

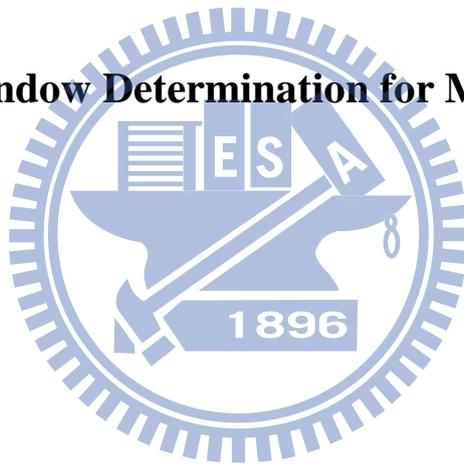
國立交通大學

電控工程研究所

碩 士 論 文

攔截飛彈的發射時間窗口決定

Launch Time Window Determination for Missile Interceptor



研 究 生 ： 林 彥 伯

指 導 教 授 ： 林 清 安 教 授

中華民國九十九年十月

攔截飛彈的發射時間窗口決定

Launch Time Window Determination for Missile Interceptor

研究生：林彥伯

Student: Yen-Po Lin

指導教授：林清安

Advisor: Prof. Ching-An Lin



Submitted to Institute of Electrical and Control Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of Master

In

Electrical and Control Engineering

October 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年十月

攔截飛彈的發射時間窗口決定

學生：林彥伯

指導教授：林清安 教授

國立交通大學電控工程研究所

中文摘要

本論文探討攔截飛彈的發射時間窗口(Launch Time Window)該如何適當選擇，如何描述飛彈及目標的位置與速度函數，以及分析攔截成功的必要條件並建立最佳化問題，透過發展處罰函數(PF)演算法及連續二次規劃(SQP)演算法，來求出發射前等待的延遲時間最大值及最小值，即發射時間窗口。最後提供三種不同來襲目標來驗證所求出的發射時間窗口是否符合實際攔截情形。透過模擬的數值結果顯示針對不同目標所求出的發射時間窗口均有不錯的準確度且亦相當符合實際攔截情形。

Launch Time Window Determination for Missile Interceptor

Student: Yen-Po Lin

Advisor: Prof. Ching-An Lin

Institute of Electrical and Control Engineering
National Chiao Tung University

Abstract

When an attacking missile is detected, the missile defense system must decide when to launch the missile interceptor. The choice of a proper launch time is important because given the physical constraints on the interceptor, a poor choice of launch time alone may cause the interception to fail. This thesis discusses the determination of launch time window for a missile interceptor, which is formulated as constrained optimization problems.

The earliest and the latest allowable launch times are determined by numerically solve the corresponding minimization and maximization problems. Three incoming threats are considered. Numerical results show good agreement with results obtained via six degree of freedom simulations employing midcourse guidance.

誌 謝

本論文可以順利完成，首先要感謝我的指導教授林清安博士在研究上提供我許多寶貴的經驗及建議，使我得以能在研究上順利進行。不僅僅是在課業上的幫助，在待人處世及面對問題該有正確、積極的態度也是我值得學習的榜樣。接著感謝口試委員鄧清政教授與陳科祥博士仔細地閱讀我的論文並給予我適當的指教。

此外也感謝所有在求學過程中幫助我的朋友，特別是606實驗室的吳建賢與林世修學長，同窗邱泰偉、蕭惟庭及學弟卓國展在日常生
活上的照顧及學習上的討論，陪伴我渡過這兩年充實且美好的歲月。

最後也要特別感謝我的家人無怨無悔的為我付出，使我衣食無缺，讓我更有學習的動力。謝謝我的女友佩蓉默默的支持我，鼓勵我。謝謝交通大學提供我如此好的學習環境，讓我能在專業領域更上層樓。

目錄

中文摘要.....	i
英文摘要.....	ii
誌謝.....	iii
目錄.....	iv
表目錄.....	vii
圖目錄.....	ix
第一章、緒論.....	1
第二章、運動方程式.....	3
2.1 座標系統.....	3
2.2 飛彈六自由度運動方程式.....	4
2.2.1 平移運動方程式.....	4
2.2.2 轉動方程式.....	5
2.2.3 尤拉角(Euler angle).....	5
2.2.4 尤拉方程式(Euler equation).....	7
2.3 目標運動方程式.....	7
第三章、飛彈及目標的速度及位置函數.....	9
3.1 各階段時間的定義.....	9
3.2 變數的定義.....	10
3.3 目標的速度及位置函數.....	11
3.4 飛彈的速度及位置函數.....	11
3.5 第二節火箭加速度分析.....	13
第四章、最佳化問題的建構.....	15
4.1 攔截的必要條件.....	15
4.2 最佳化問題的建立.....	18

4.3 延遲時間最大值與最小值.....	19
第五章、最佳化演算法的發展.....	21
5.1 處罰函數(PF)演算法	21
5.2 PF 範例	23
5.2.1 PF 範例最小值問題.....	23
5.2.2 PF 範例最大值問題.....	25
5.3 連續二次規劃(SQP)演算法.....	27
5.4 SQP 範例	29
5.4.1 SQP 範例最小值問題.....	29
5.4.2 SQP 範例最大值問題.....	30
第六章、數值計算及模擬.....	33
6.1 飛彈的推力及物性.....	33
6.2 模擬初始條件.....	34
6.3 模擬結果.....	39
6.3.1 Simulink 模擬目標一來襲.....	40
6.3.1.1 執行 PF 演算法求解目標一.....	44
6.3.1.2 執行 SQP 演算法求解目標一.....	46
6.3.2 Simulink 模擬目標二來襲.....	47
6.3.2.1 執行 PF 演算法求解目標二.....	50
6.3.2.2 執行 SQP 演算法求解目標二.....	52
6.3.3 Simulink 模擬目標三來襲.....	53
6.3.3.1 執行 PF 演算法求解目標三.....	55
6.3.3.2 執行 SQP 演算法求解目標三.....	57
6.4 延遲時間評估與驗證.....	57
6.4.1 當目標一來襲時	59
6.4.2 當目標二來襲時	62

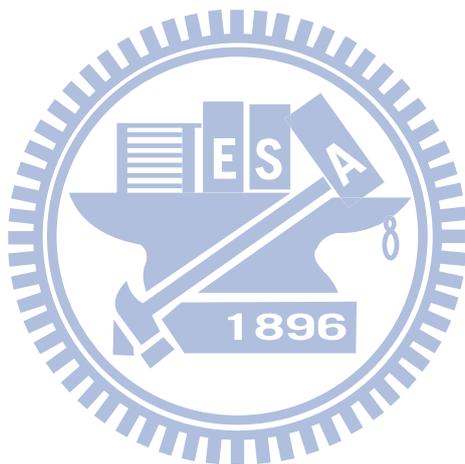
6.4.3 當目標三來襲時	64
6.5 不同 h_r 參數的數值結果	67
第七章、結論.....	71
參考資料.....	72
附錄 1 主程式碼.....	73
附錄 2 限制式程式碼.....	74
附錄 3 PF 程式碼.....	76
附錄 4 SQP 程式碼.....	86
附錄 5 SQP 求解不同目標的 LNS.....	91



表目錄

表 2.1：慣性座標轉換至體座標的旋轉順序.....	6
表 4.1：選擇不同延遲時間對攔截的影響.....	20
表 5.1：PF 求解範例最小值相關數據.....	24
表 5.2：PF 求解範例最大值相關數據.....	26
表 5.3：SQP 求解範例最小值相關數據.....	30
表 5.4：SQP 求解範例最大值相關數據.....	31
表 5.5：最小值問題方法比較.....	31
表 5.6：最大值問題方法比較.....	31
表 6.1：比較探空五號與本論文火箭的推力與作用時間.....	33
表 6.2：比較探空五號與本論文火箭的燃料質量.....	34
表 6.3：飛彈燃料質量分配狀況.....	34
表 6.4：目標資料庫.....	34
表 6.5：模擬目標一在不同延遲時間的攔截情形.....	41
表 6.6：PF 求解目標一延遲時間最大值.....	45
表 6.7：PF 求解目標一延遲時間最小值.....	46
表 6.8：SQP 求解目標一延遲時間最大值.....	46
表 6.9：SQP 求解目標一延遲時間最小值.....	47
表 6.10：模擬目標二在不同延遲時間的攔截情形.....	48
表 6.11：PF 求解目標二延遲時間最大值.....	51
表 6.12：PF 求解目標二延遲時間最小值.....	52
表 6.13：SQP 求解目標二延遲時間最大值.....	52
表 6.14：SQP 求解目標二延遲時間最小值.....	53
表 6.15：模擬目標三在不同延遲時間的攔截情形.....	54
表 6.16：PF 求解目標三延遲時間最大值.....	56

表 6.17：SQP 求解目標三延遲時間最大值	57
表 6.18：三種不同來襲目標的延遲時間窗口與可靠攔截時間.....	58
表 6.19：目標一求解可靠攔截時間代入模擬結果.....	59
表 6.20：目標二求解可靠攔截時間代入模擬結果.....	62
表 6.21：目標三求解可靠攔截時間代入模擬結果.....	64
表 6.22：當 HR=5 求解延遲時間最小值	67
表 6.23：當 HR=10 求解延遲時間最小值.....	68
表 6.24：不同高度差產生的延遲時間誤差及可靠攔截時間.....	69



圖目錄

圖 2.1：NED 座標示意圖	3
圖 2.2：尤拉角示意圖.....	6
圖 3.1：各時間參數的相對關係.....	10
圖 4.1：目標與飛彈對座標原點的距離示意圖.....	16
圖 5.1：PF 範例最小值驗證	24
圖 5.2：PF 範例最大值驗證	26
圖 6.1：飛行過程中的質量與推力隨時間變化示意圖.....	33
圖 6.2：不同目標的飛行軌跡圖(NED).....	35
圖 6.3：不同目標的飛行高度對時間圖.....	36
圖 6.4：飛彈與目標一的飛行軌跡圖(NED).....	42
圖 6.5：飛彈與目標一的飛行高度對時間圖.....	42
圖 6.6：目標一不同延遲時間對應的零力誤失距離圖.....	43
圖 6.7：目標一不同延遲時間對應的高度關係圖.....	44
圖 6.8：目標一延遲時間示意圖.....	45
圖 6.9：飛彈與目標二的飛行軌跡圖(NED).....	49
圖 6.10：目標二不同延遲時間對應的零力誤失距離圖.....	50
圖 6.11：目標二延遲時間示意圖.....	51
圖 6.12：飛彈與目標三的飛行軌跡圖(NED).....	55
圖 6.13：飛彈與目標三的飛行高度對時間圖.....	55
圖 6.14：目標三延遲時間示意圖.....	56
圖 6.15：驗證飛彈與目標一的飛行軌跡圖(NED).....	60
圖 6.16：驗證飛彈與目標一的飛行高度對時間圖.....	61
圖 6.17：驗證飛彈與目標一對座標原點的距離圖.....	61
圖 6.18：驗證飛彈與目標二的飛行軌跡圖(NED).....	63

圖 6.19：驗證飛彈與目標二的飛行高度對時間圖.....	63
圖 6.20：驗證飛彈與目標二對座標原點的距離圖.....	64
圖 6.21：驗證飛彈與目標三的飛行軌跡圖(NED).....	65
圖 6.22：驗證飛彈與目標三的飛行高度對時間圖.....	66
圖 6.23：驗證飛彈與目標三對座標原點的距離圖.....	66
圖 6.24：不同高度差與實際延遲時間所對應的零力誤失距離圖.....	69



第一章

緒論

早在 1995 及 1996 年台海危機時，中國假藉軍事演習之名發射飛彈至台灣附近海域並完成五階段「攻台模式」演練，包括發射彈道飛彈摧毀機場、制空演習、制海演習、登陸演習及城鎮山地戰，如此大膽的挑釁行為，使得國人日漸感受到中國飛彈的真實威脅。特別是近年來，中國積極部署對台飛彈，更多次阻止美國對台軍售，不僅在沿海地區部署超過一千多枚飛彈對準台灣，且隨著軍事預算年年增加，更在內陸的雲南及廣西柳州部署好幾百顆高精準度且射程不等的彈道飛彈，可以從不同的角度與距離瞄準台灣，其射程最遠超過 1500 公里。面對軍事武力如此強大的中國，我國更應積極發展攔截飛彈防衛系統，特別是面對射程不同的目標，該有不同的發射時間點，因此本論文主要探討當敵方朝我方發射飛彈來攻擊某一目標時，我方應如何選擇最佳發射時間來攔截目標飛彈。

當地面雷達站偵測到有一目標從遠方發射至我方附近領土時，在典型的飛彈防禦系統中，雷達偵測到來襲目標的距離通常遠大於防禦飛彈的射程，例如飛彈的有效射程在 200 至 300 公里而雷達偵測的能力超過 1000 公里。在這種狀況下，偵測到目標如果立即發射防禦飛彈，攔截必定失敗。所以我們需要有一個延遲時間 t_d 可以發射攔截飛彈使得在第二節中途導引[7]結束後產生足夠小的零力誤失距離(Zero Effort Miss, ZEM)，讓第三節截殺彈頭在與目標相對距離夠近時鎖定並開始追蹤目標，以致命中目標；而且攔截的高度通常也有一定的下限。因此飛彈發射的時間必須適當地選擇才能確保攔截成功。

如何在有限的時間範圍內進行攔截即飛彈可以發射的時間窗口(Launch Time Window)，將是本論文需要深入探討的問題。在探討過程中，為了計算上的方便與一致性，位置的單位皆使用公里(km)，速度的單位皆使用公里/每秒(km/s)，加速度的單位皆使用公里/每秒平方(km/s²)。

本論文討論的重點在於如何描述飛彈及目標的速度及位置函數，以及在滑行時間固定的狀況下如何建立最佳化問題來求出發射前等待的延遲時間最大值及最小值，此上下限定義了飛彈可以發射的時間窗口。之後介紹兩種演算法來針對建構出的最佳化問題進行求解，分別是處罰函數方法(Penalty Function, PF method)[3][5]及連續二次規劃的方法(Sequential Quadratic Programming, SQP method)[1][2][4]，並提供一個簡單的非線性最佳化問題來測試所提出的 PF 及 SQP 演算法是否具有可行性與比較兩種方法的差異性。最後提供三種不同來襲目標來說明延遲時間的重要性以及利用 PF 與 SQP 演算法求出的延遲時間窗口及飛彈飛行時間來驗證是否接近實際攔截的情況。

本論文第二章介紹運動方程式，第三章推導飛彈及目標的速度及位置函數，第四章建構最佳化問題，第五章介紹最佳化演算法，第六章為數值計算及模擬，第七章為結論。

第二章

運動方程式

飛彈在空間中的運動可分為平移運動與轉動，各為三個自由度的運動，因此在整個三維空間中視飛彈為六個自由度的運動。而將目標假設為點質量，不考慮其姿態變化，故只考慮目標在三維空間中的平移運動。本章先介紹所需要的座標系統，再接著介紹飛彈六自由度運動方程式及目標運動方程式。

2.1 座標系統

為了方便描述攔截飛彈與目標在三維空間飛行過程中的動態與姿態，我們需要定義以下座標系統。

地面座標系 (North-East-Down coordinate, NED)

使用地面座標是為了方便描述攔截飛彈與目標相對於地表的軌跡。地面座標是以地表的雷達站當作座標原點，其 X_N 軸指向地表平面雷達站的北方，其 Y_N 軸指向地表平面雷達站的東方，其 Z_N 軸滿足右手定則，指向地下，如圖 2.1 所示。本論文以地面座標系當作慣性座標。

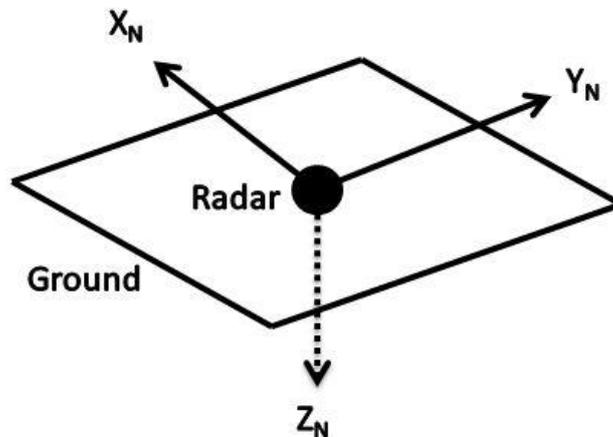


圖 2.1：NED 座標示意圖

2.2 飛彈六自由度運動方程式

2.2.1 平移運動方程式

平移運動用來描述飛彈質心在慣性座標中的位置變化，平移的運動方程式為

$$\begin{aligned}\dot{v}_{mx} &= a_{mx} \\ \dot{v}_{my} &= a_{my} \\ \dot{v}_{mz} &= a_{mz}\end{aligned}\quad (2.1)$$

$$\begin{aligned}\dot{r}_{mx} &= v_{mx} \\ \dot{r}_{my} &= v_{my} \\ \dot{r}_{mz} &= v_{mz}\end{aligned}\quad (2.2)$$

其中 (a_{mx}, a_{my}, a_{mz}) 、 (v_{mx}, v_{my}, v_{mz}) 、 (r_{mx}, r_{my}, r_{mz}) 分別為加速度、速度和位置在慣性座標中的三個分量。簡單來說，位置的微分為速度，速度的微分為加速度。

飛彈加速度包含飛彈的引擎推力造成的加速度、氣動力造成的加速度以及重力加速度

$$\begin{bmatrix} a_{mx} \\ a_{my} \\ a_{mz} \end{bmatrix} = \frac{1}{mass} T_{NB} \begin{bmatrix} F_{Tx} + F_{Ax} \\ F_{Ty} + F_{Ay} \\ F_{Tz} + F_{Az} \end{bmatrix} + \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}\quad (2.3)$$

其中 $mass$ 為飛彈質量， T_{NB} 為體座標轉換到慣性座標的轉換矩陣(如 2.2.3 節所討論)， (F_{Tx}, F_{Ty}, F_{Tz}) 、 (F_{Ax}, F_{Ay}, F_{Az}) 分別為引擎推力和氣動力在體座標中的三個分量， (g_x, g_y, g_z) 為重力加速度在慣性座標中的三個分量。

2.2.2 轉動方程式

旋轉運動在體座標中表示，轉動運動方程式可表示如下

$$\begin{aligned}\dot{p} &= \frac{M_x}{I_x} - \frac{qr}{I_x}(I_z - I_y) \\ \dot{q} &= \frac{M_y}{I_y} - \frac{rp}{I_y}(I_x - I_z) \\ \dot{r} &= \frac{M_z}{I_z} - \frac{pq}{I_z}(I_y - I_x)\end{aligned}\quad (2.4)$$

其中 (M_x, M_y, M_z) 、 (I_x, I_y, I_z) 、 (p, q, r) 分別為力矩、轉動慣量、角速度在體座標中的三個分量。

力矩包含了引擎推力造成的力矩和氣動力造成的力矩

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} M_{Tx} + M_{Ax} \\ M_{Ty} + M_{Ay} \\ M_{Tz} + M_{Az} \end{bmatrix}\quad (2.5)$$

其中 (M_{Tx}, M_{Ty}, M_{Tz}) 為引擎力矩在體座標中的三個分量， (M_{Ax}, M_{Ay}, M_{Az}) 為氣動力矩在體座標中的三個分量。

2.2.3 尤拉角(Euler angle)

慣性座標與體座標之間的關係可以透過一組尤拉角 (ϕ, θ, ψ) 來表示，通常用來描述一剛體的方位。如圖 2.2 中，首先將慣性座標與體座標的原點重合， X_B 為體座標的 X 軸， z'_B 為體座標 Z_B 軸在 Y_N-Z_N 平面上的投影， ψ 、 θ 、 ϕ 分別為三個有順序的旋轉角度。

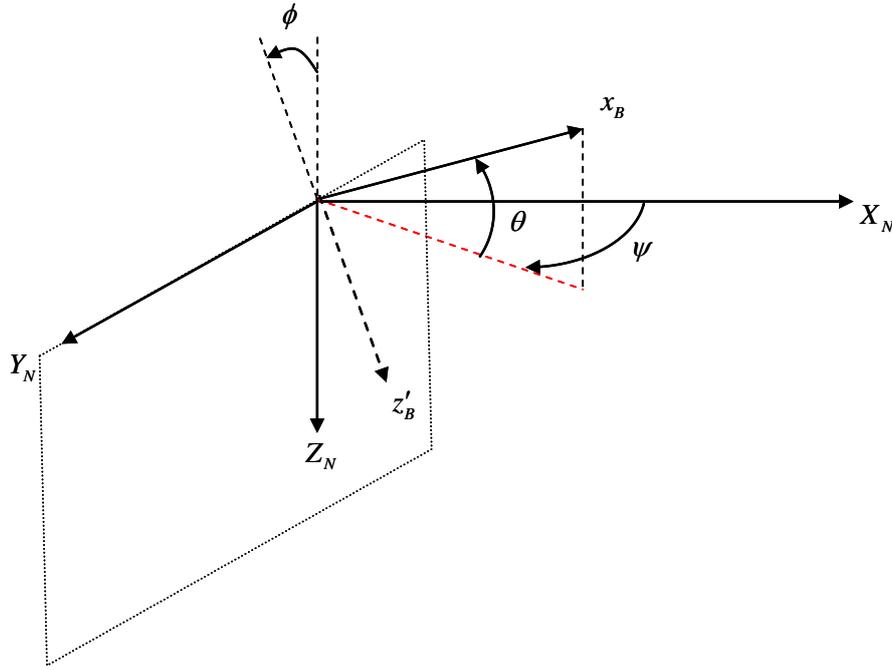


圖 2.2：尤拉角示意圖

在此將定義由慣性座標轉換至體座標的旋轉順序整理成如表 2.1 所示：

順序 1	以 Z_N 軸為轉軸並旋轉 ψ 角
順序 2	以此時的 Y_N 為轉軸並旋轉 θ 角
順序 3	以此時的 x_B 為轉軸並旋轉 ϕ 角

表 2.1：慣性座標轉換至體座標的旋轉順序

經過表 2.1 有順序的三個旋轉後，即可將慣性座標旋轉到體座標的方向，因此兩者之間的轉換矩陣即為三個旋轉矩陣有順序的相乘而得到。從慣性座標轉換到體座標的轉換矩陣為

$$\begin{aligned}
 T_{BN} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \cos \psi \sin \phi \sin \theta - \cos \phi \sin \psi & \sin \psi \sin \phi \sin \theta + \cos \phi \cos \psi & \sin \phi \cos \theta \\ \cos \psi \cos \phi \sin \theta + \sin \phi \sin \psi & \sin \psi \cos \phi \sin \theta - \sin \phi \cos \psi & \cos \phi \cos \theta \end{bmatrix} \quad (2.6)
 \end{aligned}$$

若是要計算從體座標轉換到慣性座標的轉換矩陣則為

$$T_{NB} = (T_{BN})^T \quad (2.7)$$

其中 $(T_{BN})^T$ 為 T_{BN} 的轉置矩陣。

2.2.4 尤拉方程式(Euler equation)

我們假設 (p, q, r) 分別為飛彈在滾轉、俯仰及偏航方向的角速度，再搭配前一節所定義的尤拉角 (ϕ, θ, ψ) ，透過簡單的推導可以得到尤拉角和角速度之間的一階微分方程式，稱為尤拉方程式[10]：

$$\begin{aligned} \dot{\phi} &= p - \tan \theta (q \sin \phi + r \cos \phi) \\ \dot{\theta} &= q \cos \phi - r \sin \phi \\ \dot{\psi} &= \sec \theta (q \sin \phi + r \cos \phi) \end{aligned} \quad (2.8)$$

2.3 目標運動方程式

假設目標為點質量，不考慮其姿態變化，故只考慮目標在三維空間中的平移運動，其運動方程式在慣性座標下可表示為

$$\begin{aligned} \dot{v}_{tx} &= a_{tx} \\ \dot{v}_{ty} &= a_{ty} \\ \dot{v}_{tz} &= a_{tz} \end{aligned} \quad (2.9)$$

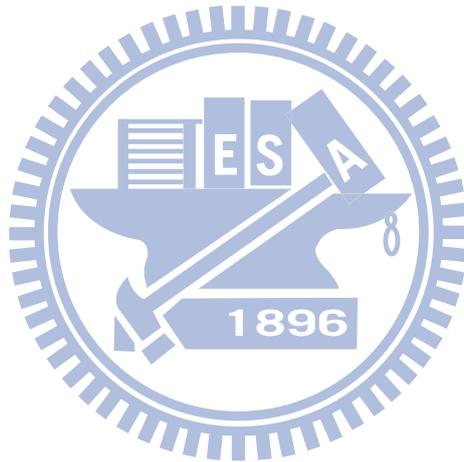
$$\begin{aligned} \dot{r}_{tx} &= v_{tx} \\ \dot{r}_{ty} &= v_{ty} \\ \dot{r}_{tz} &= v_{tz} \end{aligned} \quad (2.10)$$

其中 (a_{tx}, a_{ty}, a_{tz}) 、 (v_{tx}, v_{ty}, v_{tz}) 、 (r_{tx}, r_{ty}, r_{tz}) 分別為加速度、速度和位置在慣性座標中的三個的分量。

考慮目標加速度只受到重力和空氣阻力的影響，在慣性座標下可表示為

$$\begin{bmatrix} a_{tx} \\ a_{ty} \\ a_{tz} \end{bmatrix} = \begin{bmatrix} A_{Dx} \\ A_{Dy} \\ A_{Dz} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g_{tz} \end{bmatrix} \quad (2.11)$$

其中 (A_{Dx}, A_{Dy}, A_{Dz}) 為空氣阻力所產生的加速度在慣性座標下的三個分量， g_{tz} 為目標重力加速度在慣性座標下的Z分量。因空氣阻力所產生之加速度與目標的高度，即空氣密度，及馬赫數，即速度，有關。在數值模擬的例子中，我們採用參考資料[8]有建立的模式。



第三章

飛彈及目標的速度及位置函數

在本章先說明飛彈發射前各階段時間的定義，接著介紹推導飛彈及目標的速度及位置函數所需要的參數及變數，之後再推導出飛彈及目標的速度及位置函數。飛彈以固定的仰角及方向角發射，第一節燃畢前的位置及速度可由六自由度模擬準確地得到，第二節燃燒期間的加速度以簡單的時間一階函數近似，假設飛彈在其餘的時間的動態僅受重力的影響，則飛彈在終端航程的位置可方便地表示成時間的二次函數，方便最佳化計算。模擬的結果顯示，簡化的(無導引)飛彈軌跡與六自由度模擬中途導引的軌跡差異不是很大。

3.1 各階段時間的定義

我們假設飛彈發射的時間為時間 0 秒，即時間座標原點，而雷達偵測到目標的時間為 $-t_d$ 秒。定義以下六個相關的時間參數，各時間參數的相對關係如圖 3.1 所示。時間的單位是秒(s)。

t_d ：雷達偵測到目標至飛彈發射的延遲時間。

t_1 ：第一節火箭的燃燒時間。

t_2 ：第二節火箭的燃燒時間。

t_c ：滑行時間。

t_h ：第二節火箭脫離到第三節姿態控制開始的時間。

t_f ：預估攔截飛彈擊中目標的時間。

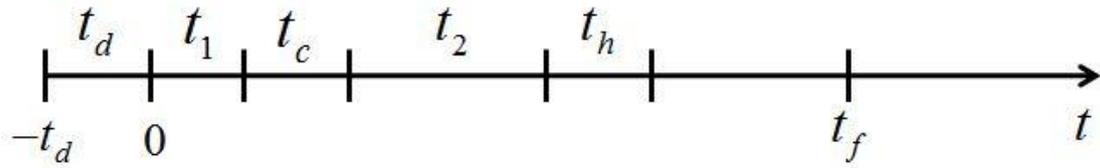


圖 3.1：各時間參數的相對關係

3.2 變數的定義

以下定義描述目標與飛彈運動的變數，距離的單位是公里(km)，時間的單位是秒(s)，質量的單位是公斤(kg)，力量的單位是牛頓(N)。

r_{mx0} 、 r_{my0} 、 r_{mz0} ：攔截飛彈初始位置，分別為慣性座標中的三個分量。

r_{mx} 、 r_{my} 、 r_{mz} ：攔截飛彈所在位置，分別為慣性座標中的三個分量。

r_{tx} 、 r_{ty} 、 r_{tz} ：目標所在位置，分別為慣性座標中的三個分量。

ΔV_1 ：第一節火箭燃燒完畢後的速度，為慣性座標中的三維向量。

ΔR_1 ：第一節火箭燃燒完畢後的位置，為慣性座標中的三維向量。

v_{t0} ：目標初始速度，為慣性座標中的三維向量。

r_{t0} ：目標初始位置，為慣性座標中的三維向量。

g ：重力加速度，為慣性座標中的三維向量。

T_2 ：第二節火箭燃燒時的最大推力。

m_{20} ：第一節火箭脫離後的質量。

\dot{m}_2 ：第二節火箭的質量燃燒速率。

3.3 目標的速度及位置函數

假設目標的初始位置為 r_{i0} ，初始速度為 v_{i0} ，只考慮重力的影響，則在任意時間 $t \geq -t_d$ ，目標速度為

$$v_i(t) = v_{i0} + g \cdot (t + t_d) \quad (3.1)$$

目標位置為

$$r_i(t) = r_{i0} + v_{i0} \cdot (t + t_d) + \frac{1}{2} g \cdot (t + t_d)^2 \quad (3.2)$$

其中 g 為重力加速度向量， $g = (0, 0, 0.0098) \text{ km/s}^2$ 。由(3.2)式可以看出目標的位置為 t 及 t_d 的二次函數。由於目標的飛行在攔截的過程中均在 30 公里以上的高空，氣動力的影響不大，(3.2)式的模式與實際三自由度模擬的結果差異不大。

3.4 飛彈的速度及位置函數

地面雷達站位於慣性座標原點 $(0, 0, 0)$ 。假設飛彈發射點的位置為 $(r_{mx0}, r_{my0}, r_{mz0})$ ，則高度為 $-r_{mz0}$ 公里。以下描述飛彈在每一個飛行階段的位置及速度函數。

令第一節火箭燃畢，即時間 $t = t_1$ ，飛彈的位置與速度分別為 ΔR_1 與 ΔV_1 。由於飛彈的推力物性及氣動模式固定， ΔR_1 及 ΔV_1 只和飛彈發射的角度有關。在不同的發射角度下，這兩個向量可利用六自由度模擬很準確且很有效率的取得。

當時間 $t = t_1 + t_c$ ，即當滑行結束第二節火箭點火前，飛彈的位置與速度分別為

$$r_m(t_1 + t_c) = \Delta R_1 + \Delta V_1 t_c + \frac{1}{2} g t_c^2 \quad (3.3)$$

$$v_m(t_1+t_c) = \Delta V_1 + gt_c \quad (3.4)$$

(3.3)式及(3.4)式的計算只考慮重力忽略了氣動力的影響因此有些誤差。如果 t_c 為固定值，則 $r_m(t_1+t_c)$ 及 $v_m(t_1+t_c)$ 可以利用模擬比較準確的取得。大致來說，(3.3)式及(3.4)式所計算的數值會略高於模擬所得到的數值。

第二節火箭點火之後至燃畢前，即 $t_1+t_c < t \leq t_1+t_c+t_2$ ，推力所產生的加速度因質量遞減而遞增。當中途導引律啟動時，速度的方向隨攔截狀況而改變，因此加速度的方向並不固定。為了分析方便，也因為實際的加速度方向無從得知，假設加速度的方向固定並且可近似為線性函數

$$a_m(t) \approx a + (t-t_1-t_c) \cdot b \quad (3.5)$$

其中 a 與 b 為常數向量，其決定在 3.5 節介紹。因此在第二節點火時間內飛彈的位置與速度分別為

$$r_m(t) = \Delta R_1 + \Delta V_1(t-t_1) + \frac{1}{2}g(t-t_1)^2 + \frac{1}{2}(t-t_1-t_c)^2 \cdot a + \frac{1}{6}(t-t_1-t_c)^3 \cdot b \quad (3.6)$$

$$v_m(t) = \Delta V_1 + g(t-t_1) + (t-t_1-t_c) \cdot a + \frac{1}{2}(t-t_1-t_c)^2 \cdot b \quad (3.7)$$

將 $t=t_1+t_c+t_2$ 代入(3.6)式及(3.7)式，可得時間 $t=t_1+t_c+t_2$ 時飛彈的位置與速度分別為

$$r_{m2f} = \Delta R_1 + \Delta V_1 \cdot (t_c+t_2) + \frac{1}{2}g(t_c+t_2)^2 + \frac{1}{2}t_2^2 \cdot a + \frac{1}{6}t_2^3 \cdot b \quad (3.8)$$

$$v_{m2f} = \Delta V_1 + g(t_c+t_2) + t_2 \cdot a + \frac{1}{2}t_2^2 \cdot b \quad (3.9)$$

(3.8)式與(3.9)式分別為飛彈在第二節火箭燃畢時的位置與速度。

當 $t > t_1 + t_c + t_2$ ，即截殺彈頭終端飛行階段，其位置與速度分別為

$$r_m(t) = r_{m2f} + v_{m2f} \cdot (t - t_1 - t_c - t_2) + \frac{1}{2} g(t - t_1 - t_c - t_2)^2 \quad (3.10)$$

$$v_m(t) = v_{m2f} + g(t - t_1 - t_c - t_2) \quad (3.11)$$

其中 r_{m2f} 及 v_{m2f} 分別定義於(3.8)式及(3.9)式。

3.5 第二節火箭加速度分析

在真正的攔截飛行中，飛彈中途階段的加速度方向由中途導引律提供，但是在射控階段，則必須對加速度做一些簡單的假設以利攔截時間的分析。我們假設飛彈在滑行完畢中途飛行時，飛行路徑角維持不變且攻角為零。因此在這個階段，即 $t_1 + t_c \leq t \leq t_1 + t_c + t_2$ ，飛彈的加速度為

$$a_m(t) = \frac{T_2}{m_{20} - (t - t_1 - t_c) \cdot \dot{m}_2} \cdot L_{v1} \quad (3.12)$$

$$L_{v1} = \frac{\Delta V_1 + g t_c}{\|\Delta V_1 + g t_c\|} \quad (3.13)$$

其中 T_2 為第二節燃燒時的最大推力， m_{20} 為第一節脫離後的質量，常數 \dot{m}_2 為第二節火箭的燃燒速率， L_{v1} 為第一節滑行時間結束後速度的單位向量。

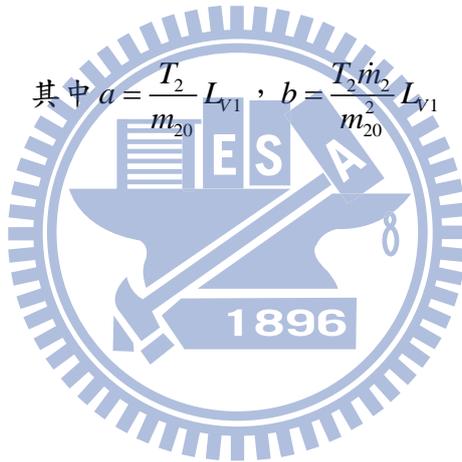
在滑行時間 t_c 固定的狀況下，在(3.13)式中， L_{v1} 為一常數向量，但是 $a_m(t)$ 並非時間 t 的線性函數；為了計算方便，可將(3.12)式右邊作線性化得

$$\frac{T_2}{m_{20} - (t - t_1 - t_c) \cdot \dot{m}_2} \cdot L_{v1}$$

$$\begin{aligned}
&= \frac{T_2 L_{V1}}{m_{20}} \cdot \frac{1}{1 - (t - t_1 - t_c) \frac{\dot{m}_2}{m_{20}}} \\
&\approx \frac{T_2 L_{V1}}{m_{20}} \left[1 + (t - t_1 - t_c) \frac{\dot{m}_2}{m_{20}} \right] \\
&= \frac{T_2 L_{V1}}{m_{20}} + (t - t_1 - t_c) \cdot \frac{T_2 \dot{m}_2 L_{V1}}{m_{20}^2}
\end{aligned} \tag{3.14}$$

因此(3.5)式可表示成時間的線性函數為

$$a_m(t) \approx a + (t - t_1 - t_c) \cdot b$$



第四章

最佳化問題的建構

在本章首先說明飛彈能夠攔截成功的必要條件，隨後介紹如何建立一個有效的非線性最佳化問題來求出發射前的延遲時間及飛彈飛行時間，並透過 Matlab 程式撰寫求出的延遲時間的最大值與最小值，根據此上下限將可以建立發射時間窗口。最後透過表 4.1 可以初步了解延遲時間的選擇對攔截是否成功的影響。

4.1 攔截的必要條件

一個成功的攔截必須經過準確的中途導引及終端導引階段，而在發射之初無法預知中途及終端階段的飛彈加速度。因此如果要探討攔截是否可能，數值上必須對飛彈的加速度做一些簡化的假設以利軌跡及速度函數的建立。此即第三章所建立之結果。

一個成功的攔截的充分條件是在某一個時間 $t_f > 0$ ，飛彈及目標在同一位置，即 $r_m(t_f) = r_t(t_f)$ 。如上所提，這個條件必須經過準確的中途及終端導引才有可能成立。因此探討攔截的可能性我們考慮其必要條件。其中一個最明顯的必要條件是

$$\|r_m(t_f)\| = \|r_t(t_f)\| \quad (4.1)$$

其中 $\|\cdot\|$ 代表向量的長度。如果(4.1)式不滿足，攔截一定無法達成；如果(4.1)式滿足，則加上適當的導引律攔截有達成的可能。如圖 4.1 所示，當(4.1)式成立表示當時間 t_f 時，飛彈與目標同時位於以圓心為座標原點的某個圓球上。

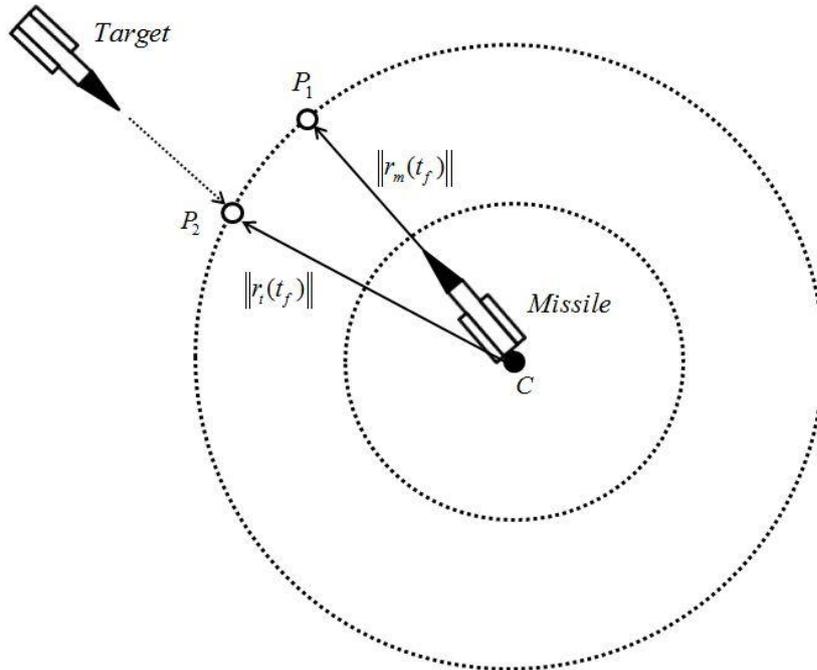


圖 4.1：目標與飛彈對座標原點的距離示意圖

除了(4.1)式的條件之外，成功攔截的必要條件還包括

$$-r_{tz}(t_f) \geq h \quad (4.2)$$

$$t_f \geq t_1 + t_c + t_2 + t_h \quad (4.3)$$

$$t_c \geq t_{cmin} \quad (4.4)$$

$$t_d \geq 0 \quad (4.5)$$

$$-r_{mz}(t_f) \geq h \quad (4.6)$$

$$-r_{tz}(t) + r_{mz}(t) \leq h_r \quad (4.7)$$

上列所述六個必要條件的意義分別說明如下：

(4.2)式代表攔截的目標高度必須高於 h 公里，此為確保截殺彈頭在空氣足夠稀薄的高空運作簡化姿態控制；(4.3)式表示攔截時飛彈必須是在終端導引階段，其中 t_h 為第二節火箭脫離至目標鎖定所需的最短時間，為一個定值；(4.4)式表示滑行

時間必須大於一個固定的最小值，以確保第一節火箭有足夠的時間可以安全脫離；(4.5)式要求飛彈的發射必須在雷達偵測到目標之後；(4.6)式代表攔截的飛彈高度必須高於 h 公里；(4.7)式代表目標與飛彈的高度差必須小於 h_r 公里。

在此將考慮比較簡單的情況下即 t_c 為一固定值，所以飛彈的位置為時間 t 的函數而目標的位置為時間 t 及延遲時間 t_d 的函數，分別如(3.10)及(3.2)式所列。令變數 $x_1 = t_d$ ， $x_2 = t$ ，且定義函數

$$h(x_1, x_2) = \|r_t(t)\|^2 - \|r_m(t)\|^2$$

$$g_1(x_1, x_2) = r_{tz}(t) + h$$

$$g_2(x_1, x_2) = t_1 + t_c + t_2 + t_h - x_2$$

$$g_3(x_1, x_2) = -x_1$$

$$g_4(x_1, x_2) = r_{mz}(t) + h$$

$$g_5(x_1, x_2) = -r_{tz}(t) + r_{mz}(t) - h_r$$

則(4.1)、(4.2)、(4.3)、(4.5)、(4.6)及(4.7)的必要條件可分別表示為

$$h(x_1, x_2) = 0 \quad (4.8)$$

$$g_1(x_1, x_2) \leq 0 \quad (4.9)$$

$$g_2(x_1, x_2) \leq 0 \quad (4.10)$$

$$g_3(x_1, x_2) \leq 0 \quad (4.11)$$

$$g_4(x_1, x_2) \leq 0 \quad (4.12)$$

$$g_5(x_1, x_2) \leq 0 \quad (4.13)$$

其中(4.8)為等式限制式，(4.9)至(4.13)為不等式限制式。

4.2 最佳化問題的建立

令函數

$$f(x_1, x_2) = x_1 + x_2 \quad (4.14)$$

即 $f(x_1, x_2) = t_d + t$ ，為雷達偵測到目標至飛彈發射後 t 秒的總時間。我們考慮的最佳化問題之一為

(P1)

$$\max_{x_1, x_2} f(x_1, x_2)$$

$$\text{subject to } \begin{cases} h(x_1, x_2) = 0 \\ g_1(x_1, x_2) \leq 0 \\ g_3(x_1, x_2) \leq 0 \\ g_4(x_1, x_2) \leq 0 \end{cases}$$

透過此最佳化問題可以求出從雷達偵測到目標至攔截時間之最大值與其所對應的延遲時間 t_d ，飛彈飛行時間 t_f ，及預估的攔截點 $r_i(t_f)$ 。所求得的 x_1 及 x_2 ，仍必須代入檢查(4.10)限制式是否滿足，即 $g_2(x_1, x_2) \leq 0$ ，如果(4.10)式不滿足即代表飛彈尚未進入終端階段不可能攔截成功，飛彈不應發射。這樣的狀況通常發生於目標被偵測時已經太接近雷達站(或飛彈的發射點)。如果(4.10)式滿足則將所求得的延遲時間 t_d 視為容許最大的延遲時間。

我們考慮的另一個最佳化問題是

(P2)

$$\begin{aligned} & \min_{x_1, x_2} f(x_1, x_2) \\ & \text{subject to } \begin{cases} h(x_1, x_2) = 0 \\ g_1(x_1, x_2) \leq 0 \\ g_2(x_1, x_2) \leq 0 \\ g_3(x_1, x_2) \leq 0 \\ g_4(x_1, x_2) \leq 0 \\ g_5(x_1, x_2) \leq 0 \end{cases} \end{aligned}$$

透過此最佳化問題可以求出從雷達站偵測到目標至攔截時間之最小值與其
所對應的延遲時間 t_d ，飛彈飛行時間 t_f ，及預估的攔截點 $r_i(t_f)$ 。與問題(P1)不同
的是，(P2)求取 $f(x_1, x_2)$ 的最小值且限制式包括(4.10)式。只有在(P1)有解的情況
下才考慮(P2)問題，並將所求得的延遲時間 t_d 視為最短的延遲時間。

4.3 延遲時間最大值與最小值

如果最佳化問題(P1)及(P2)均有可行解，且定義其分別算得的延遲時間為
 $t_{d \max}$ 及 $t_{d \min}$ ，且 $0 \leq t_{d \min} < t_{d \max}$ ，則時間區間 $[t_{d \min}, t_{d \max}]$ 即為延遲時間的窗口，將
可提供攔截飛彈系統參考。大致上來說，雷達偵測到目標之後，飛彈至少需等待
 $t_{d \min}$ 秒之後才能發射且不能等待超過 $t_{d \max}$ 秒，以確保攔截有可能成功。我們可以
透過表 4.1 初步了解上述關於延遲時間的說明，在此假設 $t_{d \min} > 0$ 。

狀況	發射前的延遲時間 t_d	攔截結果
Case 1	$t_d < t_{d\min}$	失敗
Case 2	$t_{d\min} \leq t_d \leq t_{d\max}$	有可能成功
Case 3	$t_d > t_{d\max}$	失敗

表 4.1：選擇不同延遲時間對攔截的影響



第五章

最佳化演算法的發展

根據第四章所討論的最佳化問題可以用下列標準非線性規劃的問題表示：

$$\begin{aligned} & \text{Minimize } f(\mathbf{x}) \\ & \text{subject to } \begin{cases} h_j(\mathbf{x}) = 0, j = 1, \dots, n \\ g_i(\mathbf{x}) \leq 0, i = 1, \dots, m \end{cases} \end{aligned} \quad (5.1)$$

其中 $\mathbf{x} \in \mathcal{R}^l$ ， $g_i(\mathbf{x}) \leq 0$ 為不等式限制式， $h_j(\mathbf{x}) = 0$ 為等式限制式。一般來說，除了非常特殊且簡單的情況，(5.1)的標準問題並沒有解析解，而必須使用遞迴的方式求解。因此在本章先後介紹兩種演算法來計算問題(5.1)的解，第一種方法是處罰函數(Penalty Function, PF)的演算法。利用所謂的處罰函數是嘗試將含有等式及不等式的最佳化問題轉換成對等的沒有限制式(unconstrained)的問題，以方便一般求最小值方法的使用，例如梯度法(gradient method)。第二種方法是所謂的連續二次規劃(Sequential Quadratic Programming, SQP)的演算法。並設計一個簡單的非線性最佳化問題來測試所提出的兩種演算法是否可行，最後再比較兩種方法的差異及可行性。

5.1 處罰函數(PF)演算法

考慮問題(5.1)定義處罰函數

$$P(\mathbf{x}) = \sum_{i=1}^m (\max(0, g_i(\mathbf{x})))^2 + \sum_{j=1}^n |h_j(\mathbf{x})|^2 \quad (5.2)$$

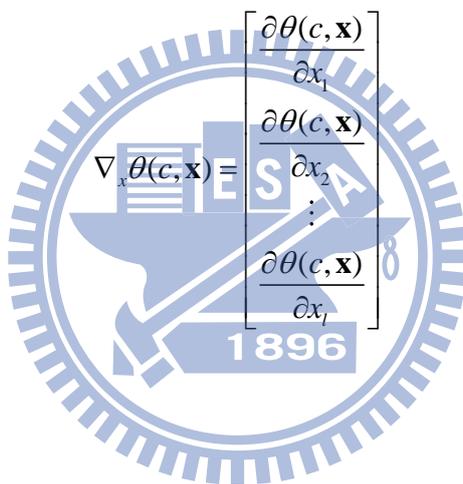
及新的目標函數

$$\theta(c, \mathbf{x}) = f(\mathbf{x}) + cP(\mathbf{x}) \quad (5.3)$$

其中 c 為正數。PF 方法的主要精神是如果(5.1)的限制式不滿足，則(5.2)的 $P(\mathbf{x})$ 大於零，如果參數 c 很大，則將(5.3)式最小化可確定 $P(\mathbf{x})$ 很小，當 $c \rightarrow \infty$ 時，則 $P(\mathbf{x}) \rightarrow 0$ ，即限制式均滿足。據此將(5.1)的問題轉換成下面沒有限制式的問題

$$\text{Minimize } \{ \theta(c, \mathbf{x}), \mathbf{x} \in \mathcal{R}^l \} \quad (5.4)$$

在此利用 Gradient Search method，求出 x 使得 $\theta(c, \mathbf{x})$ 最小。數值求解(5.4)需要選擇三個參數分別為成長函數 $\eta > 1$ ，停止參數 $\varepsilon > 0$ ，及初始處罰參數 $c_0 > 0$ 。定義梯度向量為



$$\nabla_{\mathbf{x}} \theta(c, \mathbf{x}) = \begin{bmatrix} \frac{\partial \theta(c, \mathbf{x})}{\partial x_1} \\ \frac{\partial \theta(c, \mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial \theta(c, \mathbf{x})}{\partial x_l} \end{bmatrix}$$

詳細求解步驟如下：

步驟 1：選擇 η ， ε ， c_0 ，及初始值 x^0

步驟 2：迴圈求解， $k=1,2,3,\dots$

由 \mathbf{x}^{k-1} 開始尋找 \mathbf{x}^k 使得 $\theta(c_k, \mathbf{x}^k)$ 最小，並記錄下每個 \mathbf{x}^k 所對應的 $\theta(c_k, \mathbf{x}^k)$

步驟 3：計算 $\nabla_{\mathbf{x}} \theta(c_{k-1}, \mathbf{x}^{k-1})$ 。

步驟 4： $\mathbf{x}^k = \mathbf{x}^{k-1} - \alpha^k \cdot \nabla_{\mathbf{x}} \theta(c_{k-1}, \mathbf{x}^{k-1})$

其中 α^k 為搜尋距離， α^k 可利用 $\frac{d\theta(c_{k-1}, \mathbf{x}^k)}{d\alpha} = 0$ 求得。

以下過程為判斷是否達到收斂條件：

If ($\|\mathbf{x}^k - \mathbf{x}^{k-1}\| < \varepsilon$ or $\|f(\mathbf{x}^k) - f(\mathbf{x}^{k-1})\| < \varepsilon$)

跳出迴圈， \mathbf{x}^k 即為預估最小值的最佳解；

Else $c_k = \eta \cdot c_{k-1}$ ，返回步驟 2，重新產生 $\theta(c_k, \mathbf{x}^k)$ ；

End

=====

5.2 PF 範例

經過 5.1 節詳細介紹 PF 演算法後，接下來要透過 Matlab 程式撰寫求解一個非線性的最佳化問題，並用圖形來驗證所求出的最小值與最大值是否接近實際的最小值與最大值。

5.2.1 PF 範例最小值問題



$$\begin{aligned} & \text{Minimize } f(\mathbf{x}) = x_1 + x_2 \\ & \text{subject to } \begin{cases} h(\mathbf{x}) = 3x_1^2 + 2x_2^2 - 2 = 0 \\ g(\mathbf{x}) = 2x_1^2 + 3x_2 + 2 \leq 0 \end{cases} \end{aligned}$$

考慮本題之最小值問題，選擇 $\eta = 1.5$ ， $\varepsilon = 0.01$ ， $c_0 = 0.05$ ，初始值設定為 $(x_1, x_2) = (-1, -2)$ ，經 Matlab 程式執行共 11 次後收斂，當 $(x_1, x_2) = (-0.4865, -0.8200)$ ，目標函數有最小值 $f(x) = -1.3065$ 。表 5.1 為 PF 求解結果的相關數據。

k	x_1	x_2	c_k	$f(x_1, x_2)$
0	-1.0000	-2.0000	0.05	-3.0000
1	-0.5873	-1.2205	0.05	-1.8078

2	-0.6492	-1.0172	0.075	-1.6664
3	-0.6426	-0.9662	0.1125	-1.6088
4	-0.6088	-0.9198	0.1688	-1.5286
5	-0.5776	-0.8845	0.2531	-1.4621
6	-0.5519	-0.8600	0.3797	-1.4119
7	-0.5312	-0.8436	0.5695	-1.3748
8	-0.5149	-0.8330	0.8543	-1.3479
9	-0.5024	-0.8264	1.2814	-1.3288
10	-0.4931	-0.8224	1.9222	-1.3155
11	-0.4865	-0.8200	2.8833	-1.3065

表 5.1：PF 求解範例最小值相關數據

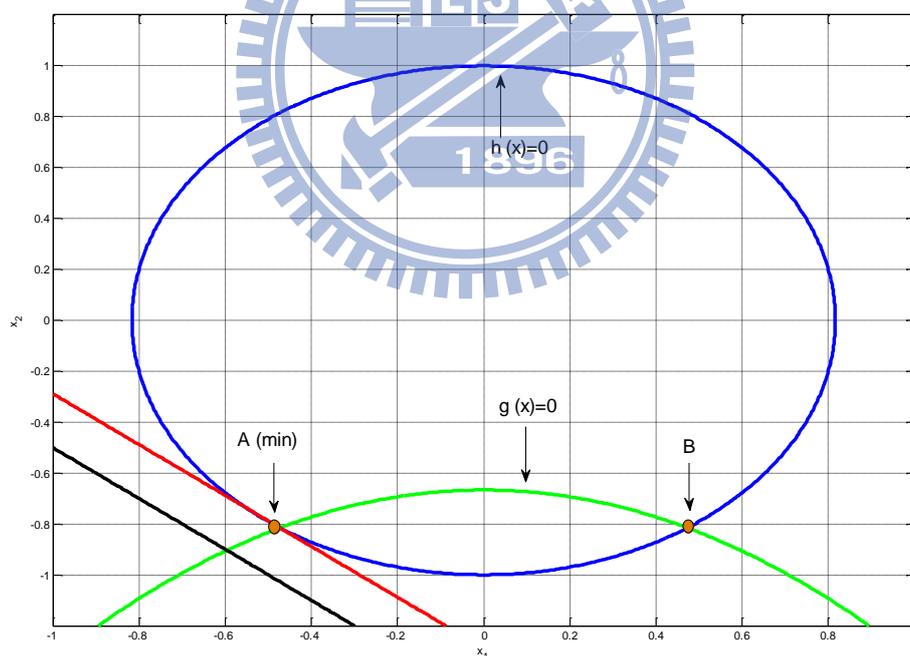


圖 5.1：PF 範例最小值驗證

我們可以由圖 5.1 來檢視利用 PF 作法是否接近實際的最小值。其中綠色線段代表限制式 $g(\mathbf{x})=0$ ，綠色線段下方代表 $g(\mathbf{x})\leq 0$ 之解集合，藍色線段代表限

制式 $h(\mathbf{x})=0$ ，因此滿足等式及不等式限制式之可行解區域為劣弧 AB。利用 PF 演算法所求出的可行解為 A 點，即 $(x_1, x_2) = (-0.4865, -0.8200)$ 。紅色線段代表當目標函數最小值為 -1.3065，剛好通過 A 點，而黑色線段則代表目標函數值為 -1.5000 時，若求出的可行解在黑色線段上，則不同時滿足 $g(\mathbf{x}) \leq 0$ 與 $h(\mathbf{x}) = 0$ 。因此利用 PF 演算法可以有效求得 A 點已經相當接近實際的最小值。

5.2.2 PF 範例最大值問題

$$\begin{aligned} & \text{Maximize } f(\mathbf{x}) = x_1 + x_2 \\ & \text{subject to } \begin{cases} h(\mathbf{x}) = 3x_1^2 + 2x_2^2 - 2 = 0 \\ g(\mathbf{x}) = 2x_1^2 + 3x_2 + 2 \leq 0 \end{cases} \end{aligned}$$

考慮本題之最大值問題，則選擇 $\eta = 1.5$ ， $\varepsilon = 0.01$ ， $c_0 = 0.05$ ，初始值設定為 $(x_1, x_2) = (1, 1)$ ，經 Matlab 程式執行共 11 次後收斂，當 $(x_1, x_2) = (0.4913, -0.8042)$ ，目標函數有最大值 $f(x) = -0.3129$ 。表 5.2 為 PF 求解結果的相關數據。

k	x_1	x_2	c_k	$f(x_1, x_2)$
0	1.0000	1.0000	0.05	2.0000
1	0.5025	0.6822	0.05	1.1847
2	0.3994	0.2719	0.075	0.6713
3	0.8037	-0.2417	0.1125	0.5620
4	0.5807	-0.4299	0.1688	0.1508
5	0.6466	-0.6633	0.2531	-0.0167
6	0.5708	-0.7105	0.3797	-0.1397
7	0.5540	-0.7562	0.5695	-0.2022

8	0.5259	-0.7744	0.8543	-0.2485
9	0.5123	-0.7897	1.2814	-0.2774
10	0.4987	-0.7977	1.9222	-0.2990
11	0.4913	-0.8042	2.8833	-0.3129

表 5.2：PF 求解範例最大值相關數據

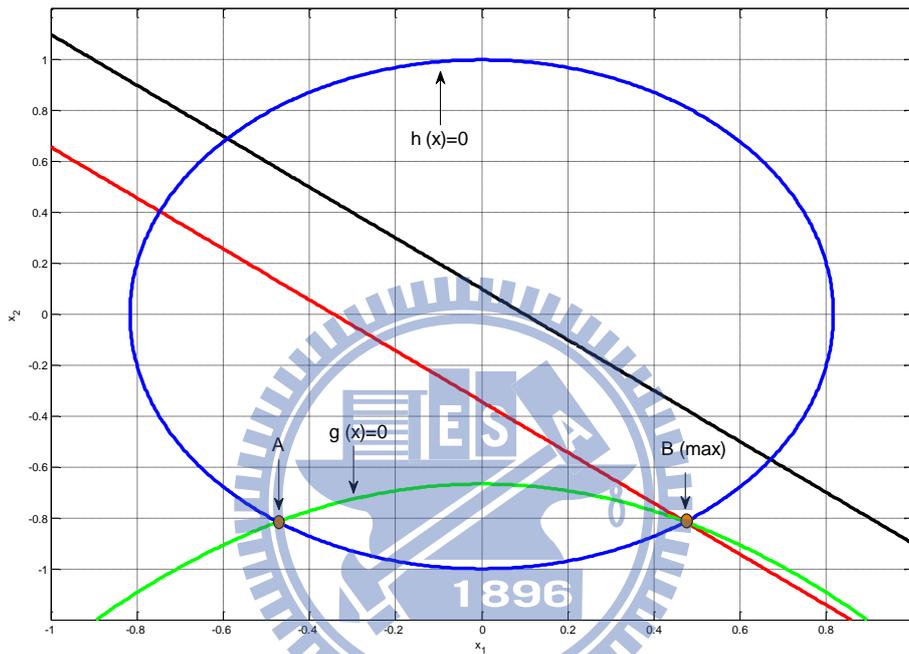


圖 5.2：PF 範例最大值驗證

同樣地，我們可以由圖 5.2 來檢視利用 PF 作法是否接近實際的最大值。其中綠色線段代表限制式 $g(\mathbf{x})=0$ ，綠色線段下方代表 $g(\mathbf{x})\leq 0$ 之解集合，藍色線段代表限制式 $h(\mathbf{x})=0$ ，因此滿足等式及不等式限制式之可行解區域為劣弧 AB。利用 PF 演算法所求出的可行解為 B 點，即 $(x_1, x_2) = (0.4913, -0.8042)$ 。紅色線段代表當目標函數最大值為 -0.3129，剛好通過 B 點，而黑色線段代表目標函數值為 0.1 時，若求出的可行解在黑色線段上，則不同時滿足 $g(\mathbf{x})\leq 0$ 與 $h(\mathbf{x})=0$ 。因此利用 PF 演算法可以有效求得 B 點已經相當接近實際的最大值。

5.3 連續二次規劃(SQP)演算法

連續二次規劃方法 SQP，是常見且最簡單的非線性規劃最佳化的求解工具。

考慮標準問題(5.1)，以下說明僅考慮限制式為等式部份。

定義等式限制式為

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_n(\mathbf{x}) \end{bmatrix}$$

使用 Lagrange 函數為

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^n \lambda_j h_j(\mathbf{x}) \quad (5.5)$$

其中 λ_j 為 Lagrange multipliers。

根據KKT(Karush-Kuhn-Tucker)必要條件為

$$\nabla f(\mathbf{x}) + \sum_{j=1}^n \lambda_j \nabla h_j(\mathbf{x}) = \mathbf{0} \quad (5.6)$$

$$h_j(\mathbf{x}) = 0 \quad (5.7)$$

即 $\begin{bmatrix} \nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) \\ \nabla_\lambda L(\mathbf{x}, \boldsymbol{\lambda}) \end{bmatrix} = \mathbf{0}$ 。針對(5.6)式及(5.7)式左邊的函數利用泰勒式在 $(\mathbf{x}^k, \boldsymbol{\lambda}^k)$ 展開可

得到

$$\begin{bmatrix} \nabla_x L(\mathbf{x}^k, \boldsymbol{\lambda}^k) \\ \nabla_\lambda L(\mathbf{x}^k, \boldsymbol{\lambda}^k) \end{bmatrix} + \begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k) & \nabla \mathbf{h}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \end{bmatrix} = \mathbf{0} \quad (5.8)$$

利用(5.8)式計算 $\delta \mathbf{x}$ 及 $\delta \boldsymbol{\lambda}$ 並令

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \delta \mathbf{x} \quad (5.9)$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \delta \boldsymbol{\lambda} \quad (5.10)$$

(5.8)式、(5.9)式、(5.10)式所列的三個方程式為主要的遞迴步驟。

其中，在(5.8)式中 Lagrange 函數的 Hessian 矩陣為

$$\nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}) = \nabla^2 f(\mathbf{x}) + \sum_{j=1}^n \lambda_j \nabla^2 h_j(\mathbf{x}) \quad (5.11)$$

將(5.8)式經過移項整理之後可以得到下列矩陣

$$\begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k) & \nabla \mathbf{h}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}^k) - \nabla \mathbf{h}(\mathbf{x}^k) \cdot \boldsymbol{\lambda}^k \\ -\mathbf{h}(\mathbf{x}^k) \end{bmatrix} \quad (5.12)$$

在此假設 $\mathbf{d}^k = \delta \mathbf{x}$ ，並將(5.10)式代入(5.12)式後可得到下列矩陣

$$\begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k) & \nabla \mathbf{h}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d}^k \\ \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}^k) - \nabla \mathbf{h}(\mathbf{x}^k) \cdot \boldsymbol{\lambda}^k \\ -\mathbf{h}(\mathbf{x}^k) \end{bmatrix}$$

$$\begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k) & \nabla \mathbf{h}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d}^k \\ \boldsymbol{\lambda}^{k+1} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}^k) - \nabla \mathbf{h}(\mathbf{x}^k) \cdot \boldsymbol{\lambda}^k + \nabla \mathbf{h}(\mathbf{x}^k) \cdot \boldsymbol{\lambda}^k \\ -\mathbf{h}(\mathbf{x}^k) \end{bmatrix}$$

$$\begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k) & \nabla \mathbf{h}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d}^k \\ \boldsymbol{\lambda}^{k+1} \end{bmatrix} = - \begin{bmatrix} \nabla f(\mathbf{x}^k) \\ \mathbf{h}(\mathbf{x}^k) \end{bmatrix} \quad (5.13)$$

上述(5.13)式稱為 Newton-Lagrange System (LNS)。

定義不等式限制式為

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ \vdots \\ g_m(\mathbf{x}) \end{bmatrix}$$

在此考慮不等式限制式 $g_i(\mathbf{x}) \leq 0$ 與等式限制式 $h_j(\mathbf{x}) = 0$ 同時存在時，則只有

當不等式限制式 $g_i(\mathbf{x}) \geq 0$ 時，才將此不等式限制式加入 LNS 中，其矩陣為

$$\begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k, \boldsymbol{\beta}^k) & \nabla \mathbf{h}(\mathbf{x}^k) & \nabla \mathbf{g}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & \mathbf{0} & \mathbf{0} \\ \nabla \mathbf{g}(\mathbf{x}^k)^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d}^k \\ \boldsymbol{\lambda}^{k+1} \\ \boldsymbol{\beta}^{k+1} \end{bmatrix} = - \begin{bmatrix} \nabla f(\mathbf{x}^k) \\ \mathbf{h}(\mathbf{x}^k) \\ \mathbf{g}(\mathbf{x}^k) \end{bmatrix} \quad (5.14)$$

其中 Lagrange 函數改變為

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_{j=1}^n \lambda_j h_j(\mathbf{x}) + \sum_{i=1}^m \beta_i g_i(\mathbf{x}) \quad (5.15)$$

定義收斂參數： $\varepsilon > 0$

詳細求解步驟如下：

=====

步驟 1：決定初始值 $(\mathbf{x}^0, \boldsymbol{\lambda}^0, \boldsymbol{\beta}^0)$

步驟 2：迴圈求解 LNS， $k=1, 2, 3, \dots$ ，可求出新的增加量 \mathbf{d}^k 、 $\boldsymbol{\lambda}^{k+1}$ 與 $\boldsymbol{\beta}^{k+1}$

步驟 3：更新 $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k$

步驟 4：檢查是否收斂，收斂條件 $\mathbf{d}^{k^T} \cdot \mathbf{d}^k \leq \varepsilon$ ，若滿足，則停止。

步驟 5：令 $k = k + 1$ ，重新回到步驟 2 直至收斂為止

=====

5.4 SQP 範例

經過 5.3 節詳細介紹 SQP 演算法後，接下來要透過 Matlab 程式撰寫求解一個非線性的最佳化問題，此範例與 5.2 節所提出的 PF 範例相同，目的是要來比較所求出的可行解是否接近，最後使用表 5.5 及表 5.6 來比較兩種演算法的差異及可行性。

5.4.1 SQP 範例最小值問題

$$\begin{aligned} & \text{Minimize } f(\mathbf{x}) = x_1 + x_2 \\ & \text{subject to } \begin{cases} h(\mathbf{x}) = 3x_1^2 + 2x_2^2 - 2 = 0 \\ g(\mathbf{x}) = 2x_1^2 + 3x_2 + 2 \leq 0 \end{cases} \end{aligned}$$

若考慮本題之最小值問題，選擇 $(\lambda^0, \beta^0) = (0.5, 0.5)$ ， $\varepsilon = 10^{-6}$ ，初始值設定為 $(x_1, x_2) = (2, -5)$ ，經 Matlab 程式執行共 7 次後收斂，當 $(x_1, x_2) = (-0.4725, -0.8155)$ ，目標函數有最小值 $f(x) = -1.2880$ 。

k	x_1	x_2	\mathbf{d}^k	λ^{k+1}	β^{k+1}	$f(x_1, x_2)$
1	0.6022	-2.8387	(-1.3978, 2.1613)	0.2661	0.5000	-2.2365
2	-0.4373	-1.8304	(-1.0394, 1.0083)	0.1826	0.5000	-2.2677
3	-0.8183	-0.9735	(-0.3810, 0.8569)	0.2221	0.5000	-1.7918
4	-0.5510	-0.8215	(0.2672, 0.1520)	0.3417	0.0651	-1.3725
5	-0.4781	-0.8155	(0.0729, 0.0059)	0.3337	0.0295	-1.2936
6	-0.4726	-0.8155	(0.0056, 9.0850e-06)	0.3333	0.0291	-1.2881
7	-0.4725	-0.8155	(3.2821e-05, 2.1267e-11)	0.3333	0.0291	-1.2880

表 5.3：SQP 求解範例最小值相關數據

5.4.2 SQP 範例最大值問題

$$\text{Maximize } f(\mathbf{x}) = x_1 + x_2$$

$$\text{subject to } \begin{cases} h(\mathbf{x}) = 3x_1^2 + 2x_2^2 - 2 = 0 \\ g(\mathbf{x}) = 2x_1^2 + 3x_2 + 2 \leq 0 \end{cases}$$

若考慮本題之最大值問題，則選擇 $(\lambda^0, \beta^0) = (0.5, 0.5)$ ， $\varepsilon = 10^{-6}$ ，初始值設定為 $(x_1, x_2) = (2, -5)$ ，經 Matlab 程式執行共 7 次後收斂，當 $(x_1, x_2) = (0.4725, -0.8155)$ ，目標函數有最大值 $f(x) = -0.3430$ 。

k	x_1	x_2	\mathbf{d}^k	λ^{k+1}	β^{k+1}	$f(x_1, x_2)$
1	1.4624	-2.3226	(-0.5376, 2.6774)	0.2177	0.5000	-0.8602
2	1.7495	-0.4148	(0.2872, 1.9078)	0.0712	0.5000	1.3347
3	0.9609	-0.8677	(-0.7886, -0.4529)	0.0195	0.3871	0.0932
4	0.5972	-0.8162	(-0.3637, 0.0515)	0.0323	0.3694	-0.2190
5	0.4855	-0.8155	(-0.1116, 6.8202e-04)	0.0631	0.4020	-0.3300
6	0.4727	-0.8155	(-0.0128, 1.1985e-07)	0.0752	0.4152	-0.3428
7	0.4725	-0.8155	(-1.7424e-04, 3.7251e-15)	0.0757	0.4156	-0.3430

表 5.4：SQP 求解範例最大值相關數據

由於本題與 5.2 節 PF 範例完全相同，因此為了檢視 SQP 與 PF 演算法的差異及可行性，可參考表 5.5 及表 5.6 分別為利用兩種方法來求解最小值及最大值的求解結果。

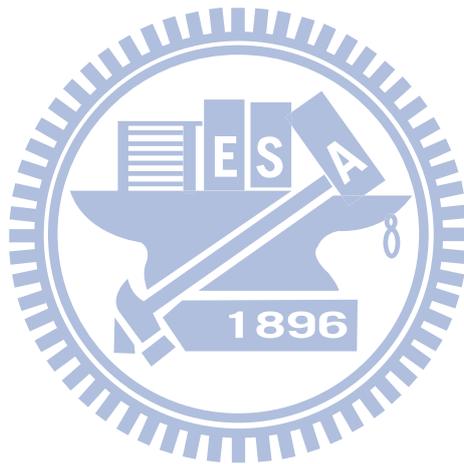
方法	執行次數	可行解 x_1	可行解 x_2	目標函數最小值
PF	11	-0.4865	-0.8200	-1.3065
SQP	7	-0.4725	-0.8155	-1.2880

表 5.5：最小值問題方法比較

方法	執行次數	可行解 x_1	可行解 x_2	目標函數最大值
PF	11	0.4913	-0.8042	-0.3129
SQP	7	0.4725	-0.8155	-0.3430

表 5.6：最大值問題方法比較

透過表 5.5 及表 5.6 中可以發現兩種方法所求得的可行解相當接近，且由圖 5.1 及圖 5.2 可得知求出的可行解均相當接近實際的最小值與最大值。經 Matlab 程式執行以 SQP 方法執行次數較少，PF 執行次數較多但是最終亦可準確求出可行解。因此若不考慮執行次數多寡，PF 與 SQP 演算法皆可作為我們所要探討最佳化問題(P1)及(P2)的求解工具。



第六章

數值計算及模擬

6.1 飛彈的推力及物性

本論文飛彈的推力設定為修改探空五號[9]的資料。圖 6.1 分別為攔截飛彈在飛行過程的質量變化與推力大小的設定。表 6.1 為比較各節火箭推力與作用時間，表 6.2 為比較各節的燃料質量。表 6.3 是飛彈各節質量的分配狀況。

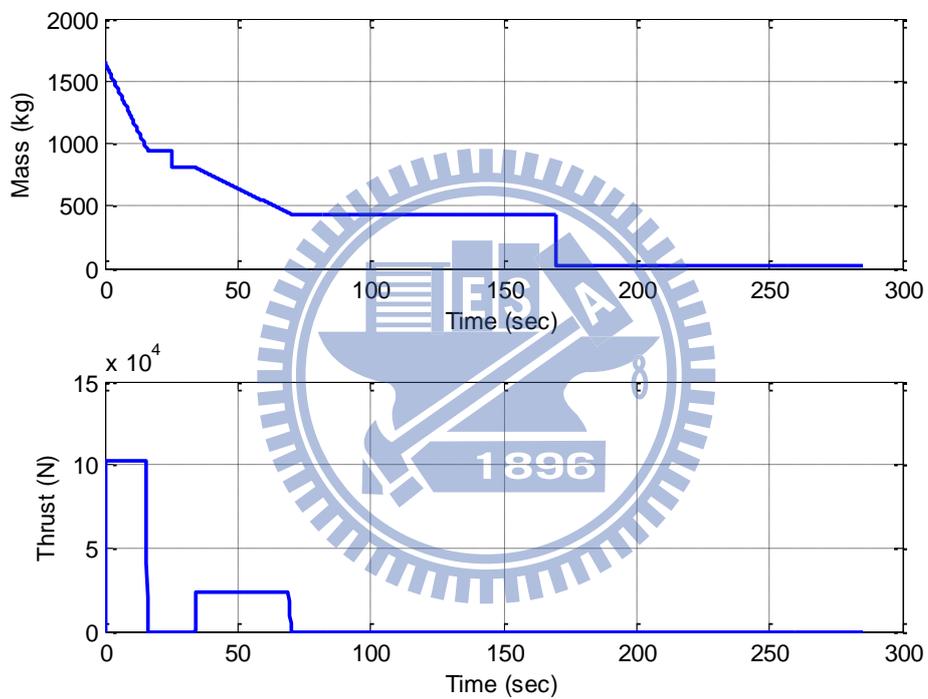


圖 6.1：飛行過程中的質量與推力隨時間變化示意圖

	本論文火箭		探空五號	
	第一節	第二節	第一節	第二節
推進器				
總衝量	161,500(kg-s)	854,20(kg-s)	102,968(kg-s)	148,560(kg-s)
總燃燒時間	16s	36s	5.4s	29s

表 6.1：比較探空五號與本論文火箭的推力與作用時間

	本論文火箭	探空五號
第一節	703 (kg)	429.20(kg)
第二節	373 (kg)	645.17(kg)
燃料總質量	1076 (kg)	1074.37(kg)

表 6.2：比較探空五號與本論文火箭的燃料質量

飛彈總質量	1650(kg)
飛彈燃料總質量	1076(kg)
第一節燃料質量	703(kg)
第二節燃料質量	373(kg)
第一節燃料用完時質量	947(kg)
第一節飛彈脫離時質量	810(kg)
第二節燃料用完時質量	437(kg)
第二節飛彈脫離時質量	21(kg)

表 6.3：飛彈燃料質量分配狀況

6.2 模擬初始條件

分別建立三種不同來襲目標的資料庫如下表所示

	初始位置(km)	初始速度(km/s)	最大高度(km)
目標一	(900,-900,-30)	(-2.1,2.0,-2.0)	239.13
目標二	(750,-710,-30)	(-1.8,1.9,-1.6)	161.65
目標三	(350,-280,-97)	(-1.8,1.4,-0.1)	97.526

表 6.4：目標資料庫

目標一在雷達站的北方 900 公里，西方 900 公里，高度 30 公里，距離雷達站為 1273.1 公里。目標二在雷達站的北方 750 公里，西方 710 公里，高度 30 公里，距離雷達站為 1033.2 公里。目標三在雷達站的北方 350 公里，西方 280 公里，高度 97 公里，距離雷達站為 458.59 公里。三個來襲目標均朝向東南方飛行至雷達站附近落地，如圖 6.2 所示。

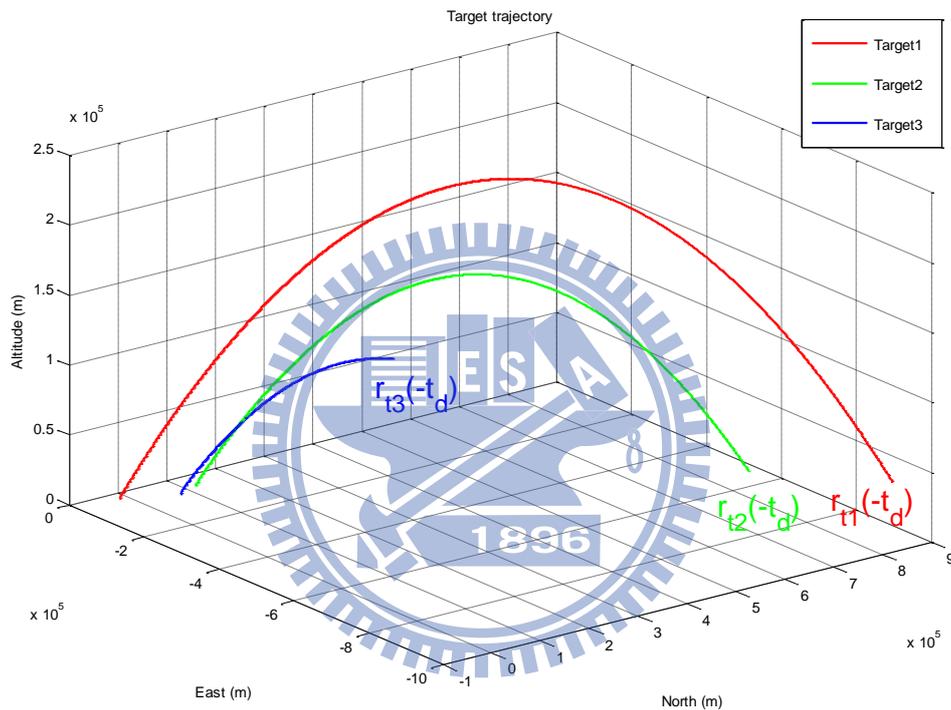


圖 6.2：不同目標的飛行軌跡圖(NED)

圖 6.3 為三個來襲目標的飛行高度隨時間變化關係圖，可以很明顯看出目標一飛行高度最高且被雷達偵測時處於逐漸上升階段，而目標三飛行高度最低且被雷達偵測時處於正在下降階段，目標二飛行最大高度則介於目標一與目標三之間且被雷達偵測時屬於逐漸上升階段。

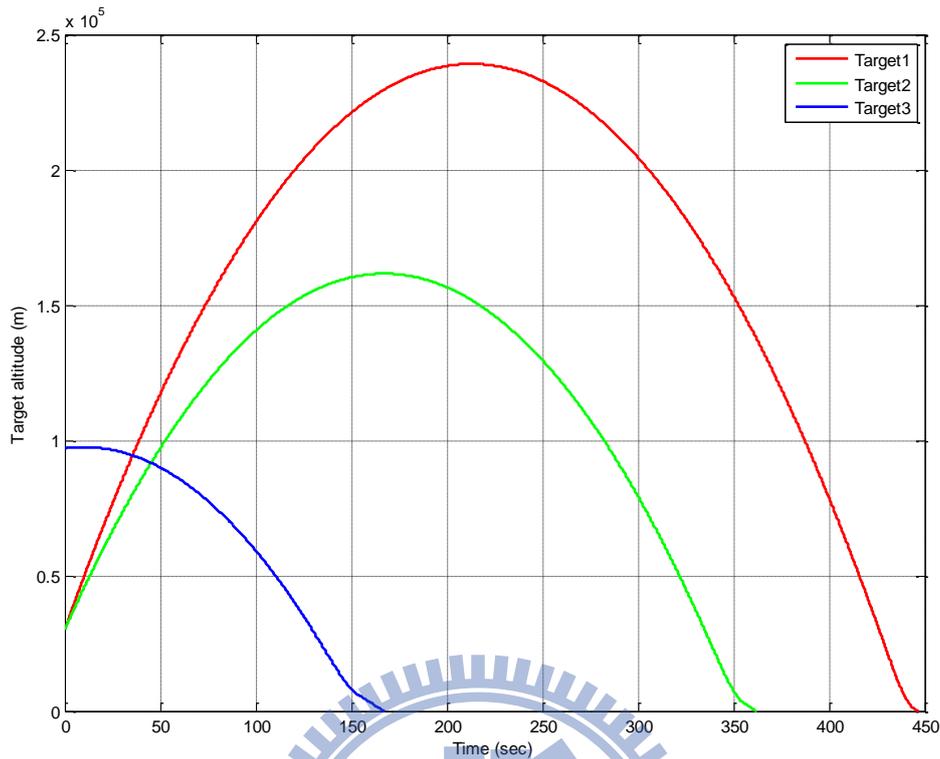


圖 6.3：不同目標的飛行高度對時間圖

飛彈初始位置 $(-1,0,-0.01)$ km。發射點在雷達站的南方 1 公里，高度 10 公尺。飛彈靜止於發射點，發射仰角固定為 70 度，發射的方向角固定為粗估的攔截點方向，即目標初始位置，更精確的發射方向角將在 6.4 節介紹。

其他時間參數 $t_1 = 16s$ ， $t_c = 18s$ ， $t_2 = 36s$ ， $t_h = 10s$ 。第一節飛彈脫離後的質量 $m_{20} = 810\text{kg}$ ，第二節燃燒時的最大推力 $T_2 = 23580\text{N}$ ，第二節推力的質量燃燒速率 $\dot{m}_2 = \frac{373}{36}\text{kg/s}$ ，重力加速度 $g = (0,0,0.0098)\text{km/s}^2$ 。攔截必要條件(4.9)式及(4.12)式參數 $h = 30$ ，(4.13)式參數 $h_r = 10$ 。

根據 3.3 節所推導的目標位置依三個目標分別為

目標一

$$r_i(t) = \begin{bmatrix} 900 \\ -900 \\ -30 \end{bmatrix} + \begin{bmatrix} -2.1 \\ 2.0 \\ -2.0 \end{bmatrix} (t+t_d) + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t+t_d)^2, \quad t \geq -t_d$$

目標二

$$r_i(t) = \begin{bmatrix} 750 \\ -710 \\ -30 \end{bmatrix} + \begin{bmatrix} -1.8 \\ 1.9 \\ -1.6 \end{bmatrix} (t+t_d) + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t+t_d)^2, \quad t \geq -t_d$$

目標三

$$r_i(t) = \begin{bmatrix} 350 \\ -280 \\ -97 \end{bmatrix} + \begin{bmatrix} -1.8 \\ 1.4 \\ -0.1 \end{bmatrix} (t+t_d) + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t+t_d)^2, \quad t \geq -t_d$$

目標速度依三個目標分別為

目標一

$$v_i(t) = \begin{bmatrix} -2.1 \\ 2.0 \\ -2.0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t+t_d)$$

目標二

$$v_i(t) = \begin{bmatrix} -1.8 \\ 1.9 \\ -1.6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t+t_d)$$

目標三

$$v_i(t) = \begin{bmatrix} -1.8 \\ 1.4 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t+t_d)$$

透過 Simulink 程式模擬將飛彈發射自由飛行至第二節點火前，根據 3.4 節所推導

飛彈的位置如下。當第一節火箭燃畢，即 $t=t_1$ 時，

飛彈的位置為

$$r_m(t_1) = \Delta R_1 = \begin{bmatrix} 1.7311 \\ -2.6115 \\ -6.2311 \end{bmatrix}$$

飛彈的速度為

$$v_m(t_1) = \Delta V_1 = \begin{bmatrix} 0.3781 \\ -0.3616 \\ -0.7866 \end{bmatrix}$$

當滑行完畢後，即 $t=t_1+t_c$ 時，

$$r_m(t_1+t_c) = \begin{bmatrix} 1.7311 \\ -2.6115 \\ -6.2311 \end{bmatrix} + \begin{bmatrix} 0.3781 \\ -0.3616 \\ -0.7866 \end{bmatrix} \cdot 18 + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} \cdot 18^2$$

當第二節火箭點火之後，即 $t_1+t_c \leq t \leq t_1+t_c+t_2$ 時，推力產生的加速度為

$$a_m(t) = \begin{bmatrix} 0.0133 \\ -0.0127 \\ -0.0214 \end{bmatrix} + (t-t_1-t_c) \begin{bmatrix} 1.6963 \cdot 10^{-4} \\ -1.6221 \cdot 10^{-4} \\ -2.7372 \cdot 10^{-4} \end{bmatrix}$$

飛彈的位置為

$$r_m(t) = \begin{bmatrix} 1.7311 \\ -2.6115 \\ -6.2311 \end{bmatrix} + \begin{bmatrix} 0.3781 \\ -0.3616 \\ -0.7866 \end{bmatrix} (t-16) + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t-16)^2 + \frac{1}{2} (t-34)^2 \begin{bmatrix} 0.0133 \\ -0.0127 \\ -0.0214 \end{bmatrix} + \frac{1}{6} (t-34)^3 \begin{bmatrix} 1.6963 \cdot 10^{-4} \\ -1.6221 \cdot 10^{-4} \\ -2.7372 \cdot 10^{-4} \end{bmatrix}$$

第二節火箭燃畢時，將 $t = t_1 + t_c + t_2$ 代入上式，可得飛彈的位置為

$$r_{m2f} = \begin{bmatrix} 1.7311 \\ -2.6115 \\ -6.2311 \end{bmatrix} + \begin{bmatrix} 0.3781 \\ -0.3616 \\ -0.7866 \end{bmatrix} \cdot 54 + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} \cdot 54^2 + \frac{1}{2} \cdot 36^2 \cdot \begin{bmatrix} 0.0133 \\ -0.0127 \\ -0.0214 \end{bmatrix} + \frac{1}{6} \cdot 36^3 \cdot \begin{bmatrix} 1.6963 \cdot 10^{-4} \\ -1.6221 \cdot 10^{-4} \\ -2.7372 \cdot 10^{-4} \end{bmatrix}$$

$$= \begin{bmatrix} 32.3895 \\ -31.9299 \\ -50.9403 \end{bmatrix}$$

飛彈的速度為

$$v_{m2f} = \begin{bmatrix} 0.3781 \\ -0.3616 \\ -0.7866 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} \cdot 54 + 36 \cdot \begin{bmatrix} 0.0133 \\ -0.0127 \\ -0.0214 \end{bmatrix} + \frac{1}{2} \cdot 36^2 \cdot \begin{bmatrix} 1.6963 \cdot 10^{-4} \\ -1.6221 \cdot 10^{-4} \\ -2.7372 \cdot 10^{-4} \end{bmatrix}$$

$$= \begin{bmatrix} 0.9847 \\ -0.9417 \\ -1.2362 \end{bmatrix}$$

當 $t \geq t_1 + t_c + t_2$ 時，即第三節截殺彈頭飛行時，飛彈的位置為

$$r_m(t) = r_{m2f} + v_{m2f} \cdot (t - 70) + \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0.0098 \end{bmatrix} (t - 70)^2$$

$$= \begin{bmatrix} -36.5412 + 0.9847t \\ 33.9880 - 0.9417t \\ 59.6039 - 1.9222t + 0.0049t^2 \end{bmatrix}$$

6.3 模擬結果

先透過 Simulink 程式實際模擬有中途導引狀況下當三種不同來襲目標與飛彈交錯時的攔截狀況，之後再利用 PF 及 SQP 演算法求出延遲時間最大值及最小值來驗證是否接近實際情況。

6.3.1 Simulink 模擬目標一來襲

利用 6.2 節所設定的初始條件，並且透過 Simulink 程式來模擬目標一在不同延遲時間的攔截狀況如表 6.5 所示，可以得到當目標與飛彈交錯時的零力誤失距離、目標高度、飛彈飛行時間、 $r_m(t)$ 與 $r_t(t)$ ，其中延遲時間每隔 10 秒模擬一次。

延遲時間 (s)	零力誤失 距離 (km)	目標高度 (km)	飛行時間 (s)	$r_m(t)$ (km)	$r_t(t)$ (km)
0	123.48	182.94	323.93	343.00	387.90
10	102.88	174.73	321.70	327.64	364.37
20	82.266	166.08	319.43	311.80	340.91
30	67.998	160.40	314.28	302.35	326.16
40	54.313	154.76	308.88	293.20	311.94
50	41.479	149.31	303.18	284.50	298.58
60	29.470	144.08	297.15	276.35	286.08
70	18.688	139.25	290.84	269.82	274.78
80	15.745	139.11	280.78	269.47	274.47
90	13.374	138.84	271.06	269.87	273.84
100	11.276	138.02	261.66	268.74	271.95
110	9.4430	136.78	252.54	266.63	269.11
120	7.9499	135.13	243.70	263.54	265.33
130	6.6508	133.31	235.05	259.67	261.22
140	5.5609	131.20	226.09	255.09	256.47
150	4.6438	128.64	218.31	249.87	250.76
160	3.8183	125.91	210.22	244.15	244.71
170	3.0696	122.99	202.26	237.82	238.34

180	2.5609	119.74	194.42	231.09	231.31
190	2.0304	116.29	186.73	223.80	223.93
200	1.6978	112.64	179.11	216.27	216.22
210	1.3078	108.78	171.63	208.21	208.17
220	1.0306	104.70	164.25	199.77	199.76
230	0.9190	100.39	156.94	191.11	191.01
240	0.7070	95.858	149.72	182.12	181.91
250	0.6000	91.247	142.59	172.69	172.78
260	0.5168	86.223	135.55	163.11	162.98
270	0.4759	81.114	128.59	153.12	153.16
280	0.4123	75.569	121.70	143.04	142.66
290	0.3141	69.926	114.90	132.56	132.15
300	0.3090	64.185	108.14	121.84	121.63
310	0.4850	58.162	101.49	110.76	110.81
320	0.4321	51.661	94.980	99.374	99.363
330	0.8890	44.853	88.546	87.776	87.685
340	1.3120	37.346	82.403	75.425	75.233

表 6.5：模擬目標一在不同延遲時間的攔截情形

圖 6.4 為目標一在延遲時間為零秒的情況下，目標與飛彈的飛行軌跡圖，可以看出當兩者交錯時飛彈的高度已明顯低於目標高度，造成最後飛彈比目標提早落地而導致攔截失敗。目標一與飛彈詳細的飛行高度可參考圖 6.5。

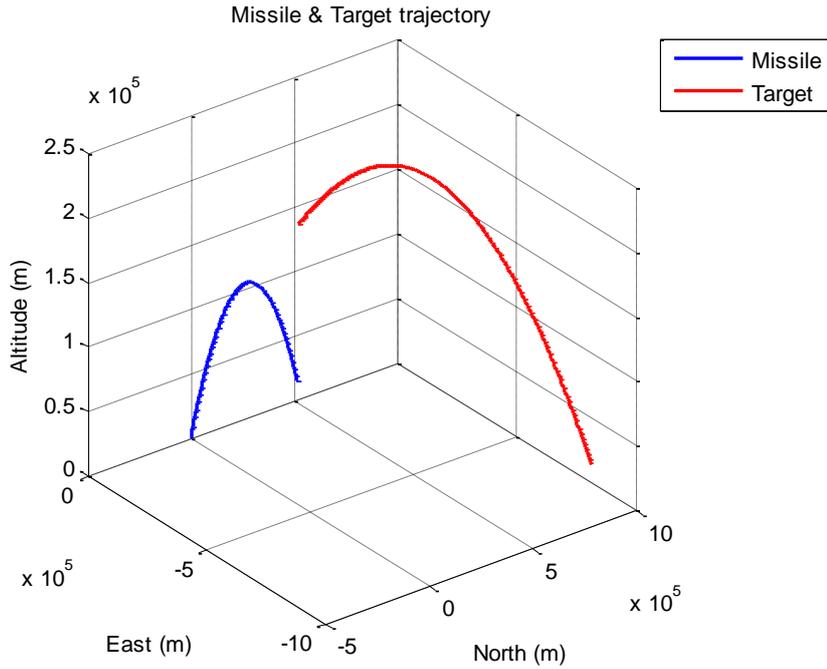


圖 6.4：飛彈與目標一的飛行軌跡圖(NED)

圖 6.5 為目標一在延遲時間為零秒的情況下，飛彈飛行 367.60 秒之後比目標提早落地，此時目標仍然在空中飛行且高度為 129.79 公里，導致最終攔截失敗，因此當目標一來襲時必須要有延遲時間等待目標接近才允許發射飛彈。

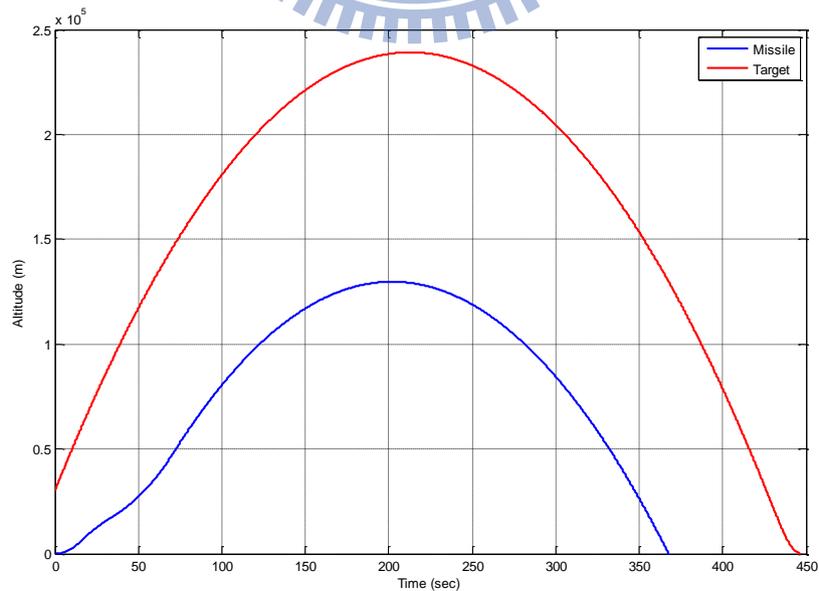


圖 6.5：飛彈與目標一的飛行高度對時間圖

圖 6.6 為延遲時間為 70 秒至 340 秒的零力誤失距離圖，當延遲 70 秒發射時，飛彈與目標一交錯的零力誤失距離為 18.69 公里，隨著延遲時間的增加，所對應的零力誤失距離愈來愈小。

圖 6.7 同樣為延遲時間為 70 秒至 340 秒的目標高度對時間圖，同樣隨著延遲時間的增加，所對應的目標高度愈來愈低，當延遲時間為 340 秒的目標高度接近 30 公里，且由表 6.5 查表可知此時飛彈飛行總時間為 82.40 秒，剛進入第三節終端飛行階段。

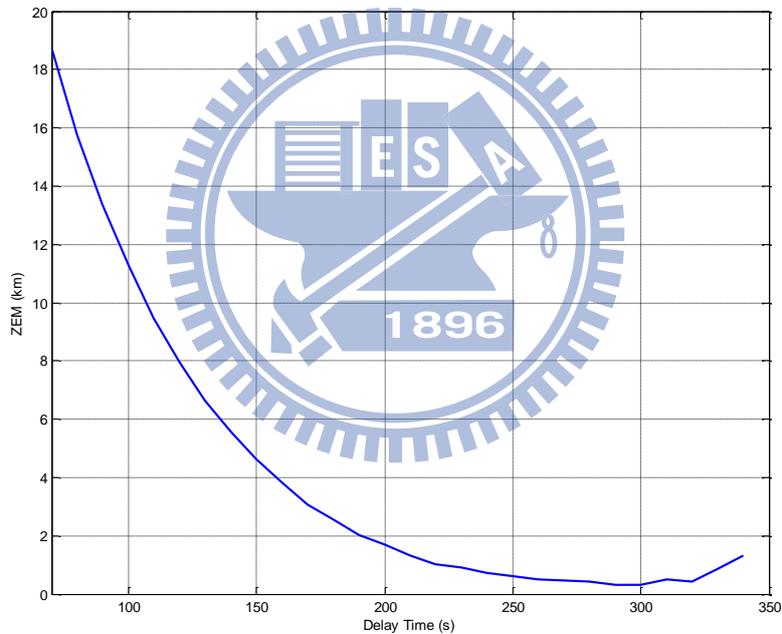


圖 6.6：目標一不同延遲時間對應的零力誤失距離圖

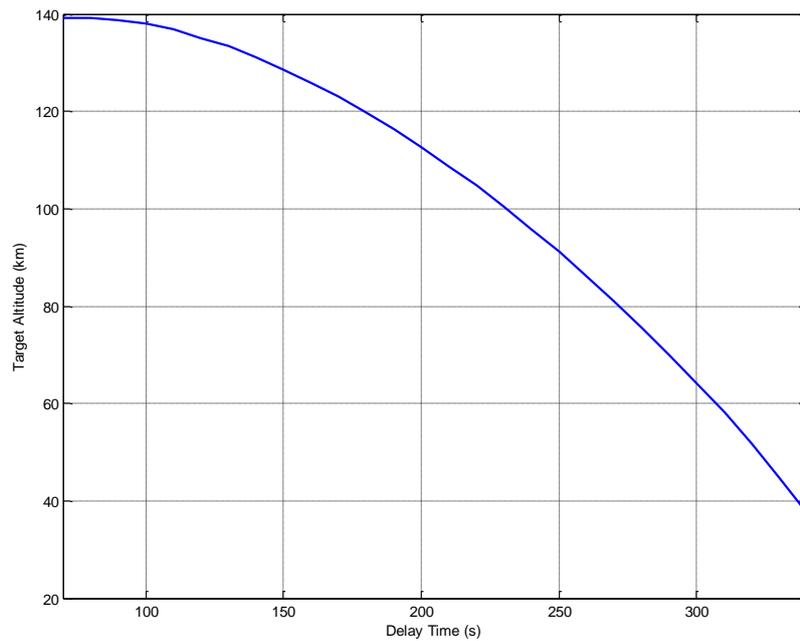


圖 6.7：目標一不同延遲時間對應的高度關係圖

6.3.1.1 執行 PF 演算法求解目標一

當目標一來襲時，利用 5.1 節所發展的 PF 演算法，透過 Matlab 程式撰寫可求出第四章所提出的最佳化問題(P1)及(P2)。圖 6.8 為根據第四章所提到的等式限制式 $h(x_1, x_2) = 0$ 及不等式限制式 $g_1(x_1, x_2) \leq 0$ 與 $g_5(x_1, x_2) \leq 0$ 所畫的示意圖，橫軸代表延遲時間 $x_1 \geq 0$ ，縱軸代表飛彈飛行時間 $x_2 > 0$ ，其中藍色曲線代表 $h(x_1, x_2) = 0$ ；綠色曲線代表 $g_1(x_1, x_2) = 0$ ，綠色曲線下方即為 $g_1(x_1, x_2) \leq 0$ 的可行解區域；紅色曲線代表 $g_5(x_1, x_2) = 0$ ，紅色曲線上方即為 $g_5(x_1, x_2) \leq 0$ 的可行解區域。因此從圖形可看出欲同時滿足等式及不等式限制式的延遲時間窗口為藍色線段 $\overline{m_1 m_2}$ ，即延遲時間最小值為 $m_1 \approx 115$ ，延遲時間最大值為 $m_2 \approx 321$ 。

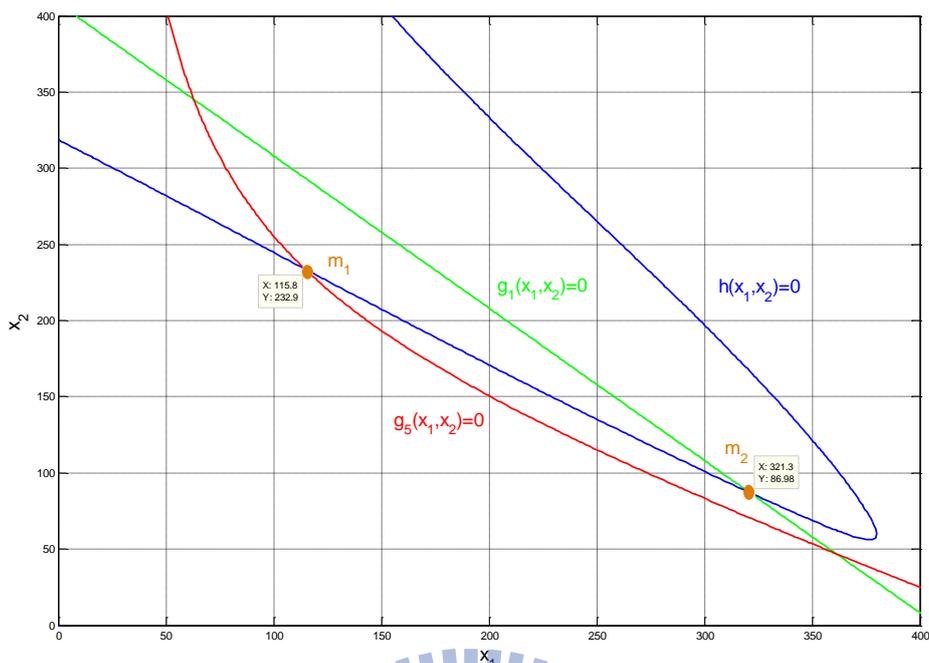


圖 6.8：目標—延遲時間示意圖

(i). 欲求解 $\max : f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最大值

兩個初始值分別為 $(x_1, x_2) = (335, 90)$ 及 $(x_1, x_2) = (280, 120)$ ，詳細求解結果如表 6.6 所示

演算法	初始值 (x_1, x_2)	延遲時間 最大值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(335,90)	320.95	87.189	408.13
	(280,120)	320.39	87.550	407.94

表 6.6：PF 求解目標—延遲時間最大值

(ii).欲求解 $\min: f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最小值

兩個初始值分別為 $(x_1, x_2) = (150, 220)$ 及 $(x_1, x_2) = (0, 130)$ ，詳細求解結果如表 6.7 所示

演算法	初始值 (x_1, x_2)	延遲時間 最小值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(150,220)	115.20	233.35	348.55
	(0,130)	115.11	233.42	348.53

表 6.7：PF 求解目標一延遲時間最小值

6.3.1.2 執行 SQP 演算法求解目標一

當目標一來襲時，利用 5.3 節所發展的 SQP 演算法，透過 Matlab 程式撰寫可求出第四章所提出的最佳化問題(P1)及(P2)。

(i).欲求解 $\max: f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最大值

兩個初始值分別為 $(x_1, x_2) = (335, 90)$ 及 $(x_1, x_2) = (280, 120)$ ，詳細求解結果如表 6.8 所示

演算法	初始值 (x_1, x_2)	延遲時間 最大值 (s)	飛行時間 (s)	目標函數值 (s)
SQP	(335,90)	321.01	87.148	408.15
	(280,120)	321.01	87.148	408.15

表 6.8：SQP 求解目標一延遲時間最大值

(ii).欲求解 $\min: f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最小值

兩個初始值分別為 $(x_1, x_2) = (150, 220)$ 及 $(x_1, x_2) = (0, 130)$ ，詳細求解結果如表 6.9 所示

演算法	初始值 (x_1, x_2)	延遲時間 最小值 (s)	飛行時間 (s)	目標函數值 (s)
SQP	(150,220)	114.69	233.73	348.42
	(0,130)	114.69	233.73	348.42

表 6.9：SQP 求解目標一延遲時間最小值

透過 PF 及 SQP 演算法所求得的延遲時間窗口大約為 [115 , 320]，而經由模擬得知：當延遲時間為零秒，由於飛彈與目標初始距離太遠且目標屬於加力上升階段，造成飛彈比目標提早落地而攔截失敗，因此必須要等待延遲時間才能發射；當延遲時間為 110 秒的零力誤失距離約 9 公里，且此時目標及飛彈高度均遠大於 30 公里；而當延遲時間為 340 秒的零力誤失距離約 1 公里，此時目標及飛彈高度相當接近 30 公里且飛彈剛進入第三節終端飛行階段，若延遲時間再增加，將會造成攔截失敗。依數值結果與實際模擬的情形差異不大且相當符合實際情形。

6.3.2 Simulink 模擬目標二來襲

在此我們考慮目標二來襲時，利用 6.2 節所設定的初始條件，並且透過 Simulink 程式來模擬目標二在不同延遲時間的攔截狀況如表 6.10 所示，可以得到當目標與飛彈交錯時的零力誤失距離、目標高度、飛彈飛行時間、 $r_m(t)$ 與 $r_t(t)$ ，其中延遲時間每隔 10 秒模擬一次。

延遲時間 (s)	零力誤失 距離 (km)	目標高度 (km)	飛行時間 (s)	$r_m(t)$ (km)	$r_t(t)$ (km)
0	8.0242	95.919	285.80	310.10	311.66
10	6.5246	95.138	276.59	308.28	309.73
20	5.3273	93.900	267.64	305.77	306.69
30	4.1689	92.309	259.02	302.28	302.82
40	3.2672	90.467	250.64	297.99	298.39
50	2.5631	88.248	242.51	293.01	293.12
60	2.0561	85.756	234.63	287.20	287.30
70	1.5113	83.100	226.84	281.19	281.19
80	1.2426	80.025	219.32	274.36	274.24
90	0.9214	76.891	211.81	267.53	267.29
100	0.7037	73.439	204.51	260.04	259.78
110	0.5520	69.785	197.35	252.09	251.98
120	0.4330	65.787	190.31	243.95	243.63
130	0.3241	61.703	183.37	235.36	235.28
140	0.3799	57.110	176.61	226.42	226.12
150	0.3058	52.414	169.95	217.12	216.97
160	0.3346	47.468	163.33	207.80	207.59
170	0.4421	42.113	156.96	197.79	197.71
180	0.6175	36.487	150.67	187.62	187.64
190	0.8747	30.281	144.63	177.00	176.91

表 6.10：模擬目標二在不同延遲時間的攔截情形

圖 6.9 為目標二在延遲時間為零秒的情況下，目標與飛彈的飛行軌跡圖，可以明顯看出當兩者交錯時的高度相當接近，且此時的零力誤失距離約為 8 公里。

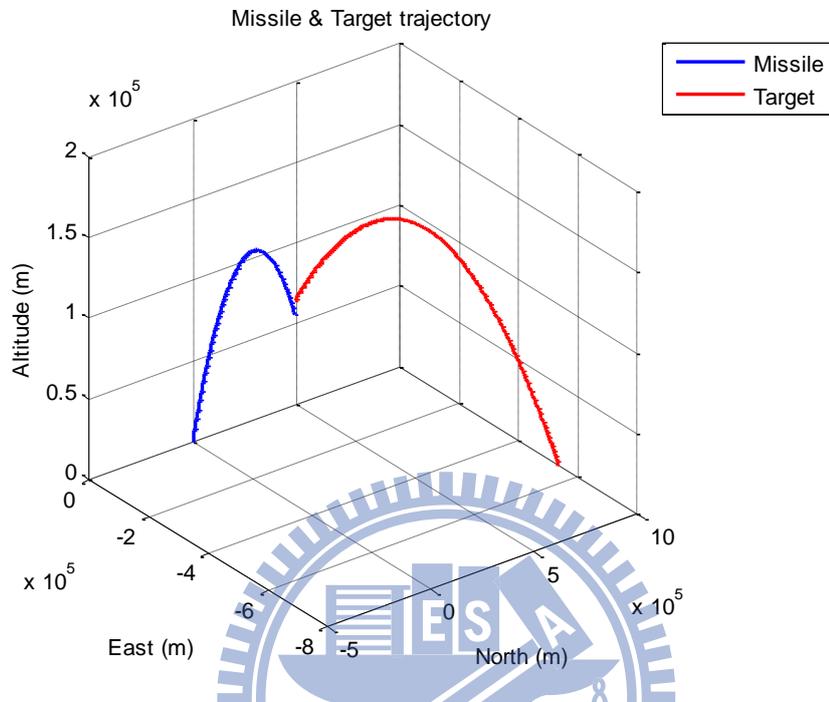


圖 6.9：飛彈與目標二的飛行軌跡圖(NED)

圖 6.10 為目標二在不同延遲時間所對應的零力誤失距離圖，當延遲零秒發射時，飛彈與目標交錯的零力誤失距離約為 8 公里，隨著延遲時間的增加，所對應的零力誤失距離愈來愈小，且 $r_m(t)$ 與 $r_t(t)$ 愈來愈接近；當延遲 190 秒發射時，兩者交錯的產生的零力誤失距離小於 1 公里，且由表 6.10 查表可知此時目標的高度相當接近 30 公里，若延遲時間再增加，高度將會低於 30 公里。

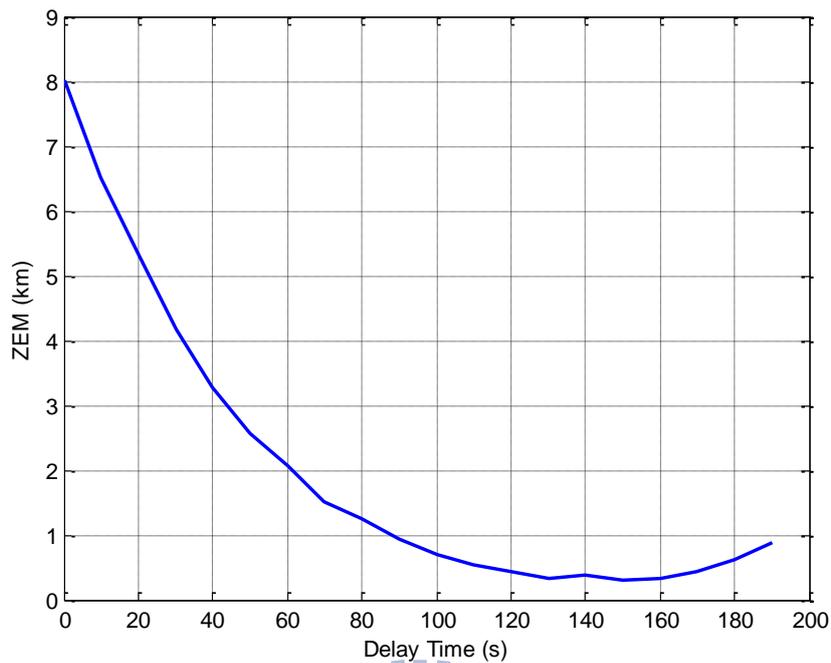


圖 6.10：目標二不同延遲時間對應的零力誤失距離圖

6.3.2.1 執行 PF 演算法求解目標二

當目標二來襲時，利用 5.1 節所發展的 PF 演算法，透過 Matlab 程式撰寫可求出第四章所提出的最佳化問題(P1)及(P2)。圖 6.11 為根據第四章所提到的等式限制式 $h(x_1, x_2) = 0$ 及不等式限制式 $g_1(x_1, x_2) \leq 0$ 與 $g_5(x_1, x_2) \leq 0$ 所畫的示意圖，橫軸代表延遲時間 $x_1 \geq 0$ ，縱軸代表飛彈飛行時間 $x_2 > 0$ ，其中藍色曲線代表 $h(x_1, x_2) = 0$ ；綠色曲線代表 $g_1(x_1, x_2) = 0$ ，綠色曲線下方即為 $g_1(x_1, x_2) \leq 0$ 的可行解區域；紅色曲線代表 $g_5(x_1, x_2) = 0$ ，紅色曲線上方即為 $g_5(x_1, x_2) \leq 0$ 的可行解區域。因此從圖形可看出欲同時滿足等式及不等式限制式的延遲時間窗口為藍色線段 $\overline{m_1 m_2}$ ，即延遲時間最小值為 $m_1 \approx 0$ ，延遲時間最大值為 $m_2 \approx 181$ 。

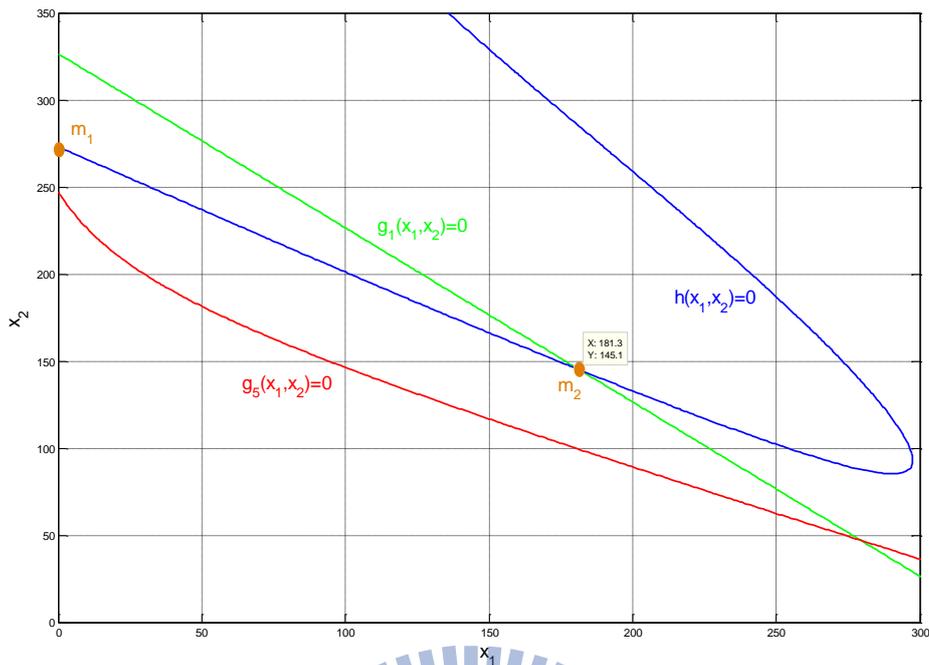


圖 6.11：目標二延遲時間示意圖

(i). 欲求解 $\max : f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最大值

兩個初始值分別為 $(x_1, x_2) = (200, 110)$ 及 $(x_1, x_2) = (160, 150)$ ，詳細求解結果如表

6.11 所示

演算法	初始值 (x_1, x_2)	延遲時間 最大值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(200,110)	181.14	145.23	326.37
	(160,150)	181.12	145.24	326.36

表 6.11：PF 求解目標二延遲時間最大值

(ii).欲求解 $\min: f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最小值

兩個初始值分別為 $(x_1, x_2) = (50, 190)$ 及 $(x_1, x_2) = (0, 300)$ ，詳細求解結果如表 6.12 所示

演算法	初始值 (x_1, x_2)	延遲時間 最小值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(50,190)	0.8959	272.38	273.27
	(0,300)	0.0637	272.98	273.04

表 6.12：PF 求解目標二延遲時間最小值

6.3.2.2 執行 SQP 演算法求解目標二

當目標二來襲時，利用 5.3 節所發展的 SQP 演算法，透過 Matlab 程式撰寫可求出第四章所提出的最佳化問題(P1)及(P2)。

(i).欲求解 $\max: f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最大值

兩個初始值分別為 $(x_1, x_2) = (200, 110)$ 及 $(x_1, x_2) = (160, 150)$ ，詳細求解結果如表 6.13 所示

演算法	初始值 (x_1, x_2)	延遲時間 最大值 (s)	飛行時間 (s)	目標函數值 (s)
SQP	(200,110)	181.59	144.93	326.52
	(160,150)	181.59	144.93	326.52

表 6.13：SQP 求解目標二延遲時間最大值

(ii).欲求解 $\min: f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最小值

兩個初始值分別為 $(x_1, x_2) = (50, 190)$ 及 $(x_1, x_2) = (0, 300)$ ，詳細求解結果如表 6.14 所示

演算法	初始值 (x_1, x_2)	延遲時間 最小值 (s)	飛行時間 (s)	目標函數值 (s)
SQP	(50,190)	0.0000	273.02	273.02
	(0,300)	0.0000	273.02	273.02

表 6.14：SQP 求解目標二延遲時間最小值

透過 PF 及 SQP 演算法所求得的延遲時間窗口大約為 $[0, 180]$ ，而經由模擬得知：當延遲時間為零秒，目標與飛彈交錯時的零力誤失距離為 8 公里，若配合第三節適當的終端導引是有機會可以攔截成功；而當延遲時間為 190 秒的零力誤失距離小於 1 公里，此時目標及飛彈高度相當接近 30 公里，若延遲時間再增加，將會造成目標高度低於 30 公里而導致攔截失敗。依數值結果與實際模擬的情形差異不大且相當符合實際情形。

6.3.3 Simulink 模擬目標三來襲

最後，我們考慮目標三來襲時，利用 6.2 節所設定的初始條件，並且透過 Simulink 程式來模擬目標三在不同延遲時間的攔截狀況如表 6.15 所示，可以得到當目標與飛彈交錯時的零力誤失距離、目標高度、飛彈飛行時間、 $r_m(t)$ 與 $r_t(t)$ ，其中延遲時間每隔 10 秒模擬一次。

延遲時間 (s)	零力誤失 距離 (km)	目標高度 (km)	飛行時間 (s)	$r_m(t)$ (km)	$r_t(t)$ (km)
0	0.2954	30.000	129.93	155.27	155.00
10	0.5551	24.387	124.40	144.43	144.27
20	0.8638	19.004	119.15	133.59	133.46
30	1.4198	13.181	114.41	122.52	122.36
40	1.4839	7.2535	111.26	111.86	111.83

表 6.15：模擬目標三在不同延遲時間的攔截情形

表 6.15 值得注意的是不同延遲時間所對應的目標高度，雖然當目標與飛彈交錯時的零力誤失距離均在 2 公里以下，且飛彈也進入終端飛行階段，但是當延遲時間愈久，目標與飛彈的高度也愈低。特別是當延遲時間為 40 秒兩者交錯時的目標高度低於 10 公里，代表延遲時間過長。

圖 6.12 與圖 6.13 分別為目標三在延遲時間為零秒的情況下與飛彈的飛行軌跡圖及飛行高度對時間圖，可以清楚看出目標與飛彈交錯時的高度很接近 30 公里，應此可以推測若發射前的延遲時間大於零秒，則目標高度將低於 30 公里。

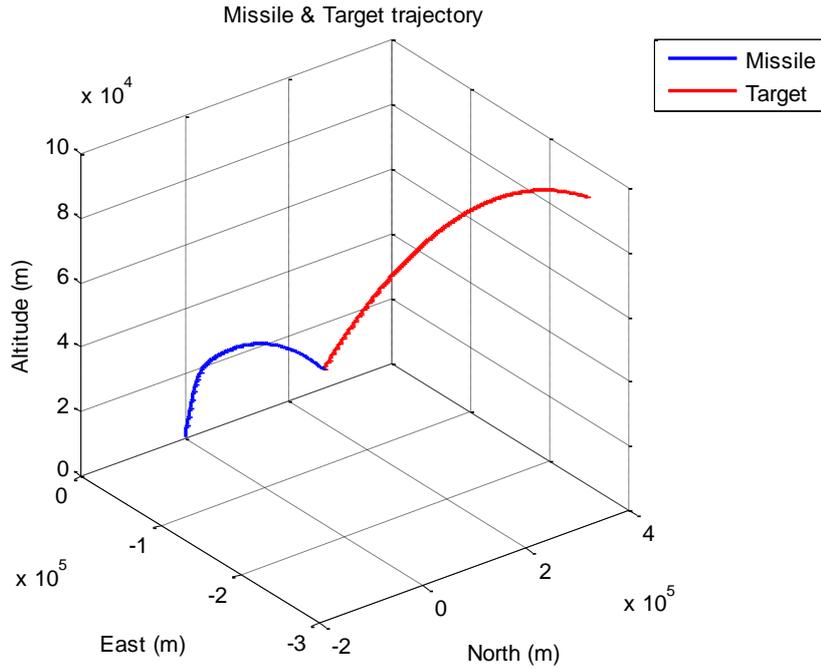


圖 6.12：飛彈與目標三的飛行軌跡圖(NED)

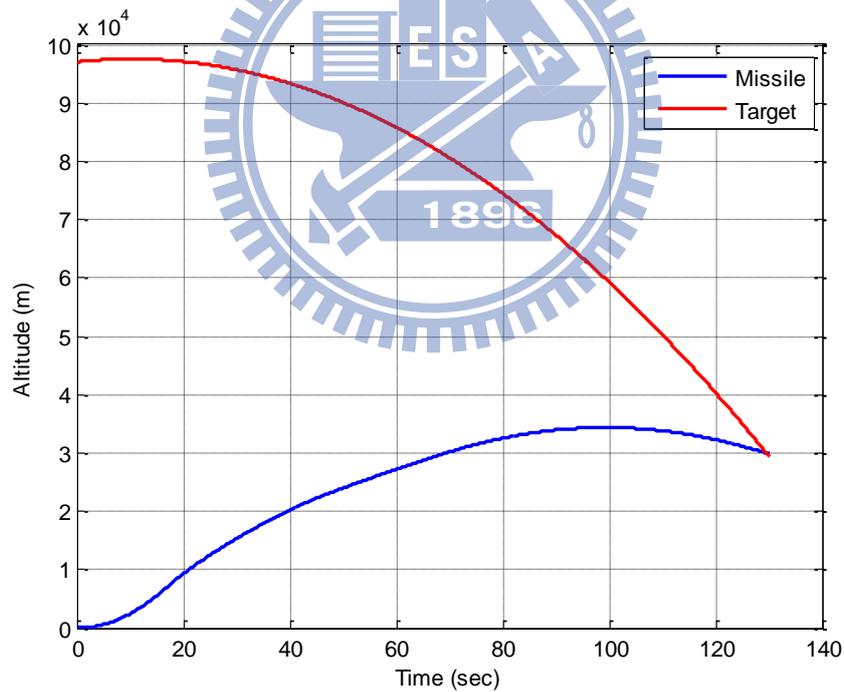


圖 6.13：飛彈與目標三的飛行高度對時間圖

6.3.3.1 執行 PF 演算法求解目標三

當目標三來襲時，利用 5.1 節所發展的 PF 演算法，透過 Matlab 程式撰寫可求出第四章所提出的最佳化問題(P1)及(P2)。圖 6.14 為根據第四章所提到的等式

限制式 $h(x_1, x_2) = 0$ 及不等式限制式 $g_1(x_1, x_2) \leq 0$ 所畫的示意圖，橫軸代表延遲時間 $x_1 \geq 0$ ，縱軸代表飛彈飛行時間 $x_2 > 0$ ，其中藍色曲線代表 $h(x_1, x_2) = 0$ ；綠色曲線代表 $g_1(x_1, x_2) = 0$ ，綠色曲線下方即為 $g_1(x_1, x_2) \leq 0$ 的可行解區域。因此從圖形可看出欲同時滿足等式及不等式限制式的延遲時間最大值為 m_2 ，相當接近於零，因此可以推測延遲時間最小值為零。

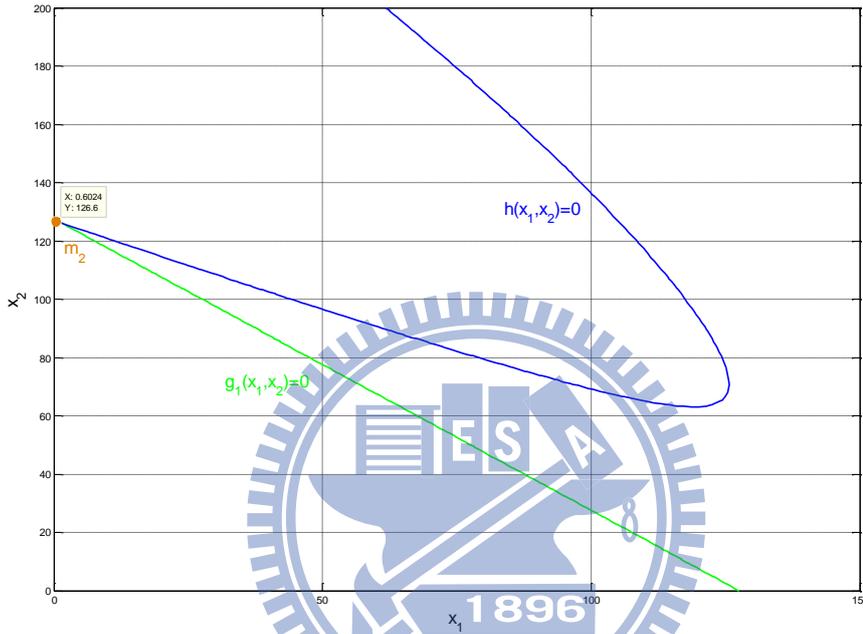


圖 6.14：目標三延遲時間示意圖

(i). 欲求解 $\max : f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最大值

兩個初始值分別為 $(x_1, x_2) = (250, 250)$ 及 $(x_1, x_2) = (300, 200)$ ，詳細求解結果如表

6.16 所示

演算法	初始值 (x_1, x_2)	延遲時間 最大值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(250,250)	1.5877	125.87	127.45
	(300,200)	1.5558	125.89	127.44

表 6.16：PF 求解目標三延遲時間最大值

6.3.3.2 執行 SQP 演算法求解目標三

當目標三來襲時，利用 5.3 節所發展的 SQP 演算法，透過 Matlab 程式撰寫可求出第四章所提出的最佳化問題(P1)及(P2)。

(i). 欲求解 $\max : f(x_1, x_2) = x_1 + x_2$ ，即延遲時間與飛行時間最大值

兩個初始值分別為 $(x_1, x_2) = (250, 250)$ 及 $(x_1, x_2) = (300, 200)$ ，詳細求解結果如表

6.17 所示

演算法	初始值 (x_1, x_2)	延遲時間 最大值 (s)	飛行時間 (s)	目標函數值 (s)
SQP	(250,250)	1.8974	125.68	127.57
	(300,200)	1.8974	125.68	127.57

表 6.17：SQP 求解目標三延遲時間最大值

由於目標三經計算所得的最大延遲時間為 1 秒，代表幾乎不能延遲發射，因此延遲時間最小值為 0 秒，必須立刻發射，否則目標與飛彈高度將低於 30 公里。

6.4 延遲時間評估與驗證

針對以上三個不同來襲目標經由 PF 及 SQP 演算法所計算出的延遲時間窗口可以整理成表 6.18 所述，其中延遲時間窗口定義為 $[t_{d \min}, t_{d \max}]$ ，飛行時間

$(t_{f \min}, t_{f \max})$ 代表 $t_{d \min}$ 所對應的飛行時間為 $t_{f \min}$ 秒， $t_{d \max}$ 所對應的飛行時間為

$t_{f \max}$ 秒。定義可靠攔截時間 t_s 為

$$t_s = \frac{t_{d \min} + t_{d \max}}{2} \quad (6.1)$$

即延遲時間最小值與延遲時間最大值之和的一半，代表若延遲 t_s 秒發射飛彈，將可以順利攔截成功，即在中途導引結束後的零力誤失距離在 2 公里以內、目標及飛彈高度均保持在 30 公里以上與飛彈順利進入第三節終端階段飛行。將可靠攔截時間所對應的飛彈飛行時間可定義為相對飛行時間 t_{sf} ，其表示如下

$$t_{sf} = \frac{t_{f \min} + t_{f \max}}{2} \quad (6.2)$$

即延遲時間最小值對應的飛行時間與延遲時間最大值對應的飛行時間之和的一半。

	演算法	延遲時間窗口	飛行時間	可靠攔截時間	相對飛行時間
目標一	PF	[115,320]	(233,87)	217	160
	SQP	[114,321]	(233,87)	217	160
目標二	PF	[0,181]	(272,145)	90	208
	SQP	[0,181]	(273,144)	90	208
目標三	PF	[0,1]	(125,125)	0	125
	SQP	[0,1]	(125,125)	0	125

表 6.18：三種不同來襲目標的延遲時間窗口與可靠攔截時間

我們利用上述所定義的可靠攔截時間來驗證求出的可靠攔截時間是否與實際情形相符，以下將根據三種不同來襲目標來進行模擬。其中飛彈發射仰角為 70 度，發射的方向角可以由預估攔截點 $r_i(t_f)$ 的位置來決定。預估攔截點可以表示為

$$r_t(t_f) = r_{t0} + v_{t0} \cdot (t_d + t_f) + \frac{1}{2} g \cdot (t_d + t_f)^2 \quad (6.3)$$

則初始攔截方向 w 可表示為

$$w = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = r_{t0} + v_{t0} \cdot (t_d + t_f) + \frac{1}{2} g \cdot (t_d + t_f)^2 - r_{m0} \quad (6.4)$$

其中 r_{m0} 為飛彈初始位置。飛彈發射的初始方向角即為朝著 w 的方向

$$\psi(0) = \tan^{-1} \frac{w_y}{w_x} \quad (6.5)$$

6.4.1 當目標一來襲時

當目標一來襲時，由於初始距離較遠且飛行最大高度最高，因此必須要等待延遲時間才能發射，透過計算所得延遲時間最小值為 115 秒且延遲時間最大值為 320 秒，代表目標一來襲時不能立刻發射反而必須等待 115 秒或是最久等待至 320 秒再發射。以下部分將討論若選擇可靠攔截時間來發射，是否符合實際攔截情形。

將 $t_d = t_s = 217$ ， $t_f = t_{sf} = 160$ 代入(6.4)式及(6.5)式可以求得飛彈發射的初始方向

角，經 Simulink 模擬可得下列結果：

方向角 (deg)	零力誤失 距離 (km)	目標高度 (km)	飛行時間 (s)	$r_m(t)$ (km)	$r_t(t)$ (km)
-53.18	0.8705	106.28	166.28	202.93	203.00

表 6.19：目標一求解可靠攔截時間代入模擬結果

當目標一來襲時，若選擇可靠攔截時間為 217 秒，相對飛行時間為 160 秒，透過模擬當飛彈與目標交錯時的零力誤失距離小於 2 公里，且目標高度為 106 公里，飛彈也進入第三節終端飛行，此時目標與飛彈距離雷達站約 203 公里，因此經計算所得的可靠攔截時間相當精準，可以有效的攔截目標。圖 6.15、圖 6.16 與圖 6.17 分別為飛彈與目標一的飛行軌跡圖、飛行高度對時間圖與對座標原點的距離圖。

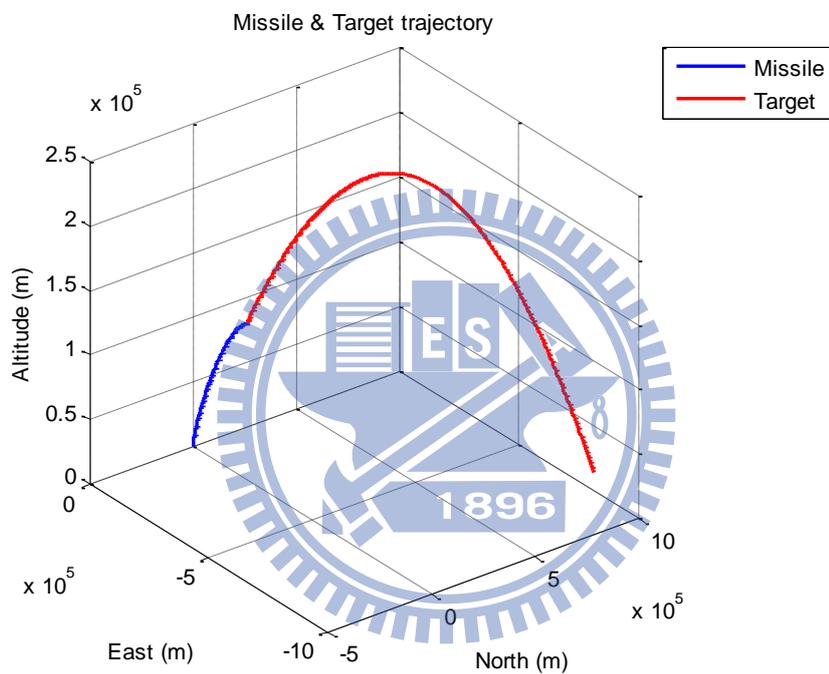


圖 6.15：驗證飛彈與目標一的飛行軌跡圖(NED)

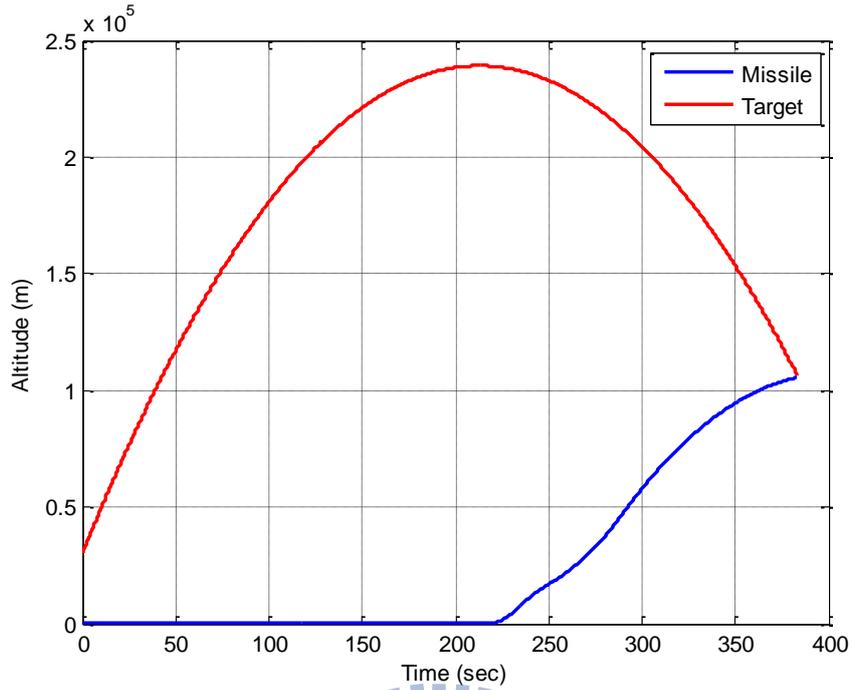


圖 6.16：驗證飛彈與目標一的飛行高度對時間圖

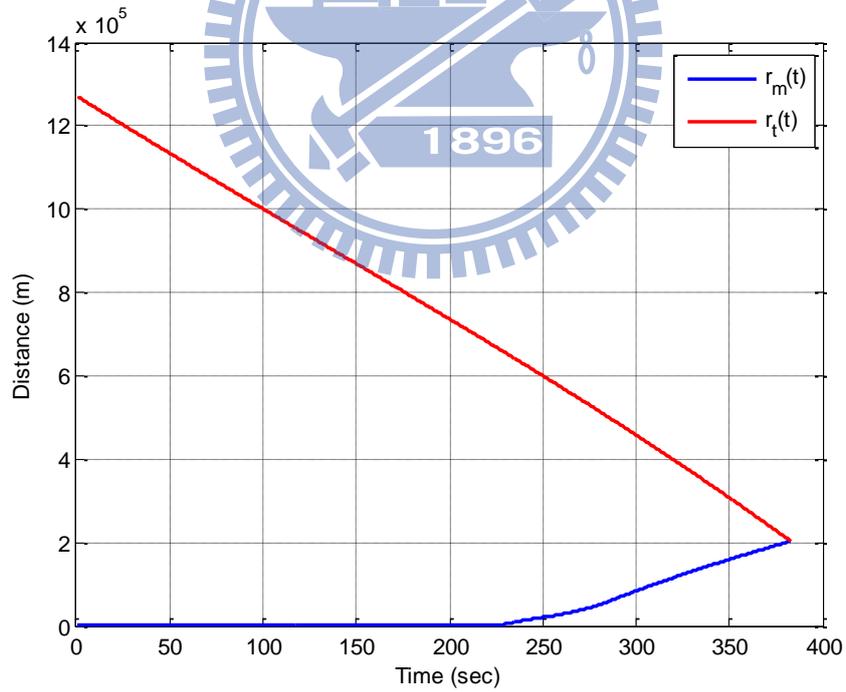


圖 6.17：驗證飛彈與目標一對座標原點的距離圖

6.4.2 當目標二來襲時

當目標二來襲時，與目標一不同的是目標二所到達的最大高度並未如目標一那麼高，透過計算所得延遲時間最小值為 0 秒且延遲時間最大值為 181 秒，代表目標二來襲時可以立刻發射或是最久等待至 181 秒再發射。以下部分將討論若選擇可靠攔截時間來發射，是否符合實際攔截情形。將 $t_d = t_s = 90$ ， $t_f = t_{sf} = 208$ 代入(6.4)式及(6.5)式可以求得飛彈發射的初始方向角，經 Simulink 模擬可得下列結果：

方向角 (deg)	零力誤失 距離 (km)	目標高度 (km)	飛行時間 (s)	$r_m(t)$ (km)	$r_t(t)$ (km)
-33.82	0.4681	77.397	211.44	268.58	268.40

表 6.20：目標二求解可靠攔截時間代入模擬結果

當目標二來襲時，透過模擬當飛彈與目標交錯時的零力誤失距離小於 1 公里，且目標高度為 77 公里，飛彈也進入第三節終端飛行，此時目標與飛彈距離雷達站約 268 公里，因此經計算所得的可靠攔截時間相當精準，可以有效的攔截目標。圖 6.18、圖 6.19 與圖 6.20 分別為飛彈與目標二的飛行軌跡圖、飛行高度對時間圖與對座標原點的距離圖。

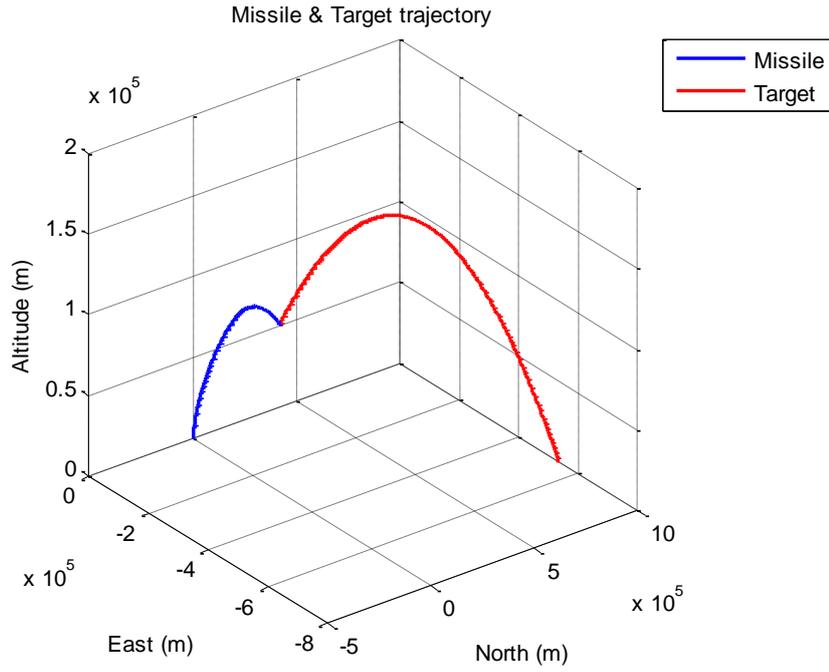


圖 6.18：驗證飛彈與目標二的飛行軌跡圖(NED)

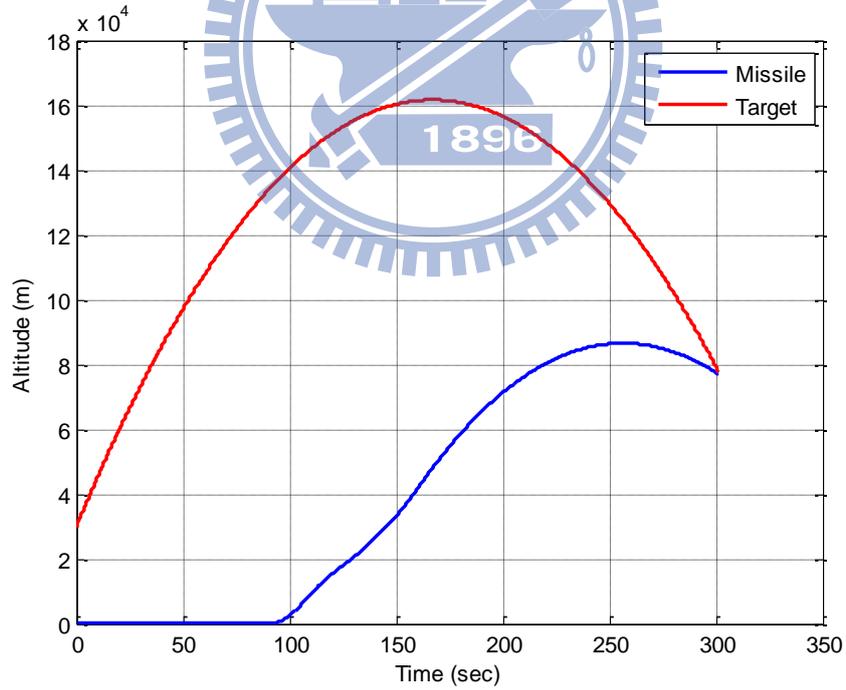


圖 6.19：驗證飛彈與目標二的飛行高度對時間圖

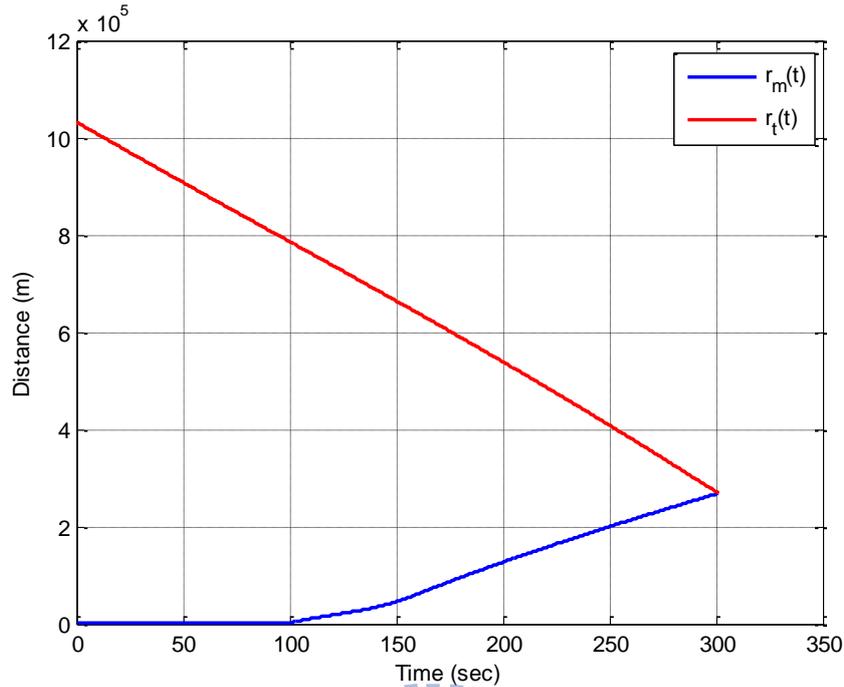


圖 6.20：驗證飛彈與目標二對座標原點的距離圖

6.4.3 當目標三來襲時

當目標三來襲時，是屬於將要下降的短距離飛彈，應此必須立刻發射飛彈，透過計算所得延遲時間最大值為 1 秒，代表目標三來襲時必須立刻發射，若延遲發射時間，則目標高度將低於 30 公里，因此延遲時間最小值為 0 秒。以下部分將討論若選擇可靠攔截時間來發射，是否符合實際攔截情形。將 $t_d = t_s = 0$ ， $t_f = t_{sf} = 125$ 代入(6.4)式及(6.5)式可以求得飛彈發射的初始方向角，經 Simulink 模擬可得下列結果：

方向角 (deg)	零力誤失 距離 (km)	目標高度 (km)	飛行時間 (s)	$r_m(t)$ (km)	$r_t(t)$ (km)
-39.80	0.2917	30.000	129.92	155.32	155.00

表 6.21：目標三求解可靠攔截時間代入模擬結果

當目標三來襲時，透過模擬當飛彈與目標交錯時的零力誤失距離小於 1 公里，且目標高度剛好為 30 公里，飛彈也進入第三節終端飛行，此時目標與飛彈距離雷達站約 155 公里。所以若延遲時間發射，則目標高度將低於 30 公里，因此經計算所得的可靠攔截時間相當精準，可以有效的攔截目標。圖 6.21、圖 6.22 與圖 6.23 分別為飛彈與目標三的飛行軌跡圖、飛行高度對時間圖與對座標原點的距離圖。

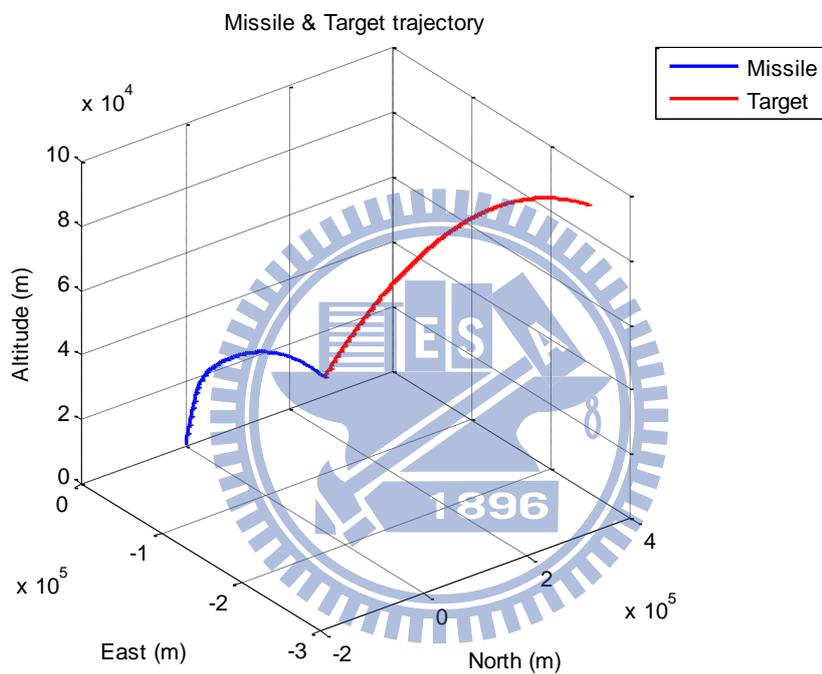


圖 6.21：驗證飛彈與目標三的飛行軌跡圖(NED)

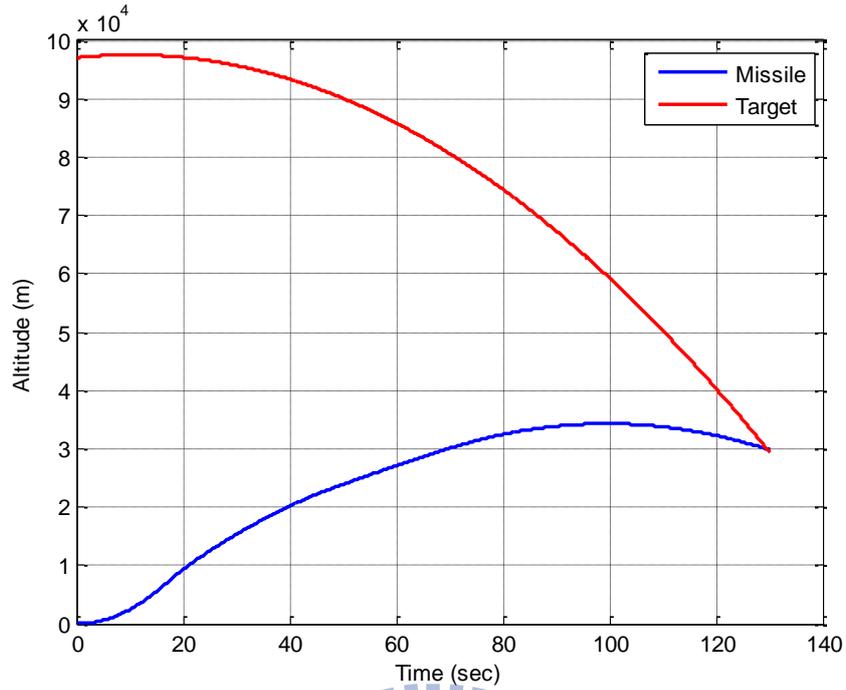


圖 6.22：驗證飛彈與目標三的飛行高度對時間圖

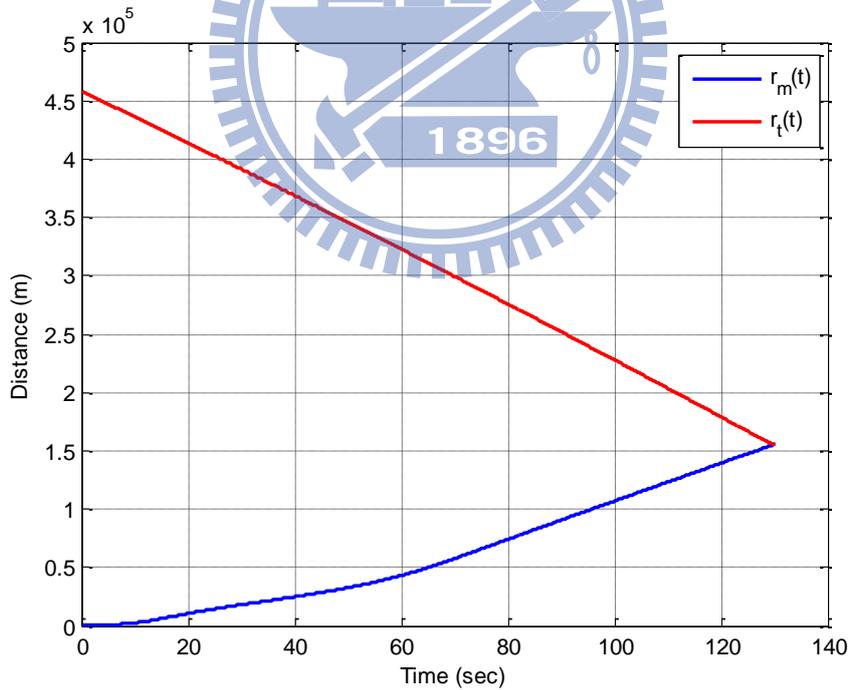


圖 6.23：驗證飛彈與目標三對座標原點的距離圖

由 6.4 節所整理分析當三種不同目標來襲時，透過 PF 及 SQP 方法可有效求得三種不同來襲飛彈的延遲時間窗口及所定義的可靠攔截時間與相對飛行時間

相當符合實際模擬的情形。因此顯示第三章所推導目標與飛彈的位置函數及第四章最佳化問題的建構相當成功，透過兩種演算法皆可在射控階段順利求出飛彈發射時間且亦有相當不錯的準確度。

6.5 不同 h_r 參數的數值結果

由於目標和飛彈的高度差 h_r 對零力誤失距離有很大的影響，因此我們考慮將兩者之間的高度差設定在某為一個範圍之內，來滿足當延遲時間最小的情況下，兩者的高度差必須要在此範圍內，才能符合攔截成功的必要條件。即第四章的(4.7)式

$$-r_{tz}(t) + r_{mz}(t) \leq h_r$$

由 6.2 節模擬初始條件可知 $h_r = 10$ ，在本節所要探討的是當 h_r 參數改變時對數值計算結果所造成的差異，我們將選取 $h_r = 5$ 及 $h_r = 10$ 來作數值結果的比較，即目標與飛彈的相對高度差為 5 公里及 10 公里對求解延遲時間最小值的差異。在此僅考慮當目標一來襲時，利用 PF 及 SQP 演算法求得延遲時間最小值如表 6.22 及表 6.23 所示。

(i). 當 $h_r = 5$ ，即 $g_5(x_1, x_2) = -r_{tz}(t) + r_{mz}(t) - 5 \leq 0$

演算法	初始值 (x_1, x_2)	延遲時間 最小值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(150,220)	123.74	226.98	350.72
	(0,130)	122.83	227.65	350.48
SQP	(150,220)	122.83	227.66	350.49
	(0,130)	122.83	227.66	350.49

表 6.22：當 $h_r=5$ 求解延遲時間最小值

根據表 6.22 得知經 PF 及 SQP 所求出的延遲時間最小值為 123 秒，且由表 6.6 得知延遲時間最大值為 320 秒，根據(6.1)式可算出可靠攔截時間為 222 秒。

(ii).當 $h_r=10$ ，即 $g_5(x_1, x_2) = -r_{tz}(t) + r_{mz}(t) - 10 \leq 0$

演算法	初始值 (x_1, x_2)	延遲時間 最小值 (s)	飛行時間 (s)	目標函數值 (s)
PF	(150,220)	115.16	233.38	348.54
	(0,130)	115.12	233.41	348.53
SQP	(150,220)	114.69	233.73	348.42
	(0,130)	114.69	233.73	348.42

表 6.23：當 $h_r=10$ 求解延遲時間最小值

根據表 6.23 得知經 PF 及 SQP 所求出的延遲時間最小值為 115 秒，且由表 6.6 得知延遲時間最大值為 320 秒，根據(6.1)式可算出可靠攔截時間為 217 秒。

圖 6.24 為不同高度差與實際延遲時間所對應的零力誤失距離圖，其中紅色線段與藍色線段分別代表經 PF 及 SQP 演算法與利用模擬所求出的延遲時間最小值對應的零力誤失距離。

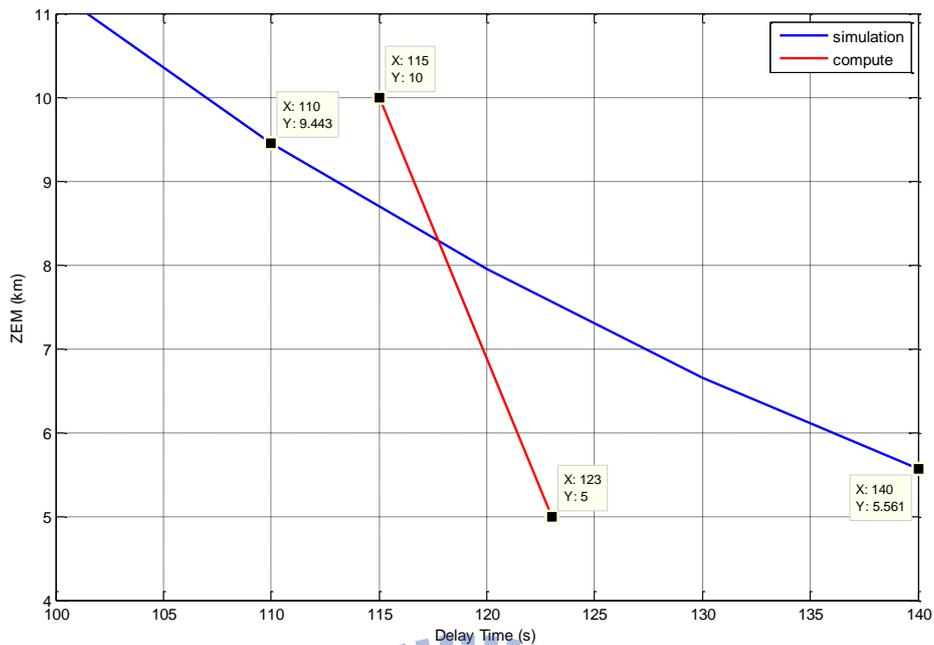


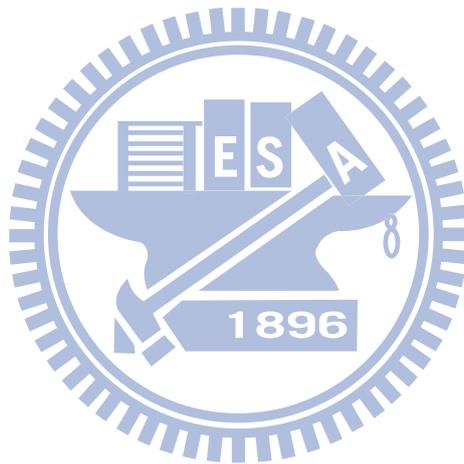
圖 6.24：不同高度差與實際延遲時間所對應的零力誤失距離圖

透過圖 6.24 可以清楚看出當目標一來襲時，當 $g_5(x_1, x_2) \leq 5$ ，經計算所得的延遲時間最小值為 123 秒，而實際當延遲 140 秒發射所產生的零力誤失距離約為 5 公里，兩者產生的延遲時間誤差為 17 秒；當 $g_5(x_1, x_2) \leq 10$ ，經計算所得的延遲時間最小值為 115 秒，而實際當延遲 110 秒發射所產生的零力誤失距離約為 9 公里，兩者產生的延遲時間誤差為 5 秒。將不同高度差所產生的延遲時間誤差及可靠攔截時間整理如表 6.24 所述。

	延遲時間誤差(s)	可靠攔截時間(s)
$g_5(x_1, x_2) \leq 5$	17	222
$g_5(x_1, x_2) \leq 10$	5	217

表 6.24：不同高度差產生的延遲時間誤差及可靠攔截時間

由表 6.24 可看出不同高度差對可靠攔截時間的影響不大，但是 $g_5(x_1, x_2) \leq 10$ 產生的延遲時間誤差較小，因此我們選擇當目標與飛彈的高度差在 10 公里內，即 $g_5(x_1, x_2) \leq 10$ ，為滿足延遲時間最小的攔截成功必要條件。



第七章

結論

攔截來襲的彈道飛彈是一個相當困難且具有挑戰性的問題，一直是世界各國在國防科技研究上的重要發展目標。執行任務時間短，速度快，誤差距離小為彈道飛彈的特性，其射程從數百公里至數千公里不等，因此為了要攔截各種射程不等的來襲目標，我們必須要在射控階段求出發射時間窗口，再透過準確的中途導引及終端導引階段才能夠攔截成功。

本論文以三個不同來襲目標來做數值計算以及分析，探討不同射程的目標在發射前所需要等待的延遲時間，即發射時間窗口。由於第二節火箭的加速度受到中途導引律的影響，但是在射控階段無從得知飛彈的加速度，因此在數值上必須作簡化的假設以及採用近似線性的方式來表示。此外，求解過程中也會受到飛彈與目標高度的限制範圍有些許影響，例如在本論文中預設攔截時的目標與飛彈高度需大於 30 公里以及目標與飛彈的相對高度差為 10 公里，對於日後的相關攔截研究可以朝向此方面來進行深入探討。

透過數值模擬結果顯示，本論文所發展的 PF 及 SQP 演算法與推導飛彈與目標的位置和速度函數以及建構出的最佳化問題相當可行並有不錯的準確度。

本論文針對攔截飛彈的研究只能算是一小步，對於攔截飛彈的導引與控制還有許多部份需要進一步的探討，希望透過模擬所得的相關數據和結果可提供未來攔截飛彈硬體的發展作些許的貢獻並有其參考的價值。

參考資料

- [1] Stephen G.Nash and Ariela Sofer , “Linear and nonlinear programming,”
McGraw-Hill,1996.
- [2] Mokhtar S. Bazaraa and Hanif D. Sherali and C.M. Shetty , “Nonlinear
programming theory and algorithms , second edition ,” *John Wiley & Sons,Inc,*
1993.
- [3] N.K. Kwak and Marc J. Schniederjans , “ Introduction to mathematical
programming,” *Robert E. Krieger Publishing Company Co.,Inc,* 1987.
- [4] P.Venkataraman , “Applied optimization with matlab programming , second
edition ,” *John Wiley & Sons,Inc,* 2009.
- [5] Zhenxiao Gao and Tianyuan Xiao and Wenhui Fan , “Adaptive relaxation penalty
function method for equal constrained optimization in differential evolution, ”
2009 fifth international conference in natural computation.
- [6] 林清安, “轉向姿態導控系統設計(三),” 期末報告, 2009 年 11 月.
- [7] 林世修, “攔截飛彈的中途導引與控制,” 碩士論文, 2009 年 7 月.
- [8] 陳科祥, “轉向與姿態控制研究初步分析,” 2009 年 5 月.
- [9] “探空五號發動機設計審查報告”, 中科院二所固推組, 2005 年 6 月.
- [10] B. Etkin , “ Dynamics of Atmospheric Flight,” *Wiley,* 1972.

附錄 1 主程式碼

```
clear all;

clc;

rmx0=-1000; rmy0=0; rmz0=-10;

txs=1; % 選擇目標

if (txs==1)

%===case1===%

rtx0=900000; rty0=-900000; rtz0=-30000;

Vtx0=-2100; Vty0=2000; Vtz0=-2000;

elseif (txs==2)

%===case2===%

rtx0=750000; rty0=-710000; rtz0=-30000;

Vtx0=-1800; Vty0=1900; Vtz0=-1600;

elseif (txs==3)

%===case3===%

rtx0=350000; rty0=-280000; rtz0=-97000;

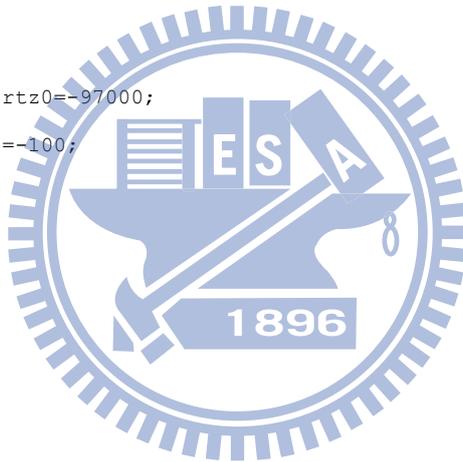
Vtx0=-1800; Vty0=1400; Vtz0=-100;

end

Constraint_INI % 限制式

PF_COM % 執行PF

SQP_COM % 執行SQP
```



附錄 2 限制式程式碼

```

x1=sym('x1'); % x1為延遲時間
x2=sym('x2'); % x2為飛行時間
RMx0=rmx0/1000;RMy0=rmy0/1000;RMz0=rmz0/1000; % 飛彈初始位置
rt0x=rtx0/1000;rt0y=rt0y/1000;rt0z=rtz0/1000; % 目標初始位置
vt0x=Vtx0/1000;vt0y=Vty0/1000;vt0z=Vtz0/1000; % 目標初始速度
rm0x=1.7311;rm0y=-2.6115;rm0z=-6.2311; % 第一節燃畢後的位置
vm0x=0.3781;vm0y=-0.3616;vm0z=-0.7866; % 第一節燃畢後的速度
gm1=0;gm2=0;gm3=0.0098; % 飛彈重力
gt1=0;gt2=0;gt3=0.0098; % 目標重力
t1=16; % 第一節燃燒時間
t2=36; % 第二節燃燒時間
tc=18; % 滑行時間
T2=23580; % 第二節最大推力
m20=810; % 第二節初始質量
m2d=373/36; % 質量燃燒率
v1_norm=sqrt(vm0x^2+vm0y^2+vm0z^2);
Lv1x=(vm0x+gm1*tc)/sqrt((vm0x+gm1*tc)^2+(vm0y+gm2*tc)^2+(vm0z+gm3*tc)^2);
Lv1y=(vm0y+gm2*tc)/sqrt((vm0x+gm1*tc)^2+(vm0y+gm2*tc)^2+(vm0z+gm3*tc)^2);
Lv1z=(vm0z+gm3*tc)/sqrt((vm0x+gm1*tc)^2+(vm0y+gm2*tc)^2+(vm0z+gm3*tc)^2);
a1=T2/(m20*1000)*Lv1x; % X-方向線性加速度a
a2=T2/(m20*1000)*Lv1y; % Y-方向線性加速度a
a3=T2/(m20*1000)*Lv1z; % Z-方向線性加速度a
b1=(T2*m2d/(m20^2*1000))*Lv1x; % X-方向線性加速度b
b2=(T2*m2d/(m20^2*1000))*Lv1y; % Y-方向線性加速度b
b3=(T2*m2d/(m20^2*1000))*Lv1z; % Z-方向線性加速度b

%%% 目標位置 %%%
rtx=rt0x+vt0x*(x2+x1)+0.5*gt1*(x2+x1)^2;
rty=rt0y+vt0y*(x2+x1)+0.5*gt2*(x2+x1)^2;
rtz=rt0z+vt0z*(x2+x1)+0.5*gt3*(x2+x1)^2;
Rt=sqrt((rtx-0)^2+(rty-0)^2+(rtz-0)^2);

%%% 第二節燃畢後速度 %%%
vm2fx=vm0x+gm1*(tc+t2)+t2*a1+0.5*t2^2*b1;
vm2fy=vm0y+gm2*(tc+t2)+t2*a2+0.5*t2^2*b2;
vm2fz=vm0z+gm3*(tc+t2)+t2*a3+0.5*t2^2*b3;

```

```

%%% 第三節飛行 %%%

rm3x_max=rm0x+vm0x*(tc+t2)+0.5*gm1*(tc+t2)^2+0.5*t2^2*a1+1*t2^3*b1/6+0.5*gm1*(x2-t1-tc
-t2)^2+vm2fx*(x2-t1-tc-t2);

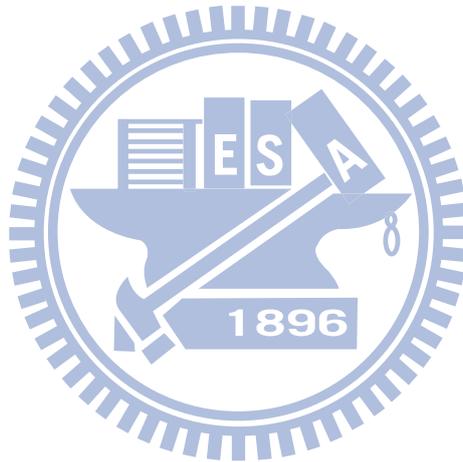
rm3y_max=rm0y+vm0y*(tc+t2)+0.5*gm2*(tc+t2)^2+0.5*t2^2*a2+1*t2^3*b2/6+0.5*gm2*(x2-t1-tc
-t2)^2+vm2fy*(x2-t1-tc-t2);

rm3z_max=rm0z+vm0z*(tc+t2)+0.5*gm3*(tc+t2)^2+0.5*t2^2*a3+1*t2^3*b3/6+0.5*gm3*(x2-t1-tc
-t2)^2+vm2fz*(x2-t1-tc-t2);

Rm3_max=sqrt((rm3x_max-0)^2+(rm3y_max-0)^2+(rm3z_max-0)^2);

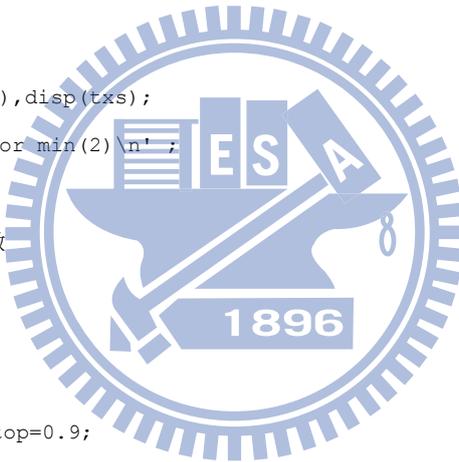
h_stg3=Rt^2-Rm3_max^2;      % Rm(t)=Rt(t)
al_t=-rtz;                  % 目標高度
al_m=-rm3z_max;            % 飛彈高度
g_tm=(-rtz)-(-rm3z_max);   % 目標與飛彈高度差

```



附錄 3 PF 程式碼

```
%%%===== input =====%%%  
  
% x(1,1) 為延遲時間初始值  
% x(1,2) 為飛行時間初始值  
  
%%%===== output =====%%%  
  
% T(1,i) 為延遲時間值  
% T(2,i) 為飛行時間值  
% T(3,i) 為目標函數值  
% T(4,i) 為g1不等式值  
% T(5,i) 為h等式值  
% T(6,i) 為g3不等式值  
% T(7,i) 為收斂值  
% T(8,i) 為c值  
  
syms x1 x2 f g1 h1 c  
fprintf( 'Target Case :' ),disp( txs );  
string1 = 'choose max(1) or min(2) \n';  
xs = input( string1 );  
f = (x1+x2); % 目標函數  
fobj = f;  
if (xs==1)  
    x(1,1)=300;x(1,2)=200;  
    C(1,1)=0.5;growth=2;stop=0.9;  
elseif (xs==2)  
    x(1,1)=0;x(1,2)=300;  
    C(1,1)=0.5;growth=2;stop=0.9;  
end  
renew=1;  
while (renew==1)  
    h_altitude=30; % h參數  
    h_r=10; % hr參數  
    i=1;t1=16;tc=18;t2=36;th=10;  
    if (xs==1)  
        h1=h_stg3; % 等式限制式  
        g1=h_altitude-(a1_t); % 不等式限制式  
        g3=-x1; % 不等式限制式
```



```

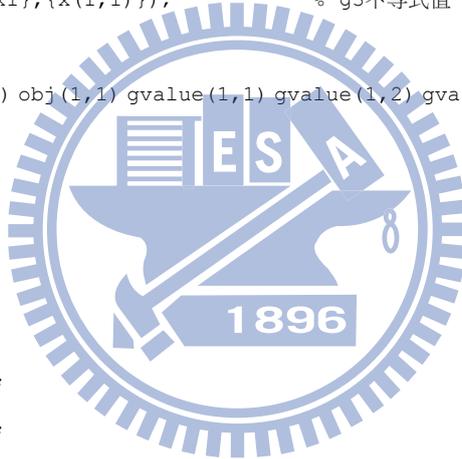
elseif (xs==2)
    h1=h_stg3;           % 等式限制式
    g1=(g_tm)-h_r;      % 不等式限制式
    g2=-x2+tc+t1+t2+th; % 不等式限制式
    g3=-x1;             % 不等式限制式
end
start1=0;start2=0;
while (start1==0)
    g1n=subs(g1,{x1,x2},{x(1,1),x(1,2)});
    if (xs==2)
        g2n=subs(g2,{x2},{x(1,2)});
    end
    g3n=subs(g3,{x1},{x(1,1)});
    h1n=subs(h1,{x1,x2},{x(1,1),x(1,2)});
    f1n=subs(fobj,{x1,x2},{x(1,1),x(1,2)});
    if (xs==1)
        theta_c_x=f1n-C(1,1)*(max(0,g1n)^2+max(0,g3n)^2+abs(h1n)^2);
        thetacx=f-c*(g1^2+g3^2+h1^2);
    elseif (xs==2)
        theta_c_x=f1n+C(1,1)*(max(0,g1n)^2+max(0,g2n)^2+max(0,g3n)^2+abs(h1n)^2);
        thetacx=f+c*(g1^2+g2^2+g3^2+h1^2);
    end
    k(1,1)=1;
    Jthetacx=[diff(thetacx,'x1');diff(thetacx,'x2')];
    Jvalue=subs(Jthetacx,{x1,x2,c},{x(1,1),x(1,2),C(1,1)});
    syms t
    if (xs==1)
        xold1=x(1,1)+t*Jvalue(1); % x1'= x1+t*Df(x)
        xold2=x(1,2)+t*Jvalue(2); % x2'= x2+t*Df(x)
    elseif (xs==2)
        xold1=x(1,1)-t*Jvalue(1); % x1'= x1-t*Df(x)
        xold2=x(1,2)-t*Jvalue(2); % x2'= x2-t*Df(x)
    end
    fxnew=subs(thetacx,{x1,x2,c},{xold1,xold2,C(1,1)});
    fxnewdiff=diff(fxnew,t);
    sol=solve(fxnewdiff,t); % solve t
    if (xs==1)
        xnew1=x(1,1)+sol*Jvalue(1); % x1'= x1+t*Df(x)
        xnew2=x(1,2)+sol*Jvalue(2); % x2'= x2+t*Df(x)

```

```

elseif (xs==2)
    xnew1=x(1,1)-sol*Jvalue(1); % x1'= x1-t*Df(x)
    xnew2=x(1,2)-sol*Jvalue(2); % x2'= x2-t*Df(x)
end
x(1,1)=real(xnew1(1,1)); % new x1
x(1,2)=real(xnew2(1,1)); % new x2
x(1,3)=k(1,1);
theta_cx(1,1)=subs(theta_c_x,{x1,x2,c},{x(1,1),x(1,2),C(1,1)});
obj(1,1)=subs(fobj,{x1,x2},{x(1,1),x(1,2)}); % 目標函數值
gvalue(1,1)=subs(g1,{x1,x2},{x(1,1),x(1,2)}); % g1不等式值
gvalue(1,2)=subs(h1,{x1,x2},{x(1,1),x(1,2)}); % h1等式值
if (xs==2)
    gvalue(1,3)=subs(g2,{x2},{x(1,2)}); % g2不等式值
end
gvalue(1,4)=subs(g3,{x1},{x(1,1)}); % g3不等式值
exam(1,1)=0;
T(1:8,1)=[x(1,1) x(1,2) obj(1,1) gvalue(1,1) gvalue(1,2) gvalue(1,4) exam(1,1) C(1,1)];
start1=start1+1;
break;
end
while (start2==0)
    i=i+1;
    x(i,1)=real(x(i-1,1));
    x(i,2)=real(x(i-1,2));
    C(i,1)=growth*C(i-1,1);
    if (xs==1)
        h1=h_stg3; % 等式限制式
        g1=h_altitude-(al_t); % 不等式限制式
        g3=-x1; % 不等式限制式
    elseif (xs==2)
        h1=h_stg3; % 等式限制式
        g1=(g_tm)-h_r; % 不等式限制式
        g2=-x2+tc+t1+t2+th; % 不等式限制式
        g3=-x1; % 不等式限制式
    end
    g1n=subs(g1,{x1,x2},{x(i,1),x(i,2)});
    h1n=subs(h1,{x1,x2},{x(i,1),x(i,2)});
    if (xs==2)
        g2n=subs(g2,{x2},{x(i,2)});

```



```

end

g3n=subs(g3,{x1},{x(i,1)});

f1n=subs(fobj,{x1,x2},{x(i,1),x(i,2)});

if (xs==1)

theta_c_x=f1n-C(i,1)*(max(0,g1n)^2+max(0,g3n)^2+abs(h1n)^2);

thetacx=f-c*(g1^2+g3^2+h1^2);

elseif (xs==2)

theta_c_x=f1n+C(i,1)*(max(0,g1n)^2+max(0,g2n)^2+max(0,g3n)^2+abs(h1n)^2);

thetacx=f+c*(g1^2+g2^2+g3^2+h1^2);

end

Jthetacx=[diff(thetacx,'x1');diff(thetacx,'x2')];

if (i>2)

    Jvalue=subs(Jthetacx,{x1,x2,c},{x(i,1),x(i,2),C(i,1)});

else

    Jvalue=subs(Jthetacx,{x1,x2,c},{x(i,1),x(i,2),C(1,1)});

end

syms tt

if (xs==1)

    xold1=x(i,1)+tt*Jvalue(1); % x1'= x1+t*Df(x)

    xold2=x(i,2)+tt*Jvalue(2); % x2'= x2+t*Df(x)

elseif (xs==2)

    xold1=x(i,1)-tt*Jvalue(1); % x1'= x1-t*Df(x)

    xold2=x(i,2)-tt*Jvalue(2); % x2'= x2-t*Df(x)

end

if (i>2)

    fxnew=subs(thetacx,{x1,x2,c},{xold1,xold2,C(i,1)});

else

    fxnew=subs(thetacx,{x1,x2,c},{xold1,xold2,C(1,1)});

end

fxnewdiff=diff(fxnew,tt);

sol2=solve(fxnewdiff,tt); % solve t

if (xs==1)

    xnew1=x(i,1)+sol2*Jvalue(1); % x1'= x1+t*Df(x)

    xnew2=x(i,2)+sol2*Jvalue(2); % x2'= x2+t*Df(x)

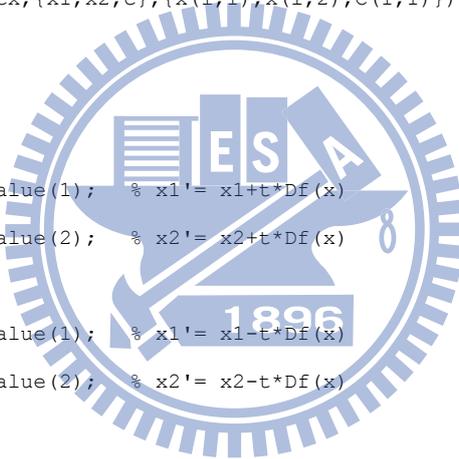
elseif (xs==2)

    xnew1=x(i,1)-sol2*Jvalue(1); % x1'= x1-t*Df(x)

    xnew2=x(i,2)-sol2*Jvalue(2); % x2'= x2-t*Df(x)

end

```

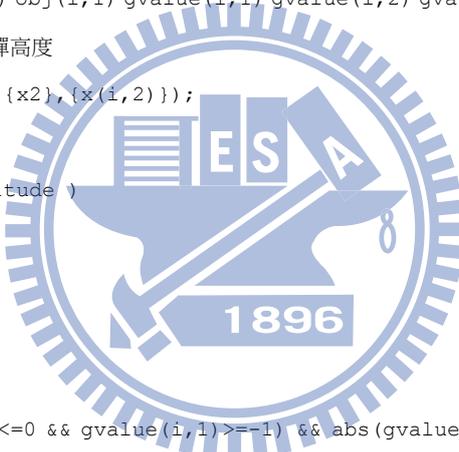


```

x(i,1)=real(xnew1(1,1));          % new x1
x(i,2)=real(xnew2(1,1));          % new x2
theta_cx(1,i)=real(subs(theta_c_x,{x1,x2,c},{x(i,1),x(i,2),C(i,1)}));
obj(i,1)=real(subs(fobj,{x1,x2},{x(i,1),x(i,2)}));      % 目標函數值
gvalue(i,1)=real(subs(g1,{x1,x2},{x(i,1),x(i,2)}));    % g1不等式值
gvalue(i,2)=real(subs(h1,{x1,x2},{x(i,1),x(i,2)}));    % h1等式值
if (xs==2)
    gvalue(i,3)=real(subs(g2,{x2},{x(i,2)}));          % g2不等式值
end
gvalue(i,4)=real(subs(g3,{x1},{x(i,1)}));              % g3不等式值
k(i,1)=i;
x(i,3)=k(i,1)
exam(1,i)=sqrt((x(i,1)-x(i-1,1))^2+(x(i,2)-x(i-1,2))^2); % 收斂值
%%%=====輸出結果=====%%
T(1:8,i)=[x(i,1) x(i,2) obj(i,1) gvalue(i,1) gvalue(i,2) gvalue(i,4) exam(1,i) C(i,1)];
heightm=al_m; % 檢查飛彈高度
height_m=subs(heightm,{x2},{x(i,2)});
kk=0;
if ( height_m < h_altitude )
    kk=kk+1;
end

if (xs==1)
    if ( (gvalue(i,1)<=0 && gvalue(i,1)>=-1) && abs(gvalue(i,2))<=1 && x(i,1) >= 0 &&
        x(i,2) >= 0 && exam(1,i) < stop )
        if ( height_m < h_altitude || x(i,2) < t1+tc+t2+th )
            x(1,1)=x(i,1)-5*kk;x(1,2)=x(i,2)+5*kk;
            C(1,1)=0.5;growth=2;stop=0.9;
        else
            renew=2;
            break;
        end
    elseif ( ( gvalue(i,1)<=0 || abs(gvalue(i,1))<=1) && abs(gvalue(i,2))<=1 && ( x(i,1)
        <=1 && x(i,1) >=0) && x(i,2) >= 0 )
        if ( height_m < h_altitude || x(i,2) < t1+tc+t2+th )
            x(1,1)=x(i,1)-5*kk;x(1,2)=x(i,2)+5*kk;
            C(1,1)=0.5;growth=2;stop=0.9;
        else
            renew=2;
        end
    end
end

```



```

        break;
    end
end
if ( gvalue(i,1) > 0 )
    if (gvalue(i,1) > 20)
        xx1=fix(x(i,1));
        xx2=fix(x(i,2));
        check_f=subs(fobj,{x1,x2},{xx1,xx2});
        limitg1=subs(g1,{x1},{0});
        limit_tf=max(double(solve(limitg1)));
        if ( check_f > limit_tf && x(i,2) >= t1+tc+t2+th )
            cc=subs(g1,{x1},{xx1});
            cc_tf=max(double(solve(cc)));
            miss_tf=x(i,2)-cc_tf;
            x(i,2)=x(i,2)-miss_tf;
            mp=solve(h1,g1);
            s1=double(mp.x1);
            s2=double(mp.x2);
            count=length(s1);
            for j=1:count
                ix1(j,1)=imag(s1(j,1));
                ix2(j,1)=imag(s2(j,1));
                rx1(j,1)=real(s1(j,1));
                rx2(j,1)=real(s2(j,1));
            end
            ccix1=find(ix1(1:count,1)==0);
            cclenix=length(ccix1);
            for jj=1:cclenix
                ac1(jj,1)=ccix1(jj,1);
                for jjj=1:cclenix
                    acr11(jjj,jj,1)=rx1(ac1(jj,1),1);
                    acr22(jjj,jj,1)=rx2(ac1(jj,1),1);
                end
            end
            for jjj=1:cclenix
                acr1(jjj,1)=acr11(1,jjj,1);
                acr2(jjj,1)=acr22(1,jjj,1);
            end
            for jjjj=1:cclenix

```

```

        if (acr1(jjjj)>0 && acr2(jjjj)>0)
            ktd=acr1(jjjj);ktf=acr2(jjjj);
        end
    end
    miss_td=x(i,1)-ktd;
    x(i,1)=x(i,1)-miss_td;
    elseif ( check_f > limit_tf && x(i,2) < tc+t1+t2+th )
        x(i,1)=x(i,1)-3;
    end
elseif (gvalue(i,1) <= 20)
    if ( abs(gvalue(i,1)) <= 1 )
        x(i,2)=x(i,2)+1;
    else
        x(i,2)=x(i,2)+3;
    end
end
elseif ( (gvalue(i,1)) < 0 )
    if ( (gvalue(i,1)) < -20 )
        x(i,1)=x(i,1)+10;
    elseif ( (gvalue(i,1)) < -10 && (gvalue(i,1)) >= -20 )
        x(i,1)=x(i,1)+5;
    elseif ( (gvalue(i,1)) < -4 && (gvalue(i,1)) >= -10 )
        x(i,1)=x(i,1)+3;
    elseif ( (gvalue(i,1)) < -1 && (gvalue(i,1)) >= -4)
        x(i,1)=x(i,1)+2;
    elseif ( (gvalue(i,1)) >= -1 )
        x(i,1)=x(i,1)+1;
    end
end
elseif(xs==2)
    if ( ( (gvalue(i,1)<=0 && gvalue(i,1)>=-1) && abs(gvalue(i,2))<=1 && gvalue(i,3)<=0
        && x(i,1) >= 0 && x(i,2) >= 0 && exam(1,i) < stop) )
        if ( height_m < h_altitude )
            x(1,1)=x(i,1)-5*kk;x(1,2)=x(i,2)+5*kk;
            C(1,1)=0.5;growth=2;stop=0.9;
        else
            renew=2;
            break;
        end
    end
end

```

```

elseif ( ( gvalue(i,1)<=0 || abs(gvalue(i,1))<=1) && abs(gvalue(i,2))<=1 &&
    gvalue(i,3)<=0 && ( x(i,1) <=1 && x(i,1) >=0) && x(i,2) >= 0 )
    if ( height_m < h_altitude )
        x(1,1)=x(i,1)-5*kk;x(1,2)=x(i,2)+5*kk;
        C(1,1)=0.5;growth=2;stop=0.9;
    else
        renew=2;
        break;
    end
elseif ( gvalue(i,1)<=-20 && abs(gvalue(i,2))<=1 && ( x(i,1) <=1 && x(i,1) >=0) &&
    x(i,2) >= 0 )
    renew=2;
    break;
end
if ( (gvalue(i,1)) > 0 )
    if ( (gvalue(i,1)) > 20 )
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+10;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-10;
        end
    elseif ( (gvalue(i,1)) > 10 && (gvalue(i,1)) <= 20 )
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+5;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-5;
        end
    elseif ( (gvalue(i,1)) > 4 && (gvalue(i,1)) <= 10 )
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+3;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-3;
        end
    elseif ( (gvalue(i,1)) > 1 && (gvalue(i,1)) <= 4)
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+2;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-2;
        end
    end
end

```

```

elseif ( (gvalue(i,1)) <= 1)
    if ( x(i,1) < x(i-1,1) )
        x(i,2)=x(i,2)+1;
    elseif ( x(i,1) > x(i-1,1) )
        x(i,2)=x(i,2)-1;
    end
end
elseif ( (gvalue(i,1)) < 0 )
    if ( (gvalue(i,1)) < -20 && abs(gvalue(i,1)) >= 100 )
        if ( x(i,1) <= -10 )
            x(i,2)=x(i,2)-5;
        elseif ( x(i,1) <= -2 && x(i,1) > -10 )
            x(i,2)=x(i,2)-3;
        elseif ( x(i,1) >= -2)
            x(i,2)=x(i,2)-1;
        end
    elseif ( (gvalue(i,1)) < -20 && abs(gvalue(i,1)) < 100 )
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+10;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-10;
        end
    elseif ( (gvalue(i,1)) < -10 && (gvalue(i,1)) >= -20 )
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+5;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-5;
        end
    elseif ( (gvalue(i,1)) < -1 && (gvalue(i,1)) >= -10)
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+2;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-2;
        end
    elseif ( (gvalue(i,1)) >= -1 )
        if ( x(i,1) < x(i-1,1) )
            x(i,2)=x(i,2)+1;
        elseif ( x(i,1) > x(i-1,1) )
            x(i,2)=x(i,2)-1;

```

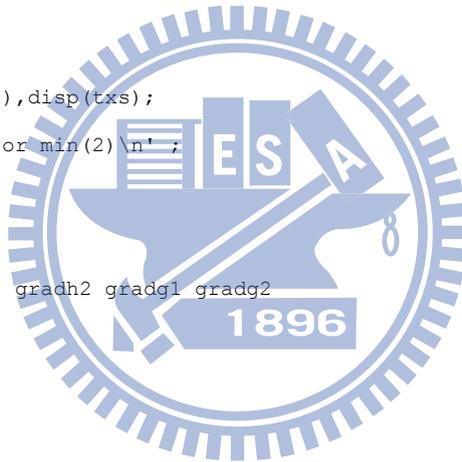

附錄 4 SQP 程式碼

```

%%%===== input =====%%%
% xb (1) 為延遲時間初始值
% xb (2) 為飛行時間初始值
%%%===== output =====%%%
% XX (1, i) 為x1的改變量
% XX (2, i) 為x2的改變量
% XX (3, i) 為新的lamda1
% XX (4, i) 為新的lamda2
% XX (5, i) 為延遲時間值
% XX (6, i) 為飛行時間值
% XX (7, i) 為h等式值
% XX (8, i) 為g不等式值
% XX (9, i) 為目標函數值

fprintf( 'Target Case :' ), disp(txs);
string1 = 'choose max(1) or min(2)\n';
xs = input(string1);
syms x1 x2 f g h
syms gradf1 gradf2 gradh1 gradh2 gradg1 gradg2
i=1;
t1=16;tc=18;t2=36;th=10;
lamda1=0.5;lamda2=0.5;
h_altitude=30;h_r=10; % 初始參數設定
if (xs==1)
    f = -(x1+x2); % 目標函數(max)
    xb(1)=250;
    xb(2)=250;
    h=h_stg3; % 等式限制式
    g=h_altitude-a1_t; % 不等式限制式
elseif (xs==2)
    f = (x1+x2); % 目標函數(min)
    xb(1)=0;
    xb(2)=100;
    h=h_stg3; % 等式限制式
    g=-h_r+g_tm; % 不等式限制式
end

```



```

k=0;
renew=1;
while (renew==1)
clear XX
k=k+1;it=1000;
for i = 1:it
% the gradient functions
gradf1 = diff(f,x1);
gradf2 = diff(f,x2);
gradh1 = diff(h,x1);
gradh2 = diff(h,x2);
gradg1 = diff(g,x1);
gradg2 = diff(g,x2);
% the hessian functions
hessf = [diff(gradf1,x1), diff(gradf1,x2); diff(gradf2,x1), diff(gradf2,x2)];
hessh = [diff(gradh1,x1), diff(gradh1,x2); diff(gradh2,x1), diff(gradh2,x2)];
hessg = [diff(gradg1,x1), diff(gradg1,x2); diff(gradg2,x1), diff(gradg2,x2)];
f1 = subs(f,{x1,x2},{xb(1),xb(2)});
g1 = subs(g,{x1,x2},{xb(1),xb(2)});
h1 = subs(h,{x1,x2},{xb(1),xb(2)});
% the values of the gradient functions and hessian functions
gf1 = double(subs(gradf1,{x1,x2},{xb(1),xb(2)}));
gf2 = double(subs(gradf2,{x1,x2},{xb(1),xb(2)}));
hess1 = double(subs(hessf,{x1,x2},{xb(1),xb(2)}));
gh1 = double(subs(gradh1,{x1,x2},{xb(1),xb(2)}));
gh2 = double(subs(gradh2,{x1,x2},{xb(1),xb(2)}));
gg1 = double(subs(gradg1,{x1,x2},{xb(1),xb(2)}));
gg2 = double(subs(gradg2,{x1,x2},{xb(1),xb(2)}));
g_value=(subs(g,{x1,x2},{xb(1),xb(2)}) > 0);
if (g_value >= 0)
hess=hessf+lamda1*hessh+lamda2*hessg;
A=[subs(hess,{x1,x2},{xb(1),xb(2)}) [gh1;gh2] [gg1;gg2] ;[gh1 gh2 0 0];[gg1 gg2 0 0] ];
B=[-[gf1;gf2];-subs(h,{x1,x2},{xb(1),xb(2)});-subs(g,{x1,x2},{xb(1),xb(2)})];
X=inv(A)*B;
xb(1) = xb(1) + X(1); % x1'= x1+d1
xb(2) = xb(2) + X(2); % x2'= x2+d2
dx1best = X(1);
dx2best = X(2);

```

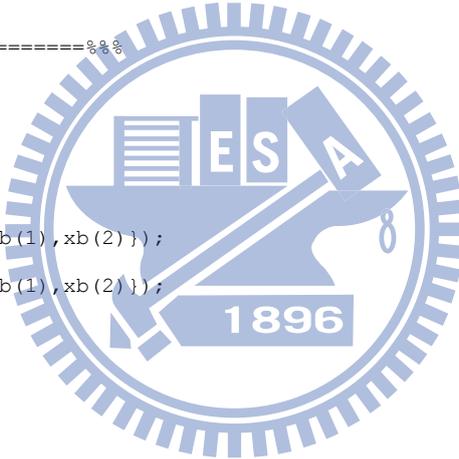
```

else
    hess=hessf+lamda1*hessh;
    A=[subs(hess,{x1,x2},{xb(1),xb(2)}) [gh1;gh2];[gh1 gh2 0]];
    B=[-[gf1;gf2];-subs(h,{x1,x2},{xb(1),xb(2)})];
    X=inv(A)*B;
    xb(1) = xb(1) + X(1);    % x1'= x1+d1
    xb(2) = xb(2) + X(2);    % x2'= x2+d2
    dx1best = X(1);
    dx2best = X(2);
    X(4)=lamda2;
end
lamda1=X(3);
lamda2=X(4);
obj=subs(f,{x1,x2},{xb(1),xb(2)});

%%=====輸出結果=====%%
XX(1:3,i)=X(1:3);
XX(4,i)=X(4);
XX(5:6,i)=[xb(1);xb(2)];
XX(7,i)=subs(h,{x1,x2},{xb(1),xb(2)});
XX(8,i)=subs(g,{x1,x2},{xb(1),xb(2)});
if xs==1
    XX(9,i)=-obj;
else
    XX(9,i)=obj;
End

if xs==1
    if ([dx1best dx2best]*[dx1best dx2best]') <= 1.0e-06
        break;
    end
elseif xs==2
    if ([dx1best dx2best]*[dx1best dx2best]') <= 1.0e-06
        break;
    end
end
end
end

```



```

height_m=al_m; % 檢查飛彈高度
if (xs==1)
    if subs(height_m,{x2},{xb(2)}) < h_altitude || XX(6,i) < t1+tc+t2+th
        xb(1)=xb(1)-5*k;xb(2)=xb(2)+5*k;
        renew=1;
    else
        renew=2;
        break
    end
elseif (xs==2)
    if subs(height_m,{x2},{xb(2)}) < h_altitude || XX(6,i) < t1+tc+t2+th
        xb(1)=xb(1)-5*k;xb(2)=xb(2)+5*k;
        renew=1;
    else
        renew=2;
        break
    end
end
end
end

if (XX(5,i)<=0)
    XX(5,i)=0;
    XXX=real(double(solve(subs(h,{x1},{0})))));
    for j=1:length(XXX)
        if (XXX(j)>0 && XXX(j)<=400)
            XX(6,i)=XXX(j);
        end
    end
end

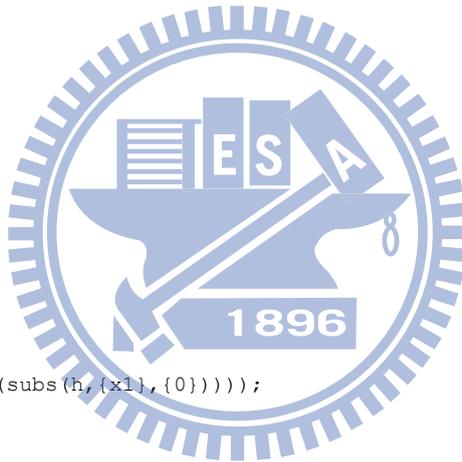
end

clc;

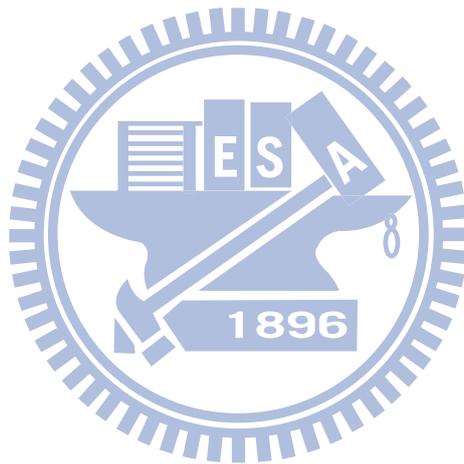
fprintf( 'Target Case :' ),disp(txs);

if (xs==1)
    fprintf('\n =====MAX=====');
    fprintf('\n Iteration Number :'),disp(i);
    fprintf('\n Best Fly Time (tf) :'),disp(XX(6,i));
    fprintf('\n Best Delay Time (td) :'),disp(XX(5,i));
    fprintf('\n Objective Function :'),disp(XX(9,i));
    fprintf('\n =====MAX=====\n');
end

```



```
elseif (xs==2)
    fprintf('\n =====MIN=====');
    fprintf('\n Iteration Number :'),disp(i);
    fprintf('\n Best Fly Time (tf) :'),disp(XX(6,i));
    fprintf('\n Best Delay Time (td) :'),disp(XX(5,i));
    fprintf('\n Objective Function :'),disp(XX(9,i));
    fprintf('\n =====MIN=====\\n');
end
```



附錄 5 SQP 求解不同目標的 LNS

式(5.14)的 LNS 為

$$\begin{bmatrix} \nabla_x^2 L(\mathbf{x}^k, \boldsymbol{\lambda}^k, \boldsymbol{\beta}^k) & \nabla \mathbf{h}(\mathbf{x}^k) & \nabla \mathbf{g}(\mathbf{x}^k) \\ \nabla \mathbf{h}(\mathbf{x}^k)^T & 0 & 0 \\ \nabla \mathbf{g}(\mathbf{x}^k)^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d}^k \\ \boldsymbol{\lambda}^{k+1} \\ \boldsymbol{\beta}^{k+1} \end{bmatrix} = - \begin{bmatrix} \nabla f(\mathbf{x}^k) \\ \mathbf{h}(\mathbf{x}^k) \\ \mathbf{g}(\mathbf{x}^k) \end{bmatrix}$$

根據不同來襲目標可以求出不同的 $h(\mathbf{x})=0$ ， $g(\mathbf{x})\leq 0$ 及 $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta})$ 。

1. 求解目標一最大值

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{h}(\mathbf{x}) = 1.6149 \cdot 10^6 - 6894.9x_2 + 5.9806x_2^2 - 0.0588x_1^2x_2 - 0.0588x_2^2x_1 - 0.0196x_1^3 - 7.6242 \cdot 10^{-4}x_2^3 + 9.6040 \cdot 10^{-5}x_1^3x_2 \\ + 1.4406 \cdot 10^{-4}x_1^2x_2^2 + 9.6040 \cdot 10^{-5}x_2^3x_1 + 24.2320x_1x_2 + 2.4010 \cdot 10^{-5}x_1^4 - 7260x_1 + 12.1160x_1^2$$

$$\mathbf{g}(\mathbf{x}) = -2x_2 - 2x_1 + 0.0049x_2^2 + 0.0098x_1x_2 + 0.0049x_1^2$$

$$\nabla \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$h_1 = -0.1176x_1x_2 + 9.6040 \cdot 10^{-5}x_2^3 + 2.8812 \cdot 10^{-4}x_2^2x_1 + 2.8812 \cdot 10^{-4}x_1^2x_2 \\ - 0.0588x_2^2 - 0.0588x_1^2 + 9.6040 \cdot 10^{-5}x_1^3 + 24.2320x_2 - 7260 + 24.2320x_1$$

$$h_2 = -6894.9 - 0.0588x_1^2 + 2.8812 \cdot 10^{-4}x_2^2x_1 + 2.8812 \cdot 10^{-4}x_1^2x_2 + 9.6040 \cdot 10^{-5}x_1^3 \\ - 0.1176x_1x_2 - 0.0023x_2^2 + 24.2320x_1 + 11.9611x_2$$

$$\nabla \mathbf{g}(\mathbf{x}) = \begin{bmatrix} -2 + 0.0098x_1 + 0.0098x_2 \\ -2 + 0.0098x_1 + 0.0098x_2 \end{bmatrix}$$

$$\nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta}) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$$

$$h_{11} = -0.1176\lambda x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 - 0.1176\lambda x_1 + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 24.2320\lambda + 0.0098\beta$$

$$h_{12} = -0.1176\lambda x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 - 0.1176\lambda x_1 + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 24.2320\lambda + 0.0098\beta$$

$$h_{21} = -0.1176\lambda x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 - 0.1176\lambda x_1 + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 24.2320\lambda + 0.0098\beta$$

$$h_{22} = 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_1^2 - 0.1176\lambda x_1 - 0.0046\lambda x_2 + 11.9611\lambda + 0.0098\beta$$

如此一來，將可利用 LNS 順利求出 \mathbf{d}^k ， $\boldsymbol{\lambda}^{k+1}$ 及 $\boldsymbol{\beta}^{k+1}$ 。

2. 求解目標一最小值

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{h}(\mathbf{x}) = 1.6149 \cdot 10^6 - 6894.9x_2 + 5.9806x_2^2 - 0.0588x_1^2x_2 - 0.0588x_2^2x_1 - 0.0196x_1^3 - 7.6242 \cdot 10^{-4}x_2^3 + 9.6040 \cdot 10^{-5}x_1^3x_2 \\ + 1.4406 \cdot 10^{-4}x_1^2x_2^2 + 9.6040 \cdot 10^{-5}x_2^3x_1 + 24.2320x_1x_2 + 2.4010 \cdot 10^{-5}x_1^4 - 7260x_1 + 12.1160x_1^2$$

$$\mathbf{g}(\mathbf{x}) = 79.6039 + 0.0778x_2 + 2x_1 - 0.0098x_1x_2 - 0.0049x_1^2$$

$$\nabla \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$h_1 = -0.1176x_1x_2 + 9.6040 \cdot 10^{-5}x_2^3 + 2.8812 \cdot 10^{-4}x_2^2x_1 + 2.8812 \cdot 10^{-4}x_1^2x_2 \\ - 0.0588x_2^2 - 0.0588x_1^2 + 9.6040 \cdot 10^{-5}x_1^3 + 24.2320x_2 - 7260 + 24.2320x_1$$

$$h_2 = -6894.9 - 0.0588x_1^2 + 2.8812 \cdot 10^{-4}x_2^2x_1 + 2.8812 \cdot 10^{-4}x_1^2x_2 + 9.6040 \cdot 10^{-5}x_1^3 \\ - 0.1176x_1x_2 - 0.0023x_2^2 + 24.2320x_1 + 11.9611x_2$$

$$\nabla \mathbf{g}(\mathbf{x}) = \begin{bmatrix} 2 - 0.0098x_1 - 0.0098x_2 \\ 0.0778 - 0.0098x_1 \end{bmatrix}$$

$$\nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta}) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$$

$$h_{11} = -0.1176\lambda x_2 - 0.1176\lambda x_1 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 24.2320\lambda + 2.8812 \cdot 10^{-4}\lambda x_1^2 - 0.0098\beta \\ h_{12} = -0.1176\lambda x_2 - 0.1176\lambda x_1 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 24.2320\lambda + 2.8812 \cdot 10^{-4}\lambda x_1^2 - 0.0098\beta \\ h_{21} = -0.1176\lambda x_2 - 0.1176\lambda x_1 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 24.2320\lambda + 2.8812 \cdot 10^{-4}\lambda x_1^2 - 0.0098\beta \\ h_{22} = 11.9611\lambda - 0.1176\lambda x_1 - 0.0046\lambda x_2 + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 5.7624 \cdot 10^{-4}\lambda x_1x_2$$

如此一來，將可利用 LNS 順利求出 \mathbf{d}^k ， $\boldsymbol{\lambda}^{k+1}$ 及 $\boldsymbol{\beta}^{k+1}$ 。

3. 求解目標二最大值

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{h}(\mathbf{x}) = 1.0615 \cdot 10^6 - 4936.9x_2 + 2.9806x_2^2 - 0.0470x_1^2x_2 - 0.0470x_2^2x_1 - 0.0157x_1^3 + 0.0032x_2^3 + 9.6040 \cdot 10^{-5}x_1^3x_2 \\ + 1.4406 \cdot 10^{-4}x_1^2x_2^2 + 9.6040 \cdot 10^{-5}x_2^3x_1 + 18.2320x_1x_2 + 2.4010 \cdot 10^{-5}x_1^4 - 5302x_1 + 9.1160x_1^2$$

$$\mathbf{g}(\mathbf{x}) = -1.6x_2 - 1.6x_1 + 0.0049x_2^2 + 0.0098x_1x_2 + 0.0049x_1^2$$

$$\nabla \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$\begin{aligned}
h_1 &= -0.0941x_1x_2 - 0.0470x_2^2 - 0.0470x_1^2 + 18.2320x_2 + 2.8812 \cdot 10^{-4}x_1^2x_2 \\
&\quad + 2.8812 \cdot 10^{-4}x_2^2x_1 + 9.6040 \cdot 10^{-5}x_2^3 - 5302 + 18.2320 * x_1 + 9.6040 \cdot 10^{-5}x_1^3 \\
h_2 &= -4936.9 + 5.9611x_2 - 0.0470x_1^2 - 0.0941x_1x_2 + 0.0095x_2^2 + 18.2320x_1 \\
&\quad + 9.6040 \cdot 10^{-5}x_1^3 + 2.8812 \cdot 10^{-4}x_1^2x_2 + 2.8812 \cdot 10^{-4}x_2^2x_1
\end{aligned}$$

$$\nabla \mathbf{g}(\mathbf{x}) = \begin{bmatrix} -1.6 + 0.0098x_1 + 0.0098x_2 \\ -1.6 + 0.0098x_1 + 0.0098x_2 \end{bmatrix}$$

$$\nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta}) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$$

$$\begin{aligned}
h_{11} &= -0.0941\lambda x_2 - 0.0941\lambda x_1 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 18.2320\lambda + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 0.0098\beta \\
h_{12} &= -0.0941\lambda x_2 - 0.0941\lambda x_1 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 18.2320\lambda + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 0.0098\beta \\
h_{21} &= -0.0941\lambda x_2 - 0.0941\lambda x_1 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 2.8812 \cdot 10^{-4}\lambda x_2^2 + 18.2320\lambda + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 0.0098\beta \\
h_{22} &= 5.9611\lambda - 0.0941\lambda x_1 + 0.0189\lambda x_2 + 2.8812 \cdot 10^{-4}\lambda x_1^2 + 5.7624 \cdot 10^{-4}\lambda x_1x_2 + 0.0098\beta
\end{aligned}$$

如此一來，將可利用 LNS 順利求出 \mathbf{d}^k ， $\boldsymbol{\lambda}^{k+1}$ 及 $\boldsymbol{\beta}^{k+1}$ 。

4. 求解目標二最小值

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{aligned}
\mathbf{h}(\mathbf{x}) &= 1.0615 \cdot 10^6 - 4936.9x_2 + 2.9806x_2^2 - 0.0470x_1^2x_2 - 0.0470x_2^2x_1 - 0.0157x_1^3 + 0.0032x_2^3 + 9.6040 \cdot 10^{-5}x_1^3x_2 \\
&\quad + 1.4406 \cdot 10^{-4}x_1^2x_2^2 + 9.6040 \cdot 10^{-5}x_2^3x_1 + 18.2320x_1x_2 + 2.4010 \cdot 10^{-5}x_1^4 - 5302x_1 + 9.1160x_1^2
\end{aligned}$$

$$\mathbf{g}(\mathbf{x}) = 79.6039 - 0.3222x_2 + 1.6x_1 - 0.0098x_1x_2 - 0.0049x_1^2$$

$$\nabla \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$\begin{aligned}
h_1 &= -0.0941x_1x_2 - 0.0470x_2^2 - 0.0470x_1^2 + 18.2320x_2 + 2.8812 \cdot 10^{-4}x_1^2x_2 \\
&\quad + 2.8812 \cdot 10^{-4}x_2^2x_1 + 9.6040 \cdot 10^{-5}x_2^3 - 5302 + 18.2320 * x_1 + 9.6040 \cdot 10^{-5}x_1^3 \\
h_2 &= -4936.9 + 5.9611x_2 - 0.0470x_1^2 - 0.0941x_1x_2 + 0.0095x_2^2 + 18.2320x_1 \\
&\quad + 9.6040 \cdot 10^{-5}x_1^3 + 2.8812 \cdot 10^{-4}x_1^2x_2 + 2.8812 \cdot 10^{-4}x_2^2x_1
\end{aligned}$$

$$\nabla \mathbf{g}(\mathbf{x}) = \begin{bmatrix} 1.6 - 0.0098x_1 - 0.0098x_2 \\ -0.3222 - 0.0098x_1 \end{bmatrix}$$

$$\nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta}) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$$

$$\begin{aligned} h_{11} &= -0.0941\lambda x_2 - 0.0941\lambda x_1 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 2.8812 \cdot 10^{-4} \lambda x_2^2 + 18.2320\lambda + 2.8812 \cdot 10^{-4} \lambda x_1^2 - 0.0098\beta \\ h_{12} &= -0.0941\lambda x_2 - 0.0941\lambda x_1 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 2.8812 \cdot 10^{-4} \lambda x_2^2 + 18.2320\lambda + 2.8812 \cdot 10^{-4} \lambda x_1^2 - 0.0098\beta \\ h_{21} &= -0.0941\lambda x_2 - 0.0941\lambda x_1 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 2.8812 \cdot 10^{-4} \lambda x_2^2 + 18.2320\lambda + 2.8812 \cdot 10^{-4} \lambda x_1^2 - 0.0098\beta \\ h_{22} &= 5.9611\lambda - 0.0941\lambda x_1 + 0.0189\lambda x_2 + 2.8812 \cdot 10^{-4} \lambda x_1^2 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 \end{aligned}$$

如此一來，將可利用 LNS 順利求出 \mathbf{d}^k ， $\boldsymbol{\lambda}^{k+1}$ 及 $\boldsymbol{\beta}^{k+1}$ 。

5. 求解目標三最大值

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= 2.0427 \cdot 10^5 - 1659.5x_2 - 1.8760x_2^2 - 0.0029x_1^2x_2 - 0.0029x_2^2x_1 - 9.8000 \cdot 10^{-4}x_1^3 + 0.0179x_2^3 + 9.6040 \cdot 10^{-5}x_1^3x_2 \\ &\quad + 1.4406 \cdot 10^{-4}x_1^2x_2^2 + 9.6040 \cdot 10^{-5}x_2^3x_1 + 8.5188x_1x_2 + 2.4010 \cdot 10^{-5}x_1^4 - 2024.6x_1 + 4.2594x_1^2 \end{aligned}$$

$$\mathbf{g}(\mathbf{x}) = -67 - 0.1x_2 - 0.1x_1 + 0.0049x_2^2 + 0.0098x_1x_2 + 0.0049x_1^2$$

$$\nabla \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$\begin{aligned} h_1 &= -0.0059x_1x_2 - 0.0029x_2^2 - 0.0029x_1^2 + 2.8812 \cdot 10^{-4}x_1^2x_2 + 2.8812 \cdot 10^{-4}x_2^2x_1 \\ &\quad + 9.6040 \cdot 10^{-5}x_2^3 + 8.5188x_2 + 9.6040 \cdot 10^{-5}x_1^3 - 2024.6 + 8.5188x_1 \end{aligned}$$

$$\begin{aligned} h_2 &= -1659.5 - 3.7521x_2 - 0.0029x_1^2 - 0.0059x_1x_2 + 0.0536x_2^2 + 9.6040 \cdot 10^{-5}x_1^3 \\ &\quad + 2.8812 \cdot 10^{-4}x_1^2x_2 + 2.8812 \cdot 10^{-4}x_2^2x_1 + 8.5188x_1 \end{aligned}$$

$$\nabla \mathbf{g}(\mathbf{x}) = \begin{bmatrix} -0.1 + 0.0098x_1 + 0.0098x_2 \\ -0.1 + 0.0098x_1 + 0.0098x_2 \end{bmatrix}$$

$$\nabla_x^2 L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\beta}) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$$

$$\begin{aligned} h_{11} &= -0.0059\lambda x_2 - 0.0059\lambda x_1 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 2.8812 \cdot 10^{-4} \lambda x_2^2 + 2.8812 \cdot 10^{-4} \lambda x_1^2 + 8.5188\lambda + 0.0098\beta \\ h_{12} &= -0.0059\lambda x_2 - 0.0059\lambda x_1 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 2.8812 \cdot 10^{-4} \lambda x_2^2 + 2.8812 \cdot 10^{-4} \lambda x_1^2 + 8.5188\lambda + 0.0098\beta \\ h_{21} &= -0.0059\lambda x_2 - 0.0059\lambda x_1 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 2.8812 \cdot 10^{-4} \lambda x_2^2 + 2.8812 \cdot 10^{-4} \lambda x_1^2 + 8.5188\lambda + 0.0098\beta \\ h_{22} &= -3.7521\lambda - 0.0059\lambda x_1 + 0.1071\lambda x_2 + 2.8812 \cdot 10^{-4} \lambda x_1^2 + 5.7624 \cdot 10^{-4} \lambda x_1 x_2 + 0.0098\beta \end{aligned}$$

如此一來，將可利用 LNS 順利求出 \mathbf{d}^k ， $\boldsymbol{\lambda}^{k+1}$ 及 $\boldsymbol{\beta}^{k+1}$ 。