最佳化方法與應用二
MA5038-*

Chapter 15. Fundamentals of Algorithms for Nonlinear Constrained Optimization

Introduction

在本章中，我們開始討論解決一般受限優化 (constrained optimization) 問題的演算法，該問題的表達如下：

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0 & \text{if } i \in \mathcal{E}, \\ c_i(x) \geqslant 0 & \text{if } i \in \mathcal{I}, \end{cases} \tag{1}$$

其中目標函數 f 和限制函數 c_i 都是定義在 $\mathbb{R}^n$ 的一個子集上的光滑 (smooth)、實數值函數，而 $\mathcal{I}$ 和 $\mathcal{E}$ 則分別是不等式和等式限制式的有限 index set。在第 12 章中，我們使用了這個一般性的問題陳述來推導表徵其解的最優條件。這些理論有助於啟發本書接下來的章節討論的各種藉由迭代產生一系列解 x_* 的估計值的演算法，而我們期望這些估計值趨近於解。在某些情況下，它們還會生成與限制式相關聯的 Lagrange 乘子的猜測數列。與上學期所述關於無受限優化的章節一樣，我們僅研究尋找 (1) 的局部解的算法。

Introduction

在本章中，我們開始討論解決一般受限優化 (constrained optimization) 問題的演算法，該問題的表達如下：

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0 & \text{if } i \in \mathcal{E}, \\ c_i(x) \geqslant 0 & \text{if } i \in \mathcal{I}, \end{cases} \tag{1}$$

其中目標函數 f 和限制函數 c_i 都是定義在 $\mathbb{R}^n$ 的一個子集上的光滑 (smooth)、實數值函數，而 $\mathcal{I}$ 和 $\mathcal{E}$ 則分別是不等式和等式限制式的有限 index set。在第 12 章中，我們使用了這個一般性的問題陳述來推導表徵其解的最優條件。這些理論有助於啟發本書接下來的章節討論的各種藉由迭代產生一系列解 x_* 的估計值的演算法，而我們期望這些估計值趨近於解。在某些情況下，它們還會生成與限制式相關聯的 Lagrange 乘子的猜測數列。與上學期所述關於無受限優化的章節一樣，我們僅研究尋找 (1) 的局部解的算法。

Introduction

在本章中，我們開始討論解決一般受限優化 (constrained optimization) 問題的演算法，該問題的表達如下：

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0 & \text{if } i \in \mathcal{E}, \\ c_i(x) \geqslant 0 & \text{if } i \in \mathcal{I}, \end{cases} \tag{1}$$

其中目標函數 f 和限制函數 c_i 都是定義在 $\mathbb{R}^n$ 的一個子集上的光滑 (smooth)、實數值函數，而 $\mathcal{I}$ 和 $\mathcal{E}$ 則分別是不等式和等式限制式的有限 index set。在第 12 章中，我們使用了這個一般性的問題陳述來推導表徵其解的最優條件。這些理論有助於啟發本書接下來的章節討論的各種藉由迭代產生一系列解 x_* 的估計值的演算法，而我們期望這些估計值趨近於解。在某些情況下，它們還會生成與限制式相關聯的 Lagrange 乘子的猜測數列。與上學期所述關於無受限優化的章節一樣，我們僅研究尋找 (1) 的局部解的算法。

Introduction

我們注意到，本章不涉及單個算法本身，而是涉及與多個演算法共有的基本概念和構建模塊。閱讀了 §15.1 和 §15.2 後，讀者可能希望瀏覽 §15.3、§15.4、§15.5 和 §15.6 的內容，在後續章節的學習過程中根據需要返回這些部分。

§15.1 Categorizing Optimization Algorithms

在本節中我們整理本書接下來幾章中呈現的演算法。對於非線性優化演算法，目前沒有標準的分類體系，而我們分類如下：

1. 在第 16 章中，我們研究解決二次規劃 (quadratic programming) 問題的演算法。我們將這個類別單獨考慮，是因為它具有本質的重要性，其特定特徵可以被高效的演算法所使用，而且在解決非線性規劃問題時需要使用一連串的二次規劃方法和某些內點法 (interior-point method) 來處理相關的二次規劃子問題。我們將討論 active set、內點法和梯度投影法 (gradient projection method)。
2. 在第 17 章中，我們討論懲罰法 (penalty method) 和擴增 Lagrangian 方法 (augmented Lagrangian method)。這兩個方法的共同特點是通過將目標函數和限制函數結合成一個懲罰函數，我們可以通過解決一系列無受限問題來處理問題。

§15.1 Categorizing Optimization Algorithms

在本節中我們整理本書接下來幾章中呈現的演算法。對於非線性優化演算法，目前沒有標準的分類體系，而我們分類如下：

1. 在第 16 章中，我們研究解決二次規劃 (quadratic programming) 問題的演算法。我們將這個類別單獨考慮，是因為它具有本質的重要性，其特定特徵可以被高效的演算法所使用，而且在解決非線性規劃問題時需要使用一連串的二次規劃方法和某些內點法 (interior-point method) 來處理相關的二次規劃子問題。我們將討論 active set、內點法和梯度投影法 (gradient projection method)。
2. 在第 17 章中，我們討論懲罰法 (penalty method) 和擴增 Lagrangian 方法 (augmented Lagrangian method)。這兩個方法的共同特點是通過將目標函數和限制函數結合成一個懲罰函數，我們可以通過解決一系列無受限問題來處理問題。

§15.1 Categorizing Optimization Algorithms

舉例來說，如果在 (1) 中只存在等式限制，我們可以定義二次懲罰函數：

$$f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x), \tag{2}$$

其中 $\mu > 0$ 被稱為懲罰參數 (penalty parameter)。我們對一系列遞增的 μ 值最小化這個無受限函數，直到找到足夠精確的受限優化問題的解。

對於等式限制問題，我們也可能可以通過解以下的無受限優化問題

$$\min_x f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)|,$$

找到 (1) 的局部解。儘管此優化問題中的目標函數通常是不可微的，但還是可以通過解一系列光滑的子問題來最小化精確懲罰函數。

§15.1 Categorizing Optimization Algorithms

舉例來說，如果在 (1) 中只存在等式限制，我們可以定義二次懲罰函數：

$$f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x), \tag{2}$$

其中 $\mu > 0$ 被稱為懲罰參數 (penalty parameter)。我們對一系列遞增的 μ 值最小化這個無受限函數，直到找到足夠精確的受限優化問題的解。

對於等式限制問題，我們也可能可以通過解以下的無受限優化問題

$$\min_x f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)|,$$

找到 (1) 的局部解。儘管此優化問題中的目標函數通常是不可微的，但還是可以通過解一系列光滑的子問題來最小化精確懲罰函數。

§15.1 Categorizing Optimization Algorithms

在擴增 Lagrangian 方法中，我們定義了一個結合 Lagrangian 和二次懲罰函數 (2) 特性的函數。這個所謂的擴增 Lagrangian 函數 (augmented Lagrangian function) 在只具等式限制問題中具有以下形式：

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x) .$$

擴增 Lagrangian 方法首先將 λ 固定為最優 Lagrange 乘子向量的某個估計值，並將 μ 固定為某個正值，然後找到一個使得擴增 Lagrangian 近似最小化的 x 值。在這個新的 x 迭代點上，λ 和 μ 可能會被更新；然後重複這個過程。這種方法避免了與最小化二次懲罰函數 (2) 相關的某些缺點。

§15.1 Categorizing Optimization Algorithms

在擴增 Lagrangian 方法中，我們定義了一個結合 Lagrangian 和二次懲罰函數 (2) 特性的函數。這個所謂的擴增 Lagrangian 函數 (augmented Lagrangian function) 在只具等式限制問題中具有以下形式：

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x).$$

擴增 Lagrangian 方法首先將 λ 固定為最優 Lagrange 乘子向量的某個估計值，並將 μ 固定為某個正值，然後找到一個使得擴增 Lagrangian 近似最小化的 x 值。在這個新的 x 迭代點上，λ 和 μ 可能會被更新；然後重複這個過程。這種方法避免了與最小化二次懲罰函數 (2) 相關的某些缺點。

§15.1 Categorizing Optimization Algorithms

在擴增 Lagrangian 方法中，我們定義了一個結合 Lagrangian 和二次懲罰函數 (2) 特性的函數。這個所謂的擴增 Lagrangian 函數 (augmented Lagrangian function) 在只具等式限制問題中具有以下形式：

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x) .$$

擴增 Lagrangian 方法首先將 λ 固定為最優 Lagrange 乘子向量的某個估計值，並將 μ 固定為某個正值，然後找到一個使得擴增 Lagrangian 近似最小化的 x 值。在這個新的 x 迭代點上，λ 和 μ 可能會被更新；然後重複這個過程。這種方法避免了與最小化二次懲罰函數 (2) 相關的某些缺點。

§15.1 Categorizing Optimization Algorithms

在擴增 Lagrangian 方法中，我們定義了一個結合 Lagrangian 和二次懲罰函數 (2) 特性的函數。這個所謂的擴增 Lagrangian 函數 (augmented Lagrangian function) 在只具等式限制問題中具有以下形式：

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x) .$$

擴增 Lagrangian 方法首先將 λ 固定為最優 Lagrange 乘子向量的某個估計值，並將 μ 固定為某個正值，然後找到一個使得擴增 Lagrangian 近似最小化的 x 值。在這個新的 x 迭代點上，λ 和 μ 可能會被更新；然後重複這個過程。這種方法避免了與最小化二次懲罰函數 (2) 相關的某些缺點。

§15.1 Categorizing Optimization Algorithms

3. 在第 18 章中，我們描述了順序二次規劃 (SQP) 方法，這些方法通過在每次迭代中使用二次規劃子問題來近似問題 (1)，並定義搜索方向為該子問題的解：在迭代點 (x_k, λ_k)，搜索方向 p_k 為二次規劃問題 (3) 的解

$$\min_p \frac{1}{2} p^{\mathrm{T}} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) p + \nabla f(x_k)^{\mathrm{T}} p \tag{3a}$$

subject to

$$\nabla c_i(x_k)^{\mathrm{T}} p + c_i(x_k) = 0\,, i \in \mathcal{E}\,, \tag{3b}$$

$$\nabla c_i(x_k)^{\mathrm{T}} p + c_i(x_k) \geqslant 0\,, i \in \mathcal{I}\,, \tag{3c}$$

其中 $\mathcal{L}$ 是 Lagrangian。在這個子問題中，目標函數是近似於從 x_k 移動到 $x_k + p$ 時 Lagrangian 的變化，而限制函數來自於 (1) 中限制式的線性化。

§15.1 Categorizing Optimization Algorithms

3. 在第 18 章中，我們描述了順序二次規劃 (SQP) 方法，這些方法通過在每次迭代中使用二次規劃子問題來近似問題 (1)，並定義搜索方向為該子問題的解：在迭代點 (x_k, λ_k)，搜索方向 p_k 為二次規劃問題 (3) 的解

$$\min_p \frac{1}{2} p^{\mathrm{T}} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) p + \nabla f(x_k)^{\mathrm{T}} p \tag{3a}$$

subject to

$$\nabla c_i(x_k)^{\mathrm{T}} p + c_i(x_k) = 0\,, i \in \mathcal{E}\,, \tag{3b}$$

$$\nabla c_i(x_k)^{\mathrm{T}} p + c_i(x_k) \geqslant 0\,, i \in \mathcal{I}\,, \tag{3c}$$

其中 $\mathcal{L}$ 是 Lagrangian。在這個子問題中，目標函數是近似於從 x_k 移動到 $x_k + p$ 時 Lagrangian 的變化，而限制函數來自於 (1) 中限制式的線性化。

§15.1 Categorizing Optimization Algorithms

為了對步進量 p_k 的長度和質量有所控制，可以在 (3) 中添加一個信賴域 (trust-region) 限制，並且可以使用 quasi-Newton 近似 Hessian 來替代 $\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$。另外有一個被稱為順序線性二次規劃 (sequential linear-quadratic programming) 的 SQP 方法的變體中，步進量 p_k 是通過兩個階段計算的。首先，我們解一個線性規劃問題，該問題的定義是通過將目標 (3a) 中的第一項（二次項）刪除，並向 (3) 添加一個信賴域限制而得到的。接下來，我們解一個只具等式限制的二次規劃子問題來取新的步進量 p_k，這個子問題中的等式限制來自於對線性規劃所得之解所對應的 active constraints，而所有其它的不等式限制則被忽略。

§15.1 Categorizing Optimization Algorithms

為了對步進量 p_k 的長度和質量有所控制，可以在 (3) 中添加一個信賴域 (trust-region) 限制，並且可以使用 quasi-Newton 近似 Hessian 來替代 $\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$。另外有一個被稱為順序線性二次規劃 (sequential linear-quadratic programming) 的 SQP 方法的變體中，步進量 p_k 是通過兩個階段計算的。首先，我們解一個線性規劃問題，該問題的定義是通過將目標 (3a) 中的第一項（二次項）刪除，並向 (3) 添加一個信賴域限制而得到的。接下來，我們解一個只具等式限制的二次規劃子問題來取新的步進量 p_k，這個子問題中的等式限制來自於對線性規劃所得之解所對應的 active constraints，而所有其它的不等式限制則被忽略。

§15.1 Categorizing Optimization Algorithms

4. 在第 19 章，我們研究了用於非線性規劃的內點法。這些方法可以被視為是在第 14 章中討論的線性規劃 primal-dual 內點法的延伸。我們也可以將它們視為解決以下優化問題

$$\min_{x,s} f(x) - \mu \sum_{i=1}^{m} \log s_i$$

$$\text{subject to}$$

$$c_i(x) = 0, i \in \mathcal{E},$$

$$c_i(x) = s_i, i \in \mathcal{I},$$

時用以生成步進量 p_k 的障礙方法 (barrier methods)，其中 $\mu > 0$ 是障礙參數 (barrier parameter)，而 $s_i > 0$ 則為鬆弛變數 (slacks)。內點法是非線性規劃中最新的一類方法，並被證明是順序二次規劃方法相當強大的競爭對手。

§15.1 Categorizing Optimization Algorithms

4. 在第 19 章，我們研究了用於非線性規劃的內點法。這些方法可以被視為是在第 14 章中討論的線性規劃 primal-dual 內點法的延伸。我們也可以將它們視為解決以下優化問題

$$\min_{x,s} f(x) - \mu \sum_{i=1}^{m} \log s_i$$

subject to

$$c_i(x) = 0, i \in \mathcal{E},$$
$$c_i(x) = s_i, i \in \mathcal{I},$$

時用以生成步進量 p_k 的障礙方法 (barrier methods)，其中 $\mu > 0$ 是障礙參數 (barrier parameter)，而 $s_i > 0$ 則為鬆弛變數 (slacks)。內點法是非線性規劃中最新的一類方法，並被證明是順序二次規劃方法相當強大的競爭對手。

§15.1 Categorizing Optimization Algorithms

消除技術 (elimination techniques) 在①、③ 和④ 類別的演算法中被使用，這些消除技術使用了限制條件來消除問題中的某些自由度。為了瞭解這些演算法的背景，我們在 §15.3 中討論了消除的概念。在隨後的章節中，我們將討論 merit functions 和 filters，這是促使非線性規劃演算法從遠處起始點收斂的重要機制。

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

在解決非線性規劃問題中的一個主要挑戰在於處理不等式限制，特別是決定對於（未知）真解哪些限制條件是 active 哪些不是。一個具備 active set method 精神的方法，是首先猜測對真解來說為等式的不等式限制之集合 (optimal active set) $\mathcal{A}_*$。我們將這個猜測稱為工作集 (working set) 並以 $\mathcal{W}$ 來表示。然後，我們解 "在 $\mathcal{W}$ 中的限制被強制規定為等式而不在 $\mathcal{W}$ 中的限制則被忽略的" 優化子問題。再然後我們檢查是否存在 Lagrange 乘子使得對於這個 $\mathcal{W}$ 獲得的解 x_* 滿足 KKT 條件。如果存在這樣的 Lagrange 乘子，我們將 x_* 接受為問題 (1) 的局部解。否則，我們選擇不同的工作集 $\mathcal{W}$ 並重複這個過程。這種方法是基於一般而言解決等式限制問題要比解非線性規劃問題簡單得多的這個觀察所設計出來的最佳化方法。

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

在解決非線性規劃問題中的一個主要挑戰在於處理不等式限制，特別是決定對於（未知）真解哪些限制條件是 active 哪些不是。一個具備 active set method 精神的方法，是首先猜測對真解來說為等式的不等式限制之集合 (optimal active set) $\mathcal{A}_*$。我們將這個猜測稱為工作集 (working set) 並以 $\mathcal{W}$ 來表示。然後，我們解 "在 $\mathcal{W}$ 中的限制被強制規定為等式而不在 $\mathcal{W}$ 中的限制則被忽略的" 優化子問題。再然後我們檢查是否存在 Lagrange 乘子使得對於這個 $\mathcal{W}$ 獲得的解 x_* 滿足 KKT 條件。如果存在這樣的 Lagrange 乘子，我們將 x_* 接受為問題 (1) 的局部解。否則，我們選擇不同的工作集 $\mathcal{W}$ 並重複這個過程。這種方法是基於一般而言解決等式限制問題要比解非線性規劃問題簡單得多的這個觀察所設計出來的最佳化方法。

Theorem 12.1 – KKT conditions

Theorem (First-Order Necessary Conditions)

Suppose that x_* is a local solution of problem

$$\min_{x\in\mathbb{R}^n} f(x) \quad \textit{subject to} \quad \begin{cases} c_i(x) = 0, i \in \mathcal{E}\,, \\ c_i(x) \geqslant 0, i \in \mathcal{I}\,, \end{cases} \tag{1}$$

that the functions f and c_i in (1) are continuously differentiable, and that the LICQ holds at x_*. Then there is a Lagrange multiplier vector λ_*, with components λ_i^*, $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied.

$$\nabla_x \mathcal{L}(x_*, \lambda_*) = 0\,, \tag{32a}_{12}$$

$$c_i(x_*) = 0 \quad \textit{for all } i \in \mathcal{E}, \tag{32b}_{12}$$

$$c_i(x_*) \geqslant 0 \quad \textit{for all } i \in \mathcal{I}, \tag{32c}_{12}$$

$$\lambda_i^* \geqslant 0 \quad \textit{for all } i \in \mathcal{I}, \tag{32d}_{12}$$

$$\lambda_i^* c_i(x_*) = 0 \quad \textit{for all } i \in \mathcal{E} \cup \mathcal{I}. \tag{32e}_{12}$$

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

工作集 $\mathcal{W}$ 的選擇數目是 $2^{|\mathcal{I}|}$，其中 $|\mathcal{I}|$ 是不等式限制的數量（我們可以通過觀察每個 $\mathcal{I}$ 的 index i 是否在 $\mathcal{W}$ 中來得到這個估計）。由於可能的工作集數量隨著不等式數量的增加呈指數成長，這種現象被稱為非線性規劃的組合難度 (combinatorial difficulty)，我們無法期望通過考慮所有可能的 $\mathcal{W}$ 來設計一個實用的算法。

以下的例子表明，即使對於少量的不等式限制，確定 optimal active set 也並不是一個簡單的任務。

Example

Consider the problem

$$\min_{x,y} f(x,y) = \frac{1}{2}(x-2)^2 + \frac{1}{2}\left(y-\frac{1}{2}\right)^2 \tag{4}$$

subject to

$$(x+1)^{-1} - y - \frac{1}{4} \geqslant 0, \quad x \geqslant 0, \quad y \geqslant 0.$$

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

工作集 $\mathcal{W}$ 的選擇數目是 $2^{|\mathcal{I}|}$，其中 $|\mathcal{I}|$ 是不等式限制的數量（我們可以通過觀察每個 $\mathcal{I}$ 的 index i 是否在 $\mathcal{W}$ 中來得到這個估計）。由於可能的工作集數量隨著不等式數量的增加呈指數成長，這種現象被稱為非線性規劃的組合難度 (combinatorial difficulty)，我們無法期望通過考慮所有可能的 $\mathcal{W}$ 來設計一個實用的算法。

以下的例子表明，即使對於少量的不等式限制，確定 optimal active set 也並不是一個簡單的任務。

Example

Consider the problem

$$\min_{x,y} f(x,y) = \frac{1}{2}(x-2)^2 + \frac{1}{2}\left(y-\frac{1}{2}\right)^2 \tag{4}$$

subject to

$$(x+1)^{-1} - y - \frac{1}{4} \geqslant 0, \quad x \geqslant 0, \quad y \geqslant 0.$$

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

We label the constraints, in order, with the indices 1 through 3. Figure 1 illustrates the contours of the objective function (dashed circles). The feasible region is the region enclosed by the curve and the two axes. We see that only the first constraint is active at the solution, which is $(x_*, y_*)^{\mathrm{T}} = (1.953, 0.089)^{\mathrm{T}}$.

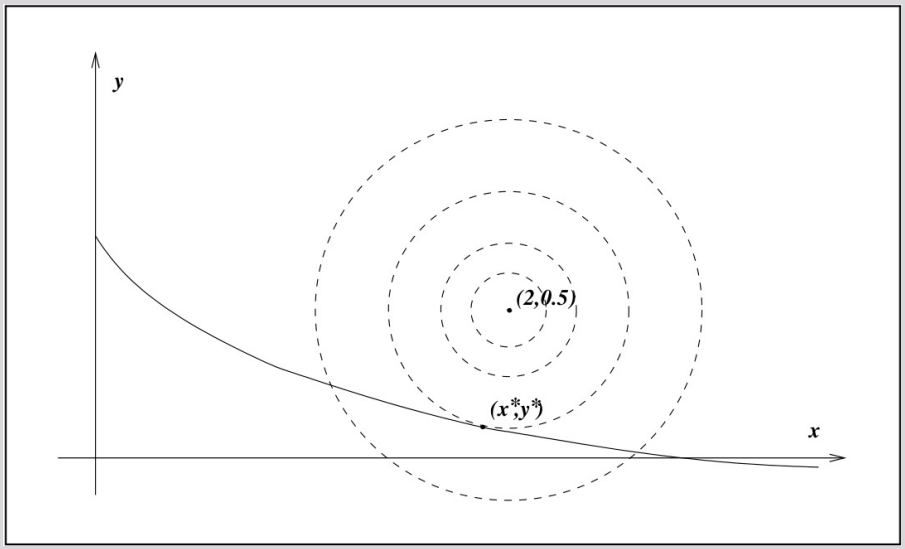

Figure 1: Graphical illustration of problem (4)

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

Let us now apply the working-set approach described above to problem (4), considering all $2^3 = 8$ possible choices of $\mathcal{W}$. Since $\nabla f = (x-2, y-1/2)^{\mathrm{T}}$, we see that the unconstrained minimum of f lies outside the feasible region. Hence, $\mathcal{W}$ cannot be empty.

There are seven other possibilities. First, all three constraints could be active (that is, $\mathcal{W} = \{1,2,3\}$); however, a glance at Figure 1 shows that this does not happen for our problem since the three constraints do not share a common point of intersection. Three further possibilities are obtained by making a single constraint active (that is, $\mathcal{W} = \{1\}$, $\mathcal{W} = \{2\}$, and $\mathcal{W} = \{3\}$), while the final three possibilities are obtained by making exactly two constraints active (that is, $\mathcal{W} = \{1,2\}$, $\mathcal{W} = \{1,3\}$, and $\mathcal{W} = \{2,3\}$). We consider three of these cases in detail.

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

Let us now apply the working-set approach described above to problem (4), considering all $2^3 = 8$ possible choices of $\mathcal{W}$. Since $\nabla f = (x-2, y-1/2)^{\mathrm{T}}$, we see that the unconstrained minimum of f lies outside the feasible region. Hence, $\mathcal{W}$ cannot be empty.

There are seven other possibilities. First, all three constraints could be active (that is, $\mathcal{W} = \{1,2,3\}$); however, a glance at Figure 1 shows that this does not happen for our problem since the three constraints do not share a common point of intersection. Three further possibilities are obtained by making a single constraint active (that is, $\mathcal{W} = \{1\}$, $\mathcal{W} = \{2\}$, and $\mathcal{W} = \{3\}$), while the final three possibilities are obtained by making exactly two constraints active (that is, $\mathcal{W} = \{1,2\}$, $\mathcal{W} = \{1,3\}$, and $\mathcal{W} = \{2,3\}$). We consider three of these cases in detail.

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

Let us now apply the working-set approach described above to problem (4), considering all $2^3 = 8$ possible choices of $\mathcal{W}$. Since $\nabla f = (x-2, y-1/2)^{\mathrm{T}}$, we see that the unconstrained minimum of f lies outside the feasible region. Hence, $\mathcal{W}$ cannot be empty.

There are seven other possibilities. First, all three constraints could be active (that is, $\mathcal{W} = \{1,2,3\}$); however, a glance at Figure 1 shows that this does not happen for our problem since the three constraints do not share a common point of intersection. Three further possibilities are obtained by making a single constraint active (that is, $\mathcal{W} = \{1\}$, $\mathcal{W} = \{2\}$, and $\mathcal{W} = \{3\}$), while the final three possibilities are obtained by making exactly two constraints active (that is, $\mathcal{W} = \{1,2\}$, $\mathcal{W} = \{1,3\}$, and $\mathcal{W} = \{2,3\}$). We consider three of these cases in detail.

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

1. $\mathcal{W} = \{2\}$; that is, only the constraint $x = 0$ is active. If we minimize f enforcing only this constraint, we obtain the point $(0, 1/2)^{\mathrm{T}}$. A check of the KKT conditions $(32)_{12}$ shows that no matter how we choose the Lagrange multipliers, we cannot satisfy all these conditions at $(0, 1/2)^{\mathrm{T}}$. We must have $\lambda_1 = \lambda_3 = 0$ to satisfy the KKT condition

$$\lambda_i^* c_i(x_*, y_*) = 0 \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}. \tag{32e}_{12}$$

which implies that we must set $\lambda_2 = -2$ to satisfy

$$\nabla_{(x,y)} \mathcal{L}(x_*, y_*, \lambda_*) = 0\,; \tag{32a}_{12}$$

but this value of λ_2 violates the KKT condition

$$\lambda_i^* \geqslant 0 \quad \text{for all } i \in \mathcal{I}, \tag{32d}_{12}$$

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

2. $\mathcal{W} = \{1, 3\}$, which yields the single feasible point $(3, 0)^{\mathrm{T}}$. Since constraint 2 is inactive at this point, we have $\lambda_2 = 0$, so by solving

$$\nabla_{(x,y)} \mathcal{L}(x_*, y_*, \lambda_*) = 0\,, \qquad (32a)_{12}$$

for the other Lagrange multipliers, we obtain $\lambda_1 = -16$ and $\lambda_3 = -16.5$. These values are negative, so they violate

$$\lambda_i^* \geqslant 0 \quad \text{for all } i \in \mathcal{I}, \qquad (32d)_{12}$$

and $(x, y) = (3, 0)^{\mathrm{T}}$ cannot be a solution of (1).

3. $\mathcal{W} = \{1\}$. Solving the equality-constrained problem in which the first constraint is active, we obtain $(x, y)^{\mathrm{T}} = (1.953, 0.089)^{\mathrm{T}}$ with Lagrange multiplier $\lambda_1 = 0.411$. By setting $\lambda_2 = \lambda_3 = 0$, the remaining KKT conditions are satisfied.

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

2. $\mathcal{W} = \{1, 3\}$, which yields the single feasible point $(3, 0)^{\mathrm{T}}$. Since constraint 2 is inactive at this point, we have $\lambda_2 = 0$, so by solving

$$\nabla_{(x,y)} \mathcal{L}(x_*, y_*, \lambda_*) = 0\,, \qquad (32a)_{12}$$

for the other Lagrange multipliers, we obtain $\lambda_1 = -16$ and $\lambda_3 = -16.5$. These values are negative, so they violate

$$\lambda_i^* \geqslant 0 \quad \text{for all } i \in \mathcal{I}, \qquad (32d)_{12}$$

and $(x, y) = (3, 0)^{\mathrm{T}}$ cannot be a solution of (1).

3. $\mathcal{W} = \{1\}$. Solving the equality-constrained problem in which the first constraint is active, we obtain $(x, y)^{\mathrm{T}} = (1.953, 0.089)^{\mathrm{T}}$ with Lagrange multiplier $\lambda_1 = 0.411$. By setting $\lambda_2 = \lambda_3 = 0$, the remaining KKT conditions are satisfied.

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

Therefore, $(1.953, 0.089)^{\mathrm{T}}$ is a KKT point. Furthermore, the second-order sufficient conditions are satisfied, as the Hessian of the Lagrangian is positive definite.

在這個小例子中，考慮所有可能的 $\mathcal{W}$ 也是非常繁瑣的過程。然而若能利用問題中的函數及其導數的知識，便可以消除一些 $\mathcal{W}$ 的選擇。第 16 章中描述的 active set 方法便使用此類訊息對工作集進行一系列有根據的猜測，避免選擇明顯不會得到解的 $\mathcal{W}$。

在第 19 章討論的內點（或障礙）法採用了一種不同的方法。這些方法生成的迭代點離由不等式限制定義的可行區域的邊界尚有一小段距離（即接近邊界的障礙效應）。隨著接近非線性規劃的解，障礙效應會被削弱，以允許對解的估計更加準確。通過這種方式，內點方法避免了非線性規劃的組合困難。

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

Therefore, $(1.953, 0.089)^{\mathrm{T}}$ is a KKT point. Furthermore, the second-order sufficient conditions are satisfied, as the Hessian of the Lagrangian is positive definite.

在這個小例子中，考慮所有可能的 $\mathcal{W}$ 也是非常繁瑣的過程。然而若能利用問題中的函數及其導數的知識，便可以消除一些 $\mathcal{W}$ 的選擇。第 16 章中描述的 active set 方法便使用此類訊息對工作集進行一系列有根據的猜測，避免選擇明顯不會得到解的 $\mathcal{W}$。

在第 19 章討論的內點（或障礙）法採用了一種不同的方法。這些方法生成的迭代點離由不等式限制定義的可行區域的邊界尚有一小段距離（即接近邊界的障礙效應）。隨著接近非線性規劃的解，障礙效應會被削弱，以允許對解的估計更加準確。通過這種方式，內點方法避免了非線性規劃的組合困難。

§15.2 The Combinatorial Difficulty of Inequality-Constrained Problems

Example (cont'd)

Therefore, $(1.953, 0.089)^{\mathrm{T}}$ is a KKT point. Furthermore, the second-order sufficient conditions are satisfied, as the Hessian of the Lagrangian is positive definite.

在這個小例子中，考慮所有可能的 $\mathcal{W}$ 也是非常繁瑣的過程。然而若能利用問題中的函數及其導數的知識，便可以消除一些 $\mathcal{W}$ 的選擇。第 16 章中描述的 active set 方法便使用此類訊息對工作集進行一系列有根據的猜測，避免選擇明顯不會得到解的 $\mathcal{W}$。

在第 19 章討論的內點（或障礙）法採用了一種不同的方法。這些方法生成的迭代點離由不等式限制定義的可行區域的邊界尚有一小段距離（即接近邊界的障礙效應）。隨著接近非線性規劃的解，障礙效應會被削弱，以允許對解的估計更加準確。通過這種方式，內點方法避免了非線性規劃的組合困難。

§15.3 Elimination of Variables

當處理受限優化問題時，嘗試使用限制條件來消除問題中的一些變數，以獲得一個更簡單、自由度較少的問題是非常自然的。然而，必須謹慎使用消除技術，因為它們可能改變問題或引入 ill conditioning。

我們首先舉一個消除變數是安全和方便的例子。在問題

$$\min f(x_1, x_2, x_3, x_4) \quad \text{subject to} \quad \begin{cases} x_1 + x_3^2 - x_4 x_3 = 0, \\ -x_2 + x_4 + x_3^2 = 0, \end{cases}$$

中設定

$$x_1 = x_4 x_3 - x_3^2, \quad x_2 = x_4 + x_3^2$$

是安全的，這樣做我們可以得到一個只有兩個變數的函數

$$h(x_3, x_4) = f(x_4 x_3 - x_3^2, x_4 + x_3^2, x_3, x_4),$$

然後我們可以使用在先前章節中描述的無受限優化技術對其進行最小化。

§15.3 Elimination of Variables

當處理受限優化問題時，嘗試使用限制條件來消除問題中的一些變數，以獲得一個更簡單、自由度較少的問題是非常自然的。然而，必須謹慎使用消除技術，因為它們可能改變問題或引入 ill conditioning。

我們首先舉一個消除變數是安全和方便的例子。在問題

$$\min f(x_1, x_2, x_3, x_4) \quad \text{subject to} \quad \begin{cases} x_1 + x_3^2 - x_4 x_3 = 0, \\ -x_2 + x_4 + x_3^2 = 0, \end{cases}$$

中設定

$$x_1 = x_4 x_3 - x_3^2, \quad x_2 = x_4 + x_3^2$$

是安全的，這樣做我們可以得到一個只有兩個變數的函數

$$h(x_3, x_4) = f(x_4 x_3 - x_3^2, x_4 + x_3^2, x_3, x_4),$$

然後我們可以使用在先前章節中描述的無受限優化技術對其進行最小化。

§15.3 Elimination of Variables

Example (非線性消除的危險性)

Consider the problem

$$\min x^2 + y^2 \quad \text{subject to} \quad (x-1)^3 = y^2.$$

The contours of the objective function and the constraints are illustrated in Figure 2, which shows that the solution is $(x, y) = (1, 0)$.

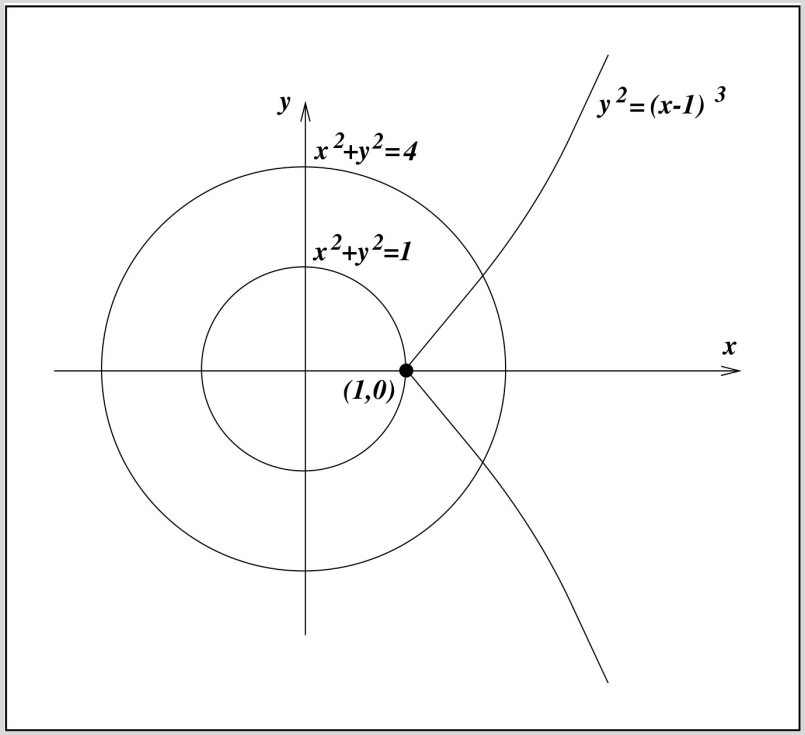

Figure 2: The danger of nonlinear elimination.

§15.3 Elimination of Variables

Example (非線性消除的危險性 (cont'd))

We try to solve this problem by eliminating y. Doing so, we obtain

$$h(x) = x^2 + (x-1)^3.$$

Clearly, $h(x) \to -\infty$ as $x \to -\infty$.

By blindly applying this transformation we may conclude that the problem is unbounded, but this view ignores the fact that the constraint $(x-1)^3 = y^2$ implicitly imposes the bound $x \geqslant 1$ that is active at the solution. Hence, if we wish to eliminate y, we should explicitly introduce the bound $x \geqslant 1$ into the problem.

這個例子表明，使用非線性方程式來消除變數可能導致難以追踪的錯誤。因此，大多數優化算法不使用非線性消除。相反地，許多算法將限制式線性化，然後對簡化後的問題應用消除技術。接下來，我們將描述使用線性限制式執行變數消除的系統程序。

§15.3 Elimination of Variables

Example (非線性消除的危險性 (cont'd))

We try to solve this problem by eliminating y. Doing so, we obtain

$$h(x) = x^2 + (x-1)^3.$$

Clearly, $h(x) \to -\infty$ as $x \to -\infty$.

By blindly applying this transformation we may conclude that the problem is unbounded, but this view ignores the fact that the constraint $(x-1)^3 = y^2$ implicitly imposes the bound $x \geqslant 1$ that is active at the solution. Hence, if we wish to eliminate y, we should explicitly introduce the bound $x \geqslant 1$ into the problem.

這個例子表明，使用非線性方程式來消除變數可能導致難以追踪的錯誤。因此，大多數優化算法不使用非線性消除。相反地，許多算法將限制式線性化，然後對簡化後的問題應用消除技術。接下來，我們將描述使用線性限制式執行變數消除的系統程序。

§15.3 Elimination of Variables

- **Simple elimination using linear constraints**

我們考慮在一組線性等式限制下的最佳化的問題

$$\min f(x) \quad \text{subject to} \quad Ax = b, \tag{5}$$

其中 A 是一個 $m \times n$ 矩陣且 $m \leqslant n$。簡單起見，我們假設 A 有滿秩 (full rank)：若 A 非滿秩，則不是限制條件不一致（因而無解）就是某些限制條件是冗餘的可以刪除而不影響問題的解。

在 A 有滿秩的假設下，可以找到 A 中線性獨立的 m 行。將這些行組成一個 $m \times m$ 的矩陣 B（稱為基底矩陣 basis matrix），並定義一個將這些線性獨立行交換到 A 的前 m 行位置的 $n \times n$ 排列 (permutation) 矩陣 Π，則有

$$A\Pi = \left[B \vdots N\right], \tag{6}$$

其中 N 表示 A 在 B 中 m 行外剩餘的 $n - m$ 行。

§15.3 Elimination of Variables

- **Simple elimination using linear constraints**

我們考慮在一組線性等式限制下的最佳化的問題

$$\min f(x) \quad \text{subject to} \quad Ax = b, \tag{5}$$

其中 A 是一個 $m \times n$ 矩陣且 $m \leqslant n$。簡單起見，我們假設 A 有滿秩 (full rank)：若 A 非滿秩，則不是限制條件不一致（因而無解）就是某些限制條件是冗餘的可以刪除而不影響問題的解。

在 A 有滿秩的假設下，可以找到 A 中線性獨立的 m 行。將這些行組成一個 $m \times m$ 的矩陣 B（稱為基底矩陣 basis matrix），並定義一個將這些線性獨立行交換到 A 的前 m 行位置的 $n \times n$ 排列 (permutation) 矩陣 Π，則有

$$A\Pi = \left[B \vdots N\right], \tag{6}$$

其中 N 表示 A 在 B 中 m 行外剩餘的 $n - m$ 行。

§15.3 Elimination of Variables

我們定義子向量 $x_B \in \mathbb{R}^m$ 與 $x_N \in \mathbb{R}^{n-m}$ 如下：

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = \Pi^{\mathrm{T}} x, \tag{7}$$

注意到 Π 是正交矩陣，所以可以將限制條件 $Ax = b$ 重新改寫為

$$b = Ax = A\Pi(\Pi^{\mathrm{T}} x) = Bx_B + Nx_N .$$

通過重新排列這個公式，我們得出 basic variable x_B 可以表示為

$$x_B = B^{-1}b - B^{-1}Nx_N . \tag{8}$$

因此，x_N 可視為自由變數，而滿足限制條件 $Ax = b$ 的可行點 (feasible point) 可根據公式 (7) 與 (8) 來求得。因此，問題 (5) 等價於無限制問題

$$\min_{x_N} h(x_N) \equiv f\left(\Pi \begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix}\right).$$

我們將 (8) 中的代換稱為變數的簡單消除 (simple elimination)。

§15.3 Elimination of Variables

我們定義子向量 $x_B \in \mathbb{R}^m$ 與 $x_N \in \mathbb{R}^{n-m}$ 如下：

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = \Pi^{\mathrm{T}} x, \tag{7}$$

注意到 Π 是正交矩陣，所以可以將限制條件 $Ax = b$ 重新改寫為

$$b = Ax = A\Pi(\Pi^{\mathrm{T}} x) = Bx_B + Nx_N .$$

通過重新排列這個公式，我們得出 basic variable x_B 可以表示為

$$x_B = B^{-1}b - B^{-1}Nx_N . \tag{8}$$

因此，x_N 可視為自由變數，而滿足限制條件 $Ax = b$ 的可行點 (feasible point) 可根據公式 (7) 與 (8) 來求得。因此，問題 (5) 等價於無限制問題

$$\min_{x_N} h(x_N) \equiv f\left(\Pi \begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix}\right).$$

我們將 (8) 中的代換稱為變數的簡單消除 (simple elimination)。

§15.3 Elimination of Variables

Example

Consider the problem

$$\min_{x_1,x_2,x_3,x_4,x_5,x_6} \sin(x_1+x_2)+x_3^2+\frac{1}{3}\big(x_4+x_5^4+\frac{x_6}{2}\big) \tag{9a}$$

subject to

$$\begin{aligned} 8x_1-6x_2+x_3+9x_4+4x_5&=6\,,\\ 3x_1+2x_2-x_4+6x_5+4x_6&=-4\,. \end{aligned} \tag{9b}$$

By defining the permutation matrix Π so as to reorder the components of x as $x^{\mathrm{T}}=(x_3,x_6,x_1,x_2,x_4,x_5)^{\mathrm{T}}$, we find that the coefficient matrix $A\Pi$ is

$$A\Pi=\left[\begin{array}{cc:cccc} 1&0&8&-6&9&4\\ 0&4&3&2&-1&6 \end{array}\right].$$

The basis matrix B is diagonal and therefore easy to invert.

§15.3 Elimination of Variables

Example (cont'd)

We obtain from (8) that

$$\begin{bmatrix} x_3 \\ x_6 \end{bmatrix} = -\begin{bmatrix} 8 & -6 & 9 & 4 \\ \frac{3}{4} & \frac{1}{2} & \frac{-1}{4} & \frac{3}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 6 \\ -1 \end{bmatrix}.$$

By substituting for x_3 and x_6 in (9a), the problem becomes

$$\begin{aligned} \min_{x_1,x_2,x_4,x_5} \quad & \sin(x_1+x_2) + (8x_1 - 6x_2 + 9x_4 + 4x_5 - 6)^2 \\ & + \frac{1}{3}\left[x_4 + x_5^4 - \left(\frac{1}{2} + \frac{3}{8}x_1 + \frac{1}{4}x_2 - \frac{1}{8}x_4 + \frac{3}{4}x_5\right)\right]. \end{aligned}$$

We could have chosen two other columns of the coefficient matrix A (that is, two variables other than x_3 and x_6) as the basis for elimination in the system (9b), but the matrix $B^{-1}N$ would not have been so simple.

§15.3 Elimination of Variables

Example (cont'd)

We obtain from (8) that

$$\begin{bmatrix} x_3 \\ x_6 \end{bmatrix} = -\begin{bmatrix} 8 & -6 & 9 & 4 \\ \frac{3}{4} & \frac{1}{2} & \frac{-1}{4} & \frac{3}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 6 \\ -1 \end{bmatrix}.$$

By substituting for x_3 and x_6 in (9a), the problem becomes

$$\begin{aligned} \min_{x_1,x_2,x_4,x_5} \quad & \sin(x_1+x_2) + (8x_1 - 6x_2 + 9x_4 + 4x_5 - 6)^2 \\ & + \frac{1}{3}\left[x_4 + x_5^4 - \left(\frac{1}{2} + \frac{3}{8}x_1 + \frac{1}{4}x_2 - \frac{1}{8}x_4 + \frac{3}{4}x_5\right)\right]. \end{aligned}$$

We could have chosen two other columns of the coefficient matrix A (that is, two variables other than x_3 and x_6) as the basis for elimination in the system (9b), but the matrix $B^{-1}N$ would not have been so simple.

§15.3 Elimination of Variables

一般來說，可以通過高斯消去法選擇一組線性獨立的 m 行。以線性代數的術語來說，即是計算矩陣的 row echelon form，並將 pivot columns 選擇為基底 B 的行。理想情況下，我們希望 B 容易進行分解並為 well-conditioned。有一個能達成這些目標的技術是稀疏高斯消去法 (sparse Gaussian elimination)，該方法嘗試在保持稀疏性的同時控制捨入誤差 (rounding/round-off errors)。這個算法的一個廣為人知的實現方式是來自 HSL library 的 MA48 [96]。然而，正如我們下面即將討論的，並不能保證高斯消去過程會確定最佳的基底矩陣選擇。

§15.3 Elimination of Variables

一般來說，可以通過高斯消去法選擇一組線性獨立的 m 行。以線性代數的術語來說，即是計算矩陣的 row echelon form，並將 pivot columns 選擇為基底 B 的行。理想情況下，我們希望 B 容易進行分解並為 well-conditioned。有一個能達成這些目標的技術是稀疏高斯消去法 (sparse Gaussian elimination)，該方法嘗試在保持稀疏性的同時控制捨入誤差 (rounding/round-off errors)。這個算法的一個廣為人知的實現方式是來自 HSL library 的 MA48 [96]。然而，正如我們下面即將討論的，並不能保證高斯消去過程會確定最佳的基底矩陣選擇。

§15.3 Elimination of Variables

剛才所描述的簡單消除變數法有一個有趣的解釋。為了簡化符號，從現在起我們假設 $\Pi = \mathrm{I}$，亦即 A 的前 m 行線性獨立。

從 (7) 和 (8) 我們可以看出，滿足線性限制式 $Ax = b$ 的任何可行點 x 都可以寫成

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = Yb + Zx_N, \tag{10}$$

其中

$$Y = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix}_{n\times m}, \quad Z = \begin{bmatrix} -B^{-1}N \\ \mathrm{I} \end{bmatrix}_{n\times(n-m)}. \tag{11}$$

注意到 Z 有 $n-m$ 個線性獨立的行（由於 Z 的下方區塊為單位矩陣），並且滿足 $AZ = 0$。因此，Z 中的行構成 A 的 null space 的一組基底。此外，矩陣 Y 的行和矩陣 Z 的行構成一組線性獨立的集合。同時，我們還可以從 (11) 和 (6) 中注意到，Yb 是線性限制式 $Ax = b$ 的一個特解。

§15.3 Elimination of Variables

剛才所描述的簡單消除變數法有一個有趣的解釋。為了簡化符號，從現在起我們假設 $\Pi = \mathrm{I}$，亦即 A 的前 m 行線性獨立。

從 (7) 和 (8) 我們可以看出，滿足線性限制式 $Ax = b$ 的任何可行點 x 都可以寫成

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = Yb + Zx_N, \tag{10}$$

其中

$$Y = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix}_{n\times m}, \quad Z = \begin{bmatrix} -B^{-1}N \\ \mathrm{I} \end{bmatrix}_{n\times(n-m)}. \tag{11}$$

注意到 Z 有 $n-m$ 個線性獨立的行（由於 Z 的下方區塊為單位矩陣），並且滿足 $AZ = 0$。因此，Z 中的行構成 A 的 null space 的一組基底。此外，矩陣 Y 的行和矩陣 Z 的行構成一組線性獨立的集合。同時，我們還可以從 (11) 和 (6) 中注意到，Yb 是線性限制式 $Ax = b$ 的一個特解。

§15.3 Elimination of Variables

剛才所描述的簡單消除變數法有一個有趣的解釋。為了簡化符號，從現在起我們假設 $\Pi = \mathrm{I}$，亦即 A 的前 m 行線性獨立。

從 (7) 和 (8) 我們可以看出，滿足線性限制式 $Ax = b$ 的任何可行點 x 都可以寫成

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = Yb + Zx_N, \tag{10}$$

其中

$$Y = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix}_{n\times m}, \quad Z = \begin{bmatrix} -B^{-1}N \\ \mathrm{I} \end{bmatrix}_{n\times(n-m)}. \tag{11}$$

注意到 Z 有 $n-m$ 個線性獨立的行（由於 Z 的下方區塊為單位矩陣），並且滿足 $AZ = 0$。因此，Z 中的行構成 A 的 null space 的一組基底。此外，矩陣 Y 的行和矩陣 Z 的行構成一組線性獨立的集合。同時，我們還可以從 (11) 和 (6) 中注意到，Yb 是線性限制式 $Ax = b$ 的一個特解。

§15.3 Elimination of Variables

換句話說，簡單消去變數法將可行點表示為 $Ax = b$ 的特解 ((10) 式中的 Yb) 與沿著限制的 null space 的偏移 ((10) 式中的 Zx_N) 的總和：

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = Yb + Zx_N\,, \tag{10}$$

其中特解 Yb 是通過將 x 的 $n - m$ 個分量保持為零同時放鬆其他 m 個分量（x_B 中的分量）直到它們滿足限制式而獲得。特解 Yb 有時被稱為坐標放鬆步驟 (coordinate relaxation step)。在（下頁）圖 3 中，我們可以看到通過選擇基底矩陣 B 為 A 的第一行所獲得的坐標放鬆步驟 Yb。如果我們將 B 選擇為 A 的第二行，坐標放鬆步驟將沿著 x_2 軸。

§15.3 Elimination of Variables

換句話說，簡單消去變數法將可行點表示為 $Ax = b$ 的特解 ((10) 式中的 Yb) 與沿著限制的 null space 的偏移 ((10) 式中的 Zx_N) 的總和：

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix}_{n\times m} b + \begin{bmatrix} -B^{-1}N \\ \mathrm{I} \end{bmatrix}_{n\times(n-m)} x_N, \qquad (10)$$

其中特解 Yb 是通過將 x 的 $n-m$ 個分量保持為零同時放鬆其他 m 個分量（x_B 中的分量）直到它們滿足限制式而獲得。特解 Yb 有時被稱為坐標放鬆步驟 (coordinate relaxation step)。在（下頁）圖 3 中，我們可以看到通過選擇基底矩陣 B 為 A 的第一行所獲得的坐標放鬆步驟 Yb。如果我們將 B 選擇為 A 的第二行，坐標放鬆步驟將沿著 x_2 軸。

§15.3 Elimination of Variables

換句話說，簡單消去變數法將可行點表示為 $Ax = b$ 的特解 ((10) 式中的 Yb) 與沿著限制的 null space 的偏移 ((10) 式中的 Zx_N) 的總和：

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix}_{n\times m} b + \begin{bmatrix} -B^{-1}N \\ \mathrm{I} \end{bmatrix}_{n\times(n-m)} x_N, \qquad (10)$$

其中特解 Yb 是通過將 x 的 $n-m$ 個分量保持為零同時放鬆其他 m 個分量（x_B 中的分量）直到它們滿足限制式而獲得。特解 Yb 有時被稱為坐標放鬆步驟 (coordinate relaxation step)。在（下頁）圖 3 中，我們可以看到通過選擇基底矩陣 B 為 A 的第一行所獲得的坐標放鬆步驟 Yb。如果我們將 B 選擇為 A 的第二行，坐標放鬆步驟將沿著 x_2 軸。

§15.3 Elimination of Variables

換句話說，簡單消去變數法將可行點表示為 $Ax = b$ 的特解 ((10) 式中的 Yb) 與沿著限制的 null space 的偏移 ((10) 式中的 Zx_N) 的總和：

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix}_{n\times m} b + \begin{bmatrix} -B^{-1}N \\ \mathrm{I} \end{bmatrix}_{n\times(n-m)} x_N, \qquad (10)$$

其中特解 Yb 是通過將 x 的 $n-m$ 個分量保持為零同時放鬆其他 m 個分量（x_B 中的分量）直到它們滿足限制式而獲得。特解 Yb 有時被稱為坐標放鬆步驟 (coordinate relaxation step)。在（下頁）圖 3 中，我們可以看到通過選擇基底矩陣 B 為 A 的第一行所獲得的坐標放鬆步驟 Yb。如果我們將 B 選擇為 A 的第二行，坐標放鬆步驟將沿著 x_2 軸。

§15.3 Elimination of Variables

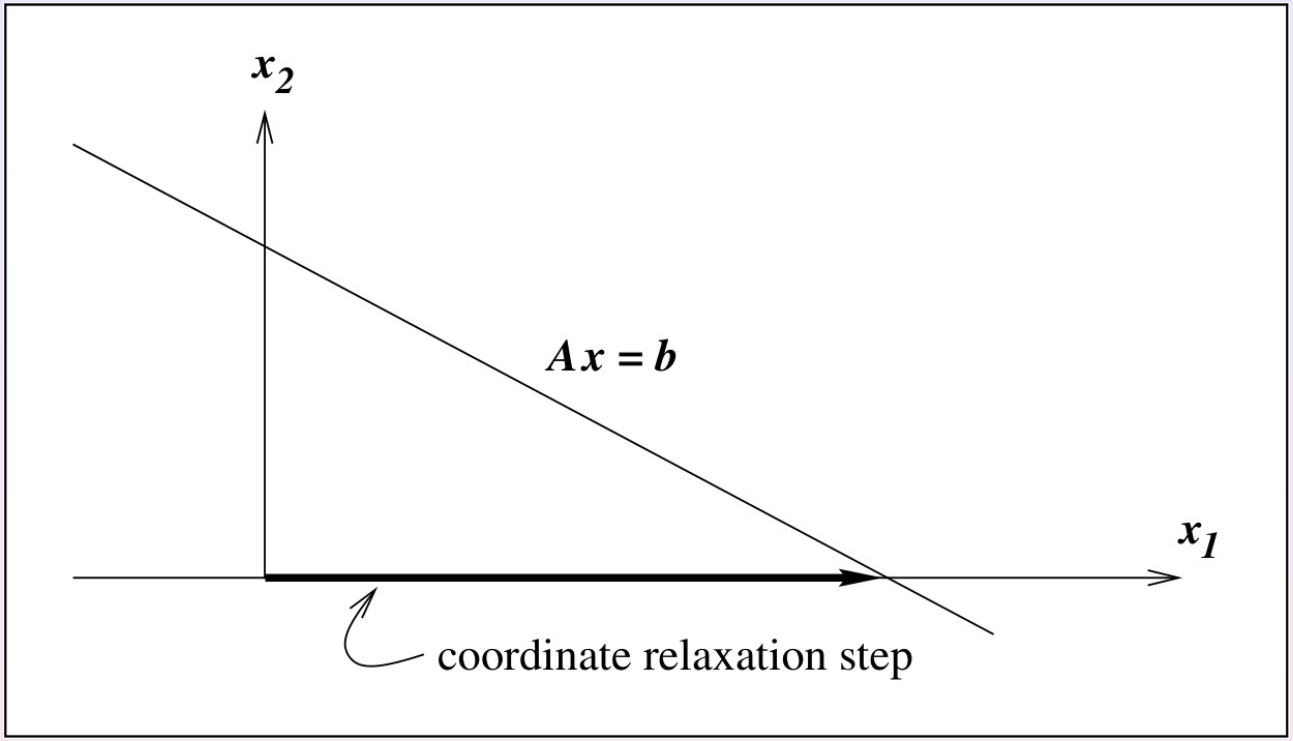

Figure 3: Simple elimination, showing the coordinate relaxation step obtained by choosing the basis to be the first column of A.

§15.3 Elimination of Variables

簡單消除變數法計算量較低，但可能產生數值不穩定。假如在圖 3 中的可行集合 (feasible set) 是一條幾乎平行於 x_1 軸的直線，沿著這個軸的坐標鬆弛步驟之 magnitude 會非常大。此時，我們計算實際的 minimizer x_* 時，會將其視為與超長向量 Yb 間的差異，這將導致數值上的抵銷效應 (numerical cancellation)。在這種情況下，最好選擇沿著 x_2 軸的特解，也就是選擇不同的基底。然而一般來說，選擇最佳基底並非一個簡單的任務。為了克服坐標鬆弛步驟過大的危險，我們可以將特解選擇為滿足限制的最小（範數）步進量。這種方法是更一般性的消除策略的特例，我們接下來進行描述。

§15.3 Elimination of Variables

簡單消除變數法計算量較低，但可能產生數值不穩定。假如在圖 3 中的可行集合 (feasible set) 是一條幾乎平行於 x_1 軸的直線，沿著這個軸的坐標鬆弛步驟之 magnitude 會非常大。此時，我們計算實際的 minimizer x_* 時，會將其視為與超長向量 Yb 間的差異，這將導致數值上的抵銷效應 (numerical cancellation)。在這種情況下，最好選擇沿著 x_2 軸的特解，也就是選擇不同的基底。然而一般來說，選擇最佳基底並非一個簡單的任務。為了克服坐標鬆弛步驟過大的危險，我們可以將特解選擇為滿足限制的最小（範數）步進量。這種方法是更一般性的消除策略的特例，我們接下來進行描述。

§15.3 Elimination of Variables

- **General reduction strategies for linear constraints**

為了推廣 (10) 與 (11) 式，我們選擇滿足

$$[Y \vdots Z] \in \mathbb{R}^{n\times n} \text{ is non-singular}, \quad AZ = 0\,. \tag{12}$$

的 $n \times m$ 實矩陣 Y 及 $n \times (n-m)$ 實矩陣 Z。就像在 (11) 中一樣，這些性質說明矩陣 Z 的行構成 A 的 null space 的基底。由於 A 有滿秩，因此 $A[Y \vdots Z] = [AY \vdots 0]$ 也有滿秩，這意味著 $m \times m$ 矩陣 AY 是 non-singular。因為線性限制式 $Ax = b$ 的解可表示為

$$x = Yx_Y + Zx_Z\,, \tag{13}$$

其中 $x_Y \in \mathbb{R}^m$ 且 $x_Z \in \mathbb{R}^{n-m}$，將 (13) 代入限制方程式可得

$$Ax = (AY)x_Y = b;$$

因此，由於 AY 是 non-singular，x_Y 可以被明確地寫成

$$x_Y = (AY)^{-1}b. \tag{14}$$

§15.3 Elimination of Variables

- **General reduction strategies for linear constraints**

為了推廣 (10) 與 (11) 式，我們選擇滿足

$$[Y \vdots Z] \in \mathbb{R}^{n\times n} \text{ is non-singular}, \quad AZ = 0\,. \tag{12}$$

的 $n \times m$ 實矩陣 Y 及 $n \times (n-m)$ 實矩陣 Z。就像在 (11) 中一樣，這些性質說明矩陣 Z 的行構成 A 的 null space 的基底。由於 A 有滿秩，因此 $A[Y \vdots Z] = [AY \vdots 0]$ 也有滿秩，這意味著 $m \times m$ 矩陣 AY 是 non-singular。因為線性限制式 $Ax = b$ 的解可表示為

$$x = Yx_Y + Zx_Z\,, \tag{13}$$

其中 $x_Y \in \mathbb{R}^m$ 且 $x_Z \in \mathbb{R}^{n-m}$，將 (13) 代入限制方程式可得

$$Ax = (AY)x_Y = b;$$

因此，由於 AY 是 non-singular，x_Y 可以被明確地寫成

$$x_Y = (AY)^{-1}b. \tag{14}$$

§15.3 Elimination of Variables

- **General reduction strategies for linear constraints**

為了推廣 (10) 與 (11) 式，我們選擇滿足

$$[Y \vdots Z] \in \mathbb{R}^{n\times n} \text{ is non-singular}, \quad AZ = 0\,. \tag{12}$$

的 $n \times m$ 實矩陣 Y 及 $n \times (n-m)$ 實矩陣 Z。就像在 (11) 中一樣，這些性質說明矩陣 Z 的行構成 A 的 null space 的基底。由於 A 有滿秩，因此 $A[Y \vdots Z] = [AY \vdots 0]$ 也有滿秩，這意味著 $m \times m$ 矩陣 AY 是 non-singular。因為線性限制式 $Ax = b$ 的解可表示為

$$x = Yx_Y + Zx_Z\,, \tag{13}$$

其中 $x_Y \in \mathbb{R}^m$ 且 $x_Z \in \mathbb{R}^{n-m}$，將 (13) 代入限制方程式可得

$$Ax = (AY)x_Y = b;$$

因此，由於 AY 是 non-singular，x_Y 可以被明確地寫成

$$x_Y = (AY)^{-1}b. \tag{14}$$

§15.3 Elimination of Variables

通過將 (14) 這個表達式代入 (13)，我們得出結論：任何滿足 $Ax = b$ 的向量 x 都可以寫成

$$x = Y(AY)^{-1}b + Zx_Z \tag{15}$$

其中 $x_Z \in \mathbb{R}^{n-m}$，反之亦然。因此，問題 (5) 可以等價於以下的無限制問題

$$\min_{x_Z \in \mathbb{R}^{n-m}} f\left(Y(AY)^{-1}b + Zx_Z\right).$$

理想情況下，我們希望選擇 Y 使得矩陣 AY 的條件數盡可能好（小），因為 (14) 式中需要計算 $(AY)^{-1}b$。我們可以透過計算 A^{T} 的 QR 分解來計算 $(AY)^{-1}b$，其形式為

$$A^{\mathrm{T}}\Pi = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{16}$$

其中矩陣 $\left[Q_1 \vdots Q_2\right]$ 是正交矩陣，R 是 $m \times m$ 的上三角 non-singular 矩陣，Π 是一個 $m \times m$ 的置換矩陣。

§15.3 Elimination of Variables

通過將 (14) 這個表達式代入 (13)，我們得出結論：任何滿足 $Ax = b$ 的向量 x 都可以寫成

$$x = Y(AY)^{-1}b + Zx_Z \tag{15}$$

其中 $x_Z \in \mathbb{R}^{n-m}$，反之亦然。因此，問題 (5) 可以等價於以下的無限制問題

$$\min_{x_Z \in \mathbb{R}^{n-m}} f\left(Y(AY)^{-1}b + Zx_Z\right).$$

理想情況下，我們希望選擇 Y 使得矩陣 AY 的條件數盡可能好（小），因為 (14) 式中需要計算 $(AY)^{-1}b$。我們可以透過計算 A^{T} 的 QR 分解來計算 $(AY)^{-1}b$，其形式為

$$A^{\mathrm{T}}\Pi = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{16}$$

其中矩陣 $\left[Q_1 \vdots Q_2\right]$ 是正交矩陣，R 是 $m \times m$ 的上三角 non-singular 矩陣，Π 是一個 $m \times m$ 的置換矩陣。

§15.3 Elimination of Variables

在 (16) 中矩陣 $Q_1 \in \mathbb{R}^{n\times m}$ 和 $Q_2 \in \mathbb{R}^{n\times(n-m)}$ 具有彼此正交的行向量。若定義

$$Y = Q_1\,, \quad Z = Q_2\,.$$

則 Y 和 Z 的行形成 $\mathbb{R}^n$ 的一組正交基底，且展開 (16) 並重新排列後可得

$$AY = \Pi R^{\mathrm{T}}, \quad AZ = 0\,.$$

因此，Y 和 Z 滿足

$$[Y \vdots Z] \in \mathbb{R}^{n\times n} \text{ is non-singular}, \quad AZ = 0\,, \tag{12}$$

且 AY 的條件數與 R 的條件數相同，進而等於 A 本身的條件數。

從 (15) 我們看到，任何滿足 $Ax = b$ 的解都可以表示為

$$x = Q_1 R^{-\mathrm{T}} \Pi^{\mathrm{T}} b + Q_2 x_z\,,$$

其中 x_z 為某個向量。$R^{-\mathrm{T}}\Pi^{\mathrm{T}}b$ 的計算量並不高，因其只需進行一次逆下三角矩陣操作。

§15.3 Elimination of Variables

在 (16) 中矩陣 $Q_1 \in \mathbb{R}^{n\times m}$ 和 $Q_2 \in \mathbb{R}^{n\times(n-m)}$ 具有彼此正交的行向量。若定義

$$Y = Q_1\,, \quad Z = Q_2\,.$$

則 Y 和 Z 的行形成 $\mathbb{R}^n$ 的一組正交基底，且展開 (16) 並重新排列後可得

$$AY = \Pi R^{\mathrm{T}}, \quad AZ = 0\,.$$

因此，Y 和 Z 滿足

$$[Y \vdots Z] \in \mathbb{R}^{n\times n} \text{ is non-singular}, \quad AZ = 0\,, \tag{12}$$

且 AY 的條件數與 R 的條件數相同，進而等於 A 本身的條件數。

從 (15) 我們看到，任何滿足 $Ax = b$ 的解都可以表示為

$$x = Q_1 R^{-\mathrm{T}} \Pi^{\mathrm{T}} b + Q_2 x_z\,,$$

其中 x_z 為某個向量。$R^{-\mathrm{T}}\Pi^{\mathrm{T}}b$ 的計算量並不高，因其只需進行一次逆下三角矩陣操作。

§15.3 Elimination of Variables

在 (16) 中矩陣 $Q_1 \in \mathbb{R}^{n\times m}$ 和 $Q_2 \in \mathbb{R}^{n\times(n-m)}$ 具有彼此正交的行向量。若定義

$$Y = Q_1\,, \quad Z = Q_2\,.$$

則 Y 和 Z 的行形成 $\mathbb{R}^n$ 的一組正交基底，且展開 (16) 並重新排列後可得

$$AY = \Pi R^{\mathrm{T}}, \quad AZ = 0\,.$$

因此，Y 和 Z 滿足

$$[Y \vdots Z] \in \mathbb{R}^{n\times n} \text{ is non-singular}, \quad AZ = 0\,, \tag{12}$$

且 AY 的條件數與 R 的條件數相同，進而等於 A 本身的條件數。

從 (15) 我們看到，任何滿足 $Ax = b$ 的解都可以表示為

$$x = Q_1 R^{-\mathrm{T}} \Pi^{\mathrm{T}} b + Q_2 x_Z\,,$$

其中 x_Z 為某個向量。$R^{-\mathrm{T}}\Pi^{\mathrm{T}}b$ 的計算量並不高，因其只需進行一次逆下三角矩陣操作。

§15.3 Elimination of Variables

進一步計算顯示 $Q_1 R^{-\mathrm{T}} \Pi^{\mathrm{T}} b = A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1} b$，而右邊項為問題

$$\min \|x\|_2 \quad \text{subject to} \quad Ax = b$$

的解。也就是說，特解是方程 $Ax = b$ 的最小範數解。請參見圖 4 以了解這一步驟的示意圖。

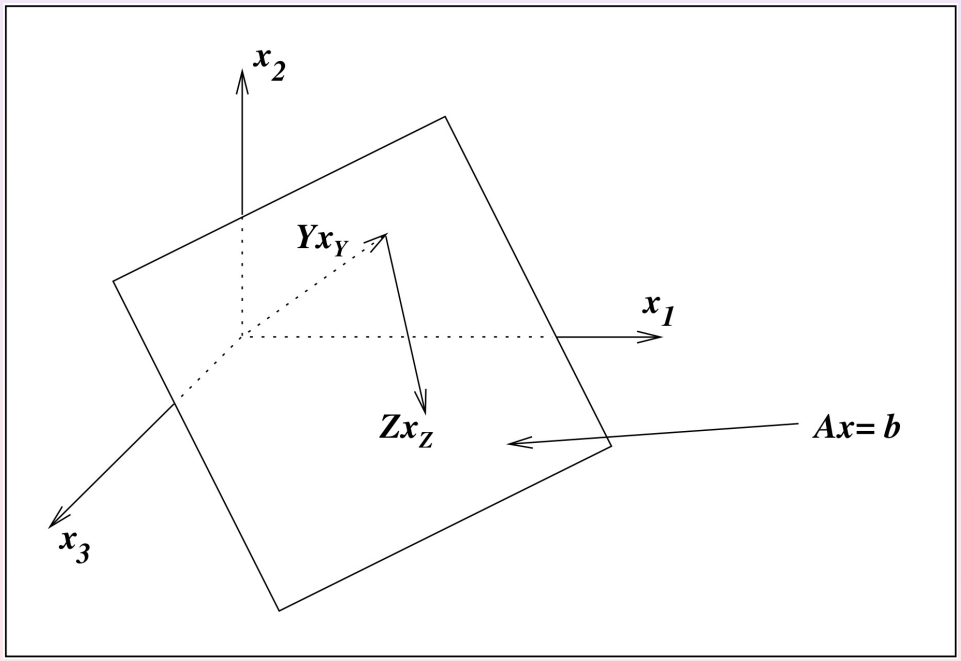

Figure 4: General elimination: Case in which $A \in \mathbb{R}^{1\times 3}$, showing the particular solution and a step in the null space of A.

§15.3 Elimination of Variables

從數值穩定性的角度來看，通過 Q_1, Q_2 進行變數消除是理想的，而與此簡化策略相關的主要成本在於計算 QR 分解 (16)。不幸的是，對於 A 大且稀疏 (sparse) 的問題，稀疏 QR 分解的計算成本可能比簡單消除中使用的稀疏高斯消除策略更高。因此，已經發展出其他消除策略，它們在這兩種技術之間尋求折衷。

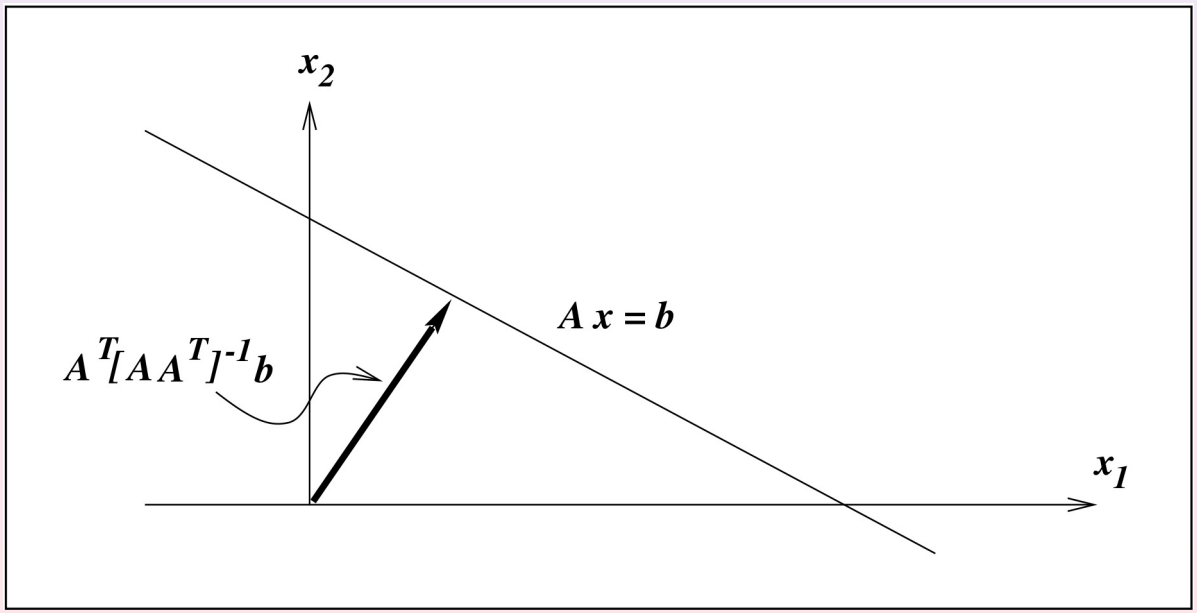

Figure 5: The minimum-norm step.

§15.3 Elimination of Variables

- **Effect of inequality constraints**

如果問題同時存在不等式限制，消除變數並非總是有益的。例如，若問題

$$\min \sin(x_1 + x_2) + x_3^2 + \frac{1}{3}\left(x_4 + x_5^4 + \frac{x_6}{2}\right) \tag{9a}$$

subject to

$$\begin{aligned} 8x_1 - 6x_2 + x_3 + 9x_4 + 4x_5 &= 6\,, \\ 3x_1 + 2x_2 - x_4 + 6x_5 + 4x_6 &= -4\,, \end{aligned} \tag{9b}$$

中有額外的限制 $x \geqslant 0$，那麼在消去變數 x_3 和 x_6 後，我們將得到在滿足限制

$$\begin{aligned} (x_1, x_2, x_4, x_5) &\geqslant 0\,, \\ 8x_1 - 6x_2 + 9x_4 + 4x_5 &\leqslant 6\,, \\ \frac{3}{4}x_1 + \frac{1}{2}x_2 - \frac{1}{4}x_4 + \frac{3}{2}x_5 &\leqslant -1\,, \end{aligned}$$

§15.3 Elimination of Variables

（承上頁）的情況下最小化函數

$$\sin(x_1 + x_2) + (8x_1 - 6x_2 + 9x_4 + 4x_5 - 6)^2 + \frac{1}{3}\left[x_4 + x_5^4 - \left(\frac{1}{2} + \frac{3}{8}x_1 + \frac{1}{4}x_2 - \frac{1}{8}x_4 + \frac{3}{4}x_5\right)\right].$$

的問題。因此，消除等式限制 (9b) 的代價是使不等式變得比簡單的界限 $x \geqslant 0$ 更複雜，對於許多算法而言，這種轉換可能不會帶來任何好處。然而，如果問題 (9) 包括一個一般的不等式限制 $3x_1 + 2x_3 \geqslant 1$，那麼消除變數 x_3, x_6 可以將問題轉化為在滿足不等式限制

$$-13x_1 + 12x_2 - 18x_4 - 8x_5 \geqslant -11.$$

的情況下最小化上述函數的問題。在這種情況下，消除等式限制後，不等式限制不會變得更複雜，因此進行消除可能是值得的。

§15.3 Elimination of Variables

（承上頁）的情況下最小化函數

$$\sin(x_1 + x_2) + (8x_1 - 6x_2 + 9x_4 + 4x_5 - 6)^2 + \frac{1}{3}\left[x_4 + x_5^4 - \left(\frac{1}{2} + \frac{3}{8}x_1 + \frac{1}{4}x_2 - \frac{1}{8}x_4 + \frac{3}{4}x_5\right)\right].$$

的問題。因此，消除等式限制 (9b) 的代價是使不等式變得比簡單的界限 $x \geqslant 0$ 更複雜，對於許多算法而言，這種轉換可能不會帶來任何好處。然而，如果問題 (9) 包括一個一般的不等式限制 $3x_1 + 2x_3 \geqslant 1$，那麼消除變數 x_3, x_6 可以將問題轉化為在滿足不等式限制

$$-13x_1 + 12x_2 - 18x_4 - 8x_5 \geqslant -11.$$

的情況下最小化上述函數的問題。在這種情況下，消除等式限制後，不等式限制不會變得更複雜，因此進行消除可能是值得的。

§15.3 Elimination of Variables

（承上頁）的情況下最小化函數

$$\sin(x_1 + x_2) + (8x_1 - 6x_2 + 9x_4 + 4x_5 - 6)^2 + \frac{1}{3}\left[x_4 + x_5^4 - \left(\frac{1}{2} + \frac{3}{8}x_1 + \frac{1}{4}x_2 - \frac{1}{8}x_4 + \frac{3}{4}x_5\right)\right].$$

的問題。因此，消除等式限制 (9b) 的代價是使不等式變得比簡單的界限 $x \geqslant 0$ 更複雜，對於許多算法而言，這種轉換可能不會帶來任何好處。然而，如果問題 (9) 包括一個一般的不等式限制 $3x_1 + 2x_3 \geqslant 1$，那麼消除變數 x_3, x_6 可以將問題轉化為在滿足不等式限制

$$-13x_1 + 12x_2 - 18x_4 - 8x_5 \geqslant -11.$$

的情況下最小化上述函數的問題。在這種情況下，消除等式限制後，不等式限制不會變得更複雜，因此進行消除可能是值得的。

§15.4 Merit Functions and Filters

假設解非線性規劃問題

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0 & \text{if } i \in \mathcal{E}, \\ c_i(x) \geqslant 0 & \text{if } i \in \mathcal{I}, \end{cases} \tag{1}$$

的某個演算法（例如第 17 章提到的 penalty 法、擴增 Lagrangian 法以及第 18 章的 SQP 法都是這類的演算法）生成了一個「可以減少目標函數值但卻增加了違反限制條件的個數」的步進量，我們應該接受這個步進量嗎？這個問題並不容易回答。我們必須尋找一種平衡減少目標函數值並滿足限制條件這兩個（通常是互相競爭的）目標的方法。接下來我們要介紹的是可以實現這種平衡的兩種方法：merit function 和 filter。在典型的受限優化算法中，只有當步進量導致 merit function 充分減少，或者它在 filter 中是可接受的時候，步進量才會被接受。這些概念在本節的其餘部分進行解釋。

§15.4 Merit Functions and Filters

假設解非線性規劃問題

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0 & \text{if } i \in \mathcal{E}, \\ c_i(x) \geqslant 0 & \text{if } i \in \mathcal{I}, \end{cases} \tag{1}$$

的某個演算法（例如第 17 章提到的 penalty 法、擴增 Lagrangian 法以及第 18 章的 SQP 法都是這類的演算法）生成了一個「可以減少目標函數值但卻增加了違反限制條件的個數」的步進量，我們應該接受這個步進量嗎？這個問題並不容易回答。我們必須尋找一種平衡減少目標函數值並滿足限制條件這兩個（通常是互相競爭的）目標的方法。接下來我們要介紹的是可以實現這種平衡的兩種方法：merit function 和 filter。在典型的受限優化算法中，只有當步進量導致 merit function 充分減少，或者它在 filter 中是可接受的時候，步進量才會被接受。這些概念在本節的其餘部分進行解釋。

§15.4 Merit Functions and Filters

- **Merit functions**

在無受限優化的討論中，目標函數 f 是 merit function 的自然選擇：本書描述的所有無受限優化方法都要求在每一步（或至少在一定數量的迭代中）降低 f 的函數值。對於受限優化的可行方法 (feasible methods)，若要求起始點和所有後續迭代點滿足問題中的所有限制式，則目標函數仍然是一個合適的 merit function。另一方面，允許後續迭代點違反限制式的演算法需要某種手段來評估步進量和迭代的質量，在這種情況下，merit function 將目標函數與違反限制的度量結合在一起。

§15.4 Merit Functions and Filters

- **Merit functions**

在無受限優化的討論中，目標函數 f 是 merit function 的自然選擇：本書描述的所有無受限優化方法都要求在每一步（或至少在一定數量的迭代中）降低 f 的函數值。對於受限優化的可行方法 (feasible methods)，若要求起始點和所有後續迭代點滿足問題中的所有限制式，則目標函數仍然是一個合適的 merit function。另一方面，允許後續迭代點違反限制式的演算法需要某種手段來評估步進量和迭代的質量，在這種情況下，merit function 將目標函數與違反限制的度量結合在一起。

§15.4 Merit Functions and Filters

- **Merit functions**

在無受限優化的討論中，目標函數 f 是 merit function 的自然選擇：本書描述的所有無受限優化方法都要求在每一步（或至少在一定數量的迭代中）降低 f 的函數值。對於受限優化的可行方法 (feasible methods)，若要求起始點和所有後續迭代點滿足問題中的所有限制式，則目標函數仍然是一個合適的 merit function。另一方面，允許後續迭代點違反限制式的演算法需要某種手段來評估步進量和迭代的質量，在這種情況下，merit function 將目標函數與違反限制的度量結合在一起。

§15.4 Merit Functions and Filters

在非線性規劃問題 (1) 中，一個常用的 merit function 是由 ℓ_1 懲罰函數 (penalty function) 定義的，其表達式如下：

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^- ,$$

在此符號 $[z]^-$ 代表實數 z 的負部，其定義為 $[z]^- = \max\{0, -z\}$。正參數 μ 稱為懲罰參數 (penalty parameter)，它決定我們相對於目標最小化給予限制滿足的權重。ℓ_1 merit function ϕ_1 由於絕對值和 $[\cdot]^-$ 函數的存在而不可微，但它具有準確 (exact) 性的重要特性。

Definition (Exact Merit Function)

A merit function $\phi(x;\mu)$ is **exact** if there is a positive scalar μ_* such that for any $\mu > \mu_*$, any local solution of the nonlinear programming problem (1) is a local minimizer of $\phi(x;\mu)$.

§15.4 Merit Functions and Filters

在非線性規劃問題 (1) 中，一個常用的 merit function 是由 ℓ_1 懲罰函數 (penalty function) 定義的，其表達式如下：

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^- ,$$

在此符號 $[z]^-$ 代表實數 z 的負部，其定義為 $[z]^- = \max\{0, -z\}$。正參數 μ 稱為懲罰參數 (penalty parameter)，它決定我們相對於目標最小化給予限制滿足的權重。ℓ_1 merit function ϕ_1 由於絕對值和 $[\cdot]^-$ 函數的存在而不可微，但它具有準確 (exact) 性的重要特性。

Definition (Exact Merit Function)

A merit function $\phi(x;\mu)$ is **exact** if there is a positive scalar μ_* such that for any $\mu > \mu_*$, any local solution of the nonlinear programming problem (1) is a local minimizer of $\phi(x;\mu)$.

§15.4 Merit Functions and Filters

在稍後的定理中，我們證明在一定的假設下，ℓ_1 merit function $\phi_1(x;\mu)$ 是 exact，其門檻值參數 (threshold value) μ_* 為

$$\mu_* = \max\left\{|\lambda_i^*| \,\middle|\, i \in \mathcal{E} \cup \mathcal{I}\right\},$$

其中 λ_i^* 是與最優解 x_* 相關聯的 Lagrange 乘子。然而，由於最優 Lagrange 乘子事先是未知的，基於 ℓ_1 merit function 的演算法通常會包含在有理由相信懲罰參數不夠大（或過大）時調整懲罰參數的規則。這些規則取決於優化算法的選擇，並在接下來的章節中進行討論。

另一個有用的 exact merit function 是使用了 ℓ_2 norm：對於等式限制問題 exact ℓ_2 merit function 的形式為

$$\phi_2(x;\mu) = f(x) + \mu\|c(x)\|_2\,.$$

§15.4 Merit Functions and Filters

在稍後的定理中，我們證明在一定的假設下，ℓ_1 merit function $\phi_1(x;\mu)$ 是 exact，其門檻值參數 (threshold value) μ_* 為

$$\mu_* = \max\left\{|\lambda_i^*| \,\middle|\, i \in \mathcal{E} \cup \mathcal{I}\right\},$$

其中 λ_i^* 是與最優解 x_* 相關聯的 Lagrange 乘子。然而，由於最優 Lagrange 乘子事先是未知的，基於 ℓ_1 merit function 的演算法通常會包含在有理由相信懲罰參數不夠大（或過大）時調整懲罰參數的規則。這些規則取決於優化算法的選擇，並在接下來的章節中進行討論。

另一個有用的 exact merit function 是使用了 ℓ_2 norm：對於等式限制問題 exact ℓ_2 merit function 的形式為

$$\phi_2(x;\mu) = f(x) + \mu\|c(x)\|_2\,.$$

§15.4 Merit Functions and Filters

然而因為 2-norm 項沒有平方，函數 ϕ_2 並非到處可微的：它在滿足 $c(x) = 0$ 的 x 點的導數未定義。有些 merit function 既平滑又 exact，但為同時確保平滑性與 exactness，我們必須在 merit function 中加入額外的項。例如對於只具等式限制的最佳化問題，Fletcher 的增廣 Lagrangian 如下：

$$\phi_F(x;\mu) = f(x) - \lambda(x)^{\mathrm{T}} c(x) + \frac{1}{2}\mu \sum_{i\in\mathcal{E}} c_i(x)^2 , \tag{17}$$

其中 $\mu > 0$ 是懲罰參數，而若以 $A(x)$ 表示 $c(x)$ 的 Jacobian matrix，則 λ 為

$$\lambda(x) = \left[A(x)A(x)^{\mathrm{T}}\right]^{-1} A(x)\nabla f(x) . \tag{18}$$

這個 merit function 雖具有一些有趣的理論性質，但實際上存在一些限制，其中包括在 (18) 中計算 $\lambda(x)$ 的計算量。

§15.4 Merit Functions and Filters

然而因為 2-norm 項沒有平方，函數 ϕ_2 並非到處可微的：它在滿足 $c(x) = 0$ 的 x 點的導數未定義。有些 merit function 既平滑又 exact，但為同時確保平滑性與 exactness，我們必須在 merit function 中加入額外的項。例如對於只具等式限制的最佳化問題，Fletcher 的增廣 Lagrangian 如下：

$$\phi_F(x;\mu) = f(x) - \lambda(x)^{\mathrm{T}} c(x) + \frac{1}{2}\mu \sum_{i\in\mathcal{E}} c_i(x)^2 , \tag{17}$$

其中 $\mu > 0$ 是懲罰參數，而若以 $A(x)$ 表示 $c(x)$ 的 Jacobian matrix，則 λ 為

$$\lambda(x) = \left[A(x)A(x)^{\mathrm{T}}\right]^{-1} A(x)\nabla f(x) . \tag{18}$$

這個 merit function 雖具有一些有趣的理論性質，但實際上存在一些限制，其中包括在 (18) 中計算 $\lambda(x)$ 的計算量。

§15.4 Merit Functions and Filters

然而因為 2-norm 項沒有平方，函數 ϕ_2 並非到處可微的：它在滿足 $c(x) = 0$ 的 x 點的導數未定義。有些 merit function 既平滑又 exact，但為同時確保平滑性與 exactness，我們必須在 merit function 中加入額外的項。例如對於只具等式限制的最佳化問題，Fletcher 的增廣 Lagrangian 如下：

$$\phi_F(x;\mu) = f(x) - \lambda(x)^{\mathrm{T}} c(x) + \frac{1}{2}\mu \sum_{i\in\mathcal{E}} c_i(x)^2 , \tag{17}$$

其中 $\mu > 0$ 是懲罰參數，而若以 $A(x)$ 表示 $c(x)$ 的 Jacobian matrix，則 λ 為

$$\lambda(x) = \left[A(x)A(x)^{\mathrm{T}}\right]^{-1} A(x)\nabla f(x) . \tag{18}$$

這個 merit function 雖具有一些有趣的理論性質，但實際上存在一些限制，其中包括在 (18) 中計算 $\lambda(x)$ 的計算量。

§15.4 Merit Functions and Filters

一個非常不同的 merit function 是（標準的）增廣 Lagrangian，對於等式限制問題，其形式為

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \lambda^{\mathrm{T}} c(x) + \frac{1}{2}\mu \|c(x)\|_2^2 .$$

我們通過比較測試點 (trial point) (x^+, λ^+) 的 $\mathcal{L}_A(x^+, \lambda^+; \mu)$ 值與當前迭代點 (x, λ) 的值來評估其可接受性。嚴格來說，$\mathcal{L}_A$ 在解非線性規劃問題方面並不是一個 merit function，因為 (x_*, λ_*) 通常不是 $\mathcal{L}_A(x, \lambda; \mu)$ 的 minimizer 而只是一個 stationary point。儘管一些序列二次規劃 (SQP) 方法通過自適應地修改 μ 和 λ 成功地將增廣 Lagrangian $\mathcal{L}_A$ 作為一個 merit function，但在之後的章節我們不會將其視為一個 merit function 來使用。相反地，我們將主要聚焦於非平滑的 exact 懲罰函數 ϕ_1 與 ϕ_2。

§15.4 Merit Functions and Filters

一個非常不同的 merit function 是（標準的）增廣 Lagrangian，對於等式限制問題，其形式為

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \lambda^{\mathrm{T}} c(x) + \frac{1}{2}\mu \|c(x)\|_2^2 .$$

我們通過比較測試點 (trial point) (x^+, λ^+) 的 $\mathcal{L}_A(x^+, \lambda^+; \mu)$ 值與當前迭代點 (x, λ) 的值來評估其可接受性。嚴格來說，$\mathcal{L}_A$ 在解非線性規劃問題方面並不是一個 merit function，因為 (x_*, λ_*) 通常不是 $\mathcal{L}_A(x, \lambda; \mu)$ 的 minimizer 而只是一個 stationary point。儘管一些序列二次規劃 (SQP) 方法通過自適應地修改 μ 和 λ 成功地將增廣 Lagrangian $\mathcal{L}_A$ 作為一個 merit function，但在之後的章節我們不會將其視為一個 merit function 來使用。相反地，我們將主要聚焦於非平滑的 exact 懲罰函數 ϕ_1 與 ϕ_2。

§15.4 Merit Functions and Filters

由 line search 生成的試驗步 $x^+ = x + \alpha p$ 若對 merit function $\phi(x;\mu)$ 之值產生足夠的遞減量 (sufficient decrease)，則此步進量將被接受。定義「函數值有足夠遞減量」這個概念的一種方式與無受限優化中使用的 Armijo condition 類似：其函數值減少的量相對於在步進中函數的預測變化不能太小。

ℓ_1 和 ℓ_2 merit functions 並不可微，但它們在所有方向的方向導數都存在。在以下的討論中，我們將 $\phi(x;\mu)$ 在方向 p 上的方向導數表示為 $D(\phi(x;\mu);p)$.

§15.4 Merit Functions and Filters

由 line search 生成的試驗步 $x^+ = x + \alpha p$ 若對 merit function $\phi(x;\mu)$ 之值產生足夠的遞減量 (sufficient decrease)，則此步進量將被接受。定義「函數值有足夠遞減量」這個概念的一種方式與無受限優化中使用的 Armijo condition 類似：其函數值減少的量相對於在步進中函數的預測變化不能太小。

ℓ_1 和 ℓ_2 merit functions 並不可微，但它們在所有方向的方向導數都存在。在以下的討論中，我們將 $\phi(x;\mu)$ 在方向 p 上的方向導數表示為 $D(\phi(x;\mu);p)$.

§15.4 Merit Functions and Filters

在 line search 方法中，函數值有足夠遞減量這個條件要求步長參數 $\alpha \geqslant 0$ 足夠小，使得不等式

$$\phi(x+\alpha p;\mu) \leqslant \phi(x;\mu) + \eta\alpha D(\phi(x;\mu);p)\,, \tag{19}$$

對於某個 $\eta \in (0,1)$ 成立。

當想要使用 trust-region 方法處理受限優化問題時，通常會使用一個二次模型 $m(p)$ 來估計在步進之後 merit function ϕ 的值，而函數值有足夠遞減量的條件可以根據這個模型的減小來陳述。具體來說，在 trust-region 方法中函數值有足夠遞減量的條件是存在 $\eta \in (0,1)$ 使得

$$\phi(x+p;\mu) \leqslant \phi(x;\mu) - \eta(m(0) - m(p)), \tag{20}$$

注意到 (20) 式中的最後一項是正的，因為步進量 p 被計算為使模型 m 的函數值減少。關於這部份的具體細節請參見 §18.5。

§15.4 Merit Functions and Filters

在 line search 方法中，函數值有足夠遞減量這個條件要求步長參數 $\alpha \geqslant 0$ 足夠小，使得不等式

$$\phi(x+\alpha p;\mu) \leqslant \phi(x;\mu) + \eta\alpha D(\phi(x;\mu);p)\,, \tag{19}$$

對於某個 $\eta \in (0,1)$ 成立。

當想要使用 trust-region 方法處理受限優化問題時，通常會使用一個二次模型 $m(p)$ 來估計在步進之後 merit function ϕ 的值，而函數值有足夠遞減量的條件可以根據這個模型的減小來陳述。具體來說，在 trust-region 方法中函數值有足夠遞減量的條件是存在 $\eta \in (0,1)$ 使得

$$\phi(x+p;\mu) \leqslant \phi(x;\mu) - \eta(m(0) - m(p)), \tag{20}$$

注意到 (20) 式中的最後一項是正的，因為步進量 p 被計算為使模型 m 的函數值減少。關於這部份的具體細節請參見 §18.5。

§15.4 Merit Functions and Filters

- **Filters**

Filter 的技術是基於優化多個目標函數 (multi-objective optimization) 的想法所發展出來的步進量接受機制 (step acceptance mechanisms)。非線性規劃有兩個目標：最小化目標函數和滿足限制條件。如果我們定義一個不可行性度量 (measure of infeasibility) 為

$$h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \left[c_i(x)\right]^- , \tag{21}$$

我們可以將這兩個目標表示為

$$\min_x f(x) \quad \text{and} \quad \min_x h(x) . \tag{22}$$

與將兩個問題合併為一個最小化問題的 merit function 不同之處在於，filter 方法裡面 (22) 中的兩個目標函數始終分開考慮。

§15.4 Merit Functions and Filters

- **Filters**

Filter 的技術是基於優化多個目標函數 (multi-objective optimization) 的想法所發展出來的步進量接受機制 (step acceptance mechanisms)。非線性規劃有兩個目標：最小化目標函數和滿足限制條件。如果我們定義一個不可行性度量 (measure of infeasibility) 為

$$h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \left[c_i(x)\right]^{-}, \tag{21}$$

我們可以將這兩個目標表示為

$$\min_x f(x) \quad \text{and} \quad \min_x h(x)\,. \tag{22}$$

與將兩個問題合併為一個最小化問題的 merit function 不同之處在於，filter 方法裡面 (22) 中的兩個目標函數始終分開考慮。

§15.4 Merit Functions and Filters

- **Filters**

Filter 的技術是基於優化多個目標函數 (multi-objective optimization) 的想法所發展出來的步進量接受機制 (step acceptance mechanisms)。非線性規劃有兩個目標：最小化目標函數和滿足限制條件。如果我們定義一個不可行性度量 (measure of infeasibility) 為

$$h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \left[c_i(x)\right]^{-}, \tag{21}$$

我們可以將這兩個目標表示為

$$\min_x f(x) \quad \text{and} \quad \min_x h(x)\,. \tag{22}$$

與將兩個問題合併為一個最小化問題的 merit function 不同之處在於，filter 方法裡面 (22) 中的兩個目標函數始終分開考慮。

§15.4 Merit Functions and Filters

若試探步驟 x^+ 的函數值對 $(f(x^+), h(x^+))$ 不被算法先前所生成的函數值對 $(f_\ell, h_\ell) = (f(x_\ell), h(x_\ell))$ 所 dominated（定義在下），filter 方法將接受 x^+ 作為新的迭代點。這些概念的定義如下。

Definition

1. A pair (f_k, h_k) is said to dominate another pair (f_ℓ, h_ℓ) if both $f_k \leqslant f_\ell$ and $h_k \leqslant h_\ell$.
2. A filter is a list of pairs (f_ℓ, h_ℓ) such that no pair dominates any other.
3. An iterate x_k is said to be acceptable to the filter if (f_k, h_k) is not dominated by **any** pair in the filter.

當一個迭代點 x_k 被 filter 接受時，我們（通常）將 (f_k, h_k) 加入 filter，並刪除被 (f_k, h_k) dominate 的所有函數值對。

§15.4 Merit Functions and Filters

若試探步驟 x^+ 的函數值對 $(f(x^+), h(x^+))$ 不被算法先前所生成的函數值對 $(f_\ell, h_\ell) = (f(x_\ell), h(x_\ell))$ 所 dominated（定義在下），filter 方法將接受 x^+ 作為新的迭代點。這些概念的定義如下。

Definition

1. A pair (f_k, h_k) is said to dominate another pair (f_ℓ, h_ℓ) if both $f_k \leqslant f_\ell$ and $h_k \leqslant h_\ell$.
2. A filter is a list of pairs (f_ℓ, h_ℓ) such that no pair dominates any other.
3. An iterate x_k is said to be acceptable to the filter if (f_k, h_k) is not dominated by **any** pair in the filter.

當一個迭代點 x_k 被 filter 接受時，我們（通常）將 (f_k, h_k) 加入 filter，並刪除被 (f_k, h_k) dominate 的所有函數值對。

§15.4 Merit Functions and Filters

圖 6（下頁）顯示了一個 filter，其中 filter 中的每對 (f_ℓ, h_ℓ) 都以黑點表示。filter 中的每一點都以其為矩形的左下角創建一個（無窮大的）矩形區域，該矩形區域中的點不被 filter 接受，而（未被刪除的）每一個點所創建的矩形區域的聯集定義了不被 filter 接受的函數值對集合。更具體地說，在圖 6 中，如果試探點 x^+ 的函數值對 (f^+, h^+) 位於實線的下方或左側，則它會被 filter 接受的。

§15.4 Merit Functions and Filters

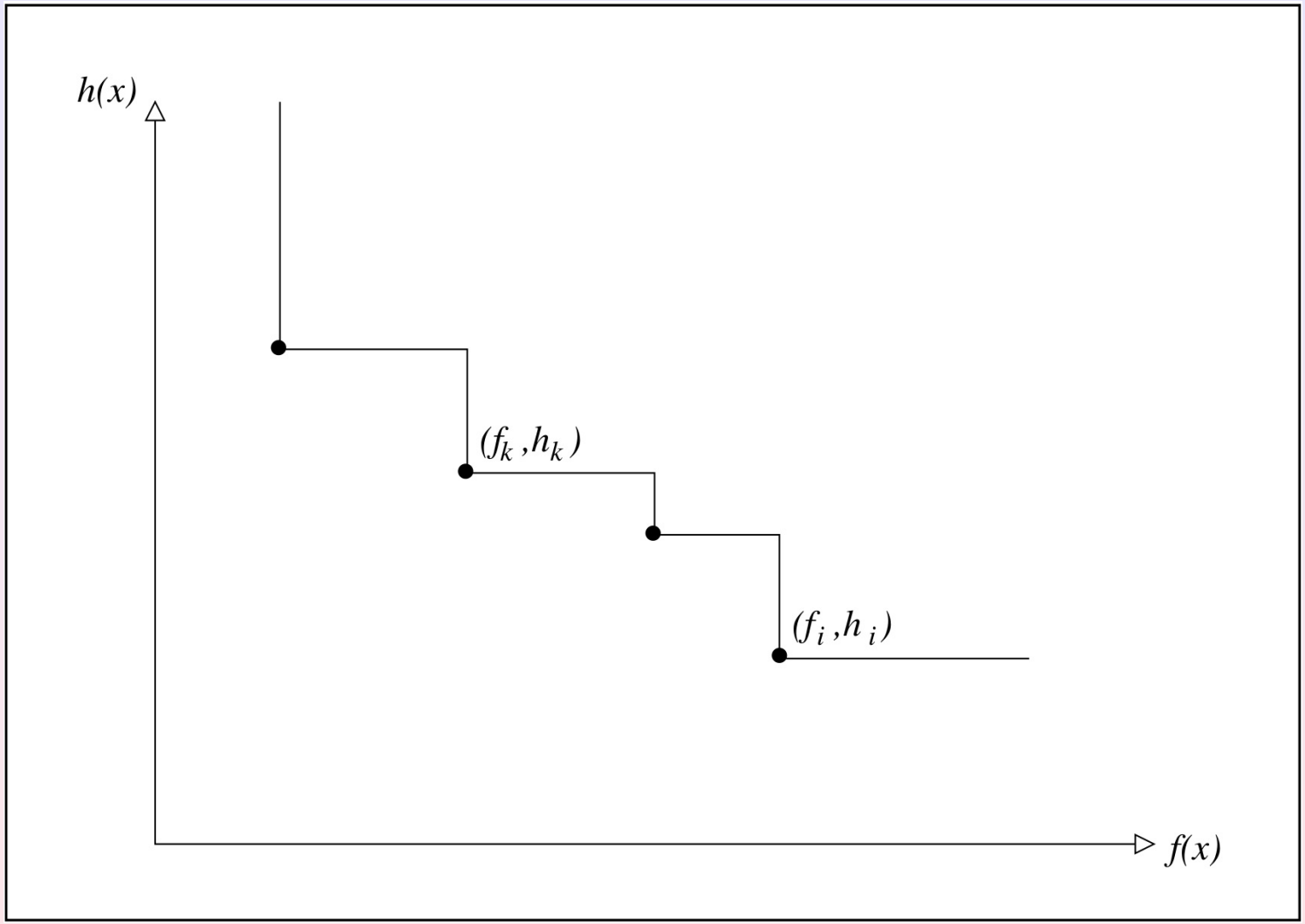

Figure 6: Graphical illustration of a filter with four pairs.

§15.4 Merit Functions and Filters

為了比較 filter 和 merit function 方法，我們在圖 7（下頁）中畫出滿足 $f+\mu h = f_k + \mu h_k$ 之函數值對 (f, h) 的等高線，其中 x_k 為當前迭代點。該線左側的區域對應於降低 merit function $\phi(x;\mu) = f(x) + \mu h(x)$ 的函數值對；顯然，這個集合與 filter 可接受的點的集合相當不同。

如果由 line search 方法生成的試探步驟 $x^+ = x_k + \alpha_k p_k$ 得到一個滿足 filter 的函數值對 (f^+, h^+)，我們設定 $x_{k+1} = x^+$；否則，就進行 backtracking line search。在 trust-region 方法中，如果步驟不符合 filter 的條件，信賴域將被縮小，並重新計算新的步驟。

§15.4 Merit Functions and Filters

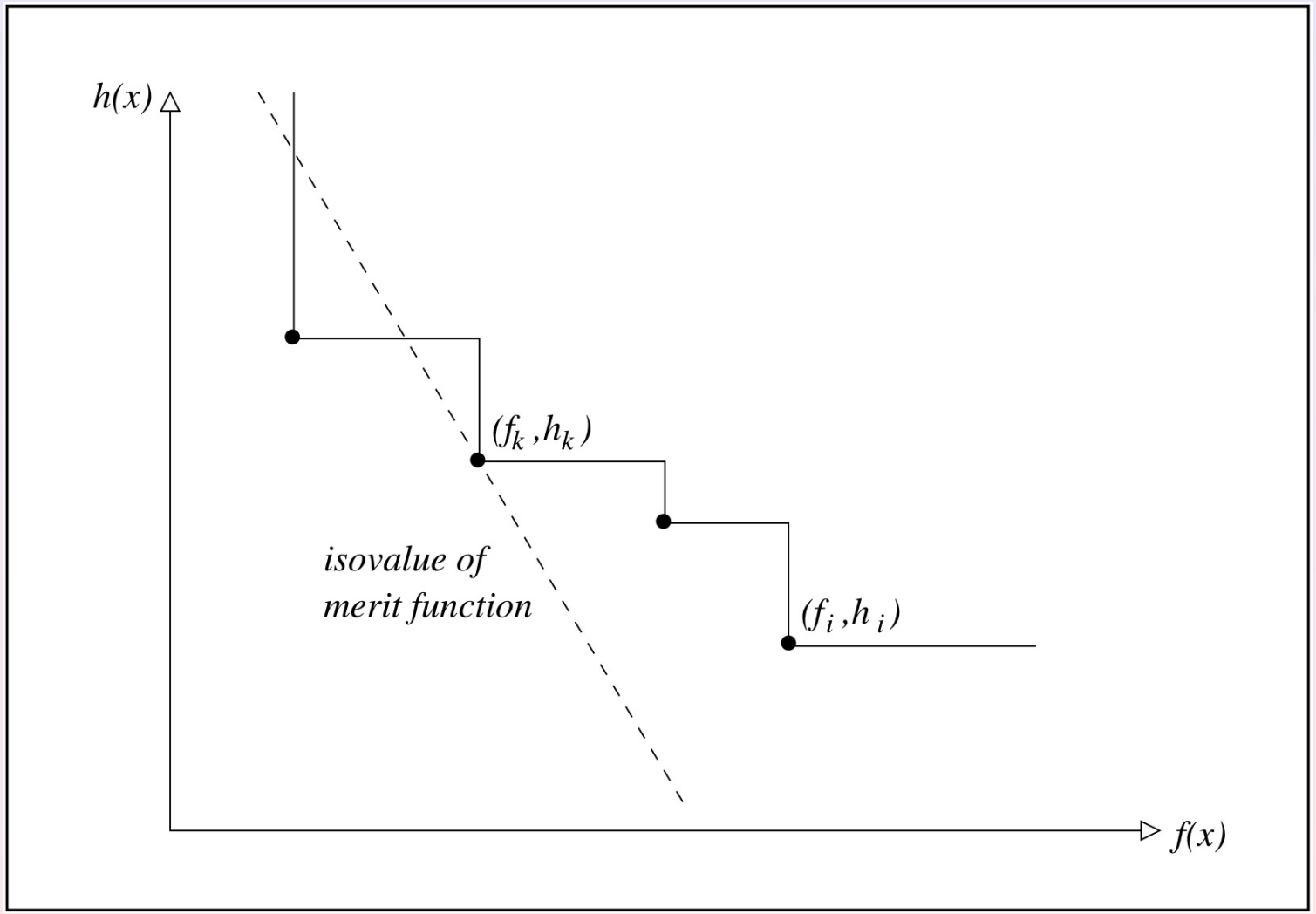

Figure 7: Comparing the filter and merit function techniques.

§15.4 Merit Functions and Filters

為了比較 filter 和 merit function 方法，我們在圖 7（下頁）中畫出滿足 $f+\mu h = f_k + \mu h_k$ 之函數值對 (f,h) 的等高線，其中 x_k 為當前迭代點。該線左側的區域對應於降低 merit function $\phi(x;\mu) = f(x) + \mu h(x)$ 的函數值對；顯然，這個集合與 filter 可接受的點的集合相當不同。

如果由 line search 方法生成的試探步驟 $x^+ = x_k + \alpha_k p_k$ 得到一個滿足 filter 的函數值對 (f^+, h^+)，我們設定 $x_{k+1} = x^+$；否則，就進行 backtracking line search。在 trust-region 方法中，如果步驟不符合 filter 的條件，信賴域將被縮小，並重新計算新的步驟。

§15.4 Merit Functions and Filters

為了獲得全域收斂和良好的實際性能，需要對這種 filter 技術進行一些修改以增強效果。首先，我們需要確保非常接近當前函數值對 (f_k, h_k) 或 filter 中的另一函數值對的 $(f(x^+), h(x^+))$ 不被接受。

我們可以通過修改可接受標準並要求函數值有足夠遞減量來實現這一點。如果有一固定 $\beta \in (0,1)$ 使得對於 filter 中的所有函數值對 (f_j, h_j)，試探步驟 x^+ 符合以下條件

$$f(x^+) \leqslant f_j - \beta h_j \qquad \text{or} \qquad h(x^+) \leqslant h_j - \beta h_j,$$

則我們接受該試探步驟為下一迭代點。上述試探步驟接受條件在實作中使用如 $\beta = 10^{-5}$ 這樣的值是有效的，但是為了進行分析我們常將第一個不等式替換為

$$f(x^+) \leqslant f_j - \beta h^+.$$

§15.4 Merit Functions and Filters

第二個增強措施是用來處理 filter 機制的一些奇怪的面向。在特定情況下，filter 得接受由 line search 方法中在搜尋方向上的任意小的步長 α_k，而這種現象可能導致算法停滯 (stall) 和失敗。為防範此情況，如果 backtracking line search 生成的步長小於給定閾值 $\alpha_{\min}$，則算法轉換為可行性修復階段 (feasibility restoration phase)。同樣地，在 trust-region 方法中，如果一系列試探步驟被 filter 拒絕，信賴域半徑可能被降低到使得信賴域子問題變得不可行（請參見 §18.5）。在這種情況下，也會啟動可行性修復階段。我們要強調的是，雖然還有其他機制可用於處理這種情況，但正如我們在稍後會討論到的，可行性修復階段有助於演算法實現其他有用的目標。

§15.4 Merit Functions and Filters

第二個增強措施是用來處理 filter 機制的一些奇怪的面向。在特定情況下，filter 得接受由 line search 方法中在搜尋方向上的任意小的步長 α_k，而這種現象可能導致算法停滯 (stall) 和失敗。為防範此情況，如果 backtracking line search 生成的步長小於給定閾值 $\alpha_{\min}$，則算法轉換為可行性修復階段 (feasibility restoration phase)。同樣地，在 trust-region 方法中，如果一系列試探步驟被 filter 拒絕，信賴域半徑可能被降低到使得信賴域子問題變得不可行（請參見 §18.5）。在這種情況下，也會啟動可行性修復階段。我們要強調的是，雖然還有其他機制可用於處理這種情況，但正如我們在稍後會討論到的，可行性修復階段有助於演算法實現其他有用的目標。

§15.4 Merit Functions and Filters

第二個增強措施是用來處理 filter 機制的一些奇怪的面向。在特定情況下，filter 得接受由 line search 方法中在搜尋方向上的任意小的步長 α_k，而這種現象可能導致算法停滯 (stall) 和失敗。為防範此情況，如果 backtracking line search 生成的步長小於給定閾值 $\alpha_{\min}$，則算法轉換為可行性修復階段 (feasibility restoration phase)。同樣地，在 trust-region 方法中，如果一系列試探步驟被 filter 拒絕，信賴域半徑可能被降低到使得信賴域子問題變得不可行（請參見 §18.5）。在這種情況下，也會啟動可行性修復階段。我們要強調的是，雖然還有其他機制可用於處理這種情況，但正如我們在稍後會討論到的，可行性修復階段有助於演算法實現其他有用的目標。

§15.4 Merit Functions and Filters

可行性修復階段的目標僅在於減少違反限制式的度量，即尋找問題的近似解

$$\min_x h(x)\,.$$

儘管

$$h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \left[c_i(x)\right]^-, \tag{21}$$

並非光滑函數，但我們在第 17 章中會展示如何使用光滑的受限優化子問題對其進行最小化。此階段在達到具有足夠小 h 值且與 filter 兼容的迭代時終止。

我們在下一頁提出一個針對迭代過程是採取 trust-region 方法生成的 filter 方法的框架。有關受限優化的信賴域方法的討論，請參見 §18.5。

§15.4 Merit Functions and Filters

可行性修復階段的目標僅在於減少違反限制式的度量，即尋找問題的近似解

$$\min_x h(x)\,.$$

儘管

$$h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \left[c_i(x)\right]^-, \tag{21}$$

並非光滑函數，但我們在第 17 章中會展示如何使用光滑的受限優化子問題對其進行最小化。此階段在達到具有足夠小 h 值且與 filter 兼容的迭代時終止。

我們在下一頁提出一個針對迭代過程是採取 trust-region 方法生成的 filter 方法的框架。有關受限優化的信賴域方法的討論，請參見 §18.5。

§15.4 Merit Functions and Filters

Algorithm 15.1 (General Filter Method)

Choose a starting point x_0 and an initial trust-region radius Δ_0;
Set $k \leftarrow 0$;
repeat until a convergence test is satisfied
  if the step-generation sub-problem is feasible
    Compute a trial iterate $x^+ = x_k + p_k$;
    if (f^+, h^+) is acceptable to the filter
      Set $x_{k+1} = x^+$ and add (f_{k+1}, h_{k+1}) to the filter;
      Choose Δ_{k+1} such that $\Delta_{k+1} \geqslant \Delta_k$;
      Remove all pairs from the filter that are dominated by (f_{k+1}, h_{k+1});
    else
      Reject the step, set $x_{k+1} = x_k$;
      Choose $\Delta_{k+1} < \Delta_k$;
    end if
    $k \leftarrow k + 1$;

§15.4 Merit Functions and Filters

if the step-generation sub-problem is feasible
  Compute a trial iterate $x^+ = x_k + p_k$;
  if (f^+, h^+) is acceptable to the filter
    Set $x_{k+1} = x^+$ and add (f_{k+1}, h_{k+1}) to the filter;
    Choose Δ_{k+1} such that $\Delta_{k+1} \geqslant \Delta_k$;
    Remove all pairs from the filter that are dominated by (f_{k+1}, h_{k+1});
  else
    Reject the step, set $x_{k+1} = x_k$;
    Choose $\Delta_{k+1} < \Delta_k$;
  end if
  $k \leftarrow k + 1$;
else
  Compute x_{k+1} using the feasibility restoration phase;
end if
end (repeat)

§15.4 Merit Functions and Filters

Algorithm 15.1 (General Filter Method)

Choose a starting point x_0 and an initial trust-region radius Δ_0;
Set $k \leftarrow 0$;
repeat until a convergence test is satisfied
  if the step-generation sub-problem is feasible
    Compute a trial iterate $x^+ = x_k + p_k$;
    if (f^+, h^+) is acceptable to the filter
      Set $x_{k+1} = x^+$ and add (f_{k+1}, h_{k+1}) to the filter;
      Choose Δ_{k+1} such that $\Delta_{k+1} \geqslant \Delta_k$;
      Remove all pairs from the filter that are dominated by (f_{k+1}, h_{k+1});
    else
      Reject the step, set $x_{k+1} = x_k$;
      Choose $\Delta_{k+1} < \Delta_k$;
    end if
    $k \leftarrow k + 1$;
  else
    Compute x_{k+1} using the feasibility restoration phase;
  end if
end (repeat)

§15.4 Merit Functions and Filters

Algorithm 15.1 (General Filter Method)

Choose a starting point x_0 and an initial trust-region radius Δ_0;
Set $k \leftarrow 0$;
repeat until a convergence test is satisfied
  if the step-generation sub-problem is infeasible
    Compute x_{k+1} using the feasibility restoration phase;
  else
    Compute a trial iterate $x^+ = x_k + p_k$;
    if (f^+, h^+) is acceptable to the filter
      Set $x_{k+1} = x^+$ and add (f_{k+1}, h_{k+1}) to the filter;
      Choose Δ_{k+1} such that $\Delta_{k+1} \geqslant \Delta_k$;
      Remove all pairs from the filter that are dominated by (f_{k+1}, h_{k+1});
    else
      Reject the step, set $x_{k+1} = x_k$;
      Choose $\Delta_{k+1} < \Delta_k$;
    end if
    $k \leftarrow k + 1$;
  end if
end (repeat)

§15.5 The Maratos Effect

某些基於 merit function 或 filter 的演算法可能因為它們不接受「對當前迭代點取得良好進展的步進量」而無法迅速收斂。這種不希望其發生的現象通常被稱為 Maratos 效應，因為它最初由 Maratos 在 [199] 的論文中觀察到。以下是一個例子，其中步進量 p_k 如果被接受的話將產生二次收斂，但接受的結果卻是導致目標函數值和限制式違反度量均增加。

Example

Consider the problem

$$\min f(x_1, x_2) = 2(x_1^2 + x_2^2 - 1) - x_1 \quad \text{subject to} \quad x_1^2 + x_2^2 = 1\,.$$

One can verify (see Figure 8) that the optimal solution is $x_* = (1, 0)^{\mathrm{T}}$, that the corresponding Lagrange multiplier is $\lambda_* = \frac{3}{2}$, and that $\nabla^2_{xx}\mathcal{L}(x_*, \lambda_*) = \mathrm{I}$.

§15.5 The Maratos Effect

某些基於 merit function 或 filter 的演算法可能因為它們不接受「對當前迭代點取得良好進展的步進量」而無法迅速收斂。這種不希望其發生的現象通常被稱為 Maratos 效應，因為它最初由 Maratos 在 [199] 的論文中觀察到。以下是一個例子，其中步進量 p_k 如果被接受的話將產生二次收斂，但接受的結果卻是導致目標函數值和限制式違反度量均增加。

Example

Consider the problem

$$\min f(x_1, x_2) = 2(x_1^2 + x_2^2 - 1) - x_1 \quad \text{subject to} \quad x_1^2 + x_2^2 = 1\,.$$

One can verify (see Figure 8) that the optimal solution is $x_* = (1, 0)^{\mathrm{T}}$, that the corresponding Lagrange multiplier is $\lambda_* = \frac{3}{2}$, and that $\nabla^2_{xx}\mathcal{L}(x_*, \lambda_*) = \mathrm{I}$.

§15.5 The Maratos Effect

Example (cont'd)

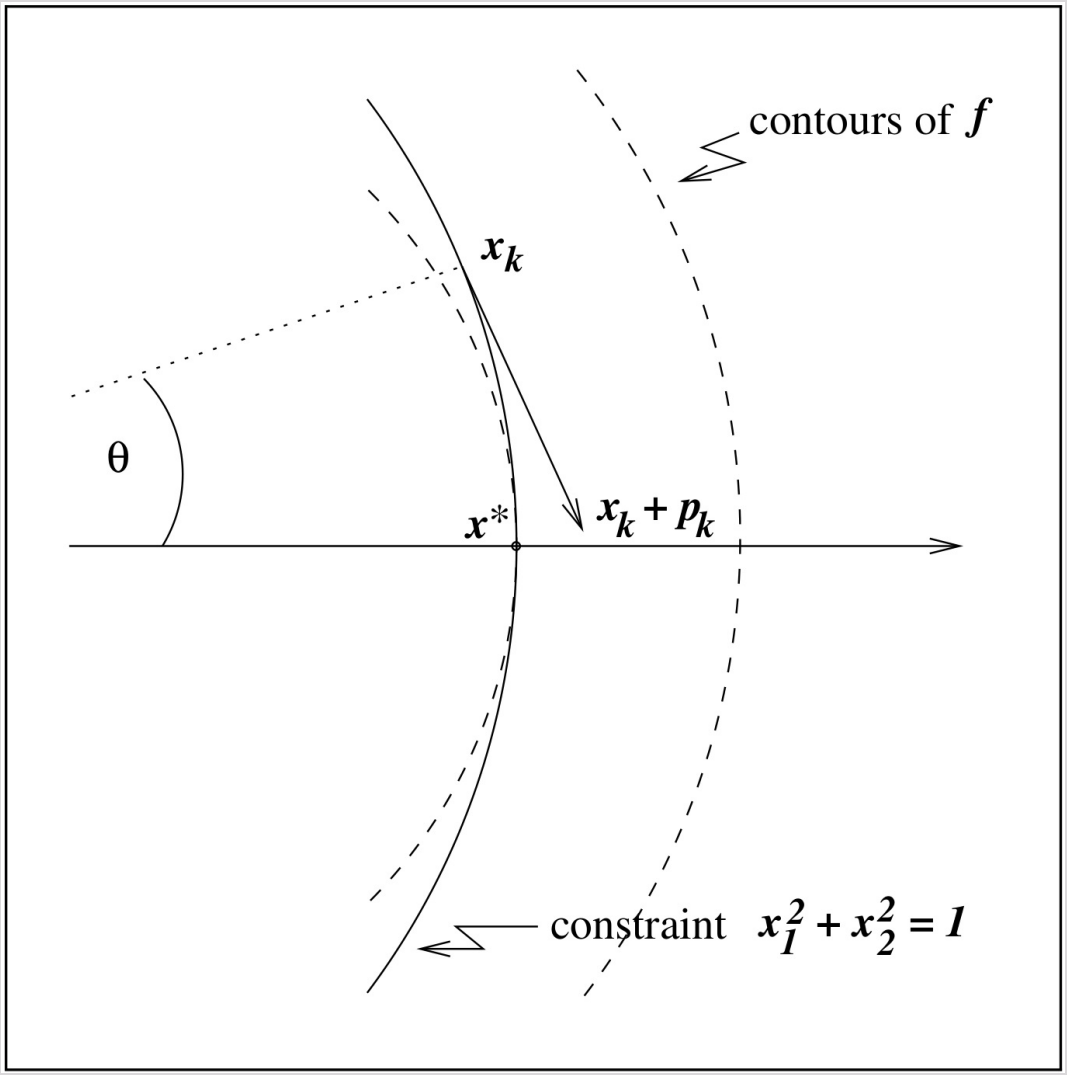

Figure 8: Maratos Effect: Note that the constraint is no longer satisfied after the step from x_k to $x_k + p_k$, and the objective value has increased.

§15.5 The Maratos Effect

Example (cont'd)

Let us consider an iterate x_k of the form $x_k = (\cos\theta, \sin\theta)^{\mathrm{T}}$, which is feasible for any value of θ. Suppose that our algorithm computes the following step:

$$p_k = \left(\sin^2\theta, -\sin\theta\cos\theta\right)^{\mathrm{T}}, \tag{23}$$

which yields a trial point

$$x_k + p_k = \left(\cos\theta + \sin^2\theta, \sin\theta(1-\cos\theta)\right)^{\mathrm{T}}.$$

By using elementary trigonometric identities, we have that

$$\|x_k + p_k - x_*\| = 2\sin^2(\theta/2), \quad \|x_k - x_*\| = 2\left|\sin(\theta/2)\right|,$$

and therefore

$$\frac{\|x_k + p_k - x_*\|}{\|x_k - x_*\|^2} = \frac{1}{2}.$$

§15.5 The Maratos Effect

Example (cont'd)

Hence, this step approaches the solution at a rate consistent with Q-quadratic convergence. However, we have that

$$f(x_k + p_k) = \sin 2\theta - \cos\theta > -\cos\theta = f(x_k),$$
$$c(x_k + p_k) = \sin^2\theta > c(x_k) = 0\,,$$

so that both the objective function value and the constraint violation increase over this step. This behavior occurs for any nonzero value of θ, even if the initial point is arbitrarily close to the solution.

在上述例子中，任何 filter 以及要求具備以下形式

$$\phi(x;\mu) = f(x) + \mu h(c(x))\,, \quad h \geqslant 0,\ h(0) = 0\,,$$

的 merit function 函數值下降的演算法，都會拒絕 (23) 所給的步進量（因為 $(f(x_k+p_k), h(x_k+p_k))$ 被 (f_k, h_k) 所 dominated）。因此，所有這些方法都將受到 Maratos 效應的影響。

§15.5 The Maratos Effect

Example (cont'd)

Hence, this step approaches the solution at a rate consistent with Q-quadratic convergence. However, we have that

$$f(x_k + p_k) = \sin 2\theta - \cos\theta > -\cos\theta = f(x_k),$$
$$c(x_k + p_k) = \sin^2\theta > c(x_k) = 0\,,$$

so that both the objective function value and the constraint violation increase over this step. This behavior occurs for any nonzero value of θ, even if the initial point is arbitrarily close to the solution.

在上述例子中，任何 filter 以及要求具備以下形式

$$\phi(x;\mu) = f(x) + \mu h(c(x))\,, \quad h \geqslant 0,\ h(0) = 0\,,$$

的 merit function 函數值下降的演算法，都會拒絕 (23) 所給的步進量（因為 $(f(x_k+p_k), h(x_k+p_k))$ 被 (f_k, h_k) 所 dominated）。因此，所有這些方法都將受到 Maratos 效應的影響。

§15.5 The Maratos Effect

如果不採取補救措施，Maratos 效應可能會干擾演算法在遠離解的時之良好步進量，並阻礙 superlinear 收斂。避免 Maratos 效應的策略包括以下幾點：

1. 我們可以使用一個不受 Maratos 效應影響的 merit function。一個例子是 Fletcher 的擴增 Lagrangian (17)。
2. 我們可以使用二階修正，在 p_k 中添加一步進量 $\hat{p}_k$，該步進量在 $c(x_k + p_k)$ 處計算，並減少限制違反。
3. 我們可以允許 merit function ϕ 在某些迭代中增加；換句話說，我們可以使用非單調策略。

我們將在下一節中討論後兩種方法。

§15.6 Second-Order Correction and Non-monotone Techniques

通過添加一個能降低限制式違反式度量的修正項，很多演算法能夠克服與 Maratos 效應相關的困難。以下我們在具等式限制的最佳化問題下描述這個技巧，在這裡限制式被簡寫成 $c(x) = 0$，其中 $c : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{E}|}$。

給定一個步進量 p_k，二階修正步進量 $\hat{p}_k$ 的定義為

$$\hat{p}_k = -A_k^{\mathrm{T}}(A_k A_k^{\mathrm{T}})^{-1} c(x_k + p_k)\,, \tag{24}$$

其中 $A_k = A(x_k)$ 是在點 x_k 處的限制函數 c 的 Jacobian matrix。注意到此定義下的 $\hat{p}_k$ 具有滿足限制函數 c 在點 $x_k + p_k$ 處的線性化的特性，亦即

$$A_k \hat{p}_k + c(x_k + p_k) = 0\,.$$

事實上，$\hat{p}_k$ 是上述方程的最小範數解。有關二階修正的另一種解釋可參見 §18.3。

§15.6 Second-Order Correction and Non-monotone Techniques

通過添加一個能降低限制式違反式度量的修正項，很多演算法能夠克服與 Maratos 效應相關的困難。以下我們在具等式限制的最佳化問題下描述這個技巧，在這裡限制式被簡寫成 $c(x) = 0$，其中 $c : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{E}|}$。

給定一個步進量 p_k，二階修正步進量 $\widehat{p}_k$ 的定義為

$$\widehat{p}_k = -A_k^{\mathrm{T}}(A_k A_k^{\mathrm{T}})^{-1} c(x_k + p_k)\,, \tag{24}$$

其中 $A_k = A(x_k)$ 是在點 x_k 處的限制函數 c 的 Jacobian matrix。注意到此定義下的 $\widehat{p}_k$ 具有滿足限制函數 c 在點 $x_k + p_k$ 處的線性化的特性，亦即

$$A_k \widehat{p}_k + c(x_k + p_k) = 0\,.$$

事實上，$\widehat{p}_k$ 是上述方程的最小範數解。有關二階修正的另一種解釋可參見 §18.3。

§15.6 Second-Order Correction and Non-monotone Techniques

修正步進量 $\hat{p}_k$ 的效果是將 $\|c(x)\|$ 降低到 $\|x_k - x_*\|^3$ 的程度，前提是主步進量 p_k 滿足 $A_k p_k + c(x_k) = 0$。這個估計表明，從 x_k 到 $x_k + p_k + \hat{p}_k$ 的步進量將至少在解附近降低 merit function 的函數值。這種增強的代價包括在 $x_k + p_k$ 處額外對限制函數 c 取值以及從 (24) 計算步驟 $\hat{p}_k$ 所需的計算量。

接下來我們描述一種使用 merit function、line-search 策略和二階修正步驟的演算法。我們假設使得 merit function 的下降方向之搜索方向 p_k 和懲罰參數 μ_k 已被計算出（這是第 18 和 19 章的主題之一）；即

$$D(\phi(x_k; \mu_k); p_k) < 0.$$

該演算法的關鍵特點是，如果步長 $\alpha_k = 1$ 未能使 merit function 的函數值產生滿意的下降量，我們會在沿著原始方向 p_k 回溯之前嘗試二階修正步驟。

§15.6 Second-Order Correction and Non-monotone Techniques

修正步進量 $\hat{p}_k$ 的效果是將 $\|c(x)\|$ 降低到 $\|x_k - x_*\|^3$ 的程度，前提是主步進量 p_k 滿足 $A_k p_k + c(x_k) = 0$。這個估計表明，從 x_k 到 $x_k + p_k + \hat{p}_k$ 的步進量將至少在解附近降低 merit function 的函數值。這種增強的代價包括在 $x_k + p_k$ 處額外對限制函數 c 取值以及從 (24) 計算步驟 $\hat{p}_k$ 所需的計算量。

接下來我們描述一種使用 merit function、line-search 策略和二階修正步驟的演算法。我們假設使得 merit function 的下降方向之搜索方向 p_k 和懲罰參數 μ_k 已被計算出（這是第 18 和 19 章的主題之一）；即

$$D(\phi(x_k; \mu_k); p_k) < 0.$$

該演算法的關鍵特點是，如果步長 $\alpha_k = 1$ 未能使 merit function 的函數值產生滿意的下降量，我們會在沿著原始方向 p_k 回溯之前嘗試二階修正步驟。

§15.6 Second-Order Correction and Non-monotone Techniques

Algorithm 15.2 (Generic Algorithm with Second-Order Correction).

Choose parameters $\eta \in (0, 0.5)$ and τ_1, τ_2 with $0 < \tau_1 < \tau_2 < 1$;
Choose initial point x_0; set $k \leftarrow 0$;
repeat until a convergence test is satisfied
    Compute a search direction p_k;
    Set $\alpha_k \leftarrow 1$, `newpoint` $\leftarrow$ `false`;
    while `newpoint` $=$ `false`
        if $\phi(x_k + \alpha_k p_k; \mu) \leqslant \phi(x_k; \mu) + \eta \alpha_k D(\phi(x_k; \mu); p_k)$
            Set $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
            Set `newpoint` $\leftarrow$ `true`;
        else if $\alpha_k = 1$
            Compute $\widehat{p}_k$ from (24);
            if $\phi(x_k + p_k + \widehat{p}_k; \mu) \leqslant \phi(x_k; \mu) + \eta D(\phi(x_k; \mu); p_k)$
                Set $x_{k+1} \leftarrow x_k + p_k + \widehat{p}_k$;
                Set `newpoint` $\leftarrow$ `true`;
            else

§15.6 Second-Order Correction and Non-monotone Techniques

while newpoint = false
  if $\phi(x_k + \alpha_k p_k; \mu) \leqslant \phi(x_k; \mu) + \eta \alpha_k D(\phi(x_k; \mu); p_k)$
    Set $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    Set newpoint $\leftarrow$ true;
  else if $\alpha_k = 1$
    Compute $\widehat{p}_k$ from (24);
    if $\phi(x_k + p_k + \widehat{p}_k; \mu) \leqslant \phi(x_k; \mu) + \eta D(\phi(x_k; \mu); p_k)$
      Set $x_{k+1} \leftarrow x_k + p_k + \widehat{p}_k$;
      Set newpoint $\leftarrow$ true;
    else
      Choose new α_k in $[\tau_1 \alpha_k, \tau_2 \alpha_k]$;
    end
  else
    Choose new α_k in $[\tau_1 \alpha_k, \tau_2 \alpha_k]$;
  end
end while (接下來是否要 $k \leftarrow k+1$？書上沒有這一步)
end repeat

§15.6 Second-Order Correction and Non-monotone Techniques

在這個演算法中，如果完整的 (full) 二階修正步進量 $\hat{p}_k$ 未能使 merit function 函數值減少，因為 $p_k + \hat{p}_k$ 不一定是 merit function 的下降方向，我們不會再沿著 $p_k + \hat{p}_k$ 這個方向繼續回溯而捨棄此修正步進量。這個演算法的一個變形是僅在不滿足函數值有足夠遞減條件

$$\phi(x + \alpha p; \mu) \leqslant \phi(x; \mu) + \eta \alpha D(\phi(x; \mu); p)\,, \tag{19}$$

的情況下才應用二階修正步進量，這是由於限制式的範數 $\|c(x)\|$ 增加而導致的。

在實作中，二階修正策略是有效的。對限制函數 c 進行額外的取值和 (24) 中的額外反解的成本被增加的穩健性和效率所抵銷。

§15.6 Second-Order Correction and Non-monotone Techniques

在這個演算法中，如果完整的 (full) 二階修正步進量 $\hat{p}_k$ 未能使 merit function 函數值減少，因為 $p_k + \hat{p}_k$ 不一定是 merit function 的下降方向，我們不會再沿著 $p_k + \hat{p}_k$ 這個方向繼續回溯而捨棄此修正步進量。這個演算法的一個變形是僅在不滿足函數值有足夠遞減條件

$$\phi(x + \alpha p; \mu) \leqslant \phi(x; \mu) + \eta \alpha D(\phi(x; \mu); p)\,, \tag{19}$$

的情況下才應用二階修正步進量，這是由於限制式的範數 $\|c(x)\|$ 增加而導致的。

在實作中，二階修正策略是有效的。對限制函數 c 進行額外的取值和 (24) 中的額外反解的成本被增加的穩健性和效率所抵銷。

§15.6 Second-Order Correction and Non-monotone Techniques

- **Non-monotine (watchdog) strategy**

Maratos 效應引起的低效率也可以通過偶爾接受「增加 merit function 函數值」的步進量來避免；這樣的步進量被稱為放鬆步進量 (relaxed step)。然而，我們對這種容忍是有限度的。如果在放鬆步進量的幾個迭代（比如說，$\hat{t}$ 個迭代）內並未使得 merit function 的函數值有足夠的遞減，則我們會返回到放鬆步進量之前的迭代，進行正常的迭代，使用 line search 或其他技術來強制降低 merit function 的函數值。

與目標僅在改善滿足限制式的二階修正不同，這種非單調策略總是採取演算法的正規步進量 p_k，其目標既在改善可行性同時又能優化。我們希望在單個步進量中 merit function 函數值的增加只是暫時的，而隨後的步進將補償這一增加量。

§15.6 Second-Order Correction and Non-monotone Techniques

接下來我們描述一種非單調方法的特定實例，稱為 "watchdog" 策略。在此策略中 $\hat{t}$ 被設定成 1，也就是說在 "watchdog" 策略下 merit function 的函數值有足夠遞減量的兩個迭代間 merit function 的函數值被允許增加一次。與前面一樣，我們將我們的討論集中在一種使用非光滑 merit function ϕ 的 line search 演算法上。

我們假設懲罰參數 μ 直到成功完成一個週期之前都不會更改。如此一來，在一個週期內我們可以忽略 ϕ 對 μ 的依賴而將 merit function 記為 $\phi(x)$，並將 merit function 沿 p_k 方向的方向導數記為 $D(\phi(x); p_k)$。

§15.6 Second-Order Correction and Non-monotone Techniques

接下來我們描述一種非單調方法的特定實例，稱為 "watchdog" 策略。在此策略中 $\hat{t}$ 被設定成 1，也就是說在 "watchdog" 策略下 merit function 的函數值有足夠遞減量的兩個迭代間 merit function 的函數值被允許增加一次。與前面一樣，我們將我們的討論集中在一種使用非光滑 merit function ϕ 的 line search 演算法上。

我們假設懲罰參數 μ 直到成功完成一個週期之前都不會更改。如此一來，在一個週期內我們可以忽略 ϕ 對 μ 的依賴而將 merit function 記為 $\phi(x)$，並將 merit function 沿 p_k 方向的方向導數記為 $D(\phi(x); p_k)$。

§15.6 Second-Order Correction and Non-monotone Techniques

Algorithm 15.3 (Watchdog).

Choose a constant $\eta \in (0, 0.5)$ and an initial point x_0;

Set $k \leftarrow 0$, $\mathcal{S} \leftarrow \{0\}$;

repeat until a convergence test is satisfied

Compute a step p_k;

Set $x_{k+1} \leftarrow x_k + p_k$;

if $\phi(x_{k+1}) \leqslant \phi(x_k) + \eta D(\phi(x_k); p_k)$

$k \leftarrow k + 1$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$;

else

Compute a search direction p_{k+1} from x_{k+1};

Find α_{k+1} such that

$$\phi(x_{k+1} + \alpha_{k+1} p_{k+1}) \leqslant \phi(x_{k+1}) + \eta \alpha_{k+1} D(\phi(x_{k+1}); p_{k+1});$$

Set $x_{k+2} \leftarrow x_{k+1} + \alpha_{k+1} p_{k+1}$;

if $\phi(x_{k+1}) \leqslant \phi(x_k)$ or $\phi(x_{k+2}) \leqslant \phi(x_k) + \eta D(\phi(x_k); p_k)$

§15.6 Second-Order Correction and Non-monotone Techniques

if $\phi(x_{k+1}) \leqslant \phi(x_k)$ or $\phi(x_{k+2}) \leqslant \phi(x_k) + \eta D(\phi(x_k); p_k)$
  $k \leftarrow k+2$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$;
else if $\phi(x_{k+2}) > \phi(x_k)$ ($\star$ return to x_k and search along p_k $\star$)
  Find α_k such that

$$\phi(x_k + \alpha_k p_k) \leqslant \phi(x_k) + \eta \alpha_k D(\phi(x_k); p_k);$$

  Compute $x_{k+3} = x_k + \alpha_k p_k$;
  $k \leftarrow k+3$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$;
else
  Compute a direction p_{k+2} from x_{k+2};
  Find α_{k+2} such that

$$\phi(x_{k+2} + \alpha_{k+2} p_{k+2}) \leqslant \phi(x_{k+2}) + \eta \alpha_{k+2} D(\phi(x_{k+2}); p_{k+2});$$

  Set $x_{k+3} \leftarrow x_{k+2} + \alpha_{k+2} p_{k+2}$;
  $k \leftarrow k+3$, $\mathcal{S} \leftarrow \mathcal{S} \cup k$;
end
end
end repeat

§15.6 Second-Order Correction and Non-monotone Techniques

該演算法並不需要集合 $\mathcal{S}$，$\mathcal{S}$ 只是用來標識已經獲得足夠的效能函數減少的迭代。注意，至少三分之一的迭代具有屬於 $\mathcal{S}$ 的索引。利用這一事實，可以證明使用 watchdog 技術的各種受限最佳化方法是全域收斂的。同樣可以證明，對於所有足夠大的 k，步長為 $\alpha_k = 1$，且收斂速度是 superlinear。

在實作中，允許 merit function 在多個迭代中增加可能是有利的。典型的 $\hat{t}$ 值可以選擇為 5 或 8。正如本討論所示，實現 watchdog 技術具有一定程度的複雜性，但由於這種方法具有良好的實際性能，所以增加的複雜性是值得的。Watchdog 技術相對於二階修正策略的一個潛在優勢是限制函數可能較少被取值。在最好的情況下，大多數步進將是 full step，很少需要返回到先前的點。

§15.6 Second-Order Correction and Non-monotone Techniques

該演算法並不需要集合 $\mathcal{S}$，$\mathcal{S}$ 只是用來標識已經獲得足夠的效能函數減少的迭代。注意，至少三分之一的迭代具有屬於 $\mathcal{S}$ 的索引。利用這一事實，可以證明使用 watchdog 技術的各種受限最佳化方法是全域收斂的。同樣可以證明，對於所有足夠大的 k，步長為 $\alpha_k = 1$，且收斂速度是 superlinear。

在實作中，允許 merit function 在多個迭代中增加可能是有利的。典型的 $\hat{t}$ 值可以選擇為 5 或 8。正如本討論所示，實現 watchdog 技術具有一定程度的複雜性，但由於這種方法具有良好的實際性能，所以增加的複雜性是值得的。Watchdog 技術相對於二階修正策略的一個潛在優勢是限制函數可能較少被取值。在最好的情況下，大多數步進將是 full step，很少需要返回到先前的點。