國立臺灣師範大學數學系碩士班碩士論文

指導教授：蔡碧紋 博士

超飽和設計因子篩選方法比較

研 究 生：謝逸安

中 華 民 國　一零六 年 一 月

誌謝

謝謝指導教授蔡碧紋教授兩年來專業細心的指導，很榮幸能成為您的學生，在此致上最深的謝意。

謝謝口試委員陳瑞彬教授、蔡蓉青教授在口試期間對論文提出許多指正與建議，使我的論文更加完整。

謝謝夥伴朱彥羽的討論，以及小助手陳奕璇的幫忙，讓我能順利的完成論文。

最後感謝家人與朋友們的支持與鼓勵。

謝逸安謹識於

國立臺灣師範大學數學系碩士班

中華民國一零六年一月

摘要

近年來有許多學者對大量因子的實驗設計感興趣，但是這類實驗可能僅有少數真正重要的因子，採用完全因子設計或傳統的部分因子設計來配置實驗可能不利於成本的控制，此時就可以考慮使用超飽和設計來篩選因子。

本文首先討論超飽和設計的矩陣建構方法，包含 Lin 和 Wu 分別提出的折半法和交互法，並進行比較。接下來介紹逐步向前法、最小絕對值壓縮選擇法 (LASSO) 以及 Dantzig selector 三種分析分法應用在超飽和設計的因子篩選上。最後在不同的模擬條件設定下，引入重要因子挑選比比較三種因子篩選方法在超飽和設計表現的優劣，並應用於實例之中。

關鍵字：超飽和設計、逐步向前法、最小絕對值壓縮選擇法 (LASSO)、Dantzig selector、重要因子挑選比

Abstract

In recent years, there are many scholars interested in experimental designs of a large number of factors, but such experiments may only have a few really important factors. It is detrimental to cost control if we using full factorial designs or traditional fractional factorial designs to construct these experiments, so we could consider using supersaturated designs in this situation to do factor screening.

In this article, we first discuss the design matrix construction methods of supersaturated designs, including the split-half method and the interaction method proposed by Lin and Wu, and then compare with them. Next, we introduce forward stepwise selection, LASSO and Dantzig selector as three factor selection methods of supersaturated designs. Finally, we compare the three factor selection methods of supersaturated designs by important factor selection ratio under different simulation conditions, and applied it to real examples.

Keywords: Supersaturated Designs, Forward Stepwise Selection, LASSO, Dantzig selector, Important Factor Selection Ratio

目錄

圖目錄

表目錄

1 緒論

在科技的進步之下，近年有許多學者開始對大量因子的實驗設計感興趣。但是由於成本的原因，此類實驗設計可能無法如傳統實驗設計有那麼多運行數 (Run size)，甚至會出現運行數小於因子數 (Number of factors) 的情形，我們稱運行數小於因子數的實驗設計為超飽和設計 (Supersaturated designs)。

超飽和設計的理念起源於 Satterthwaite(1959) ，他提出隨機平衡設計 (Random balanced designs) 的想法建構超飽和設計，而 Booth 和 Cox(1962) 則首先利用電腦程式，有系統地建構出七個超飽和設計。然而當時除了複雜的電腦演算，並沒有發現可以有效率建構出超飽和設計的方法，所以超飽和設計在之後的三十年間沉寂了下來，直到 Lin(1993) 和 Wu(1993) 以 Hadamard 矩陣為基礎，可以快速且一般性的建構出超飽和設計的設計矩陣，超飽和設計才又開始被學者們重視。時至今日，超飽和設計被廣泛應用在機器學習、基因工程、工業製造以及巨量資料等領域中，超飽和設計已是實驗設計領域中備受矚目的議題，許多相關的研究在過去二十年來相繼出現，例如 Lin(1993) 、Wu(1993) 、Nguyen(1996) 、Li 和 Wu(1997) 、Cheng(1997) 。

實驗設計中，水準 (Levels) 是指每個因子的可能選項，有可能是量化的 (Quantitative) 選項，如不同的溫度、不同的壓力等等；也可能是質性的 (Qualitative) 選項，如不同的顏色、不同的結構等等。兩水準因子設計 (Two-level factorial designs) 代表一個因子中只會有兩個不同的選項，如在溫度這個因子中設定攝氏 100 度與攝氏 40 度來配置實驗後，不會再有其他溫度的可能性。因子被設定的兩個水準以符號 $+1$ 與 -1 表示，我們可以將攝氏 100 度時的溫度因子以 $+1$ 表示，攝氏 40 度時的溫度因子以 -1 表示，以 $+1$ 表示與以 -1 表示的情形是可以互換的。本文之後皆以兩水準因子設計作為主要討論對象，其原因是探討超飽和設計時，成本經常有所限制，若再增加因子水準的數目，將不利於成本的控制。

在情況允許之下，會使用完全因子設計 (Full factorial designs) 來配置實驗。完全因子設計就是在實驗中用上了因子水準全部組合的設計，當因子數為 p 時，

兩水準完全因子設計所需要的運行數為 2^p，隨著因子數增加，所需運行數也將會快速增加，所以在因子數量較多的情況之下，使用完全因子設計是有些不切實際的，會導致經濟成本或時間成本成指數型增加。而部分因子設計 (Fractional factorial designs) 是僅執行完全因子設計中的部分實驗的設計，當資源有限或因子數很大時，部分因子設計是很好的選擇，因為部分因子設計使用了更少的運行數，超飽和設計就是部分因子設計的一種，特別指運行數比因子數少的部分因子設計。部分因子設計中，因為某些因子的主效應 (Main effects) 會與其它因子間的交互作用產生效應混淆 (Aliased)，造成無法有效區分效應來源。Box 和 Meyer(1986) 提出效應稀疏原則 (Sparsity-of-effects principle)，說明實驗通常是由因子的主效應與低階交互作用主導，高階的交互作用是非常罕見的，故通常假設高階的交互作用不存在，以便能以較少運行數取得主效應與低階交互作用的資訊。超飽和設計最主要的分析方式是因子篩選 (Factor selection)，是因為超飽和設計中因子的主效應個數本身就已多於運行數，考慮二階及以上交互作用容易因混淆情形影響分析結果。另外，因為超飽和設計中某些因子並不正交 (Non-orthogonality) 的關係，不足的運行數也可能導致選取重要因子時，同時選取到其它混淆程度較高的因子，造成錯誤選取重要因子的情況，此亦為超飽和設計的困境。為了避免以少量運行數估計較多因子參數，超飽和設計通常會進行因子稀疏性的假設。假設稀疏性的統計模型只有少部分因子扮演重要角色，其餘大部分因子的效應都可以忽略不計，亦即模型中只會有少數不為零的參數。

本文第二章介紹折半法與交互法兩種超飽和設計的設計矩陣建構方法，並進行討論與比較。第三章介紹三種分析分法應用在超飽和設計的因子篩選上，分別為逐步向前法 (Forward stepwise selection)、最小絕對值壓縮選擇法 (Least absolute shrinkage and seletion operator，LASSO) 以及 Dantzig selector。第四章為挑選比分析，引入重要因子挑選比，利用不同設定之下模擬的結果比較三種因子篩選方法表現的優劣。第五章為結論，對前四章的結果進行總結。

2 超飽和設計建構方法

考慮常態假設的線性迴歸模型：

$$y = X\beta + \epsilon \tag{2.1}$$

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} \quad \epsilon \sim N(0, \sigma^2 I_n)$$

y 為觀測值 (Observations)，X 為 $n \times p$ 的設計矩陣 (Design matrix)，β 為因子的參數 (Parameters)，ϵ 為誤差項 (Error terms)，p 為因子數 (Number of factors)，n 為運行數 (Run size)。超飽和設計就是指因子數大於運行數的實驗配置，若 (2.1) 式為超飽和設計模型，其設計矩陣 X 的行數 (Number of columns) 會多於其列數 (Number of rows)，即 $p > n$ 的情形。

超飽和設計最主要的課題之一就是設計矩陣的建構，本章接下來探討兩種可以一般性的建構出兩水準因子的超飽和設計的設計矩陣方法。兩種矩陣設計方法都是以 Hadamard 矩陣作為開端，Hadamard 矩陣為一方陣，其特性如下：

1 矩陣中每個元素都是 $+1$ 或 -1。

2 矩陣中有一行全部都是 $+1$(或都是 -1)，其餘每一行 $+1$ 與 -1 的數量各半。

3 矩陣中任意兩行互相正交。

若以 H 表示 Hadamard 矩陣，m 表示其方陣階數，則 Hadamard 矩陣會滿足 $HH^T = mI_m$，其中 I_m 為 m 階單位矩陣 (Identity matrix)，同階下 Hadamard 矩陣並不唯一。

若 p 為大於 2 的正整數，可以找到 $2^p \times p$ 的直交表配置因子數為 p 的 2^p 完全因子設計實驗，若不想執行如此多運行數，一個常見的方法是使用 2^{p-l} 部

分因子設計，其中 l 為生成子 (Generators)。事實上 2^{p-l} 部分因子設計就是 2^p 完全因子設計取出其中某些列來執行，有些學者稱這類部分因子設計為正規設計 (Regular designs)，用以區別不是用這種方法產生的部分因子設計。而在設計矩陣中，若其中兩行符號完全相同 (或相反)，則稱此兩行完全混淆 (Fully aliased)；若其中兩行沒有完全混淆且不正交，則稱此兩行部分混淆 (Partially aliased)。某些 Hadamard 矩陣在去掉都是 $+1$(或都是 -1) 的行後，會是一個正規設計，如表 2.1的 8 階 Hadamard 矩陣，這類 Hadamard 矩陣在使用本章的兩個超飽和設計矩陣建構法時，會因因子之間完全混淆的緣故，不能形成因子數比運行數多的設計矩陣，無法用於超飽和設計。現今的實驗設計中，經常使用 Plackett 和 Burman(1946) 設計 (Plackett–Burman designs) 所生成的 Hadamard 矩陣，這種利用生成列向量 (Generating row vectors) 的循環產生的 Hadamard 矩陣為不正規設計，因建構出的超飽和設計的設計矩陣只有部分混淆，能夠運用在超飽和設計中，大部分階數為 4 的倍數的 Hadamard 矩陣皆可以用此方式生成。

表 2.1: 8 階 Hadamard 矩陣 (正規設計)

$$\begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \end{pmatrix}$$

以下介紹 Lin(1993) 和 Wu(1993) 如何以 Hadamard 矩陣建構出超飽和設計的設計矩陣。

2.1 折半法

Lin(1993) 提出將 Hadamard 矩陣折半來產生超飽和設計的設計矩陣，本文稱此法為折半法，折半法可以建構出運行數為 2 的倍數的超飽和設計矩陣。首先將 Hadamard 矩陣中都是 $+1$(或都是 -1) 的行去掉，在餘下的行中任取一行當作分支行 (Branching column)，得出分支行中元素為 $+1$(或 -1) 的列數，取出這些列形成

新的矩陣，再去掉分支行。最後檢查新的矩陣是否有完全混淆的行，有則將其去掉。若最後沒有完全混淆的行，則 m 階 Hadamard 矩陣，可以利用折半法建構出 $\frac{m}{2}$ 列 $(m-2)$ 行的設計矩陣，此矩陣可用於因子數為 $(m-2)$，運行數為 $\frac{m}{2}$ 的超飽和設計。

以 12 階 Plackett 和 Burman 設計的 Hadamard 矩陣為例，如表 2.2，首先將全部都是 $+1$ 的第一行刪除，再以第二行作為分支行取其 $+1$ 部分，為原矩陣中第 1、4、8、9、10、12 列。將原矩陣這些列提取出來並刪除分支行，可得 6 列 10 行的新矩陣，如表 2.3。新矩陣並沒有完全混淆的行可以刪除，故可用於因子數為 10，運行數為 6 的超飽和設計。

表 2.2: 12 階 Hadamard 矩陣 (Plackett 和 Burman 設計)

$$\begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 \\ +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 \\ +1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 \\ +1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 \\ +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 \\ +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 \end{pmatrix}$$

表 2.3: 折半法範例：形成 6×10 的設計矩陣

$$\begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 \\ +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 \\ -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\ -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 \end{pmatrix}$$

2.2 交互法

Wu(1993) 提出將 Hadamard 矩陣加入部分混淆交互作用 (Partially aliased interactions) 來產生超飽和設計的設計矩陣，本文稱為交互法，交互法可以建構出運行數為 4 的倍數的超飽和設計矩陣。首先將 Hadamard 矩陣中都是 $+1$(或都是 -1) 的行去掉，再取餘下的行中行與行的交互作用形成新的行。將行數增加到設定的因子數，最後檢查新的矩陣是否有完全混淆的行，有則將其去掉。

m 階 Hadamard 矩陣去掉都是 $+1$(或都是 -1) 的行後，本文將餘下的 $(m-1)$ 行稱為交互前行，也有一些文獻將交互前行稱為基向量 (Base vectors)。將交互前行中兩兩進行交互作用，會得到 $\binom{m-1}{2}$ 行，本文將交互作用得出的新行稱為交互後行。交互前行加上交互後行，若沒有完全混淆的行，則 m 階 Hadamard 矩陣可以利用交互法建構出 m 列 $\binom{m}{2}$ 行的設計矩陣，此矩陣可用於最多因子數為 $\binom{m}{2}$，運行數為 m 的超飽和設計。但交互法中，交互後行並不見得都要用盡，所以此方法建構出的超飽和設計，因子數較具彈性。若我們有 p 個設計因子，其中 $p > m$，則交互前行可觀測 $(m-1)$ 個設計因子，剩下的 $(p-m+1)$ 個設計因子可以從交互後行挑出。挑出的方法可以採用固定順序挑選或隨機挑選，固定順序挑選就是從 $\binom{m-1}{2}$ 行交互後行中挑選前 $(p-m+1)$ 行來觀測剩下的設計因子，而隨機挑選顧名思義是從 $\binom{m-1}{2}$ 行交互後行隨機挑選 $(p-m+1)$ 行來觀測剩下的設計因子。因為交互後行的行數往往比運行數多出許多的關係，隨機挑選不可避免地經常挑到許多混淆程度很高的行，相較之下採取固定順序挑選產生的設計矩陣行與行間能保持更高的正交性，故本文的交互法皆採用固定順序挑選挑出剩下的設計因子。

同樣以表 2.2的 12 階 Plackett 和 Burman 設計的 Hadamard 矩陣為例，首先將全部都是 $+1$ 的第一行刪除，得出 11 行交互前行，如表 2.4中的 $PartA$，再將交互前行中的第一行與第二行進行交互作用，得出交互後行的第一行，將交互前行中的第一行與第三行進行交互作用，得出交互後行的第二行，以此類推，共得出 55 行交互後行，形成表 2.4中 $PartB$ 的部分。最後可得 12 列 66 行的新矩陣，如表 2.4的所有部分。新矩陣並沒有完全混淆的行，故可用於因子數為 66、運行數為 12 的超飽和設計。若所需的設計因子數少於 66，除了 11 行交互前行必選外，

餘下的因子由 55 行交互後行的前面行補齊。

表 2.4: 交互法範例：形成 12×66 的設計矩陣

$$
\begin{array}{c}
\quad\overbrace{\hspace{20em}}^{PartA}\quad\overbrace{\hspace{9em}}^{PartB}\\
\left(\begin{array}{ccccccccccc|ccccc}
+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & & +1\\
-1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & +1 & -1 & +1 & & -1\\
-1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & +1 & & -1\\
+1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 & & +1\\
-1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & +1 & +1 & -1 & & -1\\
-1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & -1 & & -1\\
-1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 & \cdots & +1\\
+1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & & +1\\
+1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 & & -1\\
+1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & & +1\\
-1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & +1 & & +1\\
+1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 & +1 & & -1
\end{array}\right)
\end{array}
$$

2.3 建構方法的比較

本章介紹的折半法與交互法，都是目前最常見的超飽和設計的設計矩陣建構方法，兩種方法也各有其使用時機。在固定運行數的情況下，折半法產生的設計矩陣，本文稱為折半法標準矩陣，折半法標準矩陣擁有特定的行數，若實驗並沒有這麼多因子要觀測，則必須刪除其中某幾行，理論上刪除混淆程度低的行較佳。但若要刪除的行有一定數目時，計算每一行行與行間的混淆程度再予以刪除其實並不容易，所以比較直接的方式是採用隨機刪行，只是隨機刪行可能導致設計矩陣的正交性快速下降。所以使用折半法設計時，對於因子數的限制較高，往往因子數與運行數需要兼顧，不能與折半法標準矩陣差距太大。

交互法的優勢就是折半法的劣勢，固定運行數的情況下，交互法對因子數的限制較小，即此方法產生的設計矩陣彈性較大。有些沒有辦法利用折半法建構的超飽和設計，可以利用交互法來建構。例如運行數為 n 的超飽和設計，若沒有完全混淆的行，折半法最多僅能觀測 $(2n-2)$ 個因子，但是交互法最多可以觀測到 $\binom{n}{2}$ 個因子，當 n 越大時兩方法的最大觀測因子數差距會越大。利用折半法建構的設計矩陣不能與折半法標準矩陣差距太大，但是交互法就無此限制，交互後

行的運用相當有彈性，設計矩陣的正交性也不會因為沒有用完交互後行而快速下降。交互法因為其特性，也常運用在重要與不重要的因子明顯分成兩群時，交互前行因正交性較高，可以用來觀測重要因子，確保重要因子不會與其他不重要因子混淆，而將不重要因子擺在較為彈性的交互後行。但交互法的劣勢也相當明顯，就是無法用在觀察因子之間的交互作用上，因為交互後行正是交互前行的交互作用所產生。所以當各因子重要性不明顯時，或需要探討因子之間的交互作用時，用折半法產生的設計矩陣會更有優勢。

超飽和設計的設計矩陣因子之間不會完全正交，必有部分混淆的情形發生，但是混淆會干擾分析的結果，好的設計要儘量降低混淆情形的出現，亦即設計的因子之間保持較高的正交性，故正交性越好的設計矩陣代表越佳的設計。但是如何能判斷設計矩陣的正交性？最常使用的判斷準則是由 Booth 和 Cox(1962) 提出的 $E(s^2)$。若 X 是 $n \times p$ 的設計矩陣，則 X 的 $E(s^2)$ 是將 X 任意兩行內積平方和取平均，其定義如下：

$$E(s^2) = \sum_{i<j} \frac{s_{ij}^2}{\binom{p}{2}} \tag{2.2}$$

其中 $s_{ij} = c_i^T c_j$，c_i 與 c_j 分別代表設計矩陣的第 i 行及第 j 行。其實 X 的 $E(s^2)$ 就是 $X^T X$ 去掉主對角線後，每一個元素平方相加，最後取平均，$E(s^2)$ 越小表示混淆程度越低，即設計矩陣的正交性越好。以本文中兩種超飽和設計的設計方法在固定運行數 n 為 120 之下，分別在六個不同因子數 p 的情況下建構設計矩陣並求其 $E(s^2)$，結果如表 2.5。因為兩種方法建構的設計矩陣都不唯一，所以表 2.5的 $E(s^2)$ 平均是兩種方法在六個情況各建構 100 次平均的結果。使用折半法可以形成 120×238 的折半法標準矩陣，在因子數與折半法標準矩陣的行數相差不大的情況下，兩種方法的設計矩陣 $E(s^2)$ 平均相當接近，但隨著因子數的減少，交互法所建構的設計矩陣，其 $E(s^2)$ 平均會小於使用折半法建構的設計矩陣，即此時使用交互法建構的設計矩陣，其正交性要好於以折半法所建構，且因子數越小差距越大。而在其他不同運行數的情況下也有類似的結果，所以若因子數與折半法標準矩陣的行數相差過大，則只建議使用交互法生成設計矩陣；反之若因子數與折半法標準矩陣的行數相差不大時，兩種生成超飽和設計的方法都同樣出色，此時只需依照方便的情況挑選設計方法即可。

另外，本章在介紹交互法時曾提到，在交互後行挑選剩下的因子時，使用固定順序挑選會比隨機挑選保持更好的正交性，所以表 2.5的交互法是採用固定順序挑選剩下的因子。而表 2.6顯示若交互法改成使用隨機挑選剩下的因子時，在固定運行數 n 為 120 與同樣的六個不同因子數 p 的情況下各建構 100 次設計矩陣的 $E(s^2)$ 平均。比較表 2.5與表 2.6，可知在所有不同因子數的情況下，隨機挑選的 $E(s^2)$ 平均都會比固定順序挑選的 $E(s^2)$ 平均大上許多，即固定順序挑選的正交性明顯好於隨機挑選，這也是本文使用交互法時，皆採用固定順序挑選挑出剩下因子的原因。

表 2.5: $n = 120$，p 不同之下折半法與交互法的 $E(s^2)$ 平均比較 (100 次)

$E(s^2)$ 平均	$p = 238$	$p = 235$	$p = 230$	$p = 220$	$p = 200$	$p = 150$
折半法	60.504	60.497	60.479	60.494	60.372	60.381
交互法	60.894	60.494	60.431	60.099	58.320	39.680

表 2.6: $n = 120$，p 不同之下採用隨機挑選的交互法的 $E(s^2)$ 平均 (100 次)

$E(s^2)$ 平均	$p = 238$	$p = 235$	$p = 230$	$p = 220$	$p = 200$	$p = 150$
交互法 (隨機挑選)	89.724	88.975	87.683	84.740	77.461	44.562

3 超飽和設計因子篩選方法

自 Lin(1993) 和 Wu(1993) 重新喚起學者們對於超飽和設計的重視後，近年來發展出許多超飽和設計的分析方法。超飽和設計最普遍的應用就是進行因子篩選 (Factor selection)，本文也以因子篩選作為分析超飽和設計的主軸。Lin 本人是以逐步向前法 (Forward stepwise selection) 進行因子篩選挑出顯著因子，此後 Chipman et al.(1997)、Westfall et al.(1998)、Beattie et al.(2002)、Li 和 Lin(2002)、Fang et al.(2003)、Lu 和 Wu(2004)、Zhang et al.(2007)、Candes 和 Tao(2007) 、Phoa et al.(2009) 都有提出或分析超飽和設計的因子篩選方法。

本章接下來介紹和比較的超飽和設計的因子篩選方法，有傳統上統計分析經常使用的逐步向前法、由 Tibshirani(1996) 提出的最小絕對值壓縮選擇法 (Least absolute shrinkage and seletion operator，LASSO)，以及由 Candes 和 Tao(2007) 提出的 Dantzig selector ，以下介紹這三種因子篩選方法。

3.1 逐步向前法

逐步迴歸 (Stepwise regression) 是複迴歸分析中挑選變數和模型的一種方法，也是實驗設計中最常見的因子篩選方法之一。逐步迴歸有分成向前及向後兩個方向，因為超飽和設計因子數大於運行數，方向向後會有自由度的問題而無法進行因子篩選，所以本文逐步迴歸皆採用方向向前，簡稱為逐步向前法。

逐步向前法是以 t 檢驗 (Student's t-test) 的 t 值作為是否挑選因子的指標，如果某個因子的 $|t|$ 觀測值大於其他因子的 $|t|$ 觀測值，且大於設定的顯著水準 α 理論 $|t|$ 值，則將此因子加入模型。之後再對剩下的因子執行同樣步驟，進行因子篩選，直到選取到所設定的因子數，或最大的 $|t|$ 觀測值小於理論 $|t|$ 值為止。有時也會採用赤池訊息準則 (Akaike information criterion，AIC) 或貝氏訊息準則 (Bayesian information criterion，BIC)，協助進行因子篩選。逐步向前法原理直觀且在大部分情況下表現不錯，經常用於迴歸分析或實驗設計中挑選變數或因子上。

但是在某些因子數遠大於運行數的超飽和設計中，因為此時因子間混淆程度較為嚴重的緣故，逐步向前法的表現可能就沒有其它方法來的好。另外在資訊量較大時，逐步向前法因計算的限制，也會比其他方法需要更多的時間，運用在需要及時運算的情況下會有挑戰。

3.2 最小絕對值壓縮選擇法 (LASSO)

當要估計一個迴歸模型的參數，最常見的方法是使用最小平方法。考慮 (2.1) 式的模型，若要以最小平方法估計 β，就要找到使得 $||y - X\beta||^2$ 最小的 β。Tibshirani 提出的最小絕對值壓縮選擇法 (以下簡稱 LASSO) 就是在 L_1 範數 (L_1-norm) 式中，加入最小平方法的條件，使 LASSO 的 β 估計 $\hat{\beta}_{LA}$ 如下：

$$\hat{\beta}_{LA} = \underset{\beta}{\operatorname{argmin}} ||y - X\beta||^2 \quad s.t. \quad ||\beta||_1 \leq \delta \tag{3.1}$$

其中，L_1 範數 $||\beta||_1 = \sum_{i=1}^{p} |\beta_i|$，$\delta$ 為調整參數 (Tuning parameters)，不同的調整參數會影響估計，使得挑選的因子或挑選因子的數目會有些許不同。由 (3.1) 式可知當 δ 超過最小平方法 β 估計的 L_1 範數 $||\hat{\beta}_{LSE}||_1$ 時，$\hat{\beta}_{LA}$ 就是最小平方法的 β 估計。當 δ 從 $||\hat{\beta}_{LSE}||_1$ 減少，$\hat{\beta}_{LA}$ 中某些參數開始強迫為 0，δ 越小則 $\hat{\beta}_{LA}$ 中為 0 的參數越多，在 δ 減少的過程中，參數越慢降為 0 的因子越為重要，因此可以進行因子篩選。但要快速解出在不同的 δ 之下的 $\hat{\beta}_{LA}$ 其實並不容易，所以直到 Efron et al.(2004) 提出最小角度迴歸 (Least-angle regression，LARS)，此演算法大幅改進 LASSO 的運算速度，才使 LASSO 成為受歡迎的因子篩選方法之一。

為了解決可能存在多解的問題而引入額外的條件，稱為正規化 (Regularization) 的過程。如在 (3.1) 式中，$||\beta||_1 \leq \delta$ 在 δ 不為零時 β 存在無限多組解，但是若加入 $\underset{\beta}{\operatorname{argmin}} ||y - X\beta||^2$ 的條件，β 會只剩下一組解。因為 $||\beta||_1$ 是 L_1 範數，故也稱 LASSO 為 L_1 範數的正規化問題，或簡稱為 L_1 正規化，L_0 正規化與 L_2 正規化同理。事實上就因子篩選的觀點，最直觀的方法是使用 L_0 正規化，亦即使用 L_0 範數式 $||\beta||_0 = \sum_{i=1}^{p} I_{\{\beta_i \neq 0\}} \leq \delta$ 作為目標。為了方便說明，這裡以兩參數模型為例，如圖 3.1。圖 3.1(a) 表示兩參數模型的 L_0 正規化，右上方的點 $\hat{\beta}_{LSE}$ 為兩參

數的最小平方法估計值，因為圖中虛線區域為 L_0 範數式，解被限制在虛線區域上，故以 $\hat{\beta}_{LSE}$ 為中心的 $||y - X\beta||^2$ 圖形會鬆弛 (Relax) 成橢圓形軌跡。將 L_0 正規化的 β 估計以 $\hat{\beta}_{L_0}$ 表示，其中 $\hat{\beta}_{L_0} = (\hat{\beta}_{1L_0}, \hat{\beta}_{2L_0})$，$\hat{\beta}_{L_0}$ 即為鬆弛後的橢圓形軌跡與虛線區域的交點。在圖 3.1(a) 中只有 $\hat{\beta}_{1L_0}$ 為 0，代表此時參數為 β_1 的因子不顯著，所以可以進行因子篩選。因為使用 L_0 範數式，L_0 正規化篩選因子非常直觀而且明確，理論上 L_0 正規化是最適合進行因子篩選的方法，可惜因為 L_0 正規化為非凸問題 (Non-convex problems)，計算相當困難。Natarajan(1995) 說明此為一個 NP-hard 問題，事實上當因子數大於 40 就無法求解，所以現實中 L_0 正規化無法有效利用在因子篩選上。

脊迴歸 (Ridge regression) 為使用 L_2 範數式 $||\beta||_2 = \sqrt{\sum_{i=1}^{p} \beta_i^2} \leq \delta$ 作為目標的 L_2 正規化，又稱為吉洪諾夫正規化 (Tikhonov regularization)。由於此為嚴格凸問題 (Convex problems) 的緣故，與 L_0 正規化及 LASSO 相較之下，脊迴歸計算相當快速。相對於最小平方法，脊迴歸雖然允許少許的偏誤 (bias) 存在，但其估計會有更小的變異數。圖 3.1(c) 表示兩參數模型的脊迴歸，脊迴歸的 β 估計以 $\hat{\beta}_{RG}$ 表示，其中 $\hat{\beta}_{RG} = (\hat{\beta}_{1RG}, \hat{\beta}_{2RG})$，在圖中 L_2 範數式為虛線部分圍成的圓型區域，其與鬆弛後的橢圓形軌跡的交點 $\hat{\beta}_{RG}$ 坐落於第一象限中，因 $\hat{\beta}_{1RG}$ 與 $\hat{\beta}_{2RG}$ 皆不為 0，無法進行因子篩選。事實上由於 L_2 範數式沒有尖點的特性，脊迴歸只能用於參數估計，並不能進行因子篩選。而圖 3.1(b) 表示兩參數模型的 LASSO，$\hat{\beta}_{LA} = (\hat{\beta}_{1LA}, \hat{\beta}_{2LA})$，在圖中只有 $\hat{\beta}_{1LA}$ 為 0，所以這裡 LASSO 如同 L_0 正規化能夠進行因子篩選，因為遠較 L_0 正規化有效率解決問題，所以 LASSO 常用來近似 L_0 正規化的結果。LASSO 如今被廣泛地運用在因子篩選上，主要原因之一就是其妥協性質，因子篩選只會發生在範數式有尖點時，即 L_p 範數式中 $p \leq 1$ 的情況，但相對也只有在 $p \geq 1$ 時 L_p 正規化為凸問題，可以使用凸最佳化 (Convex optimization) 的方式計算。LASSO 亦有其缺點，如因子篩選準確度不如 L_0 正規化，可能嘗試縮減係數太過，導致出現許多偽因子；而計算速度也不及脊迴歸，因 LASSO 並非嚴格凸問題，所以仍有可能存在不同的解。在因子篩選中，LASSO 是不得已但也是最佳的選擇，近年亦有一些新興的方法，試圖以 L_1 正規化的變形改進因子篩選的結果，例如 Dantzig selector。

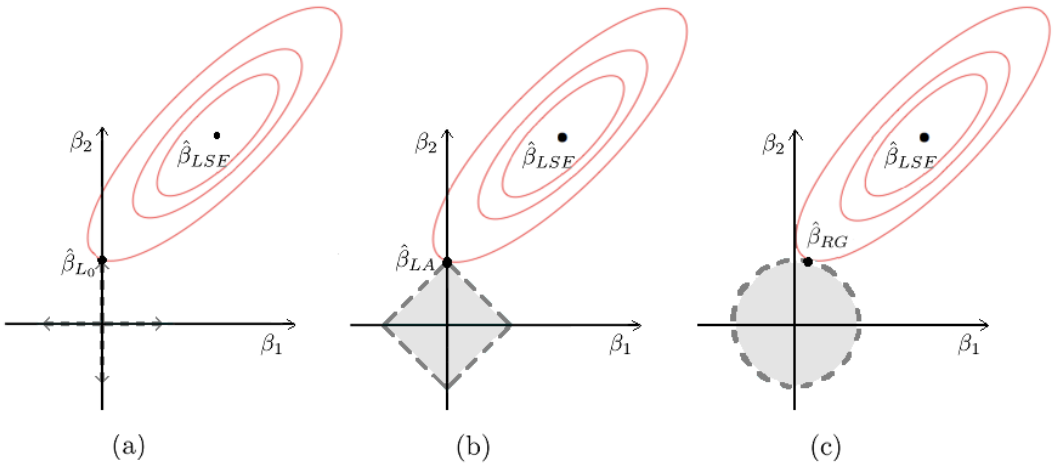

圖 3.1: 兩參數模型不同範數正規化：(a)L_0 正規化；(b)LASSO(L_1 正規化)；(c) 脊迴歸 (L_2 正規化)

3.3 Dantzig selector

Dantzig selector 由 Candes 和 Tao 提出，目的是在當參數數目遠大於觀測值數目時，能提供一個更佳的估計方式。考慮 (2.1) 式的模型，Dantzig selector 的 β 的估計 $\hat{\beta}_{DS}$ 如下：

$$\hat{\beta}_{DS} = \min_{\beta} ||\beta||_1 \quad s.t. \quad ||X^T(y - X\beta)||_\infty \leq \delta \tag{3.2}$$

其中，$||X^T(y - X\beta)||_\infty = \max_{1\leq i\leq p} |(X^T(y - X\beta))_i|$，$\delta$ 為調整參數。因為以 L_1 範數 $||\beta||_1$ 最小做為目標，加入 $||X^T(y - X\beta)||_\infty \leq \delta$ 的條件，Dantzig selector 亦是一個 L_1 正規化問題。由 (3.2) 式可知在 δ 很小時 $\hat{\beta}_{DS}$ 中每個參數都不為 0。當 δ 增加，$\hat{\beta}_{DS}$ 中某些參數會強迫為 0，δ 越大則 $\hat{\beta}_{DS}$ 為 0 的參數會越多，在 δ 提升的過程中，參數越慢降為 0 的因子越為重要，因此可以進行因子篩選。圖 3.2表示兩參數模型的 Dantzig selector，$\hat{\beta}_{DS} = (\hat{\beta}_{1DS}, \hat{\beta}_{2DS})$，右上方的橢圓形區域為條件 $||X^T(y - X\beta)||_\infty \leq \delta$，與 LASSO 不同的是解被限制在橢圓形區域上，所以 Dantzig selector 要調整虛線部分 L_1 範數的軌跡。$\hat{\beta}_{DS}$ 即為虛線軌跡與橢圓形區域的交點，發生在圖中較小的正方形虛線軌跡時，此時只有 $\hat{\beta}_{1DS}$ 為 0，所以可以進行因子篩選。

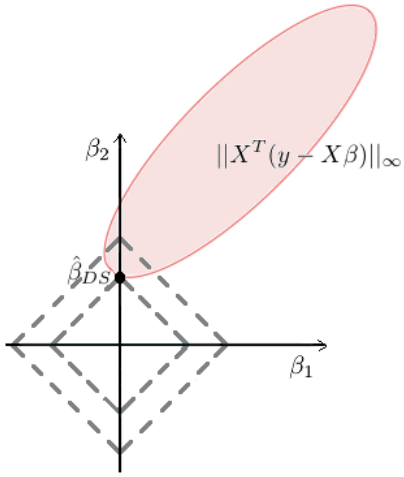

圖 3.2: 兩參數模型 Dantzig selector

但為何要使用 $||X^T(y - X\beta)||_\infty$ 做為條件？其中一個好處就是正交變換 (Orthonormal transformation) 後保持不變性。若有一個正交變換矩陣 U，U^TU 為單位矩陣，考慮 $y' = Uy$，一個好的估計應該與 U 無關，即無論觀測值為 y 還是經正交變換後的 y'，估計值 $\hat{\beta}$ 應保持不變。同樣的情形在設計矩陣 X 也類似，考慮 $X' = UX$，有以下性質：

$$X'^T(y' - X'\beta) = (UX)^T(Uy - UX\beta) = X^Ty - X^TX\beta = X^T(y - X\beta) \qquad (3.3)$$

(3.3) 式說明若使用 $||X^T(y - X\beta)||_\infty$ 作為條件，則 y 與 X 都經正交變換後，其條件不會發生改變。因為 Dantzig selector 良好的理論性質，Dantzig selector 在超飽和設計，尤其是因子數 p 遠大於運行數 n 的情況表現相當不錯，是近年來備受矚目的超飽和設計篩選因子的方法。

4 超飽和設計挑選比分析

逐步向前法、LASSO 與 Dantzig selector 都可以用於超飽和設計的因子篩選分析，本章在特定的模擬條件下，使用重要因子挑選比探討若要篩選某個數量的因子時，三種因子篩選方法的優劣。模擬條件包含了矩陣設計法、因子數、運行數、因子係數等設定。之後在其它設定保持不變的情況下，改變其中的某項設定，使用重要因子挑選比觀察此設定的變化是否對三種因子篩選方法產生影響。

在介紹重要因子挑選比之前，本文假設因子篩選數會等於重要因子數。因子篩選數 c 就是指使用因子篩選方法時，挑選出 c 個因子；而重要因子數是指真實模型中，顯著影響觀測值的因子個數。這個假設說明若我們已知真實模型的重要因子的個數，就指定該數作為使用因子篩選方法篩選因子的個數。其實現階段並沒有一個很有效的方法可以判斷進行因子篩選時，應該篩選出幾個因子。交叉驗證 (Cross-validation) 雖經常用來判斷因子篩選數，但其準確度不高，甚至在一些情況下完全無法判斷，只能得出選取因子越多越好的這類結果。在各式判斷準則，例如 R^2_{adj}、赤池訊息準則 (AIC) 或貝氏訊息準則 (BIC) 皆無法有效幫助的情況下，選取因子的個數常常是我們煩惱的課題。因為本章採用模擬的方式，確實可以知道所設定的重要因子數，故可以讓因子篩選數等於重要因子數，但在真實的情況下，不可能知道確切要篩選出多少因子，所以因子篩選數與重要因子數可能存在落差。

4.1 重要因子挑選比 (挑選比)

本章使用重要因子挑選比 (以下簡稱挑選比) 作為三種因子篩選方法比較的基準。我們定義模擬次數 t，因子篩選數 c 的挑選比 $R_t(c)$ 如下：

$$R_t(c) = \frac{\bar{s}}{c} = \frac{1}{tc}\sum_{i=1}^{t} s_i \tag{4.1}$$

其中，s_i 為第 i 次模擬中挑選的因子是重要因子的個數。挑選比其實就是 t 次模擬重要因子數為 c 的資料中，用因子篩選方法挑出的 c 個因子裡是重要因子的比例平均，挑選比越大越好，代表越能選到正確的重要因子。產生模擬資料的挑選比 $R_t(c)$ 的步驟如下：

步驟 1： 設定模擬條件，決定超飽和設計的矩陣設計法、因子數 p、運行數 n、重要因子係數 β_s、不重要因子係數 β_{ns}、模型誤差 ϵ 以及模擬次數 t。

步驟 2： 在設計矩陣中隨機挑選 c 行為重要因子，依條件設定產生模擬資料，可以得到一筆觀測值，將觀測值中心化。

步驟 3： 利用逐步向前法、LASSO 與 Dantzig selector 挑出 c 個因子，並記錄各方法所挑出的 c 個因子中重要因子的個數。

步驟 4： 步驟 2 到步驟 3 重複執行 t 次。

步驟 5： 計算 t 次模擬中各因子篩選方法挑出重要因子個數的平均數，將平均數除以 c，即可得 $R_t(c)$。

因為設計矩陣沒有包含常數項，為了確保因子篩選結果不被模型中的常數項影響，會將每一個觀測值減去觀測值平均數，稱為觀測值中心化。而為了避免某些因子混淆程度太高，嚴重影響分析，每次模擬都要重新生成重要因子與模型，以巨量資料中亂數隨機的概念消除因子間的混淆對統計結果的影響。另外，這裡的因子篩選都是篩選特定數量的因子，並非以顯著水準 α 或各式的判斷準則作為篩選因子數的考量。在本文，表 4.1的設定稱為範例條件，在之後的模擬中扮演比較的標準。這裡以產生範例條件中挑選比 $R_{1000}(20)$ 為例說明。

依範例條件，可以利用交互法建構出因子數為 235，運行數為 120 的設計矩陣，在 235 個因子中，隨機挑選 20 個因子為重要因子。再以常態分配亂數產生因子係數，重要因子係數以 $N(0, 2^2)$ 的亂數產生，而不重要因子係數以 $N(0, 0.25^2)$ 的亂數產生，為了保證重要因子確實比不重要因子顯著，亂數產生的重要因子係數其絕對值在 1 以下重選，而亂數產生的不重要因子係數其絕對值在 1 以上重選。常數項的產生方式如同重要因子，重要因子範例如表 4.2，有了因子係數即可產生

表 4.1: 範例條件

範例條件
矩陣設計法：交互法
因子數 $p = 235$
運行數 $n = 120$
重要因子係數 $\beta_s \sim N(0, 2^2)$ 且 $\lvert\beta_s\rvert \geq 1$
不重要因子係數 $\beta_{ns} \sim N(0, 0.25^2)$ 且 $\lvert\beta_{ns}\rvert \leq 1$
模型誤差 $\epsilon \sim N(0, 1^2)$
模擬次數 $t = 1000$

表 4.2: 重要因子範例

因子	常數	$p1$	$p10$	$p19$	$p30$	$p48$	$p63$
係數	-3.89	-2.83	-2.71	1.47	1.75	-3.15	3.06
因子	$p72$	$p89$	$p92$	$p96$	$p108$	$p119$	$p139$
係數	3.40	1.90	1.33	-3.13	-2.49	2.76	-1.10
因子	$p152$	$p166$	$p175$	$p181$	$p187$	$p206$	$p230$
係數	-1.11	-1.67	-1.83	-1.03	-2.35	-2.48	1.96

真實模型，並加入模型誤差項模擬一筆觀測值。為了方便比較，本文模擬的模型誤差項都由期望值為 0 變異數為 1 的常態分配生成。

將觀測值中心化之後，分別使用逐步向前法、LASSO 和 Dantzig selector 三種因子篩選方法進行因子排序，挑選前 20 個顯著因子，如表 4.3。表中深色部分為重要因子，在該次模擬中，逐步向前法選出 18 個重要因子表現最好，Dantzig selector 選出 17 個重要因子次之，而 LASSO 只選出 16 個重要因子。重複模擬 1000 次後，將每次模擬三種因子篩選方法挑出重要因子的個數記錄下來，如表 4.4，並計算 1000 次模擬中三種因子篩選方法挑出重要因子的個數之平均數 $\bar{s}$，將平均數除以因子篩選數 20，即可得到挑選比 $R_{1000}(20)$，如表 4.5。

可以利用相同的方式產生不同因子篩選數 c 的挑選比。範例條件中因子篩選數為 1 到 60 的挑選比如表 4.6所示，將表格作圖呈現，如圖 4.1。為了方便，之後皆以這類散佈圖進行挑選比的比較，圖 4.1中的空心圓點、實心圓點、菱形點分別代表範例條件中逐步向前法、LASSO 與 Dantzig selector 中的挑選比。

表 4.3: 不同因子篩選方法挑選因子範例

逐步向前法									
$p72$	$p63$	$p108$	$p10$	$p48$	$p1$	$p96$	$p230$	$p206$	$p119$
$p187$	$p175$	$p19$	$p89$	$p181$	$p30$	$p92$	$p166$	$p9$	$p159$
LASSO									
$p72$	$p63$	$p187$	$p108$	$p10$	$p206$	$p48$	$p230$	$p1$	$p96$
$p119$	$p159$	$p175$	$p200$	$p30$	$p19$	$p133$	$p89$	$p92$	$p118$
Dantzig selector									
$p72$	$p63$	$p108$	$p48$	$p206$	$p10$	$p187$	$p1$	$p96$	$p230$
$p119$	$p159$	$p175$	$p19$	$p89$	$p92$	$p30$	$p181$	$p173$	$p200$

表 4.4: $R_{1000}(20)$ 記錄重要因子個數範例

個數	9	10	11	12	13	14	15	16	17	18	19	20
逐步向前法	1	1	1	1	21	35	55	107	181	272	252	73
LASSO	0	0	4	6	49	79	202	257	244	125	28	6
Dantzig selector	0	0	0	6	17	32	94	212	260	248	113	18

表 4.5: $R_{1000}(20)$ 產生範例

	$\bar{s}$	$R_{1000}(20)$
逐步向前法	17.56	0.88
LASSO	16.05	0.80
Dantzig selector	16.93	0.85

4.2 不同設定的挑選比分析

本節以範例條件的挑選比分析作為開端，探討三種不同的因子篩選方法在不同因子篩選數的挑選比表現。之後探討在不同的設計矩陣方法、不同的因子數或不同的重要因子係數之下挑選比的變化情形。

4.2.1 範例條件下的挑選比分析

圖 4.1為範例條件的挑選比，可以看出各因子篩選方法的挑選比連線軌跡呈現類似曲線的圖樣，若模擬次數 t 越多則軌跡會越平滑。在因子篩選數 c 不大時 (如 c 為 3 以前) 三種因子篩選方法的挑選比表現並沒有明顯差異，而當 c 越來越大時，三種因子篩選方法的挑選比表現都開始下降，因為這代表著要命中 c 個重

表 4.6: 範例條件因子篩選數 c 為 1 到 60 的挑選比 $R_{1000}(c)$

c	1	2	3	4	5	6	7	8	9	10
逐步向前法	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.91	0.90	0.90
LASSO	0.90	0.90	0.89	0.88	0.88	0.87	0.87	0.86	0.86	0.85
Dantzig selector	0.90	0.90	0.90	0.90	0.89	0.89	0.89	0.89	0.89	0.88
c	11	12	13	14	15	16	17	18	19	20
逐步向前法	0.90	0.90	0.90	0.90	0.89	0.89	0.89	0.88	0.88	0.88
LASSO	0.85	0.84	0.84	0.83	0.83	0.82	0.82	0.81	0.81	0.80
Dantzig selector	0.88	0.88	0.88	0.87	0.87	0.87	0.86	0.86	0.85	0.85
c	21	22	23	24	25	26	27	28	29	30
逐步向前法	0.88	0.88	0.87	0.86	0.85	0.85	0.84	0.83	0.82	0.81
LASSO	0.80	0.80	0.79	0.78	0.78	0.77	0.77	0.76	0.76	0.75
Dantzig selector	0.84	0.84	0.84	0.83	0.83	0.82	0.82	0.81	0.80	0.80
c	31	32	33	34	35	36	37	38	39	40
逐步向前法	0.79	0.79	0.77	0.77	0.76	0.74	0.73	0.71	0.69	0.69
LASSO	0.75	0.74	0.74	0.73	0.73	0.72	0.72	0.72	0.71	0.70
Dantzig selector	0.79	0.79	0.78	0.78	0.77	0.76	0.76	0.75	0.75	0.74
c	41	42	43	44	45	46	47	48	49	50
逐步向前法	0.67	0.66	0.65	0.63	0.62	0.61	0.60	0.59	0.58	0.56
LASSO	0.70	0.69	0.69	0.69	0.69	0.68	0.68	0.67	0.67	0.66
Dantzig selector	0.73	0.72	0.72	0.71	0.71	0.70	0.70	0.69	0.69	0.68
c	51	52	53	54	55	56	57	58	59	60
逐步向前法	0.56	0.55	0.54	0.53	0.52	0.52	0.51	0.51	0.50	0.49
LASSO	0.66	0.66	0.66	0.66	0.65	0.65	0.64	0.64	0.64	0.64
Dantzig selector	0.68	0.68	0.67	0.67	0.66	0.66	0.65	0.65	0.65	0.64

要因子會更加困難。直到 c 約為 30 以前，比起其它兩種方法，逐步向前法的挑選比下降較慢，換言之其表現會較好，尤其是在 c 約為 21 左右時，逐步向前法的挑選比會比起第二好的 Dantzig selector 的挑選比還多出 0.04 左右。但是當 c 超過 30 之後，逐步向前法的挑選比表現會急遽下降，在 c 為 31 左右開始不如 Dantzig selector，在 c 為 38 左右開始不如 LASSO，之後當 c 越大時與其它兩種方法的挑選比表現差距越大。至於同為 L_1 正規化的 LASSO 與 Dantzig selector，不像與逐步向前法比較有輸有贏，LASSO 在 c 為 1 到 60 時表現皆不如 Dantzig selector。所以在範例條件之下，當 c 小於 31 時，傾向使用逐步向前法篩選因子；當 c 大於 31 時，傾向使用 Dantzig selector 篩選因子。

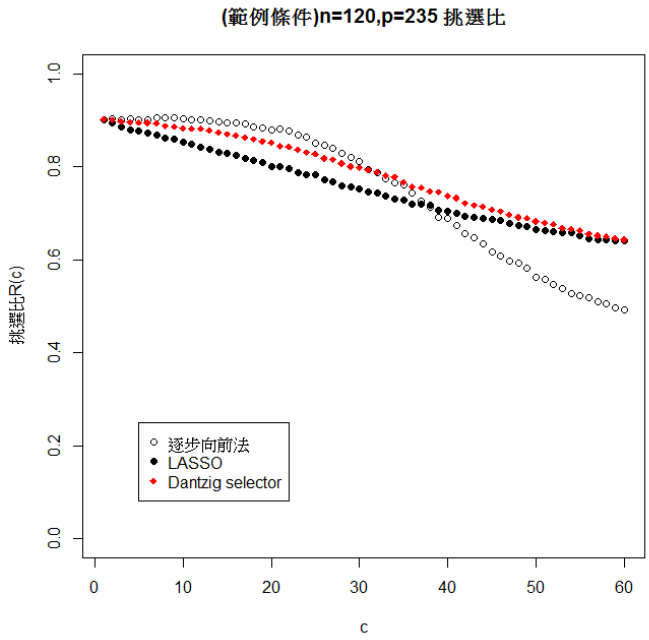

圖 4.1: 範例條件的挑選比

4.2.2 不同設計矩陣建構方法的挑選比分析

第二章有介紹可以使用 $E(s^2)$ 來判斷設計矩陣的優劣，在運行數為 120、因子數為 235 的設計矩陣中，使用折半法與使用交互法建構的設計矩陣 $E(s^2)$ 差異相當小。那若以這兩種不同的矩陣建構法建構出的設計矩陣進行挑選比分析，得出的結果是否會有差異？圖 4.2左圖與圖 4.1完全相同，為範例條件下的挑選比；而作為對比，圖 4.2右圖為範例條件將矩陣的建構方法從交互法修改為半分法，其餘設定與範例條件相同的挑選比。兩圖中三種因子篩選方法的挑選比表現都相當接近，顯現出在 $E(s^2)$ 相近時不同的矩陣建構方法對挑選比分析的影響並不會有顯著的差異。

但是若使用不同矩陣建構方法的設計矩陣 $E(s^2)$ 差異並不小時，挑選比結果是否會存在差異？圖 4.3左圖與圖 4.3右圖分別也是使用交互法與使用折半法建構設計矩陣的挑選比，但是改為運行數為 120、因子數為 150 的設計矩陣，在這個規模的設計矩陣中，兩種矩陣建構方法的 $E(s^2)$ 會有顯著差異，交互法的 $E(s^2)$ 約比折半法的 $E(s^2)$ 少 21 左右。圖 4.3可以看出使用交互法的挑選比表現的確會好於使用折半法，儘管差距並不是很明顯。這顯示若是建構的設計矩陣 $E(s^2)$ 有

顯著差異時，$E(s^2)$ 較小的設計矩陣建構方法的挑選比表現會更好，也驗證第二章所提及過 $E(s^2)$ 越小則設計矩陣越好。

4.2.3 不同因子數的挑選比分析

當其它設定保持不變，僅改變範例條件中的因子數設定時，挑選比結果會如何變化？圖 4.4左圖為將範例條件中的因子數改為 119 的挑選比，因為此時因子數小於運行數，並非超飽和設計，此情形可以找到並使用行與行間完全正交的設計矩陣。因為因子之間沒有任何混淆的緣故，此時不論因子篩選數 c 為何，三種因子篩選方法的挑選比都會相當趨近 1。而圖 4.4右圖、圖 4.5左圖與圖 4.5右圖分別為將範例條件中的因子數改為 150、320 與 400 的挑選比，可以發現在相同的 c 下，當因子數越大時，其挑選比表現會越差，這是因為要討論的因子數一多，不可避免地會更有機會出現混淆程度較高的因子，造成錯誤選取重要因子的可能性增加。至於三種因子篩選方法的挑選比表現，在 c 不大的情形之下，逐步向前法的挑選比表現仍舊較好，但若因子數越大時，逐步向前法與其它兩方法挑選比交叉的 c 會越小，即逐步向前法的挑選比表現會下降。其它不同設定的模擬也顯示出，在因子數遠大於運行數的超飽和設計中，逐步向前法的挑選比表現就沒有如此優異。如圖 4.5右圖，儘管在 c 約為 15 之前，逐步向前法的挑選比表現仍舊較佳，但與表現第二好的 Dantzig selector 的挑選比都已經相當接近，差距皆在 0.01 以下，而當 c 超過 15 以後，逐步向前法的挑選比表現會完全不如 Dantzig selector，在 c 超過 25 以後也開始不如 LASSO，且 c 越大時差異會越明顯。另外，不論因子數的設定為何，LASSO 的挑選比表現仍然完全不如 Dantzig selector。

當其它設定保持不變，僅改變範例條件中的運行數設定時，挑選比結果又會如何變化？其實固定因子數增加運行數與固定運行數減少因子數，對挑選比的結果影響相當類似。所以在其它設定相同的情況下，若運行數越大時，逐步向前法與其它兩方法挑選比交叉的 c 會越大，即逐步向前法的挑選比表現會上升。在因子數沒有與運行數差距太大的超飽和設計中，逐步向前法的挑選比表現會較好。

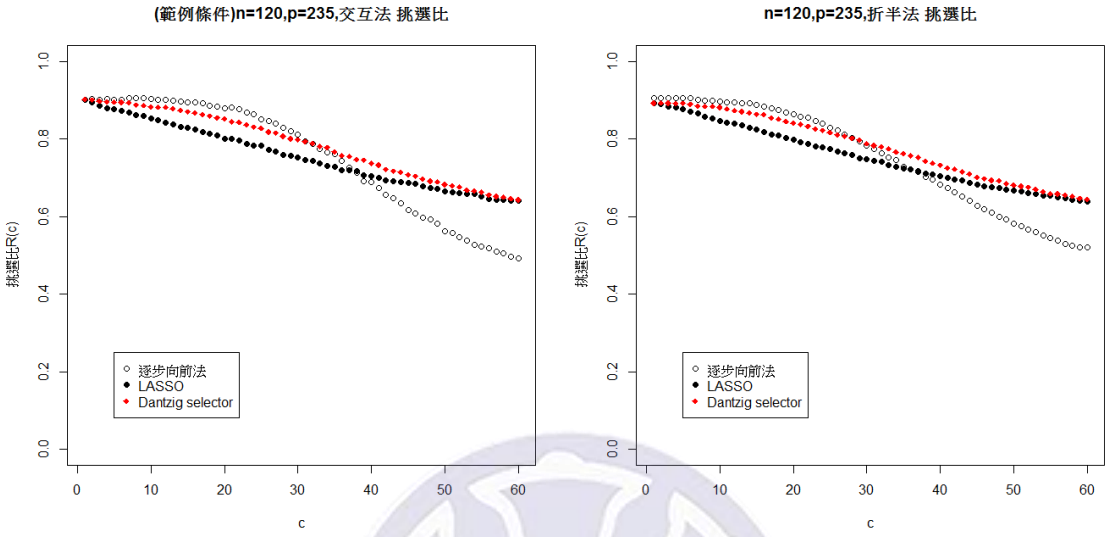

圖 4.2: $E(s^2)$ 相近時不同設計矩陣建構方法的挑選比：左圖為交互法、右圖為折半法

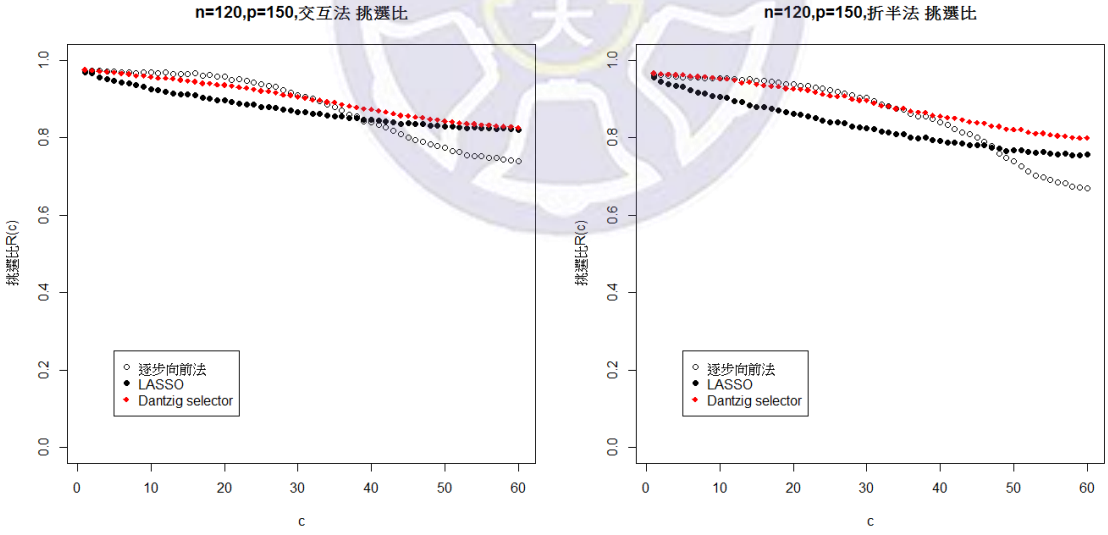

圖 4.3: $E(s^2)$ 不相近時不同設計矩陣建構方法的挑選比：左圖為交互法、右圖為折半法

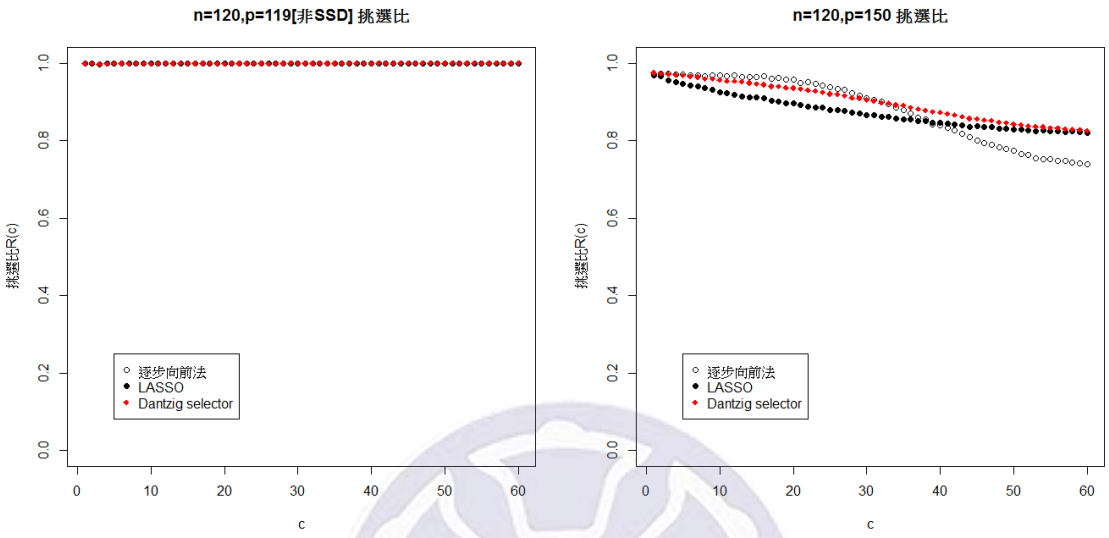

圖 4.4: 不同因子數的挑選比 (一)：左圖因子數為 119、右圖因子數為 150

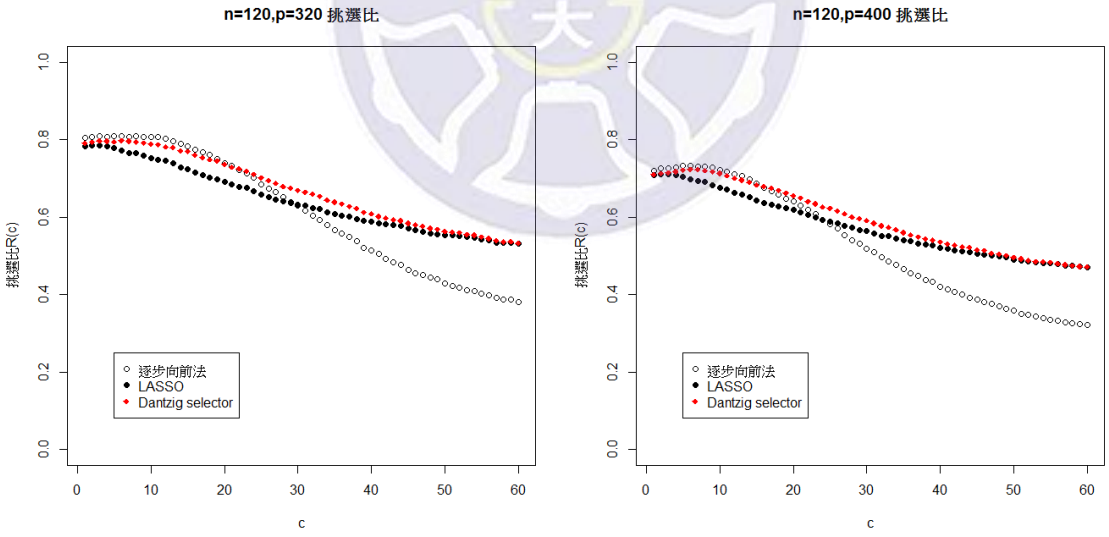

圖 4.5: 不同因子數的挑選比 (二)：左圖因子數為 320、右圖因子數為 400

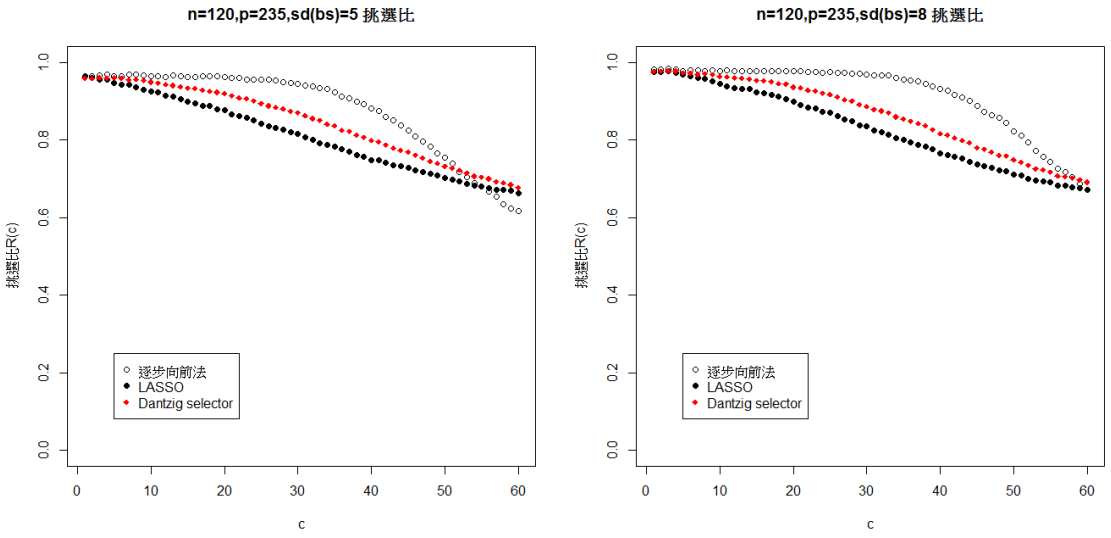

圖 4.6: 不同重要因子係數的挑選比：左圖的重要因子係數從標準差為 5 常態分配挑出、右圖的重要因子係數從標準差為 8 常態分配挑出

4.2.4 不同重要因子係數的挑選比分析

那保持其它設定不變的情況下，只改變範例條件中的重要因子係數時，挑選比結果會如何變化？圖 4.6左圖與圖 4.6右圖分別是將範例條件中的重要因子係數改為從標準差為 5 與標準差為 8 的常態分配挑出的挑選比。可由圖中看出挑出重要因子係數的常態分配標準差越大時，LASSO 與 Dantzig selector 的挑選比並不會發生明顯改變，但是逐步向前法的挑選比表現會顯著提升。甚至如圖 4.6右圖，逐步向前法與其它兩方法挑選比交叉的 c 約為 60，換言之其它兩方法的挑選比表現會完全不如逐步向前法。所以挑出重要因子係數的常態分配標準差越大時，逐步向前法的挑選比表現相對較佳，反之則 LASSO 與 Dantzig selector 的挑選比表現相對較佳。

4.3 挑選比分析的應用

本節以超飽和設計的實際資料來說明挑選比分析的應用。表 4.7的原始資料來自於 Williams(1968)，是因子數為 24、運行數為 28 採用 Plackett 和 Burman 設計的部分因子設計。Lin(1993) 首先以折半法將這筆資料處理，因為折半後因子 13 與因子 16 完全混淆，所以將因子 13 刪除，形成因子數為 23、運行數為 14 超飽和設

計。本文稱表 4.7的資料為半 Williams 資料，此資料後來成為超飽和設計的經典資料，經常用於分析超飽和設計。

表 4.7: 半 Williams 資料

運行	因子																							觀測值
	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21	22	23	24	y
1	+	+	+	−	−	−	+	+	+	+	+	−	−	−	+	+	−	−	+	−	−	−	+	133
2	+	−	−	−	−	−	+	+	+	−	−	−	+	+	+	−	+	−	−	+	+	−	−	62
3	+	+	−	+	+	−	−	−	−	+	−	+	+	+	+	+	−	−	−	−	+	+	−	45
4	+	+	−	+	−	+	−	−	−	+	+	−	−	+	+	−	+	+	+	−	−	−	−	52
5	−	−	+	+	+	+	−	+	+	−	−	−	−	+	+	+	−	−	+	−	+	+	+	56
6	−	−	+	+	+	+	+	−	+	+	+	−	+	+	−	+	+	+	+	+	+	−	−	47
7	−	−	−	−	+	−	−	+	−	+	−	+	+	−	+	+	+	+	+	+	−	−	+	88
8	−	+	+	−	−	+	−	+	−	+	−	−	−	−	−	−	−	+	−	+	+	+	−	193
9	−	−	−	−	−	+	+	−	−	−	+	+	−	+	−	+	+	−	−	−	−	+	+	32
10	+	+	+	+	−	+	+	+	−	−	−	+	+	+	−	+	−	+	−	+	−	−	+	53
11	−	+	−	+	+	−	−	+	+	−	+	−	+	−	−	−	+	+	−	−	−	+	+	276
12	+	−	−	−	+	+	+	−	+	+	+	+	−	−	+	−	−	+	−	+	+	+	+	145
13	+	+	+	+	+	+	−	+	−	+	−	−	+	−	−	−	+	−	+	+	−	+	−	130
14	−	−	+	−	−	−	−	−	−	−	+	+	+	−	−	−	−	−	+	−	+	−	−	127

圖 4.7為半 Williams 資料設計矩陣的挑選比分析，其中圖 4.7左圖與圖 4.7右圖的重要因子係數分別從標準差為 2 與標準差為 20 的常態分配挑出，而不重要因子係數、模型誤差與模擬次數皆與範例條件設定相同。但是兩圖中何者更能顯示出真實情形？其實這與重要因子跟不重要因子的係數差距有關，取決於我們相信兩者的差距大或不大。因為這裡在分析選取率時，都將不重要因子係數與模型誤差固定住，所以重要因子係數抽取分配的標準差差異，也就是重要因子與不重要因子係數差距，圖 4.7左圖可以理解為重要因子與不重要因子係數差距不大的情況，而圖 4.7右圖則是重要因子與不重要因子係數有明顯差距的情況。但是無論圖 4.7的左圖與右圖，都可以發現 c 在 4 以前逐步向前法的表現幾乎都是三種方法中最好的，所以若要篩選 4 個以內的因子，可以優先考慮逐步向前法所篩選的因子。以三種因子篩選方法在 Williams 原始資料中篩選 4 個因子，都一致指出因子 15、因子 20、因子 17、因子 4 為重要因子，但因為超飽和設計中因子之間會有混淆情形出現的緣故，三種因子篩選方法在半 Williams 資料的因子篩選結果會出現分歧，結果如表 4.8所示。若將原始資料因子篩選結果的 4 個因子視為重要因子，

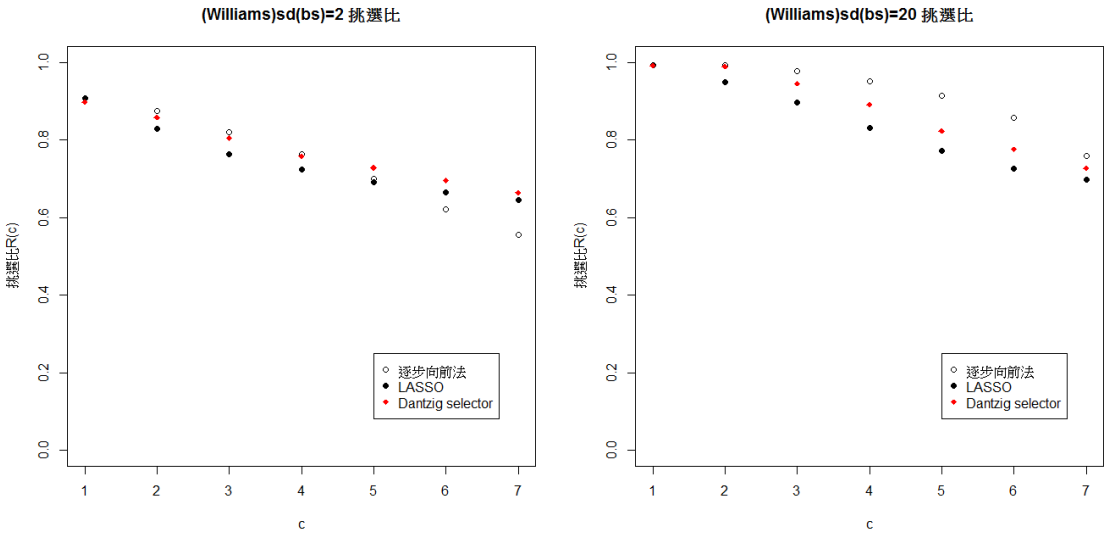

圖 4.7: 半 Williams 資料的挑選比：左圖的重要因子係數從標準差為 2 常態分配挑出、右圖的重要因子係數從標準差為 20 常態分配挑出

以深色表示，則在半 Williams 資料中逐步向前法、LASSO 與 Dantzig selector 分別可以挑出 3、2、2 個重要因子，顯示優先考慮逐步向前法是合理的。值得一提的是，因子 12 的顯著程度在 Williams 原始資料中相當低，但是在半 Williams 資料篩選 4 個因子時，三種因子篩選方法都有挑出因子 12，這正是超飽和設計中的混淆情形所帶來的影響。不足的運行數會導致在進行超飽和設計的因子篩選時，被因子間的混淆程度所影響，進而可能挑選到錯誤的重要因子。

表 4.8: Williams 原始資料與半 Williams 資料在因子篩選數為 4 時三種因子篩選方法的篩選因子比較

Williams 原始資料				
三種因子篩選方法	因子 15	因子 20	因子 17	因子 4
半 Williams 資料				
逐步向前法	因子 15	因子 12	因子 20	因子 4
LASSO	因子 15	因子 17	因子 12	因子 2
Dantzig selector	因子 15	因子 20	因子 12	因子 16

5 結論

本文中使用逐步向前法、LASSO 與 Dantzig selector 來進行超飽和設計的因子篩選，並以挑選比的表現比較三種因子篩選方法。結果顯示在同樣的設定之下，若篩選的因子數量不多，逐步向前法的挑選比表現會相當不錯，但是在因子篩選數大於某個數量後，Dantzig selector 的挑選比表現會開始好於逐步向前法，再後來 LASSO 的挑選比表現也會好於逐步向前法。雖然 LASSO 的表現始終不如 Dantzig selector，但是當因子篩選數較大的時候差距會相當小。

不同的矩陣建構法在 $E(s^2)$ 接近時挑選比的的差異並不大。在因子數比運行數大上許多的情況，逐步向前法的挑選比表現會下降，又或者在重要因子與不重要因子差距不明顯時，也不利於逐步向前法的使用，此時採用 Dantzig selector 進行因子篩選更有優勢；反之逐步向前法在因子數與運行數差距不大，或因子之間的重要程度區別明顯時挑選比表現相當優異。

挑選比也可以應用在分析超飽和設計的資料上，在其設計矩陣加入適當的因子係數設定後，可以由挑選比的結果判斷當篩選某個數量的因子時，哪一種因子篩選方法更佳。但是挑選比分析仍有限制存在，其假設重要因子數等於因子篩選數就是一個主要的限制，我們不太可能知道真實資料中的重要因子數，所以因子篩選數與重要因子數往往會存在差距。未來若要繼續使用挑選比來分析因子篩選方法時，可以考慮當重要因子數與因子篩選數不同時的情形，此時要討論重要因子數大於或小於因子篩選數兩種狀況，挑選比的式子也必須進行修正，情況就會複雜很多。

參考文獻

Beattie, S. D., Fong, D. K. H., & Lin, D. K. J. (2002). A two-stage bayesian model selection strategy for supersaturated designs. *Technometrics*, *44*(1), 55–63.

Booth, K. H. V., & Cox, D. R. (1962). Some systematic supersaturated designs. *Technometrics*, *4*(4), 489–495.

Box, G. E. P., & Meyer, R. D. (1986). An analysis for unreplicated fractional factorials. *Technometrics*, *28*(1), 11–18.

Candes, E., & Tao, T. (2007). The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, *35*(6), 2313–2351.

Cheng, C.-S. (1997). E(s2)-optimal supersaturated designs. *Statistica Sinica*, *7*(4), 929–939.

Chipman, H., Hamada, M., & Wu, C. F. J. (1997). A bayesian variable-selection approach for analyzing designed experiments with complex aliasing. *Technometrics*, *39*(4), 372–381.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*(2), 407–499.

Fang, K.-T., Lin, D. K. J., & Liu, M.-Q. (2003). Optimal mixed-level supersaturated design. *Metrika*, *58*(3), 279–291.

Li, R., & Lin, D. K. J. (2002). Data analysis in supersaturated designs. *Statistics & Probability Letters*, *59*(2), 135–144.

Li, W. W., & Wu, C. F. J. (1997). Columnwise-pairwise algorithms with applications to the construction of supersaturated designs. *Technometrics*, *39*(2), 171–179.

Lin, D. K. J. (1993). A new class of supersaturated designs. *Technometrics*, *35*(1), 28–31.

Lu, X., & Wu, X. (2004). A strategy of searching active factors in supersaturated screening experiments. *Journal of Quality Technology*, *36*(4), 392–399.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal*

on Computing, *24*(2), 227–234.

Nguyen, N.-K. (1996). An algorithmic approach to constructing supersaturated designs. *Technometrics*, *38*(1), 69–73.

Phoa, F. K. H., Pan, Y.-H., & Xu, H. (2009). Analysis of supersaturated designs via the dantzig selector. *Journal of Statistical Planning and Inference*, *139*(7), 2362–2372.

Plackett, R. L., & Burman, J. P. (1946). The design of optimum multifactorial experiments. *Biometrika*, *33*(4), 305–325.

Satterthwaite, F. E. (1959). Random balance experimentation. *Technometrics*, *1*(2), 111–137.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (methodological)*, *58*(1), 267–288.

Westfall, P. H., Young, S. S., & Lin, D. K. J. (1998). Forward selection error control in analysis of supersaturated designs. *Statistica Sinica*, *8*(1), 101–117.

Williams, K. R. (1968). Designed experiments. *Rubber Age*, *100*, 65–71.

Wu, C. F. J. (1993). Construction of supersaturated designs through partially aliased interactions. *Biometrika*, *80*(3), 661–669.

Zhang, Q.-Z., Zhang, R.-C., & Liu, M.-Q. (2007). A method for screening active effects in supersaturated designs. *Journal of Statistical Planning and Inference*, *137*(6), 2068–2079.