

102-130 -1340
MOTC-IOT-101-PBA032

臺鐵系統連鎖延滯之可靠度 模式分析



交通部運輸研究所

中華民國 102 年 7 月

102-130 -1340
MOTC-IOT-101-PBA032

臺鐵系統連鎖延滯之可靠度 模式分析

著者：劉昭榮、張恩輔

交通部運輸研究所

中華民國 102 年 7 月

國家圖書館出版品預行編目(CIP)資料

臺鐵系統連鎖延滯之可靠度模式分析 / 劉昭榮, 張恩輔著. -- 初版. -- 臺北市 : 交通部運研所, 民 102. 07

面 ; 公分
ISBN 978-986-03-7447-6(平裝)

1. 鐵路管理 2. 運輸系統

557

102013264

臺鐵系統連鎖延滯之可靠度模式分析

著 者：劉昭榮、張恩輔

出版機關：交通部運輸研究所

地 址：10548 臺北市敦化北路 240 號

網 址：www.iot.gov.tw (中文版>圖書服務>本所出版品)

電 話：(02)23496789

出版年月：中華民國 102 年 7 月

印 刷 者：安頡企業社

版(刷)次冊數：初版一刷 50 冊

本書同時登載於交通部運輸研究所網站

定 價：70 元

展 售 處：

交通部運輸研究所運輸資訊組・電話：(02)23496880

國家書店松江門市：10485 臺北市中山區松江路 209 號・電話：(02)25180207

五南文化廣場：40042 臺中市中山路 6 號・電話：(04)22260330

GPN：1010201337

ISBN：978-986-03-7447-6 (平裝)

著作財產權人：中華民國（代表機關：交通部運輸研究所）

本著作保留所有權利，欲利用本著作全部或部分內容者，須徵求交通部運輸研究所書面授權。

交通部運輸研究所合作研究第 1 類計畫出版品摘要表

出版品名稱：臺鐵系統連鎖延滯之可靠度模式分析			
國際標準書號（或叢刊號） ISBN978-986-03-7447-6(平裝)	政府出版品統一編號 1010201337	運輸研究所出版品編號 102-130-1340	計畫編號 101-PBA032
本所主辦單位：運輸計畫組 主管：蘇振維 計畫主持人：劉昭榮 研究人員：劉昭榮 聯絡電話：(02)23496809 傳真號碼：(02)25450428		合作研究單位： 計畫主持人： 研究人員：張恩輔 地址：臺北市南京東路 5 段 171 號 聯絡電話：(02)27692131 ext. 20976 傳真號碼：(02)27655010	
研究期間 自 101 年 1 月 至 101 年 12 月			
關鍵詞：隨機特性、初始延滯、連鎖延滯、衝擊波			
<p>摘要：</p> <p>鑑於影響連鎖延滯之部分關鍵因素具有隨機特性，為確實反映臺鐵列車因初始延滯發生後所產生之連鎖延滯，及深入探討產生連鎖延滯之班表穩定度問題，本研究以七堵—新竹路段為案例深入分析站間運轉時間及停站時間之隨機特性，除蒐集該路段對號列車及通勤電聯車之站間運轉時間資料，並依尖、離峰時段分開彙整停站時間之隨機資料，據以應用所構建之模擬模式推估其連鎖延滯。研究結果顯示，無論中間車站或末端車站（七堵及新竹站），其尖峰或離峰時段之連鎖延滯模擬結果皆會收斂。本研究所提出之分析架構及方法，有助於釐清連鎖延滯關鍵影響因素及其影響程度，所構建之模擬模式可作為相關改善策略之分析工具，研究成果則可作為營運單位排班規劃、系統可靠度分析及服務品質改善參考。</p> <p>另有關站間容許列車數對連鎖延滯之敏感度之分析結果證實，當站間容許列車數由 1 提昇至 2 時，總延滯時間之降幅最大亦即系統可靠度大幅提昇，此係由整個路段之站間閉塞區間及站內股道配置狀況共同決定之路線容量及可靠度結果。另有關 2 次初始延滯對連鎖延滯之衝擊波分析，研究結果顯示若選擇南港站之連鎖延滯進行分析，並假設上午 7:10 於南下松山—台北間發生 1 小時之第 1 次初始延滯，當第 2 次初始延滯於該區間發生之時間夠接近時，其對連鎖延滯之衝擊波將產生交互作用影響，且當 2 次初始延滯發生時間（即雙峰）逐漸靠近至 1.4 小時，雙峰就合併成單峰且峰度越高，顯示 2 次初始延滯之影響發生加成效果，並大幅衝擊系統的可靠度。</p>			
出版日期	頁數	定價	本出版品取得方式
102 年 7 月	91	70	凡屬機密性出版品均不對外公開。普通性出版品，公營、公益機關團體及學校可函洽本所免費贈閱；私人及私營機關團體可按定價價購。
<p>機密等級：</p> <p><input type="checkbox"/>密 <input type="checkbox"/>機密 <input type="checkbox"/>極機密 <input type="checkbox"/>絕對機密</p> <p>（解密條件：<input type="checkbox"/> 年 月 日解密，<input type="checkbox"/>公布後解密，<input type="checkbox"/>附件抽存後解密， <input type="checkbox"/>工作完成或會議終了時解密，<input type="checkbox"/>另行檢討後辦理解密）</p> <p><input checked="" type="checkbox"/>普通</p>			
備註：本研究之結論與建議不代表交通部之意見。			

PUBLICATION ABSTRACTS OF RESEARCH PROJECTS
INSTITUTE OF TRANSPORTATION
MINISTRY OF TRANSPORTATION AND COMMUNICATIONS

TITLE: Analyzing the Reliability Analysis Model for the Stochastic Knock-on Delays of Taiwan Railways Administration System			
ISBN(OR ISSN) ISBN978-986-03-7447-6(pbk.)	GOVERNMENT PUBLICATIONS NUMBER 1010201337	IOT SERIAL NUMBER 102-130-1340	PROJECT NUMBER 101-PBA032
DIVISION: Planning Division DIVISION CHIEF: Cheng-Wei Su PRINCIPAL INVESTIGATOR: Jau-Rong Liu PROJECT STAFF: Jau-Rong Liu PHONE: 886-2-23496809 FAX: 886-2-25450428			PROJECT PERIOD FROM January 2012 TO December 2012
RESEARCH AGENCY: PRINCIPAL INVESTIGATOR: PROJECT STAFF: En-Fu Chang ADDRESS: 171 Nanking E. RD. SEC. 5, Taipei, Taiwan, R.O.C. PHONE: 886-2-27692131 ext. 20976 FAX: 886-2-27655010			
KEY WORDS: Stochastic nature, First delay, Knock-on delay, Shock waves			
ABSTRACT: <p>Based on the stochastic features concerning the key factors influencing knock-on delays, this project endeavored to correctly reflect the knock on delays that are produced by the initial delays of Taiwan Railways Administration (TRA) trains and investigate schedule stability issues that are caused by knock-on delays. To achieve these goals, the Cidu-Hsinchu section of the TRA was selected as the research site to analyze stochastic features of between-station and in-station operation times. In addition to collecting data concerning the between-station operational times of express and commuter trains, the in-station stochastic data for these trains were also categorized according to peak and off-peak periods and analyzed. A comprehensive simulation model was developed and employed to process and evaluate the knock-on delays. The results suggest that the knock-on delays at all stations and two end stations during peak and off-peak hours converged to constant values. Therefore, this research concludes that the framework and the simulation model proposed in this research may be helpful for identifying the key factors affecting knock-on delays, but also the degree of delay impact. The results are also beneficial for timetable scheduling, analyzing system reliability, and improving service quality.</p> <p>Based on the results concerning the tolerated number of trains between stations and their sensitivity towards knock-on delays, increasing the tolerated number of trains between stations from 1 to 2 demonstrated the greatest reduction in overall delay times; that is, the greatest increase in system reliability. These results were consequently based on the route capacity and sensitivity results of between-station block sections and in-station track configuration conditions of all railway sections. Furthermore, this project performed an additional double-delay test to analyze the impact of initial delays on knock-on delays. In this test, the knock-on delay of Nangang Train Station was observed. This test assumed that the first initial delay event (duration=1h) occurred along the Songshan-Taipei section on the southbound track at 7:10 a.m. When the second initial delay event occurred in an appropriate interval to the first event, the impact of the knock-on delay generates an interactive effect linking the two events. When the time of the second initial delay event (i.e., dual-peak) reaches 1.4 h, the dual peaks converge into a single peak with a higher extreme. These results suggest the influence of the second initial delay was even amplified and significantly impacted the reliability of the system.</p>			
DATE OF PUBLICATION July 2013	NUMBER OF PAGES 91	PRICE 70	CLASSIFICATION <input type="checkbox"/> RESTRICTED <input type="checkbox"/> CONFIDENTIAL <input type="checkbox"/> SECRET <input type="checkbox"/> TOP SECRET <input checked="" type="checkbox"/> UNCLASSIFIED
The views expressed in this publication are not necessarily those of the Ministry of Transportation and Communications.			

目 錄

第一章 緒論	1
1.1 研究緣起	1
1.2 研究目的	2
1.3 研究範圍與對象	3
1.4 研究內容及流程	3
第二章 文獻回顧與評析	5
2.1 軌道列車延滯發生原因分析	5
2.2 班表可靠度與延滯關係及分析方法	5
2.3 列車運行條件與延滯關係	7
2.4 運轉整理與連鎖延滯擴散問題	7
2.5 綜合評析	8
第三章 臺鐵列車連鎖延滯模擬模式	11
3.1 模式架構	11
3.2 模式假設與限制	13
3.3 模擬機制	14
3.4 模擬程式設計	16
3.5 模擬模式驗證	21
第四章 案例分析	23
4.1 資料蒐集分析	23
4.2 延滯特性參數校估分析	24
4.3 延滯特性參數分布分析	25
4.4 隨機特性連鎖延滯模擬結果	33
4.5 「站間容許列車數」連鎖延滯敏感度分析	36
4.6 二次初始延滯對連鎖延滯之衝擊波分析	39
第五章 結論與建議	45
5.1 結論	45
5.2 建議	46
參考文獻	49
附錄 程式碼及模擬結果說明	53

圖目錄

圖 1-1 研究流程圖.....	4
圖 3-1 模式架構圖.....	12
圖 3-2 型 I 車站軌道佈設.....	13
圖 3-3 型 II 車站軌道佈設.....	13
圖 3-4 型 III_R 車站軌道佈設.....	14
圖 3-5 型 III_L 車站軌道佈設.....	14
圖 3-6 型 VI 車站軌道佈設.....	14
圖 3-7 同向列車離一到時隔示意圖.....	15
圖 3-8 同向列車到一到時隔示意圖.....	15
圖 3-9 同向列車離一離時隔示意圖.....	16
圖 3-10 型 III 月臺之平面交叉時隔示意圖.....	16
圖 3-11 類別設計架構圖.....	18
圖 3-12 StationII 型車站股道之平面交叉路徑設定.....	18
圖 3-13 StationIII_R 型車站股道之平面交叉路徑設定.....	19
圖 3-14 StationIII_L 型車站股道之平面交叉路徑設定.....	19
圖 3-15 連鎖延滯之模擬流程圖.....	20
圖 3-16 受影響列車樣本之模擬延滯與實際延滯差異分布圖.....	22
圖 4-1 對號列車之站間運轉時間分布圖.....	28
圖 4-2 通勤電聯車之站間運轉時間分布圖.....	29
圖 4-3 對號列車於尖峰時段之停站時間分布圖.....	30
圖 4-4 對號列車於離峰時段之停站時間分布圖.....	31
圖 4-5 通勤電聯車於尖峰時段之停站時間分布圖.....	32
圖 4-6 通勤電聯車於離峰時段之停站時間分布圖.....	33
圖 4-7 所有車站之連鎖延滯模擬值分布及收斂情形.....	34
圖 4-8 七堵及新竹二端末車站之連鎖延滯模擬值分布及收斂情形.....	35
圖 4-9 站間容許列車數搭配不同調度策略之所有站總延滯比較圖.....	36
圖 4-10 站間容許列車數搭配不同調度策略之最末站總延滯比較圖.....	37
圖 4-11 站間容許列車數搭配不同程度初始延滯之所有站總延滯比較.....	38
圖 4-12 站間容許列車數搭配不同程度初始延滯之最末站總延滯比較.....	38
圖 4-13 不同調度策略下之衝擊波比較圖.....	39
圖 4-14 不同程度初始延滯下的衝擊波比較.....	40
圖 4-15 不同程度初始延滯下的衝擊波比較（局部放大）.....	40
圖 4-16 不同車站之衝擊波比較.....	41
圖 4-17 不同車站衝擊波比較（局部放大）.....	41
圖 4-18 初始延滯間隔 8 小時之衝擊波.....	42
圖 4-19 各種初始延滯間隔值(2~9hr)之衝擊波.....	43
圖 4-20 各種初始延滯間隔值(1~2hr)之衝擊波.....	43

表 目 錄

表 3.1 基本輸入資料表.....	12
表 3.2 初始延滯之設定參數表.....	12
表 3.3 各種不同事件的衝突檢查項目.....	20
表 3.4 受影響列車樣本之模擬延滯與實際延滯差異統計表.....	22
表 4.1 參數隨機樣本資料分布統計.....	27

第一章 緒論

1.1 研究緣起

本所於 99~101 年度與財團法人中興工程顧問社合作辦理「軌道系統容量與可靠度分析」系列研究計畫，初步已將臺鐵系統連續路段之容量分析模式、軟體及連續路段之延滯可靠度分析模式研析完成，並期能完整構建區域鐵路系統(臺鐵系統)之容量解析模式及構建開發操作軟體。

軌道運輸系統的可靠度攸關運輸服務品質，而其中列車服務的可靠度更是受到社會的期望，列車延滯即為衡量服務可靠度的重要指標之一，國外學者 Rietveld et al. (2001)曾提出下列幾種指標以作為評估軌道系統可靠度的準則，包括：(1)準點率；(2)列車提早離站的機率；(3)實際到達時間與表定到達時間的差異，也就是延滯時間；(4)當一列車延滯時，其續行列車的平均延滯時間；(5)當一列車延滯超過某一時間，對續行列車所造成的平均延滯時間；(6)到達時間的標準差；(7)總旅客延滯小時/旅次數；(8)準點旅次數/旅次數；(9)延滯的列車數等，由上述評估指標亦可顯示「延滯」對軌道系統營運是否正常及服務品質之良窳扮演關鍵重要角色。

列車延滯依其類型可分為「列車排班的交會待避延誤 (waiting time due to scheduled meeting and overtaking)」以及「列車實際運行的延滯 (delay in current operation)」兩類 (鍾志成，2005)，前者係列車排班作業過程中，列車因排除衝突所需額外增加的停等時間，亦稱為排班延滯 (scheduled delay)，通常是用來分析時刻表容量；而後者則是列車實際運行過程中所發生的延滯，亦稱為非排班延滯 (unscheduled delay)，一般列車服務可靠度所討論的延滯通常是指後者。若細究延滯發生的原因，則列車實際運行延滯又可分為初始延滯 (first delay/primary delay/exogenous delay) 和連鎖延滯 (secondary delay/knock-on delay) 兩類 (Olsson et al., 2004)。初始延滯是由於外在

因素直接影響列車運行所導致的延滯，而連鎖延滯則是由於初始延滯的發生，造成列車間相互影響所導致的延滯，例如列車因旅客量過多導致停站時間過長，或列車車輛故障及事故所導致之列車停等，皆為初始延滯；而因初始延滯導致其續行列車必須機外停車，便是連鎖延滯。

有鑑於連鎖延滯擴散效應對於臺鐵系統正常營運之影響至為關鍵，且連鎖延滯牽涉之影響因素又相當錯綜複雜，而過去相關研究主要著重在以平均延滯角度分析延滯影響因素，或就特定路段（如車站）小範圍作各項延滯影響程度之解析式分析，尚無就整體軌道系統影響連鎖延滯之各項條件因素進行綜合性分析，即使探討連鎖延滯之研究亦僅就單一車站範圍構建解析模式進行影響因素之交互作用分析，實難一窺連鎖延滯議題全貌並完全反映實際列車運作及延滯之交互影響現象。故本研究除嘗試有效釐清影響連鎖延滯之關鍵因子、交互作用及各種情境條件所產生連鎖延滯之影響程度，更希望透過本研究建構連鎖延滯分析推估模式，以作為後續理論研究與實務應用之參考。

1.2 研究目的

前述各年度研究主要係針對臺鐵系統容量及可靠度模式從規劃面進行實務應用研析，但有關營運面連鎖延滯可靠度部分之議題，因礙於經費及時間因素，並未列為該系列研究之工作項目，惟鑑於「延滯」議題尤其因外生初始延滯所衍生之連鎖延滯問題，一直是營運單位臺鐵局最困擾及急需因應解決之課題，實務上由於營運單位無法有效精準掌握影響連鎖延滯之關鍵因素，皆於列車排班時以運轉寬裕時間或於發生營運特殊突發事件時，以趕點時間來降低延滯之衝擊，故本研究除欲釐清連鎖延滯之影響因素及其影響程度，更希望能進一步有效釐清影響容量之路線、交通及控制條件對連鎖延滯之交互作用影響，以協助營運單位更精準掌握系統之可靠度，爰辦理本研究。

1.3 研究範圍與對象

為確實反映部分參數資料於實際營運之隨機特性，本研究主要目的係針對列車延滯影響最鉅之站間運轉時間及停站時間進行隨機特性分析，再將該些隨機資料納入模擬模式推估連鎖延滯。由於臺鐵系統之列車種類甚多、特性各異，故本研究將其概分為對號列車及通勤電聯車二類，蒐集所分析路段之延滯資料，並進行二類列車之各別站間運轉時間及停站時間的隨機資料分布分析。其中由於各站之停站時間資料分布變異頗大，故本研究除先將資料蒐集時段內過年及地震等異常事件易造成列車延滯之樣本去除外，為利分析特性一致並進一步將該資料分類為尖、離峰分布資料，再進行後續之連鎖延滯推估。

1.4 研究內容及流程

本研究利用自行建構之模擬模式，考量列車流量、車種組成、站間運轉時間以及號誌時距等因素，分析釐清不同影響因素對連鎖延滯之影響程度，及其與路線、交通及控制等條件之交互作用關係；另為分析可靠度問題，本研究亦將影響列車營運關鍵之站間運轉時間(running time)及停站時間(dwell time)的隨機特性充分納入模擬模式進行分析。綜上，本研究之各項研究內容主要包括：隨機模式構建、連鎖延滯隨機資料之處理、連鎖延滯分析應用程式開發、連鎖延滯分析應用程式測試、運用連鎖延滯分析應用程式執行案例分析(含「站間容許列車數」敏感度分析、二次 First Delay 之衝擊波分析)。至於本研究之研究流程如下圖 1-1 所示。

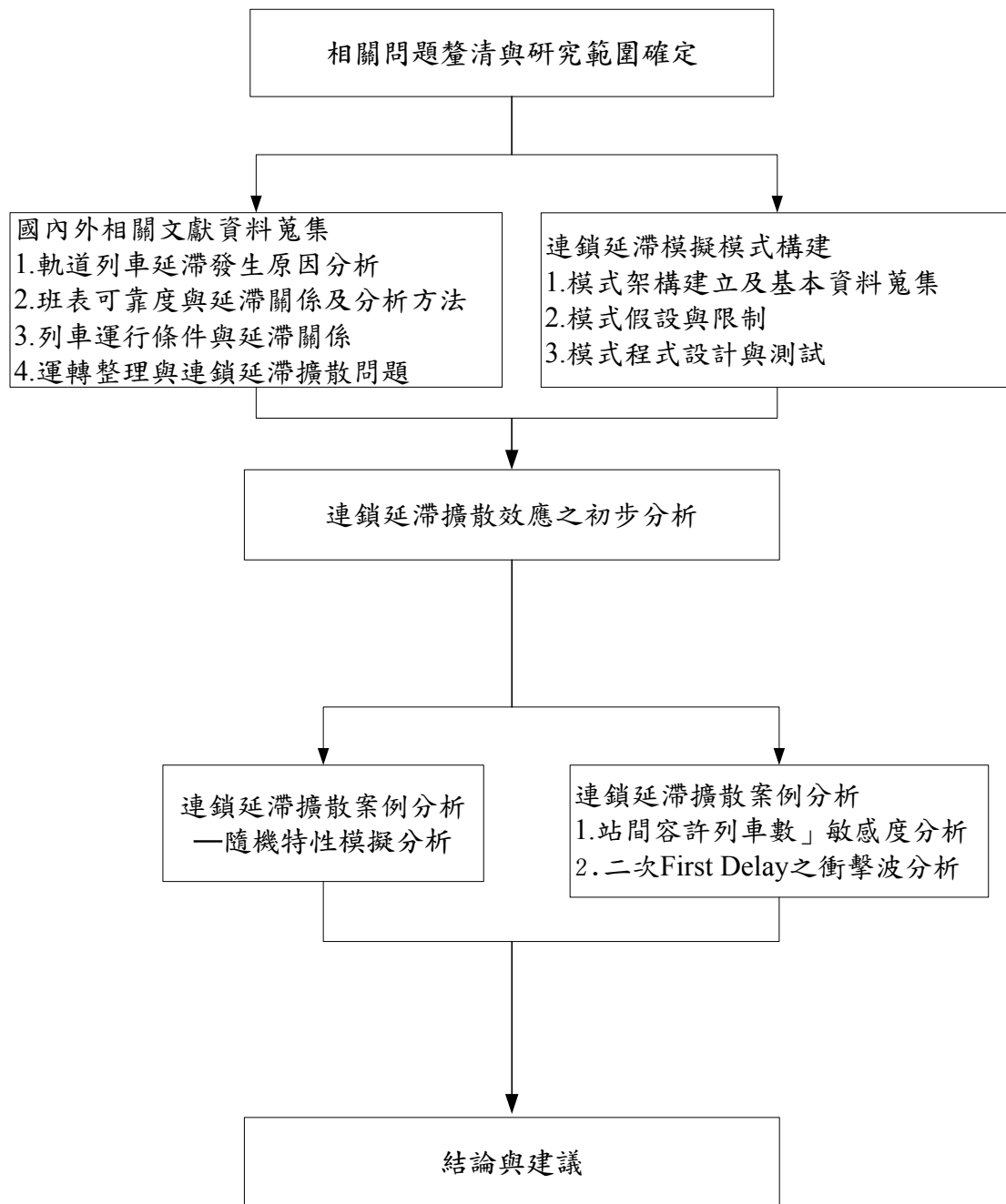


圖 1-1 研究流程圖

第二章 文獻回顧與評析

為探究臺鐵系統延滯(尤其連鎖延滯)之各項複雜問題，並尋求連鎖延滯之可行改善策略，需先有效釐清延滯之類型及其影響因素。因此本研究首先就與列車延滯關係密切之班表可靠度、列車運行條件、路線容量等要項之相關文獻進行回顧，接續再就列車延滯及服務可靠度之分析方法，與運轉調度策略對列車延滯改善分析之相關文獻進行探討，以完整掌握連鎖延滯分析之理論架構及分析方法之邏輯，以作為本研究後續各項研析內容之參考基礎。茲將上述各類相關文獻之回顧結果彙整分述如下：

2.1 軌道列車延滯發生原因分析

Jong *et al.* (2010) 曾以臺灣高鐵系統為例，進一步套用風險矩陣的概念進行延滯分析，依不同延滯原因分別計算各種延滯發生之頻率與嚴重度。該研究並以國內「鐵路行車規則」第一百二十二條律定的17項事故，配合臺灣高鐵自營運起39個月的延滯資料進行分析，得到列車延滯的風險矩陣。另 Yun *et al.* (2011) 利用時間節省控制及黃金路段搜尋等方法，以啟發式模擬模式確認在行車衝突之交叉路段於執行號誌提升之列車控制計畫下，可使列車以最適行車速率行駛通過衝突路段，研究結果亦顯示相較於傳統最大速度控制法及綠波 (green wave) 控制法，本研究方法對於列車整體運轉時間及延滯之降低效果更好。

2.2 班表可靠度與延滯關係及分析方法

由於軌道列車運行延滯與可靠度之間存在抵換 (trade-off) 關係，故在理論研究與實務應用上，常藉由班表可靠度之探討來衡量延滯之衝擊影響，而準點率則常用來評估列車服務可靠度之良窳，但因影響延滯與可靠度之因素錯綜複雜，故準點率並非唯一指標，適用評估指標之選定常需視分析問題特性及研究方法類型而定。Middelkoop

and Bouwman (2002) 曾應用 SIMONE 模擬軟體來比較兩個小型路網及時刻表之穩定度 (stability)，其中對可靠度的評估是採用延滯的列車數、恢復時間，以及初始與最終延滯之比率等三個項目來作為評比的標準，並進行荷蘭 Weesp 車站轉乘等待時間的案例分析，發現增加 6 列貨物列車排入班表後，會造成延滯的程度增加 10%，顯示列車流量的提高會導致可靠度下降。惟其主要內容係強調將 SIMONE 商業模擬軟體，應用於荷蘭鐵路實際案例以進行時刻表穩定度之實驗測試結果，並無探討如何處理類似臺鐵多車種、複線運轉及多類型月臺型式之鐵路系統，及其路線、交通及控制條件交互作用產生複雜延滯擴散影響等問題。而 Briggs and Beck (2007) 則是發展更複雜的指標：先統計在某一車站的列車實際運行資料，找出一個最能夠代表其延誤時間和發生率關係的 q -exponential 函數 (q -exponential functions)，再透過該函數中的各項係數來評估該車站在可靠度上的表現。此方法的優點為可分別評估各車站的可靠度，其結果更能反映旅客的感受，但缺點為其過程相當繁雜。Higgins *et al.* (1995) 則考量了每個軌道路段、列車和班表，分別針對車站 (含端點站)、軌道相關以及列車相依等三種型態的延滯建構解析模式，可用於改善班表的可靠度，之後又分別以貨運和客運為對象，以解析方法預估其可能延滯時間 (Higgins and Kozan, 1998)。

Carey (1999) 為評估複線列車運行之可靠度 (reliability) 及準點率 (punctuality)，以解析式方法構建連鎖延滯之機率密度函數計算公式，其係為給定班表之列車班距及外生初始延滯機率密度函數之函數關係，並透過該函數計算出連鎖延滯之擴散延滯量；另因為外生初始延滯之機率密度函數不易獲得，其亦應用一些啟發式求解方法進行問題之求解。而 Carey and Carville (2000) 則以模擬模式分析單線區間之交會待避或多月臺車站特定區位之延滯擾動暨班表可靠度。有關班表的可靠度分析，除了解析方法亦有若干研究採用模擬分析。Barter (1998) 即以 RailPlan 模擬模式之基本概念，分析討論容量與列車服務之間的關係以及可靠度規劃 (planning for reliability) 問題。另在 EUROPE-TRIP (2000) 中則是利用 VISION 模式，探討時刻表每多排入不同列車數對整體路線延滯之影響情形，發現排入的列車數與整體

的延滯時間，兩者之間存在著抵換關係。另由於列車在軌道上運行的原理與等候理論 (queueing theory) 相當類似，因此以該理論為基礎，可對列車延滯行為與現象進行探討，Hansen (2000) 就利用了等候理論搭配 Max-Plus 代數法 (max-plus algebra)，來推估車站容量與列車服務的可靠度。

2.3 列車運行條件與延滯關係

另 Vromans *et al.* (2006) 分析列車服務頻率與列車平均延滯之關係，並釐清列車服務頻率與延滯及列車異質性間之關係，更以最短班距倒數和 (The Sum of Shortest Headway Reciprocals, SSHR) 及抵達班距倒數和 (The Sum of Arrival Headway Reciprocals, SAHR) 等二項指標來評估列車之異質性，證實提升列車營運可靠度最佳方法就是降低因列車異質性所導致之連鎖延滯效應。而 Huisman and Boucherie (2001) 藉由所蒐集之延滯資料構建一隨機模式來描述排班及非排班列車之移動情形，並藉由求解軌道系統之線性不等式得到每列車之行駛時間 (running time) 分布，研究亦發現影響行駛時間之關鍵因素包括列車數、列車異質性、初始延滯、列車行駛順序及運轉寬裕時間等。Carey and Crawford (2007) 針對一個具大量不同列車速度組合、多月臺車站及複雜路線之鐵路路網，研析實務上考量列車運行延滯、避免列車運行衝突之最佳化排班問題。為解決初始班表之列車運行衝突問題，該研究發展一啟發式求解方法，並透過模擬方法獲得無運行衝突之最佳班表，提供探討實際可運行班表、營運策略、車站配置及隨機延滯等問題之參考。

2.4 運轉整理與連鎖延滯擴散問題

運轉整理 (rescheduling) 之主要目的係在列車實際運轉過程中確認及解決列車衝突，並透過運轉整理使列車營運儘速回復至最初班表，Hansen and Pachl (2008) 於 *Railway Timetable & Traffic* 一書中即提到運轉整理之功能包括：使列車運轉重回無列車衝突之班表、提供

列車重新運轉之資訊、偵測列車衝突、自動解決列車衝突、產生無衝突重新指派之新班表及提供列車運轉交通控制所需之資料等。另該書亦指出解決列車衝突重新排班之方法包括：使用替代路線、延長停站時間、延長站間運轉時間、重新安排不停靠車站、運轉需要之增停車站及取消部分列車等，而此亦即是本研究運轉調度策略 (timetable recovery strategy) 所探討之範疇。

2.5 綜合評析

根據本章所蒐集整理之列車延滯發生原因分析、班表可靠度、列車運行條件、路線容量、列車延滯及服務可靠度分析方法與運轉調度策略對列車延滯改善分析等議題之相關文獻可知，鐵路列車延滯相關問題之分析十分複雜，除需界定所分析之延滯類型及其對正常營運之關鍵影響(如本研究之連鎖延滯)之外，亦需掌握影響延滯之路線、交通及控制等三大條件關聯之可靠度、路線容量及運轉調度策略等要項之關係及影響程度。茲將回顧之重要發現彙整如下：

一、鐵路列車運轉延滯之複雜性

鐵路列車運轉之延滯，除不可控制之外生初始延滯外，其衍生之連鎖延滯擴散更是營運者不易有效控制之問題，且因延滯所涉及之影響因素包括路線、交通及控制等諸多條件，且各項因素彼此間之交互作用相當複雜，不易掌握分析，故相關文獻之分析做法皆將研究範圍限縮至較小路段甚至車站區位。欲就整個路段進行連鎖延滯分析確有一定之難度，尤其對於具有多車種、多列車運轉特性之臺鐵系統難度更高。黃承傳、劉昭榮(民 100) 以臺鐵之七堵—樹林路段為案例構建模擬模式分析關鍵因素對於連鎖延滯之影響，除研析列車密度(train density)、初始延滯及運轉調度策略(timetable recovery strategy)對連鎖延滯之個別及其整體交互作用之影響外，更以迴歸分析方法構建列車連鎖延滯與該三項因子之間的指數關係函數以作為估算連鎖延滯的簡便工具，即希望針對鐵路列車運轉延滯複雜問題的研究能夠有所突破。

二、營運及延滯資料蒐集處理之困難度

臺鐵之營運資料本身即摻雜延滯(包括初始延滯及連鎖延滯)成分在內，故欲有效釐清所蒐集之運轉資料以作為模擬模式建構及後續案例分析，難度甚高，再加上許多關鍵參數(例如本研究之站間運轉時間及停站時間)運轉資料具有隨機特性，故有關所蒐集資料之前處理工作(如扣除有重大事件之營運資料)相當重要。由於相關資料內部之交互作用及複雜性甚高，故所蒐集資料之關鍵參數及變數判斷，及挑選之前處理工作將影響後續模式構建及連鎖延滯推估分析結果之成效。

三、延滯分析方法之限制

如本章文獻彙整所示，列車延滯及列車服務可靠度的研究方法主要包括解析方法、微觀模擬模式和統計分析三大類，惟各類方法論皆有其適用之問題特性及優缺點，但由於鐵路列車運轉延滯具有高度之複雜性，營運及延滯資料蒐集處理亦具有一定之困難度，故過去相關文獻研究若採解析方法和統計分析皆需將研究之問題範圍適度限縮始能處理，相較之下對於鐵路列車延滯複雜交互作用問題之處理，仍以微觀模擬模式有較好之成效，但有關其大量參數資料輸入處理、各項複雜運轉機制規則之設定等問題，皆需設法予以克服。

第三章 臺鐵列車連鎖延滯模擬模式

本章首先將考量鐵路列車運行之路線、交通及控制等條件之複雜交互作用關係及延滯之各種特性，蒐集臺鐵系統各項軟、硬體基礎資料及模式所需之各項參數資料，構建適合臺鐵連鎖延滯模擬分析之模擬模式先進行初步之連鎖延滯模擬分析，並作為後續各項連鎖延滯相關問題分析之基礎工具。軌道系統之延滯主要係為路線容量不足所導致，而路線容量又受路線條件、交通條件及控制條件所影響，故延滯亦將受上述三大類條件之影響。而因列車運行延滯在軌道系統營運階段係扮演列車班距 (train headway) 及列車流量 (traffic flow) 之關鍵決定要素，故本研究所建構之模擬模式主要功能在於連鎖延滯之推估，其估算結果可作為後續營運可靠度 (train reliability) 分析及營運排班調度之參考 (Hwang and Liu, 2010；黃承傳與劉昭榮，2011)。

3.1 模式架構

本模式之架構如圖 3-1 所示，主要輸入資料包括基本條件資料、計畫班表、初始延滯，輸出資料則為模擬之實際班表，並據以計算連鎖延滯。各部分內容茲分述如下：

1. 基本條件資料：包括軌道容量相關的路線條件、交通條件與控制條件如表 3.1。
2. 計畫班表：包括各車次於每個車站的預定到/開時間、各車次之起迄點、列車順序及停站型態，故本模式可處理多種列車、不同列車等級、不同站間運轉時間、不同起迄站、不同停靠車站、不同停靠時間之情境。
3. 初始延滯：本研究旨在處理連鎖延滯之影響，故模擬模式必須設定一項初始延滯的相關輸入資訊，初始延滯可設定在軌道路線之任一位置，延滯參數之設定如表 3.2 所示。
4. 模擬之實際班表：模擬後之班表為模式之最後產出，其呈現列車受延滯影響後之實際到開時間，而模擬班表與計畫班表中各列車到開時間之差距即為連鎖延滯。

表 3.1 基本輸入資料表

項目 \ 條件	路線條件	交通條件	控制條件
輸入參數 內容	-路線 -車站列表 -車站軌道佈設	-計畫班表各項 列車資訊 -最短停站時分 -最短站間運轉 時間	-同向時隔 -反向時隔 -站間容量 -趕點能力

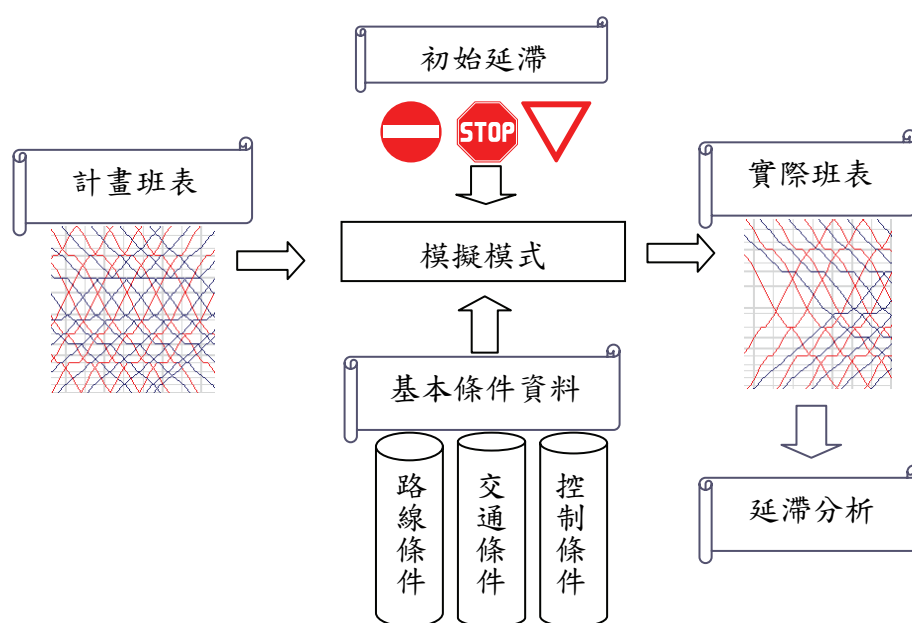


圖 3-1 模式架構圖

表 3.2 初始延滯之設定參數表

延滯參數	說明
延滯地點	若為站間軌道：需指定方向 若為站內軌道：需進一步指定股道
起始時間	該軌道資源開始無法被使用
解除時間	該軌道資源可再被使用之時間
延滯量	將「解除時間」減去「起始時間」即為延滯量 (或連鎖延滯)

3.2 模式假設與限制

本模式之假設與操作限制如下。

1. 車站間均有兩股道同時以複線方式運轉。
2. 臺鐵系統列車於北上、南下之站間股道係獨立運轉，但當產生延滯時，於站內之共用股道則可依營運需要彈性調度不同方向之列車順序，故雙向之列車運轉將會產生交互作用。
3. 臺鐵區域鐵路系統之車站配置型態大致可歸納為 5 種型式，如圖 3-2~圖 3-6。
4. 由於臺鐵系統號誌設置之限制，各站間依不同條件皆有允許駐留列車數之上限限制（為利模式操作本研究假設為 2 列車）。
5. 運轉調度之站間趕點時間以站間運轉時間固定的縮減比率設定之（本研究假設為 0.9）。

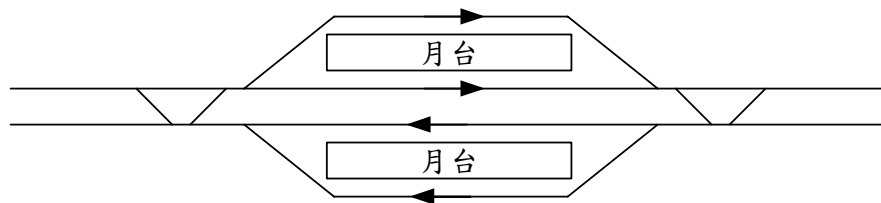


圖 3-2 型 I 車站軌道佈設

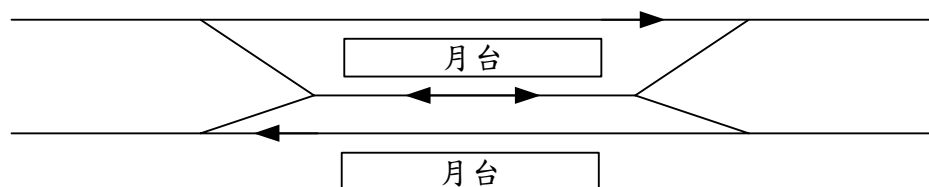


圖 3-3 型 II 車站軌道佈設

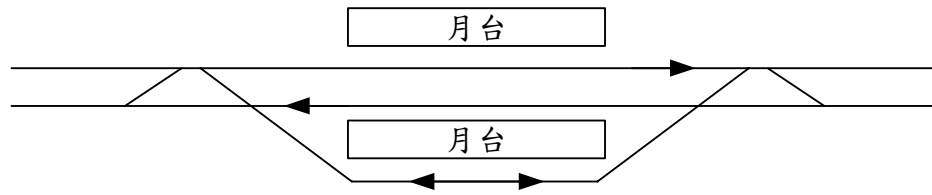


圖 3-4 型 III_R 車站軌道佈設

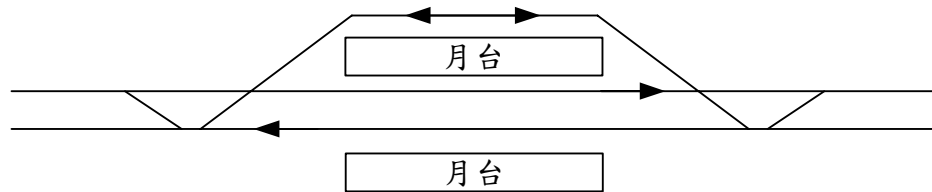


圖 3-5 型 III_L 車站軌道佈設

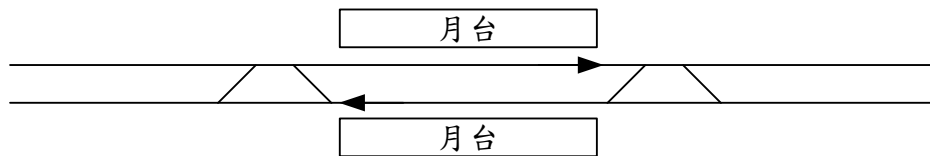


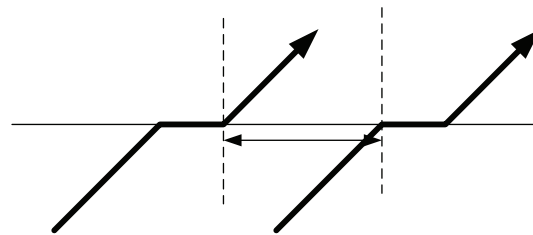
圖 3-6 型 VI 車站軌道佈設

3.3 模擬機制

為準確推估連鎖延滯，本模擬模式必須考慮所有列車運行之影響因素，相關因素之處理方式彙整如下：

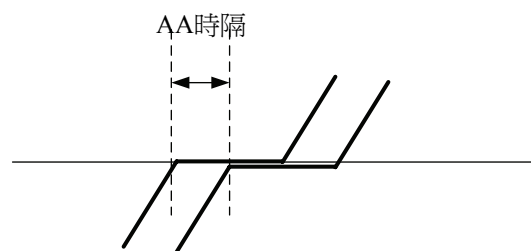
1. 列車衝突解決：站間軌道之列車運行規則為先進先出，且不允許站間追越；另站間軌道有容量限制，不允許過多的列車滯留同一站間；站內每一股道只允許一列車佔用，而先行列車離開車站後，續行列車進入該股道必須符合離—到時隔限制，如圖 3-7 所示；二連續列車抵達同一車站之不同股道，必須符合到—到時隔限制，如圖 3-8 所示；至於二連續列車由同一車站之不同股道離開，則必須符合離—離時隔限制，如圖 3-9 所示；另假如二列車由不同方向通過平面交叉點，則必須符合平面交叉時隔限制，如圖 3-10 所示。

2. 列車延滯排除：當模式執行時，若續行列車違反前述之運行規則而無法前進時，即產生延滯及可能之連鎖延滯現象；當模式偵測到此現象時，均統一將該事件的執行時間（系統時間）延後 1 秒執行，此方法可以使列車運行無限的往後遞延直到前方的事故（延滯）原因排除為止。
3. 運轉調度策略：鐵路營運單位在列車延誤時，通常會啟動運轉調整機制，而本模式所考慮的「運轉調度策略」允許使用單一或多種策略同時搭配使用，包括：
 - (1) 站內趕點：縮短停站時間，但是必須遵守停站時間最短限制，而且需依公開時刻表不可提早開車。
 - (2) 站間趕點：縮短站間運轉時間，但仍需遵守趕點限制。
 - (3) 同時採取站內趕點與站間趕點。



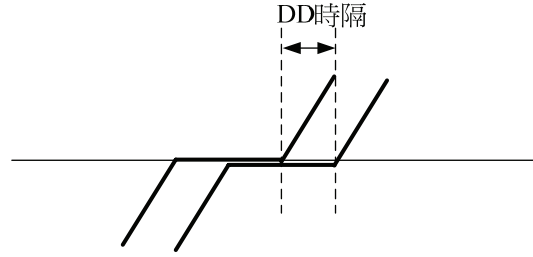
註：DA 時隔 (departure-arrival headway)

圖 3-7 同向列車離一到時隔示意圖



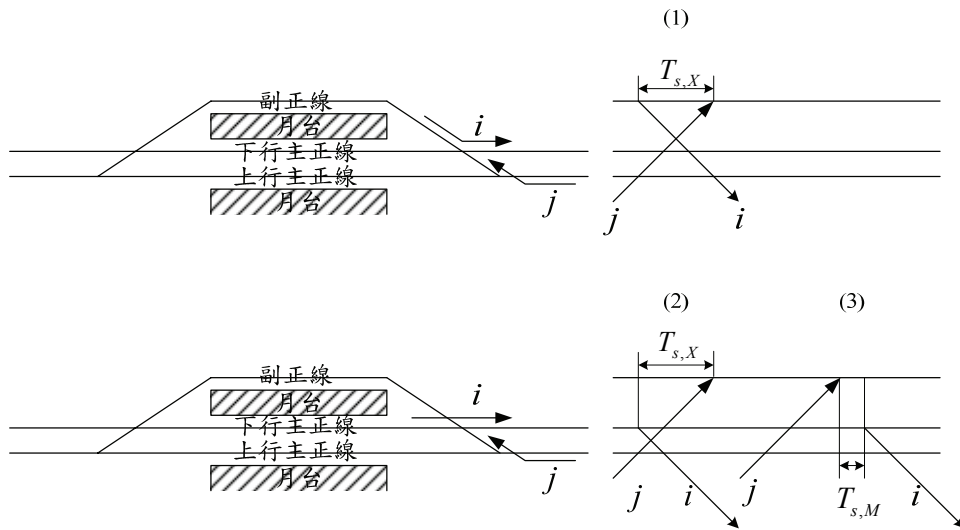
註：AA 時隔 (arrival-arrival headway)

圖 3-8 同向列車到一到時隔示意圖



註：DD 時隔 (departure-departure headway)

圖 3-9 同向列車離—離時隔示意圖



註： $T_{s,X}$ 為先行列車離站與續行列車進站之號誌安全時距；

$T_{s,M}$ 為先行列車進站與續行列車離站之號誌安全時距。

資料來源：鍾志成 (2008)

圖 3-10 型 III 月臺之平面交叉時隔示意圖

3.4 模擬程式設計

本研究模擬程式之設計與開發，旨在設計一套資料結構來描述整個軌道行車系統之各種狀態，包含列車、車站、軌道等，同時針對列車運轉邏輯設計對應的狀態改變，透過不斷更新電腦記憶體中的變數來達到系統模擬之目的，最後將本研究關注的變數 (即列車延滯) 匯

出，以利後續的績效評析。以下將分別就物件類別、行車系統模擬流程、初始延滯等三項設計進行深入說明。

3.4.1 物件類別 (Class) 設計

類別設計為物件導向程式設計中最重要之基礎 (周斯畏, 2002)，一般而言包括：(1) 邊界類別 (boundary class)、(2) 實體類別 (entity class) 及 (3) 控制類別 (control classes) 三大類，其中第一類是用來處理使用者介面，第二類則是用來處理問題領域的核心資料，最後一類則是用來處理前兩類資料的協調與互動；由於 (1) 與 (3) 類與核心模式較無關聯，故以下僅就與模式相關之實體類別部分進行說明。本研究共設計了 9 個類別來描述整個軌道系統，包括 RailSystem、Train、DwellPlan、Station、StationI、StationII、StationIII_R、StationIII_L、Tracks 等 (如圖 3-11)。其中 RailSystem 是最上層的類別，除一些 Global 的變數之外，所有其他的類別物件均包含於其中；而 Station 類別則是其他四個 Station 開頭類別的父類別，父類別用來描述最基本的車站 (即圖 3-6 的捷運化車站)，另外四個子類別則分別描述其他四型車站如圖 3-2~圖 3-5。Tracks 類別是被 Station 類別所包含，而 Train 與 DwellPlan 則是用來描述「計畫班表」與「實際班表」。另其中 Train 與 Station 都以集合物件的形式和 RailSystem 以合成 (aggregation/composition) (周斯畏, 2002) 關係存在，其他諸如 Train 與 DwellPlan、Station 與 Track 均為合成關係。至於 StationI、StationII、StationIII_R、StationIII_L 等四個類別則繼承 (inheritance) 自 Station 類別，用以實作各種不同型式車站的行車限制。

另本研究依圖 3-11 執行進一步的虛擬碼撰寫時發現，StationII、StationIII_R、StationIII_L 三個類別有許多屬性與行為是類似，包括站內有三股軌道、第三股軌道共用、共用軌道的使用牽涉反向時隔 (平面交叉) 等屬性。故為使程式容易維護，本研究進一步將平面交叉的行為獨立處理，建立 CrossOverInfo 類別處理各種平面交叉，如此將可以簡化圖 3-11 的類別圖，亦即只要在車站物件建構 (construction) 時加入指定的平面交叉路徑設定即可。經整理後得到 StationII、StationIII_R、StationIII_L 三種車站的平面交叉路徑設定分別如圖

3-12、圖 3-13、圖 3-14 所示，在程式撰寫時即可根據前述三種路徑設定將平面交叉類型寫入程式，以處理避免列車衝突且需足夠間距 (headway) 之問題，如此即能達到程式碼共用與容易維護的目的。

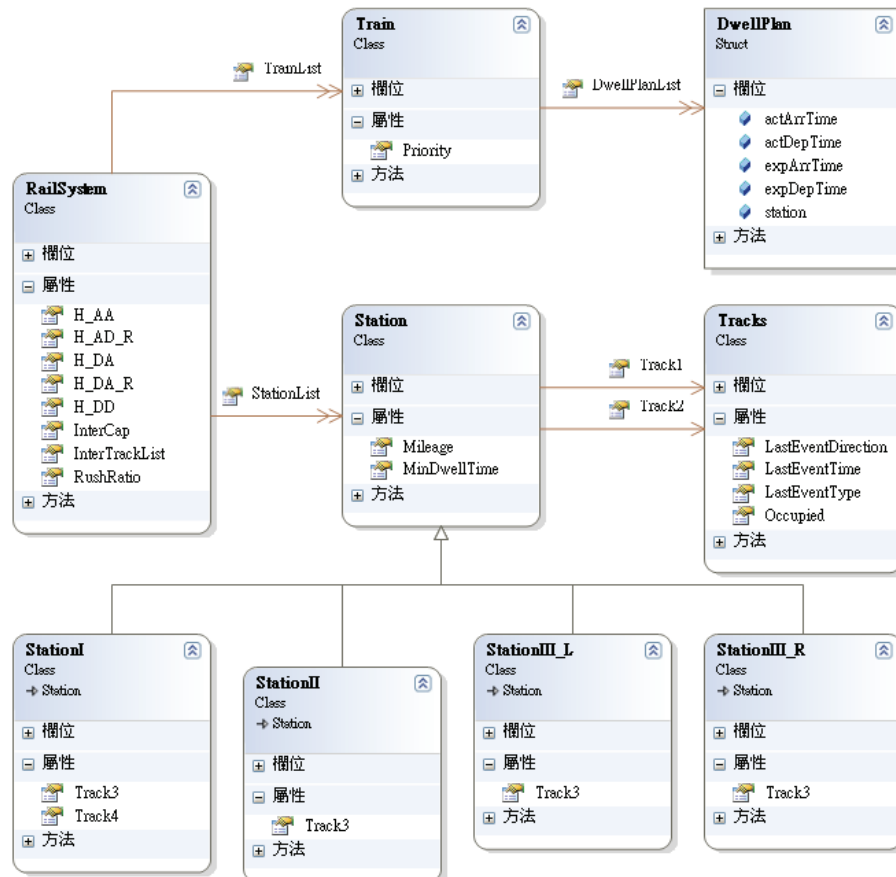


圖 3-11 類別設計架構圖

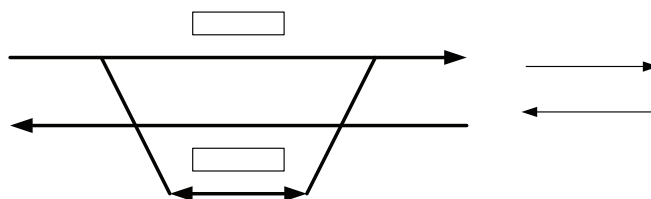


圖 3-12 StationII 型車站股道之平面交叉路徑設定

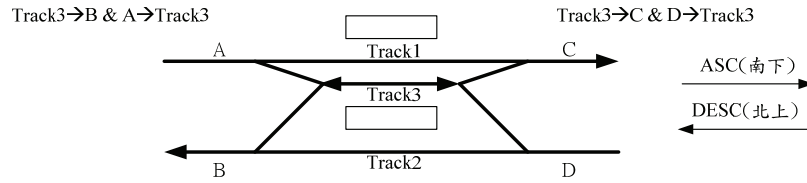


圖 3-13 StationIII_R 型車站股道之平面交叉路徑設定

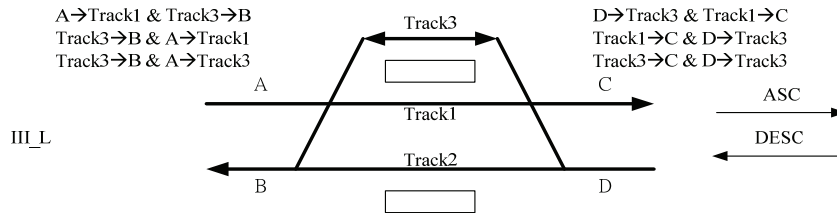


圖 3-14 StationIII_L 型車站股道之平面交叉路徑設定

3.4.2 行車系統模擬流程

本模擬模式估算連鎖延滯之整體流程如圖 3-15 所示，主要是以事件導向的方式進行，一開始將所有列車的第一個事件加入事件列表，經時間排序後每次依序執行第一個事件，共計有進站、通過與離站三種事件；若有列車衝突則都在延滯處理時把時間加計一秒後重新加回事件列表，當所有的事件均處理完成後，事件列表沒有任何事件，即完成模擬。至於衝突項目的檢查方式則視事件種類有所差異如表 3.3 所示。

至於本模式之程式語言選擇，因臺鐵之連鎖延滯推估問題錯綜複雜，經評估仍宜採模擬模式進行，但目前並無合適的商用軟體可供使用，故自行撰寫程式。考量類別庫之完整性及未來的擴充支援，本研究採用微軟.NET Framework 類別庫；原則上只要符合 CLI (Common Language Infrastructure) 標準程式語言均可呼叫.NET Framework 類別庫，且目前支援最完整也是主流的.NET 語言即是 C#，故本研究以 C#撰寫之，同時 IDE (Integrated Development Environment) 整合開發平臺則選用 Visual Studio 2005 Professional。

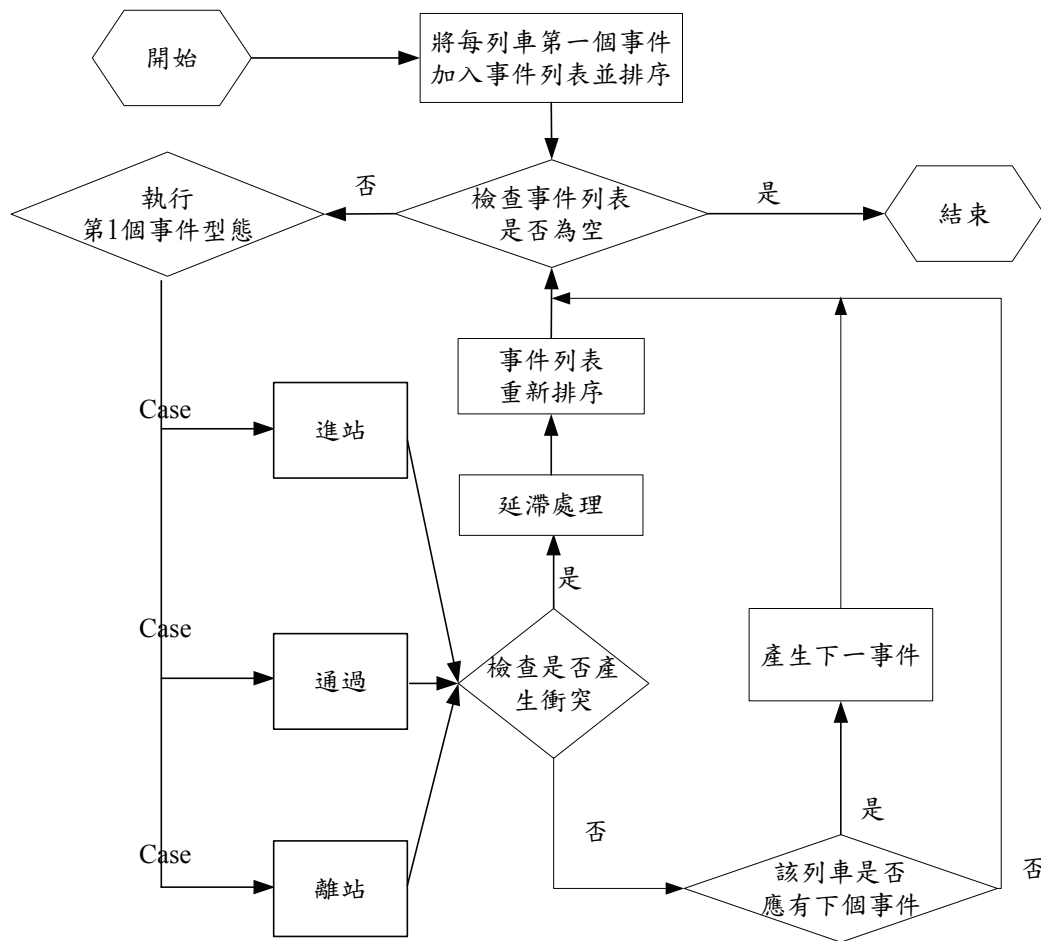


圖 3-15 連鎖延滯之模擬流程圖

表 3.3 各種不同事件的衝突檢查項目

事件種類	進站	通過	離站
檢查項目清單	是否存在站間追越 站內股道數是否足夠 進站時隔是否充足 反向時隔是否充足	是否存在站間追越 站內股道數是否足夠 進站時隔是否充足 反向時隔是否充足 離站時隔是否充足 股道容量是否充足	離站時隔是否充足 反向時隔是否充足 股道容量是否充足

3.4.3 初始延滯運作

由於初始延滯可能於不同時空發生且程度不一，而本模擬模式係著重處理列車的基本運轉以及連鎖延滯推擠效應，程式未具有模擬「初始延滯」之功能，故於推估連鎖延滯時須從事件驅動引擎中新增兩種事件，包括列車運轉之障礙開始與障礙結束（如圖 3-15 模擬架

構)。第一種事件安排於初始延滯發生時執行，觸動時將會鎖定使用者所設定之軌道，使其無法被列車佔用；第二種事件則在第一種事件發生後的若干時間（使用者指定之延滯程度）執行，觸動時將會解除鎖定，透過上述方式將可在任意時空產生不同程度的初始延滯。

3.5 模擬模式驗證

為測試模擬模式之準確適用性，本模式選用臺鐵系統交通量最繁忙之七堵—樹林路段，擷取 2009 年 7 月 29 日（週三）一般日之中央列車控制 CTC（Centralized Train Control）資料庫，選定北上萬華—臺北路段於上午 10:10 北上 42 車次有 18.2 分鐘之初始延滯所產生實際列車到—開之延滯資料進行驗證分析，結果顯示計有臺北、松山、南港、汐止及七堵等 5 個站共 40 部列車受該初始延滯影響，相關驗證分析討論茲述如下：

一、模式驗證指標選定

檢視上述 40 筆列車樣本，其實際延滯值不僅頗分散（平均值為 282.6 秒、標準差為 428.8 秒）且延滯值跳動、無如預期向下游擴散之現象，經研判該結果可能係實務上調度員遇事故時，常依經驗判斷採行不同列車調度策略所致，且調度策略因時因地因車皆不同；另該延滯結果亦可能是整體軌道系統、前後路段及各項複雜外生事件因素交互作用之結果，故所蒐集之實際延滯資料屬性不易釐清歸納其趨勢傾向，亦不適於進行各項統計檢定分析。另由於其中 13 筆樣本之實際延滯值為 0，若以一般慣用之平均絕對百分比誤差（Mean Absolute Percentage Error, MAPE）或均方根誤差（Root Mean Square Error, RMSE）作為模式推估精確度之驗證指標，將不適用且產生極大偏誤之現象，故在臺鐵實際延滯資料具高度複雜性不易釐清之前提下，鑑於臺鐵營運實務上若列車延滯不超過 5 分鐘則不視為誤點之認定標準，本模式之驗證指標將以受影響列車之模式推估延滯與該列車實際延滯之差異絕對值是否超過 5 分鐘，作為模擬模式之精確度評估指標。

二、模式驗證結果分析

應用上述模擬模式之精確度評估指標，經彙整本案例 40 筆受影響列車樣本之模式推估延滯與實際延滯資料，其差異分布及其差異絕

對值之統計分如圖 3-16 及表 3.4 所示，結果顯示有 30%之模擬延滯時間與實際延滯時間幾近相同，有 90%之列車樣本模擬延滯時間與實際資料差距在 5 分鐘之內，故整體而言顯示本模擬模式估算結果之準確度可被接受。

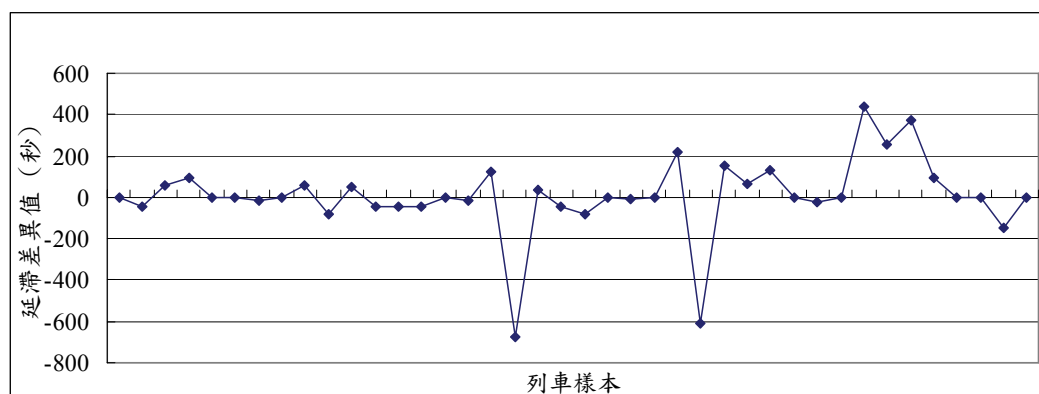


圖 3-16 受影響列車樣本之模擬延滯與實際延滯差異分布圖

表 3.4 受影響列車樣本之模擬延滯與實際延滯差異統計表

延滯差異之絕對值(秒)	列車樣本數	樣本百分比	累積百分比
diff.=0	12	30.0%	30.0%
0<diff.<=60	13	32.5%	62.5%
60<diff.<=120	6	15.0%	77.5%
120<diff.<=180	3	7.5%	85.0%
180<diff.<=240	1	2.5%	87.5%
240<diff.<=300	1	2.5%	90.0%
diff.>300	4	10.0%	100.0%
合計 = 40		100.0%	

第四章 案例分析

有關列車連鎖延滯與其關鍵影響因素之關聯分析，黃承傳與劉昭榮 (2011) 所建構之確定性模擬模式已曾針對外生初始延滯地點、持續時間及不同運轉調度策略等因素釐清其彼此間之因果關係。惟由該確定性模擬分析過程可發現，影響連鎖延滯結果之部分關鍵參數確具有隨機特性，故本研究即以隨機性模擬模式進行評估分析，深入探討有初始延滯發生後產生連鎖延滯之班表穩定度問題，以真實反映列車之實際交互作用狀況及其影響程度。由於站間運轉時間及停站時間係為運轉調度策略中最關鍵二項參數，為確實反映其隨機特性，本研究針對該二項參數進行隨機特性分析，並將其隨機資料納入模擬模式推估連鎖延滯。故本研究案例之連鎖延滯分析成果除可作為班表穩定度之排班規劃參考，參數隨機特性分析結果亦可作為臺鐵系統營運之運轉調度策略擬訂之依據。

另鑑於臺鐵系統之列車種類甚多且特性各異，為利分析亦將其概分為對號列車及通勤電聯車二類進行研究路段延滯資料之蒐集分析，再進行二類列車之個別站間運轉時間及停站時間的隨機資料分布分析；而為使隨機參數之分布資料具有一致性及代表性，資料處理過程中除先將特性異常樣本 (outliers) 先予去除外，亦針對分佈變異較大之停站時間資料再依尖、離峰進行分群後分別分析。

有關本研究之隨機特性分析，茲分別就資料蒐集分析、延滯特性參數校估、延滯特性參數分析及隨機特性連鎖延滯模擬結果等部分說明如下。

4.1 資料蒐集分析

為利進行隨機特性連鎖延滯模擬分析，首先必須蒐集臺鐵系統之列車延滯資料；惟依據鍾志成 (2011) 之研究顯示，過去有關列車延滯相關分析的研究，大多是以終點站之列車平均延滯作為量測基準，少部份才會針對服務範圍內之車站逐站統計列車延滯。惟不同的時間

範圍內，由於列車的密度、車種可能會不一樣，因此列車的延滯也會隨時間而異，但同時分析列車延滯的時空變化情形者並不多。而本節所用進行臺鐵列車延滯時空分布分析的資料，是根據臺鐵中央行車控制系統所記錄的列車實際到離站資訊來計算，依 CTC 紀錄的列車實際到離站情況，列車延滯量即為公告時刻表的表訂到/離站時間與實際到/離站時間之差值。

有關本章所需之臺鐵系統列車延滯資料，資料來源為 2009 年 12 月 1 日到 2010 年 3 月 25 日期間臺鐵 CTC 所記錄的列車到離站延滯時間，至於空間範圍為臺鐵西部幹線北部區域之七堵站至新竹站，共計蒐集 18 個車站、17 個站區間資料 (部分車站及站間資料缺漏)；而為利延滯分析，本節針對站間運轉時間及停站時間二項參數進行分析，共計蒐集 421,869 筆站間運轉時間及 372,327 筆停站時間資料，並將其與表訂班表時間比較即可得到列車延滯資料。另為利區隔不同車種運轉特性，本節亦概分對號列車及通勤電聯車二類車種搭配站間運轉時間及停站時間，進行資料蒐集整理及後續之校估分析。

4.2 延滯特性參數校估分析

為利延滯特性參數之校估，應先了解延滯特性參數之觀測方式，茲參考鍾志成 (2011) 之研究建議，有關其觀測方式在表定時刻表與實際到開資料均蒐集齊全的前提下，可能的觀測方式至少包括以下四種，茲說明其優缺點如下。

一、觀測「到站延滯」與「離站延滯」之分布

此為最直接的觀測方式，因臺鐵系統即有公告時刻表，但列車可能提早進站，故「到站延滯」的分布可能存在負值；然就「離站延滯」的分布而言，則不存在延滯值小於零的狀況，因為列車不允許提早發車。惟此觀測方式最大的缺點就是延滯值可能會累積先前上游車站的延滯，造成重複計算的問題，故此觀測方式不適用於本連鎖延滯推估模式。

二、觀測「實際站間運轉時間絕對值」與「實際停站時間絕對值」之分布

此觀測方式雖可改善前一種觀測方式之上游車站延滯累積問題，亦即觀察實際列車的站間運轉時間變異及停站時間變異，蒐集系統的變異特性參數可供延滯分析模式使用，但分析過程需將樣本進行適當的分類，例如依不同的列車種別、不同時段、區間來分類。特別是不同區間由於路線長度不同，路線較長的區間可能運轉時間變異就會較大，在沒有標準化的情況下，可能每個區間均需分析，另停站時間亦因有車站規模大小不一而需要標準化的問題存在，故此觀測方式對於本研究可能過於複雜不適用。

三、觀測「站間運轉時間實際值與表定值差值」與「停站時間實際值與表定值差值」之分布

為了克服前一種觀測方式存在有不同區間長度不一的狀況，本觀測方式直接計算站間運轉時間的實際值與表定值之差，由於觀測值均減掉一個標準值（即表定站間運轉時間或表定停站時間），因此統計出來的參數就可以套用到各種不同長度的區間或套用在各種不同規模的車站。

四、觀測「站間運轉時間實際值與表定值比值」與「停站時間實際值與表定值比值」之分布

此種觀測方式類似前一種，亦是要解決第二種觀測方式沒有依區間長度或車站規模來標準化的問題，只是本觀測方式採用的標準化方法並非用實際值減去表定值來處理，而是利用實際值除以表定值來呈現，對於本研究而言，此種觀測方式最為簡便，故本節後續之特性參數校估即運用此方法。

4.3 延滯特性參數分布分析

依據上述參數校估方式，本節後續之延滯模擬分析將分為對號列車及通勤電聯車二車種，分別先彙整處理「站間運轉時間實際值與表定值比值」與「停站時間實際值與表定值比值」之累積分布資料，以

作為後續連鎖延滯模擬分析之參數資料輸入基礎。本研究隨機特性分析之目的係為擷取隨機資料，而不在於探討校估各項隨機資料之分布函數型態，故並無推估隨機參數之數值，且站間運轉時間及停站時間之隨機分布資料主要係呈現實際值與表訂值比值累積機率佔大宗之資料，並據以模擬分析具隨機特性之連鎖延滯。至於有關延滯特性參數資料處理方式，茲分述如下：

1. 樣本資料：有關站間運轉時間資料，其中對號列車部分原計有實際通過 17 個站間共 176,248 筆，通勤電聯車部分則原計有實際通過 17 個站間共 245,620 筆資料。至於停站時間資料，其中對號列車部分原計有實際於 18 個車站共計 92,903 筆，通勤電聯車則原計有於 20 個車站共計 279,424 筆之停站時間資料。
2. 延滯特性參數分布圖繪製：有關站間運轉時間分布圖，係先計算各樣本資料與該站間運轉時間表定值之比值後並進行排序加總，再將各站間之「站間運轉時間實際值與表定值比值」作為 x 軸數值、與其累計機率作為 y 軸數值，再繪製其整體分布情形。至於停站時間分布圖亦類似，但需先計算樣本之「停站時間實際值與表定值比值」，再依上述方法繪製整體分布圖。
3. 隨機延滯特性參數擷取方式：依據前述二類參數資料分布圖，其隨機時間只要透過亂數種子產生之亂數，對應該參數分佈曲線，即可透過該站間運轉時間或停站時間之表訂值還原得其隨機值。
4. 隨機延滯特性參數分布分析：經由上述方式原可得二類隨機參數資料分布圖，惟進一步檢視前述彙整之站間運轉時間及停站時間分布資料，發現許多停站時間樣本資料之比值皆顯示異常，推測所蒐集之臺鐵系統列車運轉資料可能存在許多干擾因素，以致停站時間資料出現許多異常結果，故顯見資料仍需再分群過濾始能應用分析。鑑於列車營運所產生之延滯皆可能擴散導致大量班次之站間運轉時間與停站時間異常，故為利探討列車在一般營運情況下的延滯狀況，經檢視案例資料期間發現新年連續假期（12 月 31 日到 1 月 3 日）、春節連續假期（2 月 12 日到 21 日），以及當時於 3 月 4 日和 8 日兩天國內發生規模較大之地震，故共計 16 天

的異常營運資料將去除不列入分析。另由於各站之停站時間分布型態較為異常且各站之差距頗大，為利後續連鎖延滯分析，本節再將停站時間資料區分為尖、離峰進行分析，其中尖峰時段係定義為上午 6~9 點及下午 4~7 點，其餘則為離峰時段。有關站間運轉時間與停站時間之隨機資料分布型態，經統計如表 4.1 所示，原則上其應接近常態分配；至於去除異常事件日後之各分群資料茲分析如下。

表 4.1 參數隨機樣本資料分布統計

隨機參數之 分布機率 樣本分布區間	站間運轉時間之 分布機率		停站時間之分布機率			
	對號列 車	通勤電 聯車	尖峰時段		離峰時段	
			對號列 車	通勤電 聯車	對號列 車	通勤電 聯車
$\Pr[\mu - \sigma \leq \bar{X} \leq \mu + \sigma]$	79.08%	80.93%	89.89%	93.81%	88.20%	95.20%
$\Pr[\mu - 2\sigma \leq \bar{X} \leq \mu + 2\sigma]$	96.69%	95.73%	96.61%	97.34%	96.78%	98.03%
$\Pr[\mu - 3\sigma \leq \bar{X} \leq \mu + 3\sigma]$	98.99%	98.05%	98.35%	98.36%	98.56%	98.87%

4.3.1 對號列車之站間運轉時間分布

此項資料係以實際通過 17 個站間之對號列車共計 149,231 筆站間運轉時間資料，分別計算其與該站間運轉時間表定值之比值後並進行排序加總，再分別將各站間之「站間運轉時間實際值與表定值比值」作為 x 軸數值、與其累計機率作為 y 軸數值，繪製其分布如圖 4-1，另由於各站間之運轉時間分布尚稱一致並未有變異特別大之資料，故本研究亦以所有站間樣本直接進行「站間運轉時間實際值與表定值比值」之分布分析，可得一「總計」曲線，則其隨機資料可直接以該曲線進行擷取。

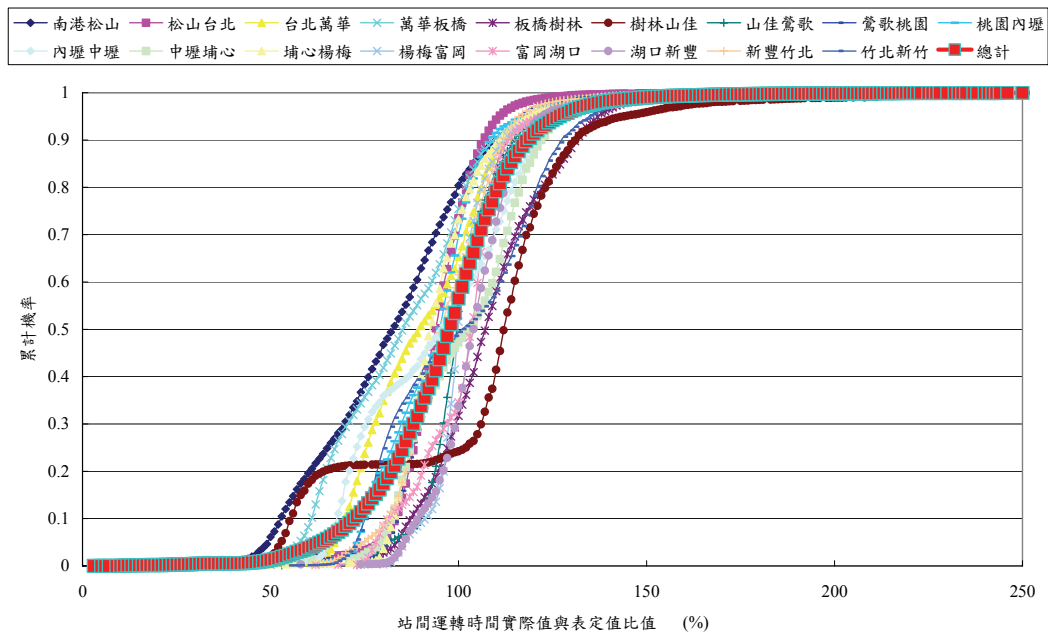


圖 4-1 對號列車之站間運轉時間分布圖

4.3.2 通勤電聯車之站間運轉時間分布

此項資料係以實際通過 17 個站間之通勤電聯車共計 211,778 筆站間運轉時間資料，同樣分別計算其與該站間運轉時間表定值之比值後並進行排序加總，再分別將各站間之「站間運轉時間實際值與表定值比值」作為 x 軸數值、與其累計機率作為 y 軸數值，繪製其分布如圖 4-2，另如同前述對號列車之作法，由於各站間之運轉時間分布尚稱一致並未有變異特別大之資料，故本研究亦以所有站間樣本直接進行「站間運轉時間實際值與表定值比值」之分布可得一「總計」曲線，並直接以該曲線進行隨機資料之擷取。

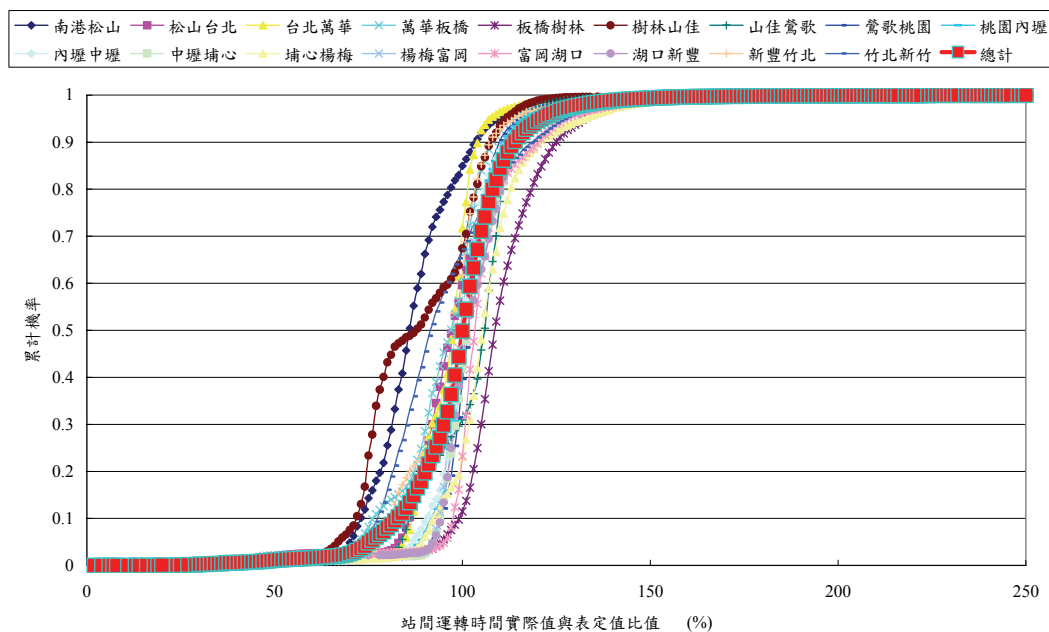


圖 4-2 通勤電聯車之站間運轉時間分布圖

4.3.3 對號列車於尖峰時段之停站時間分布

此項資料係以實際所有對號列車於 18 個車站之尖峰時段共計 26,983 筆停站時間資料，分別計算各站所有列車每筆實際停站時間與表定值之比值後，再分別將各站之比值先排序後累加，並以「停站時間實際值與表定值比值」作為 x 軸數值、其累計機率作為 y 軸數值，分別繪製各站之停站時間分布如圖 4-3。

有關各站於尖峰時段停站時間之特性顯示於各站之比值分布情形，其各站停站時間之各項特性彙整顯示，因內壢及埔心站於尖峰時段無對號列車停靠，故僅剩 16 個車站資料；由所蒐集之資料顯示，經去除異常事件日資料後，各站樣本資料之比值超過 1000%特別異常之比例更降低至約 0.037%，大部分資料之比值更集中約介於 50%~200%之間，若以比值介於 80%~120%資料為例，其比例即達約 39%，顯見分群後之對號列車於尖峰時段之停站時間分布資料已更為一致，應更能作為進一步分析連鎖延滯之基礎。

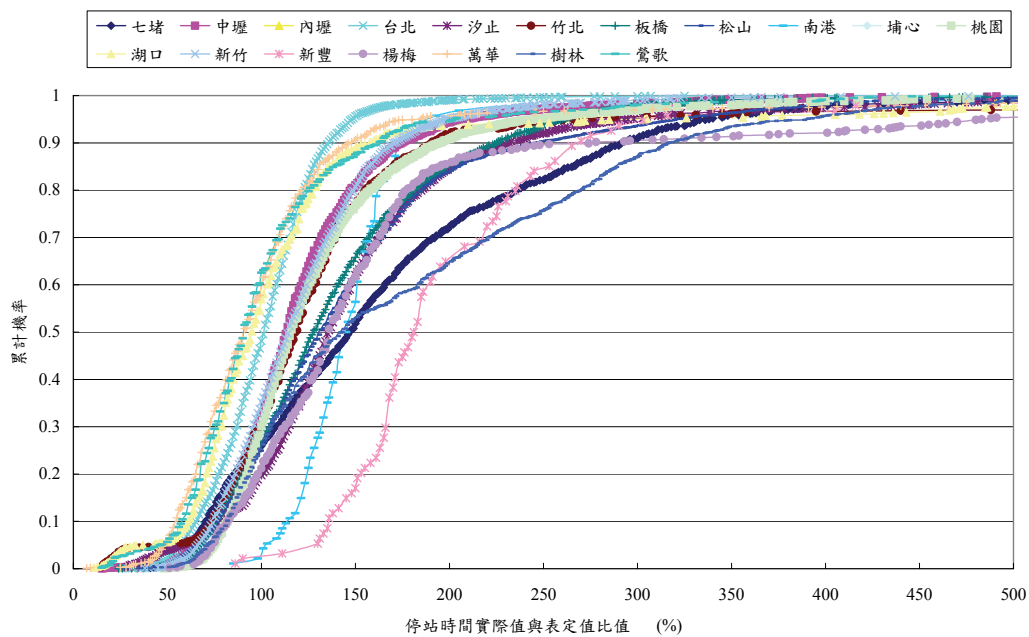


圖 4-3 對號列車於尖峰時段之停站時間分布圖

4.3.4 對號列車於離峰時段之停站時間分布

此項資料係以實際所有對號列車於 18 個車站之離峰時段共計 51,680 筆停站時間資料，分別計算各站所有列車每筆實際停站時間與表定值之比值後，再分別將各站之比值先排序後累加，並以「停站時間實際值與表定值比值」作為 x 軸數值、其累計機率作為 y 軸數值，分別繪製各站之停站時間分布如圖 4-4。

有關其各站於離峰時段停站時間之特性於各站之比值分布情形，其各站停站時間之各項特性彙整顯示，因新豐站於離峰時段無對號列車停靠，故僅剩 17 個車站資料；由所蒐集之資料顯示，經去除異常事件日資料後，各站樣本資料之比值超過 1000% 特別異常之比例更降低至約 0.033%，大部分資料之比值更集中約介於 50%~200% 之間，若以比值介於 80%~120% 資料為例，其比例即達約 36%，顯見分群後之對號列車於離峰時段之停站時間分布資料也更為一致。

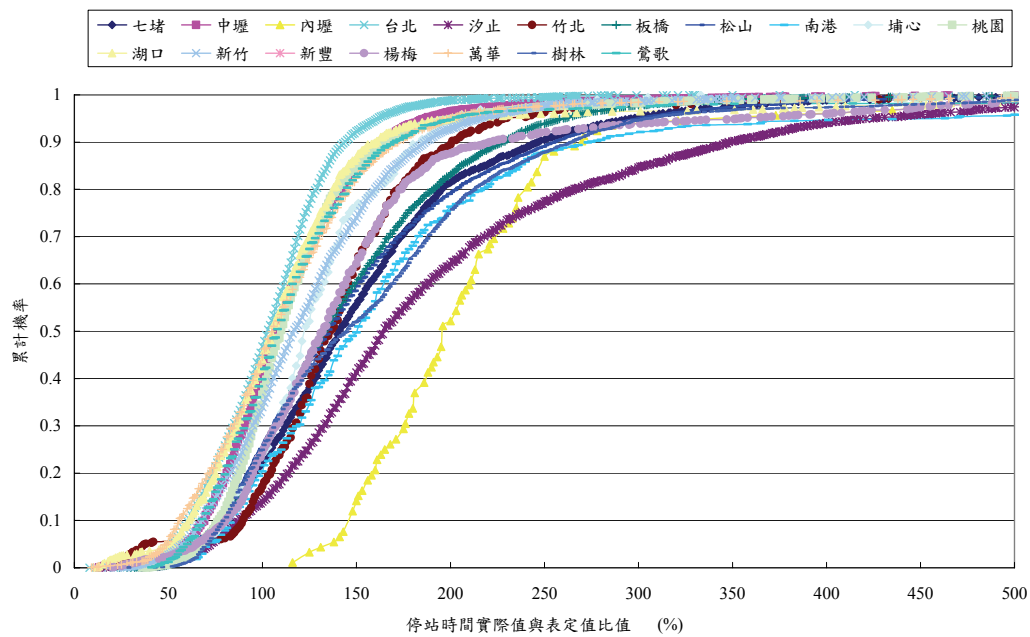


圖 4-4 對號列車於離峰時段之停站時間分布圖

4.3.5 通勤電聯車於尖峰時段之停站時間分布

此項資料係以實際所有通勤電聯車於 20 個車站之尖峰時段共計 91,764 筆停站時間資料，分別計算各站所有列車每筆實際停站時間與表定值之比值後，再分別將各站之比值先排序後累加，並以「停站時間實際值與表定值比值」作為 x 軸數值、其累計機率作為 y 軸數值，分別繪製各站之停站時間分布如圖 4-5。

有關其各站於尖峰時段停站時間之特性顯示於各站之比值分布情形，由所蒐集之資料顯示，經去除異常事件日資料後，各站樣本資料之比值超過 1000%特別異常之比例已由原先之 0.42%降低至約 0.37%，大部分資料之比值更集中約介於 20%~250%之間，若以比值介於 80%~120%資料為例，其比例即達約 30%。

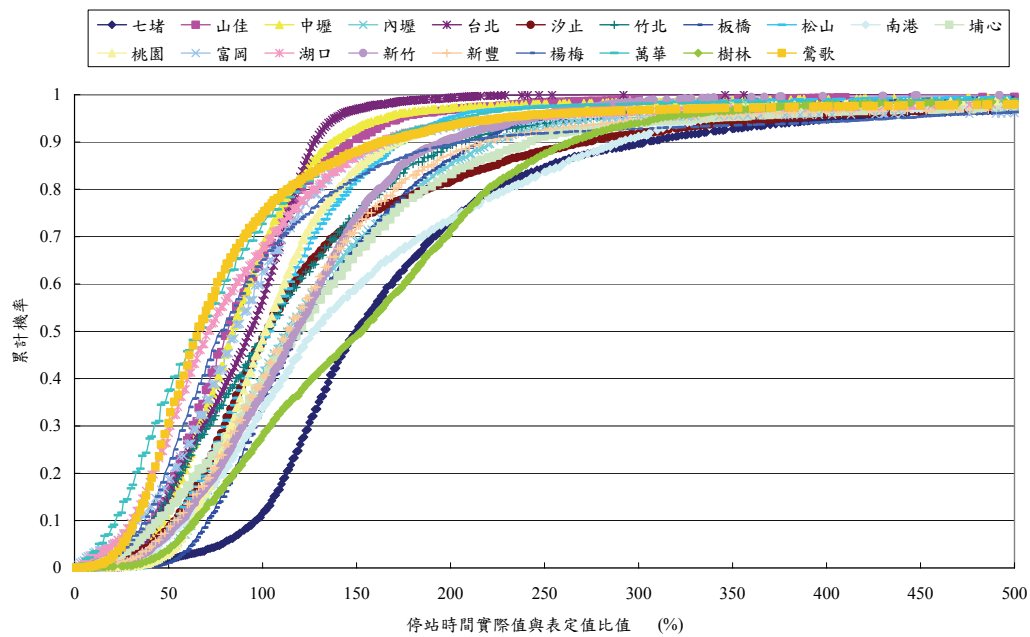


圖 4-5 通勤電聯車於尖峰時段之停站時間分布圖

4.3.6 通勤電聯車於離峰時段之停站時間分布

此項資料係以實際所有通勤電聯車於 20 個車站之離峰時段共計 148,892 筆停站時間資料，分別計算各站所有列車每筆實際停站時間與表定值之比值後，再分別將各站之比值先排序後累加，並以「停站時間實際值與表定值比值」作為 x 軸數值、其累計機率作為 y 軸數值，分別繪製各站之停站時間分布如圖 4-6。

有關其各站於離峰時段停站時間之特性顯示於各站之比值分布情形，由所蒐集之資料顯示，經去除異常事件日資料後，各站樣本資料之比值超過 1000%特別異常之比例已由原先之 0.42%降低至約 0.38%，大部分資料之比值更集中約介於 20%~280%之間，若以比值介於 80%~120%資料為例，其比例可達約 28%。

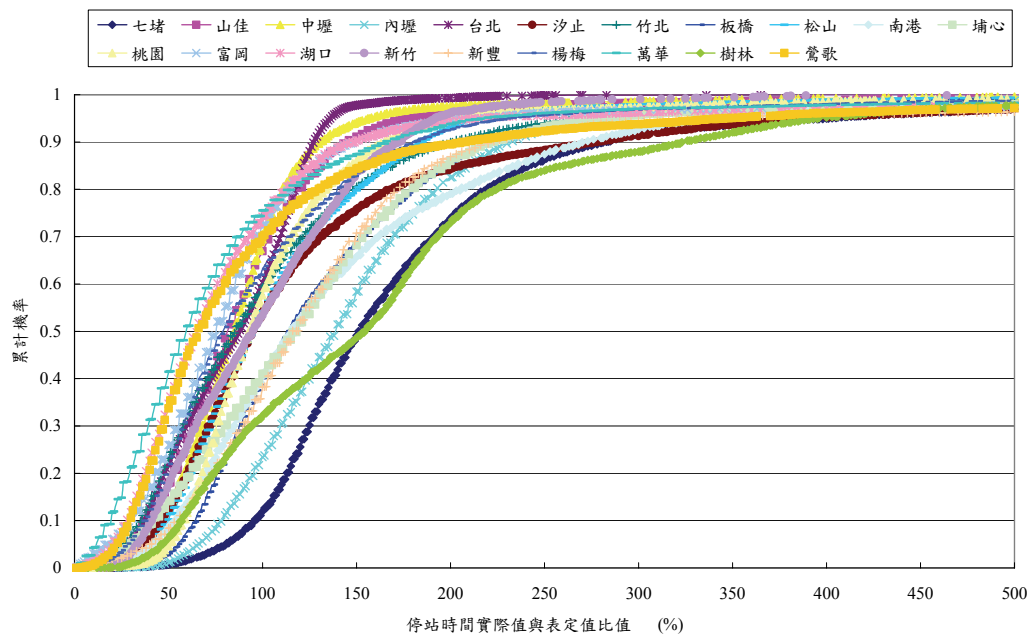


圖 4-6 通勤電聯車於離峰時段之停站時間分布圖

4.4 隨機特性連鎖延滯模擬結果

經過前述延滯特性參數分布分析結果得知，本研究將以 4.3 節去除 16 天異常事件日之延滯特性參數分布資料進行後續連鎖延滯模擬分析，並依據站間運轉時間與停站時間之關鍵延滯參數分布特性，區分為對號列車及通勤電聯車等二種車種，及尖、離峰等二種時段，分別針對研究案例臺鐵西部幹線七堵站至新竹站區間「所有車站」及「二端末車站」之連鎖延滯進行分析。

有關前述連鎖延滯分析之情境假設包括：尖峰時段定義為每日上午 6~9 點及下午 4~7 點，其餘則為離峰時段；另尖峰情境之連鎖延滯分析係假設於上午 7:15 於南下松山—臺北站間有 1 小時之初始延滯，而離峰情境之連鎖延滯分析則假設於上午 10:15 於南下松山—臺北站間有 1 小時之初始延滯。

至於隨機站間運轉時間之擷取係依據前述 4.3 節不同車種之站間運轉時間分布「總計」曲線，依產生之亂數對應而得；而隨機停站時間之擷取作法類似，但需依不同車種、尖離峰時段及不同車站，依產生之亂數對應而得。有關各情境之模擬分析結果茲分述如下。

4.4.1 所有車站之連鎖延滯模擬結果

本案例分析是依據 4.3 節之各項隨機資料，及班表列車順序及前述各項情境假設，分別依據上午 7:15 (尖峰) 及 10:15 (離峰) 於南下松山—臺北站間有 1 小時之初始延滯假設，模擬得到產生連鎖延滯之模擬班表，最後再與表定班表各列車時間比較，即可得到所有車站之連鎖延滯模擬結果。另由於隨機模擬結果具有擾動震盪之特性，故本研究由系統亂數種子所產生的 1,000 組亂數進行連鎖延滯模擬推估，並透過連鎖延滯推估量逐次累計再取單次模擬平均值之作法，嘗試尋找連鎖延滯之收斂值，並彙整七堵—新竹區間於尖、離峰時段有 1 小時初始延滯所有 20 個車站之連鎖延滯模擬結果。

由彙整結果可發現，當南下松山—臺北站間於尖峰時段有 1 小時之初始延滯發生時，其所有 20 個車站之連鎖延滯值約經過 100 次模擬即會收斂於約 173 小時。另當同樣南下松山—臺北站間於離峰時段有 1 小時之初始延滯發生時，其所有 20 個車站之連鎖延滯值約經過 123 次模擬即會收斂於約 153 小時，其連鎖延滯模擬值分布及收斂情形如圖 4-7 所示；而由圖中模擬值亦可發現，尖峰之連鎖延滯模擬值始終大於離峰之連鎖延滯模擬值，可見模擬結果亦符合先驗知識。

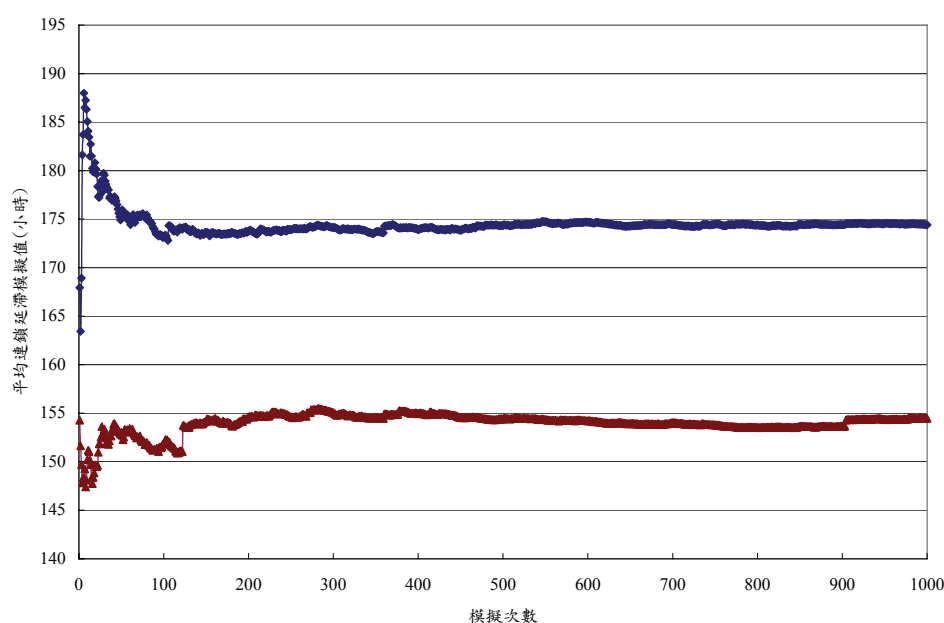


圖 4-7 所有車站之連鎖延滯模擬值分布及收斂情形

4.4.2 二端末車站之連鎖延滯模擬結果

有關二端末車站之連鎖延滯模擬作法，如同前述所有車站之作法及各項情境假設，分別依據上午 7:15 (尖峰) 及 10:15 (離峰) 於南下松山—臺北站間有 1 小時初始延滯假設，模擬得到產生連鎖延滯之模擬班表，最後針對分析路段七堵—新竹路段二端末車站各列車時間與表定班表各列車時間比較，即可得到二端末車站連鎖延滯模擬結果。另本分析亦由系統亂數種子所產生的 1,000 組亂數進行連鎖延滯模擬推估，並透過連鎖延滯推估量逐次累計再取單次模擬平均値之作法，可得到連鎖延滯之收斂值，並彙整七堵—新竹區間於尖、離峰時段有 1 小時初始延滯於七堵及新竹二端末車站之合計連鎖延滯模擬結果。

由彙整結果可發現，當南下松山—臺北站間於尖峰有 1 小時初始延滯發生時，其七堵及新竹二端末車站之連鎖延滯值約經過 108 次模擬即會收斂於約 32.7 小時。另當同樣南下松山—臺北站間於離峰有 1 小時之初始延滯發生時，其七堵及新竹二端末車站之連鎖延滯值約經過 175 次模擬即會收斂於約 28.5 小時，其連鎖延滯模擬值分布及收斂情形如圖 4-8 所示；而由圖中模擬值亦可發現，尖峰之連鎖延滯模擬值始終大於離峰之連鎖延滯模擬值，其模擬結果係符合先驗知識。

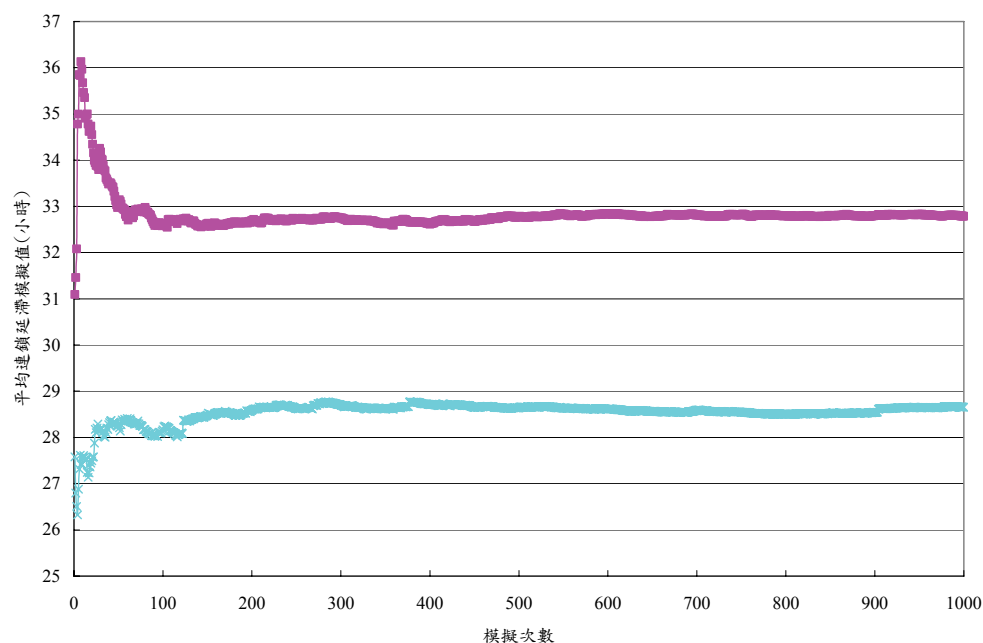


圖 4-8 七堵及新竹二端末車站之連鎖延滯模擬值分布及收斂情形

4.5 「站間容許列車數」連鎖延滯敏感度分析

有關站間容許列車數之分析情境係承襲前述 4.3 節之隨機參數資料為基礎，進一步分析「站間容許列車數」參數的敏感度。為利分析，以不同之「站間容許列車數」搭配不同之運轉調度策略，分析其對連鎖延滯之影響。圖 4-9 與圖 4-10 中 x 軸為「站間容許列車數」，該值分別設定 1~5，分析四種調度策略（無策略、站間趕點、站內趕點、站間+站內趕點）下之所有車站延滯及最末站延滯。

由該圖之分析結果可發現，當站間容許列車數由 1 提昇至 2 時，總延滯時間大幅降低(即系統可靠度大幅提昇)；但當站間容許列車數由 2 提昇至 3 或 4 時，則總延滯時間僅小幅降低(即系統可靠度僅有小幅提昇)。由此可知，站間閉塞區間之數量即使大幅增加，整體之系統可靠度亦不一定大幅提升，因為整體路段之運轉及容量仍未有效提升。

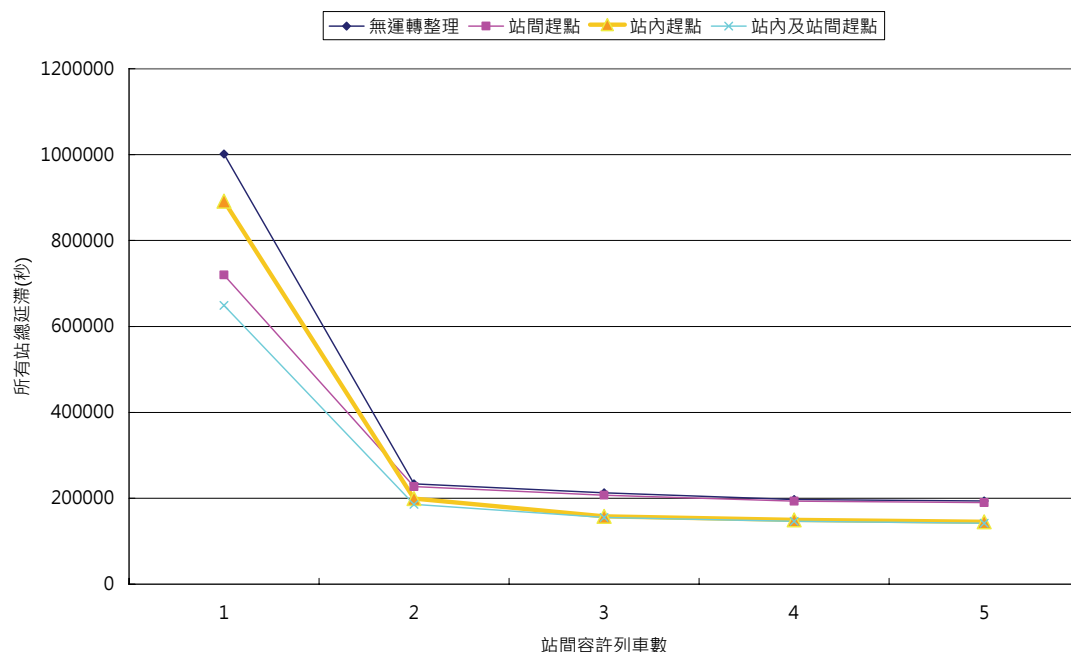


圖 4-9 站間容許列車數搭配不同調度策略之所有站總延滯比較圖

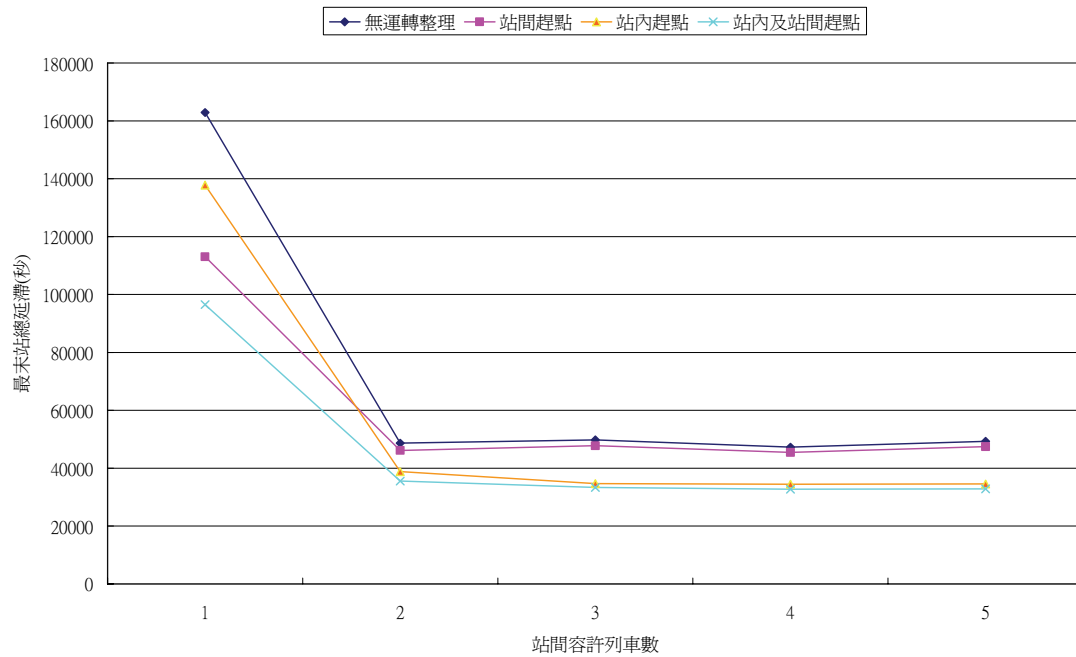


圖 4-10 站間容許列車數搭配不同調度策略之最末站總延滯比較圖

另為分析各種不同初始延滯程度下，「站間容許列車數」所帶來的效益，本研究另以「站間＋站內趕點」之情境進行分析。圖 4-11 及圖 4-12 即為以 0.5hr、1.0hr、1.5hr、2.0hr、2.5hr、3.0hr 之情境所分析之總延滯時間，由圖可看出不同初始延滯及「站間容許列車數」雖然會有不同變化，但仍可得到前述兩點相同之結論。

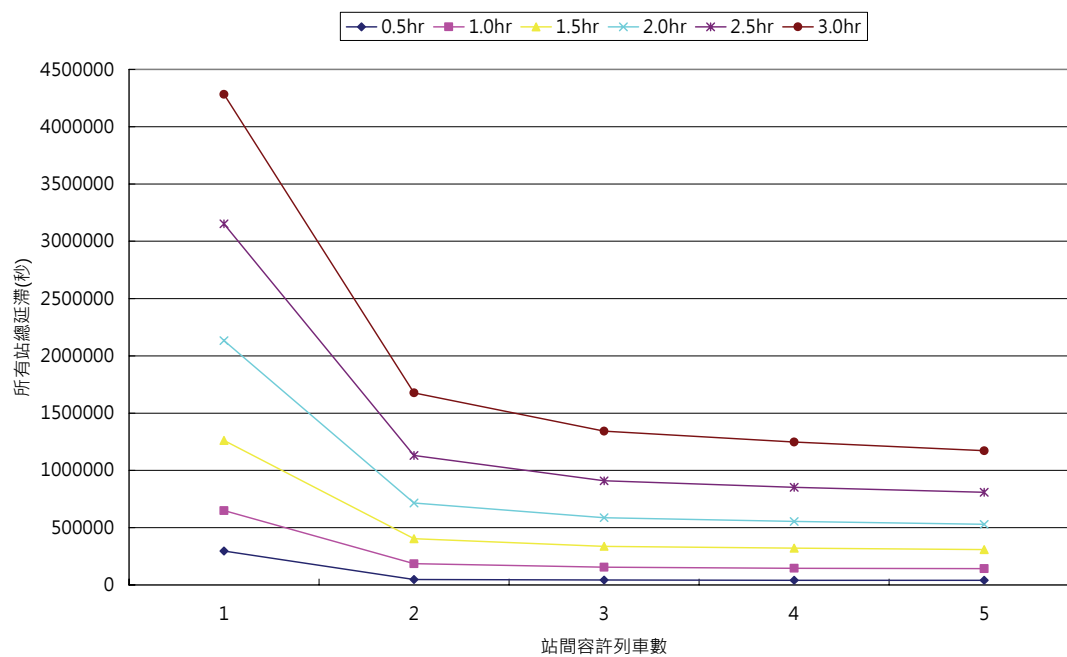


圖 4-11 站間容許列車數搭配不同程度初始延滯之所有站總延滯比較

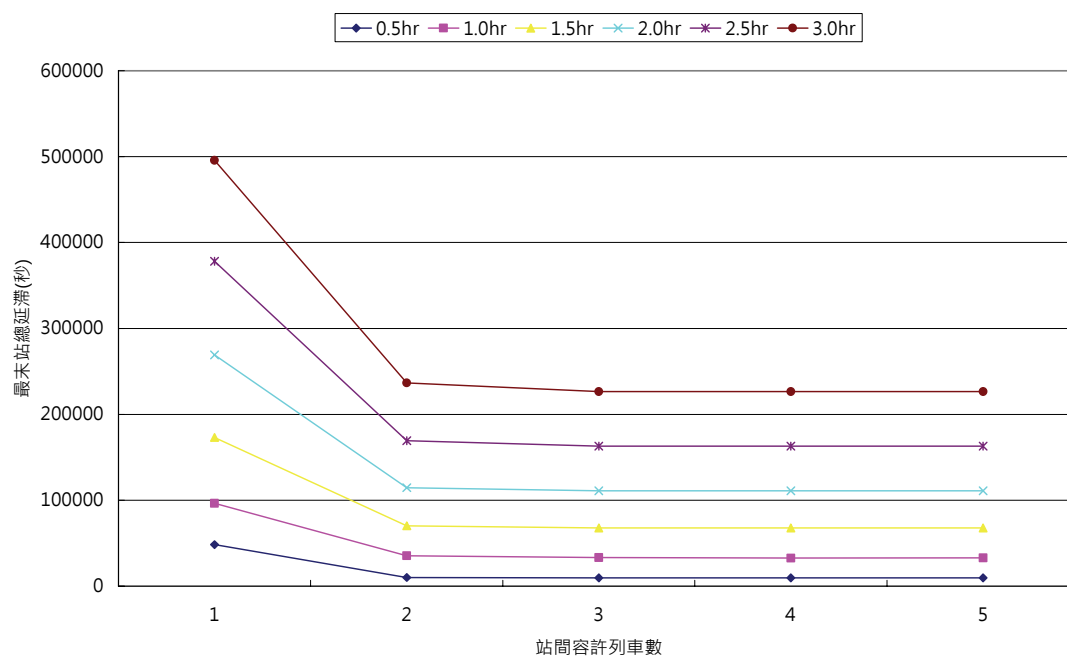


圖 4-12 站間容許列車數搭配不同程度初始延滯之最末站總延滯比較

4.6 二次初始延滯對連鎖延滯之衝擊波分析

前述有關連鎖延滯之分析皆為在一次初始延滯之假設情境進行分析，惟為了解實務上 2 次以上初始延滯下其彼此交互作用是否對連鎖延滯是否產生影響，本研究續以所構建之模擬模式及相關資料進行 2 次初始延滯對連鎖延滯之衝擊波分析。

先前本研究所開發之臺鐵系統營運可靠度暨延滯最佳化模式應用程式僅能處理單一初始延滯對連鎖延滯之影響問題，故在本研究裡將其延伸成可處理 2 次初始延滯，並以本節所述之情境資料進行案例分析。

4.6.1 衝擊波圖形簡介

本研究所謂的衝擊波是觀察不同時段（x 軸）的延滯總量（y 軸）所繪製出的波型，波型越高代表延滯越嚴重；波型越寬代表影響時間越長。若以假設基本情境（上午 7:10 於松山台北間發生 1 小時之初始延滯）搭配 4 種調度策略（無策略、站間趕點、站內趕點、站間＋站內趕點）進行分析，結果可繪製衝擊波圖型如圖 4-13。

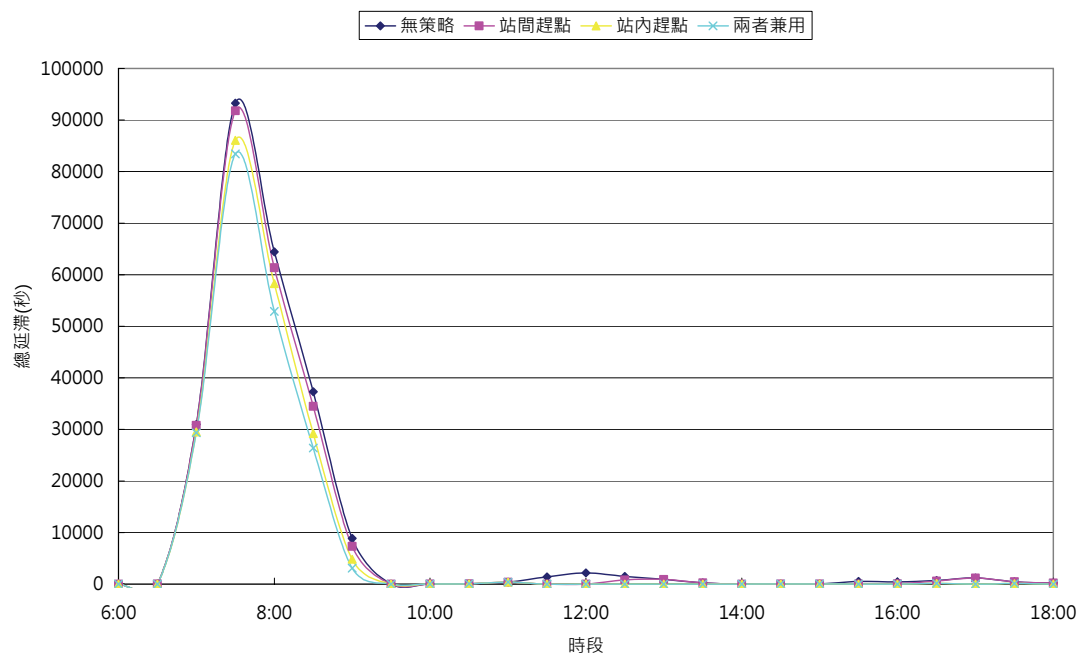


圖 4-13 不同調度策略下之衝擊波比較圖

如前所述，初始延滯越大衝擊波的振幅就會越高，圖 4-14 比較各種不同初始延滯下的衝擊波，的確可觀察到上述現象，且若將圖 4-14 的 x 軸放大展開如圖 4-15，將可更清楚觀察波型的成長變化。

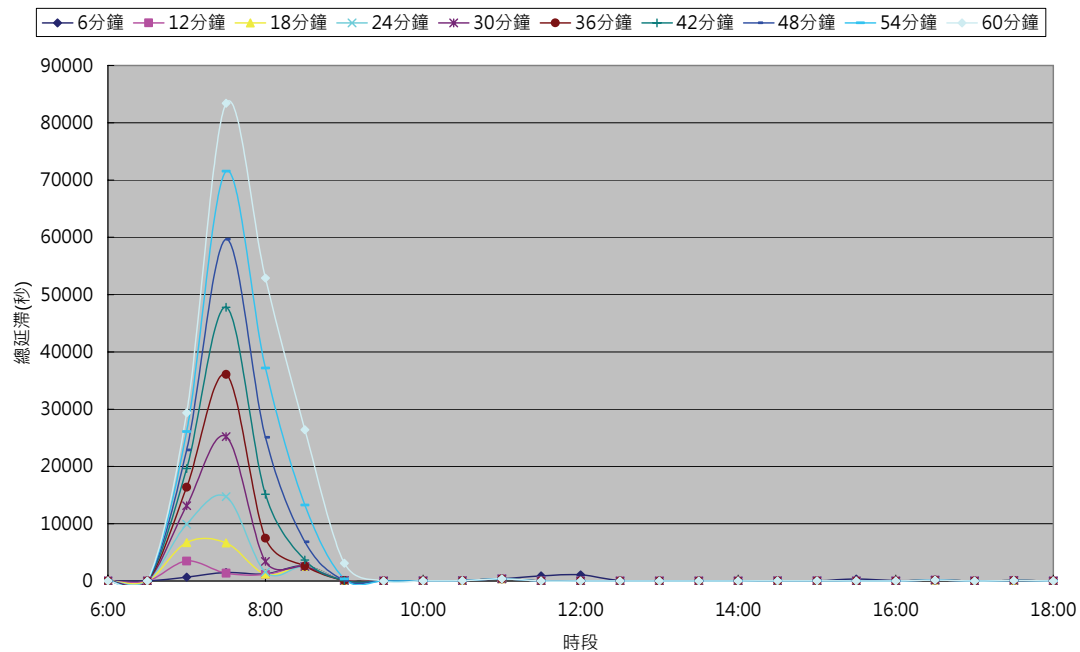


圖 4-14 不同程度初始延滯下的衝擊波比較

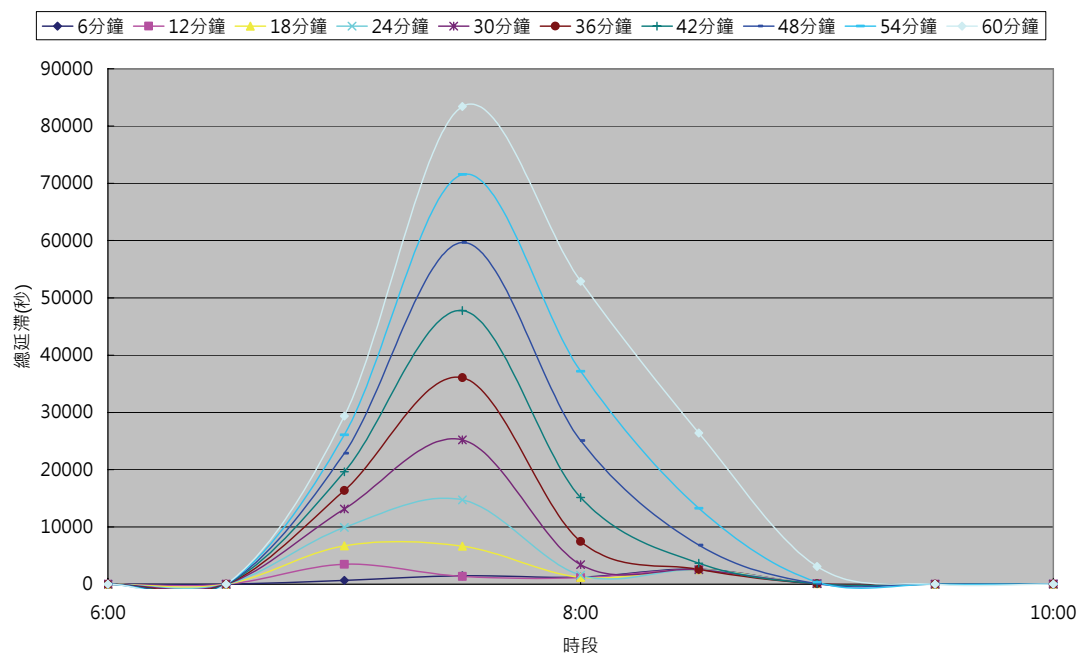


圖 4-15 不同程度初始延滯下的衝擊波比較（局部放大）

另為利比較各站之衝擊波狀況，本研究亦逐站分析計算衝擊波的數值，圖 4-16 係分別計算七堵、百福等 9 站的衝擊波，圖 4-17 則進一步將 x 軸放大設定在上午 6 點至 10 點，以利可明顯觀察各站所受之不同到連鎖延滯衝擊波影響。

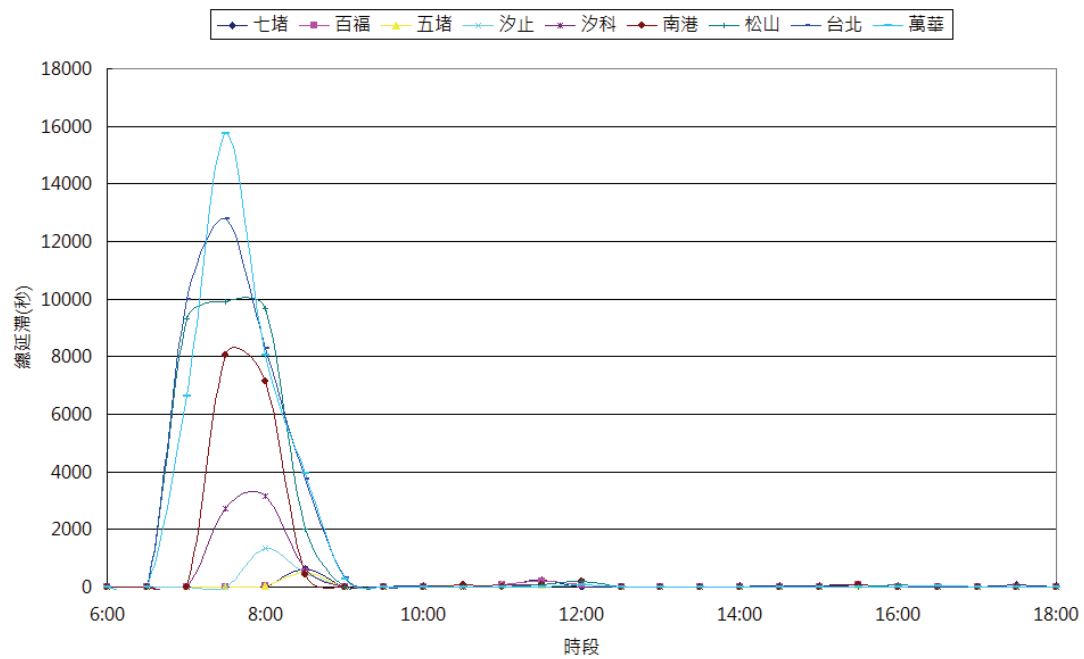


圖 4-16 不同車站之衝擊波比較

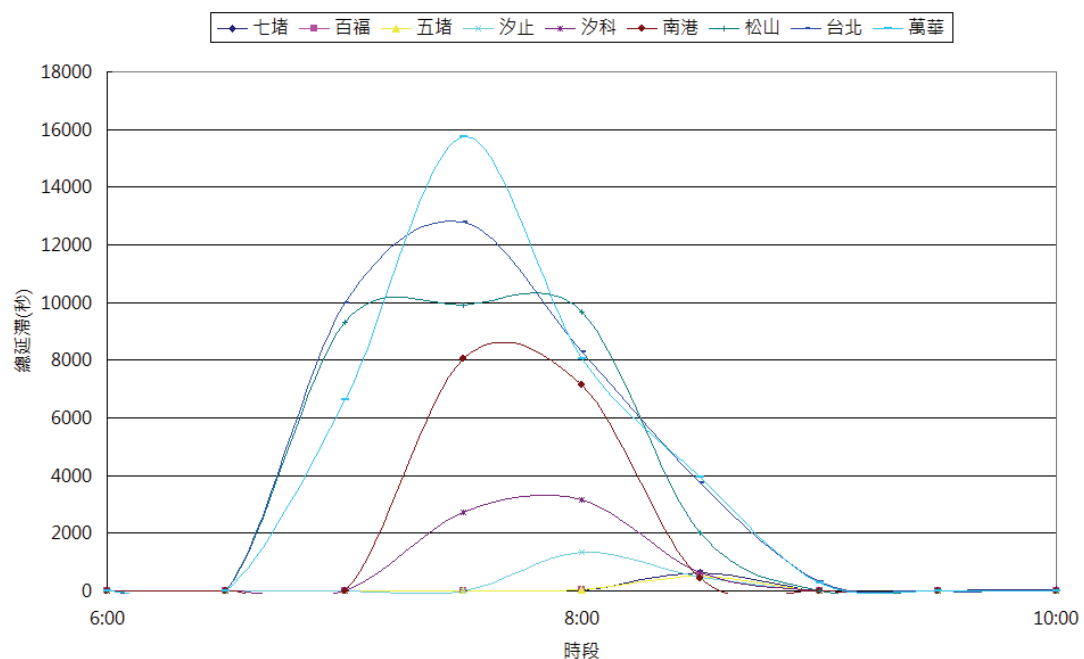


圖 4-17 不同車站衝擊波比較（局部放大）

4.6.2 二次初始延滯之衝擊波圖型分析

接續前述衝擊波圖型理論，本研究進一步以衝擊波圖型繪製 2 次初始延滯的分析結果，圖 4-18 即是假設 2 次初始延滯在相隔 8 小時的情況下所模擬之連鎖延滯衝擊波分析結果，圖中可發現 2 個波峰，但由於第 2 次初始延滯的假設發生時間並非在尖峰時段，故對系統的衝擊較小、回復時間也較快。

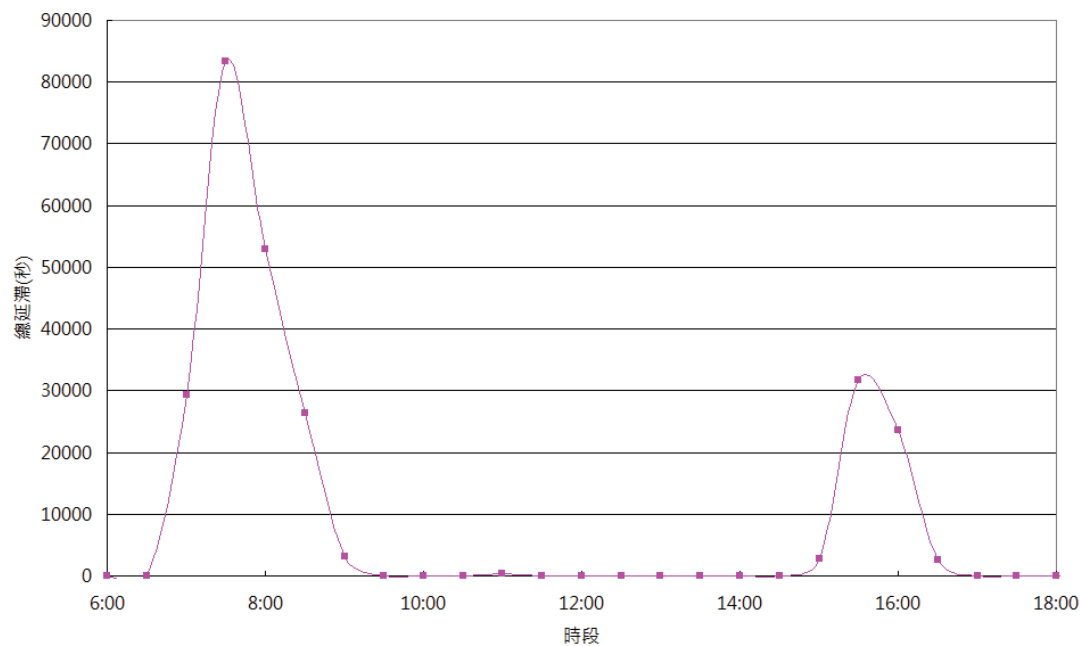


圖 4-18 初始延滯間隔 8 小時之衝擊波

另為分析 2 次初始延滯對連鎖延滯之衝擊波交互作用影響，本研究進一步將 2 次初始延滯之發生時間縮短，圖 4-19 則選擇南港站初始延滯間隔 2~9 小時進行連鎖延滯衝擊波分析，其假設上午 7:10 於南下松山—台北間發生 1 小時之第 1 次初始延滯，從中可發現不管間隔幾小時均是雙峰的圖型，但第 2 次峰的高度則略有不同。而隨著 2 次初始延滯的接近，2 個波會相互影響並造成更高的延滯，為了明確觀察其中變化，圖 4-20 以每 6 分鐘為間隔，分別模擬 1.0 小時、1.1 小時、1.2 小時直到 2.0 小時的衝擊波變化，從中發現當 2 次初始延滯發生時間(即雙峰)靠近至 1.4 小時，雙峰就合併成單峰且峰度越高，顯示 2 次初始延滯之影響發生加成效果，並大幅衝擊系統的可靠度。

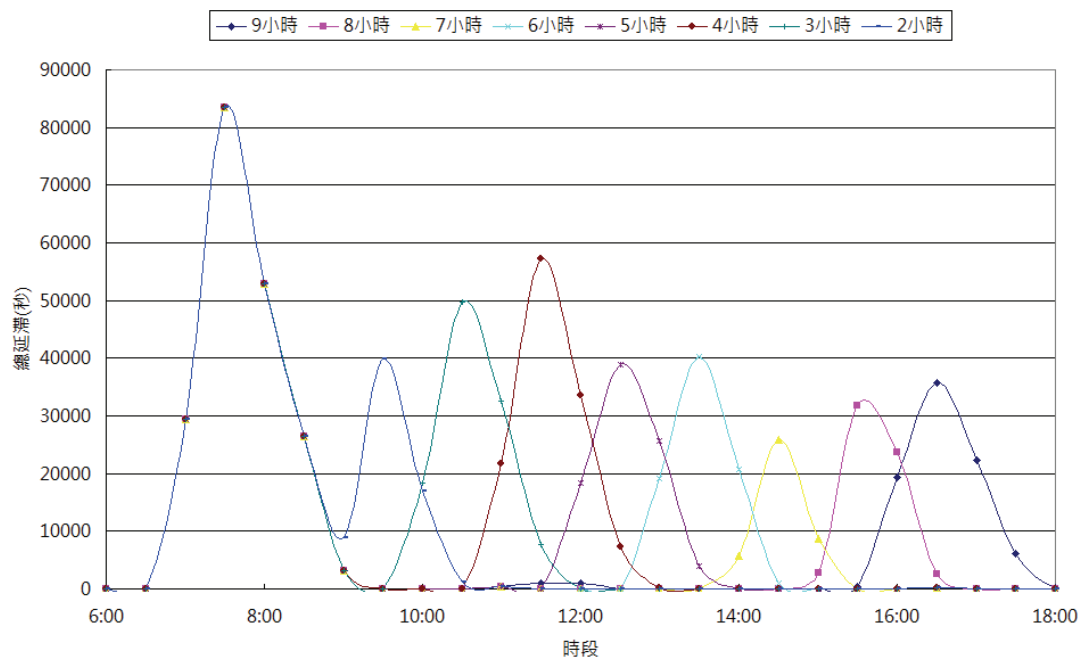


圖 4-19 各種初始延滯間隔值(2~9hr)之衝擊波

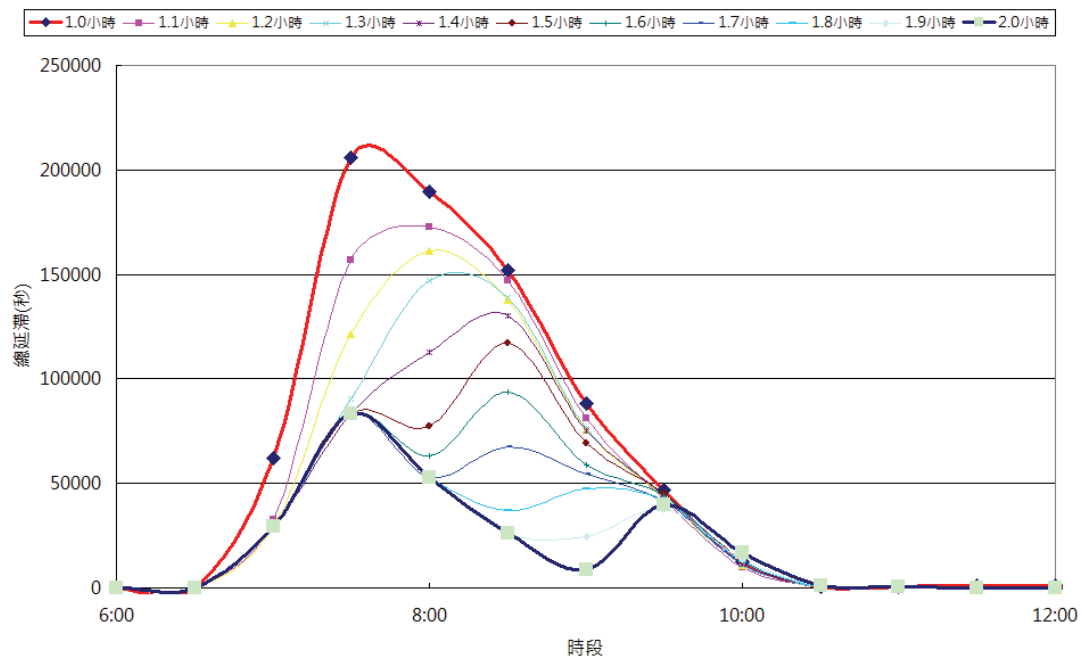


圖 4-20 各種初始延滯間隔值(1~2hr)之衝擊波

第五章 結論與建議

鐵路系統容量不足往往是導致列車延滯之主因，尤其在初始延滯(first delay)發生後，後續所引發的連鎖延滯(knock-on delay)擴散效應對列車正常營運之影響甚大，因此欲確保系統之可靠度及服務品質，快速有效地降低連鎖延滯一直是重要課題。為有效釐清造成連鎖延滯之複雜因素及其交互作用，本研究依據臺鐵系統之運轉特性構建一模擬模式用以推估連鎖延滯，並以臺鐵西部幹線北部路段為案例，分析連鎖延滯之各種相關問題。鑑於影響連鎖延滯之部分關鍵因素具有隨機特性，故為確實反映臺鐵列車實際運轉因初始延滯發生後所產生之連鎖延滯特性，及深入探討產生連鎖延滯之班表穩定度問題，本研究進一步分析站間運轉時間及停站時間之隨機特性，並將其納入模擬模式推估連鎖延滯。另本研究亦針對站間容許列車數對連鎖延滯之敏感度，及二次初始延滯對連鎖延滯之衝擊波進行分析，相關研究結論及建議茲彙整如下。

5.1 結論

1. 本研究之重點在於考量站間運轉時間及停站時間所具有之隨機特性，並確實反映臺鐵列車實際運轉之交互作用及可能產生之連鎖延滯狀況，研究成果可作為深入探討初始延滯發生後所產生連鎖延滯之班表穩定度的基礎。若能再配合本模擬模式所具有的功能，進一步探討列車密度/初始延滯/調度策略等關鍵因素對連鎖延滯之影響情形，以充分掌握不同路線條件、交通條件及控制條件之複雜交互作用，及其對連鎖延滯之擴散效應與各種運轉調度策略之改善成效評估（黃承傳與劉昭榮，2011），將可作為後續營運單位排班規劃、系統可靠度分析及服務品質改善之參考。
2. 由於分析路段間各站之隨機停站時間資料分布變異頗大，顯示臺鐵系統之營運資料內部隱含許多「初始延滯」及「連鎖延滯」，故本研究需先將所蒐集時段內過年及地震等旅次特性異常之樣本去

除。此外，因發現其資料可能存在時段差異特性，故再將該資料分類為尖、離峰時段分開處理，再分別進行後續之連鎖延滯模擬分析，其結果更為合宜。

- 3.經假設上午尖峰時段內之 7:15 及離峰時段內之 10:15 於南下松山—臺北站間有 1 小時之初始延滯，經本模擬模式推估結果顯示，無論所有車站或二端末車站(七堵及新竹站)之連鎖延滯模擬其平均值皆會穩定接近於一定值，其中所有車站之連鎖延滯於尖、離峰有初始延滯情境下，其平均模擬值分別約接近於 173 小時及 153 小時；而二端末車站之連鎖延滯於尖、離峰有初始延滯情境下，其平均模擬值則分別約接近於 32.7 小時及 28.5 小時，相關分析結果皆符合所有車站大於二端末車站之連鎖延滯，及尖峰大於離峰之連鎖延滯等先驗知識，應可作為營運單位實際運轉調度之參考。
- 4.由站間容許列車數對連鎖延滯敏感度之分析結果可發現，當站間容許列車數由 1 提昇至 2 時，總延滯時間大幅降低(即系統可靠度大幅提昇)；但當站間容許列車數由 2 提昇至 3 或 4 時，則總延滯時間僅小幅降低(即系統可靠度僅有小幅提昇)。由此可知，站間閉塞區間之數量即使大幅增加，整體之系統可靠度亦不一定大幅提升，因為整體路段之運轉及容量並未有效提升，仍需由整個路段之站間及站內閉塞區間及股道配置狀況共同決定路線容量及可靠度。
- 6.有關 2 次初始延滯對連鎖延滯之衝擊波分析，研究結果顯示若選擇南港站之連鎖延滯進行分析，並假設上午 7:10 於南下松山—台北間發生 1 小時之第 1 次初始延滯，當第 2 次初始延滯於該區間發生之時間夠接近時，其對連鎖延滯之衝擊波將產生交互作用影響，且當 2 次初始延滯發生時間(即雙峰)逐漸靠近至 1.4 小時，雙峰就合併成單峰且峰度越高，顯示 2 次初始延滯之影響發生加成效果，並大幅衝擊系統的可靠度。

5.2 建議

- 1.本研究於分析過程中發現，臺鐵目前之營運資料中隱含有延滯(包

括初始延滯及連鎖延滯) 成分在內，需耗費許多時間先加以有效處理，始能應用於模擬模式建構及案例分析。由於列車運轉之許多關鍵參數皆具有隨機特性，同樣亦面臨蒐集資料前之複雜處理工作(如扣除有重大事件之營運資料)，極為費時。由於各類型延滯量及相關重要參數資料皆屬延滯相關研究所必須，故建議營運單位宜設法建立完善之延滯資料庫，並提供初步之延滯時空及原因分析，以利後續之相關研究分析。

2. 本研究雖已針對營運單位實務上最常採用之站間趕點及站內趕點等運轉調度策略，對於連鎖延滯之改善效果進行模擬分析，惟尚有許多運轉整理策略(如減班、變換列車順序、改變列車交通組成等)、單線雙向運轉調度策略、初始延滯發生於站內股道之列車行駛股道選擇及運轉時隔等相關問題值得探討，建議可以持續加以研究分析，以補足對連鎖延滯各面向議題之深入了解。
3. 由於路線、交通、控制等三大類條件之交互作用皆會影響整體列車營運之「連鎖延滯」，各項因素彼此間之交互作用相當複雜不易掌握分析，且由於模擬模式具依個案所需輸入資料不同且數量極大之特性，故除前述 CTC 營運資料庫之建立外，若欲進行軟、硬體之路線、交通、控制等各項條件改善對於連鎖延滯之全方位影響分析時，將無法因應。建議仍需儘速結合臺鐵系統之運務、工務、機務、電務等四大部門建立整合型臺鐵系統營運資料庫，以符所需。
4. 本研究雖已針對二次初始延滯對連鎖延滯之衝擊波進行分析，惟實務上軌道系統運轉仍存在多次初始延滯及不同時空條件之初始延滯複雜狀況，故後續若能將本模擬模式擴充應用於處理多次及不同時空條件初始延滯之連鎖延滯分析，並配合衝擊波之分析呈現，將更能即時提供營運單位之運轉調度參考。

參考文獻

- 李治綱、鍾志成、林杜寰、張仕龍、張恩輔、陳一昌、張開國、吳熙仁 (2009), 「公共運輸之安全績效：臺灣鐵管理局之個案分析」, *運輸計劃季刊*, 第 38 卷, 第 4 期, 頁 381-406。
- 周斯畏 (2002), *物件導向系統分析與設計使用 UML 與 C++*, 臺北市：全華科技圖書股份有限公司。
- 鍾志成 (2005), 軌道容量研究－臺鐵系統容量模式之建構分析 (一), 交通部運輸研究所委託研究。
- 鍾志成 (2008), 運輸系統容量分析暨應用研究—軌道系統 (2/4), 交通部運輸研究所委託研究。
- 鍾志成 (2011), 軌道系統容量與可靠度分析研究 (1/3), 交通部運輸研究所委託研究。
- 黃承傳、劉昭榮 (2011), 「鐵路列車密度與初始延滯以及調度策略對連鎖延滯之影響分析」, *運輸計劃季刊*, 第 40 卷, 第 1 期, 頁 63-98。
- Barter, W. M. (1998), Application of Computer Simulation to Rail Capacity Planning, In Brebbia, C. A. *et al.* (Eds.), *Computers in Railways VI*, Southampton: WIT Press, pp. 199-211.
- Briggs, K. and Beck, C. (2007), “Modeling Train Delays with Q-exponential Functions,” *Physica A: Statistical Mechanics and its Applications*, Vol. 378, No. 2, pp. 498-504.
- Carey, M. (1999), “Ex ante Heuristic Measures of Schedule Reliability,” *Transportation Research Part B: Methodological*, Vol. 33, No. 7, pp. 473-494.
- Carey, M. and Carville, S. (2000), “Testing Schedule Performance and Reliability for Train Stations,” *Journal of the Operational Research Society*, Vol. 51, No. 6, pp. 666-682.

- Carey, M. and Crawford, I. (2007), "Scheduling Trains on a Network of Busy Complex Stations," *Transportation Research Part B: Methodological*, Vol. 41, No. 2, pp. 159-178.
- EuROPE-TRIP (2000), *European Railways Optimisation Planning Environment - Transportation Railways Integrated Planning*, RA-97-AM-1165, Ferrovie dello Stato Spa, Divisione Infrastruttura, Roma.
- Hansen, I. A. (2000), Station Capacity and Stability of Train Operations, In Allan, J. *et al.* (Eds.), *Computers in Railways VII*, Southampton: WIT Press, pp. 809-816.
- Hansen, I. A. and Pachl, J. (2008), *Railway Timetable and Traffic: Analysis – Modelling – Simulation*, Hamburg: Railway Gazette.
- Higgins, A., Kozan, E., and Ferreira, L. (1995), "Modeling Delay Risks Associated with Train Schedules," *Transportation Planning and Technology*, Vol. 19, No. 2, pp. 89-108.
- Higgins, A. and Kozan, E. (1998), "Modeling Train Delay in Urban Networks," *Transportation Science*, Vol. 32, No. 4, pp. 346-357.
- Huisman, T. and Boucherie, R. J. (2001), "Running Times on Railway Sections with Heterogeneous Train Traffic," *Transportation Research Part B: Methodological*, Vol. 35, No. 3, pp. 271-292.
- Hwang, C. C. and Liu, J. R. (2010), "A Simulation Model for Estimating Knock-on Delay of Taiwan Regional Railway," *Journal of the Eastern Asia Society for Transportation Studies*, Vol. 8, pp. 1082-1097.
- Jong, J. C., Lin, T. H., Lee, C. K., and Hu, H. L. (2010), "The Analysis on Train Reliability of Taiwan High Speed Rail," *Proceedings, 12th COMPRAIL Conference*, pp. 169-180.
- Middelkoop, D. and Bouwman, M. (2002), Testing the Stability of the Rail Network, In Allan, J. *et al.* (Eds.), *Computers in Railways VIII*, Southampton: WIT Press, pp. 995-1002.
- Olsson, N. O. E. and Haugland, H. (2004), "Influencing Factors on Train Punctuality—Results from Some Norwegian Studies," *Transport Policy*, Vol. 11, No. 4, pp. 387-397.

- Vromans, M. J., Dekker, R., and Kroon, L. G. (2006), “Reliability and Heterogeneity of Railway Services,” *European Journal of Operational Research*, Vol. 172, No. 2, pp. 647-665.
- Yun, B., Tinkin, H., and Baohua, M. (2011), “Train Control to Reduce Delays upon Service Disturbances at Railway Junctions,” *Journal of Transportation Systems Engineering and Information Technology*, Vol. 11, No. 5, pp. 114-122.

附錄 程式碼及模擬結果說明

壹、原始程式碼

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ClassLibrary1
{
    public class TrainEvent : IComparable<TrainEvent>
    {
        public static int Num = 0;
        public int CompareTo(TrainEvent otherEvent)
        {
            if (Time > otherEvent.Time)
            {
                return 1;
            }
            else if (Time < otherEvent.Time)
            {
                return -1;
            }
            else
            {
                return m_SerNum.CompareTo(otherEvent.m_SerNum);
            }
        }

        private Directions m_Dir;
        private int m_Time = -RailSystem.DayTime;
        private EventTypes m_Type;
        private string m_TrainID = null;
        private string m_Station = null;
        private int m_SerNum = 0;

        public TrainEvent(string TrainID, int Time, string Station, EventTypes Type, Directions
Dir)
        {
            Num++;
            m_TrainID = TrainID;
            m_Time = Time;
            m_Station = Station;
            m_Type = Type;
            m_Dir = Dir;
            m_SerNum = Num;
        }

        public string TrainID
        {
            get
            {
                return m_TrainID;
            }
            set
            {
                m_TrainID = value;
            }
        }

        public string Station
        {
            get
            {
```

```

        return m_Station;
    }
    set
    {
        m_Station = value;
    }
}

public int Time
{
    get
    {
        return m_Time;
    }
    set
    {
        m_Time = value;
    }
}

public Directions Dir
{
    get
    {
        return m_Dir;
    }
    set
    {
        m_Dir = value;
    }
}

public EventTypes Type
{
    get
    {
        return m_Type;
    }
    set
    {
        m_Type = value;
    }
}
}

}

using System;
using System.Collections.Generic;
using System.Text;

namespace ClassLibrary1
{
    public class DwellPlan
    {
        public string station;
        public int expArrTime;
        public int expDepTime;
        public int actArrTime;

        public int actDepTime;

        public DwellPlan(string stationName, int arrivalTime, int departureTime)
        {
            station = stationName;
            expArrTime = arrivalTime;
            expDepTime = departureTime;
            actArrTime = int.MinValue;
            actDepTime = int.MinValue;
        }
    }
}

```



```

public class Train
{
    private int m_priority;
    public int Priority
    {
        get { return m_priority; }
        set { m_priority = value; }
    }

    private List<DwellPlan> m_dwellPlanList;
    public List<DwellPlan> DwellPlanList
    {
        get { return m_dwellPlanList; }
    }

    public Train()
    {
        m_dwellPlanList = new List<DwellPlan>();
    }

    private Directions m_Dir;
    public Directions Dir
    {
        get { return m_Dir; }
        set { m_Dir = value; }
    }

    private int m_Task=0;
    public int Task
    {
        get { return m_Task; }
    }
    public bool NextTask(){
        m_Task++;
        if (m_Task < DwellPlanList.Count)
            return true;
        else
            return false;
    }
    public void ResetTask()
    {
        m_Task = 0;
    }

    private string m_Location = null;
    public string Location
    {
        get { return m_Location; }
        set { m_Location = value; }
    }

    public int IsArrDelay(int Time)
    {
        return Time - DwellPlanList[Task].expArrTime;
    }

    public int IsDepDelay(int Time)
    {
        return Time - DwellPlanList[Task].expDepTime;
    }

    public int DefaultDwellTime()
    {
        return DwellPlanList[Task].expDepTime - DwellPlanList[Task].expArrTime;
    }
    public int DefaultRunningTime()
    {
        return DwellPlanList[Task].expArrTime - DwellPlanList[Task-1].expDepTime;
    }
}
}
using System;
using System.Collections.Generic;

```

```

using System.Text;
using System.Diagnostics;

namespace ClassLibrary1
{
    public enum EventTypes { ARRIVE, DEPART, PASS, FD_START, FD_STOP };
    public enum Directions { ASC=0, DESC=1 };

    public class Tracks
    {
        public Tracks(string Name)
        {
            m_Name = Name;
            Reset();
        }
        private string m_Name;
        public string Name
        {
            get { return m_Name; }
        }

        private bool m_blocked;
        public bool Blocked
        {
            get { return m_blocked; }
            set { m_blocked = value; }
        }

        private string m_occupied;
        public string Occupied
        {
            get { return m_occupied; }
        }

        private EventTypes m_type;
        public EventTypes LastEventType
        {
            get { return m_type; }
        }

        private Directions m_direction;
        public Directions LastEventDirection
        {
            get { return m_direction; }
        }

        private int m_time = -RailSystem.DayTime;
        public int LastEventTime
        {
            get { return m_time; }
        }

        private int m_LastArrTime = -RailSystem.DayTime;
        public int LastArrTime
        {
            get { return m_LastArrTime; }
        }

        public void Reset()
        {
            m_time = -RailSystem.DayTime;
            m_occupied = null;
            m_blocked = false;
            m_LastArrTime = -RailSystem.DayTime;
        }

        public void SetLastEvent(ref TrainEvent TE)
        {
            m_type = TE.Type;
            m_time = TE.Time;
            m_direction = TE.Dir;
        }
    }
}

```

```

        if (EventTypes.ARRIVE==m_type)
        {
            m_LastArrTime = TE.Time;
            m_occupied = TE.TrainID;
        }
        else if (EventTypes.DEPART == m_type)
        {
            Debug.Assert(m_occupied == TE.TrainID);
            m_occupied = null;
        }
        else if (EventTypes.PASS == m_type)
        {
            m_occupied = null;
        }
        else
        {
            Debug.Assert(false);
        }
    }
}

}

using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;

namespace ClassLibrary1
{
    public class Station
    {
        public int totalDelay = 0;

        private int m_headwayDA;
        public int H_DA
        {
            get { return m_headwayDA; }
            set { m_headwayDA = value; }
        }

        private int m_headwayAA;
        public int H_AA
        {
            get { return m_headwayAA; }
            set { m_headwayAA = value; }
        }

        private int m_headwayDD;
        public int H_DD
        {
            get { return m_headwayDD; }
            set { m_headwayDD = value; }
        }

        private int m_headwayAD_R;
        public int H_AD_R
        {
            get { return m_headwayAD_R; }
            set { m_headwayAD_R = value; }
        }

        private int m_headwayDA_R;
        public int H_DA_R
        {
            get { return m_headwayDA_R; }
            set { m_headwayDA_R = value; }
        }

        protected Tracks m_track1;
        public Tracks Track1
        {
            get { return m_track1; }
        }
    }
}

```

```

    }

    protected Tracks m_track2;
    public Tracks Track2
    {
        get { return m_track2; }
    }

    private int m_mileage;
    /// <summary>
    /// </summary>
    public int Mileage
    {
        get { return m_mileage; }
        set { m_mileage = value; }
    }

    private int m_minDwellTime;
    public int MinDwellTime
    {
        get { return m_minDwellTime; }
        set { m_minDwellTime = value; }
    }

    public Station()
    {
        m_track1 = new Tracks("Track1");
        m_track2 = new Tracks("Track2");
    }

    public virtual string GetStationType()
    {
        return "Type IV";
    }

    protected bool CheckTrackNoneOccupy(ref Tracks track)
    {
        if (null == track.Occupied && !track.Blocked)
            return true;
        else
            return false;
    }

    protected bool CheckDA(ref Tracks track, int Time)
    {
        if ( this.H_DA <= (Time - track.LastEventTime) )
            return true;
        else
            return false;
    }

    protected bool CheckAA(ref TrainEvent TE, ref Tracks AnotherOne)
    {
        if (AnotherOne.LastEventTime < 0)
            return true;
        if (AnotherOne.LastEventDirection != TE.Dir)
            return true;

        if ((TE.Time - AnotherOne.LastArrtTime) >= H_AA)
        {
            return true;
        }
        else {
            return false;
        }
    }

    if (EventTypes.ARRIVE == AnotherOne.LastEventType || EventTypes.PASS ==
AnotherOne.LastEventType)
    {
        if ((TE.Time - AnotherOne.LastEventTime) >= H_AA)
            return true;
        else

```

```

        return false;
    }
    else
    {
        return true;
    }
}
protected bool CheckDD(ref TrainEvent TE, ref Tracks AnotherOne)
{
    if (AnotherOne.LastEventTime < 0)
        return true;
    if (AnotherOne.LastEventDirection != TE.Dir)
        return true;

    if (EventTypes.DEPART == AnotherOne.LastEventType)
    {
        if ((TE.Time - AnotherOne.LastEventTime) >= H_DD)
            return true;
        else
            return false;
    }
    else
    {
        return true;
    }
}

protected bool CheckAP_DP(ref TrainEvent TE, ref Tracks SecTrack)
{
    if (SecTrack.LastEventTime < 0)
        return true;
    if (TE.Dir != SecTrack.LastEventDirection)
        return true;

    Debug.Assert(EventTypes.ARRIVE == SecTrack.LastEventType || EventTypes.DEPART ==
SecTrack.LastEventType);
    if (EventTypes.ARRIVE == SecTrack.LastEventType)
    {
        if ((TE.Time - SecTrack.LastEventTime) >= H_AA)
            return true;
        else
            return false;
    }
    else
    {
        if ((TE.Time - SecTrack.LastEventTime) >= H_DD)
            return true;
        else
            return false;
    }
}

public virtual bool IsAllTrackEmpty()
{
    if (null == m_track1.Occupied && null == m_track2.Occupied)
        return true;
    else
        return false;
}

public virtual void Reset()
{
    m_track1.Reset();
    m_track2.Reset();
}

public virtual bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing)
{
    if (Directions.ASC == TE.Dir)
    {
        if (CheckTrackNoneOccupy(ref m_track1) && CheckDA(ref m_track1, TE.Time))
        {

```

```

        m_track1.SetLastEvent(ref TE);
        return true;
    }
    else
    {
        return false;
    }
}
else
{
    if (CheckTrackNoneOccupy(ref m_track2) && CheckDA(ref m_track2, TE.Time))
    {
        m_track2.SetLastEvent(ref TE);
        return true;
    }
    else
    {
        return false;
    }
}
}

public virtual bool Depature(ref TrainEvent TE)
{
    if (Directions.ASC == TE.Dir)
    {
        m_track1.SetLastEvent(ref TE);
    }
    else
    {
        m_track2.SetLastEvent(ref TE);
    }

    return true;
}
public virtual bool Passing(ref TrainEvent TE)
{
    return Arrival(ref TE, false);
}

public virtual void SetTrackBlock(string track, bool block)
{
    Debug.Assert(track != "Track3");
    Debug.Assert(track != "Track4");
    if (m_track1.Name == track)
    {
        m_track1.Blocked = block;
    }
    else if (m_track2.Name == track)
    {
        m_track2.Blocked = block;
    }
}

}

public class StationI : Station
{
    private Tracks m_track3;
    public Tracks Track3
    {
        get { return m_track3; }
    }

    private Tracks m_track4;
    public Tracks Track4
    {
        get { return m_track4; }
    }

    public StationI()
    {
        m_track3 = new Tracks("Track3");
    }
}

```

```

        m_track4 = new Tracks("Track4");
    }

    public override string GetStationType()
    {
        return "Type I";
    }

    public override bool IsAllTrackEmpty()
    {
        if (null == m_track1.Occupied && null == m_track2.Occupied && null ==
m_track3.Occupied && null == m_track4.Occupied)
            return true;
        else
            return false;
    }

    public override void Reset()
    {
        m_track1.Reset();
        m_track2.Reset();
        m_track3.Reset();
        m_track4.Reset();
    }

    public override bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing)
    {
        if (Directions.ASC == TE.Dir)
            return Arrival(ref TE, IsFollowingTrainPassing, ref m_track1, ref m_track3);
        else
            return Arrival(ref TE, IsFollowingTrainPassing, ref m_track2, ref m_track4);
    }

    private bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing, ref Tracks Pri,
ref Tracks Sec)
    {
        if (CheckTrackNoneOccupy(ref Sec) && CheckDA(ref Sec, TE.Time) && CheckAA(ref TE,
ref Pri))
        {
            Sec.SetLastEvent(ref TE);
            return true;
        }
        else if (CheckTrackNoneOccupy(ref Pri) && CheckDA(ref Pri, TE.Time) && CheckAA(ref
TE, ref Sec))
        {
            Pri.SetLastEvent(ref TE);
            return true;
        }
        else
        {
            return false;
        }
    }

    public override bool Depature(ref TrainEvent TE)
    {
        if (Directions.ASC == TE.Dir)
            return Depature(ref TE, ref m_track1, ref m_track3);
        else
            return Depature(ref TE, ref m_track2, ref m_track4);
    }

    private bool Depature(ref TrainEvent TE, ref Tracks Pri, ref Tracks Sec)
    {
        Debug.Assert((Pri.Occupied == TE.TrainID && Sec.Occupied != TE.TrainID) ||
(Pri.Occupied != TE.TrainID && Sec.Occupied == TE.TrainID));
        if (Pri.Occupied == TE.TrainID)
        {
            if ( CheckDD(ref TE, ref Sec) )
            {
                Pri.SetLastEvent(ref TE);
            }
        }
    }

```

```

        return true;
    }
    else
    {
        return false;
    }
}
else
{
    if (CheckDD(ref TE, ref Pri))
    {
        Sec.SetLastEvent(ref TE);
        return true;
    }
    else
    {
        return false;
    }
}
}

public override bool Passing(ref TrainEvent TE)
{
    if (Directions.ASC == TE.Dir)
        return Passing(ref TE, ref m_track1, ref m_track3);
    else
        return Passing(ref TE, ref m_track2, ref m_track4);
}

private bool Passing(ref TrainEvent TE, ref Tracks Pri, ref Tracks Sec)
{
    if (CheckTrackNoneOccupy(ref Pri) && CheckDA(ref Pri, TE.Time) && CheckAP_DP(ref TE,
ref Sec) )
    {
        Pri.SetLastEvent(ref TE);
        return true;
    }
    else
    {
        return false;
    }
}

public override void SetTrackBlock(string track, bool block)
{
    if (Track3.Name == track)
    {
        Track3.Blocked = block;
    }
    else if (Track4.Name == track)
    {
        Track4.Blocked = block;
    }
    else
    {
        base.SetTrackBlock(track, block);
    }
}

}

public class StationII : Station
{
    private Tracks m_track3;
    public Tracks Track3
    {
        get { return m_track3; }
    }

    public StationII()
    {
        m_track3 = new Tracks("Track3");
    }
}

```



```

    }

    public override string GetStationType()
    {
        return "Type II";
    }

    public override bool IsAllTrackEmpty()
    {
        if (null == m_track1.Occupied && null == m_track2.Occupied && null ==
m_track3.Occupied)
            return true;
        else
            return false;
    }
    public override bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing)
    {
        return true;
    }
    public override bool Depature(ref TrainEvent TE)
    {
        return true;
    }
    public override bool Passing(ref TrainEvent TE)
    {
        return true;
    }
}

public class StationIII_R : Station
{
    private Tracks m_track3;
    public Tracks Track3
    {
        get { return m_track3; }
    }

    private string m_Type;
    public string Type
    {
        get { return m_Type; }
    }

    public StationIII_R(string type)
    {
        m_track3 = new Tracks("Track3");
        m_Type = type;
        CrossOverInfo Rule;
        switch (type)
        {
            case "III_R":
                Rule = new CrossOverInfo("0==>Track3", "Track2==>1", false);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("Track2==>1", "0==>Track3", true);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("Track3==>1", "0==>Track3", true);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("1==>Track2", "Track3==>0", false);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("Track3==>0", "1==>Track2", true);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("Track3==>0", "1==>Track3", true);
                CrossInfoList.Add(Rule);
                break;
            case "III_L":
                Rule = new CrossOverInfo("0==>Track1", "Track3==>1", false);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("Track3==>1", "0==>Track1", true);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("Track3==>1", "0==>Track3", true);
                CrossInfoList.Add(Rule);
                Rule = new CrossOverInfo("1==>Track3", "Track1==>0", false);

```

```

        CrossInfoList.Add(Rule);
        Rule = new CrossOverInfo("Track1==>0", "1==>Track3", true);
        CrossInfoList.Add(Rule);
        Rule = new CrossOverInfo("Track3==>0", "1==>Track3", true);
        CrossInfoList.Add(Rule);
        break;
    case "II":
        Rule = new CrossOverInfo("Track3==>1", "0==>Track3", true);
        CrossInfoList.Add(Rule);
        Rule = new CrossOverInfo("Track3==>0", "1==>Track3", true);
        CrossInfoList.Add(Rule);
        break;
    default:
        Debug.Assert(false);
        break;
    }
}

public override string GetStationType()
{
    return "Type III_R";
}

public override bool IsAllTrackEmpty()
{
    if (null == m_track1.Occupied && null == m_track2.Occupied && null ==
m_track3.Occupied)
        return true;
    else
        return false;
}

public override void Reset()
{
    m_track1.Reset();
    m_track2.Reset();
    m_track3.Reset();

    foreach (CrossOverInfo rule in CrossInfoList)
    {
        rule.Reset();
    }
}

private List<CrossOverInfo> CrossInfoList = new List<CrossOverInfo>();

private void UpdateCrossoverInfo(ref TrainEvent TE, string TrackName)
{
    string RouteName = null;

    switch (TE.Type)
    {
        case EventTypes.ARRIVE:
            RouteName = ((int)TE.Type).ToString() + RailSystem.AndSymbol + TrackName;
            UpdateCrossoverInfo(TE.Time, RouteName);
            break;
        case EventTypes.DEPART:
            RouteName = TrackName + RailSystem.AndSymbol + ((int)TE.Type).ToString();
            UpdateCrossoverInfo(TE.Time, RouteName);
            break;
        case EventTypes.PASS:
            RouteName = ((int)TE.Type).ToString() + RailSystem.AndSymbol + TrackName;
            UpdateCrossoverInfo(TE.Time, RouteName);
            RouteName = TrackName + RailSystem.AndSymbol + ((int)TE.Type).ToString();
            UpdateCrossoverInfo(TE.Time, RouteName);
            break;
        default:
            Debug.Assert(false);
            break;
    }
}

private void UpdateCrossoverInfo(int Time, string RouteName)

```

```

{
    foreach( CrossOverInfo rule in CrossInfoList )
        rule.UpdateFirstRoute(Time, RouteName);
}

private bool CheckCrossoverHeadway(ref TrainEvent TE, string TrackName)
{
    string RouteName=null;

    switch (TE.Type)
    {
        case EventTypes.ARRIVE:
            RouteName= ((int)TE.Type).ToString()+RailSystem.AndSymbol+TrackName;
            return CheckCrossoverHeadway(TE.Time, RouteName);
            //break;
        case EventTypes.DEPART:
            RouteName= TrackName+RailSystem.AndSymbol+((int)TE.Type).ToString();
            return CheckCrossoverHeadway(TE.Time, RouteName);
            //break;
        case EventTypes.PASS:
            RouteName= ((int)TE.Type).ToString()+RailSystem.AndSymbol+TrackName;
            bool b1 = CheckCrossoverHeadway(TE.Time, RouteName);
            RouteName= TrackName+RailSystem.AndSymbol+((int)TE.Type).ToString();
            bool b2 = CheckCrossoverHeadway(TE.Time, RouteName);
            return ( b1 && b2);
            //break;
        default:
            Debug.Assert(false);
            return true;
    }
}

private bool CheckCrossoverHeadway(int Time, string RouteName)
{
    foreach (CrossOverInfo rule in CrossInfoList)
    {
        if (false == rule.CheckHeadway(Time, RouteName, H_DA_R, H_AD_R) )
            return false;
    }
    return true;
}

public override bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing)
{
    if (Directions.ASC == TE.Dir)
        return Arrival(ref TE, IsFollowingTrainPassing, ref m_track1, ref m_track3);
    else
        return Arrival(ref TE, IsFollowingTrainPassing, ref m_track2, ref m_track3);
}

private bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing, ref Tracks Pri,
ref Tracks Sec)
{
    if (IsFollowingTrainPassing)
    {
        if (CheckTrackNoneOccupy(ref Sec) && CheckDA(ref Sec, TE.Time) && CheckAA(ref
TE, ref Pri) && CheckCrossoverHeadway(ref TE, Sec.Name))
        {
            Sec.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Sec.Name);
            return true;
        }
        }else if ( CheckTrackNoneOccupy(ref Pri) && CheckDA(ref Pri, TE.Time) &&
CheckAA(ref TE, ref Sec) && CheckCrossoverHeadway(ref TE, Pri.Name) )
        {
            Pri.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Pri.Name);
            return true;
        }
        }else
        {
            return false;
        }
    }
}

```

```

        if (CheckTrackNoneOccupy(ref Pri) && CheckDA(ref Pri, TE.Time) && CheckAA(ref
TE, ref Sec) && CheckCrossoverHeadway(ref TE, Pri.Name))
        {
            Pri.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Pri.Name);
            return true;
        }
        else if (CheckTrackNoneOccupy(ref Sec) && CheckDA(ref Sec, TE.Time) &&
CheckAA(ref TE, ref Pri) && CheckCrossoverHeadway(ref TE, Sec.Name))
        {
            Sec.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Sec.Name);
            return true;
        }
        else
        {
            return false;
        }
    }
}

public override bool Depature(ref TrainEvent TE)
{
    if (Directions.ASC == TE.Dir)
        return Depature(ref TE, ref m_track1, ref m_track3);
    else
        return Depature(ref TE, ref m_track2, ref m_track3);
}

private bool Depature(ref TrainEvent TE, ref Tracks Pri, ref Tracks Sec)
{
    Debug.Assert((Pri.Occupied == TE.TrainID && Sec.Occupied != TE.TrainID) ||
(Pri.Occupied != TE.TrainID && Sec.Occupied == TE.TrainID));
    if (Pri.Occupied == TE.TrainID)
    {
        if (CheckDD(ref TE, ref Sec) && CheckCrossoverHeadway(ref TE, Pri.Name) )
        {
            Pri.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Pri.Name);
            return true;
        }
        else
        {
            return false;
        }
    }
    else
    {
        if (CheckDD(ref TE, ref Pri) && CheckCrossoverHeadway(ref TE, Sec.Name))
        {
            Sec.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Sec.Name);
            return true;
        }
        else
        {
            return false;
        }
    }
}

public override bool Passing(ref TrainEvent TE)
{
    if (Directions.ASC == TE.Dir)
        return Passing(ref TE, ref m_track1, ref m_track3);
    else
        return Passing(ref TE, ref m_track2, ref m_track3);
}

private bool Passing(ref TrainEvent TE, ref Tracks Pri, ref Tracks Sec)
{
    if (CheckTrackNoneOccupy(ref Pri) && CheckDA(ref Pri, TE.Time) && CheckAP_DP(ref TE,
ref Sec) && CheckCrossoverHeadway(ref TE, Pri.Name))

```

```

        {
            Pri.SetLastEvent(ref TE);
            UpdateCrossoverInfo(ref TE, Pri.Name);
            return true;
        }
        else
        {
            return false;
        }
    }

    public override void SetTrackBlock(string track, bool block)
    {
        Debug.Assert(track != "Track4");
        if (Track3.Name == track)
        {
            Track3.Blocked = block;
        }
        else
        {
            base.SetTrackBlock(track, block);
        }
    }
}

public class StationIII_L : Station
{
    private Tracks m_track3;
    public Tracks Track3
    {
        get { return m_track3; }
    }

    public StationIII_L()
    {
        m_track3 = new Tracks("Track3");
    }

    public override string GetStationType()
    {
        return "Type III_L";
    }

    public override bool IsAllTrackEmpty()
    {
        if (null == m_track1.Occupied && null == m_track2.Occupied && null ==
m_track3.Occupied)
            return true;
        else
            return false;
    }

    public override bool Arrival(ref TrainEvent TE, bool IsFollowingTrainPassing)
    {
        return true;
    }

    public override bool Depature(ref TrainEvent TE)
    {
        return true;
    }

    public override bool Passing(ref TrainEvent TE)
    {
        return true;
    }
}

public class CrossOverInfo
{
    private int m_LastFirstRouteTime=-RailSystem.DayTime;
    private string m_FirstRoute;
    private string m_SecondRoute;
    bool m_Is_H_DA_R;

```

```

public CrossoverInfo(string FirstRoute, string SecondRoute, bool Is_H_DA_R)
{
    m_FirstRoute = FirstRoute;
    m_SecondRoute = SecondRoute;
    m_Is_H_DA_R = Is_H_DA_R;
}
public void UpdateFirstRoute(int NowTime, string RouteName)
{
    if (m_FirstRoute == RouteName)
        m_LastFirstRouteTime = NowTime;
}

public void Reset()
{
    m_LastFirstRouteTime = -RailSystem.DayTime;
}
public bool CheckHeadway(int NowTime, string RouteName, int H_DA_R, int H_AD_R)
{
    if (m_SecondRoute == RouteName )
    {
        if (m_Is_H_DA_R)
        {
            if ((NowTime - m_LastFirstRouteTime) >= H_DA_R)
                return true;
            else
                return false;
        }
        else
        {
            if ((NowTime - m_LastFirstRouteTime) >= H_AD_R)
                return true;
            else
                return false;
        }
    }
    else
    {
        return true;
    }
}
}
}
}

```

貳、資料輸入及模擬結摘要說明

一、模式輸入

本程式須從輸入檔讀入資料後方能進行模擬，而輸入檔內容必須符合格式，才能使程式正確讀入進行分析，以獲得正確結果，本章將介紹輸入檔格式以及程式輸入介面的操作方式。

(一)輸入檔格式

程式的輸入檔內容如附圖 1 所示。

INPUT.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

#####全域變數

*同向到達-到達時隔(秒)
180

*同向離開-到達時隔(秒)
190

*同向離開-離開時隔(秒)
200

*反向到達-離開時隔(秒)
210

*反向離開-到達時隔(秒)
220

*站間容量
2

*起點時運轉時間:原有時間比例(0.0<x<1.0)
0.900000

#####路線車站

*車站數目
9

*名稱	車站里程(m)	車站型式	停靠列車最短停站時間
南港,	18700,	1,	60
松山,	22200,	1,	60
台北,	28600,	1,	90
萬華,	31200,	2,	30
板橋,	36400,	1,	60
樹林,	41000,	1,	60
山佳,	44800,	5,	30
鶯歌,	49300,	1,	30
桃園,	57500,	3,	60

#####車次資料

*列車數目
6

*車種	車次	停站計畫
1,	車次A,	松山, 松山, 台北, 台北, 萬華, 萬華, 板橋, 板
		38400, 38520, 38940, 39180, 39360, 39360, 39660, 397
1,	車次B,	松山, 松山, 台北, 台北, 萬華, 萬華, 板橋, 板
		39600, 39720, 40140, 40380, 40560, 40560, 40860, 409
2,	車次C,	台北, 台北, 萬華, 萬華, 板橋, 板橋, 樹林, 樹
		41340, 41580, 41760, 41760, 42060, 42180, 42660, 427

全域參數

路線車站

車次資料

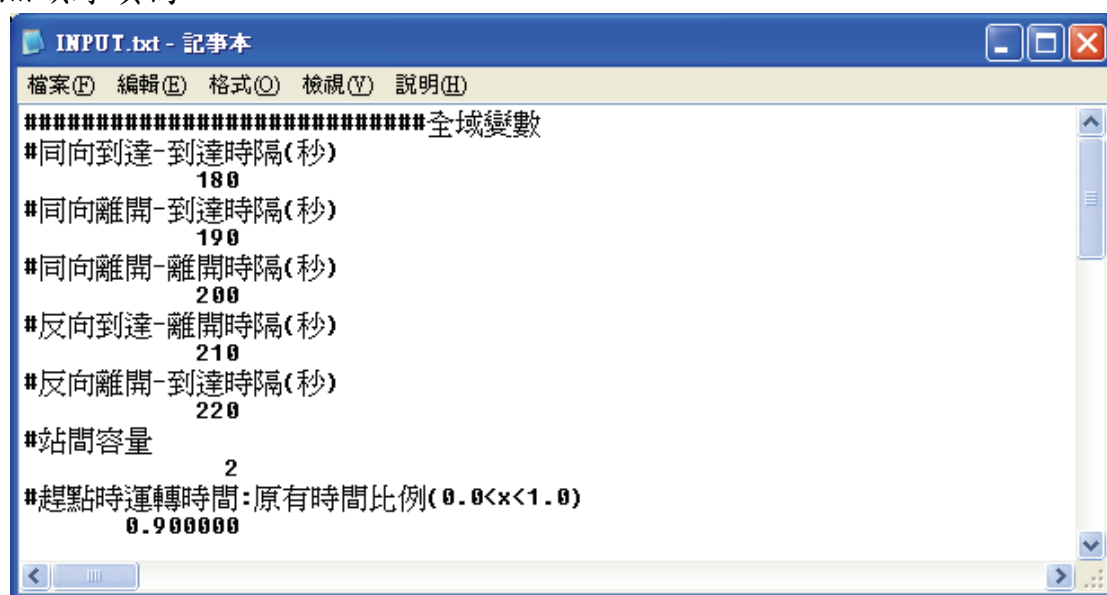
附圖 1 程式輸入檔

其中以「#」符號開始的為註解，程式在讀取輸入檔時，會自動略過該行之資料，此功能可供使用者加入額外資訊以供檢視。例如圖

中第二行「#同向到達-到達時隔(秒)」，便是用來說明第三行的「180」的意義。一份完整的輸入檔共可分為三個主要片段，分別為全域參數、路線車站和車次等資料，將依序介紹於後續章節。

(二)全域變數資料

全域變數的格式如附圖 2，所包含的各項參數與其相關說明限制彙整於附表 1，各項參數必須以空白、換行或「,」符號區隔，且須按照順序填寫。



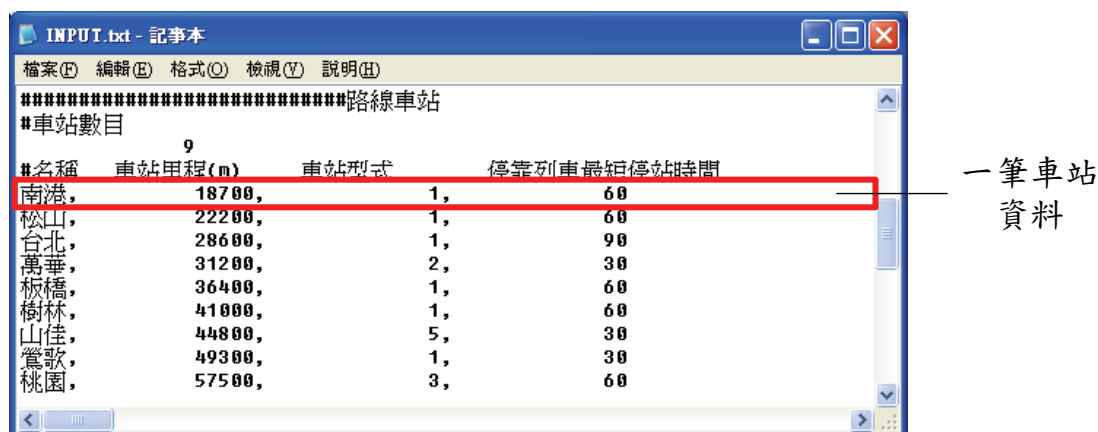
附圖 2 全域變數輸入格式

附表 1 全域變數資料參數型態、說明與限制彙整表

參數	型態	說明	限制
同向到達-到達時隔	整數	以秒為單位，連續兩列同向列車進站所需保持的時隔	大於或等於 0
同向離開-到達時隔	整數	以秒為單位，先行列車離站後，到續行列車進站之間所需保持的時隔	大於或等於 0
同向離開-離開時隔	整數	以秒為單位，連續兩列同向列車離站所需保持的時隔	大於或等於 0
反向到達-離開時隔	整數	以秒為單位，平面交叉時隔	大於或等於 0
反向離開-到達時隔	整數	以秒為單位，平面交叉時隔	大於或等於 0
站間容量	整數	站間軌道可容納的列車數	大於或等於 0
趕點比例	實數	列車於站間進行趕點時，其最小運轉時間與基準運轉時間的比率	介於 0~1 之間

(三)路線車站資料

路線車站的資料片段格式如附圖 3 所示，首先為欲分析路線的車站總數，接著為各車站的資料，每一筆車站資料包含車站名稱、里程、車站型式，和停靠列車最短停站時間等參數，其相關說明限制彙整於附表 2，各項參數必須以空白、換行或「,」符號區隔，且須按照順序填寫。此外需特別注意的是，各車站資料需以里程遞增的次序填寫。



附圖 3 路線車站輸入格式

附表 2 路線車站資料參數型態、說明與限制彙整表

參數	型態	說明	限制
車站名稱	文字	車站的名稱	不得含空白、「,」等特殊字元
車站里程	整數	以公尺為單位，車站的里程	大於 0
車站型式	列舉	車站內軌道佈設型式	註
停靠列車最短 停站時間	整數	以秒為單位，若列車停靠該 車站，最少的停站時間	大於或等於 0

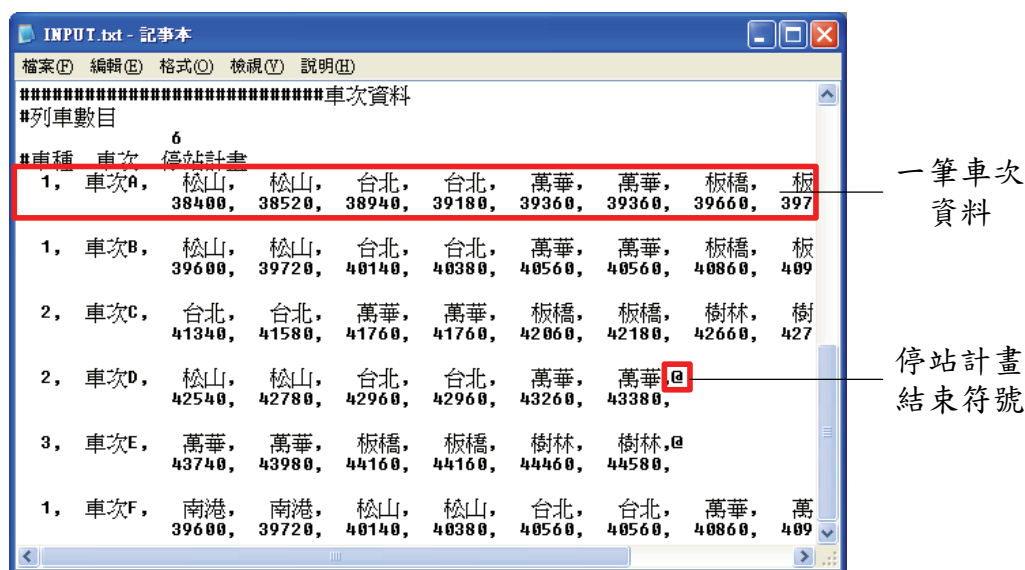
註：本程式考量 5 種佈設型式，以數字 1~5 表示。

(四)車次資料

在車次資料的片段中，資料格式如附圖 4，首先為列車數目，表示所有車次的總數，接著為各車次資料，每一筆車次資料佔兩行，每筆車次資料之間必須空一行，每一筆車次資料皆包含車種、車次名稱和停站計畫，其相關說明限制彙整於附表 3。

每筆車次資料的各項參數必須以空白、換行或「,」符號區隔，並且須按照順序填寫，首先為車種，其次為車次名稱，最後為停站計

畫，其中停站計畫包含該車所經過車站的進站時間與離站時間，如附圖 4 所示，於車次名稱之後開始填寫該車所經過的車站名稱，而第二行填寫於該車站的進站與離站時間。在此需特別注意，對於每一車站皆要有進站與離站時間，若該列車為通過該站，則於該站的進站與離站時間為相等，此外在最後一個離站的車站名稱之後需加上「@」符號，以表示結束。



附圖 4 車次資料輸入格式

附表 3 車次資料參數型態、說明與限制彙整表

參數	型態	說明	限制
車種	整數	由使用者自行定義不同的數字代表不同的車種	大於 0
車次名稱	文字	車次的名稱	不得含空白、「,」等特殊字元
停站計畫	站	文字	列車進/離站所經過的車站名稱
	時	整數	列車在各車站進/離站的時間，從 0 點開始計算，以秒為單位，例如早上 6 點為 21600 秒
			名稱以及順序必須和「路線車站」所定義的資料一致
			介於 0~86399 之間

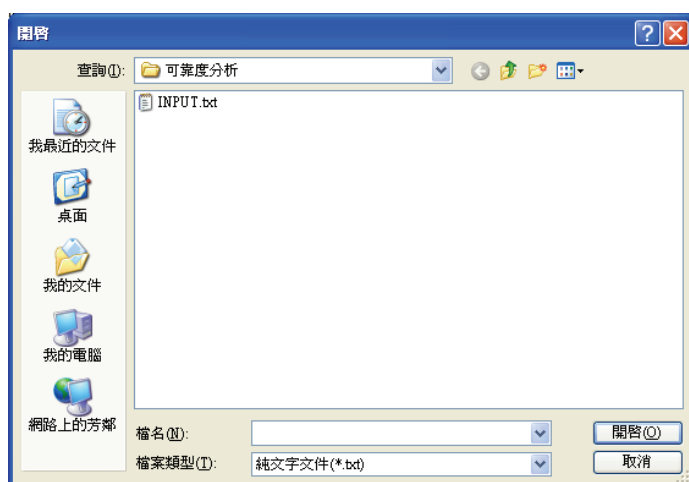
(五)程式輸入介面

程式介面如附圖 5，選擇左上角【開啟輸入檔】按鈕，會出現附

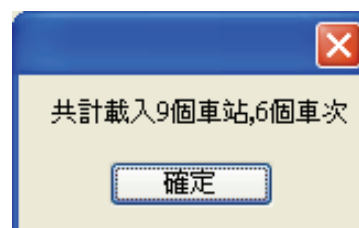
圖 6 之對話盒來協助使用者選擇欲讀取的輸入檔，選定輸入檔後按下【開啟】按鈕，程式便會讀取輸入檔，當讀取成功後會出現附圖 7 之訊息，顯示該輸入檔所包含的車站數與車次數，若該數量與輸入檔的設定有所出入，請重新檢視輸入檔設定是否有誤。



附圖 5 程式介面



附圖 6 開啟輸入檔對話盒



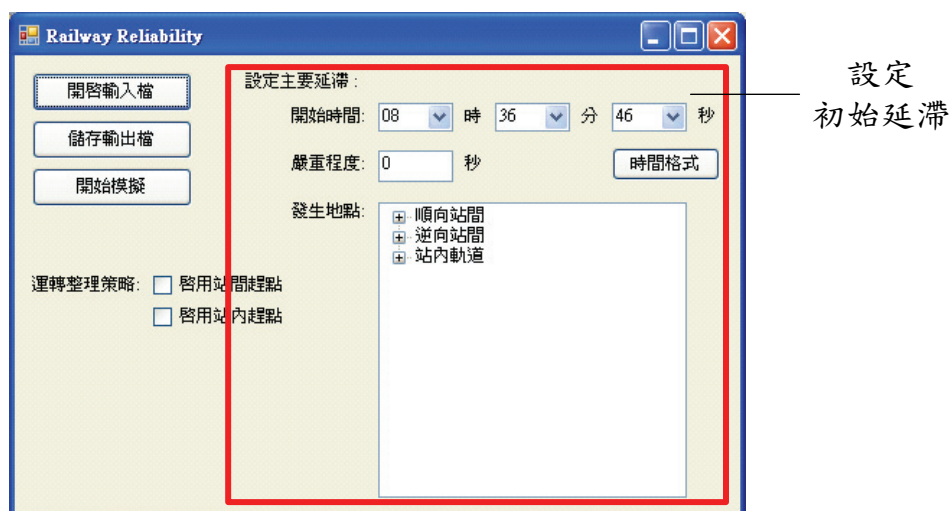
附圖 7 輸入檔讀取成功之訊息

二、模擬執行

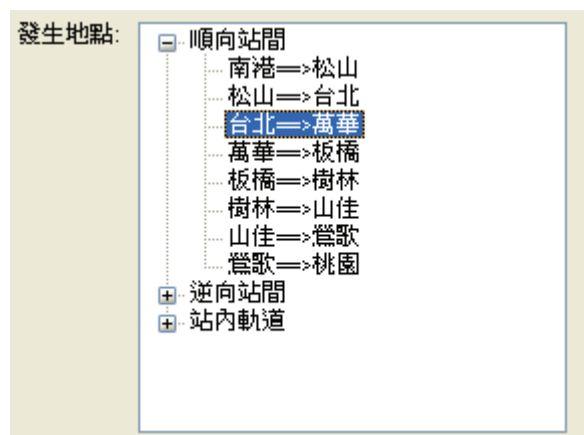
本程式係針對一可行班表，模擬初始延滯(亦稱主要延滯)發生對整體班表的影響，因此模擬的執行過程須先設定初始延滯的時間地點，此外還可選擇不同運轉整理策略，以比較不同策略對可靠度的改善效果。

在程式的介面中，設定初始延滯的區域在介面的右方，如附圖 8，

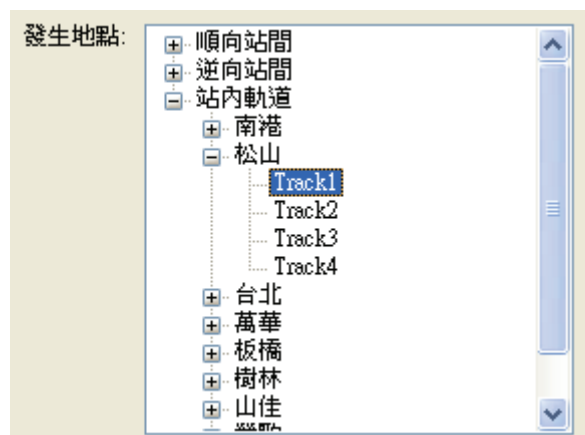
在該區中可設定初始延滯的開始時間、嚴重程度和發生地點。其中開始時間表示初始延滯發生的時間，嚴重程度則是表示初始延滯的持續時間，發生地點則可選擇初始延滯是發生在站間軌道還是站內軌道，分別如附圖 9 和附圖 10。此外，本程式另外提供時間格式轉換小工具，其介面如附圖 11，協助使用者轉換不同的時間格式，免去換算的麻煩。



附圖 8 設定初始延滯



附圖 9 選擇初始延滯發生在站間軌道

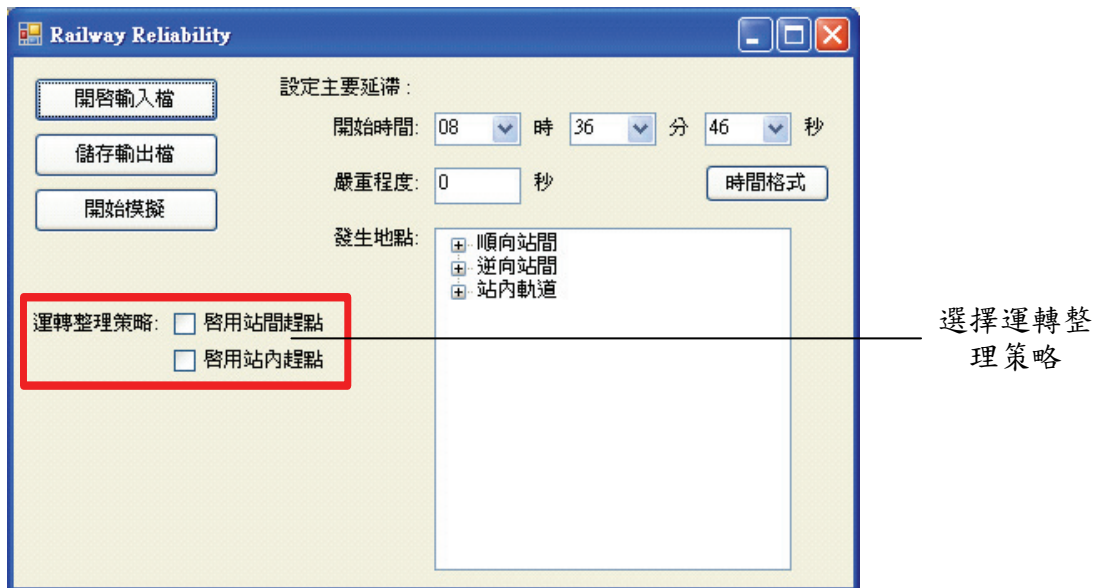


附圖 10 選擇初始延滯發生在站內軌道



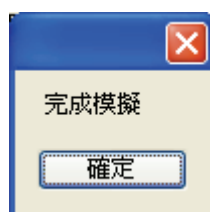
附圖 11 時間格式轉換小工具

在運轉整理策略方面，本程式提供兩種策略供使用者選用，分別是(1)站間趕點和(2)站內趕點，於介面上直接勾選便可啟用，如附圖 12，兩種策略互相獨立，可同時選用。



附圖 12 選擇運轉整理策略

在完成初始延滯的設定與運轉整理策略的選擇後，於介面上按下【開始模擬】按鈕，程式即開始進行模擬運算，當附圖 13 之訊息出現後，表示模擬運算完成。



附圖 13 完成模擬之訊息

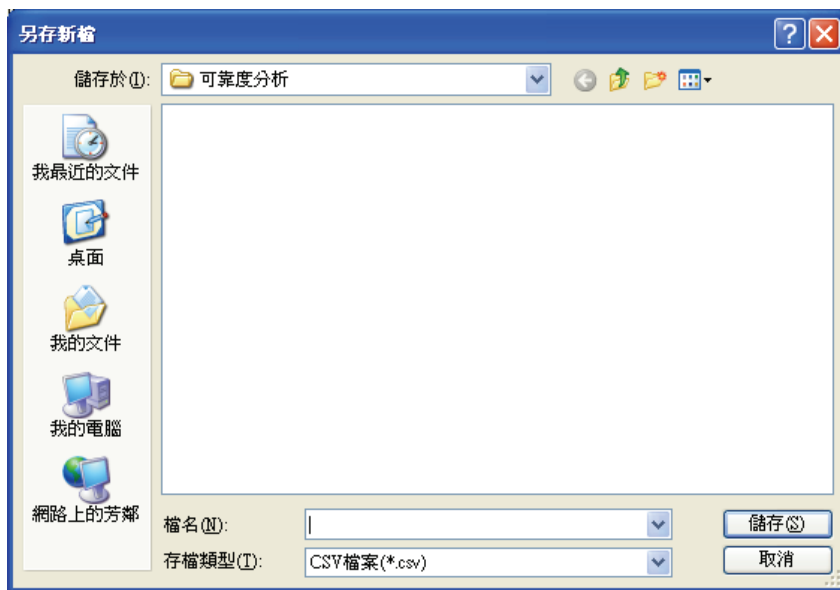
三、模式輸出

本程式可將模擬結果輸出成文字檔，但是僅從文字檔較難看出初始延滯對列車運行的影響，因此本研究另外利用 MS EXCEL 製作了 EXCEL 工具，供模擬結果後處理使用。

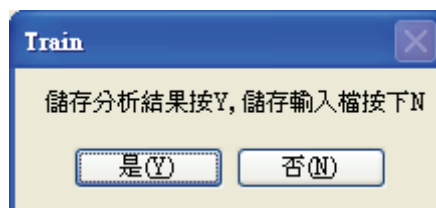
(一)程式輸出介面

在程式介面（如附圖5）選擇【儲存輸出檔】按鈕，會出現附圖 14 之對話盒來協助使用者進程式輸出工作，透過對話盒選定路徑與指定檔名後，按下【儲存】按鈕後，會再出現附圖 15 之對話盒，若要儲存模擬分析的結果就按【是】；若要儲存原輸入檔的內容則按【否】，如此便能完成程式輸出工作。

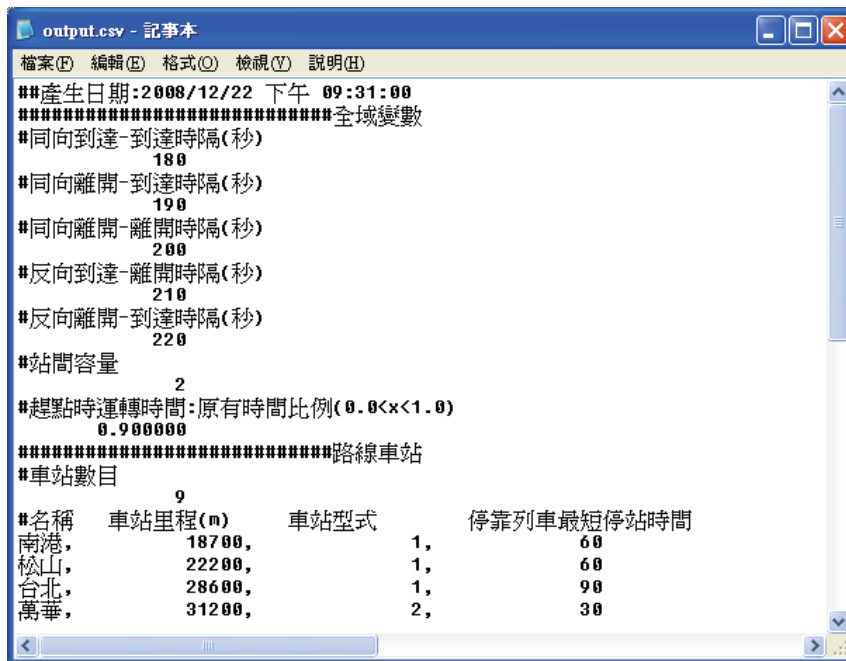
程式的輸出檔類型為「CSV」檔案，以 Notepad 開啟的畫面如附圖 16，此外亦可由 MS EXCEL 直接開啟。其格式內容和輸入檔相同，唯獨在車次資料的部分是列出在完成模擬後，各列車於各車站的到離時間，而非原本輸入檔中的時間。



附圖 14 儲存輸出檔對話盒



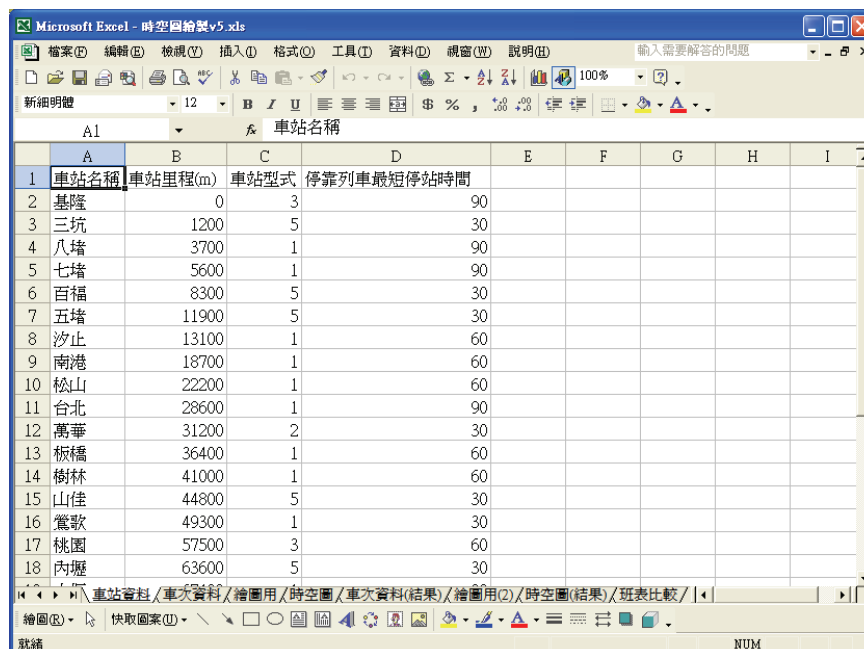
附圖 15 選擇輸出項目對話盒



附圖 16 輸出檔格式

(二)EXCEL 工具

本研究所製作的 EXCEL 工具如附圖 17，一共包含 8 個工作表、2 個表單和 6 個巨集，各工作表的內容說明如附表 4，本工具提供時空圖繪製和班表比較兩項功能，其操作方式分別說明於下。



附圖 17 EXCEL 工具介面

附表 4 EXCEL 工具中各工作表之內容說明

工作表名稱	內容說明
車站資料	輸入檔中各車站的資料
車次資料	輸入檔中各車次的資料
繪圖用	供繪製時空圖內部使用，勿隨意更改
時空圖	輸入檔中各車次的運行時空圖
車次資料（結果）	輸出檔中各車次的資料
繪圖用(2)	供繪製時空圖內部使用，勿隨意更改
時空圖（結果）	輸出檔中各車次的運行時空圖
班表比較	根據所選車次比較發生延滯前後的時空圖

(三)輸入資料

在進行時空圖繪製和班表比較之前，需先將程式的輸入與輸出檔的資料輸入到 EXCEL 工具中，首先在「車站資料」工作表中輸入各車站資料，如附圖 18，第一筆車站資料必須從儲存格 A2~D2 開始，依序填寫車站名稱、里程、軌道型式和最短停站時間，每筆車站資料佔用一行，且各車站必須按里程遞增次序填寫。


	A	B	C	D	
1	車站名稱	車站里程(m)	車站型式	停靠列車最短停站時間	
2	基隆	0	3	90	一筆車站資料
3	三坑	1200	5	30	
4	八堵	3700	1	90	
5	七堵	5600	1	90	
6	百福	8300	5	30	
7	五堵	11900	5	30	
8	汐止	13100	1	60	
9	南港	18700	1	60	
10	松山	22200	1	60	
11	台北	28600	1	90	
12	萬華	31200	2	30	
13	板橋	36400	1	60	
14	樹林	41000	1	60	
15	山佳	44800	5	30	
16	鶯歌	49300	1	30	
17	桃園	57500	3	60	
18	內壢	63600	5	30	



附圖 18 輸入車站資料

接著於「車次資料」工作表中填入輸入檔的車次資料，如附圖 19，第一筆車次資料必須從第 2 行開始，每筆車次資料佔用兩行，各車次資料間需空一行。每一車次資料於第一行 A 欄填寫車種，B 欄填寫車次名稱，C 欄以後填寫該車次各到離站的車站名稱，而第二行 C

欄以後填寫於各站到離站的時間。


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	車種	車次	停站計畫										
2	1	車次A	松山	松山	台北	台北	萬華	萬華	板橋	板橋	樹林	樹林	@
3			38400	38520	38940	39180	39360	39360	39660	39780	40260	40380	
4													
5	1	車次B	松山	松山	台北	台北	萬華	萬華	板橋	板橋	樹林	樹林	@
6			39600	39720	40140	40380	40560	40560	40860	40980	41460	41580	
7													
8	2	車次C	台北	台北	萬華	萬華	板橋	板橋	樹林	樹林	@		
9			41340	41580	41760	41760	42060	42180	42660	42780			
10													
11	2	車次D	松山	松山	台北	台北	萬華	萬華	@				
12			42540	42780	42960	42960	43260	43380					
13													
14	3	車次E	萬華	萬華	板橋	板橋	樹林	樹林	@				
15			43740	43980	44160	44160	44460	44580					
16													
17	1	車次F	南港	南港	松山	松山	台北	台北	萬華	萬華	板橋	板橋	@
18			39600	39720	40140	40380	40560	40560	40860	40980	41460	41580	

一筆車次
資料

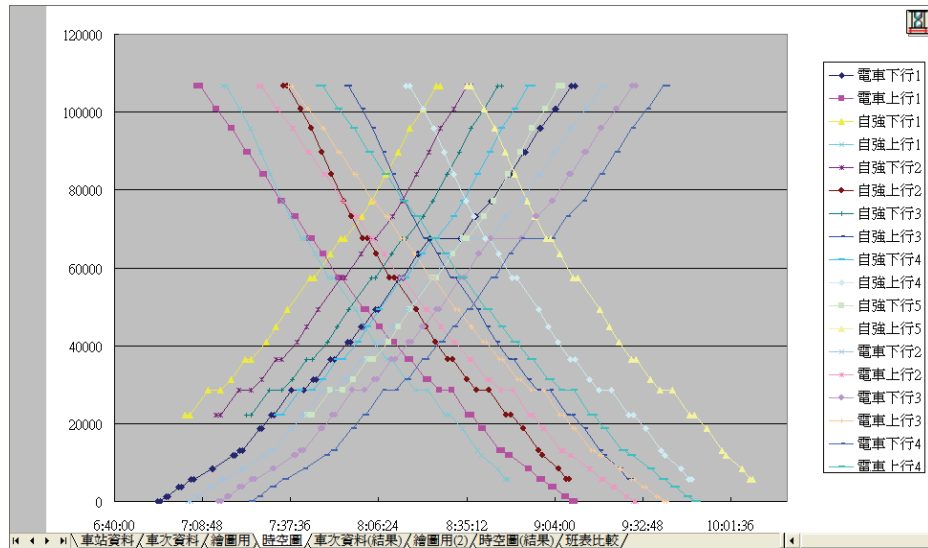
附圖 19 輸入車次資料

最後在「車次資料（結果）」工作表中填寫輸出檔的車次資料，該表的格式和「車次資料」工作表相同，填寫的原則亦相同，在此不再贅述。

(四)時空圖繪製

當資料輸入完成後，於「車次資料」和「車次資料（結果）」工作表中，按下【】圖示，此時便呼叫巨集繪製時空圖，其結果分別呈現於「時空圖」和「時空圖（結果）」工作表中，如附圖 20。

欲調整時空圖檢視的時間範圍，可在「時空圖」或「時空圖（結果）」工作表右上方，按下【】圖示，會出現附圖 21 對話盒，設定起始時間和結束時間後按下【確定】按鈕，兩工作表中的時空圖時間範圍便會根據設定同時進行調整。




附圖 20 時空圖繪製結果

附圖 21 設定時間軸範圍

(五)班表比較

當時空圖繪製完成後，於 EXCEL 工具中切換「時空圖」或「時空圖（結果）」兩工作表，可達到班表比較的效果，但是當車次數量龐大時，使用者將難以判斷其差異，因此在本工具中另外提供針對某單一車次的班表比較功能。

按下「班表比較」工作表中右上方的【】圖示，會出現附圖 22 之對話盒，從下拉式選單中選擇欲比較的車次後按下【確定】按鈕，畫面中將該車次原始與模擬後的班表同時繪在一起，如附圖 23，使用者便可一目了然該車次的延滯情形。

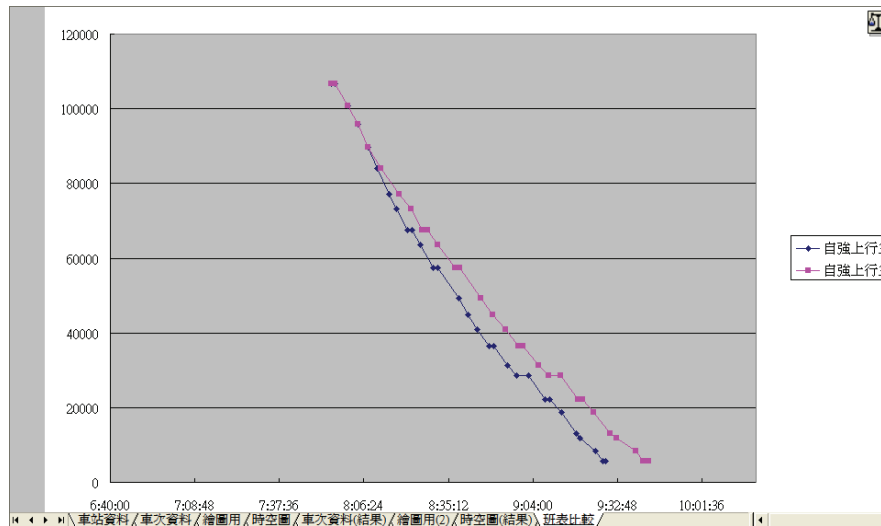
分析比較

選擇比較的車次

電車下行1

確定

附圖 22 選擇比較的車次



附圖 23 班表比較結果