國立臺灣大學工學院工業工程學研究所

碩士論文

Graduate Institute of Industrial Engineering

College of Engineering

National Taiwan University

Master Thesis

一般化縮減信賴域搜尋及其在多目標統計模型最佳化之應用

Generalized Reduced Trust-region Search and Its Applications to Statistical Multi-Objective Optimization

陳彥廷

Yen-Ting Chen

指導教授：陳正剛 博士

Advisor: Argon Chen, Ph.D.

中華民國 97 年 8 月

August, 2008

謝誌

看著電腦中一個又一個的論文備份檔，與碩士學位論文及格證明，一次又一次的 Group Meeting，經歷了四個寒暑，無數個看日出的清晨，令人難忘的畢業旅行，這其中最令人感到欣慰的一件事，莫過於論文的順利完成！在我的研究生涯中，首要感謝的當然是我的指導教授陳正剛老師，老師對於學術研究的熱誠，與對事物的嚴謹，與那融會貫通的教學方式，都是我一生當中值得學習的對象，也要感謝老師在論文階段，對本論文內容做到滴水不露、逐字斧正的程度，實令我深為感佩!另外，感謝口試委員的不吝指導，包括兩位從新竹不辭路途遙遠趕來的柯志明與蔡雅蓉經理、與元智大學的范書愷老師，您的著作是本論文相當重要的基石、所上的兩位老師，張時中老師與吳政鴻老師，承蒙五位口試委員在論文口試上的指正與建議，使得本論文更加完備，在此也深表感激。感謝所辦最熱心助理們，Monica、淑云、相怡，給與我許多生活與行政工作上的協助，與常常提醒我要工讀的琍文，幫老師傳話的助理宇珊。感謝兩位學識淵博的博班學長，Amos 在修課上的幫助與小藍在本論文的相關理論的討論與指導。也要感謝患難與共的戰友們，回家與吃飯的好伙伴小耀、同為台南人全身名牌的靖淵、一起住半年才了解他在想什麼的政緯、躺著也中槍但脾氣超好的建名、研究室守護者所辦愛將凌誠、垂直正交化之神育維，講話無理頭的宅宅青杉、還有其他好同學們，勝傑、于珈、哲欣、俊男、皙杰、俊儒、尚樺、建宏、韋銓，同為阿剛 Group 的學弟妹們，小薛薛、博尉、惟婷、士晉、信融。最後要感謝我最摯愛的家人們，特別是無怨無悔對我的栽培與付出我的爸媽，常常關心我的爺爺與奶奶，相當支持我的叔叔嬸嬸，還有大、小姑姑與大、小姑丈，還有一起長大的兄弟姐妹們，沒有你們 24 年來的支持也不會有今天的我。最後也要感謝這最後半年來，常常聽我抱怨，但又為我鼓勵與打氣的采琳，願未來的路能與你共同努力。

彥廷 謹誌於台大工工所

民國九十七年八月二十七日

論文摘要

一般化縮減梯度(Generalized Reduced Gradient)法是一個廣受喜愛的非線性規劃問題解法，但於具有四次目標式之多目標統計最佳化(Statistical Multi-objective Optimization)問題中，一般化縮減梯度法容易出現搜尋路徑曲折(Zigzagging)的現象。於本研究中，我們改善了信賴域(Trust Region)搜尋法，並發展了一般化縮減信賴域(Generalized Reduced Trust Region)搜尋法。此方法結合了一般化縮減梯度與信賴域搜尋法，將具有限制式的非線性規劃問題，轉化成由非基礎變數(Nonbasic variable)所構成的不具現制式的非線性規劃問題，並且在縮減空間(Reduced Space)中獲得最佳改善的方向，且於案例中克服了一般化縮減梯度法的缺點，此外，我們也結合了一般化縮減信賴域搜尋法與 Zoutendijk’s 搜尋法以改善搜尋效果。最後，為了驗證該演算法的成效，我們利用一個眾所皆知且具四次目標式的測試問題：Rosenbrock’s function 與三個案例來測試，第一個案例是關於半導體可製造性設計(DFM)之問題，而第二個案例是半導體供應鏈穩健配置之案例，最後一個案例為半導體製造過程中，臨界尺寸均勻度(CDU)在軌道系統之曝光後烘烤(PEB)步驟下之最佳化。經由與商業套裝軟體 Lingo 的結果比較，我們可以在相似的計算時間內獲得同樣甚至更好的最佳解。

關鍵字：非線性規劃，多目標統計最佳化，一般化縮減梯度法，一般化縮減信賴域法，信賴域法

ABSTRACT

"Generalized Reduced Gradient" method is a popular NLP method, but it often incurs a zigzagging search path especially for the statistical multi-objective optimization (SMOO) problem where the objective function is a quartic function. In this study, we improve the "Trust Region (TR)" search method and develop the "Generalized Reduced Trust Region" (GRT) search method which combines the GRG method and the improved TR method. The GRT search transforms the constrained NLP problem to an unconstrained NLP problem consisting of only the nonbasic variables and searches the best improving direction in the reduced space. The proposed method is shown to overcome the zigzagging problem of the GRG method. To verify the performance of our methods, we study a well know test problem and three cases. The test problem is called Rosenbrock's function which has a quartic objective function with two decision variables. The first case is a semiconductor design for manufacturing (DFM) problem. The second case is the problem to configure a robust semiconductor supply chain. The final case is the "Track System PEB CDU Optimization". Compared against the result of the commercial software "Lingo", the same or better solutions are obtained by our methods with comparable computation time.

Keywords: Nonlinear Programming, Statistical Multi-Objective Optimization, Generalized Reduced Gradient Method, Generalized Reduced Trust Region Method, Trust Region Method

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Problem Definition and Formulation

Response surface methodology (RSM) is a powerful technique for quality and productivity improvement.

Many processes such as chemical processes, manufacturing processes, development processes, etc, are critical to productivity. These processes transform inputs into outputs. In chemical processes, reaction time and reaction temperature are inputs and the yield is output. Actually, engineers usually want to know how inputs affect outputs. Response Surface Methodology (RSM) is a set of mathematical and statistical techniques used by researchers and engineers to aid in the solution of certain types of problems. In RSM, we call the inputs and the outputs "explanatory (independent) variables" and "responses". The response is normally measured on a continuous scale and is a measure representing the most important function of the system. The independent variables are the fade affecting the response and are usually controllable.

Suppose the yield of a chemical process is affected by two factors. The first one is reaction time and the second one is reaction temperature. At beginning we only know the relationship between yield and these two factors is expressed as follows:

$$yeild = f(reaction\ \ tme, reaction\ \ temperature) + \varepsilon\ . \tag{1.1}$$

In order to disinter the function *f*, engineers use RSM procedures involve experimental strategy, mathematical methods, and statistical inference which, when combined, enable them to make an efficient empirical exploration of the system in which they are interested. First, they design a set of designs using experimental strategy (design). The purpose of the experimental strategy (design) is to enable the analyst to explore the response surface [12] with equal precision, in any direction. Subsequently engineers collect a set of data by performing these designs.

After obtaining these data, a model can be built by using regression analysis. Here we suppose the linear multiple regression is applied. The relationship between response and factors is express as:

$$yield = \beta_0 + \beta_1 \times reaction\ \ time + \beta_2 \times reaction\ \ temperature + \varepsilon \tag{1.2}$$

, where β_1 and β_2 mean the effects to yield by changing one unit reaction time and reaction temperature respectively. These two coefficients are useful information for analyzing the chemical process. It is also convenient to view the response surface in the two-dimensional time-temperature plane, as in Figure 1.1.

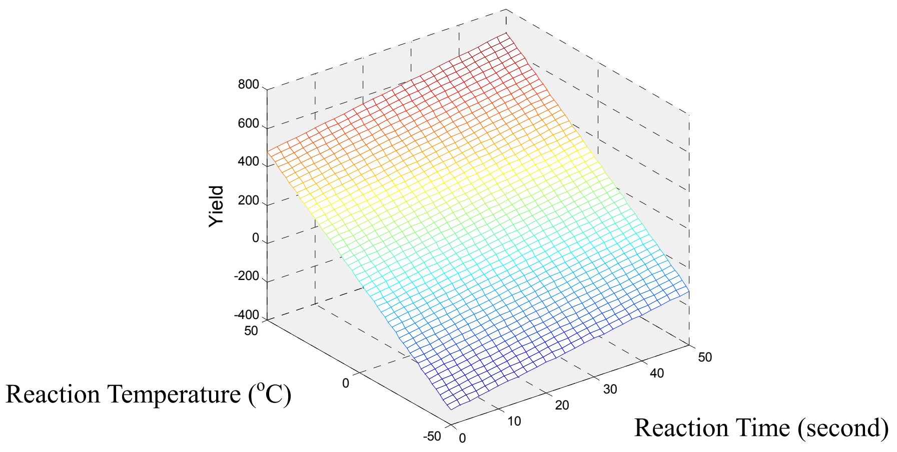


Figure 1.1 Response surface of chemical process

Normally, there are two stages of performing RSM. The first stage is called response surface design which is mentioned in last two paragraphs. The second stage is called response surface optimization or response surface analysis. In the latter stage, engineers use optimization techniques such as steepest descent/ascent method to decide the search direction for obtaining the best value of independent variable i.e. reaction time and reaction temperature in our example.

In most engineering problem, the linear response surface model is not satisfactory. Indeed the relationship between response and predictor variable is nonlinear relationship. So we need nonlinear functions help us to describe the relationship well. Now we consider there are n predictor variables, and the i-th expected response is denoted as $\hat{y}_i$. The nonlinear response surface model can be expressed as a quadratic function as follows:

$$
\begin{aligned}
\hat{y}_i &= f_i(x_1 \cdots x_n) \\
&= b_{i0} + b_{i1}x_1 + b_{i2}x_2 + \cdots + b_{in}x_n + \hat{\beta}_{i11}x_1^2 + \cdots + \hat{\beta}_{inn}x_n^2 \\
&\quad + \hat{\beta}_{i12}x_1x_2 = \hat{\beta}_{i13}x_1x_3 + \cdots + \hat{\beta}_{i,n-1,n}x_{n-1}x_n \\
&= b_{i0} + \mathbf{b}_i^T\mathbf{x} + \mathbf{x}^T\mathbf{B}_i\mathbf{x}
\end{aligned}
\tag{1.3}
$$

, where $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ denotes n variables;

$$
\mathbf{b}_i = \begin{pmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{in} \end{pmatrix}
$$

, and

$$
\mathbf{B}_i = \begin{pmatrix} \hat{\beta}_{i11} & \hat{\beta}_{i12}/2 & \cdots & \hat{\beta}_{i1n}/2 \\ & \hat{\beta}_{i22} & \cdots & \hat{\beta}_{i2n}/2 \\ & & \ddots & \vdots \\ sym & & & \hat{\beta}_{inn} \end{pmatrix}
$$

are the linear and quadratic coefficients, respectively.

Most engineering requirements would specify a desired target T_i for each response $\hat{y}_i$. That is, the difference between $\hat{y}_i$ and its target T_i should be as small as possible. Here, the quadratic loss function [1] can be used to measure the total difference. That is,

$$
Min \sum_i w_i(\hat{y}_i - T_i)^2 = \sum_i w_i(b_{i0} + \mathbf{b}_i^T\mathbf{x} + \mathbf{x}^T\mathbf{B}_i\mathbf{x} - T_i)^2 . \tag{1.4}
$$

In (1.4), w_i is a user-specified weight representing the relative importance for $\hat{y}_i$ to conform to the target. In this research, without loss of generality, all w_i are assumed to equal to 1. In addition to the target, the response $\hat{y}_i$ should be located in a specification widow with an upper specification limit U_i and a lower specification limit L_i:

$$L_i \leq \quad b_{i0} + \mathbf{b}_i{}^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} \quad \leq U_i . \tag{1.5}$$

Moreover, each input variable x_n usually has an experimental region with an upper bound U_{x_n} and a lower bound L_{x_n} :

$$L_{x_n} \leq x_n \leq U_{x_n} .$$

(1.6)

The purpose of the region is that if variable x_n is out of the experimental region, the estimated response may be incorrect. So the regional constraints are needed. There are also some technical restrictions that can be expressed as linear equality constraints. In our example, we sometime request the linear combination of reaction time and reaction temperature hits the target T$_p$.

$$a_{p0} + \mathbf{a}_p{}^{\mathbf{T}} \mathbf{x} = T_p \tag{1.7}$$

, where

$$\mathbf{a}_p = \begin{pmatrix} a_{p1} \\ a_{p2} \\ \vdots \\ a_{pn} \end{pmatrix}$$

are the linear coefficients for the p-th linear equality constraint.

Here we give a summary for our problem, let *T* represent the set of responses with targets; *S* represent the set of responses with specification windows; and *Z* represent the set of linear equality constraints. The optimization problem can be formulated as:

$$\underset{\mathbf{x}}{Minimize} \quad f(\mathbf{x}) = \sum_i (b_{i0} + \mathbf{b}_i^{\,T}\mathbf{x} + \mathbf{x}^T\mathbf{B}_i\mathbf{x} - T_i)^2 \quad i \in T\,. \tag{1.8}$$

$$\begin{aligned} subject\ to: L_j \le \quad & b_{j0} + \mathbf{b}_j^{\,T}\mathbf{x} + \mathbf{x}^T\mathbf{B}_j\mathbf{x} \quad \le U_j; && j = 1,\dots,m_1 \in S \\ & a_{p0} + \mathbf{a}_p^{\,\mathbf{T}}\mathbf{x} = T_p; && p = 1,\dots,m_2 \in Z \\ & L_{x_q} \le x_q \le U_{x_q}. && q = 1,\dots,n \end{aligned}$$

This is a nonlinear minimization problem subject to linear equality constraints, and nonlinear inequality constraints. In particular, the objective function is a quartic programming problem with the objective function being a “quadratic” of “quadratic” and nonlinear inequality constraints being quadratic inequalities. Actually there are numerous non-linear programming (NLP) methods which can solve this problem, and these methods will be introduced in next section.

1.2 Current NLP Methods Review

For engineers, optimization is really a practical procedure. There are numerous NLP methods developed in recent half century. The most conventional class is “Primal Method” also called “Methods of Feasible Direction” [2, 11]. The following strategy is typical feasible direction algorithm. Given a feasible point **x**, a feasible direction **d** is generated by main algorithm and the step size λ is also determined.

Thus, these methods keep two properties (1) $\mathbf{x}+\lambda\mathbf{d}$ feasible, and (2) the objective value of current iteration smaller than last iteration. These methods usually have the following three advantages [2]:

1. Because these methods generate a feasible direction for minimizing the objective. Consequently the sequence of these points generated by these methods is feasible too.
2. If these methods generate a convergent sequence, the limit of the sequence will often satisfy the convergence prosperity, i.e., these methods are usually shown to converge to KKT solutions.
3. These methods are not limited to solve convex problem.

1.2.1 Generalized Reduced Gradient (GRG) Method

When dealing with linear constraint optimization, it is natural to add slack variables and use the linear equality constraints to eliminate some of the variables from the problem. Reduced Gradient method uses this idea and avoids the use of penalty parameter to search optimal solution. After that Generalized Reduced Gradient (GRG) method is developed for nonlinear constrained optimization problem. Today GRG is already verified to be a precise and accurate method for solving NLP problems. There are many commercial optimization software packages like LINGO, Microsoft Excel, Lotus and MINOS are all developed base on GRG.

The reduced gradient method can be viewed as the logical extension of the gradient method to constrained optimization problems. We start with linearly constrained optimization problems and consider the following linear equality constraint problem.

$$\begin{aligned} &\textit{Minimize} : f(\mathbf{x}) \\ &\textit{Subject to}: \ \mathbf{Ax} = \mathbf{b} \\ &\qquad\qquad\quad \mathbf{x} \geq 0 \end{aligned} \tag{1.9}$$

, where **A** is $m \times n$ matrix of rank m; **b** is m-vector.

There are some assumptions of this problem [2]:

1. f is continuously differentiable;
2. Every subset of m columns of the $m \times n$ matrix **A** is linearly independent;
3. Each extreme point of the feasible set has at least m positive components (non-degeneracy assumption).

Now let **x** be a feasible solution. The basic idea of reduced gradient method is dividing all variables into two sets, the set of basic variables $\mathbf{x}_B$ and nonbasic variables $\mathbf{x}_N$. For simplicity of notation we assume that we can partition the matrix **A** as **A** = [**B**, **N**] where **B** is an $m \times n$ invertible matrix. We partition **x** accordingly: $\mathbf{x}^T = [\mathbf{x}_B, \mathbf{x}_N]^T$. Thus we can rewrite **Ax** = **b** as the follows.

$\mathbf{Bx}_B + \mathbf{Nx}_N = \mathbf{b}$

, where

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N. \tag{1.10}$$

Now the basic variables $\mathbf{x}_B$ can be eliminated by (1.10), and then the problem will be

$$\begin{aligned}&\textit{Minimize}: f_N(\mathbf{x}_N)\\&\textit{Subject to:} \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N \geq 0.\\&\qquad\qquad \mathbf{x}_N \geq 0\end{aligned} \tag{1.11}$$

, where $f_N(\mathbf{x}_N) = f(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N, \mathbf{x}_N)$.

In (1.11), the variables we concerned are reduced to $\mathbf{x}_N$. If we have $\mathbf{x}_N$, we can obtain $\mathbf{x}_B$ by substituting $\mathbf{x}_N$ into (1.10). Now let's consider the choice of search direction. Suppose **d** is a feasible direction, by the definition **d** should satisfy the condition $\nabla f(\mathbf{x})^T \mathbf{d} < 0$. And then we also translate the condition into (1.9) by dividing $\nabla f(\mathbf{x})$ and **d** into two sets.

$$\nabla_{\mathrm{B}} f(\mathbf{x})^T \mathbf{d}_{\mathrm{B}} + \nabla_{\mathrm{N}} f(\mathbf{x})^T \mathbf{d}_{\mathrm{N}} < 0 \tag{1.12}$$

,where $\nabla_{\mathrm{B}} f(\mathbf{x})$ is the gradient with respect to the basic variables.

If d is a feasible direction, and then d satisfies the condition $\mathbf{A}\mathbf{d}=0$, i.e. $\mathbf{B}\mathbf{d}_B + \mathbf{N}\mathbf{d}_N = \mathbf{0}$.

This means $\mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{N}\mathbf{d}_N$. (1.13)

And then substitute (1.13) into (1.12) to yield:

$$\nabla f(\mathbf{x})\mathbf{d} = -\nabla f_B(\mathbf{x})^T \mathbf{B}^{-1}\mathbf{N} + \nabla f_N(\mathbf{x})^T \mathbf{d}_N < 0 \tag{1.14}$$

In (1.14), we call $\mathbf{r} = -\nabla f_B(\mathbf{x})^T \mathbf{B}^{-1}\mathbf{N} + \nabla f_N(\mathbf{x})^T \mathbf{d}_N$ the reduced gradient of f at **x** for the given basis **B**. In other words, the reduced gradient **r** plays the same role in the reduced problem as the gradient ∇f did in the original problem. In fact, the reduced gradient is exactly the gradient of the function f_N with respect to $\mathbf{x}_N$ in the reduced problem. Actually, the reduced gradient method can be generalized for solving nonlinearly constrained optimization problems by linearizing the nonlinear constraints. So we can solve the problem similarly to the linearly constrained case. The nonlinearly constrained problem with bounded constraints is express as follows.

$$\begin{array}{lll} \underset{\mathbf{x}}{Minimize} & f(\mathbf{x}) & for \quad \mathbf{x} \in E^n \\ subject\ to & g_{\tilde{i}}(\mathbf{x}) \le 0 & for \quad \tilde{i} = 1,\ldots,m \\ & \overline{L}_{\tilde{j}} \le x_{\tilde{j}} \le \overline{U}_{\tilde{j}} & for \quad \tilde{j} = 1,\ldots,n \end{array} \tag{1.15}$$

Given a nondegeneracy assumption, i.e., any columns of $\nabla \mathbf{h}(\mathbf{x})$ given by linearization inequality constraints are linear independent, a summary of Generalized Reduced Gradient method is given as follows [2]:

- Step 1:

Add slack variables to inequality constraints $g_{\tilde{i}}(\mathbf{x}) \le 0$ and obtain equality constraints $h_{\tilde{i}}(\mathbf{x}) = 0, \tilde{i} = 1,\ldots,m$. Let $\mathbf{x}^{(k)}$ be a feasible solution at the *k*-th search step. Linearize the constraints and get $\nabla \mathbf{h}(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) = 0$. Decompose variables into basic and nonbasic sets ($\mathbf{x}_{\mathrm{B}}^{(k)}, \mathbf{x}_{\mathrm{N}}^{(k)}$). Furthermore, the Jacobian matrix $\nabla \mathbf{h}(\mathbf{x}^{(k)})$ is decomposed into $\nabla \mathbf{h}(\mathbf{x}_{\mathrm{B}}^{(k)})$ and $\nabla \mathbf{h}(\mathbf{x}_{\mathrm{N}}^{(k)})$, such that $\nabla \mathbf{h}(\mathbf{x}_{\mathrm{B}}^{(k)})$ is invertible.

- Step 2:

Let $\mathbf{r}^T = \nabla_{\mathrm{N}} f(\mathbf{x}^{(k)})^T - \nabla_{\mathrm{B}} f(\mathbf{x}^{(k)})^T \nabla_{\mathrm{B}} \mathbf{h}(\mathbf{x}^{(k)})^{-1} \nabla_{\mathrm{N}} \mathbf{h}(\mathbf{x}^{(k)})$. Compute the vector $\mathbf{d}_{\mathrm{N}}$ whose $\tilde{j}^{th}$ component $d_{\tilde{j}}$ is

$$d_{\tilde{j}} = \begin{cases} 0 & \text{if } x_{\tilde{j}}^{(k)} = \overline{L}_{\tilde{j}} \text{ and } r_{\tilde{j}} > 0, \text{or } x_{\tilde{j}}^{(k)} = \overline{U}_{\tilde{j}} \text{ and } r_{\tilde{j}} < 0 \\ -r_{\tilde{j}} & \text{otherwise} \end{cases} \tag{1.16}$$

, where $r_{\tilde{j}}$ is the $\tilde{j}^{th}$ component of $\mathbf{r}$.

If $\mathbf{d}_{\mathrm{N}} = 0$, stop. $\mathbf{x}^{(k)}$ is a KKT point; otherwise, go to step 3.

- Step 3.1:

Find a solution to satisfy the nonlinear constraints by Newton-Raphson method. Choose $\underline{\varepsilon} > 0$ and a positive integer $\overline{T}$. Let $\theta > 0$ be such that $\overline{\mathbf{L}}_{\mathrm{N}} \le \tilde{\mathbf{x}}_{\mathrm{N}}^{(k)} \le \overline{\mathbf{U}}_{\mathrm{N}}$, where $\tilde{\mathbf{x}}_{\mathrm{N}}^{(k)} = \mathbf{x}_{\mathrm{N}}^{(k)} + \theta\,\mathbf{d}_{\mathrm{N}}$. Let $\mathbf{y}^{(1)} = \mathbf{x}_{\mathrm{B}}^{(k)}$ and $t = 1$.

- Step 3.2:

Compute $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \nabla_{\mathrm{B}}\mathbf{h}(\mathbf{y}^{(t)}, \tilde{\mathbf{x}}_{\mathrm{N}}^{(k)})^{-1}\mathbf{h}(\mathbf{y}^{(t)}, \tilde{\mathbf{x}}_{\mathrm{N}}^{(k)})$. If $\overline{\mathbf{L}}_{\mathrm{B}} \le \mathbf{y}^{(t+1)} \le \overline{\mathbf{U}}_{\mathrm{B}}$, $f(\mathbf{y}^{(t+1)}, \tilde{\mathbf{x}}_{\mathrm{N}}^{(k)}) < f(\mathbf{x}_{\mathrm{B}}^{(k)}, \mathbf{x}_{\mathrm{N}}^{(k)})$, and $\left\|\mathbf{h}(\mathbf{y}^{(t+1)}, \tilde{\mathbf{x}}_{\mathrm{N}}^{(k)})\right\| < \underline{\varepsilon}$, let $\mathbf{x}^{(k+1)} = (\mathbf{y}^{(t+1)}, \tilde{\mathbf{x}}_{\mathrm{N}}^{(k)})$ and go to step 1; otherwise, go to step 3.3.

- Step 3.3:

If $t = \overline{T}$, replace θ by $\theta/2$. Let $\tilde{\mathbf{x}}_{\mathrm{N}}^{(k)} = \mathbf{x}_{\mathrm{N}}^{(k)} + \theta\,\mathbf{d}_{\mathrm{N}}$ and $\mathbf{y}^{(1)} = \mathbf{x}_{\mathrm{B}}^{(k)}$. Replace t by 1 and repeat step 3.2. Otherwise, replace t by $t + 1$ and repeat step 3.2.

The contour in original space of the NLP problem is shown in Figure 1.2. In step 1, all inequality constraints are transformed into linearized equality constraints as shown in Figure 1.3. The selected basic variables are x_1 and x_2 and the selected nonbasic variables are x_3 and x_4. In step 2, the original NLP problem is transformed into a NLP problem without equality constraints in the reduced space of nonbasic variables, x_3 and x_4. The improving direction of the nonbasic variables is the opposite direction of the reduced gradient. However, the variables should not be negative. The improving direction of the nonbasic variables is modified as shown in Figure 1.4. In step 3, the improving direction after transformed into the original space is actually the direction along the tangent of the binding constraint. The optimal solution is then found through Newton-Raphson method as shown in Figure 1.5.

Example 1.1:

$$\underset{\mathbf{x}}{Minimize}: f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2,$$

$$\begin{aligned} subject\ to: \ & x_1 + x_2 \le 2; \\ & x_1 + \frac{17}{6}x_2^2 \le 2.8; \\ & x_1, x_2 \ge 0. \end{aligned}$$

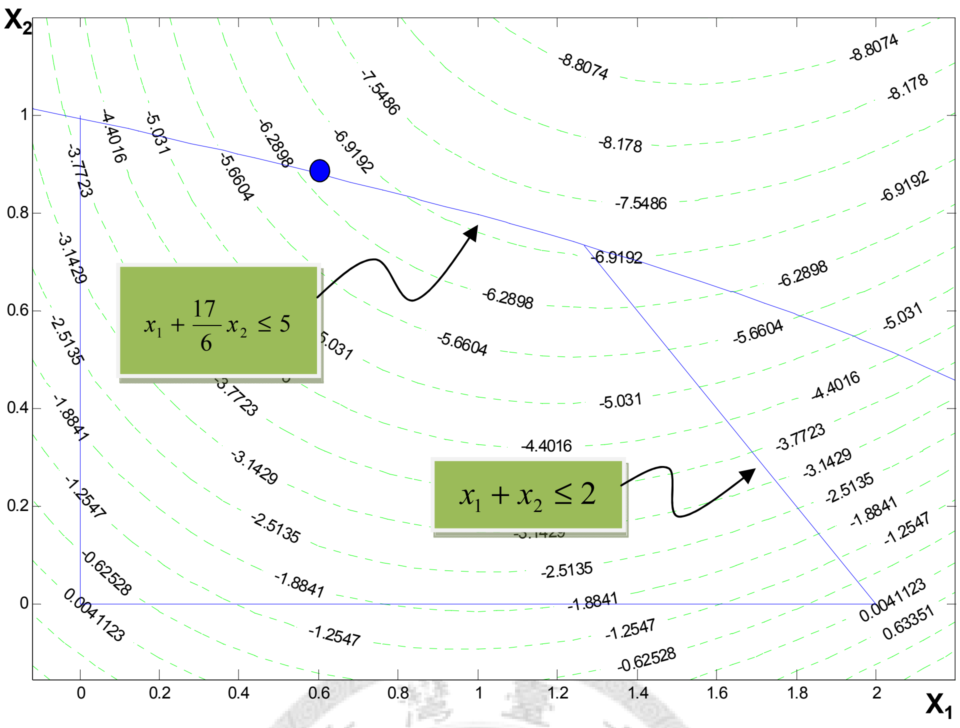


Figure 1.2 Feasible Region of Example 1.1 with the nonlinear constraints

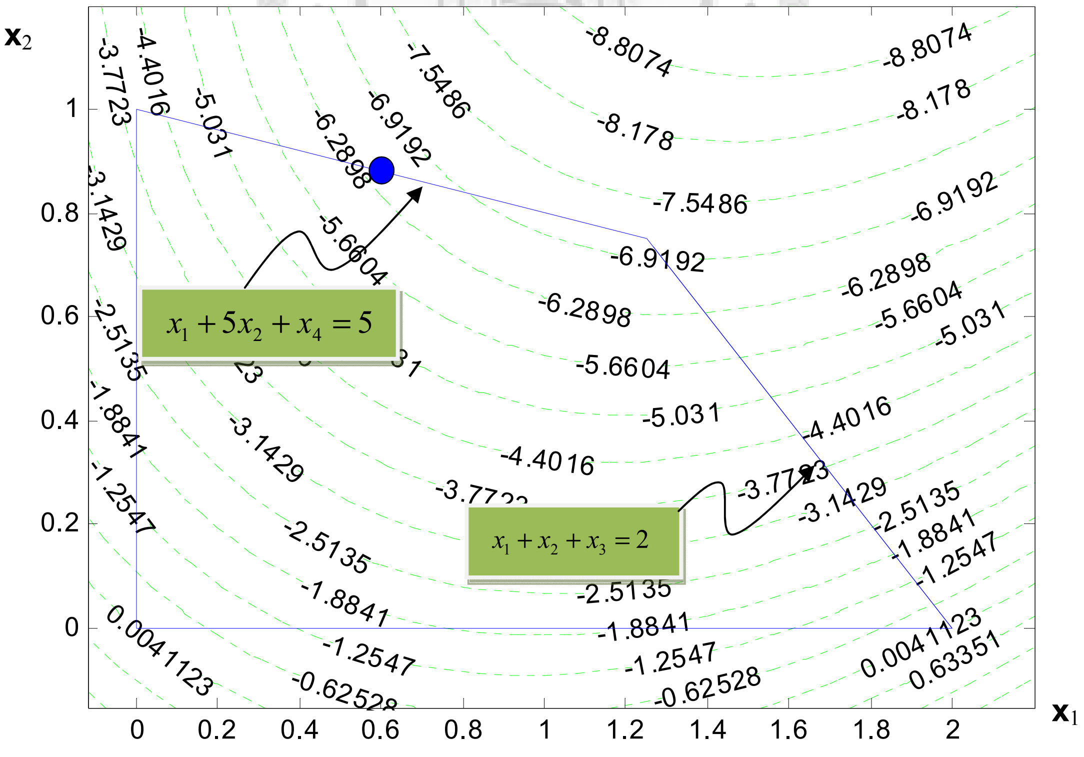


Figure 1.3 Feasible Region of Example 1.1 with the linearized constraints

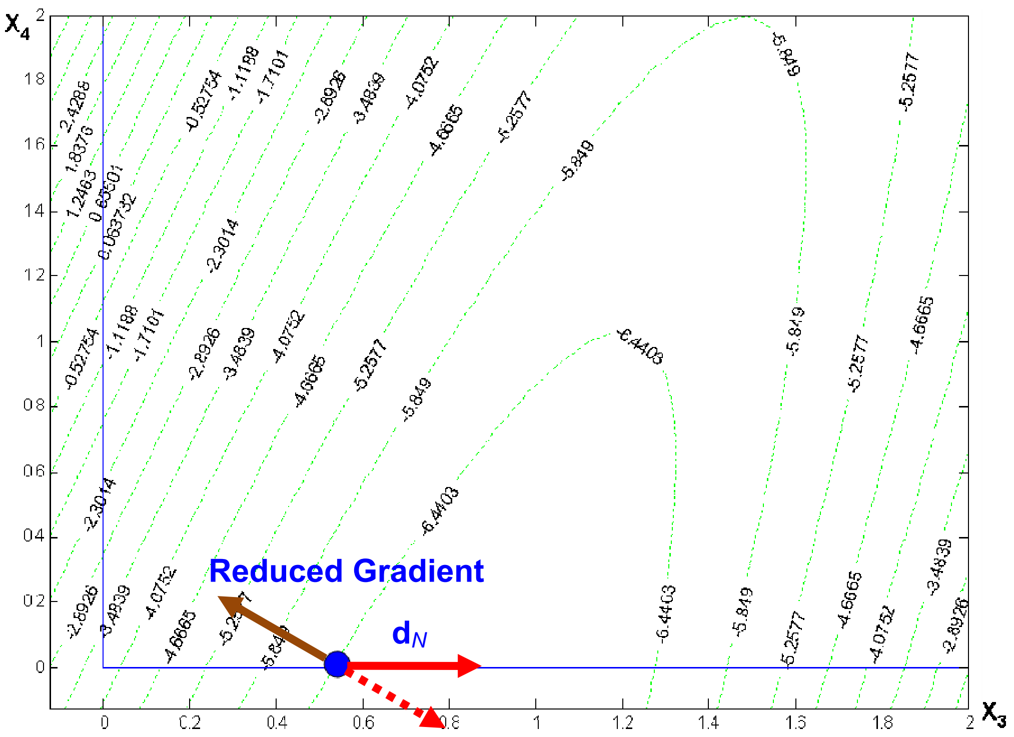


Figure 1.4 The Example 1.1 in reduced space of nonbasic variables

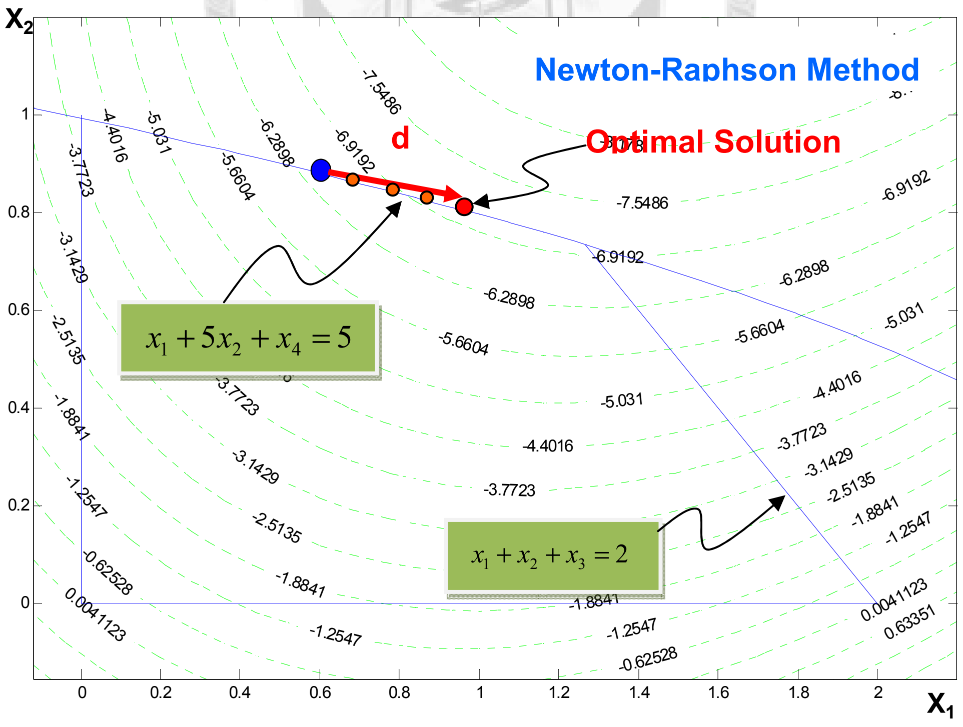


Figure 1.5 Newton-Raphson method in the original space

1.2.2 Ridge Analysis and Ridge Search Method

In RSM, ridge analysis is a method for exploring optimal factor levels of a response surface. Ridge analysis helps us to find maximum or minimum a quadratic response surface under a spheral constraint. The purpose of spheral constraint is to fixed distances from the center of the experimental region. Due to the formulation ridge analysis, it is a nonlinearly constrained optimization problem.

The concept of ridge analysis is finding an absolute minimum or maximum on the spheral constraint of a certain radius you trusted. Additionally, we can adjust the radius to increase the sphere size if the point on the spheral constraint is still inside the experimental region. In other words we can find an optimum corresponding to a distinct radius, all optimums with various radiuses construct a "ridge path" as shown in Figure 1.6. In fact, control radius of the region is hard in ridge analysis. We discuss the issue in the following subsection.

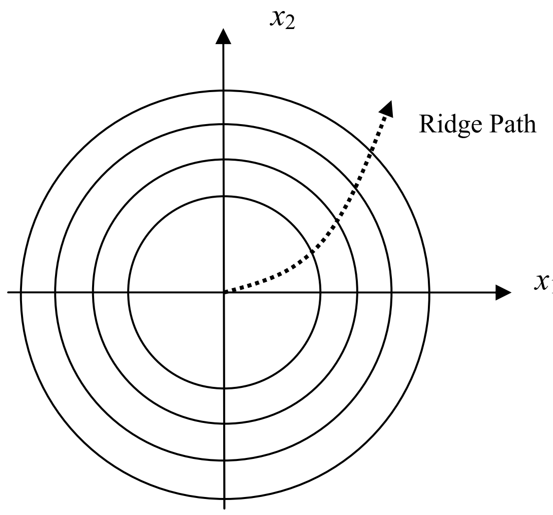


Figure 1.6 Ridge path of all stationary points with various radiuses

Consider the following problem.

$$\begin{aligned} &\textit{Minimize } (\textit{Maximize})\ \hat{y} = b_0 + \boldsymbol{\beta}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} \\ &\qquad\quad \textit{subject to }\ \mathbf{x}^T\mathbf{x} = \Delta^2 \end{aligned} \tag{1.17}$$

, where **x** is an *n*-vector; **G** is the matrix contains quadratic coefficients; $\boldsymbol{\beta}$ is a vector expressed first order coefficients.

Under the problem formulation, global constrained optima are typically obtained using the Lagrangian multiplier approach. By introducing the Lagrangian multiplier μ, the problem will be an unconstrained optimization problem.

$$\textit{Minimize } (\textit{Maximize})\ L = \hat{y} + \mu(\mathbf{x}^T\mathbf{x} - \Delta^2) \tag{1.18}$$

Differentiate (1.18) with respect to **x**, and then set the derivative equal to zero. The equation which includes a stationary point $\hat{\mathbf{x}}_s$ will hold:

$$\boldsymbol{\beta} + (\mathbf{G} + \mu\mathbf{I})\hat{\mathbf{x}}_s = 0 \tag{1.19}$$

Given a fixed value of μ, the stationary point $\hat{\mathbf{x}}_s$ on the sphere with radius Δ can be estimated to be:

$$\hat{\mathbf{x}}_s = -(\mathbf{G} + \mu\mathbf{I})^{-1}\boldsymbol{\beta} \tag{1.20}$$

Theoretically, there are totaling n+1 equation, namely, the spherical constraint in (1.17) and (1.18) let us solve $\hat{\mathbf{x}}_s$ and μ. Practically the radius Δ is also unknown, i.e. there are n+2 unknown variables. That is we can't solve $\hat{\mathbf{x}}_s$ directly. So ridge analysis considers the following strategy to solve the problem.

1. Regard Δ as variable, but fix μ instead.
2. Choose μ as a fixed value and substitute μ with the fixed value into (1.19) to obtain $\hat{\mathbf{x}}_s$.
3. Evaluate $\hat{y}$ by (1.17)

Even if we have the above strategies, there is still a problem that is how to choose μ. Providentially, there are some properties of ridge analysis help us choose μ appropriately. These properties are described as follows [7]:

1. At $\mu = -\infty \text{ or } \infty$ then Δ = 0 and Δ increases exponentially to infinity at $\mu = \lambda_i$.
2. If we wish to find the ridge path as Δ varies, we can substitute any value of μ larger than $-\lambda_1$.
3. As Δ increases, $\hat{y}$ passes through the ridge path toward a minimum.

The value μ determines the radius Δ, i.e., Δ is a function of μ. Figure 1.7 shows the relation between radius and Lagrangian multiplier. $\lambda_1,\ldots,\lambda_k$ are the eigenvalues of the **G**, and $\lambda_k > \lambda_{k-1} > \ldots > \lambda_1$.

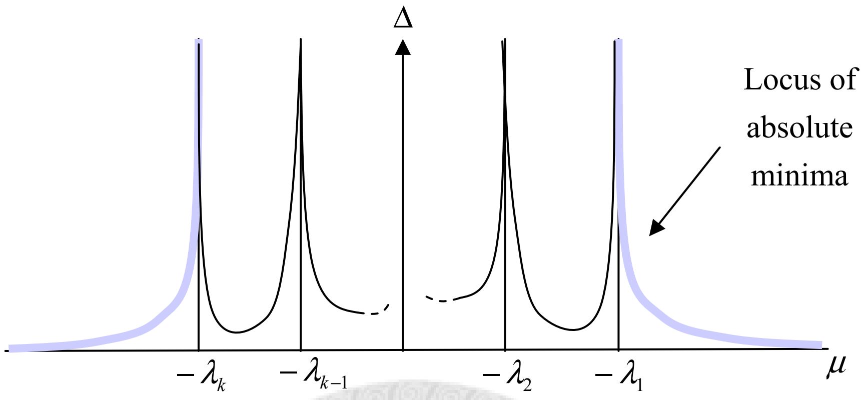


Figure 1.7 The dependence of radius on Lagrangian multiplier

We consider the following example to show you the relationship between Δ and μ.

Example 1.2:

$$\textit{Minimize } y = 15 + 2x_1 + x_2 + \frac{1}{2}x_1^2 + 2x_1x_2 + \frac{1}{2}x_2^2$$
$$\textit{subject to } \sqrt{x_1^2 + x_2^2} = \Delta$$

Express in matrix notation.

$$\textit{Minimize}: \ y = 15 + \boldsymbol{\beta}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix} \mathbf{G} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$\textit{subject to}: \sqrt{\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}} = \Delta$$

, where $\boldsymbol{\beta} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$; $\mathbf{G} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$; the eigenvalues of $\mathbf{G}$ are -1 and 3; the corresponding eivenvectors $[-1, 1]^T$ and $[1, 1]^T$.

Introduce Lagrangian multiplier μ and then we have

$$\tilde{\mathbf{x}}_s = -(\mathbf{G} + \mu\mathbf{I})^{-1}\boldsymbol{\beta} = -\begin{bmatrix} 1+\mu & 2 \\ 2 & 1+\mu \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{-2\mu}{-3+2\mu+\mu^2} \\ -\dfrac{-3+\mu}{-3+2\mu+\mu^2} \end{bmatrix} \tag{1.21}$$

We also have

$$\begin{aligned} \Delta &= \sqrt{\tilde{\mathbf{x}}_s^T \tilde{\mathbf{x}}_s} \\ &= \sqrt{\begin{bmatrix} \dfrac{-2\mu}{-3+2\mu+\mu^2} \\ -\dfrac{-3+\mu}{-3+2\mu+\mu^2} \end{bmatrix}^T \begin{bmatrix} \dfrac{-2\mu}{-3+2\mu+\mu^2} \\ -\dfrac{-3+\mu}{-3+2\mu+\mu^2} \end{bmatrix}} \\ &= \sqrt{\frac{9-6\mu+5\mu^2}{(-3+2\mu+\mu^2)^2}} \end{aligned} \tag{1.22}$$

That is Δ is a function of μ, figure 1.8 shows the relationship on a two dimension space.

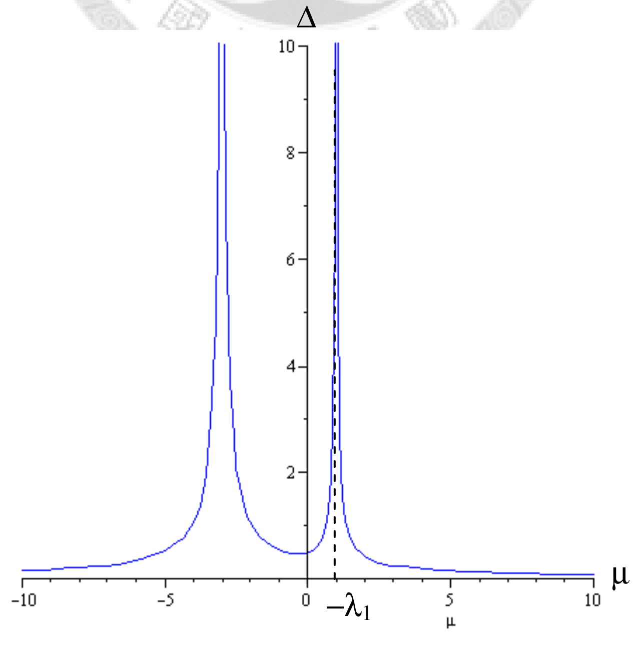


Figure 1.8 The dependence radius against Lagrangian multiplier

In (1.22), the denominator in the radical has two factors. We can factorize the denominator as:

$$-3+2\mu+\mu^2=(\mu+3)(\mu-1) \tag{1.23}$$

(1.22) also implies that if μ is equal to the subtractive eigenvalue of **G**, i.e. −3 or 1, the denominator is close to 0. Therefore Δ goes to positive infinity. On the other hand, if μ goes to positive or negative infinity, thus the denominator is close to positive zero, i.e., Δ goes to zero.

1.2.3 Zoutendijk Method

Zoutendijk's method searches a feasible improving direction. Compared with the GRG method, the direction may be less effective. To consider (1.15) (a minimization problem), if the direction is an opposite direction to objective function's gradient, it is an improving direction. Moreover, if the direction is an opposite direction to binding constraint's gradient, it is a feasible direction. Zoutendijk's method solves a linear program to generate a direction satisfying the above two requirements; to improve and to be feasible.

Zoutendijk's method is described as follows:

- Step 1:

Let $\mathbf{x}^{(k)}$ be a feasible solution at the *k*-th search step. Check the binding nonlinear constraints. Let $\widetilde{W} = \{ \widetilde{w} \ : \ g_{\widetilde{w}}(\mathbf{x}^{(k)}) = 0 \ \}$. Compute the gradients of the objective function and the binding constraints $\nabla f(\mathbf{x}^{(k)})$ and $\nabla g_{\widetilde{w}}(\mathbf{x}^{(k)})$ with respect to $\mathbf{x}^{(k)}$.

- Step 2:

Solve the following linear program with decision variables $\mathbf{d}_Z$ and z:

$$\underset{z,\,\mathbf{d}}{Minimize}: z\,.$$

$$\begin{aligned} subject\ to: &\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}_Z - z \le 0; \\ &\nabla g_{\widetilde{w}}(\mathbf{x}^{(k)})^T \mathbf{d}_Z - z \le 0 \quad \forall\, \widetilde{w} \in \widetilde{W}; \\ &-1 \le d_j \le 1 \quad j = 1 \ldots n. \end{aligned}$$

Let (z^*, $\mathbf{d}_Z^*$) be the optimal solution. If $z^* = 0$, stop; $\mathbf{x}^{(k)}$ is an optimal point. Else, go to the step 3.

- Step 3:

Do Line Search along $\mathbf{d}_Z^*$. Let the feasible solution be $\mathbf{x}^{(k+1)}$. Return to step 1.

Here we also use the example 1.1 to introduce Zoutendijk's method. Suppose that we start from iteration with current solution (x_1, x_2) = (0.5889, 0.8833) is also binding on the second constraint in example 1.1. The gradients of objective function and constraint are described as follows:

$$\nabla f(x_1, x_2)^{\mathrm{T}} = (-3.4110, -3.6442)$$

, and

$$\nabla g_{(1.11)}(x_1, x_2)^{\mathrm{T}} = (1,5).$$

Now we consider the following linear programming problem:

$$\underset{z,\,\mathbf{d}}{Minimize} : z \,.$$

$$\begin{aligned} subject\ to : & (-3.4110, -3.6442)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} - z \le 0; \\ & (1,5)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} - z \le 0; \\ & -1 \le d_1 \le 1; \\ & -1 \le d_2 \le 1. \end{aligned}$$

By performing simplex method, we can obtain the improving direction **d** is (d_1, d_2) = (1, −0.5102). We sketch the direction in Figure 1.9. We can see that the next solution can leave the binding constraint by performing line search along Zoutendijk's direction. This will help us develop the main search algorithm of this thesis. It is detailed in chapter 3.

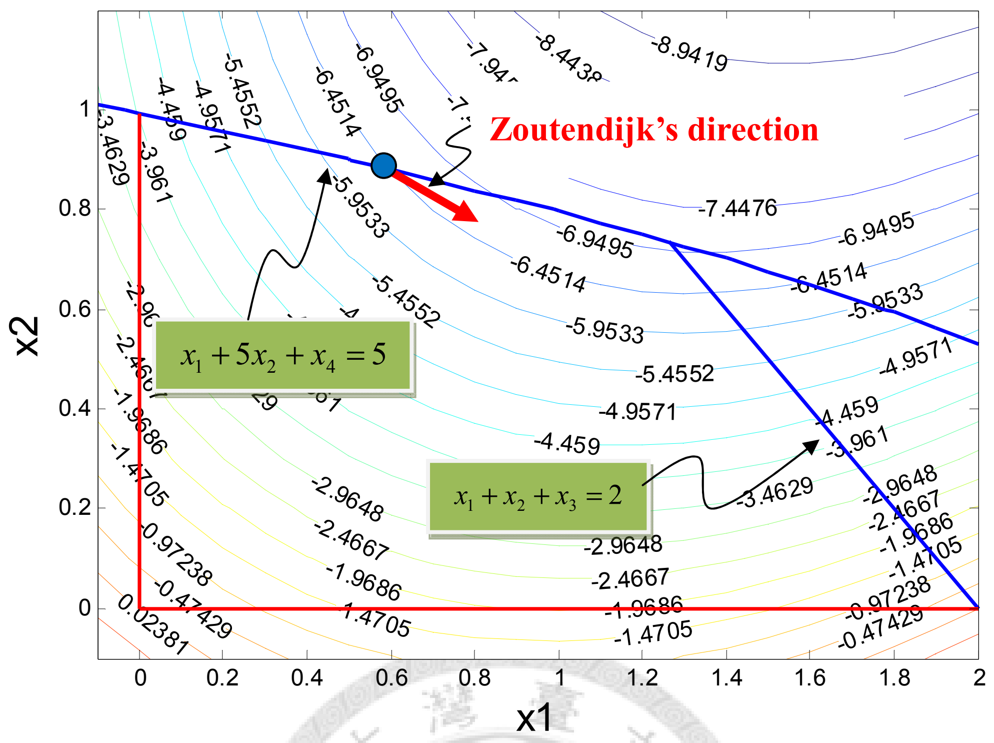


Figure 1.9 Zoutendijk's direction of Example 1.1.

1.3 Shortcomings of Current NLP Methods

1.3.1 Shortcomings of Generalized Reduced Gradient Method

One motivation of this study is to overcome some unexpected phenomenon rose by the GRG method, although the GRG method is applied intensively in practice. The phenomenon is called "zigzagging" or "jamming". Zigzagging usually appears at the later phase of search and causes a poor convergence. As mentioned earlier, the GRG method employs the first-order approximation: $f(\mathbf{x}^{(k)} + \lambda\mathbf{d}) = f(\mathbf{x}^{(k)}) + \lambda\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}$ $+e$, where **d** is the search direction, e is the error of the linearization approximation. When $\mathbf{x}^{(k)}$ is close to the stationary point, $\left\|\nabla f(\mathbf{x}^{(k)})\right\|$ becomes very small so is the term

$\lambda \nabla f(\mathbf{x}^{(k)})^T \mathbf{d}$. The error term, thus, becomes relatively significant. The error term caused by the linearization approximation thus induces the search path to zigzag.

For example the response surface resembling an inclined trough, will cause the GRG method to zigzag easily. That is, the search direction of the GRG method, i.e., the reduced gradient, moves toward the bottom of the trough, not to the inclined direction. The number of moving steps will be enormous, and it becomes difficult to reach the optimal point. The objective function of the SMOO problem in (1.8) could certainly form an inclined trough and cause the zigzagging problem.

As described earlier, the quartic objective function in SMOO problem is the "the quadratic of the quadratic". If the response, expressed as a quadratic function of input variables, isn't absolutely positive or negative, the quartic objective function could form a trough. For an example with two input variables, the quadratic function, $z = 10x^2 + 10y^2 - 4$, will exhibit a shape as shown in Figure 1.10. After squaring the quadratic function, a trough ring will be created. Figure 1.11 shows the quartic response surface. Because in a typical SMOO problem, the objective function is the sum of multiple quartic functions, a trough will be easily formed and cause the zigzagging problem.

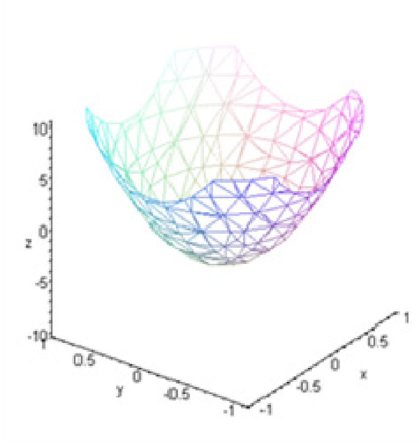


Figure 1.10 Quadratic response surface

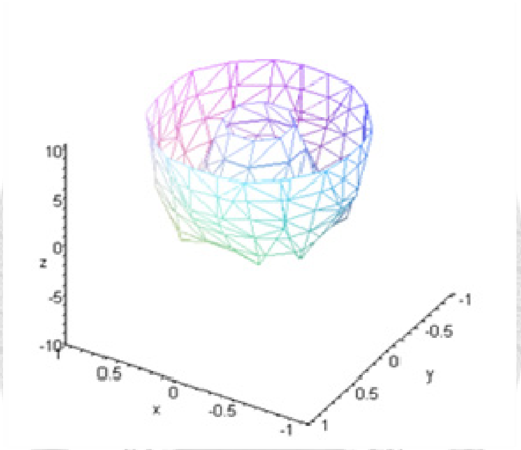


Figure 1.11 Quartic response surface

Example 1.3 is a well known problem for testing optimization algorithm. The objective function is called Rosenbrock's Function or Rosenbrock's valley. The function is a summation of a square of quadratic function and a square of a linear function. The value of this function thus must be greater or equal to zero. Because of the global minimum f (x1, x2) =0 at (x1=1, x2=1) is inside a long, narrow and inclined trough. To find the valley is trivial, however to converge to the global minimum is difficult. The function form is expressed in the following equation and figure 1.12 shows the corresponding surface and contour plot.

Example 1.3:

Minimize: $(1-x_1)^2+100\times(x_2-x_1^2)^2$,

Subject to: $-2 \le x_1 \le 2;\ 0 \le x_2 \le 4$.

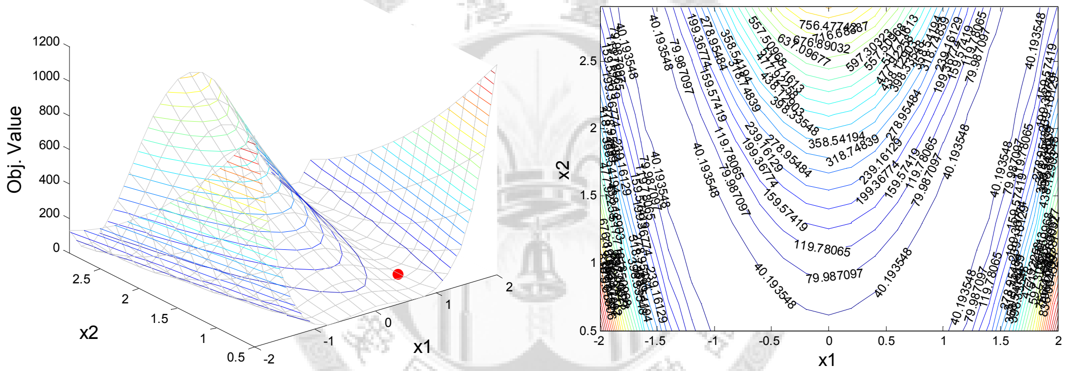


Figure 1.12 Surface plot and contour map of Example 1.3

Suppose the initial point is $(x_1, x_2) = (-2, 0.5)$ with the terminal criterion to be 10^{-10} and the maximal iteration number to be 3000. The objective value of the second iteration is 0.0173683 with $(x_1, x_2) = (1.1316, 1.2812)$. The objective value of the latest iteration is 0.000657 with $(x_1, x_2) = (1.0256, 1.0520)$. There are 2417 iterations in total from the second iteration to the latest iteration. From the result, there are two major drawbacks of GRG method. The search path of the GRG is sketched on the

Figure 1.13. Figure 1.15 shows the zigzagging phenomenon of the GRG method. Due to zigzagging phenomenon, the GRG method sometimes is unable to converge to the global minimum $(x_1, x_2) = (1, 1)$.

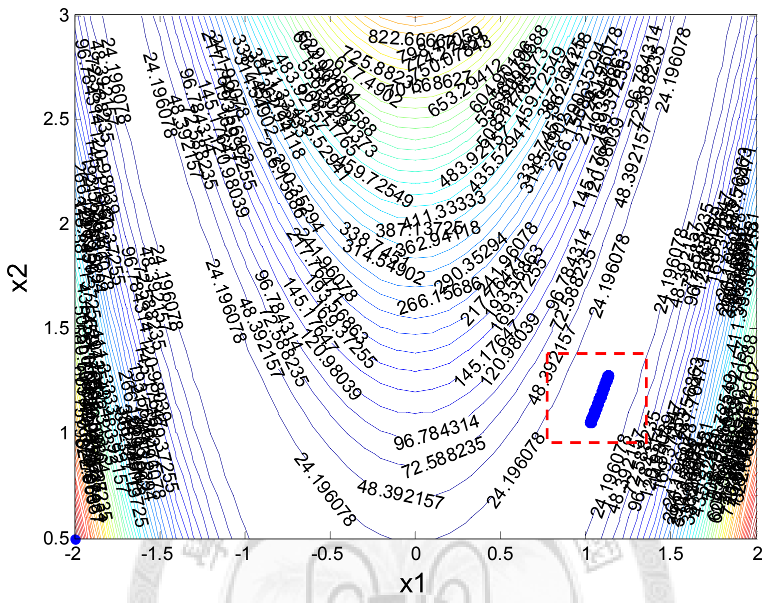


Figure 1.13 Search path of GRG in Example 1.3

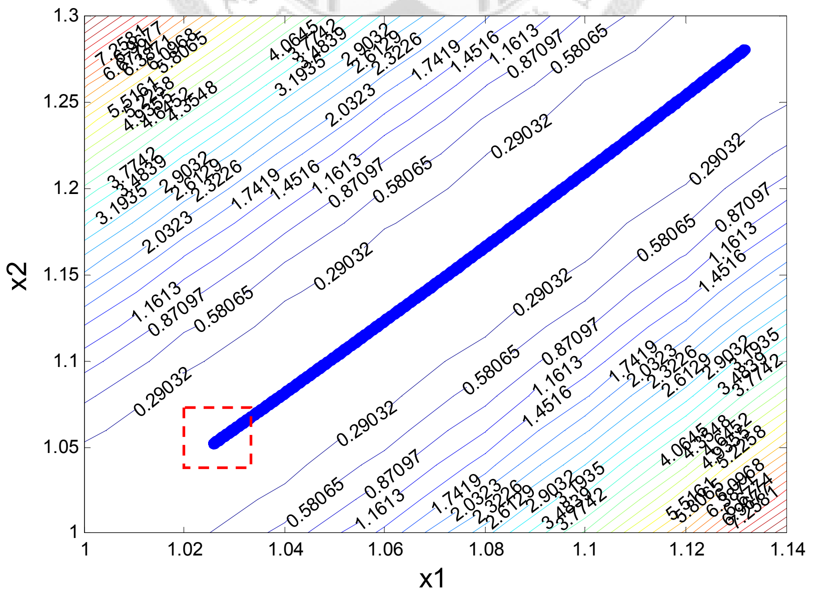


Figure 1.14 Dash-line region in Figure 1.13

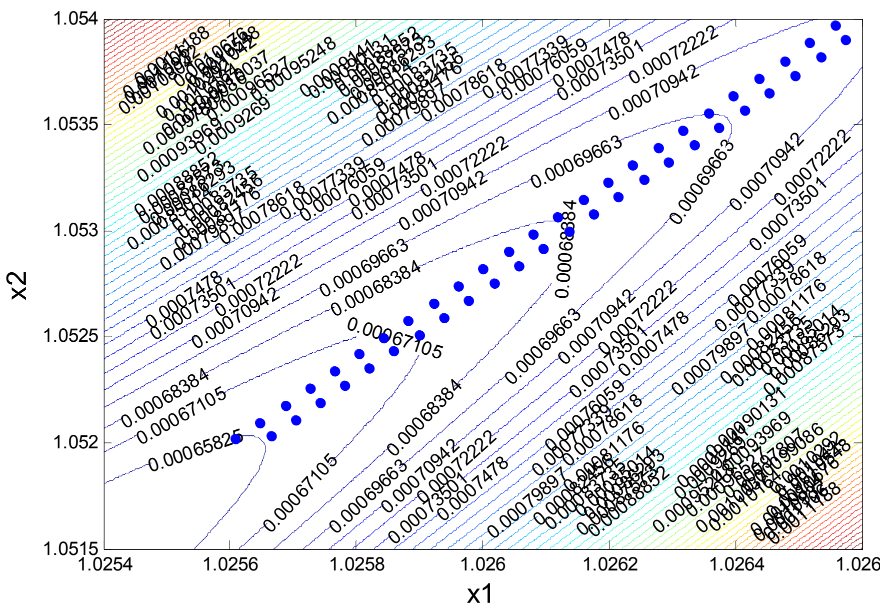


Figure 1.15 Zigzagging path in dash-line region of Figure 1.14

1.3.2 Shortcomings of Ridge Search Method

Although the ridge analysis helps us to find the minimum without zigzagging phenomenon, the required optimal Lagrangian multiplier is difficult to find or is inefficiently found. What is known is that the optimal Lagrangian multiplier should be smaller than the smallest eigenvalue of the quadratic coefficient matrix **G** if we want to minimize the objective function. The RS search uses the following formula to search for the optimal Lagrangian multiplier to calculate the stationary point and obtain the corresponding objective value. The updating formula of Lagrangian multiplier is:

$$\mu^{(\gamma+1)} = \mu^{\gamma} - \Delta \times \alpha^{\gamma} \tag{1.24}$$

, where γ is the search step index; Δ is the step size and is set to be proportional to the smallest eigenvalue and α is the parameter to approximate the exponential relationship between the radius and μ. When we use (1.24) to search the optimal Lagrangian multiplier, there exist three drawbacks. In (1.24) there are two manipulatable parameters Δ and α, the search result of the GRR algorithm is in fact quite sensitive to these two parameters. This is also an important reason motivating us to develop a new algorithm with less parameter settings.

Considering Example 1.3, with an initial point $(x_1, x_2) = (-2, 0.5)$, we first set α =10 and Δ=100 and then change the setting to α=100 and Δ=100. Figure 1.16, Figure 1.17 and Table 1.1 show the search processes and the comparisons including objective value, iterations and computing time under the two different settings.

Table 1.1 Comparison of two different settings

Settings	Obj. Value	Number of Iterations	Computing time
1st	3.368948E-07	181	0.06
2nd	1.100042E-10	976	0.30

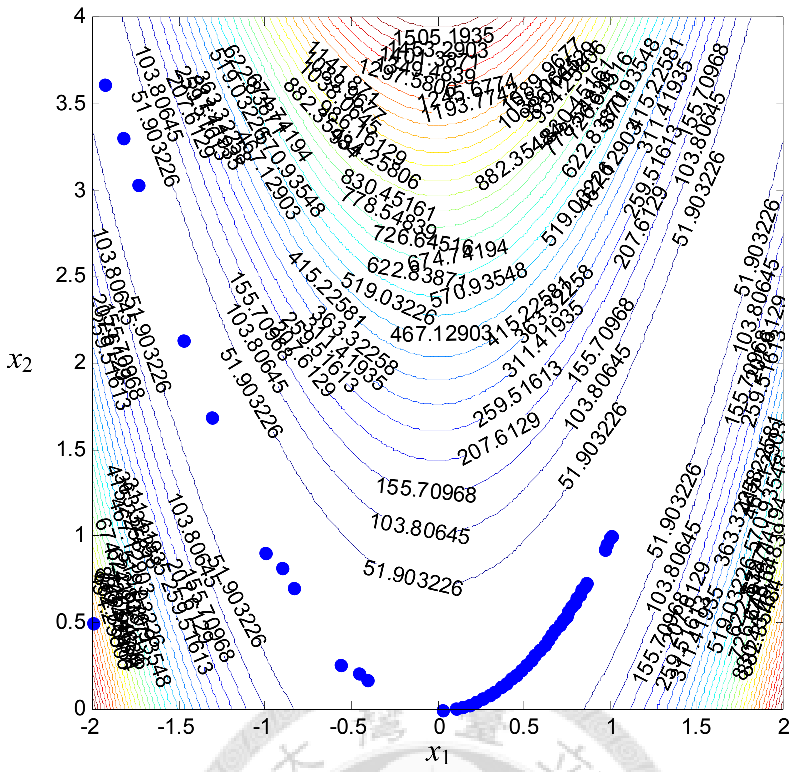


Figure 1.16 Search process of 1st setting

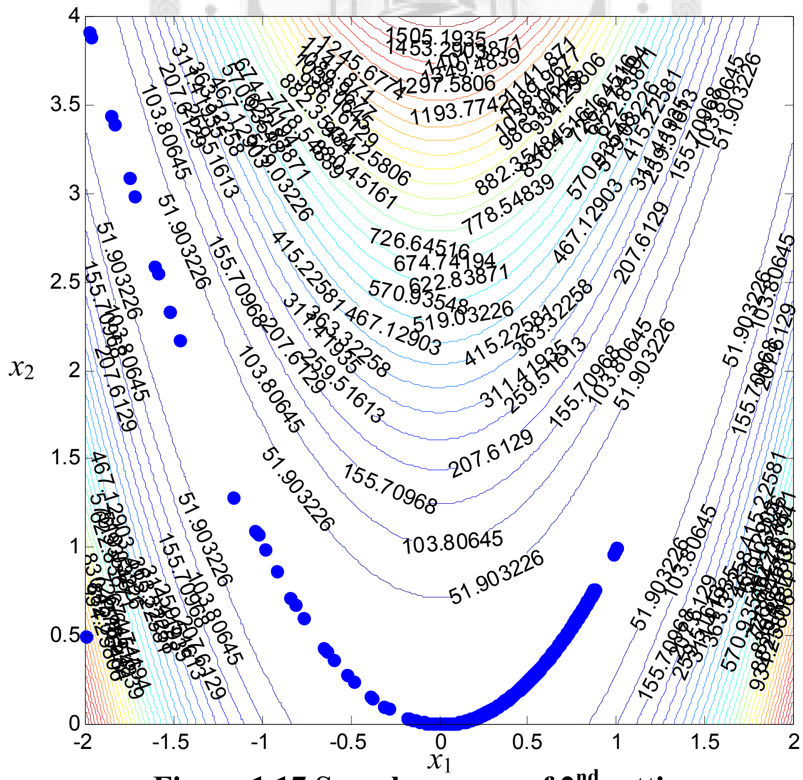


Figure 1.17 Search process of 2nd setting

Although the GRR search method can find the near-optimal objective value by different setting, but the difference between the two settings is large, and the computing time is also greatly depending to the setting. That is, the first drawback of the GRR search is the difficulty of parameter selection. The second drawback is due to some issues of numerical calculation. For a large SMOO problem, the quadratic coefficient matrix **G** is easy to be singular or near singular with the smallest eigenvalue near zero. Under the circumstances, the Δ, set to be proportional to the smallest eigenvalue, in (1.24) is also near zero and cause the search to be extremely slow. Finally, the GRR uses (1.24) to approximate the relationship between the radius and the Lagrangian multiplier. This sometimes results in an over-large radius is large sometimes. In fact, the objective function of the SMOO problem is a quartic function. In order to perform the GRR search, the algorithm needs to approximate the objective function to second-order function by the Taylor series expansion. If the radius is too large, the solution solved by GRR may not in the region of trusted approximate. This is the third drawback of the GRR algorithm.

1.4 Research Objectives

The formulation of the statistical multi-objective optimization (SMOO) problem is exactly a nonlinear programming problem (NLP) with nonlinear inequality and linear equality constraints. So this thesis will focus on developing a constrained optimization algorithm for solving the quadratic programming NLP problem. Furthermore, the objective function of SMOO problem is a quartic function and is not guaranteed to be a convex function. This is the first challenge we need to face. On the other hand, there are three drawbacks of GRR we discussed in subsection 1.3.2. we then attempt to develop a new algorithm that prevents the three drawbacks of the GRR search.

There are some commercial optimization softwares, such as “Lingo”, adopts “Generalized Reduced Gradient method” together with “Successive Linear Programming method” in its algorithm. The two methods used by Lingo are actually Feasible Direction Methods. These methods are also subject to the zigzagging. Since one of our research objectives is to avoid zigzagging, our research results will be compared to Lingo’s to validate the proposed algorithm. To Summary, our research objectives are to develop a constrained optimization algorithm for solving the SMOO problem and this algorithm must (1) overcome the three drawbacks of GRR and (2) avoid the zigzagging phenomenon.

In specific, there are four research objectives:

1. Develop a nonlinear constrained optimization algorithm called Generalized Reduced Trust-Region (GRT) search method based on trust-region method.
2. Develop a algorithm using the developed GRT method and the Zoutendijk's method.
3. Propose the convergence proof of GRT search algorithm
4. Test the proposed search algorithm with four cases: (1) A well-known test problem for NLP algorithm called Rosenbrock's function, (2) Geometric Layout Design for Semiconductor Manufacturability, (3) Robust Configuration of Semiconductor Supply Chain, (4) Track System PED CDU Optimization.

1.5 Thesis Organization

In this chapter, we describe the background, problem definition, current methodology review, and drawbacks of these NLP algorithm and the research objectives. Chapter 2 introduces the trust-region method and subproblem of the trust-region method. Moreover, the hard case of the trust-region method will be also mentioned in this chapter. Finally we do some modification of the traditional trust-region algorithm is also be introduced here. In chapter 3, we describe the algorithm of generalized reduced trust region method. The convergence proof of GRT is also proposed in this chapter. In chapter 4, the test problem and result will be presented. Every result will be compared against Lingo's result. Finally, some conclusions and suggestions are presented in Chapter 5.

2 Trust Region Method

Due to the drawbacks of the GRG method and the GRR method, this research develops an algorithm based on a method known in numerical optimization are "Trust Region" (TR) method. In Section 2.1, the basic ideas and the problem formulation of the TR method will be introduced. In Section 2.2, we study an algorithm to help us solve the "Trust Region Subproblem" (TRS). Some numerical issues called "Hard Case" of the TR method in the literature will be discussed in Section 2.3. Finally, we make some modifications to the TR method to improve its numerical implementation in Section 2.4.

2.1 Trust Region Method

The TR method and the Ridge Analysis (RA), in effect, share the same mathematical formulation, i.e., minimizing or maximizing a quadratic function subject to a spheral region constraint. The quadratic function can be an approximation of any objective function. For example, we can approximate the quartic SMOO problem to a quadratic function and solve it by the TR method or the RA method. Though the problem formulation is the same, there are still fundamental differences between two methods. First, the TR method finds a solution inside the spheral region, while the RA method only considers the boundary solutions. Second, as we discussed in Subsection 1.2.2, the RA method regards radius of the spheral region as a variable and makes guess on the value of the Lagrangian multiplier iteratively by (1.24). In contrast, the TR method finds the optimal Lagrangian multiplier directly by solving a sequence of Trust Region Subproblems (TRS). This method for TRS is discussed in the next section. The TR method allows us to adjust the radius directly without guessing on the value of the Lagrangian multiplier.

Determination of the trust region radius with TR method is critical. If the radius of the region is too small, the algorithm misses the chance to move faster to a minimum of the objective function. If it's too large, the approximated model may become a poor approximate of the objective function and the minimum found inside the region may be far from the global minimum. Thus the TR algorithm gradually shrinks the size of the region in its search steps. In every iteration, the algorithm uses the approximate performance of the previous iterate to determine radius of the trust region. If the approximation is good, we enlarge the size else we shrink the size of the trust region. Such update of the trust region radius is introduced in next Chapter. Figure 2.1 shows the TR approach for a function *f* of two variables on a contour plot. The contour of quadratic model function ϕ (in dashed line) is constructed from the derivative information at the current iterate $\tilde{\mathbf{x}}_0$

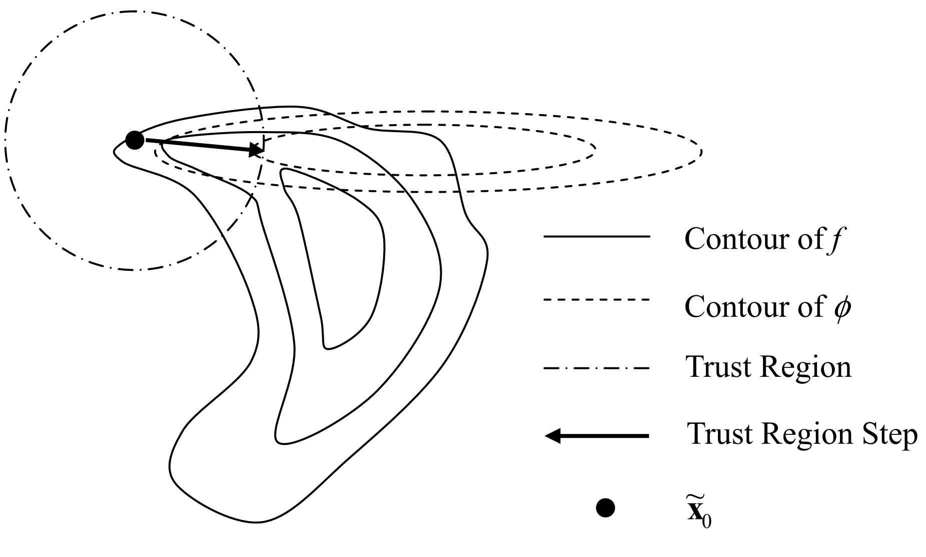


Figure 2.1 Trust-Region and Trust-Region step

The TR method approximates any differentiable function to a quadratic function by the Taylor series expansion. Consider the following TR problem

$$\begin{aligned}&\mathit{Minimize}: f(\mathbf{x}) + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} \\ &\mathit{Subject\ to}: \|\mathbf{x}\| \le \Delta,\end{aligned} \tag{2.1}$$

where $\|\cdot\|$ is the Euclidean norm; $\mathbf{g}^T$ is the gradient vector, i.e., $\nabla f(\mathbf{x})$; $\mathbf{H}$ is the Hessian Matrix, i.e., $\nabla^2 f(\mathbf{x})$; and Δ is the radius of the trust region.

Now considering the SMOO problem, the objective function of (1.8) is a quartic function. We approximate the objective function of the SMOO problem with respect to a given point $\mathbf{x}^{(k)}$ by the second-order approximation and apply the TR method as follows:

$$\underset{\mathbf{x}^{(k+1)}}{\mathit{Minimize}} : f(\mathbf{x}^{(k)}) + \boldsymbol{\beta}^{(k)^T}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) + (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T \mathbf{G}^{(k)} (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})/2;$$

$$\mathit{subject\ to}: \left\| \left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \right) \right\| \le \Delta^{(k)} \tag{2.2}$$

, where the superscript (k) denotes the *k*-th iteration index; the vector $\mathbf{x}^{(k+1)}$ is the minimizer of (2.2), i.e., $\mathbf{x}^{(k+1)}$ minimizes the (2.2) at a given point $\mathbf{x}^{(k)}$; the $\Delta^{(k)} > 0$ denotes the trust region radius at current iteration; the partial derivative matrix $\boldsymbol{\beta}^{(k)} = \nabla f\left(\mathbf{x}^{(k)}\right)$ and the Hessian matrix $\mathbf{G}^{(k)} = \nabla^2 f\left(\mathbf{x}^{(k)}\right)$ are calculated with the following formulas:

$$\begin{aligned}
\boldsymbol{\beta} &= \nabla f(\mathbf{x}) \\
&= \frac{\partial \sum_i \left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T_i\right)^2}{\partial \mathbf{x}} \\
&= \sum_i \left[2\left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T_i\right) \frac{\partial\left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T_i\right)}{\partial \mathbf{x}} \right] \\
&= \sum_i \left[2\left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T_i\right)\left(\mathbf{b}_i + 2\mathbf{B}_i \mathbf{x}\right) \right]
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{G} &= \nabla^2 f(\mathbf{x}) \\
&= \frac{\partial}{\partial \mathbf{x}} \left[\frac{\partial \sum_i \left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T_i\right)^2}{\partial \mathbf{x}} \right] \\
&= \frac{\partial \sum_i \left[2\left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T\right)\left(\mathbf{b}_i + 2\mathbf{B}_i \mathbf{x}\right) \right]}{\partial \mathbf{x}} \\
&= \sum_i 2\left[\left(\mathbf{b}_i + 2\mathbf{B}_i \mathbf{x}\right)\left(\mathbf{b}_i + 2\mathbf{B}_i \mathbf{x}\right)^T + \left(b_{i0} + \mathbf{b}_i^T \mathbf{x} + \mathbf{x}^T \mathbf{B}_i \mathbf{x} - T\right) 2\mathbf{B}_i \right].
\end{aligned}$$

The solution of (2.2) is derived in the next Section.

2.2 Iterative Solution of Trust Region Subproblem (TRS)

To solve (2.2), a sequence of the "Trust Region Subproblems" (TRS) has to be solved. The TRS is to find the minimum of (2.2) with a given trust region radius Δ. Actually, $\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right)$ in (2.2) is the improving direction $\mathbf{d}^{(k)}$ to be found. Therefore we replace $\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right)$ by the improving direction $\mathbf{d}^{(k)}$. Without loss generality, we drop the superscript (k) and consider the following direction-finding problem.

$$\underset{\mathbf{d}}{Minimize}: f(\mathbf{x}) + \boldsymbol{\beta}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{G}\mathbf{d} \qquad (2.3)$$

$$subject\,to: \left\|\mathbf{d}^T\mathbf{d}\right\| \le \Delta$$

First, we characterize the exact solution of (2.3) by the theorem 2.1 which shows that the improving direction **d** satisfies

$$(\mathbf{G} + \mu\mathbf{I})\mathbf{d} = -\boldsymbol{\beta} \qquad (2.4)$$

Theorem 2.1 [9, 13]

The vector $\mathbf{d}$ is a global solution of the TR problem

$$\begin{aligned} &\underset{\mathbf{d}}{Minimize}: f + \boldsymbol{\beta}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{G}\mathbf{d} \\ &Subject\,to: \left\|\mathbf{d}\right\| \le \Delta \end{aligned} \qquad (2.5)$$

*if and only if **d** is feasible with the Lagrange multiplier* $\mu \ge 0$ *such that the following conditions are satisfied:*

$$(\mathbf{G} + \mu\mathbf{I})\mathbf{d} = -\boldsymbol{\beta}; \qquad (2.6)$$

$$\mu(\Delta - \|\mathbf{d}\|) = 0\,; \tag{2.7}$$

$$(\mathbf{G} + \mu\mathbf{I}) \text{ is positive semidefinite.} \tag{2.8}$$

Any solutions of (2.5) lies either in the interior or on the boundary of the feasible set (trust region), i.e. the set $\{\mathbf{d} \mid \|\mathbf{d}\| \le \Delta\}$. Equation (2.5) has no solution on the boundary if and only if **G** is positive define and $\|\mathbf{G}^{-1}\boldsymbol{\beta}\| < \Delta$. In this case, the solution of (2.5) is $\mathbf{d} = \mathbf{G}^{-1}\boldsymbol{\beta}$ with the Lagrangian multiplier $\mu^* = 0$.

In (2.4), the hessian matrix **G** and the gradient vector **β** are known. The unknowns in (2.4) are the solution **d** and the Lagrangian multiplier μ. The solution of **d** in (2.3) is shown to be:

$$\mathbf{d} = -(\mathbf{G} + \mu\mathbf{I})^{-1}\boldsymbol{\beta}\,. \tag{2.9}$$

According to (2.7), either $\Delta - \|\mathbf{d}\| = 0$ or $\mu = 0$ must hold. If $\mu = 0$ then the solution **d** is in the interior of the region else **d** is on the boundary. In the latter case $\Delta - \|\mathbf{d}\| = 0$ hold and then the norm of the solution **d,** $\|\mathbf{d}\|$**,** equals to the trust region radius Δ, i.e., $\Delta = \|\mathbf{d}\|$. Due to the equality relationship between the radius and the norm of the solution, (2.9) becomes

$$\|\mathbf{d}\| = \|-(\mathbf{G} + \mu\mathbf{I})^{-1}\boldsymbol{\beta}\| = \Delta\,. \tag{2.10}$$

From (2.10), the solution **d** is a function of μ. To find **d**, we have to find μ first. Finding μ is a typical root-finding problem of a nonlinear equation. We can apply

Newton method to help us find the optimal μ with a given radius Δ. Now we define the function $\phi(\mu)$ as

$$\phi(\mu) = \|\mathbf{d}(\mu)\| = \left\| -(\mathbf{G} + \mu\mathbf{I})^{-1}\boldsymbol{\beta} \right\| = \Delta\,. \tag{2.11}$$

Equation (2.11) describes the equality relationship between the radius and the Lagrangian multiplier, much like what we have discussed for (1.22) in Subsection 1.2.2 where we also have sketched the relationship on a two dimension space like Figure 1.8. It shows that if the Newton's method is applied to find the root of (2.11), the root finding procedure is slow and inefficient due to the nonlinearity of the function $\|\phi(\mu)\|$ with μ on the interval of $(-\lambda_1,\infty)$.

Fortunately, the Newton's method can perform quite efficiently with the following transformation to (2.11). The attempt is to reformulate (2.11) to become almost linear with μ on the interval of $(-\lambda_1,\infty)$. We define the reformulated equation as follows:

$$\phi_1(\mu) = \frac{1}{\Delta} - \frac{1}{\|\mathbf{d}(\mu)\|} = \frac{1}{\Delta} - \frac{1}{\left\| -(\mathbf{G} + \mu\mathbf{I})^{-1}\boldsymbol{\beta} \right\|} = 0 \tag{2.12}$$

As shown in Figure 2.2, $\phi_1(\mu)$ becomes a near-linear function of μ. Now the Newton's method can perform better to find the root.

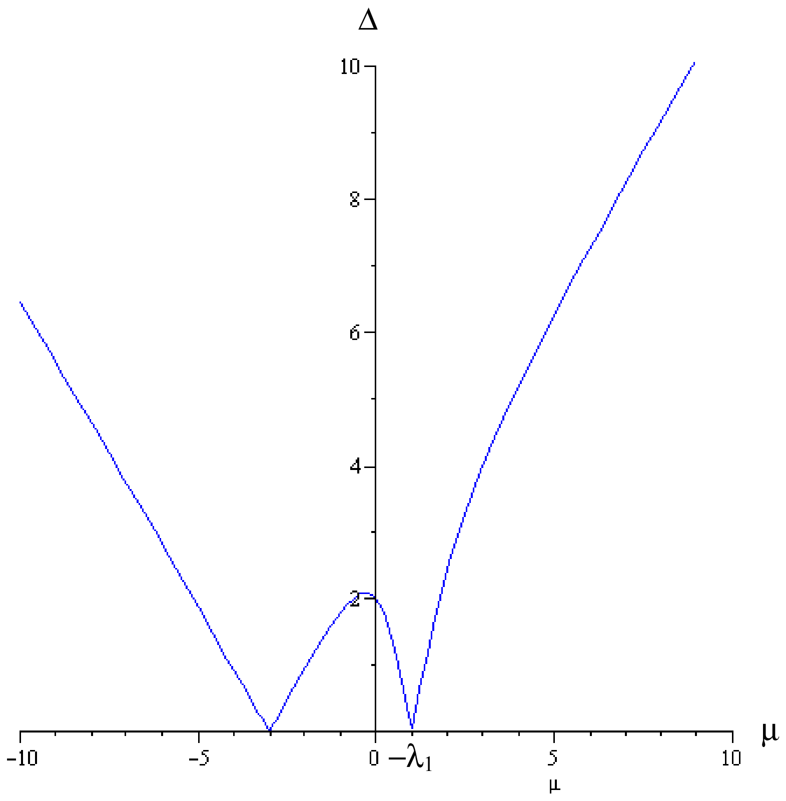


Figure 2.2 The relationship of $1/\Delta$ and μ in example 1.2

To apply the one-dimensional Newton's method:

$\tilde{\mu} = \mu - \frac{\phi_1(\mu)}{\phi_1'(\mu)}$, where $\phi_1'(\mu)$ is the first derivative of $\phi_1(\mu)$ and $\tilde{\mu}$ denotes the next Lagrangian multiplier found by the Newton's iterates. In order to perform the Newton's method $\phi_1(\mu)$ and $\phi_1'(\mu)$ must be evaluated. That can be obtained by solving a linear system involving $(\mathbf{G}+\mu\mathbf{I})$. Because in the range of interest, $(\mathbf{G}+\mu\mathbf{I})$ is definite positive, we may use its Cholesky factors $(\mathbf{G}+\mu\mathbf{I}) = \mathbf{U}(\mu)^T\mathbf{U}(\mu)$, where $\mathbf{U}(\mu)$ is an upper triangular matrix. To solve the problem, computation demanding calculation of the eigen-system of G is thus avoided.

However, to be able to use the Cholesky factorization, we have to ensure that $(\mathbf{G}+\mu\mathbf{I})$ is positive definite. In other words, μ has to be in the interval of $(-\lambda_1, \infty)$. A safeguard mechanism is therefore needed to ensure the success of the Newton's method. Here, we don't discuss the Newton's method and the safeguard mechanism in detail. For a more detailed explanation, please see Section 2.4 and Appendix B.

2.3 The Hard Case

Although the Newton's iterates can be used to find root of $\phi_1(\mu)$, there are some computation difficulties. The numerical difficulty is called "Hard Case" in the TR literature. The hard case occurs when the eigenvector corresponding to the smallest eigenvalue is perpendicular to the gradient vector $\boldsymbol{\beta}$, i.e., $\mathbf{q}_1{}^T\boldsymbol{\beta}=0$, where the eigenvector with respect to the smallest eigenvalue is denoted as $\mathbf{q}_1$. When there are multiple eigenvectors, i.e., an eigenspace corresponding to the smallest eigenvalue provided that $\mathbf{Q}_1{}^T\mathbf{b}=0$, where $\mathbf{Q}_1$ is the matrix whose columns span the eigenspace corresponding to the smallest eigenvalue. The hard case is caused by the failure of the limit condition $\lim_{\mu\to\lambda_i}\|\mathbf{d}(\mu)\|=\infty$. Therefore, there may not exist a value in $(-\lambda_1, \infty)$ to solve $\|\mathbf{d}(\mu)\|=\Delta$. We use an example to illustrate the hard case condition followed by a geometric interpretation.

Consider the following example with a current point at (d_1, d_2) = (0, 0).

Example 2.1

$$\begin{aligned} &Minimize{:}\,15{+}d_1+d_2{+}\frac{1}{2}d_1^2{+}2d_1d_2{+}\frac{1}{2}d_2^2. \\ &subject\,to:\sqrt{d_1^2+d_2^2}\le\Delta. \end{aligned} \qquad (2.13)$$

Express (2.14) in matrix notation:

$$\begin{aligned} &Minimize:\ 15{+}\boldsymbol{\beta}^T\mathbf{d}(\mu){+}\frac{1}{2}\mathbf{d}(\mu)^T\mathbf{G}\mathbf{d}(\mu); \\ &subject\ to:\|\mathbf{d}(\mu)\|<\Delta, \end{aligned}$$

where the gradient vector $\boldsymbol{\beta}=\begin{bmatrix}1\\1\end{bmatrix}$; the hessian matrix $\mathbf{G}=\begin{bmatrix}1&2\\2&1\end{bmatrix}$; the eigenvalue of **G** are -1 and 3; the corresponding eivenvectors are $[-1,1]^T$ and $[1,1]^T$. It can be seen that the gradient vector $\boldsymbol{\beta}$ be perpendicular to the eigenvector corresponding to smallest eigenvalue -1. The relationship among the radius, μ and the solution $\mathbf{d}(\mu)$ becomes:

$$\mathbf{d}(\mu)=-(\mathbf{G}+\mu\mathbf{I})^{-1}\boldsymbol{\beta}=-\begin{bmatrix}1+\mu&2\\2&1+\mu\end{bmatrix}^{-1}\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}\dfrac{-(\mu-1)}{(\mu+3)(\mu-1)}\\\dfrac{-(\mu-1)}{(\mu+3)(\mu-1)}\end{bmatrix}$$

$$\begin{aligned}
\Delta &= \|\mathbf{d}(\mu)\| \\
&= \sqrt{\left(\frac{-\mu+1}{(\mu-1)(\mu+3)}\right)^2+\left(\frac{-\mu+1}{(\mu-1)(\mu+3)}\right)^2} \\
&= \sqrt{2}\sqrt{\left(\frac{\mu-1}{(\mu-1)(\mu+3)}\right)^2} \\
&= \sqrt{2}\sqrt{\left(\frac{1}{(\mu+3)}\right)^2}
\end{aligned} \tag{2.14}$$

We also sketch (2.14) in a two dimension space. Figure 2.2 shows that there is only one pole corresponding to the second eigenvalue, that is, the pole with respect to the first eigenvalue is vanished. This is because the denominator $(\mu+3)(\mu-1)$ is eliminated by its factor $(\mu-1)$. However the pole which is corresponding to the second smallest eigenspace is still existed but the solution $\mu \in (-3,1)$ is only a local minimum on the spheral constraint. Fortunately, More, J. J. and D. C. Sorensen (1983) [13] propose a solution to solve the hard case which will be discussed latter.

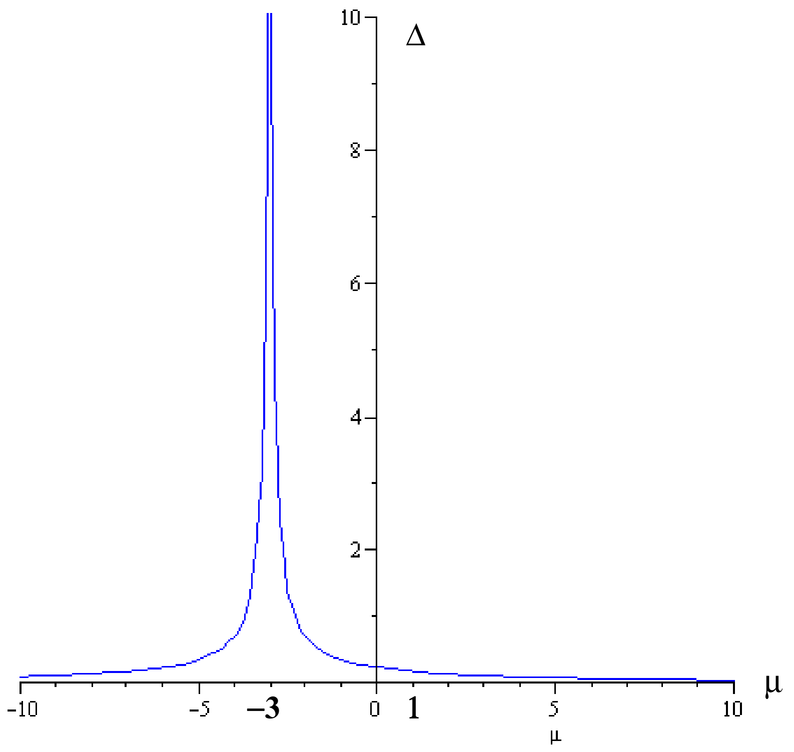


Figure 2.3 The vanished pole with respect to the smallest eigenspace

The hard case is a special situation in which the boundary solution of (2.3) is not unique. It can be shown that the hard case can only occur when the hessian matrix **G** is positive semidefinite, indefinite; the gradient vector $\boldsymbol{\beta}$ is perpendicular to the eigenspace with respect to the smallest eigenvalue of **G**; and $\Delta > \left\| -(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta} \right\|$, where the superscript ($^{+}$) indicates the pseudo inverse. That is, if any of the above three conditions is not met, the hard case cannot occur see also [17]. Let $\tau^2 = \Delta^2 - \phi^2(\lambda_1)$ and $\mathbf{d} = -(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}$. If $\lambda_1 \le 0$ and $\|\mathbf{d}\| < \Delta$ then the solution to the hard case is defined as:

$$-(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta} + \tau\mathbf{q}_1 = \mathbf{d} + \tau\mathbf{q}_1 \tag{2.15}$$

, where $\mathbf{q}_1 \in \mathbf{E}_{\min}$ (the eigenspace with respect to smallest eigenvalue) and $\|\mathbf{q}_1\| = 1$;

If **d** solves the (2.4) then **d** must satisfy the condition (2.7) to (2.9), see also Appendix A.

Now we explain the geometric interpretation of the hard-case solution. Consider the following example with a current point at $\mathbf{x}_0 = (x_1, x_2) = (-3,0)$.

$$\begin{aligned} &\textit{Minimize}: 15 + x_1 + x_1^2 - 2x_2^2. \\ &\textit{subject to}: x_1^2 + x_2^2 \le \Delta, \end{aligned}$$

where the Hessian matrix $\mathbf{G} = \begin{bmatrix} 2 & 0 \\ 0 & -4 \end{bmatrix}$ and the gradient vector $\boldsymbol{\beta} = \begin{bmatrix} -5 \\ 0 \end{bmatrix}$ at current point $(-3,0)$ and $\left\| -(\mathbf{G} - \lambda_1 \mathbf{I})^+ \boldsymbol{\beta} \right\| = 5/6$.

With the indefinite Hessian matrix **G**, we firstly show the geometric interpretation for the case with the gradient vector **β** orthogonal to $\mathbf{E}_{\min}$ but $\Delta < \left\| -(\mathbf{G} - \lambda_1 \mathbf{I})^+ \boldsymbol{\beta} \right\|$. Figure 2.4 shows a two-dimensional example where Δ is chosen to be 0.6. When the radius is chosen to be $\Delta < \left\| -(\mathbf{G} - \lambda_1)^+ \boldsymbol{\beta} \right\| = 5/6$, there is still a unique solution because the intersection of the sphere and the contour of the optimal *y* along the **d** direction is a unique point.

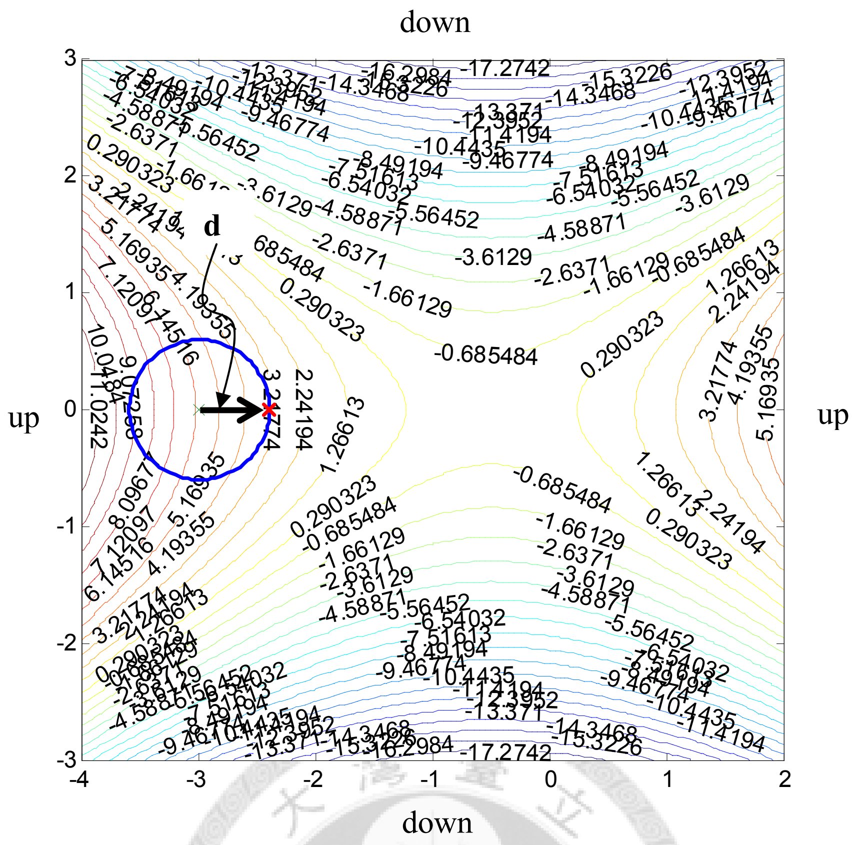


Figure 2.4 The Easy Case for an Indefinite Hessian

Suppose the trust-region radius Δ is chosen to be 1.5 and is greater than $5/6$, i.e., the gradient vector $\boldsymbol{\beta}$ is still orthogonal to $\mathbf{E}_{\min}$ but $\Delta > \left\| -(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta} \right\|$. Figure 2.5 shows how the solution for this case becomes not unique. In Figure 2.5 the length from the current point $\mathbf{x}_0 = (-3,0)$ to $\mathbf{x}_0 + \mathbf{d}$, $\left(-2\frac{1}{6},0\right)$, is $5/6$ (the length of the bold line in Figure 2.5) and less than 1.5. In this case, there are actually two solutions by adding $\mathbf{d}$ with $\tau\mathbf{q}_1$ and $-\tau\mathbf{q}_1$ (dotted lines): $\mathbf{d}_{H1} = -(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta} + \tau\mathbf{q}_1$ and $\mathbf{d}_{H2} = -(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta} - \tau\mathbf{q}_1$, i.e., two bold dashed lines in Figure 2.5.

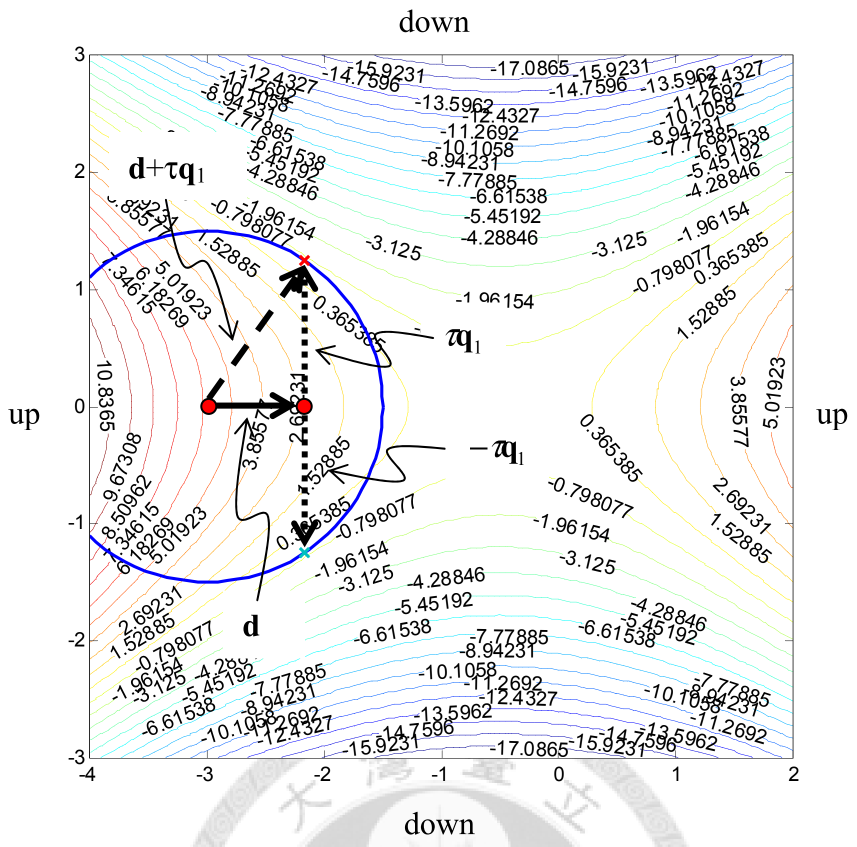


Figure 2.5 The Hard Case for an Indefinite Hessian

2.4 Modifications of Trust Region Algorithm

We do some modification to the TRS algorithm [9], the conventional TRS algorithm is detailed in Appendix B. The first is that because we need to compute the eigenvectors with respect to the smallest eigenvalue to solve the hard case, we use a more numerical computation robust method, namely, Singular Value Decomposition (SVD), to compute the eigen-system. Cholesky factorization is therefore replaced by SVD.

We first derive the all ingredients for the Trust Region algorithm. The root-finding problem applied Newton's method generates a sequence of iterate of $\tilde{\mu}$ by setting

$$\tilde{\mu} = \mu - \phi_1(\mu)/\phi_1'(\mu) \tag{2.16}$$

, where *k* is the *k*-th search index; $\tilde{\mu}$ is the next Lagrangian multiplier found in the Newton's iterates and

$$\begin{aligned}\phi_1'(\mu) &= \frac{-d\left(\|\mathbf{d}(\mu)\|^{-1}\right)}{d\mu} = -\left[-\frac{1}{2}\boldsymbol{\beta}^T(\mathbf{G}+\mu\mathbf{I})^{-2}\boldsymbol{\beta}\right]^{-\frac{3}{2}}\left[-2\boldsymbol{\beta}^T(\mathbf{G}+\mu\mathbf{I})^{-3}\boldsymbol{\beta}\right] \\ &= -\left[\boldsymbol{\beta}^T(\mathbf{G}-\mu\mathbf{I})^{-2}\boldsymbol{\beta}\right]^{-\frac{3}{2}}\left[\boldsymbol{\beta}^T(\mathbf{G}-\mu\mathbf{I})^{-3}\boldsymbol{\beta}\right]\end{aligned} \tag{2.17}$$

In trust region literature, the first order derivative can solve by solving linear system. Due to the matrix $(\mathbf{G}+\mu\mathbf{I})$ is positive definite with μ on the interval $(-\lambda_1,\infty)$ and $(\mathbf{G}+\mu\mathbf{I})$ is also a symmetric matrix so it can be factorized by Cholesky factorization

as $(\mathbf{G}+\mu\mathbf{I}) = \mathbf{U}^T\mathbf{U}$ (2.18)

, where **U** is a upper triangular matrix.

By substituting (2.18) into (2.4) yields

$$\mathbf{U}^T\mathbf{U}\,\mathbf{d} = \mathbf{U}^T\mathbf{U}\,\mathbf{d}(\mu) = -\boldsymbol{\beta}. \tag{2.19}$$

Solve the linear system (2.19) we have the the solution $\mathbf{d}(\mu)$ becomes

$$\mathbf{d}(\mu) = -\mathbf{U}^{-1}\mathbf{U}^{-T}\boldsymbol{\beta} \tag{2.20}$$

, and $\mathbf{d}^T(\mu)\mathbf{d}(\mu) = \boldsymbol{\beta}^T\mathbf{U}^{-1}\mathbf{U}^{-T}\mathbf{U}^{-1}\mathbf{U}^{-T}\boldsymbol{\beta} = \boldsymbol{\beta}^T(\mathbf{G}+\mu\mathbf{I})^{-2}\boldsymbol{\beta}$. (2.21)

Also, solve the linear system $\mathbf{U}^{\mathrm{T}}\mathbf{U}\ \mathbf{y}(\mu) = \mathbf{d}(\mu)$, the solution $\mathbf{y}\ (\mu)$ is

$$\mathbf{y}\ (\mu) = \ \mathbf{U}^{-1}\mathbf{U}^{-T}\mathbf{d}(\mu) = -\mathbf{U}^{-1}\mathbf{U}^{-T}\mathbf{U}^{-1}\mathbf{U}^{-T}\boldsymbol{\beta} = -(\mathbf{G}+\mu\mathbf{I})^{-2}\boldsymbol{\beta}\ . \tag{2.22}$$

Besides we also have

$$\mathbf{d}^{T}(\mu)\mathbf{y}(\mu) = \boldsymbol{\beta}^{T}(\mathbf{G}+\mu\mathbf{I})^{-3}\boldsymbol{\beta} \tag{2.23}$$

Substituting (2.21) and (2.23) into (2.17) yields

$$\begin{aligned}\phi_1{}'(\mu) &= -\left[\mathbf{d}^{T}(\mu)\mathbf{d}(\mu)\right]^{-\frac{3}{2}}\left[\mathbf{d}^{T}(\mu)\mathbf{y}(\mu)\right] = -\left[\|\mathbf{d}(\mu)\|^{2}\right]^{-\frac{3}{2}}\left[\mathbf{d}^{T}(\mu)\mathbf{y}(\mu)\right] \\ &= -\|\mathbf{d}(\mu)\|^{-3}\left[\mathbf{d}^{T}(\mu)\mathbf{y}(\mu)\right]\end{aligned} \tag{2.24}$$

Also, substitute (2.12) and (2.24) into (2.16), and we have the formula of Newton's iterates

$$\begin{aligned}\tilde{\mu} &= \mu - \left(\frac{1}{\Delta} - \frac{1}{\|\mathbf{d}(\mu)\|}\right) \Big/ \left(-\|\mathbf{d}(\mu)\|^{-3}\left[\mathbf{d}^{T}(\mu)\mathbf{y}(\mu)\right]\right) \\ &= \mu - \left(\frac{\|\mathbf{d}(\mu)\| - \Delta}{\Delta\|\mathbf{d}(\mu)\|}\right)\left(\frac{-\|\mathbf{d}(\mu)\|^{3}}{\left[\mathbf{d}^{T}(\mu)\mathbf{y}(\mu)\right]}\right) \\ &= \mu + \left(\frac{\|\mathbf{d}(\mu)\| - \Delta}{\Delta}\right)\left(\frac{\|\mathbf{d}(\mu)\|^{2}}{\left[\mathbf{d}^{T}(\mu)\mathbf{y}(\mu)\right]}\right)\end{aligned} \tag{2.25}$$

Now we have derived the formula for performing Newton's iteration. The detailed algorithm is described in Appendix A.

Because (2.12) and (2.17) involve the term $(\mathbf{G}+\mu\mathbf{I})^{-p}$ where $p \in \Re$. By applying the SVD method to $(\mathbf{G}+\mu\mathbf{I})$ we have

$$(\mathbf{G}+\mu\mathbf{I}) = \mathbf{Q}\boldsymbol{\Sigma}\mathbf{Q}^{T} \tag{2.26}$$

, where $\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} = \begin{bmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_n} \end{bmatrix}$ and $\lambda_1, ..., \lambda_n$ are the eigenvalues of $(\mathbf{G} + \mu\mathbf{I})$; $\mathbf{Q}$ is the n by n matrix with columns consisting of orthonormal eigenvectors of $(\mathbf{G} + \mu\mathbf{I})$.

Therefore, the inverse of $(\mathbf{G} + \mu\mathbf{I})$ with any order p could be calculated by the following formula.

$$(\mathbf{G} + \mu\mathbf{I})^{-p} = (\mathbf{G} + \mu\mathbf{I})^{+p} = \mathbf{Q}\Sigma^{+p}\mathbf{Q}^T = \mathbf{Q}\begin{bmatrix} \frac{1}{\sigma_1^{\ p}} & & \\ & \ddots & \\ & & \frac{1}{\sigma_n^P} \end{bmatrix}\mathbf{Q}^T. \tag{2.27}$$

To proceed with our algorithm, we also have to do the following transformation.

$$\boldsymbol{\theta} = \mathbf{Q}\boldsymbol{\beta} \tag{2.28}$$

, where the components of $\boldsymbol{\theta}$ denotes as θ_i which is the product of the eigenvectors $\mathbf{q}_i$ and the gradient vector $\boldsymbol{\beta}$.

The second modification of the TRS algorithm is to find the lower-bound for the Lagrangian multiplier μ more efficiently. The purpose of the lower-bound is to prevent the unsuccessful iterates of the Newton's method. As presented in Section 2.2, the safeguard mechanism of the TRS algorithm is designed to prevent this situation. Figure 2.6 shows Newton's method leads μ beyond the logical interval. Moreover, the traditional TRS algorithm doesn't compute the eigensystem, i.e., they do not use the information of $-\lambda_1$ to safeguard the possible failure of Newton's method. Since the S.V.D has been used to help us solve the problem, and the smallest eigenvalue of the Hessian matrix is also obtained. We may establish a new lower-bound based on the current Lagrangian multiplier. It will be shown that this new lower-bound will be better than the lower bound $\mu_{\min}^{(S)}$ proposed by Semple, J. (1997) [18]. To derive the lower-bound, we first define

$$\begin{aligned}\Phi(\mu) &= \|\mathbf{d}(\mu)\|^2 = \mathbf{d}^T(\mu)\mathbf{d}(\mu) \\ &= \boldsymbol{\beta}^T(\mathbf{G}+\mu\mathbf{I})^{-2}\boldsymbol{\beta} \\ &= \left(\boldsymbol{\beta}^T\mathbf{Q}^{-T}\right)(\boldsymbol{\Lambda}+\mu\mathbf{I})^{-1}\left(\mathbf{Q}^{-1}\mathbf{Q}^{-T}\right)(\boldsymbol{\Lambda}+\mu\mathbf{I})^{-1}\left(\mathbf{Q}^{-1}\boldsymbol{\beta}\right) \\ &= \left(\mathbf{Q}^T\boldsymbol{\beta}\right)^T(\boldsymbol{\Lambda}+\mu\mathbf{I})^{-2}\left(\mathbf{Q}^T\boldsymbol{\beta}\right) \\ &= \sum_{i=1}^{n}\frac{r_i^2}{(\lambda_i+\mu)^2}\end{aligned} \tag{2.29}$$

Differentiating (2.29) with respect to μ produces

$$\Phi'(\mu) = -2\boldsymbol{\beta}(\mathbf{G}+\mu\mathbf{I})^{-3}\boldsymbol{\beta} = -2\mathbf{d}^T(\mu)\mathbf{y}(\mu) \tag{2.30}$$

The lower bound is estimated by the following inequality:

$$\frac{-\Phi(\mu)}{\Phi'(\mu)} = \left(\frac{\sum_{i=1}^{k}\left(\frac{\gamma_i^2}{(\lambda_i+\mu)^2}\right)}{2\sum_{i=1}^{k}\left(\frac{\gamma_i^2}{(\lambda_i+\mu)^3}\right)}\right) \geq \frac{\lambda_1+\mu}{2} \tag{2.31}$$

, whenever $\mu \in (-\lambda_1, \infty)$.

Both elements in (2.29) and (2.30) are calculated in the Newton's iterate, so it is easy to identify the estimated lower bound becomes

$$-\lambda_1 \geq \mu + \frac{2\Phi(\mu)}{\Phi'(\mu)} = \mu + \frac{2\|\mathbf{d}(\mu)\|^2}{-2\mathbf{d}^T(\mu)\mathbf{y}(\mu)} = \mu - \frac{\|\mathbf{d}(\mu)\|^2}{\mathbf{d}^T(\mu)\mathbf{y}(\mu)}. \tag{2.32}$$

Then the lower-bound proposed by Semple, J denotes as $\mu_{\min}^{(S)}$ can be written as:

$$\mu_{\min}^{(S)} = \mu - \frac{\|\mathbf{d}(\mu)\|^2}{\mathbf{d}^T(\mu)\mathbf{y}(\mu)}.$$

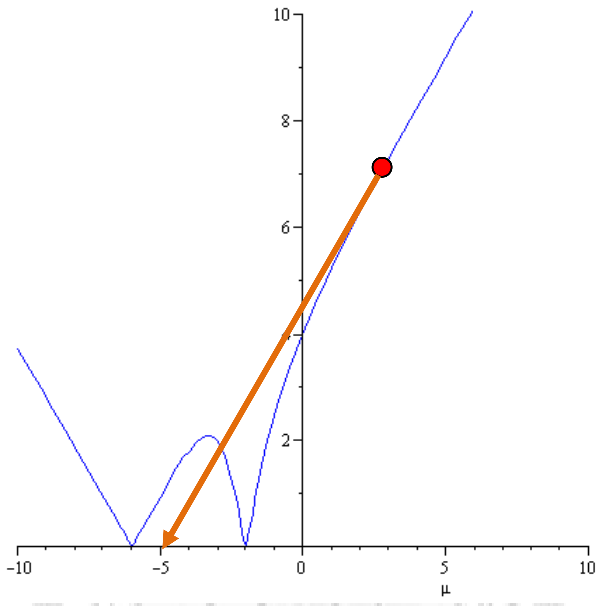


Figure 2.6 The failure Newton's iteration

On the other hand, by using SVD our lower-bound $\mu_{\min}^{(T)}$ can be calculated by the following formula:

$$\begin{aligned}\mu_{\min}^{(T)} &= \mu - \frac{\phi(\mu)}{\phi'(\mu)} \\ &= \mu - \frac{\|\mathbf{d}(\mu)\| - \Delta}{-\left(\boldsymbol{\beta}(\mathbf{G}+\mu\mathbf{I})^{-2}\boldsymbol{\beta}\right)^{-\frac{1}{2}}\left(\boldsymbol{\beta}(\mathbf{G}+\mu\mathbf{I})^{-3}\boldsymbol{\beta}\right)} \\ &= \mu + \frac{\left(\|\mathbf{d}(\mu)\| - \Delta\right)\|\mathbf{d}(\mu)\|}{\left(\boldsymbol{\beta}(\mathbf{G}+\mu\mathbf{I})^{-3}\boldsymbol{\beta}\right)},\end{aligned} \tag{2.33}$$

Thus the lower bound of μ can be then set to

$$\mu_{\min} = \max\left(-\lambda_1, \mu_{\min}^{(T)}\right). \tag{2.34}$$

That is, the larger value between the negative smallest eigenvalue and the lower bound in (2.33) is set to be $\mu_{\min}$. We use Example 1.2 to show (2.34) is better than $\mu_{\min}^{(S)}$ and their geometric meanings in three different situations, i.e., three different positions of the current point.

Situation 1:

For being a positive definite matrix **G** + μ**I**, let $\Delta = 1$ and consider μ_0 = 3 as the current point. Calculate the two lower-bounds and the two lower-bounds are illustrated in Figure 2.7.

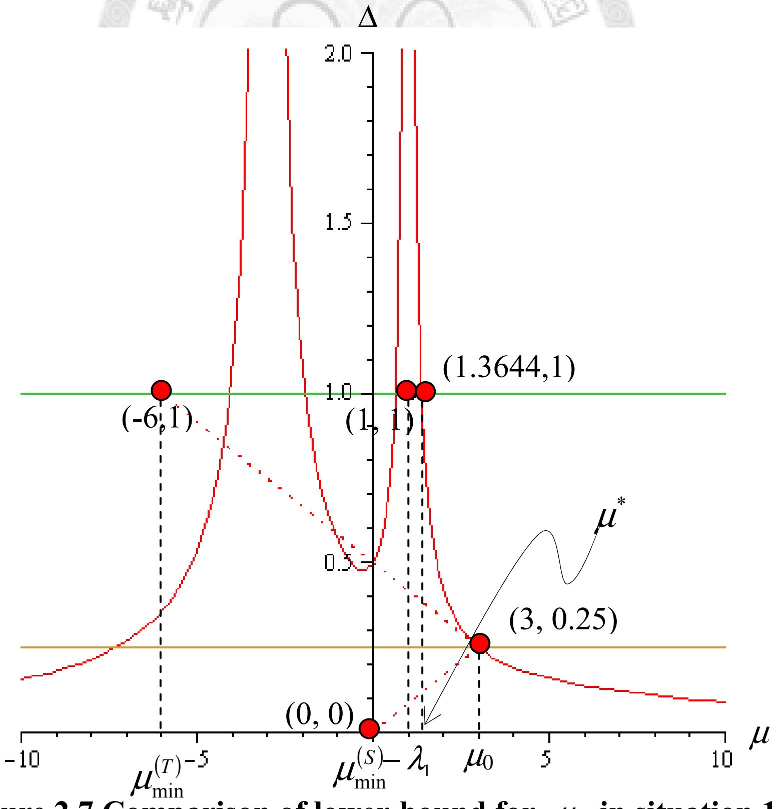


Figure 2.7 Comparison of lower-bound for μ in situation 1

Since $\mu_{\min}^{(T)} < -\lambda_1$, we set $\mu_{\min}$ to be $-\lambda_1$ according to (2.34). With the help of $-\lambda_1$, we a obtain better $\mu_{\min}$ than $\mu_{\min}^{(S)}$.

Situation 2:

In this situation, we consider μ_0 = 1.5 (at the right of the optimal μ^*) to be the current point as shown in Figure 2.8. The two lower-bound are calculated and shown in Figure 2.8.

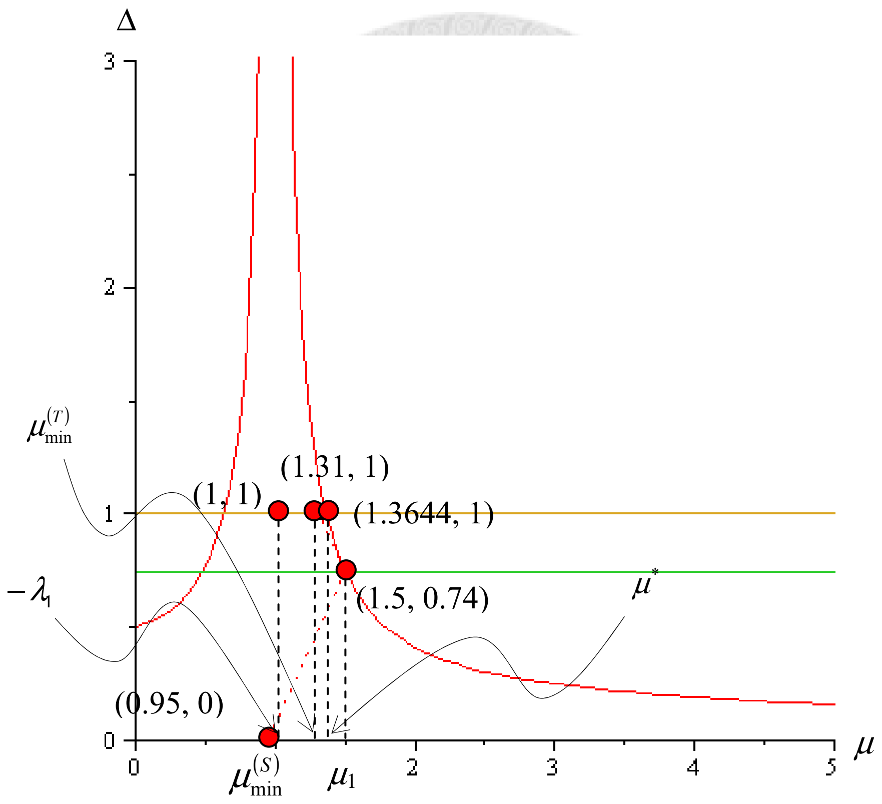


Figure 2.8 Comparison of lower-bound for μ in situation 2

As shown in Figure 2.8, $\mu_{\min}^{(T)}$ is set to be 1.31, which appears to be very close to the optimal μ^* and also lower than $\mu_{\min}^{(T)}$.

Situation 3:

In this situation, we consider $\mu_0 = 1.2$ (at the left of the optimal μ^*) to be the current point as shown in Figure 2.9. Again the two lower-bound are also calculated and shown in Figure 2.9.

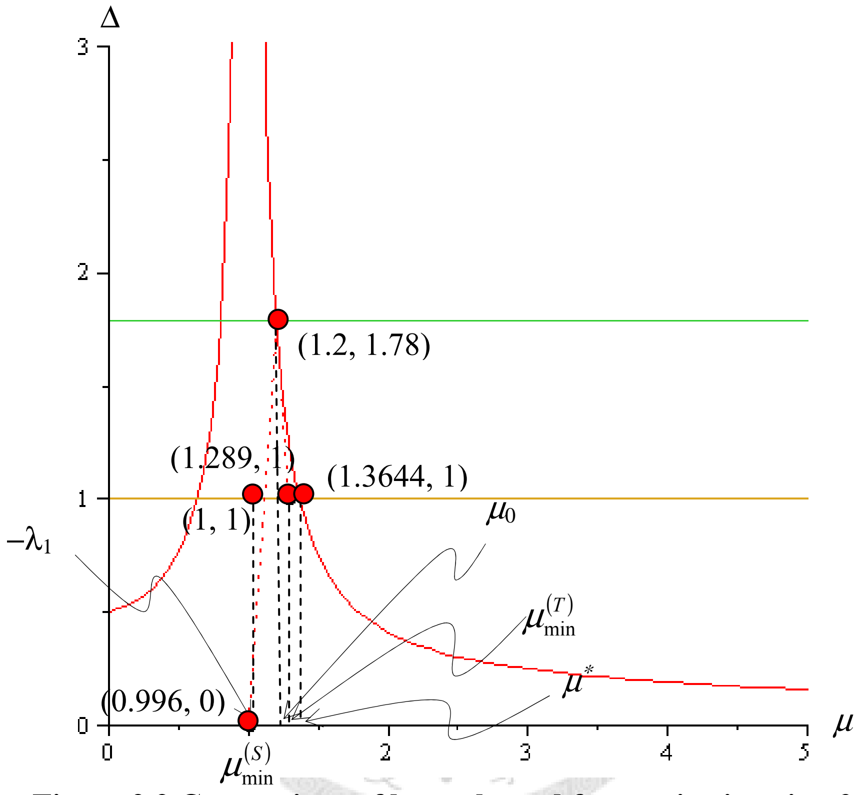


Figure 2.9 Comparison of lower-bound for μ in situation 3

We find that $\mu^{(T)}_{\min}$ is still larger than $\mu^{(S)}_{\min}$ even if the current point μ_0 is on the left hand side of μ^*. We already demonstrate, without proof, that $\mu^{(T)}_{\min}$ better than $\mu^{(S)}_{\min}$. When we use the lower-bound to safeguard the Newton's method from invalid solutions, this new lower bound helps the Newton's method to converge quickly.

We summarize the algorithm to solve the Trust Region Subproblem as follows.

```
Algorithm 2.1 (Trust Region Subproblem Algorithm)
Begin
    Perform S.V.D to $(\mathbf{G}+\mu\mathbf{I})$ by (2.27)
    Calculate $\boldsymbol{\theta}$ by (2.28)
    If G is positive definite then
        Return $\mu^*=0$ and the solution $\mathbf{d}(0)=-\mathbf{G}^{-1}\boldsymbol{\beta}$    (2.35)
    Else If $\boldsymbol{\beta}\perp\mathbf{E}_{\min}$ then
        Calculate $\tau^2=\left\{\Delta^2-\left[(\frac{\theta_2}{\mu_2+\lambda_1})^2+\ldots+(\frac{\theta_k}{\mu_k+\lambda_1})^2)\right]\right\}$    (2.36)
        If $\tau^2>0$ then
            $\mathbf{D}\leftarrow\begin{bmatrix}0\\ \frac{\theta_2}{\mu_2+\lambda_1}\\ \vdots\\ \frac{\theta_k}{\mu_k+\lambda_1}\end{bmatrix}\pm\tau\begin{bmatrix}1\\0\\ \vdots\\0\end{bmatrix}=\begin{bmatrix}\pm\tau\\ \frac{\theta_2}{\mu_2+\lambda_1}\\ \vdots\\ \frac{\theta_k}{\mu_k+\lambda_1}\end{bmatrix}$    (2.37)
                Return a better solution d by evaluating the objective of original
                objective function
        Else
                Go to Algorithm 2.2 (the problem is a good case)
        End If
    Else
        Go to Algorithm 2.2 (the problem is a good case).
    End If
End
```

Notice that, when the hard case occurs the optimal solution must be chosen by evaluating the objective value of original objective function. The algorithm for the TR Hard Case is complete. For "Good Case" of TRS, the Newton's iterates algorithm is shown below:

```
Algorithm 2.2 (Algorithm for Good Case)
Input:
δ : the tolerance for convergence of the solution $\mathbf{d}(\mu)$
$\varepsilon$ : the tolerance for ensuring (ensure $(\mathbf{G}+\mu\mathbf{I})$ is P.D.)
G:the hessian matrix of (2.2)
β: the gradient vector of (2.2)
Δ: the given trust region radius
$\lambda_1$: the smallest eigenvalue
Begin
    $\mu \leftarrow -\lambda_1 + \varepsilon$ (ensure $(\mathbf{G}+\mu\mathbf{I})$ is P.D.).    (2.38)
    Repeat while $\left|\|\mathbf{d}(\mu)\| - \Delta\right| > \delta$
        $\mu_{\min} \leftarrow \max\left(-\lambda_1, \mu + \frac{(\|\mathbf{d}(\mu)\| - \Delta)\|\mathbf{d}(\mu)\|}{(\boldsymbol{\beta}(\mathbf{G}+\mu\mathbf{I})^{-3}\boldsymbol{\beta})}\right)$    (2.39)
        (set the lower-bound for $\mu$).
        If $\|\mathbf{d}(\mu)\| < \Delta$ (at the right of the root) then
            $\mu_{\max} \leftarrow \mu$    (2.40)
        Else
            $\mu_{\min} \leftarrow \mu$    (2.41)
        End If
        $\tilde{\mu} \leftarrow \mu + \dfrac{\dfrac{1}{\Delta} - \dfrac{1}{\|\mathbf{d}(\mu)\|}}{\left[\boldsymbol{\beta}^T (\mathbf{G} + \mu\mathbf{I})^{-2} \boldsymbol{\beta}\right]^{-\frac{3}{2}} \left[\boldsymbol{\beta}^T (\mathbf{G} + \mu\mathbf{I})^{-3} \boldsymbol{\beta}\right]}$        (2.42)
        **If** $\tilde{\mu} < \mu_{\min}$ **then**
            $\tilde{\mu} \leftarrow \dfrac{\mu_{\max} + \mu_{\min}}{2}$.        (2.43)
        **End If**
    **End Repeat**
    **Return** $\mu^* = \tilde{\mu}$
**End**
```

Example 1.2 is used again to demonstrate the TRS good case algorithm. With the explanation of the geometric meanings, first let the given trust region radius to be 1;ε = 2; the procedure is detailed as follows.

Preparation:

The eigenvalue of the Hessian matrix are 3 and -1 respectively, that is, **G** is an indefinite matrix. We first set $\mu = -(-1) + 2 = 3$ according to (2.38), and then proceed to the algorithm.

Iteration 1:

By (2.39), we have $\mathbf{d}(\mu)=\mathbf{d}(3)=\begin{bmatrix}-0.25\\0\end{bmatrix}$ and $\|\mathbf{d}(\mu)\|=0.25<1$. Thus we enter the TRS problem solving step. The $\mu_{\min}$ is first found to be $\max(1,-6)=1$ according to (2.39) and is shown in Figure 2.10. Because $\|\mathbf{d}(\mu)\|=0.25<1$, to obtain an valid μ, we can set $\mu_{\max}=3$. With μ^* known to be in the interval of $(\mu_{\min},\mu_{\max})=(1,3)$, the first Newton's iterate can be performed by (2.42), as shown in Figure 2.11 $\breve{\mu}=0.75$ Because $\breve{\mu}$ is not in (1, 3) according to the safeguard mechanism (2.43), we take the average of $\mu_{\min}$ and $\mu_{\max}$ to replace $\breve{\mu}$, i.e., $\tilde{\mu}=(1+3)/2=2$. The two bold-dashed line in Figure 2.11 indicate $\mu_{\min}$ and $\mu_{\max}$ in the space of $1/\Delta$.

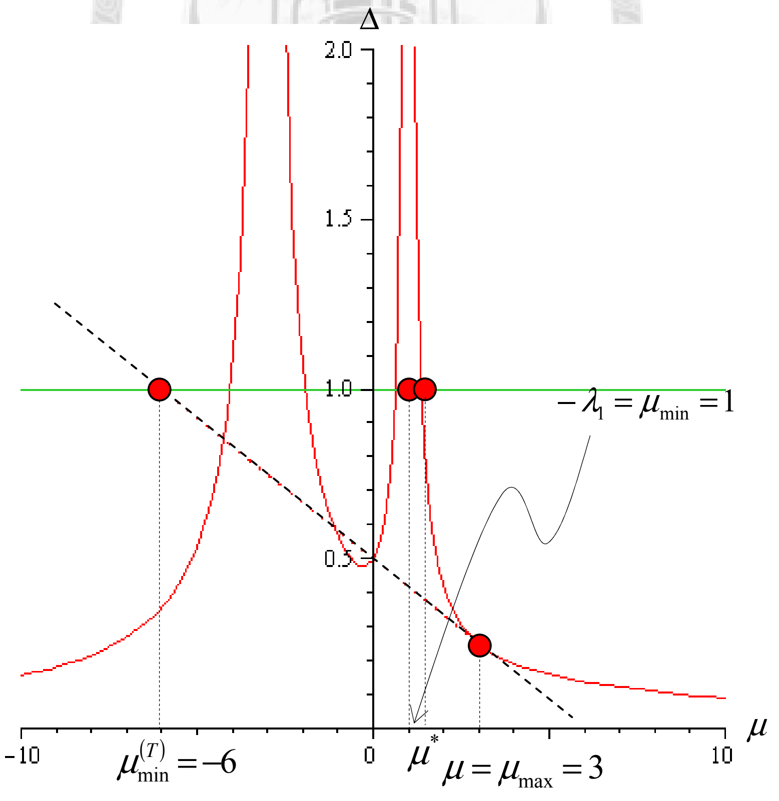


Figure 2.10 $\mu_{\max}$, $\mu_{\min}$, $\mu_{\min}^{(T)}$ **and** μ_1 **on two-dimensional space**

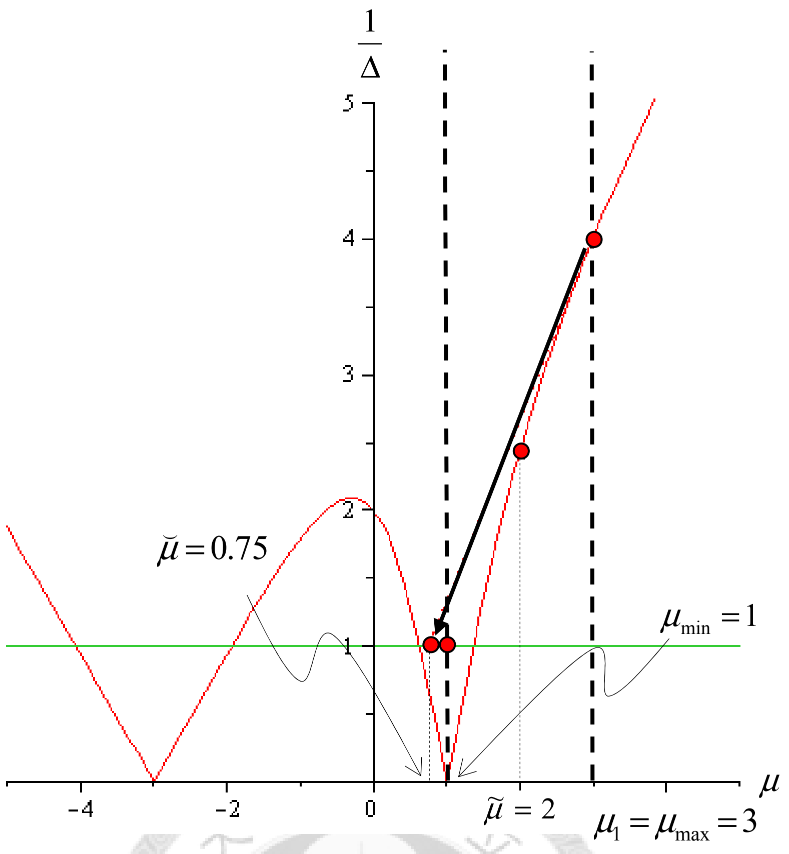


Figure 2.11 Safeguard mechanism for Newton's iterate in the $1/\Delta$ space

With $\tilde{\mu}=2$, the remaining iteration is listed in the following table.

Table 2.1 The iterative results of example 1.2 solved by the TRS algorithm

Iteration	m_k	$\mathbf{d}(\mu_k)$	$\lVert\mathbf{d}(\mu_k)\rVert$
1	3	Safeguarded	Safeguarded
2	2	(-0.4,0.1)	0.41231
3	1.2544	(-1.1588,0.8063)	1.41181
4	1.3623	(-08618,0.5179)	1.0055
5	1.3644	(-0.8577,0.5140)	≈ 1

3 Generalized Reduced Trust Region (GRT) Search

In this Chapter, we develop an effective search algorithm based on the trust region method. In Section 3.1, we introduce a conventional search algorithm based on the trust region method. In Section 3.2, we propose our search algorithm by considering the modified TRS algorithm in Chapter 2 and the generalized space reduction method. In the final Section, we provide a convergence proof for the proposed search algorithm.

3.1 Trust Region Search Method

In Chapter 2, the trust region method and the related algorithm are introduced. Now we consider the use of the trust region method for optimization. The choice of the trust region radius will be an important issue during optimization. The problem will be approached by considering the approximation quality of the current iteration. Given a step **d** from the current $\mathbf{x}^{(k)}$, the response improving ratio $\rho^{(k)}$ is defined as follows:

$$\rho^{(k)} = \frac{f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k)} + \mathbf{d})}{m(0) - m(\mathbf{d})} \tag{3.1}$$

, where the numerator and the denominator are called the actual reduction and the predicted reduction; the superscript (k) denotes the k-th iteration; $f(\bullet)$ is original objective function and $m(\bullet)$ is the approximated objective function by the Taylor expansion.

Notice that because the trust region method finds the solution inside the entire trust region so the denominator must be greater than or equal to zero, i.e., the predicted solution of the current iteration must not be worse than the solution found by the previous iteration. When we substitute the solution solved by the trust region method into the original function *f*, the new objective value $f\left(\mathbf{x}^{(k)}+\mathbf{d}\right)$ may be greater or less than $f\left(\mathbf{x}^{(k)}\right)$. That is, the numerator may be greater or less than zero and determine the sign of $\rho^{(k)}$. If $\rho^{(k)} < 0.25$, then the actual reduction provided by **d** is smaller than the predicted reduction, thus the step **d** must be rejected. On the other hand, if $\rho^{(k)}$ is close to 1 that means the predicted reduction is quite close to the actual reduction; namely, the function $m(\bullet)$ is a good approximate of the original objection function $f(\bullet)$ and it is also safe to enlarge the trust-region radius for the next iteration. But if $\rho^{(k)} < 0$ and $\rho^{(k)}$ is significantly smaller than 1 then we shrink the trust region by reducing the trust-region radius Δ for the next iteration. Such a trust-region radius adjustment strategy is expected to remedy the approximation deficiency of the GRR search.

The following algorithm describes an iteration of the search process without constraints.

```
Algorithm 3.1
Input:
$\hat{\Delta}$: an overall upper bound on the step lengths and $\hat{\Delta} > 0$
$\mathbf{x}^{(k)}$: the current point
$\Delta^{(k)}$: initial trust region radius and $\Delta^{(1)} \in (0, \Delta)$
$\mathbf{d}^{(k)}$: an improving direction from current $\mathbf{x}^{(k)}$
$\eta$: threshold above which $\rho$ is considered to be a trusted improvement, where
    $\eta \in [0, 0.25)$.
Begin
    For $k = 0, 1, 2, \ldots$ do
        Obtain $\mathbf{d}^{(k)}$ by solving algorithm 2.1.
        Evaluate $\rho^{(k)}$ by (3.1).
        If $\rho^{(k)} < \frac{1}{4}$
            $\Delta^{(k+1)} \leftarrow \frac{1}{4}\Delta^{(k)}$                                    (3.2)
        Else If $\rho^{(k)} > \frac{3}{4}$ and $\left\|\mathbf{d}^{(k)}\right\| = \Delta^{(k)}$ (boundary solution)
            $\Delta^{(k+1)} \leftarrow \min\left(2\Delta^{(k)}, \widehat{\Delta}\right)$                    (3.3)
        Else
            $\Delta^{(k+1)} \leftarrow \Delta^{(k)}$;                                      (3.4)
        End If
        If $\rho^{(k)} > \eta$
            $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$                            (3.5)
        Else
```

```
            $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)}$;                    (3.6)
        End if
    End For
End
```

Equation (3.3) means that if we want to enlarge the trust region, the solution found must be already as far away from the current point as possible, i.e., on the boundary $\|\mathbf{d}^{(k)}\| = \Delta^{(k)}$. The purpose of the (3.5) and (3.6) is to determine if the improvement is worth moving the current point to the next point.

Again, we consider the Rosenbrock's function as our example for performing Algorithm 3.1.

Example 3.1:

Minimize: $(1-x_1)^2+100\times(x_2-x_1^2)^2$.

Settings: Initial Point: $(x_1, x_2) = (-2, 0.5)$; $\hat{\Delta} = 2$; $\Delta^{(1)} = 1$; $\eta = 0.25$.

Iteration 1:

Solving the trust region subproblem yields $d^{(1)} = (0.4351, 0.9003)$. Because $\rho^{(1)} = 1.09 > 0.25$ and the norm of $\mathbf{d}^{(1)}$ is equal to 1.00000, i.e., $\mathbf{d}^{(1)}$ is a boundary solution according to (3.5), we have $\rho^{(1)} > 0.25$ and set $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \mathbf{d}^{(1)} = (-1.5648, 1.4003)$. According to (3.3), because $\rho^{(1)} = 1.09 > 0.75$, we enlarge the trust region radius $\Delta^{(2)}$

to be $2\Delta^{(1)}$ for $\mathbf{x}^{(2)}$. Figure 3.1 shows the processes of Iteration 1 and the improving direction.

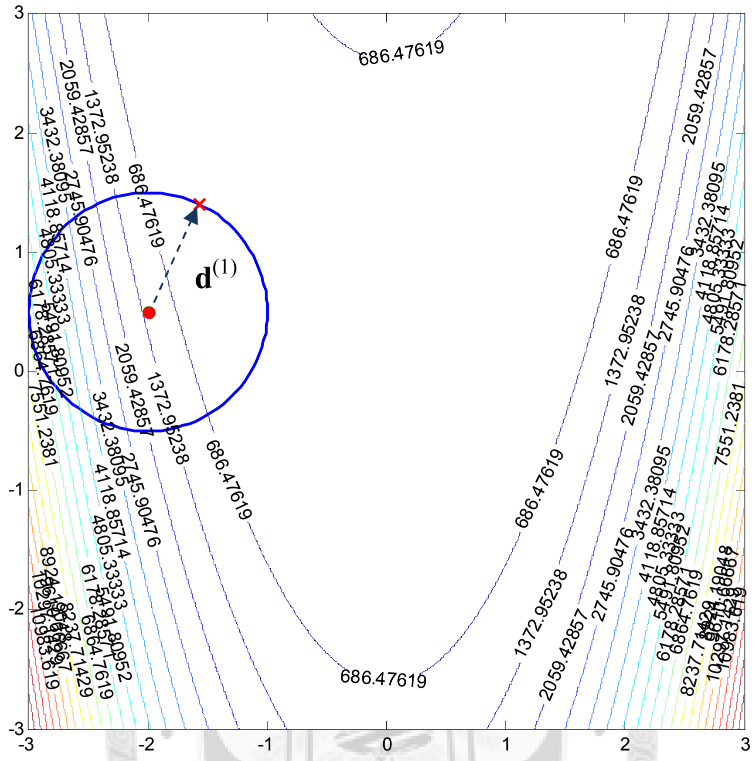


Figure 3.1 Positions of $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ at iteration1 with $\Delta^{(1)} = 0.5$

Iteration 2:

Solve the TRS for $\mathbf{x}^{(2)}$ with $\Delta^{(2)}$ and yield $\mathbf{d}^{(2)} = (0.0121, 1.010)$. We find that $\rho^{(2)}$ = 1.002 > 0.25 and move $x^{(2)}$ to $\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \mathbf{d}^{(2)} = (-1.5526, 2.4105)$ according to (3.5). But the norm of $\mathbf{d}^{(2)}$ = 1.0102 < $\Delta^{(2)}$ = 2, i.e., not on the boundary. According to (3.4), we do not need to enlarge the trust region for $\mathbf{x}^{(3)}$ and $\Delta^{(3)}$ remains to be 2. This is because the Hessian matrix is already positive definite thus the optimal Lagrangian multiplier is 0 and the solution is inside the trust region. Figure 3.2 shows the search processes and the inside solution with trust region radius $\Delta^{(2)} = 2$.

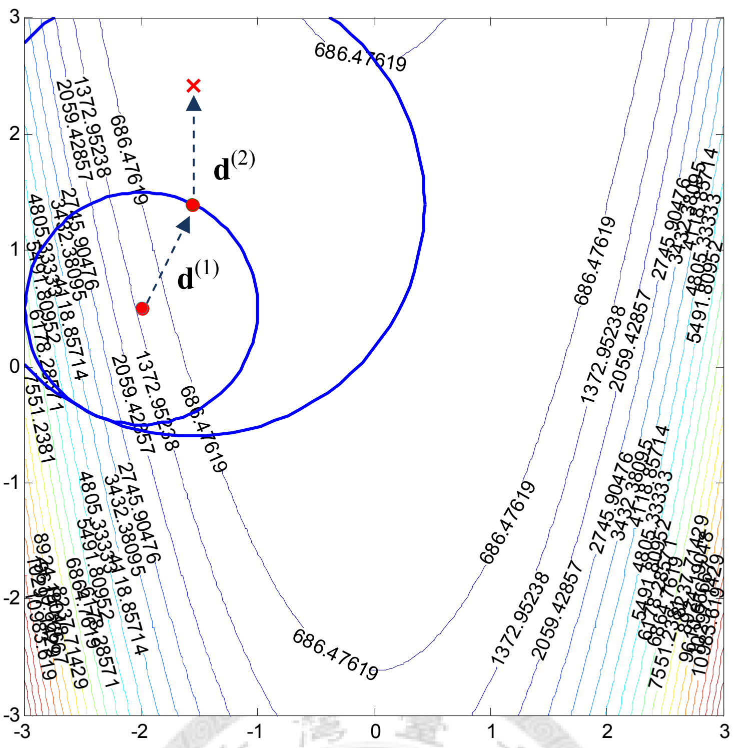


Figure 3.2 The inside solution of Iteration 2.

Iteration 3:

We continue to solve the TRS for $\mathbf{x}^{(3)}$ with the trust region radius $\Delta^{(3)} = 2$, obtain $\mathbf{d}^{(3)} = (0.6147, -1.9031)$. Also evaluate $\rho^{(3)}$ and we have $\rho^{(3)} = -4.0421$. Because $\rho^{(3)}$ is smaller than 0, i.e., the objective value is worse than that of the last iteration. We have to reject $\mathbf{d}^{(3)}$ according to (3.6) and shrink the trust region radius $\Delta^{(4)}$ to be $0.25\Delta^{(3)} = 0.5$ according to (3.2). Figure 3.3 shows the process of Iteration 3.

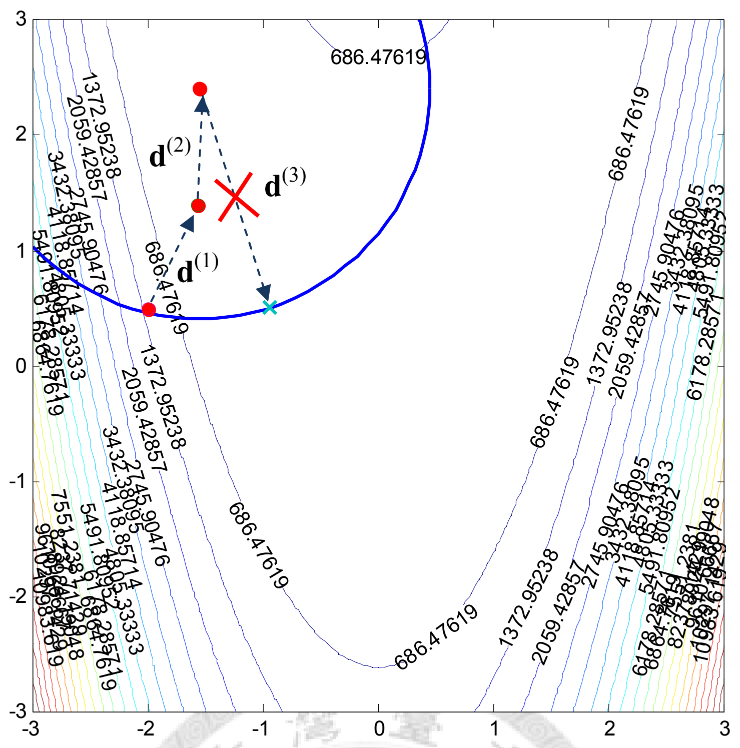


Figure 3.3 The rejected direction of iteration 3.1

Iteration 4:

By solving the TRS for $\mathbf{x}^{(4)} = \mathbf{x}^{(3)}$ with $\Delta^{(4)} = 0.5$ then we have $\mathbf{d}^{(4)} = (0.1553, -0.4752)$ and $\rho^{(4)} = 0.96 > 0.75$. According to (3.5), $\mathbf{x}^{(5)} = \mathbf{x}^{(4)} + \mathbf{d}^{(4)} = (-1.3973, 1.9353)$ and $\Delta^{(5)}$ for $\mathbf{x}^{(5)}$ is enlarged to $2\Delta^{(4)} = 1$ because $\rho^{(4)} = 0.96 > 0.75$ according to (3.3). Figure 3.4 the process of Iteration 4. The rest of the iterations are listed in Table 3.1 and illustrated in Figure 3.5

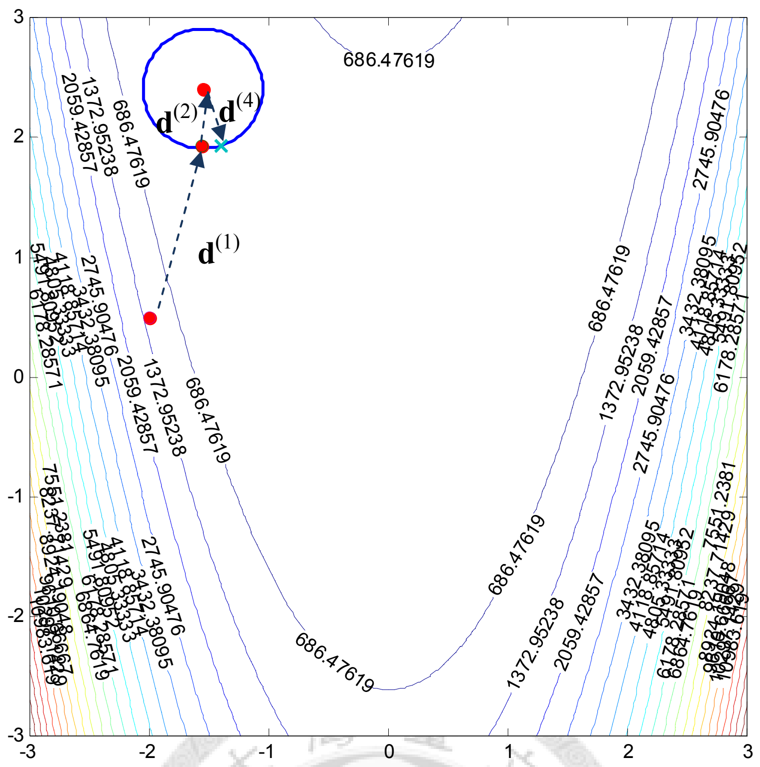


Figure 3.4 The accepted direction of iteration 3.2

Table 3.1 The all iterations of Example 3.1

Iteration	Obj. Value	x_1	x_2	$\Delta^{(k)}$
1	1234.000000	-2.000000	0.500000	1
2	116.476261	-1.564822	1.400344	2
3	6.516007	-1.552647	2.410563	2
4	6.516007	-1.552647	2.410563	0.5
5	5.776516	-1.397305	1.935301	1
6	5.776516	-1.397305	1.935301	0.25
7	5.316688	-1.305694	1.702691	0.5
8	4.579766	-1.123898	1.236912	1
9	4.579766	-1.123898	1.236912	0.25
10	4.049016	-1.010759	1.013977	0.5
11	3.392050	-0.783780	0.568466	1
12	2.681523	-0.608367	0.339341	1
13	2.169678	-0.383544	0.096561	1
14	1.609145	-0.259002	0.051572	1
15	1.609145	-0.259002	0.051572	0.25
16	1.322277	-0.030596	-0.050069	0.25
17	0.888112	0.061413	-0.004694	0.25
18	0.888112	0.061413	-0.004694	0.0625
19	0.773370	0.122282	0.009493	0.125
20	0.599444	0.242763	0.042798	0.25
21	0.437176	0.421899	0.145909	0.25
22	0.253857	0.499832	0.243759	0.25
23	0.201670	0.673872	0.423231	0.25
24	0.079203	0.719329	0.515367	0.25
25	0.061329	0.861224	0.721197	0.25
26	0.012504	0.888424	0.788557	0.25
27	0.009131	0.985619	0.961998	0.25
28	0.000088	0.990596	0.981256	0.25
29	0.000001	0.999954	0.999820	0.25
30	0.000000	0.999999	0.999998	-

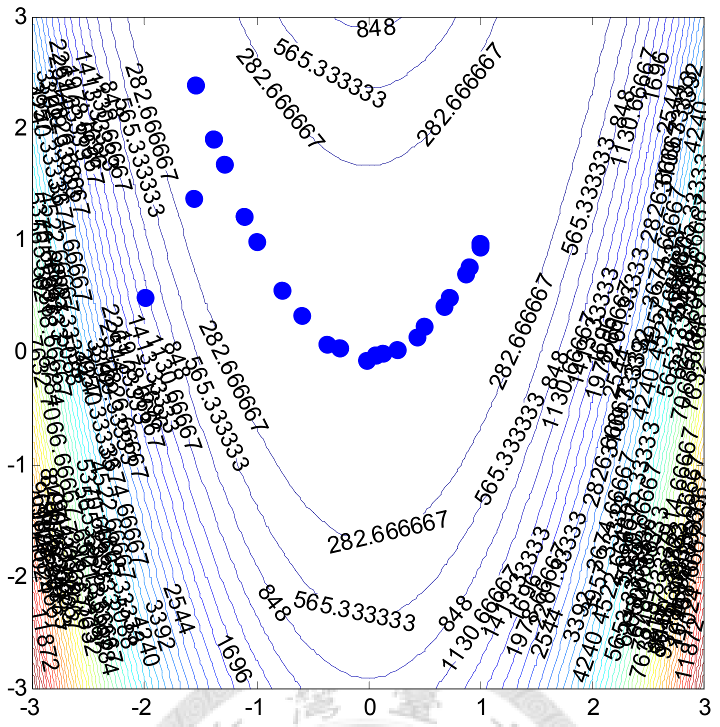


Figure 3.5 The search process of Example 3.1

From Figure 3.5, we see that Algorithm 3.1 avoids the zigzagging phenomenon significantly and the solution also converges to the global minimum (1, 1). But this algorithm is only available for the unconstrained problem. In order to solve the SMOO problem, we develop the Generalized Reduced Trust Region (GRT) method in the next Section.

3.2 Generalized Reduced Trust Region Method

Although Algorithm 3.1 solves the Rosenbrock's function effectively by avoiding the zigzagging phenomenon, further development is still needed to solve the SMOO problem, because there exist bounded constraints and inequality constraints in this problem. In order to consider these constraints, we first add "Line Search" method into our algorithm to solve the constrained problem. The search direction **d** is provided by TR method and the Line Search is then performed along this direction to search for a better solution. Again, using the Rosenbrock's function as an example, the improving direction of iteration 4 in Table 3.1 is equal (−1.397305, 1.935301) − (−1.552647, 2.410563) = (0.155342, −0.475262). The objective value of iteration 5 is equal to 5.776516. If we apply the Line Search here, we further move the solution to (−1.303346, 1.647840) and the objective value becomes 5.564209 as shown in Figure 3.6.

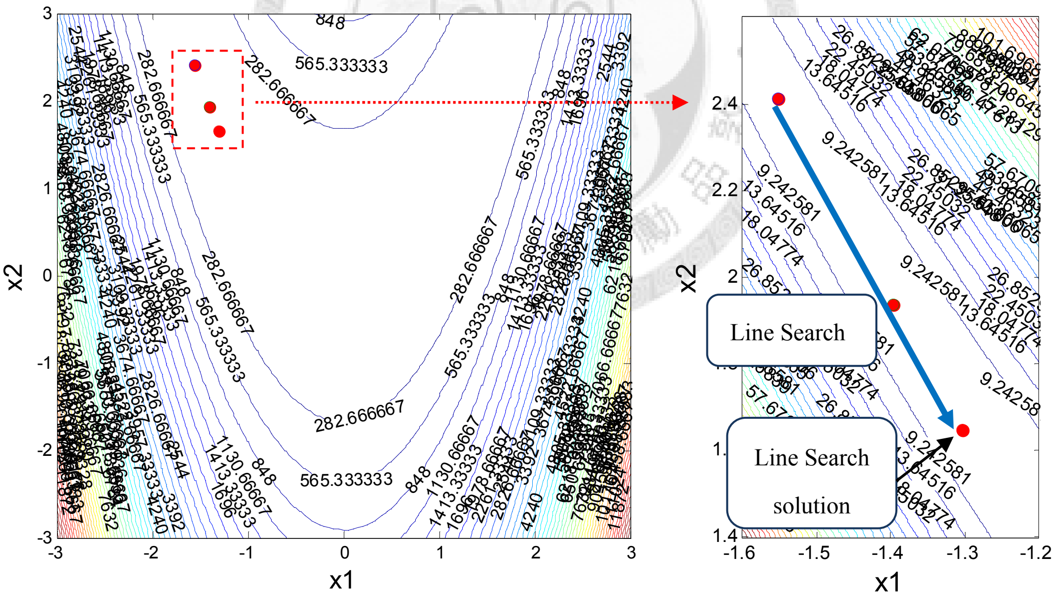


Figure 3.6 The line search solution of iteration 3

The global minimum of the unconstrained optimization problem is changed when we add the constraints into the problem because the solution has to satisfy the constraints, i.e., be inside the feasible region. When the global minimum is not inside the feasible region, the Line Search usually leads to a solution on the constraints. Figure 3.7 shows critical constraints imposed on the Rosenbrock's problem and the solutions generated by the Lien Search with various directions. It can be seen that the Line Search solution all stay on the bounded constraint.

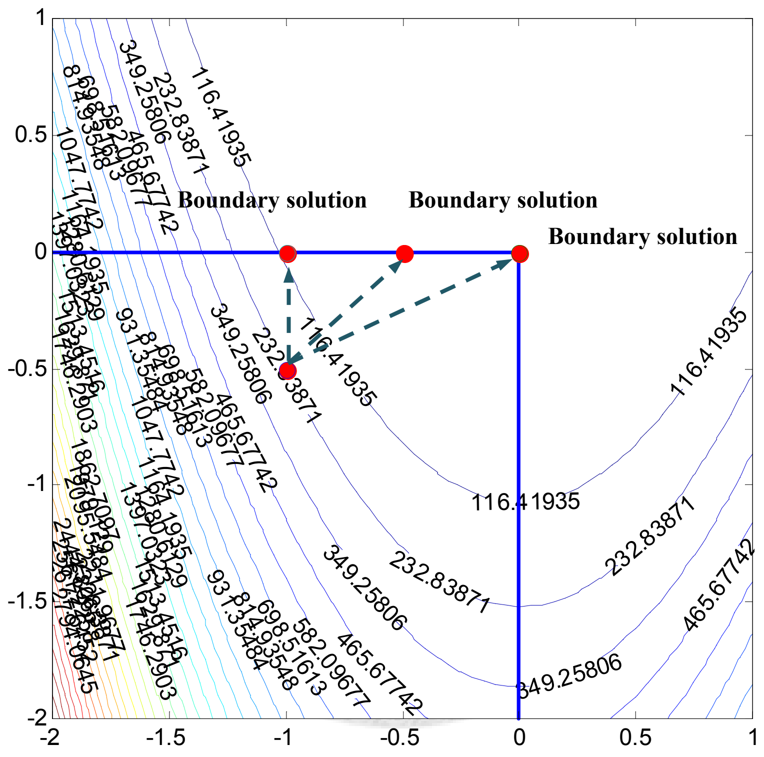


Figure 3.7 Boundary solutions by performing line search

Therefore if we continue to perform the TR method to the boundary solution, we usually get an infeasible direction even if the direction is an improving direction in the unconstrained problem. Figure 3.8 shows the infeasible direction generated by the TR method.

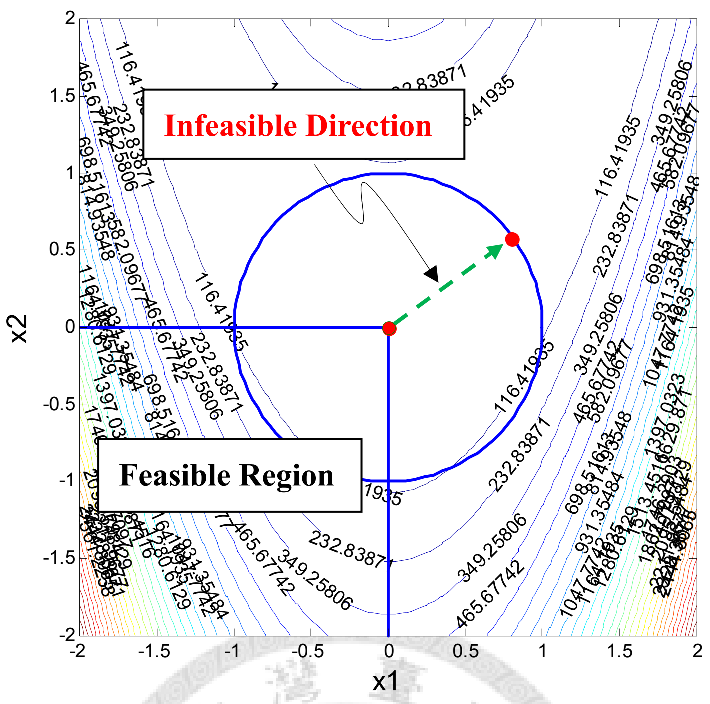


Figure 3.8 The infeasible direction generated by the TR method

By this reason, this research uses the concept of the GRG to develop "Generalized Reduced Trust Region" (GRT) method to generate a feasible direction by considering the constraints. Similar to the GRG method, the GRT search decomposes all variables into the basic and the nonbasic variables, then, finds the improving direction in the "reduced" spaced, found by the nonbasic variables by solving the TR method. After the improving direction of nonbasic variables is found, the direction of the basic variables is in the reduced space then adjusted to meet the linearized constraints. Since, the number of the variables considered by the TR method is only the number of the nonbasic variables. Thus, the computation required is less than that needed by the TR method where all variables have to be accounted for.

We now consider the objective function of (2.2) and add the inequality constraints and the bounded constraints in to our problem. Rewrite the problem by linearizing the constraints as follows.

$$\underset{\mathbf{x}^{(k+1)}}{Minimize}: m(\mathbf{x}) = f(\mathbf{x}^{(k)}) + \boldsymbol{\beta}^{(k)T}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) + \left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right)^T \mathbf{G}^{(k)}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right)/2 . \quad (3.7)$$

$$\begin{aligned} subject\ \ to: &\nabla\mathbf{H}(\mathbf{x}^{(k)})\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = 0; \\ &L_{x_q} \le x_q \le U_{x_q} \quad for \quad q = 1,\ldots,n \end{aligned}$$

, where

$$\nabla\mathbf{H}(\mathbf{x}^{(k)}) = \begin{bmatrix} \left(\mathbf{b}_1 + 2\mathbf{B}_1\mathbf{x}^{(k)}\right)^T \\ \vdots \\ \left(\mathbf{b}_{m_1} + 2\mathbf{B}_{m_1}\mathbf{x}^{(k)}\right)^T \\ a_1^T \\ \vdots \\ a_{m_2}^T \end{bmatrix}$$

is the Jacobian matrix of the binding constraints of (1.5) and (1.7).

We also replace $\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right)$ by the improving direction $\mathbf{d}$ as before. Then, decompose $\mathbf{d}$, $\boldsymbol{\beta}^{(k)}$ and $\nabla\mathbf{H}(\mathbf{x}^{(k)})$ into the basic and the nonbasic variables:

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_B \\ \mathbf{d}_N \end{bmatrix}, \boldsymbol{\beta}_k = \begin{bmatrix} \boldsymbol{\beta}_B^{(k)} \\ \boldsymbol{\beta}_N^{(k)} \end{bmatrix}, \nabla\mathbf{H}(\mathbf{x}_k) = \left[\nabla_B\mathbf{H}\left(\mathbf{x}^{(k)}\right) \quad \nabla_N\mathbf{H}\left(\mathbf{x}^{(k)}\right)\right].$$

The method to decompose the variables will be introduced later. Different from the GRG method, the GRT search should decompose the Hessian Matrix $\mathbf{G}^{(k)}$ into four sets:

$$\mathbf{G}^{(k)} = \begin{array}{c|cc} & B & N \\ \hline B & \mathbf{G}_{BB}^{(k)} & \mathbf{G}_{BN}^{(k)} \\ N & \mathbf{G}_{NB}^{(k)} & \mathbf{G}_{NN}^{(k)} \end{array}.$$

The nonlinear problem (3.7) can be then generalized to:

$$\begin{aligned} \underset{\mathbf{d}_B,\mathbf{d}_N}{Minimize} \quad & f(\mathbf{x}_k) + \boldsymbol{\beta}_B^{(k)T}\mathbf{d}_B + \boldsymbol{\beta}_N^{(k)T}\mathbf{d}_N + \frac{\mathbf{d}_B{}^T\mathbf{G}_{BB}^{(k)}\mathbf{d}_B}{2} + \frac{\mathbf{d}_B{}^T\mathbf{G}_{BN}^{(k)}\mathbf{d}_N}{2} \\ & + \frac{\mathbf{d}_N{}^T\mathbf{G}_{NB}^{(k)}\mathbf{d}_B}{2} + \frac{\mathbf{d}_N{}^T\mathbf{G}_{NN}^{(k)}\mathbf{d}_N}{2} \\ subject\ to \quad & \nabla_B\mathbf{H}(\mathbf{x}^{(k)})\mathbf{d}_B + \nabla_N\mathbf{H}(\mathbf{x}^{(k)})\mathbf{d}_N = 0 \\ & L_{x_q} \le x_q^k \le U_{x_q} \quad for \quad q = 1,\ldots,n. \end{aligned} \tag{3.8}$$

In (3.8), the equality constraints could be rewritten as

$$\mathbf{d}_B = -\nabla_B\mathbf{H}(\mathbf{x}^{(k)})^{-1}\nabla_N\mathbf{H}(\mathbf{x}^{(k)})\mathbf{d}_N \tag{3.9}$$

(3.9) should hold for the constraints to be met. Substituting (3.9) into the objective function of (3.8), the objective function is reduced to be the function of nonbasic variables. The nonlinear programming (3.8) becomes:

$$\underset{\mathbf{x}_N^{(k+1)}}{Minimize} \quad f(\mathbf{x}^{(k)}) + \mathbf{b}_R^T\left(\mathbf{x}_N^{(k+1)} - \mathbf{x}_N^{(k)}\right) + \left(\mathbf{x}_N^{(k+1)} - \mathbf{x}_N^{(k)}\right)^T\mathbf{B}_R\left(\mathbf{x}_N^{(k+1)} - \mathbf{x}_N^{(k)}\right)/2 \tag{3.10}$$

$$subject\ to \quad L_{x_q} \le x_q^{(k)} \le U_{x_q} \quad for \quad q = 1,\ldots,n$$

, where

$$\mathbf{b}_R = \boldsymbol{\beta}_N^{(k)} - \nabla_N\mathbf{H}\left(\mathbf{x}^{(k)}\right)^T\nabla_B\mathbf{H}\left(\mathbf{x}^{(k)}\right)^{-1}\boldsymbol{\beta}_B^{(k)} \tag{3.11}$$

$$\mathbf{B}_R = \mathbf{G}_{NN}^{(k)} + \nabla_N \mathbf{H}\left(\mathbf{x}^{(k)}\right)^T \nabla_B \mathbf{H}\left(\mathbf{x}^{(k)}\right)^{-1} \mathbf{G}_{BB}^{(k)} \nabla_B \mathbf{H}\left(\mathbf{x}^{(k)}\right)^{-1} \nabla_N \mathbf{H}\left(\mathbf{x}^{(k)}\right)$$

$$- 2\nabla_N \mathbf{H}\left(\mathbf{x}^{(k)}\right)^T \nabla_B \mathbf{H}\left(\mathbf{x}^{(k)}\right)^{-1} \mathbf{G}_{BN}^{(k)} \tag{3.12}$$

The TR method in Section 3.1 is then applied to (3.10) and generates the improving direction with the nonbasic variables subject to a sphere constraint:

$$\begin{aligned} &\underset{\mathbf{d}_N}{Minimize} : f(\mathbf{x}^{(k)}) + \mathbf{b}_R^T \mathbf{d}_N + \mathbf{d}_N{}^T \mathbf{B}_R \mathbf{d}_N / 2. \\ &subject\ to : \mathbf{d}_N{}^T \mathbf{d}_N \le \Delta^{(k)2}; \\ &\qquad L_{x_q} \le x_q^{(k)} \le U_{x_q} \quad for \quad q = 1, \dots, n \end{aligned} \tag{3.13}$$

, where $\mathbf{d}_N = \left(\mathbf{x}_N^{(k+1)} - \mathbf{x}_N^{(k)}\right)$ the improving direction in the reduced space.

Moreover, we need to consider the upper bound and the lower bound of the decision variables. The improving direction of the nonbasic variable $\mathbf{d}_N$ should be further adjusted by:

$$x_q^{(k+1)} - x_q^{(k)} = d_q = 0 \qquad \text{if } x_q^{(k)} = L_{x_q} \text{ and } d_q < 0, \text{or } x_q^{(k)} = U_{x_q} \text{ and } d_q > 0 \tag{3.14}$$

, where d_q is the q-th component of $\mathbf{d}_N$, and $x_q^{(k)}$ is the q-th component of $\mathbf{x}^{(k)}$.

With the above adjustment, the improving direction of the basic variables can be then is calculated by (3.9). That ensures that the improving direction is feasible and effective.

At each iteration of the GRT search, $\mathbf{x}^{(k)}$ is partitioned into basic variables $\mathbf{x}_B^{(k)}$ and nonbasic variables $\mathbf{x}_N^{(k)}$, and $\nabla\mathbf{H}(\mathbf{x}^{(k)})$ is also partitioned into $\nabla_B\mathbf{H}(\mathbf{x}^{(k)})$ and $\nabla_N\mathbf{H}(\mathbf{x}^{(k)})$. Here, the number of the basic variables is the number of the binding constraints, and the basic variables $\mathbf{x}_B^{(k)}$ should satisfy two requirements. First, $\nabla_B \mathbf{H}(\mathbf{x}^{(k)})$, the bases of $\nabla \mathbf{H}(\mathbf{x}^{(k)})$, should be nonsingular. It ensures that the (3.9) holds. Second, $\mathbf{x}_B^{(k)}$ should be larger than $\mathbf{L}_{xB}$ and smaller than $\mathbf{U}_{xB}$. Because once the improving direction of the nonbasic variables is determined, the direction of the basic variables is indirectly generated by (3.9). If some elements of $\mathbf{x}_B^{(k)}$ are on the upper bounds or the lower bounds, no feasible solutions after Line Search can be found through the GRT direction and the solution will be stuck at the boundary.

To satisfy the above two requirements, we first rank all variables by their distances to the bounds. The distances between variables and bounds are computed as follows:

$$distance_q = \begin{cases} U_{x_q} - x_q^{(k)} & \text{if } x_q^{(k)} \notin \text{slack variables and } \left.\dfrac{\partial f(\mathbf{x})}{\partial x_q}\right|_{\mathbf{x}=\mathbf{x}^{(k)}} < 0 \\ x_q^{(k)} - L_{x_q} & \text{if } x_q^{(k)} \notin \text{slack variables and } \left.\dfrac{\partial f(\mathbf{x})}{\partial x_q}\right|_{\mathbf{x}=\mathbf{x}^{(k)}} > 0 \\ 0 & \text{if } x_q^{(k)} \in \text{slack variables} \end{cases} \tag{3.15}$$

, where distance$_q$ is the distance of q-th variable to its bound.

We would like to choose the variables farther from the bounds as the basic variables. To do this, we rearrange columns of $\nabla \mathbf{H}(\mathbf{x}^{(k)})$ by *distance$_q$* and choose the bases from the front columns. In addition, we want to ensure that the chosen bases are independent. Choosing independent columns can be done by Gaussian elimination [3].

Pivots obtained by Gaussian elimination will locate the independent columns. The second method to choose independent columns is rather straightforward. Starting from the first column of the rearranged $\nabla\mathbf{H}(\mathbf{x}^{(k)})$, every time a column is picked its independence from the chosen columns will be checked with "Singular Value Decomposition (SVD)" to prevent singularity. We observe that the results by the two methods are similar.

The biggest difference between the GRT search and the GRG search is that we need to specify the trust region radius for the GRT search. Even the algorithm 3.1 provides a strategy for updating the trust-region radius, we still need to make some modifications for constrained problems and to make the algorithm more intelligent. In this research, the modified algorithm is proposed as follows:

```
Algorithm 3.2
Input:
    $\mathbf{x}^{(k)}$: a given current point
    $j$: the iteration index of trust region radius adjustment algorithm which is set
      to be 0
    $\rho^{(k)}$: the response improvement ratio of $k$-th search iteration.
    $\Delta^{(k)}$: a given trust region radius for current point $\mathbf{x}^{(k)}$
    $\eta$: a radio measures how we trust this step and $\eta \in \left[0, \frac{1}{4}\right)$
Output:
    d : an improving direction to current point $\mathbf{x}^{(k)}$
    $\Delta^{(k+1)}$: the trust region radius of next point $\mathbf{x}^{(k+1)}$
Procedure Trust Region Radius Adjustment Algorithm
Begin
```

```
    Repeat Until $\rho^{(k)} > \eta$
        Perform the algorithm 2.1 to obtain improving direction $\mathbf{d}$.
        Evaluate response improvement ratio $\rho^{(k)}$ by (3.1).
        If $\rho_j < 0.25$ then
            $\Delta^{(k+1)} \leftarrow \Delta^{(k)} \times \left(0.25 + 10^{\left(\rho_j - 0.25 + \log(0.75)\right)}\right)$        (3.16)
        Else if $\rho^{(k)} < 0.75$ then
            $\Delta^{(k+1)} \leftarrow \Delta^{(k)}$        (3.17)
        Else If $\mathbf{d}$ is a boundary solution then
            $\Delta^{(k+1)} \leftarrow \min\left(\Delta^{(k)} \times \left(2 - 10^{\left(-\rho_j + 0.75\right)}\right), \hat{\Delta}\right)$        (3.18)
        End If
        $k \leftarrow k + 1$
    End Repeat
End
```

In Algorithm 3.2, we establish a mechanism to decide trust region radius dynamically. This algorithm is supposed to be more intelligent than the Algorithm 3.1. First, we set $\widehat{\Delta}$ to be the largest distance from the current point $\mathbf{x}^{(k)}$ to the constraints boundary. Second, instead of shrinking the trust region radius to one-fourth, we dynamically shrink the radius according to the degree of the improvement ratio $\rho^{(k)}$ by multiplying $0.25 + 10^{\left(\rho^{(k)} - 0.25 + \log(0.75)\right)}$. Similarly, instead of enlarging the radius twice as large we dynamically enlarge it by multiplying $\left(2 - 10^{\left(-\rho^{(k)}\right)}\right)$. To explain the radius adjustment mechanism, Figure 3.9 and Figure 3.10 show how the two multipliers change as ρ decreases or increases. The multiplier in (3.16) maps $\left\{\rho^{(k)} \mid -\infty < \rho^{(k)} < 0.25\right\}$ to a shrinking factor $\left\{\tilde{\rho}^{(k)} \mid 0.25 < \tilde{\rho}^{(k)} < 1\right\}$; i.e., as $\rho^{(k)}$ is a large negative value the radius for the next iteration will be approaching $0.25 \times \Delta^{(k)}$. The multiplier in (3.18) maps $\left\{\rho^{(k)} \mid 0.75 < \rho^{(k)} < \infty\right\}$ to an enlarging factor $\left\{\tilde{\rho}^{(k)} \mid 1 < \tilde{\rho}^{(k)} < 2\right\}$; i.e., when ρ is greater than 0.75 and becomes large the radius for the next iteration will be approaching $2 \times \Delta^{(k)}$. Thus, with the help of the two multipliers, we can adjust the trust region radius dynamically.

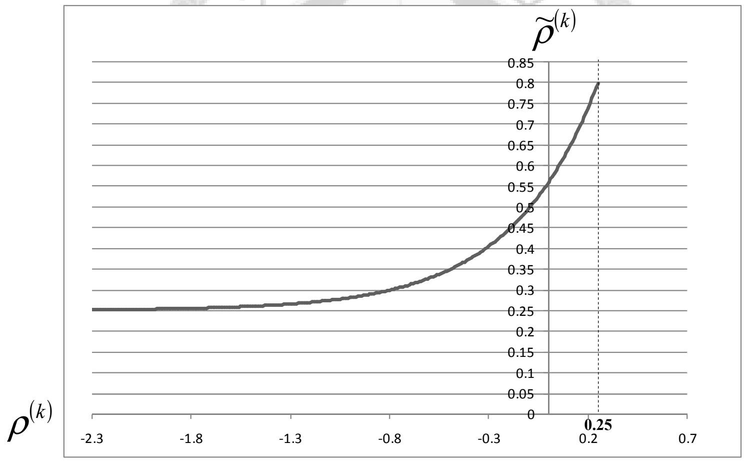


Figure 3.9 The mapping of the shrinking factor in (3.16)

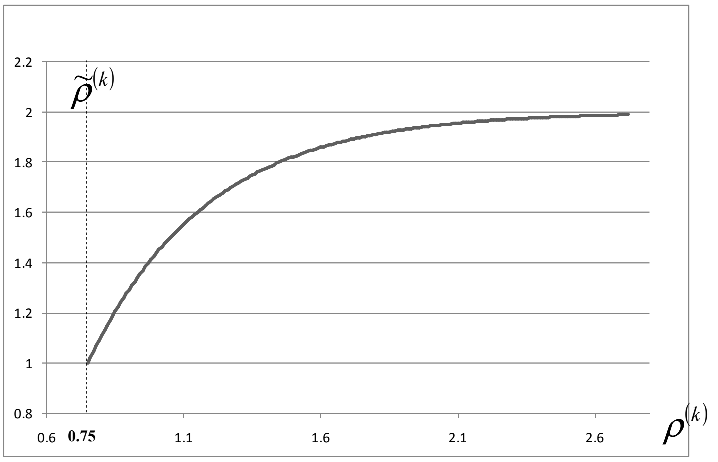


Figure 3.10 The mapping of the enlarging factor in (3.18)

Although our algorithm consider the constraints and find the improving direction in the reduced space but there still exists another problem, that is, the linearization of nonlinear constraints. GRG deals with the problem by Newton-Raphson method. As we discuss in GRG algorithm, the Newton-Raphson maintain the feasibility of the solution. However, there are two disadvantages in the Newton-Raphson method. First, the Newton-Raphson method relaxes the feasibility by allowing solution deviating slightly from the constraints. Determining the tolerance of feasibility ε is an issue. Similarly, determining the initial step length of nonbasic variables θ isn't easy. Second, the computation required by the Newton-Raphson method is intensive. In particular, the term $\nabla_B \mathbf{h}(\mathbf{y}^{(t)}, \widetilde{\mathbf{x}}_N^{(k)})$ in step 3.2 may not be invertible. Based on the above reasons, we replace Newton-Raphson method by the Line Search. By the Line Search, each solution is feasible and acceptable in the actual problem.

There is a strong assumption in the GRG method or the GRT search. Both methods linearize the constraints. However, the feasible region of nonlinear problem (1.8) may not be a polyhedron. Because we use the Line Search instead of Newton-Raphson method, there may be no feasible solutions along the linearized constraints. For example, Figure 3.11 shows an initial solution on the quadratic constraint boundary. The linearized constraint is actually the tangent of the curve. The direction derived by both the GRG and the GRT search are the direction along the tangent, but the tangent is outside the feasible region except the point of contact. This study uses the ideas of the GRR algorithm, that is, we combine the Zoutendijk's method into the algorithm for this issue.

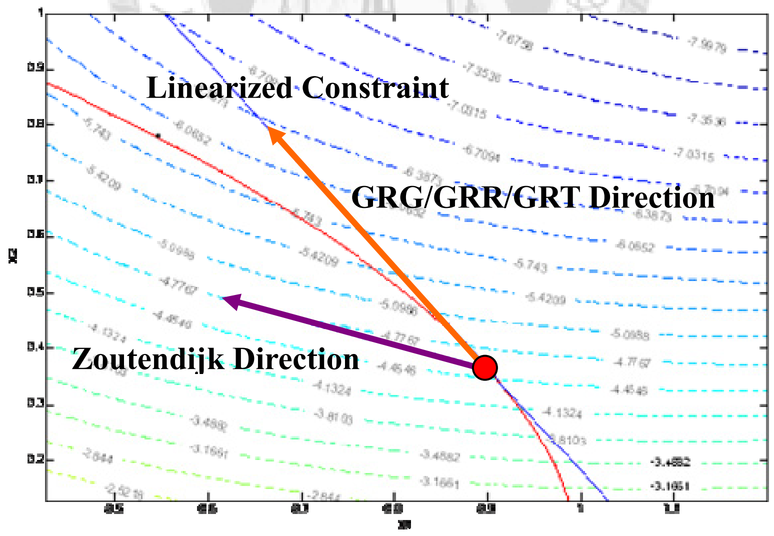


Figure 3.11 Example of Zoutendijk's method

As explained in Section 1.2.3, the Zoutendijk's method generates an improving direction such that the angle between this direction and the constraint tangent must be greater than zero and within feasible region. However, the direction found by the Zoutendijk's method is less effective. When there are feasible solutions after the Line Search along the direction found by the GRG method or the GRT search, the direction should be preferred. Otherwise, the Zoutendijk's method is applied only when the Line Search fails to improve. The algorithm combining the Zoutendijk's method will be shown as follows.

Now we summarize the algorithm of Generalized Reduced Trust Region search method as follows:

- Step 1:

Let $\mathbf{x}^{(k)}$ be a feasible solution at the *k*-th search step. Choose a threshold $\underline{e} > 0$. Check the binding constraints and add slack variables (the slack variables are zeros) to the binding inequality constraints. Set the number of basic variables equal to the number of the binding constraints. Approximate the objective function as a quadratic function and linearize the binding constraints as the formulation of (3.7).

- Step 2:

Compute *distance*$_q$ by (3.15). Rearrange columns of $\nabla\mathbf{H}(\mathbf{x}^{(k)})$ in (3.7) by *distance*$_q$ and choose the independent bases from the columns in the front as basic variables, the other variables as nonbasic variables. Then, decompose all matrices and vectors into the set of basic variables and the set of nonbasic variables. In particular, the Hessian matrix $\mathbf{G}^{(k)}$ is decomposed into $\mathbf{G}_{BB}^{(k)}$, $\mathbf{G}_{BN}^{(k)}$, $\mathbf{G}_{NB}^{(k)}$, and $\mathbf{G}_{NN}^{(k)}$.

- Step 3:

Perform Algorithm 3.2 to get an improving direction $\mathbf{d}_N$ by solving (3.13). Adjust $\mathbf{d}_N$ according to (3.14). Calculate $\mathbf{d}_B$ by (3.9). Combine $\mathbf{d}_N$ and $\mathbf{d}_B$ as the improving direction $\mathbf{d}$.

- Step 4.1:

Do Line Search from $\mathbf{x}^{(k)}$ along direction $\mathbf{d}$ to find $\mathbf{x}^{(k+1)}$ in the feasible region of (1.8). If there are no improving solutions after performing the Line Search, go to step 4.2; else take the feasible solution to replace $\mathbf{x}^{(k+1)}$. Go to step 5.

- Step 4.2

Calculate Zoutendijk's steps $\mathbf{d}_Z$ according the Zoutendijk's method in Subsection 1.2.3 to replace **d.** Do Line Search from $\mathbf{x}^{(k)}$ along direction $\mathbf{d}_Z$ in the feasible region of (1.8). Go to step 5.

- Step 5:

If $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)}) < \underline{e}$, stop and $\mathbf{x}^* = \mathbf{x}^{(k+1)}$; otherwise, go to step 1.

Here we use two examples to show the process of the GRT search method and a test problem to verify the GRT search algorithm. First we use the same example (Example 1.2) of the GRG method to show the search direction in the reduced space. Second we use the Rosenbrock's function as the example to show the GRT direction could be more effective than the GRR direction and the GRT algorithm could also avoid the three drawbacks of the GRR search method.

Figure 3.12 shows the improving direction (dashed-line) in the reduced space and modified improving direction (bolded-line). Figure 3.13 shows the improving direction (dashed-line) in the original space is infeasible thus we have to replace this direction by Zoutendijk's direction (bolded-line).

Example 3.2

Initial Setting: set trust region radius to be 0.5 at current point.

$$\textit{Minimize}: f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2,$$

$$\textit{subject to}: x_1 + x_2 \le 2;$$

$$x_1 + \frac{17}{6}x_2^2 \le 2.8;$$

$$x_1, x_2 \ge 0.$$

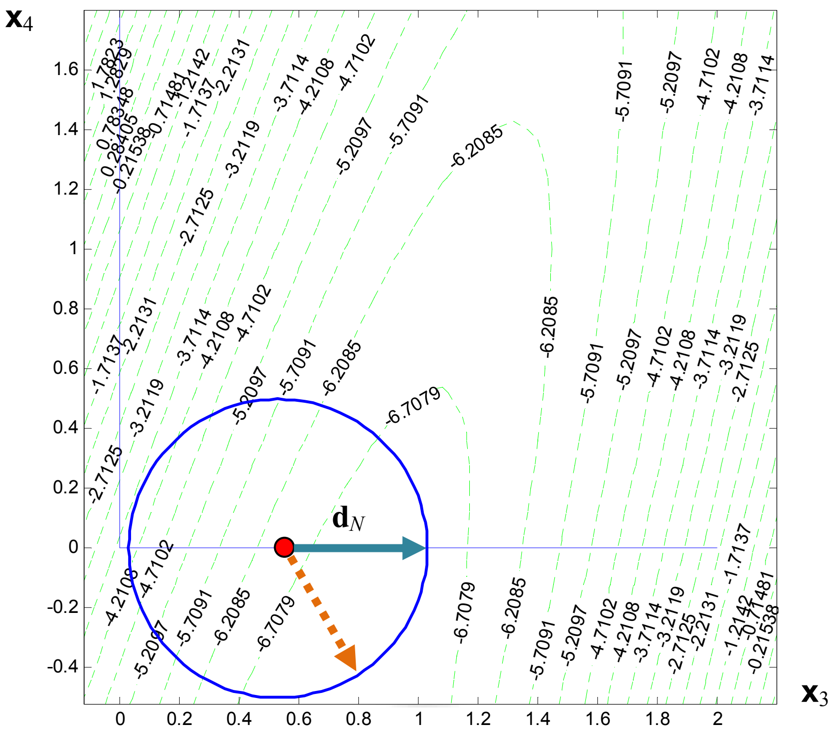


Figure 3.12 The improving direction in the reduced space

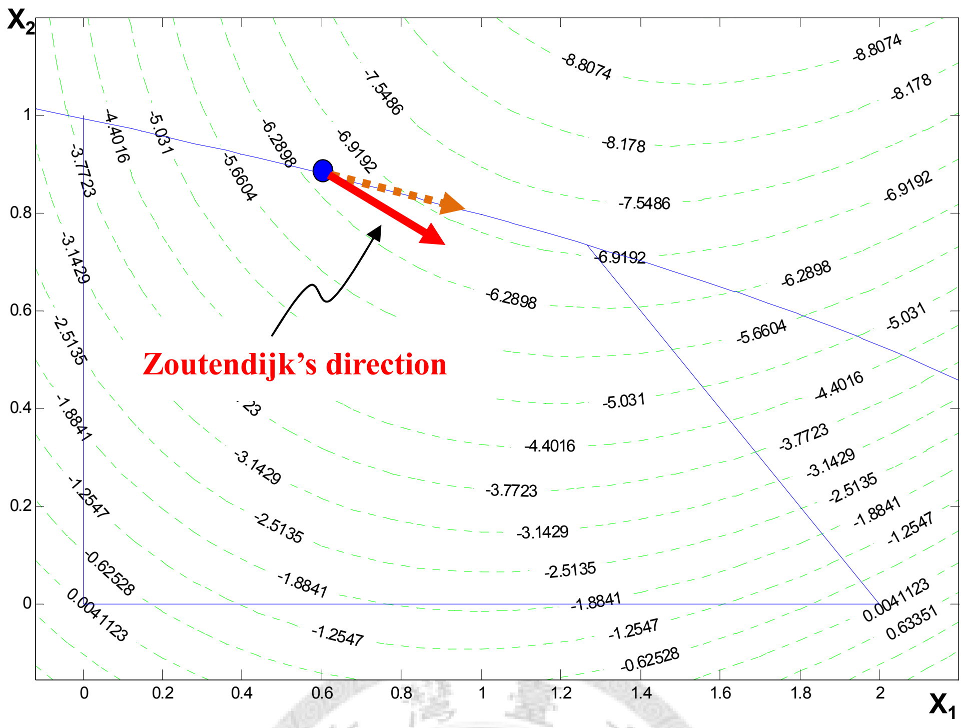


Figure 3.13 The improving direction in the original space

In the following example, we demonstrate that the GRT algorithm is more effective than the GRR algorithm. The first purpose of the example is to show that the advantage of GRT search algorithm consider the solution inside the trust region. The second purpose is to show the GRT algorithm avoid two drawbacks, i.e., (1) the hessian matrix is singular or near singular; (2) the approximation issue of the quadratic model.

Example 3.3:

Initial point: $(x_1, x_2) = (1.4213545{,}2.0252484)$

Minimize : $(1-x_1)^2+100\times(x_2-x_1^2)^2$, *Subject to* : $-2 \le x_1 \le 2$; $0 \le x_2 \le 4$.

The Hessian matrix **G** is equal to $\begin{bmatrix} 808.0994 & -284.2709 \\ -284.2709 & 100.0000 \end{bmatrix}$; the eigenvalue of **G** is equal $\begin{bmatrix} 908.09948 \\ 0.000004 \end{bmatrix}$. Because the smallest eigenvalue almost equals to zero, the Hessian matrix **G** is almost a singular matrix. We roughly set the trust region radius to be 1, and then perform the GRT search algorithm to find the next point. Table 3.2 shows the result by solving the TRS. With the response improving ratio $\rho^{(1)} < 0.25$, the approximation of first iteration is poor. Thus we shrink the trust region radius and set it to be $1 \times \left(0.25 + 10^{(-3.54233 - 0.25 + \log(0.75))}\right) = 0.250087$. The response improving ratio of the second iteration $\rho^{(2)}$ can be accepted and we also get an improving objective value. By the way, the algorithm only needs a few number of iteration for solving TRS.

Table 3.2 GRT search result of example 3.2

Interaion	Lagrangian multiplier	Corresponding Obj. Value	ρ_k	Radius
1	−1.3981900E−01	1.175425E+00	−3.542328	1
2	−5.2386169E−01	1.155079E-01	0.849160	0.2500887

Now we use the GRR algorithm to search the optimal Lagrangian multiplier by (1.24). The GRR algorithm wants to find a Lagrangian multiplier which minimizes the original objective by adjusting the Lagrangian multiplier by (1.24). With δ and α are set to be 10 and 100, Table 3.3 shows the iterative results of the GRR search method. Compare the two result generated by two method, we find that the GRT uses less iterations and also gets a better objective value.

Table 3.3 GRR search result of example 3.3

Interaion	Lagrangian multiplier	Corresponding Obj. Value
1	4.6806226E–06	9.276490E+25
2	4.2078324E–06	6.335962E+21
3	–5.2006918E–07	6.110703E+17
4	–4.7799085E–05	6.088703E+13
5	–5.2058925E–04	6.086232E+09
6	–5.2484909E–03	6.083651E+05
7	–5.2527507E–02	6.055326E+01
8	–5.2531767E–01	1.156180E-01

Finally this thesis will solve the test problem and the cases by these methods: "Generalized Reduced Gradient method and Zoutendijk method" [24], "Generalized Reduced Ridge method and Zoutendijk method" [24], "Generalized Reduced Trust Region method and Zoutendijk method", and commercial software "Lingo". Solutions by three different methods will be also discussed. Table 3.2 shows the methods with different settings are compared in our research. Moreover, we consider the same method [24] to generate the initial points.

Table 3.2 The methods compared in our research

Methods
GRG + Zoutendijk
GRR + Zoutendijk (Δ=100, α=10)
GRR + Zoutendijk (Δ=100, α=20)
GRR + Zoutendijk (Δ=100, α=30)
GRT + Zoutendijk with Conventional Radius Adjustment (CRA)
GRT + Zoutendijk with Dynamic Radius Adjustment (DRA)
Lingo (Steepest Edge)
Lingo (SLP Directions)
Lingo (Steepest + SLP)

In the Subsection 1.3.1, we use the Rosenbrock's function to show the strong zigzagging phenomena by using the GRG search. This study uses the same problem to test the search methods listed in Table 3.2. We select four corner points in the feasible region to be the initial point and suppose the terminal criterion is less than 10^{-6} between two iterative objective values or more than seven hundred steps of search. The initial points are listed in Table 3.3 and the local search results are listed in Table 3.4.

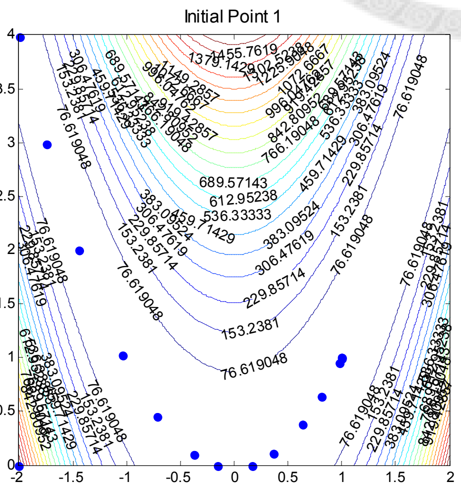
Table 3.3 The initial points of the Rosenbrock's function

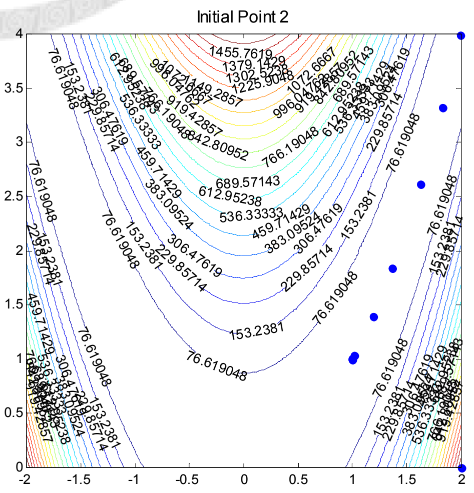
Index	x_1	x_2
1	-2	0
2	2	0
3	-2	4
4	2	4

Table 3.4 The Results of Rosenbrock's function (Local Search)

Methods	Average Objective Value	Best Objective Value	Average Number of Iterations	Average Computing Time (seconds)
GRG + Zoutendijk	5.0039460E-01	2.924208E-04	352	0.12
GRR + Zoutendijk (Δ=100, α=10)	1.4376088E-07	2.863833E-08	117	0.07
GRR + Zoutendijk (Δ=100, α=20)	7.2339749E-07	6.192558E-28	101.25	0.04
GRR + Zoutendijk (Δ=100, α=30)	1.6651505E-07	6.192558E-28	186.75	0.07
GRT + Zoutendijk with CRA	**8.8623310E-18**	**1.467099E-19**	**13**	**0.03**
GRT + Zoutendijk with DRA	**9.9622481E-18**	**7.101449E-19**	**13**	**0.02**
Lingo (Steepest Edge)	2.2556653E-08	2.254358E-08	159.75	< 1
Lingo (SLP Directions)	2.2560715E-08	2.255993E-08	146	< 1
Lingo (Steepest + SLP)	2.2556500E-08	2.254766E-08	147.75	< 1

The "GRT + Zoutendijk" methods have better performance in objective value and computing time against the "GRR + Zoutendijk" methods and the methods of Lingo. Moreover, the "GRR + Zoutendijk" methods are very sensitive to the parameters. This is one of the drawbacks of the algorithm with "GRR + Zoutendijk" approach as we discussed before. In order to verify the "GRT + Zoutendijk" methods avoid the zigzagging phenomena, we plot the search processes of all initial points by using the "GRT + Zoutendijk" method with $\eta = 0.25$.





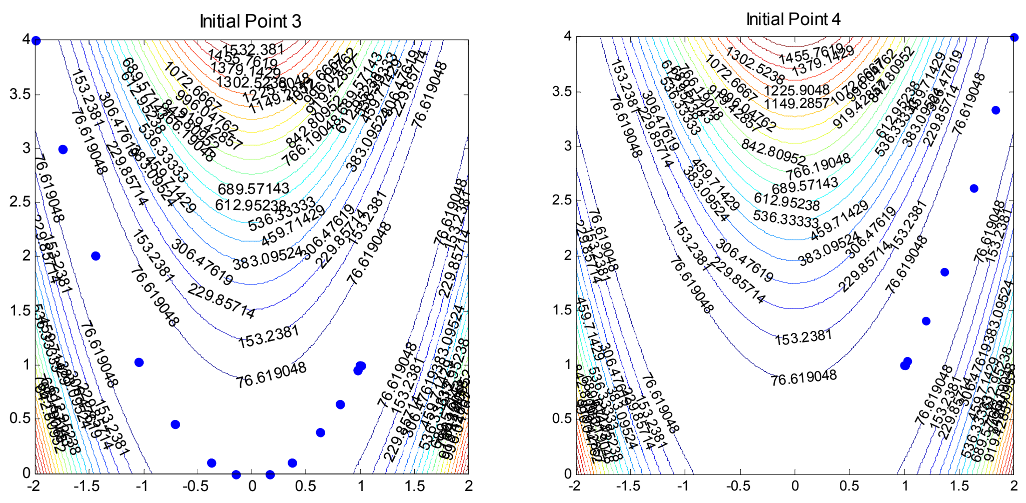


Figure 3.14 The search process of the "GRT + Zoutendijk" method

Figure 3.14 shows that the "GRT + Zoutendijk" methods avoid the zigzagging phenomena successfully and the search path advance along the inclined trough of the Rosenbrock's function.

3.3 Convergence Proof of Generalized Reduced Trust Region Method

In this Section, we propose a convergence proof of the GRT search method based on the Algorithm 3.1. The convergence combines two convergence theories. The first theory is about trust region method. It shows that the sequence of gradient $\{\boldsymbol{\beta}^{(k)}\}$ generated by Algorithm 3.1 has an accumulation point at zero, and in fact converges to zero when η is strictly positive. Under this condition, another theorem about the convergence of GRG claims that the $\boldsymbol{\beta}^{(k)}$ is equal to zero if and only if the current point $\mathbf{x}^{(k)}$ is a KKT point. We then start the convergence analysis by obtaining an estimate of the decrease in the model function m in (3.7) two-dimensional subspace minimization algorithms and Steihaug's algorithm produce approximation solution **d** of the (3.7) that satisfy the following estimate of decrease in the model function [14]:

$$m(0)-m(\mathbf{d})\geq c_1\left\|\boldsymbol{\beta}^{(k)}\right\|\min\left(\Delta^{(k)},\frac{\left\|\boldsymbol{\beta}^{(k)}\right\|}{\left\|\mathbf{G}^{(k)}\right\|}\right) \tag{3.19}$$

We assume throughout that the Hessian matrix $\mathbf{G}^{(k)}$ in (3.7) is uniformly bounded in norm, and that *f* in (1.8) is bounded below on the level set

$$S:=\{s\,|\,f(\mathbf{x})\leq f(\mathbf{x}_0)\}. \tag{3.20}$$

We define an open neighborhood of this set by

$$S(R_0):=\{\mathbf{x}\,|\,\left\|\mathbf{x}-\mathbf{z}\right\|<R_0 \text{ for some } \mathbf{z}\in S\}, \tag{3.21}$$

where R_0 is a positive constant.

To allow our results to be applied more generally, we also allow the length of the approximate solution **d** of (3.7) to exceed the trust-region bound, provided that it stays within some fixed multiple of the bound; that is,

$$\left\|\mathbf{d}\right\|\leq\gamma\,\Delta^{(k)} \text{ for some constant } \gamma\geq 1. \tag{3.22}$$

The following result deals with the case η = 0.

Theorem 3.1 [14]

Let $\eta = 0$ in Algorithm 3.1. Suppose that $\left\|\mathbf{G}^{(k)}\right\|\leq\chi$ for some constant χ, that f is bounded below on the level set S defined by (3.21) and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all approximate *solutions of (3.13) satisfy the inequalities (3.20) and (3.22), for some positive constants c_1 and γ. We then have*

$$\liminf_{k\to\infty} \left\|\boldsymbol{\beta}^{(k)}\right\| = 0. \qquad (3.23)$$

Proof:

See also Appendix C.

Theorem 3.2 [2]

*Consider the problem (3.7) without the bounded constraints to minimize $m(\mathbf{x})$ subject to $\nabla\mathbf{H}(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)})=0$, $\mathbf{x}\geq 0$. Let **x** be a feasible solution such that $\mathbf{x}^T=(\mathbf{x}_B^T,\mathbf{x}_N^T)$ and $\mathbf{x}_B$ > **0**, where $\nabla\mathbf{H}(\mathbf{x}^{(k)})$ is decomposed into $[\nabla_B\mathbf{H}(\mathbf{x}^{(k)})$ $\nabla_N\mathbf{H}(\mathbf{x}^{(k)})]$ and $\nabla_B\mathbf{H}(\mathbf{x}^{(k)})$ is **an** invertible matrix. Suppose that m is differentiable at **x**, and let $\mathbf{r}^T=\boldsymbol{\beta}^{(k)}-\boldsymbol{\beta}_B^{(k)T}\nabla_B\mathbf{H}(\mathbf{x}^{(k)})^{-1}\nabla\mathbf{H}(\mathbf{x}^{(k)})$. Let $\mathbf{d}^T=(\mathbf{x}-\mathbf{x}^{(k)})^T=[\mathbf{d}_B^T,\mathbf{d}_N^T]$ be the direction formed as follows. For each nonbasic component j, let $\mathbf{d}_j=-\mathbf{r}_j$ if $\mathbf{r}_j\leq 0$ and $\mathbf{d}_j=-\mathbf{x}_j\mathbf{r}_j$ if $\mathbf{r}_j>0$, and let $\mathbf{d}_B=-\nabla_B\mathbf{H}(\mathbf{x}^{(k)})^{-1}\nabla_N\mathbf{H}(\mathbf{x}^{(k)})\mathbf{d}_N$. If $\mathbf{d}\neq 0$, then **d** is an improving feasible direction. Furthermore, **d = 0** if and only if **x** is a KKT point.*

Corollary 3.1

Consider problem (3.13). The Generalized Reduced Trust Region search algorithm will reach a KKT point at $\mathbf{x}^{(k)}$ as $k\to\infty$.

Proof: We will have $\liminf_{k\to\infty}\|\mathbf{b}_R\| = 0$ for problem (3.13) based on Theorem 3.1. Since $\mathbf{d}_N = -(\mathbf{B}_R + \mu\mathbf{I})^{-1}\mathbf{b}_R$ for problem (3.13), $\mathbf{d}_B = -\nabla_B\mathbf{H}(\mathbf{x}^{(k)})^{-1}\nabla_N\mathbf{H}(\mathbf{x}^{(k)})\mathbf{d}_N$, and $\mathbf{d} = [\mathbf{d}_N \quad \mathbf{d}_B]$ thus we have $\liminf_{k\to\infty}\|\mathbf{d}\| = 0$. Therefore, by Theorem 3.2 $\mathbf{x}^{(k)}$ is the KKT point as $k \to \infty$.

4 Case Study

In this chapter, three cases about semiconductor are described. We formulate the cases as the SMOO problems like Equation (1.8) and solve by the methods listed in Table 3.2.

4.1 Geometric Layout Design for Semiconductor Manufacturability

The information about how different geometric styles of layouts impact the circuit performance is important for fables design houses. Some slight changes of the channel length and width often lead to unexpected variations in the electricity signals. The rounding phenomenon will occur in the corners of poly-silicon after photolithography. Generally speaking, the "Active-Area" is the main cause of the variation. Examples with rounding phenomenon are shown in Figure 4.1.

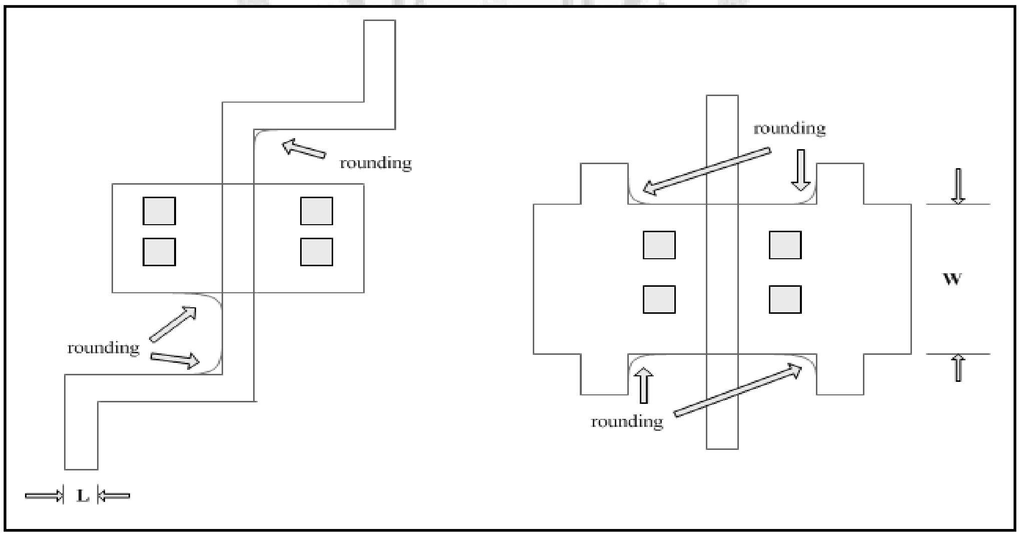


Figure 4.1 Two SPICE models with the rounding phenomenon

The rounding phenomenon would increase the channel drawn length, *Len*, and the channel drawn width, *Wid,* in Figure 4.1. The change of *Len* and *Wid* directly influences the width-to-length ration of a transistor: ${Wid}/{Len}$. However, the major observations in E-Tests, saturation current (I_{Dsat}) and the threshold voltage (V_t), would be proportional to ${Wid}/{Len}$ [21, 16]. Thus, the design house would like to obtain a setting of the design layout which has less variation and close to the desired electrical performances.

In this case study, the design factors on the device layout are shown in Figure 4.2 and the upper bound and lower bound of these factors are summarized as Table 4.1. This design is a NMOS transistor and the rounding phenomenon occurs around the fillister in the center of Active-Area.

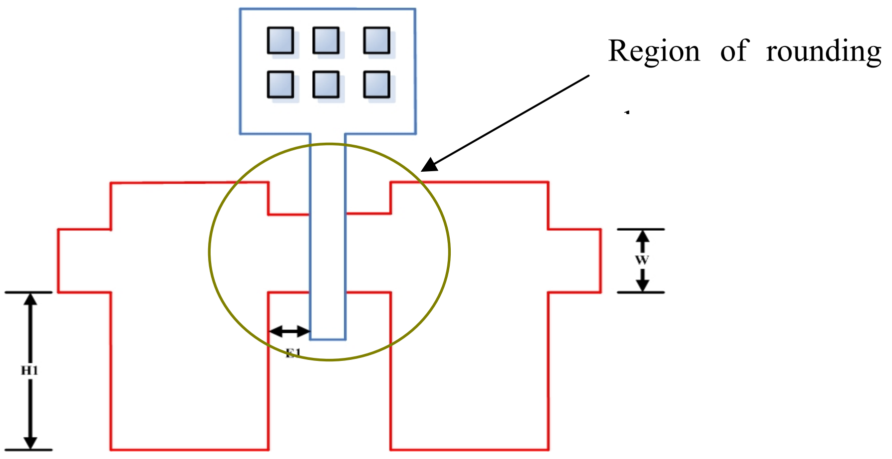


Figure 4.2 Design factors on geometric layout

It is a three-factor layout design problem. Ten ET parameters are measured in a CCD experiment and ten response surface models are built as below:

Table 4.1 Upper bounds and lower bounds for 3 factors

Factor	Factor Name	Lower Bound	Upper Bound
x_1	$H1$	0	0.4
x_2	$E1$	0.05	0.15
x_3	W	0.1	0.3

$$\hat{Y}_{\mathrm{A1}} = 0.48+0.86H1-0.89E1-0.87W-0.1H1^2+7.18E1^2+3.43W^2+1.19H1E1-2.57H1W-0.59E1W$$
$$\hat{Y}_{\mathrm{A2}} = 0.09+0.99H1+2.23E1+1.63W-0.37H1^2-0.19E1^2-0.91W^2-2.3H1E1-1.95H1W-8.53E1W$$
$$\hat{Y}_{\mathrm{A3}} = -0.26-0.48H1+0.36E1+0.03W+0H1^2-2.54E1^2-0.2W^2+1.71H1E1+1.16H1W-1.03E1W$$
$$\hat{Y}_{\mathrm{A4}} = -0.26+0H1-0.48E1-0.13W-0.08H1^2+0.14E1^2-0.13W^2-0.67H1E1-0.1H1W+3.25E1W$$
$$\hat{Y}_{\mathrm{A5}} = 0.51+0.66H1-2.06E1+0.31W-0.36H1^2+9.6E1^2+0.08W^2+0.35H1E1-1.54H1W-0.34E1W$$
$$\hat{Y}_{\mathrm{A6}} = -0.29-0.46H1+0.5E1-0.3W+0.15H1^2-3.12E1^2+0.67W^2+0.8H1E1+0.96H1W+0.19E1W$$
$$\hat{Y}_{\mathrm{A7}} = 0.01+1.48H1+2E1+1.94W-0.4H1^2+2.39E1^2-2.54W^2-5.77H1E1-2.49H1W-5.29E1W$$
$$\hat{Y}_{\mathrm{A8}} = -0.39-0.11H1+0.55E1+0.45W+0.26H1^2-3.46E1^2-0.79W^2+0.4H1E1-0.71H1W+0.33E1W$$
$$\hat{Y}_{\mathrm{B1}} = 0.32-0.17H1+0.59E1+1.47W-0.14H1^2-2.35E1^2-3.02W^2+0.21H1E1+0.68H1W-1.77E1W-0.32$$
$$\hat{Y}_{\mathrm{C1}} = 1098.08+600.73H1-8716.68E1-5754.12W+1716.24H1^2+28624.78E1^2+12017.57W^2$$
$$+3734.83H1E1-5760.43H1W+13181E1W$$

Because the rounding phenomenon would be influenced by $H1$ and $E1$, the design rule would like the $H1$ to be as large as possible and the $E1$ to be as small as possible. Thus, the term, $(H1-0.4)^2+(E1-0.05)^2$, are added into our objective function to ensure the design factors close to the targets. In addition, the designers are asked to minimize the rounding effect caused by the design factors $E1$ and $H1$. That is, they hope that changes in $E1$ and $H1$ should not have minimum influence on the responses. Therefore, additional terms of $\left(\frac{\partial ET_i}{\partial x_q}-0\right)^2$, $q=1,2$, are added in the objective function. Each of the ten ET has a specification window and a desired target. We generate the desired targets close to the responses corresponding to the setting of (0.4, 0.05, 0.25). Furthermore, the specification limits are generated by ±10% of these responses. These requirements are summarized as in Table 4.2 and the details of problem formulation are in Appendix D.

Table 4.2 Desired targets and specification windows for DFM

Response	Response name	Desired target (T_i)	Specification window L_i	U_i
$A1$	IdsatN-592	0.54	0.483975	0.591525
$A2$	IdsatN-593	0.54	0.486743	0.594908
$A3$	IdsatP-592	-0.31	-0.33883	-0.27722
$A4$	IdsatP-593	-0.32	-0.35184	-0.28787
$A5$	IdsatN-104	0.57	0.511785	0.625515
$A6$	IdsatP-104	-0.35	-0.38671	-0.3164
$A7$	IdsatN-107	0.54	0.48573	0.59367
$A8$	IdsatP-107	-0.37	-0.40623	-0.33237
$B1$	VtN	0.48	0.433845	0.530255
$C1$	IoffN	225	202.2143	247.1508

Here, we solve the SMOO problem by the three methods. The optimum design, the corresponding responses, and the effects are summarized in Table 4.3, Table 4.4, and Table 4.5, respectively.

Table 4.3 Optimum design of DFM case

Factor	H1	E1	W
Optimum setting	0.155624	0.1341821	0.1184904

Table 4.4 Responses given the optimum design

A1	A2	A3	A4	A5	A6	A7	A8	B1	C1
IdsatN-592	IdsatN-593	IdsatP-592	IdsatP-593	IdsatN-104	IdsatP-104	IdsatN-107	IdsatP-107	VtN	IoffN
0.5344	0.4917	-0.2907	-0.3052	0.5118	-0.3357	0.4857	-0.3466	0.4478	247.1508

Table 4.5 Sensitivity effects given the optimum design

	A1	A2	A3	A4	A5	A6	A7	A8	B1	C1
H1	0.684	0.335	-0.113	-0.127	0.412	-0.192	0.286	-0.595	-0.105	953.498
E1	1.152	0.81	-0.178	-0.162	0.53	-0.19	1.117	-0.277	-0.218	1108.238
W	-0.536	-0.034	0.025	0.26	0.044	0.034	0.241	0.197	0.623	-2033.993

Here we allocate 2^3 initial solutions for the global search. Seven feasible initial solutions could be found. We suppose the terminal criterion is less than 10^{-6} between two iterative objective values or more than seven hundred steps of search. Table 4.6 describes the results of all local optimums with the seven feasible initial solutions by the methods listed in Table 3.2.

Table 4.6 Results of DFM case (Local Search)

Methods	Average Objective Value	Best Objective Value	Average Number of Iterations	Average Computing Time (seconds)
GRG + Zoutendijk	2.0196656E+07	1.5592227E+07	434.00	1.20
GRR + Zoutendijk (Δ=100, α=10)	1.5592227E+07	1.5592227E+07	116.14	0.63
GRR + Zoutendijk (Δ=100, α=20)	1.5592227E+07	1.5592227E+07	332.14	1.40
GRR + Zoutendijk (Δ=100, α=30)	1.5870851E+07	1.5592227E+07	477.43	2.14
GRT + Zoutendijk with CRA	**1.5592227E+07**	**1.5592227E+07**	**34.71**	**0.23**
GRT + Zoutendijk with DRA	**1.5592227E+07**	**1.5592227E+07**	**34.43**	**0.22**
Lingo (Steepest Edge)	1.5592233E+07	1.5592220E+07	12.00	< 1
Lingo (SLP Directions)	1.5592230E+07	1.5592230E+07	12.86	< 1
Lingo (Steepest + SLP)	1.5592230E+07	1.5592230E+07	12.29	< 1

In the above results, the algorithms with the “GRR + Zoutendijk” and “GRT + Zoutendijk” approach and software “Lingo” have better performance. All initial solutions could reach the global optimum. However, the algorithm with the “GRG + Zoutendijk” approach can’t converge to the global solution no matter what initial solution is used possibly due to the zigzagging phenomenon. Moreover, the steps and the computing time of the algorithm with “GRR + Zoutendijk” are very sensitive to the parameter.

4.2 Robust Configuration of Semiconductor Supply Chain

Semiconductor fabrication is a very complicated manufacturing process. The global, cross-company supply chain operations as shown in Figure 4.3 are even more complicated and dynamic.

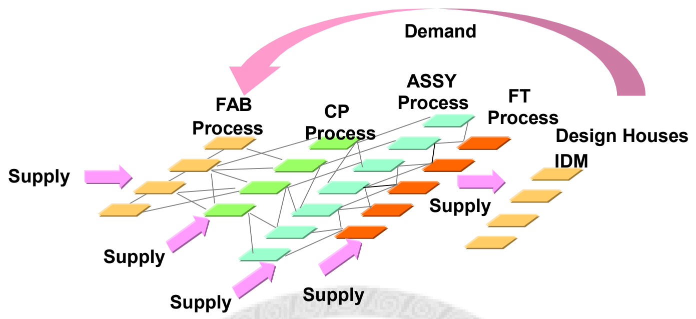


Figure 4.3 Semiconductor supply chain

For the complexity, a usual planning and scheduling solutions have become impossible to employ. Thus, both statistical optimization and control techniques have been proposed and applied to semiconductor manufacturing systems [6]. The empirical supply chain model describes how the supply chain configuration affects the chosen performance metrics and their variability. With such models, an optimal supply chain configuration can be found for different types of products, priorities, and routes.

There are several performance metrics of the semiconductor supply chain. From the entire supply chain point of view, this case chooses “the mean of X-factor” and "the variability (standard deviation) of cycle time" as the metrics to evaluate the supply chain performance in semiconductor manufacturing. In addition, lots of allocation decision variables in semiconductor manufacturing may affect the supply chain performance metrics. In this case, these allocation decision variables are defined as follows: $\pi_{\tilde{k}\tilde{q}}$, the percentage of product $\tilde{k}$ assigned to be produced at the priority $\tilde{q}$, and $\rho_{\tilde{k}\tilde{r}}$, the percentage of product $\tilde{k}$ assigned to be produced at the route $\tilde{r}$. The relationship among these allocation decision variables is shown in Figure 4.4; these metrics are defined as follows: $E(X - factor_{\tilde{q}})$, the mean of X-factor to all products assigned to be produced at the priority $\tilde{q}$, and $SD(CT_{\tilde{q}})$, the standard deviation of cycle time to all products assigned to be produced at the priority $\tilde{q}$. Besides, we assume that the priority mix is independent of the supply chain route mix without loss of generality.

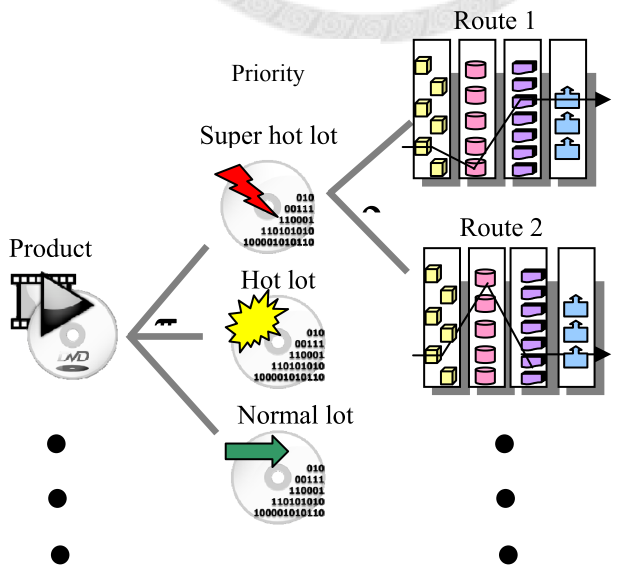


Figure 4.4 Supply chain allocation decision variables

By collecting the data from research papers and personal interviews, this case build an empirical supply chain simulation model as shown in Figure 4.5. The production environmental setting in our simulation is shown in Figure 4.5.

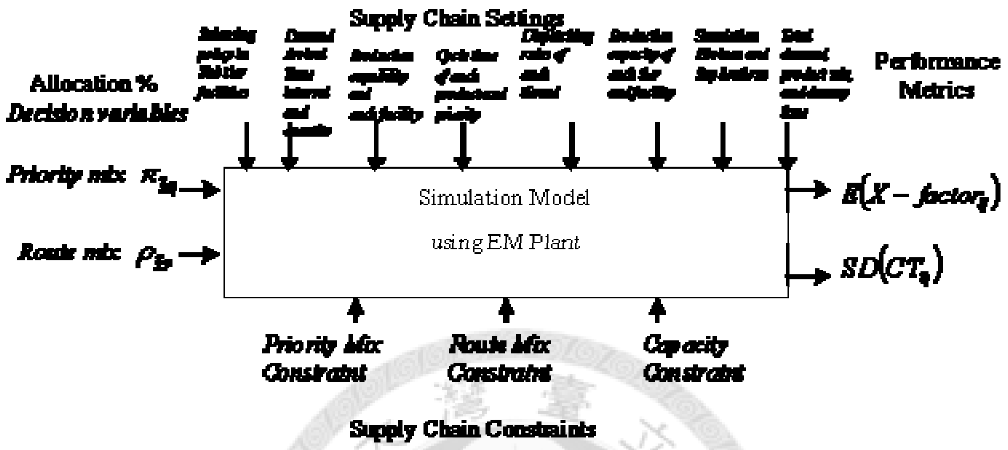


Figure 4.5 Supply chain simulation model

Table 4.7 The Environment setting of model

Index	Value
Number of facilities in each tier	3 tiers; 6 : 2 : 2
Product Capabilities of each facilities	See Table 3.9
Simulation Horizon Setting	90days
Total Demand Quantity	6465K wafers
Production Capacity of each tier (wafer per month)	6465K wafers for each tier
Average bottleneck processing time of each facility of each tier (Capacity Constraint)	Product A：B：C = 2：1.7：1

Table 4.8 The capacity at each facility of each tier

FAB	Capacity	Assem.	Capacity	FT	Capacity
FAB1	1468K	Assem1	3265K	FT1	3200K
FAB2	1376K	Assem2	3200K	FT2	3265K
FAB3	922K				
FAB4	1133K				
FAB5	1202K				
FAB6	689K				
Total	6465K	Total	6465K	Total	6465K

Besides, there are three priorities for each product, and the required delivery durations are very different: the required delivery duration for super hot lot is 1.3 times the row process time of each product; 2.1 times for hot lot and 3 times for the regular. We have total nine possible routes in this example including six routes, four routes and two routes for three different products, respectively.

We also assume that the production cycle time is infinite if capacity utilization rate approaches to 200% and the production cycle time is raw processing time if capacity utilization rate is 0%. By following this assumption, the product cycle time for each product at each plant in different priorities can be estimated based on different utilization rate for each product at each plant in different priorities. The general function of cycle time is an exponential curve. Our simulation model is based on 80% capacity utilization rate. The expected cycle times and raw process time for each product at each plant of different priorities in FAB, Assembly, and Final test are listed in Appendix E.

Moreover, we design 5 levels for each factor, but total levels in this experimental design have only 15 factors because the sum of decision variables $\pi_{\tilde{k}\tilde{q}}$ and $\rho_{\tilde{k}\tilde{r}}$ must add up to 1. Next, a D-Optimal method is adopted such that 180 simulation runs are further developed. Finally, each run is performed 20 replicates. The corresponding performance metrics for each run were collected. After that, a response surface model is generated to indicate the interrelationships between $E(X - factor_{\tilde{q}})$, $SD(CT_{\tilde{q}})$ and $\pi_{\tilde{k}\tilde{q}}$, $\rho_{\tilde{k}\tilde{r}}$. Thus, an optimal configuration model in a semiconductor manufacturing is ready to be developed.

Since this model must consider several performance metrics simultaneously, the subjective weights of performance metrics for priority 1, 2 and 3 are supposed to 15, 5, and 1, respectively:

$$Min \quad \sum_{\tilde{q}=1}^{3} w_{\tilde{q}} \left[E(X - factor_{\tilde{q}}) - 1\right]^2 + \sum_{\tilde{q}=1}^{3} w_{\tilde{q}} \left[SD(CT_{\tilde{q}}) - 0\right]^2$$

where $w_1 = 15, w_2 = 5, w_3 = 1$. The target of the X-factor and the standard deviation are one and zero.

In addition to the target, there are the lower bounds of the X-factor and the standard deviation:

$$E(X - factor_{\tilde{q}}) \geq 1 \qquad \forall \tilde{q} ,$$

$$SD(CT_{\tilde{q}}) \geq 0 \qquad \forall \tilde{q} .$$

Moreover, there are several sets of constraints, which are explained as follows: The proportion of a product assigned to be produced at different priority levels should be added up to 1:

$$\sum_{\tilde{q}} \pi_{\tilde{k}\tilde{q}} = 1 \qquad \forall \tilde{k} .$$

The proportion of a product assigned to be produced at different routes should be added up to 1:

$$\sum_{\tilde{r}} \rho_{\tilde{r}\tilde{k}} = 1 \qquad \forall \tilde{k} .$$

The total proportion of demands to be produced at the priority level $\tilde{q}$ must locate within a predetermined upper limit and lower limit:

$$\xi_{\tilde{q}} \le \sum_{\tilde{k}} \tilde{p}_{\tilde{k}} \cdot \pi_{\tilde{k}\tilde{q}} \le \eta_{\tilde{q}} \qquad \forall \tilde{q} ,$$

where $\eta_{\tilde{q}}$ is the maximum percentage of products produced at the priority $\tilde{q}$, $\xi_{\tilde{q}}$ is the minimum percentage of products produced at the priority $\tilde{q}$, and $\tilde{p}_{\tilde{k}}$ is the percentage of product $\tilde{k}$ in product mix.

The total proportion of demands to be produced at the specific route must locate within a predetermined upper limit and lower limit:

$$\tilde{\beta}_{\tilde{r}} \le \sum_{\tilde{k}} \tilde{p}_{\tilde{k}} \cdot \rho_{\tilde{r}\tilde{k}} \le \tilde{\alpha}_{\tilde{r}} \qquad \forall \tilde{r} ,$$

where $\tilde{\alpha}_{\tilde{r}}$ is the maximum percentage of products produced at production route $\tilde{r}$ and $\tilde{\beta}_{\tilde{r}}$ is the minimum percentage of products produced at production route $\tilde{r}$.

The capacity constraints of each facility in each supply chain tier:

$$E_{\tilde{t}} \sum_{\tilde{r}:\phi \in \tilde{r}} \sum_{\tilde{k}} \tilde{p}_{\tilde{k}} \rho_{\tilde{r}\tilde{k}} \frac{PT_{\tilde{k}\tilde{t}\phi}}{PT_{\tilde{t}\phi}} \le C_{\tilde{t}\phi} \qquad \forall \tilde{t}, \phi,$$

where $E_{\tilde{t}}$ is the utilization rate of tier $\tilde{t}$, $C_{\tilde{t}\phi}$ is the capacity ratio at factory ϕ of tier $\tilde{t}$, $PT_{\tilde{t}\phi}$ is the average production cycle time of single product at factory ϕ of tier $\tilde{t}$, and $PT_{\tilde{k}\tilde{t}\phi}$ is the average production cycle time of product $\tilde{k}$ at factory ϕ of tier $\tilde{t}$.

The upper bounds and the lower bounds of each decision variable:

$$U_{\pi_{\tilde{k}\tilde{q}}} \ge \pi_{\tilde{k}\tilde{q}} \ge L_{\pi_{\tilde{k}\tilde{q}}} \qquad \forall \tilde{k}, \tilde{q},$$

$$U_{\rho_{\tilde{r}\tilde{k}}} \ge \rho_{\tilde{r}\tilde{k}} \ge L_{\rho_{\tilde{r}\tilde{k}}} \qquad \forall \tilde{r}, \tilde{k},$$

where $U_{\pi_{\tilde{k}\tilde{q}}}$ and $L_{\pi_{\tilde{k}\tilde{q}}}$ are the upper bound and the lower bound of the percentage of product $\tilde{k}$ assigned to be produced at the priority $\tilde{q}$. $U_{\rho_{\tilde{r}\tilde{k}}}$ and $L_{\rho_{\tilde{r}\tilde{k}}}$ are the upper bound and the lower bound of the percentage of product $\tilde{k}$ assigned to be produced at the route $\tilde{r}$. The details of the supply chain problem formulation are in Appendix F. Here, we solve the SMOO problem again by the three methods. The optimum solution and the corresponding responses are summarized in and Table 4.10, respectively.

Table 4.9 Optimum design of supply chain case

Factor	π_{11}	π_{12}	π_{13}	π_{21}	π_{22}	π_{23}	π_{31}
Optimum setting	5	10	85	15	10	75	15
Factor	π_{32}	π_{33}	ρ_{11}	ρ_{14}	ρ_{15}	ρ_{17}	ρ_{18}
Optimum setting	10	75	10	10	20	17.1498	19.9088
Factor	ρ_{19}	ρ_{22}	ρ_{24}	ρ_{26}	ρ_{27}	ρ_{33}	ρ_{34}
Optimum setting	22.9414	30	23.3269	15.9212	30.7519	45.4128	54.5871

(Unit of all decision variables: %)

Table 4.10 Responses given the optimum design

$E(X-factor_1)$	$E(X-factor_2)$	$E(X-factor_3)$	$SD(CT_1)$	$SD(CT_2)$	$SD(CT_3)$
1.39722	1.63383	2.10121	0.24881	0.2734	1.58036

(Unit of the standard deviation: Month)

We generate 32 feasible initial points and perform the search algorithm listed in Table 3.2. Table 4.11 shows the search results of each method. Although, the GRT + Zoutendijk and the GRG + Zoutendijk method can find almost the same optimum but the GRG + Zoutendijk consumes less computing time and uses less number of iterations. Moreover, we use the local-search option to perform algorithm thus the objective values of the Lingo's method are worse than other methods.

Table 4.11 Results of supply chain case (Local Search)

Methods	Average Objective Value	Best Objective Value	Average Number of Iterations	Average Computing Time (seconds)
GRG + Zoutendijk	1.1831251E+01	9.379897E+00	107.23	3.68
GRR + Zoutendijk (Δ=100, α=10)	1.2398509E+01	9.379897E+00	442.67	38.72
GRT + Zoutendijk with CRA	**1.2190163E+01**	**9.379897E+00**	**307.81**	**12.10**
GRT + Zoutendijk with DRA	**1.2048563E+01**	**9.379897E+00**	**285.48**	**10.63**
Lingo (Steepest Edge)	1.6123810E+01	1.612381E+01	27.00	< 1
Lingo (SLP Directions)	1.6126714E+01	1.613550E+01	27.54	< 1
Lingo (Steepest + SLP)	1.6123701E+01	1.612673E+01	25.32	< 1

4.3 Track System PEB CDU Optimization

In semiconductor fabrication industry, the Critical Dimension Uniformity (CDU) control is essential for today's high performance IC device. The desired control of the CDU is just under 2.6nm (3-sigma) for 65-nm technology. The across-wafer gate critical dimension uniformity (CDU) strongly affects the final chip-to-chip performance spread in terms of speed and power. Thus it motivates us to improve the CDU for better yield. This study uses the methods including Design of Experiment (DOE), Response Surface Methodology (RSM) and the GRT search to improve the CDU. There is a paper, see also [23], discusses the improvement of the CDU by using different approach can be compared with our result. Many semiconductor fabrication technologies in our study are also can be found in this paper.

Within-wafer CD uniformity is mainly affected by the temperature non-uniformity on the post-exposure-bake (PEB) hot plate. Therefore the temperature control of the PEB step has an important impact of the final CDU. There are a lot of source contributes to CD variation throughout the lithography and etch sequence. Table 4.12 shows the possible source to CD variation [23]. The PEB step has become very critical in controlling gate CD since the thermal dose diffuses acid and catalyzes the chemical reaction of the chemically amplified resist after exposure. This study will focus on reducing the variation source of the PEB step. The simplest and most straightforward approach to reduce across-wafer CD variation is to make each processing step spatially uniform. Modern wafer track systems include a multizone/multicontroller bake plate meant to be adjusted to deliver more spatially uniform PEB temperature distribution. In this study, the distribution of the seven zones is shown in Figure 4.6.

Table 4.12 Source and characteristic of several types of CD variation

Process step	Tool	Type of variation
Coat	Track	across wafer (AW), wafer-to-wafer (w2w)
Bake	Track	AW, w2w
Expose	Scanner	AW, intra die, w2w
Develop	Track	AW, w2w
Etch	Etcher	AW, w2w

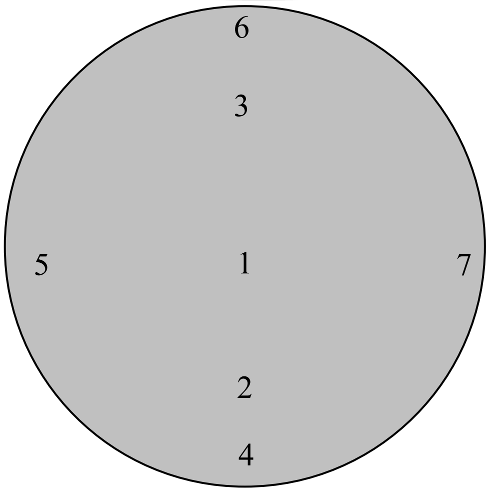


Figure 4.6 The distribution of multizone PEB bake plate

The manipulatable parameter of the PEB bake plate is the temperature of each zone. In our modeling approach, we have the same assumptions as [23]. We also assume that the actual steady-state PEB temperature on a wafer at a location over each zone of the multizone/multicontroller bake plate is decided by the temperature setpoint, the corresponding offset of the zone controller, and the effect of other zones, due to the good conductivity of bake plate.

In practice, there are 577 sites in one wafer and the CDs of these sites are affected by the temperature of the seven zones, i.e., the offsets of the seven zones. For this reason, we can construct 577 "Linear Regression Models" for these 577 sites. Each model can be written as the following equation:

$$CD_i = b_{i,0} + \mathbf{b}_i^T \times \mathbf{Zone\ Offsets} + \varepsilon_i ,$$

where $i = \{1,...,577\}$, $\mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ b_{i,2} \\ \vdots \\ b_{i,7} \end{bmatrix}$, $\mathbf{Zone\,Offses} = \begin{bmatrix} Zone_1\ Offset \\ Zone_2\ Offset \\ \vdots \\ Zone_7\ Offset \end{bmatrix}$ and ε is the error term of the model.

But the only concern is the CDU of the wafer not the CDs thus we use the "Mean Difference" approach to construct these models. The idea of the "Mean Difference" is to construct a model describes the relationship between the differences (the CD of each site − the overall mean of the wafer) and offset of each zone. Because our ultimate goal is to reduce the CDU of the wafer thus we will care about the difference of the CD of each site to the overall mean. If the total differences become small, that is, the CDU of the wafer becomes small simultaneously. Therefore, the set of optimal offset found by the "Mean Difference" model can minimizes the difference of each site to overall mean and also reduces the CDU of the wafer. For this reason, we choose the "Mean Difference" approach to construct the models. Thus the "Mean Difference" models can be rewritten as follows.

$$(CD_i - M)_{i,D} = b_{i,0,D} + \mathbf{b}_{i,D}^T \times \mathbf{Zone\,Offsets} + \varepsilon_i ,$$

where $i = \{1,...,577\}$ and M is an overall mean CD value of the wafer.

In this study, the data is obtained from 32 experiments. Actually, with the help of "Design of Experiment" (DOE), we also can build the models by fewer experiments. But we do not emphasize the importance of using the DOE in this study. The effects of all sites on a wafer can be plot on the color grid charts. The different colors of the color grid chart denote the degrees of the effects. The darker color means the stronger effect of the site. Figure 4.7 shows the effect maps of the seven zones.

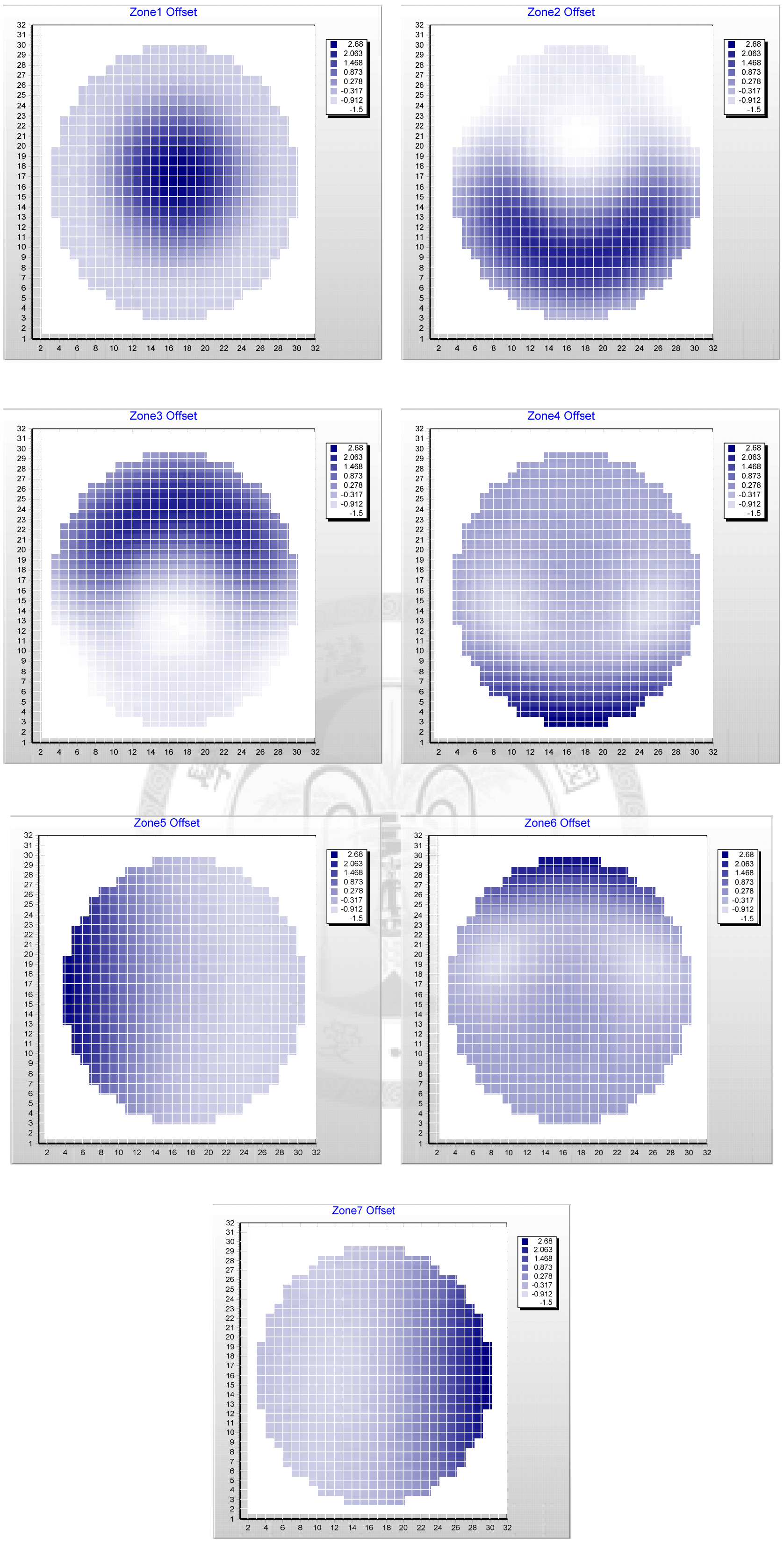


Figure 4.7 The effect map of the seven zones.

In practice, there are three different requirements need to be meet. The first requirement is to find the set of optimal offset minimizes the CDU. The second requirement is to find the set of optimal offset minimizes the CDU and the value of overall mean CD of the wafer hit the specified target. The third requirement is to find the set of optimal offset minimizes the CDU and the value of overall mean CD the wafer satisfies the specified bounded constraints. To satisfy the second and third requirements of the optimization, we also need to evaluate the model describes the overall mean of the wafer and the offsets. The overall mean model can be written as follows.

$$\mathit{Overall\ Mean} = b_{0,M} + \mathbf{b}_M^T \times \textbf{Zone Offsets} + \varepsilon_M,$$

where $\mathbf{b}_M = \begin{bmatrix} b_{1,M} \\ b_{2,M} \\ \vdots \\ b_{7,M} \end{bmatrix}$, $\textbf{Zone Offses} = \begin{bmatrix} Zone_1\ Offset \\ Zone_2\ Offset \\ \vdots \\ Zone_7\ Offset \end{bmatrix}$ and ε_M is the error term of the overall mean model.

Sometimes, the offsets of the PEB plate have physical limitation, i.e., the optimization model needs to consider the constraints for the offsets. We now summarize the three optimization models as follows.

Optimization Model for Requirement (1):

$$\underset{Zone_i\ Offset}{Minimize}: \sum_{i=1}^{577}\left[b_{i,D,0} + \mathbf{b}_{i,D}^T \times \mathbf{Zone\ Offsets}\right]^2 ;$$

$$Subject\ to: L_j \le Zone_j\ Offset \le U_j ,$$

where $j = \{1,...,7\}$, L_j and U_j denote the lower-bound and upper-bound of the offset of Zone *j*, $\mathbf{b}_{i,D} = \begin{bmatrix} b_{i,D,1} \\ b_{i,D,2} \\ \vdots \\ b_{i,D,7} \end{bmatrix}$, $\mathbf{Zone\ Offses} = \begin{bmatrix} Zone_1\ Offset \\ Zone_2\ Offset \\ \vdots \\ Zone_7\ Offset \end{bmatrix}$.

Optimization Model for Requirement (2):

$$\underset{Zone_i\ Offset}{Minimize}: \sum_{i=1}^{577}\left[b_{i,D,0} + \mathbf{b}_{i,D}^T \times \mathbf{Zone\ Offsets}\right]^2 ;$$

$$\begin{aligned} Subject\ to: & b_{0,M} + \mathbf{b}_M^T \times \mathbf{Zone\ Offsets} = T_M ; \\ & L_j \le Zone_j\ Offset \le U_j , \end{aligned}$$

where $j = \{1,...,7\}$, T_M is the target of the overall mean and L_j and U_j denote the lower-bound and upper-bound of the offset of Zone *j* respectively.

Optimization Model for Requirement (3):

$$\underset{Zone_i\ Offset}{Minimize}: \sum_{i=1}^{577}\left[b_{i,D,0} + \mathbf{b}_{i,D}^T \times \mathbf{Zone\ Offsets}\right]^2 ;$$

$$\begin{aligned} Subject\ to: & L_M \le b_{0,M} + \mathbf{b}_M^T \times \mathbf{Zone\ Offsets} \le U_M ; \\ & L_j \le Zone_j\ Offset \le U_j \end{aligned}$$

, where $j = \{1,...,7\}$, U_M and L_M are the upper-bound and lower-bound of the overall mean and L_j and U_j denote the lower-bound and upper-bound of the offset of Zone j respectively.

We consider the third requirement to be the optimization model. We allocate 2^3 initial solutions for the global search. We use the same terminal criterion, 10^{-6} between two iterative objective values or seven hundred steps as other cases. Table 4.13 lists the results of all local optimums starting with the eight feasible initial solutions by the methods in Table 3.2.

Table 4.13 Results of CDU optimization case (Local Search)

Methods	Average Objective Value	Best Objective Value	Average Number of Iterations	Average Computing Time (seconds)
GRG + Zoutendijk	3.0783512E+01	3.0109657E+01	700	70.48
GRR + Zoutendijk (Δ=100, α=10)	3.0092226E+01	3.0092226E+01	12.375	0.73
GRR + Zoutendijk (Δ=100, α=20)	3.0092226E+01	3.0092226E+01	63.125	3.13
GRR + Zoutendijk (Δ=100, α=30)	3.0092226E+01	3.0092226E+01	154.75	6.95
GRT + Zoutendijk with CRA	**3.0092226E+01**	**3.0092226E+01**	**5.5**	**0.45**
GRT + Zoutendijk with DRA	**3.0092226E+01**	**3.0092226E+01**	**2.625**	**0.37**
Lingo (Steepest Edge)	3.0092230E+01	3.0092230E+01	39.75	≈ 1
Lingo (SLP Directions)	3.0092230E+01	3.0092230E+01	44	≈ 1
Lingo (Steepest + SLP)	3.0092230E+01	3.0092230E+01	38	≈ 1

Due to the zigzagging phenomena, the method “GRG + Zoutendijk” methods also cannot converge to the global minimum again. The search result including the steps and the computing time of the “GRR + Zoutendijk” methods is sensitive to the parameter again. If we compare the objective value, the consuming steps and the computing time, the “GRT + Zoutendijk” methods are all comparable to the methods of “Lingo”.

5 Conclusions

In this research, we solve the “Trust Region Subproblem (TRS)”, using SVD. With the help of SVD, we enhance the TRS algorithm by proposing a better lower bound for safeguarding the Newton’s iterates and also provide a new mechanism to adjust the trust-region radius dynamically. To solve the SMOO problem, a nonlinear constrained problem, we then develop the “Generalized Reduced Trust Region (GRT)” search method with the above modifications. We have also proved the convergence of the proposed GRT algorithm.

To verify our algorithm, a test problem and three SMOO problems were studied. The following results were observed:

1. The GRT search method avoids the zigzagging phenomena often incurred by the GRG method and gets a better solution.
2. The GRT search combined with the Zoutendijk search method can effectively reach the optimal point in every case.
3. The GRT search method with dynamic radius adjustment can reduce the number of iterations and computing time by about 5% to 10% as compared to the conventional radius adjustment in a large scale problem such as the cases of the DFM problem and the robust semiconductor supply chain optimizations.
4. Compared against Lingo’s solution, our search algorithm usually converges at the same or better solution with comparable computation time.

Although, the four cases lend support to this research, there are still much room to be improved.

1. In order to deal with any kind of optimization problems, the Hessian matrix can be calculated and updated more efficiently. Moreover, the Hessian matrix indeed could be approximated for shorter computing time [4].
2. In late 1980s, many researchers try to solve the trust region problem more efficiently like the dogleg method and indefinite dogleg method [5, 14, 22]. They are all approximate techniques of the trust region problem and also lead to the same global and local convergence properties, i.e., these methods can shorten the computing time without loss of optimality conditions.
3. In this research, the SVD replaces the Cholesky factorization to compute and perform the Newton's iterates. However the SVD is too costly for large matrices, the method is applicable only for small problems. There have been many researches on how to reduce the computational efforts of the Cholesky factorization [10].
4. Although this research propose the convergence property of the GRT algorithm but the Corollary 3.1 does not cover the Line Search method. There have been some algorithms combine the Trust Region method and Line Search method and also provider convergence properties [8, 15, 20]
5. In this research, we propose a dynamic strategy to update the trust-region radius. In 2005, some researchers discussed about the trust region radius update [19].
6. Zoutendijk's method sometimes incurs the zigzagging phenomenon. It may influence the search performance of the GRT search. There should be some

enhancements when searching a feasibly improving direction at the boundary of feasible set.

7. Multiple initial solutions could increase the probability to reach the global optimum, but there exists a systematic method. In Lingo's algorithm, the "Branch and Bound" algorithm is adopted. It divides the nonlinear programming problem into several approximate convex optimization problems, then, searches the global optimum iteratively.

REFERENCE

[1] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Dover Publications, 2003.

[2] M. S. Bazaraa, H. D. Sherali, C. M. Shetty and Wiley InterScience (Online service), *Nonlinear programming [electronic resource] : theory and algorithms*, Wiley-Interscience, Hoboken, N.J., 2006.

[3] A. D. Belegundu and T. R. Chandrupatla, *Optimization concepts and applications in engineering*, Prentice Hall.

[4] R. H. Byrd, H. F. Khalfan and R. B. Schnabel, *Analysis of a Symmetric Rank-One Trust Region Method*, SIAM, 1996, pp. 1025.

[5] R. H. Byrd, R. B. Schnabel and G. A. Shultz, *Approximate Solution of the Trust Region Problem by Minimization over Two-Dimensional Subspaces*, Mathematical Programming, 40 (1988), pp. 247-263.

[6] A. Chen, P. S. Guo and P. Lin, *Statistical analysis and design of semiconductor manufacturingsystems*, 2000, pp. 335-338.

[7] N. R. Draper, *Ridge analysis of response surfaces*, JSTOR, 1963, pp. 469-479.

[8] J. Y. Fan, W. B. Ai and Q. Y. Zhang, *A line search and trust region algorithm with trust region radius converging to zero*, Journal of Computational Mathematics, 22 (2004), pp. 865-872.

[9] S. K. S. Fan, *A different view of ridge analysis from numerical optimization*, Engineering Optimization, 35 (2003), pp. 627-647.

[10] S. K. S. Fan, *THE HOUSEHOLDER TRIDIAGONALIZATION STRATEGY FOR SOLVING A CONSTRAINED QUADRATIC MINIMIZATION PROBLEM*, Taylor & Francis, 2001, pp. 261-277.

[11] D. G. Luenberger, *Linear and Nonlinear Programming*, Springer, 2003.

[12] D. C. Montgomery and D. C. Montgomery, *Design and analysis of experiments*, Wiley New York, 1991.

[13] J. J. More and D. C. Sorensen, *Computing a Trust Region Step*, Siam Journal on Scientific and Statistical Computing, 4 (1983), pp. 553-572.

[14] J. Nocedal, S. J. Wright and SpringerLink (Online service), *Numerical Optimization [electronic resource]*, Springer Science+Business Media LLC., New York, NY, 2006.

[15] J. Nocedal and Y. Yuan, *Combining trust region and line search techniques*, Berlin: Kluwer, 1998, pp. 175.

[16] J. M. Rabaey, A. Chandrakasan and B. Nikolic, *Digital integrated circuits*, Prentice Hall Upper Saddle River, NJ, 2002.

[17] M. Rojas and D. C. Sorensen, *A Trust-Region Approach to the Regularization of Large-Scale Discrete Forms of Ill-Posed Problems*, 2002, pp. 1843-1861.

[18] J. Semple, *Optimality conditions and solution procedures for nondegenerate dual-response systems*, Springer, 1997, pp. 743-752.

[19] J. M. B. Walmag and E. J. M. Delhez, *A Note on Trust-Region Radius Update*, SIAM, 2005, pp. 548.

[20] R. A. Waltz, J. L. Morales, J. Nocedal and D. Orban, *An interior algorithm for nonlinear optimization that combines line search and trust region steps*, Mathematical Programming, 107 (2006), pp. 391-408.

[21] W. Wolf, *Modern VLSI Design: Systems on Silicon, 2nd Editon Prentice Hall*, Inc, 1996.

[22] J. Zhang and C. Xu, *A Class of Indefinite Dogleg Path Methods for Unconstrained Minimization*, SIAM, 1999, pp. 646.

[23] Q. Zhang, K. Poolla and C. J. Spanos, *Across Wafer Critical Dimension Uniformity Enhancement Through Lithography and Etch Process Sequence: Concept, Approach, Modeling, and Experiment*, 2007, pp. 488-505.

[24] 陳彥良, *使用一般化縮減脊線搜尋與 Zoutendijk 方法於多目標統計模型最佳化*, *工業工程學研究所*, 臺灣大學, pp. 60.

Appendix A. Proof of the solution to the Hard Case

To prove **d** satisfies the condition (2.7) [9], observe

$$(\mathbf{G}-\lambda_1\mathbf{I})\mathbf{d} = -(\mathbf{G}-\lambda_1\mathbf{I})\left((\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}+\tau\mathbf{q}_1\right) = -(\mathbf{G}-\lambda_1\mathbf{I})(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}-\tau(\mathbf{G}-\lambda_1\mathbf{I})\mathbf{q}_1 .$$

, where $(\mathbf{G}-\lambda_1\mathbf{I})(\mathbf{G}-\lambda_1\mathbf{I})^{+}=\mathbf{I}$ thus we have

$$(\mathbf{G}-\lambda_1\mathbf{I})\mathbf{d} = -\boldsymbol{\beta}-\tau(\mathbf{G}-\lambda_1\mathbf{I})\mathbf{q}_1$$

and since $\tau\mathbf{q}\in N(\mathbf{G}-\lambda_1\mathbf{I})$ we conclude

$$(\mathbf{G}-\lambda_1\mathbf{I})\mathbf{d} = -\boldsymbol{\beta}$$

, which complete this proof.

For the condition (2.8) we have the squared Euclidean distance of **d** is decomposed as follows

$$\|\mathbf{d}\|^2 = \left\|-(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}+\tau\mathbf{q}_1\right\|^2 = \left\|-(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}\right\|^2 + 2\times\left\|\mathbf{q}_1{}^{\mathrm{T}}(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}\right\| + \|\tau\mathbf{q}_1\|^2$$

, where $\mathbf{q}_1^T(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}=0$.

So we have

$$\|\mathbf{d}\|^2 = \left\|-(\mathbf{G}-\lambda_1\mathbf{I})^{+}\boldsymbol{\beta}\right\|^2 + \|\tau\mathbf{q}\|^2$$

and then $\tau = \pm[(\Delta^2-\phi^2(\lambda_1)]^{\frac{1}{2}}$ can be determined to meet $\|\mathbf{d}\|=\Delta$.

For the condition (2.9), it can be seen that **d** is a KKT point that satisfies KKT first- and second-order conditions for establishing only local optimality.

Appendix B. Trust Region Algorithm

```
Algorithm B.1 [9]
Input:
$\mu_0 = \|\mathbf{G}\|_F$ (where $\|\bullet\|_F$ is the Frobenius matrix norm) to ensure that $(\mathbf{G}+\mu_0\mathbf{I})$
    i s P.D.
$\delta_1$ = tolerance for convergence of the solution $\mathbf{d}(\mu)$
$\delta_2$ = tolerance for convergence of $\mu_k$ to signal the hard case
$\varepsilon$ = tolerance used in the method of iteration
$\mu_{\min}$ = some large negative number (in our implementation, we use the minimum
    value of double)
$k = 0$ (reset the iteration index)
Begin
    Repeat while $\left|\|\mathbf{d}(\mu)\| - \Delta\right| > \delta_1$
        Factor $(\mathbf{G}+\mu\mathbf{I}) = \mathbf{U}^T\mathbf{U}$ (Cholesky Factorization)            (B. 1)
        If $(\mathbf{G}+\mu\mathbf{I})$ is P.D. then
            Solve the two linear system:
            $\mathbf{U}^T\mathbf{U}\mathbf{d}(\mu) = -\boldsymbol{\beta}$ and $\mathbf{U}^T\mathbf{U}\mathbf{y}(\mu) = \mathbf{d}(\mu)$            (B. 2)
            $\mu_{\min} \leftarrow \left[\mu_{\min}, \mu - \frac{\mathbf{d}^T(\mu)\mathbf{d}(\mu)}{\mathbf{d}^T(\mu)\mathbf{y}(\mu)}\right]$            (B. 3)
            If $\|\mathbf{d}(\mu)\| < \Delta$ (at the right of the root) then
                $\mu_{\max} \leftarrow \mu_k$            (B. 4)
            Else
```

```
                $\mu_{\min} \leftarrow \mu_k$                                                    (B. 5)
            End If
            $\tilde{\mu} \leftarrow \mu - \frac{1-\|\mathbf{d}(\mu)\|}{\Delta} \cdot \frac{\|\mathbf{d}(\mu)\|^2}{\|\mathbf{d}(\mu)\|\|\mathbf{y}(\mu)\|}$ (Newton's iterate)          (B. 6)
            If $\tilde{\mu} < \mu_{\min}$ then
                $\tilde{\mu} \leftarrow \frac{(\mu_{\max} + \mu_{\min})}{2}$ (safeguarding)                     (B. 7)
            End If
        Else
            $\mu_{\min} \leftarrow \max\{\mu_{\min}, \mu\}$ and $\tilde{\mu} \leftarrow \frac{(\mu_{\max} + \mu_{\min})}{2}$ (safeguarding)
                                                                                 (B. 8)
        End If
        If $|\mu_{\max} - \mu_{miin}| < \delta_2$ then
            Compute the eigenvector q via the method of inversed iteration
            applied to $(\mathbf{G} + (\mu + \varepsilon)\mathbf{I})$ and then determine $\pi$ (Problem is hard
            case). Return (Solutions are $\mathbf{d}^* = \mathbf{d}(\mu) + \pi\mathbf{q}$; $\mu^* = \mu \approx -\lambda_1$)
                                                                                 (B. 9)
        End If
    End Repeat
End
```

Appendix C. Proof of Theorem 3.1 (Convergence to Stationary Point)

By performing some technical manipulation with the ratio $\rho^{(k)}$ from Algorithm (3.1), we obtain

$$\begin{aligned}\left|\rho^{(k)}-1\right| &= \left|\frac{(f(\mathbf{x}^{(k)})-f(\mathbf{x}^{(k)}+\mathbf{d}))-(m(0)-m(\mathbf{d}))}{m(0)-m(\mathbf{d})}\right| \\ &= \left|\frac{m(\mathbf{d})-f(\mathbf{x}^{(k)}+\mathbf{d})}{m(0)-m(\mathbf{d})}\right|,\end{aligned} \tag{C. 1}$$

where $f\left(\mathbf{x}^{(k)}\right)=m(\mathbf{d})$.

Since from Taylor's theorem we have that

$$f(\mathbf{x}^{(k)}+\mathbf{d})=f(\mathbf{x}^{(k)})+\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}+\int_0^1[\nabla f(\mathbf{x}^{(k)}+t\mathbf{d})-\nabla f(\mathbf{x}^{(k)})]^T\mathbf{d}dt, \tag{C. 2}$$

for some t ∈ (0, 1), it follows from the definition (3.7) of *m* that

$$\begin{aligned}\left|(m(\mathbf{d})-f(\mathbf{x}^{(k)}+\mathbf{d})\right| &= \left|\frac{1}{2}\mathbf{d}^T\mathbf{G}^{(k)}\mathbf{d}-\int_0^1[\nabla f(\mathbf{x}^{(k)}+t\mathbf{d})-\nabla f(\mathbf{x}^{(k)})]^T\mathbf{d}dt\right| \\ &\le \left(\frac{\chi}{2}\right)\|\mathbf{d}\|^2+\chi_1\|\mathbf{d}\|^2,\end{aligned} \tag{C. 3}$$

where we have used χ_1 to denote the Lipschitz constant for $\nabla f\left(\mathbf{x}^{(k)}\right)$ on the set S(R_0), and assumed that $\|\mathbf{d}\|\le R_0$ to ensure that $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k)}$ + t**d** both lie in the set S(R_0). Suppose for contradiction that there is ε> 0 and a positive index K such that

$$\left\|\boldsymbol{\beta}^{(k)}\right\|\ge\varepsilon, \text{ for all } k\ge K. \tag{C. 4}$$

From (3.20), we have for $k \ge$ K that

$$m(0)-m(\mathbf{d})\ge c_1\left\|\boldsymbol{\beta}^{(k)}\right\|\min\left(\Delta^{(k)},\frac{\left\|\boldsymbol{\beta}^{(k)}\right\|}{\left\|\mathbf{G}^{(k)}\right\|}\right)\ge c_1\varepsilon\min\left(\Delta^{(k)},\frac{\varepsilon}{\chi}\right). \tag{C. 5}$$

Using (C. 3), (C. 5), and the bound (3.22), we have

$$\left|\rho^{(k)}-1\right| \le \frac{\gamma^2 \Delta^{(k)2}\left(\frac{\chi}{2}+\chi_1\right)}{c_1 \varepsilon \min\left(\Delta^{(k)}, \frac{\varepsilon}{\chi}\right)}. \tag{C. 6}$$

We now derive a bound on the right-hand-side that holds for all sufficiently small values of $\Delta^{(k)}$, that is, for all $\Delta^{(k)} \le \overline{\Delta}$, where $\overline{\Delta}$ is defined as follows:

$$\overline{\Delta} = \min\left(\frac{1}{2}\frac{c_1\varepsilon}{\gamma^2(\chi/2)+\chi_1}, \frac{R_0}{\gamma}\right). \tag{C. 7}$$

The R_0/γ term in this definition ensures that the bound (C.3) is valid (because $\|\mathbf{d}\| \le \gamma\Delta_k \le \gamma\overline{\Delta} \le R_0$). Note that since $c_1 \le 1$ and $\gamma \ge 1$, we have $\overline{\Delta} \le \varepsilon/\chi$. The latter condition implies that for all $\Delta^{(k)} \in \left[0, \overline{\Delta}\right]$, we have $\min\left(\Delta^{(k)}, \varepsilon/\chi\right) = \Delta^{(k)}$, so from (A. 6) and (3.31), we have

$$\left|\rho^{(k)}-1\right| \le \frac{\gamma^2 \Delta^{(k)2}\left(\frac{\chi}{2}+\chi_1\right)}{c_1 \varepsilon \Delta^{(k)}} = \frac{\gamma^2 \Delta^{(k)}\left(\frac{\chi}{2}+\chi_1\right)}{c_1 \varepsilon} \le \frac{\gamma^2 \overline{\Delta}\left(\frac{\chi}{2}+\chi_1\right)}{c_1 \varepsilon} \le \frac{1}{2}. \tag{C. 8}$$

Therefore, $\rho^{(k)} > \frac{1}{4}$, and so by the workings of Algorithm 3.1, we have $\Delta^{(k+1)} \ge \Delta^{(k)}$ whenever $\Delta^{(k)}$ falls below the threshold $\overline{\Delta}$. It follows that reduction of $\overline{\Delta}$ (by a factor of $\frac{1}{4}$) can occur in our algorithm only if

$\Delta^{(k)} \ge \overline{\Delta}$,

and therefore we conclude that

$$\Delta^{(k)} \ge \min\left(\Delta^{(K)}, \overline{\Delta}/4\right) \text{ for all } k \ge K. \tag{C. 9}$$

Suppose now that there is an infinite subsequence κ such that $\rho^{(k)} \ge \frac{1}{4}$ for $k \in \kappa$. For $k \in \kappa$ and $k \ge \kappa$, we have from (C. 5) that

$$
\begin{aligned}
f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k)} + \mathbf{d}^{(k)}) \\
&\geq \frac{1}{4}\left[m(0) - m(\mathbf{d})\right] \\
&\geq \frac{1}{4} c_1 \varepsilon \min\left(\Delta_k, \varepsilon/\chi\right).
\end{aligned}
\tag{C. 10}
$$

Since f is bounded below, it follows from this inequality that

$$
\lim_{k \in \kappa, k \to \infty} \Delta^{(k)} = 0\,, \tag{C. 11}
$$

contradicting (C. 9). Hence no such infinite subsequence κ can exist, and we must have $\rho^{(k)} < \frac{1}{4}$ for all k sufficiently large. In this case, $\Delta^{(k)}$ will eventually be multiplied by $\frac{1}{4}$ at every iteration, and we have $\lim_{k \to \infty} \Delta^{(k)} = 0$, which again contradicts (C. 9). Hence, our original assertion (C. 4) must be false, giving (3.23).

Complete the proof of Theorem 3.1.

Appendix D. Problem Formulation of DFM Case

Minimize:

$$
\begin{aligned}
&(0.48 + 0.86x_1 - 0.89x_2 - 0.87x_3 - 0.1x_1^2 + 7.18x_2^2 + 3.43x_3^2 + 1.19x_1x_2 - 2.57x_1x_3 - 0.59x_2x_3 - 0.54)^2 \\
&+ (0.09 + 0.99x_1 + 2.23x_2 + 1.63x_3 - 0.37x_1^2 - 0.19x_2^2 - 0.91x_3^2 - 2.3x_1x_2 - 1.95x_1x_3 - 8.53x_2x_3 - 0.54)^2 \\
&+ (-0.26 - 0.48x_1 + 0.36x_2 + 0.03x_3 + 0x_1^2 - 2.54x_2^2 - 0.2x_3^2 + 1.71x_1x_2 + 1.16x_1x_3 - 1.03x_2x_3 + 0.31)^2 \\
&+ (-0.26 + 0x_1 - 0.48x_2 - 0.13x_3 - 0.08x_1^2 + 0.14x_2^2 - 0.13x_3^2 - 0.67x_1x_2 + 0.1x_1x_3 + 3.25x_2x_3 + 0.32)^2 \\
&+ (0.51 + 0.66x_1 - 2.06x_2 + 0.31x_3 - 0.36x_1^2 + 9.6x_2^2 + 0.08x_3^2 + 0.35x_1x_2 - 1.54x_1x_3 - 0.34x_2x_3 - 0.57)^2 \\
&+ (-0.29 - 0.46x_1 + 0.5x_2 - 0.3x_3 + 0.15x_1^2 - 3.12x_2^2 + 0.67x_3^2 + 0.8x_1x_2 + 0.96x_1x_3 + 0.19x_2x_3 + 0.35)^2 \\
&+ (0.01 + 1.48x_1 + 2x_2 + 1.94x_3 - 0.4x_1^2 + 2.39x_2^2 - 2.54x_3^2 - 5.77x_1x_2 - 2.49x_1x_3 - 5.29x_2x_3 - 0.54)^2 \\
&+ (-0.39 - 0.11x_1 + 0.55x_2 + 0.45x_3 + 0.26x_1^2 - 3.46x_2^2 - 0.79x_3^2 + 0.4x_1x_2 - 0.71x_1x_3 + 0.33x_2x_3 + 0.37)^2 \\
&+ (0.32 - 0.17x_1 + 0.59x_2 + 1.47x_3 - 0.14x_1^2 - 2.35x_2^2 - 3.02x_3^2 + 0.21x_1x_2 + 0.68x_1x_3 - 1.77x_2x_3 - 0.48)^2 \\
&+ (1098.08 + 600.73x_1 - 8716.68x_2 - 5754.12x_3 + 1716.24x_1^2 + 28624.78x_2^2 + 12017.57x_3^2 + 3734.83x_1x_2 \\
&- 5760.43x_1x_3 + 13181x_2x_3 - 225)^2 \\
&+ (x_1 - 0.4)^2 + (x_2 - 0.05)^2 \\
&+ (0.86 - 0.2x_1 + 1.19x_2 - 0.59x_3 - 0)^2 + (0.99 - 0.74x_1 - 2.3x_2 - 8.53x_3 - 0)^2 \\
&+ (-0.48 + 0x_1 + 1.71x_2 - 1.03x_3 - 0)^2 + (0 - 0.16x_1 - 0.67x_2 - 0.1x_3 - 0)^2 \\
&+ (0.66 - 0.72x_1 + 0.35x_2 - 0.34x_3 - 0)^2 + (-0.46 + 0.3x_1 + 0.8x_2 + 0.19x_3 - 0)^2 \\
&+ (1.48 - 0.8x_1 - 5.77x_2 - 5.29x_3 - 0)^2 + (-0.11 + 0.52x_1 + 0.4x_2 + 0.33x_3 - 0)^2 \\
&+ (-0.17 - 0.28x_1 + 0.21x_2 - 1.77x_3 - 0)^2 + (600.73 + 3432.48x_1 + 3734.83x_2 + 13181x_3 - 0)^2 \\
&+ (-0.89 + 1.19x_1 + 14.36x_2 - 0.59x_3 - 0)^2 + (2.23 - 2.3x_1 - 0.38x_2 - 8.53x_3 - 0)^2 \\
&+ (0.36 + 1.71x_1 - 5.08x_2 - 1.03x_3 - 0)^2 + (-0.48 - 0.67x_1 + 0.28x_2 + 3.25x_3 - 0)^2 \\
&+ (-2.06 + 0.35x_1 + 19.2x_2 - 0.34x_3 - 0)^2 + (0.5 + 0.8x_1 - 6.24x_2 + 0.19x_3 - 0)^2 \\
&+ (2 - 5.77x_1 + 4.78x_2 - 5.29x_3 - 0)^2 + (0.55 + 0.4x_1 - 6.92x_2 + 0.33x_3 - 0)^2 \\
&+ (0.59 + 0.21x_1 - 4.7x_2 - 1.77x_3 - 0)^2 + (-8716.68 + 3734.83x_1 + 57249.56x_2 + 13181x_3 - 0)^2 \\
&+ (-0.87 - 2.57x_1 - 0.59x_2 + 6.86x_3 - 0)^2 + (1.63 - 1.95x_1 - 8.53x_2 - 1.82x_3 - 0)^2 \\
&+ (0.03 + 1.16x_1 - 1.03x_2 - 0.4x_3 - 0)^2 + (-0.13 + 0.1x_1 + 3.25x_2 - 0.26x_3 - 0)^2 \\
&+ (0.31 - 1.54x_1 - 0.34x_2 + 0.16x_3 - 0)^2 + (-0.3 + 0.96x_1 + 0.19x_2 + 1.34x_3 - 0)^2 \\
&+ (1.94 - 2.49x_1 - 5.29x_2 - 5.08x_3 - 0)^2 + (0.45 - 0.71x_1 + 0.33x_2 - 1.58x_3 - 0)^2 \\
&+ (1.47 + 0.68x_1 1.77x_2 - 6.04x_3 - 0)^2 + (-5754.12 - 5760.43x_1 + 13181x_2 + 24035.14x_3 - 0)^2
\end{aligned}
$$

subject to:

$$
\begin{aligned}
&0.483975 \le 0.48 + 0.86x_1 - 0.89x_2 - 0.87x_3 - 0.1x_1^2 + 7.18x_2^2 + 3.43x_3^2 + 1.19x_1x_2 - 2.57x_1x_3 - 0.59x_2x_3 - 0.54 \le 0.591525 \\
&0.486743 \le 0.09 + 0.99x_1 + 2.23x_2 + 1.63x_3 - 0.37x_1^2 - 0.19x_2^2 - 0.91x_3^2 - 2.3x_1x_2 - 1.95x_1x_3 - 8.53x_2x_3 - 0.54 \le 0.594908 \\
&-0.33883 \le -0.26 - 0.48x_1 + 0.36x_2 + 0.03x_3 + 0x_1^2 - 2.54x_2^2 - 0.2x_3^2 + 1.71x_1x_2 + 1.16x_1x_3 - 1.03x_2x_3 + 0.31 \le -0.27722 \\
&-0.35184 \le -0.26 + 0x_1 - 0.48x_2 - 0.13x_3 - 0.08x_1^2 + 0.14x_2^2 - 0.13x_3^2 - 0.67x_1x_2 + 0.1x_1x_3 + 3.25x_2x_3 + 0.32 \le -0.28787 \\
&0.511785 \le 0.51 + 0.66x_1 - 2.06x_2 + 0.31x_3 - 0.36x_1^2 + 9.6x_2^2 + 0.08x_3^2 + 0.35x_1x_2 - 1.54x_1x_3 - 0.34x_2x_3 - 0.57 \le 0.625515 \\
&-0.38671 \le -0.29 - 0.46x_1 + 0.5x_2 - 0.3x_3 + 0.15x_1^2 - 3.12x_2^2 + 0.67x_3^2 + 0.8x_1x_2 + 0.96x_1x_3 + 0.19x_2x_3 + 0.35 \le -0.3164 \\
&0.48573 \le 0.01 + 1.48x_1 + 2x_2 + 1.94x_3 - 0.4x_1^2 + 2.39x_2^2 - 2.54x_3^2 - 5.77x_1x_2 - 2.49x_1x_3 - 5.29x_2x_3 - 0.54 \le 0.59367 \\
&-0.40623 \le -0.39 - 0.11x_1 + 0.55x_2 + 0.45x_3 + 0.26x_1^2 - 3.46x_2^2 - 0.79x_3^2 + 0.4x_1x_2 - 0.71x_1x_3 + 0.33x_2x_3 + 0.37 \le -0.33237 \\
&0.433845 \le 0.32 - 0.17x_1 + 0.59x_2 + 1.47x_3 - 0.14x_1^2 - 2.35x_2^2 - 3.02x_3^2 + 0.21x_1x_2 + 0.68x_1x_3 - 1.77x_2x_3 - 0.48 \le 0.53025 \\
&202.2143 \le 1098.08 + 600.73x_1 - 8716.68x_2 - 5754.12x_3 + 1716.24x_1^2 + 28624.78x_2^2 + 12017.57x_3^2 + 3734.83x_1x_2 \\
&- 5760.43x_1x_3 + 13181x_2x_3 - 225 \le 247.1508 \\
&0 \le x_1 \le 0.4 \\
&0.05 \le x_2 \le 0.15 \\
&0.1 \le x_3 \le 0.3
\end{aligned}
$$

Appendix E. Expected Cycle Times and Raw Process Time of Supply Chain

The estimated cycle time with raw process time for products, plants and priorities in

FAB:

Product		Produc1		Produc2		Produc3	
FAB	**Priority**	**Expect Cycle Time**	**Row Process Time**	**Expect Cycle Time**	**Row Process Time**	**Expect Cycle Time**	**Row Process Time**
FAB1	**Priority1**	107786.3	43545.6	101322.4	55065.6	106238.2	65491.2
(Min)	**Priority2**	138157.9	46425.6	143855.6	59745.6	154811.1	69393.6
	Priority3	198175.9	49305.6	200123.4	63705.6	191290.5	72720
FAB2	**Priority1**	106754.1	46569.6	110731.8	55209.6	112257.4	66974.4
	Priority2	140035.7	49449.6	144816.8	56433.6	155978.6	70905.6
	Priority3	203083.4	52329.6	196421.5	63849.6	193376.6	75643.2
FAB3	**Priority1**	116164.1	45576	112373	57096	not	not
	Priority2	138316.2	48456	147136.1	59587.2	not	not
	Priority3	202811.6	51336	206241.8	62856	not	not
FAB4	**Priority1**	140333.1	29966.4	117665.1	55886.4	not	not
	Priority2	138654	47246.4	146215.2	58348.8	not	not
	Priority3	194274.4	50126.4	211231.9	63086.4	not	not
FAB5	**Priority1**	112853.2	45748.8	not	not	not	not
	Priority2	139512.7	48628.8	not	not	not	not
	Priority3	198760.6	51508.8	not	not	not	not
FAB6	**Priority1**	136227	43027.2	not	not	not	not
	Priority2	137850.2	45907.2	not	not	not	not
	Priority3	206452.7	48787.2	not	not	not	not

The estimated cycle time with raw process time for products, plants and priorities in

Assembly:

Product		Product1		Product2		Product3	
Fab	**Priority**	**Expect Cycle Time**	**Row Process Time**	**Expect Cycle Time**	**Row Process Time**	**Expect Cycle Time**	**Row Process Time**
Asse1	**Priority1**	16819.8	8523.07	17240.97	9001.94	17598.24	9403.24
(Min)	**Priority2**	21227.3	9963.07	21754.13	10585.94	22099.07	10987.24
	Priority3	26258.9	12123.07	25022.45	12745.94	28684.21	13147.24
Asse2	**Priority1**	16852.1	8560.02	17368.85	9146.06	17727.48	9547.3
	Priority2	19715.8	10000.02	20231.61	10586.06	20588.89	10987.3
	Priority3	24732.4	12160.02	25239.6	12746.06	25590.77	13147.3

The estimated cycle time with raw process time for products, plants and priorities in Final test:

Product		Product1		Product2		Product3	
Fab	**Priority**	**Expect Cycle Time**	**Row Process Time**	**Expect Cycle Time**	**Row Process Time**	**Expect Cycle Time**	**Row Process Time**
FT1	**Priority1**	22869.36	15051.3	24142.48	16376.1	24261.33	16499.34
(Min)	**Priority2**	27098.66	16491.3	28347.25	17816.1	28463.96	17939.34
	Priority3	33953.16	22251.3	35223.73	23576.1	35342.23	23699.34
FT2	**Priority1**	22014.74	15170.94	23511	16713.24	23352.4	16550.16
	Priority2	27210.93	16610.94	28666.72	18153.24	28512.11	17990.16
	Priority3	33093.73	22370.94	34583.09	23913.24	34425.29	23750.16

Appendix F. Problem Formulation of Supply Chain Case

Minimize:

$$\begin{aligned}
&15(3.04028-0.76932\rho_{18}-5.324071\pi_{11}\rho_{18}-1.70306\rho_{22}+2.55627\pi_{11}\rho_{22}-3.21206\rho_{24}+2.0617\pi_{11}\rho_{24}+3.35593\rho_{22}\rho_{24}\\
&+6.45807\rho_{24}^{2}-6.01907\rho_{26}+2.64128\pi_{11}\rho_{26}+2.43013\rho_{18}\rho_{26}+3.95107\rho_{22}\rho_{26}+3.59718\rho_{24}\rho_{26}+6.27775\rho_{26}^{2}\\
&-2.08025\rho_{24}\rho_{33}+0.95507\rho_{33}^{2}-0.89682\rho_{22}\pi_{12}-2.73278\rho_{22}\pi_{21}-2.94563\rho_{24}\pi_{21}+4.33951\pi_{11}\pi_{31}+2.20954\rho_{22}\pi_{31}\\
&+2.26427\rho_{26}\pi_{31}-3.90953\rho_{33}\pi_{31}+3.88601\pi_{11}\pi_{32}-0.69994\rho_{33}\pi_{32}-3.96577\pi_{11}\rho_{11}-1.57582\rho_{22}\rho_{14}-1.4345\rho_{15}\\
&-6.01892\pi_{11}\rho_{15}+4.19858\rho_{26}\rho_{15}+6.76398\mathrm{x}3\rho_{15}-1)^{2}\\
&+5(6.82052-9.14259*\pi_{11}^{2}-12.10215\rho_{17}-3.46978\pi_{11}\rho_{17}+20.72719\rho_{17}^{2}-9.47746\rho_{18}+3.59012\pi_{11}\rho_{18}+4.71436\rho_{17}\rho_{18}\\
&+17.3784\rho_{18}^{2}-1.04104\rho_{22}+1.06997\pi_{11}\rho_{22}-1.52753\rho_{17}\rho_{22}+2.08932\rho_{24}+3.13456\pi_{11}\rho_{24}-2.39502\rho_{18}\rho_{24}\\
&+0.98932\rho_{22}\rho_{26}+1.09759\rho_{24}\rho_{26}-1.16146\rho_{26}^{2}-10.02525\rho_{33}+2.97032\rho_{22}\rho_{33}-2.4221\rho_{24}\rho_{33}+0.81731\rho_{26}\rho_{33}\\
&+8.44666\rho_{33}^{2}-2.15257\rho_{24}\pi_{12}+3.91556\pi_{21}-4.36691\pi_{11}\pi_{21}+9.21784\rho_{17}\pi_{21}-3.07236\rho_{24}\pi_{21}-0.97223\rho_{33}\pi_{21}\\
&-24.31008\pi_{21}^{2}-3.78231\pi_{22}+4.3232\pi_{11}\pi_{22}+6.59604\rho_{17}\pi_{22}+4.53952\rho_{18}\pi_{22}-2.54099\rho_{22}\pi_{22}-3.81107\rho_{24}\pi_{22}\\
&-4.19952\rho_{26}\pi_{22}+2.13798\rho_{33}\pi_{22}+3.96131\pi_{21}\pi_{22}-1.44225\pi_{31}+2.77296\pi_{11}\pi_{31}+9.04688\rho_{17}\pi_{31}-2.53356\rho_{22}\pi_{31}\\
&-2.78439\rho_{24}\pi_{31}-1.32614\rho_{33}\pi_{31}+3.9015\pi_{22}\pi_{31}+2.3744\rho_{18}\pi_{32}+1.65015\rho_{33}\pi_{32}-1.93648\pi_{21}\pi_{32}+2.0288\pi_{22}\pi_{32}\\
&-5.22684\pi_{32}^{2}-3.75599\rho_{11}+3.06233\rho_{17}\rho_{11}+1.88448\rho_{22}\rho_{11}+1.75657\rho_{24}\rho_{11}+2.42335\rho_{26}\rho_{11}+2.63329\pi_{12}\rho_{11}\\
&+3.61077\pi_{21}\rho_{11}+3.59251\pi_{31}\rho_{11}-4.41291\rho_{14}+8.54303\rho_{17}\rho_{14}+8.54006\rho_{18}\rho_{14}+2.04767\rho_{24}\rho_{14}-2.5092\rho_{26}\rho_{14}\\
&+4.82369\pi_{21}\rho_{14}+1.48026\pi_{22}\rho_{14}+2.17926\pi_{31}\rho_{14}+1.67976\pi_{32}\rho_{14}+3.64868\rho_{11}\rho_{14}+6.19947\rho_{17}\rho_{15}+2.85905\rho_{18}\rho_{15}\\
&-0.9847\rho_{22}\rho_{15}+1.70062\rho_{26}\rho_{15}+4.36643\rho_{33}\rho_{15}-3.35128\pi_{21}\rho_{15}+9.41675\pi_{22}\rho_{15}+2.36581\rho_{11}\rho_{15}+2.76393\rho_{14}\rho_{15}\\
&-19.03243\rho_{15}^{2}-1)^{2}\\
&+(2.12136-5.26463\rho_{17}\rho_{18}+1.23765\rho_{17}\rho_{22}+1.29732\rho_{22}^{2}+3.45607\rho_{17}\rho_{24}+1.0604\rho_{18}\rho_{24}-1.50806\rho_{22}\rho_{24}\\
&+1.71873\rho_{17}\rho_{26}-1.44593\rho_{22}\rho_{26}-2.52953\rho_{24}\rho_{26}-0.61691\rho_{33}-1.39983\rho_{18}\rho_{33}+1.00621\rho_{24}\rho_{33}+0.42542\rho_{26}\rho_{33}\\
&-0.80321\pi_{11}\pi_{12}-0.52938\pi_{12}\pi_{21}+1.61145\rho_{18}\pi_{32}+1.08404\rho_{26}\pi_{32}+2.15541\rho_{22}\rho_{11}+1.59615\rho_{33}\rho_{14}-2.36941\pi_{32}\rho_{14}\\
&-3.6599\rho_{17}\rho_{15}+3.59611\rho_{18}\rho_{15}+3.08604\rho_{26}\rho_{15}+2.50117\rho_{33}\rho_{15}-0.74715\pi_{22}\rho_{15}-4.90862\rho_{15}^{2}-1)^{2}\\
&+15(10.76424-30.39664\rho_{18}+20.91593\rho_{17}\rho_{18}+44.38031\rho_{18}^{2}-7.04376\rho_{22}+3.75327\pi_{11}\rho_{22}-19.49143\rho_{17}\rho_{22}-14.41669\rho_{24}\\
&-19.90259\rho_{17}\rho_{24}+17.13541\rho_{22}\rho_{24}+19.52764\rho_{24}^{2}-16.44176\rho_{26}-20.93092\rho_{17}\rho_{26}+15.80926\rho_{22}\rho_{26}+16.97536\rho_{24}\rho_{26}\\
&+23.21307\rho_{26}^{2}+5.01638\rho_{22}\pi_{22}-6.09423\pi_{31}+9.03262\rho_{22}\pi_{31}+9.70458\rho_{24}\pi_{31}+7.29345\pi_{12}\pi_{31}-16.26179\rho_{11}\\
&+18.03107\rho_{17}\rho_{11}+27.58228\rho_{18}\rho_{11}-5.73555\pi_{12}\rho_{11}+28.52365\rho_{17}\rho_{14}+29.47225\rho_{18}\rho_{14}-5.43344\rho_{22}\rho_{14}+23.20821\rho_{11}\rho_{14}\\
&-55.31215\rho_{14}^{2}-17.66627\rho_{15}+17.71866\rho_{17}\rho_{15}+22.28432\rho_{18}\rho_{15}+11.24435\rho_{26}\rho_{15}-5.38951\pi_{22}\rho_{15}+32.58414\rho_{11}\rho_{15}\\
&+25.65426\rho_{14}\rho_{15}-0)^{2}\\
&+5(1.63076+12.34\pi_{11}-23.40161\rho_{17}+20.84788\pi_{11}\rho_{17}+83.34814\rho_{17}^{2}-37.43988\rho_{18}+66.00568\rho_{18}^{2}-16.55726\pi_{11}\rho_{22}\\
&-19.97751\rho_{17}\rho_{24}-11.0329\rho_{18}\rho_{24}+17.01881\rho_{24}^{2}+15.22815\rho_{26}-6.40817\rho_{17}\rho_{26}-17.29442\rho_{18}\rho_{26}+5.59763\rho_{22}\rho_{26}\\
&-7.18731\rho_{24}\rho_{26}-26.26289\rho_{26}^{2}-6.38914\rho_{17}\rho_{33}+5.69988\rho_{18}\rho_{33}-17.48533\pi_{11}\pi_{12}+27.66893\rho_{18}\pi_{12}-9.23598\rho_{26}\pi_{12}\\
&+4.73927\rho_{33}\pi_{12}-12.08031\rho_{17}\pi_{21}+7.07731\rho_{22}\pi_{21}-6.65594\rho_{24}\pi_{21}+16.89891\rho_{26}\pi_{21}-20.36283\pi_{12}\pi_{21}-10.46857\rho_{17}\pi_{22}\\
&+11.55669\rho_{18}\pi_{22}+7.36825\pi_{21}\pi_{22}+28.55784\pi_{31}-10.71991\rho_{22}\pi_{31}-26.40251\rho_{26}\pi_{31}+8.96146\rho_{33}\pi_{31}+14.11937\pi_{12}\pi_{31}\\
&-31.24011\pi_{21}\pi_{31}-14.06794\pi_{22}\pi_{31}-106.56485\pi_{31}^{2}-28.0642\pi_{11}\pi_{32}+8.3371\rho_{17}\pi_{32}+16.97524\rho_{18}\pi_{32}+10.80967\rho_{22}\pi_{32}\\
&+6.79337\rho_{24}\pi_{32}-8.22914\rho_{26}\pi_{32}+22.48927\pi_{21}\pi_{32}-22.4352\pi_{32}^{2}-13.69445\rho_{11}+30.29128\rho_{17}\rho_{11}+23.13033\rho_{18}\rho_{11}\\
&+18.90353\rho_{24}\rho_{11}-7.51322\pi_{12}\rho_{11}+39.77696\rho_{14}-27.57\pi_{11}\rho_{14}+11.53332\rho_{18}\rho_{14}-8.72286\rho_{24}\rho_{14}-11.28578\rho_{33}\rho_{14}\\
&-23.51398\pi_{12}\rho_{14}-11.83482\pi_{22}\rho_{14}+14.90828\pi_{31}\rho_{14}+13.26411\pi_{32}\rho_{14}+18.3684\rho_{11}\rho_{14}-111.78177\rho_{14}^{2}-11.657\rho_{15}\\
&+15.08398\rho_{17}\rho_{15}+37.72783\rho_{18}\rho_{15}-10.90103\rho_{22}\rho_{15}-19.83233\rho_{24}\rho_{15}+24.32188\rho_{26}\rho_{15}+18.62692\pi_{12}\rho_{15}+16.55446\pi_{22}\rho_{15}\\
&-16.78231\pi_{31}\rho_{15}14.60959\rho_{14}\rho_{15}-0)^{2}\\
&+(4.5013+5.9327\pi_{11}\rho_{18}-15.36454\rho_{18}^{2}-7.60712\rho_{22}+5.69776\rho_{18}\rho_{22}+3.95581\rho_{22}\rho_{24}+3.26237\rho_{24}^{2}-9.91956\rho_{26}+8.08492\rho_{22}\rho_{26}\\
&+3.67909\rho_{24}\rho_{26}+11.14849\rho_{26}^{2}-6.14005\pi_{11}\rho_{33}+6.81903\rho_{22}\rho_{33}-7.9766\rho_{24}\rho_{33}-4.16798\rho_{22}\pi_{12}+5.63863\pi_{11}\pi_{22}+5.87533\pi_{11}\rho_{11}\\
&-6.22671\rho_{18}\rho_{11}+7.59146\rho_{22}\rho_{11}-5.04495\pi_{22}\rho_{11}-4.23574\rho_{22}\rho_{14}+6.20288\rho_{24}\rho_{14}-8.91496\rho_{15}+13.54669\rho_{18}\rho_{15}+3.29294\rho_{22}\rho_{15}\\
&+16.43614\rho_{26}\rho_{15}+4.453\rho_{33}\rho_{15}+3.02565\pi_{12}\rho_{15})^{2}
\end{aligned}$$

subject to:

$$3.04028 - 0.76932\rho_{18} - 5.324071\pi_{11}\rho_{18} - 1.70306\rho_{22} + 2.55627\pi_{11}\rho_{22} - 3.21206\rho_{24} + 2.0617\pi_{11}\rho_{24} + 3.35593\rho_{22}\rho_{24}$$

$$+ 6.45807\rho_{24}^{2} - 6.01907\rho_{26} + 2.64128\pi_{11}\rho_{26} + 2.43013\rho_{18}\rho_{26} + 3.95107\rho_{22}\rho_{26} + 3.59718\rho_{24}\rho_{26} + 6.27775\rho_{26}^{2}$$

$$- 2.08025\rho_{24}\rho_{33} + 0.95507\rho_{33}^{2} - 0.89682\rho_{22}\pi_{12} - 2.73278\rho_{22}\pi_{21} - 2.94563\rho_{24}\pi_{21} + 4.33951\pi_{11}\pi_{31} + 2.20954\rho_{22}\pi_{31}$$

$$+ 2.26427\rho_{26}\pi_{31} - 3.90953\rho_{33}\pi_{31} + 3.88601\pi_{11}\pi_{32} - 0.69994\rho_{33}\pi_{32} - 3.96577\pi_{11}\rho_{11} - 1.57582\rho_{22}\rho_{14} - 1.4345\rho_{15}$$

$$- 6.01892\pi_{11}\rho_{15} + 4.19858\rho_{26}\rho_{15} + 6.76398\text{x}3\rho_{15} \geq 1$$

$$6.82052 - 9.14259 * \pi_{11}^{2} - 12.10215\rho_{17} - 3.46978\pi_{11}\rho_{17} + 20.72719\rho_{17}^{2} - 9.47746\rho_{18} + 3.59012\pi_{11}\rho_{18} + 4.71436\rho_{17}\rho_{18}$$

$$+ 17.3784\rho_{18}^{2} - 1.04104\rho_{22} + 1.06997\pi_{11}\rho_{22} - 1.52753\rho_{17}\rho_{22} + 2.08932\rho_{24} + 3.13456\pi_{11}\rho_{24} - 2.39502\rho_{18}\rho_{24}$$

$$+ 0.98932\rho_{22}\rho_{26} + 1.09759\rho_{24}\rho_{26} - 1.16146\rho_{26}^{2} - 10.02525\rho_{33} + 2.97032\rho_{22}\rho_{33} - 2.4221\rho_{24}\rho_{33} + 0.81731\rho_{26}\rho_{33}$$

$$+ 8.44666\rho_{33}^{2} - 2.15257\rho_{24}\pi_{12} + 3.91556\pi_{21} - 4.36691\pi_{11}\pi_{21} + 9.21784\rho_{17}\pi_{21} - 3.07236\rho_{24}\pi_{21} - 0.97223\rho_{33}\pi_{21}$$

$$- 24.31008\pi_{21}^{2} - 3.78231\pi_{22} + 4.3232\pi_{11}\pi_{22} + 6.59604\rho_{17}\pi_{22} + 4.53952\rho_{18}\pi_{22} - 2.54099\rho_{22}\pi_{22} - 3.81107\rho_{24}\pi_{22}$$

$$- 4.19952\rho_{26}\pi_{22} + 2.13798\rho_{33}\pi_{22} + 3.96131\pi_{21}\pi_{22} - 1.44225\pi_{31} + 2.77296\pi_{11}\pi_{31} + 9.04688\rho_{17}\pi_{31} - 2.53356\rho_{22}\pi_{31}$$

$$- 2.78439\rho_{24}\pi_{31} - 1.32614\rho_{33}\pi_{31} + 3.9015\pi_{22}\pi_{31} + 2.3744\rho_{18}\pi_{32} + 1.65015\rho_{33}\pi_{32} - 1.93648\pi_{21}\pi_{32} + 2.0288\pi_{22}\pi_{32}$$

$$- 5.22684\pi_{32}^{2} - 3.75599\rho_{11} + 3.06233\rho_{17}\rho_{11} + 1.88448\rho_{22}\rho_{11} + 1.75657\rho_{24}\rho_{11} + 2.42335\rho_{26}\rho_{11} + 2.63329\pi_{12}\rho_{11}$$

$$+ 3.61077\pi_{21}\rho_{11} + 3.59251\pi_{31}\rho_{11} - 4.41291\rho_{14} + 8.54303\rho_{17}\rho_{14} + 8.54006\rho_{18}\rho_{14} + 2.04767\rho_{24}\rho_{14} - 2.5092\rho_{26}\rho_{14}$$

$$+ 4.82369\pi_{21}\rho_{14} + 1.48026\pi_{22}\rho_{14} + 2.17926\pi_{31}\rho_{14} + 1.67976\pi_{32}\rho_{14} + 3.64868\rho_{11}\rho_{14} + 6.19947\rho_{17}\rho_{15} + 2.85905\rho_{18}\rho_{15}$$

$$- 0.9847\rho_{22}\rho_{15} + 1.70062\rho_{26}\rho_{15} + 4.36643\rho_{33}\rho_{15} - 3.35128\pi_{21}\rho_{15} + 9.41675\pi_{22}\rho_{15} + 2.36581\rho_{11}\rho_{15} + 2.76393\rho_{14}\rho_{15}$$

$$- 19.03243\rho_{15}^{2} \geq 1$$

$$2.12136 - 5.26463\rho_{17}\rho_{18} + 1.23765\rho_{17}\rho_{22} + 1.29732\rho_{22}^{2} + 3.45607\rho_{17}\rho_{24} + 1.0604\rho_{18}\rho_{24} - 1.50806\rho_{22}\rho_{24}$$

$$+ 1.71873\rho_{17}\rho_{26} - 1.44593\rho_{22}\rho_{26} - 2.52953\rho_{24}\rho_{26} - 0.61691\rho_{33} - 1.39983\rho_{18}\rho_{33} + 1.00621\rho_{24}\rho_{33} + 0.42542\rho_{26}\rho_{33}$$

$$- 0.80321\pi_{11}\pi_{12} - 0.52938\pi_{12}\pi_{21} + 1.61145\rho_{18}\pi_{32} + 1.08404\rho_{26}\pi_{32} + 2.15541\rho_{22}\rho_{11} + 1.59615\rho_{33}\rho_{14} - 2.36941\pi_{32}\rho_{14}$$

$$- 3.6599\rho_{17}\rho_{15} + 3.59611\rho_{18}\rho_{15} + 3.08604\rho_{26}\rho_{15} + 2.50117\rho_{33}\rho_{15} - 0.74715\pi_{22}\rho_{15} - 4.90862\rho_{15}^{2} \geq 1$$

$$10.76424 - 30.39664\rho_{18} + 20.91593\rho_{17}\rho_{18} + 44.38031\rho_{18}^{2} - 7.04376\rho_{22} + 3.75327\pi_{11}\rho_{22} - 19.49143\rho_{17}\rho_{22} - 14.41669\rho_{24}$$

$$- 19.90259\rho_{17}\rho_{24} + 17.13541\rho_{22}\rho_{24} + 19.52764\rho_{24}^{2} - 16.44176\rho_{26} - 20.93092\rho_{17}\rho_{26} + 15.80926\rho_{22}\rho_{26} + 16.97536\rho_{24}\rho_{26}$$

$$+ 23.21307\rho_{26}^{2} + 5.01638\rho_{22}\pi_{22} - 6.09423\pi_{31} + 9.03262\rho_{22}\pi_{31} + 9.70458\rho_{24}\pi_{31} + 7.29345\pi_{12}\pi_{31} - 16.26179\rho_{11}$$

$$+ 18.03107\rho_{17}\rho_{11} + 27.58228\rho_{18}\rho_{11} - 5.73555\pi_{12}\rho_{11} + 28.52365\rho_{17}\rho_{14} + 29.47225\rho_{18}\rho_{14} - 5.43344\rho_{22}\rho_{14} + 23.20821\rho_{11}\rho_{14}$$

$$- 55.31215\rho_{14}^{2} - 17.66627\rho_{15} + 17.71866\rho_{17}\rho_{15} + 22.28432\rho_{18}\rho_{15} + 11.24435\rho_{26}\rho_{15} - 5.38951\pi_{22}\rho_{15} + 32.58414\rho_{11}\rho_{15}$$

$$+ 25.65426\rho_{14}\rho_{15} \geq 0$$

$$1.63076 + 12.34\pi_{11} - 23.40161\rho_{17} + 20.84788\pi_{11}\rho_{17} + 83.34814\rho_{17}^{2} - 37.43988\rho_{18} + 66.00568\rho_{18}^{2} - 16.55726\pi_{11}\rho_{22}$$

$$- 19.97751\rho_{17}\rho_{24} - 11.0329\rho_{18}\rho_{24} + 17.01881\rho_{24}^{2} + 15.22815\rho_{26} - 6.40817\rho_{17}\rho_{26} - 17.29442\rho_{18}\rho_{26} + 5.59763\rho_{22}\rho_{26}$$

$$- 7.18731\rho_{24}\rho_{26} - 26.26289\rho_{26}^{2} - 6.38914\rho_{17}\rho_{33} + 5.69988\rho_{18}\rho_{33} - 17.48533\pi_{11}\pi_{12} + 27.66893\rho_{18}\pi_{12} - 9.23598\rho_{26}\pi_{12}$$

$$+ 4.73927\rho_{33}\pi_{12} - 12.08031\rho_{17}\pi_{21} + 7.07731\rho_{22}\pi_{21} - 6.65594\rho_{24}\pi_{21} + 16.89891\rho_{26}\pi_{21} - 20.36283\pi_{12}\pi_{21} - 10.46857\rho_{17}\pi_{22}$$

$$+ 11.55669\rho_{18}\pi_{22} + 7.36825\pi_{21}\pi_{22} + 28.55784\pi_{31} - 10.71991\rho_{22}\pi_{31} - 26.40251\rho_{26}\pi_{31} + 8.96146\rho_{33}\pi_{31} + 14.11937\pi_{12}\pi_{31}$$

$$- 31.24011\pi_{21}\pi_{31} - 14.06794\pi_{22}\pi_{31} - 106.56485\pi_{31}^{2} - 28.0642\pi_{11}\pi_{32} + 8.3371\rho_{17}\pi_{32} + 16.97524\rho_{18}\pi_{32} + 10.80967\rho_{22}\pi_{32}$$

$$+ 6.79337\rho_{24}\pi_{32} - 8.22914\rho_{26}\pi_{32} + 22.48927\pi_{21}\pi_{32} - 22.4352\pi_{32}^{2} - 13.69445\rho_{11} + 30.29128\rho_{17}\rho_{11} + 23.13033\rho_{18}\rho_{11}$$

$$+ 18.90353\rho_{24}\rho_{11} - 7.51322\pi_{12}\rho_{11} + 39.77696\rho_{14} - 27.57\pi_{11}\rho_{14} + 11.53332\rho_{18}\rho_{14} - 8.72286\rho_{24}\rho_{14} - 11.28578\rho_{33}\rho_{14}$$

$$- 23.51398\pi_{12}\rho_{14} - 11.83482\pi_{22}\rho_{14} + 14.90828\pi_{31}\rho_{14} + 13.26411\pi_{32}\rho_{14} + 18.3684\rho_{11}\rho_{14} - 111.78177\rho_{14}^{2} - 11.657\rho_{15}$$

$$+ 15.08398\rho_{17}\rho_{15} + 37.72783\rho_{18}\rho_{15} - 10.90103\rho_{22}\rho_{15} - 19.83233\rho_{24}\rho_{15} + 24.32188\rho_{26}\rho_{15} + 18.62692\pi_{12}\rho_{15} + 16.55446\pi_{22}\rho_{15}$$

$$- 16.78231\pi_{31}\rho_{15}14.60959\rho_{14}\rho_{15} \geq 0$$

$$4.5013 + 5.9327\pi_{11}\rho_{18} - 15.36454\rho_{18}^{2} - 7.60712\rho_{22} + 5.69776\rho_{18}\rho_{22} + 3.95581\rho_{22}\rho_{24} + 3.26237\rho_{24}^{2} - 9.91956\rho_{26} + 8.08492\rho_{22}\rho_{26}$$

$$+ 3.67909\rho_{24}\rho_{26} + 11.14849\rho_{26}^{2} - 6.14005\pi_{11}\rho_{33} + 6.81903\rho_{22}\rho_{33} - 7.9766\rho_{24}\rho_{33} - 4.16798\rho_{22}\pi_{12} + 5.63863\pi_{11}\pi_{22} + 5.87533\pi_{11}\rho_{11}$$

$$- 6.22671\rho_{18}\rho_{11} + 7.59146\rho_{22}\rho_{11} - 5.04495\pi_{22}\rho_{11} - 4.23574\rho_{22}\rho_{14} + 6.20288\rho_{24}\rho_{14} - 8.91496\rho_{15} + 13.54669\rho_{18}\rho_{15} + 3.29294\rho_{22}\rho_{15}$$

$$+ 16.43614\rho_{26}\rho_{15} + 4.453\rho_{33}\rho_{15} + 3.02565\pi_{12}\rho_{15} \geq 0$$

$$\pi_{11} + \pi_{12} + \pi_{13} = 1$$

$$\pi_{21} + \pi_{22} + \pi_{23} = 1$$

$$\pi_{31} + \pi_{32} + \pi_{33} = 1$$

$$\rho_{11} + \rho_{14} + \rho_{15} + \rho_{17} + \rho_{18} + \rho_{19} = 1$$

$$\rho_{22} + \rho_{24} + \rho_{26} + \rho_{27} = 1$$

$$\rho_{33} + \rho_{34} = 1$$

$$5 \le \frac{2}{4.7}\pi_{11} + \frac{1.7}{4.7}\pi_{21} + \frac{1}{4.7}\pi_{31} \le 15$$

$$10 \le \frac{2}{4.7}\pi_{12} + \frac{1.7}{4.7}\pi_{22} + \frac{1}{4.7}\pi_{32} \le 25$$

$$60 \le \frac{2}{4.7}\pi_{13} + \frac{1.7}{4.7}\pi_{23} + \frac{1}{4.7}\pi_{33} \le 85$$

$$4.2553191 \le \frac{2}{4.7}\rho_{11} \le 8.5106383$$

$$3.6170213 \le \frac{1.7}{4.7}\rho_{22} \le 10.8510638$$

$$8.5106383 \le \frac{1.7}{4.7}\rho_{33} \le 12.7659574$$

$$16.383 \le \frac{2}{4.7}\rho_{14} + \frac{1.7}{4.7}\rho_{24} + \frac{1}{4.7}\rho_{34} \le 32.1277$$

$$4.2553191 \le \frac{2}{4.7}\rho_{15} \le 8.5106383$$

$$3.6170213 \le \frac{1.7}{4.7}\rho_{26} \le 10.8510638$$

$$7.8723 \le \frac{2}{4.7}\rho_{17} + \frac{1.7}{4.7}\rho_{27} \le 33.8298$$

$$4.2553191 \le \frac{2}{4.7}\rho_{18} \le 8.5106383$$

$$0 \le \frac{2}{4.7}\rho_{19} \le 21.2765957$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{11} \times 0.831313 + \frac{1.7}{4.7}\rho_{22} \times 1.051238 + \frac{1}{4.7}\rho_{34} \times 1.250269 \right) \le 21.62003$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{14} \times 0.861823 + \frac{1.7}{4.7}\rho_{24} \times 1.021716 + 1\frac{1}{4.7}\rho_{34} \times 1.239437 \right) \le 20.2651$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{15} \times 0.895949 + \frac{1.7}{4.7}\rho_{26} \times 1.122413 \right) \le 13.57879$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{17} \times 0.845686 + \frac{1.7}{4.7}\rho_{27} \times 1.181546 \right) \le 16.6863$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{18} \right) \le 17.7025$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{19} \right) \le 10.14728$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{11} \times 0.959422 + \frac{1.7}{4.7}\rho_{22} \times 1.013327 + \frac{1}{4.7}\rho_{33} \times 1.0585 + \frac{2}{4.7}\rho_{15} \times 0.959422 \quad \frac{1.7}{4.7}\rho_{26} \times 1.013327 \right.$$
$$\left. + \frac{2}{4.7}\rho_{18} \times 0.959422 \right) \le 50$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{14} \times 0.953016 + \frac{1.7}{4.7}\rho_{24} \times 1.018262 + \frac{1.7}{4.7}\rho_{34} \times 1.062922 + \frac{2}{4.7}\rho_{17} \times 0.953016 + \frac{1.7}{4.7}\rho_{27} \times 1.018262 \right.$$
$$\left. + \frac{2}{4.7}\rho_{19} \times 0.953016 \right) \le 50$$

$$0.8 \times \left(\frac{2}{4.7}\rho_{11} \times 0.93226 + \frac{2}{4.7}\rho_{14} \times 0.93226 + \frac{1.7}{4.7}\rho_{24} \times 1.014317 + \frac{1}{4.7}\rho_{34} \times 1.111142 + \frac{2}{4.7}\rho_{18} \times 0.93226 \right.$$
$$\left. + \frac{2}{4.7}\rho_{19} \times 0.93226 \right) \le 50$$

$$0.8 \times \left(\frac{1.7}{4.7}\rho_{22} \times 1.023555 + \frac{1.7}{4.7}\rho_{33} \times 1.101756 + \frac{2}{4.7}\rho_{15} \times 0.929101 + \frac{1.7}{4.7}\rho_{26} \times 1.023555 + \frac{2}{4.7}\rho_{17} \times 0.929101 \right.$$
$$\left. + \frac{1.7}{4.7}\rho_{27} \times 1.023555 \right) \le 50$$

$5 \le \pi_{11} \le 15$

$10 \le \pi_{12} \le 25$

$0 \le \pi_{13} \le 100$

$5 \le \pi_{21} \le 15$

$10 \le \pi_{22} \le 25$

$0 \le \pi_{23} \le 100$

$5 \le \pi_{31} \le 15$

$10 \le \pi_{32} \le 25$

$0 \le \pi_{33} \le 100$

$10 \le \rho_{11} \le 20$

$10 \le \rho_{14} \le 20$

$10 \le \rho_{15} \le 20$

$10 \le \rho_{17} \le 20$

$10 \le \rho_{18} \le 20$

$0 \le \rho_{19} \le 100$

$10 \le \rho_{22} \le 30$

$10 \le \rho_{24} \le 30$

$10 \le \rho_{26} \le 30$

$0 \le \rho_{27} \le 100$

$40 \le \rho_{33} \le 60$

$0 \le \rho_{34} \le 100$