Calibration of Mesoscopic Traffic Simulation Models for Dynamic Traffic Assignment

by

Kunal Kamlakar Kundé

B.Tech. in Civil Engineering (2000)

Indian Institute of Technology, Bombay, India

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

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2002

Author .....
Department of Civil and Environmental Engineering
August 9, 2002

Certified by .....
Moshe E. Ben-Akiva
Edmund K.Turner Professor
Department of Civil and Environmental Engineering
Thesis Supervisor

Certified by .....
Haris N. Koutsopoulos
Operations Research Analyst
Volpe National Transportation Systems Center
Thesis Supervisor

Accepted by .....
Oral Buyukozturk
Chairman, Department Committee on Graduate Students

Calibration of Mesoscopic Traffic Simulation Models for Dynamic Traffic Assignment

by

Kunal Kamlakar Kundé

B.Tech. in Civil Engineering (2000)

Indian Institute of Technology, Bombay, India

Submitted to the Department of Civil and Environmental Engineering
on August 9, 2002, in partial fulfillment of the
requirements for the degree of
Master of Science in Transportation

Abstract

This thesis tackles the calibration of mesoscopic flow propagation models in the traffic dynamics simulator of the DynaMIT Dynamic Traffic Assignment system. A three-stage methodology is developed to carry out calibration in a sequential manner at increasing levels of aggregation. Two stochastic optimization approaches – one based on stochastic gradient approximation and one that does not employ gradients – are applied to carry out the calibration along the lines of a simulation optimization problem. The parameters to be calibrated are the $u_{min}, u_f, k_0, k_{jam}, \alpha$, and β parameters in the speed-density relationship for every road segment and the capacities of all road segments and intersections. The methodology is applied to a study network extracted from the Orange County region in California using traffic surveillance data obtained from the California Department of Transportation (Caltrans).

Thesis Supervisor: Moshe E. Ben-Akiva
Title: Edmund K.Turner Professor
Department of Civil and Environmental Engineering

Thesis Supervisor: Haris N. Koutsopoulos
Title: Operations Research Analyst
Volpe National Transportation Systems Center

Acknowledgments

I would like to thank:

Professor Moshe E. Ben-Akiva for his invaluable guidance and support. He has been an inspiration and a role model in more ways than one, and it has been a great privilege and honor to work with such a colossal transportation engineering researcher.

Dr. Haris Koutsopoulos for his help, encouragement and constructive comments. Working with him has provided me with edification on aspects of life well outside the scope of simulation and transportation research. He belongs soundly in the category of the few people that I think are worthy of emulation.

Siva for his unwavering friendship, for the encouraging zwrites, and for sharing his athena account with me to expedite my thesis completion.

Rama for his help as Latex guru and for his help with all the data needed for the calibration.

Srini for his help with the DynaMIT code and related bug fixes.

Anshul Sood for lending me his account on ORC's superfast alfred. Virtually a heaven-sent saviour.

Darda for his suggestions on tackling calibration, and for lending a patient ear whenever I felt the urge to vent my spleen.

Akhil for the peppy talk and the telephone calls to enliven my research enthusiasm.

Marge for helping me with my jobhunt and for lifting my sagging spirits in the face of impossible deadlines.

Paussems for initiating me into the fine art of placing research in the perspective of Belgian hedonism (I am not sure I agree with him), and for playing his self-titled "code monkey" role to perfection.

Constantinos for his thesis-writing advice and for his UNIX help.

Manish for his help with DynaMIT and for the engaging discussions on life.

Sood for lending a homely air to #9, 905 Main Street, and being a fine debating foe.

My best American friend Dan Morgan for helping me decipher American slang, being a great listener, and showing me the rewards of never making excuses.

Yosef Brandriss, Angus Davol, Tomer Toledo and Shlomo Bekhor for their help and friendship, and Joe Scariza and Zhili Tian for making 3cc the place that it is.

Cynthia Stewart for being ever-accommodating to all my thesis-submission-related requests, and for her sound professional-plus-matronly advice and support.

The Center for Transportation and Logistics (formerly the Center for Transportation Studies) for giving me the opportunity to study at MIT and the United States Federal Highway Administration for financing my studies.

Finally, and most importantly, I would like to thank my family for their unconditional love and moral support, and to thank God for helping me fulfill my dream of earning an MIT degree.

Contents

1 Introduction **15**

1.1 Dynamic Traffic Assignment (DTA) 16

1.2 DynaMIT and DynaMIT-P 17

1.3 Thesis Focus 18

1.4 Thesis Outline 19

2 Literature Review **21**

2.1 Development of Traffic Models 21

2.2 Data sources 22

2.2.1 Infrastructure-based detectors 22

2.2.2 Non-infrastructure-based detectors 22

2.3 Traffic Model Parameters 23

2.4 Calibration of Traffic Simulation Models 24

2.5 Conclusion 30

3 Calibration Methodology **31**

3.1 Ideal Calibration Methodology 31

3.2 Calibration of the DynaMIT system 33

3.2.1 Supply Simulator Parameters 34

3.2.2 Demand Simulator Parameters 36

3.3 Proposed Calibration Methodology 37

3.3.1 Three-stage Calibration 37
3.3.2 The Tool for Calibration 39

4 Stochastic Optimization **43**

4.1 Stochastic Optimization of Simulation Systems 43
4.1.1 Issues 43
4.1.2 Notation 44
4.1.3 Classification 45
4.2 Path search methods 45
4.2.1 Response Surface Methodology 45
4.2.2 Stochastic Approximation 46
4.3 Pattern search methods 51
4.3.1 Hooke and Jeeves Method 51
4.3.2 Nelder and Mead (Simplex Search) Method 52
4.3.3 Box Complex Method 52
4.4 Random methods 53
4.5 Summary 56

5 Case Studies **59**

5.1 The Irvine Data 59
5.1.1 Network Description 60
5.1.2 Data Description 60
5.2 The First Stage of Calibration 63
5.3 The Second Stage of Calibration 68
5.4 Network-Level Calibration 71
5.4.1 The Box Complex Algorithm 73
5.4.2 The SPSA Algorithm 75
5.5 Results 79
5.6 Runtime and Convergence 92

5.7 Summary 96

6 Conclusion **97**

6.1 Research Contribution and Findings 97

6.2 Future Research 98

A MATLAB Code for the Box Complex Algorithm **99**

B MATLAB Code for the SPSA Algorithm **111**

List of Figures

1-1 Dynamic Traffic Management Overview 17

2-1 Calibrated speed-flow relationship for Interstate 4 in Orlando, Florida 26

3-1 A Generic Calibration Framework for Traffic Models [15] 32
3-2 Three-stage Calibration Methodology 38
3-3 Sub-network Calibration 40
3-4 Stochasticity in the Supply Simulator Output 42

4-1 The White-Box Approach to Simulation Optimization 48
4-2 The Black-Box Approach to Simulation Optimization 49

5-1 Map of the Study Network Area 61
5-2 The Irvine Network 62
5-3 Primary OD Pairs in the Irvine Network 64
5-4 5-day Variability in Arterial Sensor Counts 65
5-5 5-day Variability in Freeway Sensor Counts 65
5-6 Typical Calibrated Speed-Density Curve 67
5-7 The SSC Network 68
5-8 Freeway sensor counts on the SSC network 69
5-9 Off-ramp sensor counts on the SSC network 69
5-10 On-ramp sensor counts on the SSC network 70
5-11 (Second) off-ramp sensor counts on the SSC network 70

5-12 Calibrated SSC Network Speeds Comparison 72
5-13 Improving SPSA Performance with Gradient Averaging 78
5-14 Flows for 04:00-04:15 and 04:15-04:30 82
5-15 Flows for 04:30-04:45 and 04:45-05:00 83
5-16 Flows for 05:00-05:15 and 05:15-05:30 84
5-17 Flows for 05:30-05:45 and 05:45-06:00 85
5-18 Flows for 06:00-06:15 and 06:15-06:30 86
5-19 Flows for 06:30-06:45 and 06:45:-07:00 87
5-20 Flows for 07:00-07:15 and 07:15-07:30 88
5-21 Flows for 07:30-07:45 and 07:45-08:00 89
5-22 Flows for 08:00-08:15 and 08:15-08:30 90
5-23 Flows for 08:30-08:45 and 08:45-09:00 91
5-24 Convergence of the Box Complex Algorithm 94
5-25 Convergence of the SPSA Algorithm 95

List of Tables

4.1 Gradient Estimation Approaches for Stochastic Approximation (Fu [11]) 51

5.1 Results of Individual Segment Calibration 66

5.2 Results of SSC Network Calibration 71

5.3 Root Mean Square Error values for the Different Optimizations . . . 79

5.4 Calibrated Values Versus Starting Values 81

5.5 Convergence of the Box Complex Algorithm 93

5.6 Runtimes of the Box Complex Algorithm 93

5.7 Runtimes of the SPSA Algorithm 95

Chapter 1

Introduction

Recent years have witnessed a spurt in the development and deployment of Intelligent Transportation Systems (ITS). Much of the impetus for the development of these systems was derived from a paucity of investment funds, and more importantly, from a lack of public willingness to expand roadway capacity at the cost of detriments to the environment. ITS helps in bolstering the efficiency, productivity and safety of extant transportation infrastructure through the use of the latest data processing, communication and surveillance technologies.

Traffic information devices such as Variable Message Signs (VMS) are now ubiquitous; most ITS service providers also provide in-vehicle trip advisory to their subscribers regarding accidents and bottlenecks. However, many ITS sub-systems such as Advanced Traffic Management Systems (ATMS), Advanced Traveler Information Systems (ATIS), Advanced Public Transportation Systems (APTS), Commercial Vehicle Operations (CVO) and Emergency Management Systems (EMS) would get a much-needed fillip with the ability to assimilate wide-area estimates of prevalent and emerging traffic. Dynamic Traffic Assignment (DTA) systems[1] are being developed to serve this very need for a Traffic Estimation and Prediction System (TrEPS).

[1]Sometimes synonymously referred to as Dynamic Traffic Management Systems (DTMS)

1.1 Dynamic Traffic Assignment (DTA)

The dynamic traffic assignment problem has been the focus of research for more than three decades now ([39], [24], [25], [7]). The advancement of intelligent transportation systems in the last decade has intensified DTA research and led to the development of DTA systems aimed at dynamic traffic management [9], [10], [21], [27].

DTA systems aim to improve general traffic conditions on a proactive basis. Such systems are designed to address and alleviate traffic congestion using advanced communication and surveillance technologies managed by real-time, intelligent software systems. Strategies to ameliorate the congestion call for the use of ITS sub-systems such as ATMS and ATIS.

ATMS refers to control systems that impose constraints on traffic flows. Such constraints include traffic signal lights (based on fixed timing or on proactive rules), ramp metering, speed limit signs (fixed or variable) and lane use signs. Statutory restrictions demand drivers' compliance with such systems. ATMS can thereby modify the capacity of the network and affect transportation supply.

ATIS refers to information systems that provide traffic information and travel recommendations to drivers both before and during their trips. Such guidance may be through means such as radio forecast, web-based or on-board navigation systems and variable message signs. The information provided by these systems may have a bearing on drivers' trip-related choices: the decision to travel or not, which destination(s) to travel to, the departure time, the travel mode, and route choice. By influencing drivers' travel decisions, ATIS influence transportation demand.

ATIS differ from ATMS in that drivers are not obligated to follow its prescriptive recommendations. Also, an ATMS, designed to control the traffic, is driven by system optimal objectives while an ATIS, designed to influence demand, is driven by user optimal objectives. Figure 1-1 illustrates the roles of ATMS and ATIS in the overall management of a road network and its traffic.

Researchers at MIT have developed the DynaMIT and DynaMIT-P DTA systems.

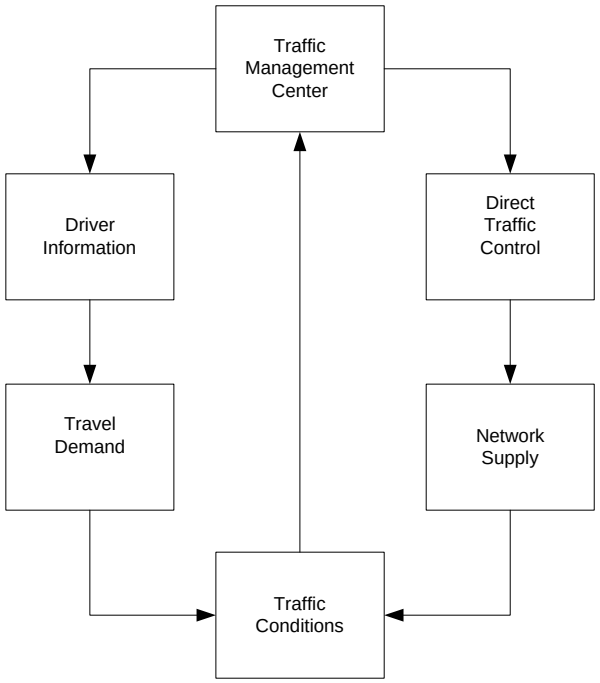

Figure 1-1: Dynamic Traffic Management Overview

1.2 DynaMIT and DynaMIT-P

DynaMIT [2] is a state-of-the-art DTA system designed for real-time traffic estimation and prediction, and generation of traveler information and route guidance. DynaMIT has been designed to reside in a traffic management center (TMC) and to support ATIS and ATMS operations.

DynaMIT-P is a DTA-based planning tool developed on the DynaMIT platform. It has been designed to assist transportation planners and policymakers in assessing possible operational or infrastructural modifications in the local or regional transportation networks.

Both DynaMIT and DynaMIT-P use two main simulation tools – the **demand simulator** and the **supply simulator**.

The demand simulator is a microscopic simulator that deals mainly with esti-

[2] **Dy**namic **N**etwork **A**ssignment for the **M**anagement of **I**nformation to **T**ravelers

mation and prediction of time-dependent OD flows, demand disaggregation to model the socioeconomic characteristics of drivers, and their decisions of departure time and route choice using (MNL-type) behavioral models.

The supply simulator is a mesoscopic traffic simulator used to simulate vehicular movements [3] on the given network. It can be used to infer traffic flows, queue lengths, speeds, travel times, and densities at all points on the network, and thereby serve to indicate network performance.

1.3 Thesis Focus

The focus of this thesis is the calibration of the supply simulation module within DynaMIT-P.

Calibration is the estimation process to determine correct values of the model parameters, based on traffic measurements. It is aimed at optimally adapting model parameters and coefficients so that the calibrated model is able to replicate field observations to a sufficient level of accuracy.

The calibration process is almost always mated to the process of validation. Validation is attempted subsequent to calibration: the objective is to test whether the calibrated model is able to reproduce field observations to a sufficient level of accuracy, both from a qualitative and quantitative point of view.

The importance of the calibration and validation of models in efficient and correct model application studies cannot be overstated. Regular updating model parameters and relations affords ever-increasing accuracy in modeling results. A congestion-related study by the Dutch Ministry of Transport [33] concludes that models are seldom exhaustively calibrated and validated (if at all!) for their considered application.

[3] also referred to as 1. network performance, or 2. traffic dynamics, or 3. flow propagation elsewhere in this document

1.4 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 explores the literature related to calibration of traffic simulation models. More specifically, it dwells on the calibration of the components of a DTA system, especially the traffic dynamics simulation models.

Chapter 3 outlines a methodology for the calibration of mesoscopic flow propagation models on the basis of the literature reviewed.

Chapter 4 discusses the topic of simulation optimization in light of the optimization-pronged attack needed to tackle the calibration problem. Different stochastic optimization algorithms are looked into, and this exercise is used to identify the algorithms that will be used for the calibration purpose on hand.

Chapter 5 presents case studies. It begins by outlining a three-stage calibration methodology, and then employs a random search algorithm (the Box Complex algorithm) and a path search algorithm (the Simultaneous Perturbation Stochastic Approximation algorithm) for calibration of a large real network.

A summary of the research and directions for further work are presented in the final chapter.

Chapter 2

Literature Review

The objective of this chapter is to study the different approaches employed by researchers in calibrating traffic simulation models. This literature review is used to avoid the pitfalls, overcome the deficiencies and adopt the plus-points of past approaches in the calibration methodology we will outline and implement.

2.1 Development of Traffic Models

Traffic model development usually follows the lines of:

1. Establishing and testing theories on the basis of microscopic/macroscopic traffic flow observations, either by inductive or by deductive means. This involves developing qualitative and mathematical relations based on empirical knowledge and qualitative data analysis.[1]

2. **Calibration of the models for a specific application using empirical data**.

[1] also referred to as
A. Model verification – process of determining how well the underlying model theory and logic reflect reality, and
B. Model validation – process of determining if the proposed model logic is correctly represented by the computer code

3. Validation of the calibrated models using different data

2.2 Data sources

This section briefly touches upon the different infrastructure-based and non-infrastructure-based data sources.

2.2.1 Infrastructure-based detectors

The infrastructure-based detector systems are of two types: the inductive loop-based systems and the video, infrared and radar systems. Loop-based systems can provide either lane-aggregate or lane-specific speeds and flow-rates at varying frequencies (minute-aggregate/hour-average etc. depending on the type of loop-based system). Some such systems also enable collection of individual vehicle data. Video, radar and infrared techniques are used to collect individual vehicle data (from which macroscopic data can be determined) in a small region. However, these are not as accurate as the loop-based systems.

2.2.2 Non-infrastructure-based detectors

This category refers to the use of vehicles themselves being used for data collection purposes. Such *probe* vehicles are equipped with on-board computers (OBC) or on-board units (OBU) from which positioning (via GPS) and speed measurements may be obtained. Combining probe trajectories with fixed detector data allows for the estimation of flow characteristics of non-equipped vehicles.

Alternatively, aerial photographs and video data can be collected. With sufficiently detailed data, densities can be directly determined. Also, quantities such as space-mean-speeds and distance-headways can be determined by comparing photographs of consecutive time instants.

2.3 Traffic Model Parameters

Traffic simulation models are usually classified as macroscopic, mesoscopic or microscopic on the basis of the level of detail of simulation.

Macroscopic and mesoscopic models use the analogy between vehicular flow and fluid flow. When compared to microscopic models, the number of unknown model relations and parameter relations is relatively low.

Calibration of first-order macroscopic and mesoscopic models requires the construction of speed-density (or flow-density) curves – mesoscopic models do, of course, require a more detailed modeling relationship (with more governing parameters) than macroscopic models. Traffic data covering the congestion spectrum from free-flow to congested flow ensures a complete fundamental diagram.

Higher-order and microscopically-based models require estimation techniques for model parameters other than those implicit in speed-density curves. For instance, the *constant anticipation factor* c_0 reflects the spread in velocities while a so-called *viscosity coefficient* captures the transition rate from brisk to careful driving. To determine c_0, one needs to either

1. determine the velocity of shockwave propagation, or

2. use speed variances

In general, macroscopic model calibration is relatively straightforward relative to the microscopic case.

Calibration of microscopic models is concerned with tuning parameters that determine vehicle-vehicle interactions, such as the ones in car-following and lane-changing models of driver behavior.

The relatively large number of parameters dictates that complex microscopic models should be dis-assembled, calibrated and tested in a step-by-step fashion, whenever possible. Sometimes the number of degrees of freedom in a microscopic model is so

large that multiple parameter value combinations yield the same model behavior. To aid microscopic model calibration, sensitivity analysis can be performed.

Data requirements for microscopic model calibration are very stringent, in that individual vehicle data in the form of trajectories and pair-wise vehicle dependency data are needed. In contrast, data requirement criteria for model validation are less stringent – they need reflect only the specific traffic flow behaviors that the calibrated model is expected to describe.

2.4 Calibration of Traffic Simulation Models

While there is no dearth of literature on the calibration of stand-alone microscopic traffic simulation models, literature on calibration of traffic simulation models in a DTA setting is sorely lacking. We review some of the literature as regards calibration of traffic simulation models, especially the calibration of mesoscopic/macroscopic traffic dynamics simulation models in DTA systems.

Hellinga [15] provides an excellent discussion of the requirements for the calibration of traffic simulation models. The paper tries to address the key issue of what constitutes adequate model calibration and what measures of performance (MOPs) should be used in calibration. The author argues that a model can be deemed to be calibrated if its outputs are comparable to field data and meet pre-specified calibration criteria established prior to the commencement of any modeling. Some of the potential MOPs listed are: link volume, link speed, link travel time, queue size and location, trip travel time by origin, destination, and departure time, total travel time, average trip length, average fuel consumption, tailpipe emissions and average accident risk. The author also points out that sufficient field data to quantify statistical confidence limits on the MOP(s) of interest are very rarely available, and this lack thereof rules out such a rigorous statistical approach. Consequently, it is not uncommon to see the use of terms such as *reasonable, adequate*, and *representative*

when referring to the adequacy of calibration.

The author addresses the loss of accuracy in network modeling when choices have to be made on the spatial extents of the network and on source/sink zones. The paper briefly dwells upon the correct specification of macroscopic speed-density relationships and points out how it is beneficial to classify links into several categories and specify a unique speed-density relationship for each category rather than for each individual link. This idea is carried forth in the current research where similar road segments are grouped together for the purpose of calibration. Also pointed out is the fact that free speed is likely the least critical parameter, since it can be estimated with reasonable accuracy from the mandated speed limit itself. It is stressed that the impacts of speed at capacity flow, the capacity and the jam density are far more significant as they influence the formation, extent and dissipation of queues.

Route choice behavior is yet another aspect of calibration for networks where the drivers have more than one viable route choice. Field data to capture route choice can rarely be obtained; an inappropriate route choice parameter value can annul the accurate calibration of all other parameters. And finally, O-D demands, a must as input for nearly all simulation models, cannot be directly observed and must be derived via other means, most commonly from link flows.

The paper concludes by pointing out that simulation results are often incorrect because of the use of an insufficiently validated and verified model on the part of the model developer or an incorrectly calibrated model on the part of the user. It recommends a clearly and realistically defined data collection process for minimal negative impact on the model user's calibration process.

Kurian [19] uses an experimental design methodology to identify the set of sensitive parameters in the car-following model in MITSIM, a microscopic traffic simulator. He uses an optimization-based framework and two forms of an objective function to quantify the deviation between observed and simulated values. Stochasticity is shown to have a very significant impact on the optimal parameter values. The thesis

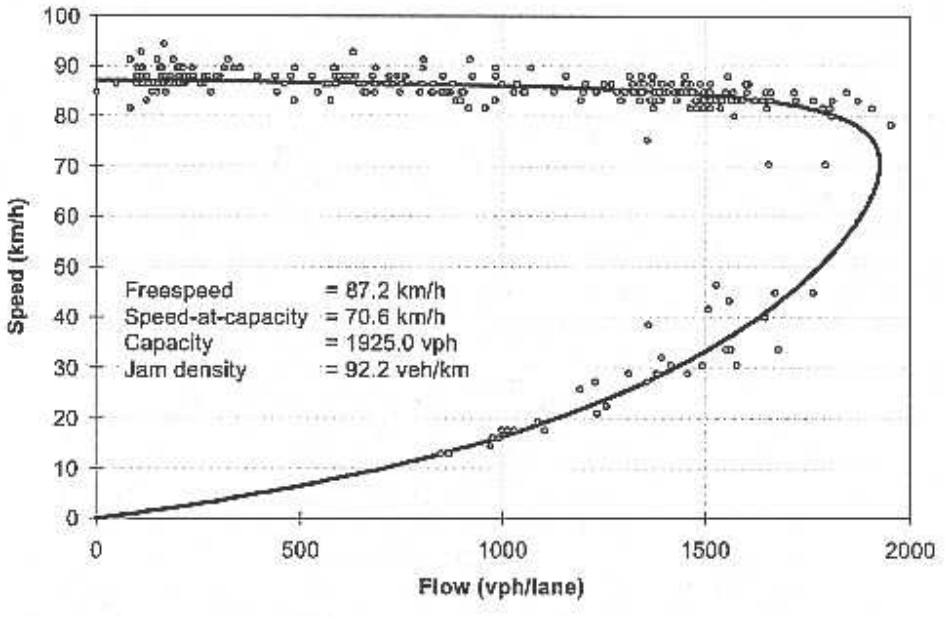

Figure 2-1: Calibrated speed-flow relationship for Interstate 4 in Orlando, Florida (Van Aerde and Rakha [1])

concludes with a very strong endorsement of calibration studies, mentioning that simulation performance was significantly enhanced by appropriate calibration.

Darda [8] has developed a module for **joint** model parameter calibration and OD estimation in the MITSIM microscopic traffic simulator. Again, an optimization-based framework driven through the Box Complex algorithm is used while taking into account the interaction between various model parameters and OD flows.

Van Aerde and Rakha [1] have carried out calibration of speed-flow relationships using loop detector data aggregated to a 5-minute average. Figure 2-1 shows the parameter values corresponding to the curve fitted for Interstate 4 in Orlando, Florida. Such curve-fitting exercises, however, cannot be carried out for realistically sized networks with hundreds of segments; the issue is not the prohibitively large number of segments, more a lack of data!

Jayakrishnan et al [16] discuss calibration issues concerning microscopic simulation for ATIS and ATMS. They describe a hybrid simulation approach wherein Paramics,

a microscopic traffic simulator, is integrated with the routing and behavior response schemes in the DYNASMART mesoscopic model. The calibration scheme uses a weighted least squares objective function to match simulated values with field data. However the termination criterion is described as

> If the comparisons between observed and modeled value comparisons (sic) are within recommended guidelines and the graphic visualisation of vehicles is realistic, then the model is deemed calibrated.

Wall et al [38] perform calibration of a microscopic model with the intent of using it in conjunction with real-time loop detector data to predict downstream traffic volumes and speeds. The calibration is performed in a least squares sense of matching model output with observed data. The model parameters are updated using a finite difference approximation for the differential.

Mahmassani and Tavana [22] use transfer function methods to capture the dynamic characteristics of speed-density time series data, including the existence of serial correlation, in the specification and estimation of dynamic speed-density relations for traffic simulation and ATMS applications.
The prevailing speed at any location is surmised to be a function of past values of speed, density and traffic conditions upstream and downstream. The general model is represented as

$$u_{t,x} = \{u(\eta, \xi), k(\eta, \xi), u^e[k(q)]\}$$

where

$u_{t,x}$	= speed at time t and location x,
$u(\eta, \xi)$	= distribution of speed values over space, ξ, and time η
$k(\eta, \xi)$	= distribution of density over space, ξ, and time η
$u^e[k(q)]$	= static equilibrium speed for a given density, which in turn itself can be a function of the flow, q.

In the context of data received at an ATMS control center, the above relationship is operationalized as

$$u_i^t = f\left[u_{i\pm j}^{t-h}, k_{i\pm j}^{t-h}, u^e(k_i^{t-h}), a_i^{t-h}\right] \qquad j,\ h\ \epsilon\ \{0,1,2,3,...\}$$

where

$$a_i^t \qquad = \text{shock introduced to the system at time } t \text{ and location } i.$$

The authors show that transfer function methods outperform [2] regression calibration models when the equilibrium speed-density relationship is assumed to be known (the dependent variable is taken to be the absolute value of the differenced speed variation). They also claim tranferability of the model to new sites [3] without the need for recalibration. Furthermore, it is conjectured that even though traffic conditions at the selected site were in the uncongested regime, transfer function models will exhibit even better performance under congested conditions in which dynamic effects are more significant.

It is worth emphasizing that calibration of transfer function models would need very detailed data. Also, if we are to include spatial effects, we need a closely spaced set of sensors. Finally, the above research was based upon data from freeway sensors and applicability of transfer function models to wide-area networks needs further investigation. Nevertheless, transfer function models hold much promise in online calibration for the purpose of supporting real-time operational decisions. They could be one of the strategies for mining expansive real-time traffic databases.

He et al [14] outline a three-step framework for the calibration of an *analytical* DTA model. Assuming knowledge of OD trip tables, they propose sequential calibration of:

1. Dynamic Link Travel Time Functions

[2] In their analysis of data obtained at a single station on Interstate 10, the authors ignore spatial effects ($j = 0$).

[3] The other site was also in the San Antonio region.

2. Route Choice Model

3. Flow Propagation

with 1 and 2 being implemented offline or online and 3 being implemented online. For offline calibration, the authors assume availability of adequate surveillance data and survey data of route choice, traveler demographics, travel times and flow propagation. They propose calibration of linear or nonlinear link travel time formulae for different types of links: freeway links, pre-timed signalized links, actuated signalized links, stop-controlled links etc. They also postulate that various route choice formulae could be calibrated offline for different types of travelers.
For the online calibration of dynamic link travel time functions, the authors propose introduction of an error term for fitting their proposed formulae with detector data. They propose use of the maximum likelihood method for online calibration of route choice, again contingent of the availability of "sufficient route choice data". The online calibration of flow propagation is also proposed to be carried out by adjustment via an error term.
A major weakness of the proposed methodology is that it is heavily reliant on a rich archive/supply of data and the calibration suggestions are heavily qualified with "can be". The test network used to illustrate the methodology is a very simplistic 3 X 3 Manhattan metric of freewat links and assumes prior knowledge of time-dependent OD matrices, a very restrictive assumption for the purpose of DTA. The scalability of this methodology to heterogeneous realistically-sized networks is not investigated.

Hawas [13] ranks individual DTA components to determine the order in which calibration should be performed. However, he ignores the interaction between individual components that is the prime essence of any DTA system. A further weakness of this research is that the case study uses the components of a traffic simulator rather than a DTA system.

Balakrishna [3] addresses the overall problem of jointly calibrating the OD estimation and route choice models in the demand simulation component of the DTA

prototype DynaMIT.

2.5 Conclusion

The literature reviewed indicates the need for a systematic calibration of the traffic dynamics simulator in a DTA system. Drawing upon the ideas of past research investigations, the next chapter outlines a calibration methodology customized for the calibration of the supply simulation module in DynaMIT.

Chapter 3

Calibration Methodology

This chapter starts with a blueprint for an "ideal" calibration methodology and then proposes one for our own problem so that it adheres to the same. In so doing, we draw heavily on the approaches used by researchers cited in the last chapter while factoring in the characteristics unique to our mesoscopic traffic dynamics simulator, the network of interest, and the dataset.

3.1 Ideal Calibration Methodology

Guidelines for the calibration and validation of traffic simulation models are outlined in [30] and [26]. An unambiguous, clear, structured and operational calibration methodology would ideally include the following, interdependent stages:

1. Identification of model application area

2. Determination of assessment objectives

3. Determination of performance measures: definition of performance indicator, its calculation, nature of the indicator in terms of the effect it aims to quantify, scope of the indicator and data collection needs

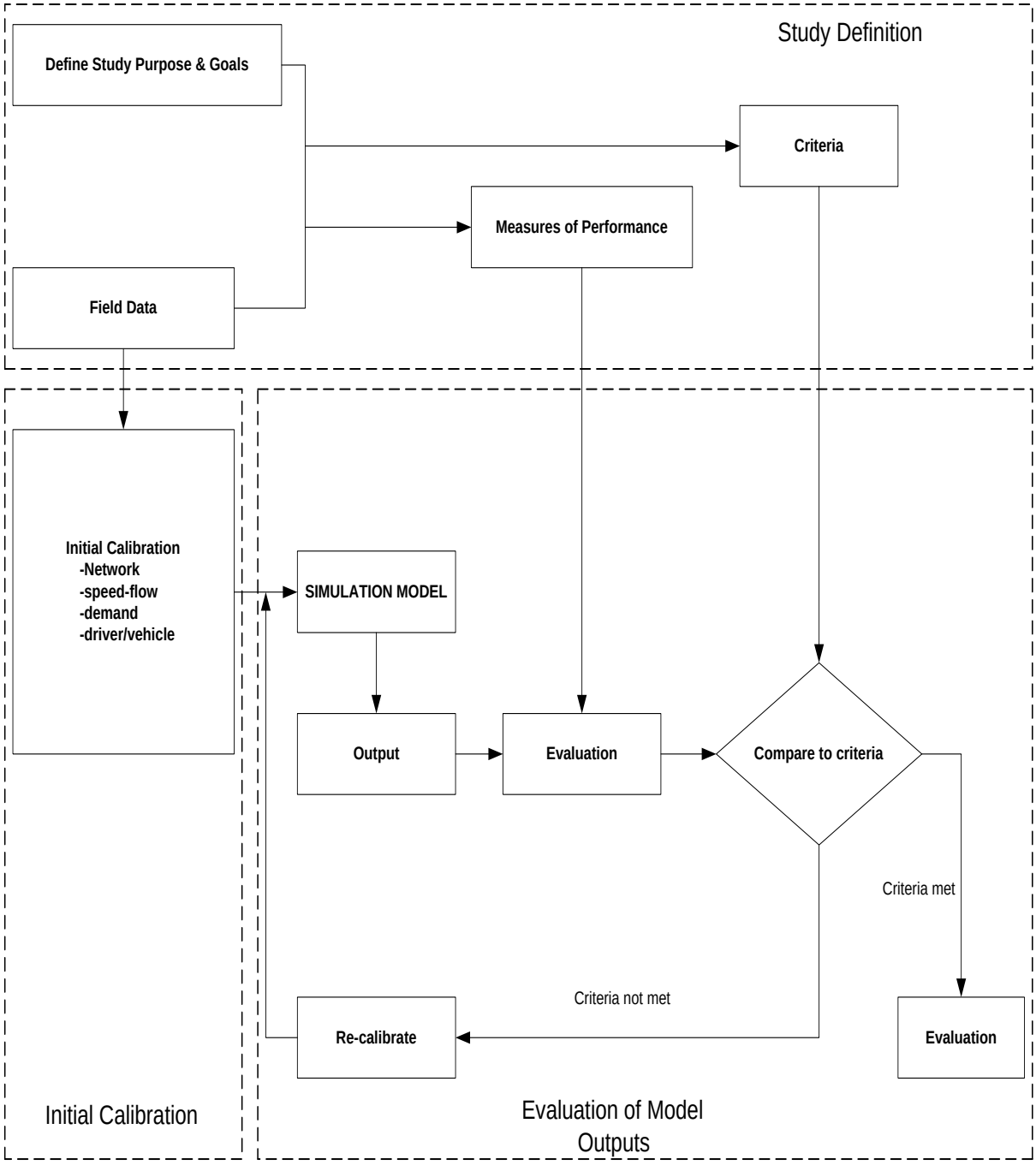

Figure 3-1: A Generic Calibration Framework for Traffic Models [15]

4. Description of general tools for calibration, e.g. sensitivity analysis, step-by-step calibration, mathematical programming algorithms, t-tests, analysis of variance

5. Overall definition of success

6. Experimental set-up

The main advantage of such a methodology is the systematically organized calibration and validation process, and a vantage ground for cross-comparison between models. Figure 3-1 shows a generic three-phase, eight-component calibration framework for traffic models (Hellinga [15]).

The first phase comprises exercises such as definition of study goals, identification of data requirements vis-a-vis the field data available, identification of measures of performance consistent with the objectives of the study, and of simulation capabilities; and the specification of criteria to evaluate the calibration. All these activities are undertaken prior to the commencement of any modeling.

The second phase involves the actual calibration – namely the representation of the network, the macroscopic speed-density (or speed-flow) relationships, driver behavior models and estimation of O-D flows.

In the last phase, the results from the model are compared with field data and against the pre-specified criteria. If the model meets these criteria, it is deemed to be adequately calibrated and is ready for testing non-base-case scenarios. If, however, the model is found to be wanting in meeting these pre-specified criteria, refinements and/or modifications, and sometimes even a recalibration from scratch must be done.

3.2 Calibration of the DynaMIT system

In the current context, we are interested in calibrating the DynaMIT system. A holistic calibration of the DynaMIT system involves joint calibration of the demand and supply simulators. The current research assumes the availability of a calibrated

demand simulator and focuses exclusively on the calibration of the supply simulator, which we describe next. For the sake of completeness, however, we also enumerate the demand simulation parameters at the end of this section.

3.2.1 Supply Simulator Parameters

The supply simulator in DynaMIT is a **mesoscopic** traffic simulator that simulates vehicular movements and provides metrics of network performance such as the temporal evolution of flows, speeds, densities, queue lengths, and travel times at all points on the network.

The supply simulator obtains the network description and the list of *packets* to be moved on the network through its interfaces with the network topology component and the list of packets component respectively. It then simulates the movement of vehicles on the network for the given supply simulation time interval.

The network consists of a set of nodes, links and loading elements (loaders). The nodes correspond to the intersections or merge points in the actual network, while links represent the unidirectional pathways between them. Loading elements represent areas where traffic is generated and/or attracted. The links in the network are subdivided into segments to capture changing geometries. Further, the lanes within each segment are grouped into lane groups to account for turn-specific capacities at diversions/merge points and intersections.

Unlike microscopic simulators, there are no lane-changing or car-following models; lanes are used primarily to capture queuing behavior. There is also the concept of *packets*, whereby two or more vehicles having the same origin and destination are aggregated – the simulator concentrates on aggregate results rather than modeling individual vehicle behavior. Detailed **mesoscopic** models are used to capture traffic dynamics and accurately model the formation and dissipation of queues, i.e. there are two major models: 1. a deterministic queuing model and, 2. a speed model.

The queuing model, with the ability to explicitly model spillbacks, makes heavy

use of the input and output capacities of individual segments; the movement of vehicles from one segment to the next is governed by a host of capacity calculations and the physical space available on the downstream segment. Furthermore, capacities are also used to accurately model incidents and signalized controls at intersections.

Flow propagation is based on the following aggregate speed-density relationship

$$u = \mathbf{max}\ (u_{min}, u_{cal})$$

$$u_{cal} = \begin{cases} u_f, & \text{if } k \leq k_0 \\ u_f[1 - (\frac{k-k_0}{k_{jam}})^{\beta}]^{\alpha}, & \text{otherwise} \end{cases}$$

where

u	= speed (mph),
u_{min}	= minimum speed,
u_{cal}	= speed calculated as per the two-regime equation,
u_f	= free-flow speed,
k	= density,
k_0	= maximum density at which free-flow is possible,
k_{jam}	= jam density,
α	= parameter,
β	= parameter.

The simulation of the traffic network operations proceeds in a number of *update* phases. Nested within each update phase are a number of *advance* phases. The update phase is used to update the dynamic traffic characteristics (e.g. speeds, densities, capacities) whereas the advance phase is used to advance vehicles to their new positions.

Depending on the demands of the application, the supply simulator can be operated at different levels of granularity – this flexibility allows for a full investigation

of tradeoffs between accuracy and run-time efficiency, and facilitates the selection of the most suitable combination for each case.

We list below the parameters that are key to calibrating the supply simulator:

- Segment-specific speed-density parameters $u_f, u_{min}, k_0, k_{jam}, \alpha$, and β,
- Lane group capacities on freeway and arterial segments, and
- Lane group capacities at the signalized intersections.

3.2.2 Demand Simulator Parameters

The demand simulation parameters are those embedded in the O-D estimation (and prediction) module, and the route choice model parameters.

Route Choice Parameters

The key calibration parameters for the route choice model are:

- Parameters in the path choice set generation algorithm
- Parameters in the path utility specification
- Path-size exponent γ

O-D Estimation (and Prediction) Parameters

The key calibration parameters in the O-D estimation (and prediction) module are:

- The historical database of O-D flows, $\mathbf{x_h^H}$
- The variance-covariance matrix $\mathbf{V_h}$ associated with indirect measurement errors
- The variance-covariance matrix $\mathbf{W_h}$ associated with direct measurement errors
- The matrices $\mathbf{f_h^p}$ of autoregressive factors.

For further details, the reader is referred to Balakrishna [3] and Ashok [2].

3.3 Proposed Calibration Methodology

Being comprised of a demand microsimulation model and a mesoscopic supply simulation model, a holistic calibration of the DynaMIT system would need to proceed in a two-stage iterative loop. We reiterate that the calibration methodology we propose will focus solely on the calibration of the supply simulator; it is assumed that the demand simulator has been adequately calibrated separately.

In proposing a calibration methodology of our own, we would like to use the following ideas encountered in the literature review:

- Calibration of speed-density relationships for individual segments using the curve-fitting approach for loop detector data, as done in Van Aerde and Rakha [1].
- Classification of sgements into categories, with a unique speed-density relationship for each category rather than for each individual segment (Hellinga [15]).
- Performing calibration in the least squares sense of matching model output with observed data (Wall et al [38]), or better still, using a weighted least squares objective function to match simulated values with field data, as outlined in Jayakrishnan et al [16].

These ideas motivate us to carry out the calibration exercise in three different, sequential stages at three correspondingly different levels of aggregation.

3.3.1 Three-stage Calibration

As shown in Figure 3-2, the calibration of the supply simulator for a large-sized real network is done at the following levels:

disaggregate level This involves calibration of the speed-density relationships and capacities at the level of each individual segment. The capacities of the inter-

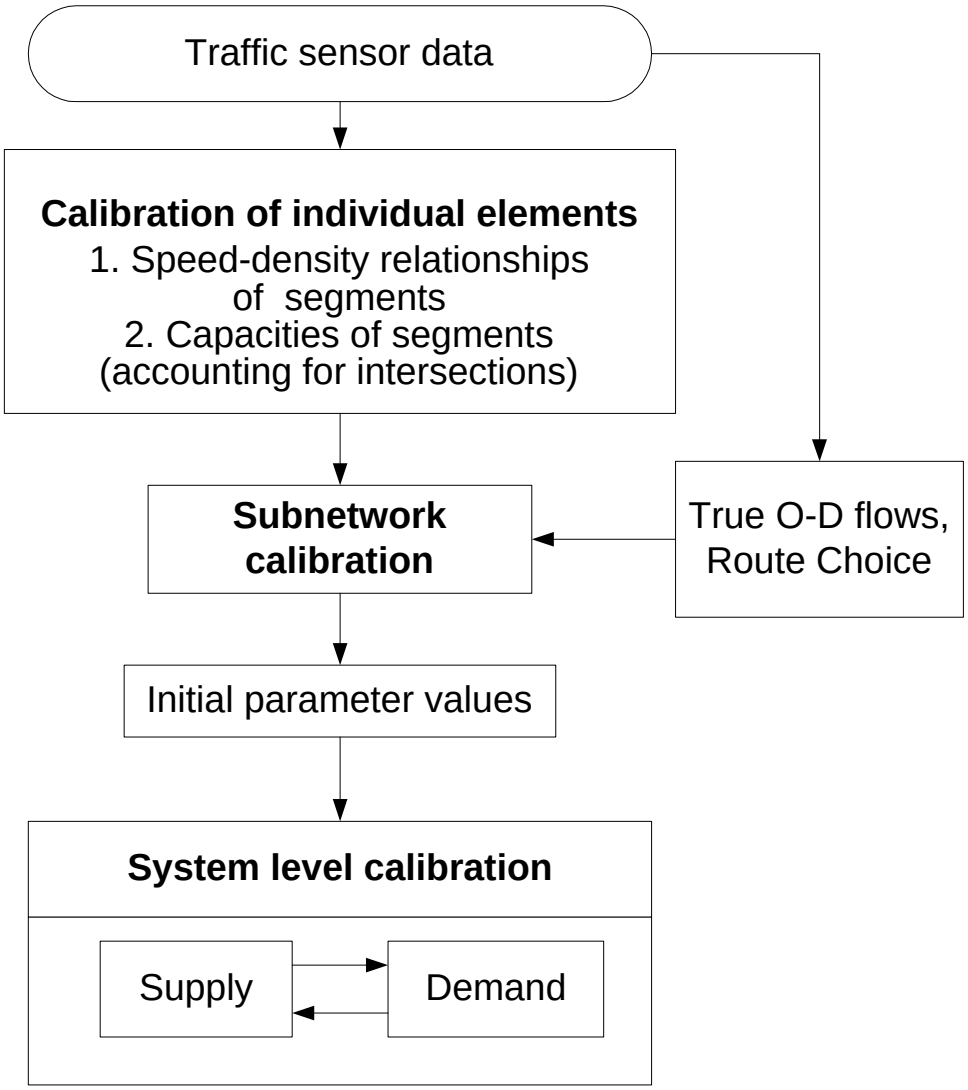

Figure 3-2: Three-stage Calibration Methodology

sections are also individually calibrated. Such a calibration assumes that there is no interaction between contiguous segments.

sub-network level This involves calibration of a sub-network where origin-destination flows can be accurately estimated from the sensor counts and there are very limited or no route choice possibilities. However, the subnetwork should consist of links and intersections with several interactions. Using such a sub-network enables us to carry out the simulation without the errors inherent in demand estimation in the presence of route choice.

The schematic in Figure 3-3 outlines such a sub-network calibration.

entire network level Finally, the calibration is done at the level of the entire network. This takes into account all the interactions that come into play between the various segments. Such a calibration takes care of the stochasticities and errors arising out of the demand estimation.

3.3.2 The Tool for Calibration

Before deciding upon the technique that needs to be used to tackle the calibration of the supply simulator, we need to emphasize the two most vital characteristics of this problem:

Noisy Output Like most simulation models, the supply simulator in DynaMIT too is a stochastic model[1] that uses random numbers to generate a sequence of stochastic values. It is therefore very important to realize that each simulation result is only a single point in a wide distribution of values. In fact, studies of NETSIM have indicated that misleading results will be obtained if the variability of the simulation results is ignored (Benekohal et al [5]).

[1]Strictly speaking, the variation in the simulated flows is due to the stochasticity in the departure times and less of an attribute of the supply simulator models

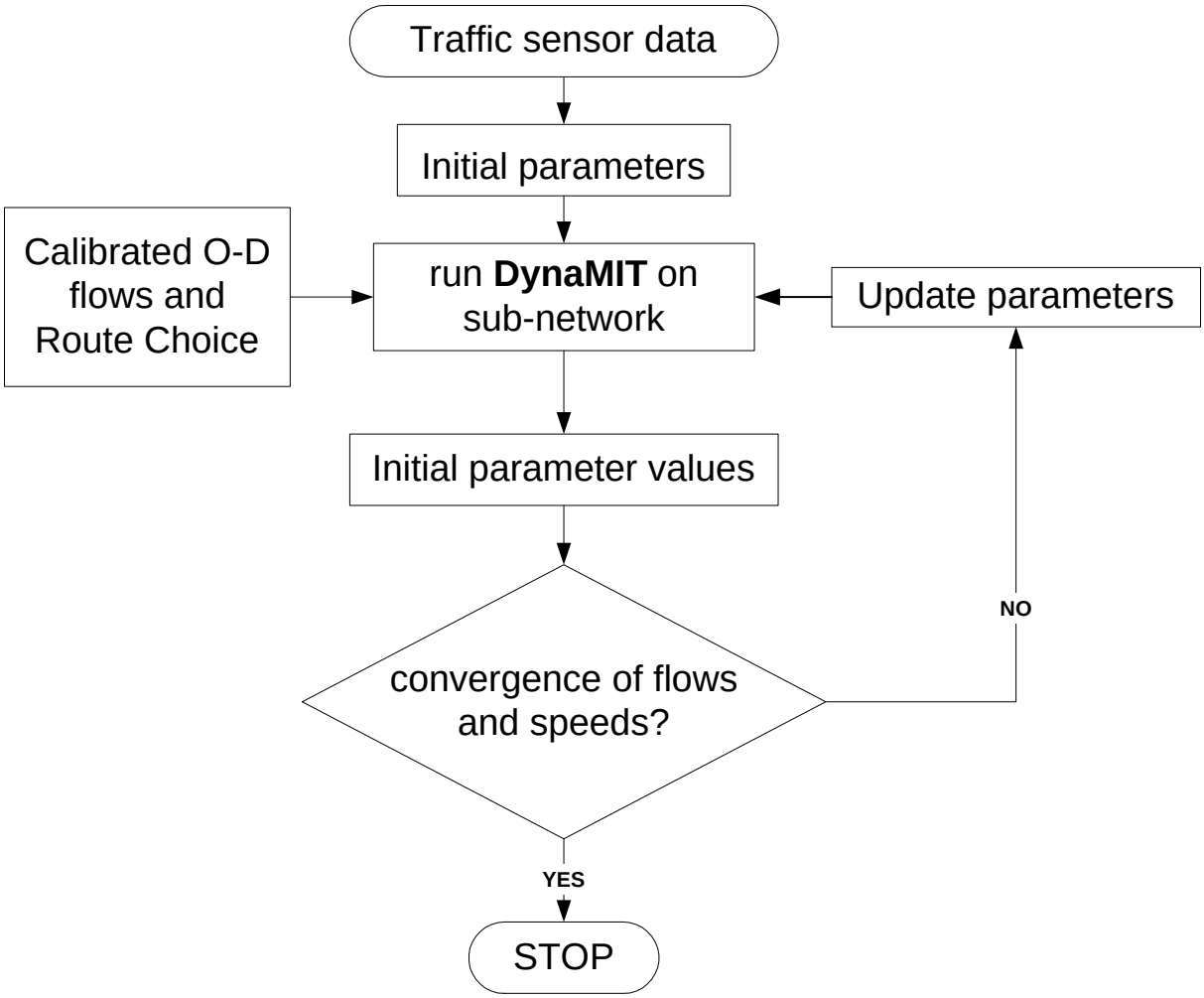

Figure 3-3: Sub-network Calibration

Figure 3-4 underlines this important facet of the supply simulator – the graphs represent simulated flows of two sensors in the Irvine study network, as recorded over four simulations.

Problem Dimension A traffic simulation network usually has a very large number (typically in the hundreds) of segments. Calibration of the entire study network requires the calibration of the 7-tuple

$\{u_f,\ k_{jam},\ \alpha,\ \beta,\ \texttt{capacity},\ u_{min},\ k_0\}$

for each and every segment in the network – a very large hyperspace.

The supply simulator calibration problem thereby lends itself to mathematical programming techniques; more specifically, we need to use a stochastic optimization technique. In the following chapter, we examine some of the stochastic optimization techniques that can be used for simulation optimization.

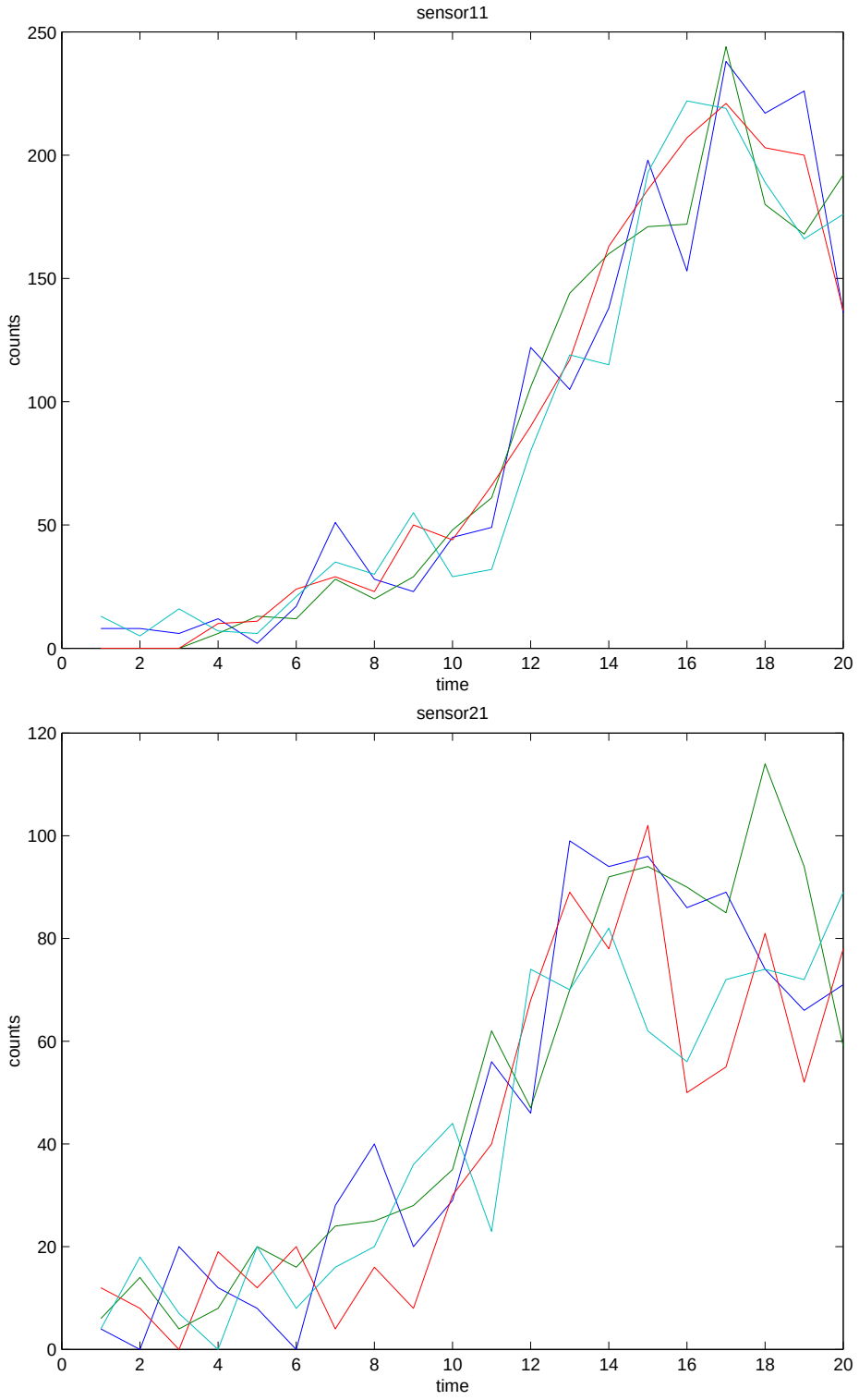

Figure 3-4: Stochasticity in the Supply Simulator Output

Chapter 4

Stochastic Optimization

The most commonly used optimization techniques – linear, nonlinear and (mixed) integer programming – require an explicit objective function. The stochastic context of simulation, however, defies analytical tractability, and thereby precludes the direct use of the expanse of optimization methods suited for solving deterministic problems. The reader is referred to Law and Kelton [20] and Andradottir [4] for recent literature on simulation optimization. Fu [12] provides an excellent exposition of simulation optimization. This chapter discusses the various methods of stochastic optimization.

4.1 Stochastic Optimization of Simulation Systems

Stochastic optimization refers to optimization of systems that produce output with inherent system noise. A simulation optimization problem is one where the objective function and some constraint(s) are implicit, stochastic functions of the decision parameters of the system, and as such, can be evaluated only by computer simulation.

4.1.1 Issues

Absence of analytical expressions of the objective functions and/or constraints eliminates the possibility of differentiation and exact calculation of local gradients. Also,

stochasticity means that we cannot decide upon the best point in the sample space by evaluating the objective function at the points of interest merely once. Furthermore, running computer-simulation programs is far more time-intensive than evaluating analytical functions and there is a great premium on the efficiency of the optimization algorithms. Lastly, interfacing generic optimization toolboxes with the simulation package is no mean feat, especially so since the simulation modeling language is, more often than not, different from the optimization programming language.

4.1.2 Notation

To discuss the various methods of stochastic optimization, the notation we use assumes that the optimization problem is of the form

$$\min_{\theta \epsilon \Theta} l(\theta),$$

where

θ = the controllable set of parameters,
Θ = the constraint set on θ,
$l(\theta)$ = the objective function.

The objective function is an expectation , i.e.,

$$l(\theta) = E[L(\theta, \omega)],$$

where

ω = a sample path (or simulation replication),
$L(\theta, \omega)$ = corresponding sample performance estimate.

4.1.3 Classification

The different approaches of searching for an optimum may be classified into three major categories: path search methods, pattern search methods, and random methods. Path search methods have been the most widely studied approach for solving the simulation optimization problem, and the section devoted to the same is more detailed than the others.

4.2 Path search methods

Path search methods involve the estimation of a direction to move from the current vector of parameters to an improved point in the feasible region. The most common direction of movement is the gradient.

4.2.1 Response Surface Methodology

Response Surface Methodology (RSM) is a widely studied approach that tries to locally fit a polynomial response surface model to a set of sample observations in a particular region of the search space. The choice of these observation points is made by experimental design techniques, and the polynomial that is fitted is usually of first or second order. RSM is applied to the problem of simulation optimization either in the form of 1. metamodels, or 2. sequential procedures.

Sequential RSM involves fitting a linear regression model around a given parameter setting, defining a linear response surface from the ordinary least squares estimate, and moving in the direction of the gradient until the linear response surface stops improving the response function value. Phase II is then implemented by fitting a quadratic model to the response, and the same procedure as Phase I, namely moving in the gradient direction, is performed until the magnitude of the gradient becomes sufficiently close to zero. If need be, higher degree regression models can then be utilized in analogous manner. Smith has developed an automatic optimum-seeking

program based on RSM that can be interfaced with independently built simulation models. The reader is referred to Safizadeh [32] for more details.

Using a metamodel simply means two separate problems of estimation and optimization. After choosing points θ based on a statistical design of experiments methodology, the outputs at these points are used to fit a response curve or metamodel. This functional relationship between the output and the simulation paramters is then optimized using deterministic procedures. An extensive discussion of metamodels is given in Kleijnen [18].

RSM has the advantages of being based on sound statistical theory that is easily understood, and of ease of implementation. The disadvantage is that a large number of simulation runs are needed. Also, it has been shown to be inadequate for complex functions with sharp ridges and flat valleys.

4.2.2 Stochastic Approximation

Stochastic Approximation (SA) is another path search method. The basic underlying principle is that the optimization problem can be solved by finding the zero of the gradient. The general form of the stochastic algorithm is given by

$$\theta_{n+1} = \Pi_\Theta(\theta_n - a_n \widehat{\nabla} L(\theta_n))$$

where

θ_n	= parameter value at the beginning of iteration n,
$\widehat{\nabla} L(\theta_n)$	= estimate of $\nabla L(\theta_n)$ from iteration n ,
$\{a_n\}$	= positive sequence of step-size multipliers (gain sequence),
Π_Θ	= projection onto Θ, the constraint set on θ.

When an unbiased estimator is used for $\nabla L(\theta_n)$, the algorithm is called a Robbins-Monro algorithm [31] and when a finite difference estimate is used, the algorithm is called a Kiefer-Wolfowitz algortithm [17]. The optimal asymptotic convergence rates

are $p^{-1/2}$ for the Robbins-Monro algorithm, and $p^{-1/3}$ ($p^{-1/4}$) for the Kiefer-Wolfowitz symmetric (one-sided) differences[1] (Pflug [29]).

Convergence The pivotal question to investigate is whether the iterate θ_n converges to θ^* as n gets large. The rich convergence theory developed for SA over the years makes it possible to formally establish convergence[2] for any stochastic optimization algorithm of the SA form.

One of the many sufficient conditions given for a.s. convergence involves the definition of an underlying ordinary differential equation that roughly emulates the SA algorithm for large n and disappearing random effects. It turns out that the convergence properties of this *deterministic* differential equation are closely related to the *stochastic* convergence properties of the general SA equation.

The most famous of the convergence conditions for SA are based upon the gain sequence $\{a_n\}$. The conditions perform a tightrope act of not damping out the noise effects as we near the solution ($a_n \to 0$) and, at the same time, avoiding premature (false) convergence of the algorithm ($\sum_{n=0}^{\infty} a_n = \infty$). The scaled harmonic sequence $\{\frac{a}{(n+1)}\}$, $a > 0$, is the best-known example of a gain sequence that satisfies the gain conditions (and, is an optimal choice with respect to the theoretical rate of convergence, although in practice other decay rates may be superior in finite samples). Usually some numerical analysis needs to be done before deciding upon the best scale factor for the gain sequence decay. Other important convergence criteria relate to the smoothness of the objective function, the magnitude of the noise, and the initial position.

Also of importance is the probability distribution of the iterate which happens to be a random vector in a stochastic setting. Having knowledge of the distribution provides key insight into two major aspects: (1) error bounds for the iterate, and (2)

[1] p is the dimension of the parameter vector.

[2] Convergence is in the probabilistic sense in the stochastic context. The most common form of convergence established for SA is in the *almost sure* (a.s.) sense.

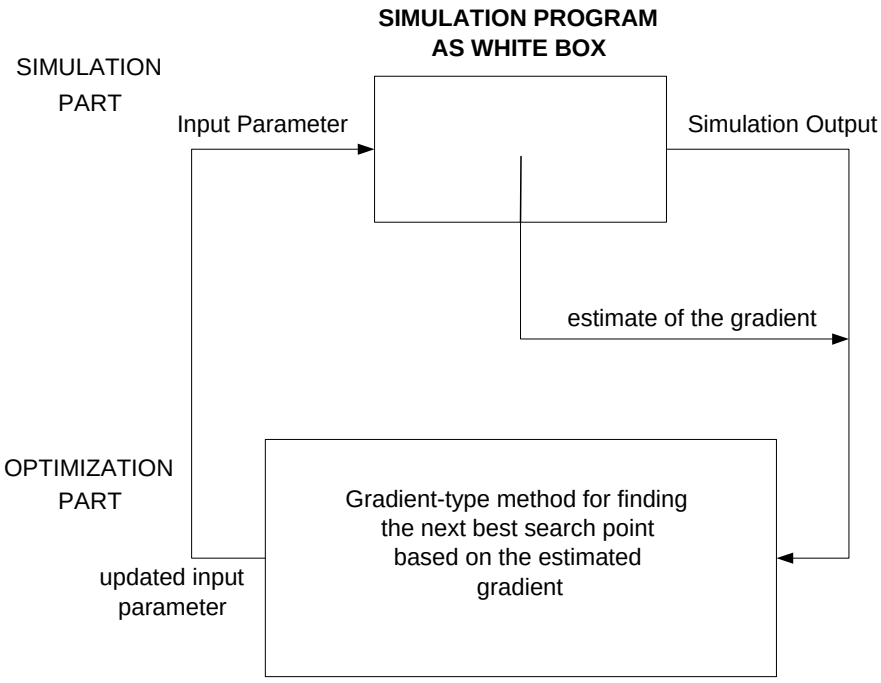

Figure 4-1: The White-Box Approach to Simulation Optimization

guidance in the choice of the optimal gain sequence $\{a_n\}$ so as to minimize the likely deviation of $\hat{\theta}_n$ from θ^*.

"White Box" Approaches Approaches that provide an unbiased estimator of the gradient rely on some knowledge of the system being studied, and include techniques such as perturbation analysis, the likelihood ratio/score function method, and weak derivatives. Since these approaches require knowledge of the underlying system, they are referred to as "white box" approaches to simulation optimization. Figure 4-1 shows the white-box approach.

"Black Box" Approaches Figure 4-2 shows the black-box approach. When the simulator needs to be treated as a black box, the usual approach is to use finite differences, either one-sided (FD) or symmetrical (SD), given respectively by:

$$[L(\theta_n + c_n e_i, \omega_n^{i+}) - L(\theta_n, \omega_n)]/c_n,$$

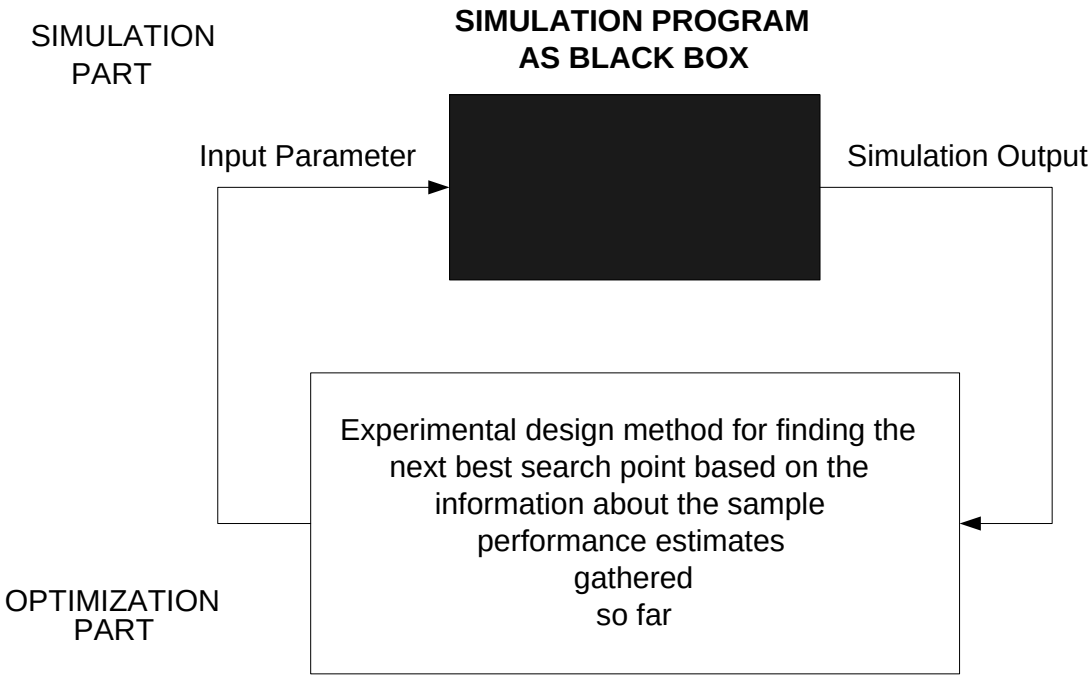

Figure 4-2: The Black-Box Approach to Simulation Optimization

$$[L(\theta_n + c_n e_i, \omega_n^{i+}) - L(\theta_n - c_n e_i, \omega_n^{i-})]/2c_n,$$

where

e_i = unit vector in the ith direction,

$\{c_n\}$ = positive sequence converging to zero,

ω_n^{i+}, ω_n^{i-} = pair of sample paths (simulation replications) used for the ith component of the nth iterate of the algorithm,

ω_n = original sample path (replication) used to estimate the performance measure itself.

In both cases, the estimate requires $O(p)$ (p is the dimension of the parameter vector) simulation replications.

Newer approaches that treat the simulator as a black box include harmonic differences based on frequency domain experimentation and simultaneous perturbations. A frequency domain experiment is one where selected input parameters are oscillated

sinusoidally at different frequencies during one long simulation run. The output variable values are then subjected to Fourier analysis. If the output variable is sensitive to an input parameter, the sinusoidal oscillation of that parameter should induce corresponding (amplified) variations in the output response.

The Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm avoids having to perform $O(p)$ simulation replications by perturbing all components of θ_n simultaneously, but randomly. Each component of $\widehat{\nabla} L(\theta_n)$ is formed from a ratio involving the individual components in the perturbation vector and the difference in the two corresponding measurements.

Let Δ_n be an i.i.d. vector sequence of perturbations of i.i.d. components $\{(\Delta_n)_i, i = 1, ..., p\}$ symmetrically distributed about 0 with $E|(\Delta_n)_i^{-2}|$ uniformly bounded. The best choice is a sequence of i.i.d. symmetric Bernoulli random variables, i.e., $P((\Delta_n)_i = 1) = P((\Delta_n)_i = -1) = 0.5$. The simultaneous perturbation estimator is then given by

$$[L(\theta_n + c_n e_i, \omega_n^{i+}) - L(\theta_n - c_n e_i, \omega_n^{i-})]/2c_n(\Delta_n),$$

where the symbols have their usual meanings. Again, the method of common random numbers takes $\omega_n^{i+} = \omega_n^{i-} = \omega_n$. The key point to be noted is that the estimate $L(\theta, \omega)$ is computationally expensive relative to the generation of Δ_n. The symmetric difference estimator requires two different estimates in the numerator for each parameter dimension and thus requires $2p$ simulation runs; the one-sided difference estimator similarly requires p+1 simulation runs. On the other hand, the simultaneous perturbation estimator requires only **2** simulations, irrespective of the dimensionality of the parameter vector, an order of magnitude savings in simulation replications.

Spall [37] reports: "Under reasonably general conditions, Kiefer-Wolfowitz finite-difference-based SA (FDSA) and SPSA achieve the same level of statistical accuracy for a given number of iterations even though SPSA uses **p** times fewer function evaluations than FDSA (since each gradient approximation uses only $1/p$ the number of

function evaluations)." This theoretical result has been confirmed in many numerical studies, even in cases where p is of the order of a few hundred or thousand.

Spall [35] mentions that it is usually helpful to average several gradient approximations since noisy loss function measurements can produce very noisy gradient estimates.

Table 4.1 provides a brief summary of gradient estimation approaches for stochastic approximation.

Approach	**Number of Simulations**	**Key Features**	**Disadvantages**
Infinitesimal PA	1	highly efficient, easy to implement	limited applicability
other PA	usually > 1	model-specific implementations	difficult to apply
LR/SF	1	requires model input distributions	high variance
SD	$2p$	widely applicable, model-free	generally noisier
FD	$p + 1$	widely applicable, model-free	generally noisier
SD	2	widely applicable, model-free	generally noisier

Table 4.1: Gradient Estimation Approaches for Stochastic Approximation (Fu [11])

4.3 Pattern search methods

Pattern search methods require neither gradient estimates nor randomization procedures, but use some characteristic or pattern of the observations to obtain an improved point. The implicit assumption of inputs and outputs being continuous made in gradient estimation methods is not necessary in pattern search methods.

4.3.1 Hooke and Jeeves Method

The Hooke and Jeeves method is based on the idea of moving in the direction that has produced a favorable change in the optimal value. The pattern from which previous improving changes have been made is used to obtain a better parameter vector, and eventually, the optimal parameter vector.

The method starts out with a set of incremental values for each parameter. Starting at an initial base point, it checks if an incremental change in the first parameter yields an improved response value. The resulting improved setting becomes the new intermediate base point. This is repeated for each of the p paranmeters until a new setting is obtained. The method then moves directly from the initial base point towards, and through the new setting. This procedure is continued until optimal changes cannot be made with the given incremental values. Then the incremental values are decreased and the procedure is repeated from the beginning. When the incremental values reach a prespecified tolerance, the procedure terminates with the optimal parameter setting.

4.3.2 Nelder and Mead (Simplex Search) Method

This method starts out with a set of p +1 parameter vectors in R^p. Then, by comparing their objective function values, the worst point is eliminated and replaced by the centroid of all these p +1 parameter vectors. The resulting simplex grows or shrinks depending, depending on the objective function value for the new vector. The procedure continues until no more improvements can be made by eliminating the worst-valued vector and the resulting simplex is small.

4.3.3 Box Complex Method

The Complex Search is an extension of the Nelder-Mead Simplex Search described above. The search starts with the evaluation of points in a simplex consisting of p +1 vertices in the feasible region. It proceeds by continuously dropping the worst point from among the points in the simplex and adding a new point determined by the reflection of this worst point through the centroid of the p others.

The major issue in applying this to simulation models is that stochasticity may result in an apparently worst point being discarded, when it was actually one of the better points, and thus take the search away from the optimum region.

4.4 Random methods

Random methods are those that use a random approach to select parameter settings, with the hope of obtaining an improving, and eventually, optimal setting. The major problem with these methods is that they are slow to converge (if they converge at all) to an optimum. Typically, previous information is not used at each iteration, and a huge number of simulation runs is needed.

Random search methods are best-suited for problems where the feasible region Θ is discrete. Approaches worthy of mention include ranking, selection, and multiple comparison methods, methods for solving the "multi-armed bandit" problem, and learning automata procedures.

Andradottir has developed two random search methods for discrete parameter simulation optimization; the value of the objective function is estimated (via simulation) at two neighboring feasible points at each iteration of both these methods, and the better point is passed onto the next iteration. The most-visited feasible point is used to estimate the optimal solution.

Yan and Mukai have proposed a random search algorithm called the stochastic ruler algorithm. Gong, Ho and Zhai have used the above three authors' ideas to analyze a method known as the stochastic comparison method. But the most widely used technique in random search methods appears to be simulated annealing, described below.

Simulated Annealing

Simulated annealing (SAN) was originally developed for discrete optimization problems, but has recently been extended for applicability to continuous optimization problems. The main virtue of this technique is that it is capable of traversing multiple local minima on its way to the global minimum. Further, there is no need to assume the existence of a gradient, much less compute it.

SAN attempts to capture mathematically the process of controlled cooling associ-

ated with physical processes, the aim being to reach the lowest value of the objective function in the face of multiple local minima. As with the physical cooling process, where temporary higher-energy states may be attained in the course of the realignment of molecules, SAN allows for temporary increases in the objective function as the learning process captures the information necessary to reach the global minimum.

SAN derives the above-mentioned property from the Boltzmann (or Gibbs) probability distribution of statistical mechanics, describing the probability of a system having a particular energy state:

$$P(energy = x) = c_T exp(-\frac{x}{c_b T})$$

where

$c_T > 0$	=a normalizing constant,
$c_b > 0$	=the Boltzmann constant,
T	=the temperature of the system.

Note that at high temperatures, the system is more likely to be in a high-energy state than at low temperatures; however, even at lower temperatures, the system has a nonzero probability of reaching a high-energy state. Thus the SAN process sometimes goes uphill, but the probability of this decreases as the temperature is lowered. Thus, during the early iterations, when the temperature is high, there is a probability of getting out of a local minimum in favor of finding a global minimum.

The general sequence of steps is:

1. Choose an initial temperature T and a set of current parameters θ_{curr}; determine $L(\theta_{curr})$.

2. Randomly determine a new value of θ, θ_{new}, that is "close" to the current value θ_{curr}, and determine $L(\theta_{new})$.

3. Let $\delta = L(\theta_{new}) - L(\theta_{curr})$. Accept θ_{new} if $\delta < 0$. Alternatively, if $\delta \geq 0$, accept the new point only if a uniform (0,1) random variable U (generated by Monte Carlo) satisfies $U \leq exp(-\delta/T)$.

4. Repeat steps 2 and 3 for some period until either the *budget* of function evaluations allocated for that T has been used or the system reaches some state of equilibrium.

5. Lower T according to the annealing schedule, and return to step 2. Continue the process until the total budget for function evaluations has been used or some indication of convergence is satisfied.

The specifics of the implementation of these five steps can vary greatly. Aside from the variations in the implementation, SAN is critically dependent on the specific values of various algorithm parameters and decision criteria. In particular, these include the initial temperature T, the specific distribution of the perturbation vector from which the θ_{new} is generated, the cooling schedule as per which the temperature T is lowered, and the criterion for determining when to lower the temperature (e.g., the *budget* of function evaluations for a given T).

SAN has been found wanting in dealing with noisy function measurements $y(.)$, especially since the value of δ can be altered from its underlying true value according to the level of noise in the function measurements. Empirical evidence points to the fact that even a modest amount of noise will frequently alter the sign of δ , which is likely to influence the selection/rejection of the new point θ_{new}. The obvious way to cope with this difficulty is to average a lot many function evaluations $y(.)$ at each θ value; however, this dramatically increases the optimization expense. An alternative way is to alter the acceptance criteria $\delta < 0$ or $\delta \geq 0$ with $\delta < \rho\sigma$ or $\delta \geq \rho\sigma$, where σ is the function measurement noise standard deviation and ρ the number of multiples of the standard deviation that will be tolerated. Changing the unconditional acceptance criterion from $\delta < 0$ to $\delta < \rho\sigma$ allows for a greater number of cases where

$L(\theta_{new}) < L(\theta_{curr})$ even though $y(\theta_{new}) \geq y(\theta_{curr})$ due to the noise. With $\rho \approx 2$, one can be sure (through the Chebyshev inequality of probability) that most such cases will be caught, although at the expense of accepting some θ_{new} values that increase the objective function.

4.5 Summary

In this chapter, we made an attempt to examine stochastic optimization algorithms that could be applied to the calibration of the supply simulator. In particular, we are interested in algorithms that do not require too many simulation replications and can be easily interfaced with mathematical programming toolboxes for easy tractability of the large hyperspace domain. It would also be preferable to have an algorithm with a certificate of convergence. Lastly, in a stochastic problem such as simulation optimization, we would like to obviate the calculation of gradients, which we expect to be noisier than the noisy function estimates on which they are based.

None of the stochastic optimization algorithms we reviewed meet **all** of the above requirements. We therefore choose to employ one pattern search algorithm, namely the Box Complex algorithmm, and one path search algorithm, namely the SPSA algorithm for the calibration of the supply simulator.

The Box Complex algorithm, while not requiring the calculation of gradients, also has the advantage of being a tried and trusted algorithm in traffic engineering problems. Also, it is far easier to apply than the Hooke and Jeeves pattern search algorithm, and finally, unlike the Nelder-Mead (Simplex Search) method – which may result in an improvement or a worsening of the objective function, the Box Complex algorithm has been empirically proven to improve the worst objective function values at least in the first few iterations.

The SPSA algorithm scores over the Box Complex algorithm in that it is guaranteed to converge with a well-chosen gain sequence. It has been successfully applied

to problems of queuing systems, industrial quality improvement, pattern recognition, neural network training, adaptive control of dynamic systems, statistical model parameter estimation and fault detection, sensor placement and configuration, and vehicle traffic management. The SPSA algorithm also outshines other stochastic approximation algorithms with its wide applicability **and** the fact that one needs to perform only two simulation replications to obtain a single gradient estimate, irrespective of the dimensionality of the parameter vector.

Chapter 5

Case Studies

So far in this thesis, we reviewed literature related to the calibration of traffic simulation models, used the same to outline our own methodology customized to the calibration of the mesoscopic supply simulator on a large-sized network, and then examined various stochastic optimization algorithms to narrow in on the Box Complex and SPSA algorithms. This chapter focuses on the application of the three-stage calibration methodology and the above two stochastic approximation algorithms to an actual large-sized network in Irvine, California.

We begin with a description of the network and the surveillance data, and then proceed to present results from each stage of the three-stage calibration process. We conclude with a discussion on the relative performance of the two algorithms.

5.1 The Irvine Data

The dataset used in this research was collected from Irvine in Orange County, California. A brief description of the network and the surveillance data follows.

5.1.1 Network Description

Figure 5-1 shows a map of the area from which the study network has been extracted. The network comprises three main freeways – the I-5 Interstate, the I-405 Interstate and State Route 133 – and a dense network of arterials. This area lies along the heavily traversed traffic corridor connecting Los Angeles and San Diego and attracts a varied mix of travelers owing to the presence of several schools, universities and an important regional airport.

Figure 5-2 shows the network as coded in MITSIM.

The network is represented as a set of 298 nodes connected by 618 directed links. These links are subdivided into 1373 segments to account for changing section geometry. Almost all of the 80 intersections within the network are signalized, and are controlled by vehicle-actuated signal logic. A high fraction of the signals along the primary arterials (Barranca Parkway, Alton Parkway and Irvine Center Drive) are co-ordinated to minimize the number of stops.

5.1.2 Data Description

The data was derived primarily from the following sources:

- PARAMICS network files
- O-D flows from OCTAM planning study
- Time-dependent detector data
- Signal timing and co-ordination plans

Network-specific information was extracted mainly from a set of input files created for the PARAMICS traffic simulator. This set included descriptions of network geometry, link and lane connectivity, sensor locations and signal phase timing plans.

The OCTAM planning study generated a static OD matrix covering 61 zones over the morning peak period. 655 primary OD pairs were extracted from the set of 61*60

Figure 5-1: Map of the Study Network Area

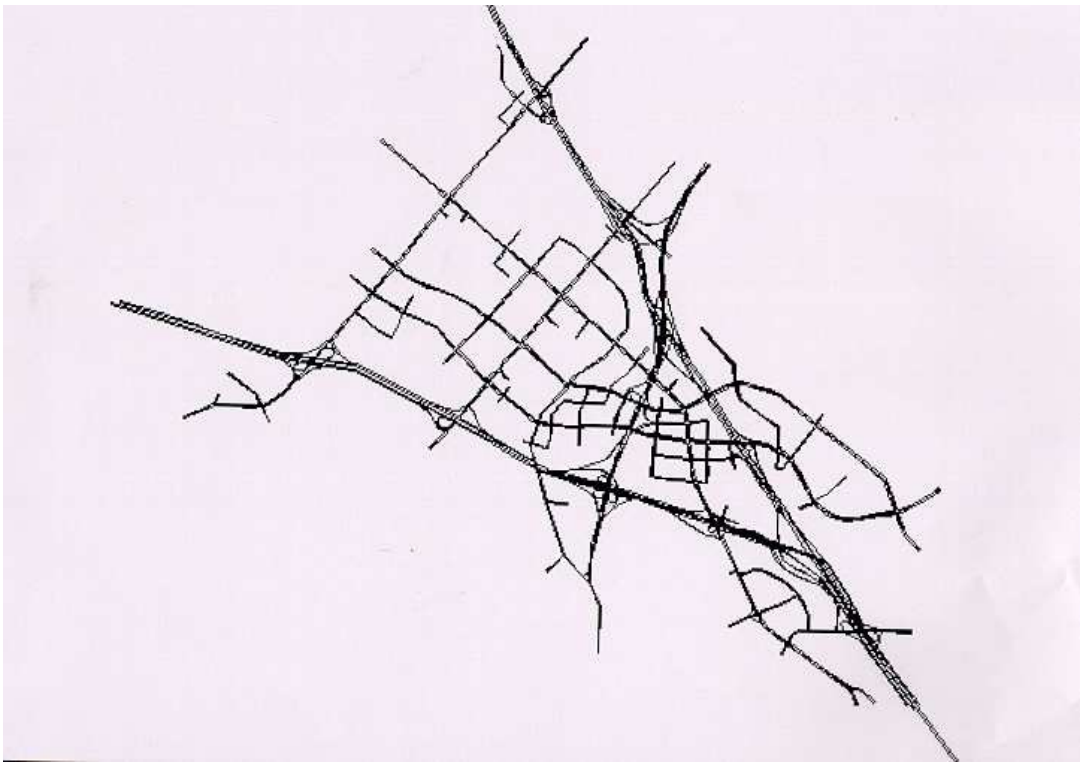

Figure 5-2: The Irvine Network

pairs. Figure 5-3 shows the primary OD pairs. The thickness of the lines between the nodes is indicative of the magnitude of the demand.

Time-varying freeway and arterial data recorded over 5 working days were available from the site. The data consisted of counts and occupancies measured by lane-specific sensors on freeway links and lane-group-specific sensors on arterial links. The freeway sensors' reporting interval was 30 seconds while the arterial sensors aggregated data every 5 minutes. Only 68 of the 225 sensors reported usable data; 30 of these were on freeway and ramp links while 38 were on arterial links.

Analysis of the temporal evolution of 5 days of sensor data indicated very little day-to-day variability in the data (Figure 5-4 and Figure 5-5). These data [1] were aggregated into 15-minute time slices for simulation purposes.

Signal timing and co-ordination charts from the City of Irvine specified the signal phasing, timing, actuation and coordination details.

5.2 The First Stage of Calibration

The Irvine network is a huge network with 1373 segments. Notwithstanding the fact that individual calibration for each of these segments would be an impractical task, the overruling constraint that obviates this possibility is the limited number of sensors. Of the 225 sensors in the network, only 68 reported usable data; furthermore, 9 of these 68 sensors were bad.

Also, densities were deduced from occupancy values, and the latter assumed one of only five discrete values. This meant that curve-fitting had to be carried out on a set of highly disjoint and clustered points.

The above practical considerations dictated that calibration of individual segments be performed only for representative network segments that afford a good set of data points for curve-fitting. In our case, we chose segments such as the mainline section of

[1] densities were calulated from occupancy values assuming a mean vehicle length of 5 meters

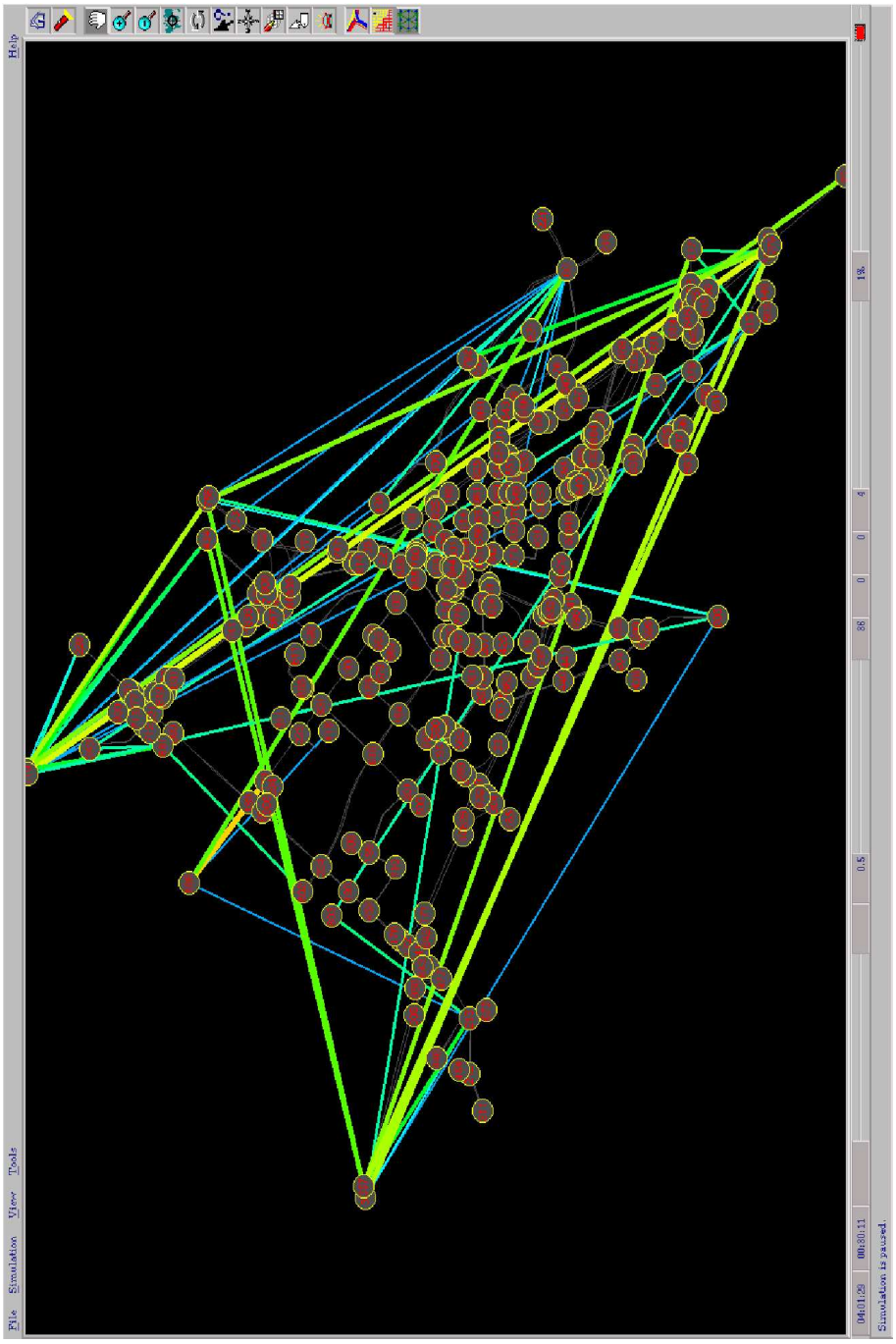

Figure 5-3: Primary OD Pairs in the Irvine Network

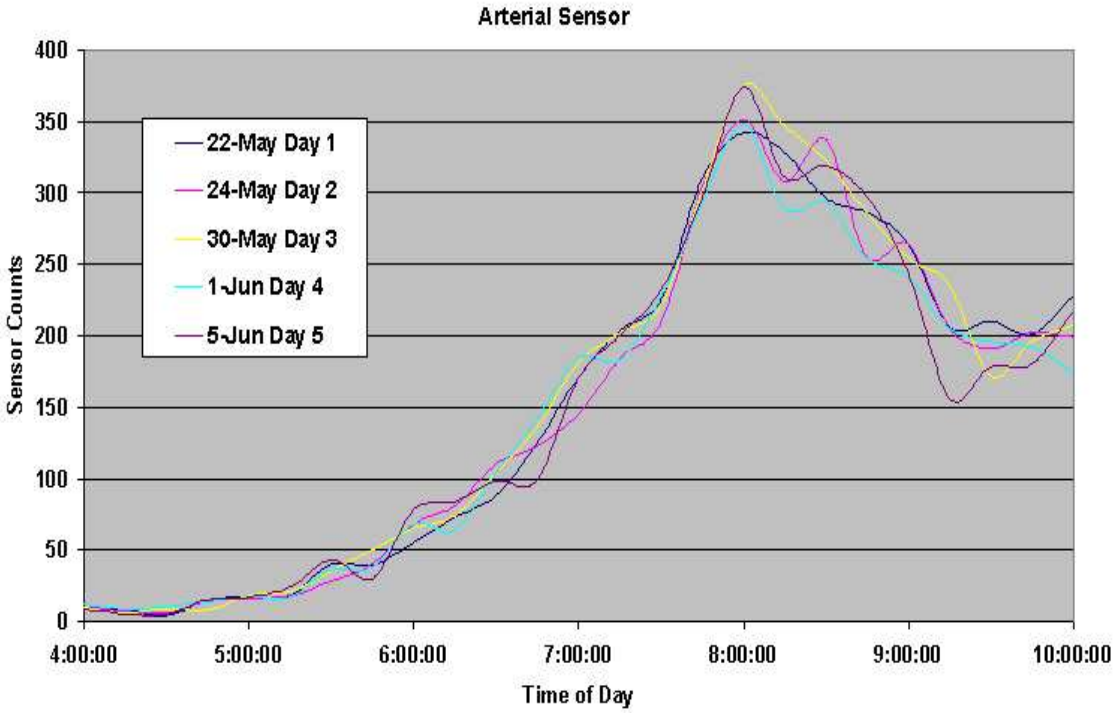

Figure 5-4: 5-day Variability in Arterial Sensor Counts

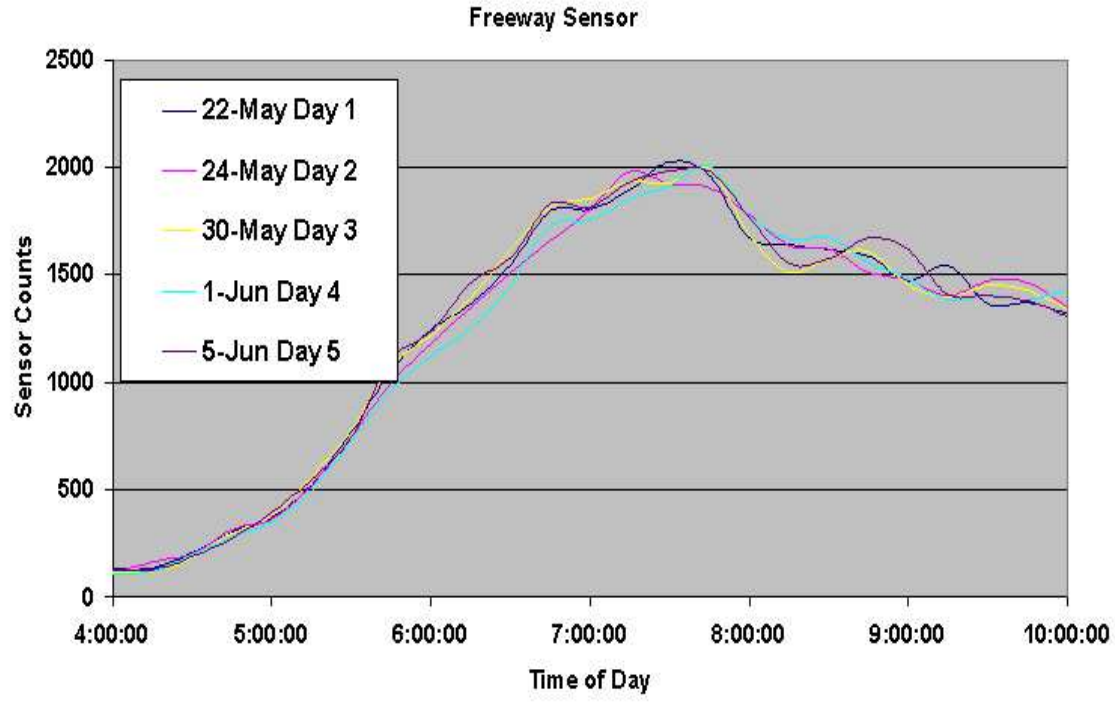

Figure 5-5: 5-day Variability in Freeway Sensor Counts

freeway, weaving section of freeway, arterial segment leading into an intersection, arterial segment leading out of an intersection, on-ramp, off-ramp etc. All the segments in the network can be classified into one of these representative categories.

Figure 5-6 shows a typical calibrated speed-density curve.

Table 5.1 shows the results of the calibration for different segment categories in the Irvine network.

Segment Type	u_f mph	k_{jam} veh/lane-m	α	β	u_{min} mph	k_0 veh/lane-m
Arterial next to an intersection	40	0.075	1.6	0.5	10	0.0009375
Off-ramp On-ramp	40	0.125	2	0.45	10	0.0125
Off-ramp 2 On-ramp 2	40	0.14375	2	0.4	10	0.0125
Arterial next to an intersection 2	45	0.05625	1.7	0.37	10	0.00125
Arterial next to an intersection 3	45	0.0875	1.38	0.348	10	0.0028125
Arterial next to an intersection 4	45	0.10625	2	0.3	10	0.0009375
Arterial	55	0.0625	1.5	0.4	10	0.0009375
Arterial 2	55	0.075	1.7	0.4	10	0.00125
Mainline section of freeway	70	0.10625	1.75	0.5	10	0.015625
Mainline section of freeway 2	70	0.11375	1.2	0.35	10	0.015625
Weaving section of freeway	70	0.12125	1.5	0.4	10	0.015625

Table 5.1: Results of Individual Segment Calibration

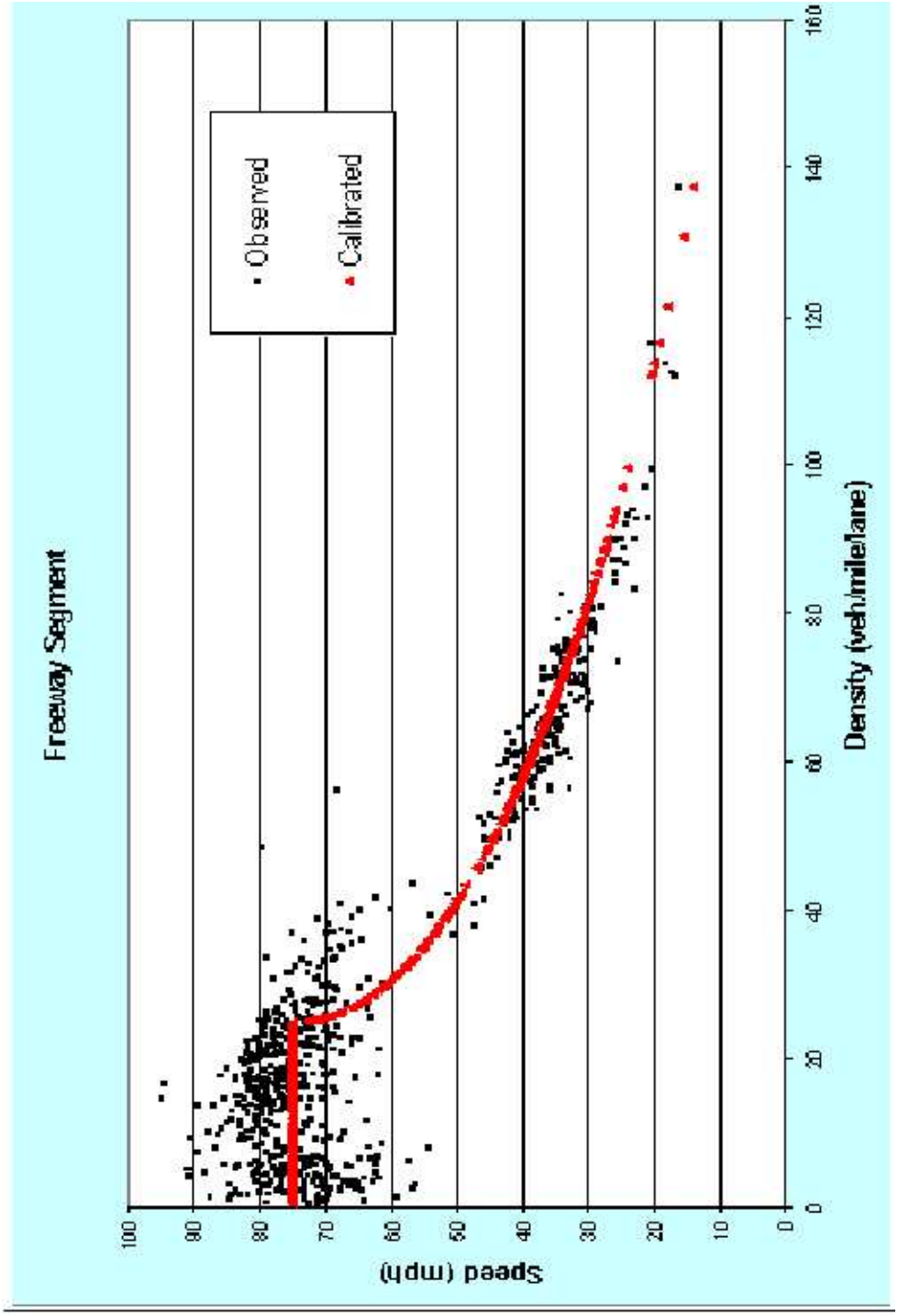

Figure 5-6: Typical Calibrated Speed-Density Curve

5.3 The Second Stage of Calibration

As was discussed in Chapter 3, the second stage of calibration involves a sub-network with a number of representative segments, and minimal or no route choice. Such a sub-network allows for accurate estimation of OD demands and enables us to simulate the traffic dynamics without OD estimation errors. Figure 5-7 shows the sub-network[2] that was chosen to carry out the second stage calibration.

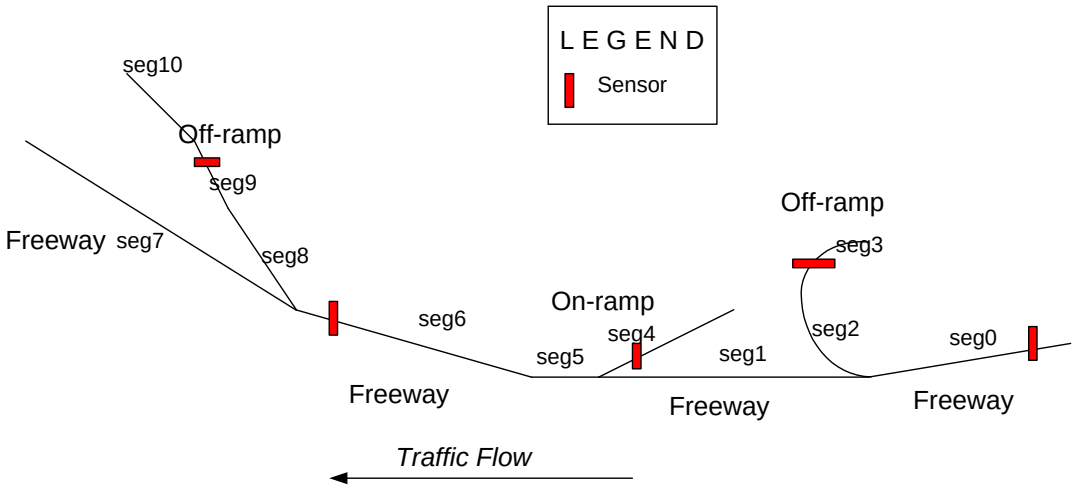

Figure 5-7: The SSC Network

The calibration of this highly linear network was easily carried out by trial and error and enabled us to easily verify the correctness of the speed-density parameters calibrated in the first stage. As shown in Table 5.2, the results of this calibration exercise are not very significantly different from the values obtained from the curve-fitting exercises in the first stage.

Figures 5-8 through 5-11 show the simulated sensor counts vis-a-vis the field counts for simulation performed on the SSC network for the 4:00 a.m. to 10:00 a.m. period.

Having achieved a very good match for the flows, we turned our attention to

[2]henceforth referred to as the SSC network

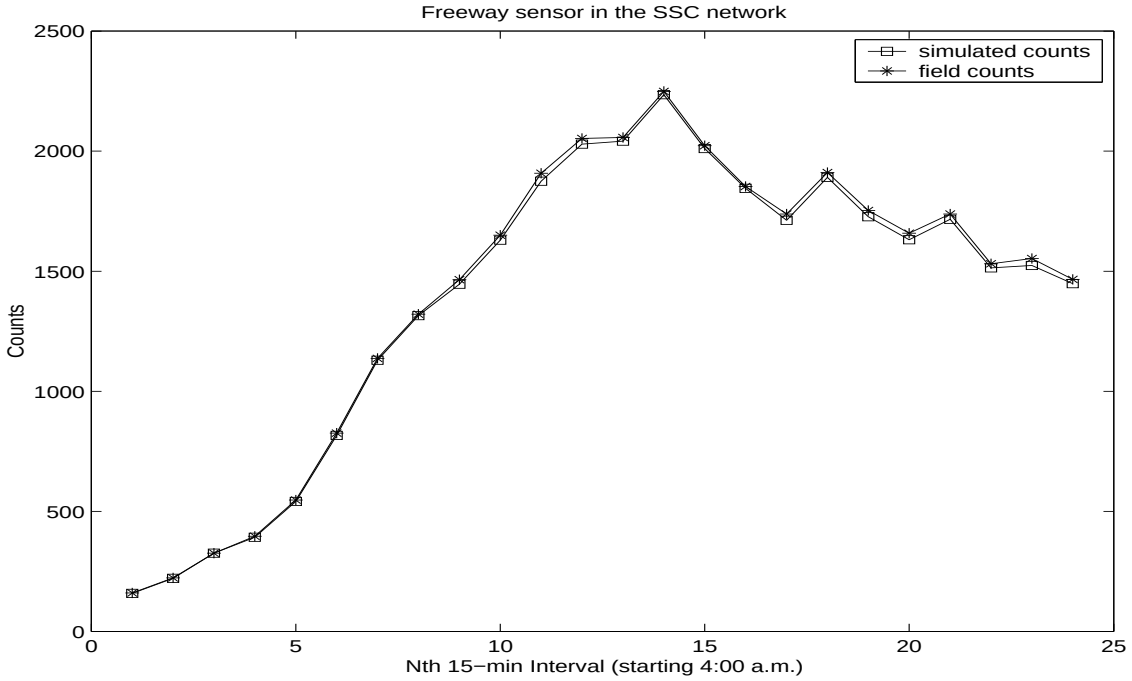

Figure 5-8: Freeway sensor counts on the SSC network

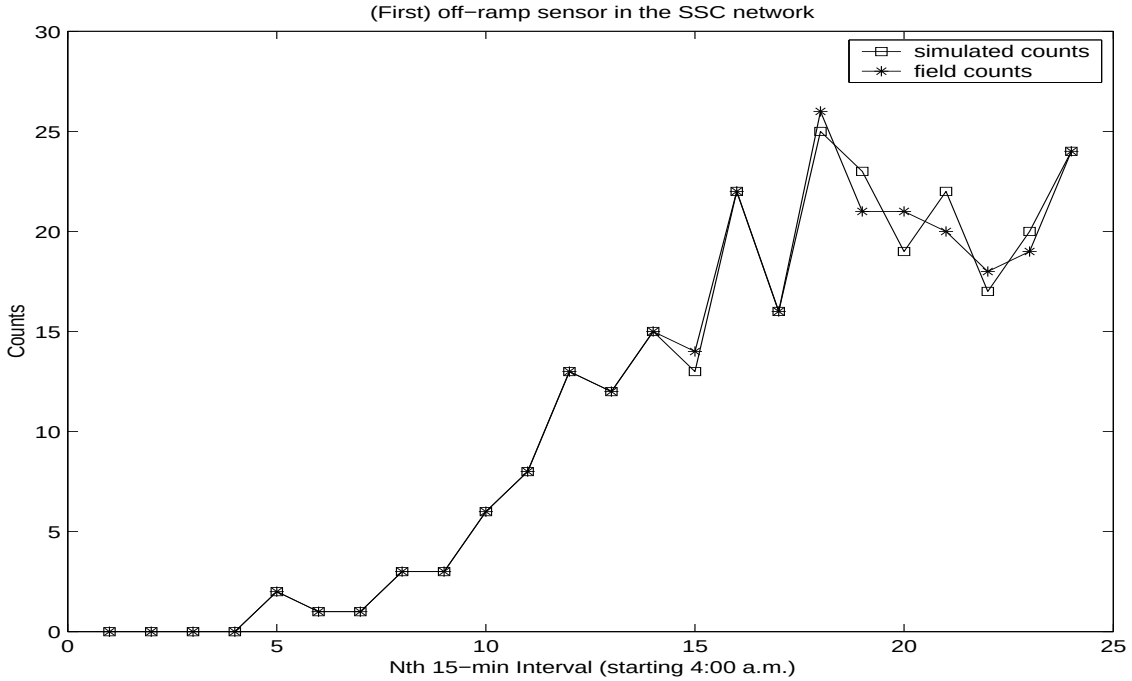

Figure 5-9: Off-ramp sensor counts on the SSC network

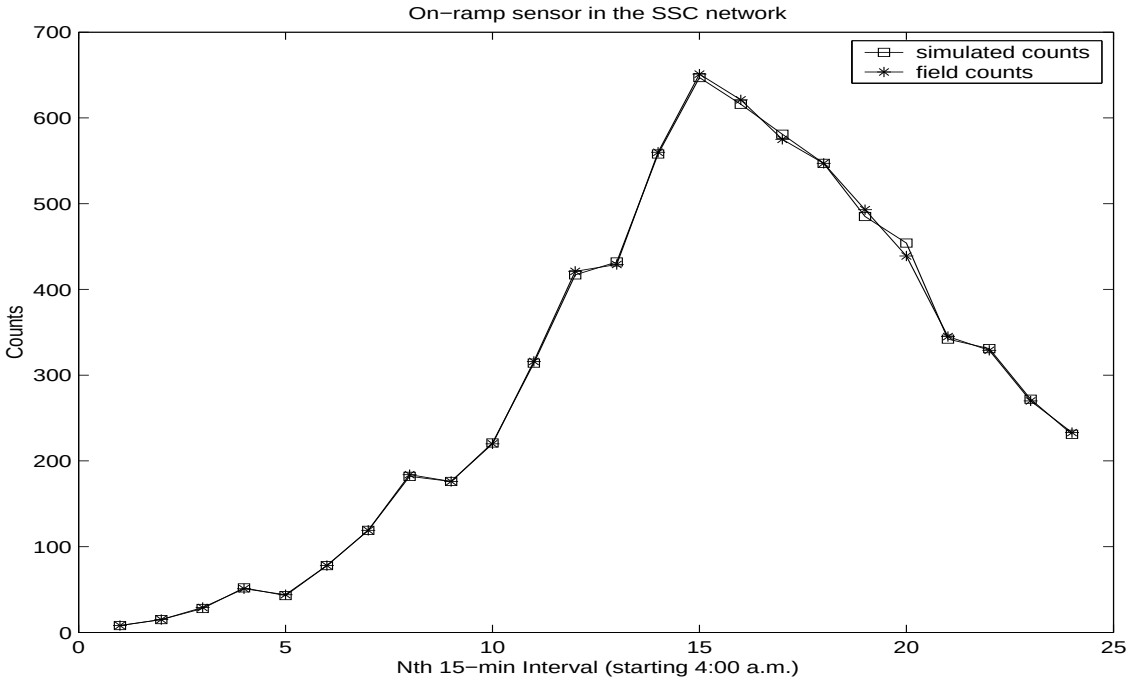

Figure 5-10: On-ramp sensor counts on the SSC network

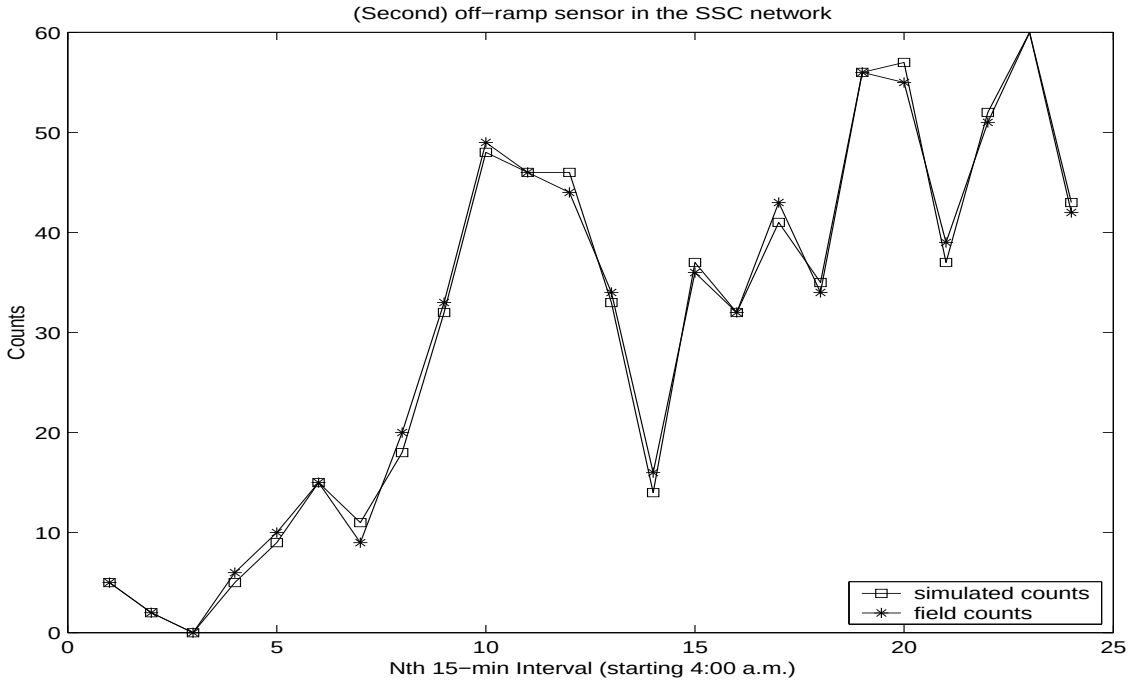

Figure 5-11: (Second) off-ramp sensor counts on the SSC network

Segment Number	u_f mph	k_{jam} veh/lane-m	α	β	u_{min} mph	k_0 veh/lane-m
0	70	0.10625	1.5	0.6	10	0.021875
1	70	0.10625	1.5	0.6	10	0.021875
2	40	0.14375	2	0.4	10	0.0125
3	40	0.14375	2	0.4	10	0.0125
4	40	0.10625	1.75	0.7	10	0.015
5	70	0.10625	1.5	0.6	10	0.021875
6	70	0.10625	1.5	0.6	10	0.021875
7	70	0.10625	1.5	0.6	10	0.021875
8	40	0.14375	2	0.4	10	0.0125
9	40	0.14375	2	0.4	10	0.0125
10	40	0.14375	2	0.4	10	0.0125

Table 5.2: Results of SSC Network Calibration

matching speeds. This involved the adjustment of the segment acceptance capacity ω in the simulations. The exercise was manually carried out and an optimal value of $\omega = 7.8$ was obtained. Figure 5-12 shows the match of the speeds over the 4:00 a.m. to 10:00 a.m. period.

5.4 Network-Level Calibration

Calibration of the supply simulator at the level of the entire network is a very large-sized, stochastic optimization problem. The Irvine network, for instance, involves exploring a subset[3] of the $\Re^7$ hyperspace for each of the 1373 segments. To reduce the scope of the search space, we group similar segments[4] together for calibration purposes.

As has been impressed upon the reader earlier, we will employ the Box Complex and the SPSA algorithms to tackle the current problem. A very brief description of

[3]it is possible to accurately bound from above and below each of the 7 segment-specific parameters, e.g. u_f is expected to be not more than ± 10 mph from the legal speed limit on a freeway segment, and k_{jam} is expected to be in the vicinity of 200 vehicles per lane per mile.

[4]we consider segments to be similar if they are described by the same 7-tuple $\{u_f,\ k_{jam},\ \alpha,\ \beta,\ \texttt{capacity},\ u_{min},\ k_0\}$

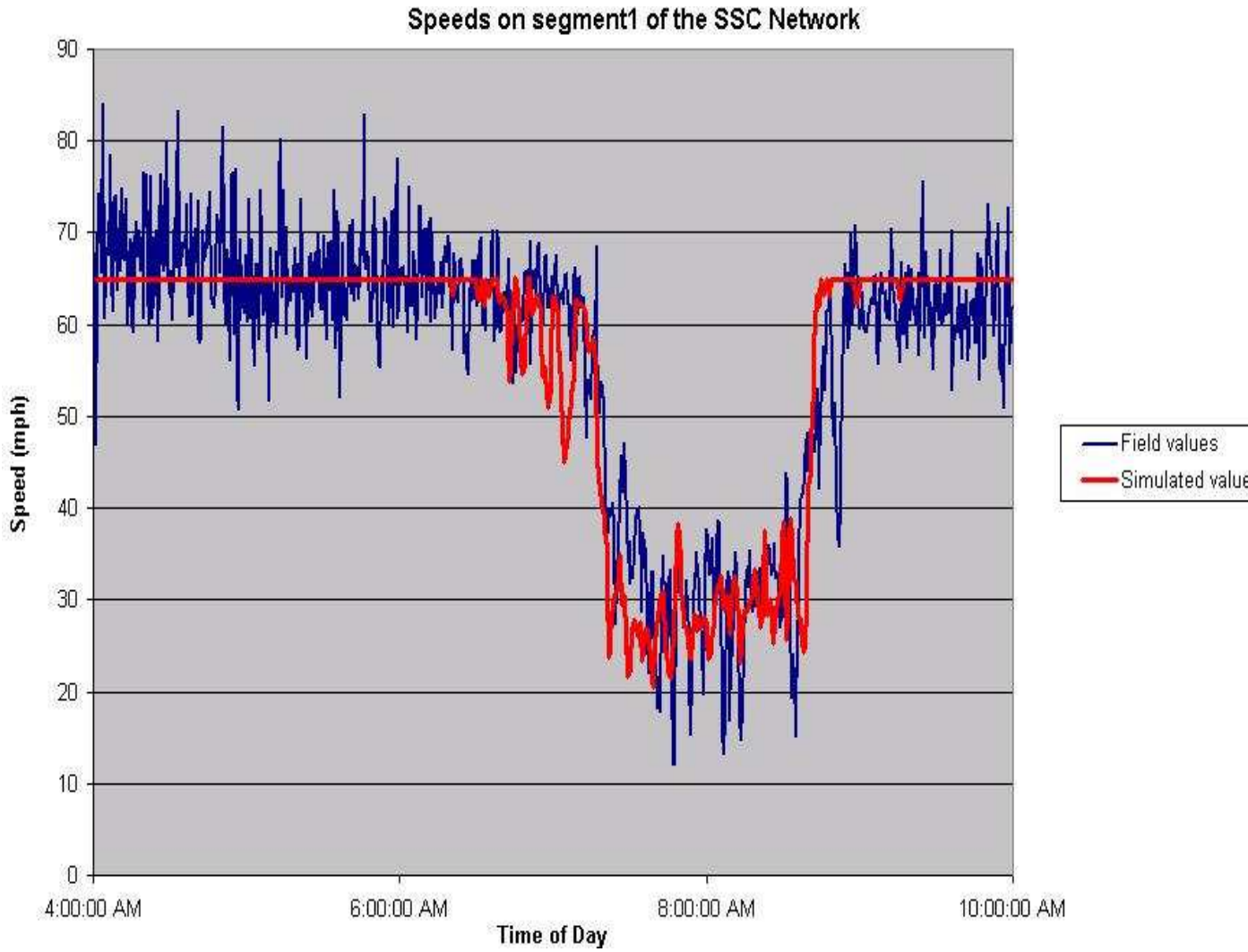

Figure 5-12: Calibrated SSC Network Speeds Comparison

how each individual algorithm is adapted to our problem is given next.

5.4.1 The Box Complex Algorithm

The algorithm attempts to find the global minimum of a multivariate, nonlinear function subject to nonlinear inequality constraints:

minimize $l(X_1, X_2, ..., X_m)$

subject to $G_k \leq X_k \leq H_k, \qquad k = 1, 2, ..., M$

The implicit variables $X_{m+1}, ...X_M$ are dependent functions of the explicit independent variables $X_1, X_2, ..., X_m$. The upper and lower constraints G_k and H_k are either constants or functions of the independent variables.

The basic idea of this pattern search algorithm is to generate a complex of random points[5], evaluate the objective function at each point, and then successively drop the point with the worst objective function value in favor of one obtained by reflecting the worst point through the centroid of the others. Using the notation defined in Chapter 4, the steps are:

step1 Generate a complex of $N > m + 1$ points (see footnote 5) by starting from a feasible initial point and generating $N - 1$ additional points using random numbers and the constraints for each independent variable:

$X_{i,j} = G_i + r_{i,j}(H_i - G_i), \quad i = 1, 2, ..., m, \quad j = 1, 2, ...N - 1$

where $r_{i,j}$ are random numbers between 0 and 1.

step2 All the randomly generated variables must satisfy all the implicit constraints. If an implicit constraint is violated, the variable value is moved one-half of the distance to the centroid of the remaining variable values, i.e.

[5]A point θ_j is a combination of variable values $X_{1,j}, X_{2,j}, ..., X_{m,j}$.

$X_{i,j}^{new} = [X_{i,j}^{old} + \bar{X}_{i,c}] * 0.5, \qquad i = 1, 2, ..., m$

where the coordinates of the centroid of the remaining variables, $\bar{X}_{i,c}$, is given by

$\bar{X}_{i,c} = \frac{1}{N-1}[\sum_{j=1}^{k} X_i^j + X_{i,j}^{old}], \qquad i = 1, 2, ..., m$

This process is repeated as necessary until all the implicit constraints are met.

step3 The objective function value is evaluated at each point. Let $(\theta_0^n, ..., \theta_{N-1}^n)$ be the complex of N points at step n.

Let $l(\theta_j^n)$ denote the objective function value at the jth point at iteration n. More specifically, $l(\theta_1^n)$ is used to denote the mapping of the `supplyparam.dat` file θ_1^n to the objective function value $l(\theta_1^n)$ on the real line. In our case, the objective function is a "loss" or "misfit" value of the simulated flow values versus the field data flow values, namely the norm of the vector of differences between the two.

Let $l(\theta_h^n) = max(l(\theta_0^n), ..., l(\theta_{N-1}^n))$ and $l(\theta_l^n) = min(l(\theta_0^n), ..., l(\theta_{N-1}^n))$.

Let θ_c^n be the centroid of all points excluding $l(\theta_h^n)$.

Reflect θ_h^n through the centroid by computing

$\theta_r^n = \theta_c^n + \alpha((\theta_c^n) - (\theta_h^n))$ for some $\alpha > 0$.

step4 The new point is checked against the constraints and adjusted as before if any constraints are violated. The objective function is then computed for this new point θ_r^n.

step5 If a point repeats in giving the highest objective function value on consecutive trials, it is moved one-half the distance to the centroid of the remaining points.

step6 The algorithm terminates when the highest objective function value is sufficiently close to the lowest value of the objective function, as evaluated at all N points; the termination criterion for the algorithm is thus a user-specified

convergence tolerance combined with an upper limit N_{max} on the maximum number of iterations. The user also has control over the reflection coefficient α and the size of the complex (number of points to generate) to start the process.

We used $\alpha = 1.3$ as recommended by Box [6] and experimented with the size N of the complex and the maximum number N_{max} of iterations.

5.4.2 The SPSA Algorithm

The idea underlying this path search algorithm is one of stochastic gradient approximation.

Using the earlier notation (subscripts are indicative of the iteration):

Let θ_n denote the parameter matrix (in our case, this is the `supplyparam.dat` file) and $l(\theta_n)$ be the corresponding objective function value[6].

Let $l(\theta_n + c_n\Delta_n)$ and $l(\theta_n - c_n\Delta_n)$ be the loss function values for the two perturbed parameter matrices $(\theta_n + c_n\Delta_n)$ and $(\theta_n - c_n\Delta_n)$, where c_n is the current value of the gain sequence $\{c_n\}$ and Δ_n is the current perturbation matrix.

On the basis of these loss function values, we can calculate the estimate of the gradient $\widehat{\nabla} l(\theta_n)$ with a finite difference estimate

$$\frac{l(\theta_n + c_n\Delta_n) - l(\theta_n - c_n\Delta_n)}{2c_n\Delta_n}$$

and use the same to update θ_n according to

$$\theta_{n+1} = \Pi_\Theta(\theta_n - a_n\widehat{\nabla} l(\theta_n))$$

(Here Π_Θ is (if need be) the projection back onto Θ, the constraint set on θ.)

A step-by-step implementation [36] adapted to our calibration problem is outlined below:

[6]Here too, the same objective function – the norm of the vector of deviations between the simulated and field values of flow – is used.

step1 *Initialization and Coefficient Selection* Set counter index $n = 0$. Start with an initial guess $\hat{\theta_0}$ and nonnegative coefficients a, c, A, α, and γ in the SPSA gain sequences $a_n = \frac{a}{(n+A)^\alpha}$ and $c_n = \frac{c}{n^\gamma}$.

Maryak and Chin [23] recommend the values A=60, a=1, α=0.602, c=2 and γ=0.101. The same are used in this implementation.

step2 *Generation of Simultaneous Perturbation Matrix* Generate by Monte Carlo a random perturbation matrix Δ_n, where each component of Δ_n is independently generated from a zero-mean probability distribution satisfying the conditions in Spall [34]. A simple and theoretically valid choice is the Bernoulli ± 1 distribution with probability of 0.5 for each ± 1 outcome. The uniform and normal random variables are not allowed for the elements of Δ_n since they have finite inverse moments.

In our case, the parameter matrix θ represents the `supplyparam.dat` file with elements having widely varying magnitudes. The ± 1 elements in Δ_n are therefore scaled to values representative of the maximum perturbation that is reasonable for a given element[7].

step3 *Loss Function Evaluations* Obtain two measurements of the loss function $l(.)$ based on the simultaneous perturbation around the current $\hat{\theta_n}$: $l(\theta_n + c_n\Delta_n)$ and $l(\theta_n - c_n\Delta_n)$ with the c_n and Δ_n from steps 1 and 2.

step4 *Gradient Approximation* Generate the simultaneous perturbation approximation to the unknown gradient $\widehat{\nabla} l(\theta_n)$:

$$[\widehat{\nabla} l(\theta_n)]_{i,j} = \frac{l(\theta_n + c_n\Delta_n) - l(\theta_n - c_n\Delta_n)}{2c_k * [\Delta_k]_{i,j}}$$

where $[\Delta_k]_{i,j}$ is the i,jth component of the Δ_n matrix; note that the common numerator in all components of $\widehat{\nabla} l(\theta_n)$ reflects the simultaneous perturbation in

[7] We restrict the variation of u_f to 15 mph, the variation of capacities to not more than 2200 vehicles/lane-hour and of k_{jam} to not more than 220 vehicles/lane-mile etc

all components of θ_n in contrast to the component-by-component perturbations in the standard finite-difference approximation.

step5 *Updating θ Estimate* Use the standard SA form

$$\theta_{n+1} = \theta_n - a_n \widehat{\nabla} l(\theta_n)$$

to update θ_n to a new value θ_{n+1}.

step6 *Iteration or Termination* Return to Step 2 with $k+1$ replacing k. Terminate the algorithm if there is litle change in several successive iterates or the maximum allowable number of iterations has been reached.

Gradient Averaging As noted in Spall [35], noisy loss measurements can produce very noisy gradient estimates – one simple way to obviate noisy gradient estimates is to average several gradient approximations. Despite the expense of additional function evaluations, it is especially useful to average several SP gradient approximations (each with an independent value of Δ_n) at a given θ_n when the noise levels in the $l(\theta)$ evaluations are high. Such averaging can often mitigate the fact that SPSA may be more unstable than finite difference stochastic approximation (FDSA) due to its potentially poorer quality gradient approximation. Even a low amount of averaging (two to four SP gradient estimates) can make SPSA more stable than FDSA due to the reduced effective noise contributions. Spall [34] provides theoretical justification for net efficiency improvements by such gradient averaging.

Figure 5-13 shows the experimental findings of the benefits of gradient averaging. Even though the SPSA algorithm did not converge within 10 updates of the θ parameter matrix for any of the three cases depicted, the less kinky objective function value curve for the case where a gradient estimate is averaged over three gradient approximations reinforces the claim that gradient averaging will be useful in noisy settings.

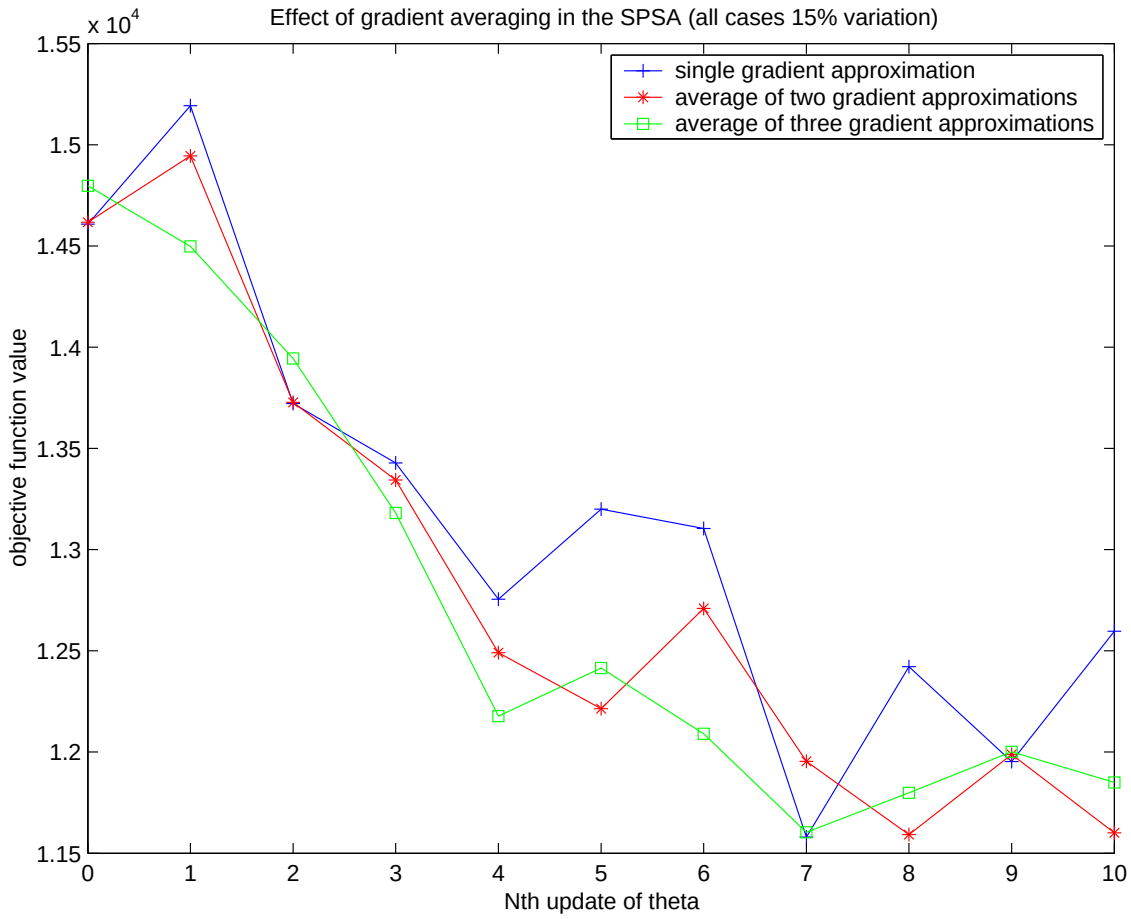

Figure 5-13: Improving SPSA Performance with Gradient Averaging

The results presented in this thesis are based on an average of three gradient approximations.

5.5 Results

We now present results from the calibration of the supply simulator using both algorithms. These results are the best match obtained as judged by the Root Mean Square Error (RMSE) statistic[8] on six different Box Complex optimizations and three different SPSA optimizations, all for simulations performed on the 4:00 a.m. to 9:00 a.m. peak[9] period. Table 5.3 summarizes these values.

Optimization Approach	**Case**	**RMSE**
No Optimization– Simulation using θ_0		365.1342
Box	50 points, 100 (max) iterations, 50%-150%	**318.004**
Box	50 points, 100 (max) iterations, 75%-125%	339.3709
Box	40 points, 80 (max) iterations, 50%-150%	318.1670
Box	40 points, 80 (max) iterations, 75%-125%	339.3980
Box	30 points, 60 (max) iterations, 50%-150%	318.004
Box	30 points, 60 (max) iterations, 75%-125%	339.6963
SPSA3	maximum perturbation=15%	**321.3124**
SPSA3	maximum perturbation=25%	334.8698
SPSA3	maximum perturbation=35%	328.1182

Table 5.3: Root Mean Square Error values for the Different Optimizations

Table 5.4 shows the comparison of the starting and calibrated values of the speed-density parameters and capacities for representative road segments, as obtained by the respective optimization algorithm.

Without exception, the free-flow and minimum speeds increase significantly beyond their starting values. There is also a very noticeable downward adjustment in

[8]The error statistic has the form Root Mean Square Error (RMSE) = $\sqrt{\frac{\Sigma_i (y_i - \hat{y}_i)^2}{N}}$

[9]Flows were found to drop beyond 8:30 a.m.

both the α and β values, more so in the former. There is, however, no such distinct upshift or downshift in the density values of k_{jam} and k_0. The capacities, interestingly, do not depart much from their initial values.

Seg.	u_f^0 BOX u_f^* SPSA u_f^* mph	k_{jam}^0 BOX k_{jam}^* SPSA k_{jam}^* veh/lane-m	α^0 BOX α^* SPSA α^*	β^0 BOX β^* SPSA β^*	u_{min}^0 BOX u_{min}^* SPSA u_{min}^* mph	k_0^0 BOX k_0^* SPSA k_0^* veh/lane-m	*Capacity* BOX *Cap.** SPSA *Cap.** veh/seg-sec
SSC	70	0.10625	1.5	0.6	10	0.021875	3.667
seg0	78.45766	0.10535	1.177425	0.600825	13.60406	0.017936	3.790525
	70.61792	0.126429	1.170875	0.474949	11.28525	0.015811	3.661436
SSC	70	0.10625	1.5	0.6	10	0.021875	3.056
seg1	76.35982	0.102159	0.97878	0.590998	14.98241	0.018092	3.37375
	73.34443	0.122556	1.17523	0.522876	12.74351	0.015793	3.093197
SSC	40	0.14375	2	0.4	10	0.0125	0.444
seg2	47.23401	0.116296	1.188293	0.367277	14.15509	0.014969	0.486851
	48.80577	0.126379	1.567718	0.339813	12.75233	0.012411	0.458136
SSC	40	0.14375	2	0.4	10	0.0125	0.444
seg3	47.23401	0.116296	1.188293	0.367277	14.15509	0.014969	0.486851
	48.80577	0.126379	1.567718	0.339813	12.75233	0.012411	0.458136
SSC	40	0.10625	1.75	0.7	10	0.015	0.889
seg4	48.72804	0.151936	1.470721	0.301933	13.15917	0.011801	0.962241
	51.83441	0.11925	1.528141	0.314389	11.89201	0.011886	0.986965
SSC	70	0.10625	1.5	0.6	10	0.021875	3.056
seg5	76.35982	0.102159	0.97878	0.590998	14.98241	0.018092	3.37375
	73.34443	0.122556	1.17523	0.522876	12.74351	0.015793	3.093197
SSC	70	0.10625	1.5	0.6	10	0.021875	3.056
seg6	76.35982	0.102159	0.97878	0.590998	14.98241	0.018092	3.37375
	73.34443	0.122556	1.17523	0.522876	12.74351	0.015793	3.093197
SSC	70	0.10625	1.5	0.6	10	0.021875	3.056
seg7	76.35982	0.102159	0.97878	0.590998	14.98241	0.018092	3.37375
	73.34443	0.122556	1.17523	0.522876	12.74351	0.015793	3.093197
SSC	40	0.14375	2	0.4	10	0.0125	0.444
seg8	47.23401	0.116296	1.188293	0.367277	13.15509	0.014969	0.486851
	48.80577	0.126379	1.567718	0.339813	12.75233	0.012411	0.458136
SSC	40	0.14375	2	0.4	10	0.0125	0.444
seg9	47.23401	0.116296	1.188293	0.367277	14.15509	0.014969	0.486851
	48.80577	0.126379	1.567718	0.339813	12.75233	0.012411	0.458136
SSC	40	0.14375	2	0.4	10	0.0125	0.889
seg10	52.81555	0.09825	1.601761	0.35645	14.08403	0.009814	0.856658
	50.71475	0.120768	1.59631	0.296459	12.63504	0.012187	0.907172

Table 5.4: Calibrated Values Versus Starting Values

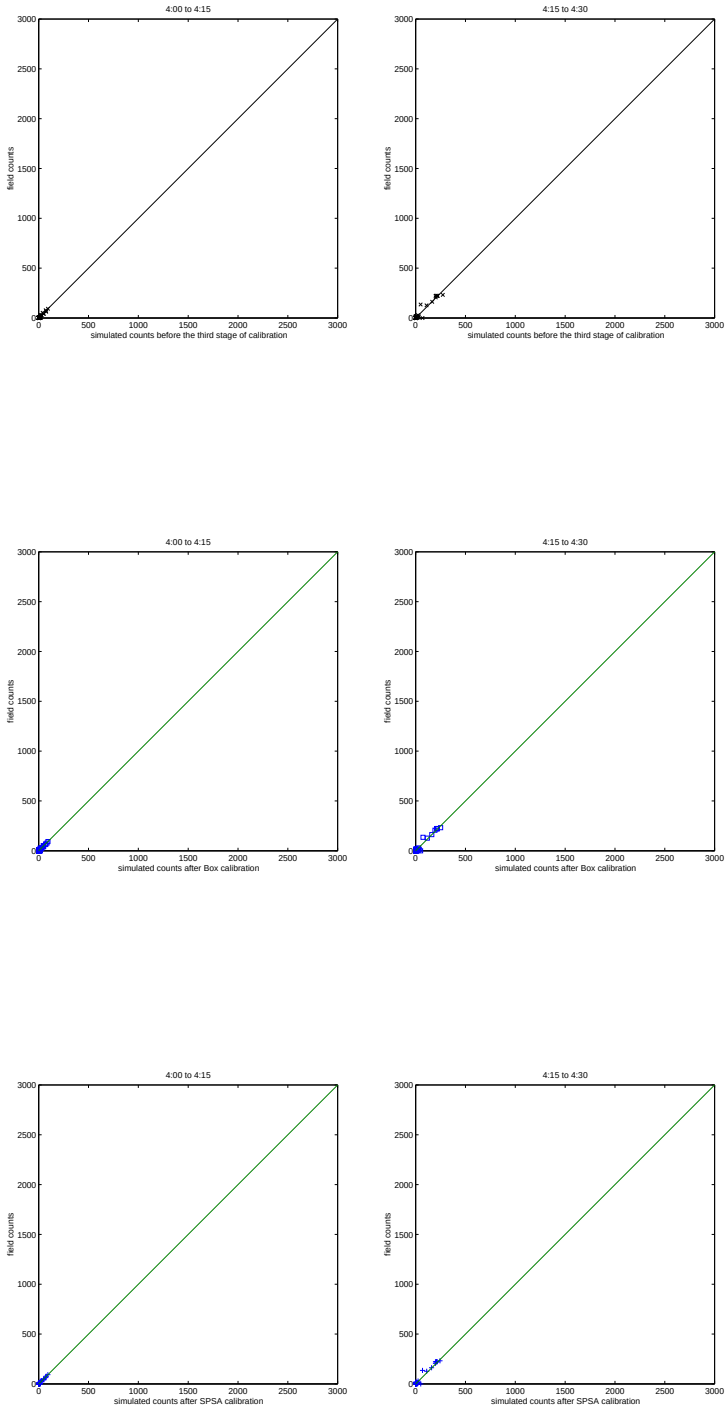

Figure 5-14: Flows for 04:00-04:15 and 04:15-04:30

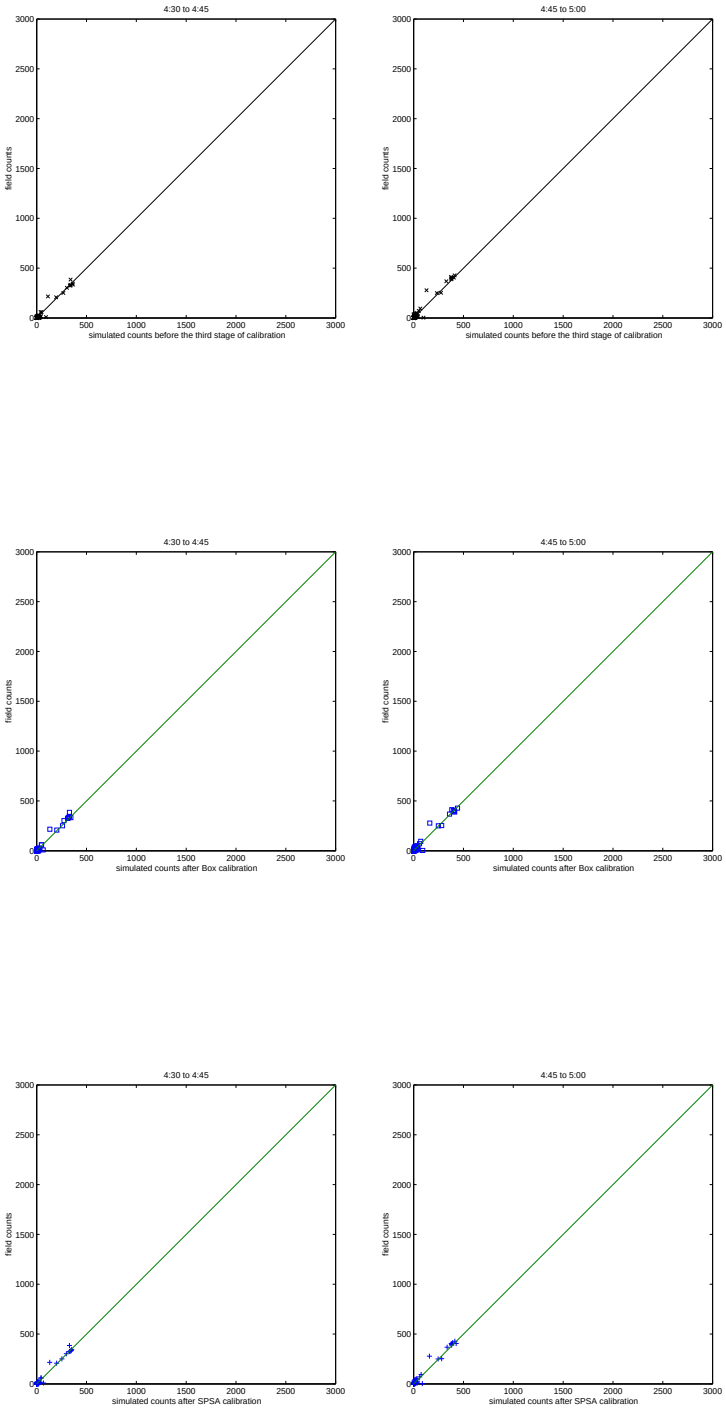

Figure 5-15: Flows for 04:30-04:45 and 04:45-05:00

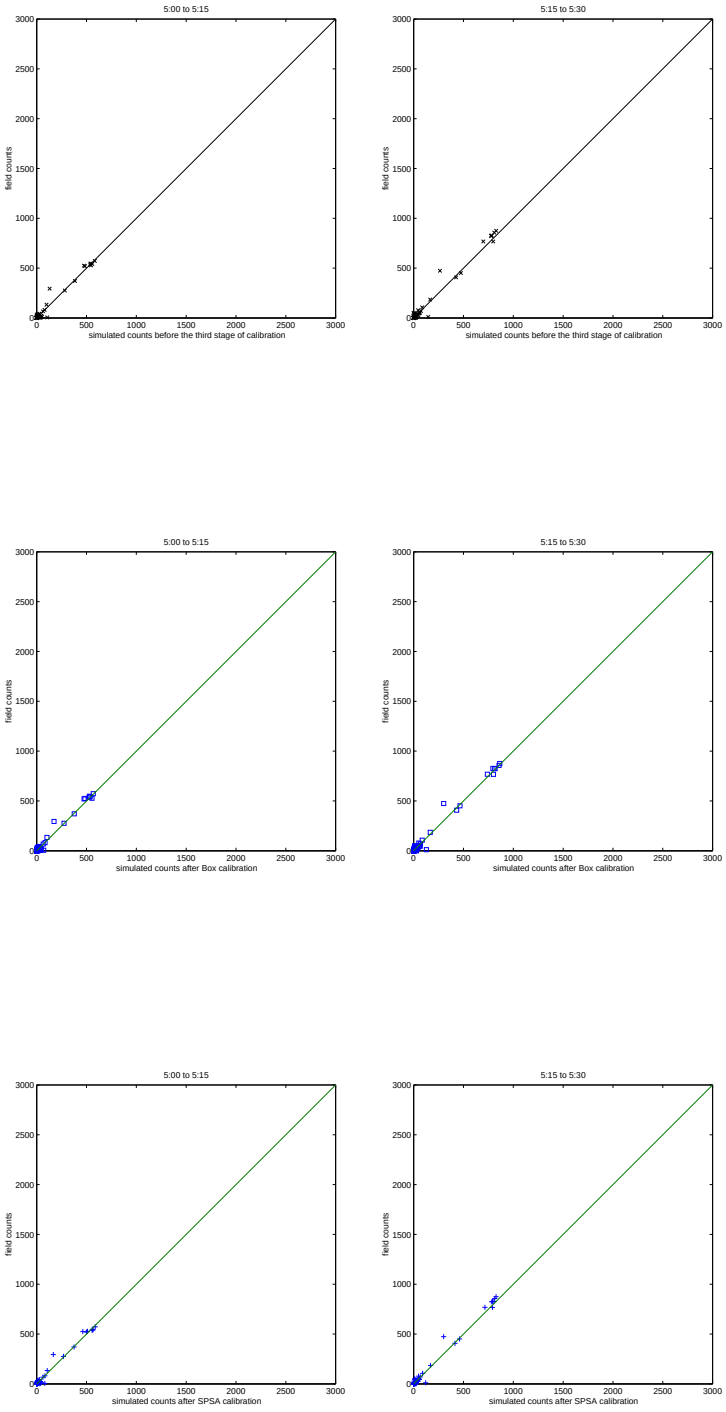

Figure 5-16: Flows for 05:00-05:15 and 05:15-05:30

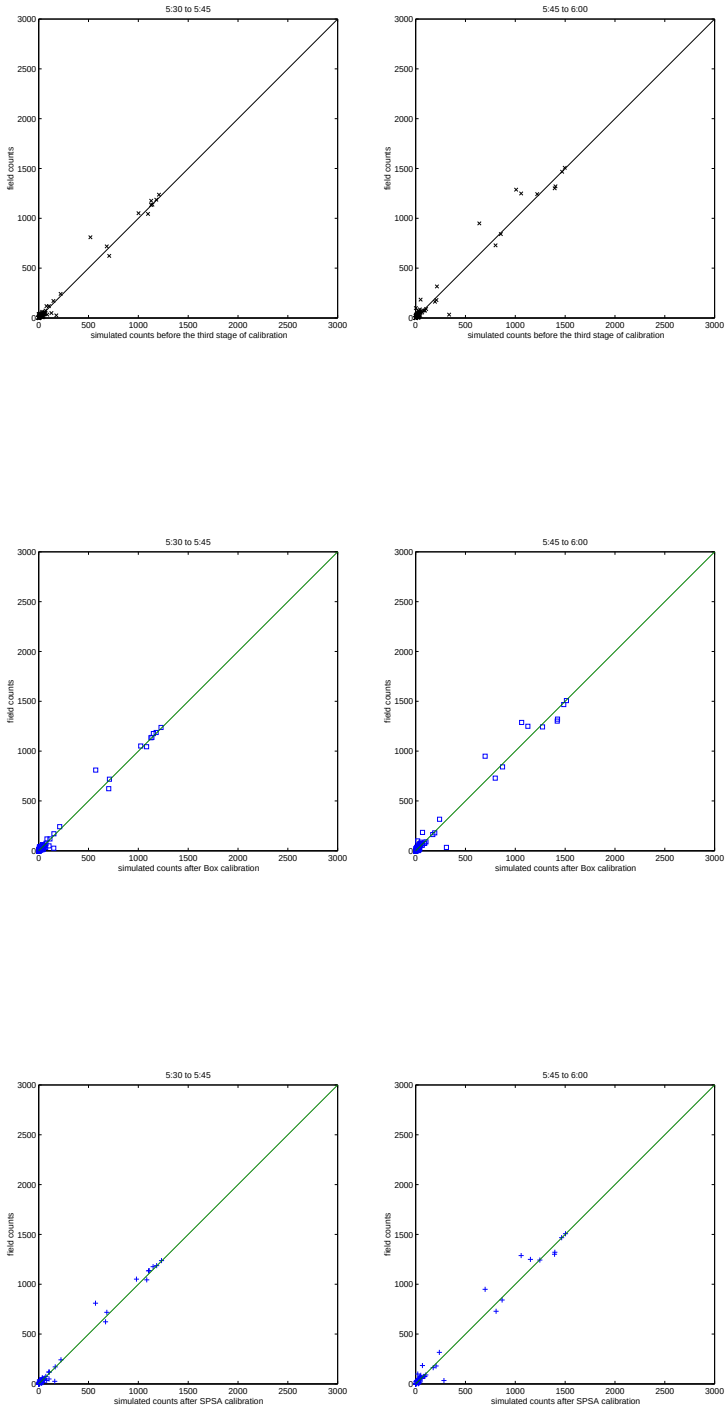

Figure 5-17: Flows for 05:30-05:45 and 05:45-06:00

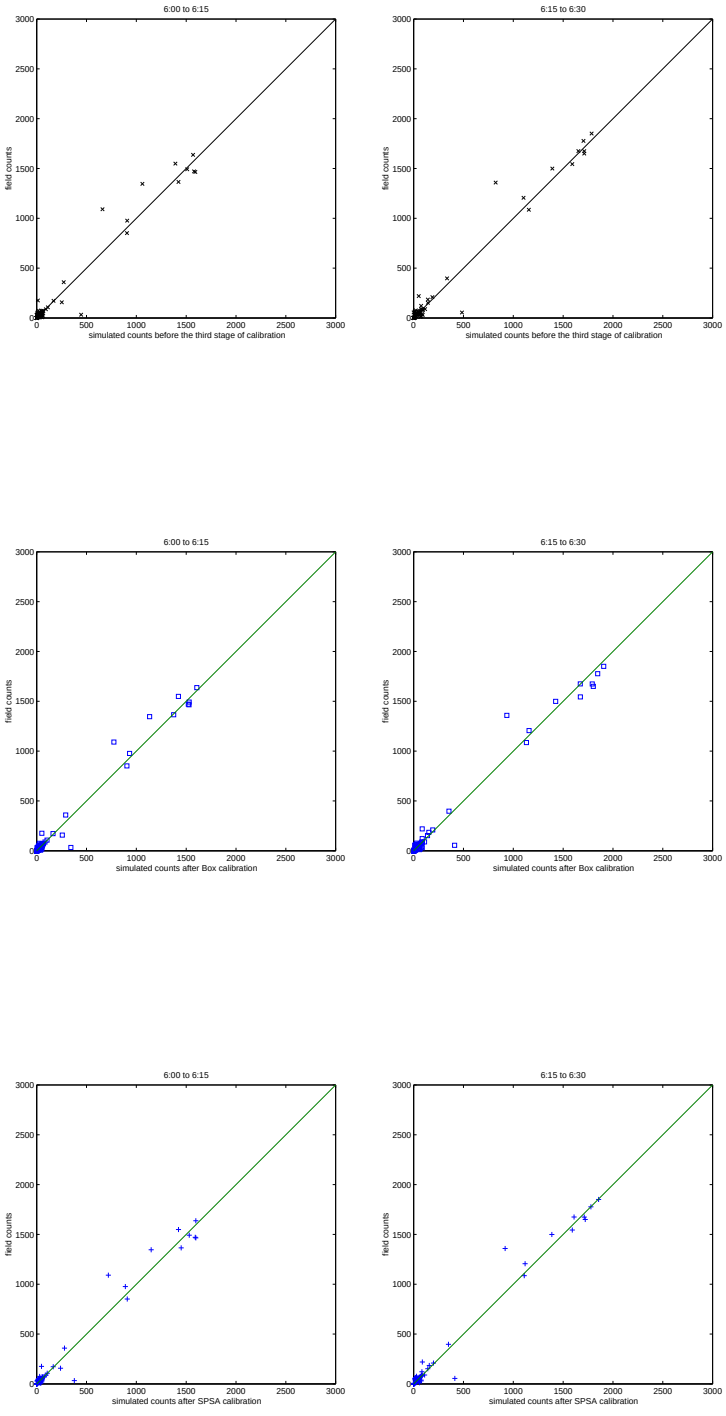

Figure 5-18: Flows for 06:00-06:15 and 06:15-06:30

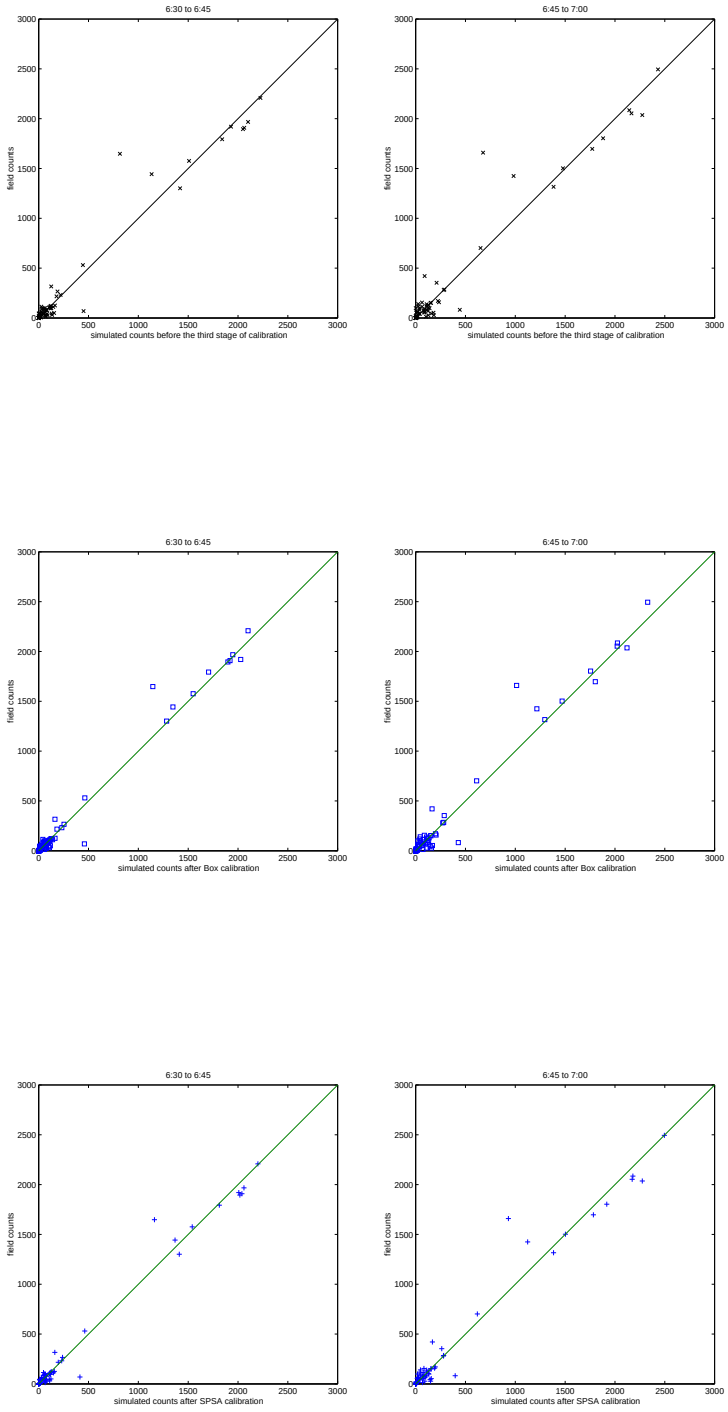

Figure 5-19: Flows for 06:30-06:45 and 06:45:-07:00

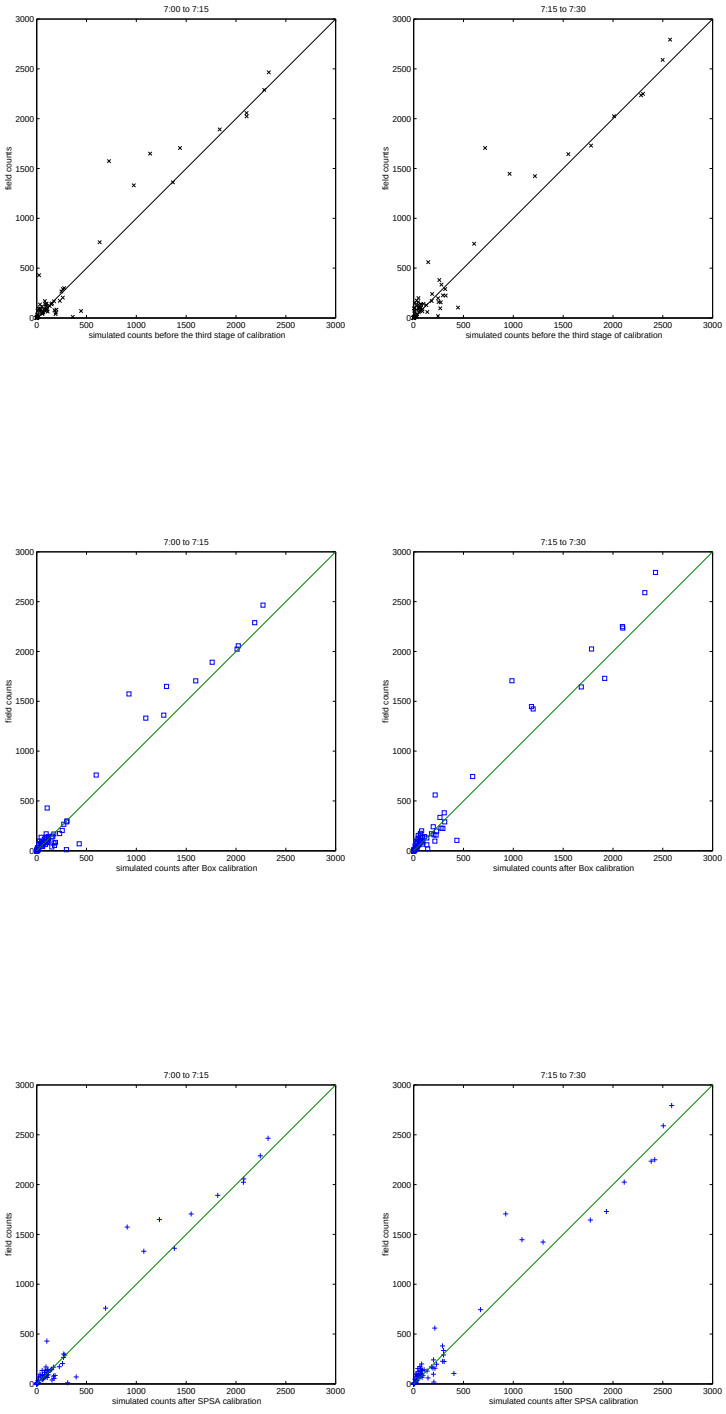

Figure 5-20: Flows for 07:00-07:15 and 07:15-07:30

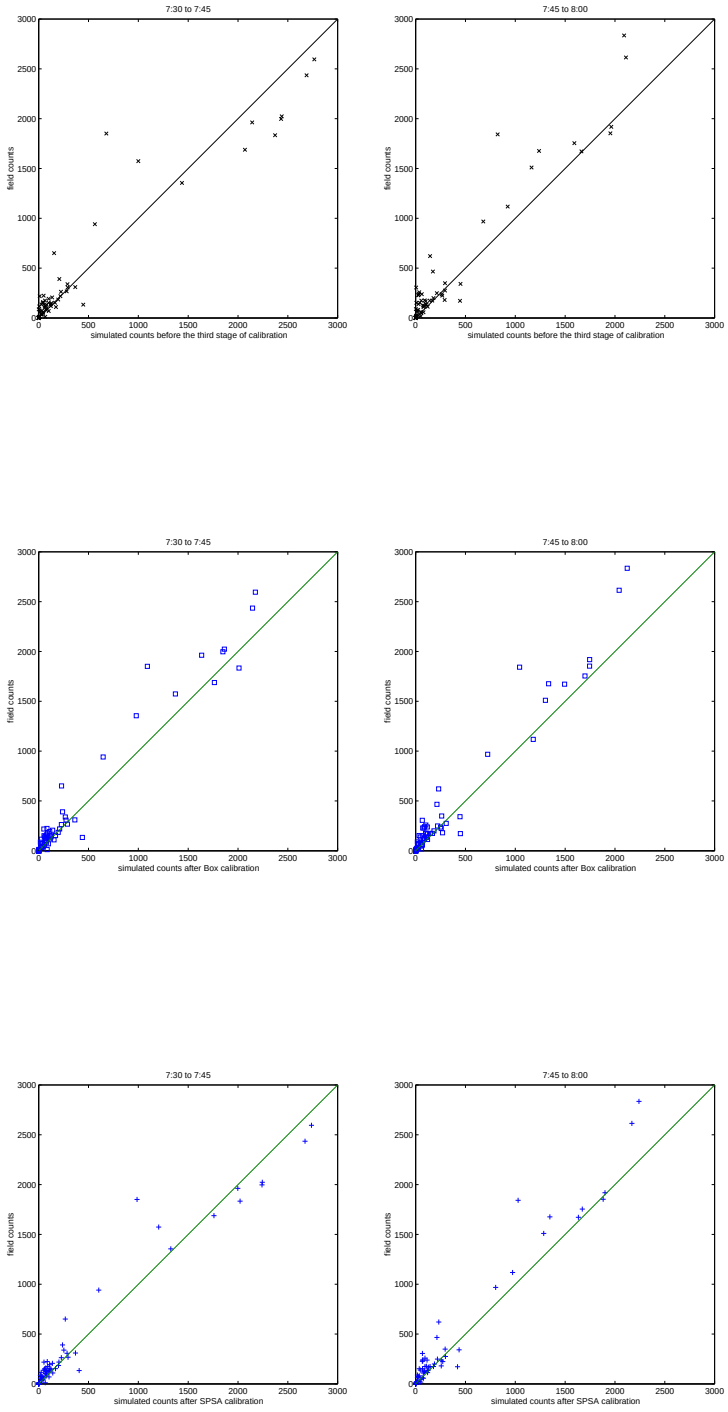

Figure 5-21: Flows for 07:30-07:45 and 07:45-08:00

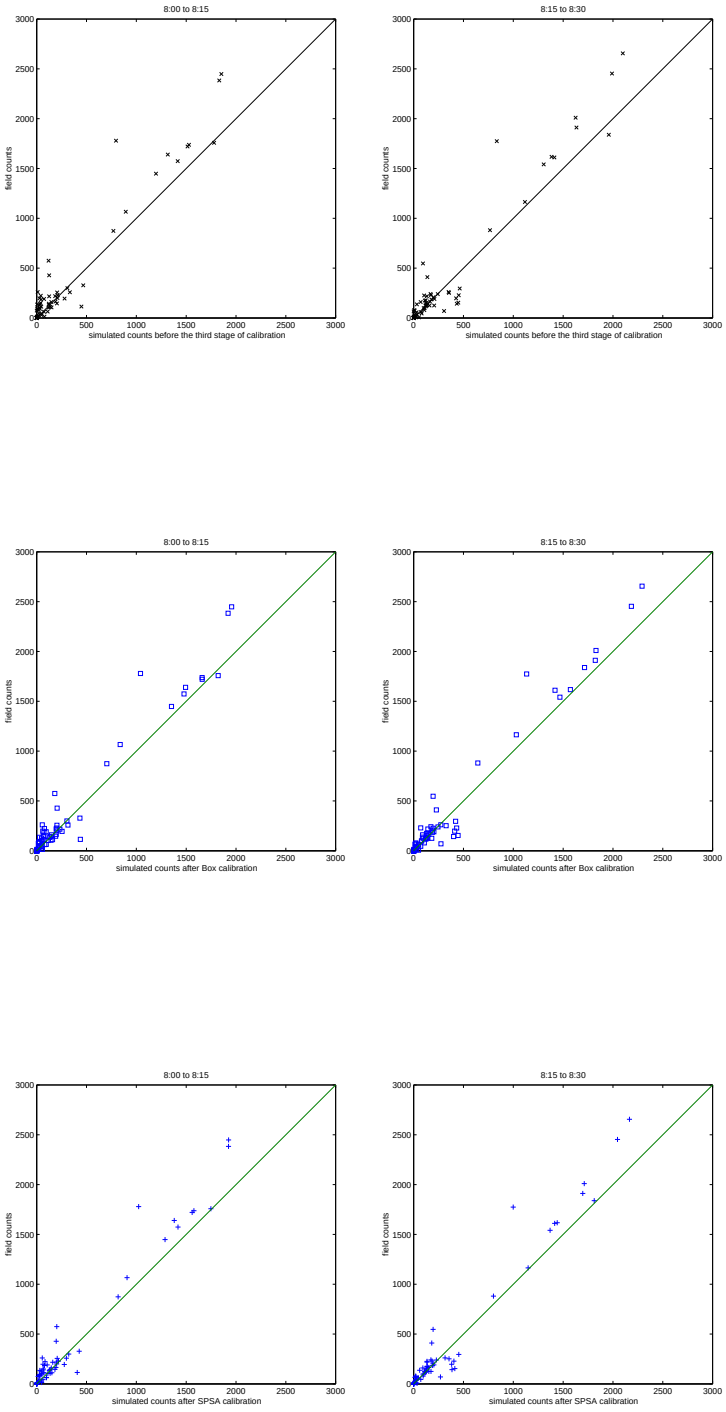

Figure 5-22: Flows for 08:00-08:15 and 08:15-08:30

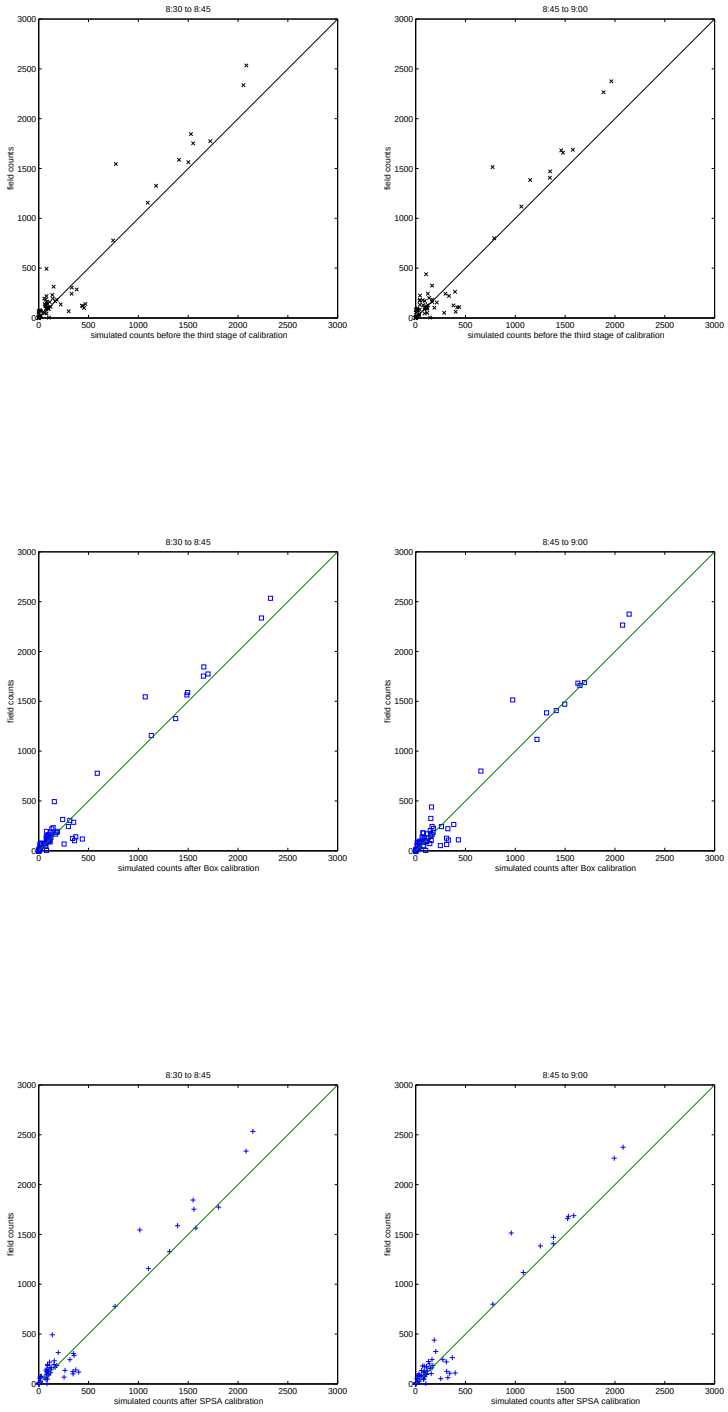

Figure 5-23: Flows for 08:30-08:45 and 08:45-09:00

As shown in Figures 5-14 through 5-23, both the algorithms achieve a very good fit[10] between observed and simulated flow values inspite of not strictly converging as defined by the convergence criterion.

5.6 Runtime and Convergence

Before presenting the running times of each algorithm for different cases, we state again the convergence criterion for each algorithm:

Box Complex algorithm The algorithm is deemed to have converged when the maximum objective function value is within the user-defined tolerance level as measured from the minimum objective function value. The tolerance level used in all the Box Complex optimizations was 1%.

SPSA algorithm The SPSA algorithm is deemed to have converged when the stochastic gradient approximation becomes zero.

In both these cases, the stochastic nature of the problem combined with the long time required for a single simulation forces us to terminate algorithm execution if a user-defined limit on the maximum number of iterations is exceeded.

The Box algorithm was run for a number of cases in which the number of initial points in the complex and the maximum number of iterations to converge were varied. Also varied was the percentage leeway in terms of the upward or downward drift permitted to each parameter. The improvement in the objective function values is shown in Figure 5-24. Table 5.5 shows the information regarding the percentage deviation of the worst-valued point w.r.t. the best-valued point at termination in each case. Table 5.6 shows the corresponding runtimes.

[10] The plots do not have points corresponding to the bad sensors identified by ids 6, 12, 18, 20, 24, 31, 50, 57 and 66; these sensors' flow values have been filtered out.

The SPSA algorithm was run with the gain sequences prescribed in Maryak and Chin [23] with different limits on the maximum perturbation permitted on the parameter matrix. It may be recalled that in the SPSA algorithm, we average three gradient samplings for the gradient estimate at any θ_n. Thus a single updating of the parameter matrix from θ_n to θ_{n+1} necessitates six simulations. The maximum number of updates was therefore set to 10. The runtime information for the SPSA optimizations is shown in Table 5.7. Figure 5-25 shows the convergence of the SPSA algorithm for three different limits on the maximum perturbation, namely 15%, 25% and 35%.

All the simulations were carried out on IBM IntelliStation EPro (Linux) workstations with 2.0 GHz Pentium IV processors and 512 MB SDRAM.

Percent Variation	N	N_{max}	N_{end}	$\frac{l(\theta_h^{end}-l(\theta_l^{end})}{l(\theta_l^{end})}$	$\frac{l(\theta_h^{start}-l(\theta_l^{start})}{l(\theta_l^{start})}$	$l(\theta_l^{end})$	$l(\theta_l^{start})$
50 to 150 (Box1)	50	100	100	11.09%	14.42%	11728	11728
75 to 125 (Box2)	50	100	100	3.82%	5.56%	12516	12516
50 to 150 (Box3)	40	80	80	11.19%	14.49%	11734	11734
75 to 125 (Box4)	40	80	80	3.93%	5.54%	12517	12517
50 to 150 (Box5)	30	60	60	10.81%	14.69%	11728	11728
75 to 125 (Box6)	30	60	60	3.90%	5.64%	12528	12528

Table 5.5: Convergence of the Box Complex Algorithm

Case	Start Time	End Time	Runtime
Box1 (50 points, 100 iterations, 50%-150%)	01:16:22	11:56:18	10:39:56
Box2 (50 points, 100 iterations, 75%-125%)	01:16:32	10:20:45	09:04:13
Box3 (40 points, 80 iterations, 50%-150%)	01:16:40	09:45:50	08:29:10
Box4 (40 points, 80 iterations, 75%-125%)	01:16:45	08:38:58	07:22:13
Box5 (30 points, 60 iterations, 50%-150%)	01:16:52	07:32:10	06:15:18
Box6 (30 points, 60 iterations, 75%-125%)	01:17:05	06:47:03	05:29:58

Table 5.6: Runtimes of the Box Complex Algorithm

Both algorithms' best solution quality is roughly the same as judged by the lowest objective function value. Also, as is borne out by Table 5.4 the solutions do not differ

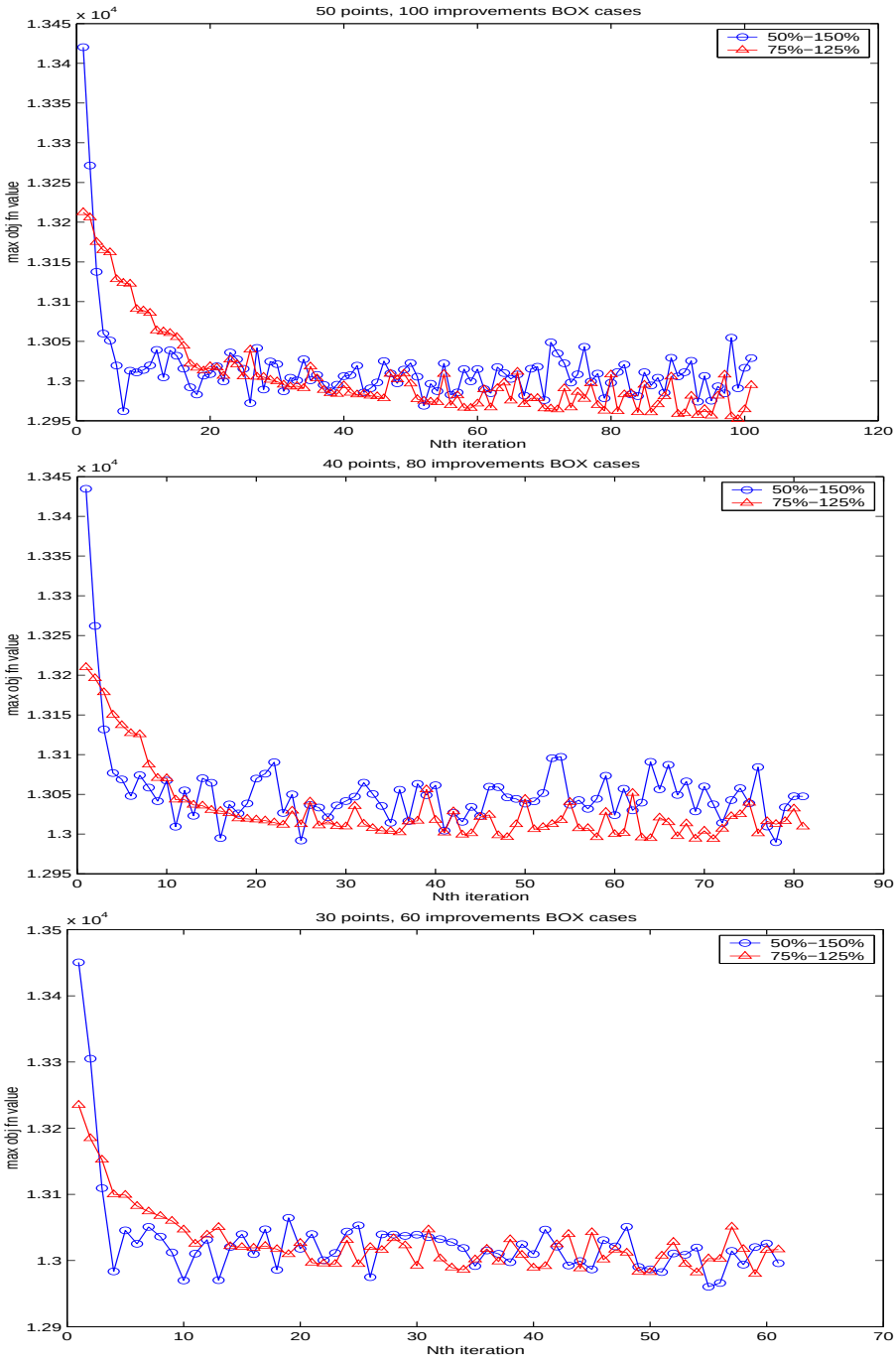

Figure 5-24: Convergence of the Box Complex Algorithm

Case	Start Time	End Time	Runtime
SPSA1 (maximum perturbation = 15%)	00:56:17	05:26:57	04:30:40
SPSA2 (maximum perturbation = 25%)	00:57:40	07:06:55	06:09:15
SPSA3 (maximum perturbation = 35%)	01:08:40	08:37:48	07:29:08

Table 5.7: Runtimes of the SPSA Algorithm

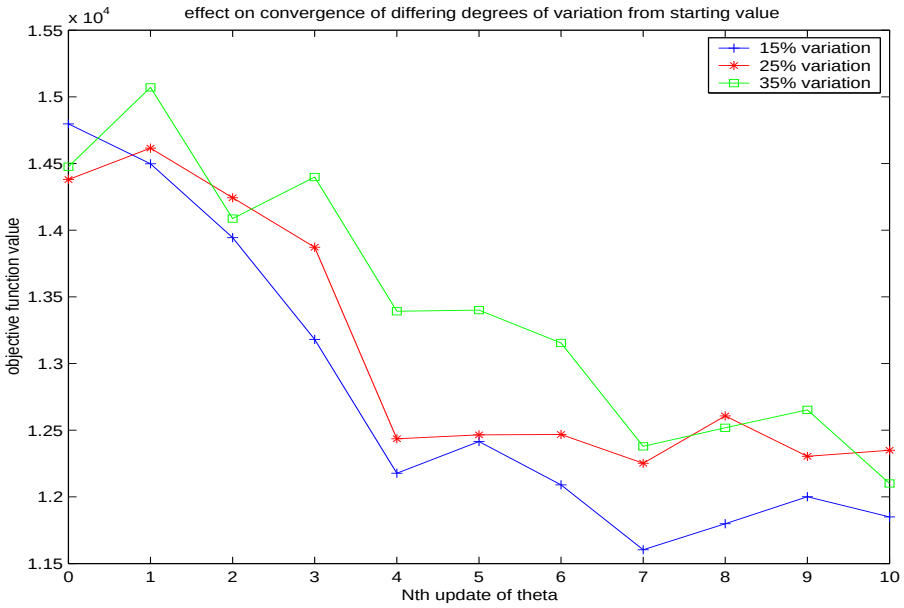

Figure 5-25: Convergence of the SPSA Algorithm

much. Quite surprisingly, the Box Complex algorithm performs better when given more leeway (±50%) to explore the domain, whereas the SPSA algorithm achieves its best solution when restricted to ±15% variation from the starting values.

The Box Complex algorithm was found to never improve upon the initial best point in the generated complex; the lack of aggressively exploring the domain after generating the initial complex is a major weakness of this algorithm.

We also observed that the SPSA algorithm needs to base its gradient estimate on many gradient approximations in order to be less erratic in its convergence – this entails many more function evaluations. However, for large-dimensional problems like the current one, gradient averaging seems to be beneficial rather than detrimental in

time to converge and final solution quality.

5.7 Summary

In this chapter, we calibrated the supply simulator in DynaMIT/DynaMIT-P using archived sensor data from Irvine, California.

Successful calibration involved a prudent choice of representative individual segments in the first stage and the judicious choice of a subnetwork with segment interactions but minimal route choice in the second stage. In the final stage, two stochastic optimization algorithms were employed to improve upon the starting values of parameters calibrated sequentially through the first two stages. Investigation of runtimes and level-of-fit statistics indicated that the SPSA algorithm outperforms the Box Complex algorithm. The three-stage calibration methodology outlined in Chapter 3 was thus successfully implemented using a stochastic-optimization-pronged approach.

Chapter 6

Conclusion

This thesis focused on the calibration of mesoscopic traffic simulation models for a large real network. The prototype used in the research was the supply simulator module in the DynaMIT DTA system. We begin this chapter with a brief description of the research contribution and findings and end with directions for further research.

6.1 Research Contribution and Findings

A three-stage calibration methodology was developed to work around practical limitations of scant sensor data and successfully executed with stochastic optimization techniques to tackle the stochasticity inherent in simulation. The estimation accuracy was found to increase with each progressive stage of calibration. Also, both stochastic algorithms were found to perform better when allowed more leeway in changing the parameters. However, more allowable percentage variation in the parameters was also found to increase the runtime in both cases.

The quality of the Box algorithm solution was found to be independent of the size of the complex and more a function of the allowable limits for parameter variation. It is interesting to note that the minimum-valued point in the initial complex was never improved upon in any of the six Box optimizations, and that the Box algorithm

concentrates more on making the worst points in the randomly generated complex less worse off rather than finding a global optimum.

The SPSA algorithm was found to take lesser number of simulation replilcations, and therefore lesser runtime, than the Box algorithm to achieve a comparable level-of-fit as judged by the RMSE statistic. This finding must however, be qualified with the fact that each parameter matrix update in the SPSA algorithm was performed by averaging three gradient approximations, which involves performing six simulation replications.

6.2 Future Research

One of the algorithms used in the current research was the SPSA algorithm based on gradient approximation. Like most gradient approximation algorithms, the SPSA algorithm has no provision for using old data. It would be interesting to examine memory-based stochastic optimization (Moore and Schneider [28]) to build a global non-linear model of the expected output instead of discarding the data that produced the gradient estimate. Moore and Schneider [28] note significant improvements in time to converge and final solution quality as compared to conventional stochastic optimization.

Also, having tackled the calibration of the supply simulator, a natural extension to the current research would be to focus on the integrated calibration of both the demand and supply simulator modules in a DTA system.

Appendix A

MATLAB Code for the Box Complex Algorithm

```
!date > date.txt % start time of program execution
 day1_field_counts_0400_0900_=[ 6 28 29 0 41 65 2 63 0 69 8 67 29 3 43
  2 25 6 2 36 0 25 52 25 8 9 6 7 8 8 123 93 2 1 7 0 3 4 6 4 8 6 10 2 3
  3 4 1 0 402 6 0 0 3 1 11 808 2 9 0 2 6 1 78 2 26 1 0 9 206 29 0 221
  290 2 222 0 223 6 223 127 16 223 0 135 82 8 181 20 162 222 101 13 7
  6 12 14 4 162 4 0 1 6 2 11 8 6 6 6 13 5 1 5 2 1 3 0 16 0 0 0 11 5 11
  426 1 7 0 0 10 4 232 15 47 1 4 21 303 59 0 325 421 0 345 3 335 12
  218 208 19 326 9 216 111 7 332 12 251 385 74 9 13 9 11 10 15 164 14
  1 2 6 1 7 11 6 15 6 15 12 5 2 4 9 2 0 62 4 0 2 15 4 20 7 7 11 1 0 15
  4 333 29 52 0 59 31 368 71 0 405 535 6 428 18 389 2 180 250 40 396 3
  278 52 16 321 8 253 412 179 19 11 11 17 20 17 238 7 0 2 11 5 9 14 7
  21 9 21 23 4 6 1 8 3 0 10 4 1 6 39 3 23 571 8 21 0 0 45 3 405 51 43
  8 95 26 520 67 2 538 663 10 573 13 542 17 247 276 38 547 5 294 115
  19 486 0 371 525 205 20 24 14 20 18 18 364 81 2 6 22 5 13 20 15 21 7
  35 13 5 2 6 7 5 2 8 6 2 2 25 7 17 34 8 11 1 1 29 10 527 44 63 2 134
  50 768 106 1 824 1004 15 875 19 857 51 402 407 53 826 11 474 120 47
```

678 28 451 826 333 39 45 26 40 38 40 134 70 2 5 27 4 23 25 12 30 9
38 27 5 5 10 17 5 2 70 4 1 0 30 12 29 6 11 24 1 0 34 10 767 78 111 0
185 62 1051 171 1 1134 1412 9 1237 34 1187 28 602 718 119 1137 26
810 106 39 881 40 623 1176 539 40 49 25 75 45 39 193 15 4 9 31 13 26
34 32 47 24 48 56 8 14 14 22 6 2 206 6 1 0 43 12 58 6 10 44 3 0 51
11 1044 119 82 19 242 74 1243 163 3 1301 1640 20 1507 49 1467 23
754 842 180 1321 34 949 232 99 981 20 729 1288 706 43 57 30 90 43 55
216 28 6 19 35 20 32 43 23 75 28 86 64 16 33 8 26 7 4 134 14 2 4 72
14 75 56 21 61 3 0 73 15 1249 184 99 37 316 107 1365 172 3 1471 1725
33 1637 66 1492 45 747 977 157 1465 33 1091 183 54 1210 29 852 1346
240 72 64 41 88 71 73 147 45 4 21 38 20 58 54 49 55 45 75 45 47 22
13 33 12 31 127 33 5 3 42 21 48 194 53 39 3 1 59 24 1549 176 126 7
359 152 1544 185 6 1674 2066 49 1851 64 1777 49 930 1206 209 1650 55
1359 174 76 1455 56 1086 1499 359 74 82 42 50 79 88 130 53 4 26 47
26 88 63 64 89 45 122 64 33 38 13 45 21 7 69 33 3 0 50 39 69 69 31
51 4 2 54 30 1675 220 108 12 397 217 1793 232 8 1895 2419 46 2208
117 1967 122 1086 1443 266 1908 69 1648 299 86 1861 46 1301 1576 485
104 111 79 40 96 125 215 53 10 32 65 27 109 96 36 83 40 114 98 31 46
17 52 31 2 66 29 5 5 83 45 98 99 46 77 2 3 99 50 1920 316 144 13 531
280 1803 285 13 2085 2667 44 2494 151 2036 105 1319 1425 353 2053 82
1659 278 140 1893 99 1316 1502 268 138 158 77 41 130 171 217 82 14
42 82 27 152 114 58 121 55 155 89 56 62 20 66 27 6 92 55 14 4 98 45
98 111 70 69 1 19 118 53 1697 421 175 13 702 300 1892 266 12 2023
2753 34 2465 137 2289 138 1322 1331 290 2057 70 1574 328 135 1926 94
1361 1649 358 144 172 90 43 146 204 204 82 14 43 97 39 168 116 71
120 69 170 86 56 60 31 85 54 6 168 69 11 6 116 78 92 96 82 64 6 15
134 84 1705 429 264 11 760 335 2025 290 15 2235 2883 16 2793 175
2590 198 1483 1447 381 2249 105 1706 420 154 2027 96 1423 1644 772

```
 139 227 98 38 171 225 261 129 34 58 125 60 241 142 84 136 72 200 94
 98 71 48 96 97 12 406 101 28 3 131 160 86 97 104 67 6 10 158 157
 1730 560 362 19 745 339 1962 307 14 1997 2703 36 2595 151 2435 153
 1421 1574 391 2023 134 1851 396 218 1839 86 1355 1688 476 175 267
 134 43 185 310 331 219 44 80 147 70 263 152 74 125 58 224 120 104 60
 73 116 148 10 137 120 32 10 136 193 110 144 138 97 12 12 160 206
 1834 651 484 12 941 349 1672 224 22 1918 2638 32 2835 139 2614 238
 1506 1510 466 1853 172 1842 336 306 1572 89 1118 1676 379 177 275
 174 47 181 342 371 200 57 113 135 62 249 126 73 152 63 229 168 109
 81 74 152 174 8 247 120 25 12 228 258 174 166 138 123 20 11 241 239
 1754 621 524 18 968 299 1640 228 16 1721 2372 43 2448 127 2383 217
 1396 1448 428 1738 115 1779 406 260 1518 74 1066 1574 677 191 258
 135 29 195 328 349 144 47 103 141 64 219 112 82 151 93 197 160 91 62
 86 135 146 13 221 105 25 10 185 256 165 125 106 113 10 11 223 200
 1758 575 524 14 874 261 1617 240 26 2010 2726 34 2655 160 2453 226
 1404 1541 410 1911 154 1774 417 230 1773 66 1164 1611 629 177 252
 125 45 186 296 302 132 57 117 137 70 191 125 75 151 78 218 150 98 60
 79 175 143 11 295 99 44 12 208 197 165 327 125 126 12 14 241 228
 1839 547 524 6 881 305 1587 232 21 1845 2474 56 2534 149 2336 136
 1323 1327 314 1753 118 1545 313 195 1729 71 1157 1563 398 160 243
 102 45 184 286 288 112 49 95 163 68 167 109 78 121 72 194 126 90 72
 62 139 101 17 290 98 22 18 185 124 142 123 119 90 8 10 219 140 1774
 493 408 4 778 243 1471 244 21 1682 2346 55 2375 178 2265 155 1222
 1384 325 1658 110 1514 317 201 1661 83 1118 1407 526 170 221 96 45
 184 264 484 100 51 88 118 52 161 103 97 129 87 181 104 80 72 39 134
 63 11 632 74 23 14 173 126 130 155 95 84 13 8 225 107 1688 439 318 4
 800 ]';
n_inner_iterations=50; % number of points to be generated in the complex
```

```
 N_max=100; % maximum number of centroid movements to converge




%% ##########################################################################
%% ################### UNIX file-processing cmds ############################
%% ##########################################################################
!grep -v '<END>' supplyparam_jan_new.dat >! supplyparam_jan_new_wo_END.dat
!awk '{ line=""; for (i=3;i<=9;i++) line=line" "$i; print line }'
  supplyparam_jan_new_wo_END.dat >!
    supplyparam_jan_new_wo_END_or_braces_or_seg_number.dat
!sort -n -k 1 -k 2 -k 3 -k 4 -k 5 -k 6 -k 7
  supplyparam_jan_new_wo_END_or_braces_or_seg_number.dat >!
    sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number.dat
load sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number.dat
SEED_SUPPLYPARAM =
  sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number;
[n_rows n_columns] = size(SEED_SUPPLYPARAM);
%% ##########################################################################
!grep -v '<END>' supplyparam_jan_new.dat >! supplyparam_jan_new_wo_END.dat
!awk '{ line=""; for (i=3;i!=7;i++) line=line" "$i; for (i=8;i!=10;i++)
  line=line" "$i; print line }' supplyparam_jan_new_wo_END.dat >!
    supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat
!sort -n -k 1 -k 2 -k 3 -k 4 -k 5 -k 6
  supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat >!
    sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat
```

```
load sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat
SUPPLYPARAM =
  sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps;
%% ##############################################################################
[num_rows num_columns] = size(SEED_SUPPLYPARAM);
n_distinct_segs=1;
for row_index = 2:num_rows
 row2=SEED_SUPPLYPARAM(row_index,:);
 row1=SEED_SUPPLYPARAM(row_index-1,:);
 comparison_result  = row2==row1;
 for parameter_index = 1:7
  if comparison_result(1,parameter_index)==0
    n_distinct_segs=n_distinct_segs+1;
    break;
  end
 end
end
n_distinct_segs;
%% ##############################################################################
!awk '{ line=""; for (i=2;i<=9;i++) line=line" "$i; print line }'
  supplyparam_jan_new_wo_END.dat >! supplyparam_jan_new_wo_END_or_braces.dat
!sort -n -k 1 supplyparam_jan_new_wo_END_or_braces.dat >!
  sorted_supplyparam_jan_new_wo_END_or_braces.dat
load sorted_supplyparam_jan_new_wo_END_or_braces.dat;
duplicate_sorted_supplyparam_jan_new_wo_END_or_braces=
  sorted_supplyparam_jan_new_wo_END_or_braces;
updated_sorted_supplyparam_jan_new_wo_END_or_braces=
  sorted_supplyparam_jan_new_wo_END_or_braces;
```

```
%% ##############################################################################
%% ###################  UNIX processing ends here  ##############################
%% ##############################################################################









%% ##############################################################################
%% ######################  Generating the cplx  ##################################
%% ##############################################################################
N=1;
while N<=n_inner_iterations
%------------------------------------------------------------------------------
%------------------------------------------------------------------------------
n=1;
flags=zeros(n_rows,1);
while n<=n_rows
%------------------------------------------------------------------------------
if flags(n,1)==0
  for i=2:8
    duplicate_sorted_supplyparam_jan_new_wo_END_or_braces(n,i)=
      0.5*sorted_supplyparam_jan_new_wo_END_or_braces(n,i)
        +(rand(1,1)*sorted_supplyparam_jan_new_wo_END_or_braces(n,i));
    flags(n,1)=1;
  end
```

```
end

  for ii=1:n_rows
     comparison=duplicate_sorted_supplyparam_jan_new_wo_END_or_braces(ii,:)
       ==updated_sorted_supplyparam_jan_new_wo_END_or_braces(n,:);
     if sum(comparison)==7
       for iii=2:8
         duplicate_sorted_supplyparam_jan_new_wo_END_or_braces(ii,iii)=
           duplicate_sorted_supplyparam_jan_new_wo_END_or_braces(n,iii);
         flags(ii,1)=1;
       end
     end
  end
ARRAY_of_supplyparams(:,:,N)=
  duplicate_sorted_supplyparam_jan_new_wo_END_or_braces;
sorted_supplyparam_jan_new_wo_END_or_braces;
%-------------------------------------------------------------------------
n=n+1;
end
updated_sorted_supplyparam_jan_new_wo_END_or_braces=
  duplicate_sorted_supplyparam_jan_new_wo_END_or_braces;
duplicate_sorted_supplyparam_jan_new_wo_END_or_braces;
dlmwrite('duplicate_supplyparam_jan_new_wo_END_or_braces.dat',
  duplicate_sorted_supplyparam_jan_new_wo_END_or_braces,' ');
!cat duplicate_supplyparam_jan_new_wo_END_or_braces.dat |
  awk '{ print "{ "$0" }" }'>! supplyparam_jan_new_wo_END.dat
!echo '<END>' >> supplyparam_jan_new_wo_END.dat
!mv -f supplyparam_jan_new_wo_END.dat supplyparam_jan_new_generated.dat
```

```
!~/Linux dtaparam4to9.dat
!rm __equilibriumUnfinished.dat
!cat simSensorFlows\[04:00:00,04:15:00].dat
     simSensorFlows\[04:15:00,04:30:00].dat
     simSensorFlows\[04:30:00,04:45:00].dat
     simSensorFlows\[04:45:00,05:00:00].dat
     simSensorFlows\[05:00:00,05:15:00].dat
     simSensorFlows\[05:15:00,05:30:00].dat
     simSensorFlows\[05:30:00,05:45:00].dat
     simSensorFlows\[05:45:00,06:00:00].dat
     simSensorFlows\[06:00:00,06:15:00].dat
     simSensorFlows\[06:15:00,06:30:00].dat
     simSensorFlows\[06:30:00,06:45:00].dat
     simSensorFlows\[06:45:00,07:00:00].dat
     simSensorFlows\[07:00:00,07:15:00].dat
     simSensorFlows\[07:15:00,07:30:00].dat
     simSensorFlows\[07:30:00,07:45:00].dat
     simSensorFlows\[07:45:00,08:00:00].dat
     simSensorFlows\[08:00:00,08:15:00].dat
     simSensorFlows\[08:15:00,08:30:00].dat
     simSensorFlows\[08:30:00,08:45:00].dat
     simSensorFlows\[08:45:00,09:00:00].dat >! simulated_counts_0400_0900_.txt
load simulated_counts_0400_0900_.txt
diff = day1_field_counts_0400_0900_ - simulated_counts_0400_0900_;
norm(diff);
ARRAY_of_diff_vectors(:,N)=diff;
ARRAY_of_norm_obj_fn_values(N)=norm(diff);
```

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
N=N+1;
end
%% ##########################################################################
%% #######  Generating the cplx and evaluating obj fn values ends here  #######
%% ##########################################################################
tolerance=0.01;
deviation=1000;
N_outer=0;
while deviation>tolerance & N_outer<N_max
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
ARRAY_of_norm_obj_fn_values;
[max_norm_obj_fn_value, index_of_max_norm_obj_fn_value]=
  max(ARRAY_of_norm_obj_fn_values);
[min_norm_obj_fn_value, index_of_min_norm_obj_fn_value]=
  min(ARRAY_of_norm_obj_fn_values);
sum_for_centroid=zeros(n_rows,n_columns+1);
index_for_sum_for_centroid=1;

while index_for_sum_for_centroid<=n_inner_iterations
sum_for_centroid=
  sum_for_centroid + ARRAY_of_supplyparams(:,:,index_for_sum_for_centroid);
index_for_sum_for_centroid=index_for_sum_for_centroid+1;
end
```

```
centroid_of_all_exc_max=
  (sum_for_centroid-ARRAY_of_supplyparams(:,:,index_of_max_norm_obj_fn_value))
    *(1/(n_inner_iterations-1))
ARRAY_of_supplyparams(:,:,index_of_max_norm_obj_fn_value);
ARRAY_of_supplyparams(:,:,index_of_max_norm_obj_fn_value)=
  ARRAY_of_supplyparams(:,:,index_of_max_norm_obj_fn_value)
    -1.3*(ARRAY_of_supplyparams(:,:,index_of_max_norm_obj_fn_value)
                                          -centroid_of_all_exc_max);
dlmwrite('duplicate_supplyparam_jan_new_wo_END_or_braces.dat',
  ARRAY_of_supplyparams(:,:,index_of_max_norm_obj_fn_value),' ');
!cat duplicate_supplyparam_jan_new_wo_END_or_braces.dat |
  awk '{ print "{ "$0" }" }'>! supplyparam_jan_new_wo_END.dat
!echo '<END>' >> supplyparam_jan_new_wo_END.dat
!mv -f supplyparam_jan_new_wo_END.dat supplyparam_jan_new_bullshit.dat

!~/Linux dtaparam4to9.dat
!rm __equilibriumUnfinished.dat
!cat simSensorFlows\[04:00:00,04:15:00].dat
     simSensorFlows\[04:15:00,04:30:00].dat
     simSensorFlows\[04:30:00,04:45:00].dat
     simSensorFlows\[04:45:00,05:00:00].dat
     simSensorFlows\[05:00:00,05:15:00].dat
     simSensorFlows\[05:15:00,05:30:00].dat
     simSensorFlows\[05:30:00,05:45:00].dat
     simSensorFlows\[05:45:00,06:00:00].dat
     simSensorFlows\[06:00:00,06:15:00].dat
     simSensorFlows\[06:15:00,06:30:00].dat
     simSensorFlows\[06:30:00,06:45:00].dat
```

```
     simSensorFlows\[06:45:00,07:00:00].dat
     simSensorFlows\[07:00:00,07:15:00].dat
     simSensorFlows\[07:15:00,07:30:00].dat
     simSensorFlows\[07:30:00,07:45:00].dat
     simSensorFlows\[07:45:00,08:00:00].dat
     simSensorFlows\[08:00:00,08:15:00].dat
     simSensorFlows\[08:15:00,08:30:00].dat
     simSensorFlows\[08:30:00,08:45:00].dat
     simSensorFlows\[08:45:00,09:00:00].dat >! simulated_counts_0400_0900_.txt
load simulated_counts_0400_0900_.txt
diff = day1_field_counts_0400_0900_ - simulated_counts_0400_0900_;
ARRAY_of_diff_vectors(:,index_of_max_norm_obj_fn_value)=diff;
norm(diff);
ARRAY_of_norm_obj_fn_values(index_of_max_norm_obj_fn_value)=norm(diff);
deviation =
  (max(ARRAY_of_norm_obj_fn_values)-min(ARRAY_of_norm_obj_fn_values))/
                                        min(ARRAY_of_norm_obj_fn_values);
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
ARRAY_to_track_max_min_diff(N_outer+1)=
  max_norm_obj_fn_value-min_norm_obj_fn_value;
ARRAY_to_track_max_obj_fn_value(N_outer+1)=max_norm_obj_fn_value;
ARRAY_to_track_min_obj_fn_value(N_outer+1)=min_norm_obj_fn_value;
N_outer=N_outer+1;
end
ARRAY_to_track_max_min_diff(N_outer+1)=
  max(ARRAY_of_norm_obj_fn_values)-min(ARRAY_of_norm_obj_fn_values);
```

```
ARRAY_to_track_max_obj_fn_value(N_outer+1)=max(ARRAY_of_norm_obj_fn_values);
ARRAY_to_track_min_obj_fn_value(N_outer+1)=min(ARRAY_of_norm_obj_fn_values);
!date >> date.txt
```

Appendix B

MATLAB Code for the SPSA Algorithm

```
!date >! date-spsa.txt
A=60;
a=1;
alpha=0.602;
c=2;
gamma=0.101;
n_SPSA_iter=10;
percentage=0.2;
convergence_tolerance_limit=0.01;
 day1_field_counts_0400_0900_=[ 6 28 29 0 41 65 2 63 0 69 8 67 29 3 43
  2 25 6 2 36 0 25 52 25 8 9 6 7 8 8 123 93 2 1 7 0 3 4 6 4 8 6 10 2 3
  3 4 1 0 402 6 0 0 3 1 11 808 2 9 0 2 6 1 78 2 26 1 0 9 206 29 0 221
  290 2 222 0 223 6 223 127 16 223 0 135 82 8 181 20 162 222 101 13 7
  6 12 14 4 162 4 0 1 6 2 11 8 6 6 6 13 5 1 5 2 1 3 0 16 0 0 0 11 5 11
  426 1 7 0 0 10 4 232 15 47 1 4 21 303 59 0 325 421 0 345 3 335 12
  218 208 19 326 9 216 111 7 332 12 251 385 74 9 13 9 11 10 15 164 14
```

1 2 6 1 7 11 6 15 6 15 12 5 2 4 9 2 0 62 4 0 2 15 4 20 7 7 11 1 0 15
4 333 29 52 0 59 31 368 71 0 405 535 6 428 18 389 2 180 250 40 396 3
278 52 16 321 8 253 412 179 19 11 11 17 20 17 238 7 0 2 11 5 9 14 7
21 9 21 23 4 6 1 8 3 0 10 4 1 6 39 3 23 571 8 21 0 0 45 3 405 51 43
8 95 26 520 67 2 538 663 10 573 13 542 17 247 276 38 547 5 294 115
19 486 0 371 525 205 20 24 14 20 18 18 364 81 2 6 22 5 13 20 15 21 7
35 13 5 2 6 7 5 2 8 6 2 2 25 7 17 34 8 11 1 1 29 10 527 44 63 2 134
50 768 106 1 824 1004 15 875 19 857 51 402 407 53 826 11 474 120 47
678 28 451 826 333 39 45 26 40 38 40 134 70 2 5 27 4 23 25 12 30 9
38 27 5 5 10 17 5 2 70 4 1 0 30 12 29 6 11 24 1 0 34 10 767 78 111 0
185 62 1051 171 1 1134 1412 9 1237 34 1187 28 602 718 119 1137 26
810 106 39 881 40 623 1176 539 40 49 25 75 45 39 193 15 4 9 31 13 26
34 32 47 24 48 56 8 14 14 22 6 2 206 6 1 0 43 12 58 6 10 44 3 0 51
11 1044 119 82 19 242 74 1243 163 3 1301 1640 20 1507 49 1467 23
754 842 180 1321 34 949 232 99 981 20 729 1288 706 43 57 30 90 43 55
216 28 6 19 35 20 32 43 23 75 28 86 64 16 33 8 26 7 4 134 14 2 4 72
14 75 56 21 61 3 0 73 15 1249 184 99 37 316 107 1365 172 3 1471 1725
33 1637 66 1492 45 747 977 157 1465 33 1091 183 54 1210 29 852 1346
240 72 64 41 88 71 73 147 45 4 21 38 20 58 54 49 55 45 75 45 47 22
13 33 12 31 127 33 5 3 42 21 48 194 53 39 3 1 59 24 1549 176 126 7
359 152 1544 185 6 1674 2066 49 1851 64 1777 49 930 1206 209 1650 55
1359 174 76 1455 56 1086 1499 359 74 82 42 50 79 88 130 53 4 26 47
26 88 63 64 89 45 122 64 33 38 13 45 21 7 69 33 3 0 50 39 69 69 31
51 4 2 54 30 1675 220 108 12 397 217 1793 232 8 1895 2419 46 2208
117 1967 122 1086 1443 266 1908 69 1648 299 86 1861 46 1301 1576 485
104 111 79 40 96 125 215 53 10 32 65 27 109 96 36 83 40 114 98 31 46
17 52 31 2 66 29 5 5 83 45 98 99 46 77 2 3 99 50 1920 316 144 13 531
280 1803 285 13 2085 2667 44 2494 151 2036 105 1319 1425 353 2053 82

1659 278 140 1893 99 1316 1502 268 138 158 77 41 130 171 217 82 14
42 82 27 152 114 58 121 55 155 89 56 62 20 66 27 6 92 55 14 4 98 45
98 111 70 69 1 19 118 53 1697 421 175 13 702 300 1892 266 12 2023
2753 34 2465 137 2289 138 1322 1331 290 2057 70 1574 328 135 1926 94
1361 1649 358 144 172 90 43 146 204 204 82 14 43 97 39 168 116 71
120 69 170 86 56 60 31 85 54 6 168 69 11 6 116 78 92 96 82 64 6 15
134 84 1705 429 264 11 760 335 2025 290 15 2235 2883 16 2793 175
2590 198 1483 1447 381 2249 105 1706 420 154 2027 96 1423 1644 772
139 227 98 38 171 225 261 129 34 58 125 60 241 142 84 136 72 200 94
98 71 48 96 97 12 406 101 28 3 131 160 86 97 104 67 6 10 158 157
1730 560 362 19 745 339 1962 307 14 1997 2703 36 2595 151 2435 153
1421 1574 391 2023 134 1851 396 218 1839 86 1355 1688 476 175 267
134 43 185 310 331 219 44 80 147 70 263 152 74 125 58 224 120 104 60
73 116 148 10 137 120 32 10 136 193 110 144 138 97 12 12 160 206
1834 651 484 12 941 349 1672 224 22 1918 2638 32 2835 139 2614 238
1506 1510 466 1853 172 1842 336 306 1572 89 1118 1676 379 177 275
174 47 181 342 371 200 57 113 135 62 249 126 73 152 63 229 168 109
81 74 152 174 8 247 120 25 12 228 258 174 166 138 123 20 11 241 239
1754 621 524 18 968 299 1640 228 16 1721 2372 43 2448 127 2383 217
1396 1448 428 1738 115 1779 406 260 1518 74 1066 1574 677 191 258
135 29 195 328 349 144 47 103 141 64 219 112 82 151 93 197 160 91 62
86 135 146 13 221 105 25 10 185 256 165 125 106 113 10 11 223 200
1758 575 524 14 874 261 1617 240 26 2010 2726 34 2655 160 2453 226
1404 1541 410 1911 154 1774 417 230 1773 66 1164 1611 629 177 252
125 45 186 296 302 132 57 117 137 70 191 125 75 151 78 218 150 98 60
79 175 143 11 295 99 44 12 208 197 165 327 125 126 12 14 241 228
1839 547 524 6 881 305 1587 232 21 1845 2474 56 2534 149 2336 136
1323 1327 314 1753 118 1545 313 195 1729 71 1157 1563 398 160 243

```
  102 45 184 286 288 112 49 95 163 68 167 109 78 121 72 194 126 90 72
  62 139 101 17 290 98 22 18 185 124 142 123 119 90 8 10 219 140 1774
  493 408 4 778 243 1471 244 21 1682 2346 55 2375 178 2265 155 1222
  1384 325 1658 110 1514 317 201 1661 83 1118 1407 526 170 221 96 45
  184 264 484 100 51 88 118 52 161 103 97 129 87 181 104 80 72 39 134
  63 11 632 74 23 14 173 126 130 155 95 84 13 8 225 107 1688 439 318 4
  800 ]';
%% ....and the bad sensors are....
%% ================================
bad_sensor_ids=[6 12 18 20 24 31 50 57 66]';
%% ================================================
%%   this array will be made use of to ignore these
%%    sensors' simulated counts in the optimization

%% setting the 9 bad sensors' counts to 0
%% ======================================
 %% this nested loop sets the bad sensors' counts to 0
 for i=0:19   %% 20 15-minute intervals in the 4 a.m. to 9 a.m. period
  for j=1:9   %% 9 bad sensors in the Irvine network
   day1_field_counts_0400_0900_(bad_sensor_ids(j,1)+i*68 ,1)=0;
  end
 end
%% ================================================
%% done with setting the 9 bad sensors' counts to 0


%% ##########################################################################
%% ############### get a matrix from the supplyparam file ####################
```

```
%% ###### without the the braces, the cap.s, the seg numbers, the <END> ######
%% ##########################################################################
!grep -v '<END>' supplyparam_jan_new.dat >! supplyparam_jan_new_wo_END.dat
!awk '{ line=""; for (i=3;i!=10;i++) line=line" "$i; print line }'
  supplyparam_jan_new_wo_END.dat >!
    supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat
!sort -n -k 1 -k 2 -k 3 -k 4 -k 5 -k 6 -k 7
  supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat >!
    sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat
load sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps.dat
SUPPLYPARAM =
  sorted_supplyparam_jan_new_wo_END_or_braces_or_seg_number_or_caps;
%% ##########################################################################

%% ##########################################################################
%% ############### find the number of groups of segments ####################
%% ##########################################################################
[num_rows num_columns] = size(SUPPLYPARAM);
n_distinct_segs=1;
for row_index = 2:num_rows
 row2=SUPPLYPARAM(row_index,:);
 row1=SUPPLYPARAM(row_index-1,:);
 comparison_result  = row2==row1;
 for parameter_index = 1:7
  if comparison_result(1,parameter_index)==0
    n_distinct_segs=n_distinct_segs+1;
    break;
  end
```

```
 end
end
n_distinct_segs;
%% ##########################################################################
%% ######## done finding the number of groups of segments #################
%% ##########################################################################

%% ##########################################################################
%% ##  UNIX cmds to get an n_segs*8 matrix from the supplyparam file  ##
%% ##########################################################################
!grep -v '<END>' supplyparam_jan_new.dat >! supplyparam_jan_new_wo_END.dat
!awk '{ line=""; for (i=2;i!=10;i++) line=line" "$i; print line }'
  supplyparam_jan_new_wo_END.dat >! supplyparam_jan_new_wo_END_or_braces.dat
!sort -n -k 2 -k 3 -k 4 -k 5 -k 6 -k 7 -k 8
  supplyparam_jan_new_wo_END_or_braces.dat >!
    sorted_supplyparam_jan_new_wo_END_or_braces.dat
load sorted_supplyparam_jan_new_wo_END_or_braces.dat;
THETA=sorted_supplyparam_jan_new_wo_END_or_braces;
[n_rows n_columns] = size(THETA);
%% ##########################################################################
GROUP_indices=zeros(n_distinct_segs,2);
group_starts_at=1;
group_ends_at=2;
while group_ends_at>group_starts_at & group_ends_at<=n_rows
  if THETA(group_starts_at,2)==THETA(group_ends_at,2) &
     THETA(group_starts_at,3)==THETA(group_ends_at,3) &
     THETA(group_starts_at,4)==THETA(group_ends_at,4) &
     THETA(group_starts_at,5)==THETA(group_ends_at,5) &
```

```
     THETA(group_starts_at,6)==THETA(group_ends_at,6) &
     THETA(group_starts_at,7)==THETA(group_ends_at,7) &
     THETA(group_starts_at,8)==THETA(group_ends_at,8)
                                  group_ends_at=group_ends_at+1;
  else GROUP_indices(index_for_GROUP_indices_matrix,1)=
       group_starts_at;group_starts_at=group_ends_at;
 GROUP_indices(index_for_GROUP_indices_matrix,2)=group_ends_at-1;
 index_for_GROUP_indices_matrix=index_for_GROUP_indices_matrix+1;
 group_ends_at=group_ends_at+1;
end
GROUP_indices(index_for_GROUP_indices_matrix,1)=group_starts_at;
GROUP_indices(index_for_GROUP_indices_matrix,2)=group_ends_at-1;
end

%% ######################################################################
g_hat=zeros(n_rows,n_columns); scale=zeros(n_rows,n_columns);
judgement=percentage*THETA; loop_entered=0;
%% ######################################################################
for k=1:n_SPSA_iter
 a_k=a/((k+A)^alpha);
 c_k=c/(k^gamma);

for index=1:3
 delta=2*round(rand(n_rows,n_columns))-1; % generate a matrix of +1s and -1s
 delta_initial = delta;
 delta=percentage*delta.*THETA;  % this single line scales the +1s and -1s
 delta_scaled = delta;
 delta(:,1)=0*delta(:,1); % since we don't want to meddle with the segment
```

```
                        % num.s, we initialize the first column to zero(e)s.
delta_scaled_and_1st_col_zeros = delta;

if loop_entered<2
 for i=2:n_columns
  for j=1:n_rows
   if max(abs(g_hat(:,i)))~=0
    scale(j,i)=judgement(j,i)/max(abs(g_hat(:,i)));
   end
  end
 end
 loop_entered=loop_entered+1;
 array_of_scale_matrices(:,:,loop_entered)=scale;
end

for j=1:n_distinct_segs
 for i=GROUP_indices(j,1):GROUP_indices(j,2)
    delta(i,:)=delta(GROUP_indices(j,1),:);
 end
end
% the above loop ensures that all segments belonging to a particular
% group have the characteristic parameters of that group

THETA_plus=THETA+c_k*delta;
array_of_THETA_pluses(:,:,k)=THETA_plus;
THETA_minus=THETA-c_k*delta;
array_of_THETA_minuses(:,:,k)=THETA_minus;
```

```
dlmwrite('supplyparam_jan_new_wo_END_or_braces_plus.dat', THETA_plus,' ');
!cat supplyparam_jan_new_wo_END_or_braces_plus.dat |
  awk '{ print "{ "$0" }" }' >! supplyparam_jan_new_wo_END_plus.dat
!echo '<END>' >> supplyparam_jan_new_wo_END_plus.dat
!mv -f supplyparam_jan_new_wo_END_plus.dat supplyparam_jan_new_plus.dat

dlmwrite('supplyparam_jan_new_wo_END_or_braces_minus.dat', THETA_minus,' ');
!cat supplyparam_jan_new_wo_END_or_braces_minus.dat |
  awk '{ print "{ "$0" }" }' >! supplyparam_jan_new_wo_END_minus.dat
!echo '<END>' >> supplyparam_jan_new_wo_END_minus.dat
!mv -f supplyparam_jan_new_wo_END_minus.dat supplyparam_jan_new_minus.dat

% ################## y_plus=loss(theta_plus); ####################
!~/Linux dtaparam_jan_new_plus_4to9.dat
!rm __equilibriumUnfinished.dat
!cat simSensorFlows\[04:00:00,04:15:00].dat
    simSensorFlows\[04:15:00,04:30:00].dat
    simSensorFlows\[04:30:00,04:45:00].dat
    simSensorFlows\[04:45:00,05:00:00].dat
    simSensorFlows\[05:00:00,05:15:00].dat
    simSensorFlows\[05:15:00,05:30:00].dat
    simSensorFlows\[05:30:00,05:45:00].dat
    simSensorFlows\[05:45:00,06:00:00].dat
    simSensorFlows\[06:00:00,06:15:00].dat
    simSensorFlows\[06:15:00,06:30:00].dat
    simSensorFlows\[06:30:00,06:45:00].dat
    simSensorFlows\[06:45:00,07:00:00].dat
    simSensorFlows\[07:00:00,07:15:00].dat
```

```
    simSensorFlows\[07:15:00,07:30:00].dat
    simSensorFlows\[07:30:00,07:45:00].dat
    simSensorFlows\[07:45:00,08:00:00].dat
    simSensorFlows\[08:00:00,08:15:00].dat
    simSensorFlows\[08:15:00,08:30:00].dat
    simSensorFlows\[08:30:00,08:45:00].dat
    simSensorFlows\[08:45:00,09:00:00].dat >!
                              simulated_counts_0400_0900_plus.txt
load simulated_counts_0400_0900_plus.txt

%% this nested loop sets the bad sensors' simulated counts to 0
for i=0:19   %% 20 15-minute intervals in the 4 a.m. to 9 a.m. period
 for j=1:9   %% 9 bad sensors in the Irvine network
  simulated_counts_0400_0900_plus(bad_sensor_ids(j,1)+i*68 ,1)=0;
 end
end
%% done with setting the bad sensors' simulated counts to 0

simulated_counts_0400_0900_plus;
day1_field_counts_0400_0900_;
diff_plus = day1_field_counts_0400_0900_ - simulated_counts_0400_0900_plus;
array_of_diff_pluses(:,:,k)=diff_plus;
y_plus=norm(diff_plus);
array_of_y_pluses(index,1)=y_plus;
% ####################################################################

% ################## y_minus=loss(theta_minus); ##################
!~/Linux dtaparam_jan_new_minus_4to9.dat
```

```
!rm __equilibriumUnfinished.dat
!cat simSensorFlows\[04:00:00,04:15:00].dat
     simSensorFlows\[04:15:00,04:30:00].dat
     simSensorFlows\[04:30:00,04:45:00].dat
     simSensorFlows\[04:45:00,05:00:00].dat
     simSensorFlows\[05:00:00,05:15:00].dat
     simSensorFlows\[05:15:00,05:30:00].dat
     simSensorFlows\[05:30:00,05:45:00].dat
     simSensorFlows\[05:45:00,06:00:00].dat
     simSensorFlows\[06:00:00,06:15:00].dat
     simSensorFlows\[06:15:00,06:30:00].dat
     simSensorFlows\[06:30:00,06:45:00].dat
     simSensorFlows\[06:45:00,07:00:00].dat
     simSensorFlows\[07:00:00,07:15:00].dat
     simSensorFlows\[07:15:00,07:30:00].dat
     simSensorFlows\[07:30:00,07:45:00].dat
     simSensorFlows\[07:45:00,08:00:00].dat
     simSensorFlows\[08:00:00,08:15:00].dat
     simSensorFlows\[08:15:00,08:30:00].dat
     simSensorFlows\[08:30:00,08:45:00].dat
     simSensorFlows\[08:45:00,09:00:00].dat >!
                                    simulated_counts_0400_0900_minus.txt
load simulated_counts_0400_0900_minus.txt

%% this nested loop sets the bad sensors' simulated counts to 0
for i=0:19 %% 20 15-min intervals in the 4 a.m. to 9 a.m. period
 for j=1:9 %% 9 bad sensors in the Irvine network
  simulated_counts_0400_0900_minus(bad_sensor_ids(j,1)+i*68 ,1)=0;
```

```
 end
end
%% done with setting the bad sensors' simulated counts to 0

simulated_counts_0400_0900_minus;
day1_field_counts_0400_0900_;
diff_minus = day1_field_counts_0400_0900_ - simulated_counts_0400_0900_minus;
array_of_diff_minuses(:,:,k)=diff_minus;
y_minus=norm(diff_minus);
array_of_y_minuses(index,1)=y_minus;

% ###################################################################
diff = abs(y_plus-y_minus);
percent_deviation=diff/min(y_plus,y_minus);
array_of_diffs(index,1)=diff;
array_of_percent_deviations(index,1)=percent_deviation;
max_percent_deviation=max(array_of_percent_deviations);
% ###################################################################

delta(:,1)=ones(n_rows,1);  % this is to prevent division by zero
g_hat=(y_plus-y_minus)./(2*c_k*delta);
g_hat(:,1)=zeros(n_rows,1); % after the division has been performed,
                            % we can revert the first column to zero
array_of_theta_revision_matrices(:,:,k)=a_k*scale.*g_hat;
array_of_g_hats(:,:,index)=g_hat;

end   %%for index=1:3 loop ends here
```

```
 array_of_max_percent_deviations(:,k)=max_percent_deviation;
 array_of_array_of_y_pluses(:,:,k)=array_of_y_pluses;
 array_of_array_of_y_minuses(:,:,k)=array_of_y_minuses;
 array_of_array_of_diffs(:,:,k)=array_of_diffs;
 average_gradient=array_of_g_hats(:,:,1)+
   array_of_g_hats(:,:,2)+array_of_g_hats(:,:,3);
 average_gradient=0.3333*average_gradient;
 array_of_average_gradients(:,:,k)=average_gradient;
 THETA=THETA-a_k*scale.*average_gradient;

 if max_percent_deviation<convergence_tolerance_limit
   num_iterations=k;
   k=n_SPSA_iter;
 end

end   %%for k=1:n_SPSA_iter loop ends here

!date >> date-spsa.txt
```

Bibliography

[1] M. Van Aerde and H. Rakha. Travtek evaluation modeling study. Technical report, Federal Highway Administration, US-DOT, 1995.

[2] K. Ashok. *Estimation and Prediction of Time-Dependent Origin-Destination Flows*. PhD thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1996.

[3] Ramachandran Balakrishna. Calibration of the demand simulator in a dynamic traffic assignment system. Master's thesis, Departmemnt of Civil and Environmental Engineering, Massachusetts Institute of Technology, 2002.

[4] J. Banks, editor. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter 9: Simulation Optimization. John Wiley & Sons, New York, 1998.

[5] R.F. Benekohal and G. Abu-Lebdeh. Variability analysis of traffic simulation outputs: Practical approach for traf-netsim. *Transportation Research Record*, 1994.

[6] E.P. Box. A new method for constrained optimization and a comparison iwth other methods. *The Computer Journal*, 1956.

[7] M. Carey. A constraint qualification for a dynamic traffic assignment model. *Transportation Science*, 20(55-58), 1986.

[8] Deepak Darda. Joint calibration of a microscopic traffic simulator and estimation of origin-destination flows. Master's thesis, Departmemnt of Civil and Environmental Engineering, Massachusetts Institute of Technology, 2002.

[9] DYNA. DYNA – A dynamic traffic model for real-time applications – DRIVE II project. Annual review reports and deliverables, Commision of the European Communities - R&D programme telematics system in the area of transport, 1992-1995.

[10] FHWA. DTA RFP. Technical report, Federal Highway Administration, US-DOT, McLean, Virginia, April 1995.

[11] M.C. Fu. Optimization for simulation: Theory vs. practice. January 2001; revised May 2001.

[12] M.C. Fu. Optimization via simulation: A review. *Annals of Operations Research*, 1994.

[13] Y.E. Hawas. Calibrating Simulation Models for ATIS/ATMS Applications. *Submitted for publication in Transportation Research*, 2000.

[14] R. He, S. Miaou, B. Ran, and C. Lan. Developing an On-Line Calibration Process for an Analytical Dynamic Traffic Assignment Model. *78th Annual Meeting of the Transportation Research Board*, 1999.

[15] Bruce Hellinga. Requirements for the calibration of traffic simulation models. Department of Civil Engineering, University of Waterloo.

[16] R. Jayakrishnan, Jun-Seok Oh, and Abd-El Kader Sahraoui. Calibration and path dynamics issues in micrsoscopic simulation for advanced traffic management and information systems. *80th Annual Meeting of the Transportation Research Board*, 2001.

[17] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Stat.*, 1952.

[18] J.P.C. Kleijnen. *Statistical Tools for Simulation Practitioners.* Marcel Dekker, 1987.

[19] Mathew Kurian. Calibration of a microscopic traffic simulator. Master's thesis, Departmemnt of Civil and Environmental Engineering, Massachusetts Institute of Technology, 2000.

[20] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis, 3rd edition*, chapter 12: Experimental Design, Sensitivity Analysis and Optimization. McGraw-Hill, New York, 2000.

[21] Hani S. Mahmassani, Ta-Yin Hu, Sriniva Peeta, and Athanasios Ziliaskopoulos. Development and testing of dynamic traffic assignment and simulation procedures for ATIS/ATMS applications. Report DTFH61-90-R-00074-FG, U.S. DOT, Federal Highway Administration, McLean, Virgina, 1994.

[22] Hani S. Mahmassani and Hossein Tavana. Estimation and application of dynamic speed-density relations by using transfer function models. *Transportation Research Record*, 2000.

[23] John L. Maryak and Daniel C. Chin. Global random optimization by simultaneous perturbation stochastic approximation. In *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[24] D.K. Merchant and G.L. Nemhauser. A model and an algorithm for the dynamic traffic assignment problems. *Transportation Science*, 12(183-199), 1978.

[25] D.K. Merchant and G.L. Nemhauser. Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(200-207), 1978.

[26] Ronald T. Milam and Fred Choa. Recommended guidelines for the calibration and validation of traffic simulation models. Fehr & Peers Associates, Inc., Roseville CA 95661.

[27] MIT. Development of a deployable real-time dynamic traffic assignment system. Technical Report Task B-C, Massachusetts Inst. of Tech., Intelligent Transportation Systems Program and Center for Transportation Studies, Cambridge, MA, 1996. Interim reports submitted to the Oak Ridge National Laboratory.

[28] Andrew W. Moore and Jeff Schneider. Memory-based stochastic optimization.

[29] George Ch. Pflug. *Optimization of Stochastic Models: The Interface Between Simulation and Optimization.* Kluwer Academic Publishers, 1996.

[30] H. Rakha, B. Hillinga, M. Van Aerde, and W. Perez. Systematic verification, validation and calibration of traffic simulation models. Department of Civil Engineering, Queen's University; SAIC, McLean, VA 22102.

[31] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 1951.

[32] M.H. Safizadeh. Optimization in simulation: Current issues and the future outlook. *Naval Research Logistics*, 1990.

[33] Stef Smulders, Serge Hoogendoorn, and Tom Alkim. Traffic flow operations during congestion. Technical report, Ministry of Transport, Public Works and Water Management; AVV Transport Research Center, The Netherlands, 2000.

[34] James C. Spall. Multivariate stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[35] James C. Spall. Developments in stochastic optimization algorithms with gradient approximations based on function measurements. In *Proceedings of the 1994 Winter Simulation Conference*, 1994.

[36] James C. Spall. Implementation of the Simultaneous Perturbation Algorithm for Stochastic Approximation. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3), 1998.

[37] James C. Spall. *Wiley Encyclopedia of Electrical and Electronics Engineering, Volume 20*, chapter Stochastic Optimization, Stochastic Approximation and Simulated Annealing. John Wiley & Sons, New York, 1999.

[38] Z. Wall, R. Sanchez, and D.J. Dailey. A general automata calibrated with roadway data for traffic prediction. *80th Annual Meeting of the Transportation Research Board*, 2001.

[39] S. Yagar. Dynamic traffic assignment by individual path minimization and queueing. *Transportation Research*, 5(179-196), 1971.