

淡江大學運輸管理學系運輸科學碩士班

碩士論文

指導教授：陶冶中 博士

不同行車制度下台鐵區段容量分析模擬模式之
研究

A Simulation Model for Line Section Capacity Analysis
of Taiwan Railway Administration with various train
operation systems

研究生：劉嵩瀚 撰

中 華 民 國 95 年 6 月

論文題目：不同行車制度下台鐵區段容量分析模擬模式之研究

頁數：144

校所組別：淡江大學運輸管理學系運輸科學碩士班

畢業時間及提要別：94 學年度第 2 學期碩士學位論文提要

畢業生：劉嵩瀚

指導教授：陶冶中 博士

論文提要內容：

國內近年來重大交通建設舉凡捷運、高鐵乃至於台鐵轉型，由於其能夠大量且快速達到運輸之目的，因此漸受重視。在各國如火如荼地運用高科技於軌道運輸以提升安全、效率的趨勢下，我國對於軌道運輸系統智慧化之規劃與建置計畫，亦應及時進行。目前台鐵採用固定自動閉塞系統來確保列車行車安全，然而當因應特殊或緊急需求而加開列車時，由於大部分班表固定且行車制度上又有安全間距之考量，縮短發車間隔以提升容量之策略即難以奏效。因此提升鐵路運輸容量的根本方法即為改變行車制度，亦即在保證安全的前提下，藉由不同行車控制方式，進行多種列車調度以滿足乘客最大的需求。

本研究係以台鐵西正線下行松山-新竹段作為研究對象，建構符合台鐵特性多車種混跑、允許超車、越行之列車運行模式，探討採行「移動自動閉塞行車制」對於軌道容量提升之變化。

軌道容量方法論大致可分為解析模式、最佳化模式、模擬模式等三種。解析模式與最佳化模式由於隱含達到系統之穩定狀態，因此較適於相關班表排點問題或時刻表容量問題之求解，且其隱含穩定之系統狀態與現實情況不符，致使求得之軌到容量相對較小且與實際狀況較不符合。模擬模式精確度最高，與系統相依程度高，能夠高度反應真實情況且有利於各種情境之分析模擬。本研究係採用模擬模式並擷取解析模式中若干數學方程式作為模式構建之依據。

模擬結果須與理論值(即時刻表)或實際列車運行實際值經由統計檢定進行驗證之後，始能宣稱模擬模式可代表真實世界；然後進行敏感度分析，以確認各影響變數對於軌道容量之影響程度，最後進行情境模擬，在「固定自動閉塞」與「移動自動閉塞」兩種行車制下，檢視不同車種組成、不同發車間隔下所產生的軌道容量變化值並進行比較。

研究結果發現：在軌道容量計算上，應選定不受列車任務影響之最瓶頸觀察基準點，並以加權平均方式計算列車運轉最小時隔以考慮列車車種組成。在現行列車任務及車種組成下，採行「固定自動閉塞」行車制與「移動自動閉塞」行車制之觀察樣本服務時間可由 14000 秒減至 10000 秒，軌道容量則可由 122(TU/day)增為 273(TU/day)，因此「移動自動閉塞」行車制的成效可見一斑。

關鍵詞：軌道容量分析、固定自動閉塞、移動自動閉塞、模擬模式

Title of Thesis:

Total Pages: 144

A Simulation model for line section capacity analysis of Taiwan Railway Administration with various train operation systems

Keywords: Capacity, Fixed Auto-blocked System(FAS), Moving Auto-blocked System(MAS), Simulation Model

Name of Institute: Graduate Institute of Transportation Science, Tamkang University

Grade Date: June, 2006

Degree Conferred: Master Degree

Name of Student: Song-Han Liu
劉 嵩 瀚

Advisor: Dr. Chi-Chung Tao
陶 治 中 博士

Abstract:

The Applications of high technologies in the field of railways to improve safety and efficiency have become a global trend recently, such as Moving Auto-blocked System (MAS) has been promoted in USA, Europe and Japan. Taiwan Railway Administration (TRA) is often not able to cope with changeable travel demands to increase capacity because of using Fixed Auto-blocked System (FAS). It is, therefore, the main purpose of this study to quantify the possible capacity by comparing MAS and FAS.

The “Song Sang-Hsinchu” line section of TRA is chosen as the empirical case study to verify and validate the proposed simulation model considering the characteristics of running and overtaking behaviors with multiple train classes. First, speed-distance diagrams for each train class are identified by using mathematical formals derived from analytical models. Then the logical flows of train movements from starting station to end station for time-space diagrams with two different train operation systems are transformed into a simulation program by using C++. Verifications and validations are proceeded stepwise to ensure the simulation model to be almost realistic with statistical proofs.

It is concluded that a solid capacity analysis is feasible only when observed points without train tasks disturbances on a bottleneck line section are fixed. The weight averages method is also recommended to calculate train operation headways considering various train classes. In the case of existing train tasks and train classes, it is shown significantly that the capacity with FAS is 122 (TU/day), while the capacity with MAS is 273(TU/day). The existing capacity almost increases 124%.

誌 謝

兩年，可以改變一個人多少？從當初懵懂的加入這個大家族，到現在已經要邁向人生的下一個階段，在經過淡江運科所的木人巷磨練之後，我有自信可以抬頭挺胸地面對未來的任何挑戰。

論文撰寫的期間，首要感謝指導教授 陶冶中博士不論是在論文寫作以及日常做人做事態度的身教，讓我獲益良多，從大一上您的第一堂運輸學，我跟同學就決定未來如果有幸能唸研究所，一定要跟隨您的腳步。啟銘，我做到了。陶老師一直是我的偶像，也將會是我未來在社會上的典範。再次感謝陶老師不厭其煩地對駑鈍的學生不吝指導。所外口試期間，承蒙中華大學 林祥生老師及中興顧問 鍾志成博士撥冗前來惠賜卓見和批評指正，讓學生的論文更臻完備；以及在論文寫作過程中，台北運務段段長 朱來順老師及台鐵綜合調度所 張鎮濟副所長不吝提供資源，鼎力協助學生資料蒐集，特此感謝。

感謝父母對我的包容，雖然離家僅一小時車程，但最後三個月內回家不到五次，你們仍對我支持及鼓勵，研究所是我第一次離家住，真的離開了家才發現這些從前習慣的愛更要好好珍惜，也恭喜婷婷考上研究所，未來的日子我們要一起讓爸媽為我們驕傲。

感謝一起共患難的研究所同學，沒有你們的鼓勵與支持就沒有今天的我，我愛你們大家，未來的日子希望還能有這群好朋友作伴。小臉蓉，謝謝妳在我最低潮的時候聽我說話，即便是論文都快要做不完了，妳還是願意花時間給我建議，妳是一個很棒的女孩子，祝你未來跟妳男朋友一直很幸福地走下去，妳這個朋友我是永遠都不會忘記的；飛影文龍，很高興我們終於變成了同門兄弟，你的能力自然不在話下，在我論文寫作的期間，你一直都是對我幫助最多的，陪我一起打混一起玩一起努力一起煩心，未來如果能有一番成就的話，一定要把你挖角過來；文賢爸爸，你真的是我在淡水的父親，為我的論文以及感情操操心，不時的用你親身經驗開導我，有新知識新資訊也都不吝與我們分享，大嫂也是我們班的一份子，每次都忙進忙出的張羅我們貪吃的嘴，希望如同文賢爸爸來唸研究所的精神一樣，祝你們全家未來都沒有做不到的事情；隊長 pona，許多個煩心的夜晚能夠與隊長您奮勇殺敵，讓我調劑壓力，是我最開心的時候；金門，跟你同班了六年，你的成長我可是比任何人都清楚，你真的很棒，未來也要好好的待佳佳，我們一定會去喝你喜酒的；誌嘉，你做事的認真是值得大家學習的，未來的半年要繼續發光發熱，雖然你家小臉蓉，你知道的，未來你一定可以找到最適合你的，不要一直往牛角尖裡鑽，更何況陳院長已經說要幫你介紹對象，我們大家可都是證人呢；俊昇，雖然平時神龍見首不見尾，可是我們班的活動你可是一次都不缺席過，希望你可以半年後進入社會，也祝你跟你女朋友開心順利；阿吉，你也是個真性情的人，雖然偷偷瞞著我們跟巧克力在一起，可是我們還是會祝福你們

的，記得未來要生個胖娃娃喔；淑芳，跟你的淵源最深，你的個性是你的優點也是你的缺點，也是我跟你相處愉快的原因，你的能力絕對不容懷疑，未來在職場上祝妳一帆風順，也祝你能夠有個好姻緣；秋如，謝謝妳陪我說話也肯聽我的建議，跟學長一定要幸福堅定；羅委員，雖然每次都喜歡鬧你，這也是你人緣好的証明，記得要好好保持，期望你半年後順利畢業；Bug，你的改變大家都有看見，未來要繼續努力，趕快去國防部報到；昶閔，記得跟 Beer 結婚的時候不要忘了我們；頭源，班上的專業攝影師兼品酒師，跟你一起當家庭主夫真的超開心的，祝你未來兩個月就完成論文剩下的部份。

再次謝謝小臉蓉、秋如、頭源、羅委員、金門、pona、誌嘉在我最難過的時候陪我喝酒，跟你們喝酒是我最暢快的時候，可以放開心沒有拘束不計後果的喝，更謝謝大家都沒有給我壓力，能夠有你們這一群想到就會很窩心的摯友，是我進淡江運科所最大的收穫。

獅子培育中心的大家，欣妮姐、淑芬姐、玟玟黨員、你們真的運用所學，還買了巴斯光年資料夾當作增強物，讓我快快完成論文，還有淑玲、小阮、活動長、宇青、冠樺、雅鈞、山人、琳嬭、雅琪、雅集、盈妮，謝謝你們。還要感謝陶家學長姐對我的指導，嘉琪學姊、忠榮學長、龍文學長，不論是做人做事帶學弟，你們都是最棒的；阿伯，你的程式真的很強，要對自己更有自信一些，因為沒有你就沒有今天我的論文，謝謝你不辭辛勞的教導，還有敏琛學弟、長偉學弟，我一定會記得跟你一起去上海開心的日子，也謝謝你們對我論文的鼎力相助。

最後，深深的謝謝在這一段日子裡對我來說最重要的 nasima，跟妳在一起的四年三個月是我最快樂的日子，妳真的很好，妳一直都是我的驕傲，是我沒有好好珍惜這段緣份，錯的是我，沒有權利再要求什麼，再跟妳說一次抱歉，做過的我無法收回，只能在未來繼續付出。其實妳一直是我論文的最大支柱，只是我將愛視為理所當然，讓你難過，衝動行事一直以來都是我的缺點，經過這些風風雨雨之後，期望自己可以變得更成熟，也要認真的聽話和溝通，妳的話如同夏日甘霖，讓我清醒不少，也讓我更加確定自己，謝謝妳，抱歉。

再一次感謝論文寫作期間鼎力相助以及努力包容我的各位，我愛你們。

劉嵩瀚 謹致

2006 年 7 月 于研究室

目錄

頁次

中文摘要	
英文摘要	
誌謝	
目錄	I
圖目錄	IV
表目錄	VI
第一章 緒論	1
1.1 研究動機	1
1.2 研究目的	2
1.3 研究對象與範圍	2
1.4 研究內容	3
1.5 研究方法與流程	4
第二章 文獻回顧	6
2.1 智慧型軌道運輸系統	6
2.1.1 國外 IRS 發展現況	7
2.1.1.1 北美、加拿大先進列車控制系統 ATCS	7
2.1.1.2 歐洲鐵路運輸管理系統 ERTMS	7
2.1.1.3 法國地鐵 ASTREE 系統	8
2.1.1.4 日本新幹線 ATC 系統	8
2.1.1.5 中國智能鐵路系統(RITS)	9
2.1.1.6 德國軌道運輸系統智慧化	11
2.1.2 台灣地區智慧型軌道運輸系統架構	12
2.1.3 小結	14
2.2 閉塞系統概述	15
2.2.1 固定自動閉塞系統	16
2.2.2 準移動閉塞系統	17
2.2.3 移動自動閉塞系統	19
2.2.4 小結	22
2.3 各國移動自動閉塞系統之發展現況與比較	25
2.3.1 北美、加拿大 ATCS	26
2.3.2 歐洲鐵路運輸管理系統 ERTMS	29
2.3.3 法國 ASTREE 系統	31
2.3.4 日本新幹線 CARAT 系統	33
2.3.5 德國 LZB、FZB 系統	35

2.3.6	英國地鐵 Jubilee 朱比利線 WESTWACE 列車自動保護系統	36
2.3.7	小結	38
2.4	各國軌道容量解析模式	41
2.4.1	小結	42
第三章	研究方法	43
3.1	軌道分析方法容量	43
3.1.1	解析模式	45
3.1.2	最佳化模式	45
3.1.3	模擬模式	48
3.2	小結	51
第四章	軌道運輸路線容量分析模式構建	52
4.1	軌道容量分析基礎架構及定義	52
4.1.1	名詞定義	54
4.1.2	軌道容量分析架構	55
4.2	台鐵容量分析模式	57
4.2.1	台鐵軌道運輸系統特性	57
4.2.2	台鐵容量分析之解析模式	59
4.2.2.1	區間軌道容量	59
4.2.2.2	最小運轉時隔	59
4.2.2.3	待避損失時間之計算	60
4.2.2.4	運轉寬裕時間之決定	61
4.2.2.5	平均最小運轉時隔之計算	62
4.2.2.6	運轉方式及前後車速差之影響	63
4.3	模擬模式之構建	66
4.3.1	行車動力學	66
4.3.1.1	路線坡度對列車加減速性能之影響	66
4.3.1.2	線型曲線對列車加減速性能之影響	67
4.3.2	列車排點	69
4.3.3	列車運行	72
4.3.3.1	號誌安全時距	73
4.3.3.2	列車運行安全間距	80
4.3.4	小結	88
第五章	實證分析	92
5.1	模擬模式簡介	92
5.1.1	模擬模式模組介紹	92

5.1.2	輸入輸出資訊	92
5.2	模擬模式驗證	93
5.2.1	小結	97
5.3	敏感度分析	98
5.3.1	停站時間	98
5.3.2	列車加減速性能因子	99
5.3.3	運轉寬裕時間係數	99
5.3.4	運行列車車種組成	100
5.3.5	列車延誤及延滯	101
5.3.6	小結	102
5.4	情境分析	103
5.4.1	台鐵 FAS 軌道容量分析	103
5.4.1.1	台鐵 FAS 軌道容量現況	103
5.4.2	MAS 模擬結果	104
5.4.3	情境設定	105
5.4.3.1	現行列車任務下不同發車間距之容量分析	105
5.4.3.2	相同發車間距下不同列車組成之容量分析	108
5.4.3.3	不同發車間距下不同列車組成之容量分析	112
5.5	小結	116
第六章	結論與建議	119
6.1	結論	119
6.2	建議	120
	參考文獻	121
	附錄一 各國軌道容量解析模式變數定義	124
	附錄二 模擬程式程式碼	126

圖目錄

圖 1.1	研究流程圖	5
圖 2.1	ATCS 系統工作原理示意圖	7
圖 2.2	中國大陸智能鐵路系統系統組成示意圖	10
圖 2.3	德國軌道運輸系統智慧化系統架構示意圖	11
圖 2.4	IRS 系統實體架構	12
圖 2.5	台灣智慧型軌道運輸系統系統架構示意圖	13
圖 2.6	固定閉塞區間列車防護示意圖	17
圖 2.7	準移動閉塞區系統閉塞原理示意圖	18
圖 2.8	自動閉塞系統安全列車間隔示意圖	19
圖 2.9	移動自動閉塞系統無線訊息傳輸示意圖	20
圖 2.10	無線移動自動閉塞系統結構示意圖	21
圖 2.11	ETCS 各級系統示意圖	30
圖 2.12	CARAT 系統列車追蹤運行示意圖	33
圖 2.13	WESTRACE 移動自動閉塞系統工作原理	37
圖 4.1	軌道容量分析架構圖	56
圖 4.2	列車待避延誤示意圖	60
圖 4.3	列車運轉時間中待避損失時間之示意圖	61
圖 4.4	先行列車速度大於後續追蹤列車之瓶頸號誌時距	64
圖 4.5	先行列車速度小於後續追蹤列車之瓶頸號誌時距	65
圖 4.6	彎道阻力示意圖	67
圖 4.7	模擬列車運行軌跡圖	72
圖 4.8	固定自動閉塞列車安全時間間隔示意圖	82
圖 4.9	準移動閉塞列車安全時間間隔示意圖	84
圖 4.10	移動自動閉塞列車安全時間間隔示意圖	85
圖 4.11	SMB-V 方式移動閉塞列車間隔示意圖	86
圖 4.12	MB-V 方式移動自動閉塞列車間隔示意圖	87
圖 4.13	MB-V ₀ 方式移動自動閉塞停車間隔示意圖	88
圖 4.14	FAS 列車模擬程式邏輯程序流程圖	89
圖 4.15	FAS 列車模擬程式邏輯程序流程圖	91
圖 5.1	模擬模式驗證	94
圖 5.2	停站時間增量與容量關係圖	98
圖 5.3	列車加減速性能因子與容量關係圖	99
圖 5.4	運轉寬裕時間係數與容量關係圖	99
圖 5.5	列車組成與容量關係圖	100
圖 5.6	列車誤點與容量關係圖	101
圖 5.7	列車延滯與容量關係圖	102

圖 5.8	現行列車運轉依原班表發車時間之 MAS 列車運行圖	104
圖 5.9	發車間距=180 秒之列車運行圖	105
圖 5.10	發車間距=240 秒之列車運行圖	106
圖 5.11	發車間距=300 秒之列車運行圖	106
圖 5.12	發車間距=360 秒之列車運行圖	107
圖 5.13	不同發車間距下軌道容量比較圖	108
圖 5.14	發車間距=300 秒下(自強-莒光-電聯) 列車組成模擬運行圖	109
圖 5.15	發車間距=300 秒下(自強-電聯-莒光) 列車組成模擬運行圖	109
圖 5.16	發車間距=300 秒下(莒光-自強-電聯) 列車組成模擬運行圖	110
圖 5.17	發車間距=300 秒下(莒光-電聯-自強) 列車組成模擬運行圖	110
圖 5.18	發車間距=300 秒下(電聯-自強-莒光) 列車組成模擬運行圖	111
圖 5.19	發車間距=300 秒下(電聯-莒光-自強) 列車組成模擬運行圖	111
圖 5.20	發車間距=180 秒下(電聯-莒光-自強) 列車組成模擬運行圖	112
圖 5.21	發車間距=240 秒下(電聯-莒光-自強) 列車組成模擬運行圖	113
圖 5.22	發車間距=360 秒下(電聯-莒光-自強) 列車組成模擬運行圖	113
圖 5.23	發車間距=420 秒下(電聯-莒光-自強) 列車組成模擬運行圖	114
圖 5.24	發車間距=480 秒下(電聯-莒光-自強) 列車組成模擬運行圖	114
圖 5.25	發車間距=540 秒下(電聯-莒光-自強) 列車組成模擬運行圖	115
圖 5.26	發車間距=600 秒下(電聯-莒光-自強) 列車組成模擬運行圖	115
圖 5.27	軌道容量列車運轉時隔求解選擇示意圖	117
圖 5.28	FAS 下不同車種組成發車間距-容量圖	118
圖 5.29	MAS 下不同車種組成發車間距-容量圖	118

表目錄

表 2.1	中國智能鐵路系統組成	9
表 2.2	智慧型軌道運輸系統行控中心單元次系統	13
表 2.3	閉塞區間發展進程	15
表 2.4	閉塞系統列車停車目標點	22
表 2.5	閉塞系統優劣比較一覽表	22
表 2.6	北美鐵路 ATCS 系統試驗及發展一覽表	26
表 2.7	北美鐵路 ATCS 系統試驗實例	27
表 2.8	歐洲 ETCS 無線運用模式	30
表 2.9	ASTREE 系統試驗及發展	31
表 2.10	CARAT 系統試驗與發展	34
表 2.11	LZB 與 FZB 系統特性比較表	35
表 2.12	軌道運輸閉塞方式演進歷程	38
表 2.13	閉塞方式對列車容量之影響統整表	39
表 2.14	固定自動閉塞區間與移動自動閉塞區間系統優劣比較表	39
表 2.15	各國移動自動閉塞系統之比較	40
表 2.16	各國軌道容量解析模式比較表	41
表 3.1	軌道容量模擬模式文獻回顧一覽表	49
表 3.2	軌道容量分析方法方法論比較	51
表 4.1	定義軌道容量之基本要素及其分類	52
表 4.2	曲線速限表	68
表 5.1	發車 headway 及停站時間統計分配一覽表	93
表 5.2	模擬模式驗證 T-value 表(I-1)	94
表 5.3	模擬模式驗證 T-value 表(I-2)	95
表 5.4	模擬模式驗證 T-value 表(II-1)	95
表 5.5	模擬模式驗證 T-value 表(II-2)	96
表 5.6	列車組成與容量關係圖	100
表 5.7	模擬現行班表最大運轉時隔	103
表 5.8	採行 MAS 行車制不同發車 headway 下之軌道流量	107
表 5.9	發車間距=300 秒下不同列車組成模擬容量表	112
表 5.10	不同發車 headway 下特定列車組成模擬容量表	116

第一章 緒論

1.1 研究動機

1960 年代以降，世界先進國家積極使用先進之電子、通訊、資訊、定位、車輛、管理、控制等技術，整合運輸系統中之人、車、路等要素，建立涵蓋陸海空全方位之智慧型運輸系統(Intelligent Transportation Systems, ITS)。ITS 的發展初期多以公路運輸為主，相關交通建設預算亦以公路建設為優先，但隨著公路運輸基礎建設規模的擴增，雖大幅提高用路人的可及性，但伴隨而來的是交通日益壅塞、空氣污染、噪音、事故等社會成本的負擔；且車輛數持續成長卻產生供給創造需求的趨勢，使現有道路容量及停車空間均無法滿足需求。因此，唯有發展具有安全、快速、高運能、低污染、不受天候影響、對環境衝擊較小等特性的軌道運輸系統，方能符合運輸系統永續發展的目標，因此近年來軌道運輸系統漸受各國重視。

隨著軌道運輸與 ITS 的蓬勃發展，歐、美、日等鐵路運輸大國亦積極投入兩者結合的研發計畫，此即軌道運輸系統智慧化(Intelligent Railway Systems, IRS)。引進 IRS 可利用通訊、定位、電腦輔助等技術以改善行車制度，達到提升營運效率及減少人員財務損失之目標，在各國如火如荼地運用高科技於軌道運輸以提升安全、效率的趨勢下，我國對於軌道運輸系統智慧化之規劃與建置計畫，亦應及時進行。

目前台灣之軌道運輸遍及全國，舉凡台鐵、台北捷運、高雄捷運，以及正在建造中的高鐵，正在規劃中的台中捷運、新竹輕軌，各有其不同的服務涵蓋面。目前台鐵採用固定自動閉塞系統來確保列車行車安全，然而當因應特殊或緊急需求而加開列車時，由於大部分班表固定且行車制度上又有安全間距之考量，縮短發車間隔以提升容量之策略即難以奏效。因此提升鐵路運輸容量的根本方法即為改變行車制度，亦即在保證安全的前提下，藉由不同行車控制方式，進行多種列車調度以滿足乘客最大的需求。

目前各軌道先進國家皆由傳統列車運行模式(固定自動閉塞行車制)轉型，發展移動自動閉塞行車制，利用無線通訊技術精確掌握前後車位置、速度等重要資訊，在軌道安全第一優先的前提下，有效地縮小列車安全間距，使得因為採用傳統固定自動閉塞行車制而浪費的軌道容量得以作更有效率、更靈活的使用。

由於軌道運輸行車制度未來必然朝向低成本、低人為疏失、高安全的移動自動閉塞系統發展，因此本研究擬以台鐵某一區段為研究對象，藉由量化方法呈現採用移動自動閉塞系統後可提升之容量值，以供決策者參考。

1.2 研究目的

本研究將針對目前美國、歐洲、日本、中國大陸等軌道大國在智慧型軌道運輸上之發展進程與工作重點進行分析，揭櫫我國未來軌道運輸發展之方向。藉由回顧軌道發展先進國家的軌道容量計算方式，可提供吾人對此一方面發展之方向，並且考慮台灣特有之軌道設定條件及各相關影響因素，建構出一符合台鐵現況之軌道容量計算分析模式。運用軌道容量分析模式，可檢核目前台鐵系統所達成之軌道容量規模，及未來技術可行能夠達到之軌道容量，期能在安全行車的前提下，達到提昇軌道容量的目的。

本研究之研究目的如下：

- 1、構建符合台鐵特性之列車運行模式
- 2、建構單線列車運行模擬程式
- 3、進行敏感度分析探討影響因子對軌道容量之影響
- 4、探討採行移動自動閉塞行車制，對於軌道容量提升之變化

1.3 研究對象與範圍

軌道運輸之課題可供討論處相當多，本研究針對 ITS 中 IRS 部分下之 ATCSS(列車自動安全與控制系統)的 ATCS(列車控制系統)加以探討，並針對移動自動閉塞區間及固定自動閉塞區間之列車容量進行比較。

由於現今軌道運輸面臨多種運具交互作用，互相助益或衝突，例如航空業、客運業、甚或是未來即將上線營運之高鐵系統對於台鐵之長途運輸造成衝擊，而便利的公路運輸及客運業更是對台鐵之中短途運輸產生了莫大的影響，因此使得台鐵之運輸優勢大不如前，營運不易，目前台鐵有意取消夜間列車服務，由於其主要服務對象為長途旅次之旅客，但此一服務對象有轉移至公路客運業之傾向，故考量台鐵面對國內運輸環境一連串之衝擊，勢必將在其運輸服務上進行轉型，以達到靈活調度及提高列車容量之目的。

目前台鐵全線朝電氣化邁進，最高運行時速以自強號每小時 130 公里為最快。軌道部分採用電阻火花焊接長軌，使得鋼軌壽命得以延長約三分之一，可節省百分之五的牽引力，提高行車速度、減低車輛震度，增進乘車舒適度，降低噪音，促進行車安全；號誌系統目前採用 CTC 設備，利用電腦自動設定進路自動調度，取代現有由調度員控制進路，亦可改由調度員手動控制，有助於促進行車安全，增加路線容量及提高運輸能力之達成。

本研究係以台鐵為研究對象，其具有單/複線運轉、混合車種、及低等

級列車讓高等級列車先行.....等特性，同時亦可納入區段站內線型配置、空間環境(曲線、坡度)特殊條件等。

囿於人力物力，本研究將以台鐵多車種混合行駛、副線運轉即考量錯車因素為研究範圍，並以西正線南下「松山-新竹」段為實証對象，進行模式驗證與情境分析。

1.4 研究內容

欲探討採用不同型車制度下列車容量之改變，首先必須針對列車容量之計算方式加以探討，目前列車容量大多以通過區間之列車數乘以其列車設計容量，而相對於此，區間通過之列車數又與列車之安全時間間距及空間間隔間距有關。

列車之運行班表若能充分滿足顧客需求，並且兼顧營運上之需要，則可望獲得最大之收益及利益，此為最理想之情境；但實務運行之規劃上兩者無法兼顧，不可只以單方面之考量產生出列車運行時刻表，必須兼顧旅客之需求、列車運轉能力、環境時空限制條件.....等多方面因素，方能產生一適用之列車運行時刻班表。

透過對於軌道容量基本定義、軌道容量影響因素，及軌道容量評估方式之回顧，建構一符合國內台鐵現況之容量分析模式，以作為後續實例分析和後期研究之基礎。本研究之主要工作項目如下：

1. 各國鐵路容量分析之現況探討。
2. 台鐵容量分析考慮因素彙整及容量分析之方法探討。
3. 台鐵容量分析模式構建。
4. 考量站內錯車限制式之列車運行模式構建。
5. 進行列車模擬程式撰寫與情境分析(MAS)、敏感度分析。
6. 結論與建議。

本研究期能藉由引進移動自動閉塞區間之概念，鬆解列車運轉能力之部分限制條件，以縮短列車追蹤間隔，提高路線區間內通過列車數，達到提升軌道容量之目的，然後撰寫一列車模擬程式，提供一個衡量副線運轉區段單向容量的工具，並針對各項情境進行量化分析。

1.5 研究方法與流程

本研究之研究流程如圖 1.1 所示，各階段之工作內容說明如下：

首先針對問題加以定義，確立研究範疇及未來工作事項；本研究最終是欲以選定路段之情境，比較採取固定自動閉塞行車制與移動自動行車制下，其軌道容量之差異程度，並檢定是否具有顯著差異，藉由相關軌道容量分析方法的回顧，確立研究欲採取之方向，以模擬的方式進行列車運行情境探討。

文獻回顧方面主要根據軌道容量定義、軌道容量分析基本理論、軌道容量影響因素等多方面文獻回顧所彙整之結果，並且針對軌道容量分析方法加以探討，比較解析模式、最佳化模式、模擬模式等三種研究方法之特性以及適用情境，確定採取以電腦模擬列車實際運行為研究之方法論。方法論確立後回顧構建模擬模式所需之行車動力學、列車排點模式以期構建出能夠描述列車實際運行情況的模擬模式，並且進程式撰寫。

經由台鐵運轉資料蒐集、路線資訊、列車基本資訊，撰寫出列車模擬模式，並將之程式化。列車模擬程式之產出與現行實際營運列車驗證，確定模式可行且可代表真實世界 FAS 列車運行，便可進行 MAS 之情境模擬及敏感度分析，探討容量對於各變數變化之敏感度。最後，將兩者之結果利用軌道容量解析模式計算出兩者之差異並進行統計分析得出結論與建議。

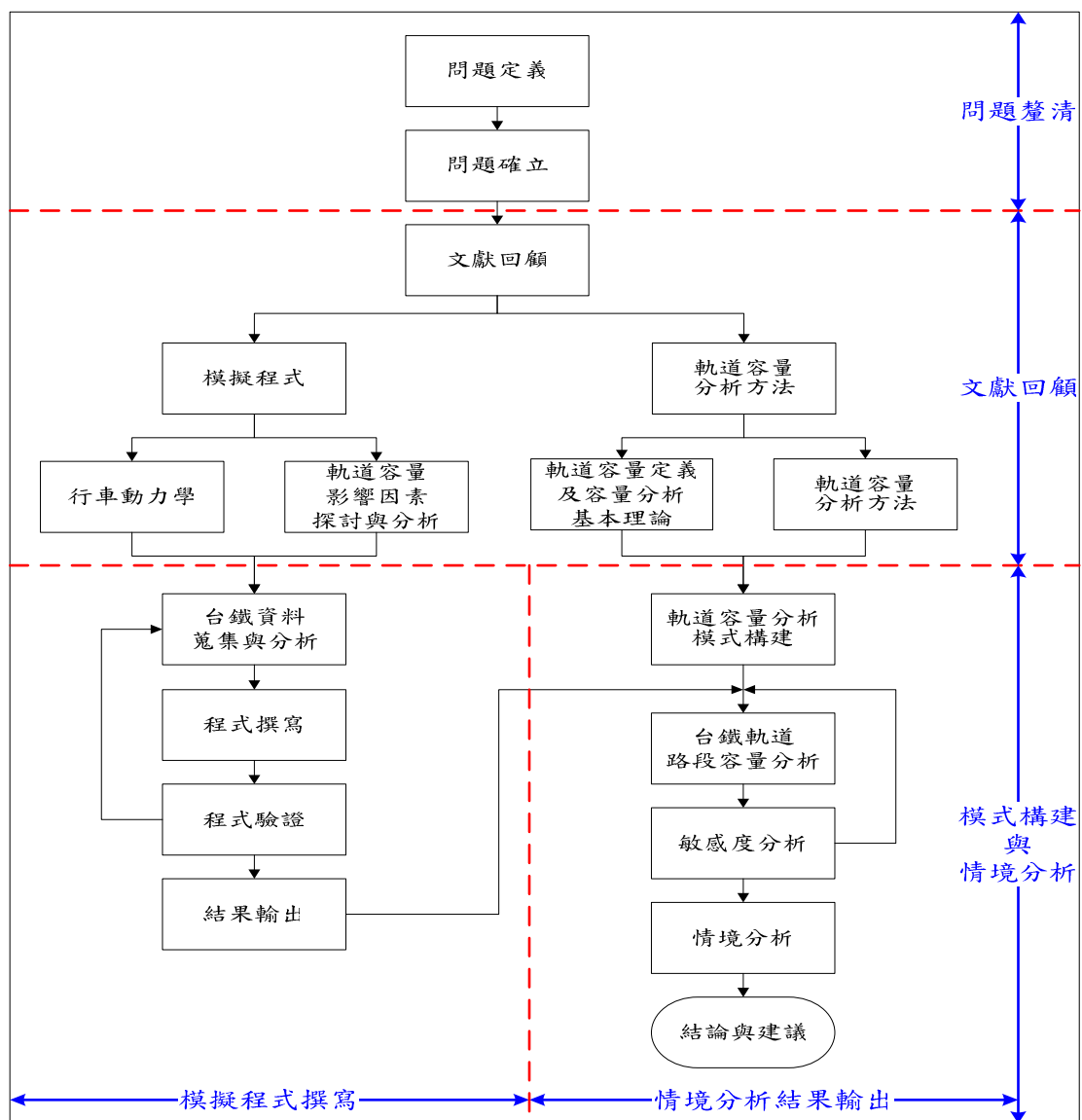


圖 1.1 研究流程圖

第二章 文獻回顧

智慧型軌道運輸系統奠基於通信系統基礎工程之上，期望能夠藉由科技之手段達到系統整合以及營運管理模式更加周延。IRS 與 ITS 最大不同在於智慧化軌道運輸系統有固定之運行時刻表，必須受到計劃性之限制，並且於區間運行時不可能改道，煞車制動距離長且慣性大，並且不論是國營亦或是私營之鐵路運輸系統，皆須對社會承擔責任，不可任意改變客、貨運之到/發列車時刻表；基於安全因素之考量，需對列車定位精度要求嚴密，其誤差不可大於公尺的範圍。由於傳統「固定自動閉塞區間」行車制之不足，無法針對需求即時反應，而浪費軌道資源，因此，希望藉助智慧型運輸系統(ITS)的概念，運用科技手段解決當前軌道運輸容量之不足，提升軌道運輸之效率。本章節將針對智慧型軌道運輸系統之發展，回顧國外各軌道先進大國之成功案例，以作為台灣軌道運輸發展之借鏡。

2.1 智慧型軌道運輸系統概述

交通部於民國 87 年 8 月成立智慧型運輸系統(Intelligent Transportation System,ITS)推動小組，先後完成「智慧型運輸系統發展與相關技術研討」、「台灣地區 ITS 綱要計畫」及「台灣地區發展智慧型運輸系統(ITS)系統架構之研究」等重要計畫，然台灣在此架構下發展之 ITS 然以公路運輸為主，對於之軌道、海運或航空等運輸系統較少探討。鑒於我國正積極發展國內捷運、輕軌、台鐵及正在構建中之高鐵等軌道運輸系統；參考美加、歐洲、日本、中國大陸等發展軌道運輸智慧化之先例，配合國內智慧型軌道運輸系統架構，達到國內軌道運輸系統之智慧化。

智慧型運輸系統結合電子、通訊、資訊、定位、車輛、控制等技術，整合於人、車、路三個面向，提供即時正確資訊，建立涵蓋路、海、空全方為之交通管理系統。而智慧型運輸系統(Intelligent Railway System,IRS)更針對列車行車安全、旅客資訊、電子票證、行車控制、列車定位、地理資訊系統、列車自動駕駛、列車監控等項目。

智慧型軌道系統，係指軌道運輸系統之營運機構於其營運管理作業中，針對預測貨實際之旅客、貨物運量需求，持續不斷以線上即時作業方式，進行資訊蒐集、整理、分析、規劃等作業，利用電腦妥為安排軌道列車營運班距、列車到、離站時刻表及司機人員執勤任務，以求在系統資源可行之限制下滿足運量需求，並使營運總成本最小、列車延滯最小、軌道容量最大之規劃作業要求(卓訓榮,蔡肇鵬,2002)。

2.1.1 國外 IRS 發展現況

近年來由於 ITS 發展，並且環境保育意識抬頭，各先進國家皆著手研擬抑制小客車成長之策略，例如道路定價(Road Pricing)、高承載(High Occupied Vehicle ,HOV)、共乘(Car Sharing)、課徵汽燃稅.....等策略，並且在京都議定書的催化下，鼓勵各種導致世界資源消耗，有助於環境維護之運輸政策，使得世界各國更加重視軌道運輸。目前軌道運輸智慧化之發展，例如北美鐵路之先進列車控制系統 ATCS 與先進鐵路電子系統 ARES、法國之及時追蹤自動化系統 ASTREE、歐洲列車控制系統 ETCS、澳洲之先進列車控制系統 AUSTRAC、日本之電腦與無線列車控制系統 CARAT、丹麥的智慧化列車 TB、荷蘭的 21 世紀鐵路 Rail 21、日本新幹線的電腦行車控制系統 COMTRAC 和列車運行管理系統 COSMOS...等，皆是成功案例。以下就目前世界各軌道先進國家之智慧型軌道運輸系統系統架構加以介紹。

2.1.1.1 北美、加拿大先進列車控制系統 ATCS

由北美鐵路協會所開發之先進列車控制系統 ATCS，將微處理器設置於鐵路移動設備上，經由數據無線電系統與中央控制系統(CCS)連接，使得提高其軌道運輸效率及安全性。

ATCS 相關列車控制之主要功能集中於地面中央控制調度中心及列車車載，因此舊有分布於軌道沿線之號誌可全數移除，中央控制調度中心之調度員根據電腦顯示之列車運行狀況，對相關之列車運行控制判斷並通過電腦下達各列車之列車運行計畫及其控制指令，而駕駛員則通過車載電腦依據中央控制調度中心經由無線電傳輸之控制指令控制列車，如圖 2.1 所示。

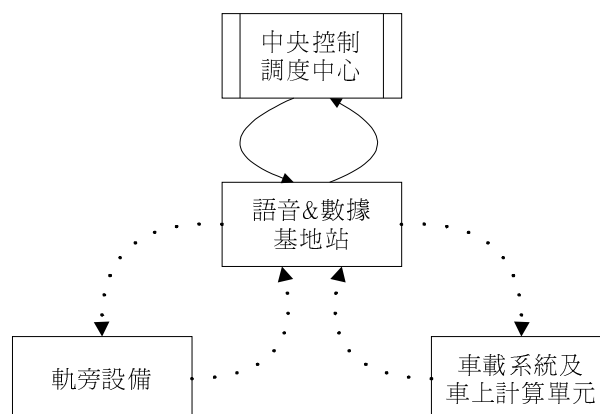


圖 2.1 ATCS 系統工作原理示意圖

2.1.1.2 歐洲鐵路運輸管理系統 ERTMS

由歐盟推動之歐洲鐵路運輸管理系統(European Railway Transportation

Management System, ERTMS)計劃，主要包含三個系統：1.先進區域號誌控制系統、2.先進資訊發佈系統、3.先進交通管理系統。其中又以資訊系統及號誌系統之發展為整合歐洲鐵路之重要關鍵。ERTMS 考量列車運行實之延誤時間進行列車自動排程，並且提供列車安全間距及列車旅行時間等資訊，提高軌道運量，延長軌道運輸之生命週期。

2.1.1.3 法國地鐵 ASTREE 系統

法國鐵路全長 52,000 公里，車站 5,000 個，網路遍及全國各大小城市，由 SNCF 國營鐵路公司所管理，尤其又以 ASTREE (Automatisation du Suivi en Temps Reel)連續即時追蹤自動化系統最具代表性。

2.1.1.4 日本新幹線 ATC 系統

日本新幹線先進列車控制系統(Advanced Train Control ,ATC)不於地面軌旁設立號誌系統，以 ATC 列車自動控制系統控制列車之運行速度，採固定自動閉塞行車制調整列車運行間隔。

ATC 列車設備接收來自軌道電路之訊息，使列車號誌顯示相對應的數字速度信號，並且通過車載電腦將接收到之允許速度與列車實際運行速度相比較，按照允許速度之要求，維持原速度繼續運行或採取減速、制動。

地面設備之功能主要如下：1.偵測軌道電路狀態(佔用、閒置)、2.偵測有無先行列車，通過連鎖裝置了解列車間之間隔以及路線條件(曲直、坡度)，依地面設備之偵測結果向列車發送列車允許速度。

新幹線 ATC 系統對於列車進行自動控制，當列車實際運行速度低於 ATC 信號指示允許速度時，列車速度由司機員控制；當列車實際運行速度高於 ATC 信號指示允許速度時，由 ATC 裝置自動剎車，直到列車速度降至接近允許速度，自動緩踩剎車。列車近站時，由 ATC 系統控制列車自動減速至 30km/hr，當列車速度降至 30km/hr 以下時，司機員須辦理確認操作，否則不緩踩剎車；當列車辦理進站作業時，由 ATC 裝置自動剎車，列車速度降至 ATC 信號以下時，自動緩踩剎車。

ATC 裝置自動減速之動作通常使用常用制動，唯有當列車速度達到或超過高一級信號指示的允許速度時，才使用緊急制動剎車。

2.1.1.5 中國智能鐵路系統(RITS)

中國大陸之智能鐵路系統(Railway Intelligent Transportation System , RITS)將先進資訊技術、導航定位技術、組合最佳化技術、影像分析技術及電腦人工智能技術等技術，綜合應用於整個鐵路運輸體系，以發揮其即時、準確、高效能之表現的鐵路運輸管理系統。

中國智能鐵路系統包括六大子系統，整理如下：

表 2.1 中國智能鐵路系統組成

1.先進運輸管理系統 ATMS	通過自動編制各級運輸計劃，對列車裝卸及運行進行控制與指揮，產生各種運輸方案
2.先進運輸信息系統 ATRIS	以鐵路運輸管理信息系統 TMIS 為主體，提供相關運輸信息給各級管理人員、列車調度、控制及指揮人員
3.先進列車控制系統 ATCS	根據列車運行情況及外界環境的變化，實施自動化或半自動化的列車運行及道岔路徑控制，使列車保持更高的速度及更短的間隔安全行駛
4.先進用戶信息系統 AUIS	先進貨運服務系統、客運預售票系統、旅客誘導及服務系統
5.先進運輸設施管理系統 ATFMS	以鐵路運輸設施的高效管理為目標，包括鐵路運輸設施的完好狀況和維護、維修需求資訊的蒐集、處理及管理方案的制定
6.先進運輸安全管理系統 ATSMS	以鐵路運輸安全、暢通為目標，即時、準確地蒐集並提供鐵路運輸事故資訊，迅速進行事故救援及處理，對鐵路災害進行預防及整治

資料來源：本研究整理

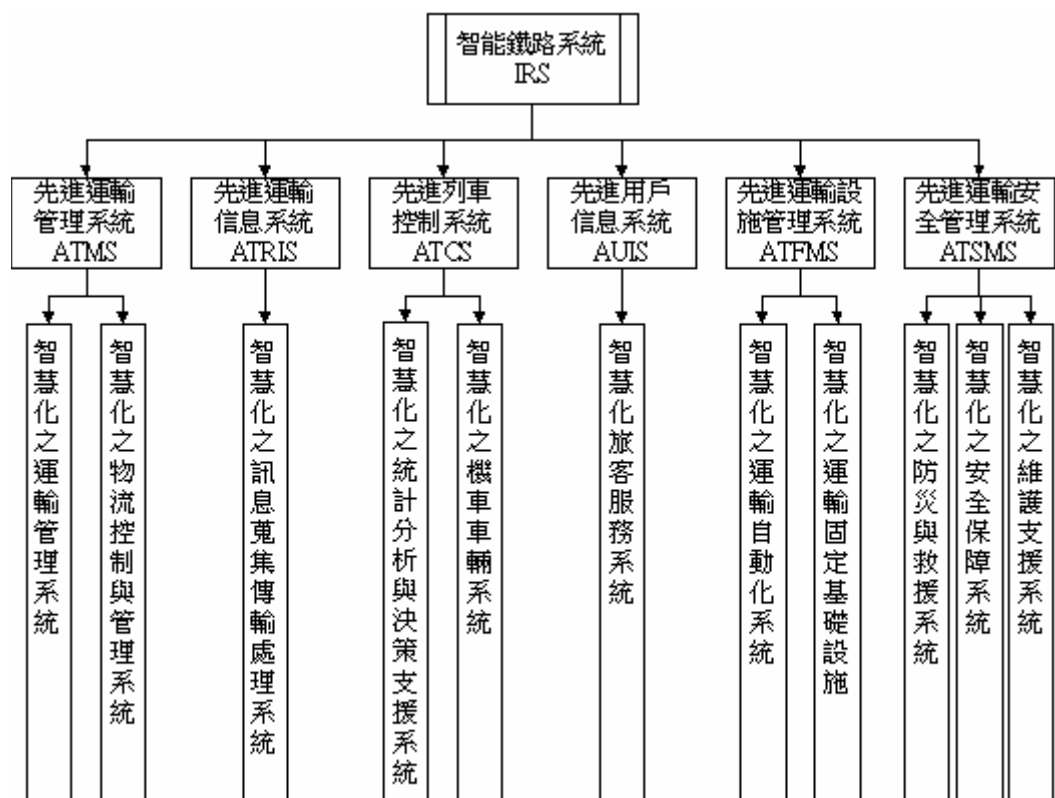


圖 2.2 中國大陸智能鐵路系統系統組成示意圖

2.1.1.6 德國軌道運輸系統智慧化

德國軌道運輸系統智慧化系統架構主要分為行車安全與營運控制、資源分配及績效管理、乘客資訊與服務三方面，並且在此三大面向下，構建各項結合電子、通信、資訊、控制等技術之子項目。完整之系統架構如圖 2.3：



圖 2.3 德國軌道運輸系統智慧化系統架構示意圖

2.1.2 台灣地區智慧型軌道運輸系統架構

智慧型軌道運輸系統實體架構由旅客(Travelers)、車輛(Vehicles)、車站與行控中心(Station and control center)、軌道(Railway)四大單元組成，如圖 2.4 所示，並透過有線/無線電傳輸系統：光纖傳輸、廣域寬頻網際網路傳輸、短距無線傳輸等組合(朱來順，2002)。

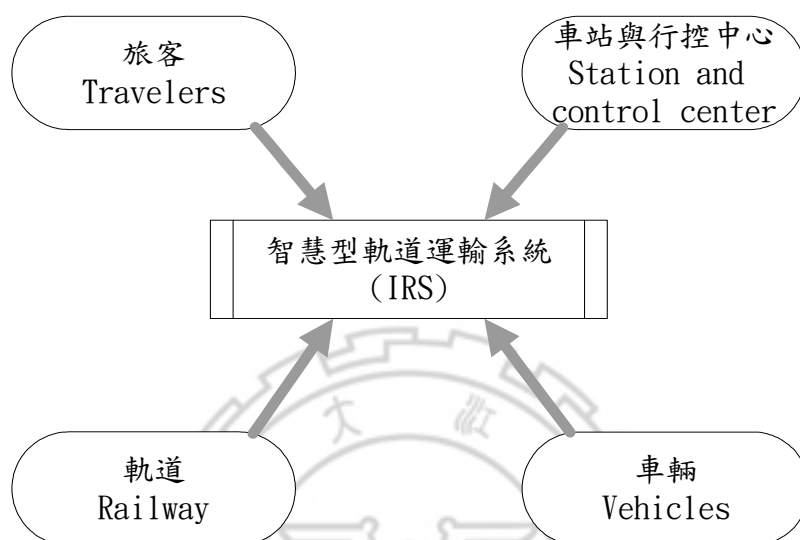


圖 2.4 IRS 系統實體架構

台灣 IRS 之應用主要是參考德國、中國大陸等軌道運輸智慧化系統架構之項目分類，並依照「台灣地區發展智慧型運輸系統綱要計劃-ITS 發展領域與使用者服務之供、需調查分析」之構想，設計出軌道運輸系統智慧化(Intelligent Railway Systems,IRS)系統架構。包括先進運輸管理系統(Advanced Transportation Management Systems,ATMS)、先進列車控制與安全系統(Advanced Train Control and Safety Systems,ATCSS)、先進交通資訊服務系統(Advanced Traffic Information System,ATIS)、先進緊急救援系統(Emergency Management Systems,EMS)、電子付費系統(Electronic Payment Systems,EPS)、先進貨運管理系統(CVO)等六大子系統，如表 2.2 所示，其細部組成如圖 2.5 所示。

表 2.2 智慧型軌道運輸系統行控中心單元次系統

1. 先進運輸管理系統 (ATMS)	資源分配、績效管理、智慧化排程排班、智慧化調度指揮系統
2. 先進列車控制與安全系統 (ATCSS)	列車行車安全、營運控制、先進號誌控制系統、監控系統、資訊擷取系統
3. 先進交通資訊服務系統 (ATIS)	乘客與貨物資訊與服務、先進軌道通信系統、軌道資訊網路系統
4. 先進緊急救援系統 (EMS)	列車自動監測與診斷系統、地面監測與診斷系統、自然災害警報系統、車站與列車內環境自動偵測與控制系統
5. 電子付費系統 (EPS)	自動收費系統(包含客貨運訂票、託運)、智慧型票務系統
6. 先進貨運管理系統 (CVO)	列車車廂調度指揮與倉儲分配、物源預測與分配

資料來源：交通部,2001



圖 2.5 台灣智慧型軌道運輸系統系統架構示意圖

2.1.3 小結

軌道運輸系統主要目的為在安全的前提下，準時、有效且大量將人、貨運送至目的地，利用先進之通訊、定位、電子、控制、車輛等技術，使軌道運輸達到智慧化，藉以提高軌道運輸之安全、準點性、以及路線容量。

藉由回顧各國之軌道運輸系統智慧化發展之現況可以發現，未來軌道運輸之發展必定朝向以科技化手段達到提高運輸系統運行安全、提高營運管理績效、提昇軌道運輸之客貨服務品質、即時掌握客源物流及運量預測之機制與能力、強化系統設施軟體演算模式及運算能力、自動更新列車運行計劃及行車班距、廣泛使用寬頻網際網路、自動偵蒐行車資訊、電腦自動化分析與應用資訊、自動運算與更新...等使軌道運輸發揮最大運輸能量之目標；國內近年來也積極發展軌道運輸，對於此一發展趨勢自然不可忽視，應用於捷運、高鐵、以及傳統之台鐵系統將可對於其運輸效能向上提昇。

本研究著重於智慧型軌道運輸系統(IRS)中容量之探討，是針對智慧型運輸系統(ITS)下智慧型軌道運輸系統(IRS)中之先進列車控制與安全系統(ATC/ATC-SS)內之自動列車控制系統(ATCS)為研究對象，探討在 IRS 架構下，列車容量之計算方式，並且更進一步探究如何可有效地提高軌道容量，達到最大效用。



2.2 閉塞系統概述

軌道運輸系統中，由於必須保證列車行車之安全，因此產生閉塞區間之概念，同一區間除了不可有對向列車闖入，同向運行之列車更應維持某一程度之安全間隔，以防止發生列車碰撞事故。列車在站間區間運行主要分為兩種方式：空間間隔運行法、時間間隔運行法。

時間間隔運行法乃指車站每隔固定時間即向區間內發送一列車，站間區間內不設號誌設施，列車必須完全按照列車運行計劃及列車時刻表之指示運行，列車與列車之運行環環相扣；一旦發生列車故障或事故，則後行列車之運行將產生骨牌效應，美國學者 Willian Fothergiel Cooke 於 1842 年首次指出其危險性，建議應在區間內增設號誌設施以提高其安全係數，如表 2.3 所示。

空間間隔運行法是指車站每隔一定之空間間隔即向區間內發送列車，在區間內，前後列車之追蹤運行也需維持一固定之空間間隔，以保證能夠安全地完成制動而不發生碰撞，有效保障軌道運輸之安全。

表 2.3 閉塞區間發展進程

1851 年	鐵路運輸始使用電報憑證，首次實現空間間隔運行法
1855 年	Messers. George Dugmore 和 George Millward 首次提出軌道電路之概念，申請英國專利，奠定空間間隔法之基礎
1911 年	美 Sedwick N.Wight 提出絕對閉塞，首次提出閉塞之概念，同一區間或同一閉塞分區內只允許一輛列車運行。

資料來源：本研究整理

一般來說，閉塞可分為電氣路籤(牌)閉塞、固定自動閉塞、準移動閉塞、以及移動自動閉塞等幾種。

固定自動閉塞是將站間軌道分成固定之區段，每個區段為一個閉塞分區，當分區內有列車佔用時，其他列車不能進入，以避免列車發生碰撞事故。

移動自動閉塞系統乃是一種基於無線通訊的閉塞技術，主要目的為在保障列車行車安全情況下，並且以此為前提下提高軌道行車密度。在移動閉塞區間行車制下，後行列車的位置不受固定區段的限制，只與前行列車的位置有關，保證列車最小運行間隔。

準移動閉塞介於固定自動閉塞與移動自動閉塞之間，主要可將其視為改良型之固定自動閉塞系統。藉由更細劃分傳統之固定閉塞系統，達到近似於移動自動閉塞系統之效果，以提高軌道容量。

任何一種閉塞方式皆是以安全為目的，至少應包括下列系統，以確保軌道運輸之安全[1]：

- (1) 列車定位系統：可有效即時地獲取列車位置資訊，以確保行車安全。
- (2) 訊息傳輸系統：確保列車與列車間、列車與車站間、列車與軌側設施間訊息傳輸無間斷即無資訊流失。
- (3) 列車間隔控制系統：確保列車運行間距，以免後行列車冒進未授權之區間發生意外。

2.2.1 固定自動閉塞系統

固定自動閉塞系統(Fixed Automatic-block System, FAS)主要特徵為路線上劃分固定之閉塞分區，並且於分區起點設置信號機進行防護，分區長度固定不變，利用軌道電路當列車鋼輪通過鋼軌時會產生短路的特性以檢測此閉塞區間之佔用情形，或利用計軸器測知。通過軌道電路、電纜、軌側信號機將列車位置速度等訊息傳送給司機或列車自動控制系統，確保列車間保持若干閉塞分區以達成安全之目的。

固定自動閉塞系統多配載自動列車保護系統(Automatic Train Protection, ATP)，ATP 系統藉由對列車間之安全間隔、超速...等控制保障列車行車安全。固定自動閉塞系統中，每一閉塞分區均有特定之速限限制，隨著前車運行狀況或自身列車運行狀況而改變，當閉塞分區之速限小於列車運行速度時，ATP 系統將會自動發生作用，緊急制動，以免冒進前車所在之閉塞分區，發生事故。ATP 系統之軌側設備以一定間隔連續地向列車發送列車速度控制訊息，作為列車之行車指導。列車速度控制碼乃是經由軌道電路、軌間應答器、感應線圈、或是無線通訊等方式傳輸予列車。列車速度控制訊息包含兩部分：(1)列車所在之閉塞分區最高速限(2)欲進入分區之速限，也稱目標速度。列車根據接收到之訊息，採取速度控制，以保障列車運行間隔。

在固定自動閉塞下，已編碼之軌道電路是列車運行中車上與地面訊息傳輸的主要通道。軌道查詢應答器、軌間電纜等作為列車位置參考點位列車提供絕對位置訊息。列車的運行權限在列車通過定點位置時由地面已機車訊號的形式發送給列車或司機員。在這種情況下訊息傳輸量相對較小，訊息的傳輸屬於點式傳輸。

固定自動閉塞系統訊息傳輸方式，乃是透過軌道電路傳輸訊息碼，因此，需要大量軌側設施，維護不易，並且訊息傳輸量與移動自動閉塞系統之無線通訊方式無法相比，若欲提高訊息傳輸量，則必須提高傳輸之頻率，此舉又會造成訊號損耗過大，傳輸距離短，因此，使用軌道電路傳輸訊息將無法滿足高速化軌道運輸之需求。

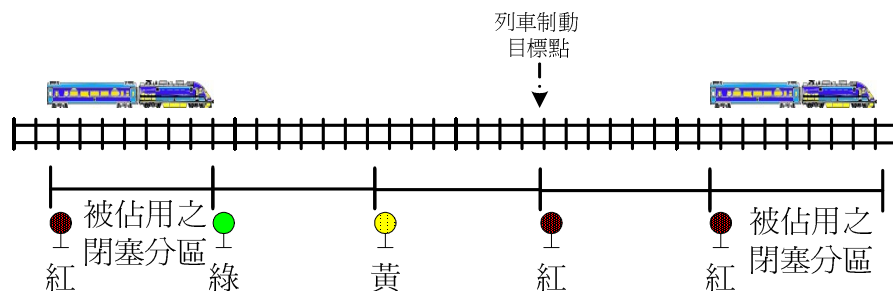


圖 2.6 固定閉塞區間列車防護示意圖

具有固定閉塞分區之固定自動閉塞系統能有效地確保行車安全，但相對的必須犧牲其軌道行車效率。閉塞分區長度之劃分乃是按照通過區段中最不利情況設計，按照通過之最長列車長度、最高列車行駛速度、最大列車載重量、最差制動率參數...等種種最不利狀況設計，以確保行車安全。採取 FAS 下所需保持之閒置閉塞分區取決於軌側信號機之顯示制度。依照上述設計決定之閉塞區間長度，因是考慮最不利狀況下之產出，對於其他不同速度、不同牽引、不同等級、不同組成、不同之列車將無法一體適用，達到最佳之配置。例如三顯示號誌之固定自動閉塞系統下，前後列車之間間隔通常至少保持兩個固定閉塞分區，若假設每個固定分區平均長度為 1.8 km，則前後列車之間間距便有可能達到 5.4 km 左右，但在實際運行時，若是以常規最高行駛速度 120 km/h 條件下，其常用制動煞車距離僅需 1.5km 即可完成煞車動作，產生浪費軌道容量之現象，是其缺點。

2.2.2 準移動閉塞系統

目前世界各國城市軌道運輸，採用音頻軌道電路，以確保列車行車安全並改善傳統固定自動閉塞系統容易產生軌道容量浪費之現象；此種軌道控制方式有別於傳統固定自動閉塞其利用軌道電路與軌側信號機連鎖，劃分出明確之閉塞分區；雖然其閉塞區間長度為可變的，但仍是以固定閉塞系統之概念延伸將軌道細分為許多段，此又與利用無線通訊技術為主，閉塞分區長度及位置隨時改變之移動自動閉塞系統有所不同，且界於此兩者之間，因此稱之為準移動閉塞系統。

利用軌道電路、感應環線、應答器等設備判斷軌道閉塞分區之佔用情況，並且利用以上設施傳輸訊息，可允許大量之訊息傳輸。後續追蹤列車再依據根據前行列車傳送之可運行距離，採取加減速或制動之行為。但採用準移動閉塞系統時，由於仍有傳統實體閉塞區間存在，因此，仍需仰賴軌道電路作為閉塞分區之劃分，後行追蹤列車之停車目標點為先行列車佔用分區之外，參見圖 2.7：

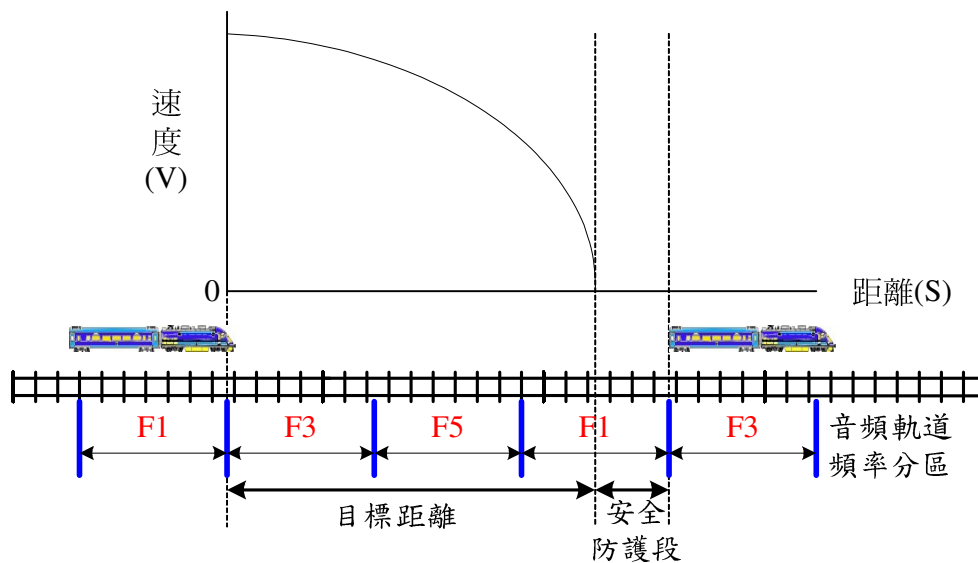


圖 2.7 準移動閉塞區系統閉塞原理示意圖

準移動閉塞乃採用音頻數字軌道將實體軌道劃分為物理上之軌道閉塞區間，並加上軌旁電路設備，檢測軌道閉塞區間之佔用情況，並且可用餘裕之頻寬作為列車傳送即時行車資訊之用，達到移動自動閉塞系統中車載控制系統列車之自動防護即自動駕駛功能。並且軌旁電路設備可做上下行切換之功能，使得軌道可靈活運用。準移動閉塞系統列車運行間隔之控制藉由即時傳輸所得之其方軌道閉塞區間之狀態、路線參數，並結合列車牽引參數透過車載計算機即時計算閉塞區間之長度，其列車間隔大小為可變，且列車間隔長度可包含數個音頻軌道區間，改進了固定自動閉塞系統列車間隔長度過長，造成軌道容量浪費之情形，可提高列車行車密度。

準移動閉塞系統利用軌道區間之分界確定列車之實際位置，作為列車車載設備計算列車位置之校正，摒除了輪軌運輸系統由於車輪滑行、列車過彎車輪空轉及磨損情形而產生列車側速與測距之誤差，提高列車定位精度，使得提高軌道容量之目的實現得以實現，台北捷運即是準移動閉塞系統最佳的例證。

2.2.3 移動自動閉塞系統

由於軌道運輸中列車運行過程，由於安全之緣故，列車間必須維持固定之追蹤運行間隔。列車追蹤運行間隔取決於後行追蹤列車之安全制動距離和緩衝距離，因此，若以此切入，藉由控制後續追蹤列車維持固定之追蹤間隔，將可縮短列車運行間隔，提高軌道容量，此即移動自動閉塞系統(Moving Automatic-block System, MAS)之原始概念。

移動自動閉塞系統取消物理之實體閉塞分區劃分，將路線分為若干路線單元，每個單元長度由幾公尺到十幾公尺之間，移動閉塞分區由數個路線單元組成，組成之單元數目隨著列車之位置及速度而變化，並且分區的長度為動態變化，故名之。

移動自動閉塞系統組成主要有無線通訊系統、車載設備、區域控制器，及控制中心等。經由無線通訊方式將列車位置、速度、及列車性能等資訊傳送給區域控制器；區域控制器追蹤列車並利用無線傳輸方式對列車發送運行權限，利用無線通訊系統進行雙向通訊；車載設備包括無線電台、車載電腦、查詢應答器。列車將蒐集到之訊息、數據經由無線通訊發送給區域控制器，以協助決策，並對接收到的命令進行確認、執行，如圖 2.8 所示。

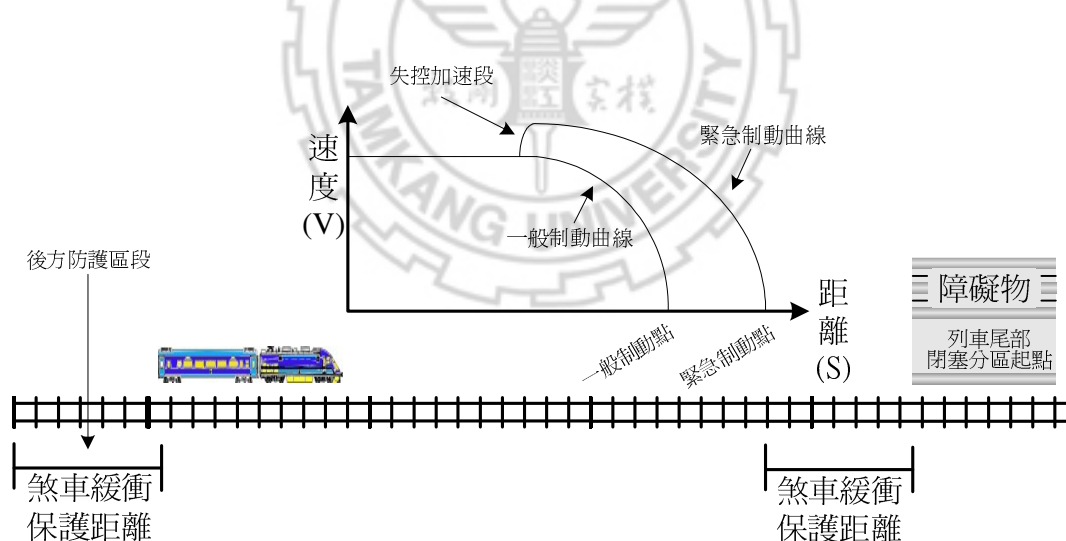


圖 2.8 自動閉塞系統安全列車間隔示意圖

在移動自動閉塞系統中，軌側不設置列車通過信號機，僅保留車站之進出站信號機，以供車站接發車之用。車站設置車站控制中心(Station Control Center, SCC)，SCC 與列車之車載計算機(On-Board Computer, OBC)進行雙向訊息傳輸，即時傳輸列車即時速度、位置、前後列車運行狀況、允許速限、列車運行準點性...等訊息，各列車依據列車運行圖及 SCC 傳送控制訊息兩相比較，考慮列車牽引特性、區間軌道地理空間特性之各項參數，採取較低之列車運行允許條件運行。對於欲進站之列車，根據 SCC 所給予之列車調

度命令決定列車是否允許辦理進站及進站股道。

移動自動閉塞根據無線通訊網，即時確定先行列車位置與速度，以進行列車安全間隔控制。閉塞方式以自身列車車首至先行列車之車尾作為閉塞區間，包括列車長度加上最大制動距離，並於自身列車後方加上一防護距離，以保證列車之行車安全。由於閉塞分區隨著列車運行、路線條件改變經由車載電腦計算改變閉塞分區長度，並且閉塞分區隨著列車同步移動，故得移動自動閉塞之名。

早期之移動閉塞系統是利用軌間之感應環線來定位列車及實現車載計算機(VOBC)與車輛控制中心(VCC)之間的連續通訊。現今大多移動自動閉塞系統採用無線通訊系統達成各子系統間的通訊。在採用軌旁基地站的無線通訊系統中，系統一般以 100%的無線通訊冗餘率進行基地站佈設，以預先消除在當某個基地站故障時能出現的訊號盲區。當列車行駛至閉塞分區端點時，後方控制器將列車到達之訊息傳送給前方控制器，並同時命令列車調整其通化頻率；兩相鄰之控制區域有部分重疊，使得可以在列車移交轉換作業時達到連續之無線通訊，如圖 2.9 所示。

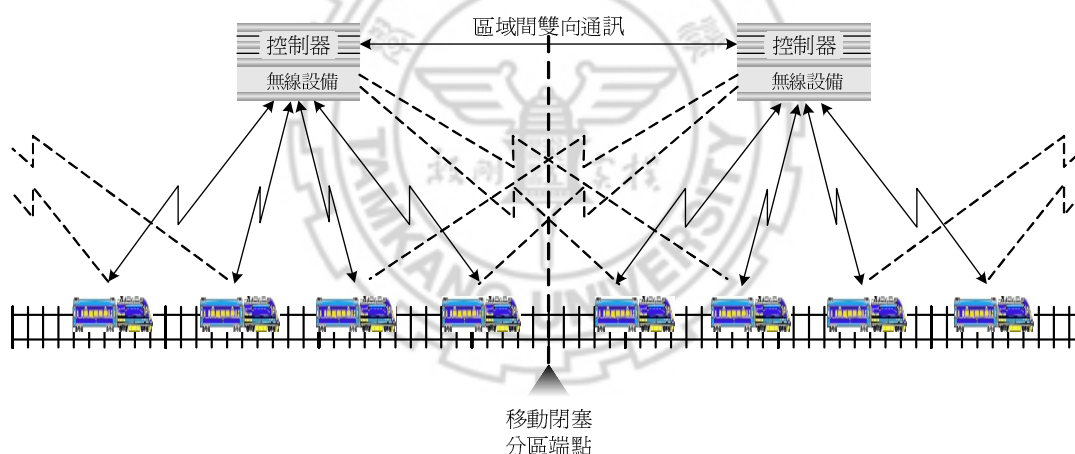


圖 2.9 移動自動閉塞系統無線訊息傳輸示意圖

移動閉塞系統採用無線數據通訊系統作為各列車運行控制子系統間相互之通訊方式，如圖 2.10 所示，採用國際通用標準 802.3(乙太網路)作為列車控制子系統間之通訊標準，以 802.11 作為無線通訊之標準，皆符合 IP(Internet Protocol)通訊協定。

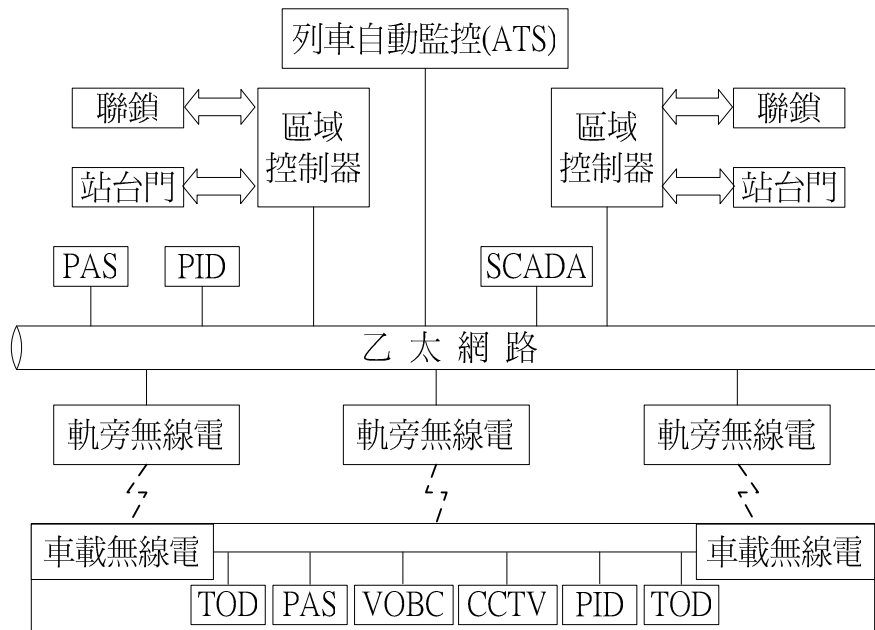


圖 2.10 無線移動自動閉塞系統結構示意圖

(CCTV：閉路電視，PAS：乘客廣播系統，PID：乘客嚮導系統，SCADA：電力監控系統，TOD：司機顯示，VOBC：車載控制器。)

資料來源：[10]

最早使用移動閉塞技術的紀錄可追溯到溫哥華的無人駕駛輕軌系統，至今已安全運行近 20 年，驗證了移動閉塞系統技術之安全性及可行性，另外，移動閉塞技術也在北美、歐洲、日本...等地有所應用。新近之移動閉塞系統建置例如漢城地鐵，或由舊系統改造之系統如紐約卡納西線、巴黎地鐵 13 號線，多是利用無線通訊技術達成。

2.2.4 小結

固定自動閉塞與移動自動閉塞其根本差異在於閉塞分區之形成方式。固定自動閉塞之分區行程式考慮運行於區間中列車最車運行參數、牽引類型、制動能力、區間坡道、彎道...等，再根據列車號誌為顯示方式(例如二顯示、三顯示、多顯示)來決定閉塞分區長度，並且使用與軌道電路聯鎖之通過信號機作為區間分隔標誌，區間一經劃分，無法任意更動，因此隨著列車運行速度之提昇或列車性能之更新，固定之閉塞分區長度勢必將限制軌道運輸之發展。

移動自動閉塞系統其閉塞分區為移動，並且取消軌側信號機及其他軌側設施，閉塞分區之形成為隨機的，依照自身列車與前方列車之即時運行速度、列車位置、車輛載重、列車節數、車廂長度、牽引能力、制動性能、坡度、彎道、天候...等列車及路線之特性與參數透過車載電腦決定閉塞長度，由自身列車車首至前行列車之車尾或進站信號機表現，因此除了無固定之閉塞分區劃分外，車站區間內之閉塞區間數也非一固定值。

前文所提及三種閉塞行車制，最主要差別在於實體設施的有無、列車追蹤停車目標點之不同，參見表 2.4。由於列車控制行車制度之不同，因此對於軌道容量及其他相關績效也有所改變，如表 2.5 所示：

表 2.4 閉塞系統列車停車目標點

閉塞系統	列車停車目標點
固定自動閉塞系統	可運行之固定閉塞區間終端端點
準移動閉塞系統	車站停車點前方+安全防護段 路線區間中軌道電路區間末端+安全防護段
移動自動閉塞系統	前行列車尾端+安全防護段

資料來源：本研究整理

表 2.5 閉塞系統優劣比較一覽表

	固定自動閉塞系統	準移動閉塞系統	移動自動閉塞系統
優點	(1) 有效保障列車行車安全。	(1) 列車定位精度高。 (2) 安全防護段位於車站停車點之前方、路線區間中軌道電路區間末端之後方，確保列車安	(1) 由於列車間隔乃按照後續列車於當時速度計算所需之煞車距離，並且加上安全保護段距離，確保列車不會發生追撞事故。 (2) 閉塞分區為移動，可有效

		<p>全。</p> <p>(3) 軌道電路設備可上下行切換，提高其靈活度。</p> <p>(4) 音頻數字軌道可實現其定位且有校正之功能；且可用於訊息之傳輸，無上下行訊息傳輸相互干擾之情形。</p> <p>(5) 提高運輸效率。</p> <p>(6) 閉塞分區具有即時計算與即時控制之功能。</p>	<p>提高軌道容量。</p> <p>(3) 無地面信號機，可有效降低成本。</p> <p>(4) 列車-軌側設施雙向傳輸訊息，訊息荷載量大，有助於實現無人駕駛。</p> <p>(5) 系統在滿足相同旅運需求下採取小編組、高密度之行車方式，減少旅客候車時間、縮小等候月台寬度及空間，節省成本。</p> <p>(6) 移動閉塞系統可直接疊加至既有 ATP 或音頻數字軌道電路系統上，有助於混合列車之運行模式。</p>
缺點	<p>(1) 閉塞分區之劃分是依最不利狀況決定，因此對於較高等級之列車則將浪費軌道容量。</p> <p>(2) 閉塞分區長度無法改變以因應不同列車編組或不同牽引之機車。</p> <p>(3) 後行追蹤列車無法掌握前行列車速度、位置、列車狀況等訊息。</p> <p>(4) 若欲以縮短閉塞分區長度來提高軌道容量，其經濟效益極低。</p> <p>(5) 對於突發狀況無法即時反應，一旦偏離列車運行計劃，容易造成大範圍之延誤發生。</p> <p>(6) 由於無法確切掌握</p>	<p>(1) 細分軌道閉塞區間致使成本提高。</p> <p>(2) 列車運行密度受軌道電路區間長度、軌道電路分界分佈設計、區間最大允許速度影響。</p>	<p>(1) 一旦通訊中斷將導致重大事故。</p> <p>(2) 民眾對其信賴度不高。</p> <p>(3) 對通訊系統依賴程度過大。</p>

	<p>運行中之列車速度、位置，使得中央列車調度員無法產生最適之調度方式。</p> <p>(7) 訊息傳輸量少。</p>		
--	---	--	--

資料來源：本研究整理

由於固定自動閉塞系統之訊息傳輸乃透過軌道電路傳輸訊息碼，無法處理大量訊息且訊號容易損耗，列車安全堪慮。因此，未來當採取移動自動閉塞系統，應使用以通訊為基礎之列車控制(Communication-Based Train Control)，採用無接縫無線通訊技術(Mesh network)，保持連續且即時通訊，以達到列車與軌側設施間雙向且大量之連續訊息傳輸，確保列車安全間隔與超速防護，根據自車實際速度及與前行列車相對速度調整閉塞分區長度，縮小行車間隔、靈活調度，達到提高軌道容量之目的。



2.3 各國移動自動閉塞系統之發展現況與比較

移動自動閉塞與傳統固定自動閉塞根本上之差異，在於其利用以通訊為基礎之列車控制(Communication-Based Train Control, CBTC)技術，基於無線通訊的移動自動閉塞系統下，需要列車將位置和速度訊息連續即時傳送給無線閉塞中心(Radio Block Center, RBC)，RBC 在將運行權限和速限等訊息發送給列車。為了滿足訊息傳輸需求並減少路測設施之建置及維護成本，採用無線傳輸(GPS)為主，查詢應答器為絕對位置參考點的訊息傳輸方式。

若充分利用智慧型運輸技術，搭配定位、通訊、電子、控制.....等技術，將可能提高軌道容量達一倍以上。此處所指為僅考慮區間運行狀況之軌道容量計算，若是軌道路線之瓶頸受限於車站、列車編組、路線配置.....等其他外在因素，則需另行考慮。

將區間行車控制改變為以無線通訊、電腦、自動控制為基礎之移動自動閉塞系統(MAS)。MAS 捨棄傳統固定之閉塞分區概念，以前方列車尾端、進/出站信號機為煞車停車目標或後行列車之追蹤目標，確保列車間保持一定的安全間距及煞車充裕距離。

以下就各國之移動閉塞系統分章討論，並於後章節統整比較，藉以指出我國智慧型軌道運輸未來發展之方向。

2.3.1 北美、加拿大 ATCS

美國與加拿大於 1982 年提出先進列車控制系統(Advanced Train Control System ,ATCS) 和 先 進 鐵 路 電 子 系 統 (Advanced Railway Electric System ,ARES)，後由北美鐵路協會共同開發，基本構想始於 1983 年在加拿大蒙特婁(Montreal)舉行之會議，與會者有加拿大和美國等北美鐵路官員。由於列車控制系統受限於技術運用及發展，因此採用微處理器裝設於鐵路移動設備上，並通過數據無線電系統與中央控制系統聯結，組成針對傳統安全性及效率作提昇之先進列車控制系統。

ATCS 分為四種不同功能級別：10 級、20 級、30 級、40 級，根據不同之路線狀況、列車條件來選用。

ATCS 之系統功能如下：

- 確認：列車位置及其他移動體。
- 定位：列車、路線單元和設備維修隊。
- 檢測：列車狀態、線路故障、超速、道岔、閉鎖。
- 監視：列車速度、加減速、制動、平交道。
- 控制：道岔、號誌、機車速度。
- 通信：數據傳輸。
- 訊息處理：自動控制、(非)安全邏輯決策訊息。

ATCS 主要組成為：中央調度室、無線電數據傳輸系統、列車及其他移動設備之車載系統、固定之軌旁設施四部分。無線電數據傳輸系統是 ATCS 的核心，負責將列車與其他移動設備上之車載系統、軌旁固定設施、中央調度控制室三方面串聯起來，因此，無線電數據傳輸系統之語音及數據基地台的設置十分重要。固定的軌旁設施包括電動道岔、道岔融雪器、軌道應答器等等，與中央調度控制室聯結，負責列車位置確認，以及作為 ATCS 預報系統之基礎。利用列車車載電腦及其他移動設備上之車載系統，計算處理各種即時訊息，搭配路線特徵、列車編組、前後車性能參數、目標停靠站特徵……等訊息，在必要時發出警告或採取制動，以維護列車行車安全。

北美鐵路 ATCS 試驗及發展整理如表 2.6：

表 2.6 北美鐵路 ATCS 系統試驗及發展一覽表

試驗名稱	試驗內容及特點
BN 柏林頓北方鐵路	<ul style="list-style-type: none">➤ ARES 之試驗已中止➤ 利用超高頻通訊結構的優點控制各種非安全應用(電碼線路通訊.列車管理功能)

UP 美國 聯合太平洋鐵路	<ul style="list-style-type: none"> ➤ 設有列車乘務工作報告系統 ➤ 取消電碼線路,利用無線根據檢測器進行區間報告
CP 加拿大太平洋鐵路	<ul style="list-style-type: none"> ➤ 利用大量的 ATCS 無線數據傳輸設備取代電碼線路
CN 加拿大國家鐵路	<ul style="list-style-type: none"> ➤ 曾裝設過原始的 ATCS 試驗,現已終止 ➤ 利用和現有計算機人工閉塞相連接的非安全計算機 ➤ 車載 OBU 裝有安全計算機 ➤ 道岔處也採用安全計算機,用於地面接口裝置(WIU)
CSX 美國切西濱海鐵路	<ul style="list-style-type: none"> ➤ 裝設大量區間 ATCS 數據無線傳輸以取代電碼線路 ➤ 幾乎北美之主要鐵路均裝設 ATCS 數據無線系統 ➤ 目前類似之系統應用於：(SP)美國南太平洋鐵路、Norfolk 鐵路、聖塔菲鐵路 ➤ 聖塔菲鐵路甚至將此一系統功能擴展到實施營運工作報告,列車管理和區間報告等

資料來源：本研究整理

表 2.7 北美鐵路 ATCS 系統試驗實例

試驗地點	計畫名稱	計畫特性
UP 聯合太平洋鐵路 BNSF 聖塔菲鐵路	PTS 可靠列車間隔 (Positive Train Separation)	<ul style="list-style-type: none"> ➤ 涵蓋 UP、BNSF 及兩者共有之 863 英里長的鐵路
Amtrak 底特律-芝加哥大走廊 Kalamazoo-New Buffalo	ITCS 漸進式 列車控制系統 (Incremental Train Control System)	<ul style="list-style-type: none"> ➤ 1996 年十月開始實驗，於 Amtrak 82 英里長之鐵路第一次進行提速試驗(時速 100mph) ➤ 1997 年投入正式營運
Hughes Aircraft 與 Morrison Knudsen 合作開發	AATC 先進自動 列車控制系統 (Advance Automated Train Control)	<ul style="list-style-type: none"> ➤ 於 BART 地區進行試驗 ➤ 得到聯邦技術再投資項目的資助

資料來源：本研究整理

南太平洋芝加哥- 聖路易斯	PTC 可靠列車控制 (Positive Train Control)	<ul style="list-style-type: none"> ➤ 於 UP 太平洋鐵路、SP 南太平洋鐵路進行試驗 ➤ 合併之前之項目,採用 ATCS 概念,位於 Dwight 和 Springfield 之間 110 英里長的鐵路上
------------------	--	---

資料來源：本研究整理

ATCS 之主要目的不在於提高軌道運輸之容量，而在於有效地運用列車及最佳化列車控制以降低生產成本，主要應用於貨運車輛，目前著名的例子有柏林頓北方鐵路，其大多數列車並無固定列車時刻表。



2.3.2 歐洲鐵路運輸管理系統 ERTMS

歐洲鐵路營運管理系統 ERTMS 之核心為 1991 年開始研究之歐洲列車運行控制系統(European Train Control System ,ETCS)和超速防護系統，利用提高列車運行速度和縮短列車間隔來提高運輸效率，並且取消地面信號系統及軌道區段空閑檢查設備及與其相應之電纜以節省路線基礎設施之建置。主要目的為提高其競爭力。其系統目標為：適應多種制式的運行條件、設備、語言、號誌型式，並利用不同級別的系統滿足不同的列車速度、列車密的之運行需要。

ETCS 系統共有分為三種等級(配置方式)：

第一級：藉由地面應答器將行車命令、列車運行權限傳送給列車，地面應答器與號誌系統聯鎖，在列車通過應答器後，依照所允許的情況下(考慮列車與下一個危險點或應答器之距離所決定出之列車運行圖)，運行至下一個應答器。採用第一級之軌道配置方式可於列車信號機前的接近區段內再加安裝一前置應答器，或使用一個半連續傳輸訊息的設備(如歐洲環線 Euroloop)，藉以提高軌道容量。

第二級：建立在第一級系統的基礎上，更增加利用無線傳輸達到對於列車下達行車命令之功能。列車行車命令從無線閉塞中心(Radio Blocked Center ,RBS)傳達列車運行權限予列車，取消地面信號機，而軌道閉塞分區之空閑/佔用情形則由地面設備檢查。

第三級：融合第二級之基礎，增加對於列車完整性之監視，以確立列車是否完整，並且可掌握後行區間之空閑情形，於第三級中，才真正達成移動閉塞系統之實現。

ETCS 主要系統組成及功用為：

- (1) 地面系統(Euro-balise)：向列車傳輸訊息的信標設施。
- (2) 車上系統(Euro-cab)：處理 Euro-balise 和 Euro-radio 傳來之訊息，以及車載子系統傳來之數據(ex：測速計、里程表)。
- (3) 無線通訊系統(Euro-radio)：GSM-R、無線數據通訊。

以上列車各級系統之示意如下圖所示：

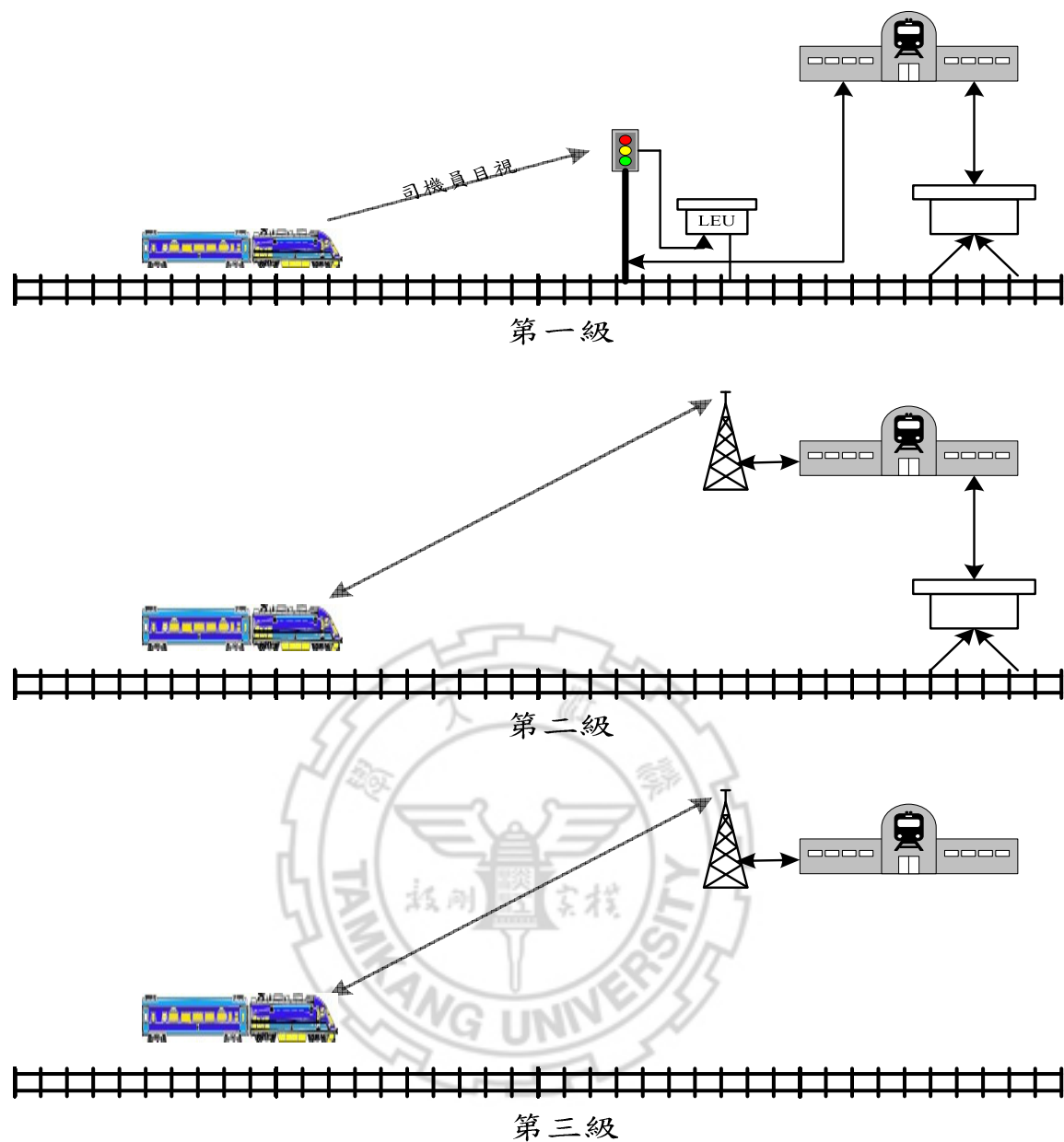


圖 2.11 ETCS 各級系統示意圖

表 2.8 歐洲 ETCS 無線運用模式

模式	適用範圍	模式運作原理
模式 A	低密度運行路線	列車於需要時向 RBC 申請運行權限
模式 B	低密度/車站調車	由 RBC 不斷發送一個包含一系列短的列車識別號和相應的運行權限,只有符合其識別號碼之列車才有可執行相應的列車運行權限
模式 C	中密度運行路線	由 RBC 向有關的列車發送運行權限
模式 D1/D2	高密度運行路線 (可實現移動閉塞之功能)	RBC 每隔幾秒即向列車發送一個有效持續時間較短的運行權限

資料來源：本研究整理

2.3.3 法國 ASTREE 系統

法國國鐵於 1986 年開始研發即時追蹤自動化系統(Automatisation du Suivi en Temps Reel , ASTREE)，採用無線技術之列車控制系統，連續偵測列車之速度與位置，利用列車無線系統(450Hz)傳送與各列車與地面設施間，其系統特點為：列車自動定位、列車安全間隔控制、監視列車運行和路線狀況、列車進路控制、列車運行調整、列車時刻表管理，及列車完整性檢測；以數據來管理列車運行而非依靠經驗，集中列車控制，並且盡量在移動設備上進行數據改造或增加設備，對固定的地面設備盡量減少更動以及裝設。

ASTREE 系統採用都普勒雷達檢測列車之位置與速度，並經由車載電腦計算，搭配軌道應答器作列車絕對位置之修正，以獲得連續之列車位置與速度。採用監視道岔和檢查聯鎖的方式控制進路設置聯鎖設備和 ASTREE 集中聯鎖，並且使用分布式數據庫以保持準確的最新路線及列車狀態數據，包括：列車動態參數(位置、速度、加速度等)、靜態參數(列車長度、重量、制動性能等)，管理信息(列車時刻表、商用訊息等)，列車完整性檢測，相關 ASTREE 系統之試驗與發展如表 2.9 所示：

表 2.9 ASTREE 系統試驗及發展

發展時間	試驗路段	計畫名稱	試驗結果
1990 1993	Bondy-Aulnay (10 km 路線)	➤ 系統可行性示範試驗	➤ 證明系統結構的實用性和技術措施的可行性
1994 	Paris-Est (80 輛機車和兩個設在巴黎 Est 湖 Gargan 的 ASTREE 營運中心<AOC-ASTREE Operations Center>)	➤ 系統的冗餘性 ➤ 兩個 AOC 的交接過程	➤ 確定計算機網路所需之規模 ➤ 測定系統的影響時間 ➤ 試驗當一個 AOC 失效或故障時，系統的重構過程
	Bondy-Aulnay	➤ 系統安全性評估試驗 ➤ 移動閉塞設備	

資料來源：本研究整理

ASTREE 系統之閉塞方法採用相對速度之方式，僅考慮列車將相對速

度，而不考慮列車之相互位置，考慮列車性能、路線環境參數.....等因素，由前後車之速度計算出允許速度。由車載電腦進行列車間隔之計算與控制，在列車前後方皆設防護區間，前方之防護區間長度根據列車相對速度改變，而列車後方之防護區間則是用以向後行追蹤列車開放進路，以達到列車運行之安全，期望提高路線容量、節省能源、增強運輸組織靈活性、提高生產率。



2.3.4 日本新幹線 CARAT 系統

日本於 1985 年提出電腦及無線支援列車控制系統(Computer and Radio Aided Train Control System ,CARAT)，主要目的為提高軌道容量，滿足列車高速、高密度之行駛需求及彈性的適應變化，改善鐵路營運之靈活性。目前處於最終性能評估及實用化評估階段，主要之工作仍是集中於列車-地面之控制。

CARAT 系統採用相對位置方式，透過對前後列車之相對位置關係，得出後行追蹤列車最大運行速度。CARAT 系統對於列車間之追蹤分為站間列車追蹤距離及站內追蹤距離兩種控制方式。站間追蹤間隔是以前行列車車尾作為停車目標，而站內追蹤間隔則是以軌道進路終端為列車停車目標點，如圖 2.13 所示：

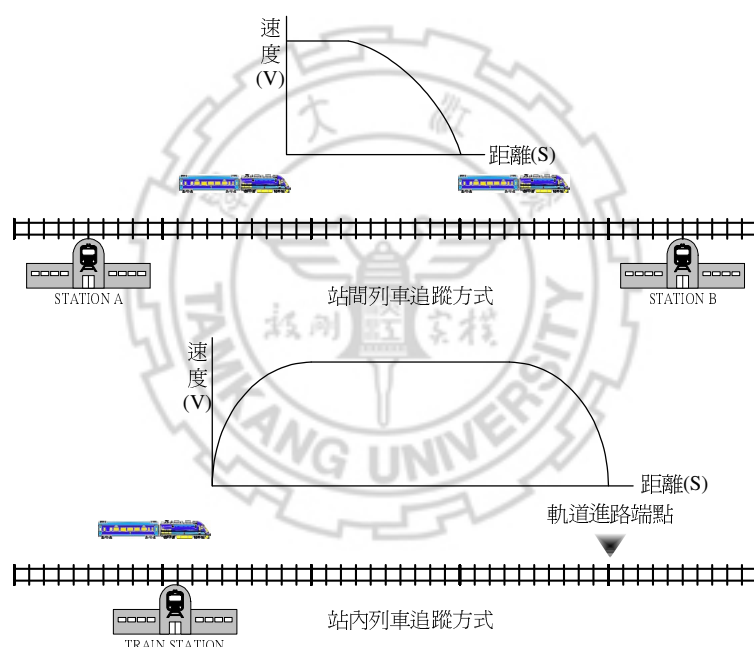


圖 2.12 CARAT 系統列車追蹤運行示意圖

CARAT 系統利用車載智慧型位置檢測系統，確認列車所在位置。最主要之系統組成有以下兩部份：地面感應器及車載計軸器。地面感應器透過感應線圈或查詢應答器，並且由車載感應裝置讀取列車之絕對位置；至於應答器佈設點之區間列車位置的確認，則透過車載計軸器計算列車車輪轉數，以獲取距離，並且經由每次通過應答器所獲得之絕對位置，兩相加總計算以獲得列車實際之位置。由於轉彎時或者是軌道扭曲之緣故，會有造成鋼輪打滑或者是空轉的情況發生，對於此一狀況之處理，採取同時對多個車輪檢測其轉數，並且用軟體修正，以達到精確定位的目的。

CARAT 之主要系統功能包括：管理各站間之訊息交換、列車間隔控制、道口控制與列車接近警報、車站綜合安全控制、列車進路控制，如表 2.10。

表 2.10 CARAT 系統試驗與發展

試驗時間	試驗地點	試驗結果	備註
1989 1990	新-岩國 (Shin-iwakuni)-德 山(Toluyoma)	<ul style="list-style-type: none"> ➤ 驗證無線傳輸系統的基本性能 ➤ 列車定位功能 ➤ 地面控制邏輯 	
1991 1990	上越新幹線新瀉 (Niigata)-長岡 (Nagaoka)	<ul style="list-style-type: none"> ➤ 無線通信的傳輸質量 ➤ 列車定位精度 ➤ 列車跟蹤和控制算法 ➤ 故障安全微處理器 	皆達到 實用化 之要求
1995 	安裝於上越新幹線 上進行實驗	<ul style="list-style-type: none"> ➤ 摒棄傳統一條進路只包含一個閉塞分區的概念 ➤ 使用新的鏈鎖表和鏈鎖裝置,在不降低列車運行安全度的前提下提高進路裝置的靈活性 ➤ 允許多列列車同時佔用一條進路 ➤ 大幅縮短列車在車站的追蹤間隔 	

資料來源：本研究整理

2.3.5 德國 LZB、FZB 系統

1972 年原聯邦德國首次在慕尼黑地鐵採用連續無級列車控制系統 LZB(Linien Zug Beeinflussung)，也因此引發後續學者研究鐵路智慧化控制與管理，例如：北美鐵路的先進列車控制系統(Advanced Train Control System ,ATCS)和先進鐵路電子系統(ARES)、法國鐵路的連續即時追蹤自動化系統(Automatisation du Suivi en Temps Reel ,ASTREE)、德國的電腦輔助綜合鐵路系統(CIR)、歐洲鐵路的列車控制系統(European Train Control System ,ETCS)、澳大利亞鐵路的先進列車控制系統、日本鐵路的電腦及無線電輔助列車控制系統(Computer and Radio Aided Train Control Sustem , CARATS)、丹麥鐵路的列車智囊(Train Brain)、荷蘭鐵路的 21 世紀鐵路(Rail 21)等。

近幾年德國西門子公司在現存之 LZB(採用軌間電纜作為訊息傳輸途徑)之基礎上，構建出 FZB 系統。

LZB 為連續式列車運行控制系統，德國於 60 年代開始研究，最高之目標行車速度為 200 km/h。發展至 70 年代後期，以達到能夠反映地面信號之顯示，並且自動控制列車的牽引和制動。相關之試驗及研究有「漢諾威-維爾次堡」之高速鐵路及「曼海姆-斯圖加特」之高速鐵路。FZB 植基於原有之 LZB 系統，在 1993 年由西門子公司推出，利用鐵路原有的無線通信站(GSM)建立車上、地面無線通信聯絡，取代 LZB 之軌間電纜。

LZB 相較於 FZB 系統，利用鐵路沿線既存之無線移動通訊網路，建立車-地通訊。FZB 無固定之閉塞分區長度，因此可適用於各等級各類型之列車於同一軌道上混合運行，皆可達到提高運行速度及提昇軌道容量之目的。兩者之比較結果可由表 2.11 看出。

表 2.11 LZB 與 FZB 系統特性比較表

	LZB	FZB
通訊方式	軌間電纜	無線通信站(GSM)
系統特性	<ul style="list-style-type: none">➤ 傳輸頻率較低➤ 信息傳輸量大➤ 數字系統抗干擾能力強➤ 適合各種速度及性能相異之列車混和運行	<ul style="list-style-type: none">➤ 傳輸頻率較高➤ 訊息傳輸量更大➤ 抗干擾能力更強➤ 適合各種速度及性能相異之列車混和運行
實際案例	<ul style="list-style-type: none">➤ 「漢諾威-維爾次堡」之高速鐵路➤ 「曼海姆-斯圖加特」之高速鐵路	<ul style="list-style-type: none">➤ 柏林-萊比錫

資料來源：本研究整理

2.3.6 英國地鐵 Jubilee 朱比利線 WESTWACE 列車自動保護系統

英國於 1923 年首次使用無線傳輸技術進行對運行列車之試驗，利用點對點無線傳輸技術取代舊有之電報傳輸技術，產生以無線傳輸達成之 ATP 系統，也是首次將無線電廣播通訊技術運用於安全調度集中的無線電子路籤(Token)閉塞(Radio Electric Token Block)中。

倫敦捷運系統主要由十三條新舊捷運路線交織而成，從最古老的 Central 線到最新的 Jubilee 線，由十二條路線及 Docklands 輕軌系統組成，每日載客量達 250 萬人次，共有近 500 輛列車，超過 260 座車站，員工數目約 16,000 人。

對於具備多種不同速限、列車間隔小的複雜路線中，正確地將訊息傳送至接收端將直接與軌道列車安全有關。由於是以無線傳輸的方式將訊息通過鋼軌上的代碼或應答器，將數據發送至列車上，但這些頻率卻會有被其他系統干擾之慮，因此完整的訊息傳輸是一大考驗；英國地鐵採用減少軌旁設備的建置，並將焦點轉為如何提高列車運輸能力。其中又以採取移動閉塞系統之朱比利線(Jubilee)為代表。

Jubilee 線於 1979 年通車，設計運量單向每小時 3 萬人次，從 Green Park 始，經 Westminster、Waterloo、Southwark、London Bridge、Bermondsey、Canada Water、Canary Wharf、North Greenwich、Canning Town、West Ham 到 Stratford。Jubilee 地鐵線十分複雜，主要分為兩部分：固有老線之繼電器電氣連鎖的常規訊號系統、延伸至倫敦東部之新線。老線升級主要包括 ATP/ATO 系統設備在軌道上之覆蓋，並保留原有之軌道電路、聯鎖系統及轉轍器。

Jubilee 採用西屋列車高級無線控制設備(WESTRACE)作為其安全處理器，採用無線通訊作為通訊及聯鎖之手段，利用電腦使得系統具備綜合診斷之功能並且提供即時之維修技術服務，列車利用車載設備，透過與速度探測相關之絕對位置參考確認系統(APR)達成對自身位置之確認。APR 系統包括車載應答器及軌旁之發射應答器，主要功能即為提供列車之確切位置予車載 ATP 控制器。由車載測速電機和都普勒雷達確知列車速度，並且經由漏波電纜連續傳送列車更新位置予沿線最近之移動閉塞處理器(MBP)，並由此移動閉塞處理器判斷列車之運行權限，車載 ATP 系統利用列車制動性能和列車運行圖及軌道路線特性，計算出列車之制動曲線並獲得「安全保護包絡線」。WESTRACE 乃採用以傳輸為基礎之訊號系統(TBS)，其工作流程如下：

(1)MBP 將 LMA 傳送予列車

(2)車載計算機計算出列車防護制動煞車曲線，並發送目標速度指令，
規範列車行駛狀況

(3)當列車逐漸駛近允許運行權限之端點，其目標速度會根據車載計算機之指令不斷向下修正，以保持列車在安全保護包絡線內行駛。

WESTRACE 每個處理器的工作至少由其他兩個處理器檢查，相互監測運行狀況是否正常，採用多樣性、差異性而非傳統利用冗餘性來達到安全的目的，並且有效降低成本。主要系統組成包括車載系統及地面系統：(1)移動閉塞處理器，安裝於地面，直接與軌道訊號系統聯鎖(2)無線通訊系統(3)ATP 車載控制器。朱比特線 ATC 系統符合 Cenelec 標準 prEN50126、prEN50128、prEN50129，包括了以下之安全保護程序：(1)危險預知分析(2)子系統危險分析(3)系統危險分析、風險定量分析(4)運行中危險分析；並且應用於其延長線上。

為了提高軌道容量，達到最小列車行車間隔，使用 WESTRACE 系統，於軌旁 ATP 系統設有一移動閉塞處理器(MBP)，用以決定每一列車之運行權限(Limit of Movement Authority, LMA)，經由 MBP 動態計算出各列車之 LMA 及所相關聯的聯鎖系統、道岔、進路、軌道電路佔用情況，配合列車運行狀況，透過漏波電纜將 LMA 傳送予列車車載之 ATP 設備；車載 ATP 裝置再根據列車制動性能及由 MBP 發送之 LMA 計算出列車制動曲線，並依此運行，此即安全保護包絡線(見圖 2.14)。

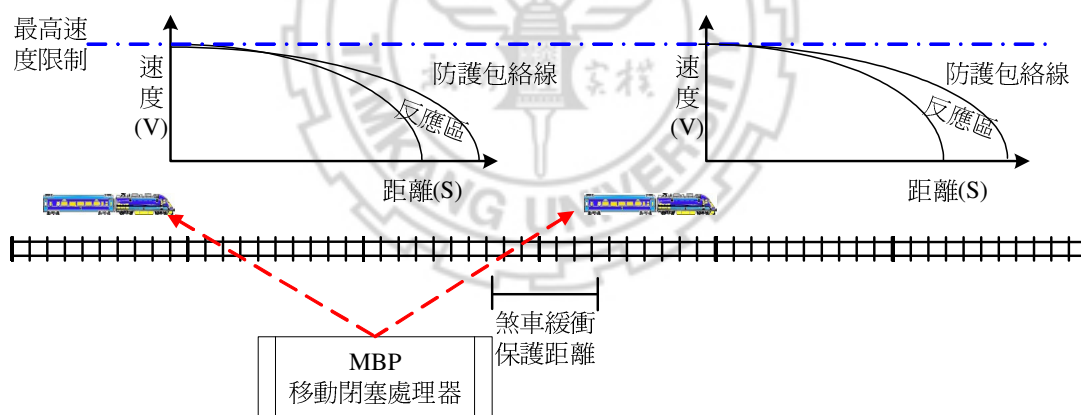


圖 2.13 WESTRACE 移動自動閉塞系統工作原理

WESTRACE 於車站和軌道接安裝 ATO 設備，提供 ATO 車載設備訊息，包括列車位置及列車行駛模式，與列車牽引設備及制動設備聯鎖，並且利用車輪轉速表計算列車位置；搭配 ATO 系統與每個車站的兩個地面 ATO 環路重新校正行駛距離，可達 $\pm 0.5m$ 的停車精度。

利用 TBS 系統使朱比利線能夠每小時每方向運行 36 輛列車，相當於是老線之 150% 的軌道運輸容量。

其他地方使用 WESTRACE 作為訊號系統之聯鎖的例子有：馬德里、巴塞隆納、里斯本。WESTRACE 系統可單獨使用，或與 ATP 系統同時使用。

2.3.7 小結

為確保列車行車安全，因此控制列車安全間隔之各種閉塞行車制有其發展必要，隨著科技進步，閉塞系統之發展進程如表 2.12 所示。由表 2.12 可看出，其發展趨勢為在安全前提下，達到提高運輸能量、降低運輸成本之目的。

表 2.12 軌道運輸閉塞方式演進歷程

時 間	項 目	功 用
西元 1872 年	軌道電路 (固定自動閉塞系統)	利用軌道電路防止車輛碰撞，一段閉塞區間使允許一輛列車佔用，藉以實現安全保障以及提高列車運行效率。
西元 1890 年	半自動閉塞系統	介於固定自動閉塞系統及移動自動閉塞系統之間，藉由細分軌道電路、音頻數字軌道電路信號系統，縮小閉塞區間，以提昇軌道容量，實現列車自動防護及自動駕駛功能。
西元 1910 年	移動自動閉塞系統	無固定閉塞區間，不需列車信號機與軌道電路連鎖作為閉塞區間之劃分，可充分利用軌道資源，提高軌道容量，即時反映需求靈活調度列車，在安全之前提下，達到最有效率之行車方式。
西元 1925 年	聯鎖系統 (Centralized Traffic Control, CTC)	將「號誌連鎖」與「閉塞區間」搭配，並由控制中心統一指揮控制，稱為 CTC。

資料來源：本研究整理

各國之傳統列車係利用列車運行曲線控制方式，在符合列車制動過程下求得最佳控制方式，進一步達到縮短列車運行間隔距離。其列車運行曲線在不同閉塞方式下會有不同之距離-速度控制曲線。在移動閉塞情況下，控制的目標煞車距離為前行列車車尾所在位置，因此此種情況下能獲得列車最小運行間隔，以提昇軌道運輸容量，各種閉塞行車制之目標煞車距離如表 2.13 所示：

表 2.13 閉塞方式對列車容量之影響統整表

閉塞方式	列車位置監控	目標煞車位置	軌道容量
固定自動閉塞	僅能確認列車所在之區間，無法精確得到列車位置	前行列車所在閉塞區間+兩個閉塞區間長度	小
準移動閉塞	區間長度較短，對列車位置掌握較佳，但仍無法確認列車即時位置	前行列車所在區間搭配數個閉塞區間	中
移動自動閉塞	藉由無線訊息傳輸列車即時位置，可對列車進行精確定位	前行列車尾端+煞車充裕距離	大

資料來源：[本研究整理]

目前由於「移動自動閉塞行車制」發展未臻完善，各國軌道運輸客運大多仍採行「固定自動閉塞行車制」進行客運服務，但各國莫不摩拳擦掌進行實驗，以期能提升軌道運輸能量並更有效率地調度及利用，相關固定自動閉塞系統及移動自動閉塞系統之比較如表 2.14 所示。

表 2.14 固定自動閉塞區間與移動自動閉塞區間系統優劣比較表

項目	固定自動閉塞系統 FAS	移動自動閉塞系統 MAS	結論
閉塞分區	固定	隨機	MAS 彈性較大
軌側設施	軌側、進出站信號機	進出站信號機	MAS 成本較低
運行模式	僅適用於同等級列車運行	允許不同等級列車混跑	MAS 靈活性高
制動方式	分區制動，每次制動都有空走時間及冗餘距離，且隨時間距離累積增加	隨機產生之閉塞分區，採用速度聯鎖消除冗餘距離	MAS 行車密度與軌道容量皆提昇
舒適度	減速度過大，制動不穩	採用速度聯鎖控制，減速度較小	MAS 舒適度較高

資料來源：本研究整理

由文獻回顧介紹各國移動自動閉塞系統之特性，影響其最根本的莫過於對於列車定位以及通訊方式，整理如表 2.15：

表 2.15 各國移動自動閉塞系統之比較

各國移動自動閉塞	列車定位及測速	訊息傳輸方式
北美、加拿大 ATCS	軌道應答器、無線電數據傳輸系統	900MHz 實現車-地訊息傳輸
歐洲 ERTMS	軌道應答器、無線傳輸系統	GSM-R、無線傳輸系統
法國 ASTREE	都普勒雷達、車載電腦、應答器	450MHz 之空間波
日本新幹線 CARAT	感應線圈、應答器、車載計軸器、電腦軟體計算修正	400MHz 之空間波和 LCX，未來擬採用準微波或毫米波
德國 LZB、FZB	軌間電纜、應答器	軌間電纜、GSM

資料來源：本研究整理

其他相關之列車定位方式有由舊金山港灣高速鐵路(BART)及休斯航空公司共同開發之 AATC(Advanced Automatic Train Control)系統，利用美國軍用高精度定位系統，藉由測量列車無線電站台與地面無線電站台之間無線電波傳播時間完成對列車之定位；由美國 Harmon 工業公司開發之 ITCS(Incremental Train Control System)系統，藉由衛星實現列車之定位，針對 ATCS 系統無法充分完成列車控制之功能加以補充。

由上表可知，先進之列車定位技術與通訊技術皆採用多種技術混和使用，以達成無間斷之列車定位，由於列車速度快、質量大，因此制動反應時間必須較長，唯有針對列車進行即時且不中斷的位置監測，才能即時發現車輛運行之異常狀況，及早反應，以達安全之要求。因此未來我國台鐵未來發展勢必朝向此一趨勢，借鏡國外軌道運輸先進國家之成功先例，達到提高安全、降低成本、以及靈活調度的目的。

2.4 各國軌道容量解析模式

軌道容量探討之方法論上，通常會以解析模式做為基礎，因其表現簡單易懂，所以在各國皆普遍使用；但也因此，若是未將考慮因素適當納入公式中，則容易產生誤差，因此解析模式大多做為初步參考的依據，若是要求得進一步精確的軌道容量，則有賴模擬模式方能達成，惟需注意各項影響因素如何納入模式中，以反映各不同運轉條件、不同情境下對於軌道容量結果之影響。

欲建立軌道模擬模式，須以解析模式為基礎，因此以下針對各主要軌道大國解析模式回顧，綜合比較如表 2.16，以期未來模式構建能針對本研究得出適用於台鐵系統容量分析及模擬程式之模式。

表 2.16 各國軌道容量解析模式比較表

國別		軌道容量解析模式	模式特性	未來 可改進方向
台灣鐵路		$C_l = \frac{1440}{\frac{t_U + t_D}{2} + t_s} \times \delta \times \eta \times n_t$	考慮辦理閉塞及號誌的時間以及行車制度對軌道容量之影響	以上下行列車運轉平均時分計算無法求得實際軌道容量以單線運轉觀點無法表現實際營運狀況
美國		$C_l = \frac{3600}{(t_s + t_d + t_m)}$	對號誌時距詳細計算，可反應列車性能、路線條件、號誌系統對號誌時距的影響	僅適用於副線運轉且車輛性能一致之捷運系統
歐洲	UIC Code 405	$C_l = \frac{T}{(t_s + t_m + t_z)}$	考慮車種組成、運轉寬裕時間	號誌時距之計算未詳加說明
	IMPROVERAIL	$C_l = \frac{T}{h} \delta$	考慮前後車速度、列車方向分布、交通組成	軌道配置、號誌系統、車輛性能反應不明顯

日本	複線區間	$C_l = \frac{1440}{h_1 r_h + (h_2 + h_3 + 1) r_l} \times \delta$	考慮不同速度列車進出站之運轉時隔，以加權平均的方式來計算	僅將列車分為高速及低速，並未考慮多車種混跑
	單線區間	$C_l = \frac{1440}{t + t_s} \times \delta$	以列車於站間之平均運轉時分計算容量	可能高估軌道容量
	通勤電車專用區間	$C_l = \frac{1440}{h} \times \delta$	單一軌道單一車種，車輛無交會、待避	未考慮到各車站之上下車、停等
中國		$C_l = \frac{1440 - t_n - t_c}{t_U + t_D + \tau_A + \tau_B}$	考慮區間兩端車站之運轉時隔	計算細節未詳加說明

資料來源：本研究整理,相關變數意義說明列於附錄一

2.4.1 小結

本研究著重於不同行車制度下，軌道容量改變量之探討，因此需構建出一體適用於不同行車制度的容量解析模式，方能在同一標準上，衡量軌道容量之變化。因此，未來模擬模式構建擬將號誌系統拔除，引入列車安全間隔概念代替號誌系統維持列車間距的功能，但由於進出站號誌機為各種行車制度均不可或缺，因此模式中也必須考慮進出站號誌時距，以納入容量計算。經由文獻回顧可知，欲改進台鐵經驗容量公式必須將各車種組成納入考慮，並用加權平均方式得出較為精確之軌道容量。

解析模式使用方便易懂，為一般常用之軌道容量評估方法，但也由於其模式簡單，使得降低其精確度，且無法產生類似模擬模式之結果，將可能會發生以及必然會發生甚或是罕見之列車運轉情形、事件表現出來，因此未來在軌道容量之研究方法仍傾向採用以解析模式為基礎之模擬模式，以避免僅採用解析模式無法考慮列車運轉速度之變異、列車間之干擾、號誌顯示之影響等隨機效應之問題。

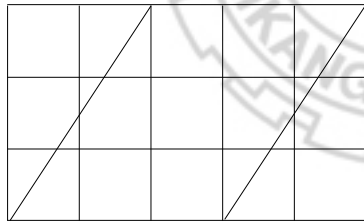
第三章 研究方法

3.1 軌道容量分析方法

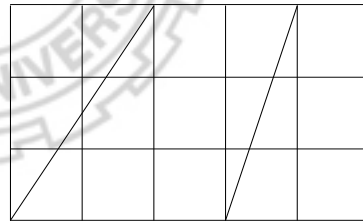
軌道容量最早由俄國工程師別爾霍夫斯基於 1878 年開始研究，一般分為列車通過能力及列車輸送能力兩種。

- (1) 列車通過能力係指在採用一定類型之列車及某一列車組成採取某種運行方式下，受限於鐵路各種固定設施，在單位時間內(通常指一日 24 小時)所能通過最大列車數或最大車廂數。
- (2) 列車輸送能力係指在一定的固定設備、列車車輛類型、列車編組及運行方式下，受限於機車車輛及人力配置，在單位時間內(通常指一年)所能運送最多貨物噸數(通常以萬噸為單位)。

本研究著重於列車通過能力之軌道容量計算方式，一般來說，固定自動閉塞系統之軌道容量可藉助列車運行圖來計算，而列車運行圖又分為平行運行圖和非平行運行圖(如下圖所示)。平行運行圖之軌道容量計算方式乃透過分析平行運行圖直接可計算出區間通過列車數，再配合列車編組、設計車廂容量等求取區間軌道容量；而非平行運行圖則是借助扣除係數在平行列車運行圖的容量計算基礎上，當列車數量、列車運行圖類型...等值為一定時，求得其軌道容量，其結果也相當精確。



平行列車運行圖



非平行列車運行圖

會有平行列車運行線與非平行列車運行線的不同結果，乃是因為研究的對象是否有多車種混跑的情形，本研究以台鐵為研究對象，需將多車種混跑納入考慮，但由於無法掌握精確扣除係數係數值，使得利用運行圖做圖求取台鐵軌道容量之方式較為不可行，更遑論未來情境分析求取採用移動自動閉塞行車制下之軌道容量。

相較於固定自動閉塞區間，移動自動閉塞區間的容量分析無法藉由傳統方式求解，因此在 FAS 條件下賴以計算軌道容量之列車運行圖運用於 MAS 將無法適用，無法表達由於列車追蹤方式之改變、列車條件、路線條件、進出站方式等因素之改變；軌道容量並非一定值，而是隨當時條件實際情況變化，並且由於 MAS 乃是基於無線通訊之基礎上與列車溝通，使得列車運行

計劃可靈活調整，因此未來軌道容量計算之模式構建需能充分表現各影響因素，方能完整且確實地反映其結果差異。

軌道容量之計算方式主要分為三種：解析模式、最佳化模式、模擬模式。根據列車運行計畫之相關資料：包括站間距離、停靠站時間、軌道佈設方式、列車班距、車站月台數、衝突點位置、列車折返點...等考量因素，計算出理論之最大容量，藉以評估軌道運輸之輸送能力。



3.1.1 解析模式

藉由鐵路設施與列車運行圖的特性，在某些假設或限制情況下，求取軌道路線容量之方法。解析模式使用簡單，隱含機率及穩定的概念，為求取軌道容量常用之方法，可得出理論上的較佳解。解析模式之優點為簡單易使用，因此常用作為軌道容量分析之工具。

由於解析模式通常有數學公式，可用此來反應解釋變數對被解釋變數之影響，但相對地若未能將重要性較大之影響因子納入公式，則產出結果將受質疑，此為利用解析模式於軌道容量計算上為人詬病之處，也是解析模式無法與模擬模式相匹敵的地方，並且解析模式未考慮隨機變數，因此解析模式多用於一般性公式推演以及結果概估。唯有架構完整定義清楚之模式(well-defined)，儘可能將重要之影響因素納入模式構建，去除共線性問題，並進行模式驗證，方可接受其結果。因解析模式對於其中參數變動非常敏感，稍有變動將大大影響其結果的精確度，針對各參數進行敏感度分析，得出一因地適宜的模式，以期能真實地反應實際情形。

解析模式在計算複雜的路網時，常將軌道系統視為許多節點與節線之組合，依各單元元素之不同特性計算容量。目前台灣鐵路管理局計算單線區間的傳統計算公式[謝文隆,1996]，日本山岸氏概算式與運轉局簡易式[日本國有鐵道]中介紹之計算式皆為著名分析路段容量之解析模式。

3.1.2 最佳化模式

軌道運輸之容量分析通常運用最佳化之時機在於求解最大排班班次、最大司機員排班模式，以最佳化之概念，在給定某些限制條件下，求解最佳值或求解出可允許最多班次等的方法，常見的研究方法有：線性規劃中之 0-1 規劃、動態規劃、整數規劃、分枝限定法、0-1 規劃搭配遺傳演算法或類神經演算法...等等，以效益最大化或成本最小化為目標函數，進行最佳化求解。

最佳化模式在容量分析上之應用常與列車時刻表、列車運行圖、司機員排班等相互配合驗證，尋求在給定之限制條件，例如列車每運轉單位時間需進行檢修之動作、司機員工作時間上限、列車調度廠站限制...等，配合列車運行班表，求取最大之軌道容量，所得之結果為一理論值。運用最佳化的方法於以解析模式產出之基礎班表，並且將之與實際運行之列車時課表相互驗證，以求解出最適班表之表定容量。

國內從事列車排點電腦化的研究最早始於 70 年代，但僅為雛形模式。台鐵因深感排點作業之複雜及人才培育的困難，乃於民國 81 年起，委請資策會進行列車排點電腦化專家統之研究。在其他研究方面，許朝勝等(民 80)

以 Fortran 語言發展 Super501 電腦排班模式；張政源等(民 83)亦提出一電腦排點專家系統的架構。同年，成功大學李治綱(民 83)首度提出列車排點數學規劃(Mathematical Programming)模式以求解最佳化之列車時刻表，其發表之模式係以單線區間為例來模式化問題；大部分之模式求解皆採啟發式(Heuristic)解法，因此不一定會求出最佳解。

國外學者 Jovanovic and Harker(1991)探討單線和複線的列車排班模式；Carey and Lockwood(1994)採用數學模式和演算法來解決此問題，其中 Carey 更將模式延伸運用於不同型態的路線、場站，和月台。

Jovanovic(1989)建構非線性混和整數規劃模式(non-linear mixed integer programming model)用以解決列車衝突問題。

目標函數式：	
➤	列車在停靠站產生延滯時間之成本函數為最小
限制式：	
➤	實際排點限制式-使列車符合預定停靠站排點的要求
➤	旅行時間限制式-各列車在交會區段內(meet point-to-meet point)的行使時間必須相同
➤	連續運行時間限制式-列車之運行必須在時空圖上保持其連續性
➤	跟車(following)與超車(overtaking)限制式-確定兩車間距為最小，且側邊軌道(sidetrack)沒有被佔用，以方便後車可以順利超越前車
➤	會車(meeting)限制式-限制列車僅能在複線區段的交會點上才可會車

資料來源：本研究整理

該模式僅適用於單、複線混和路短，雙單線及車站內部同軌道設置情形則無法加以處理。模式屬於非線性混合整數規劃。

謝汶進(民 83)參考 Jovanovic 之非線性混合整數規劃模式，使用數學規劃法構建可用於排除列車衝突的排點模式。

目標函數式：	
➤	所有列車離開各停靠站產生的總延滯最小
限制式：	
➤	實際排點限制式-使列車符合預定停靠站排點的要求
➤	旅行時間限制式-各列車到達車站的時間需大於前一車站的力開時間加上兩車站間的最小運行時間
➤	最小停等時間限制-各列車在車站的停等時間應小於在停靠站的到達與離開時間
➤	連續運行時間限制-列車之運行必須在時空圖上保持其連續性
➤	跟車與超車限制式-確定兩車間距為最小，且側邊軌道並沒有被佔用，以方便後車可以順利超越前車

- | |
|--------------------------------|
| ➤ 車站軌道佈置限制式-在車站內的列車數不得大於站內的軌道數 |
|--------------------------------|

資料來源：本研究整理

此模式分別就單線、複線、雙單線進行考量，並且將不同的站間路段、站內軌道設置列入限制式中以解決列車衝突。模式中未表現出不同等級列車延滯時間之價值權衡的不同，也針對列車等級設定不同的權重值以符合現況。由於該研究所構建的模式隨列車車種、站內軌道等因素的加入而擴大，因此造成模式過於複雜，在求解時可能產生困難。並且以務實面來說，該模式在某些停靠站會使等級較高之車種停靠較久，與現況不合。

Higgins(1996)構建非線性混合整數規劃模式，針對單線區間，依據列車因交會、待避所造成的延滯，評估列車時刻表之可靠度。

目標函數式：	
➤ 列車總延滯時間和列車營運成本為最小	
限 制 式：	
➤ 後車抵達車站之時間須為前車抵達車站之時間再加上安全距離(safety headway)	
➤ 後車之發車時間須為前車之發車時間再加上安全間距	
➤ 列車須符合預定停靠站排點的要求	
➤ 列車在各區段須符合其速限的規定	

資料來源：本研究整理

一般實際之營運情況，營運者通常是採取停等待避之方式來解決列車之衝突，但伴隨而來將導致列車延滯的情形發生，而增加無謂的營運成本。該模式是在解決時刻表之衝突問題後，藉由計算其延滯和成本，來評估時刻表的可靠度。此研究僅能從時刻表中找出最佳者，而無法自行排定產生最佳的時刻表，且該研究中僅對單線區段進行模式之構建，並未考慮各種不同型態之月台佈設、站內軌道數...等影響因素，因此在實務上之適用性可能較低。

3.1.3 模擬模式

模擬模式為軌道容量分析常用之方法，除了摒除解析模式未將隨機變數納入考慮且須完整考慮各變數之模式之缺點外，並且能反映系統些微差異對於軌道容量之影響。模擬模式能將眾多考慮因素納入，並可模組化各子系統，針對各變數進行敏感度分析，確認主要影響軌道容量之因素。

利用電腦進行多列車之模擬，以找出最佳之列車排點方式。將列車於軌道上之運轉特性完整呈現，需準確掌握車輛、路線、號誌、動力...等系統介面，目前國內外相關之研究大多著重於列車營運計畫及列車模擬，而對於列車排點之模擬大多以區域鐵路為探討對象，以降低其複雜性，因此在實務上仍有可改進的空間。

模擬可分為兩大類：離散模式(Discrete Simulation)與連續模式(Continuous Simulation)。離散模擬又稱為事件導向模擬(Discrete Event-Orientation Simulation)，連續模擬又可稱為時間掃描模擬(Continuous Time Scanning Simulation)，視需要選擇；Leilich[1998]認為事件導向之離散模擬較連續時間導向模擬適用於大量、長距離、長時間之模擬。

本研究之目標在於建構一模擬模式模擬列車之運行狀況，因此擬以連續模擬方式，期望產出為依照所給定之時間間隔，針對研究路段運行各列車在每固定時間間隔傳回受到各影響因素下之列車位置、速度、列車狀態.....等資訊，因此使用連續時間掃描模擬，對各列車進行微觀模擬。

採用模擬模式通常需考慮眾多相關變數及限制條件，但若是考量因素過多，將無法表現個別因素對於問題之影響，若考慮過少，則落入解析模式之缺點，無法真實反應系統狀況。因此通常根據所欲研究之因素深入研究，而將相關性較小之因素簡化處理，使得模擬模式得以描述所欲探討之問題，並且得出符合實際狀況之解答，藉由多次模擬及方案情境分析，比較各種情況下產生之績效作為軌道列車容量探討之產出。由於模擬模式無法得出一最佳解，但可就現實情況產出多種方案之績效供比較，常見之績效指標為路線或路段延滯時間、旅客等待時間、旅行時間、準點性、規律性...等，本研究擬採整體軌道容量(capacity)作為績效衡量之依據，藉由衡量軌道上列車行車間隔，探討在限制條件下，所能達到之最小列車行車間隔，再將此換算為單位時間通過列車數，依照容量之定義，探討通過列車數或將列車數乘以設計容量，即可得軌道運輸之最大容量。表 3.1 為軌道容量模擬模式文獻回顧彙整表。

表 3.1 軌道容量模擬模式文獻回顧一覽表

		線型	單/多列車	考慮車長	變數	號誌	使用程式語言	主要功能概述
國外文獻	美國賓州鐵路局	是	單	否	距離	否	未知	最短時間運轉模式，描述列車運行軌跡。
	法國 RATP	是	多	未知	事件	是	未知	模擬列車運轉情形，找出最小耗能之最佳化設計。
	英國鐵路局	是	單	否	距離	否	未知	找出最小耗能的運轉方式。
	Chang	是	單	未知	時間	否	未知	模擬列車運行
	Hill & Yates	是	多	是	事件	是	未知	評估列車延誤時間對後續列車之影響。
	Mckay, John & Dawson	是	多	是	時間	是	Fortran	將 TPS 加入 ATP 及 ATO 功能，使列車模擬更加符合實際狀況。
	Hill & Bond	是	多	是	事件	是	未知	模擬列車在固定閉塞和移動閉塞下，列車運行狀態分析，並推導出列車最小班距。
	Goodman, Siu & Ho	是	多	是	未知	是	未使用	彙整過去有關列車運行模擬文獻，歸納出模擬方法做整理與比較。

國內文獻	台灣鐵路管理局	是	多	否	事件	否	未知	以區段法之排點邏輯做列車模擬，並設計台鐵排班模擬系統。
	宋豐華論文	是	多	是	速度	是	Matlab & Simulink	發展列車最短運行曲線、列車最經濟運轉曲線及列車運轉動態模擬程式。
	陳智淵論文	是	單	是	時間	是	Borland C++ Builder 5.0	考慮列車行車動力學，加上號誌和閉塞長度等因素，發展出RTTMSS 模擬系統。

資料來源：本研究整理



3.2 小結

影響軌道之因素眾多而複雜，採用解析式方式無法達到全面且精確的分析，無法將影響因素完全納入考慮，並且解析式模式及最佳化模式隱含有系統達到穩定狀態之概念，與先行列車影響後續追蹤列車，無法反應真實情況。

解析模式簡單易用，為一般評估軌道容量最常見使用，但也由於解析模式公式過於簡單，使得其精確性不及模擬模式，且各影響因素皆須完整考慮，因此本研究將不建議使用解析模式作為軌道運輸容量分析之工具。解析模式之評估結果通常僅作為參考的依據，因其不同的計算公式對於同一對象將會產生極大的差異，因此若需精確地評估，仍有賴模擬模式的應用。

解析模式、最佳化模式、以及模擬模式各有其優缺點。解析模式與最佳化模式由於隱含達到系統之穩定狀態，因此較適於相關班表排點問題或時刻表容量問題之求解，且其隱含穩定之系統狀態與現實情況不符，為確保實際營運之安全，致使求得之軌到容量相對較小且與實際狀況較不符合。列車實際運轉時將會有相當多之隨機效應影響列車行車，難以達到真正穩定之狀態，且與公路運輸相同，當服務頻率接近容量上限時，對於列車之干擾或延滯之情況也會迅速增加，因此上述兩種研究方法均不適用於本研究課題，不予以採用，僅擷取其中概念作為模式求解及未來模擬模式構建之參考。

由文獻[1]中所彙整之統整比較可得模擬模式之精確度為最高，且與系統相依程度高，能夠高度反應真實情況且有利於各種情境之分析模擬，擬採用模擬模式，作為本研究未來研究之方法論。

解析模式、最佳化模式，以及模擬模式三者之綜合比較如表 3.2 所示。

表 3.2 軌道容量分析方法方法論比較

	解析模式	最佳化模式	模擬模式
所需資料量	少	中	多
精確度	低	中	高
模式適用範圍	初步規劃	班表設計	檢討、班表驗證
模式構建成本	低	高	很高
運用之便利性	較易使用	普通	普通
系統相依程度	低	中	高

資料來源：文獻[1]

第四章 軌道運輸容量分析之模式構建

經由以上章節之回顧，對於各國閉塞系統以及其相關之技術可得初步概念，接著經由文獻之回顧，採用解析式模型，配合物理之運動定律，構建出相關列車間隔、軌道容量之計算公式，可避免由於需求大於供給或是過度供給而導致浪費軌道容量之情形。就實務面而言，列車於站間運行時，需採用移動自動閉塞系統之閉塞方式，有助於列車準點性及調度靈活性之提升並且可有效地降低軌道運輸之營運成本。本研究將先定義若干基本名詞，避免造成後續討論之誤解，再從各項影響軌道容量之因素深入探討，並綜合各變因而導出一軌道容量及列車運轉時隔之泛用公式。

4.1 軌道容量分析基礎架構及定義

在對於軌道運輸之績效衡量指標中，通常利用軌道容量來衡量。由文獻回顧中可知容量之一般計算式為：

$$C = f_{\max} = \frac{T}{h_{\min}} \quad (4.1)$$

式中 C ：容量(TU/day)

f_{\max} ：最大頻率(TU/day)

T ：時間週期(s)

h_{\min} ：最小時間間隔(s)

由文獻[1]及一般容量計算公式可歸納出定義容量之四個基本要素：運轉條件、時間單位、空間參考點，及客體單位，如表 4.1 所示。

表 4.1 定義軌道容量之基本要素及其分類

運轉條件	時間單位	空間參考點	客體單位
路線條件	時	路段	乘客
交通條件	日	路線	乘位
控制條件	年	車站	車輛(廂)
		折返點	列車
		銜接點	貨物
			噸數

資料來源：文獻[1]

(1) 運轉條件

軌道系統之運轉條件可分為「路線條件」、「交通條件」、「控制條件」三種，不同之運轉條件將對軌道容量發揮莫大之影響。

(2) 時間單位

軌道系統之時間單位不一，以容量來看，若是針對都會捷運系統，由於其班距短，旅次時間短，且具有明顯之尖離峰特性，因此大多以小時最為容量分析之時間單位，相較於傳統之中長途城際鐵路，由於旅運時間較長，且並無明顯之尖離峰特性，因此多以日為單位，在實際計算上，可利用扣除非營運時間或以乘上路線利用率來表示，以求得較正確之軌道容量。

(3) 空間參考點

軌道運輸系統之空間參考點包括路段(way)、路線(line)，及車站(station)。路段代表車站之間之路線區間，不考慮車站之影響；路線代表整條運轉路線，需將車站之影響納入考慮，一條路線上含括多個路段；除了上述主要之空間參考點外，若欲詳細計算，則將路線銜接點(junction)及折返點(turn-back)一併納入分析。

(4) 客體單位

客體之研究對象可為乘客(passengers)、乘位(passenger spaces，包括座位及站位)、車輛(廂)、列車、貨物及噸數。對於都會捷運系統，習慣上以乘客或乘位做為客體單位，因其主要之營運項目乃針對客運，而傳統鐵路系統由於兼營客、貨運，因此通常採用列車數(TU)做為客體單位，以避免僅採用乘客數或噸數無法真實反應實際之營運情形。

4.1.1 名詞定義

容量之一般定義為：「在特定的情況下，單位時間內能夠通過運輸路線上一固定點的最大運輸單位數，此處所指的運輸單位(Transit Unit)可以是汽車、公車、列車、飛行器等各種不同運輸工具，運輸路線即所對應的道路、軌道、或航道。」，本研究依此定義，定義容量為「單位時間內能夠通過運輸路線之最大列車數」。根據以上定義，依不同計算對象與考慮因素的不同尚可分為：

(1)路線容量(Line capacity)：為運輸單位通過固定點的最大頻率 f_{\max} ，頻率為班距的倒數，通常班距的定義分為路線間距 h_w (way headway)與車站班距 f_s (station headway)。路線班距表示沒有車站時的班距，後者有將車站納入考慮，通常 $f_{s\min} > h_{w\min}$ ，因此路線容量公式為：

$$f_{\max} = \frac{3600}{h_{\min}} = \frac{3600}{\text{Max}(h_{s\min}, h_{w\min})} \quad (4.2)$$

(2)車輛路線容量(Vehicle line capacity)：車輛通過固定點的頻率，即路線容量乘以每運輸單位的車輛數，以符號 c 表示。例如每列車由 N 輛車廂所連結，則車輛路線容量為：

$$c = f_{\max} \times N \quad (4.3)$$

(3)最大提供路線容量(Maximum offered line capacity)：為路線上每單位時間所能運送的乘客數，即車輛路線容量乘以每車的最大乘載人數，以符號 C 表示。

$$C = cC_v = f_{\max} \times NC_v = \frac{3600NC_v}{h_{\min}} \quad (4.4)$$

(4)最大利用路線容量(Maximum utilize line capacity) C_p ：表示實際情況下所能通過一固定點的最大乘客數，由於乘客上下車而有週轉率，因此不可能用遠保持每車的最大容量，所以 $C_p < C$ 。

(5)排班路線容量(Scheduled line capacity) C_o ：依照排定之班表運作時，路線上通過一固定點的最大乘客數，通常 $C_o < C$ 。

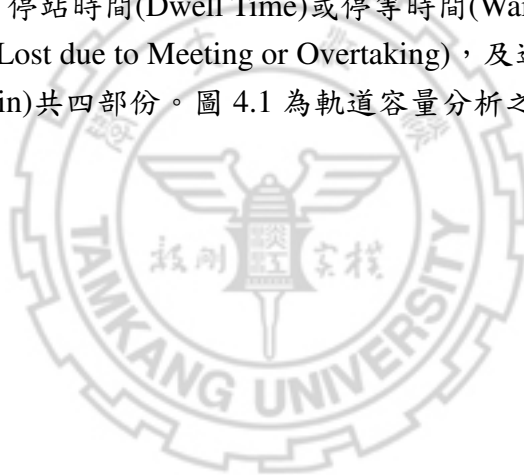
軌道運輸之路線容量可視為班距的倒數，而班距又取決於對安全之要求所必須維持固定長度之空閑時間或空間。

4.1.2 軌道容量分析架構

由前述可知，軌道容量分析之影響因素可歸結為三方面：路線條件、交通條件，及控制條件。路線條件包括線型設計、上下坡度、轉彎半徑、路段最大允許速度等等；交通條件可細分為列車性能(加減速度、制動、牽引力)、車廂設計(座位配置、車門設計)、交通特性(列車營運方式、月台停靠)等三類；控制條件則可分為包括號誌顯示、通訊定位方式、列車控制方式...等。

不同之列車運轉條件，會直接影響列車實際之運轉時隔，進而對軌道容量發生作用。列車運轉時隔乃指兩連續列車通過路線上某一點之時間間距，不同之觀測點會有不同之列車運轉時隔，實務上計算軌道容量時，會依據最瓶頸路段計算出之最大列車運轉時隔來計算，以維護正線上列車運轉之安全，並且設有運轉寬裕時間可供列車趕點。

在計算軌道容量所需關切之列車運轉時隔應包括號誌安全時距(Signal Close-in Time)、停站時間(Dwell Time)或停等時間(Waiting Time)、交會待避損失時間(Time Lost due to Meeting or Overtaking)，及運轉寬裕時間(Operating Margin)共四部份。圖 4.1 為軌道容量分析之架構。



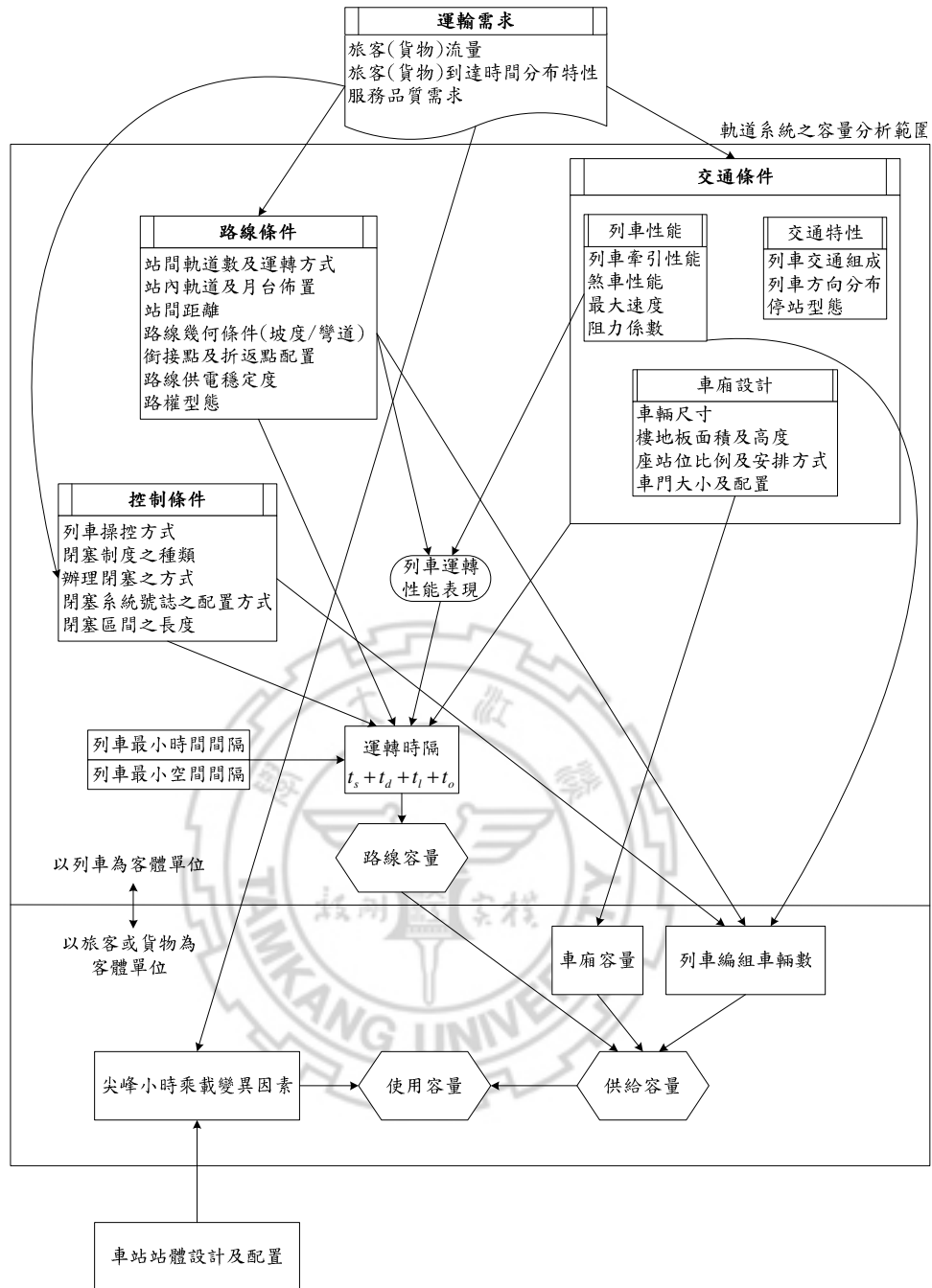


圖 4.1 軌道容量分析架構圖

資料來源：文獻[1]及本研究整理

由軌道容量分析架構圖可看出影響列車運轉時隔之影響因素，將於後續章節探討各子項目及其對列車運轉時隔之影響。

4.2 台鐵容量分析模式

台鐵為國內最早且最複雜之軌道運輸系統，由於是採取多車種混跑，且單複線混合，因此在容量上的計算，應將之一併納入考慮，方能完整描述。以下針對台鐵相關容量計算之影響因素加以整理，以作為未來模式構建時之參考。

4.2.1 台鐵軌道運輸系統特性

1. 路線環境條件

- (1) 西部幹線多為雙線區間，惟獨海線局部區域、花東線、南迴線及支線為單線區間。
- (2) 雙線區間多具備雙單線運轉之設備，但大多採用複線運轉(列車靠左行駛)為主，部分時段採用雙單線運轉。
- (3) 車站不設多為一島式月台一側式月台(岸壁式)三股道，交通繁忙之大站設有兩島式月台四股道，終點站之月台及股道數更多，未來台鐵捷運化之車站多採用兩側式月台兩股道，或一島式月台兩股道。
- (4) 平均站距約 5 公里。
- (5) 車輛基地的位置通常位於營運路線終點站後方(抽出式基地)，使得車輛往返基地對正線營運之車輛影響較小。
- (6) 路線供電電壓為 25KV，採架空線供電，變電站間隔約 40km，負責路線中前後 20km 的供電。

2. 車輛設備條件

- (1) 台鐵列車之以推拉式自強號(P&P 車)之 130km/h 為最大營運速度，莒光號則為 100km/h，通勤電車之最大營運速度為 110km/h。
- (2) 由於通勤電車停站多，站距短，因此其列車加減速度相對於其他車種較大，分列如下：

	自強號	莒光號	通勤電車
服務加速度 (km/h/s)	1.25	0.98	1.88
服務減速度 (km/h/s)	2.00	1.50	2.50

- (3) 單線區間列車方向比約為 1：1，以達到平衡列車數。

(4) 自強號、莒光號、普快列車座位配置為「非字形」，增加座位數並提高舒適度，惟車門寬度較窄，不便上下客；通勤列車座位配置採「一字形」，減少座位增加立位，由於通勤列車較多為中短程旅客，因此可容許短時間之立位乘車，車門較寬可使得旅客快速疏散。

(5) 月台設計為高月台，不利於一般乘客及殘障人士快速上下列車。

(6) 車種停站規則如下表所示：

	松 山	台 北	萬 華	板 橋	樹 林	山 佳	鶯 歌	桃 園	內 壠	中 壠	埔 心	楊 梅	富 岡	湖 口	新 豐	竹 北	新 竹
自 強	**	**		**	**			**		**							**
莒 光	**	**	**	**	**	**	**	**		**				**			**
復 興	**	**	**	**	**		**	**		**		**		**		**	**
電 聯	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**

資料來源：本研究整理

3. 控制條件

- (1) 列車控制由司機員依照路側號誌機之指示權限以人工方式控制，列車設有 ATS/ATW 裝置監控列車速度。
- (2) 閉塞方式採固定自動閉塞，由中央行控中心統一調度。
- (3) 閉塞號誌機為三位式色燈顯示。
- (4) 閉塞區間之長度視所在區間不同而不一，一般約為 1.5km~2.5km，相關之閉塞區間長度決定方式可參考前章節之敘述。

4.2.2 台鐵容量分析之解析模式

利用解析式模式的優點，可以清楚易懂模式之內涵，對於往後發展模擬模式有極大的幫助，為軌道容量評估之基礎，本章節彙整「軌道容量研究-臺鐵系統容量模式之構建分析(一)」(中興顧問工程社，交通部運研所，民國九十四年四月)，針對相關軌道容量之一般性公式為基礎進而發展台鐵容量分析之解析模式。

4.2.2.1 區間軌道容量

以目前台鐵自行發展之經驗公式為基礎，將軌道容量之公式簡化並加以修改，以得到軌道容量之一般化計算方式：

$$C = \frac{86400}{h} \times n_l \times \delta \quad (4.5)$$

其中：C=區間的軌道路線容量(TU/day)

86400=一日之時間(sec)

h=列車平均最小運轉時隔(sec)

n_l =路線軌道數

δ =路線利用率

以「日」為單位來衡量軌道容量，並且以「秒」為最小單位，以求得較為精確之軌道容量，再乘上軌道數以求取總路線容量。

4.2.2.2 最小運轉時隔

$$h_{ij} = T_s + t_l + t_m \quad (4.6)$$

其中： h_{ij} =先行列車*i*與後續追蹤列車*j*之最小運轉時隔(sec)

T_s =瓶頸號誌安全時距(sec)

t_l =列車待避損失時間(sec)

t_m =運轉寬裕時間(sec)

式中將前後兩列車*i*、*j*之瓶頸號誌安全時距、待避等時間及運轉寬裕時間加總，以求得列車最小運轉時隔。此處所表示之最小運轉時隔並未考慮

不同之列車組合及運轉方式，而是以平均之方式表現，因此宜利用加權平均的概念來求取「平均最小運轉時隔」，方能求得符合列車運行實際情況之最小運轉時隔。以車種組成比例為權重，找出瓶頸路段，求取其最小運轉時隔。

4.2.2.3 待避損失時間之計算

台鐵列車運行有低等級列車讓高等級列車先通過之考量，以增進軌道容量，交會待避之行為多發生於車站，也有少數情形是於側線待避高等級列車，直到高等級列車通過後才啟動，因而產生列車待避延誤。

列車待避延誤：

列車之待避行為原因在於前後列車在同一軌道上運行時，後續快速追蹤列車會在區間運轉中追上先行慢速列車，因此需在車站或側線先行待避，讓後續快速列車先行通過，以避免發生列車衝突，如圖 4.2 所示。

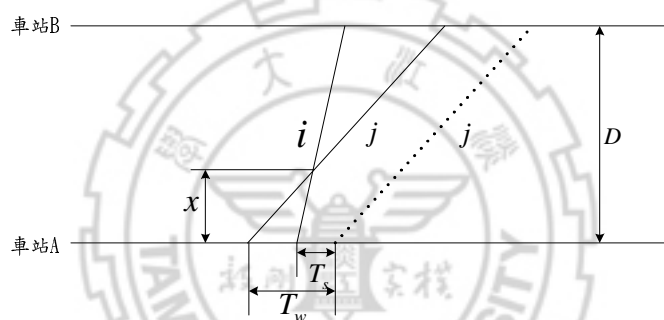


圖 4.2 列車待避延誤示意圖

先行列車 j 因待避而生之延誤為：

$$T_w = \left(\frac{1}{v_j} - \frac{1}{v_i} \right) x + T_s \quad (4.7)$$

其中： T_w =列車因待避所產生的延誤

v_i 、 v_j =列車 i 、 j 之運轉速度， $v_i > v_j$

x =列車待避的距離， $0 \leq x \leq D$

D =兩站之間的距離

T_s =瓶頸號誌時距

由式(3.46)可知，列車之待避延誤包含列車因待避之等候時間損失

$\left(\frac{1}{v_j} - \frac{1}{v_i} \right) x$ 及號誌安全時距 T_s ，較無待避情形多出了 $\left(\frac{1}{v_j} - \frac{1}{v_i} \right) x$ 的時間。

由於列車為專用路權，因此無須考慮其他交通干擾，假設列車廠站隨機

發車，待避距離 x 為 0 至 D 均勻分配，因此列車待避損失時間之期望值為：

$$t_l = \frac{1}{D} \int_0^D \left(\frac{1}{v_j} - \frac{1}{v_i} \right) x dx = \left(\frac{1}{v_j} - \frac{1}{v_i} \right) \frac{D}{2} = \frac{1}{2} (t_j - t_i) \quad (4.8)$$

其中： t_l =列車待避所產生的延誤

t_j =慢速列車 j 之站間運轉時間

t_i =快速列車 i 之站間運轉時間

若情境改為快速列車 i 等候慢速列車 j ，則待避距離變為 $D-x$ ，積分範圍不變，結果與上述計算相同，惟其不同列車等級之時間價值不同。因此可以絕對值修改上式， i 、 j 僅代表前後兩列車，所得之列車待避損失時間期望值為：

$$t_l = \frac{1}{2} |t_j - t_i| \quad (4.9)$$

列車之運轉時隔由於待避行為，使得在不同站所獲得之結果不一，例如圖 4.3 中，由車站 A 之觀點其前後列車運轉時隔僅 T_s ，若從車站 B 之觀點，則前後車運轉時隔為 $t_j - t_i$ ，因此平均運轉時隔為 $T_s + t_l = T_s + (t_j - t_i)/2$ 。

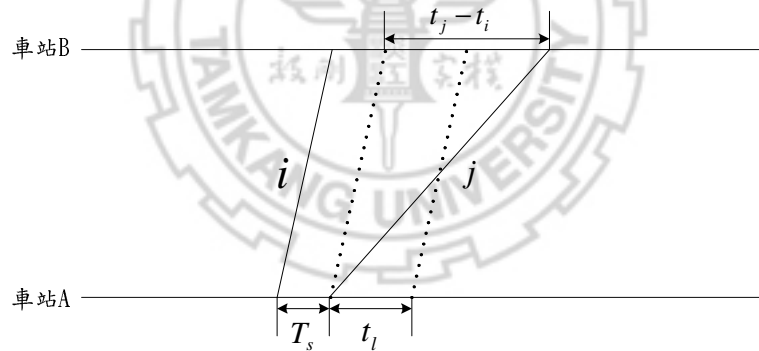


圖 4.3 列車運轉時間中待避損失時間之示意圖

4.2.2.4 運轉寬裕時間之決定

由於號誌顯示、轉轍器扳轉、停車時間及列車運轉過程皆為動態，未必每次皆可按照原定計畫完成動作，因此必須在列車運行計畫中，加上運轉寬裕時間於列車運轉時隔，以使得列車若因誤點可以趕點之空間，維持列車運行計畫之推進不致被打亂。

若是以模擬模式可進行多次模擬，以求出最小運轉時隔之分佈，並據此設定一合理之數值，以涵蓋大部分之運轉情形。本研究探討之解析模式求解時必須將運轉寬裕時間與計算出之最小運轉時隔加總，以俾列車運行使用。

運轉寬裕時間可由下列兩種方式決定：

1. 依經驗值決定

- (1) 美國軌道容量手冊之經驗值為 20 秒，日本對於號誌安全時距之寬裕時間設為 10 秒。
- (2) 依研究對象之最小運轉時隔觀測資料，或利用模擬所得之結果與解析模式之計算值相比對，以涵蓋大部分運轉情形之值與解析模式計算結果之差異即為運轉寬裕時間。

2. 按比例計算

按照比例計算之概念乃來自於號誌安全時距越大，則其產生之變異也越大，因此採用較大之運轉寬裕時間可確保時課表之穩定，即

$$t_m = \beta T_s \quad (4.10)$$

其中： t_m =運轉寬裕時間(sec)

T_s =瓶頸號誌時隔(sec)

β =運轉寬裕時間係數

運轉寬裕時間之設定，影響列車營運計畫之彈性調整空間以及準點性，採取過於樂觀之數值會使得服務水準下降甚或影響列車行車安全，若採取過於悲觀之值則使得軌道容量下降；IC Codes 405 公式中建議使用之 β 值為 0.67，可供國內參考，而在「軌道容量研究-台鐵系統容量模式之建構分析(一)」研究中利用變動運轉寬裕時間係數之方式，建議運轉寬裕時間係數 β 值採用 0.35 計算。

4.2.2.5 平均最小運轉時隔之計算

最小運轉時隔之組成包含瓶頸號誌安全時距、列車交會待避之損失時間，及運轉寬裕時間：

$$h_{ij} = T_s + t_l + t_m \quad (4.11)$$

其中： h_{ij} =先行列車 i 與後續追蹤列車 j 之最小運轉時隔(sec)

T_s =瓶頸號誌安全時距(sec)

t_l =列車交會待避損失時間(sec)

t_m =運轉寬裕時間(sec)

若營運列車之速度等級不只一種，或部分路段部分時段採單線運轉時，則最小運轉時隔必須以平均值來計算，方能反映實際之運轉情形。

1. 列車交通組成之影響

若一共有 n_c 種列車之速度等級，則先行列車即後續追蹤列車之配對方式有 n_c^2 種組合，除了自身與自身配對之外，也包括自身與其他等級列車之配對；其最小運轉時隔可依式(5.2)計算得到。對於平均最小運轉時隔之計算，若時刻表已知，即根據時刻表列車組合之相對頻率來計算，若時刻表未知，則根據營運列車數的相對頻率來計算。

(1) 時刻表已知

$$h = \sum h_{ij} \times p_{ij} \quad (4.12)$$

其中： h =平均最小運轉時隔(sec)

h_{ij} =列車 j 跟隨列車 i 之最小運轉時隔(sec)

p_{ij} =列車 j 跟隨列車 i 之相對頻率(可由時刻表統計求得)

(2) 時刻表未知

$$h = \sum h_{ij} \times p_{ij} = \sum h_{ij} \frac{n_i \cdot n_j}{n^2} = \frac{1}{n^2} \sum h_{ij} \cdot n_i \cdot n_j \quad (4.13)$$

其中： n_i =第 i 種列車之營運列車數(TU)

n_j =第 j 種列車之營運列車數(TU)

$n = \sum_i n_i$ =總營運列車數(TU)

2. 單/複線運轉之影響

單一軌道在特定時段內之運用可分為單線運轉(反方向運轉)及複線運轉(同方向運轉)。本研究之研究對象為西正線下行軌道，股道使用採複線運轉。使用複線運轉之平均最小運轉時隔較小，使用單線運轉之平均最小運轉時隔較大。修正後複線平均最小運轉時隔之計算如下：

$$h = p_s h_s \quad (4.14)$$

其中： h =平均最小運轉時隔(sec)

h_s =同方向列車組合之平均最小時隔(sec)

p_s =同方向運轉列車組合之占比例

4.2.2.6 運轉方式及前後列車速差之影響

由於台鐵於正線上運行之車種複雜，前後列車速度不一，因此列車之運轉時隔為變動。由前文所知，通常瓶頸之號誌時距發生於車站。又由於單/複線運轉，因此瓶頸之號誌時距有可能發生於車站或交會車站，以下針對前後車列車速差之不同而影響之號誌時距加以介紹。

1. 先行列車速度大於後續追蹤列車

當先行列車速度大於後續追蹤列車速度時，若在出發車站即保持安全距離，則在路段中後續追蹤列車將無法追趕上先行列車，因此將不會發生列車衝突，所以在此情境中需關注的是出發車站之離站號誌時距。

圖 4.4 中，車站 A 為出發站，車站 B 為到達站，當先行之快速列車 i 從車站 A 出發後，僅需間隔一段離站號誌時距，後續追蹤慢速列車即可離站，且在運轉的過程中不會發生衝突。其最小號誌時距為：

$$T_s = T_{s,D}^A \quad (4.15)$$

其中： T_s = 瓶頸之號誌時距(sec)

$T_{s,D}^A$ = 出發車站之離站號誌時距(sec)

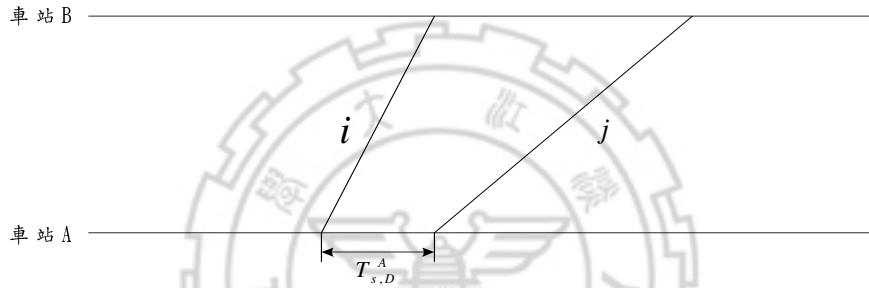


圖 4.4 先行列車速度大於後續追蹤列車之瓶頸號誌時距

2. 先行列車速度小於後續追蹤列車

當先行列車速度小於後續追蹤列車時，則在區間運轉時，先行列車有可能會被後續追蹤列車追上，因此此處需關注抵達車站之安全時距，則在區間運轉的過程中將不會發生衝突。

在圖 4.5 中，先行之慢速列車 i 抵達車站 B 後，在經過一段進站號誌時距，後續追蹤之快速列車 j 便可抵達，可確保區間運轉中之安全。其瓶頸號誌時距為：

$$T_s = T_{s,A}^B \quad (4.16)$$

其中： T_s = 瓶頸之號誌時距(sec)

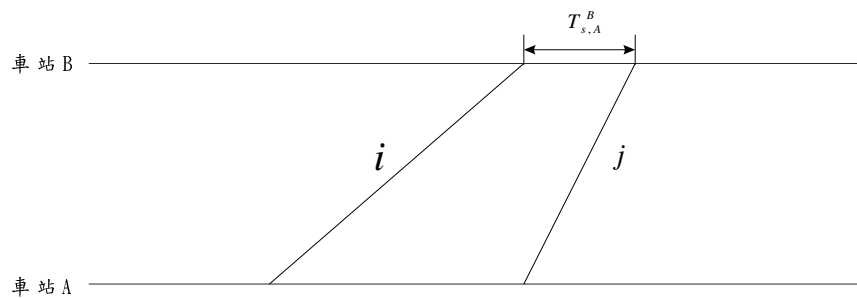


圖 4.5 先行列車速度小於後續追蹤列車之瓶頸號誌時距

3. 先行列車速度等於後續追蹤列車

當先行列車與後續追蹤列車之速度相同時，則瓶頸號誌時距不一定會發生於出發站或是抵達站，端視車站之配置，因此路線上最大號誌時距為最瓶頸處之號誌時距，即：

$$T_s = \max(T_{s,D}^A, T_{s,A}^B) \quad (4.17)$$



4.3 模擬模式之構建

以前述解析模式作為模擬模式之雛形，構建台鐵列車模擬模式。除基本列車推進外，相關之影響列車運轉性能因素諸如路線坡度、曲線、軌道佔用情形、錯車待避邏輯……等皆需納入模擬模式中考慮。本節共分為以下三部份：「行車動力學」，用以控制列車推進，加減速；「列車排點」作為控制站內軌道佔用情形以及錯車邏輯；「列車模擬」確保列車間距，使得在安全前提下，不考慮號誌系統影響仍能有效反應出現實列車運行情形，並且利於移動自動閉塞行車制之情境模擬。

4.3.1 行車動力學

運動學基本速度公式

$$V_{n+1} = V_n + dt \times a_n \quad (4.18)$$

運動學基本距離公式(以梯形公式轉換)

$$S_{n+1} = S_n + \frac{1}{2} (V_n + V_{n+1}) \times dt \quad (4.19)$$

V ：速度

a ：加(減)速度

dt ：時間增量

S ：位移

由上述公式，可得出距離及時間之公式：

$$S_n = S_{n-1} + \frac{V_{n-1}(V_n - V_{n-1})}{a_n} + \frac{(V_n - V_{n-1})^2}{2a_n} \quad (4.20)$$

$$t_n = t_{n-1} + \frac{V_n - V_{n-1}}{a_n} \quad (4.21)$$

當給定特定距離 S ，($S_n < S < S_{n+1}$)，可得出當時之列車速度：

$$V = \sqrt{V_n^2 + 2(S - S_n)a_n} \quad (4.22)$$

4.3.1.1 路線坡度對列車加減速性能之影響

軌道運輸之坡度通常是以 ‰ 為單位，坡度將會影響列車之加減速性能，可以下式估計：

$$a(G) \approx \frac{M_e a(0) - Mg \frac{G}{1000}}{M_e} = a(0) - \frac{g \times G}{1000\rho} \quad (4.23)$$

其中： $a(G)$ =列車在坡度 $G/_{00}$ 的速度(m/s^2)

$M_e = \rho M$ 即列車之等效質量(Equivalent Mass)

ρ =等效質量係數，表示列車運動過程中，轉動零件如車輪、車軸吸收的能量之等價質量

M =列車編組質量(kg)

g =重力加速度(m/s^2)， $g=9.81 m/s^2$

G =路線坡度($_{00}^0$)

等效質量係數 ρ 一般介於 1.04~1.10 之間。台鐵機車牽引之列車採用 1.06，電聯車採用 1.088。若將等效質量係數 ρ 及重力加速度 g 代入式(4.23)可以下式來趨近：

$$a(G) \approx a(0) - 0.009G \quad (4.24)$$

若為減速度，由於坡度之阻力相反，以下式來表示：

$$b(G) \approx b(0) + 0.009G \quad (4.25)$$

4.3.1.2 線型曲線對列車加減速性能之影響

在圖 4.6 中，當列車行駛於彎道時，由於鋼軌為剛性物體，因此在彎道因內外軌長度不同，產生鋼輪縱向及橫向之滑動、摩擦；又因為列車行駛於彎道時離心力產生作用，使得外側鋼輪與外側鋼軌會產生運轉摩擦；以上原因皆是產生彎道阻力之原因。另外鋼軌長期磨損，曲線變形、軌距不正確、軌道保養不良等原因也均為造成彎道阻力之主因。

國內台鐵對此之計算主要參考莫里遜氏發表關於彎道阻力公式：

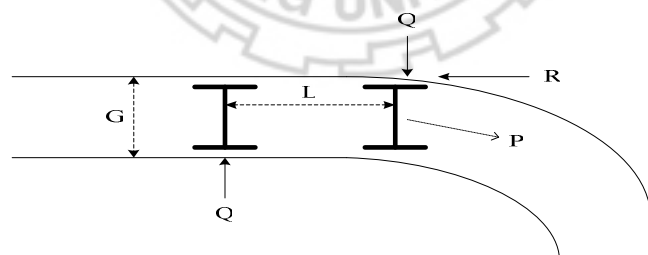


圖 4.6 彎道阻力示意圖

$$T_k(v, r) = \frac{1000\mu(G+L)}{2\gamma} \times W \quad (4.26)$$

$T_k(v, r)$ ：彎道阻力 (kg/ton)

μ ：鋼輪與鋼軌之摩擦係數

G ：軌距(m)

L ：固定軸距(m)

γ ：彎道半徑(m)

W ：列車重量(ton)

Q ：橫向阻力

R ：縱向阻力

若將台鐵軌道即列車資料代入，令 $\mu = \frac{1}{4.5}$ ， $G = 1.067$ ， $L = 4.3$ ，可得：

$$T_k(v, r) = \frac{1000 \times \frac{1}{4.5} \times (1.067 + 4.3)}{2\gamma} \times W$$
$$T_k(v, r) = \frac{600}{\gamma} \times W \quad (4.27)$$

此即為大眾所熟知之彎道阻力計算公式，一般來說，若非精密計算或半徑較小之彎道處，通常忽略彎道之阻力。

台鐵對於曲線速限依照其曲線半徑有表 4.2 之規定：

表 4.2 曲線速限表

曲線半徑	每小時速限(km/h)
900	125
800	120
700	110
600	100
500	90
450	85
400	80
350	75
300	70
250	65
225	60
200	55
150	50
125	45
100 以下	40

4.3.2 列車排點

錯車計畫主要內容為決定列車在系統中錯車之時間及地點。錯車行為為一列列車在軌道上超越另一輛同向列車。錯車條件必須有一個或多個旁軌，因此錯車之行為大多發生於車站中，或特定擁有旁軌或袋型軌之路段。旁軌之佈設位置、旁軌長度，旁軌數量直接影響到列車避讓時之速度及停等時間。參考文獻〔9〕彙整出下列規則。

符號說明：

M ：車站集合，包括實體與虛擬車站。實體車站包括發車站(令為 1 號站)至終點站(n)之各車站，虛擬車站假設為旁軌處，再以參數 $stop_i^m$ 表示列車待避或通過之情形。由於研究路段為南下路段西幹線，因此將由北往南對車站依序編號，並且定義車站 m 至車站 $m+1$ 之間的軌道車為軌道 m

I ：南下列車集合

V_{\max}^{im} ：列車 i 在軌道 m 上的行駛速率上限

V_{\min}^{im} ：列車 i 在軌道 m 上的行駛速率下限

λ_i^m ：列車 i 在車站 m 的最小停站時間

S_i^m ：列車 i 在車站 m 的最大停站時間

ini_i ：列車 i 在發車站之最早可發車時間

η_{ij}^m ：同方向列車 i 與 j 在離開車站 m 所需保持的最小時間間距

γ_{ij}^m ：同方向列車 i 與 j 在進入車站 m 所需保持的最小時間間距

ϕ ：一個大的正數

a_i^m ：列車 i 抵達車站 m 的時間， $a_i^m \geq 0$ ， $\forall i \in I, m \in M$

d_i^m ：列車 i 離開車站 m 的時間， $d_i^m \geq 0$ ， $\forall i \in I, m \in M$

0-1 變數：表示列車對於軌道之使用情況以及使用之順序

$stop_i^m$ ：1，列車 i 停靠車站 m ；0，列車 i 不停靠車站 m (即車站 m 為通過站)

b_{ij}^m ：1，同向列車 i 比列車 j 早使用軌道 m ；

0，同向列車 j 較列車 i 早使用軌道 m ； $b_{ij}^m \in \{0,1\} \quad \forall i, j \in I, m \in M$

以下將數學模式分為三類：1.單一系列車限制式，2.同向任兩列車限制式，3.同向任三列車限制式，介紹如下：

1. 單一系列車限制式：針對單一系列車在軌道上及車站內之行為限制。

限制式(1)之意義在於描述列車於車站內之到離停站行為。

$$d_i^m \geq a_i^m + stop_i^m \times \lambda_i^m \quad \forall i \in I, m \in M \quad (1)$$

限制式(2)之用意在於防止列車於車站停站時間超過容許之上限 S_i^m ，實際意義可供設定最大可接受列車於車站停站時間延滯，以供決策者決定是否取消/加派列車。

$$d_i^m \leq a_i^m + stop_i^m \times S_i^m \quad \forall i \in I, m \in M \quad (2)$$

以下兩限制式之意義為限制列車運行速率不至違反速度之上下限。

$$V_{\max}^{im} \times a_i^m - V_{\max}^{im} \times d_r^{m+1} \geq L^m \quad \forall i \in I, m \in M \quad (3)$$

$$V_{\min}^{im} \times a_i^m - V_{\min}^{im} \times d_r^{m+1} \leq L^m \quad \forall i \in I, m \in M \quad (4)$$

對於發車站來看，所有南下之列車發車時間均須大於或等於發車站起始發車時間，以限制式(5)表示之：

$$d_i^m \geq ini_i \quad \forall i \in I \quad (5)$$

2. 同向兩列車限制式：針對同向兩列車在軌道上及車站內之行為限制，以及錯車行為之描述。此一部份分為 A、控制錯車順序 0-1 變數之關係；B、同向兩列車限制。首先針對同向兩列車在運行過程中，限制其超過一次之錯車行為：

$$b_{ij}^m \leq b_{ij}^{m+1} \quad \forall i, j \in I, i < j, m \in M \quad (6)$$

A. 控制錯車順序 0-1 變數之關係：

$$b_{ij}^m + b_{ji}^m = 1 \quad \forall i, j \in I, m \in M \quad (7)$$

B. 同向兩列車限制：

當列車 j 較早離開車站 m 時，後行列車循同方向離開時，最少必須保留 η_{ij}^m 的時間間隔，若當列車 j 落後於列車 i 時，則 $b_{ij}^m = 1$ ，使得此限制式必成立，即此限制式不具任何約束力。

$$d_i^m \geq d_j^m + \eta_{ij}^m - \phi \times b_{ij}^m \quad \forall i, j \in I, i \neq j, m \in M \quad (8)$$

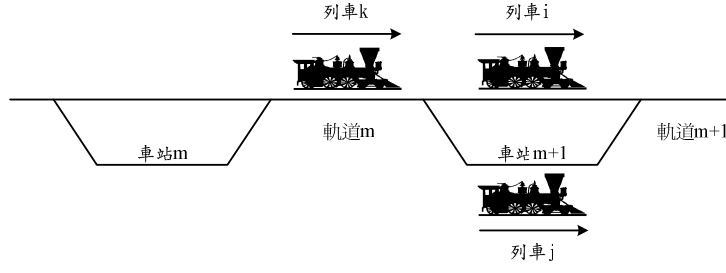
當列車 j 較早進入車站 m 時，後行同向列車應保持最小安全距離 γ_{ij}^m ，

$$a_i^{m+1} \geq a_j^{m+1} + \gamma_{ij}^m - \phi \times b_{ij}^m \quad \forall i, j \in I, i \neq j, m \in M \quad (9)$$

3. 同向三列車限制式：針對同向三列車在軌道上及車站內之行為限制及錯車行為之描述。此一部份之重點在於排除兩兩列車間相互產生之衝突。

$$\phi(1 - b_{ij}^{m+1}) + \phi(1 - b_{ik}^m) + \phi(1 - b_{jk}^m) + a_k^{m+1} \geq d_i^{m+1} \quad \forall i, j, k \in I, i \neq j \neq k, m \in M \quad (10)$$

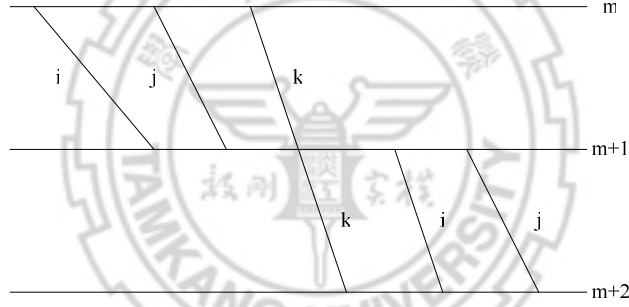
同向三列車於同一車站附近時，其分布狀況如下圖，限制式控制 k 列車必須待列車 i 或列車 j 其中之一離開該站後，方得以離開，



當 $b_{ij}^m - b_{ij}^{m+1} = 1$ 成立時，代表列車 i 被列車 j 超車，發生錯車行為；依此類推其餘表示方式。由於站內軌道有限，因此 $b_{ij}^m - b_{ij}^{m+1} = 1$ 與 $b_{jk}^m - b_{jk}^{m+1} = 1$ 不可同時成立，表示如下：

$$(b_{ij}^m - b_{ij}^{m+1}) + (b_{jk}^m - b_{jk}^{m+1}) \leq 1 \quad \forall i, j, k \in I, i < j < k, m \in M \quad (11)$$

同向列車不產生錯車而同時使用車站軌道，且後來之列車超越此兩部列車，有以下示意圖可看出，同向同時有三輛列車使用車站內軌道，因此修改上述限制式，使得列車 i 與列車 j 不同時被列車 k 超車，便不會發生下圖之情形。修改後之限制式如下：



$$(b_{ik}^m - b_{ik}^{m+1}) + (b_{jk}^m - b_{jk}^{m+1}) \leq 1 \quad \forall i, j, k \in I, i < j < k, m \in M \quad (12)$$

4.3.3 列車運行

列車運行係採用時間推進方式，推進單位時間為秒，列車運行時須受限於路段速限、前後車間距、列車任務。欲描述實際列車運行情形無法，需以標的速度點(即停站點、速限區速限)反推出「速限包絡線」、「停站包絡線」得出列車據以運行之速度-時間曲線，以圖 4.7 示之。

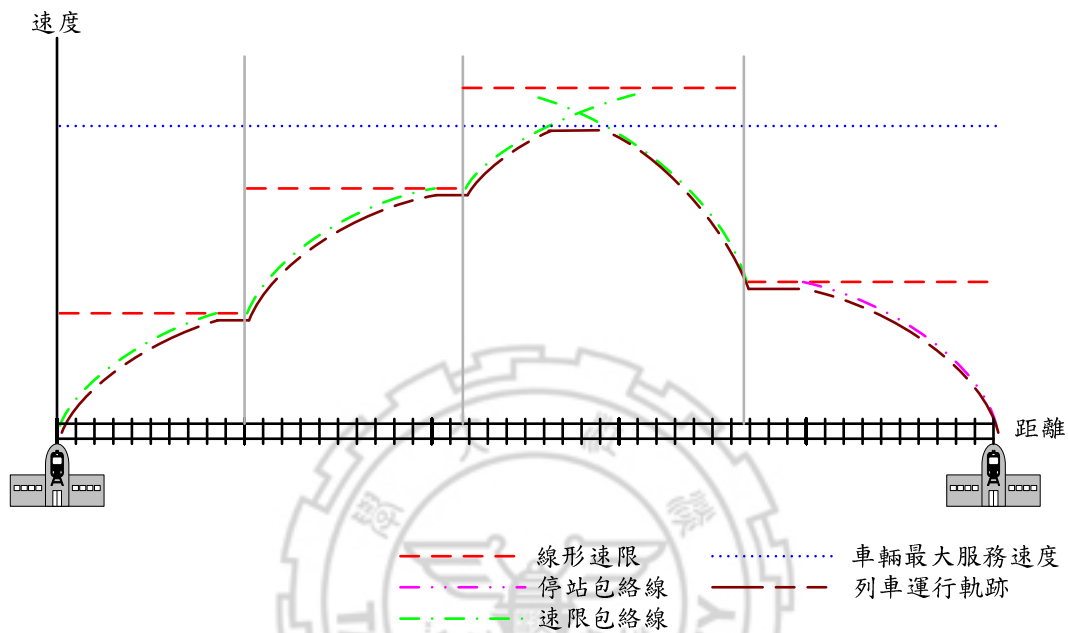


圖 4.7 模擬列車運行軌跡圖

4.3.3.1 號誌安全時距

號誌安全時距乃指受制於號制系統所得之列車最小運轉時間間隔，主因為受到不同閉塞時間之影響。台鐵之號誌主要為三位式號誌系統，以下所介紹之號誌安全時距，主要參考運研所之「軌道容量研究—台鐵系統容量模式之構建分析(一)」及國內外相關之號誌安全時距，將各種相關影響之因素納入模式中。

雖然移動自動閉塞行車制可將路段上號誌設備拔除，但進出站號誌機，道岔應答器仍保留，作為判斷列車安全進路以及確認列車進入股道。因此在本研究中，雖不考慮號誌於路段中的作用而以列車安全空間間距代替其功能，但進出站號誌時距仍需納入模式中考慮，其說明如下：

列車於路段中行駛速度較高，且若無突發狀況，通常於區間內可以連續行駛通過，因此在路線中通常出現瓶頸之路段較常發生於車站處，由於車輛進站/離站之加/減速度、上下客等行為影響列車整體運轉時隔。整段路線區間的最小運轉時隔係在任一地點任一區段所容許之運轉時隔最大者。

由於瓶頸路段多發生於車站處，最與之相關的因素即為號誌安全時距，由於考慮台鐵之特性，停站等待後行列車通過之列車通常等級較後行列車為低，速度也較慢，因此將以以下五種列車停靠站之方式討論。

1. 同向列車進站號誌安全時距-站內停靠同一軌道

若車站內僅有一軌道，後續追蹤列車必須等待先行列車淨空月台並離站方能進入，以避免發生事故。若是在先行列車離開車站通過出發號誌機後，經過解除閉塞及清除號誌時間，進站號誌機由於車站區間清空，因此進站號誌由黃燈注意轉為綠燈可通行時，後續追蹤列車恰好以原巡航速度到達距離進站號誌機一個號誌視距的長度時，由於並未因號誌而減速，因此前後兩列車之運轉時隔可望達到最小。

先行列車與後續追蹤列車進入車站之時間差即為列車進站之安全時距，可由下式表示：

$$T_{s,A1} = t_{d_i} + t_{x_i} + t_o + t_{v_j} + t_{b_j} \quad (4.28)$$

其中： $T_{s,A1}$ =同一股道進站之號誌安全時距 (sec)

t_{d_i} =先行列車之停車時間(m/s)

t_{x_i} =先行列車從靜止啟動至車尾通過出發號誌機的時間(sec)

t_o =先行列車車尾通過出發號誌機後，解除閉塞及清除號誌的時間(sec)，台鐵號誌系統以 4 秒合併計算

t_{v_j} =後續追蹤列車以巡航速度 v_j 運轉的時間(sec)

t_{b_j} =後續追蹤列車之進站緊軔時間(sec)

式(3.8)中若先行列車之停車時間為 0 秒，即先行列車煞停馬上啟動運轉，則可使 $t_{d_i}=0$ ，此時之運轉時隔即為最小號誌時距，如下所示：

$$t_s = t_{x_i} + t_o + t_{v_j} + t_{b_j} \quad (4.29)$$

其中： t_s =不含停車時間之最小號誌時距 (sec)

由於列車出發列車車尾通過出發號誌機時，通常並未加速至路段容許最高速度，因此 t_{x_i} 可由一般物理公式得到：

$$t_{x_i} = \sqrt{\frac{2(L_i + s_x)}{a_i}} \quad (4.30)$$

其中： s_x =列車於車站停車時車頭與出發號誌機之距離(m)

a_i =先行列車之加速度(m/s^2)

t_{v_j} 為續行列車已巡航速度 v_j 運轉之時間可由式(3.11)表示：

$$t_{v_j} = \frac{s_{d_j} + B_l + (B_s - s_x - s_{b_j})}{v_j} \quad (4.31)$$

其中： s_x =列車停車位置與出發號誌機的距離(m)

s_{b_j} =後續追蹤列車的進站緊軔距離(m)

由於列車進站屬可預測事件，因此司機員可及早準備，提早反應，不會有類似號誌機之反應時間、軔機作用時間之空走距離等，僅需計算列車開始煞車至煞停之距離：

$$s_{b_j} = \frac{v_j^2}{2b_j} \quad (4.32)$$

其中： b_j =後續追蹤列車之服務減速度(m/s²)

將上式整理後可得：

$$t_{v_j} = t_r - \frac{v_y^2}{2b_j v_j} + \frac{B_l + B_s - s_x}{v_j} \quad (4.33)$$

根據運動學公式，後續追蹤列車之緊軔時間可以下式計算：

$$t_{b_j} = \frac{v_j}{b_j} \quad (4.34)$$

綜合整理上式可得列車之號誌安全時距如下：

$$T_{s,Al} = \sqrt{\frac{2(L_t + s_x)}{a_i}} + \frac{v_j}{b_j} - \frac{v_y^2}{2b_j v_j} + \frac{B_l + B_s - s_x}{v_j} + t_o + t_r + t_{d_i} \quad (4.35)$$

2. 同向列車進站號誌安全時距-站內停靠不同軌道

車站內有兩股以上之軌道工列車停靠，使得先行列車及後續追蹤列車可於站內停靠不同軌道，只要轉轍器及號誌設定完成，後續追列車即可進站。

當先行列車停於車站後，車站之進站號誌機會隨著轉轍器之扳轉而由紅燈不可進入區間燈號轉為閃黃燈注意燈號，進站號誌機前一個號誌將會隨之轉為綠燈可通行燈號，若於此時，司機員恰好將列車行駛於綠燈號誌前視距可見位置，則列車可不減速直接維持原行駛速度或按照路段最高可允許速度行駛，並且可達到最小列車進站時隔。

號誌安全時距需以道岔完成先行列車服務後扳至正確股道開始計算，但列車進站時隔是以列車停車後之時間來計算，因此利用先行列車經過道岔之時間來倒推：

$$T_{s,A2} = (t_p - t_{e_i}) + t_{v_j} + t_{b_j} \quad (4.36)$$

其中： $T_{s,A2}$ =不同股道進站之號誌安全時距(sec)

t_p =先行列車通過道岔後，解除第一股道進路、扳轉轉轍器、鎖定第二股道進路，以及號誌變換的整體作業時間(sec)

t_{e_i} =先行列車車尾通過道岔直到列車停止的運轉時間(sec)

t_{v_j} =後續追蹤列車以巡航速度 v_j 運轉的時間(sec)

t_{b_j} =後續追蹤列車之進站緊軔時間(sec)

式(4.36)中之 $(t_p - t_{e_i})$ 表示先行列車停站後直到轉轍器扳轉至正確股道、進路解除及號誌變換的時間， $(t_p - t_{e_i})$ 通常為正值，若 $(t_p - t_{e_i})$ 為負值，則代表在先行列車停車之前即完成所有相關之作業。

t_p 由軌道電路、號誌及轉轍器連鎖設備之作業時間而定，台鐵之號誌系統約需 13 秒之作業時間，因此採用 15 秒以確保動作完成。先行列車車尾通過道岔直到列車停止之運轉時間 t_{e_i} ，與先行列車停車後車尾與到岔的距離 s_e 及列車減速度 b_i 有關，可用式(4.37)表示：

$$t_{e_i} = \sqrt{\frac{2s_e}{b_i}} \quad (4.37)$$

後續追蹤列車以巡航速度 v_j 運轉的時間為：

$$t_{v_j} = \frac{s_{d_j} + B_2 + B_1 + (B_s - s_x - s_{b_j})}{v_j} \quad (4.38)$$

其中： s_{d_j} =後續追蹤列車之號誌視距(m)

B_2 =車站前第二閉塞區間之長度(m)

B_1 =車站前第一閉塞區間之長度(m)

s_x =列車於車站停車時車頭與出發號誌機的距離(m)

s_{b_j} =後續追蹤列車之進站緊軔距離(m)

v_j =後續追蹤列車之巡航速度(m/s)

將上式整理可得：

$$t_{v_j} = t_r - \frac{v_y^2}{2b_j v_j} + \frac{B_2 + B_1 + B_s - s_x}{v_j} \quad (4.39)$$

$$T_{s,A2} = \frac{v_j}{b_j} - \sqrt{\frac{2s_e}{b_i} - \frac{v_y^2}{2b_j v_j}} + \frac{B_2 + B_1 + B_s - s_x}{v_j} + t_p + t_r \quad (4.40)$$

綜合比較可得知，當站內使用不同軌道供列車停靠時，列車進站運轉時隔較使用同一軌道停靠之情形減少了列車停車時間 t_{d_i} 、解除閉塞及清除號誌之時間 t_o 、先行列車離開出發號誌機之時間 t_{x_i} ，及先行列車通過道岔直到停止的運轉時間 t_{e_i} ，但也因為站內有多條股道，因此也增加了轉轍器扳轉時間及號誌變換時間 t_p 、後續追蹤列車行駛第二閉塞區間之時間 $\frac{B_2}{v_j}$ 。通常列車在車站內之緊軔、停車以及啟動加速時間遠大於以巡航速度通過一個閉塞區間之時間，因此通常進站運轉時隔會小於理論之結果。

3. 同向列車離站號誌安全時距-站內停靠同一軌道

當後續追蹤列車欲停靠車站，與先行列車停靠同一軌道時，須待先行列車車尾通過離站號誌機，並且經過解除閉塞即清除號誌的時間，當進站號誌機顯示為綠色可通過時，後續追蹤列車方得以進入車站停靠。當後續列車司機員以巡航速度行駛至進站號誌前一個號誌視距之位置時恰好與進站號誌轉為綠燈同時，則可使前後兩列車之運轉時隔達最小。式(4.41)表示先行列車離開車站直到後續追蹤列車離開車站之時間間隔：

$$T_{s,D1} = t_s + t_{d_j} = t_{n_i} + t_o + t_{v_j} + t_{b_j} + t_{d_j} \quad (4.41)$$

其中： $T_{s,D1}$ =同一股道離站之號誌安全時距(sec)

$t_s = t_{n_i} + t_o + t_{v_j} + t_{b_j}$ =最小號誌時距(sec)

t_{d_j} =後續追蹤列車之停車時間(sec)

t_{n_i} =先行列車出發後直到車尾離開第 n 個閉塞區間的運轉時間(sec)

t_o =解除閉塞及清除號誌的時間(sec)，台鐵以4秒計算

t_{v_j} =後續列車以巡航速度運轉之時間(sec)

t_{b_j} =後續列車之進站緊軔時間(sec)

先行列車通過第 n 個閉塞區間即表示其車尾通過第 $n-1$ 號號誌機，與此同時，其行駛速度可能已達區間最大允許速度 v_i 或正值加速階段。另 s_{a_i} 表示

先行列車加速至 v_i 之距離，即 $s_{a_i} = \frac{v_i^2}{2a_i}$ 。若 $L_i + s_x + B_n > \frac{v_i^2}{2a_i}$ ，表示先行

列車車尾是以 v_i 的速度通過第 $n-1$ 號號誌機，若 $L_i + s_x + B_n \leq \frac{v_i^2}{2a_i}$ ，則表示

列車仍在加速，兩種情形之 t_{n_i} 計算如下：

$$(1) L_i + s_x + B_n > \frac{v_i^2}{2a_i}$$

先行列車車尾在通過第 $n-1$ 號號誌機時即已加速至路段可運許最大速度 v_i ，則 t_{n_i} 包含加速運轉至 v_i 的時間： $\frac{v_i}{a_i}$ ，及等速運轉時間 $\frac{L_i + s_x + B_n - s_{a_i}}{v_i}$ 兩部分：

$$t_{n_i} = \frac{v_i}{a_i} + \frac{L_i + s_x + B_n - s_{a_i}}{v_i} = \frac{v_i}{2a_i} + \frac{L_i + s_x + B_n}{v_i} \quad (4.42)$$

$$(2) L_i + s_x + B_n \leq \frac{v_i^2}{2a_i}$$

列車在通過第 $n-1$ 號號誌機時尚未加速至 v_i ，因此完全處於加速運轉狀態，由運動學公式可得 $L_i + s_x + B_n = \frac{a_i t_{n_i}^2}{2}$ ，移項可得：

$$t_{n_i} = \sqrt{\frac{2(L_i + s_x + B_n)}{a_i}} \quad (4.43)$$

後續追蹤列車以巡航速度 v_j 之運轉時間為：

$$t_{v_j} = \frac{s_{d_j} + (B_s - s_x - s_{b_j})}{v_j} \quad (4.44)$$

其中： s_{d_j} = 後續追蹤列車之號誌視距(m)

B_s = 列車所在之避塞區間長度(m)

s_x = 列車停車位置車頭與出發號誌機的距離(m)

s_{b_j} = 後續追蹤列車之停車緊軔距離(m)

經代換式子後可得：

$$t_{v_j} = t_r - \frac{v_y^2}{2b_j v_j} + \frac{(B_s - s_x)}{v_j} \quad (4.45)$$

綜合整理可得列車之最小運轉時隔：

$$(1) L_i + s_x + B_n > \frac{v_i^2}{2a_i}$$

$$T_{s,D1} = \frac{v_i}{2a_i} + \frac{v_j}{b_j} - \frac{v_y^2}{2b_j v_j} + \frac{L_i + s_x + B_n}{v_i} + \frac{B_s - s_x}{v_j} + t_o + t_r + t_{d_j} \quad (4.46)$$

$$(2) L_i + s_x + B_n \leq \frac{v_i^2}{2a_i}$$

$$T_{s,D1} = \sqrt{\frac{2(L_i + s_x + B_n)}{a_i}} + \frac{v_j}{b_j} - \frac{v_y^2}{2b_j v_j} + \frac{B_s - s_x}{v_j} + t_o + t_r + t_{d_j} \quad (4.47)$$

兩者皆為於站內使用同一股軌道停站，由式(4.46)、(4.47)可知兩者之差

別在於 $\frac{L_i + s_x + B_n}{v_i} - \sqrt{\frac{2(L_i + s_x + B_n)}{a_i}}$ ，而若是將列車進站運轉時隔與列車離

站運轉時隔相比較，列車進站運轉時隔增加了後續追蹤列車行駛車站前一個閉塞區間長度之時間，減少了先行列車運行離站後第一個閉塞區間常的的時間；兩者消長之間，若假設車站前一個閉塞區間與車站後一個閉塞區間長度相同時，由於列車之加速度通常小於減速度，因此列車之進站時隔會小於列車離站時隔。

4. 同向列車離站號誌安全時距-站內停靠不同軌道

當同向之列車於站內停靠不同軌道時，只要先行列車離站後，列車車尾通過出發號誌機的下兩個號誌機，則車站之出發號誌機將會由黃燈轉為綠燈（由於台鐵為三位式顯示號誌機），此時後續追蹤列車方能離站進入下個閉塞區間。與前述同向列車停靠站內同一軌道之最大不同在於後續追蹤列車不必等候先行列車離站後才可進入車站內停靠，因此可縮短運轉時隔。

當先行列車車尾通過車站後第二個閉塞區間時，可加速至路段允許速度，因此其運轉時間包括加速運轉及等速運轉兩部分。因此先行列車離開車站至後續追蹤列車可離站之號誌安全時距可依下式表示：

$$T_{s,D2} = t_{a_i} + t_{v_i} + t_o + t_r \quad (4.48)$$

其中： $T_{s,D2}$ =不同軌道離站之號誌安全時距(sec)

t_{a_i} =先行列車之加速運轉時間(sec)

t_{v_i} =先行列車以巡航速度 v_i 運轉的時間(sec)

t_o =解除閉塞及清除號至的時間(sec)，台鐵號誌以 4 秒計算

t_r =車站離站出發號誌顯示綠燈後，關閉列車車門以及列車司機員確認出發號誌的反應時間(sec)，合併以 5 秒計算

先行列車的加速運轉時間計算如下：

$$t_{a_i} = \frac{v_i}{a_i} \quad (4.49)$$

先行列車以巡航速度 v_i 運轉的時間計算如下：

$$t_{v_i} = \frac{L_i + s_x + B_n + B_{n-1} - s_{a_i}}{v_i} \quad (4.50)$$

其中： L_i =先行列車之車身長度(m)

s_x =列車於車站停車時車頭與出發號誌機之距離(m)

B_n =出發號誌機後第一個閉塞區間長度(m)

B_{n-1} =出發號誌機後第二個閉塞區間長度(m)

s_{a_i} =先行列車之加速運轉距離

將 $s_{a_i} = \frac{v_i^2}{2a_i}$ 代入式(4.50)可得：

$$t_{v_i} = \frac{L_i + s_x + B_n + B_{n-1}}{v_i} - \frac{v_i}{2a_i} \quad (4.51)$$

將式(4.49)、式(4.51)整理可得號誌安全時距：

$$T_{s,D2} = \frac{v_i}{2a_i} + \frac{L_i + s_x + B_n + B_{n-1}}{v_i} + t_o + t_r \quad (4.52)$$

4.3.3.2 列車運行安全間距

移動自動之最主要目的乃在於確保列車行車安全之前提下，利用現今高新科技達到縮短列車運行間隔，提高軌道容量之目的。移動自動閉塞系統之內容主要在於其連續確認先行列車之位置速度，以進行列車之行車安全控制，以確保列車行車安全，避免追撞或是冒進區間。移動自動閉塞並無固定之區間，由於其是將前行列車尾部至本車車首視為閉塞區間，因此，其閉塞區間隨著兩車移動而移動，並且根據不同情況，例如前後車速度、或者是地理環境因素如上下坡等，使得車間之閉塞區間長度經由車載計算機計算後隨時改變，以即時反應列車運行狀況，確保軌道行車安全。

針對上述之構思，移動自動閉塞系統之列車間隔變顯得十分重要，一般在討論上，皆分為時間上以及空間上之列車間隔加以探討，兩者求取較大值作為實際移動自動閉塞系統之運行，以確保列車行車安全。由於固定自動閉塞至移動自動閉塞之改變過程中，可利用密集佈設軌道電路即連鎖列車通過信號機達到近似於移動自動閉塞系統之行車方式，因其仍有固定之軌側信號設施，因此並非完全屬於移動自動閉塞系統，但閉塞方式卻是類似於移動自

動閉塞系統之方式，因此將之對於列車安全間隔之探討列於本章節中，稱為「準移動閉塞系統」，這也是未來發展一定自動閉塞行車制過渡階段的可行方案。以下針對列車安全時間間隔以及列車安全空間間隔之計算決定方式加以探討，供情境分析未來施行移動自動閉塞系統之理論依據。

1. 列車運行安全時間間隔

軌道運輸之列車時間間隔乃是以連續兩列車之先行列車車首至後續追蹤列車車首(即前後列車之 headway)決定。由於列車之時間間隔會在車站區域得到相對於路線閉塞區間之較大值，因此基於安全之考量，以車站區域之列車時間間隔作為列車安全時間間隔之設定，以下針對各項符號加以定義及給予基礎之假設，作為分析列車安全時間間隔之基礎：

令研究範圍內有 M 個車站，第 i 個車站之列車間隔時間為 $t_i^i (i=1,2,\dots,M)$ ，則其列車時間間隔位：

$$T_i = \max\{t_i^1, t_i^2, \dots, t_i^M\}$$

車站列車時間間隔(t_i^i)包括以下兩部分：

$$t_i^i = t_w^i + t_H^i$$

$\begin{cases} t_w^i : \text{前行列車與後續追衝列車之安全間隔時間} \\ t_H^i : \text{列車在站內之停車時間，包括列車開關車門、完成上下客之時間} \end{cases}$

為方便起見，將列車安全時間間隔 t_w^i 簡寫為 t_w ，為使問題簡化，先作以下與實際情況相去不大之假設：

- 假設一：前後兩列車具有相同之啟動加速度 a ，相同之制動減速度 b ，區間最大允許速度為 V_{\max} 。
- 假設二：兩列車車長均為 L_z 。
- 假設三：軌道電路區段後端均設有安全防護區段 ΔL 。
- 假設四：兩列車之制動反應時間均為 t_{an} 。

利用以上之假設，針對固定自動閉塞系統、準移動閉塞、移動自動閉塞系統三者之列車時間間隔於以下篇幅探討個別之計算方式。

(1) 固定自動閉塞列車時間間隔 t_w 之計算：

固定自動閉塞系統其分區長度直接與列車容量、列車制動距離及列車最大允許運行速度有關，軌道電路僅用於列車位置檢測，不似準移動閉塞可用於訊息之傳輸，因此閉塞分區需用軌旁列車通過信號機作防護。

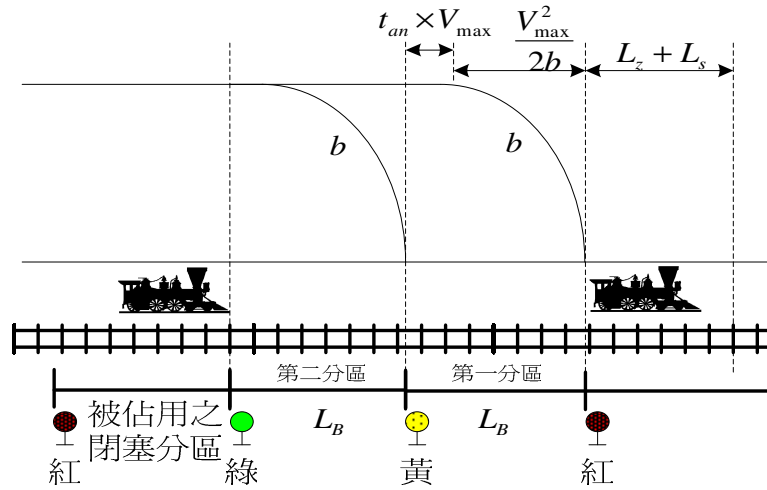


圖 4.8 固定自動閉塞列車安全時間間隔示意圖

圖 4.8 表示先行列車車站出清並駛過安全防護區段時，後行追蹤列車恰達第二分區之入口，閉塞分區之長度(L_B)為一般制動距離加上司機員反應時間內所行駛之距離：

$$L_B = t_{an} \times V_{\max} + \frac{V_{\max}^2}{2b}$$

列車之安全時間間隔由以下三部分組成：

- ①後續列車以速度 V_{\max} 行駛過第二分區 L_B 所需時間為 $t_{an} + \frac{V_{\max}}{2b}$
- ②後續列車在第一分區制動停車所需時間為： $t_{an} + \frac{V_{\max}}{b}$
- ③前行列車出清車站並駛過安全防護區段所須之時間 t_{WC} 為：

(i)當 $V_{\max} \geq \sqrt{2a(L_Z + L_S)}$ 時，以加速度 a 出清安全防護區段，則

$$t_{WC} = \sqrt{2(L_Z + L_S)/a}$$

(ii)當 $V_{\max} < \sqrt{2a(L_Z + L_S)}$ 時，以加速度 a 運行使達到 V_{\max} ，於此其間，運行距離為 $V_{\max}^2/2a$ ，運行時間為 V_{\max}/a ；接著列車以速度 V_{\max} 運行剩餘之距離： $(L_Z + L_S) - V_{\max}^2/2a$ ，運行時間為：

$$\frac{2a(L_Z + L_S) - V_{\max}^2}{2aV_{\max}}, \text{ 則}$$

$$t_{WC} = \frac{2a(L_Z + L_S) + V_{\max}^2}{2aV_{\max}}$$

由①②③可得，固定自動閉塞列車安全間隔時間為：

$$t_w = \begin{cases} \sqrt{\frac{2(L_Z + L_S)}{a}} + 2t_{an} + \frac{3V_{\max}}{2b} \dots\dots\dots V_{\max} \geq \sqrt{2a(L_Z + L_S)} \\ \frac{2a(L_Z + L_S) + V_{\max}^2}{2aV_{\max}} + 2t_{an} + \frac{3V_{\max}}{2b} \dots\dots\dots V_{\max} < \sqrt{2a(L_Z + L_S)} \end{cases} \quad (4.53)$$

(2) 準移動閉塞列車時間間隔 t_w 之計算：

由於準移動閉塞系統與固定自動閉塞系統之不同乃在於其將軌道電路單元細分，因此一個閉塞分區有可能會包括數個軌道單元長度，在此列車之間的間隔距離 L_x 以定位單元長度 L_A 的個數 N 表示，即：

$$L_x = NL_A$$

令 L 為後續追蹤時間所行駛的距離 L_{an} 和一般制動距離 L_b 之和，利用上述之推導：

$$L = L_{an} + L_b = t_{an} \times V_{\max} + \frac{V_{\max}^2}{2b}$$

當 $L/L_A = NT(L/L_A)$ 成立時，若列車營運停車點和定位單元分界處重疊，則 L 佔據 L_A 的個數： $N = NT(L/L_A)$ ；若無重疊，則 $N = 1 + NT(L/L_A)$ 。

當 $L/L_A = NT(L/L_A)$ 不成立時，若列車營運停車點和定位單元分界處重疊，則 L 佔據 L_A 的個數： $N = 1 + NT(L/L_A)$ ；反之若列車營運停車點和定位單元分界處分佈最不利的其況下， $N = 2 + NT(L/L_A)$

綜合以上所述，為不失其一般性，取最不利狀況下， N 應為：

$$N = 2 + NT(L/L_A)$$

準移動自動閉塞下列車安全間隔時間應包含圖 4.9 之三部份：

① 後續列車以 V_{\max} 均速行駛 $(L_x - L_b)$ 之時間為：

$$\frac{L_x - L_b}{V_{\max}} = \frac{L_A}{V_{\max}} \left[2 + NT\left(\frac{t_{an} V_{\max} + \frac{V_{\max}^2}{2b}}{L_A}\right) \right] - \frac{V_{\max}}{2b} \quad (4.54)$$

② 後續列車從 V_{\max} 到停車所須之時間其值為： $\frac{V_{\max}}{b}$

③ 前行列車出清車站並駛出安全防護區段所須之時間 t_{wc} ，其計算

公式如上所推導。

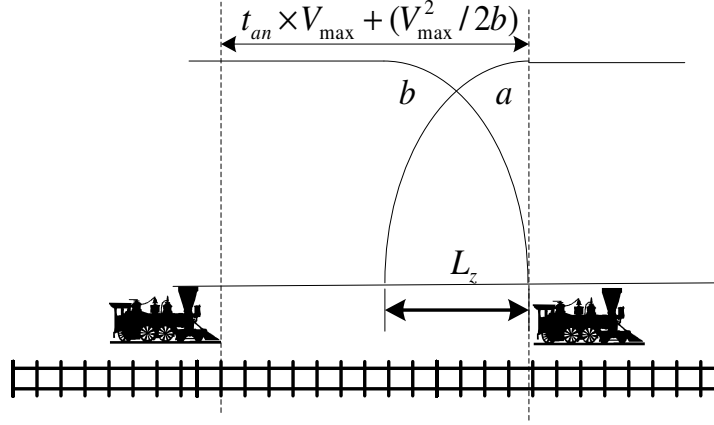


圖 4.9 準移動閉塞列車安全時間間隔示意圖

因此，綜合①②③可得其準移動閉塞列車安全時間間隔為：

$$tw = \left\{ \begin{array}{l} \frac{L_A}{V_{\max}} \left\{ 2 + NT \left[\frac{t_{an} V_{\max} + \frac{V_{\max}^2}{2b}}{L_A} \right] \right\} + \frac{V_{\max}}{2b} + \sqrt{\frac{2(L_Z + L_B)}{a}} \dots \dots V_{\max} \geq \sqrt{2a(L_Z + L_S)} \\ \frac{L_A}{V_{\max}} \left\{ 2 + NT \left[\frac{t_{an} V_{\max} + \frac{V_{\max}^2}{2b}}{L_A} \right] \right\} + \frac{V_{\max}}{2b} + \frac{2a(L_Z + L_S) + V_{\max}^2}{2aV_{\max}} \dots V_{\max} < \sqrt{2a(L_Z + L_S)} \end{array} \right\} \quad (4.55)$$

(3)移動自動閉塞列車時間間隔 t_w 之計算：

移動自動閉塞系統採用無線通訊以進行即時之定位及通訊，圖 4.10 表示前行列車剛出清車站並駛過安全防護區段 L_S ，後續追蹤列車以最大允許速度 V_{\max} 接近，其頭部至車站營運停車點的距離恰好等於後續追蹤列車之一般制動距離及在制動反應時間內以 V_{\max} 行走距離之和。其列車安全間隔時間 t_w 為：

$$t_w = t_{WC} + t_{an} + t_{WB}$$

t_{WB} ：列車從 V_{\max} 至停止所需時間，其公式為： V_{\max}/b 。

t_{an} ：由信號控制系統性能決定，包括車地訊息傳輸及處理時間，操作制動器的反應時間。

t_{WC} ：列車出清車站並駛過防護區段 L_S 所需校號的時間，與上述之計算推導方式相同。

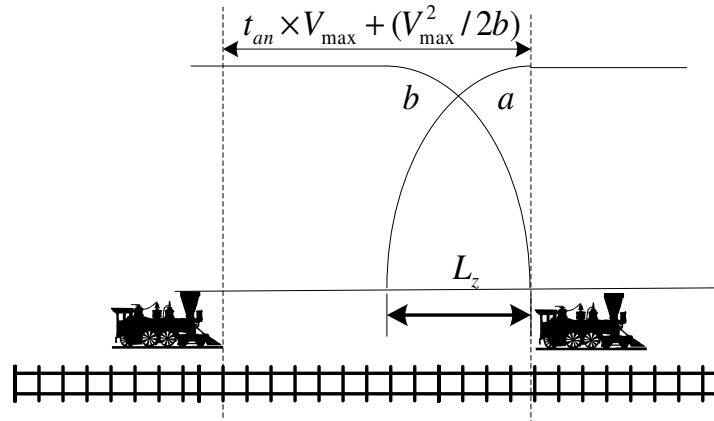


圖 4.10 移動自動閉塞列車安全時間間隔示意圖

由以上所述，可得移動自動閉塞系統之列車安全時間間隔 t_w 為：

$$t_w = \left\{ \begin{array}{l} \sqrt{\frac{2(L_z + L_s)}{a}} + t_{an} + \frac{V_{max}}{b} \dots \dots \dots V_{max} \geq \sqrt{2a(L_z + L_s)} \\ \frac{2a(L_z + L_s) + V_{max}^2}{2aV_{max}} + t_{an} + \frac{V_{max}}{b} \dots \dots \dots V_{max} < \sqrt{2a(L_z + L_s)} \end{array} \right\} \quad (4.56)$$

就移動自動閉塞系統與固定自動閉塞系統相比，不論何種情況 ($V_{max} \geq \sqrt{2a(L_z + L_s)}$ 或 $V_{max} < \sqrt{2a(L_z + L_s)}$)，自動閉塞系統之列車安全時間間隔皆比固定自動閉塞系統少 $t_{an} + \frac{V_{max}}{2b}$ 。

2.列車運行安全時間間隔

列車安全空間間隔之決定方式可以結合行車動力學之物理公式表現：

$$L = V_2 \left(\frac{V_2}{2\beta_2} + \tau_2 \right) - V_1 \left(\frac{V_2}{2\beta_1} + \tau_1 \right) + \Delta L \quad (4.57)$$

$$\left\{ \begin{array}{l} L : \text{先行列車與後續追蹤列車之最小空間間隔} \\ V_1、V_2 : \text{先行列車與追蹤列車之速度} \\ \beta_1、\beta_2 : \text{先行列車與追蹤列車之減速度} \\ \tau_1、\tau_2 : \text{先行列車與追蹤列車之空走時間} \\ \Delta L : \text{停車充裕間隔} \end{array} \right.$$

以上為基本之列車運行間隔計算方式，但在實務上，列車之安全空間間隔又可依照其分為①考慮先行列車速度之準移動閉塞系統(SMB-V)、②考慮先行列車速度之移動自動閉塞系統(MB-V)、及③不考慮先行列車速度之移動自動閉塞系統(MB-V₀)等三類，由基礎之列車運行間隔計算方式改良而得其列車安全運行間隔。

(1)考慮先行列車速度的準移動閉塞系統(MB-V₀)：將固定自動閉塞系統之閉塞區間(即軌道電路長度及相對應之軌側列車通過信號機)加以細分，使得在既有之固定自動閉塞系統功能上升級，得以考慮先行列車速度，進行列車間隔控制，當軌道電路劃分甚細，其縮短列車運行間隔之效果將近似於移動自動閉塞系統，因此稱為準移動閉塞系統(Semi Moving Block)，在此後續列車制動減速度 β_2 採一般制動減速度，前行列車減速度 β_1 採緊急制動減速度。其列車間格示意如圖 4.11 (假設 $V_1=V_2$)：

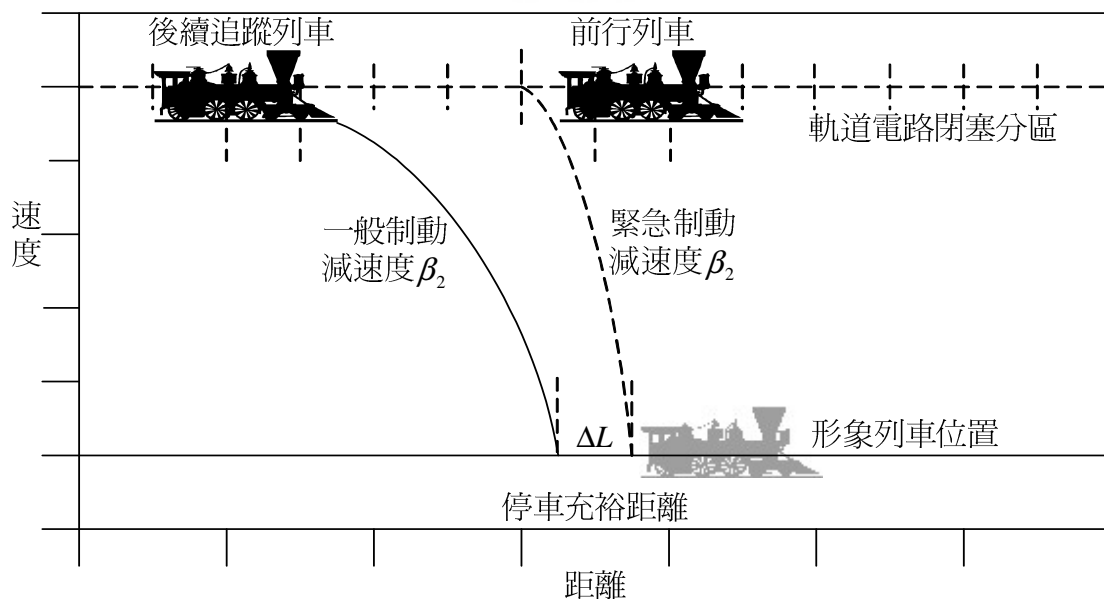


圖 4.11 SMB-V 方式移動閉塞列車間隔示意圖

(2)考慮先行列車速的移動自動閉塞系統(MB-V)：根據先行列車的速度、位置作為列車間隔控制之考量，以 β_1 為先行列車之緊急制動減速度， β_2 為後續追蹤列車之一般常用制動減速度，使得在後續列車在先行列車採取緊急制動減速度煞車時，得以以一般常用制動減速度安全停車之系統，其列車間隔示意如圖 4.12 (假設 $V_1=V_2$)：

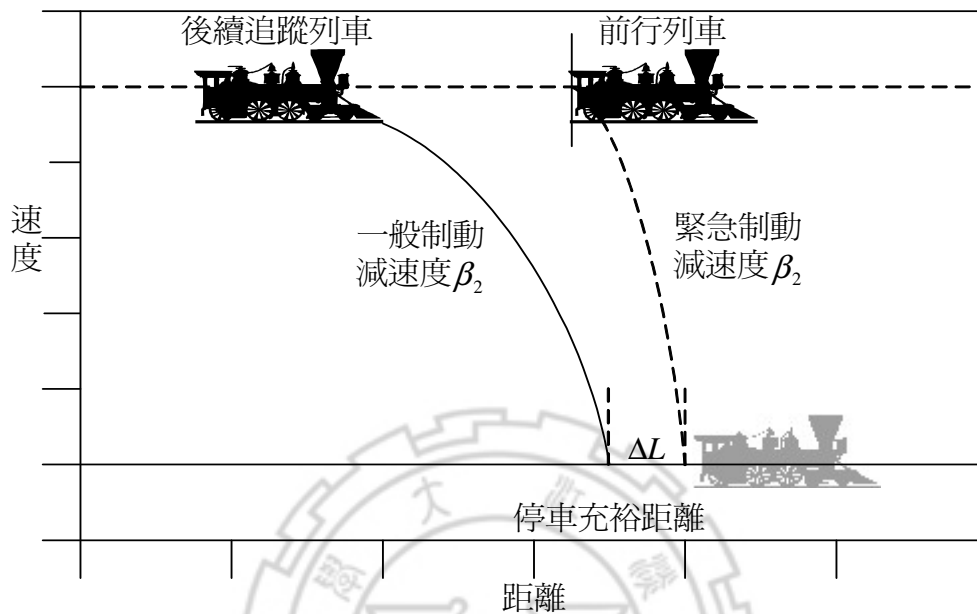


圖 4.12 MB-V 方式移動自動閉塞列車間隔示意圖

(3)不考慮先行列車速度的移動自動閉塞系統(MB- V_0)：不考慮先行列車速度，僅考慮先行列車位置，即將安全列車空間間隔基礎公式中之先行列車減速度 β_1 視為無限大，先行列車空走距離為 0，因此 $V_1(\frac{V_2}{2\beta_1} + \tau_1) = 0$ ，即將先行列車視為立即停車情況下，後續追蹤列車得以安全完成制動停車於目標停車點之外，其列車間隔示意如圖 4.13 所示：

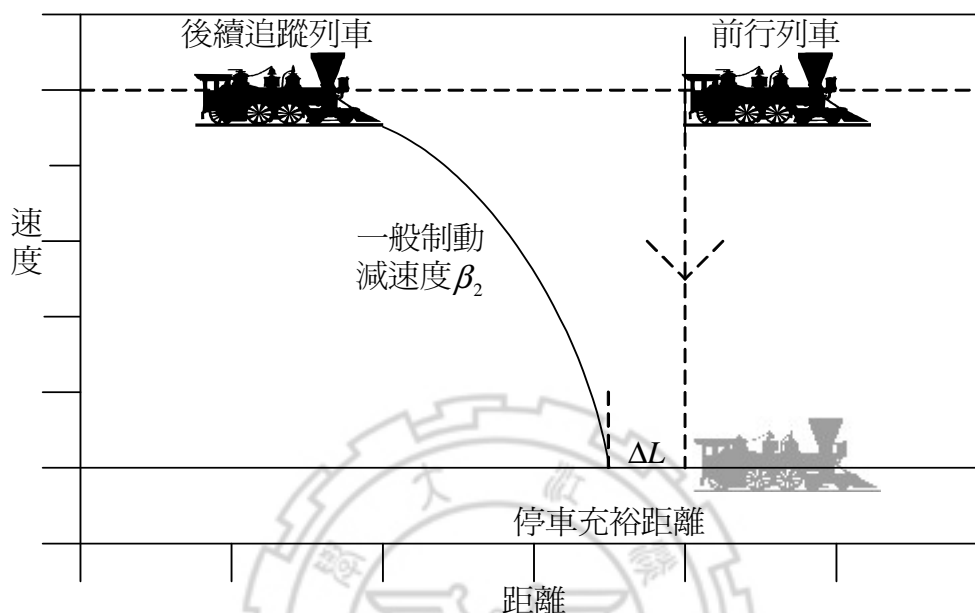


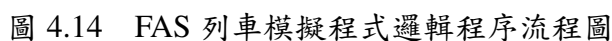
圖 4.13 MB- V_0 方式移動自動閉塞停車間隔示意圖

4.3.4 小結

移動自動閉塞之最主要目的乃在於確保列車行車安全之前提下，利用現今高新科技達到縮短列車運行間隔，提高軌道容量之目的。移動自動閉塞系統之內容主要在於其連續確認先行列車之位置速度，以進行列車之行車安全控制，以確保列車行車安全，避免追撞或是冒進區間。移動自動閉塞並無固定之區間，由於其是將前行列車尾部至本車車首視為閉塞區間，因此，其閉塞區間隨著兩車移動而移動，並且根據不同情況，例如前後車速度、或者是地理環境因素如上下坡等，使得車間之閉塞區間長度經由車載計算機計算後隨時改變，以即時反應列車運行狀況，確保軌道行車安全。

如上所述，移動自動閉塞系統之列車間隔變顯得十分重要，一般在討論上，皆分為時間上以及空間上之列車間隔加以探討，兩者求取較大值作為實際移動自動閉塞系統之運行，以確保列車行車安全。固定自動閉塞至移動自動閉塞之改變過程中，可利用密集佈設軌道電路及連鎖列車通過信號機達到近似於移動自動閉塞系統之行車方式，由於其仍有固定之軌側信號設施，因此並非完全屬於移動自動閉塞系統，但其閉塞方式卻是類似於移動自動閉塞系統之方式，因此可稱為「準移動閉塞系統」。列車安全時間間隔以及列車

根據前述章節計算公式，本研究之 FAS 與 MAS 模擬程式邏輯與流程架構分別如圖 4.14 及圖 4.15 所示。



「移動自動閉塞行車制」與傳統「固定自動閉塞行車制」最大差異在於其避塞區間之長度。在「固定自動閉塞行車制」中，其避塞區間長度多介於1.5~2.5km之間，無法對佔用區間的列車進行精確定位，使得軌道容量浪費，因此而有引進先進通訊、定位技術之「移動自動閉塞行車制」概念。

固定避塞區間之目的在於避免列車發生追撞危險，保持列車行車安全間距，因此「移動自動閉塞行車制」首重列車安全空間間距，經由採用先進通訊、定位技術，即時掌握前後車速度、位置，使得在安全的前提下，能有效地縮小列車運行安全空間間距，達到提升軌道容量之目的。

相關「移動自動閉塞行車制」之模擬程式流程圖如圖 4.15 所示，與「固定自動閉塞行車制」最大不同在於列車安全空間間距門檻值之決定，考慮前後車速度、制動性能來追蹤先行列車，目標為前後車制動至停止時，兩車仍保持一個車長之安全間距，做為停車餘裕空間。



第五章 實證分析

本研究構建出之模擬模式需經驗證流程，方能進行後續課題之探討。模擬結果須與理論值(即時刻表)或實際列車運行實際值經由統計檢定進行驗證之後，始能宣稱模擬模式可代表真實世界；然後進行敏感度分析，以確認各影響變數對於軌道容量之影響程度，最後進行情境模擬，在「固定自動閉塞」與「移動自動閉塞」兩種行車制下，檢視不同車種組成、不同發車間隔下所產生的軌道容量變化值並進行比較。

本研究所蒐集之資料為直接及間接資料，由實際調查及出版品取得，研究驗證資料採平日星期一至星期四列車實際運行資訊，擷取下午一點三十分至三點三十分離峰時刻列車運行資料作為驗證對象，以避免尖峰時刻列車延誤所產生之連鎖影響。

5.1 模擬程式簡介

本模擬程式使用軟體為 Dev C++ 4.9.9.2，為一綠色軟體，無侵權之虞。程式碼如附錄二。本模擬程式乃是針對西正線下行松山-新竹研究路段，多車種單股道混跑，容許高等級列車超越低等級列車，錯車待避行為發生於車站內，考慮進出站號誌對列車運行之影響，並且將坡度、曲線對列車加減速及速限納入考慮。相關設定介紹如下：

5.1.1 模擬程式模組介紹

本列車模擬程式主要包括以下模組：

- 1、外部資料輸入模組：讀入外部基本列車運行資訊。
- 2、包絡線模組：產生速限包絡線、停站包絡線，以控制列車運行速度。
- 3、錯車判斷控制模組：向後掃描列車決定是否採取待避動作。
- 4、安全間距控制模組：控制列車於區間運行空間間距，避免冒進追撞。
- 5、速度控制模組：控制列車採取加速、等速、減速動作。
- 6、列車推進模組：列車加減速推進、計算運行里程、速度。
- 7、結果輸出模組：計算列車運行時間、軌道容量，紀錄各列車進出站時間以產出列車運行圖。

5.1.2 輸入輸出資訊

本模擬程式輸入資訊可分為以下兩部份：交通條件、線性特徵。

交通組成部份：輸入列車組成、列車任務、發車時刻、停站時間分配、發車間距分配。

線性特徵部份：路線坡度、路線曲線半徑、路段速限、站位分布。

輸出資訊：經由模擬程式運算後，輸出各列車進出各站時間記錄、停站時間，

列車進入及離開時間、各車種組成最大運轉時隔及其累計次數及容量值。

5.2 模擬模式驗證

模擬模式之驗證主要分為兩部份：程式驗證(Verification)及模式驗證(Validation)。

本研究係針對容量計算相關之參數值進行校估，並經由統計分析，找出列車發車間距以及各車種於研究區段停站之時間分配，目的為確保本模擬程式足以表現邏輯架構合理。

模式為驗證模擬結果與真實世界之關係，是否能夠充分地描繪真實世界列車運行，以說明模擬模式對於真實世界之代表性。程式驗證在於驗證模擬程式內容是否與模式相符，模式驗證在於驗證模式是否與真實世界相符，經由模擬與實際值比較，結果與現實先去不遠，證明此模擬程式可行，相關之統計檢定說明於後。

首先針對列車發車時間間隔及各等級列車之停站時間分配進行檢定。檢定之工具採用 BestFit 最適分配函數軟體，進行各資料適合度檢定，以找出符合資料散布情形之分配。以表 5.1 說明檢定結果：

表 5.1 發車 headway 及停站時間統計分配一覽表

發車 headway 統計分配			
松山站		Weibull	
停站時間分配			
站別	車種	停站統計分配	站名
大站	自強	Weibull	台北、新竹
	莒光	Normal	
	復興	Weibull	
中站	自強	Exponential	松山、板橋、樹林、桃園、中壢
	莒光	Exponential	
	復興	Exponential	
小站	自強	Exponential	萬華、山佳、鶯歌、內壢、埔心、楊梅、富岡、湖口、新豐、竹北
	莒光	ExtValue	
	復興	ExtValue	
不分站	電車	Exponential	全線共十七站

資料來源：本研究整理

列車發車 headway 資料採實際調查取得，於一般日星期一至星期四上午十時至下午四時連續調查四天所得結果；列車停站時間則擷取台鐵綜合調度所記錄之實際列車運行及停站時間進行檢定。由於皆是實際運行之資料，因此可確保資料可用。

在模擬現行固定自動閉塞行車方式下，採用 2006 年 3 月 15 日改點後班表資料，下午一點半至下午三點半發車之列車運行。熱機切除時間由於台北-新竹段長約 84km，任一車種大約可在 90 分鐘內跑完全程，因此設定熱機切除時間為一個半小時，以擷取系統達穩定狀態之資訊。各列車發車及模擬結束時間如圖 5.1 所示，由圖中可看出，各列

車運行模擬時間與時刻表理論時間、實際運行時間之差異大部分皆在 3 分鐘之內，為台鐵列車運轉可容忍準點範圍內，證明模擬模式可行。

驗證是採 99%信心水準雙尾 t 檢定，變異數為 251.9657，樣本數為 30，檢定模擬之平均值與母體平均值是否有顯著差異。

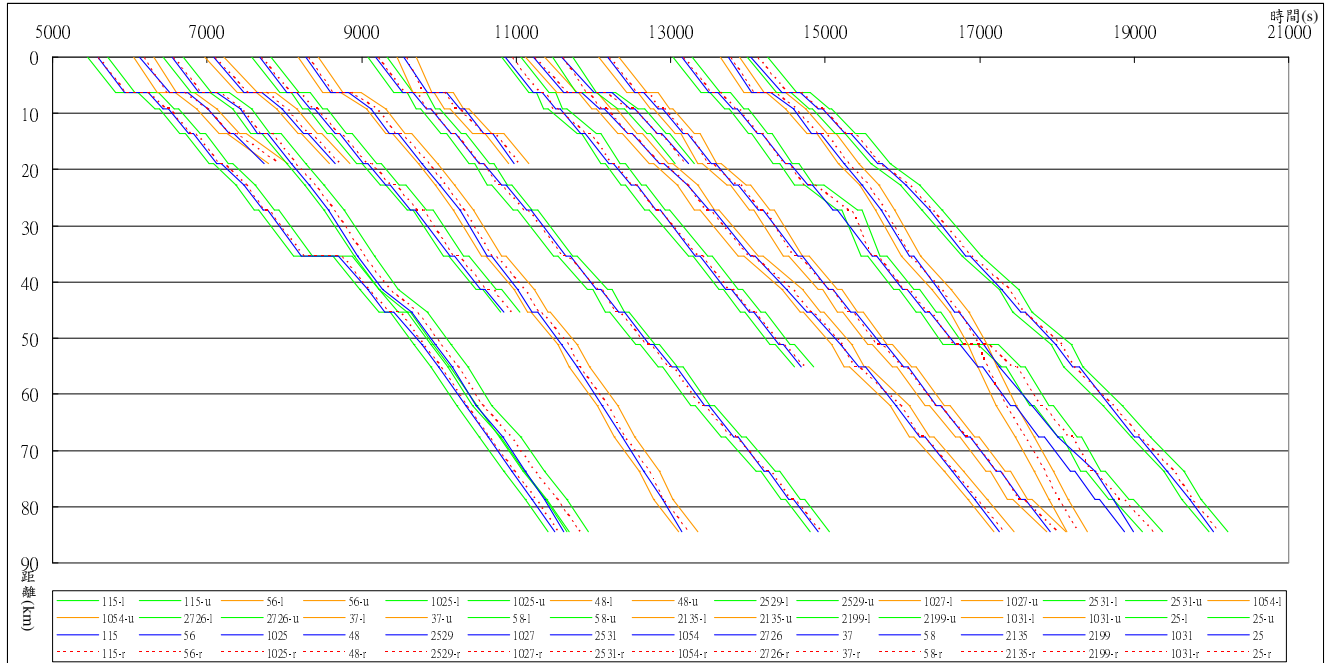


圖 5.1 模擬模式驗證

(圖例說明：115-l 表示實際值之信賴區間下限, 115-u 則為上限, 115 代表模擬列車運行線, 115[藍色實線]為模擬之列車運行線, 115-r[紅色虛線] 為實際列車運行線)

針對模擬模式產出進行檢定，結果如表 5.2：

表 5.2 模擬模式驗證 T-value 表(I-1)

T-Value	松山		台北		萬華		板橋		樹林		山佳		鶯歌		桃園	
	0		6.358		9.191		13.578		18.986		22.867		27.259		35.456	
車次	out	in	out	in	out	in	out	in	out	in	out	in	out	in	out	in
115	-0.1304	-0.2174	-0.1956	-0.3478	-0.3913	-0.3913	-0.6087	-0.3695	-0.3695	-0.4565	-0.4565	-0.2391	-0.2391	-0.3478	-0.9347	
56	-1.3043	0.8695	0.9999	0.4130	0.4130	0.1304	-0.6087	-4.0867								
1025	-0.4348	-0.6087	-0.6521	-0.5652	-0.5652	-0.6521	-1.3043	-1.4130	-1.4130	-1.4999	-1.4999	-1.5651	-1.5651	-1.7390	-1.7390	
48	-0.3261	-0.9782	-1.0869	-1.1521	-1.1521	-0.9999	-0.8043	-1.1304								
2529	-0.4348	-0.6956	-1.2173	-0.6521	-0.6956	-0.5869	-0.6521	-0.9565	-0.9999	-1.1086	-1.4782	-1.6738	-2.1086	-0.9130	-1.1739	
1027	-0.5655	-0.7595	-0.6069	-1.3213	-1.3213	-1.7011	-0.9992	-1.6707	-1.6707	-1.3199	-1.3199	-0.9695	-0.9695	-1.3015	-1.2352	
2531	-0.5434	-0.0713	-1.1609	0.0973	0.0973	-0.1304	-0.1399	-0.0295	-0.0295	1.3031	1.3031	1.0547	1.0547	0.8013	0.8013	
1054	-0.8695	1.3912	0.6521	1.7608	1.6086	0.3695	-0.4565	-1.1956								

由圖 5.1 可看出，列車運行經熱機切除後之結果，大多通過顯著性檢定，表示模擬結果與列車實際運行情形無顯著差異。僅車次 56 莒光號在停靠列車終點樹林站時，與實際值比較則早著 188 秒，與台鐵可容忍列車誤點 5 分鐘之標準相比，仍在可接受範圍

之內。

表 5.3 模擬模式驗證 T-value 表(I-2)

T-Value	內壢		中壢		埔心		楊梅		富岡		湖口		新豐		竹北		新竹
e	41.395		45.345		51.141		55.192		62		67.682		73.815		78.625		84.475
車次	in	out	in	out	in	out	in	out	in	out	in	out	in	out	in	out	in
115	-0.5652	-0.5652	-1.0652	-1.7173	-0.5000	-0.5000	-0.3913	-0.3913	-0.5000	-0.5000	-0.5217	-0.5217	-0.4348	-0.4348	-0.9347	-0.9347	-0.9999
56																	
1025	-1.8695	-1.8695	-1.6303	-1.6303	-1.5434	-1.5434	-1.6956	-1.6956	-1.8260	-1.8260	-2.1521	-2.1521	-2.4999	-2.4999	-3.5216	-3.5216	-4.1954
48																	
2529	-1.5651	-2.0216	-1.8912														
1027	-2.0431	-2.0431	-1.4066	-1.0016	-1.7525	-1.7525	-0.6619	-0.6619	-1.7150	-1.7150	-0.6500	-0.6500	-1.5156	-1.5156	-0.1605	-0.1605	-1.6655
2531	-0.3193	-0.3193	1.2482	1.2482	1.0954	1.0954	1.0218	1.0218	1.2444	1.2444	0.6130	0.6130	-0.5749	-0.5749	-0.3696	-0.3696	-0.4148
1054																	

進行檢定後，從表 5.3 中可看出除了 1025 自強號外，其餘列車皆符合列車實際運行調查結果，t-value 除 1025 車次接近終點站時位於拒絕域，其餘檢核點皆無法拒絕模擬值與實測資料有顯著差異，而 1025 車次位於拒絕域之原因乃是由於累積各站列車延誤致使得列車在終點站附近出現顯著差異，模擬值與實際值差距為竹北站早著 162 秒，新竹站早著 193 秒，但與台鐵可容忍列車誤點 5 分鐘之標準相比，皆仍在可接受範圍之內。

表 5.4 模擬模式驗證 T-value 表(II-1)

T-Value	松山	台北		萬華		板橋		樹林		山佳		鶯歌		桃園	
	0	6.358		9.191		13.578		18.986		22.867		27.259		35.456	
車次	out	in	out	in	out	in	out	in	out	in	out	in	out	in	out
2726	-1.7489	-1.6976	-1.1056	0.9309	0.9309	-1.6883	-1.6883	-0.4159	-0.4159	0.1353	0.1353	0.4598	0.4598	-0.6643	-0.6643
37	-0.4565	-0.9357	-1.6099	-0.4706	-0.6228	-0.0324	-0.0324	-0.0199	0.0888	-0.0133	-0.0133	1.2755	1.1668	0.7665	0.6579
58	-0.4782	2.0651	1.7608	0.4565	-0.1304	0.3043	0.5434	1.1521							
2135	-0.3478	-0.5666	-1.1318	-0.5036	-0.2645	-0.6388	-0.8562	1.1343	1.0691	-0.6868	-0.5564	-0.4178	-0.2656	1.2844	0.9584
2199	-0.3478	-0.4565	-0.5434	-0.6087	-0.4782	-0.2826	-0.1087	0.6521	0.1739	1.1086	-0.4565	-4.1520	-4.1737	0.3478	-0.3043
1031	-0.6254	-0.5506	-1.4553	-0.5763	-0.5763	-1.6956	-1.7105	-1.2578	-1.2578	-2.0056	-2.0056	-0.9048	-0.9048	-0.5375	0.7668
25	-1.8034	-1.3497	-0.1002	-0.3695	0.0217	0.0818	-1.3529	-0.8701	0.4342	-1.0309	-1.0309	-0.5013	-0.5013	-1.8657	-0.5614

由圖 5.1 可看出，2199 車次電聯車之模擬列車運行線大致上趨勢與實際值相仿，惟在鶯歌站早著 191 秒，早發 192 秒，但與台鐵可容忍列車誤點 5 分鐘之標準相比，皆仍在可接受範圍之內。

表 5.5 模擬模式驗證 T-value 表(II-2)

T-Value	內壢		中壢		埔心		楊梅		富岡		湖口		新豐		竹北		新竹
	41.395		45.345		51.141		55.192		62		67.682		73.815		78.625		84.475
車次	in	out	in	out	in	out	in	out	in	out	in	out	in	out	in	out	in
2726	-0.5989	-0.5989	-0.4306	-0.4306	-0.2436	-0.2436	-0.7255										
37	-1.4036	-1.4036	-1.0811	-0.7971	-0.2515	-0.2515	1.1509	1.8030	-1.1095	-1.1095	0.1538	0.1538	-0.3285	-0.3285	-0.6419	-0.6419	-1.3114
58																	
2135	0.1135	0.1135	1.0047	0.7874	1.0907	0.9385	0.4318	0.4752	0.5149	0.4062	-0.0425	0.0662	-0.0099	-0.0099	0.9481	0.8394	-1.7262
2199	-0.3913	-0.6956	-0.2174	-0.1739	0.5434	-7.9344	-9.4560	-9.4560	-8.1083	-7.9996	-7.9778	-7.9344	-5.8040	-5.8910	-6.6736	-6.6083	-7.8692
1031	-0.6011	-0.6011	-1.3030	0.0013	1.5881	1.5881	3.3272	3.3272	7.2400	7.2400	9.4138	9.4138	14.3701	14.3701	15.5439	15.5439	15.7178
25	-2.1730	-2.1730	-0.6162	0.6881	-1.7393	-1.7393	-0.2091	1.7473	-0.6445	-0.6445	-1.7878	-0.4835	-2.0317	-2.0317	-0.6613	-0.6613	-1.4304

由圖 5.1 可看出，2199 車次電聯車與 1031 自強號之模擬列車運行線從埔心站後開始與實際列車運行線可接受區間分道揚鑣。實際之列車運行 2199 列車於埔心站停站分鐘，與預訂計劃不同，追究其原因為待避對向 1022 北上自強號，也因為模擬中 2199 車並未在埔心站延長停站時間，致使同向後續追蹤列車 1031 自強無法越行，造成此兩列車埔心站後之模擬列車運行線與實際列車運行情況不符。



5.2.1 小結

由以上各分段分析可看出，56 次及 1025 次列車由於停站延誤，使得累積誤差至較接近終點站處，模擬值落於信賴區間之外；2199 次於鶯歌站產生誤差，但其早到時間與實際運轉相去不遠，此推測為停站時間隨機變數所導致。

經由上述整理各列車運轉情形與模擬模式兩相比較，可看出模擬模式無法描述 2199 及 1031 車次在埔心站及其後的運行情況，經實際調查後發現，其原因是 2199 列車除了在埔心站須待避同向 1031 自強號外，也需待避對象交會列車 1022 車次，使得停站時間拉長，導致後續兩列車運行大受影響，到站順序也與實際值不符。

經由熱機切除前後各一個半小時所得之資訊，共 15 個車次，完全通過的檢定共有 11 個車次，進一步於各站細部檢核，總共 330 個檢核點，大致上列車之模擬運行情況良好，可利用本模擬模式描述大部分之列車運行，僅 29 個檢核點之模擬列車運行情形落於信賴區間之外，而此 29 個檢核點模擬值與實際值相比，僅有 2199 電聯車及 1031 自強號在埔心站後之列車運行模擬無法滿足台鐵可容忍列車誤點範圍，探究其發生之原因乃在於列車須待避對向 1022 自強號，而此待避對向列車之行為並非本研究納入考量之因素，因此判定模擬模式有效，足可描述真實世界列車之運行。



5.3 敏感度分析

經由上述章節驗證模式可行後，進行敏感度分析以找出各影響因子對於容量之改變程度。在此選定松山至新竹段中，以板橋站為觀測基準點，因各列車之列車任務緣故，致使得列車未達山佳站及其後之車站可能早已離開系統完成服務，使得運轉時隔徒增，無法反映現實情形，因此選定板橋站為觀測間準點。

本章節探討各重要容量分析變數對容量結果之敏感度，在其他條件不變情況下，探討各變數變動對容量值之影響。以下章節將針對停站時間、列車加減速度、列車長度、運轉寬裕時間、司機員與軀機反應時間、不同列車組成…等因素，探討其對於軌道容量之影響。

5.3.1 停站時間

由於台鐵列車運行有多車種混跑的特性，各車種對於同一站之停站時間也可能有異，因此在進行相關敏感度分析時，對於停站時間增量，不宜以增加百分比率停站時間的方式，而應以共同延長相同時間增量進行敏感度分析，以描述旅客多寡及上下車所產生之停站時間延誤。

停站時間直接影響到後續列車同股道或不同股道安全進站時隔，以及與先行列車同股道或不同股道安全離站時隔，因此，對於容量計算定義通過觀測點之列車數中之列車時隔將有絕對的影響，進而改變軌道容量。停站時間愈久，則軌道容量愈小，關係圖如圖 5.2 所示。

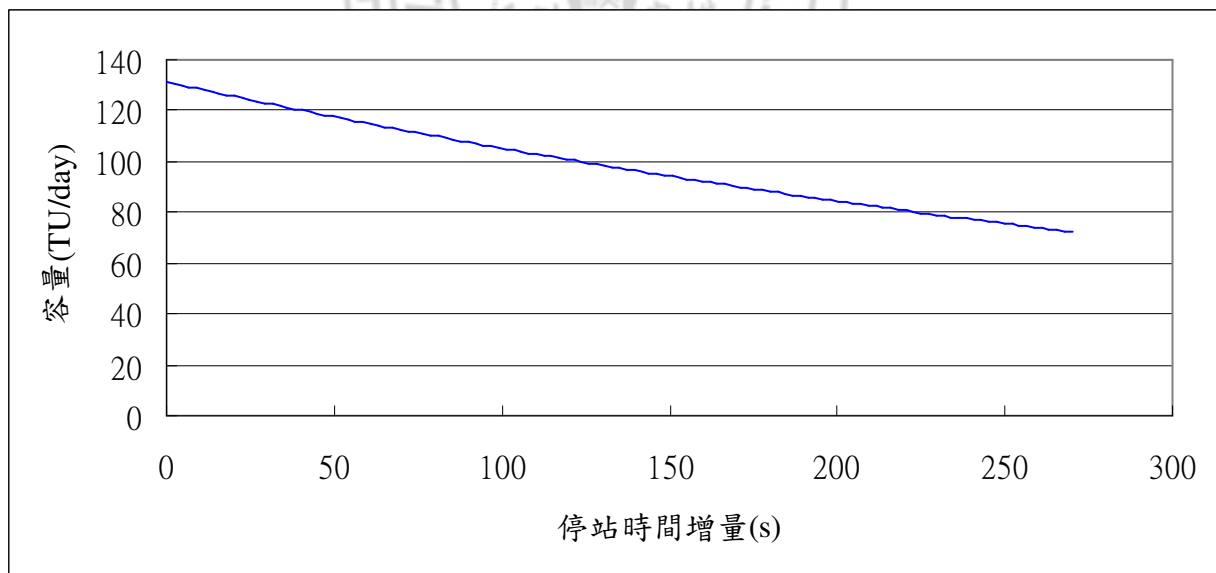


圖 5.2 停站時間增量與容量關係圖

停站時間增量之計算為直接在列車任務排定停站之停站時間延長，但原定不停站之列車任務則維持停站時間=0 秒。由圖 5.2 可知，停站時間與容量呈現負相關，且為非線性相關，相關係數為-0.94816，呈高度負相關。

5.3.2 列車加減速性能因子

列車加減速性能因子代表當司機員給予列車加減速指令時，列車能夠實際反應多少程度的實際加減速度。列車加減速性能因子與列車站間運行、進出站時間，有絕對相關，使得列車可能需花較長時間方能達到列車最大營運速度，甚或未達到列車最大營運速度即須進行減速，相關列車加減速度性能因子與容量之關係圖如圖 5.3 所示。

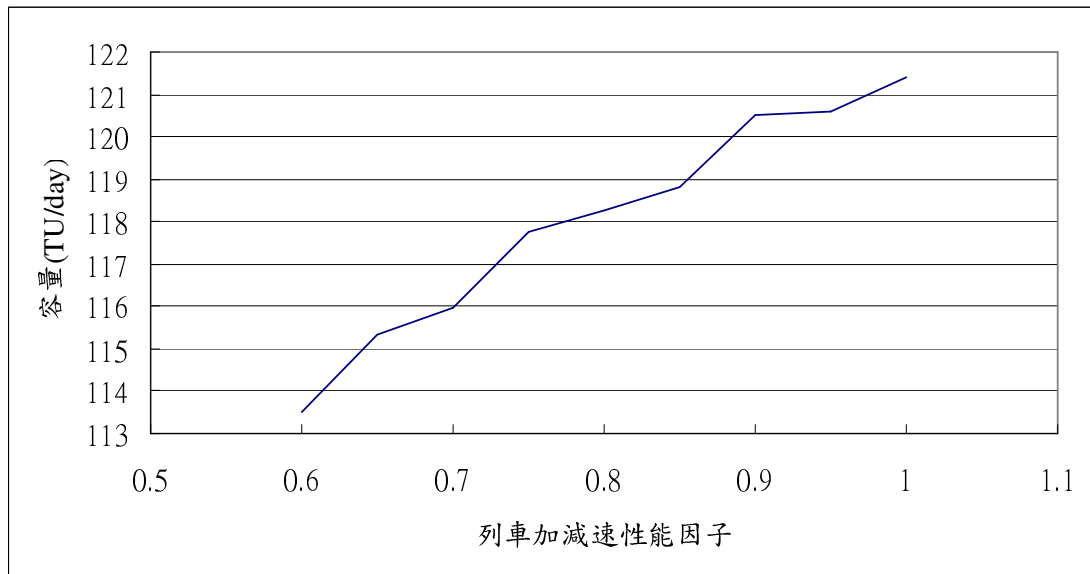


圖 5.3 列車加減速性能因子與容量關係圖

由圖 5.3 可知，列車加減速性能因子與容量呈現正相關，且為非線性相關，相關係數為 0.98502。

5.3.3 運轉寬裕時間係數

運轉寬裕時間愈大，則列車運轉時隔增加，使得軌道列車容量下降，通常運轉寬裕時間由最小運轉時隔決定，因此以運轉寬裕時間係數表現，如圖 5.4 所示。

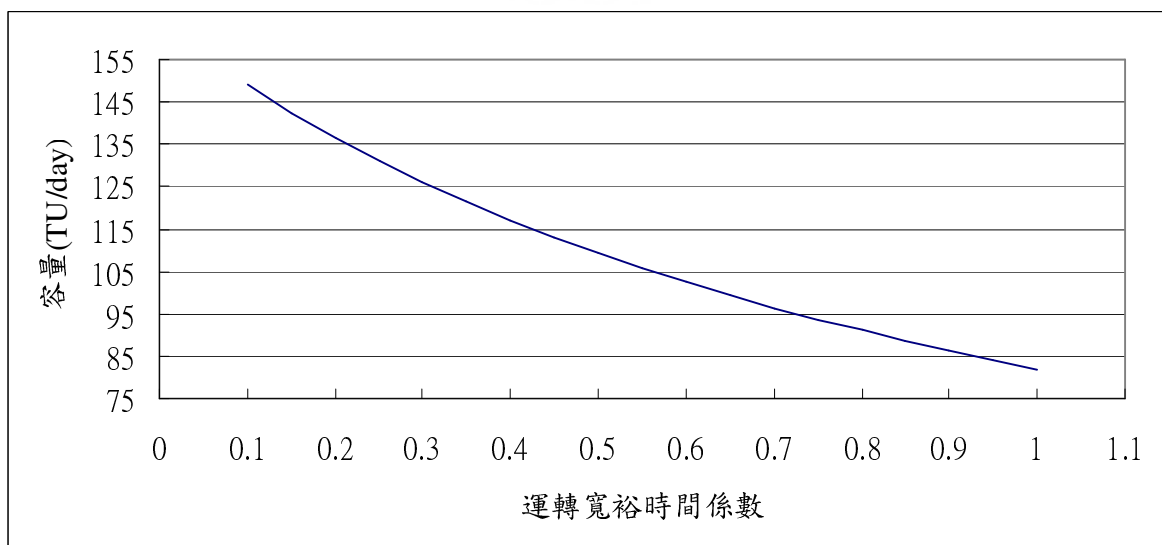


圖 5.4 運轉寬裕時間係數與容量關係圖

由圖 5.4 可知，運轉寬裕時間係數與容量呈現負相關，且為非線性相關，相關係數為-0.98135，可由此看出，本研究採取 0.35 運轉寬裕時間係數時與實際列車運行軌道容量每日約 122(TU/day)相符合。

5.3.4 運行列車車種組成

不同交通組成對於軌道影響甚鉅，台鐵軌道運輸為混合車種運轉，這也是台鐵容量最明顯浪費之處，由於初始列車排點乃是先從高等級列車著手，在高等級列車之間，若有餘裕再指派低等級列車於其間，因此使得容易犧牲零碎的軌道容量。

若依原始班表發車間距進行模擬，由於存在有餘裕列車運行空間，使得列車車種對於容量之反應不甚明顯，因此在此探討之運行列車車種組成為明顯看出車種對於軌道容量之影響，因此僅列車停站任務與原始班表相同，改變運行車種並且採取每隔 180 秒密集發車，此密集發車間隔乃參考 08：24 發車之 1099 自強號與 08：27 發車之 2712 電聯車發車間隔，這也是列車運行計畫中松山站一日最密集發車間隔。

圖 5.5 與表 5.6 為各種列車組成所得之軌道容量，在單一車種組成下，以莒光號所能產生的軌道容量為最大，軌道容量為 299.165(TU/day)；而車種組成包含兩個車種的情形則以電聯車搭配莒光號所產生的軌道容量 232.727(TU/day)為最大。

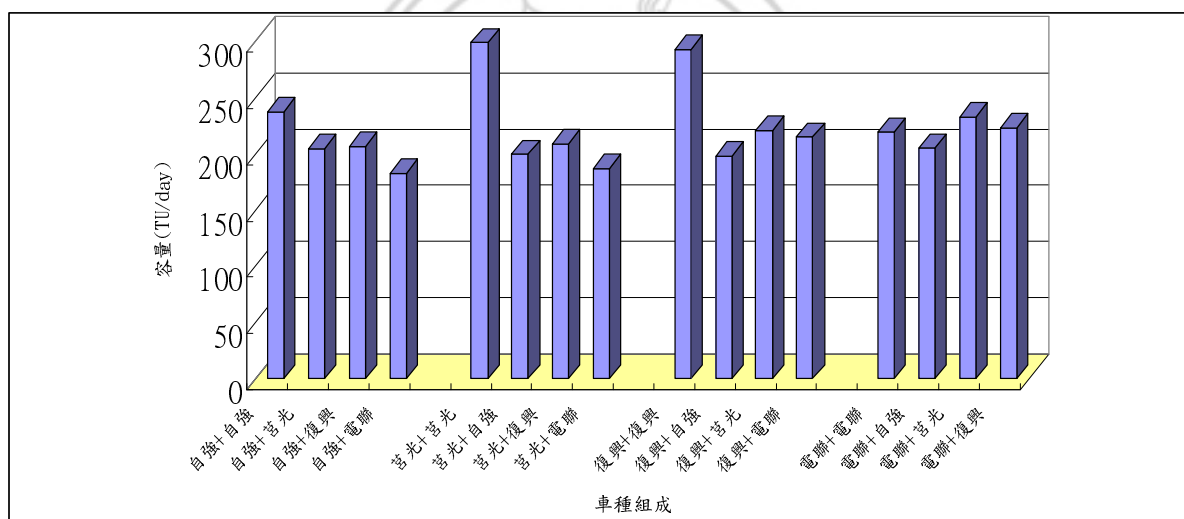


圖 5.5 列車組成與容量關係圖

表 5.6 列車組成與容量關係

車種組成	capacity	車種組成	capacity
自強+自強	237.351	自強+莒光	204.8
		自強+復興	206.69
		自強+電聯	182.485
莒光+莒光	299.165	莒光+自強	199.777
		莒光+復興	209.346
		莒光+電聯	187.252
復興+復興	292.81	復興+自強	198.23
		復興+莒光	220.69

		復興+電聯	214.868
		電聯+自強	205.034
電聯+電聯	219.339	電聯+莒光	232.727
		電聯+復興	223.441

資料來源：本研究整理

5.3.5 列車誤點及延滯

1、列車誤點

誤點定義：各列車於各站平均模擬至各檢核點(車站)時間與預期理論至各檢核點時間時間差。

$$\text{公式：}\Phi = \sum_{i=1}^{train_no} \frac{\sum_{j=1}^{station_no} X' - \bar{X}}{station_no}$$

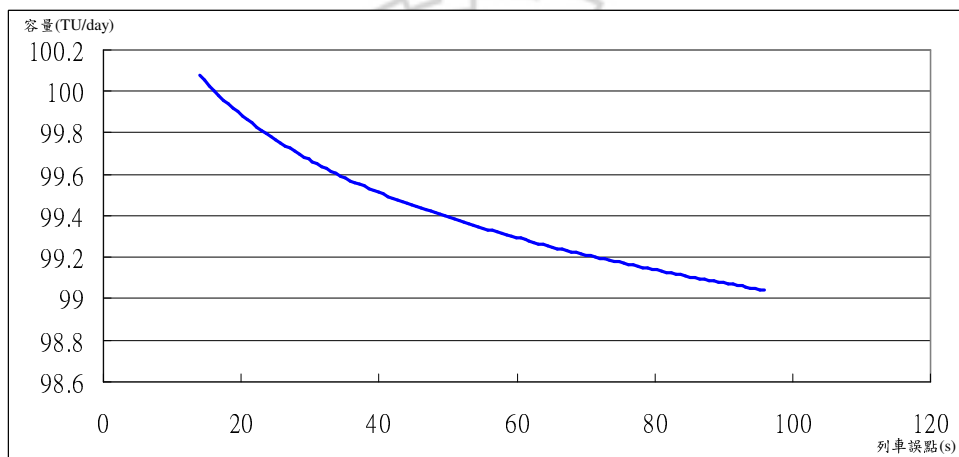


圖 5.6 列車誤點與容量關係圖

由圖 5.6 可知，列車誤點與容量呈現負相關，且為非線性相關，相關係數為 -0.934576746。

2、列車延滯

延滯定義：最後一部觀察列車模擬旅行時間與預期理論時間之差

$$\text{公式：}\phi = travel_time' - travel_time$$

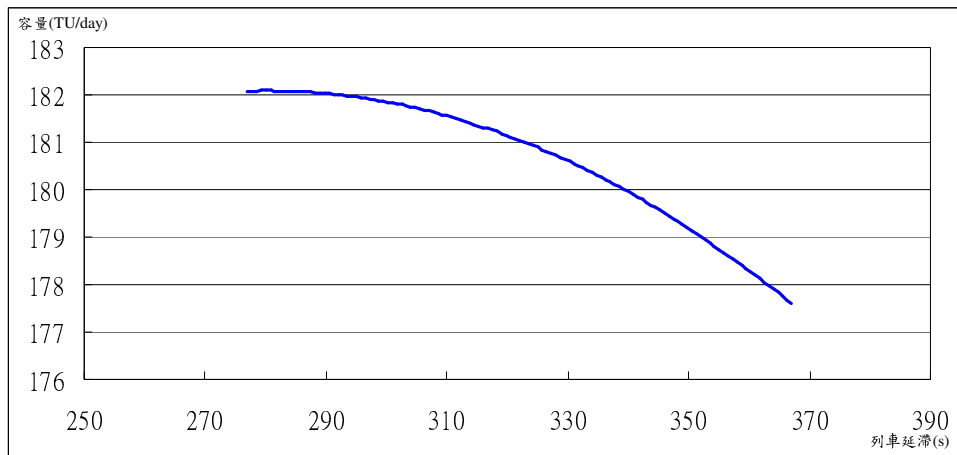


圖 5.7 列車延滯與容量關係圖

由圖 5.7 可知，列車延滯與容量呈現負相關，且為非線性相關，相關係數為 -0.940995156。

5.3.6 小結

由文獻[1]中可知，在軌道容量探討之議題中，影響容量之主要變數為「站間閉塞區間長度」，因此，在不變更車站配置及車種交通組成前提下，欲提高軌道容量，首先便可由縮短站間閉塞區間長度著手，即後續章節所欲探討之「移動閉塞行車制」，藉由通訊定位技術搭配，假想將現有閉塞區間劃分為極小等分，使得定位精準度提高，列車運行更能有效地運用軌道容量。

5.4 情境分析

情境分析係針對不同行車制度進行模擬，探討其容量值改變及模擬採取不同列車行車制度下之軌道容量值變化情形。

首先探討台鐵採用固定自動閉塞行車制度下之軌道容量分析，接者進行移動自動閉塞行車制列車行車模擬，求得軌道容量提昇值，作為未來採行移動閉塞行車制之參考依據。

5.4.1 台鐵 FAS 軌道容量分析

本章節探討目前台鐵採行「固定自動閉塞行車制」下，研究路段之軌道容量，首先針對目前現行班表軌道容量之計算，與模擬壓力測試求得採行「固定自動閉塞行車制」下，軌道容量之上限值。

5.4.1.1 台鐵 FAS 軌道容量現況

在採行「固定自動閉塞行車制」下，單股道多車種混合運轉，經由模擬一般日非尖峰下午十三點三十至十六點發車列車之軌道容量，依照現行班表模擬，以板橋站作為觀測基準站，除去熱機切除時間後，所得各車種組成最大運轉時隔如表 5.7 整理，經加權平均後所得之軌道容量為 122.248(TU/day)。

表 5.7 模擬現行班表最大運轉時隔

最大運轉 時隔 (13:30~16:00) 累計次數		後續追蹤列車車種			
		自強	莒光	復興	電聯
先行 列車 車種	自強	372/1	466/2	0/0	1255/2
	莒光	359/1	382/1	0/0	403/4
	復興	0/0	530/1	0/0	0/0
	電聯	702/4	606/2	550/1	960/5

資料來源：本研究整理

5.4.2 MAS 模擬結果

模擬程式經由更改控制邏輯，縮短列車追蹤間隔，達到提高軌道容量之目的。使用現行班表下，改變列車控制邏輯為「移動自動閉塞行車制」，則軌道流量由原來的 121.9621221 增加為 121.9908224。

模擬程式模擬現行列車運轉依原班表發車時間之列車運行圖如圖 5.8 所示。

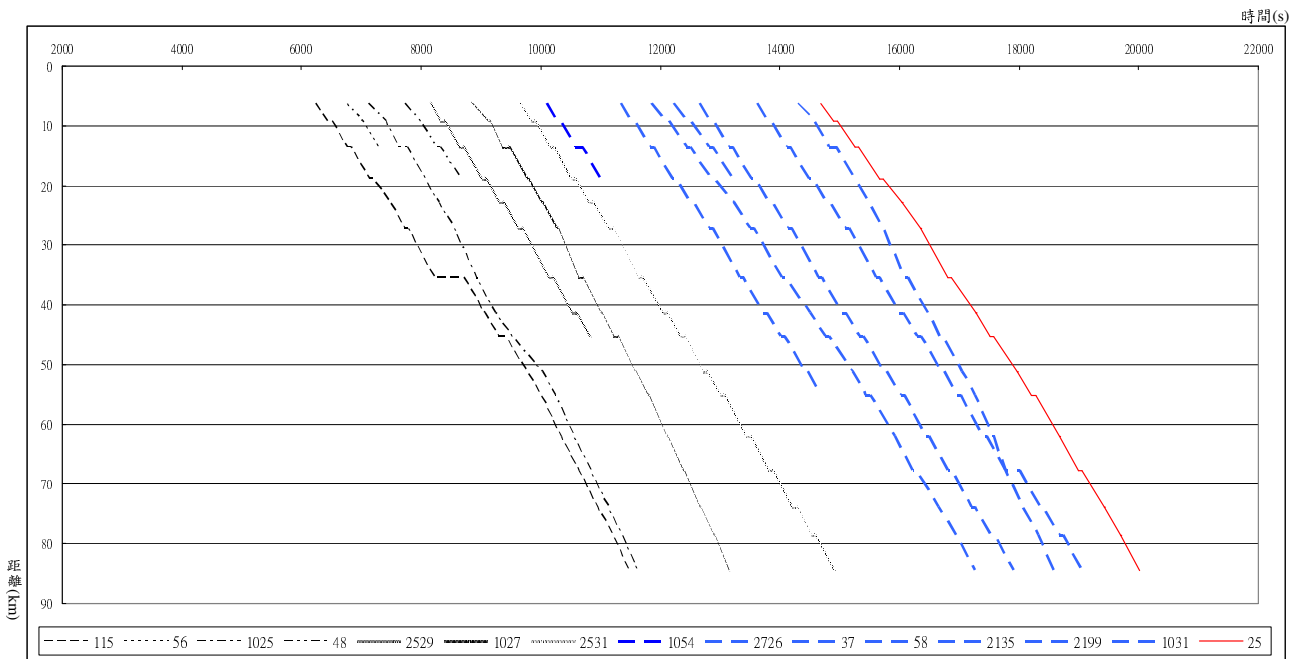


圖 5.8 現行列車運轉依原班表發車時間之 MAS 列車運行圖

由列車運行圖可看出，實際上容量與採行 FAS 行車制並無顯著增加，原因在於目前現行之班表係考慮固定閉塞區間所設，因此若依照現行班表發車時刻進行 MAS 模擬，將無法因行車制度之改變提高軌道容量。MAS 行車制在愈擁擠的形況下愈能發揮作用，但依照現行班表，列車排班對於 MAS 行車制來說稍嫌鬆散，列車較不易在站間運行中追上先行列車，無法發揮 MAS 行車制提升軌道容量之效，因此利用以下章節進行情境模擬，表現未來若採行 MAS 行車制，軌道容量之提升情形。

5.4.3 情境設定

在與現行運行情況相同下之列車停站任務、列車車種，進行不同行車制度下台鐵列車運行模擬，並且由於現行班表過於鬆散，無法有效反應出採行「移動自動閉塞行車制」之軌道容量提昇值，因此利用不同發車間距來進行各種情境模擬，以期能在相同比較基準下，清楚地比較出採行不同行車制度及發車間距對於軌道容量提升之影響。

情境設定首先比較針對現行運行班表及列車任務、車種設定進行在不同發車間距下多列車模擬，探討採行「移動自動閉塞行車制」下不同發車間距對於現行軌道運輸之容量改變情形，以證明採行「移動自動閉塞行車制」優於「固定自動閉塞行車制」之處；接著探討在各種發車間距下，各車種組成對軌道容量之影響，探討不同行車制度對於軌道容量之改變及提升。

5.4.3.1 現行列車任務下不同發車間距之容量分析

在此打破原有設計於「固定自動閉塞行車制」之發車間距，以利分析不同發車間距對於採行「移動自動閉塞行車制」之影響，參考 08：24 發車之 1099 自強號與 08：27 發車之 2712 電聯車發車間隔，這也是列車運行計畫中松山站一日最密集發車間隔，以此為發車間距下限，以下將每 180 秒、每 240 秒、每 300 秒、每 360 秒下，採行「移動自動閉塞行車制」之列車運行圖(見圖 5.9 至圖 5.12)：

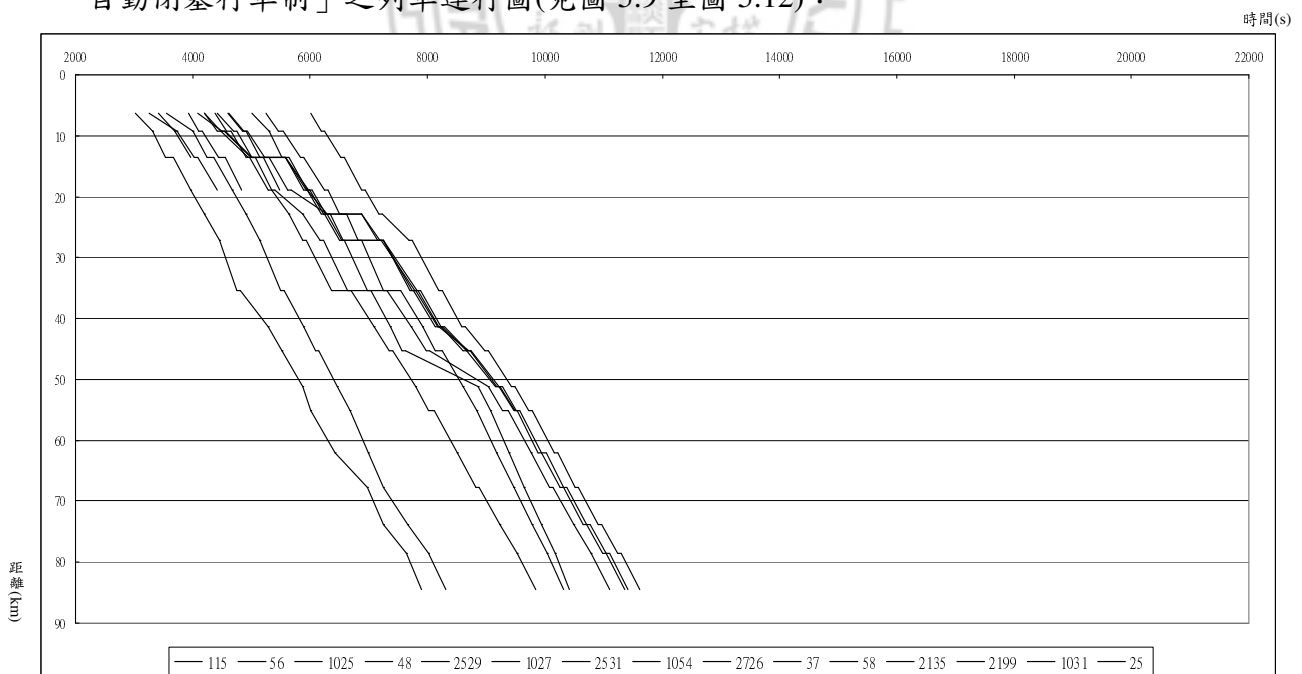


圖 5.9 發車間距=180 秒之列車運行圖(圖例按照發車順序)

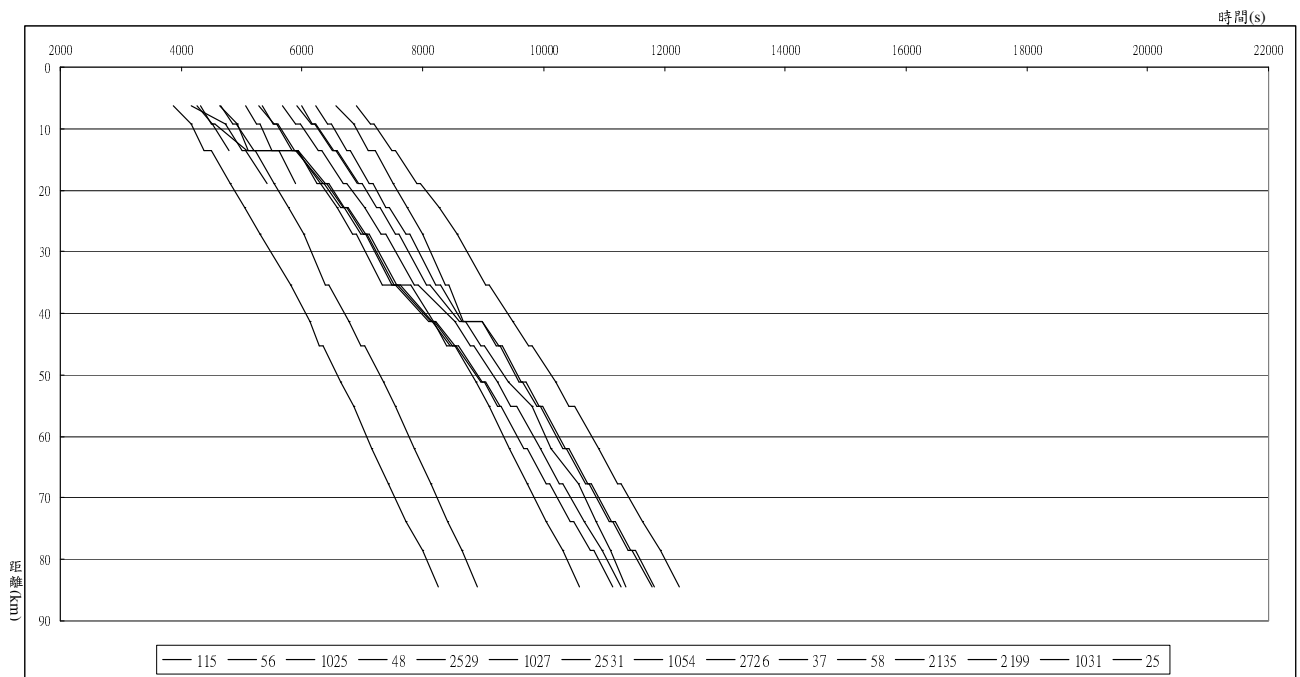


圖 5.10 發車間距=240 秒之列車運行圖(圖例按照發車順序)

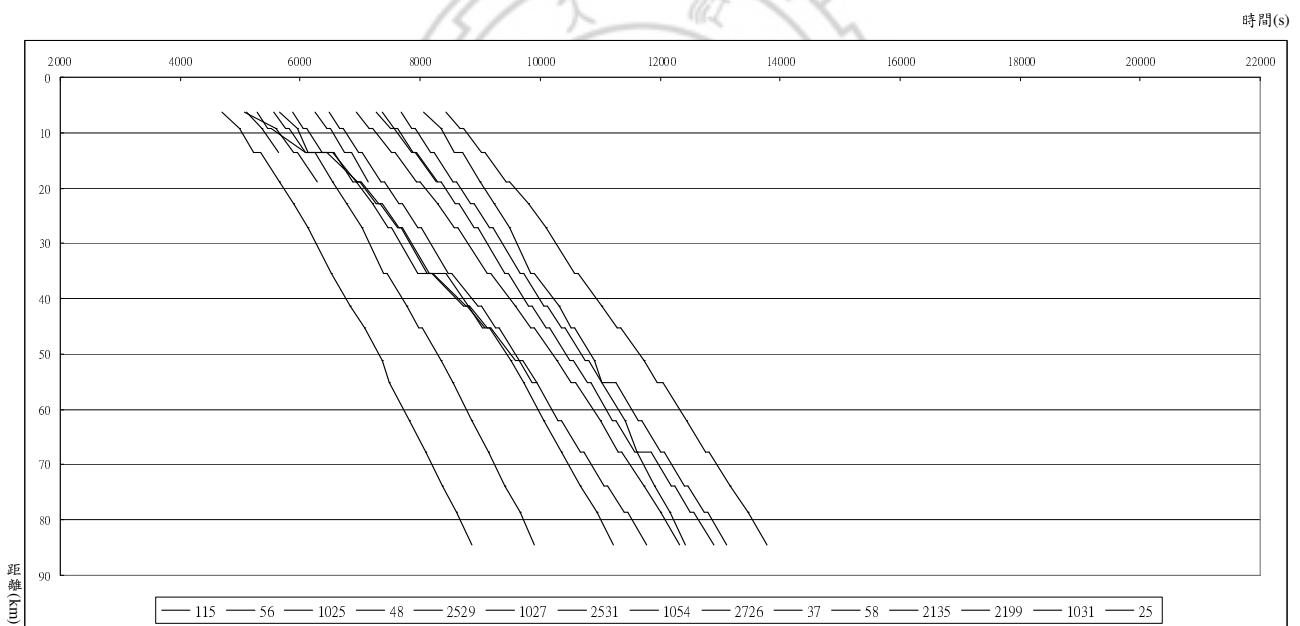


圖 5.11 發車間距=300 秒之列車運行圖(圖例按照發車順序)

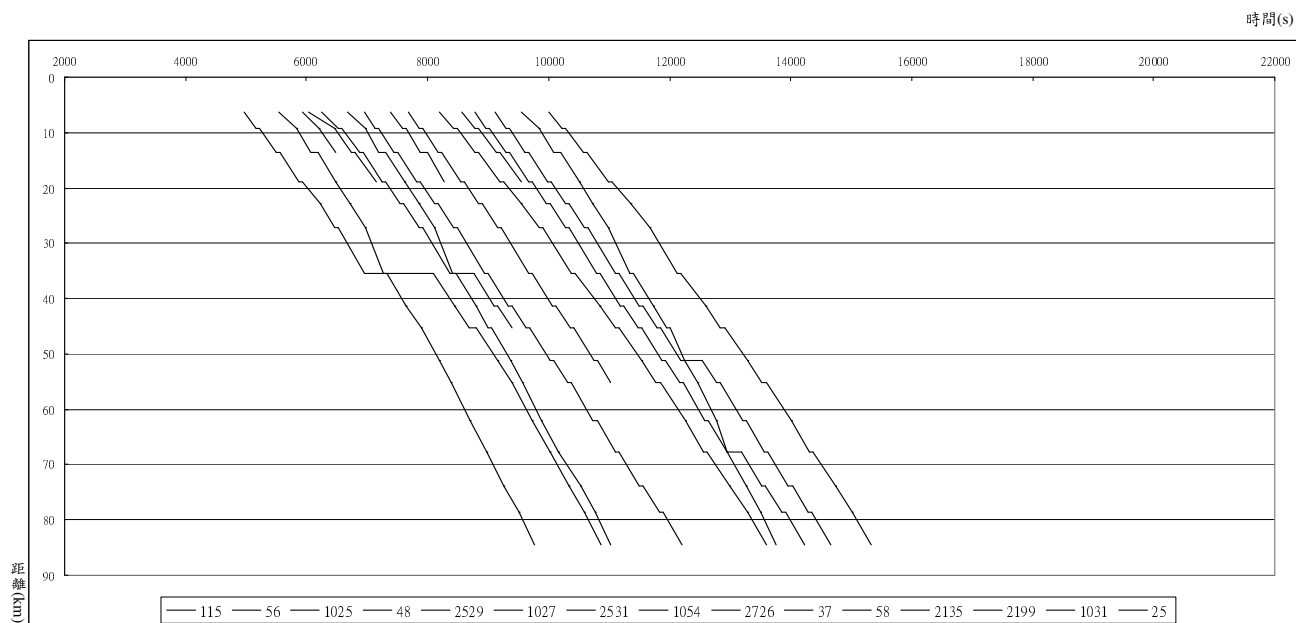


圖 5.12 發車間距=360 秒之列車運行圖(圖例按照發車順序)

由以上可看出，在不同發車間距下，各車運行之軌跡及列車追蹤情形，詳細之軌道容量值整理如表 5.8：

表 5.8 採行 MAS 行車制不同發車 headway 下之軌道流量

發車 headway(秒)	軌道容量(TU/day)
180	348.9
240	334.3
300	273
360	204.1
420	176.3
480	154.3
540	137.1
600	123.4

資料來源：本研究整理

當發車 headway 愈大時，列車運行軌跡愈接近採行「固定自動閉塞行車制」之運行情形，因此若欲提升軌道容量，除了縮短實體閉塞區間長度外，更進一步便是透過通訊、定位技術，採行「移動閉塞行車制」，以達靈活調度，提升軌道容量之目的。

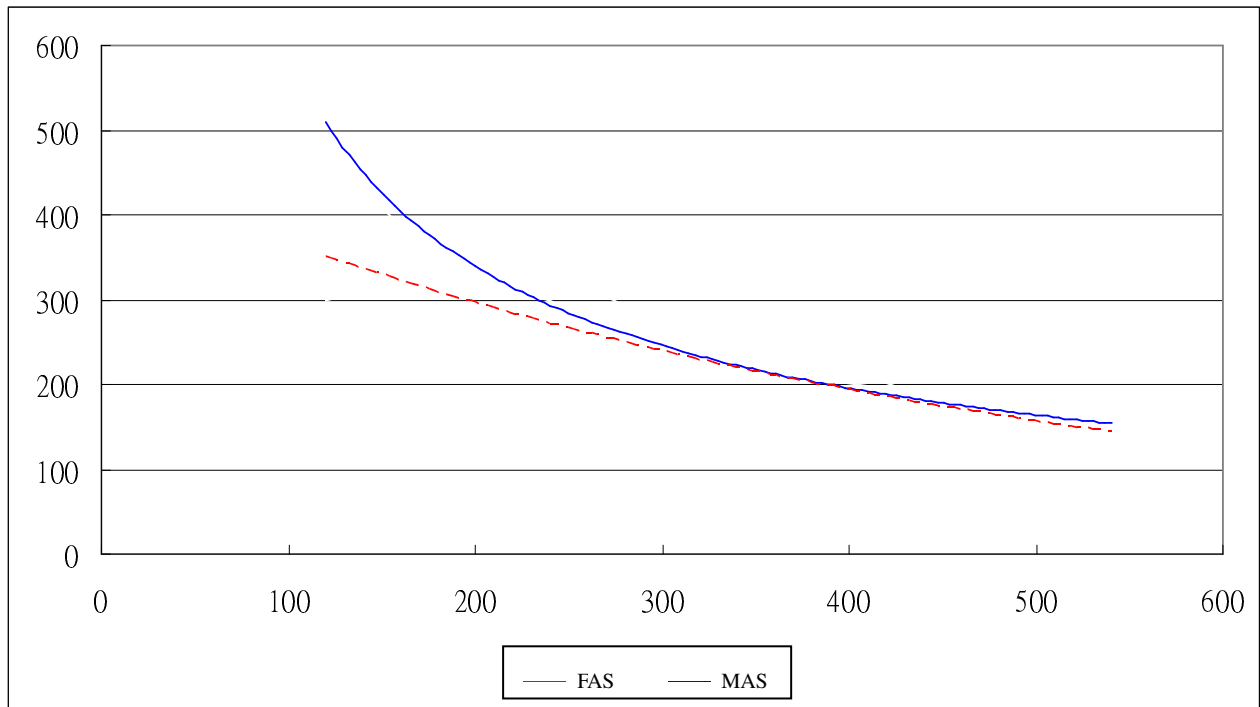


圖 5.13 不同發車間距下軌道容量比較圖

由圖 5.13 可看出，發車 headway 對於軌道流量之提升，對於「固定自動閉塞行車制」來說，無法有效地作用，原因在於列車間之追蹤間隔受實體閉塞區間影響，使得無法經由變動發車 headway 來提升，雖然站間之軌旁號誌機不為絕對號誌，司機員可依當時情況判斷列車是否減速，但由於無法確認先行列車確切位置，因此仍會保持較大之安全間距，以確保列車行車安全；反觀「移動自動閉塞行車制」，由於能及時掌握先行列車之速度與所在里程，因此對於列車之操控能更加精準，達到提升軌道容量、靈活調度，且由於不需站間軌旁號誌機，因此能以更低廉成本提供更好的服務，此又再次地驗證軌道運輸未來發展之方向，必然朝向智慧化、高科技之「移動自動閉塞行車制」。

5.4.3.2 相同發車間距下不同列車組成之容量分析

由於容量計算需要，因此情境模擬令所有列車起迄皆相同，從松山站至新竹站進行全程多車種模擬，以得出最適運行車種組合，由於列車組成設定由於實際上運行之復興號列車非常少，因此僅考慮自強號、莒光號，以及電聯車等三者組合，由於篇幅關係，在此僅繪圖發車間距為 300 秒下各種列車組成之情形，相關之列車運行圖如圖 5.14 至圖 5.19：

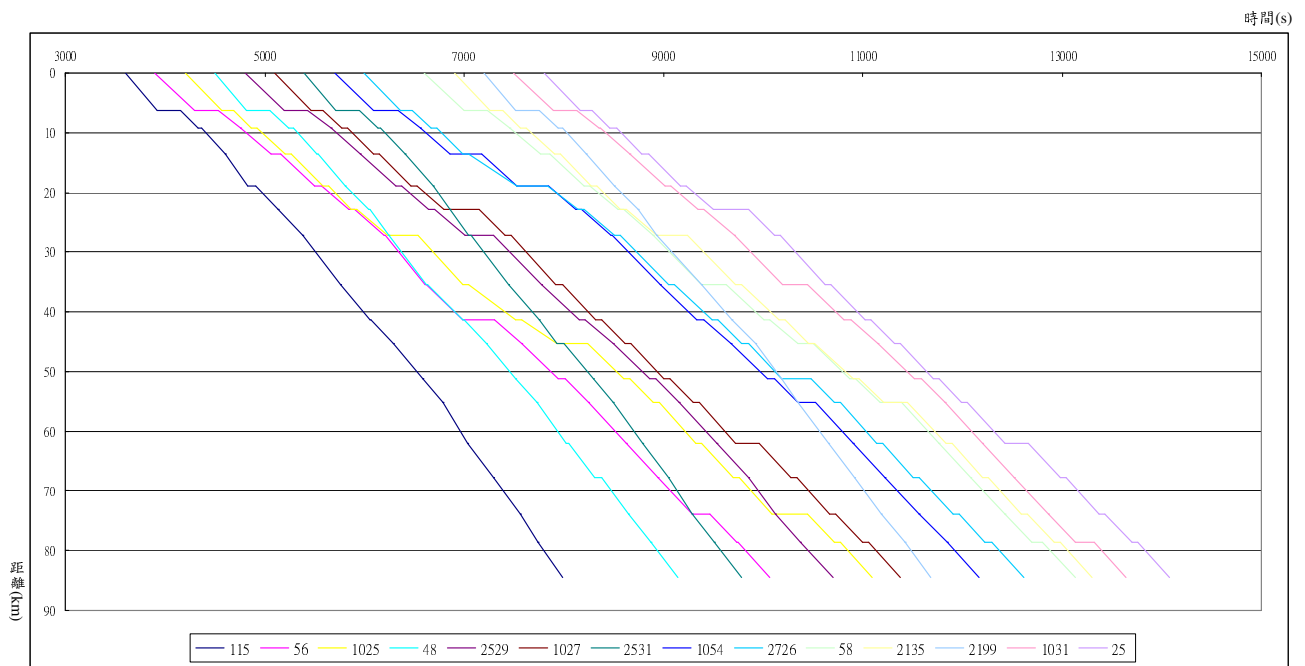


圖 5.14 發車間距=300 秒下(自強-莒光-電聯) 列車組成模擬運行圖

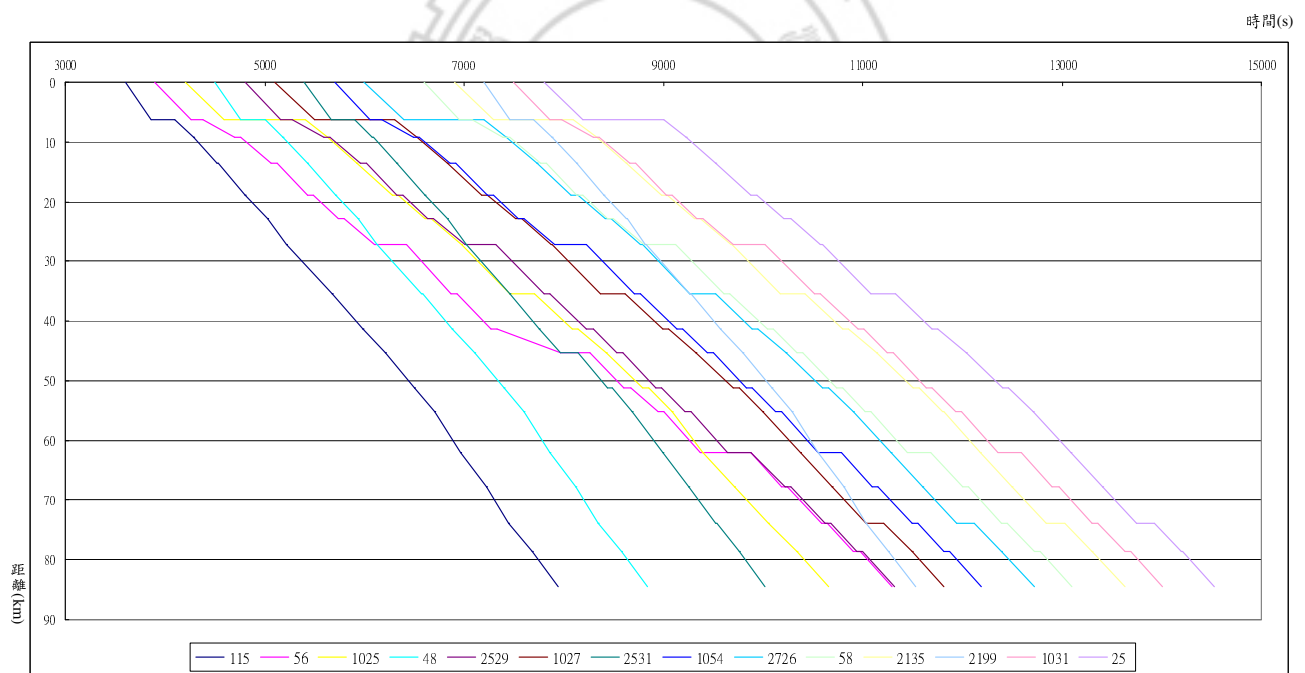


圖 5.15 發車間距=300 秒下(自強-電聯-莒光) 列車組成模擬運行圖

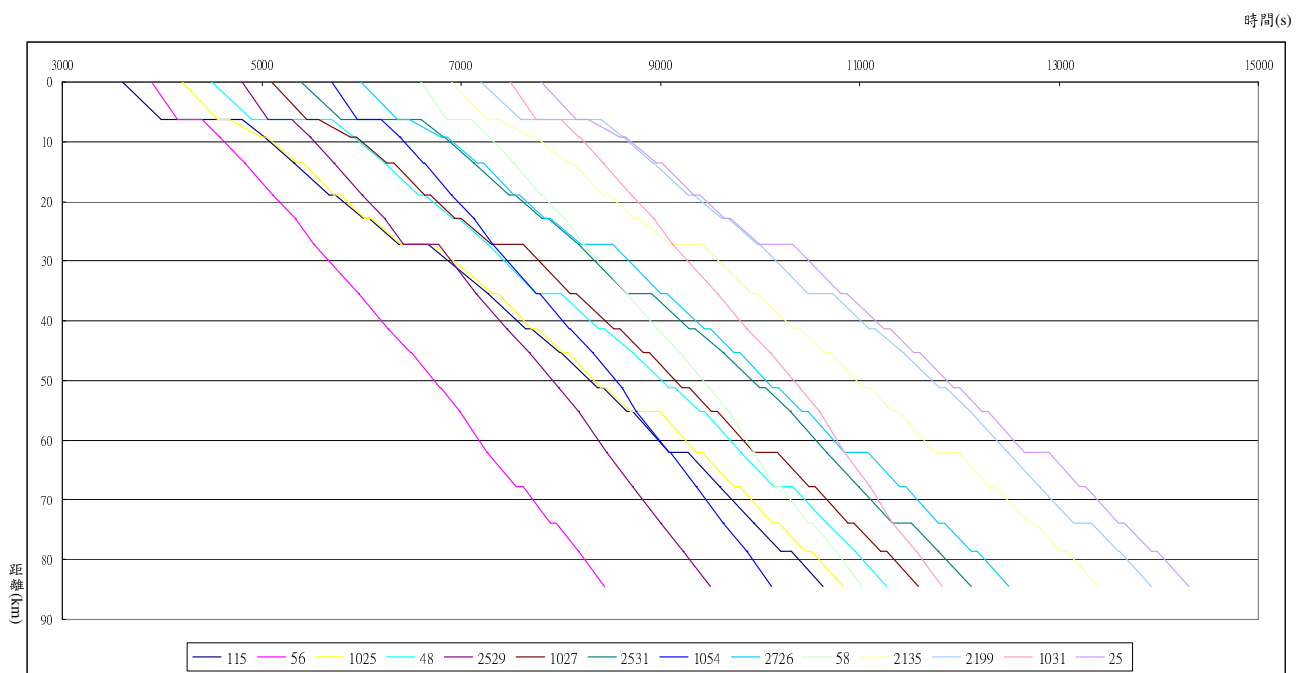


圖 5.16 發車間距=300 秒下(莒光-自強-電聯) 列車組成模擬運行圖

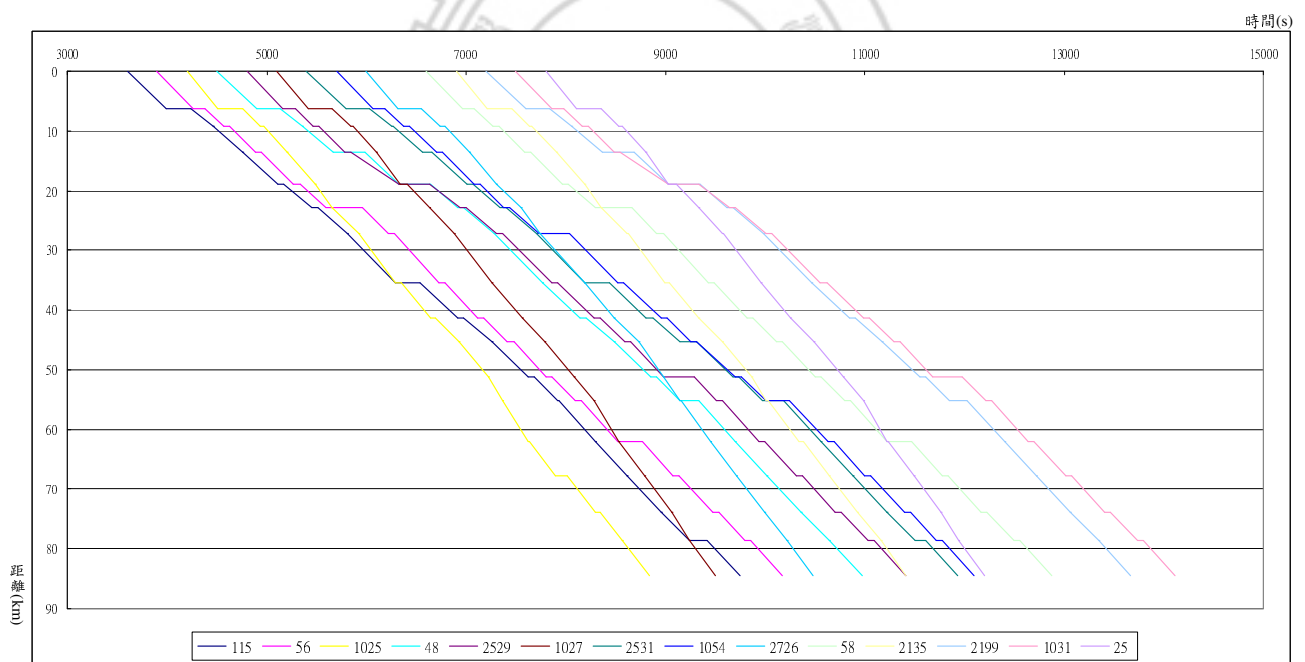


圖 5.17 發車間距=300 秒下(莒光-電聯-自強) 列車組成模擬運行圖

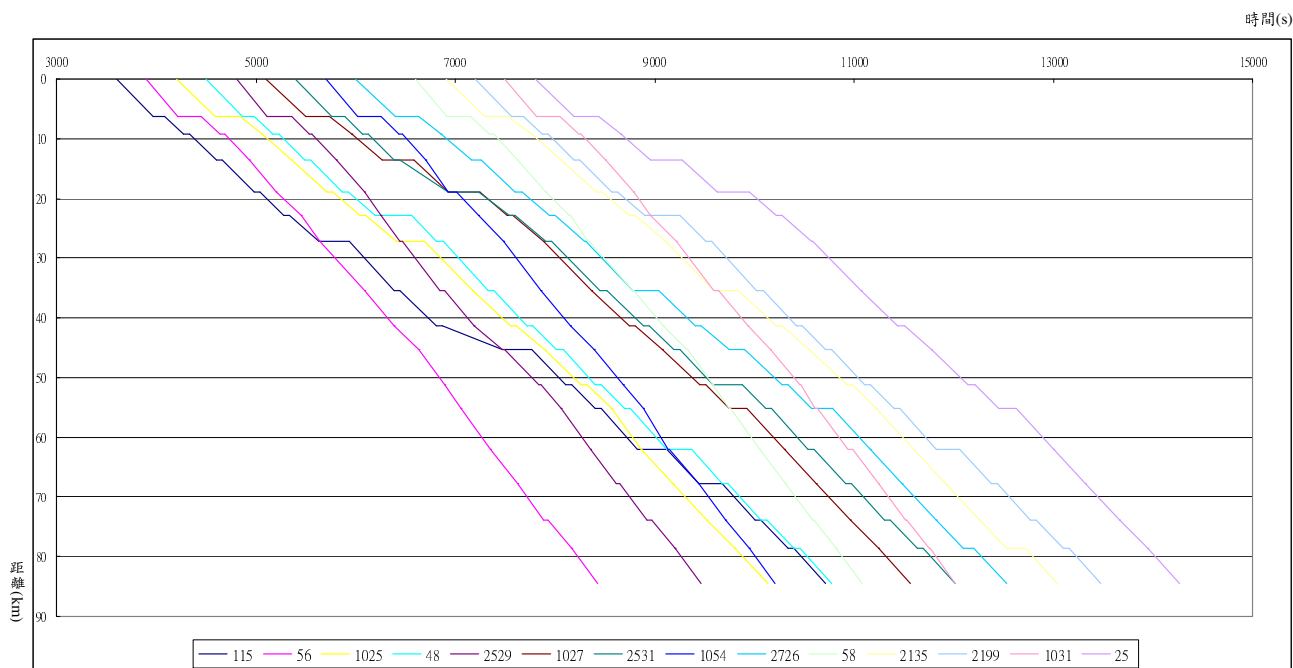


圖 5.18 發車間距=300 秒下(電聯-自強-莒光) 列車組成模擬運行圖

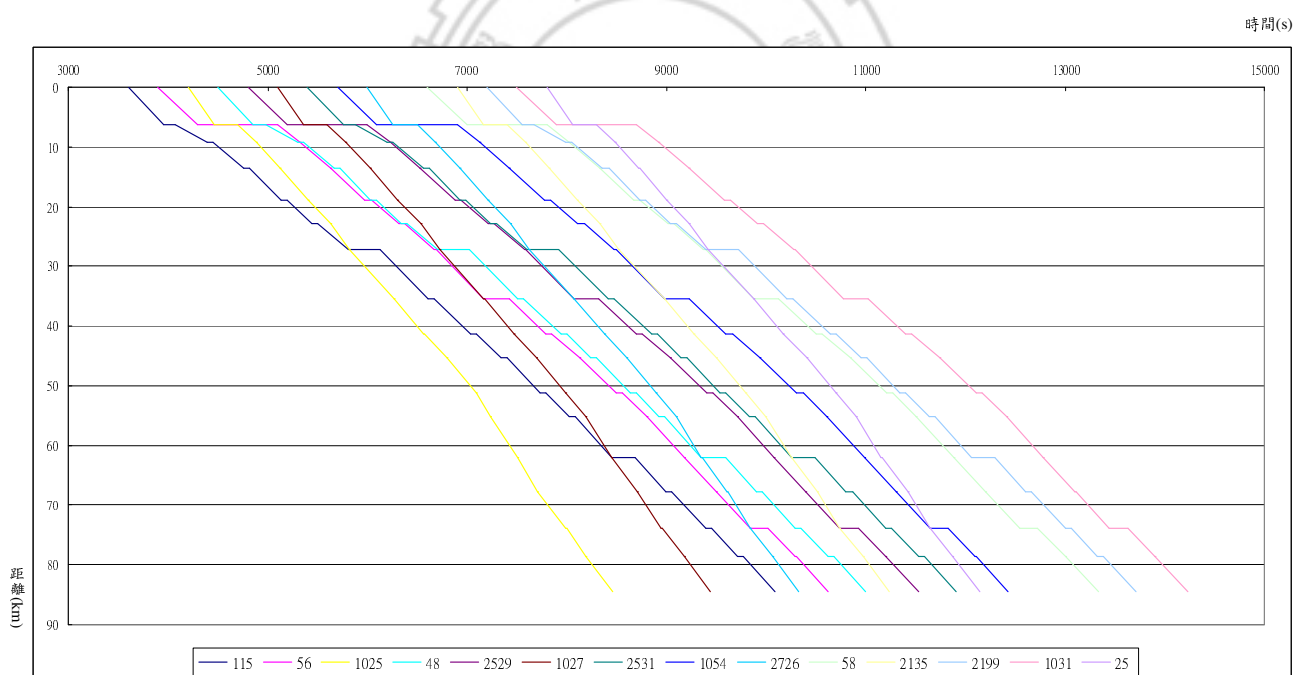


圖 5.19 發車間距=300 秒下(電聯-莒光-自強) 列車組成模擬運行圖

表 5.9 發車間距=300 秒下不同列車組成模擬容量表

列車組成	模擬容量值(TU/day)
自強-莒光-電聯	178.5738891
自強-電聯-莒光	164.7544891
莒光-自強-電聯	181.9266538
莒光-電聯-自強	202.8565838
電聯-自強-莒光	179.501385
電聯-莒光-自強	190.2385321

資料來源：本研究整理

由表 5.9 可看出，當在相同發車間距=300 秒下，不同列車組成之容量值以電聯車-莒光號-自強號之搭配 202.8565838 (TU/day) 為最高，因此判定當多車種混合運行採行「移動自動閉塞行車制」時，以此為較佳的車種組合。

5.4.3.3 不同發車間距下不同列車組成之容量分析

經由上節討論，得知在混合車種運行時，以莒光-電聯-自強之列車組成能達到最大之軌道容量，由於篇幅關係，在此僅針對此種列車組成，進行在不同發車間距下，探討其軌道容量之變化，相關之模擬列車運行圖如圖 5.20 至圖 5.26：

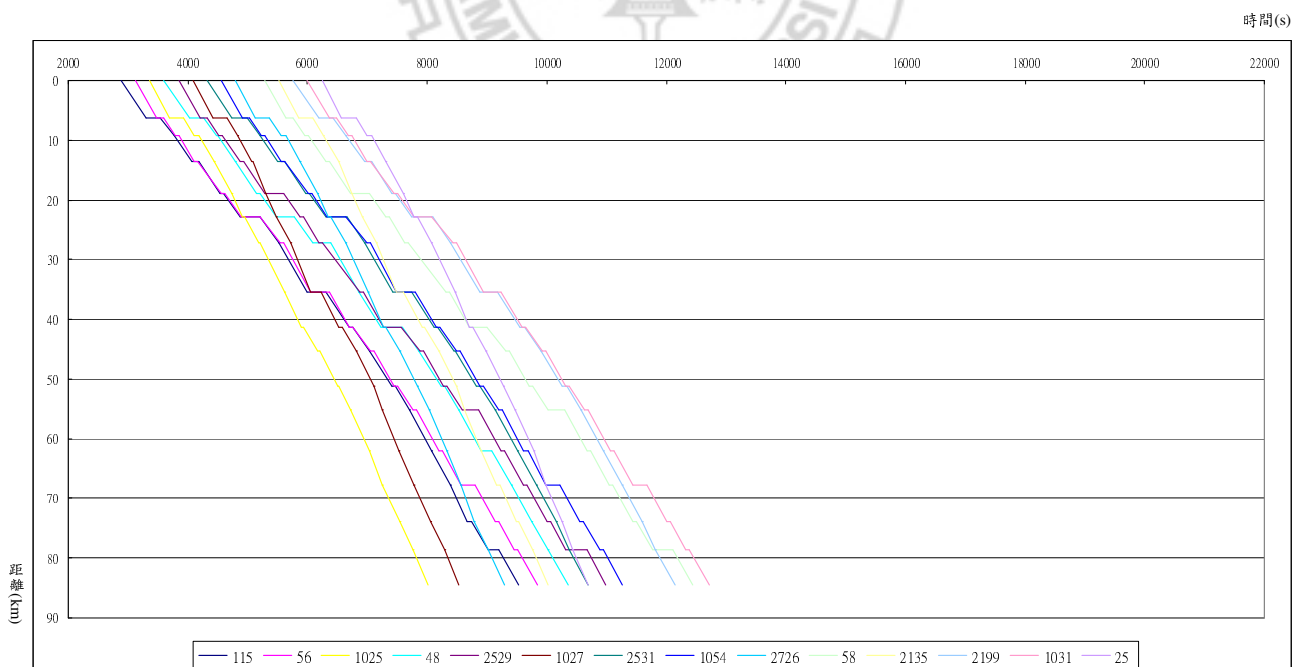


圖 5.20 發車間距=180 秒下(電聯-莒光-自強) 列車組成模擬運行圖

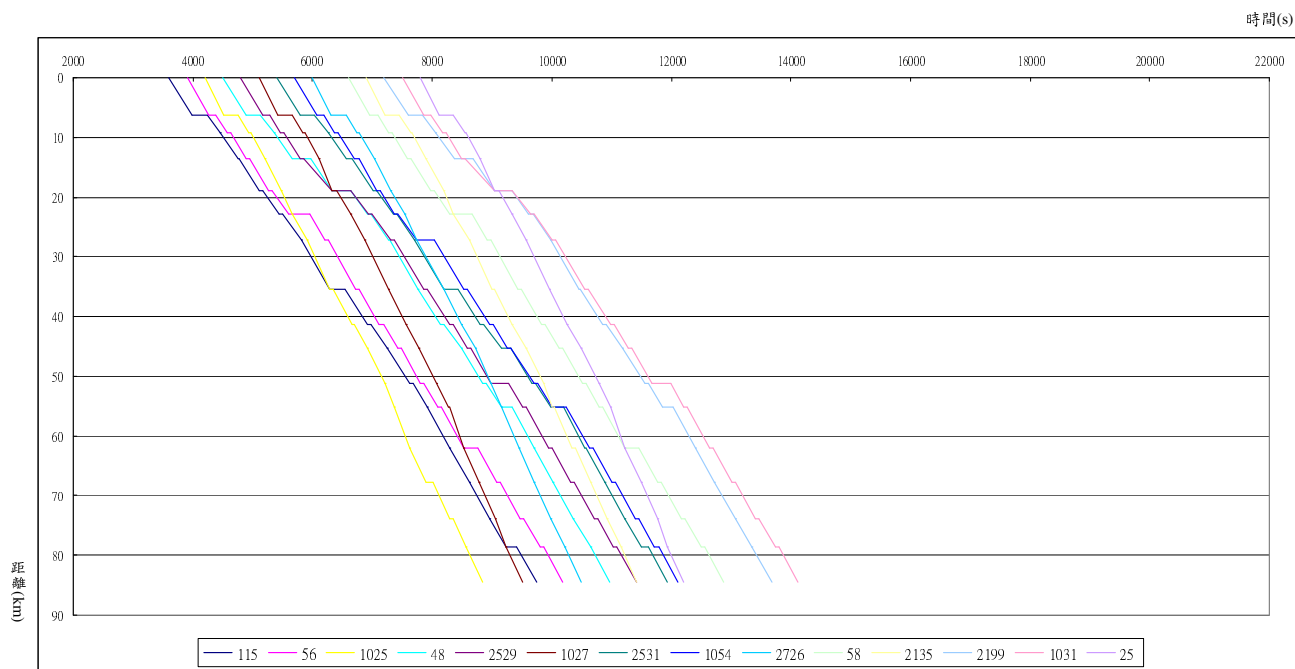


圖 5.21 發車間距=240 秒下(電聯-莒光-自強) 列車組成模擬運行圖

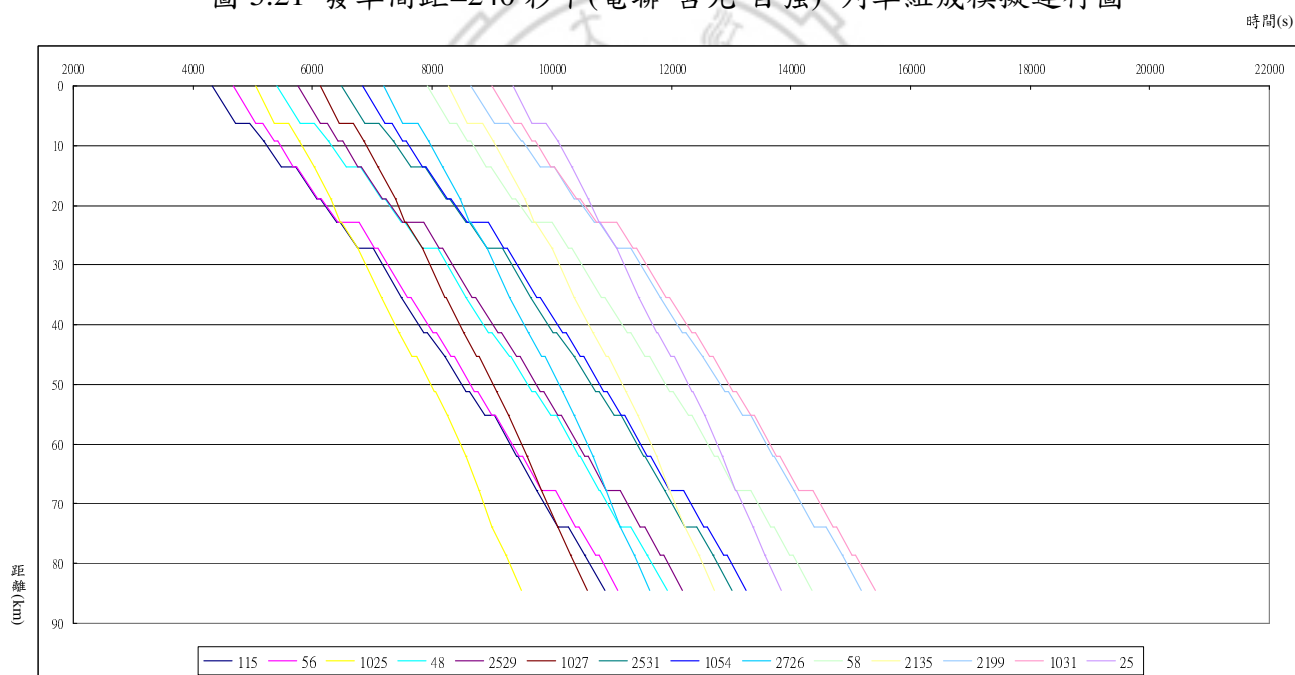


圖 5.22 發車間距=360 秒下(電聯-莒光-自強) 列車組成模擬運行圖

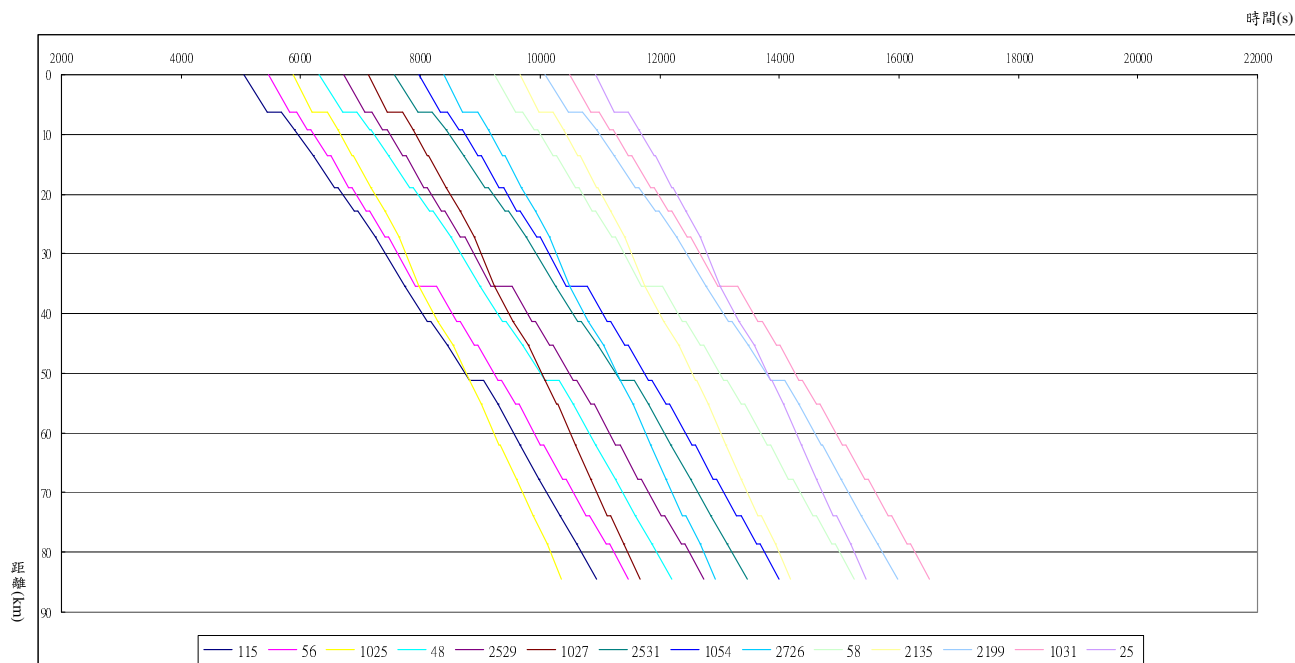


圖 5.23 發車間距=420 秒下(電聯-莒光-自強) 列車組成模擬運行圖

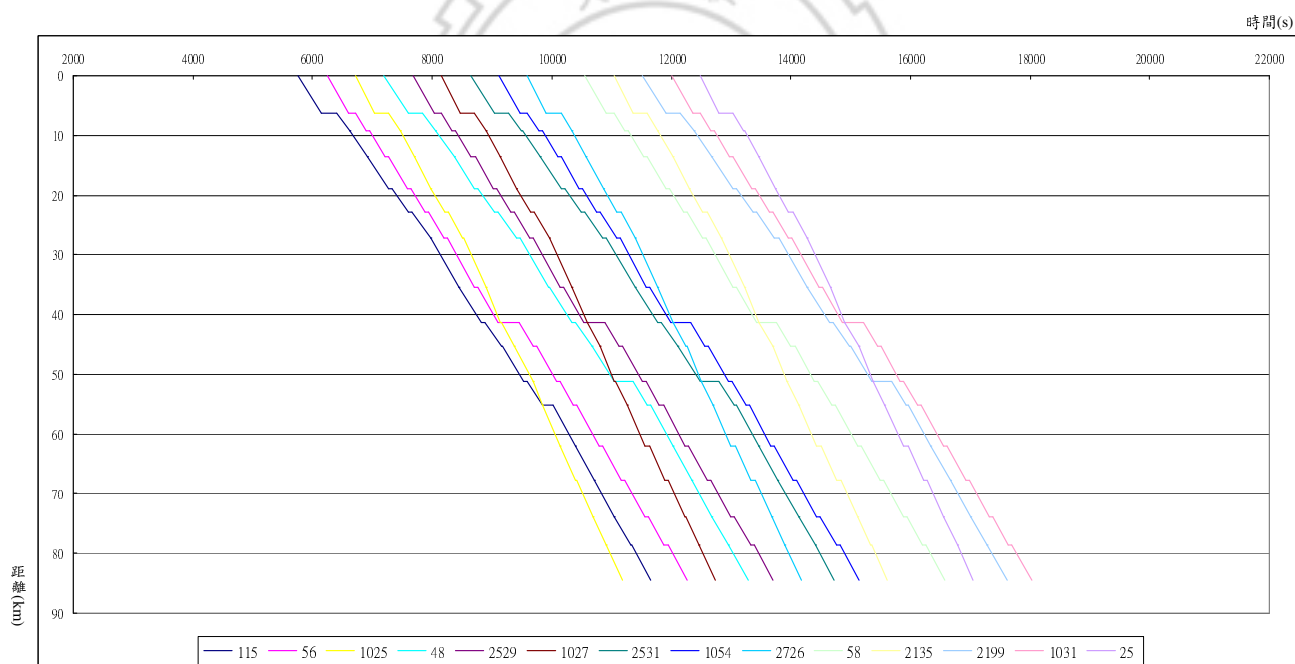


圖 5.24 發車間距=480 秒下(電聯-莒光-自強) 列車組成模擬運行圖

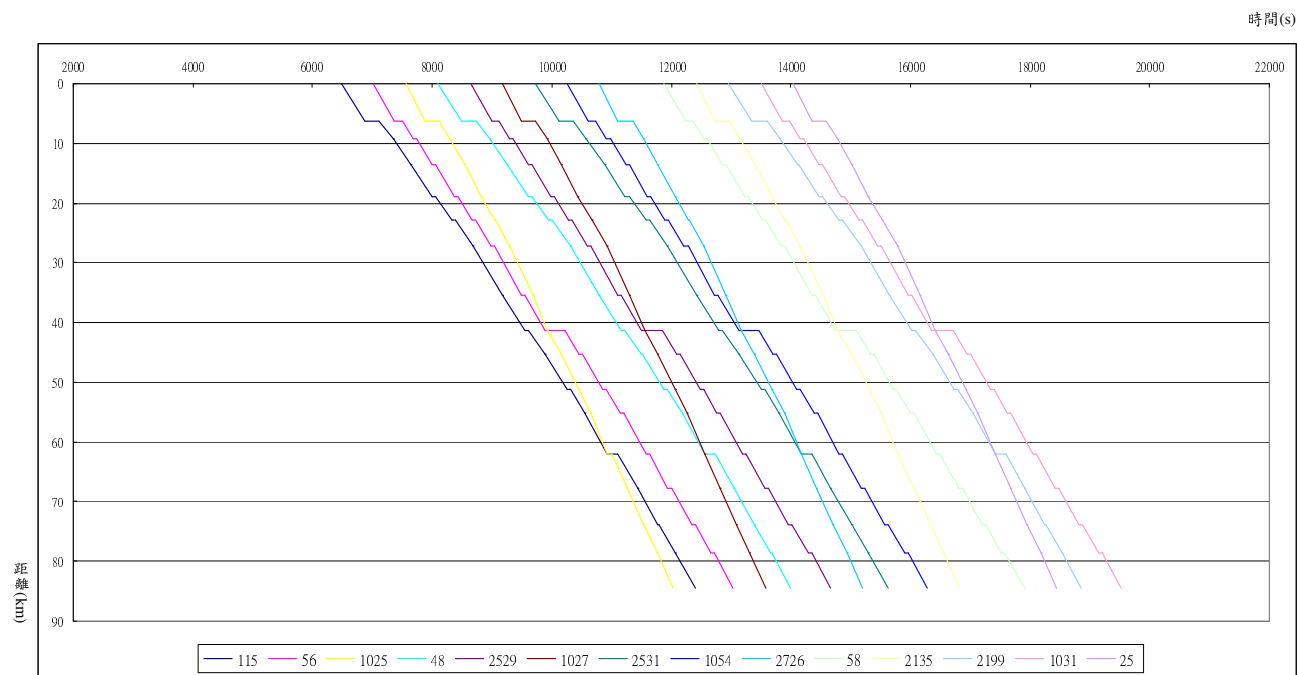


圖 5.25 發車間距=540 秒下(電聯-莒光-自強) 列車組成模擬運行圖

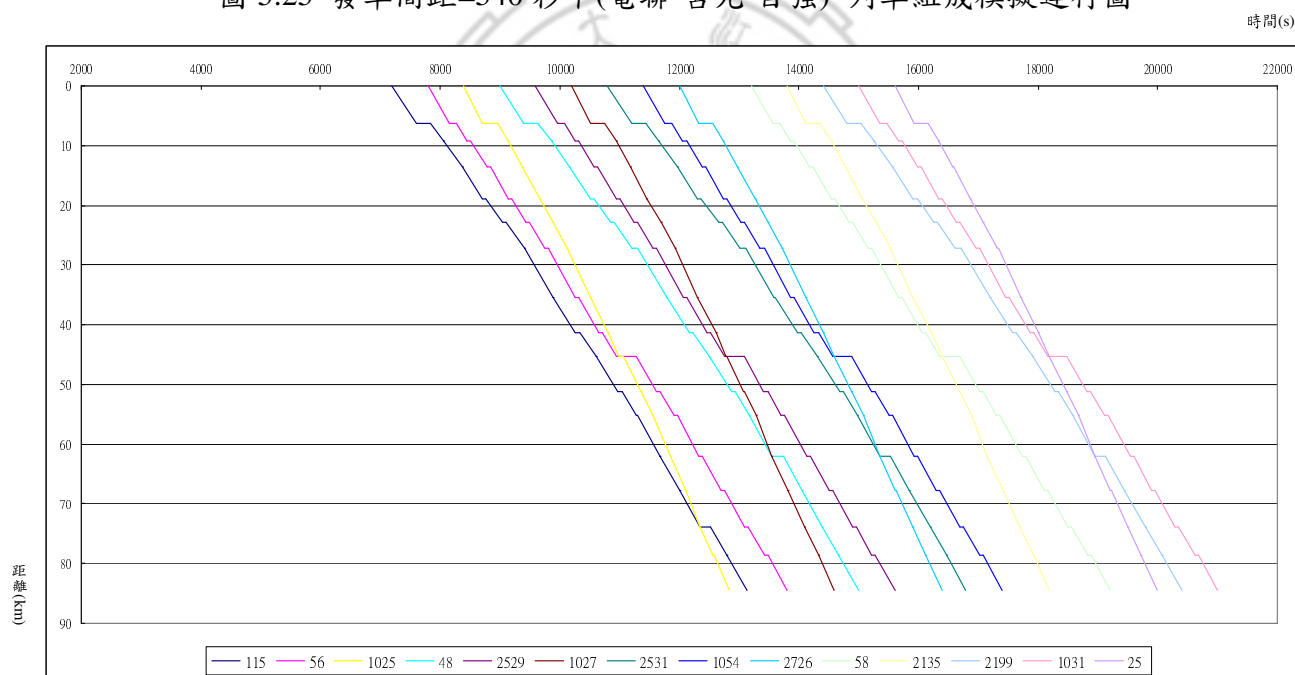


圖 5.26 發車間距=600 秒下(電聯-莒光-自強) 列車組成模擬運行圖

表 5.10 不同發車 headway 下特定列車組成模擬容量表

列車發車 headway(秒)	模擬容量值(TU/day)
240	233.0935252
300	202.8565838
360	182.0544337
420	174.5454545
480	156.3565073
540	167.902834
600	127.4492932

資料來源：本研究整理

由表 5.10 可看出，當發車間距達到 240 秒時，其軌道容量為最大，但整體並非均遵循此一趨勢，因列車組成之影響使得軌道容量之提升與列車發車間距並非呈線性相關。

5.5 小結

綜合上述，可知在採行不同列車行車控制制度時，對於軌道容量有絕對的影響。本研究之軌道容量計算公式在於考慮多車種運行下，各車種性能不同，列車任務設定不同，因此採取加權平均方式計算列車運轉時隔，以充分描述多車種混合運行之軌道容量。

軌道容量之計算，需以最瓶頸路段，通常為車站之運轉時隔，來進行容量之求解，但由於現實情況中台鐵乃多車種運行且列車任務相異，因此若以最瓶頸路段所得之列車運轉時隔，往往會產生偏誤，造成低估。

造成上述情形之原因，在於列車任務混合編排，於正線上運行之列車終點站不盡相同，以本研究為例子，大部份列車由松山出發，終點站設定為新竹站，但些許列車中途便完成服務，不再繼續運行，例如 56 次、48 次列車僅服務至樹林站，另外還有 2529 次、1054 次、2726 次…等列車，並無全程參與運行，使得在容量計算上，若瓶頸車站在樹林站或更南端之車站，便無法衡量出正確之運轉時隔。

由圖 5.27 可見，當中途有列車完成服務離開系統時，若以較南端車站計算列車運轉時隔，則必然產生偏誤。若圖中離開之列車繼續參與服務，其餘裕之空間足夠其運行，因此若以較南端車站為計算基準站，將使得取得之列車運轉時隔較大，則求解出之軌道容量將勢必低估，不可不慎。

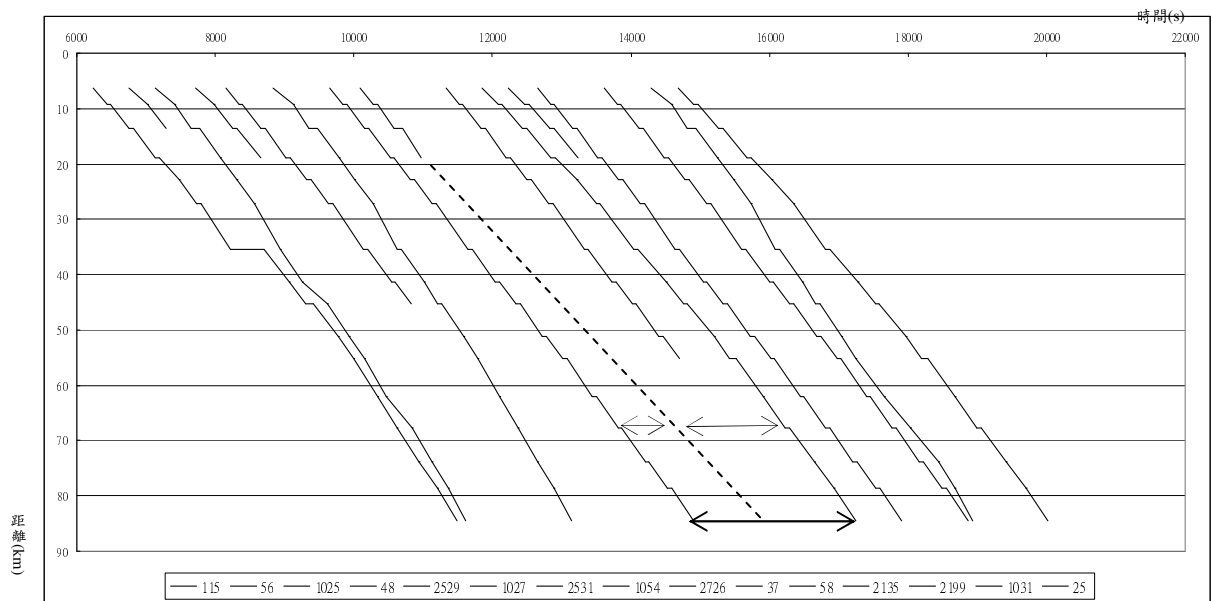


圖 5.27 軌道容量列車運轉時隔求解選擇示意圖

因此實際求解時，應以未受列車任務影響之車站為候選觀察基準站，在此之中選擇最瓶頸車站計算列車運轉時隔，方能避免軌道容量低估之問題。

採行「移動自動閉塞行車制」之軌道運輸，若欲發揮其功效，歸納必須有以下特性：

- 1、區段長度不可過短：若路線長度過短，則列車無法發揮其性能追蹤先行列車，列車安全空間間距對於列車運行影響不大，使得軌道容量之提升不明顯，因此區段長度不宜過短，使得後續列車得以追蹤並於途中車站越行，以提升軌道容量。
- 2、發車 headway 不可過長：若發車 headway 過長，則列車於區間運行中，無法在 headway 間隔時間內追上先行列車，使得列車安全空間間距無法發揮作用，更遑論「移動自動閉塞行車制」之追蹤先行列車伺機越行。發車 headway 與區段長度有直接的關係，兩者應互為消長關係，方能充分發揮「移動自動閉塞行車制」之功效。

採行「固定自動閉塞行車制」之下，不同車種組成對於軌道容量之關係可以圖 5.28 表示：

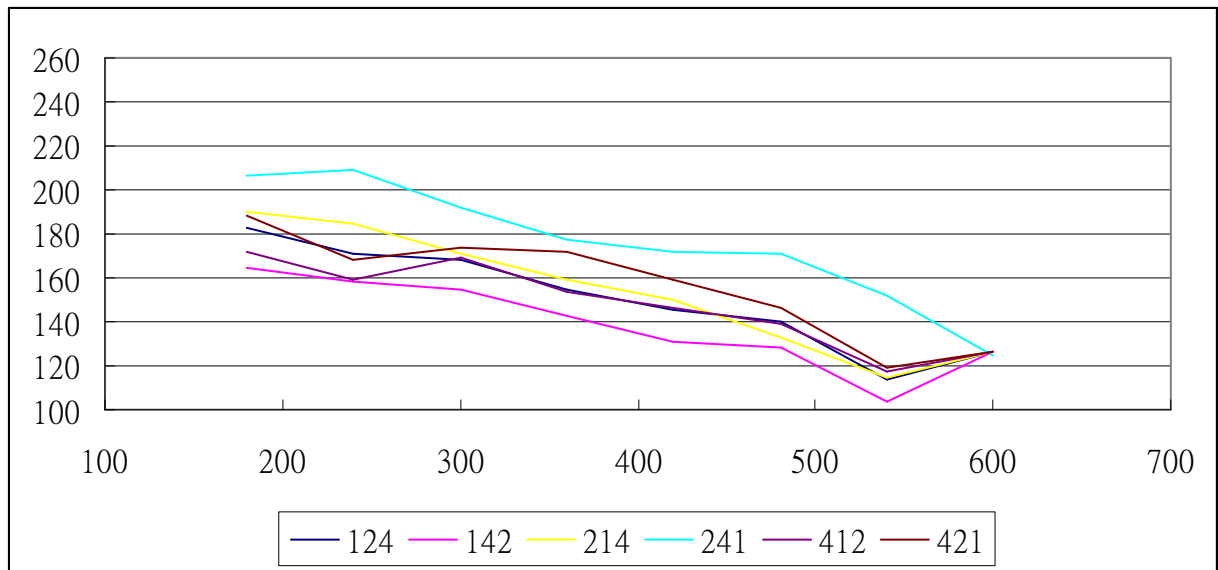


圖 5.28 FAS 下不同車種組成發車間距-容量圖

採行「移動自動閉塞行車制」之下，不同車種組成對於軌道容量之關係可以圖 5.29 表示：

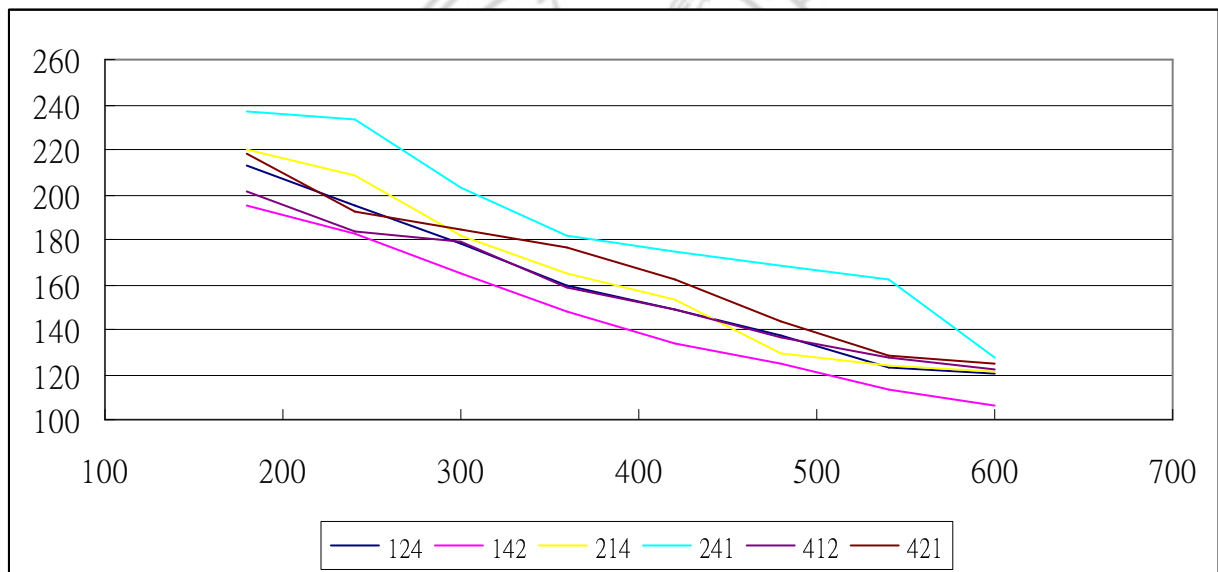


圖 5.29 MAS 下不同車種組成發車間距-容量圖

由圖 5.28 與圖 5.29 可看出，當列車在不同行車制度下，不同發車間隔對於不同列車車種組成容量之影響，將兩者疊圖後，便可看出容量在採行不同行車制度下之改變情形。

第六章 結論與建議

本研究目的在於構建一列車模擬模式，考慮多車種運行，坡度、曲線、線型速限、站內股道數以及列車待避越行。經由模式驗證證明模擬模式能充分描述真實世界列車運行。敏感度分析係找出影響軌道容量計算之影響因子對於軌道容量影響程度，情境分析則是探討當採行「移動自動閉塞行車制」下，軌道容量提升程度，以作為未來軌道運輸發展之參考。

6.1 結論

構建軌道容量解析模式，參考現行台鐵經驗公式，除用加權平均方式計算列車運轉時隔以反應車種組成外，針對觀測基準點之選擇進行初步探討，應排除列車任務提早離開系統之情形，選定不受影響之觀測點最瓶頸路段，以免低估軌道容量。

經由模式驗證，證明構建之模擬程式能充分描述現實世界列車運行情形，低等級列車會於設定待避處掃描並詢問後續追蹤列車是否將越行；沿線之速限、坡度、線型曲線不斷地對於列車運行產生作用，俾使列車處於安全運行狀態。

值得注意的是，採行「固定自動閉塞行車制」之下，觀察樣本約需 14000 秒方能完成服務，而採行「移動自動閉塞行車制」，假設列車發車間隔為 300 秒僅需不到 10000 秒即可完成服務。軌道容量由傳統固定自動閉塞行車制之 122(TU/day)提升至 273(TU/day)，提升 124%，成效顯著。由列車運行圖可看出，FAS 列車運行間仍有空閒餘裕空間，可依調度需求靈活使用。

由情境模擬中發現，欲採行「移動自動閉塞行車制」必須符合以下條件：區間長度不宜過短或過長及發車時隔不宜過長。若不符合上述條件，則無法利用「移動自動閉塞行車制」達成顯著提升軌道容量之效。區間長度過短則後續追蹤列車無法在先行列車進站前與先行列車距離列車安全間距門檻值追蹤尾隨先行列車進站，並進行越行動作，若區間長度過長，則後續多輛列車皆追蹤至先行列車列車安全間距後方，又由於站間為單軌運行，因此後續列車皆受先行列車影響，被迫慢行，降低列車運轉服務水準。發車間距若過長，則後續列車無法在站間運行時便追蹤至先行列車後方，因而產出近似平行之列車運行圖，無法收「移動自動閉塞行車制」提升軌道容量之效。

本研究主要貢獻在於建立一考慮多車種混跑、允許錯車行為之模擬模式，將坡度、曲線對於列車運行影響納入考慮，並且可針對不同行車制度進行模擬。

6.2 建議

- 1、本研究乃是針對台鐵系統多車種混跑、單股道單向運行，不考慮對向列車交會，因此本模擬模式能進行捷運、高鐵等軌道系統之模擬，唯需針對各軌道系統不同之處進行調校，方能適用。
- 2、本研究未考慮列車實際運行五個階段(加速、等速、巡航、制動、停站)裡巡航的部份。列車實際運行時，為節省能源，因此設計在某些路段上列車可採惰行方式運行，未來可修改本模擬模式以達到更精確模擬。
- 3、本模擬模式僅針對單股道單向列車運行，無法描述對向列車交會、儲車、單線運轉…等行為，因此若欲完善本模擬模式，建議由此著手，並搭配列車排點電腦化研究，兼顧供應面及需求面。
- 4、本研究未盡之處，在於僅考慮現況複線運轉下單向列車運行，而未將整體路網納入考量，因此對於列車模擬僅描述其同向錯車待避行為而無法表現對向列車交會行為，未來研究方向可描述路網中列車運行之模擬模式，將支線交互作用納入考慮。行車動力學方面，未將死帶(Dead Band)的概念納入程式中，使其更能符合列車實際運行情況，同時為求模式簡化亦忽略列車惰行運行，此方面研究應與列車運轉模擬、能源消耗模擬配合。若再能與列車排點模式配合，在台鐵實務上更可用於諸多決策之支援輔助工具。



參考文獻

(一)中文部份

1. 中興顧問工程社，軌道容量研究-臺鐵系統容量模式之構建分析(一)，交通部運研所，民國 94 年 4 月。
2. 孔慶鈴、劉其斌，鐵路運輸能力計算與加強，中國鐵道出版社，1999。
3. 王鶴鳴，鐵路行車組織(第四版)，中國鐵道出版社，1999。
4. 田長海等，提速線路列車速度密度重量，中國鐵道出版社，2001。
5. 交通部運輸研究所，臺灣地區軌道系統容量研究架構暨臺北捷運系統容量分析，交通部運輸研究所，民國 93 年 6 月。
6. 朱松年、宋瑞，「列車速度聯控行車制理論分析」，鐵道學報，Vol.19，No.6，1997。
7. 李治綱、丁國樑、黃哲旭，「鐵路列車模擬模式之研究」，中華民國運輸學會第 11 屆論文集，民國 85 年 12 月。
8. 汪希時，智能鐵路運輸系統 ITS-R，中國鐵道出版社，2004。
9. 余明興等，Borland C++ Builder 5 學習範本，松崗，民國 89 年 7 月。
10. 卓訓榮、蔡肇鵬，「軌道運輸系統智慧化之架構」，第 17 屆中華民國運輸學會論文集，民國 91 年 12 月。
11. 周潔、陳衡，「移動閉塞的原理、系統結構及功能」，城市軌道交通研究 2004，。
12. 林宜勝，軌道列車最短時間運轉曲線自動規劃程式的研發，台北科技大學機電整合研究所碩士論文，民國 90 年。
13. 楊肇夏，計算機模擬及其應用，中國鐵道出版社，1999。
14. 陶冶中等，「臺灣地區軌道運輸系統智慧化發展領域與使用者服務之供需分析」，第 16 屆中華民國運輸學會論文集，民國 90 年 11 月。
15. 張仕龍等，「臺北捷運系統路線容量分析-以 TCQSM 為例」，第 19 屆中華民國運輸學會論文集，民國 93 年 11 月。
16. 張濟民等，「準移動閉塞列車安全間隔時間的計算」，鐵道學報，Vol. 21，No.3，1999。
17. 陳一昌、胡守任，「臺灣地區發展智慧型運輸系統(ITS)綱要計畫」，第 14 屆中華民國運輸學會論文集，民國 88 年 12 月。
18. 陳潔如，考慮轉車之鐵路列車排點模式，成大交管所碩士論文，民國 90 年。
19. 陳鑑康，鐵路列車運行模擬模式之研究，成大交管所碩士論文，民國 83 年。
20. 許洋豪，「台灣鐵路旅客列車運行圖」，台鐵本舖，民國 95 年 4 月。
21. 游才譽，台鐵推拉式列車運動性能之分析與模擬，台北科技大學機電整合

- 研究所碩士論文，民國 91 年。
22. 游俊雄，多種停站方式下高速鐵路排班模擬之研究，成大交管所博士論文，民國 89 年。
 23. 游俊雄，捷運列車運行模擬之研究-以台北市中運量木柵線為例，成大交管所碩士論文，民國 83 年。
 24. 黃哲旭，捷運鐵路列車模擬模式之研究，成大交管所碩士論文，民國 82 年。
 25. 黃民仁，鐵路工程學，文笙書局，民國 82 年。
 26. 陽介平，鐵路運輸能力的計算與利用，中國鐵道出版社，2001。
 27. 趙礦英,馮俊杰，鐵路行車組織與管理，中國鐵道出版社，2002。
 28. 鍾志成等，「臺鐵列車運轉時隔之分析模式」，第 15 屆中華民國運輸學會論文集，民國 89 年 12 月，頁 609-618。
 29. 鍾志成等，「軌道容量評估方法之探討」，第 19 屆中華民國運輸學會論文集，民國 93 年 11 月。
 30. 鍾志成、張仕龍，「列車等候規則之探討」，第 16 屆中華民國運輸學會論文集，民國 90 年 11 月。
 31. 謝汶進，鐵路列車排點模式之研究，成大交管所碩士論文，民國 83 年。
 32. 謝肇桐，「移動閉塞系統」，鐵道通信信號，Vol. 32， No.2， 1996。
 33. 嚴余松、杜文，「計算機編制列車運行圖系統運用管理若干設想」，中國鐵路，Vol. 12，1999。
 34. 交通部台灣鐵路管理局網頁，網址：<http://www.railway.gov.tw>
 35. 新十大建設資訊網，網址：<http://210.200.236.10>

(二)英文部分

36. Carson, J. S.& Atala, O. M., "Using Computer Simulation for Rapid Transit Operation Strategies", Proceedings of the 1990 Winter Simulation Conference.
37. Dhingra, S.L., Marwah, B.R. & Palaniswamy, S.P., "Simulation Model for Transit Network- Modeling", Journal of Advanced Transportation, Vol.27, No.2, 1993.
38. Ferrovie dello Stato Spa – Divisione Infrastruttura, European Railways Optimisation Planning Environment – Transportation Railways Integrated Planning, EuROPE-TRIP, 2000.
39. Jovanovic, D. (1989) Improving railroad on-time performance: models, algorithms and application. Ph.D. Dissertation, Department of systems, University of Pennsylvania, Philadelphia.
40. Jovanovic, D. and Harker, P.T. (1991) Tactical scheduling of rail operations: the SCAN O system. Transportation Science. V.25. N.4. PP.46-64.
41. Kikuchi, S., "A Simulation Model of Train Travel on a Rail Transit Line", Journal of Advanced Transportation, Vol.25, No.2, 1991, pp.211-224.
42. Law, A. M. & Kelton W.D., Simulation Modeling and Analysis, 3rd Edition, McGraw-Hill International Editions, 2000.
43. Maged M. Dessouky Quan Lu & Robert C. Leahman , "Using simulation modeling to assess rail track infrastructure" , Proceedings of the 2002 Winter Simulation Conference.
44. Pacht, J., Railway Operation and Control, VTD RAIL publishing, Terrace, U.S.A., 2002.
45. Shannon, R. E., System Simulation : The Art and Science, 1976.

附錄一、各國軌道容量解析模式變數一覽表

軌道組織		模式變數定義
台灣台鐵		C_l = 路線容量(TU/h) t_u = 上行客貨列車所佔比例之混合運轉時分(min) t_d = 下行客貨列車所佔比例之混合運轉時分(min) t_s = 辦理閉塞及號誌時間(min)，目前台鐵採用 1.5min δ = 路線利用率，國外多採用 0.65~0.75，台鐵採用平均值 0.70 η = 行車制度效率因素，人工閉塞區間為 1.0，在 CTC 區間依站間區間閉塞區間的數目來制定，以兩個閉塞區間為基本值 1，每增加一個閉塞區間增加 0.1 n_t = 軌道數(單軌、雙軌、三軌)
美國		C_l = 路線容量(TU/h) t_s = 最小號誌時距(s) t_d = 瓶頸車站停車時間(s) t_m = 運轉寬裕時間(s)
歐洲	UIC Code 405	C_l = 複線區間的路線容量(TU/day) T = 計算容量的時間涵蓋範圍(min) $t_s = \sum t_{ij} * p_{ij}$ = 列車平均號誌時隔(min) t_{ij} = 後行追蹤列車 j 與先行列車 i 之號誌時隔(min) P_{ij} = 後行追蹤列車 j 與先行列車 i 之相對頻率 t_m = 列車運轉寬裕時間(min)(經驗值為 $0.67t_s$) t_z = 號誌顯示寬裕時間(min)
	IMPROVERAIL	C_l = 路線容量 T = 計算容量的時間涵蓋範圍(min) $h = \sum h_{ij} * p_{ij}$ = 平均最小運轉時隔(min) h_{ij} = 後行追蹤列車 j 與先行列車 i 之最小運轉時隔 P_{ij} = 後行追蹤列車 j 與先行列車 i 之相對頻率 δ = 路線利用率
日本	複線區間(山岸氏概算法)	C_l = 複線區間的路線容量(TU/day) h_1 = 高速列車之間轉時隔 (min) h_2 = 低速先行列車與高速續行列車之進佔運轉時隔(min) h_3 = 高速先行列車與低速續行列車之進佔運轉時隔(min) r_h = 高速列車所佔的比率 = 高速列車次數/列車總次數 r_l = 低速列車所佔的比率 = 低速列車次數/列車總次數 δ = 路線利用率，一般採 0.6

	單線區間(運轉局簡易式)	C_l =單線區間的路線容量(TU/day) t =列車於站間之平均運轉時分(min) t_s =辦理閉塞所需的時間(min)，自動閉塞制及聯動式閉塞制為 1.5min，電氣路牌式閉塞制為 2.5min δ =路線利用率，一般採 0.6
	通勤電車專用區間	C_l =路線容量(TU/day) h =最小運轉時隔(min) δ =路線利用率，一般採 0.6
中國		C_l =區間的路線容量(TU/day) t_n =接觸網(電器化路線電力供應設施)檢修封鎖時間(min) t_c =列車運行圖規定的施工封鎖時間(min) t_U =上行客貨列車區間運行時間(含停車與啟動附加時間)(min) t_D =下行客貨列車區間運行時間(含停車與啟動附加時間)(min) τ_A =A 車站之車站間隔時間(min) τ_B =B 車站之車站間隔時間(min)

資料來源：本研究整理



附錄二 模擬程式程式碼

[illegible]

```

float curve_speed[215]; //曲線速度限制
float speed_final[106]; //基本速限區和曲線半徑區的交集運算產生的速限區
float speed_final_distance[106]; //基本速限區和曲線半徑區的交集運算產生的速限區轉換點
float train_total_operation_time[train_no]; //列車存在於系統內的時間
int total_system_time=0; //系統模擬結束時間
int train_left_station_time[train_no][station_no]; //列車在系統內離站時刻
int train_join_station_time[train_no][station_no]; //列車在系統內進站時刻
int station_lines[station_no]={2,2,2,2,2,1,2,2,1,2,2,2,2,2,1,2,2}; //每站有幾個軌道
float constant_1=0; //站內股道檢查點
int capacity_temp[4][4]; //容量計算的暫存矩陣
int capacity_count[4][4]; //容量計算的權重暫存矩陣
float capacity_result[4][4]; //容量計算結果
int train_left_station_permutation[station_no][train_no]; //每站的列車離開順序
int how_many_train_passby_station[station_no]; //每站有幾輛車經過

const int count_grade=379; //有幾個坡度區段

/*-----[ 坡度值 ]-----*/
const float
grade[379]={0.97,-1.73,-8.05,-5.9,-2.1,-8.35,7.3,6.2,-2.17,1.31,2.3,7.2,8.15,0,-1.086,0,-4.07,-4.4,-0.1,-1.8,0.18,3.3,10.9,1.1,-5.2,-2.13,-9.35,-11.2,-12.1,
-10.93,8.6,10.25,8.78,9.09,2.28,0.1,-0.15,3.15,11.6,11,10.5,11.9,10.5,4.8,10.2,5.3,1,-2.8,0.18,-0.18,-0.5,-9,-8.4,-3.3,0.35,-1.7,0.8,2.5,4.8,8.5,11,5.4,2.2,
0.5,1.4,3.2,6.4,4.7,7.3,3.6,5.2,7.8,9.2,8.1,6.5,2.5,1.2,3.5,4.4,5.3,3.2,3.45,2.6,4.9,2.3,3.3,6.4,8.6,9.1,1.5,5.9,8.9,10.9,8.6,7.3,3.8,1.3,5.5,7.5,9.3,3.9,10.3,8.
2,10.4,6.7,0.6,-0.2,2.4,4.0,8.3,8,-0.8,0.9,0.4,-2.3,4.10,11.8,8.8,12.2,10.11.1,8.6,10.65,9.1,10.5,9.8,10.7,9.4,11.7,9.5,8.3,12.6,9.1,11.3,10.9,1.9,9.4,2,-1,-
7.5,-7.2,-5.2,-3.5,-0.7,1,-2.0,7,-2.45,-6.6,-5.6,-0.5,2.65,5.6,0.8,0.5,1.6,4.6,8.6,4.6,2.8,-0.15,2.5,1.4,-0.5,0.0,3,-1.0,5.2,4.6,1.1,5.0,1.3,7.8,8,10.4,8.3,11.8,
4.10,5.10,5.6,2.9,3.9,4.1,7.16,10.8,6.3,6.5,4.3,0.3,4.4,7.2,4,-0.9,1.4,6.6,3.4,3.4,9.0,0.5,2.3,5.2,8.5,10.4,9.4,7.5,1.5,7.6,5.6,9.0,4.8,6.7,9.9,10.6,9.2,8.8,10.
1,9.3,10.7,6.4,5.5,2.10,5.8,8.10,9.1,11.1,3.8,1.3,1.9,2.5,5.3,2.5,-1.5,-7.1,-10.3,-9.2,-10.4,-11.2,-10.5,-10,-8.9,-11.2,-9.2,-10.4,-9.5,-10.9,-9.5,-11.4,-8.2,-
4.8,-1.1,1,-1.5,-3,-1.9,-11.5,-9.7,-11.2,-10.4,-9.9,-3.4,0.2,0.85,0.2,-4.9,-5.9,-3.9,-5.1,-10.42,-9.8,-8.2,-11.3,-9,-6,-1.5,0.2,-9.8,-10,-9.2,-3.2,0.6,-0.7,0.9,-0
.1,-1.9,-8.3,-8.6,-9.6,-7.9,-9.5,-8.5,-9.5,-11.3,-7.7,-6.5,-3.6,-3,-9.9,-10.2,-9.5,-7.3,-6.7,-1.08,0.3,-0.9,0.4,-3.1,-11.5,-9.9,-10.8,-9.5,-10.4,-9.3,-7.3,-5.1,-4.
2,-10,-6.2,-2.3,-0.4,0.3,5.3,6.9,7.9,1.1,-2,-5.5,-7.8,-5.1,-2.0,5.0,1.1,2,-2.3,1.8,-0.8,-8.7,-10.7,-9,-10.2,-11,-8.1,-10,-10.5,-9.7,-10.7,-8.8,-10.2,-10,-8.8,-0.
3,0.2,-0.5,9.7,0.4,1.5,0.8,1.7,0,-9.3,-5.8,1.5,-2.6,-5.2,-1.8,-0.6,0.4,-0.4,0.6,0.0};

/*-----[ 坡度轉換起點 ]-----*/
float
grade_distance[379]={0.00,0.05,0.70,1.36,1.92,1.99,2.40,2.91,3.20,3.38,3.99,4.45,4.60,4.80,5.29,5.43,5.88,6.30,6.64,6.85,7.02,7.09,7.45,7.72,8.22,8.5
0.8,78.9,13.9,42.9,55.9,9.98,10.63,10.80,10.96,11.35,12.00,12.65,13.40,13.80,14.18,14.50,14.60,15.01,15.26,15.62,15.66,15.80,15.85,15.96,16.00,16.30,
16.60,16.72,16.84,17.12,17.25,17.52,17.71,17.90,18.00,18.05,18.15,18.35,18.40,18.50,18.75,18.90,19.04,19.12,19.24,19.35,19.45,19.50,19.94,20.25,2
0.40,20.80,20.95,21.10,21.44,21.62,21.85,22.10,22.27,22.39,22.46,22.52,22.74,22.95,23.08,23.20,23.38,23.50,23.65,23.75,23.94,24.05,24.11,24.17,24.
30,24.47,24.55,24.70,24.90,25.12,25.32,25.40,25.65,25.76,25.92,26.04,26.32,26.42,26.54,26.60,26.80,27.00,27.20,27.30,27.40,27.55,27.74,28.00,28.1
0,28.40,28.58,28.78,29.10,29.30,29.42,29.95,30.18,30.55,30.82,30.98,31.15,31.28,31.46,31.54,31.66,31.95,32.60,32.65,32.76,32.95,33.20,33.40,33.50,
33.65,33.90,33.95,34.05,34.27,34.36,34.60,34.66,34.80,34.95,35.10,35.45,35.58,35.66,36.00,36.26,36.55,37.10,37.42,37.65,37.95,38.25,38.52,38.60,3
8.85,38.95,39.18,39.32,39.45,39.55,39.70,39.94,40.06,40.16,40.26,40.50,40.95,41.05,41.20,41.34,41.55,41.60,41.91,42.01,42.10,42.35,42.45,42.85,43.
05,43.24,43.42,43.55,43.75,44.01,44.15,44.45,44.60,44.95,45.28,45.35,45.46,45.52,45.72,46.00,46.10,46.50,46.74,46.90,47.10,47.18,47.31,47.45,47.6
5,47.94,48.08,48.20,48.80,48.90,49.20,49.30,49.50,49.70,49.90,50.08,50.25,50.35,50.53,50.60,51.20,51.32,51.50,51.68,51.80,51.90,51.96,52.10,52.18,
52.30,52.40,52.70,53.00,53.15,53.23,53.35,53.60,53.75,53.90,54.12,54.30,54.45,54.55,54.90,55.00,55.16,55.30,55.43,55.56,55.70,55.85,56.20,56.47,5
6.55,56.65,56.85,57.00,57.25,57.55,57.75,57.90,58.35,58.58,58.72,58.90,59.05,59.12,59.25,59.70,60.05,60.65,60.92,61.02,61.20,61.50,61.75,62.05,62.
33,62.65,62.95,63.25,63.45,63.96,64.08,64.20,64.34,64.42,64.55,65.15,65.50,66.20,66.45,66.60,66.85,67.10,67.38,67.59,67.70,67.87,68.02,68.10,68.5
0,68.80,69.10,69.35,69.55,69.70,70.10,70.26,70.30,70.36,70.55,70.80,71.33,71.45,71.65,72.29,72.50,72.63,72.70,73.10,73.20,73.45,73.75,74.04,74.10,
74.16,74.26,74.36,74.44,74.66,74.85,75.10,75.35,75.50,76.25,76.52,77.02,77.14,77.28,77.48,77.64,77.74,77.95,78.35,78.67,78.88,79.35,79.78,80.06,8
0.25,80.96,81.22,81.85,82.06,82.46,82.60,82.85,83.60,83.90,84.05,84.35,84.50};

int count_speed=55; //速度限制區段個數
/*const*/ float speed[55]={ //每個速度限制區段的速限值

36.11111111,20.83333333,36.11111111,20.83333333,36.11111111,19.44444444,36.11111111,20.83333333,36.11111111,19.44444444,36.11111111,
20.83333333,36.11111111,19.44444444,36.11111111,19.44444444,36.11111111,20.83333333,36.11111111,20.83333333,36.11111111,19.44444444,
36.11111111,22.22222222,36.11111111,20.83333333,36.11111111,20.83333333,36.11111111,20.83333333,36.11111111,20.83333333,36.11111111,
19.44444444,36.11111111,18.05555556,36.11111111,22.22222222,36.11111111,19.44444444,36.11111111,26.38888889,36.11111111,25,
36.11111111,25,36.11111111,23.61111111,36.11111111,23.61111111,36.11111111,23.61111111,36.11111111,26.38888889,36.11111111};

/*const*/ float speed_distance_start[55]={ //每個速度限制區段的起點
0,14647.35,15143.2,16869.45,17246.6,19112.7,19258.7,19288.7,19439.4,22323.55,
22448.5,22585.95,22953.55,23120.3,23256.1,26470.5,27327.3,27826.4,27952.05,28176.4,
28313.5,28628.25,29104.35,29169.65,29863.30242.25,30454.65,30833.65,31151.3,31322.55,
31820.85,34338.34833.15,35609.9,35673.9,74120.85,74190.4,76296.2,76399.9,76624.2,
76995.5,77408.5,77884.55,78652.3,78691.4,78830.6,78871.7,82674.5,82754.1,82956.9,
83036.9,83525.8,83602.05,83645.2,83718.2};

/*const*/ float speed_distance_end[55]={ //每個速度限制區段的終點
14647.35,15143.2,16869.45,17246.6,19112.7,19258.7,19288.7,19439.4,22323.55,
22448.5,22585.95,22953.55,23120.3,23256.1,26470.5,27327.3,27826.4,27952.05,28176.4,
28313.5,28628.25,29104.35,29169.65,29863.30242.25,30454.65,30833.65,31151.3,31322.55,
31820.85,34338.34833.15,35609.9,35673.9,74120.85,74190.4,76296.2,76399.9,76624.2,
76995.5,77408.5,77884.55,78652.3,78691.4,78830.6,78871.7,82674.5,82754.1,82956.9,

```

```

        83036.9,83525.8,83602.05,83645.2,83718.2,84.40};
int count_station=station_no;//車站數目
int station_stop_time[4][station_no]={
    //外部輸入
};
    //每車種每站停站時間(目前有 4 個車種,17 個站)
/*const*/ float station_distance[station_no]=    //每個車站的位置點(共 17 站)
    {0,6358,9191,13578,18986,22867,27259,35456,41395,45345,51141,55192,62000,67682,73815,78625,84475};
/*const*/ int delta_t=1;//時間間隔[ set = 1 ]
/*const*/ float train_acceleration[train_no];//每輛車模擬的加速度(目前有 36 輛車)
float train_velocity[train_no];//每輛車模擬的速度(目前有 36 輛車)
float train_distance[train_no];//每輛車模擬的距離(目前有 36 輛車)
float temp_envelop[3000][2];//程式推算包絡線時的暫存矩陣(最大只能有 3000 個時間檢查點)
float speed_limit_envelop[3000][2];//速限區速度包絡線矩陣(最大只能有 3000 個時間檢查點)
float station_stop_envelop[1500][2];//停站點的速度包絡線矩陣(最大只能有 1500 個時間檢查點)
float speed_section_point[4][106];//各速限區往回推算的終點(最大 106 個跟速限區最大數目相同)
float station_stop_point[4][station_no];//各停站點往回推算的終點(最大 17 個跟車站最大數目相同)
float speed_at_station=0;//列車在車站的速度
int numbersof_speed_limit_envelop;//速限區速度包絡線矩陣的總元素數目
int numbersof_station_stop_envelop;//停站點速度包絡線矩陣的總元素數目
int system_train;//系統內的列車數

/*****變數宣告結束*****/

/*****副程式宣告開始*****/

/*=====[ 讀 外 部 檔 ]=====*/
/*int read_train_start_time(int train_start_time[train_no]);*/
/*int read_train_task(bool train_task[train_no][station_no]);*/
/*int read_train_priority(int train_priority[train_no]);*/
/*=====*/

void Calculate_Speed_Envelop(int);    //使用 pre_Calculate_Envelop()得到速限區速度包絡線矩陣
void Calculate_StationStop_Envelop(int);    //使用 pre_Calculate_Envelop()得到停站點的速度包絡線矩陣
int pre_Calculate_speed_Envelop(int,int);    //用倒推法計算出速限區包絡線數值
int pre_Calculate_station_Envelop(int,int);    //用倒推法計算出停站包絡線數值
float Calculate_pre_speed(float,float,int);    //計算前一個時間間隔的速度(用倒推法算出包絡線矩陣用)
float Calculate_pre_position(float,float,int);    //計算前一個時間間隔的列車位置(用倒推法算出包絡線矩陣用)

void train_speed_up(int);    //列車加速
void train_speed_down(int);    //列車減速
void train_speed_same(int);    //列車等速

void train_swap(int,int);//兩個列車的資料交換
float find_station_stop_envelop_speed_from_distance(float,int);    //傳入目前位置得到要停站的停站包絡線速度
float find_station_not_stop_envelop_speed_from_distance(float,int);    //傳入目前位置得到不停站的停站包絡線速度
float find_acceleration_with_grade(float,float);    //傳入(位置,原加速度)考慮坡度傳回加速度
float find_deceleration_with_grade(float,float);    //傳入(位置,原減速度)考慮坡度傳回減速度
float find_time_of_ETA(int);    //找出從站內股道檢查點到車站的模擬運行時間
int find_train_index_from_start_permutation(int);    //傳入發車順序得到離開系統的順序
void print_capacity(int);    // 列印某站的 capacity

/*****副程式宣告結束*****/

/*=====
||                                ||
=====*/

/*****主程式開始*****/
int main(){

/*=====[ 讀 外 部 檔 ]=====*/
    read_train_start_time(train_start_time);
    read_train_task(train_task);
    read_train_priority(train_priority);
/******/

for(int i=0;i<379;i++)grade_distance[i]*=1000;//公里換成公尺    //-----坡度區段起始點單位轉換 km->m -----//
for(int i=0;i<379;i++)curve_distance[i]*=1000;//公里換成公尺    //-----曲線半徑區段起始點單位轉換 km-->m ----//
for(int i=0;i<215;i++)if(curve[i]<0)curve[i]*=-1;    //曲線半徑取絕對值

```

```

//-----從曲線半徑產生曲線半徑的速限-----開始-----
for(int i=0;i<215;i++){
    if(curve[i]==0){curve_speed[i]=0;}
    if(curve[i]<=900&&curve[i]>800){curve_speed[i]=34.72222222;}
    else if(curve[i]<=800&&curve[i]>700){curve_speed[i]=27.77777778;}
    else if(curve[i]<=700&&curve[i]>600){curve_speed[i]=30.55555556;}
    else if(curve[i]<=600&&curve[i]>500){curve_speed[i]=27.77777778;}
    else if(curve[i]<=500&&curve[i]>450){curve_speed[i]=25;}
    else if(curve[i]<=450&&curve[i]>400){curve_speed[i]=23.61111111;}
    else if(curve[i]<=400&&curve[i]>350){curve_speed[i]=22.22222222;}
    else if(curve[i]<=350&&curve[i]>300){curve_speed[i]=20.83333333;}
    else if(curve[i]<=300&&curve[i]>250){curve_speed[i]=19.44444444;}
    else if(curve[i]<=250&&curve[i]>225){curve_speed[i]=18.05555556;}
    else if(curve[i]<=225&&curve[i]>200){curve_speed[i]=16.66666667;}
    else if(curve[i]<=200&&curve[i]>150){curve_speed[i]=15.27777778;}
    else if(curve[i]<=150&&curve[i]>125){curve_speed[i]=13.88888889;}
    else if(curve[i]<=125&&curve[i]>100){curve_speed[i]=12.5;}
    else if(curve[i]<=100&&curve[i]>0){curve_speed[i]=11.11111111;}
}
//-----從曲線半徑產生曲線半徑的速限-----結束-----

//-----產生最終速限區-----
int original_speed_index=0,curve_speed_index=0,speed_final_index=0;
while(curve_speed_index<215){
    if(curve_speed[curve_speed_index]==0){curve_speed_index++;continue;}
    if(speed_distance_start[original_speed_index]==curve_distance[curve_speed_index]){
        speed_final[speed_final_index]=speed[original_speed_index]<curve_speed[curve_speed_index]
            ?speed[original_speed_index]
            :curve_speed[curve_speed_index];
        speed_final_distance[speed_final_index]=speed_distance_start[original_speed_index];
        original_speed_index++;curve_speed_index++;speed_final_index++;
    }
    else if(speed_distance_start[original_speed_index]<curve_distance[curve_speed_index]){
        speed_final[speed_final_index]=speed[original_speed_index];
        speed_final_distance[speed_final_index]=speed_distance_start[original_speed_index];
        original_speed_index++;speed_final_index++;
    }
    else if(speed_distance_start[original_speed_index]>curve_distance[curve_speed_index]){
        speed_final[speed_final_index]=curve_speed[curve_speed_index];
        speed_final_distance[speed_final_index]=curve_distance[curve_speed_index];
        curve_speed_index++;speed_final_index++;
    }
}

//-----產生包絡線-----
int i;
//依車種,加減速產生不同包絡線
for(i=0;i<4;i++){
    //-----產生不停站的停站包絡線-----開始-----
    speed_at_station=2.777777;
    Calculate_StationStop_Envelop(i);
    speed_at_station=0;
    for(int j2=0;j2<1500;j2++){
        for(int k2=0;k2<2;k2++){
            station_not_stop_envelop_final[i][j2][k2]=station_stop_envelop[j2][k2];
        }
    }
    for(int a=0;a<station_no;a++)station_not_stop_point[i][a]=station_stop_point[i][a];
    //-----產生不停站的停站包絡線-----結束-----

    //-----產生速限包絡線及要停站的停站包絡線-----開始-----
    Calculate_Speed_Envelop(i);
    Calculate_StationStop_Envelop(i);
    for(int j=0;j<1500;j++){
        for(int k=0;k<2;k++){
            speed_limit_envelop_final[i][j][k]=speed_limit_envelop[j][k];
        }
    }
    for(int j1=0;j1<1500;j1++){
        for(int k1=0;k1<2;k1++){
            station_stop_envelop_final[i][j1][k1]=station_stop_envelop[j1][k1];
        }
    }
    //-----產生速限包絡線及要停站的停站包絡線-----結束-----
}

//-----修正列車優先順序-----
for(i=0;i<train_no;i++){
    train_priority[i]--;
}

```

```

//根據列車優先順序-----產生列車加減速度矩陣-----
for(i=0;i<train_no;i++){
    train_deceleration[i]=deceleration[train_priority[i]];
    train_acceleration[i]=acceleration[train_priority[i]];
}
//-----若列車 i 在第一站,其[系統發車時間][出發時間]需加上第一站的停站時間
for(i=0;i<train_no;i++){
    if(train_task[i][0]==1){
        train_start_time[i]+=station_stop_time[train_priority[i]][0];
        train_departure_time[i]+=station_stop_time[train_priority[i]][0];
    }
}
//-----產生列車終點站暫存矩陣-----開始-----
int train_int_temp[train_no]={0};
for(int i=0;i<train_no;i++){
    for(int j=0;j<station_no;j++){
        if(train_task[i][j]==true){train_final_station[i]+=train_int_temp[i]+1;train_int_temp[i]=0;}
        else{train_int_temp[i]++;}
    }
}
for(int i=0;i<train_no;i++)train_final_station[i]--;
//-----產生列車終點站暫存矩陣-----結束-----

/*****手動設定終點站*****(因為有些車中途即到達終點站)*****/
// train_final_station[發車序號-1]=[站號-1]
train_final_station[5]=9;!-->2525
train_final_station[6]=16;!-->1021(因為此列車僅停靠[松山][台北][板橋]->[台中]跳太遠)
train_final_station[7]=4;!-->2714
train_final_station[10]=9;!-->2161
train_final_station[11]=11;!-->2535
train_final_station[13]=4;!-->56
train_final_station[15]=4;!-->48
train_final_station[16]=9;!-->2529
train_final_station[19]=4;!-->1054
train_final_station[20]=11;!-->2726
train_final_station[22]=4;!-->58
train_final_station[27]=11;!-->2537
train_final_station[28]=4;!-->1066
train_final_station[32]=4;!-->1088
/*****/

//預設每輛車都在系統中
for(int i=0;i<train_no;i++)train_in_system[i]=true;
//產生發車順序矩陣
for(int i=0;i<train_no;i++)train_start_permutation[i]=i+1;
for(int i=0;i<4;i++){
    for(int j=0;j<4;j++){
        {
            capacity_count[i][j]=0;
            capacity_temp[i][j]=0;
            capacity_result[i][j]=0.0;
        }
    }
}
//將每站有幾輛列車經過歸零
for(int i=0;i<station_no;i++)how_many_train_passby_station[i]=0;

/*****/
/*****/

int simulation_time=0; //模擬秒數
int new_train_index=0; //目前要新增進入系統的列車
int current_train_index=0; //從列車陣列的第 0 個開始模擬
int train_left=0; //已離開幾輛列車
system_train=0; //預設系統內車輛數為 0

while(simulation_time<30000){ //模擬最久直到 30000 秒後強制停止
    //若系統模擬時間=系統發車時間,則系統內列車數+1,列車速度預設為 0,目前要新增進入系統的列車 index 指定為下一輛列車
    if(train_start_time[new_train_index]==simulation_time){
        system_train++;
        train_velocity[new_train_index]=0;
        new_train_index++;
    }
    //目前列車為欲離開列車則進入以下,一直做到目前列車 index=系統內列車數
    for(current_train_index=0;current_train_index<system_train;current_train_index++){
        //列車進站時刻測試輸入位置//////////
        if(train_velocity[current_train_index]==0&&train_at_station[current_train_index]==0){

```

```

        train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
//若列車不在系統內,continue
if(train_in_system[current_train_index]==false){
    continue;//-----[ A_in ]-----//
} ///如果列車被標示為離站/////當迴圈跑到他時/////跳過//////////
//如果列車已走到終點
if(train_distance[current_train_index]>=station_distance[train_final_station[current_train_index]]){
    //若列車任務排停終點站-->(則站內列車數-1,將列車設為不存在於系統中,已離開系統列車數+1)
    if(train_task[current_train_index][train_final_station[current_train_index]]==1)
        how_many_trains_in_station[train_final_station[current_train_index]]--; //系統內列車數-1
        //將列車設為不存在於系統中//////////
        train_total_operation_time[current_train_index]=simulation_time-train_start_time[current_train_index];
        train_in_system[current_train_index]=false;
        train_left++;//////////已離開系統列車數+1

//////////測試結果列印//////////
    cout<<"permutation="<<train_start_permutation[current_train_index]
        <<" priority="<<train_priority[current_train_index]
        <<" time="<<simulation_time<<"(s)="
        <<simulation_time/3600<<"h "
        <<(int)((float)simulation_time/3600-simulation_time/3600)*60<<"m "
        <<(((float)simulation_time/3600-simulation_time/3600)*60
        -(int)((float)simulation_time/3600-simulation_time/3600)*60)<<"s"<<endl;
    //若[離開系統列車數]=[模擬列車數]則終止 simulation
    if(train_left==train_no){
        cout<<endl<<"All trains have been left system."<<endl;
        total_system_time=simulation_time+station_stop_time[train_priority[current_train_index]][station_no-1];
        break;//////////終止 simulation//////////
    }
}

//////////-----列印檢查[是否有列車速度<0]
if(train_velocity[current_train_index]<0){
    cout<<"system_train="<<system_train<<" time="<<simulation_time<<" index="<<current_train_index<<" "<<train_distance[current_train_index]
        <<" "<<train_velocity[current_train_index]<<" "<<train_priority[current_train_index]
        <<" "<<train_next_station[current_train_index]<<" "<<train_departure_time[current_train_index]
        <<" "<<station_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]
        <<" "<<station_distance[train_next_station[current_train_index]]
        <<" "<<train_at_station[current_train_index]
        <<endl;
}

//檢查列車是否在站
if(train_velocity[current_train_index]>0){
    train_at_station[current_train_index]=0;
}
//若列車里程>=下一站里程 且 列車出發時間<=系統時間
if(train_distance[current_train_index]>=station_distance[train_next_station[current_train_index]]
    &&train_departure_time[current_train_index]<=simulation_time){
    //儲存列車離站時刻
    train_left_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
    how_many_train_passby_station[train_next_station[current_train_index]]++;
    train_left_station_permutation[train_next_station[current_train_index]]
        [how_many_train_passby_station[train_next_station[current_train_index]]-1]
        =train_start_permutation[current_train_index];

    //列車排定停站 則站內列車數-1,
    if(train_task[current_train_index][train_next_station[current_train_index]]==1)
        how_many_trains_in_station[train_next_station[current_train_index]]--;
    ///指定列車下一站+1
    train_next_station[current_train_index]++;
}

//若列車里程>速限區終點 則指定下一個速限區+1

if(train_distance[current_train_index]>speed_final_distance[train_next_speed_section[current_train_index]+1]){train_next_speed_section[current_train_index]++;}
//*****安全進站間距*****
if(train_distance[current_train_index]>station_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]
    -2*train_long-constant_1
    &&current_train_index!=0){
    float ETA_temp=find_time_of_ETA(current_train_index)+simulation_time;
    //判斷站內股道數 如果站內股道數為一
    if(station_lines[train_next_station[current_train_index]]==1){
        //同向列車進站號誌安全時距(站內停靠同一軌道)判斷
        if(ETA_temp>

```

```

2*train_left_station_time[current_train_index-1][train_next_station[current_train_index]]
+sqrt(2*550/train_acceleration[current_train_index-1])
+4){
    //符合同向列車進站號誌安全時距(站內停靠同一軌道)
    // do nothing
}
else{
    //不符同向列車進站號誌安全時距(站內停靠同一軌道)
    if(train_velocity[current_train_index]>0)
        train_speed_down(current_train_index);////減速
    if(train_velocity[current_train_index]<0)
        train_velocity[current_train_index]=0;
    if(train_velocity[current_train_index]<0)cout<<"Error1*****"<<endl;
    continue;
}
}
////判斷站內股道數 如果站內股道數為二
else if(station_lines[train_next_station[current_train_index]]==2){
    ////判斷站內軌道佔用情形 如果站內股道數為二 如果站內沒有車或有一輛車
    if(how_many_trains_in_station[train_next_station[current_train_index]]<=1){
        ////同向列車進站號誌安全時距(站內停靠不同軌道)判斷
        if(ETA_temp>
            2*train_left_station_time[current_train_index-1][train_next_station[current_train_index]]
            +15){
            //符合同向列車進站號誌安全時距(站內停靠不同軌道)
            // do nothing
        }
        else{
            //不符同向列車進站號誌安全時距(站內停靠不同軌道)
            if(train_velocity[current_train_index]>0)
                train_speed_down(current_train_index);////減速
            if(train_velocity[current_train_index]<0)
                train_velocity[current_train_index]=0;
            if(train_velocity[current_train_index]<0)cout<<"Error2*****"<<endl;
            continue;
        }
    }
}
////判斷站內軌道佔用情形 如果站內股道數為二 如果站內有兩輛車
else if(how_many_trains_in_station[train_next_station[current_train_index]]==2){
    ////同向列車進站號誌安全時距判斷(站內停靠同一軌道)
    if(ETA_temp>
        2*train_left_station_time[current_train_index-2][train_next_station[current_train_index]]
        +sqrt(2*550/train_acceleration[current_train_index-2])
        +4){
        //符合同向列車進站號誌安全時距(站內停靠同一軌道)
        // do nothing
    }
    else{
        //不符同向列車進站號誌安全時距(站內停靠同一軌道)
        if(train_velocity[current_train_index]>0)
            train_speed_down(current_train_index);////減速
        if(train_velocity[current_train_index]<0)
            train_velocity[current_train_index]=0;
        if(train_velocity[current_train_index]<0)cout<<"Error3*****"<<endl;
        continue;
    }
}
}
////站內停了三輛車以上
else{
    cout<<"ERROR:more than 3 trains in station("<<train_next_station[current_train_index]<<")!!!!!!!!!!"<<endl;
}
}

}

//如果到停站點

if(train_distance[current_train_index]>=station_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]){
    //如果排定要停站
    if(train_task[current_train_index][train_next_station[current_train_index]]==1){
        //如果已到站
        if(train_distance[current_train_index]>=station_distance[train_next_station[current_train_index]]-30){

```

```

        if(system_train-1!=current_train_index){//如果不為末班車
            while(true){
                if(train_priority[current_train_index+1]<train_priority[current_train_index]){//若後車列車等級<本列車列車
等級
//*****錯車門檻值******(因為最小站距台北-萬華為
2.833km)
                if(train_distance[current_train_index]-train_distance[current_train_index+1]<=1500){//若後車距離本車<
門檻值
                    /*****=[sub3]=*****低等級列車錯車後再出發時間
                    *****/
                    //若列車出發時間>系統時間,
                    if(train_departure_time[current_train_index]>simulation_time){
                        //速度暫存=min(最大列車營運速度,下個速限區速度)
                        float
temp_velocity=(train_next_speed_section[current_train_index+1]<=train_max_speed[train_priority[current_train_index]]
?train_next_speed_section[current_train_index+1]:train_max_speed[train_priority[curr
ent_train_index]]);
                        //列車再出發時間=目前系統時間+本列車停站時間+後車停站時間+9+最小離站時隔+車
長/速度
                        train_departure_time[current_train_index]=simulation_time
+(train_start_permutation[current_train_index]==0&&train_next_station[current_train_index]==11?480
:(train_start_permutation[current_train_index]==1&&train_
next_station[current_train_index]==4?420
:(train_start_permutation[current_train_index]==4&&train_
next_station[current_train_index]==11?420
:(train_start_permutation[current_train_index]==12&&train_
_next_station[current_train_index]==7?420
:station_stop_time[train_priority[current_train_index]][train_
_next_station[current_train_index]])))))
+station_stop_time[train_priority[current_train_index+1]][train_next_station[current_train_index+1]]
+9+unknow_variable+(((int)temp_velocity*3)
/(find_acceleration_with_grade(train_distance[current_train_index+1],train_acceleration[current_train_index+1])*2)
+((int)train_long/(int)temp_velocity);
                        train_swap(current_train_index,current_train_index+1);//current_train_index--;
                    }
                    //若列車不在站內 and 列車出發時間<=系統時間
                    else if(train_at_station[current_train_index]==0){
                        //速度暫存=min(最大列車營運速度,下個速限區速度)
                        float
temp_velocity=(train_next_speed_section[current_train_index+1]<=train_max_speed[train_priority[current_train_index]]
?train_next_speed_section[current_train_index+1]:train_max_speed[train_priority[curr
ent_train_index]]);
                        //列車再出發時間=目前系統時間+本列車停站時間+後車停站時間+9+最小離站時隔+車
長/速度
                        train_departure_time[current_train_index]=simulation_time
+(train_start_permutation[current_train_index]==0&&train_next_station[current_train_index]==11?480
:(train_start_permutation[current_train_index]==1&&train_
next_station[current_train_index]==4?420
:(train_start_permutation[current_train_index]==4&&train_
next_station[current_train_index]==11?420
:(train_start_permutation[current_train_index]==12&&train_
_next_station[current_train_index]==7?420
:station_stop_time[train_priority[current_train_index]][train_
_next_station[current_train_index]])))))
+station_stop_time[train_priority[current_train_index+1]][train_next_station[current_train_index+1]]
+9+unknow_variable+(((int)temp_velocity*3)
/(find_acceleration_with_grade(train_distance[current_train_index+1],train_acceleration[current_train_index+1])*2)
+((int)train_long/(int)temp_velocity);
                        //若列車不在站內 則站內停站列車數+1,將列車設為停在站內
                        if(train_at_station[current_train_index]==0){
                            how_many_trains_in_station[train_next_station[current_train_index]]++;//則站內停站
列車數+1
                        train_distance[current_train_index]=station_distance[train_next_station[current_train_index]];
                        train_velocity[current_train_index]=0;
                        train_at_station[current_train_index]=1;//////////將列車設為停在站內
                        ////////////列車進站時刻//////////

```



```

train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
    }
    //swap
    train_swap(current_train_index,current_train_index+1);//current_train_index--;
    }
    //若後車距離本車>=門檻值
    else{
        break;////////跳出掃描後車的 loop////////
    }
    //若後車列車等級>=本列車列車等級
    else{
        break;////////跳出掃描後車的 loop////////
    }
}
//////////如果沒有要 SWAP 的話////////停站時間的增加//////////
//若列車不在站內， 則 [列車離站時間]=現在模擬時間+停站時間
if(train_at_station[current_train_index]==0){
    train_departure_time[current_train_index]=simulation_time
    +(train_start_permutation[current_train_index]==0&&train_next_station[current_train_index]==11?480
next_station[current_train_index]==4?420
next_station[current_train_index]==11?420
_next_station[current_train_index]==7?420
_next_station[current_train_index])));
    train_velocity[current_train_index]=0;
    ////////////列車進站時刻//////////
    train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
}
//若列車不在站內----> 則 站內列車數+1
if(train_at_station[current_train_index]==0){
    how_many_trains_in_station[train_next_station[current_train_index]]++;
    train_distance[current_train_index]=station_distance[train_next_station[current_train_index]];
    train_velocity[current_train_index]=0;
    ////////////列車進站時刻//////////
    train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
}
//////////列車指定停於站內//////////
train_at_station[current_train_index]=1;
//////////列車進站時刻//////////
train_velocity[current_train_index]=0;
train_distance[current_train_index]=station_distance[train_next_station[current_train_index]];
//////////不同軌道最小離站時隔//////////開始//////////a//
//////當停站到可以離站時//////////
if(train_departure_time[current_train_index]==simulation_time){
    //如果不是系統內最靠近終點站的車
    if(current_train_index>0){
        //////////套用最小離站時隔公式//////////開始////
        float speed_tempa=speed_final[train_next_speed_section[current_train_index]]
<train_max_speed[train_priority[current_train_index]]?speed_final[train_next_speed_section[current_train_index]]
:train_max_speed[train_priority[current_train_index]];
        float time_tempa=train_departure_time[current_train_index]-train_departure_time[current_train_index-1];
        float
time_temp=sqrt((2*speed_tempa)/find_acceleration_with_grade(train_distance[current_train_index],train_acceleration[current_train_index]))+4;
        if(time_tempa<time_temp){
            train_departure_time[current_train_index]+=(int)time_temp-(int)time_tempa;
        }
        //////////套用最小離站時隔公式//////////結束////
    }
}
//////////不同軌道最小離站時隔//////////結束//////////a//
continue;//-----A_in-----//
}
//////////-----//////////未班車的停站時間增加//////////開始//////////
else{
    //若列車不在站內， 列車出發時間=現在時間+本列車停站時間
    if(train_at_station[current_train_index]==0){

```

```

train_departure_time[current_train_index]=simulation_time+(train_start_permutation[current_train_index]==0&&train_next_station[current_train_index]==11?480
                                                                    : (train_start_permutation[current_train_index]==1&&train_
next_station[current_train_index]==4?420
                                                                    : (train_start_permutation[current_train_index]==4&&train_
next_station[current_train_index]==11?420
                                                                    : (train_start_permutation[current_train_index]==12&&train_
_next_station[current_train_index]==7?420
                                                                    : station_stop_time[train_priority[current_train_index]][train_
_next_station[current_train_index]]));
    train_velocity[current_train_index]=0;
    train_distance[current_train_index]=station_distance[train_next_station[current_train_index]];
    ////////////列車進站時刻//////////
    train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
}
//若列車不在站內---->, 則 站內列車數+1
if(train_at_station[current_train_index]==0){
    how_many_trains_in_station[train_next_station[current_train_index]]++;
    train_distance[current_train_index]=station_distance[train_next_station[current_train_index]];
    train_velocity[current_train_index]=0;
    ////////////列車進站時刻//////////
    train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;
}
//////// 指定列車停於站內,
train_at_station[current_train_index]=1;
//////////不同軌道最小離站時隔//////////開始//////////a//
if(train_departure_time[current_train_index]==simulation_time){
    //////////如果不是系統內最靠近終點站的車
    //////////套用最小離站時隔公式//////////開始////
    float speed_tempa=speed_final[train_next_speed_section[current_train_index]]
<train_max_speed[train_priority[current_train_index]]?speed_final[train_next_speed_section[current_train_index]]
    :train_max_speed[train_priority[current_train_index]];
    float time_tempa=train_departure_time[current_train_index]-train_departure_time[current_train_index-1];
    float
time_temp=sqrt((2*speed_tempa)/find_acceleration_with_grade(train_distance[current_train_index],train_acceleration[current_train_index]))+4;
    if(time_tempa<time_temp){
        train_departure_time[current_train_index]+=(int)time_temp-(int)time_tempa;
    }
    //////////套用最小離站時隔公式//////////結束////
//
}
//////////不同軌道最小離站時隔//////////結束//////////a//
continue;//-----A_in-----/////
}
//////////-----未班車的停站時間增加//////////結束//////////
//若列車出發時間<模擬時間, 則 列車加速,出站
if(train_departure_time[current_train_index]<simulation_time){
    train_speed_up(current_train_index);
    continue;//-----A_in-----/////
}
}
//////////-----/如果未到站
else{

/*sub1 *****
//若列車里程>停站包絡線起點里程, 則 列車減速

if(train_distance[current_train_index]>station_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]{
    ////////////取得列車在此位置的停站包絡線速度限制//////////
    float
speed_temp=find_station_stop_envelop_speed_from_distance(train_distance[current_train_index],train_priority[current_train_index]);
    if(train_velocity[current_train_index]>speed_temp){
        train_speed_down(current_train_index);
        continue;//-----A_in-----/////
    }
}
//若列車速度>列車服務速度, 則減速
if(train_velocity[current_train_index]>train_max_speed[train_priority[current_train_index]]){
    train_speed_down(current_train_index);
    continue;//-----A_in-----/////
}
}

```

```

        //若列車速度=列車服務速度，則等速
    else
    if(train_velocity[current_train_index]==train_max_speed[train_priority[current_train_index]]){train_speed_same(current_train_index);continue;}
    else{
        //若列車里程>=速限區起點
    if(train_distance[current_train_index]>=speed_section_point[train_priority[current_train_index]][train_next_speed_section[current_train_index]]){
        //若列車速度>列車服務速度，則減速
        if(train_velocity[current_train_index]>train_max_speed[train_priority[current_train_index]]){
            train_speed_down(current_train_index);
            continue;//-----A_in-----/////
        }
    }
    //若列車里程<速限區起點
    else{
        train_speed_up(current_train_index);//else 列車加速
        continue;//-----A_in-----/////
    }
    }
}
}
//////////如果不停站時//////////
else{
    ////////////如果列車位置>不停站的停站包絡線起點//////////
    if(train_distance[current_train_index]>=station_not_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]){
        ////////////列車速度比 25km/hr 高時---a->減速
        if(train_velocity[current_train_index]>=2.777777){
            train_speed_down(current_train_index);////減速
            continue;//-----A_in-----/////
        }
    }
    if(train_distance[current_train_index]==station_distance[train_next_station[current_train_index]]){
        ////////////列車進站時刻//////////
        train_join_station_time[current_train_index][train_next_station[current_train_index]]=simulation_time;//newnew
    }
}
}

//////////列車安全間距//////////開始//////////

//若不為系統中運行之最前列車
if(current_train_index!=0){
    ////////////計算還在系統內的前車是系統內的第幾輛車//////////開始//////////
    int temp_count_front_train_index=current_train_index-1;//設定變數::計算前車用的指標
    //當前車中途服務結束,列車不在系統內
    while(train_in_system[temp_count_front_train_index]==false){
        if(temp_count_front_train_index==0)break;/////離開計算的迴圈
        temp_count_front_train_index--;}
    ////////////計算還在系統內的前車是系統內的第幾輛車//////////結束//////////
    //若前車速率不等於 0,意即前車不在站內---->做安全間距的判斷
    if(train_velocity[temp_count_front_train_index]!=0){
        //
        cout<<"error1 *****" <<endl;
        //若前車里程-本車里程<列車安全間距門檻值
        if(train_distance[temp_count_front_train_index]-train_distance[current_train_index]
            <train_velocity[current_train_index]
            *(
                (train_velocity[current_train_index]
                /(-2*find_deceleration_with_grade(train_distance[current_train_index],train_deceleration[current_train_index]))
                )+15
            )-
            train_velocity[current_train_index-1]
            *(
                ( train_velocity[current_train_index-1]/
                (-2*emergency)
                )
            )+15
            )+train_long*2
        ){
            //若前車速度<本車速度,-----> 則減速
            if(train_velocity[temp_count_front_train_index]<train_velocity[current_train_index]+1){
                train_speed_down(current_train_index);////減速//////////
            }
        }
    }
}

```



```

for(int i=0;i<train_no;i++){
    cout<<train_start_permutation[i]<<endl;
for(int j=0;j<station_no;j++){
    cout<<j+1<<" in " <<train_join_station_time[i][j]<<endl;
    cout<<j+1<<" out " <<train_left_station_time[i][j]<<endl;
}cout<<endl;system("pause");
}cout<<endl;

/*****
//////////列車進站離站時刻列印//////////結束//////////
system("pause");
//////////
cout<<"Total System Time : " <<total_system_time<<endl;
system("pause");

return 0;////主程式結束點//////////
}
/*****主程式結束*****/

/*****副程式內容開始*****/

//////////速限包絡線副程式//////////
void Calculate_Speed_Envelop(int train_priority_index){//////////傳入列車車種
    speed_limit_envelop[0][0]=station_distance[0];//////////從第一個速限區的位置開始計算距離
    speed_limit_envelop[0][1]=train_max_speed[train_priority_index];//////////從第一個速限區的速限開始計算速度
    int index_temp=1,temp_count;//////////計數用暫存變數
    //////////跑一個速限區的速限包絡線的迴圈//////////開始//////////
    for(int speed_limit_index_temp=0;speed_limit_index_temp<count_speed;speed_limit_index_temp++){
        //////////呼叫使用倒推法計算包絡線的副程式//////////
        temp_count=pre_Calculate_speed_Envelop(speed_limit_index_temp,train_priority_index);

    //////////將使用倒推法算出一個速限區包絡線倒轉順序//////////開始//////////
        if(temp_envelop[temp_count][1]<speed_limit_envelop[index_temp-1][1])index_temp1--;
        for(int temp_j=index_temp1;temp_j<index_temp1+temp_count;temp_j++){
            for(int temp_k=0;temp_k<2;temp_k++){
                speed_limit_envelop[temp_j][temp_k]=temp_envelop[index_temp1+temp_count-temp_j-1][temp_k];
            }
            index_temp1+=temp_count;
            speed_limit_envelop[index_temp1][0]=speed_distance_end[speed_limit_index_temp];
            speed_limit_envelop[index_temp1][1]=train_max_speed[train_priority_index];
            index_temp1++;
    //////////將使用倒推法算出一個速限區包絡線倒轉順序//////////結束//////////
        }
        //////////跑一個速限區的速限包絡線的迴圈//////////結束//////////
        speed_limit_envelop[index_temp1][0]=station_distance[count_station-1];
        speed_limit_envelop[index_temp1][1]=train_max_speed[train_priority_index];
        numbersof_speed_limit_envelop=index_temp1-1;
    }
    //////////停站包絡線副程式//////////
    void Calculate_StationStop_Envelop(int train_priority_index){
        station_stop_envelop[0][0]=station_distance[0];
        station_stop_envelop[0][1]=train_max_speed[train_priority_index];
        int index_temp=1,temp_count;
        for(int station_index_temp=1;station_index_temp<count_station;station_index_temp++){
            temp_count=pre_Calculate_station_Envelop(station_index_temp,train_priority_index);
            for(int index_temp3=index_temp1;index_temp3<index_temp1+temp_count;index_temp3++){
                for(int index_temp4=0;index_temp4<2;index_temp4++){
                    station_stop_envelop[index_temp3][index_temp4]=temp_envelop[index_temp1+temp_count-index_temp3-1][index_temp4];
                }
                index_temp1+=temp_count;
                station_stop_envelop[index_temp1][0]=station_distance[station_index_temp]+0.01;
                station_stop_envelop[index_temp1][1]=train_max_speed[train_priority_index];
                index_temp1++;
            }
            numbersof_station_stop_envelop=index_temp1-2;
        }
    }
    //////////預先計算速限包絡線副程式//////////
    int pre_Calculate_speed_Envelop(int index,int train_priority_index){
        int count=0;
        float test_speed=speed_final[index];
        float test_distance=speed_final_distance[index];
        while(true){
            if(test_speed>=train_max_speed[train_priority_index]||test_distance<=0){
                test_speed=train_max_speed[train_priority_index];
                temp_envelop[count][0]=test_distance<=0?0:test_distance;
                temp_envelop[count][1]=speed_final[index];
            }
        }
    }
}

```

```

        if(count==0)speed_section_point[train_priority_index][index]=speed_final_distance[index];
        else{speed_section_point[train_priority_index][index]=temp_envelop[count-1][0];}
        count_template[index]=count+1;

speed_limit_envelop_number[train_priority_index][index]+=count_template[index-1]+speed_limit_envelop_number[train_priority_index][index-1];
        return count;}
    else{
        temp_envelop[count][0]=test_distance;
        temp_envelop[count][1]=test_speed;
        test_distance=-Calculate_pre_position(temp_envelop[count][0],test_speed,train_priority_index);
        test_speed=Calculate_pre_speed(temp_envelop[count][0],test_speed,train_priority_index);
        count++;
    }
}
}
//////////預先計算停站包絡線副程式//////////
int pre_Calculate_station_Envelop(int index,int train_priority_index){
    int count=0;
    float test_speed=speed_at_station;
    float test_distance=station_distance[index];
    while(true){
        if(test_speed>=train_max_speed[train_priority_index]||test_distance<=station_distance[index-1]){
            test_speed=train_max_speed[train_priority_index];
            temp_envelop[count][0]=test_distance;
            temp_envelop[count][1]=test_speed;
            station_stop_point[train_priority_index][index]=test_distance;
            count_template[index]=count+1;
            return count;}
        else{
            temp_envelop[count][0]=test_distance;
            temp_envelop[count][1]=test_speed;
            test_distance=-Calculate_pre_position(temp_envelop[count][0],temp_envelop[count][1],train_priority_index);
            test_speed=Calculate_pre_speed(temp_envelop[count][0],temp_envelop[count][1],train_priority_index);
            count++;
        }
    }
}
//////////列車加速][列車等速][列車減速]求距離 and 速度//////////
float Calculate_pre_speed(float distance,float speed,int train_priority_index){
    return speed-find_deceleration_with_grade(distance,deceleration[train_priority_index])*delta_t;}
float Calculate_pre_position(float distance,float speed,int train_priority_index){
    return speed*delta_t-0.5*find_deceleration_with_grade(distance,deceleration[train_priority_index])*delta_t*delta_t;}
void train_speed_up(int current_train_index){

train_distance[current_train_index]+=train_velocity[current_train_index]*delta_t+0.5*find_acceleration_with_grade(train_distance[current_train_index],train_acceleration[current_train_index])*delta_t*delta_t;

train_velocity[current_train_index]+=find_acceleration_with_grade(train_distance[current_train_index],train_acceleration[current_train_index])*delta_t;
}
void train_speed_down(int current_train_index){

train_distance[current_train_index]+=train_velocity[current_train_index]*delta_t-0.5*find_deceleration_with_grade(train_distance[current_train_index],train_deceleration[current_train_index])*delta_t*delta_t;

train_velocity[current_train_index]+=find_deceleration_with_grade(train_distance[current_train_index],train_deceleration[current_train_index])*delta_t;
}
void train_speed_same(int current_train_index){
    train_distance[current_train_index]+=train_velocity[current_train_index]*delta_t;
}
}
/*****[ S W A P ]*****/
/*當超車行為發生後,前後車[里程][速度][發車時間][列車停站任務][車種][減速度][加速度][下一目標站]*/
/* [下一個速限區][離站再出發時間][列車是否在站內][列車任務終點站][列車是否存在於系統內] 交換 */
/*****
void train_swap(int train1_index,int train2_index){
    int int_temp;
    float float_temp;
    bool bool_temp;

    float_temp=train_distance[train1_index];          //交換前後車[里程]
    train_distance[train1_index]=train_distance[train2_index];
    train_distance[train2_index]=float_temp;

    float_temp=train_velocity[train1_index];          //交換前後車[速度]

```

```

train_velocity[train1_index]=train_velocity[train2_index];
train_velocity[train2_index]=float_temp;

int_temp=train_start_time[train1_index];          //交換前後車[發車時間]
train_start_time[train1_index]=train_start_time[train2_index];
train_start_time[train2_index]=int_temp;

for(int i=0;i<station_no;i++){
    bool_temp=train_task[train1_index][i];        //交換前後車[停站任務]
    train_task[train1_index][i]=train_task[train2_index][i];
    train_task[train2_index][i]=bool_temp;

    int_temp=train_left_station_time[train1_index][i];
    train_left_station_time[train1_index][i]=train_left_station_time[train2_index][i];
    train_left_station_time[train2_index][i]=int_temp;

    int_temp=train_join_station_time[train1_index][i];
    train_join_station_time[train1_index][i]=train_join_station_time[train2_index][i];
    train_join_station_time[train2_index][i]=int_temp;

}

int_temp=train_priority[train1_index];            //交換前後車[車種]
train_priority[train1_index]=train_priority[train2_index];
train_priority[train2_index]=int_temp;

float_temp=train_deceleration[train1_index];      //交換前後車[減速度]
train_deceleration[train1_index]=train_deceleration[train2_index];
train_deceleration[train2_index]=float_temp;

float_temp=train_acceleration[train1_index];      //交換前後車[加速度]
train_acceleration[train1_index]=train_acceleration[train2_index];
train_acceleration[train2_index]=float_temp;

int_temp=train_next_station[train1_index];        //交換前後車[下一目標站]
train_next_station[train1_index]=train_next_station[train2_index];
train_next_station[train2_index]=int_temp;

int_temp=train_next_speed_section[train1_index];  //交換前後車[下一個速限區]
train_next_speed_section[train1_index]=train_next_speed_section[train2_index];
train_next_speed_section[train2_index]=int_temp;

int_temp=train_departure_time[train1_index];      //交換前後車[離站再出發時間]
train_departure_time[train1_index]=train_departure_time[train2_index];
train_departure_time[train2_index]=int_temp;

bool_temp=train_at_station[train1_index];         //交換前後車[列車是否在站內]
train_at_station[train1_index]=train_at_station[train2_index];
train_at_station[train2_index]=bool_temp;

int_temp=train_final_station[train1_index];       //交換前後車[列車任務終點站]
train_final_station[train1_index]=train_final_station[train2_index];
train_final_station[train2_index]=int_temp;

bool_temp=train_in_system[train1_index];          //交換前後車[列車是否存在於系統內]
train_in_system[train1_index]=train_in_system[train2_index];
train_in_system[train2_index]=bool_temp;

int_temp=train_start_permutation[train1_index];   //交換前後車[發車順序]
train_start_permutation[train1_index]=train_start_permutation[train2_index];
train_start_permutation[train2_index]=int_temp;

float_temp=train_total_operation_time[train1_index];
train_total_operation_time[train1_index]=train_total_operation_time[train2_index];
train_total_operation_time[train2_index]=float_temp;

}
//////////傳入目前列車位置,計算列車所在的要停站的停站包絡線的速度限制//////////
float find_station_stop_envelop_speed_from_distance(float distance, int priority){
    int index=0;
    while(true){
        if(distance<station_stop_envelop_final[priority][index][0]){index++;}
        else{
            return station_stop_envelop_final[priority][index][1];}
    }
}

```

```

    }
}
//////////傳入目前列車位置,計算列車所在的不停站的停站包絡線的速度限制//////////
float find_station_not_stop_envelop_speed_from_distance(float distance, int priority){
    int index=0;
    while(true){
        if(distance<station_not_stop_envelop_final[priority][index][0]){index++;}
        else{
            return station_not_stop_envelop_final[priority][index][1];}
    }
}
float find_acceleration_with_grade(float distance,float acceleration){
    int i=0;
    for(i=0;grade_distance[i]<distance;i++);
    return acceleration-0.009*grade[i];
}
float find_deceleration_with_grade(float distance,float deceleration){
    int i=0;
    for(i=0;grade_distance[i]<distance;i++);
    return deceleration+0.009*grade[i];
}

float find_time_of_ETA(int current_train_index){
    float start_simulation_point=train_next_station[current_train_index]-2*train_long
    -station_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]-constant_1,
    end_simulation_point=station_distance[train_next_station[current_train_index]],
    train_position_temp=train_distance[current_train_index],
    train_velocity_temp=train_velocity[current_train_index];
    int simulation_time_temp=0;
    while(train_position_temp<end_simulation_point){simulation_time_temp++;
        //若列車里程>停站包絡線起點里程, 則 列車減速
        if(train_position_temp>station_stop_point[train_priority[current_train_index]][train_next_station[current_train_index]]){
            //////////取得列車在此位置的停站包絡線速度限制//////////
            float speed_temp=find_station_stop_envelop_speed_from_distance(train_position_temp,train_priority[current_train_index]);
            if(train_velocity[current_train_index]>speed_temp){
                train_position_temp+=train_velocity_temp*delta_t-0.5*find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
                train_velocity_temp+=find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
                continue;//-----A_in-----/////
            }
            //若列車速度>列車服務速度, 則減速
            if(train_velocity_temp>train_max_speed[train_priority[current_train_index]]){
                train_position_temp+=train_velocity_temp*delta_t-0.5*find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
                train_velocity_temp+=find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
                continue;//-----A_in-----/////
            }
            //若列車速度=列車服務速度, 則等速
            else if(train_velocity_temp==train_max_speed[train_priority[current_train_index]]){
                train_position_temp+=train_velocity_temp*delta_t;
                continue;}
            else{
                //若列車里程>=速限區起點
                if(train_position_temp>=speed_section_point[train_priority[current_train_index]][train_next_speed_section[current_train_index]]){
                    //若列車速度>列車服務速度, 則減速
                    if(train_velocity_temp>train_max_speed[train_priority[current_train_index]]){
                        train_position_temp+=train_velocity_temp*delta_t-0.5*find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
                        train_velocity_temp+=find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
                        continue;//-----A_in-----/////
                    }
                }
                //若列車里程<速限區起點
                else{
                    train_position_temp+=train_velocity_temp*delta_t+0.5*find_acceleration_with_grade(train_position_temp,train_acceleration[current_train_index])*delta_t;
                    train_velocity_temp+=find_acceleration_with_grade(train_position_temp,train_acceleration[current_train_index])*delta_t;//else 列車加速
                }
            }
        }
    }
}

```



```

        continue;//-----A_in-----/////
    }
}
}
//若列車里程<=停站包絡線起點里程，則做速限區速度判斷
//////////速限區速度判斷
//若列車速度>列車服務速限----->減速
if(train_velocity_temp>train_max_speed[train_priority[current_train_index]]){

train_position_temp+=train_velocity_temp*delta_t-0.5*find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
train_velocity_temp+=find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;
continue;//-----A_in-----/////
}
//若列車速度=列車服務速限----->等速
else if(train_velocity_temp==train_max_speed[train_priority[current_train_index]]){
train_position_temp+=train_velocity_temp*delta_t;
continue;//-----A_in-----/////
}
//若列車速度<列車服務速限----->加速
else{
//若列車所在里程>=速限區起點
if(train_position_temp>=speed_section_point[train_priority[current_train_index]][train_next_speed_section[current_train_index]]){
//////////如果列車速度>最大性能----->減速
if(train_velocity_temp>train_max_speed[train_priority[current_train_index]]){

train_position_temp+=train_velocity_temp*delta_t-0.5*find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;

train_velocity_temp+=find_deceleration_with_grade(train_position_temp,train_deceleration[current_train_index])*delta_t;//////////減速
continue;//-----A_in-----/////
}
//////////如果列車速度<=最大性能----->加速
else{

train_position_temp+=train_velocity_temp*delta_t+0.5*find_acceleration_with_grade(train_position_temp,train_acceleration[current_train_index])*delta_t;

train_velocity_temp+=find_acceleration_with_grade(train_position_temp,train_acceleration[current_train_index])*delta_t;//else 列車加速
continue;//-----A_in-----/////
}
//若列車所在里程<速限區起點----->加速
else{

train_position_temp+=train_velocity_temp*delta_t+0.5*find_acceleration_with_grade(train_position_temp,train_acceleration[current_train_index])*delta_t;

train_velocity_temp+=find_acceleration_with_grade(train_position_temp,train_acceleration[current_train_index])*delta_t;//else
列車加速
continue;//-----A_in-----/////
}
}
}
return simulation_time_temp;
}
int find_train_index_from_start_permutation(int start_index){
for(int result=0;result<train_no;result++){if(train_start_permutation[result]==start_index){return result;}}
}
void print_capacity(int station){
for(int station_index=1;station_index<station_no;station_index++){
for(int train_index=0;train_index<train_no-1;train_index++){
int front_train_index=find_train_index_from_start_permutation(
train_left_station_permutation[station_index][train_index]
),
next_train_index=find_train_index_from_start_permutation(
train_left_station_permutation[station_index][train_index+1]
);
int time_temp=0;
if(front_train_index!=0&&next_train_index!=0){
time_temp=train_left_station_time[next_train_index][station_index]
-train_left_station_time[front_train_index][station_index];
}
if(time_temp>0
&&(

```

```

        station_index==station
    )
    ){
        capacity_temp[train_priority[front_train_index]][train_priority[next_train_index]]
        =capacity_temp[train_priority[front_train_index]][train_priority[next_train_index]]<time_temp?time_temp
        :capacity_temp[train_priority[front_train_index]][train_priority[next_train_index]];
        capacity_count[train_priority[front_train_index]][train_priority[next_train_index]]++;
    }
}
}
//////////計算結果容量

for(int front_priority=0;front_priority<4;front_priority++){
    for(int priority=0;priority<4;priority++){
        if(capacity_count[front_priority][priority]!=0){
            capacity_result[front_priority][priority]
            +=capacity_temp[front_priority][priority]*capacity_count[front_priority][priority];
        }else{ capacity_result[front_priority][priority]=0; }
    }
}
int temp_count_summary=0,temp_capacity_summary=0;
for(int i=0;i<4;i++){
    for(int j=0;j<4;j++){
        temp_count_summary+=capacity_count[i][j];
        temp_capacity_summary+=capacity_temp[i][j]*capacity_count[i][j];
    }
}
//////////列印容積計算結果
cout<<"Mean Headway MAX:"<<endl;
for(int front_priority=0;front_priority<4;front_priority++){
    for(int priority=0;priority<4;priority++){
        cout<<" "<<front_priority<<" "<<priority<<"=";
        if(capacity_count[front_priority][priority]!=0){
            cout<<setw(7)<<capacity_temp[front_priority][priority];
        }
        else{cout<<setw(7)<<"None";}
        cout<<setw(6)<<" ";
    }cout<<endl;
}
cout<<"Mean Headway Count:"<<endl;
for(int front_priority=0;front_priority<4;front_priority++){
    for(int priority=0;priority<4;priority++){
        cout<<" "<<front_priority<<" "<<priority<<"=";
        if(capacity_count[front_priority][priority]!=0){
            cout<<setw(7)<<capacity_count[front_priority][priority];
        }
        else{cout<<setw(7)<<"None";}
        cout<<setw(6)<<" ";
    }cout<<endl;
}
cout<<"Mean Headway : "<<(temp_capacity_summary/temp_count_summary)*(1+uuuu)<<endl;
cout<<"Capacity : "<<(86400/((temp_capacity_summary/(temp_count_summary)*(1+uuuu)/4))*eeee<<endl;
}
/*****副程式內容結束*****/
/*=====
          讀入 列車等級
=====*/
int read_train_priority(int train_priority[train_no])
{
    FILE *p_str;
    p_str = fopen("train_priority.txt","r");
    if(p_str == NULL)
    {
        printf("\n read_train_priority function error!!! ");
        return 0;
    }

    int i;
    for(i = 0; i < train_no; i++)
    {
        fscanf(p_str,"%d",&train_priority[i]);
    }
    fclose(p_str);
}
}

```

```

/*=====
                        讀入 發車時間
=====*/
int read_train_start_time(int train_start_time[train_no])
{
    FILE *p_str;
    p_str = fopen("gogo_time.txt", "r");
    if(p_str == NULL)
    {
        printf("\n read_train_start_time function error!!! ");
        return 0;
    }
    int i;
    for(i = 0; i < train_no; i++)
    {
        fscanf(p_str, "%d", &train_start_time[i]);
    }
    fclose(p_str);
}
/*=====
                        讀入 列車任務
=====*/
int read_train_task(bool train_task[train_no][station_no])
{
    FILE *p_str;
    p_str = fopen("train_task.txt", "r");
    if(p_str == NULL)
    {
        printf("\n read_train_task function error!!! ");
        return 0;
    }
    int i, j;
    for(i = 0; i < train_no; i++)
    {
        for(j = 0 ; j < station_no ; j++ )
        {
            fscanf(p_str, "%d", &train_task[i][j]);
        }
    }
    fclose(p_str);
}

```

