Developing an Integrated Rail Simulator for SimMobility

by

Kenneth Hang Chang Koh

B.Sc Mechanical Engineering

University of Illinois at Urbana-Champaign, 2015

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements of the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Signature redacted

Author.....

Department of Civil and Environmental Engineering

June 19, 2017

Signature redacted

Certified by.....

Moshe E. Ben-Akiva

Professor of Civil and Environmental Engineering

Thesis Supervisor

Signature redacted

Certified by.....

Carlos Lima Azevedo

Research Scientist

Thesis Co-Supervisor

Signature redacted

Accepted by.....

Jesse Kroll

Professor of Civil and Environmental Engineering

Chair, Graduate Program Committee

Developing an Integrated Rail Simulator for SimMobility

by

Kenneth Hang Chang Koh

Submitted to the Department of Civil and Environmental Engineering on June 19, 2017 in Partial Fulfillment of the Requirements of the Degree of Master of Science in Transportation

Abstract

High-frequency rail transit systems such as subways play a key role in urban transportation; as such, the choices made in the design, implementation, and operation of rail networks are important. To fully understand the impacts of these choices, there is a need for an integrated analysis, where the rail system is analyzed as a component of a larger transportation network. This thesis focuses on designing a rail simulator as an integrated component of the SimMobility simulation platform, improving upon existing simulators via complete integration with the comprehensive features of SimMobility, enabling it to account for multimodal demand by dynamically adapting to changes in the entire network. A second aim is flexibility; through the implementation of a Service Controller able to read user-written LUA files, it is shown that the rail simulator is able to model a wide variety of different scenarios and policies.

The rail simulator is then calibrated using a novel sequential calibration method, where individual parameters are estimated sequentially. The advantage of this method is its computational speed and ability to operate with only automatic fare card (AFC) data, not requiring automatic train control (ATC) data. This approach is shown to result in a 30% error in travel times, making it an ideal first-step approach to generate seed values that can be used for more comprehensive calibration methods.

Finally, the rail simulator is used to conduct a historical disruption scenario and investigate the effects of a simple mitigating strategy. The effects of the various scenarios on not only the rail network, but also the road network and passenger welfare, are captured, thus demonstrating the usefulness of an integrated simulator in research and future planning.

Thesis Supervisor: Moshe E. Ben-Akiva

Title: Professor of Civil and Environmental Engineering

Acknowledgements

I would firstly like to thank my advisor, Prof. Moshe Ben-Akiva, for the assistance and guidance he has given me over the course of my studies at MIT. The rigorous, yet fair academic standards he imposes on his students have been an inspiration to me; furthermore, he has also been a pillar of support both academically and psychologically. I am honored to have had the opportunity to work with him.

I would also like to thank my thesis co-supervisor, Dr. Carlos Azevedo, for the tremendous help he has provided over these past two years. Always approachable, his uncanny ability to identify potential problems in my research, and suggest solutions to them before they even occur, has improved the quality of my research and this thesis immeasurably.

This research would not have been possible without the contributions of the researchers at SMART, who transformed the models and concepts described in this thesis into a working simulator. I am especially grateful to Kakali Basak, Zhang Huai Peng, and Jabir Karachiwala for the amazing work they have done in coding, fixing bugs, and generally making sure that everything functioned properly.

I am also indebted to the Land Transport Authority of Singapore for providing me with the financial and administrative support necessary to pursue my graduate studies at MIT. I would like to especially thank Jeremy Wong, who has efficiently helped me with various administrative matters over the course of the past four years.

To my friends in MIT, other parts of the US, or back in Singapore, thank you for the emotional and social support. Studying can be an exhausting affair, and I appreciate all the laughter, companionship, and conversations that distracted me from the stress of research and made this time more bearable. To Augustine, Auriga, Clement, Jon, and everyone else (you know who you are), thank you for helping me keep my sanity, especially (unfortunately?) during final exam week every semester.

Last, but definitely not least, I would like to thank my family: my sister, for being a constantly humorous, yet encouraging, presence in my life, and my parents, for their continuous, unwavering encouragement and support. I would not be here today, on the cusp of graduating from MIT, without their love, nor the sacrifices they have made over the past 25 years of my life. This thesis is dedicated to them.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

1.1 Motivation

High-frequency rail transit systems play a key role in urban transportation, transporting large volumes of commuters from origin to destination daily and significantly alleviating stress on the road network. Indeed, more than 50% of residents in major population-dense cities such as Hong Kong, Singapore, Tokyo, and London, rely on public transport for their daily commute (Di, 2013)[7]. As such, the choices made in the design, implementation, and operation of rail networks are of considerable importance. To understand the impacts of these choices, there is a need to perform an *integrated* analysis—considering not only effects on the rail network, but the resulting effects on future demand and the transportation network as a whole—when designing and assessing potential rail systems, or proposed modifications to existing systems. Similarly, there is a need to examine both normal operating conditions, and system disruptions, with the latter typically being harder to study given its short analysis period and the detail required to analyze system performance and traveler reactions.

A natural approach to this analysis is through simulations, which allow for the examination of a multitude of scenarios that would be too expensive or impractical to conduct in the real world. In an ideal world, we would be able to capture every relevant variable and their interactions, and yield a perfect picture of the real-world effects of theoretical disruptions; of course, this is unfortunately beyond the capabilities of anything humanly possible today. A more realistic option is to design a simulator detailed enough to represent important interactions, yet streamlined enough to be computationally efficient, and flexible enough to simulate a range of varying scenarios.

More importantly, it is necessary to design an *integrated* rail simulator that captures not only demand and supply interactions within the rail network, but also how these interactions affect other modes of transport, and how a commuter's experience in one day affects his decisions in the next. After all, this interplay of variables is a basic tenet of real-world transportation decisions—for example, if a commuter

consistently experiences train delays, he might lose faith in the system and simply start taking the bus—and to ignore it in the simulator would result in an incomplete model and limited understanding of cause-and-effect.

Such an integrated rail simulator should, via interfacing with other simulators and an overarching framework, be able to account for multimodal demand. Passenger demand should affect system performance; similarly, changes in system performance should feedback and affect future passenger demand, resulting in a clear picture of overall transportation choices. It should also be extremely flexible, able to mimic the real world by simulating a large array of scenarios.

However, the undeniable usefulness of advanced, integrated simulators gives rise to a secondary problem; that of *calibration*. The models used in these simulators are, by virtue of their complexity, typically data-eager, and require tremendous amounts of data and computational power to produce accurate and reliable results.

Calibration of supply has typically been done by simply utilizing historical data, or manually collecting data at certain stations, then extrapolating and generalizing this data to the entire train network. However, the former is insufficient for an *integrated* simulator, which demands information for which historical data may not exist (such as walking times within stations) or may be inappropriate (such as dwell times, which should vary dynamically with demand and not simply be a historical average). The latter, on the other hand, tends to be extremely labor intensive and requires the use of significant assumptions (for example, that the properties of one station represent the properties of *all* stations on a line). These methods therefore lead to inaccuracies and are counterproductive to the goal of developing an advanced simulator.

A potential solution to this lies in the use of electronically recorded data. In recent years, due to rapid advances in communications technology and the increasing ubiquity of electronic devices, automatic

fare collection (AFC) via the use of smart cards (or even phones in some cities) is being adopted by an increasing number of transit agencies worldwide. Such a method offers many advantages: not only is it a convenient, straightforward, and secure method of paying fares, it also allows the operator much more flexibility in implementing specialized fare schemes. Along with this proliferation of electronic fare collection comes a wealth of data: a detailed log of journeys made by commuters over time. Analysis of this data may yield insights into a wide array of transportation issues such as commuter behavior, response to new policies and so on. However, there is a significant drawback: commuters' movements while they are *within* the system are not tracked (that is, most systems only record at point-of-entry and, occasionally, point-of-exit). Hence, this data has often been used with train movement data (known as automatic train control, or ATC) to match passengers to trains and therefore calibrate supply. This, unfortunately, poses a problem in cases where ATC data is unavailable—which, due to privacy concerns, is becoming more prevalent.

This thesis proposes a solution to this problem via the method of *sequential calibration* to calibrate the supply of trains in a rail simulator. In this method, AFC data is used together with General Transit Feed Specification (GTFS) data and minimal manual data collection to calibrate many aspects of train supply. The goal is to utilize certain assumptions to bypass the need for ATC data, thus yielding reasonably accurate estimations of key parameters quickly.

This thesis is divided into five main parts. Chapter 1 provides an overview of the existing literature, and defines terms that will be used in this thesis. Chapter 2 focuses on the development of a rail simulator as part of the state-of-the-art SimMobility simulation platform. It presents the overall structure and design of the simulator, and demonstrates how the unique addition of the Service Controller allows the simulator to be extremely flexible. Chapter 3 focuses on the calibration of the rail simulator, and presents the method of sequential calibration as an alternative to calibrate train supply models where data is sparse. Chapter 4 focuses first on the validation of the parameters estimated in Chapter 3, and

subsequently on the validation of the rail simulator developed in Chapter 2 by comparing simulated disruptions with historical disruptions on the Singapore rail network. A simple disruption management solution is then proposed and implemented to demonstrate the capabilities of the rail simulator.

1.2 Types of Simulation Models

In the interests of clarity, it is necessary at this point to briefly digress and describe the various types of commonly used simulation models, as these terms will be used to evaluate existing simulators in the next section. There are three main types of models, *microscopic*, *macroscopic*, and *mesoscopic*, and two main types of simulators, *time-based* and *event-based*. The distinction between these choices lie in the complexity and computational resources required, and each choice can be thought of as representing a certain level of detail.

1.2.1 Microscopic

In a microscopic model, each individual entity in the simulation is explicitly modelled. In the context of a train simulator, each train and passenger is modelled explicitly, and system properties such as dwell time are determined through passenger/train behavior models that replicate the decisions made by individual passengers when boarding or alighting. Microscopic models tend to provide the most accurate simulations, but require significant computational power.

1.2.2 Macroscopic

In a macroscopic model, entities are aggregated together and not modelled individually. For example, a train simulator could simply track the number of passengers in the system, instead of modelling each passenger individually, and determine system properties such as dwell time via a function of the number of boarding passengers, alighting passengers, and overall crowdedness of the train. Macroscopic models are often orders of magnitude quicker than microscopic models, and require less information about the system, but sacrifice accuracy.

1.2.3 Mesoscopic

In a mesoscopic model, entities are implemented using both aggregated and disaggregated models. For example, the train simulator in this thesis models passengers' walking times within a station individually, but determines dwell time as a function of passenger volume to reduce computational costs. Mesoscopic models can be thought of as a "middle ground" between microscopic and mesoscopic models, with key areas simulated microscopically and the rest simulated macroscopically.

1.2.4 Time- vs. Event-Based Simulation

In time-based simulation models, the simulation is divided into evenly-spaced intervals. At each interval, the current state of the system is evaluated and the behavior of entities within the system is calculated for the *next* interval. As such, the system evolves continuously over time, and is conceptually similar to what occurs in reality. However, such a model usually requires significant computational power to generate the information required at every time-step.

In contrast, in event-based simulation models, the simulation progresses via the occurrence of a sequence of pre-defined events. When an event occurs (for example, a passenger arrives at the station), all variables relevant to that event are updated and the time of the next event calculated. As such, the simulation progresses at irregular intervals, and the updating of variables occurs asynchronously, resulting in information on the state of the system *between* events often being hazy. However, this approach substantially reduces computational effort and yields quick results.

Ultimately, the decision of what type of model to develop is primarily a decision on the trade-offs between accuracy and computational speed. In the *integrated, multimodal* rail simulator that this thesis aims to develop, time-based simulation is crucial. Given that events that occur in one simulator may affect events in another simulator, an event-based model is an inelegant approach; furthermore, given the constant feedback between supply and demand, a time-based simulation allows passengers to

instantly react to the *current* state of the system, and is ideal for capturing the performance of a transportation network in reality.

1.3 Literature Review

There is a reasonable amount of prior work involving the development of train simulators to investigate various problems such as train scheduling, supply, disruption management, and so on. (Goodman *et al.*, 1998)[8] provide a helpful framework for categorizing different types of train simulators, and their relative advantages and disadvantages. Their work has been summarized in the preceding section.

Simple macroscopic simulators such as Bahn[33] and MetroModSim[34] have been developed by independent parties to provide primarily *visual* representations of rail networks; in these simulators, trains move at deterministic speeds, with only rudimentary signal controls such as linked pairs of Stop/Go signals (when one signal says Go, the other signal says Stop, and vice versa). On the other hand, commercial simulators such as OpenTrack (Nash *et al.*, 2004)[21], a microscopic time-based railway simulator, provide a comprehensive model of train movement across long distances. However, they tend to not focus on demand-side effects such as dwell time or route choice, and are commonly used in long-distance freight transportation planning instead.

AIMSUN and VISSIM are popular commercial simulation programs used in traffic modelling (Hidas, 2005)[10] that place an emphasis on *integration*, where large-scale mesoscopic simulators interact with more detailed time-based microscopic models to simulate traffic flow and, in the case of the former, even generate demand based on geographical and socioeconomic data. However, these simulators exist primarily to model *traffic*, with agents in the simulator being *vehicles*, not *passengers*; as such, there is a heavy emphasis on multimodal interactions on the road, but a lack of breadthwise integration with other forms of transit, such as dedicated-track rail.

In academia, rail simulators are often developed to investigate various supply-side policies. For example, (Grube *et al.*, 2011)[9] and (Sánchez-Martínez, 2012)[26] both created discrete event-based simulators to increase the operational efficiency of high-frequency rail transit, such as intra-city subways, by implementing various resource allocation strategies. (Cha *et al.*, 2014)[4] constructed a similar model for Maglev trains to investigate how varying passenger arrival rates affect system performance. Others utilize rail simulation in operations research, such as to optimize bus-based disruption recovery strategies in subways in Australia (Darmanin *et al.*, 2010)[6], Singapore (Jin *et al.*, 2013)[14], and so on. However, these models are often customized to the topic at hand and hence over-simplified, only serving as a tool to demonstrate the proposed operational strategy. For example, (Grube *et al.*, 2011) assume constant speeds and ignore operational restrictions (such as safe distances) between events, while (Ravichandran, 2013)[25] and (Sánchez-Martínez, 2012) utilize historical running time data instead of actively simulating train movement. (Cha *et al.*, 2014) utilize a linear dwell time based off historical data, and ignore the effects of passenger congestion. (Darmanin *et al.*, 2010) and (Jin *et al.*, 2013) only consider deterministic choice (that is, when the subway breaks down, passengers automatically take the substitute bus) and ignore the possibility of individual agents revising their mode or destination choice due to this disruption.

Of course, more complex academic simulators do exist. MATSim is an integrated multimodal simulator that generates demand from a synthetic population (Horni *et al.*, 2016)[13]; however, it also focuses primarily on road transit, with the intricacies of demand-side effects (such as dwell time) notably absent from its embedded rail simulator. Similarly, SUMO[35] is a microscopic time-based multimodal simulator that supports passenger mode choice; however, it lacks dynamic feedback (once a commuter makes a choice, he will not change his mind) and has limited rail simulation capabilities.

SimMobility (Lu *et al.*, 2015)[19] is an integrated simulation platform that uses three simulators (Long-Term, Mid-Term, and Short-Term) to generate an accurate picture of demand and supply. The Long-

Term simulator governs decisions made over months or years (such as house/job location choice), the Mid-Term simulator governs decisions made over hours or minutes (such as route or mode choice), and the Short-Term simulator governs decisions made over seconds (such as path choice). A unique feature of the SimMobility Mid-Term is its *publish/subscribe* mechanism, where agents modify their travel plans *on-the-fly* based on current conditions—for example, a commuter that originally planned to take the train may choose to drive if the trains are exceptionally crowded. This mechanism is exceptionally useful in simulating disruption scenarios, where commuters often react to imperfect, constantly changing information. However, the SimMobility Mid-Term lacks a dedicated rail simulator—passengers who choose to take the subway are simply moved to their desired destination after a delay corresponding to distance moved. As such, it is unable to capture the nuances of rail transportation, and the effects of changes to the rail system on the overall transportation network.

A common theme of the above simulators is that, within rail transportation, passengers tend to be modelled macroscopically and feedback—reactions to the current state of the system—does not exist. There is therefore a need to develop a simulator where passengers are modelled microscopically, to better capture individual choices and responses to system conditions, as well as impacts that events occurring in the rail system have on the overall transportation network. A second consideration is developing a simulator flexible enough to simulate a wide range of different scenarios, thus reducing the need to custom-develop simulators in the future and eliminating duplication of effort.

Within a train simulator, there are various approaches to modelling aspects of train behavior. When modelling train movement, it is necessary to decide *how* trains should move between stations, as well as what type of signaling system, if any, should be in place. If train movement is not crucial to the purpose of the simulator, a typical approach as found in Bahn and the simulator developed by (Grube *et al.*, 2011) is to simply assume a constant movement speed. (Kraft, 2013)[16] demonstrates that an improvement to this approach, where trains move in regions of constant *acceleration*, constant *velocity*,

and constant *deceleration*, results in a fairly good approximation to train trajectories in real life. If more detail is required, commercial simulators such as OpenTrack allow for variations in speeds and fuel consumption depending on the tractive power of the engine, track gradient, and track curvature. However, this comes at the expense of computational cost.

(Chang *et al.*, 1998)[5] provide a good summary of the differences between the two main types (*fixed-block* and *moving-block*) of signaling systems. The former, being an older variant, is more widespread, but is steadily being replaced by the latter in rail networks worldwide, which allows for trains to operate with significantly reduced headway. (Hill *et al.*, 1992)[11] provide a framework for the implementation of these systems in a rail simulator.

Similarly, there are various approaches to modelling dwell time, ranging in complexity. Bahn adopts the simple approach of randomly and uniformly drawing from a pool of prospective dwell times. However, this approach lacks accuracy and is unlikely to mirror real-world conditions. An improved approach is to draw dwell times from a calibrated distribution, based off historical data. This approach is used in various simulators such as the one developed by (Sánchez-Martínez, 2012), however, it by default assumes a largely stable demand and is inappropriate to simulate disruptions, where passenger numbers often swell dramatically.

The Maglev simulator in developed by (Cha *et al.*, 2014) utilizes historical data to calculate the mean boarding and alighting time per passenger, which is then multiplied by the total number of alighting and boarding passengers to obtain the dwell time. However, this model relies on the simplifying assumption that dwell times scale linearly with passengers, and ignores congestion-related effects.

(Lin *et al.*, 1992)[18] proposed a function for determining dwell time, which was expanded on by (Puong, 2000)[24] and shown to be a good fit for the MBTA Red Line. This function depends on train occupancy

and the number of boarding and alighting passengers, and results in the highest degree of accuracy in dwell times across fluctuating demand. This function was adapted for the rail simulator in this thesis.

In many rail simulators such as those developed by (Cha *et al.*, 2014), (Sánchez-Martínez, 2012), (Grube *et al.*, 2011), amongst others, walking speeds within stations are often ignored, with passengers assumed to arrive directly at the platform instead of arriving at a fare-gate and then walking within the station to the platform. However, an integrated simulator cannot make this assumption, as the objective is to consider passenger trips across all modes as a whole, not just the rail component of these trips. Previous work done regarding passenger walking speeds include (Ottomanelli *et al.*, 2012)[22] and (Zhang *et al.*, 2009)[30], who suggest that asymmetric walking speeds are best modelled by a lognormal distribution. (Zhu, 2014)[32] applied this work to train stations and demonstrated that passenger walking speeds within stations followed lognormal distributions as well. As such, this thesis utilizes this distribution to estimate passenger walking times within the system.

In order to calibrate the various parameters of the simulation, electronic smart card data was used. (Pelletier *et al.*, 2011)[23] offer a comprehensive review of the usage of smart card data in various transportation studies. Three categories of studies have been identified: the *strategic* level, which focuses on long-term planning, the *tactical* level, which focuses on daily problems such as scheduling, and the *operational* level, which focuses on the examination of performance indicators such as schedule adherence. However, at all three levels, the supply is generally already assumed to be calibrated. Smart card data is instead typically used to generate demand-side data which then dictate changes in supply policy. For example, (Zhao *et al.*, 2007)[31] present a methodology of using smart card data to generate O-D matrices for rail passengers.

On the other hand, the calibration of supply is usually done via manual data collection. For example, (Lin *et al.*, 1992)[18] and (Puong, 2000)[24] calibrated the dwell time for Boston's MBTA Red Line by manually

collecting data at two stations during the AM Peak, then generalizing their results across the entire Red Line. While collecting dwell time data at *all* stations would be labor-intensive to the point of infeasibility, there are nonetheless inaccuracies associated with generating a dwell time function based on observations from only two stations.

Another common approach to calibrating supply is by simply using recorded historical data (for example, the average dwell time at a station at a certain time of day). This has been done in many simulators such as the one developed by (Sánchez-Martínez, 2012), and (Motraghi *et al.*, 2012)[20], to name a few, and is a reasonable approach to planning supply-side policy. However, this data may not be readily available and must be collected in some form. Historical data is also unable to predict how a system responds to unexpected changes in demand. Finally, ignoring passenger movements within the stations themselves leads to a lack of accuracy, especially in the case where a passenger uses multiple train lines to complete a trip. It is important to determine how much of this journey time was spent out of transit: walking to a new platform in a transfer station, waiting for new trains, and so on.

In recent years, attempts have been made to utilize electronic data to expedite the calibration process. (Sun *et al.*, 2015)[29] propose an innovative method of combining automatic fare card (AFC) with automatic train control (ATC) data to match passengers to trains, and therefore efficiently estimate the parameters in the system simultaneously. Although this is an efficient method of supply calibration, it requires two sets of data (AFC and ATC), the latter of which may not always be available due to security concerns.

(Sun *et al.*, 2012)[28] utilized AFC data exclusively to generate an average walking time, dwell time, and train movement speed for entities in Singapore's rail network. However, in their analysis they assumed that these three values were constant across all stations. In the interests of accuracy, we intend to

expand on this approach by relaxing this assumption and generating functions for each distribution at *every* station in the network.

Another similar approach to calibration was utilized by (Antoniou *et al.*, 2015)[2], which utilized a weighted simultaneous perturbation stochastic approximation algorithm to simultaneously calibrate all supply and demand parameters for traffic simulation models. Although such an approach is more comprehensive and robust, it can be time-consuming and computationally taxing. In this thesis, we propose that a *sequential* calibration framework, which requires significantly less computational power, is an effective approach to calibrating rail systems.

Finally, when evaluating the performance of a system under disruption, we consider two main performance metrics, *capacity reliability* and *travel time reliability*, as suggested in (Carrion *et al.*, 2012)[3]. The former refers to the probability that the system accommodates a certain level of demand, while the latter refers to the probability that the system allows commuters to reach their destination within a given timeframe. By evaluating these metrics, we will demonstrate that the rail simulator in this thesis has the capability to evaluate the negative effects of disruptions and the usefulness of proposed mitigation strategies.

1.4 Research Approach

To summarize the discussion in the previous sections, the goal of this thesis is to develop a comprehensive rail simulator within a multimodal agent-based simulation framework for the analysis of different operational designs. In order to provide a clear picture of supply and demand, generate a feedback loop where commuters react to the current state of the system and vice versa, and enable integration in the form of constant passenger flow between the rail simulator and other transit simulators, the simulator should be *time-based* and *mesoscopic*. Although commuters and trains should

be individually simulated for ease of integration, their effects on each other (such boarding/alighting behavior or walking speeds) can be simulated macroscopically.

1.4.1 Why Choose SimMobility?

Given the aforementioned aims, the SimMobility Mid-Term is an ideal overarching simulation platform to integrate the proposed rail simulator with. The Mid-Term already contains many key functions, such as organic demand (obtained from the Long-Term), mode/route choice models and a functioning traffic simulator, which allow for breadthwise integration of the rail simulator to other forms of transit, and the *publish/subscribe* mechanism, allowing for the development of a feedback loop between the rail system and passengers' decisions. Finally, the "blank slate" of the Mid-Term enables the rail simulator to be developed in a similarly flexible manner, allows it to easily simulate a wide range of scenarios (not just disruptions) under a wide array of operating conditions.

1.4.2 Why Sequential Calibration?

After the rail simulator is developed, it should be *tested*, and this thesis aims to apply it to a case study of a disruption in Singapore. However, due to the difficulties in obtaining ATC data, a few issues arise; firstly, the question of whether the simulator can be efficiently calibrated through AFC data *alone*, and secondly, the increasing importance of developing an efficient calibration methodology that can be applied to a variety of rail systems in different cities. As such, there are two tenets to the calibration philosophy:

1) The calibration process should rely solely on AFC data as much as possible, as obtaining additional data may be difficult depending on the transit agency in question;
2) The calibration process should minimize the need for manual data collection, in the interests of expediency and as an improvement over current methods.

The sequential calibration process satisfies both these tenets, relying almost exclusively on AFC data to generate reasonably accurate estimations of simulation parameters, and as such is an appropriate choice to calibrate the rail simulator.

1.4.3 Thesis Outline

In Chapter 2, we discuss in depth the design of the rail simulator, using the Singapore subway system as an example. We elaborate on train movement, dwell time, end-of-line behavior, and introduce the Service Controller and explain its function in making the rail simulator extremely flexible.

In Chapter 3, we discuss the calibration of the rail simulator, once again using the Singapore network as an example. We present the framework of sequential calibration, and demonstrate how iterative simulations can be used to calibrate many variables using just AFC data.

In Chapter 4, we present a validation of the methods used in Chapter 3, where the estimated parameters were used to estimate travel times for a fresh day. We demonstrate that the sequential calibration approach is valid, and subsequently apply the simulator a to case study, simulating a real-life subway disruption that occurred in Singapore in 2013. We show that the simulator comprehensively simulates the effects of a rail service disruption and can be used to investigate in detail the effects of different mitigation policies.

Finally, in Chapter 5, we summarize our findings and offer suggestions for future research. We discuss improvements to the calibration process and disruption simulation, and present suggestions for adapting the simulator to rail networks with different properties in the future.

CHAPTER 2: DEVELOPING THE RAIL SIMULATOR

2.1 Overview

The SimMobility Mid-Term simulator is a time-based mesoscopic simulator integrated both upward (with the Long-Term, which provides population and land use data and the resulting demand) and downward (with the Short-Term, which models minutiae such as path choice). The Mid-Term simulator is itself composed of multiple modules, which work in coherence to provide a clear picture of individuals' trip, route, and mode choices throughout the course of a day. The interaction between these modules is summarized in (Lu *et al.*, 2015) and reproduced in Figure 2.1 below:

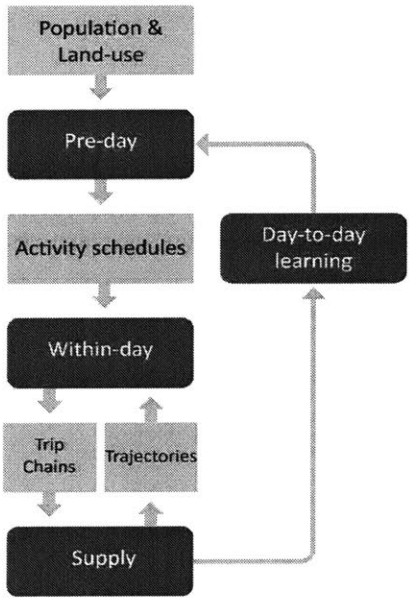

Figure 2.1: Structure of the SimMobility Mid-Term Simulator

A synthetic population is generated in the Long-Term simulator and passed to the Mid-Term simulator. In the *pre-day* module, individuals plan their trips and choose routes and modes, based on prior experiences and current needs. These schedules are then passed to the *within-day* module, which works in tandem with the *supply*—comprising the road simulator and rail simulator—to generate travel times for each trip. At the same time, the supply simulators constantly provide feedback about their current

performance, which may cause commuters to modify their schedules accordingly. For example, if a commuter becomes aware that the rail network along his desired route is congested, he may instead decide to take a taxi to his destination. As such, there is a constant dynamic interaction between the *within-day* and *supply* modules. Finally, as the day concludes, individuals "learn" from their experiences and modify their choices in the subsequent *pre-day* module accordingly. For more information, please refer to (Lu *et al.*, 2015) and (Adnan *et al.*, 2016)[1].

In order to interact with the dynamic demand from the Mid-Term simulator, the rail simulator was developed as a *mesoscopic* simulator, with demand and supply (that is, passenger-Agents and train-Agents) modelled individually, but performance effects (that is, the effects of passengers on dwell time) modelled at an aggregate level. The rail simulator itself is comprised of Agents and Entities, which are both modelled microscopically, with individual Agents/Entities having unique associated parameters. The key distinction between the two terms is that Agents are vested with decision-making abilities, while Entities make no decisions whatsoever. These are both regulated by an overarching Service Controller, which constantly evaluates the current state of the system and dynamically defines certain targets for Agents, such as speed limits or dwell times for train-Agents, or walking times for passenger-Agents. Finally, the user manipulates this Service Controller through the use of APIs to simulate various conditions and scenarios.

2.2 Framework and Components

In this section, we will provide a general overview of the infrastructure models and frameworks for passenger and train movement used in the rail simulator. The subsequent sections in this chapter will further elaborate on the models used in these frameworks, namely *train pools* (depot modelling), *train movement models* (speed and acceleration modelling), *passenger-train interaction models* (dwell time and station behavior modelling), and the *Service Controller* (external scenario modelling).

Before proceeding, it is necessary to define key terms in the proposed rail simulator. A **pool** is analogous to an "uninvolved area" (a depot, sidings, etc.), and represents an area where out-of-service trains wait (this will be further elaborated in Section 2.2). A **platform** corresponds to a specific train service in a specific direction, and is irrelevant to the number of *physical* platforms a station has. As such, most stations contain two platforms, with interchange stations containing more. A **station** represents a point of connection between the overall Mid-Term simulator to the rail simulator, where commuters are "passed" from the former to the latter. Each station has a unique name as an identifier, shared across all lines the station contains. A **block** represents a specific section of track with unique properties, and connects platforms to each other. **A line** represents a *sequence* of platforms and blocks that trains progress through sequentially during service. As such, train services are effectively represented as two different lines in the rail simulator. Finally, every train is assigned a **schedule,** which represents an ordered subset of platforms within a line that a train should serve. For example, in Singapore, the North-East Line (NEL) consists of 16 stations, NE1 through NE17 (less NE2), and is represented in the rail simulator as two lines: NE_1 consists of the ordered set of platforms {NE1_1, NE3_1, ..., NE17_1} and NE_2 consists of the ordered set {NE17_2, NE16_2, ..., NE1_2}. An express train on line NE_1 could conceivably have the schedule {NE1_1, NE3_1, NE6_1, NE12_1, NE17_1}, which only consists of interchange stations. Figure 2.2 shows a graphical depiction of the first 2 stations on the NEL:

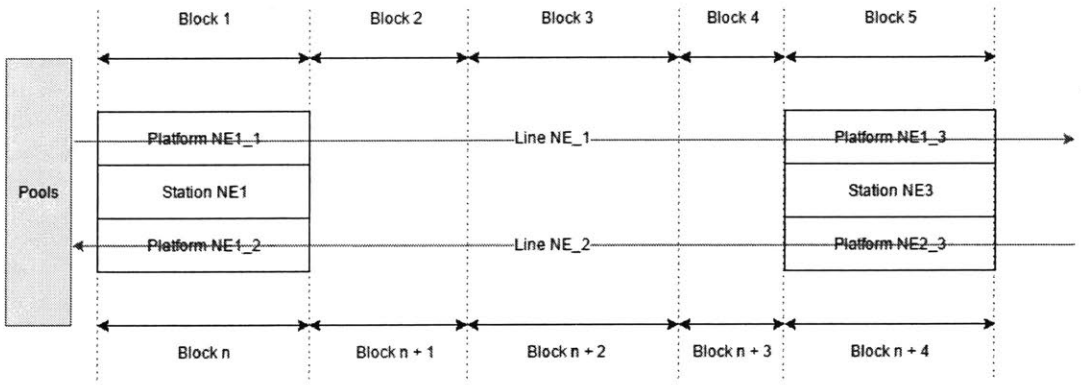

Figure 2.2: Entities in the North-East Line

The rail simulator itself consists of components which are either *Entities*, or *Agents*. The former make no decisions, but have associated attributes that affect decisions made by the latter. Pools, blocks, stations, and platforms are Entities; for example, a station-Entity has *walking time parameters* that passenger-Agents use to determine their walking time within the station. Block-Entities have *acceleration rates* and *maximum speeds* associated with them that affect how fast train-Agents decide to move on that particular block-Entity; similarly, platform-Entities have *minimum dwell times* and *maximum dwell times* that affect how long trains decide to dwell at the platform-Entity. On the other hand, passengers and trains are Agents; they constantly make decisions based on their own associated attributes, relevant Entity attributes, and instructions from the Service Controller.

More specifically, the Service Controller regulates Agent behavior via the use of internal functions that ensure smooth operation of the system. For example, the Service Controller may impose a second speed limit on a train, in addition to the speed limit already defined by the block-Entity the train is on, to keep two trains a safe distance from each other. Agents are individualistic, only focusing on their current situation; the Service Controller rectifies this by focusing on the system in its entirety.

Finally, the *user* has ultimate control over the rail simulator. Through the use of application programming interfaces (APIs), the user is able to interact with the Service Controller to create scenarios worthy of investigation. For example, the user may specify unconventional operating logic, or introduce atypical train behavior such as break-downs, to explore the sensitivity of the system to deviations from standard operation, the robustness of proposed policies, and so on.

Table 2.1 summarizes the various components of the proposed rail simulator and their associated attributes, while Table 2.2 summarizes the decisions made by the Agents and Service Controller.

Name	Type	Associated Attributes
Pool	Entity	• Pool ID • Maximum Capacity • Train Occupancy List (list of Train IDs currently in the pool) • Type (Active/Inactive) • Dispatch Schedule
Block	Entity	• Block ID • Maximum Acceleration/Deceleration Rate • Maximum Speed • Location • Length
Station	Entity	• Station ID • Walking Time Parameters • Traits (U-turn, Bypass, Interchange, Shared Track) • Maximum Capacity • Occupancy List (list of Passenger IDs currently in the station)
Platform	Entity	• Platform ID • Dwell Time Parameters • Maximum Capacity • Occupancy List (list of Passenger IDs currently at the platform)
Passenger	Agent	• Passenger ID • O-D Path (temporally ordered list of platforms to board, transfer, alight from trains) • Walking Speed Parameter
Train	Agent	• Train ID • Schedule ID • Occupancy List (list of Passenger IDs currently in the train) • Schedule • Length

Table 2.1: Attributes of Agents and Entities in the Rail Simulator

Name	Type	Decisions
Passenger	Agent	• Walking Time (within a station)
Train	Agent	• Dwell Time • Speed • Acceleration/Deceleration Rate
Service Controller	-	• Denying Access to Station • Denying Access to Platform • Denying Access to Train

Table 2.2: Decisions Made by Agents and Service Controller

Ultimately, the goal of the rail simulator is to take input from the SimMobility Mid-Term and return output in the form of travel time and associated metrics. In the Mid-Term, passengers are Agents as well, with daily activity schedules, trips, mode choices, and route choices generated by the Pre-Day module, and a "state" corresponding to their current action. When the rail-mode portion of a trip commences, the passenger-Agent's state changes to <<in the rail system>>; at this moment, he is "passed" to the rail simulator, and is hence visible to it. The exact location of this individual in the rail system is then calculated and updated at every time-step until he leaves the rail system; at this juncture, his corresponding state changes, he is "passed" back to the Mid-Term, and is no longer visible to the rail simulator. The individual's total travel time is also returned to the overall Mid-Term simulator, as well as performance metrics such as capacity reliability and travel time reliability.

Figures 2.3 and 2.4 below describe the flow of passenger-Agents and train-Agents in the rail simulator respectively. Each step is marked with a letter-number code that denotes whether it is an action taken by the *passenger-Agent* (P), *train-Agent* (V), *Service Controller* (C), or *User* (U), and its position in the sequence of events.

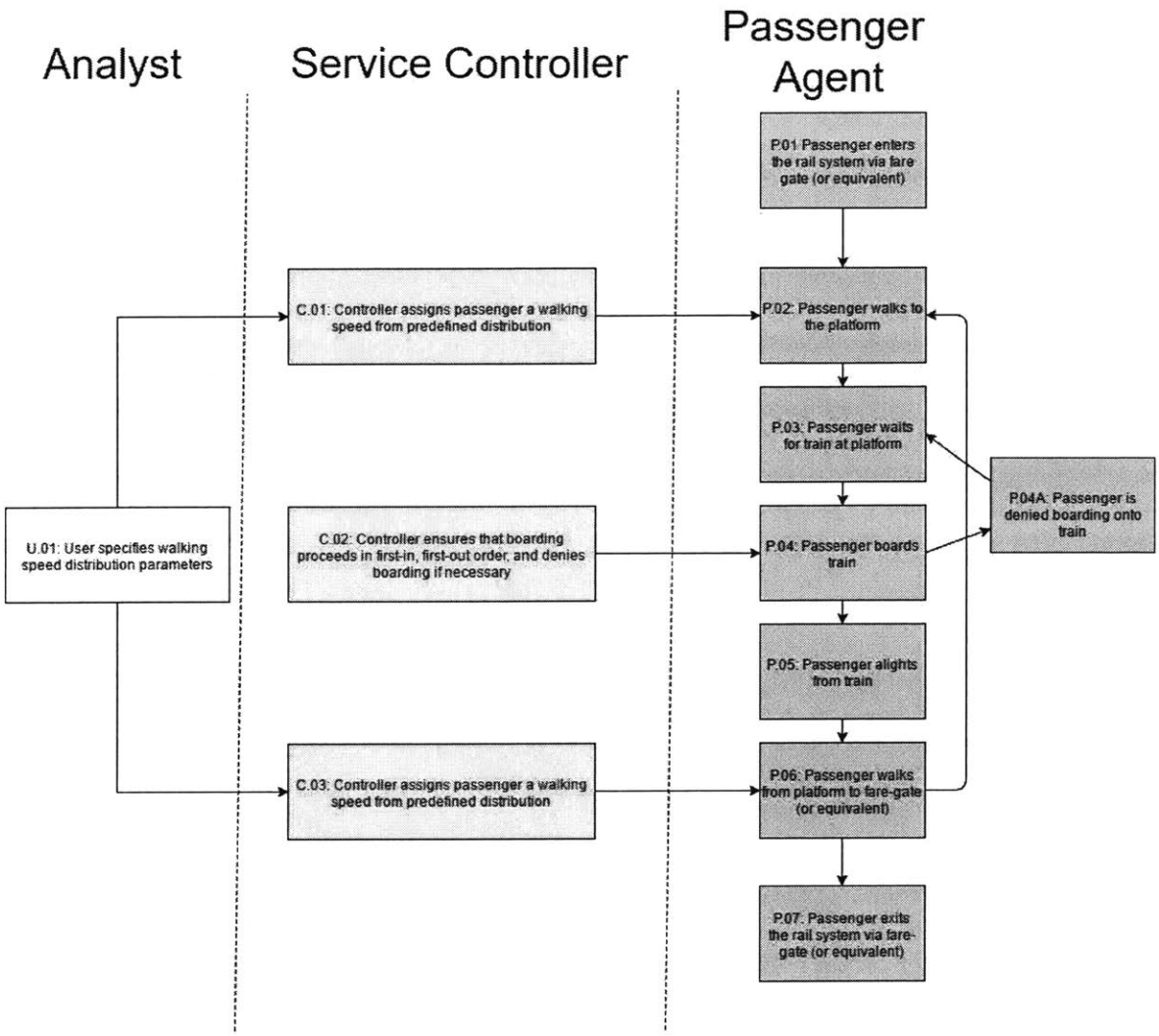

Figure 2.3: Passenger Flow in the Rail Simulator

It can be observed that passenger flow is essentially linear, consisting of 7 steps with minimal intervention from the Service Controller. The passenger agent officially enters the system (P.01) when it is passed from the overarching Mid-Term simulator to the rail simulator. This typically occurs when the passenger "taps-in" at a fare gate. He then decides his walking time to the platform, based on the attributes of the station he enters at and his own walking speed (P.02, C.01, U.01). The passenger queues in first-in, first-out (FIFO) order and is assumed to board the first train that arrives with sufficient capacity (P.03, P.04, C.02). If a train does not have sufficient capacity, the passenger is denied boarding

(P.03A), and has to wait for the subsequent train. Once boarded, the passenger-Agent "sleeps", with all movement between stations controlled by the train-Agent. The passenger is then "awoken" at his desired destination (P.05), and decides a walking time to the fare-gate (if exiting the system) or the next platform (if performing a transfer) (P.06, C.03, U.01). Note that the walking times in P.02 and P.06 are drawn from the same *percentile* of different distributions to maintain consistency in passenger walking speeds and keep the geometric features of the stations constant. Once the passenger exits the system (P.07), the train trip is determined to be complete and output is returned to the Mid-Term simulator. This will be further elaborated on in Chapter 3 as a key part of the calibration process.

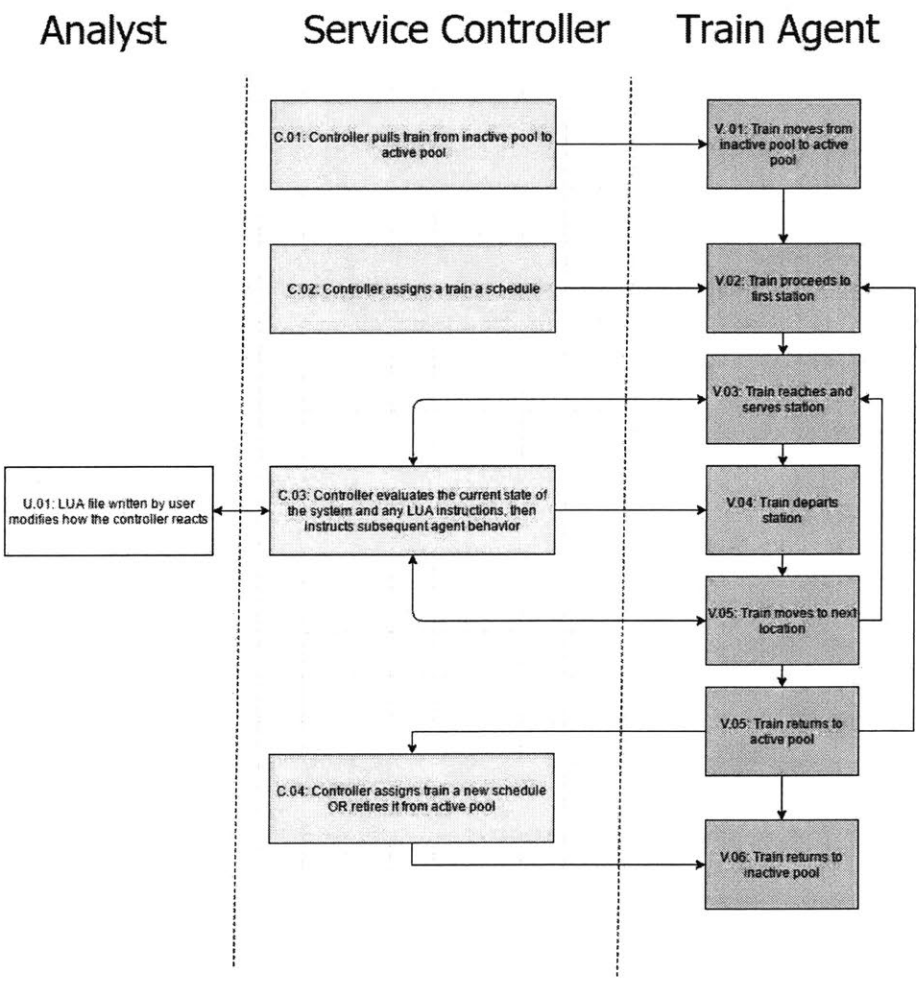

Figure 2.4: Train Flow in the Rail Simulator

In contrast to passenger flow, it can be observed that the flow of trains is significantly more complicated. Service begins when a train is pulled from the inactive pool to the active pool by the Service Controller (V.01, C.01). The former is analogous to a train depot, while the latter is analogous to an out-of-service train waiting at a terminal station. The train then receives schedule information comprising its departure time from the first station, as well as a list of stations to service and its expected arrival time at those stations (V.02, C.02). When a train arrives at a station, it relays this information to the Service Controller, which checks for any especial instructions for the train from the user (U.01). If there are none, the Service Controller boards and alights passengers, calculates a dwell time for the train based on dwell time models, and instructs its subsequent behavior (V.03, C.03). Similarly, as the train moves between stations, its relative position to other trains and stations in the line is constantly monitored by the Service Controller, and its speed adjusted accordingly subject to the train movement models and especial instructions from the user, if any (V.04, C.03, U.01). Finally, when the train reaches the end of the line, it conveys this information to the Service Controller, which typically assigns the train a new schedule—usually the corresponding return route (V.05, C.04). In rare cases, the Service Controller will instead retire the train from the active pool (V.06, C.04), essentially sending it back to the depot. This occurs when the desired arrival frequency decreases—for example, when the system changes from peak scheduling to off-peak scheduling.

In the subsequent sections, we will elaborate on the implementation of the train flow diagram, starting with the active/inactive pool system, followed by station behavior, movement and signaling, and finally the Service Controller.

2.3 Active and Inactive Pools

A typical rail system is likely to contain multiple types of trains running at once, each with different properties—for example, older trains are likely to have a slower acceleration rate and maximum speed.

In some cases, trains may differ in length and passenger capacities as well, amongst other properties. As such, it is important to distinguish between individual trains, by assigning each train an ID.

It is also important to distinguish between different *states* of a train. A train may be in active service (moving along the line and ferrying passengers); however, it may also be waiting (idle at a terminal station or side-track waiting for its next schedule to begin), or out-of-service (that is, at rest in the depot). The key difference between the latter two states is the effect they have on the system. Trains that are "waiting" are still occupying physical space in the rail network, and can be called into service very quickly. On the other hand, trains that are out-of-service occupy no space—depots are assumed to have infinite capacity—but take a while to enter service.

This distinction is simulated by having an "active pool" and "inactive pool" at each end of the line. We use the NEL, which has 28 trains total, as an example in Figure 2.5 below. The arrows represent *acceptable* directions of movement between the end stations, active pools, and inactive pools:

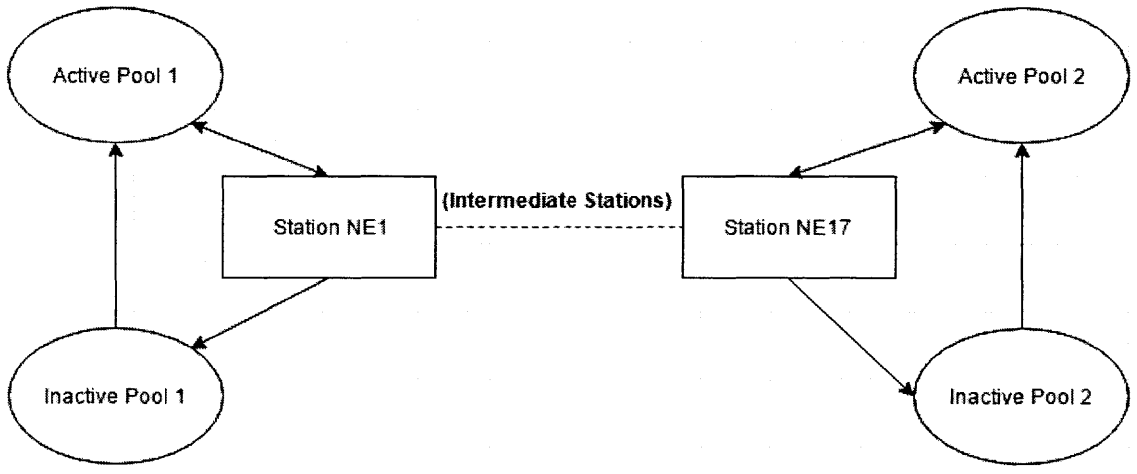

Figure 2.5: Active & Inactive Pools

Suppose that, during the AM Peak, the NEL has 20 trains in-service and 8 trains out-of-service. When the simulation is initialized, the *distribution* and *order* of trains is read from the database. For example, in the NEL trains are evenly distributed amongst both lines NE1 and NE2 and order simply corresponds to Train ID. As such, the simulation initializes with 10 trains (with IDs 1 to 10) in Active Pool 1, 10 trains (with IDs 15 to 24) in Active Pool 2, 4 trains (with IDs 11 to 14) in Inactive Pool 1, and 4 trains (with IDs 25 to 28) in Inactive Pool 2. (Naturally, the number of trains in each pool and their corresponding IDs may differ for other lines.) As the simulation runs, trains are drawn from in order AP1 and AP2 whenever a new train is dispatched from the end stations (NE1 and NE17 respectively).

When a train reaches the opposite end station from the one it departed, it by default joins the back of the queue in the active pool. Hence, for example, when Train 1 reaches NE17, it queues in Active Pool 2 behind Train 24. However, the Service Controller may alter this behavior if predefined conditions are met (for example, if the AM Peak is about to end and less trains are required in service), or if the user specifies contrary instructions via a LUA script.

A train in the inactive pool can never be called to service. However, the Service Controller may call inactive trains to the active pool, and from there they can be called into service. Similarly, trains in the active pool can be sent into the inactive pool. A time delay can be set when transferring trains between pools to mimic the real-world costs of moving an out-of-service train from depot to line, and vice versa.

Note that in the interests of simplicity, this formulation assumes that there are no restrictions on movement within the pools. For example, if there are only a few sidings behind the end station, it may result in trains forming a last-in, first-out queue. Issues may also arise with movement between the inactive and active pools—if the depots are located near the middle of the line, it may be difficult for trains to move between active and inactive pools without disrupting trains in active service. Therefore, if

additional accuracy is desired, the specific movement logic between the pools should be defined by the user via the Service Controller.

2.4 Train Movement & Signaling

2.4.1 Overview of Signaling Principles

The primary purpose of a signaling system is to help trains in a rail network maintain safe distances from each other such that each train has adequate room to brake as necessary. A "safe distance" is typically considered to be the braking distance of a train, plus some additional leeway known as the "safe operating distance". (It should be noted that in some rail networks, a safe operating *headway* is also implemented as a secondary measure; however, this can effectively be modelled in the same way as a safe operating distance). In cases where track is shared, especially when between trains running in opposite directions, signaling systems are also used to ensure that only one train runs on the track at any given time. These systems have also been utilized to implement operations research policies such as in (Keiji *et al.*, 2015)[15], where train speeds are intentionally reduced to minimize delays at stations.

Ultimately, a signaling system must be able to modify train *speeds*. There are two main types of signaling systems used worldwide: fixed-block and moving-block systems, which will be elaborated on in the next subsection. (Hill *et al.*, 1992) provides a more in-depth discussion of signaling principles and their implementation in rail simulators.

2.4.1.1 Fixed-Block Signaling

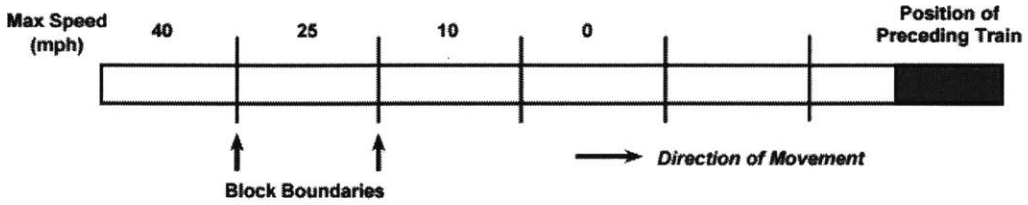

Figure 2.6: Fixed-Block Signaling

In fixed-block signaling, the track is divided into sections called blocks. Each block has a variable speed limit, as well as fixed values of train deceleration that depend on the specific track geometry that the block belongs to. Whenever a train occupies a block, the speed limit for that block and a certain number of blocks preceding it (corresponding to the safe distance) is set to zero. As the distance between a block and the preceding train increases, so does the speed limit at that block (subject to the maximum system speed limit). These limits are chosen according to the typical deceleration rates of trains on the line such that, in an emergency, a train will always be able to brake to a complete stop before coming into contact with the train in front of it. Depending on the type of signaling system used, each block can have anywhere between 3 to more than 10 different speed limit settings, known as "phases".

2.4.1.2 Moving-Block Signaling

In moving-block signaling, trains are continuously in contact with a central computer, and any one train always knows the speed and position of all the trains in the system as well as its own deceleration rate. A train operating under moving-block signaling will continuously calculate the distance to the point ahead where it must come to a stop—either the next station, or the end of the preceding train plus the safe operating distance—and continuously adjust its speed such that it will be able to decelerate fully over this distance.

2.4.1.3 Design Considerations

Moving-block signaling systems are generally easier to implement in simulation models as they only require train acceleration rates, deceleration rates, maximum speeds in a rail system, and the regulated safe operating distances or headways. In contrast, modelling fixed-block signaling systems requires knowledge of all of the above as well as block lengths along the track, the number of phases a block has, and the speed limits that correspond to these phases.

It should be noted that, even if a moving-block signaling system is used, block-Entities are still a valuable tool for capturing track properties at specific locations. For example, a block-Entity could be used to

model a section of track with a large gradient and corresponding lower-than-normal maximum acceleration rate.

2.4.2 Signaling and Train Movement Framework

Due to the popularity of both fixed-block and moving-block signaling, it is important for the rail simulator developed in this thesis to be able to simulate both systems. This is not particularly difficult, as the latter can be thought of as a special subset of the former, where block lengths approach zero and the number of phases approach infinity. As such, it is sufficient to ensure that trains:

1) Receive *information* about target speed limits, based on current system conditions; and
2) Adjust their speeds *over time* (non-instantaneously) to meet these speed limits.

We therefore require functions to determine *target speeds*, *desired accelerations/decelerations*, and the *distance travelled* at each time-step (of duration Δt), with these values then being used as inputs for the next time-step, and so on.

As demonstrated in (Kraft, 2013), train movement between stations can be approximated with a trapezoidal speed-time profile—that is, in regions of constant acceleration, constant speed, and constant deceleration—with minimal loss of accuracy:

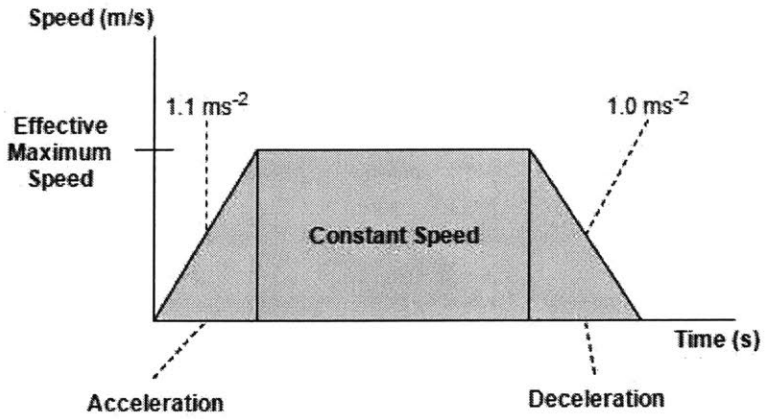

Figure 2.7: Simulated Speed-Time Profile

Note that Figure 2.7 assumes that the acceleration rate to maximum speed and deceleration rate to zero speed are constant. This is largely the situation in real life, and the rare cases where acceleration/deceleration rates change *as the train accelerates/decelerates* (when the train transitions between blocks with different attributes) are ignored in the following formulation (if necessary, these special cases of train movement can be manually modelled via the Service Controller). This allows train movement to be calculated via relatively straightforward kinematic equations:

2.4.2.1: Start of Time-Step

At each time-step, every train in the system reports its speed $u_{n,initial}$, location $x_{n,initial}$ (where n is the train ID, and x is taken relative to the first station in the line) and corresponding block acceleration and deceleration rates a_{block} and d_{block} to the Service Controller. The distance between the *front* of the train and the *end* of the train in front of it, minus the safe operating distance, is calculated by the Controller and defined as s_{trains}. The distance between the *front* of the train and the *front* of the next station, $s_{station}$, is also calculated. This is shown for "Train 1" in Figure 2.8:

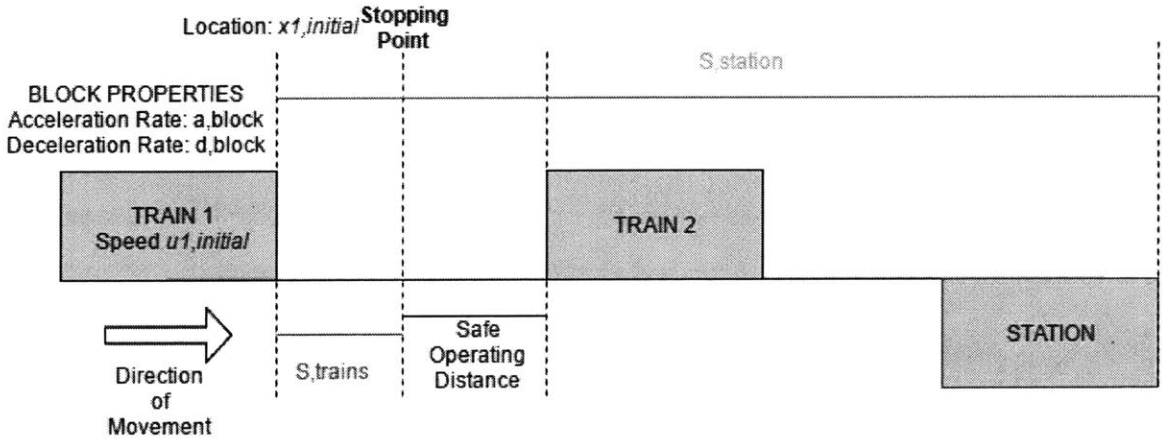

Figure 2.8: Defining Train Movement Variables

Now the remaining stopping distance s is calculated:

$$s_{stopping} = \min(s_{trains}, s_{station}) \tag{2.1}$$

In Figure 2.9, the remaining stopping distance is s_{trains}, since Train 2 less the Safe Operating Distance is nearer to Train 1 than the station is.

The theoretical speed limit $u_{max,t}$ such that the train will have an adequate braking distance is obtained:

$$u_{max,t} = \sqrt{-2d_{block}s} \tag{2.2}$$

If a safe operating headway t_{safety} is also implemented, a second theoretical speed limit $u_{max,t2}$ corresponding to this condition is calculated as well:

$$u_{max,t2} = \frac{s_{stopping}}{t_{safety}} \tag{2.3}$$

This is the speed at which the distance that the train would cover in the safe operating headway equals the distance to the train in front of it.

$u_{max,t}$ and $u_{max,t2}$ are then compared to the predefined system speed limit u_{system} and the actual speed limit $u_{max,actual}$ is taken as the minimum of the three:

$$u_{max,actual} = \min(u_{max,t}, u_{max,t2}, u_{system}) \tag{2.4}$$

There are two cases that may result:

1) A "stopping case", where the train prepares to brake to a complete halt, and
2) A "speed adjustment case", where the train simply adjusts to a safer speed.

If the stopping point ahead is derived from the rear of the train ahead, and the speed of the train ahead is not zero, a "speed adjustment case" occurs. This is the case in Figure 2.8. Train 1 simply adjusts its initial speed $u_{1,initial}$ to the calculated limit $u_{max,actual}$ by accelerating (if it is too slow) or decelerating (if it is too fast).

If the stopping point ahead represents the front of the next station, or if the train ahead has a current speed of zero, this represents a "stopping case." In this case, the train will begin to decelerate to a velocity of zero once it exceeds the speed limit $u_{max,actual}$.

2.4.2.2 Updating Variables for Subsequent Time-Step

2.4.2.2.1 Speed Adjustment Case

Since we are calculating *continuous* changes in variables at *discrete* intervals (that is, the variables are only updated at intervals of Δt), it is important to validate the estimated train trajectories to ensure that they do not "overshoot" their target speeds or stopping points. Again using Figure 2.8 as an example, we first calculate the theoretical final velocity $u_{1,final,t}$ of Train 1:

$$u_{1,final,t} = u_{1,initial} + a_{block}\Delta t \ (if\ accelerating) \tag{2.5.1}$$

$$u_{1,final,t} = u_{1,initial} - d_{block}\Delta t \ (if\ decelerating) \tag{2.5.2}$$

The actual final velocity, $u_{1,final,actual}$, is then obtained by comparing with the speed limit of the first time-step, $u_{max,actual}$:

$$u_{1,final,actual} = \min(u_{1,final,t}, u_{max,actual}) \ if\ accelerating \tag{2.6.1}$$

$$u_{1,final,actual} = \max(u_{1,final,t}, u_{max,actual}) \ if\ decelerating \tag{2.6.2}$$

This actual final velocity, $u_{1,final,actual}$, is the new train speed for time-step t+1.

The effective acceleration for Train 1, $a_{1,eff}$ (which can be either positive or negative) is thus:

$$a_{1,eff} = \frac{u_{1,final,actual} - u_{1,initial}}{\Delta t} \tag{2.7}$$

The distance travelled by Train 1 during this time-step, $s_{travelled}$, is calculated:

$$s_{travelled} = u_{1,initial}\Delta t + \frac{1}{2}a_{1,eff}(\Delta t)^2 \tag{2.8}$$

Finally, the train position is updated by adding the distance travelled by the train to its original position:

$$x_{1,final} = x_{1,initial} + s_{travelled} \tag{2.9}$$

2.4.2.2.2 Stopping Case

Similarly, there is also a need to validate train trajectories in the "stopping case". We first calculate the effective acceleration (the following notation will assume Train 1 is stopping):

$$a_{1,eff} = \frac{-u_{1,initial}^2}{2s_{stopping}} \tag{2.10}$$

This allows us to determine the final velocity:

$$u_{1,final,actual} = \max(0, u_{1,initial} + a_{1,eff}\Delta t) \tag{2.11}$$

Note that Equation 2.11 ensures that "over-decelerating", which may cause trains to move backwards, does not occur despite the discrete time-steps.

The distance travelled by Train 1 during this time-step, $s_{travelled}$, is calculated:

$$x = \frac{(u_{1,final,actual}^2 - u_{1,initial}^2)}{2a_{1,eff}} \tag{2.12}$$

Finally, the train position is updated by adding the distance travelled by the train to its original position:

$$x_{1,final} = x_{1,initial} + s_{travelled} \tag{2.13}$$

2.4.2.3 Potential Issues

It should be noted that the approach in the preceding subsections essentially applies continuous equations to discrete time-intervals. If the time-step used in the simulation is small, this is an acceptable approximation; however, if the time-step is large, unreasonably high deceleration values may occur due to loss of accuracy in estimating train positions. In order to allow the simulator to yield coherent results,

a solution was introduced, where trains to decelerate one time-step earlier than strictly necessary in certain cases.

At every time-step *t*, after a train calculates its velocity and position in the next time-step *t+1* but *before* these values are updated, it should also calculate the deceleration required in *t+1*, given these values, to stop at the stopping point. If this theoretical value is larger than the pre-defined maximum system deceleration rate, the train instead begins decelerating immediately, at time-step *t*. Its position and velocity in *t+1* are then re-calculated and updated. This will cause the train to decelerate at a slightly slower rate than what would occur in real life. However, the discrepancy diminishes as the length of the time-step *t* decreases, and is unlikely to significantly affect the accuracy of the simulation.

2.5 Passenger-Train Interaction: Station Models

2.5.1 Overview of Station Behavior

A typical rail system is likely to contain several types of stations. In the rail simulator, stations can have multiple properties from the following list:

1) **U-Turn:** There is a siding within, or in the vicinity of, the station such that trains can reverse and switch to the corresponding line heading in the opposite direction;
2) **Bypass:** There is a siding for the train to overtake the train in front of it, or allow trains behind it to overtake, or simply wait for further instructions without obstructing other trains;
3) **Interchange:** The station is shared by multiple lines and passengers may transfer between lines;
4) **Shared Track:** Multiple lines share the same track; trains may switch to different lines upon leaving the station.

A train-Agent will stop at all stations listed on its schedule. If a train is not scheduled to stop at a station, but the station is occupied, the Service Controller checks if the station is a bypass station. If so, the train

continues its route; if not, the train obeys signaling logic and waits behind the preceding train until allowed to proceed. This decision logic is shown in Figure 2.9 below:

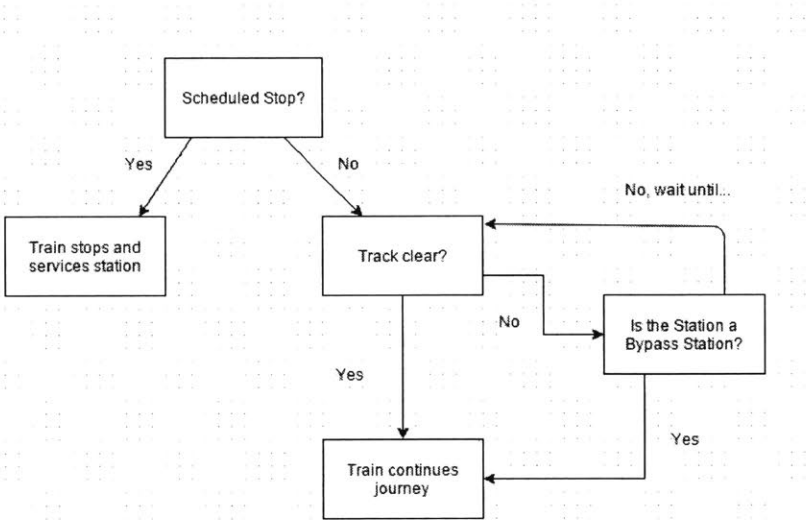

Figure 2.9: Approaching Station Decision Logic

Upon arriving at a station, a train first references its schedule, along with any altering input from the user via the Service Controller, to determine its departure behavior (to continue to the next station on the line, switch to a different line, wait at a siding, etc.). The train, station, and platform passenger lists are then updated according to the origin-destination matrix of passengers at the station, and the change in occupancy used by the Service Controller to calculate dwell times. Next, the Service Controller calculates any additional delays that may occur due to intended behavior—for example, a train attempting to "U-turn" must wait until the corresponding track on the opposite line is clear—or external factors, such as holding policies specified by the user. Finally, the train departs after the total required delay resolves.

2.5.2 Calculating Dwell Time

Dwell time is defined as the time that elapses between the "door opening" and "door closing" phases of an active train at a platform, and is primarily devoted to the loading and unloading of the train. Dwell

time is affected by both system specific factors such as passenger loads and behavior, and external factors that can affect operating conditions (Kraft, 1975)[17]. However, given that the latter is hard to quantitatively measure, they are usually not included in dwell time models, but considered through random parameters.

Previous works (Lin *et al.*, 1992), (Puong, 2000) suggest that a straightforward approach to modelling dwell time is as a linear function of passenger volume:

$$DT = \beta_0 + \beta_1 Boarding + \beta_2 Alighting \tag{2.14}$$

If additional accuracy is desired, this model can be improved to better capture the effects of congestion:

$$DT = \beta_0 + \beta_1 Boarding + \beta_2 Alighting + \beta_3 Congestion \tag{2.15}$$

where the congestion term is a function of *Boarding*, *Alighting*, and *Occupying* passengers, and can have more than one associated parameter, such as $\beta_3 AS + \beta_4 BS^2 + \cdots$.

In both models, *boarding* represents the total number of boarding passengers, *alighting* the total number of alighting passengers, and *congestion* a combined term that reflects the interactions between passengers staying on-board (through-standees) and alighting passengers; through-standees and boarding passengers; alighting and boarding passengers; interference within alighting passengers as their number increases, and interference within boarding passengers as their number increases. The composition of the *congestion* term will vary from system to system or indeed even from line to line, and can only be determined from an understanding of existing conditions and an analysis of which model best fits the available empirical data. Indeed, many complex models have been proposed to accurately estimate dwell time. For example, Puong suggests the following nonlinear model for the MBTA Red Line:

$$DT = 12.22 + 2.27 B_d + 1.82\, A_d + 6.2 * 10^{-4} T S_d^3 B_d \tag{2.16}$$

where B_d and A_d represent the boarding and alighting passengers *per door* and TS_d represents the through-standees *per door*. However, these complex models are usually difficult to estimate.

In the interests of simplicity, and for ease of the calibration in Chapter 3, we utilize a *linear* dwell time function. To capture the interference between through-standees and alighting/boarding passengers in a straightforward manner, we assume that the *congestion* term is simply the occupancy of the train:

$$DT = \beta_0 + \beta_1 Boarding + \beta_2 Alighting + \beta_3 Occupancy \quad (2.17)$$

This will be shown in Chapter 3 to be a reasonable model for the existing data while being relatively straightforward to calibrate.

2.6 The Service Controller

In the daily operations of a rail line, train service often deviates from schedule due to changing circumstances such as variability in passenger demand, driver behavior, or other unforeseen operating conditions. If these deviations are not corrected, they tend to increase in magnitude, rendering schedule adherence difficult. As such, every rail line is typically managed by a central control station, which not only defines regular operations, but also modifies them in real-time both proactively and reactively in order to ensure the system is able to continue functioning.

In this rail simulator, this central control is represented by the Service Controller, which can access internal functions (such as the dwell time function in [2.3] or the train movement functions in [2.4]) to change train behavior parameters. In conjunction with this feature, the Service Controller uses externally-written LUA scripts as an interface to modify the behavior of the rail simulator and thus simulate various scenarios. For example, a LUA script could be used to modify train speeds and accelerations to test the effects of introducing brand-new trains into the system. On a more complex note, LUA scripts may also be used to implement various policies or simulate disruptions. Section 4.3

provides an in-depth example of a LUA script being used to simulate a major disruption that occurred in Singapore in 2013.

The LUA scripts interact with the rail simulator through Application Program Interfaces (APIs), which allow different aspects of the rail simulator to be modified. Each API has a defined set of inputs, target functions to modify, and resulting outputs. For example, the API reset_speed_limit takes the inputs *new speed limit, start platform, end platform, and line ID* (in that order) to set the maximum speed of trains along a portion of a line.

There are 2 main categories of APIs. The first, *Command APIs*, are used to instruct the simulation to perform actions that deviate from typical behavior. These include modifications to train speed, scheduling, and so on. The second, *Informational APIs*, report on the status of various elements within the simulation, such as the number of current active trains or the ID of the next train to arrive at a station, and are primarily used as inputs to Command APIs. A comprehensive list of APIs (their descriptions, inputs, and expected output) can be found in Appendix 1.

LUA scripts can also be broadly categorized into two main categories, with different purposes:

1) **Reactive:** The user specifies if...then scenarios, which modify train operations in real-time during the simulation if conditions are met.
2) **Proactive:** The user *creates* scenarios that proactively modify train operations in order to simulate and observe the effects of events that may potentially occur.

A major advantage of the Service Controller is its extreme flexibility, which allows the rail simulator to be customized for vastly different scenarios. For example, headway control (a *reactive* scenario) and disruption simulation (a *proactive* scenario) are two completely different areas of research, and have typically been investigated via the construction of customized simulators (as seen in (Ravichandran, 2013), where a simplified simulator is developed to investigate service control strategies). Via the

Service Controller, these scenarios can now be easily modelled in the rail simulator, benefiting from the integration that it provides.

This is demonstrated in Section 4.2, which simulates the effects of a major disruption that did indeed occur in Singapore in 2013. The structure of the overall disruption is briefly described here as an example of the ability of the Service Controller to simulate complex scenarios. Section 4.2 elaborates further on the scenario, and the corresponding LUA code can be found in Appendix 2.

The situation of the disruption is as follows:

From 8AM to 10 AM, stations NE11 through NE17 are closed to all traffic. Trains that are caught in this affected section of track proceed to the next platform, where all passengers are forced to disembark. If the next platform is occupied, the current train ignores the safe operating distance, and continues moving under "stopping case" logic, stopping directly behind the train in front of it and force-disembarking all its passengers as well. Trains in the unaffected regions run in a loop from NE1 to NE10. Apart from this, trains otherwise behave normally—there is no change in maximum speed, movement behavior, and so on.

From 10AM onwards, all behavior returns to normal.

CHAPTER 3: CALIBRATION

3.1 Overview

The typical approach to calibrating a simulation model with many unknown parameters is exemplified in (Antoniou *et al.*, 2015). Firstly, the distributions of components within the simulation (such as walking time or dwell time) are estimated or obtained from field data in order to determine the specific parameters to calibrate. The parameters are then simultaneously estimated through multiple iterations of the simulation model. As expected, as the number of parameters to calibrate increases, so does the running time and computational resources required by this calibration method.

The goal of this section is to introduce the *sequential calibration* methodology, a disaggregate calibration of the individual models within the rail simulator that accounts for model dependencies (that is, the effects that individual models have on each other). In this approach, we relied only on AFC data and not on ATC data, which is becoming increasingly hard to obtain due to security concerns. This AFC data was used together with train network data (such as train speeds and accelerations), publicly available GTFS data, and minimal manually-obtained data to estimate parameters on a stage-by-stage basis. The results of each disaggregate calibration were then checked against existing data and then used as input for the next calibration stage (of the next associated model). For example, passenger walking time parameters were estimated, verified against station counts, and then used to estimate train dwell time parameters. This was possible due to the nature of AFC data (which captures commuters' points and times of entry and exit into/from the system, thus providing a good picture of *overall* travel times), as well as well-structured individual components of travel time, which allowed for reasonable simplifying assumptions to be made.

Ultimately, we aim to demonstrate the potential of sequential calibration as an accurate "first step" to generate appropriate initial seed values for a secondary simultaneous calibration endeavor.

To accommodate the sequential calibration method, a modified version of the rail simulator was developed with help from the Singapore-MIT Alliance for Research and Technology Centre (SMART). This

modified simulator was designed to be *unintegrated*, with passenger-Agents being created at the origin station and destroyed after exiting the destination station of their train trips. Demand data (passenger ID, origin, destination, trip start time, etc.) was obtained from a user-provided input file and limited to movement within the train network. Similarly, output data (passenger ID, trip start time, train board time, trip end time, etc.) produced by the modified simulator was limited to only the train network.

3.2 The Case Study

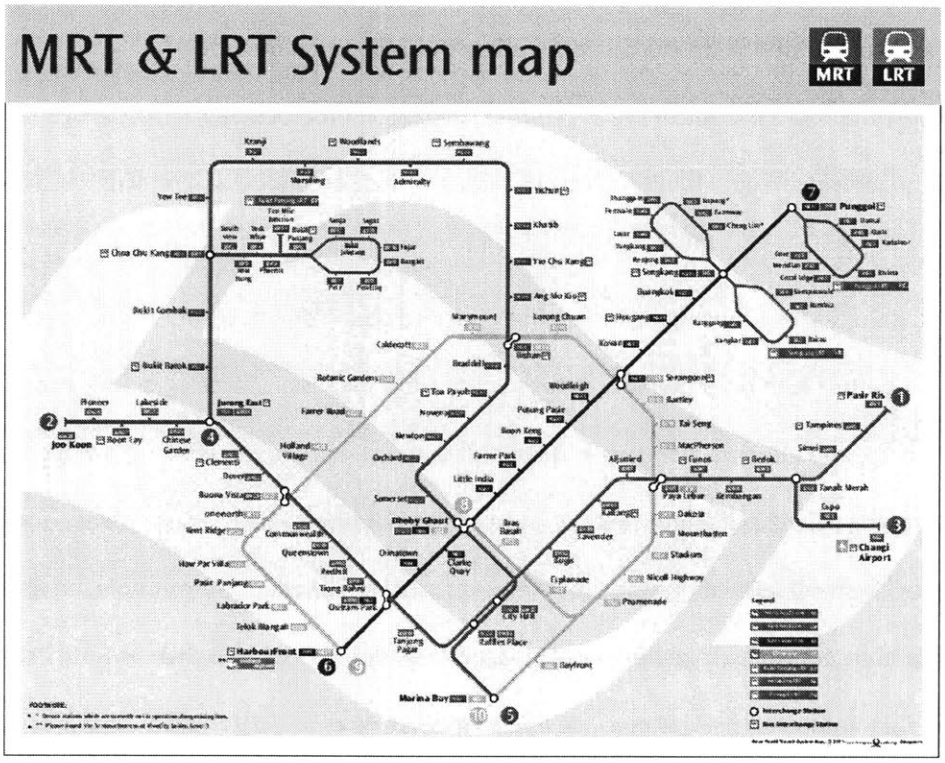

Figure 3.1: Singapore Rail Network

In this section, we use the calibration of the Singapore North-East Line (NEL) as an example, although the concept demonstrated here can be easily applied to other rail networks as well. The NEL (the purple line in Figure 3.1, with end stations ⑥ and ⑦) is fully automated and has a total of 28 operating trains servicing 16 stations across 20km. It has a daily ridership of approximately 350000 commuters, and provides a direct route from the primarily residential north-east region of Singapore to the Central

Business District in the south. The calibration process was conducted using EZ-Link (AFC) data provided by Singapore's Land Transport Authority (LTA)—approximately 173 million trips, encompassing all rail and bus trips made in the month of August 2013. Trip data included start and end locations, start and end time, and trip date. Operating specifications for the NEL were obtained from SBS (the NEL operator) and can be found in Appendix 3.

3.3 Data Structure

As shown in Figure 2.3 (Section 2.2), the flow of a passenger through the rail system consists of 5 components: walking from the fare gate to the platform, waiting at the platform for the train to depart, travelling (that is, time spent *in motion* on the train) to the destination, dwelling at intermediate stations, and walking from the platform to the fare gate at the destination. This can be expressed via the following equations:

$$\mu_{\{i,j\}} = \mu^{i}_{walk} + \mu_{wait} + \sum_{k=i}^{j-1} \mu^{\{k,k+1\}}_{travel} + \sum_{k=i+1}^{j-1} \mu^{k}_{dwell} + \mu^{j}_{walk} \tag{3.1}$$

$$\sigma^2_{\{i,j\}} = \sigma^{2^{i}}_{walk} + \sigma^2_{wait} + \sum_{k=i}^{j-1} \sigma^{2^{\{k,k+1\}}}_{travel} + \sigma^2_{dwell\{all\ stations\}} + \sigma^{2^{j}}_{walk} + 2Cov(i,j) \tag{3.2}$$

That is, the expected travel time between stations {i, j} assuming no transfers is equal to the sum of:

1) The expected walking time within station {i};
2) The expected waiting time, which we assume to be independent of station;
3) The sum of expected travel-time-between-stations on the train to the destination;
4) The sum of expected dwell times at intermediate stations, and
5) The expected walking time within station {j}.

If the components are assumed to be independent (except walking times within {i} and {j}), then the variance of the expected travel time between stations {i,j} can be expressed in a similar manner. Note

that the variance of the dwell times is represented as a variance for *total* combined dwell time, since the dwell times at stations are likely to be correlated for one specific trip.

The aim of sequential calibration is to:

1) Estimate the functions of distributions corresponding to each component;
2) Determine the corresponding mean μ and variance σ^2 for each component; and
3) Generate the appropriate function parameters for each component.

As the calibration progresses, information from the *previous* stage is used to estimate parameters of the *next* stage. It is therefore important to constantly ensure that the calibrated parameters are coherent to improve the accuracy of the calibration procedure.

In the interests of expediency, the calibration procedure makes five key assumptions:

1) The walk time from the fare gate to the platform is the same as the walk time from the platform to the fare gate in any given station *i*.
2) Passengers and trains arrive at stations independently from other passengers and trains respectively.
3) The travel time between two stations is *constant* and has *zero variance*.
4) μ_{walk} and μ_{wait} are independent.
5) Commuters are always able to board the first train that arrives (i.e. no denied boarding).

It should be noted that assumptions (1) and (4) are not strictly necessary—the former can be relaxed by doubling the number of walking time parameters, and the latter by introducing a second covariance term corresponding to the interaction between walking times and waiting times. However, these assumptions help expedite the calibration process; furthermore, walk time data collected for Section 3.3 seems to support the first assumption. Assumption (2) is a common assumption made in train

modelling, and can also be relaxed by examining passenger arrival rates for trends to predict when passengers arrive in clusters as opposed to singly. However, that is out of the scope of this thesis. Assumption (3) is a reasonable assumption given that the NEL is automated—trains are controlled by a computer, and run at almost identical speeds, with variance only occurring if a train delay affects the train behind it. However, the NEL rarely runs at peak frequency; as such, cascading delays are unlikely and travel time can be assumed to be generally constant. Finally, Assumption (5) was addressed by calibrating the rail simulator for both the off-peak and peak. In the former, denied boarding can be safely assumed to be zero; as such, it may be possible to compare trends across the two time periods and identify probabilities of denied boarding as an area of future work.

3.4 Estimating Train Speeds

As elaborated upon in Section 2.4, in the rail simulator trains move with a trapezoidal speed-time profile between stations; in regions of constant acceleration, constant speed, and constant deceleration. We once again assume that the acceleration and deceleration rates are constant, as shown:

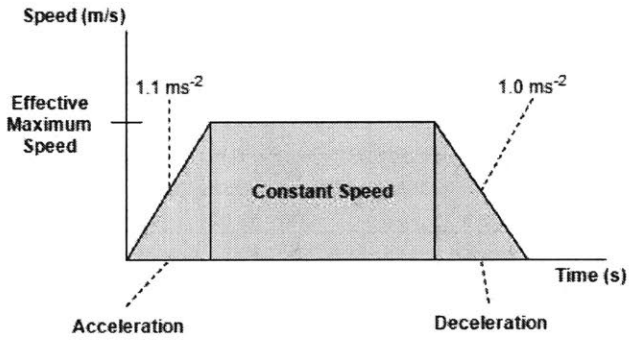

Figure 3.2: Speed-Time Profile of NEL Train

Although the NEL has a defined maximum *system* speed of 80km/h, the effective speeds between stations are affected by factors such as track gradient, curvature, section length, and so on, and as such are typically lower than the system maximum. It is therefore necessary to estimate these "effective maximum speeds" via the equation:

$$s^{\{i,i\pm1\}} = \frac{(u^{\{i,i\pm1\}})^2}{2a_{start-block}} + \frac{(u^{\{i,i\pm1\}})^2}{2d_{start-block}} + \left(\mu_{travel}^{\{i,i\pm1\}} - \frac{u^{\{i,i\pm1\}}}{a_{start-block}} - \frac{u^{\{i,i\pm1\}}}{d_{end-block}}\right) u^{\{i,i\pm1\}} \quad (3.3)$$

This is simply a corollary of the kinematic equations in Section 2.4. $s^{\{i,i\pm1\}}$ refers to the distance between two adjacent stations, $\mu_{travel}^{\{i,i\pm1\}}$ the expected travel time between these two stations, $u^{\{i,i\pm1\}}$ the effective maximum speed between two adjacent stations, $a_{start-block}$ the acceleration rate at the block immediately after Station *i*, and $d_{end-block}$ the deceleration rate at the block immediately preceding Station *i±1* respectively. Note that $\mu_{travel}^{\{i,i\pm1\}}$ is also a component of Equation 3.2.

To solve for all $u^{\{i,i\pm1\}}$, travel times $\mu_{travel}^{\{i,i\pm1\}}$ were obtained from the operator website and verified against manually collected data, while distances $s^{\{i,i\pm1\}}$ for each adjacent station pair were obtained from GTFS data. This yielded the following results:

Station ID			DIRECTION A → B (Line NE_1)		DIRECTION B → A (Line NE_2)	
A	B	Distances (m)	Travel Time (s)	Speed (km/h)	Travel Time (s)	Speed (km/h)
NE1	NE3	2720	171.5	63.3	180	59.6
NE3	NE4	470	75.1	24.7	75	24.7
NE4	NE5	700	75	39	70	43
NE5	NE6	1250	110	46	110.1	46
NE6	NE7	880	80	46.9	85	43.1
NE7	NE8	898	80	48.1	80	48.1
NE8	NE9	1160	95	51.3	90.1	55.4
NE9	NE10	1510	110	57.3	110	57.3
NE10	NE11	895	85	43.9	85	43.9
NE11	NE12	1160	90	55.5	95	51.3
NE12	NE13	1780	120	61.9	120	61.9
NE13	NE14	1530	115	54.8	115	54.8
NE14	NE15	1340	100.1	56.7	100	56.8
NE15	NE16	968	105	36.6	105	36.6
NE16	NE17	1700	125	55.5	125.3	55.3

Table 3.1: Effective Maximum Speeds on the NEL

It can be seen that the calculated speeds intuitively make sense—as the distances between stations decreases, so do the effective maximum speeds, as the train has less time to accelerate before being required to decelerate for the next station. A notable exception is the NE5 – NE6 (Clarke Quay – Dhoby Ghaut) corridor, which winds through a hilly part of Singapore and is therefore heavily affected by changes in track gradient and curvature.

3.5 Estimating Wait Times

In order to estimate the expected time (μ_{wait}) a passenger spends waiting at the platform for his train to arrive and its variance (σ^2_{wait}), the average headway between trains was assumed to correspond to the dispatch headways associated with different time periods. (Note that in the simulation model, as typical for subways and high-frequency public transit, trains only have *dispatch* timetables; they do not have departure timetables for every specific station).

According to GTFS data, there were 4 time periods with different dispatching headways:

1) 0800-0900 on weekdays/Saturday (AM Peak): 240s
2) 1715-1830 on weekdays/Saturday (PM Peak): 270s
3) All other times on weekdays/Saturday (Off-peak): 390s
4) All day on Sundays: 375s

Regardless of the actual distribution of passenger and train arrivals, the *wait time* for any individual commuter can be estimated as a uniformly distributed random variable between 0 and the headway. As such, it is trivial to calculate the expected waiting time and its variance:

$$\mu_{wait} = \frac{Headway}{2} \tag{3.4}$$

$$\sigma^2_{wait} = \frac{1}{12}(Headway)^2 \tag{3.5}$$

3.6 Estimating Walking Times

It has been shown in (Zhu, 2014) that passenger walking speeds within train stations tend to follow a lognormal distribution. Although there have been studies into what the parameters of this distribution should be, it is difficult to convert walking *speeds* into walking *times*, as the lengths of the various paths from fare gate to platform vary from station to station and are difficult to obtain. We therefore attempt to estimate walking *times* directly, by assuming an inverse lognormal distribution:

$$Walktime\{i\} = \frac{1}{e^{\alpha\{i\}+\beta\{i\}Z}} \tag{3.6}$$

where α and β are parameters unique to each station and Z is a standard normal random variable.

Note that the inverse lognormal distribution estimated in Equation 3.6 is effectively a sum of multiple inverse lognormal distributions, each corresponding to a walking time from a specific entrance to the platform. This simplification is necessary, as we lack information on the *specific* fare gate through which a commuter enters the train station; furthermore, performing an estimation at such an intricate level of detail would require much more data and computational power, thus defeating the purpose of the sequential calibration approach.

The AFC data was filtered to only include trips made between adjacent stations on the NEL. These consisted of 0.1% of the full dataset, or approximately 800,000 trips, that had *zero* dwell time at intermediate stations. The means and variances of inter-station travel time and wait time (as estimated in the previous subsections) were then subtracted from the combined travel time, resulting in the following *reduced equations:*

$$\mu^{\{i,j\}}_{combined\ walktime} = \mu^{i}_{walk} + \mu^{j}_{walk} \tag{3.7}$$

$$\sigma^{2\{i,j\}}_{combined\ walktime} = \sigma^{2^i}_{walk} + \sigma^{2^j}_{walk} + 2Cov(i,j) \tag{3.8}$$

Note that Equation 3.7 is a reduced form of Equation 3.1; similarly, Equation 3.8 is a reduced form of Equation 3.2. This results in 15 sets of mean/variance equations, one for every adjacent station pair. In

order to solve for the 16 unknown mean/variance variable pairs, walk time data was manually collected for peak and off-peak time periods at Woodleigh station. (It should be noted that since this method primarily relies on AFC data, the manual data collection can be performed at *any* station on the line, unlike traditional methods of data collection, which attempt to collect data at representative locations.) As such, Woodleigh station was chosen due to its relative small size and correspondingly few paths from fare gate to platform, thus ensuring accuracy in data collection. The system of equations was then solved to obtain the mean walk times for individual stations.

In order to estimate the *variance* of walk times at individual stations, we noted that the covariance of walking times between two stations is a function of their parameters:

$$Cov(i,j) = E[ij] - E[i]E[j] \tag{3.9}$$

$$\sigma_i^2 = E[i^2] - E[i]^2 \tag{3.10}$$

$$E[ij] = e^{-\alpha_i - \alpha_j + \frac{(\beta_i + \beta_j)^2}{2}} \tag{3.11}$$

$$and\ E[i] = e^{-\alpha_i + \frac{\beta_i^2}{2}} \tag{3.12}$$

The system of equations for walk time variances can therefore be reduced to a system of *implicit equations* for the parameters α and β, which was solved iteratively:

1) Parameters α and β were guessed for all individual stations.
2) Covariances for each adjacent station pair were calculated based on these parameters.
3) The system of equations was solved to yield variances for each individual station.
4) These resulting variances were used together with the means (estimated previously) to estimate α and β for all individual stations.

This process was repeated until convergence, with the results shown in Tables 3.2:

	OFF-PEAK				PEAK			
Station	μ^i_{walk}	$\sigma^{2^i}_{walk}$	α	β	μ^i_{walk}	$\sigma^{2^i}_{walk}$	α	β
NE1	85.27	2601.0	-4.2929	0.5530	98.22	3434.1	-4.4349	0.5519
NE3	61.03	1707.5	-3.9228	0.6143	72.86	2364.1	-4.1044	0.6069
NE4	51.95	1009.4	-3.7913	0.5637	58.45	2577	-3.7871	0.7498
NE5	32.65	128.5	-3.4288	0.3374	47.33	866.3	-3.6938	0.5717
NE6	104.68	2844.8	-4.5356	0.4804	118.52	2811.5	-4.6838	0.4271
NE7	43.48	192.5	-3.7237	0.3115	55.80	1329.4	-3.8441	0.5963
NE8	33.20	867.4	-3.2123	0.7619	48.36	1497.8	-3.6311	0.7035
NE9	39.90	11.2	-3.6828	0.0838	55.80	1150.8	-3.8646	0.5608
NE10	20.67	3.2	-3.0250	0.0867	38.50	715.9	-3.4537	0.6277
NE11	36.40	111.4	-3.5542	0.2841	38.13	317.0	-3.5424	0.4441
NE12	80.88	2804.7	-4.2146	0.5973	105.10	4470.7	-4.4850	0.5830
NE13	40.78	340.7	-3.6150	0.4317	45.53	930.0	-3.6331	0.6088
NE14	22.11	194.6	-2.9284	0.5789	46.55	1446.4	-3.5849	0.7150
NE15	48.30	330.5	-3.8112	0.3640	55.81	1254.6	-3.8527	0.5818
NE16	34.81	531.8	-3.3679	0.6033	53.94	1226.1	-3.8121	0.5930
NE17	68.25	109.5	-4.2116	0.1524	76.93	792.2	-4.2801	0.3545

Table 3.2: Station Walking Time Coefficients

In both tables, the values corresponding to NE11 (Woodleigh), the only manually-collected values, are highlighted in red. Although not shown, it was also noted that the recorded walking time distributions from gate-to-platform and platform-to-gate were generally similar. As expected, the expected walking time and its variance is larger during peak hours when compared to off-peak hours, due to both the general increase in crowdedness during these times as well as the "spillover effect" from denied boardings which occur in real life, but are unrepresented in the rail simulator. Large interchange stations such as Dhoby Ghaut and Harbourfront have very large variances, which is expected due to the many fare gates, and multiple paths that a commuter may take to the platform. Similarly, smaller stations such as Potong Pasir and Clarke Quay have significantly smaller variances. Finally, it should be noted that the estimated variances at some stations (such as Potong Pasir) during the off-peak are especially small. We attempted to correct this by bounding the minimum variance at the manually-measured amount for

Woodleigh (a small station as well). However, this was found to produce insignificant changes in α and β across the board, and hence the original values were used instead.

3.7 Estimating Dwell Times

It was previously demonstrated that dwell time can be modelled as a function of the number of boarding and alighting passengers, and the occupancy of the train. This dwell time is typically non-linear, with marginal effect of each additional passenger initially constant, but exponentially increasing as congestion effects start to occur. However, in the interests of simplicity, this thesis assumes a linear dwell time function as follows (for further elaboration, refer to Section 2.5.2):

$$DT = \beta_0 + \beta_1 Boarding + \beta_2 Alighting + \beta_3 Occupancy \quad (3.13)$$

This expedites the calibration process by reducing the number of parameters to estimate, while still being a reasonable approximation for the existing data and sufficient proof-of-concept for the sequential calibration method introduced in this chapter. (The sequential calibration method can naturally be expanded to include non-linear dwell time functions, at the cost of slightly longer calibration times).

In order to estimate dwell time parameters, the modified rail simulator described in Section 3.1 was updated with all estimated models from the previous sections. The AFC data set was then filtered to include only trips that both started and ended at NEL stations (deemed *NEL-exclusive trips*) on an arbitrarily chosen weekday (Tuesday, 6 Aug 2013). This data set, consisting of approximately 190,000 trips, was fed as input demand to the modified simulator, and the resulting simulated travel times for each passenger-Agent compared against the actual travel times for the same Agent obtained from AFC data. The dwell time parameters were then adjusted using the simultaneous perturbation stochastic approximation (SPSA) approach described in (Spall, 1998)[27], and the iterative process repeated until convergence of simulated and actual travel times. This will be further elaborated in Section 3.7.2.

3.7.1 Preparing the Data Set

3.7.1.1 Pruning the Data Set

Although the data set consisted of electronically recorded historical trips, bad data was still present due to unexpected real-world conditions. For example, one recorded trip showed a passenger covering 11 stations in less than 300 seconds, (a more likely reason, although speculative, is that the passenger forgot to "tap-in", resulting in the system assigning the passenger the most expensive possible fare), while in another, a passenger took almost 4 hours to cover 3 stations (likely indicating an alternative purpose for entering the rail network).

It was therefore necessary to eliminate bad data to ensure accurate estimation of the dwell time parameters. We would ideally like to set upper and lower bounds on plausible travel times as follows:

$$LB_{\{i,j\}} = (1 - \alpha_1) * \sum_{k=i}^{j-1} \mu_{travel}^{\{k,k+1\}} \tag{3.14}$$

$$UB_{\{i,j\}} = (1 + \alpha_2) * \sum_{k=i}^{j-1} \mu_{travel}^{\{k,k+1\}} + 2 * (WalkTimeLimit) + \alpha_3 * (MaxWaitTime) \tag{3.15}$$

That is, choosing the lower bound for a trip between stations {i, j} to be a percentage (determined by a safety factor α_1) of the expected travel time between those stations (estimated in Section 3.2). This excludes the intermediate *dwell times* as well as *walking times* to the platform, which dwarf any variations in travel time that may arise from exceptionally fast trains. Similarly, defining the upper bound was to be a percentage (determined by a safety factor α_2) of the expected travel time, plus a large amount of leeway for walking time (using the longest possible walking time) and waiting time at platform (assuming a maximum of α_3 instances of denied boarding).

Unfortunately, it is difficult to justify the "ideal" parameters to use without statistics on passenger behavior. For example, since passenger walking times are inverse lognormal and therefore can

theoretically reach infinity, it is hard to justify arbitrarily choosing a specific value. We therefore pruned the data set by simply considering 2.5th to 97.5th percentile of travel times, with trips lying out of these bounds being considered erroneous. This resulted in approximately 9500 trips (5% of the original data set) being removed, with more than sufficient (180500) trips remaining to perform an accurate estimation of dwell time parameters. As a validation, the first method was also attempted with values α_1 = 0.2, α_2 = 0.2, α_3 = 3, WalkTimeLimit = 210s (double the largest expected walk time), and MaxWaitTime = 270s (the headway during peak hour, where denied boardings are most likely). This resulted in approximately 4500 trips (2.37% of the original data set) being eliminated. These 4500 eliminated trips formed a subset of the 9500 eliminated trips calculated via the first method.

3.7.1.2 Scaling the Data Set

Since the artificial demand only consisted of NEL-exclusive trips, it was necessary to scale the demand to account for unrepresented trips which only used the NEL for a part of their journey (deemed *NEL-partial trips*). This was done via a 2-shortest-paths approach, which was deemed reasonable as the structure of the Singapore train network rarely allows for more than 2 feasible paths between stations. Path length was determined by the number of stations along the path, with transfer stations given no additional weightage. Other methods of determining path length such as historical travel time, shortest distance as judged from the map, weighting transfer stations, etc. could naturally have been used as well, but this method was chosen for its straightforwardness, as well as being an intuitive representation of how commuters might choose routes in real life (by looking at a network map).

We collated *all* rail trips made on 6 Aug 2013 and generated the two shortest paths (measured by number of intermediate stations) for each trip, with the probability of taking a specific path given by its relative length compared to its alternative. By comparing the number of *NEL-partial* trips against *NEL-exclusive* trips, we obtained appropriate scaling factors for alighting, boarding, and occupying passengers at each station across multiple time periods.

For example, Figure 3.3 demonstrates the procedure for a commuter taking an *NEL-partial* trip from Marymount to Chinatown:

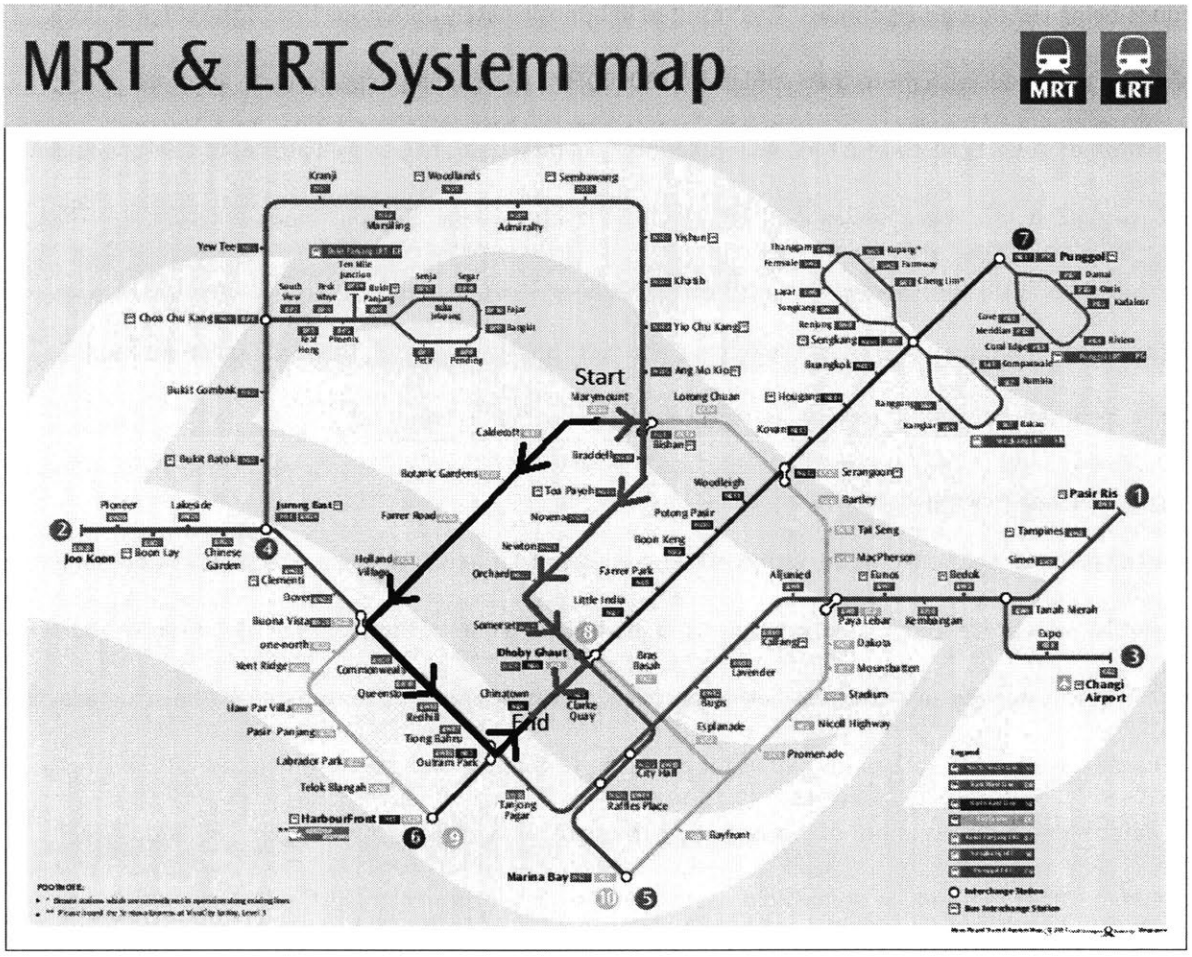

Figure 3.3: Marymount to Chinatown Route Choice Estimation

The shortest path between the two stations is highlighted in grey and consists of 11 stops, while the second-shortest path between the two stations is highlighted in black and consists of 12 stops. The probability of the commuter taking the shortest path is estimated to be 12/23 = 0.52, and the probability of the second-shortest path, 11/23 = 0.48.

The NEL segment of the shortest path is *Dhoby Ghaut – Clarke Quay – Chinatown*; therefore, this passenger would represent an **boarding** at Dhoby Ghaut, an **occupying** at Clarke Quay, and an **alighting**

at Chinatown if he chose the shortest path. Similarly, the NEL segment of the second-shortest path is *Outram Park – Chinatown* and the passenger would represent a **boarding** at Outram Park and an **alighting** at Chinatown if he chose this path instead.

The number of unrecorded boardings, alightings, and occupyings (corresponding to NEL-partial trips) were then compared against their recorded counterparts (corresponding to NEL-exclusive trips) for each station. Since the dwell time equation (Equation 3.13) only considers passenger volume within the train and station, we scale passenger volume by *station*, and not by O-Ds.

This yields the following scaling table:

	BOARDING			ALIGHTING			OCCUPYING		
Station	**0500 - 1159**	**1200 - 1659**	**1700 - 0459**	**0500 - 1159**	**1200 - 1659**	**1700 - 0459**	**0500 - 1159**	**1200 - 1659**	**1700 - 0459**
NE1	6.95	2.87	2.60	2.75	2.99	4.12	-	-	-
NE3	6.38	4.86	4.96	4.73	5.17	7.32	2.41	2.27	2.27
NE4	2.00	1.94	1.99	2.06	1.99	1.96	3.4	3.67	3.38
NE5	1.72	1.7	1.83	1.83	1.84	1.96	2.93	2.99	2.87
NE6	15.03	5.15	4.35	6.04	5.86	8.30	1.89	1.85	1.83
NE7	1.69	1.63	1.75	1.63	1.76	1.83	2.42	2.37	2.31
NE8	2.01	1.76	1.97	1.83	1.78	1.89	2.28	2.23	2.19
NE9	1.64	1.37	1.44	1.54	1.44	1.53	2.32	2.28	2.2
NE10	1.66	1.49	1.41	1.61	1.44	1.45	2.31	2.29	2.21
NE11	1.63	1.43	1.41	1.58	1.34	1.48	2.28	2.23	2.18
NE12	3.28	3.15	3.96	5.92	2.94	3.00	1.81	1.65	1.68
NE13	1.81	1.53	1.59	1.63	1.63	1.64	2.41	2.09	2.14
NE14	1.91	1.58	1.50	1.57	1.65	1.72	2.6	2.22	2.32
NE15	1.86	1.5	1.46	1.49	1.55	1.65	2.71	2.25	2.42
NE16	2.72	2.00	1.88	1.94	2.05	2.33	2.79	2.56	2.63
NE17	2.81	2.22	2.18	2.12	2.51	2.57	-	-	-

Table 3.3: Scaling Table for NEL-Partial Trips

It can be seen that the scaling table intuitively make sense. For example, station NE6 (Dhoby Ghaut) is the largest interchange station on the NEL and located in the heart of the Central Business District. As

such, it has an extremely high scaling factor of 15 for boarding passengers in the morning, as the number of passengers transferring to the NEL from other lines far outweighs the number of passengers entering via the fare gates in the city. In contrast, less-busy suburban stations such as NE14 (Hougang) have much smaller scaling factors.

3.7.2 Performing Iterative Estimation

As shown in Appendix 1, NEL trains exhibit different dwell time behavior depending on the station they are dwelling at. As set by the operator, trains dwell for a minimum of 20s at regular stations, 40s at interchange stations, and 60s at terminal stations. As such, it is necessary to have unique dwell time parameters for each type of station. This results in a total of 12 parameters (3 station types, and 4 parameters per station-type) to be estimated (see Equation 3.13).

(Hollander *et al.*, 2008)[12] provide a good summary of the advantages of different objective functions. For the purposes of this estimation, the objective function was chosen to be the root-mean-squared error (RMSE). The optimization problem was then formalized as follows:

$$\min f(\boldsymbol{b})$$

$$s.t.\, b_j \geq 0 \; \forall \, j$$

$$f(\boldsymbol{b}) = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (g(\boldsymbol{b})_i - y_i)^2} \tag{3.16}$$

where **b** is the vector of dwell time parameters of length *j*, $g(\boldsymbol{b})_i$ the simulated travel time for passenger *i* under dwell time parameters **b**, and y_i the actual historical travel time for passenger *i*. Note that all elements of **b** are constrained to be larger than zero, since an increase in the number of passengers cannot possibly result in shorter dwell times. The demand was omitted in this formulation, as it is kept constant during the calibration process.

The objective function $f(\boldsymbol{b})$ was chosen for a few reasons. Firstly, using the squared error penalizes large errors more than small errors, which is a logical approach to rail modelling, where minor fluctuations around the mean are normal and indeed expected. We did not normalize against the actual travel time, as the trip length should have no bearing on the weightage of the error corresponding to that trip. Finally, the objective function intuitively makes sense—it represents the expected difference in seconds between the simulated and actual travel times.

The root-mean-squared normalized error (RMSNE) was also calculated as an additional metric:

$$\text{RMSNE} = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} \left(\frac{g_i(\boldsymbol{b}) - y_i}{y_i} \right)^2} \tag{3.17}$$

This provides a good representation of the *degree* of error; that is, the percentage by which simulated travel times differ from actual travel times. However, it is unsuitable for comparing across different days, as the normalization by y_i causes the RMSNE to fluctuate significantly depending on the distribution of actual travel times y_i used in the input data set.

3.7.3 Results

The SPSA algorithm as described in (Spall, 1998) was run multiple times with different initial seeds, and the best results (using RMSNE as the objective function) obtained as follows:

Station Type	β_0	β_1 (Boarding)	β_2 (Alighting)	β_3 (Occupying)	RMSNE/RMSE
Normal	0.09	0.96	0.98	0.13	RMSNE: 0.2936
Interchange	1.92	0.30	1.14	0.09	RMSE: 176.23s
Terminal	21.82	1.16	2.26	0.07	

Table 3.4: Estimated Dwell Time Parameters

This took 15 iterations to reach convergence, which was defined as 3 continuous iterations with no improvement in the RMSE. The estimated parameters are observed to fit simulated travel times to actual times reasonably well, with a RMSE value of 176.23 seconds, or about 18.4% of the actual average travel time of 960 seconds, while the average deviation of each simulated trip from the actual trip duration was about 29%. Approximately 57% of simulated travel times fell within +/- 15% of the actual travel time, and approximately 68% of simulated times fell within +/- 20% of the actual travel time.

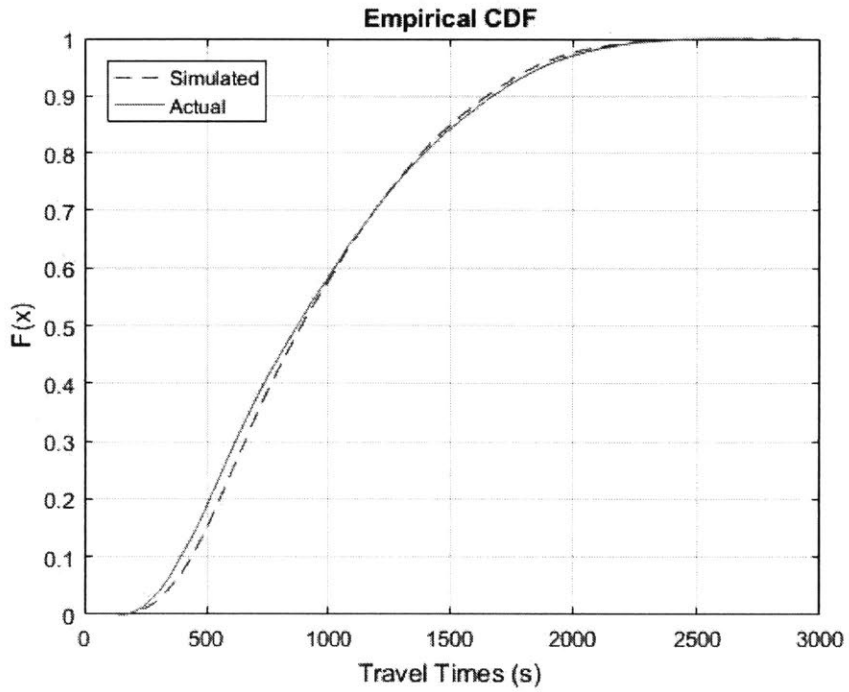

Figure 3.4: CDFs of Simulated vs. Actual Travel Times

Figure 3.4 plots the CDFs of simulated and actual travel times against each other. The curves are generally similar, although the variance of actual travel times is slightly larger than that of the simulated travel times. This is likely due to the presence of difficult-to-detect factors in actual travel times that result in erroneous data, which will be subsequently elaborated on.

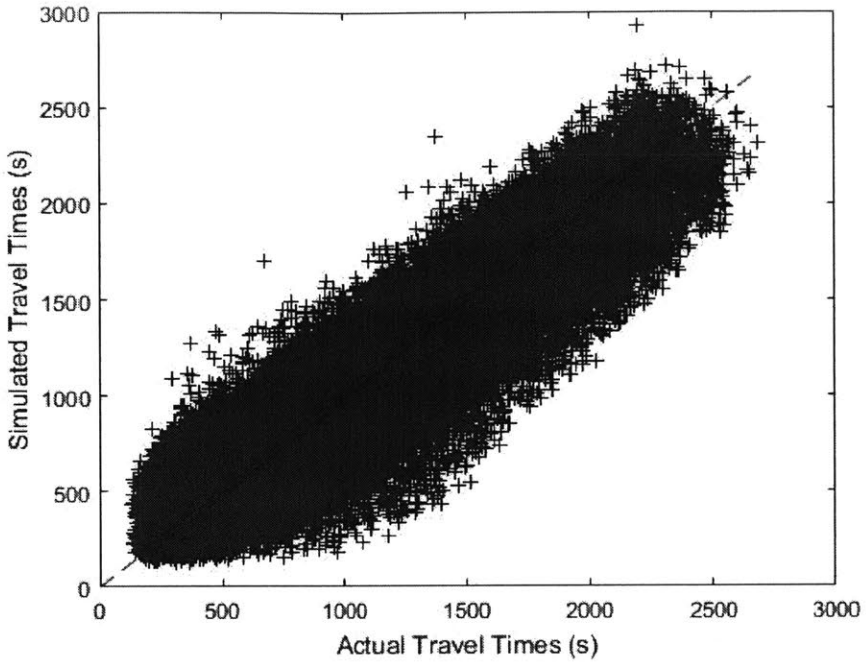

Figure 3.5: Comparing Simulated and Actual Travel Times

Figure 3.5 compares actual and simulated travel times, with the x-axis representing the former and the y-axis representing the latter. Each '+' represents a specific trip, and the red 45° line represents the ideal "perfect fit", where every simulated travel time matches actual travel time exactly. It can be seen that the estimated/actual travel time pairs generally lie along this line, demonstrating that a linear dwell time function results in a reasonable approximation for the data.

An interesting observation is that the noise to the left of the line is relatively sparse, with only sporadic outliers, while the noise to the right of the line is relatively dense and gradually tapers. This is effectively due to the simulator being unable to capture *every* source of noise that may happen in the real world. In the former region (where simulated travel times far exceed actual travel times), walking times are the only major source of variability; in the latter region (where actual travel times exceed simulated travel times), variability may occur from various sources of erratic behavior. For example, a passenger may spend time at the platform waiting for a friend, thus artificially extending his travel time—a

phenomenon only partially captured in the rail simulator, which as per the previously estimated models attributes these delays to exceptionally long walking times. Unfortunately, it is difficult to distinguish between erratic behavior and genuinely long travel times, and while more rigorous pruning of the data may result in less right-hand noise and a better fit, care must be taken to preserve legitimate trips in order to properly estimate the dwell time parameters.

CHAPTER 4: VALIDATION

In order to validate the assumptions made and work done in this thesis, we use a two-step validation process. In Section 4.1, we first validate the *sequential calibration* approach by applying the estimated parameters to a new demand data set taken from historical trips made on a different day. We show that the estimated parameters result in reasonably accurate simulated travel times, vindicating the method.

In Section 4.2, we then demonstrate the capability of the rail simulator by simulating a full-fledged historical disruption that occurred in Singapore in 2013. By showing that the simulator is able to provide in-depth information about the effects of the disruption on both supply and demand, we prove the usefulness of the simulator in providing insight into complex scenarios.

4.1 Validating Calibration Parameters

In order to validate the parameters estimated in Chapter 3, AFC data from a second arbitrarily chosen weekday (Friday, 9 Aug 2013) was used. This data set was pruned and scaled via the same method described in Section 3.7.1, resulting in approximately 170,000 plausible trips which were then fed as demand into the rail simulator. Heatmaps were used to compare the distribution of demand in the calibration day against the validation day, as shown:

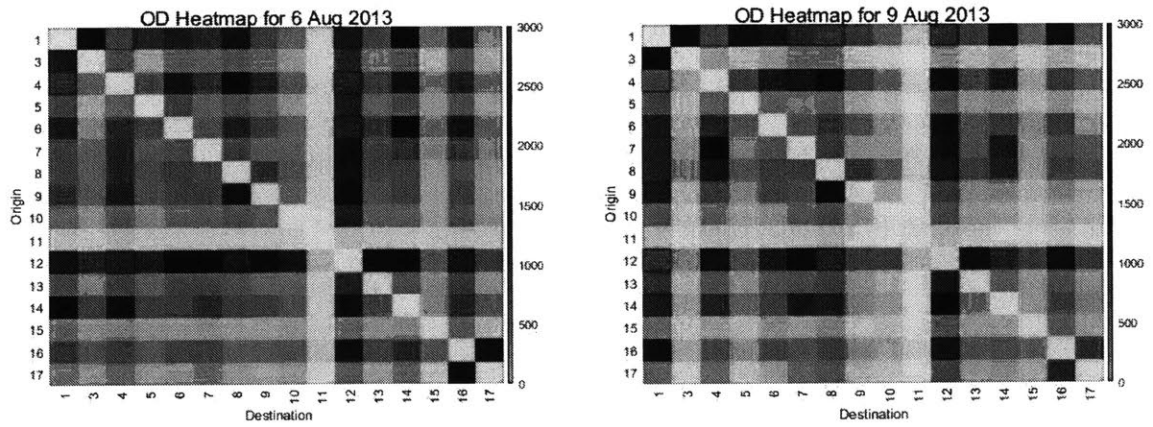

Figure 4.1: Heatmaps of Calibration Demand vs. Validation Demand

The numbers on the axes correspond to the station numbers (for example, "1" refers to station NE1). The distribution of origin-destination pairs is largely similar across both days, with major interchange stations such as NE12 (Serangoon) experiencing high commuter flow. We also note that NE1 (Harbourfront) has a significantly higher passenger flow on the validation day than the calibration day, likely due to its popularity as a recreational location.

The parameters shown in Table 3.8 resulted in a RMSE of 204.21 seconds, or 19.8% of the actual average travel time of 1034 seconds. (Note that the difference between these results and the results obtained earlier exceed the standard deviation of the RMSE, which was roughly 0.3 seconds). The RMSNE value was 0.2679, a seeming *improvement* over the estimated case; however, this was due to the longer average travel times in the validation case, which was likely a result of the flow of passengers to NE1 and back to the suburbs mentioned earlier. Approximately 52% of simulated travel times fell within +/- 15% of the actual travel time, and approximately 65% of simulated times fell within +/- 20% of the actual travel time. The parameters estimated in Table 3.8 are therefore shown to constitute a reasonably accurate model of travel time.

Figure 4.2 plots the CDFs of simulated and actual travel times against each other. We observe that simulated travel times tend to be shorter than the actual travel times, once again due to the validation day having longer travel times overall than the validation day. This problem can likely be rectified via the use of a larger sample size across multiple days for estimation, by performing separate estimations for weekdays and weekends (with Friday considered a weekend), etc.

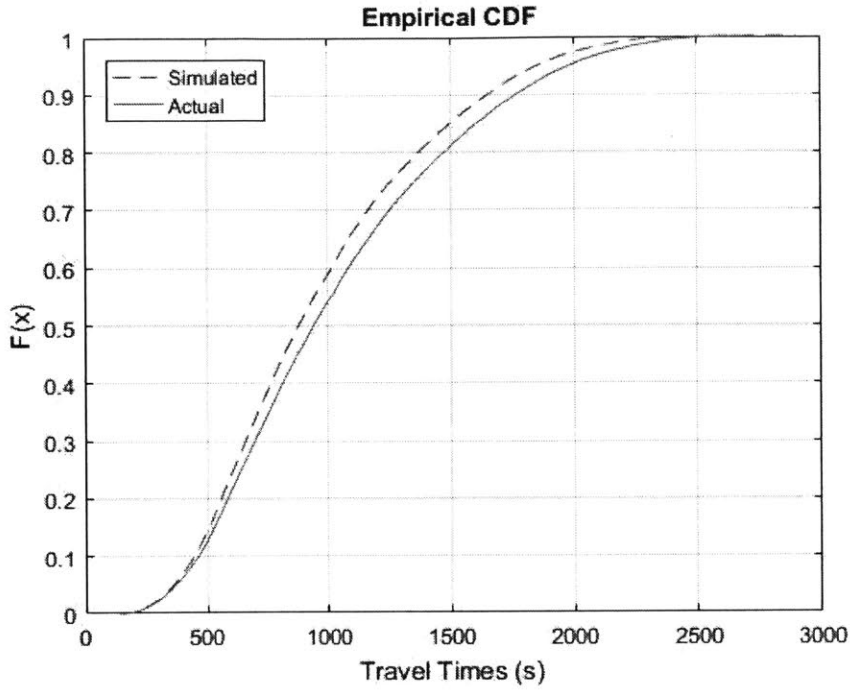

Figure 4.2 Validating CDFs of Simulated vs. Actual Travel Times

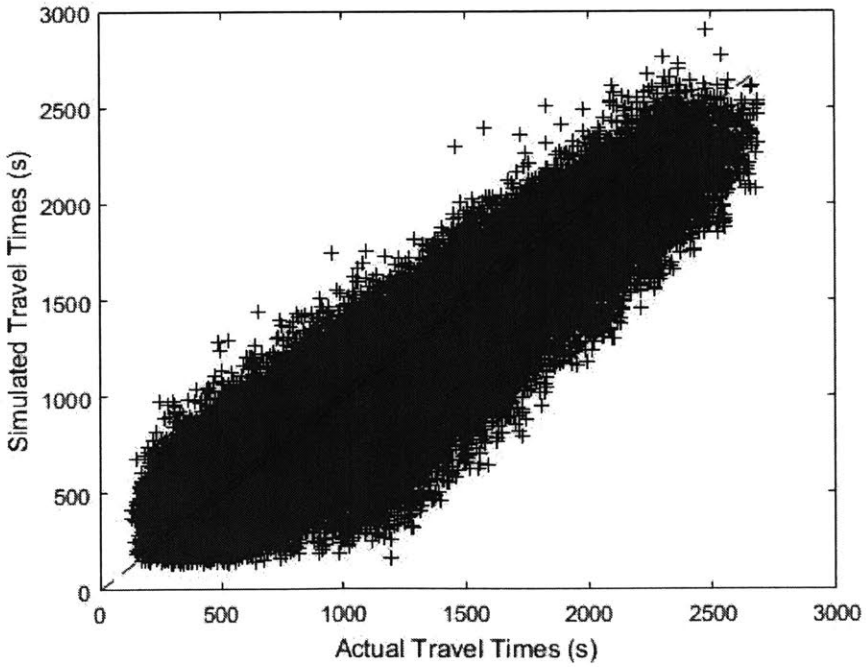

Figure 4.3: Validating Simulated and Actual Travel Times

Figure 4.3 is very similar to Figure 3.5, with the estimated/actual dwell time pairs generally lying along the line of perfect fit. We therefore conclude that the approach detailed in Chapter 3 is a valid method of estimating parameters, especially when the goal is to simply obtain appropriate initial seed values for a secondary simultaneous calibration (or in other cases where computational speed is prioritized over a high degree of accuracy). Of course, improvements can naturally be made at the cost of additional data or additional computing power to improve the accuracy of this calibration process. This will be elaborated further in Section 5.2, where we recommend potential improvements that can be made.

4.2 Disruption Scenarios

4.2.1 Introduction

In order to validate the capabilities of the rail simulator in this thesis, we perform a simple case study based on a real-life disruption that happened in Singapore. On June 19, 2013, a train on line NE_1 stalled as it approached Hougang Station (NE14), thus creating a disruption in service that lasted from 1815 hours to 2035 hours and affected stations from Woodleigh (NE11) to Punggol (NE17) inclusive. Although this disruption occurred after the PM Peak (resulting, fortunately, in minimal impact on commuters), it raised concerns about the possible effects of disruptions of similar magnitude, should they occur during peak hours. As such, in this section we aim to replicate the disruption during the AM Peak hours of 0800 to 1000, as well as explore the potential effects of simple mitigating strategies.

We aim to simulate 3 different scenarios. The first, *no disruption*, creates a baseline to compare the effects of the disruption against. The second, *simple disruption*, simulates the disruption without any mitigating strategy, demonstrating the repercussions of the event. The third, *mitigated disruption*, simulates the disruption with a simple mitigating strategy of introducing shuttle buses that stop at every station between NE11 and NE17, effectively replacing the trains.

To evaluate the impact of each scenario, we will record *average travel time* for trips that utilized the disrupted stations, *average dwell time*, *average waiting time,* and *number of denied boardings* at bus

stops near the disrupted MRT stations, and *mode share*; that is, the distribution of alternative transport choices that commuters turn to during the disruption.

Note that these metrics are multimodal; they analyze the effects of the disruption not only on passengers' travel times, but also on other modes of transport such as taxis, buses, and so on. This demonstrates the advantages of utilizing an integrated rail simulator—a larger scope of analysis and a clearer understanding of the situation.

To simulate the disruption, the original (unmodified, integrated) rail simulator was used. On the supply-side, the LUA script in Appendix 2 was used to set-up the disruption scenario for rail (holding trains at stations, u-turning trains where necessary, forcing passengers to alight, etc.), as described in Section 2.6. In addition to this, routes corresponding to the shuttle buses were introduced in the integrated bus simulator and activated in the *mitigated disruption* scenario. On the demand-side, the *pre-day* and *within-day* modules in the SimMobility Mid-Term were used to respectively generate demand parameters (time-of-day choice, route choice, mode choice, etc.) and modify these parameters in reaction to the disruption.

4.2.2 Results

As expected, the results correspond to intuition, with the *no disruption* scenario exhibiting the best results, followed by the *mitigated disruption* scenario, and finally the *simple disruption* scenario. These results are summarized in Table 4.1:

Impact Metrics	*No Disruption*	*Simple Disruption*	*Mitigated Disruption*
Average Travel Time (min)	52.8	111.6	74.8
Average Dwell Time (s)	45	58	52
Average Waiting Time (min)	4.69	8.4	5.8
# of Denied Boardings	867	13960	5539

Table 4.1: Summary of Disruption Scenario Results

There are a few interesting points of note when comparing the disruption scenarios against the base case. Firstly, the occurrence of the *simple disruption* more than doubles the average commuter time, a devastating result during the AM peak. At the same time, the simple mitigating strategy proposed in this thesis manages to alleviate the effect of the disruption significantly, demonstrating that the presence of *any* mitigating strategy, even unrefined, is better than having none at all.

The average dwell time, average waiting time, and denied boarding metrics demonstrate the advantages of having an integrated simulator, by capturing the effects of the rail disruption on the road network. In the *simple disruption* case, the dwell times of buses and the waiting times of commuters at bus stops increase significantly, as commuters rush to find alternative transport to work; however, the large amount of denied boardings demonstrate that many of these efforts are in vain, as the existing buses are simply unable to accommodate this surge in demand. On the other hand, in the *mitigated disruption* case, the number of denied boardings is significantly lower, demonstrating how the injection of the shuttle buses into the network help to somewhat alleviate this demand. At the same time, however, the dwell times and waiting times are still noticeably higher than in the *no disruption* scenario; furthermore, there is an additional downside of potential increased congestion on the road network—an area that could be further investigated via the integrated road simulator.

Table 4.2 summarizes the distribution of alternative modes that train commuters chose during the disruption:

	Simple Disruption	*Mitigated Disruption*
Walk	6%	6%
Taxi	19%	6%
Regular Bus	75%	34%
Shuttle Bus (Mitigation)	N/A	54%

Table 4.2: Mode Share in Disruption Scenarios

In the *simple disruption* scenario, a large number of commuters turned to the bus service, which was unable to cope with this surge in demand, resulting in the significant number of denied boardings shown in Table 4.1 It can be seen once again that the presence of the mitigating shuttle buses helped to alleviate the demand on the regular bus service, thus resulting in the reduced number of denied boardings (and, at the same time, reducing the demand for taxis as well).

In conclusion, this case study confirms what we intuitively knew: that the presence of *any* mitigating strategy, even if unoptimized, is a huge step towards handling major disruptions. More importantly, however, it demonstrates the capabilities of an integrated simulator, by capturing the effects of the rail disruption on the road network, on other modes of transport, and on commuters' choices. Only through such an all-encompassing analysis of the effects of disruptions and proposed mitigation strategies can a true optimal solution—one that minimizes the repercussions of the disruption across the transportation network *in its entirety*—be carefully devised.

Chapter 5: Summary and Future Work

5.1 Summary

In this thesis, we developed a comprehensive rail simulator integrated with the SimMobility simulation platform. There were three primary goals:

1) To develop an *integrated* rail simulator that works in conjunction with other simulators to capture not only supply and demand interactions within the rail network, but also how these interactions affect other modes of transport, and how a commuter's experience in one day might affect his decisions in the next;
2) To design a computationally fast method of calibration that relies only on AFC data to yield reasonably accurate estimates of simulation parameters; and
3) To showcase the features of the rail simulator through disruption scenario results obtained from the SimMobility Mid-Term.

Chapter 2 demonstrated that the first goal was met. We detailed how the rail simulator was integrated to the larger framework of the SimMobility Mid-Term, and explained how the design and structure of the rail simulator allowed it to constantly maintain a feedback loop with the Mid-Term, ensuring an accurate representation of passenger behavior. We also introduced the Service Controller, a powerful entity that allows the user to construct virtually any scenario in the rail simulator, and demonstrated its capabilities by designing and implementing a historical disruption scenario via a LUA script.

Chapters 3 and 4 demonstrated that the second goal was met. In Chapter 3, we introduced a novel approach to calibration, sequential calibration, which used certain assumptions to draw conclusions from AFC data, streamlining the estimation process while still maintaining reasonable accuracy. We demonstrated that this approach of estimating parameters resulted an error of approximately 176 seconds per trip (18.4% of the average trip duration), a reasonably accurate value.

In Chapter 4, we validated the method of sequential calibration by using the parameters estimated in Chapter 3 to estimate travel times for a new data set. This resulted in an error of approximately 204 seconds per trip (19.8% of the average trip duration), thus demonstrating that the values obtained through sequential calibration were a valid estimation of the actual parameters, and hence proving that sequential calibration is an appropriate "first step" approach to generate initial seed values for a more accurate, secondary simultaneous calibration of parameters for a specific scenario. We then used the calibrated rail simulator to run multiple disruption scenarios, and demonstrated how the integrated rail simulator was able to provide a wealth of data not only regarding the impact on the rail network, but also the road network, associated bus routes, passenger welfare, and so on—data that could then be utilized to develop a system-optimal mitigation strategy.

5.2 Recommendations for Future Work

The rail simulator developed and calibrated in this thesis has proven to be a useful tool in transportation modelling, capable of providing new insights into passenger behavior and how events may cause effects that cascade through the entire transportation network, affecting multiple modes of transport simultaneously. Nevertheless, there is still much room for improvement in improving both the *scope* and *accuracy* of the rail simulator and the methods detailed in this thesis. We suggest the following areas as potential subjects of future work:

5.2.1 Expanding the Scope

In this thesis, the Singapore NEL was used as a proof-of-concept of the rail simulator. A natural next step is to expand the scope of the simulator to the entire Singapore rail network, and subsequently, rail networks in other cities. This poses an interesting challenge in supply modelling due to the presence of *shared track*, that is, when multiple rail lines share one track. To model this, the Service Controller logic should be updated to consider the position of *all* trains in the system (not just within a line), and keep appropriate distances between trains competing for the same track.

A second challenge arises when modelling rail networks with manually driven trains, especially in cases where the train driver is given significant autonomy. For example, the Singapore North-South Line (NSL) employs train drivers; however, speeds are still automatically regulated and the train driver is simply there as a fail-safe. In contrast, the Boston Green Line gives train drivers considerable autonomy in choosing speeds and dwell times. It may therefore be necessary to introduce an additional layer of complexity in the form of *train-driver-Agents*, each with their own preferred speeds, patience (in waiting for passengers to board, thus affecting dwell time), and so on. This is an interesting area that should be further explored.

The issue of *depots* should also be more carefully modelled in the rail simulator. In this thesis, depots are represented as active/inactive pools, and are assumed to lie at the ends of the line. However, in real life depots may be located along the line; as such, difficulties may arise in moving trains in and out of service due to out-of-service trains competing with in-service trains for track. This issue should once again be solved by updating the Service Controller logic.

An ambitious long-term goal would be to expand the rail simulator to model *light surface rail*. These systems, such as the MBTA Green Line, present an additional dimension of difficulty as train-Agents interact not only with passenger-Agents, but also with road conditions (cars, buses, pedestrians, etc.) and their own drivers (the train-driver-Agents mentioned above). The models used in the Service Controller will therefore need to be significantly improved to capture the increased degree of autonomy present in such situations.

5.2.2 Improving Accuracy in Estimation

In Chapter 3, a considerable number of assumptions were used to improve the efficiency of the sequential calibration process. Although this resulted in a reasonable fit, some of these assumptions could be removed to obtain more accurate estimation parameters. For example, a natural improvement would be to use a non-linear dwell time function to replicate the increasing marginal delay that each

additional passenger has on dwell time. This approach was used by (Puong, 2000), which fit a *cubic* function to the data, but was not used in this thesis due to the additional computational demands it required.

Another improvement would be to differentiate between walking times *to* and *from* the platform. Although the gate-to-platform and platform-to-gate walking times recorded at Woodleigh were relatively similar, this phenomenon may not hold for larger or busier stations; indeed, one might expect the former (where commuters arrive relatively constantly) to exhibit less variance than the latter (where commuters arrive in "bursts" when a train arrives).

A weakness in sequential calibration is its assumption of *constant* (and hence zero-variance) inter-station travel times. As such, the variance that exists in reality is effectively "shunted" to the walking time parameters, resulting in a slight overestimation of commuter walking times. If data on inter-station travel times of trains were available, it would be possible to determine the corresponding *variance* and account for it in the sequential calibration, improving the overall accuracy of the estimated parameters.

It should also be noted that in this thesis, the NEL was calibrated in isolation out of necessity, as the other lines in the Singapore rail network were not yet developed, with scaling factors used to approximate passenger flow from (or to) non-NEL stations. The sequential calibration process should therefore be revisited once the entire Singapore rail network is modelled. By estimating dwell time parameters at *all* train stations within the network, we will be better able to capture the interactions between different train lines. For example, a particularly full train on the NEL may cause severe boarding problems on the NSL at a NEL-NSL interchange station, thus resulting in prolonged dwell times. This will lead to a more accurate estimation of dwell time parameters.

Finally, in the sequential calibration approach, the NEL was calibrated using AFC data exclusively; however, over the course of this research, preliminary attempts were made to obtain ATC data via "data scraping" real-time train arrival information from the operator's website. Although this approach

ultimately proved unreliable and was abandoned, it would be interesting to explore alternative methods of obtaining data, and how the presence of this data would affect the sequential calibration process. For example, it is conceivable that, if the "data scraping" experiment had been successful, the pseudo-ATC data could have been integrated with the AFC data to improve the accuracy of the estimation.

5.3.3 Future Case Studies

In this thesis, we conducted a simple case study to demonstrate how the rail simulator was able to provide the data necessary for an all-encompassing analysis on the impact of a rail service disruption on the overall transportation network. A logical next step is therefore to perform this analysis. For example, in Chapter 4.2, shuttle buses were introduced into the system as a potential mitigating strategy. However, while these buses helped reduce the impact of the train disruption, they are also likely to introduce additional problems, such as congestion along the road network. As such, an investigation and analysis of potential mitigating strategies, taking into account the *overall* impact on the transportation network, would be an interesting area of future work that further demonstrates the importance of the integrated SimMobility Mid-Term simulator.

Of course, given the considerable capabilities of the SimMobility Mid-Term, more ambitious projects should be attempted as well. One such project would be utilizing the integration between different supply simulators to enhance coordination between modes—for example, *holding* buses at bus bays near train stations, if a train has just arrived at that station, in order to serve the expected surge in demand—and investigate the benefits of this coordination on passenger welfare.

Indeed, *passenger welfare* is an interesting topic that merits further exploration. An interesting area of study would be to utilize the integrated simulator to attempt to *quantify* passenger welfare—for example, the negative externalities that an additional rail passenger brings (due to increased crowding, longer dwell times, and potential for denied boarding)—and impose *taxation* that shifts passenger

demand between different modes of transport, in order to obtain an optimal social equilibrium. While ambitious, this concept serves to highlight the myriad possibilities that an *integrated* simulator offers.

Finally, the ultimate goal is to utilize the entire integrated SimMobility suite of simulators. One potential application is in long-term planning of rail lines, where the SimMobility Long-Term can be used to explore the effects of rail lines on residential choices and long-term demand, and the Mid-Term used to investigate the *ability* of the proposed rail lines to serve this demand. This is an exciting area of future research that promises to showcase the practical benefits of the SimMobility suite in transportation.

Appendices

Appendix 1: List of Implemented APIs

The following table provides a list of APIs currently implemented in the rail simulator. There are 3 main categories of APIs. The first, *Command APIs*, are used to instruct the simulation to perform an extraordinary action, via the service controller. The second, *Informational APIs*, report on the status of various elements within the simulation, and are primarily used as inputs to Command APIs. The third, *Disruption-specific APIs*, are APIs that were developed specifically to help simulate a test-case "disruption scenario" (see Appendix 2) on the NEL. These APIs are unlikely to be of much use outside this "disruption scenario", but are included for the sake of completion.

#	API Name (*and notes*)	Input	Result
	Command APIs		
C1	terminate_trainservice	Line ID: NE_1	No new train dispatch after API called. All trains stop at next station, regardless of platform list. All trains empty of passengers after stopping.
C2	terminate_single_train_service	Train ID: 4 Line ID: NE_1	Train 4 stops at the next station regardless of platform list. Train 4 force-alights all passengers, then returns to the depot.
C3	reset_safe_operation_distance	Distance:1979 Train ID: 16 Line ID: NE_2	Train 4 keeps a minimum distance of 1979m away from the train in front of it.
C4	reset_safe_headway_sec	Headway: 800 Train ID: 2 Line ID: NE_1	Train 4 keeps a minimum headway of 800s behind the train in front of it.
C5	force_release_passengers	Train ID: 4 Line ID: NE_1 Boolean: True	When Train 4 next reaches a platform, it forcibly alights all its passengers.
C6	restrict_passengers 0: Boarding 1: Alighting 2: Both *Check PersonsBoarding*	Platform ID: NE6_1 Train ID: 4 Line ID: NE_1 Type: 1	Passengers are restricted from boarding, alighting, or both.

C7	update_platform_list *Used together with C8*	Train ID: 4 Platforms: {NE4_1, NE6_1} Line ID: NE_1	The train does not stop at NE6_1 and NE4_1. Speed and acceleration is calculated simply assuming NE6_1 and NE4_1 are normal stretches of track. If all platforms are removed, the train proceeds along the track back to the depot, despite having passengers on board.
C8	add_platform *Used together with C7*	Train ID: 4 Platforms: NE6_1 *(Note: After update_platform_list has already been run)* LINE ID: NE_1	The train now stops at NE6_1 again. Speed and acceleration is calculated assuming NE6_1 is a platform.
C9	reset_holding_time_at_station *If C9, C10, and C11 contradict, C9 has priority over C10 has priority over C11.*	Platform ID: NE6_1 Duration: 100 Train ID: 3 LINE ID: NE_1	The next time Train 3 reaches NE6_1, it holds for exactly 100s regardless of demand. This API **does not** affect how many passengers manage to board or alight from the train.
C10	reset_max_holding_time_at_station *If C9, C10, and C11 contradict, C9 has priority over C10 has priority over C11.*	Platform ID: NE6_1 Duration: 100 Train ID: 3 LINE ID: NE_1	The next time Train 3 reaches NE6_1, it holds for at most 100s regardless of demand. This API **does not** affect how many passengers manage to board or alight from the train.
C11	reset_min_holding_time_at_station *If C9, C10, and C11 contradict, C9 has priority over C10 has priority over C11.*	Platform ID: NE6_1 Duration: 100 Train ID: 3 LINE ID: NE_1	The next time Train 3 reaches NE6_1, it holds for at least 100s regardless of demand. This API **does not** affect how many passengers manage to board or alight from the train.
C12	insert_stop_point	Train ID: 2 LineID: NE_1 Distance: 3978.38 Duration: 50 Maximum Deceleration Rate: 1.2	When Train 2 approaches point 3978.38 on NE_1, it **slows down** and stops. If Train 2 has to decelerate at *more* than 1.2 ms^{-2} to stop, it ignores the command entirely and returns an error message. The set maximum deceleration rate cannot be lower than the system default.
C13	insert_unscheduled_train_trip	Line ID: NE_1 Start Time: 08:00:00 Start Station: NE6_1	Starting from 08:00:00, as soon as the safe headway and safe distance conditions are satisfied, a train is inserted into line NE_1, beginning service at NE6_1.

C14	set_uturn **Use in conjunction with I11 to ensure that the next station is U-turn compatible.*	Train ID: 4 Line ID: NE_1 Boolean: True Delay: 30s	The next time Train 4 reaches a station, it force alights all its passengers. When headway and distance conditions next allow it, Train 4 is teleported to the corresponding platform on the opposite line. 30 seconds after the teleportation, Train 4 allows passengers to board, then continues service in the opposite direction, using the default platform list for the new line. Minimum U-turn time is 5 seconds.
C15	set_ignore_safe_distance	Train ID: 4 Line ID: NE_1 Boolean: 1	Train 4 ignores the safe following distance when calculating its speed and acceleration. This does **not** change the actual safe following distance value.
C16	set_ignore_safe_headway	Train ID: 4 Line ID: NE_1 Boolean: 1	Train 4 ignores the safe following headway when calculating its speed and acceleration. This does **not** change the actual safe following headway value.
C17	set_force_alight_status *Refer to I9*	Train ID: 4 Line ID: NE_1 Status: 0 (= FALSE)	Sets the "forceAlighted" variable for Train 4 to FALSE.
C18	reset_moving_case	Train ID: 4 Line ID: NE_1 Case Value: 1 (Case Value = 2 only works if the next stopping point corresponds to a station, and is only used to "undo" CV = 1).	Train 4 is forced to enter the "speed adjustment case" (see Appendix 2). If the next stopping point corresponds to a station, it stops "jerkily" at the station.
C19	reset_speed_limit	Speed Limit: 50 Start Platform: NE3_1 End Platform: NE4_1 Line ID: NE_1 Start Time: 09:00:00 End Time: 10:00:00	Between 09:00:00 and 10:00:00, all trains passing between NE3_1 and NE4_1 on line NE_1 obey a system speed limit of 50 km/h.
C20	reset_acceleration	Acceleration Limit: 0.8 Start Platform: NE3_1 End Platform: NE4_1 Line ID: NE_1 Start Time: 09:00:00	Between 09:00:00 and 10:00:00, all trains passing between NE3_1 and NE4_1 on line NE_1 obey a system acceleration limit of 0.8 ms^{-2}

		End Time: 10:00:00	
Informational APIs			
I1	get_dwelltime	Train ID: 4 Line ID: NE_1 Platform ID: NE6_1	The **remaining** dwell time of Train 4 at Platform NE6_1 is returned. If Train 4 is not at NE6_1, the API returns -1.
I2	get_opposite_lineid	Line ID: NE_1	The API returns the opposite Line ID, i.e. NE_2.
I3	get_next_platform	Train ID: 4 Line ID: NE_1	If the train is between platforms, it returns the next platform the train will stop at. If the train is at a platform, it returns the current platform.
I4	get_distance_to_next_platform	Line ID: NE_1 Train ID: 4	If the train is between platforms, it returns the distance to the next platform the train will stop at. If the train is at a platform, it returns a distance of 0.
I5	get_active_trains_size	Line ID: NE_1	Returns the current number of active trains on NE_1.
I6	get_active_train_by_index	Line ID: NE_1	Returns the IDs of all trains currently active on a line, in no particular order.
I7	get_next_requested	Train ID: 4 Line ID: NE_1	It returns the next requested action for Train 4 (i.e. the action the train is supposed to take in the next frame tick): The next requested can be 1.no_requested 2.requested_at_platform 3.requested_waiting_leaving 4.requested_leaving_platform 5.requested_take_uturn 6.requested_to_platfrom 7.requested_to_depot
I8	get_trainid_train_ahead	Train ID: 4 Line ID: NE_1	Returns the ID of the train ahead of Train 4. If there is no train ahead of it, it returns a value of -1.
I9	get_force_alight_status *Refer to C17* *Refer to C2*	Train ID: 4 Line ID: NE_1	After a train force-alights all its passengers, it sets the variable "forceAlighted" to TRUE. A train cannot force-alight its passengers until this variable is reset to FALSE.

I10	get_platform_by_offset	Train ID: 4 Line ID: NE_1 Offset: 3	If Train 4 is currently approaching platform NE_3, this API returns NE_6.
I11	get_next_uturn_platform	Train ID: 4 Line ID: NE_1	If Train 4 is currently approaching NE2_1, the API returns NE3_1 (the next acceptable U-turn platform).
Disruption-specific APIs			
D1	perform_disruption	Start Station: NE_3 End Station: NE_7 Time: 08:00:00	At 08:00:00, a disruption occurs between stations NE_3 and NE_7 *in both directions*. **In the disruption region:** Trains move to the next platform, force-alight their passengers, and wait. If the next platform is occupied, trains move as close as possible to the platform, ignoring safe distances and headways, then force-alight their passengers and wait. **Out of disruption region:** Trains force-alight passengers and U-turn at the platform right before the disruption region.
D2	set_disrupted_platforms	Start Station: NE_3 End Station: NE_7 Line ID: NE_1	Platforms NE_3 through NE_7 on line NE_1 are considered disrupted. Trains cannot pass through this stretch of track.
D3	clear_disruption *Performed after D2*	Line ID: NE_1	Platforms NE_3 through NE_7 on line NE_1 are no longer disrupted.
D4	get_disrupted_platforms	Line ID: NE_1 Index: Why is this required?	The API lists NE_3, NE_4, NE_5, NE_6 and NE_7 as disrupted platforms.
D5	get_disruptedplatforms_size *Performed after D2*	Line ID: NE_1	The API returns 5 disrupted platforms.
D6	is_stranded_during_disruption	Train ID: 3 Line ID: NE_1	If Train 3 is unable to make it to a platform during a disruption, this API returns TRUE.

D7	get_disrupted_state	Train ID: 3 Line ID: NE_1	If Train 3 is waiting at a platform (or right before it) in a disrupted region, this API returns TRUE.
D8	set_subsequent_next_requested *Refer to I7*	Train ID: 4 Line ID: NE_1 Next Req: 7	This only works if Train 4 is currently in requested_waiting_leaving state. After Train 4 exits requested_waiting_leaving state, it enters requested_to_depot state.

Appendix 2: LUA code for Disruption

```
function use_servicecontroller(params,time)

if time =="10:00:00" then --this clears the disruption at 10AM
--OBJECTIVES:
--unset disrupted platforms
--unset Uturn flag
--unset force alight flag
--unset force alight status
--unset ignore safe distance and safe headway
 params:clear_disruption("NE_1")
 params:clear_disruption("NE_2")

 local size=params:get_active_trains_size("NE_1") --Performing the reset for line NE_1
 if size>0 then

       local i=0
       for i = 0,size-1 do
          local trainId=params:get_active_train_by_index(i,"NE_1")
          params:set_uturn(trainId,"NE_1",false,0)
          params:force_release_passengers(trainId,"NE_1",false)
          params:set_force_alight_status(trainId,"NE_1",false)
          params:set_ignore_safe_distance(trainId,"NE_1",false)
                  params:set_ignore_safe_headway(trainId,"NE_1",false)
          if params:is_stranded_during_disruption(trainId,"NE_1") == true then
            params:set_subsequent_next_requested(trainId,"NE_1",5)
          elseif params:get_disrupted_state(trainId,"NE_1") == true then
            params:set_subsequent_next_requested(trainId,"NE_1",1)
          end
       end
 end

size=params:get_active_trains_size("NE_2") --Performing the reset for line NE_2
if size>0 then

       local i=0
       for i = 0,size-1 do
          local trainId=params:get_active_train_by_index(i,"NE_2")
          params:set_uturn(trainId,"NE_2",false,0)
          params:force_release_passengers(trainId,"NE_2",false)
          params:set_force_alight_status(trainId,"NE_2",false)
          params:set_ignore_safe_distance(trainId,"NE_2",false)
                  params:set_ignore_safe_headway(trainId,"NE_2",false)
              if params:is_stranded_during_disruption(trainId,"NE_2") == true then
            params:set_subsequent_next_requested(trainId,"NE_2",5)
          elseif params:get_disrupted_state(trainId,"NE_2") == true then
            params:set_subsequent_next_requested(trainId,"NE_2",1)
          end
       end
 end

```

```
50
51  --Performing the disruption between 8AM and 10AM
52
53  elseif time  >= "08:00:00" and time<"10:00:00" then
54      local trainIds = {}
55      if time == "08:00:00" then
56          params:set_disrupted_platforms("NE11","NE17/PTC","NE_1")
57          params:set_disrupted_platforms("NE17/PTC","NE11","NE_2")
58      end
59      local disruptedPlatformsSize_firstLine=params:get_disrupted_platforms_size("NE_1")
60      local disruptedPlatformsSize_secLine=params:get_disrupted_platforms_size("NE_2")
61      local size=params:get_active_trains_size("NE_1")
62      if size>0 then
63
64          local i=0
65          for i = 0,size-1 do
66             local trainId=params:get_active_train_by_index(i,"NE_1")
67
68              local comingPlatform=params:get_next_platform(trainId,"NE_1")
69              local nextPlatform=params:get_platform_by_offset(trainId,"NE_1",1)
70
71              for j=0,disruptedPlatformsSize_firstLine-1 do
72               local tobreak = false
73                  if comingPlatform == params:get_disrupted_platforms("NE_1",j) then
74                  tobreak =true
75                  local forcereleasestatus=params:get_force_alight_status(trainId,"NE_1")
76                  if forcereleasestatus==false then--Need to reset the force alight status
77                      params:force_release_passengers(trainId,"NE_1",true)
78                  end
79
80              -- do something ignore safe distane bring trains as close if stranded between platform
81                              if params:get_next_requested(trainId,"NE_1") ==5 then
82
83                               local nextTrainId=params:get_trainid_train_ahead(trainId,"NE_1")
84
85                 if nextTrainId~=-1 then
86                  local nextRequested=params:get_next_requested(nextTrainId,"NE_1")
87                  if nextRequested==1 or nextRequested==2 then
88                     if params:get_next_platform(nextTrainId,"NE_1") == comingPlatform then
89                      --ignore safe distance and safe headway
90                        params:set_ignore_safe_distance(trainId,"NE_1",true)
91                        params:set_ignore_safe_headway(trainId,"NE_1",true)
92                     end
93                  end
94                 end
95
96
97              end
98
99              elseif nextPlatform == params:get_disrupted_platforms("NE_1",j) then
100                 tobreak =true
```

```
                tobreak =true
                   if params:get_force_alight_status(trainId,"NE_1")==false then
                     params:force_release_passengers(trainId,"NE_1",true)
                   end
                   params:set_uturn(trainId,"NE_1",true,0)
              -- do something take Uturn

            end
            if tobreak == true then
               break
            end

         end


         -- trainIds[i]=trainId

end
end
   size=params:get_active_trains_size("NE_2")
   if size>0 then

        local i=0
        for i = 0,size-1 do
           local trainId=params:get_active_train_by_index(i,"NE_2")

            local comingPlatform=params:get_next_platform(trainId,"NE_2")
            local nextPlatform=params:get_platform_by_offset(trainId,"NE_2",1)


            for j=0,disruptedPlatformsSize_secLine-1 do
             local tobreak = false
                 if comingPlatform == params:get_disrupted_platforms("NE_2",j) then
                 tobreak = true
                 local forcereleasestatus=params:get_force_alight_status(trainId,"NE_2")
                 if forcereleasestatus==false then--Need to reset the force alight status
                     params:force_release_passengers(trainId,"NE_2",true)
                 end

            -- do something ignore safe distane bring trains as close if stranded between platform
                          if params:get_next_requested(trainId,"NE_2") ==5 then

                           local nextTrainId=params:get_trainid_train_ahead(trainId,"NE_2")

               if nextTrainId~=-1 then
                local nextRequested=params:get_next_requested(nextTrainId,"NE_2")
                if nextRequested==1 or nextRequested==2 then
                   if params:get_next_platform(nextTrainId,"NE_2") == comingPlatform then
                    --ignore safe distance and safe headway
                     params:set_ignore_safe_distance(trainId,"NE_2",true)
                     params:set_ignore_safe_headway(trainId,"NE_2",true)
```

```
                    params:set_ignore_safe_headway(trainId,"NE_2",true)
                end
             end
            end


         end

         elseif nextPlatform == params:get_disrupted_platforms("NE_2",j) then
                tobreak = true
                if params:get_force_alight_status(trainId,"NE_2")==false then
                  params:force_release_passengers(trainId,"NE_2",true)
                end
                params:set_uturn(trainId,"NE_2",true,0)
           -- do something take Uturn

         end
         if tobreak == true then
            break
         end

      end


      -- trainIds[i]=trainId

end
end

end
```

Appendix 3: Singapore North-East Line Operating Specifics

Singapore North-East (Purple) Line – Operated by SBS	
Number of In-Service Trains	28
Passenger Capacity per Train	1600
Train Length	138m
Acceleration, Deceleration, Max System Speed	Acceleration: $1.1ms^{-2}$ Deceleration: $1.0ms^{-2}$ Max System Speed: 80 km/h
Safe Operating Distance/Headway	90s safe headway 50m safe distance
Dwell Time Restrictions	Minimum 20s at normal stations, 40s at interchange stations, and 60s at terminal stations. Maximum 120s at all stations.

Bibliography

[1] Adnan, Muhammad, et al. "SimMobility: A Multi-scale Integrated Agent-based Simulation Platform." *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record*. 2016.

[2] Antoniou, Constantinos, et al. "W–SPSA in practice: Approximation of weight matrices and calibration of traffic simulation models." *Transportation Research Procedia* 7 (2015): 233-253.

[3] Carrion, Carlos, and David Levinson. "Value of travel time reliability: A review of current evidence." *Transportation research part A: policy and practice* 46.4 (2012): 720-741.

[4] Cha, Moo Hyun, and Duhwan Mun. "Discrete event simulation of Maglev transport considering traffic waves." *Journal of Computational Design and Engineering* 1.4 (2014): 233-242.

[5] Chang, C. S., et al. "Development of train movement simulator for analysis and optimisation of railway signaling systems." (1998): 243-248.

[6] Darmanin, Tim, Calvin Lim, and H. Gan. "Public railway disruption recovery planning: a new recovery strategy for metro train Melbourne." *Proceedings of the 11th Asia pacific industrial engineering and management systems conference*. Vol. 7. 2010.

[7] Di, P., Key Transport Statistics of World Cities. *JOURNEYS, September, 2013*, pp. 105—112.

[8] Goodman, C. J., L. K. Siu, and T. K. Ho. "A review of simulation models for railway systems." *Developments in Mass Transit Systems, 1998. International Conference on (Conf. Publ. No. 453)*. IET, 1998.

[9] Grube, Pablo, Felipe Núñez, and Aldo Cipriano. "An event-driven simulator for multi-line metro systems and its application to Santiago de Chile metropolitan rail network." *Simulation Modelling Practice and Theory* 19.1 (2011): 393-405.

[10] Hidas, Peter. "A functional evaluation of the AIMSUN, PARAMICS and VISSIM microsimulation models." *Road and Transport Research* 14.4 (2005).

[11] Hill, R. John, and Theresa K. Yates. "Modelling railway block signaling systems using discrete-event simulation." *Railroad Conference, 1992. Proceedings of the ASME/IEEE Spring Joint*. IEEE, 1992.

[12] Hollander, Yaron, and Ronghui Liu. "The principles of calibrating traffic microsimulation models." *Transportation* 35.3 (2008): 347-362.

[13] Horni, Andreas, Kai Nagel, and Kay W. Axhausen. "The multi-Agent transport simulation MATSim." *Ubiquity, London* 9 (2016).

[14] Jin, Jian Gang, Kwong Meng Teo, and Lijun Sun. "Disruption response planning for an urban mass rapid transit network." *transportation research board 92nd annual meeting, Washington DC*. 2013.

[15] Keiji, Kariyazaki, Hibino Naohiko, and Morichi Shigeru. "Simulation analysis of train operation to recover knock-on delay under high-frequency intervals." *Case Studies on Transport Policy* 3.1 (2015): 92-98.

[16] Kraft, Karl Heinz. *Fahrdynamik und Automatisierung von spurgebundenen Transportsystemen*. Vol. 17. Springer-Verlag, 2013.

[17] Kraft, Walter H. "AN ANALYSIS OF THE PASSENGER VEHICLE INTERFACE OF STREET TRANSIT SYSTEMS WITH APPLICATIONS TO DESIGN OPTIMIZATION." (1975).

[18] Lin, Tyh-ming, and Nigel HM Wilson. "Dwell time relationships for light rail systems." *Transportation Research Record* 1361 (1992).

[19] Lu, Yang, et al. "Simmobility mid-term simulator: A state of the art integrated agent based demand and supply model." *94th Annual Meeting of the Transportation Research Board, Washington, DC*. 2015.

[20] Motraghi, Adam, and Marin Varbanov Marinov. "Analysis of urban freight by rail using event based simulation." *Simulation modelling practice and theory* 25 (2012): 73-89.

[21] Nash, Andrew, and Daniel Huerlimann. "Railroad simulation using OpenTrack." *Computers in railways IX* (2004): 45-54.

[22] Ottomanelli, M., Iannucci, G., and Sassanelli, D. (2012). A simplified pedestriansvehicles interaction model at road crossings based on discrete events system
[23] Pelletier, Marie-Pier, Martin Trépanier, and Catherine Morency. "Smart card data use in public transit: A literature review." *Transportation Research Part C: Emerging Technologies* 19.4 (2011): 557-568.
[24] Puong, Andre. "Dwell time model and analysis for the MBTA red line."*Massachusetts Institute of Technology Research Memo* (2000).
[25] Ravichandran, Harshavardhan. *Evaluating the robustness of crew schedules for rail transit systems*. Diss. Massachusetts Institute of Technology, 2013.
[26] Sánchez-Martínez, Gabriel Eduardo. *Running time variability and resource allocation: a data-driven analysis of high-frequency bus operations*. Diss. Massachusetts Institute of Technology, 2012.
[27] Spall, James C. "Implementation of the simultaneous perturbation algorithm for stochastic optimization." *IEEE Transactions on aerospace and electronic systems* 34.3 (1998): 817-823.
[28] Sun, Lijun, et al. "Using smart card data to extract passenger's spatio-temporal density and train's trajectory of MRT system." *Proceedings of the ACM SIGKDD international workshop on urban computing*. ACM, 2012.
[29] Sun, Yanshuo, and Paul M. Schonfeld. "Schedule-based rail transit path-choice estimation using automatic fare collection data." *Journal of Transportation Engineering* 142.1 (2015): 04015037.
[30] Zhang, R., Zhu, X., and Lei, L. (2009). Study on intersections with median brt stations: Focused on pedestrian steet-crossing charateristics. ICCTP, pages 1392 1399.
[31] Zhao, Jinhua, Adam Rahbee, and Nigel HM Wilson. "Estimating a Rail Passenger Trip Origin-Destination Matrix Using Automatic Data Collection Systems." *Computer-Aided Civil and Infrastructure Engineering* 22.5 (2007): 376-387.
[32] Zhu, Yiwen. *Passenger-to-train assignment model based on automated data*. Diss. Massachusetts Institute of Technology, 2014.
[33] (n.d.). Retrieved May 15, 2017, from http://paginas.fe.up.pt/~feliz/man/modeller.html
[34] (n.d.). Retrieved Feb 12, 2017, from http://www.jbss.de/
[35] (n.d.). Retrieved May 12, 2017, from http://www.sumo.dlr.de/userdoc/Sumo_at_a_Glance.html