾何洞察：當無約束最優解位於可行域內部時，約束對解沒有影響。

例子 **1.8** (活躍約束的非切線情況). 考慮：

$$\begin{aligned} \min \quad & f(x) = -x_1 - x_2 && (1.52)\\ \text{受約束於} \quad & g(x) = x_1^2 + x_2^2 - 1 \le 0 && (1.53) \end{aligned}$$

分析：
目標梯度 $\nabla f = (-1,-1)$ 指向 x_1 和 x_2 增加的方向。無約束最優解會在無限遠處，所以約束必須活躍。
在圓 $x_1^2 + x_2^2 = 1$ 上：- $\nabla f = (-1,-1)$ - $\nabla g = (2x_1, 2x_2)$
對於 KKT：$(-1,-1) + \lambda(2x_1, 2x_2) = (0,0)$
這給出：$-1 + 2\lambda x_1 = 0$ 且 $-1 + 2\lambda x_2 = 0$
所以 $x_1 = x_2 = \frac{1}{2\lambda}$
由約束：$2\left(\frac{1}{2\lambda}\right)^2 = 1$，給出 $\lambda = \frac{1}{\sqrt{2}}$
解：$x^* = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$
幾何洞察：在最優解處，∇f 和 ∇g 平行且指向相反方向，表明目標函數等值線與約束邊界相切。

1.4.4 法錐和切錐解釋

定義 **1.6** (切錐). 可行集 $\mathcal{S}$ 在點 x^* 處的切錐是：

$$T_{\mathcal{S}}(x^*) = \{d : \exists t_k \downarrow 0, \exists d_k \to d \text{ 使得} x^* + t_k d_k \in \mathcal{S}\}$$

定義 **1.7** (法錐). 可行集 $\mathcal{S}$ 在點 x^* 處的法錐是：

$$N_{\mathcal{S}}(x^*) = \{g : g^T d \le 0 \text{ 對所有} d \in T_{\mathcal{S}}(x^*)\}$$

定理

最優性的幾何特徵
可行點 x^* 是最優的當且僅當：

$$\nabla f(x^*) \in N_{\mathcal{S}}(x^*)$$

這意味著目標梯度必須在可行集的法錐中。

1.4.5 Fritz John 與 KKT 條件

例子 **1.9** (約束資格失效)**.** 考慮病理情況：

$$\min \quad f(x) = x_2 \tag{1.54}$$

$$\text{受約束於} \quad g_1(x) = x_1^3 - x_2 \leq 0 \tag{1.55}$$

$$g_2(x) = -x_1^3 - x_2 \leq 0 \tag{1.56}$$

在 $x^* = (0,0)$：- 兩個約束都活躍：$g_1(0,0) = g_2(0,0) = 0$ - $\nabla g_1(0,0) = (0,-1)$ 且 $\nabla g_2(0,0) = (0,-1)$
約束梯度線性相關，違反 LICQ。
Fritz John 條件：存在 $\lambda_0, \lambda_1, \lambda_2 \geq 0$，不全為零，使得：

$$\lambda_0 \nabla f + \lambda_1 \nabla g_1 + \lambda_2 \nabla g_2 = 0$$

這給出：$\lambda_0(0,1) + (\lambda_1 + \lambda_2)(0,-1) = (0,0)$
解：$\lambda_0 = \lambda_1 + \lambda_2$，任何非負值都可以。
幾何洞察：當約束資格失效時，法錐可能有「空隙」，阻止標準 KKT 條件成立。

實際應用

資源配置
製造公司必須在不同產品之間分配有限資源。問題可以表述為：

$$\max \quad \sum_{i=1}^{n} p_i x_i - \sum_{i=1}^{n} c_i x_i^2 \tag{1.57}$$

$$\text{受約束於} \quad \sum_{i=1}^{n} a_{ji} x_i \leq b_j, \quad j = 1, \ldots, m \tag{1.58}$$

$$x_i \geq 0, \quad i = 1, \ldots, n \tag{1.59}$$

其中 x_i 是生產水平，p_i 是單位價格，c_i 代表生產成本，a_{ji} 是資源需求。
幾何解釋：在最優配置處，邊際利潤梯度是資源約束梯度的非負組合，意味著沒有滿足所有資源限制的有利方向存在。

Python 程式碼

```
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt
```

```
def geometric_kkt_visualization():
    """
    可視化 KKT 條件的幾何解釋
    """

    # 範例：min (x1-2)^2 + (x2-2)^2 受約束於 x1 + x2 <= 2

    def objective_constrained(x):
        return (x[0] - 2)**2 + (x[1] - 2)**2

    def constraint_ineq(x):
        return 2 - x[0] - x[1]  # x1 + x2 <= 2

    # 求解最佳化問題
    constraints = [
        {'type': 'ineq', 'fun': constraint_ineq},
        {'type': 'ineq', 'fun': lambda x: x[0]},  # x1 >= 0
        {'type': 'ineq', 'fun': lambda x: x[1]}   # x2 >= 0
    ]

    x0 = np.array([1.0, 1.0])
    result = minimize(objective_constrained, x0, method='SLSQP',
                      constraints=constraints)

    # 建立可視化
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

    # 圖 1：目標函數等高線和約束
    x1 = np.linspace(-0.5, 3, 100)
    x2 = np.linspace(-0.5, 3, 100)
    X1, X2 = np.meshgrid(x1, x2)
    Z = (X1 - 2)**2 + (X2 - 2)**2

    # 等高線圖
    contours = ax1.contour(X1, X2, Z, levels=15, alpha=0.7, colors='blue')
    ax1.clabel(contours, inline=True, fontsize=8)

    # 約束邊界
    x1_constraint = np.linspace(0, 2.5, 100)
    x2_constraint = 2 - x1_constraint

    # 可行域
    x1_fill = np.linspace(0, 2, 100)
    x2_fill = 2 - x1_fill
    x2_lower = np.zeros_like(x1_fill)

    ax1.fill_between(x1_fill, x2_lower, x2_fill, alpha=0.3,
                     color='lightgreen', label='可行域')
    ax1.plot(x1_constraint, x2_constraint, 'r-', linewidth=2,
            label='$x_1 + x_2 = 2$')

    # 標記點
    ax1.plot(2, 2, 'bs', markersize=10, label='無約束最優解 (2,2)')
```

```
    ax1.plot(result.x[0], result.x[1], 'ro', markersize=10,
            label=f'約束最優解 ({result.x[0]:.1f},{result.x[1]:.1f})')

    ax1.set_xlim(-0.5, 3)
    ax1.set_ylim(-0.5, 3)
    ax1.set_xlabel('$x_1$')
    ax1.set_ylabel('$x_2$')
    ax1.set_title('約束最佳化問題')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # 圖 2：最優點處的梯度向量
    x_opt = result.x

    # 最優點處的目標梯度
    grad_f = 2 * (x_opt - np.array([2, 2]))

    # 約束梯度
    grad_g = np.array([1, 1])  # x1 + x2 的梯度

    # 畫梯度向量
    ax2.quiver(x_opt[0], x_opt[1], grad_f[0], grad_f[1],
              angles='xy', scale_units='xy', scale=1, color='blue',
              width=0.005, label='$\\nabla f(x^*)$')
    ax2.quiver(x_opt[0], x_opt[1], -grad_g[0], -grad_g[1],
              angles='xy', scale_units='xy', scale=1, color='red',
              width=0.005, label='$-\\nabla g(x^*)$')

    # 顯示梯度平行（KKT 條件）
    lambda_val = np.dot(grad_f, grad_g) / np.dot(grad_g, grad_g)
    scaled_grad_g = lambda_val * grad_g

    ax2.quiver(x_opt[0], x_opt[1], scaled_grad_g[0], scaled_grad_g[1],
              angles='xy', scale_units='xy', scale=1, color='green',
              width=0.005, linestyle='--',
              label=f'$\\lambda \\nabla g(x^*)$ ($\\lambda$={lambda_val:.2f})')

    # 畫最優點附近的約束線
    x1_local = np.linspace(x_opt[0]-0.5, x_opt[0]+0.5, 100)
    x2_local = 2 - x1_local
    ax2.plot(x1_local, x2_local, 'k-', linewidth=2, alpha=0.7,
            label='約束邊界')

    # 標記最優點
    ax2.plot(x_opt[0], x_opt[1], 'ro', markersize=10, label='最優點')

    ax2.set_xlim(x_opt[0]-0.8, x_opt[0]+0.8)
    ax2.set_ylim(x_opt[1]-0.8, x_opt[1]+0.8)
    ax2.set_xlabel('$x_1$')
    ax2.set_ylabel('$x_2$')
    ax2.set_title('KKT 條件：$\\nabla f + \\lambda \\nabla g = 0$')
    ax2.legend()
    ax2.grid(True, alpha=0.3)
```

```
    ax2.set_aspect('equal')

    plt.tight_layout()
    plt.show()

    return result

def tangent_normal_cone_example():
    """
    說明切錐和法錐
    """

    # 考慮可行集：x1^2 + x2^2 <= 1, x1 >= 0
    fig, ax = plt.subplots(1, 1, figsize=(10, 8))

    # 畫可行域
    theta = np.linspace(0, np.pi, 100)
    x1_circle = np.cos(theta)
    x2_circle = np.sin(theta)

    # 添加底部部分
    x1_full = np.concatenate([x1_circle, [0]])
    x2_full = np.concatenate([x2_circle, [0]])

    ax.fill(x1_full, x2_full, alpha=0.3, color='lightblue',
           label='可行域')
    ax.plot(x1_circle, x2_circle, 'b-', linewidth=2, label='約束邊界')
    ax.axvline(x=0, color='black', linewidth=2, ymin=0, ymax=0.5)

    # 邊界上的點
    point = np.array([0, 1])
    ax.plot(point[0], point[1], 'ro', markersize=10, label='邊界上的點')

    # 切錐方向
    tangent_directions = np.array([[1, 0], [0, -1], [0.5, -0.866]])
    for i, direction in enumerate(tangent_directions):
        ax.arrow(point[0], point[1], 0.3*direction[0], 0.3*direction[1],
                head_width=0.05, head_length=0.05, fc='green', ec='green',
                alpha=0.7)

    # 法錐（向外法向量）
    normal_directions = np.array([[0, 1], [-1, 0]])
    for direction in normal_directions:
        ax.arrow(point[0], point[1], 0.4*direction[0], 0.4*direction[1],
                head_width=0.05, head_length=0.05, fc='red', ec='red',
                alpha=0.7, linewidth=2)

    # 添加文字註解
    ax.text(0.2, 0.8, '切錐', color='green', fontsize=12, fontweight='bold')
    ax.text(-0.3, 1.2, '法錐', color='red', fontsize=12, fontweight='bold')

    ax.set_xlim(-1.5, 1.5)
    ax.set_ylim(-0.5, 1.5)
```

```
    ax.set_xlabel('$x_1$')
    ax.set_ylabel('$x_2$')
    ax.set_title('邊界點處的切錐和法錐')
    ax.legend()
    ax.grid(True, alpha=0.3)
    ax.set_aspect('equal')

    plt.show()

# 執行可視化
print("幾何 KKT 可視化：")
result = geometric_kkt_visualization()
print(f"最優解：{result.x}")
print(f"最優值：{result.fun}")

print("\n切錐和法錐說明：")
tangent_normal_cone_example()
```

探索

進階幾何研究
進一步探索這些幾何概念：

1. 約束曲率效應：研究約束邊界的曲率如何影響最優解的幾何性質，特別是對非線性約束。
2. 退化性分析：研究多個約束活躍且約束梯度接近線性相關的情況。觀察小擾動如何改變活躍集。
3. 二階幾何：檢查拉格朗日函數的 Hessian 在確定臨界點性質（最小值、最大值或鞍點）中的作用。
4. 參數分析：研究當約束參數變化時最優解路徑如何變化，觀察活躍集改變的分岔點。

練習 **1.3.** 考慮問題：最小化 $x_1^2 + x_2^2$ 受約束於 $x_1^2 + x_2^2 \leq 1$ 和 $x_1 + x_2 \geq 1$。建立 KKT 條件並圖解求解。識別最優點處目標梯度與約束梯度之間的幾何關係。

練習 **1.4.** 證明如果 x^* 滿足凸最佳化問題的 KKT 條件，則 x^* 是全局最小值。

Chapter 2

線性規劃

2.1 線性規劃簡介

線性規劃（LP）是約束最佳化問題的基本類別，其目標函數和約束都是線性的。

定義 **2.1** (線性規劃問題)**.** 線性規劃問題的標準形式是：

$$\begin{aligned} \min \quad & c^T x && (2.1)\\ \text{受約束於} \quad & Ax = b && (2.2)\\ & x \geq 0 && (2.3) \end{aligned}$$

其中 $c \in \mathbb{R}^n$，$A \in \mathbb{R}^{m \times n}$，$b \in \mathbb{R}^m$。

2.2 線性規劃基本定理

定理

線性規劃基本定理
如果線性規劃問題有最優解，則存在一個基本可行解（可行域的頂點）是最優解。

2.2.1 基本定理的說明例子

例子 **2.1** (兩變數線性規劃)**.** 考慮問題：

$$\begin{aligned} \max \quad & 3x_1 + 2x_2 && (2.4)\\ \text{受約束於} \quad & x_1 + 2x_2 \leq 4 && (2.5)\\ & 2x_1 + x_2 \leq 4 && (2.6)\\ & x_1, x_2 \geq 0 && (2.7) \end{aligned}$$

可行域是一個多邊形，頂點為：

- $(0,0)$：目標值 $= 0$
- $(0,2)$：目標值 $= 4$
- $\left(\frac{4}{3}, \frac{4}{3}\right)$：目標值 $= 3 \cdot \frac{4}{3} + 2 \cdot \frac{4}{3} = \frac{20}{3}$

- $(2,0)$：目標值 $= 6$

最大值在頂點 $(2,0)$ 處達到，值為 6，證實了基本定理。

例子 **2.2** (無界線性規劃)**.** 考慮：

$$\begin{aligned} \max \quad & x_1 + x_2 && (2.8)\\ \text{受約束於} \quad & -x_1 + x_2 \leq 1 && (2.9)\\ & x_1, x_2 \geq 0 && (2.10) \end{aligned}$$

可行域在方向 $(1,1)$ 上是無界的，這也是目標函數改進的方向。因此，問題向上無界。
幾何驗證：對於任何 $t \geq 0$，點 $(t, t+1)$ 都是可行的，並給出目標值 $2t+1 \to \infty$ 當 $t \to \infty$。

例子 **2.3** (不可行線性規劃)**.** 考慮：

$$\begin{aligned} \min \quad & x_1 + x_2 && (2.11)\\ \text{受約束於} \quad & x_1 + x_2 \leq 1 && (2.12)\\ & x_1 + x_2 \geq 2 && (2.13)\\ & x_1, x_2 \geq 0 && (2.14) \end{aligned}$$

約束 $x_1 + x_2 \leq 1$ 和 $x_1 + x_2 \geq 2$ 是矛盾的，使得可行集為空。

例子 **2.4** (退化基本可行解)**.** 考慮問題：

$$\begin{aligned} \max \quad & x_1 + x_2 && (2.15)\\ \text{受約束於} \quad & x_1 \leq 2 && (2.16)\\ & x_2 \leq 2 && (2.17)\\ & x_1 + x_2 \leq 3 && (2.18)\\ & x_1, x_2 \geq 0 && (2.19) \end{aligned}$$

在點 $(1,2)$，三個約束是活躍的：$x_2 = 2$，$x_1 + x_2 = 3$，以及由這兩個約束隱含的約束。這代表一個退化的基本可行解，其中超過 n 個約束是活躍的。

2.3 線性規劃的多種求解方法

2.3.1 單體法

單體法是求解線性規劃問題的經典算法，但它不是唯一的方法。

定義 **2.2** (基本可行解)**.** 基本可行解對應可行域的一個頂點，其中恰好有 n 個約束是活躍的（包括非負性約束）。

例子 **2.5** (單體法說明)**.** 對於標準形式問題：

$$\begin{aligned} \min \quad & -3x_1 - 2x_2 && (2.20)\\ \text{受約束於} \quad & x_1 + 2x_2 + s_1 = 4 && (2.21)\\ & 2x_1 + x_2 + s_2 = 4 && (2.22)\\ & x_1, x_2, s_1, s_2 \geq 0 && (2.23) \end{aligned}$$

初始表格：

	x_1	x_2	s_1	s_2	右端項
s_1	1	2	1	0	4
s_2	2	1	0	1	4
z	3	2	0	0	0

迭代 **1**：x_1 進入（最負的簡約成本），s_2 離開（最小比值測試：$\min\{4/1, 4/2\} = 2$）。
旋轉後：

	x_1	x_2	s_1	s_2	右端項
s_1	0	$\frac{3}{2}$	1	$-\frac{1}{2}$	2
x_1	1	$\frac{1}{2}$	0	$\frac{1}{2}$	2
z	0	$\frac{1}{2}$	0	$-\frac{3}{2}$	-6

由於所有簡約成本都非負，最優解是 $x_1 = 2$，$x_2 = 0$，目標值為 6。

2.3.2 內點法

內點法通過可行域的內部逼近最優解。

定義 **2.3** (線性規劃的原始-對偶內點法)**.** 原始-對偶方法同時求解原始和對偶問題：

$$\text{原始：} \quad \min c^T x \text{ 受約束於} Ax = b, x \geq 0 \tag{2.24}$$

$$\text{對偶：} \quad \max b^T y \text{ 受約束於} A^T y + s = c, s \geq 0 \tag{2.25}$$

中心路徑由 $\mu > 0$ 參數化，滿足：

$$Ax = b \tag{2.26}$$

$$A^T y + s = c \tag{2.27}$$

$$xs = \mu e \tag{2.28}$$

$$x, s > 0 \tag{2.29}$$

其中 xs 表示逐元素乘法。

例子 **2.6** (內點路徑)**.** 對於問題：

$$\min \quad -x_1 - x_2 \tag{2.30}$$

$$\text{受約束於} \quad x_1 + x_2 \leq 2 \tag{2.31}$$

$$x_1, x_2 \geq 0 \tag{2.32}$$

對於遞減的 μ，中心路徑解為：

- $\mu = 1$：$(x_1, x_2) \approx (0.5, 0.5)$
- $\mu = 0.1$：$(x_1, x_2) \approx (0.9, 0.9)$
- $\mu \to 0$：$(x_1, x_2) \to (1, 1)$（最優頂點）

路徑保持在內部直到 $\mu \to 0$，然後收斂到最優頂點。

2.3.3 對偶單體法

對偶單體法保持對偶可行性同時尋求原始可行性。

例子 **2.7** (對偶單體法應用)**.** 考慮添加約束後的問題：

$$\begin{aligned} \min \quad & x_1 + x_2 && (2.33)\\ \text{受約束於} \quad & x_1 + 2x_2 \geq 4 && (2.34)\\ & 2x_1 + x_2 \geq 4 && (2.35)\\ & x_1, x_2 \geq 0 && (2.36) \end{aligned}$$

轉換為帶鬆弛變數 $s_1, s_2 \leq 0$ 的標準形式：

$$\begin{aligned} \min \quad & x_1 + x_2 && (2.37)\\ \text{受約束於} \quad & x_1 + 2x_2 - s_1 = 4 && (2.38)\\ & 2x_1 + x_2 - s_2 = 4 && (2.39)\\ & x_1, x_2 \geq 0, \quad s_1, s_2 \leq 0 && (2.40) \end{aligned}$$

對偶單體法從 $x_1 = x_2 = 0$ 開始，給出 $s_1 = s_2 = -4 < 0$（原始不可行但對偶可行），然後迭代恢復原始可行性。

2.3.4 網絡單體法

對於網絡流問題，網絡單體法利用特殊結構。

例子 **2.8** (最小成本流問題)**.** 考慮運輸網絡：

- 供應節點：$s_1 = 10$，$s_2 = 15$
- 需求節點：$d_1 = 8$，$d_2 = 12$，$d_3 = 5$
- 弧成本：從節點 i 到節點 j 的流 c_{ij}

問題變為：

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} && (2.41)\\ \text{受約束於} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \forall i && (2.42)\\ & x_{ij} \geq 0 \quad \forall (i,j) \in A && (2.43) \end{aligned}$$

網絡單體法維護生成樹解，通過添加和移除弧來維護樹結構進行旋轉。

2.3.5 橢球法

橢球法作為線性規劃第一個多項式時間算法具有理論重要性。

例子 **2.9** (橢球法概念)**.** 從包含最優解的橢球 E_0 開始，該方法：

1. 查詢中心當前橢球 E_k 的中心 x_k
2. 檢查可行性：如果 x_k 是最優的，停止

3. 生成切割平面：找到違反的約束 $a^T x \leq b$，其中 $a^T x_k > b$

4. 更新橢球：構造更小的橢球 E_{k+1}，包含 E_k 與半空間 $a^T x \leq b$ 的交集

對於在中心 $(3,1)$ 處違反的約束 $x_1 + x_2 \leq 2$：

- 當前橢球中心：$(3,1)$
- 切割平面：$x_1 + x_2 \leq 2$
- 新橢球中心向可行域偏移

該方法在多項式時間內收斂，但由於常數因子較差而不實用。

2.4 對偶理論

定理

強對偶定理
如果原始問題有最優解且目標值為 z^*，則對偶問題也有最優解且目標值相同。

例子 **2.10** (對偶分析). 原始：

$$\begin{aligned} \max \quad & 3x_1 + 2x_2 && (2.44)\\ \text{受約束於} \quad & x_1 + 2x_2 \leq 4 && (2.45)\\ & 2x_1 + x_2 \leq 4 && (2.46)\\ & x_1, x_2 \geq 0 && (2.47) \end{aligned}$$

對偶：

$$\begin{aligned} \min \quad & 4y_1 + 4y_2 && (2.48)\\ \text{受約束於} \quad & y_1 + 2y_2 \geq 3 && (2.49)\\ & 2y_1 + y_2 \geq 2 && (2.50)\\ & y_1, y_2 \geq 0 && (2.51) \end{aligned}$$

求解過程：
首先，通過檢查頂點求解原始問題：

- $(0,0)$：目標值 $= 0$
- $(0,2)$：目標值 $= 4$
- $(2,0)$：目標值 $= 6$
- $\left(\frac{4}{3}, \frac{4}{3}\right)$：目標值 $= 3 \cdot \frac{4}{3} + 2 \cdot \frac{4}{3} = \frac{20}{3} \approx 6.67$

原始最優解是 $x^* = \left(\frac{4}{3}, \frac{4}{3}\right)$，值為 $\frac{20}{3}$。
對於對偶，使用互補鬆弛性，由於兩個原始約束都緊：

$$\begin{aligned} y_1 + 2y_2 &= 3 && (2.52)\\ 2y_1 + y_2 &= 2 && (2.53) \end{aligned}$$

求解此系統：從第二個方程：$y_1 = 1 - \frac{y_2}{2}$ 代入：$\left(1 - \frac{y_2}{2}\right) + 2y_2 = 3$ 簡化：$1 + \frac{3y_2}{2} = 3$，所以 $y_2 = \frac{4}{3}$ 且 $y_1 = \frac{1}{3}$
對偶值：$4 \cdot \frac{1}{3} + 4 \cdot \frac{4}{3} = \frac{4}{3} + \frac{16}{3} = \frac{20}{3}$

例子 **2.11** (對偶的經濟解釋)**.** 考慮生產計劃問題：

原始解釋：通過生產 x_1 單位產品 1 和 x_2 單位產品 2 來最大化利潤，受資源限制。

對偶解釋：對偶變數 y_1, y_2 代表資源的影子價格。對偶問題問：以影子價格評估資源時，所需資源的最小成本是多少？

在最優性時，影子價格 y_j^* 代表額外一單位資源 j 的邊際價值。

實際應用

實際中的多方法方案
現實世界的線性規劃通常採用多種方法：

1. 預處理：問題簡化和重新表述
2. 方法選擇：
   - 小到中等問題使用單體法
   - 大規模問題使用內點法
   - 網絡流結構使用網絡單體法
   - 敏感性分析使用對偶單體法
3. 後處理：解的解釋和敏感性分析
4. 熱啟動：當問題參數變化時使用之前的解

現代 LP 求解器根據問題特徵自動選擇最合適的方法。

探索

LP 方法的比較分析
調查不同 LP 求解方法的以下方面：

1. 理論複雜性：比較最壞情況和平均情況複雜性
2. 實際性能：在不同問題類別上進行基準測試
3. 數值穩定性：分析條件數較差問題的行為
4. 並行化潛力：檢查哪些方法可以從並行計算中受益
5. 記憶體需求：比較大規模問題的存儲需求

練習 **2.1.** 考慮線性規劃：

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 + x_3 && (2.54)\\ \text{受約束於} \quad & x_1 + x_2 + x_3 \le 6 && (2.55)\\ & 2x_1 + x_2 \le 8 && (2.56)\\ & x_2 + 2x_3 \le 7 && (2.57)\\ & x_1, x_2, x_3 \ge 0 && (2.58) \end{aligned}$$

制定對偶問題並使用互補鬆弛性驗證 $x^* = (3, 2, 1)$ 和 $y^* = (1, 1, 1)$ 分別是原始和對偶的最優解。

練習 **2.2.** 證明如果原始和對偶線性規劃都有可行解，則兩者都有最優解且目標值相等（強對偶定理）。

練習 **2.3.** 對於有 n 個節點和 m 條弧的網絡流問題，解釋為什麼網絡單體法在生成樹結構中恰好維護 $n-1$ 個基變數，並描述在此上下文中如何處理退化性。

Chapter 3

二次規劃與進階方法

3.1 二次規劃

二次規劃（QP）通過允許目標函數中的二次項擴展了線性規劃，同時保持線性約束。

定義 **3.1** (二次規劃問題)**.** 標準二次規劃問題是：

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Qx + c^T x && (3.1)\\ \text{受約束於} \quad & Ax \leq b && (3.2)\\ & Ex = d && (3.3)\\ & x \geq 0 && (3.4) \end{aligned}$$

其中 $Q \in \mathbb{R}^{n\times n}$ 是對稱矩陣。

3.1.1 二次規劃的分類

定義 **3.2** (二次規劃的類型)**.** 根據矩陣 Q 的性質：

- 凸二次規劃：$Q \succeq 0$（半正定）
- 嚴格凸二次規劃：$Q \succ 0$（正定）
- 不定二次規劃：Q 具有正負特徵值
- 凹二次規劃：$Q \preceq 0$（半負定）

定理

凸二次規劃的全局最優性
對於凸二次規劃，其中 $Q \succeq 0$，任何局部最小值都是全局最小值。此外，如果 $Q \succ 0$，則全局最小值是唯一的（如果存在）。

3.1.2 凸二次規劃

當 $Q \succeq 0$（半正定）時，問題是凸的，KKT 條件是最優性的必要且充分條件。

定理

二次規劃的 **KKT** 條件
對於具有 $Q \succeq 0$ 的二次規劃問題，點 x^* 是最優的當且僅當存在乘數 $\lambda^* \geq 0$、μ^* 和 $\nu^* \geq 0$ 使得：

$$\text{駐點條件：} \quad Qx^* + c + A^T\lambda^* + E^T\mu^* - \nu^* = 0 \tag{3.5}$$
$$\text{原始可行性：} \quad Ax^* \leq b, \quad Ex^* = d, \quad x^* \geq 0 \tag{3.6}$$
$$\text{對偶可行性：} \quad \lambda^* \geq 0, \quad \nu^* \geq 0 \tag{3.7}$$
$$\text{互補鬆弛：} \quad \lambda_i^*(A_i x^* - b_i) = 0, \quad \nu_j^* x_j^* = 0 \tag{3.8}$$

例子 **3.1** (簡單二次規劃)**.** 考慮問題：

$$\min \quad \frac{1}{2}(x_1^2 + x_2^2) + x_1 + x_2 \tag{3.9}$$
$$\text{受約束於} \quad x_1 + x_2 \leq 1 \tag{3.10}$$
$$x_1, x_2 \geq 0 \tag{3.11}$$

這裡 $Q = I$，$c = (1,1)^T$，$A = (1,1)$，$b = 1$。
求解過程：
拉格朗日函數是：

$$\mathcal{L} = \frac{1}{2}(x_1^2 + x_2^2) + x_1 + x_2 + \lambda(x_1 + x_2 - 1) - \nu_1 x_1 - \nu_2 x_2$$

KKT 條件：

$$x_1 + 1 + \lambda - \nu_1 = 0 \tag{3.12}$$
$$x_2 + 1 + \lambda - \nu_2 = 0 \tag{3.13}$$
$$x_1 + x_2 \leq 1, \quad \lambda \geq 0, \quad \lambda(x_1 + x_2 - 1) = 0 \tag{3.14}$$
$$x_1, x_2 \geq 0, \quad \nu_1, \nu_2 \geq 0, \quad \nu_1 x_1 = \nu_2 x_2 = 0 \tag{3.15}$$

情況 **1**：內點解（$\nu_1 = \nu_2 = 0$，$\lambda = 0$）由駐點條件：$x_1 = x_2 = -1 < 0$ （違反非負性）
情況 **2**：約束活躍（$x_1 + x_2 = 1$，$\nu_1 = \nu_2 = 0$）由駐點條件：$x_1 = x_2 = -1 - \lambda$ 由約束：$2(-1-\lambda) = 1 \Rightarrow \lambda = -\frac{3}{2} < 0$
情況 **3**：$x_1 = 0$（所以 $\nu_1 \geq 0$），$\nu_2 = 0$ 由駐點條件：$1 + \lambda - \nu_1 = 0$ 且 $x_2 + 1 + \lambda = 0$ 所以 $x_2 = -1 - \lambda$ 且 $\nu_1 = 1 + \lambda$
對於可行性：$x_2 \geq 0 \Rightarrow \lambda \leq -1$ 且 $\nu_1 \geq 0 \Rightarrow \lambda \geq -1$ 因此 $\lambda = -1$，給出 $x_2 = 0$，$\nu_1 = 0$。
由對稱性，最優解是 $x^* = (0,0)$，目標值為 0。

例子 **3.2** (投資組合最佳化二次規劃)**.** 經典的 Markowitz 投資組合最佳化問題：

$$\min \quad \frac{1}{2} w^T \Sigma w \tag{3.16}$$
$$\text{受約束於} \quad \mathbf{1}^T w = 1 \tag{3.17}$$
$$\mu^T w \geq r_{\min} \tag{3.18}$$
$$w \geq 0 \tag{3.19}$$

其中 w 是投資組合權重向量，Σ 是協方差矩陣，μ 是預期報酬向量。

經濟解釋：在預算約束、最小報酬要求和無賣空條件下最小化投資組合變異數。
KKT 條件：

$$\Sigma w^* + \lambda \mathbf{1} - \nu\mu - \gamma = 0 \tag{3.20}$$
$$\mathbf{1}^T w^* = 1 \tag{3.21}$$
$$\mu^T w^* \geq r_{\min}, \quad \nu \geq 0, \quad \nu(\mu^T w^* - r_{\min}) = 0 \tag{3.22}$$
$$w^* \geq 0, \quad \gamma \geq 0, \quad \gamma_i w_i^* = 0 \tag{3.23}$$

其中 λ 是預算約束的拉格朗日乘數，ν 是報酬約束的乘數，γ 是非負性約束的乘數。

例子 **3.3** (僅有等式約束的二次規劃)**.** 考慮：

$$\min \quad \frac{1}{2}x^T Q x + c^T x \tag{3.24}$$
$$\text{受約束於} \quad Ax = b \tag{3.25}$$

其中 $Q \succ 0$。KKT 系統變為：

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

由於 $Q \succ 0$ 使矩陣非奇異，此系統有唯一解。
顯式解：

$$x^* = Q^{-1}(A^T(AQ^{-1}A^T)^{-1}b - c)$$
$$\lambda^* = (AQ^{-1}A^T)^{-1}(AQ^{-1}c + b)$$

3.2 活躍集方法

活躍集方法通過迭代地使用在當前迭代點活躍的約束子集來求解二次規劃。

定義 **3.3** (活躍集方法算法)**.** 1. 初始化：從可行點 x_0 和工作集 $\mathcal{W}_0$ 開始

2. 求解等式二次規劃：以工作集約束為等式求解子問題
3. 最優性檢查：檢查當前工作集的 KKT 條件
4. 添加/移除約束：根據違反情況更新工作集
5. 迭代：重複直到收斂

定理

活躍集方法的有限收斂性
對於嚴格凸二次規劃，活躍集方法在有限次迭代內收斂到最優解。

例子 **3.4** (活躍集方法追蹤)**.** 考慮二次規劃：

$$\min \quad \frac{1}{2}(x_1^2 + x_2^2) - 2x_1 - 6x_2 \tag{3.26}$$
$$\text{受約束於} \quad x_1 + x_2 \leq 2 \tag{3.27}$$
$$-x_1 + 2x_2 \leq 2 \tag{3.28}$$
$$2x_1 + x_2 \leq 3 \tag{3.29}$$
$$x_1, x_2 \geq 0 \tag{3.30}$$

迭代 **0**：從 $x_0 = (0,0)$ 開始，工作集 $\mathcal{W}_0 = \{4,5\}$（兩個非負性約束）

以 $x_1 = x_2 = 0$ 求解等式二次規劃：拉格朗日梯度：$\nabla\mathcal{L} = (1-\nu_1, 6-\nu_2) = (0,0)$ 這給出 $\nu_1 = 1 > 0$，$\nu_2 = 6 > 0$

檢查最優性：計算 d 以在保持活躍約束下改進目標。方向 d 必須滿足 $d_1 = d_2 = 0$，所以 $d = 0$。檢查是否可以移除約束。

由於 $\nu_2 = 6$ 最大，從工作集中移除約束 $x_2 \geq 0$。

迭代 **1**：工作集 $\mathcal{W}_1 = \{4\}$（只有 $x_1 \geq 0$）

以 $x_1 = 0$ 求解最優 x：$\min \frac{1}{2}x_2^2 - 6x_2$ 給出 $x_2 = 6$

檢查約束：$x_1 = 0$，$x_2 = 6$ 違反 $x_1 + x_2 \leq 2$。

將最嚴重違反的約束添加到工作集。

繼續直到收斂...

3.3 二次規劃的內點法

內點法通過可行域的內部逼近最優解。

定義 **3.4** (二次規劃的對數障礙函數)**.** 對於具有不等式約束 $Ax \leq b$ 的二次規劃，障礙函數是：

$$\phi(x) = -\sum_{i=1}^{m} \log(b_i - A_i x)$$

障礙子問題是：

$$\min_x \frac{1}{2}x^T Q x + c^T x + \mu\phi(x)$$

其中 $\mu > 0$ 是障礙參數。

定理

二次規劃的中心路徑
對於嚴格凸二次規劃，中心路徑 $x(\mu)$ 對所有 $\mu > 0$ 都是良定義的，並且當 $\mu \to 0^+$ 時收斂到最優解。

例子 **3.5** (二次規劃的障礙法)**.** 考慮：

$$\min \quad \frac{1}{2}(x_1^2 + x_2^2) \tag{3.31}$$

$$\text{受約束於} \quad x_1 + x_2 \leq 1 \tag{3.32}$$

$$x_1, x_2 \geq 0 \tag{3.33}$$

障礙子問題是：

$$\min \frac{1}{2}(x_1^2 + x_2^2) - \mu\log(1 - x_1 - x_2) - \mu\log x_1 - \mu\log x_2$$

一階條件：

$$x_1 + \frac{\mu}{1 - x_1 - x_2} - \frac{\mu}{x_1} = 0 \tag{3.34}$$

$$x_2 + \frac{\mu}{1 - x_1 - x_2} - \frac{\mu}{x_2} = 0 \tag{3.35}$$

由對稱性，在中心路徑上 $x_1 = x_2$。令 $x_1 = x_2 = t$：

$$t + \frac{\mu}{1-2t} - \frac{\mu}{t} = 0$$

求解：$t^2(1-2t) + \mu t - \mu(1-2t) = 0$
當 $\mu \to 0$：$t^2(1-2t) = 0$，給出 $t = 0$ 或 $t = \frac{1}{2}$。由於需要 $t > 0$，極限點是 $x^* = (0, 0)$。

3.3.1 原始-對偶內點法

原始-對偶方法在保持不等式乘數正性的同時直接求解 KKT 系統。

定理

原始-對偶方法的牛頓方向
對於標準形式的二次規劃，牛頓方向通過求解以下方程獲得：

$$\begin{bmatrix} Q & A^T & E^T \\ \mathrm{diag}(\lambda)A & \mathrm{diag}(Ax-b) & 0 \\ E & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\mu \end{bmatrix} = - \begin{bmatrix} Qx + c + A^T\lambda + E^T\mu \\ \mathrm{diag}(\lambda)(Ax-b) - \sigma\mu e \\ Ex - d \end{bmatrix} \tag{3.36}$$

其中 $\sigma \in (0, 1)$ 是中心化參數。

例子 **3.6** (原始-對偶步長計算)**.** 對於上述系統，當 $E = 0$（無等式約束）時，簡化的牛頓系統變為：

$$\begin{bmatrix} Q & A^T \\ \Lambda A & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -Qx - c - A^T\lambda \\ \sigma\mu e - \Lambda S e \end{bmatrix}$$

其中 $\Lambda = \mathrm{diag}(\lambda)$ 和 $S = \mathrm{diag}(s)$，$s = b - Ax$。
步長選擇以保持正性：$\alpha = 0.95 \min\{1, \min_i\{-\lambda_i/\Delta\lambda_i, -s_i/\Delta s_i\}\}$。

3.4 序列二次規劃 (SQP)

SQP 方法通過求解二次規劃子問題序列來解決非線性規劃問題。

定義 **3.5** (SQP 子問題)**.** 在第 k 次迭代，求解：

$$\min \quad \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \tag{3.37}$$

$$\text{受約束於} \quad \nabla g_i(x_k)^T d + g_i(x_k) \le 0, \quad i = 1, \ldots, m \tag{3.38}$$

$$\nabla h_j(x_k)^T d + h_j(x_k) = 0, \quad j = 1, \ldots, p \tag{3.39}$$

其中 B_k 是拉格朗日函數 Hessian 的近似。

定理

SQP 的超線性收斂
在適當的正則性條件下，如果 B_k 收斂到拉格朗日函數的真實 Hessian，則 SQP 超線性收斂到最優解。

例子 **3.7** (非線性問題的 SQP)**.** 考慮非線性規劃：

$$\min \quad x_1^2 + x_2^2 \tag{3.40}$$

$$\text{受約束於} \quad x_1^2 + x_2^2 - 1 = 0 \tag{3.41}$$

從 $x_0 = (0.5, 0.5)$ 開始：
迭代 **1**： $\nabla f(x_0) = (1,1)$，$\nabla h(x_0) = (1,1)$，$h(x_0) = -0.5$
SQP 子問題：

$$\min \quad d_1 + d_2 + \frac{1}{2}(d_1^2 + d_2^2) \tag{3.42}$$

$$\text{受約束於} \quad d_1 + d_2 - 0.5 = 0 \tag{3.43}$$

解：$d_1 = d_2 = 0.25$，給出 $x_1 = (0.75, 0.75)$。
檢查約束：$0.75^2 + 0.75^2 = 1.125 \neq 1$。
繼續迭代直到收斂到 $x^* = (1/\sqrt{2}, 1/\sqrt{2})$。

3.5 專門的二次規劃算法

3.5.1 二次規劃的共軛梯度法

對於具有 $Q \succ 0$ 的大規模二次規劃，可以直接應用共軛梯度法。

定義 **3.6** (二次規劃的共軛梯度法)**.** 求解 $Qx = -c$（無約束二次規劃）的共軛梯度法生成搜索方向 d_k，這些方向是 Q-共軛的：

$$d_i^T Q d_j = 0 \quad \text{對於} i \neq j$$

例子 **3.8** (無約束二次規劃的共軛梯度法)**.** 對於 $\min \frac{1}{2}x^T Q x + c^T x$，其中 $Q \succ 0$：
算法：

$$x_0 = \text{初始猜測} \tag{3.44}$$

$$r_0 = Qx_0 + c, \quad d_0 = -r_0 \tag{3.45}$$

$$\alpha_k = \frac{r_k^T r_k}{d_k^T Q d_k} \tag{3.46}$$

$$x_{k+1} = x_k + \alpha_k d_k \tag{3.47}$$

$$r_{k+1} = r_k + \alpha_k Q d_k \tag{3.48}$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{3.49}$$

$$d_{k+1} = -r_{k+1} + \beta_k d_k \tag{3.50}$$

該方法在 $n \times n$ 系統的最多 n 步內收斂。

3.5.2 梯度投影法

對於箱約束二次規劃：$\min \frac{1}{2}x^T Q x + c^T x$ 受約束於 $l \leq x \leq u$。

定義 **3.7** (梯度投影)**.** 梯度投影法在以下之間交替：

1. 梯度步：$y = x - \alpha \nabla f(x)$

2. 投影步：$x^+ = P_{[l,u]}(y)$ 其中 $P_{[l,u]}(y)_i = \max\{l_i, \min\{y_i, u_i\}\}$

例子 **3.9** (箱約束二次規劃的投影梯度法)**.** 考慮：

$$\min \quad \frac{1}{2}(x_1^2 + x_2^2) + x_1 + x_2 \tag{3.51}$$

$$\text{受約束於} \quad 0 \le x_1, x_2 \le 1 \tag{3.52}$$

從 $x_0 = (0.5, 0.5)$ 開始：
$\nabla f(x_0) = (1.5, 1.5)$ $y = (0.5, 0.5) - 0.5(1.5, 1.5) = (-0.25, -0.25)$ $x_1 = P_{[0,1]}(y) = (0, 0)$
在 $x_1 = (0, 0)$：$\nabla f(x_1) = (1, 1)$ 由於兩個分量都是正的且我們在下界處，這是最優的。

實際應用

支援向量機
SVM 的對偶形式導致二次規劃問題：

$$\max \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{3.53}$$

$$\text{受約束於} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.54}$$

$$0 \le \alpha_i \le C, \quad i = 1, \ldots, n \tag{3.55}$$

其中 $K(x_i, x_j)$ 是核函數，C 是正則化參數。
這個二次規劃具有可被專門算法（如 SMO，序列最小最佳化）利用的特殊結構。

實際應用

模型預測控制
在 MPC 中，每個時間步的最優控制問題變成二次規劃：

$$\min \quad \sum_{k=0}^{N-1} \left[x_k^T Q x_k + u_k^T R u_k \right] + x_N^T P x_N \tag{3.56}$$

$$\text{受約束於} \quad x_{k+1} = A x_k + B u_k \tag{3.57}$$

$$u_{\min} \le u_k \le u_{\max} \tag{3.58}$$

$$x_{\min} \le x_k \le x_{\max} \tag{3.59}$$

MPC 的實時性要求快速二次規劃求解器，通常使用熱啟動技術。

Python 程式碼

```python
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from scipy.linalg import solve

def solve_qp_examples():
    """
```

```
    求解二次規劃範例並比較不同方法
    """

    # 範例 1：簡單凸二次規劃
    def qp_objective(x):
        Q = np.array([[1, 0], [0, 1]])
        c = np.array([1, 1])
        return 0.5 * x.T @ Q @ x + c.T @ x

    def qp_constraint(x):
        return 1 - x[0] - x[1]  # x1 + x2 <= 1

    constraints = [
        {'type': 'ineq', 'fun': qp_constraint},
        {'type': 'ineq', 'fun': lambda x: x[0]},  # x1 >= 0
        {'type': 'ineq', 'fun': lambda x: x[1]}   # x2 >= 0
    ]

    x0 = np.array([0.3, 0.3])
    result1 = minimize(qp_objective, x0, method='SLSQP', constraints=constraints)

    print("範例 1 - 簡單凸二次規劃：")
    print(f"最優解：{result1.x}")
    print(f"最優值：{result1.fun}")
    print(f"理論解：[0, 0]，值 = 0")
    print()

    # 範例 2：僅等式約束的二次規劃
    # min 0.5 * x^T Q x + c^T x 受約束於 Ax = b
    Q = np.array([[2, 0, 0], [0, 2, 0], [0, 0, 2]])
    c = np.array([1, 1, 1])
    A = np.array([[1, 1, 1]])
    b = np.array([3])

    # 使用 KKT 系統直接求解
    n = Q.shape[0]
    m = A.shape[0]

    # 建構 KKT 矩陣
    KKT_matrix = np.zeros((n + m, n + m))
    KKT_matrix[:n, :n] = Q
    KKT_matrix[:n, n:] = A.T
    KKT_matrix[n:, :n] = A

    rhs = np.concatenate([-c, b])
    solution = solve(KKT_matrix, rhs)

    x_optimal = solution[:n]
    lambda_optimal = solution[n:]

    print("範例 2 - 等式約束二次規劃：")
    print(f"最優解：{x_optimal}")
    print(f"拉格朗日乘數：{lambda_optimal}")
```

```
    print(f"目標值：{0.5 * x_optimal.T @ Q @ x_optimal + c.T @ x_optimal}")
    print()

    # 範例 3：投資組合最佳化
    # 簡化的 Markowitz 模型
    n_assets = 3
    np.random.seed(42)

    # 生成協方差矩陣（確保正定）
    A_temp = np.random.randn(n_assets, n_assets)
    Sigma = A_temp.T @ A_temp + 0.1 * np.eye(n_assets)

    # 預期報酬
    mu = np.array([0.1, 0.12, 0.15])

    # 最小報酬要求
    r_min = 0.11

    def portfolio_objective(w):
        return 0.5 * w.T @ Sigma @ w

    portfolio_constraints = [
        {'type': 'eq', 'fun': lambda w: np.sum(w) - 1},  # 預算約束
        {'type': 'ineq', 'fun': lambda w: mu.T @ w - r_min},  # 最小報酬
        {'type': 'ineq', 'fun': lambda w: w[0]},  # w1 >= 0
        {'type': 'ineq', 'fun': lambda w: w[1]},  # w2 >= 0
        {'type': 'ineq', 'fun': lambda w: w[2]}   # w3 >= 0
    ]

    w0 = np.ones(n_assets) / n_assets
    result3 = minimize(portfolio_objective, w0, method='SLSQP',
                       constraints=portfolio_constraints)

    print("範例 3 - 投資組合最佳化：")
    print(f"最優權重：{result3.x}")
    print(f"投資組合風險（變異數）：{result3.fun}")
    print(f"預期報酬：{mu.T @ result3.x:.4f}")
    print(f"夏普比率：{(mu.T @ result3.x) / np.sqrt(result3.fun):.4f}")

    return result1, x_optimal, result3

def active_set_demo():
    """
    演示活躍集方法的概念
    """
    print("\n活躍集方法演示：")
    print("="*40)

    # 問題：min 0.5(x1^2 + x2^2) - 2x1 - 6x2
    # 受約束於：x1 + x2 <= 2, x1 >= 0, x2 >= 0

    def qp_obj_demo(x):
        return 0.5 * (x[0]**2 + x[1]**2) - 2*x[0] - 6*x[1]
```

```
    def qp_grad_demo(x):
        return np.array([x[0] - 2, x[1] - 6])

    # 無約束最優解
    x_unconstrained = np.array([2, 6])
    print(f"無約束最優解：{x_unconstrained}")
    print(f"檢查約束：x1 + x2 = {x_unconstrained[0] + x_unconstrained[1]} > 2（違反）")

    # 約束最優解
    constraints_demo = [
        {'type': 'ineq', 'fun': lambda x: 2 - x[0] - x[1]},
        {'type': 'ineq', 'fun': lambda x: x[0]},
        {'type': 'ineq', 'fun': lambda x: x[1]}
    ]

    x0_demo = np.array([0.5, 0.5])
    result_demo = minimize(qp_obj_demo, x0_demo, method='SLSQP',
                           jac=qp_grad_demo, constraints=constraints_demo)

    print(f"約束最優解：{result_demo.x}")
    print(f"最優值：{result_demo.fun}")
    print(f"活躍約束：x1 + x2 = {result_demo.x[0] + result_demo.x[1]:.6f}  2")

def barrier_method_demo():
    """
    演示障礙法的中心路徑
    """
    print("\n障礙法中心路徑演示：")
    print("="*40)

    # 問題：min 0.5(x1^2 + x2^2) 受約束於 x1 + x2 <= 1, x1, x2 >= 0

    def barrier_objective(x, mu):
        if x[0] <= 0 or x[1] <= 0 or x[0] + x[1] >= 1:
            return np.inf

        obj = 0.5 * (x[0]**2 + x[1]**2)
        barrier = -mu * (np.log(x[0]) + np.log(x[1]) + np.log(1 - x[0] - x[1]))
        return obj + barrier

    mu_values = [1, 0.1, 0.01, 0.001]
    solutions = []

    for mu in mu_values:
        x0_barrier = np.array([0.2, 0.2])
        result_barrier = minimize(lambda x: barrier_objective(x, mu), x0_barrier,
                                  method='BFGS')
        solutions.append(result_barrier.x)
        print(f"  = {mu:6.3f}: 解 = [{result_barrier.x[0]:.4f},
↪ {result_barrier.x[1]:.4f}]")

    print("當  → 0 時，解收斂到 [0, 0]")
```

```
# 執行範例
print("二次規劃求解範例：")
print("="*50)
results = solve_qp_examples()
active_set_demo()
barrier_method_demo()
```

探索

進階二次規劃主題
探索二次規劃的這些進階方面：

1. 參數二次規劃：研究當問題參數變化時最優解如何變化
2. 穩健二次規劃：調查參數不確定性下的二次規劃形式
3. 整數二次規劃：檢查混合整數二次規劃問題
4. 多參數二次規劃：分析具有多個變化參數的二次規劃問題
5. 隨機二次規劃：考慮具有隨機變數的二次規劃問題

練習 **3.1.** 考慮二次規劃：

$$\begin{aligned} \min \quad & \frac{1}{2}(x_1^2 + 4x_2^2) + x_1 - 2x_2 && (3.60)\\ \text{受約束於} \quad & x_1 + x_2 = 1 && (3.61)\\ & x_1 - x_2 \leq 0 && (3.62)\\ & x_1, x_2 \geq 0 && (3.63) \end{aligned}$$

通過以下方式求解此問題：

1. 寫出完整的 KKT 系統
2. 識別所有可能的活躍集
3. 系統性地求解每種情況
4. 驗證最優解

練習 **3.2.** 推導投資組合最佳化問題的 KKT 條件：

$$\begin{aligned} \min \quad & \frac{1}{2} w^T \Sigma w && (3.64)\\ \text{受約束於} \quad & \mathbf{1}^T w = 1 && (3.65)\\ & \mu^T w \geq r_{\min} && (3.66)\\ & w \geq 0 && (3.67) \end{aligned}$$

並解釋每個拉格朗日乘數的經濟含義。證明當報酬約束不活躍時，最優投資組合具有封閉形式解：

$$w^* = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^T\Sigma^{-1}\mathbf{1}}$$

練習 **3.3.** 對於應用於二次規劃的障礙法，證明中心路徑 $x(\mu)$ 滿足：

$$\lim_{\mu \to 0^+} x(\mu) = x^*$$

其中 x^* 是原始二次規劃的最優解。討論收斂速率。

練習 **3.4.** 實現求解一般二次規劃問題的活躍集方法。你的算法應處理：

1. 找到初始可行點
2. 確定從工作集中添加/移除哪個約束
3. 當多個約束具有相同乘數值時處理退化性
4. 檢測不可行性和無界性

在具有不同風險-報酬偏好的投資組合最佳化問題上測試你的實現。