Improving the Estimation of Travel Demand for Traffic Simulation: Part 1

Final Report

Satya Muthuswamy
David Levinson
Panos Michalopoulos
Gary Davis

Department of Civil Engineering
University of Minnesota

CTS 04-11

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.
CTS 04-11		

4. Title and Subtitle	5. Report Date
Improving the Estimation of Travel Demand for Traffic Simulation: **Part 1**	March 2005
	6.

7. Author(s)	8. Performing Organization Report No.
Satya Muthuswamy, David Levinson, Panos Michalopoulos Gary Davis	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
University of Minnesota Department of Civil Engineering 500 Pillsbury Drive S.E. Minneapolis, MN 55455-0116	11. Contract (C) or Grant (G) No.

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
Intelligent Transportation Systems Institute Center for Transportation Studies University of Minnesota 511 Washington Avenue, SE Suite 200 Minneapolis, MN 55455	Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes

http://www.cts.umn.edu/pdf/CTS-04-11Part1.pdf

16. Abstract (Limit: 200 words)

Many current traffic management schemes are tested and implemented using traffic simulation. An Origin-Destination (OD) matrix is an ideal input for such simulations. The underlying travel demand pattern produces observed link counts. One could use these counts to reconstruct the OD matrix. An offline approach to estimate a static OD matrix over the peak period for freeway sections using these counts is proposed in this research. Almost all the offline methods use linear models to approximate the relationship between the on-ramp and off-ramp counts. Previous work indicates that the use of a traffic flow model embedded in a search routine performs better than these linear models. In this research, that approach is enhanced using a microscopic traffic simulator, AIMSUN, and a gradient-based optimization routine, MINOS, interfaced to estimate an OD matrix. The problem is highly non-linear and non-smooth, and the optimization routine finds multiple local minima, but cannot guarantee a global minima. However, with a number of starting "seed" matrices, an OD matrix with a good fit in terms of reproducing traffic counts can be estimated. The dominance of the mainline counts in the OD estimation and an identifiability issue is indicated from the experiments. The quality of the estimates improves as the specification error, introduced due to the discrepancy between AIMSUN and the real-world process that generates the on-ramp and off-ramp counts, reduces.

17. Document Analysis/Descriptors		18.Availability Statement
Travel Demand	MINOS	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161
Flow	Traffic	
Ramp	Origin-Destination	

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	112	

Improving the Estimation of Travel Demand for Traffic Simulation: Part I

Final Report

Prepared by:
Satya Muthuswamy
David Levinson
Panos Michalopoulos
Gary Davis

Department of Civil Engineering
University of Minnesota

March 2005

Intelligent Transportation Systems Institute
University of Minnesota

CTS 04-11

Table of Contents

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Many current traffic management schemes are tested and implemented using traffic simulation. An Origin-Destination (OD) matrix is an ideal input for such simulations. The underlying travel demand pattern produces observed link counts. One could use these counts to reconstruct the OD matrix. An offline approach to estimate a static OD matrix over the peak period for freeway sections using these counts is proposed in this research. Almost all the offline methods use linear models to approximate the relationship between the on-ramp and off-ramp counts. Previous work indicates that the use of a traffic flow model embedded in a search routine performs better than these linear models. In this research, that approach is enhanced using a microscopic traffic simulator, AIMSUN, and a gradient-based optimization routine, MINOS, interfaced to estimate an OD matrix. The problem is highly non-linear and non-smooth, and the optimization routine finds multiple local minima, but cannot guarantee a global minima. However, with a number of starting "seed" matrices, an OD matrix with a good fit in terms of reproducing traffic counts can be estimated. The dominance of the mainline counts in the OD estimation and an identifiability issue is indicated from the experiments. The quality of the estimates improves as the specification error, introduced due to the discrepancy between AIMSUN and the real-world process that generates the on-ramp and off-ramp counts, reduces.

Chapter 1 – Introduction

Travel demand estimation is one of the most challenging and interesting procedures in transportation engineering. The process of demand estimation is an attempt to understand and predict the behavioral patterns of individuals, and the choices that they make on routes and trips. Over the years, many techniques have evolved to estimate travel demands and in different forms.

Traffic management plans require travel demand estimation. Especially under emergency conditions like accidents, the travel demands are essential to efficiently re-route the traffic. Many of the traffic management methods are developed and tested using simulation so there is an added need for travel demand estimation for use in these simulation applications. Travel demand is also an important element in transportation network analysis. A combination of the knowledge of traveler behavior and demand are essential to study/predict the responses to structural changes in the network.

Traditionally the four-step transportation planning process estimates the travel demands as the number of trip interchanges between traffic zones at a given time in the trip distribution phase. The result is a trip table that represents the number of trip interchanges between the various zones. This matrix representation, known as the Origin Destination (OD) matrix, is the most commonly used form for representing travel demand. The OD matrix can also be expressed as the percentage of trips that flow from each origin zone to another destination zone.

Freeways are one of the most important parts of transportation networks. They have lower travel times and higher speeds, i.e., a better Level of Service (LOS). This attracts more vehicles onto them and hence need efficient traffic management systems to maintain the high LOS. Since travel demand is an important component of all traffic management schemes, its estimation for freeways becomes very consequential.

On freeways, the on-ramps serve as the inputs (origins) and the off-ramps the outputs (destinations). One could estimate OD matrices for freeways that depict the ramp-to-ramp flows. This is very useful information for freeway management strategies such as ramp metering. Travel demand on a freeway can also be represented as input flows and turning percentages at off-ramps. This form of demand representation does not help in traffic re-routing because the destination of a given vehicle is unknown when it enters the network. Re-routing is generally based on destination, which can be obtained from an OD matrix. Hence the travel demand estimated as an OD matrix is more useful.

The applications of an OD matrix for efficient freeway control can be explained as follows. In Figure 1.1, a bottleneck is assumed downstream of ramp 1. If the OD matrix informs that most of the trips from ramp 2 get off on-ramp 3, then this can be used to reduce the ramp metering rates of ramps upstream of the bottleneck, i.e., ramp 1 and ramp 2. In the absence of such information the ramp metering rates of both the ramps will be reduced when not called for.

Thus it is clear that there is a need for the estimation of the OD matrices, to improve traffic management techniques. In addition, since many management techniques are devised, developed, and tested by simulation, these estimated OD matrices are needed for traffic simulation for development, testing, and the implementation of traffic

management strategies. Therefore, this research focuses on estimating freeway OD matrices.

The report has been organized as follows. Chapter 2 is the literature review. Chapter 3 is a description of the adopted methodology and the related issues. Chapter 4 is a detailed explanation of the implementation of the method. Chapter 5 explains the methods used to generate the starting solutions. The test sites and the related results are described in Chapter 6, and Chapter 7 discusses the conclusions of this research.

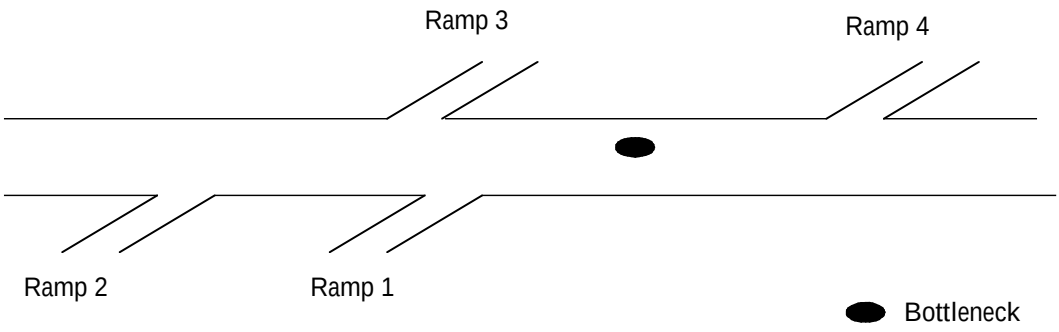

Figure 1.1 - Application Example

Chapter 2 - Literature Review

The earliest techniques for OD estimation were for planning purposes. The OD patterns between places were an essential requirement for the rational planning of new facilities. This was done using s*urveys,* the simplest method of OD estimation. In this method, a random sample of the population was chosen and their travel patterns determined. This was then extended to the population as a whole. The surveying technique could be direct or indirect. But these methods had the common drawback of being very labor intensive and expensive. As time went on, the investment into building new facilities dwindled and there was disapproval toward spending huge amounts in surveys. Hence there was a search for cheaper estimation techniques. There is a very comprehensive analysis of the different survey techniques and the associated accuracy issues in Wills and May (1981).

The first researchers in this area worked on the *gravity model*. The gravity model is a very simple and elegant representation of the spatial distribution of trips. It is based on the analogy to Newton's gravitational law. There is a detailed description of some gravity models in Wills and May (1981). The advantage of using a gravity model was in the reduction of number of unknowns. But the model did not use all the available information, so the results were of limited accuracy. The gravity model, when applied to freeways, is not as efficient as it is more suitable for a more aggregate level.

The link flows on the network is information that is routinely collected. These link flows are a direct result of the travel patterns of the users. Therefore, it was hypothesized that the *travel patterns* can be extracted from the *link flows*. This shifted the OD estimation techniques towards using these traffic counts or link flows as inputs. The network was an important element in the estimation process. The presence of alternate routes called for knowledge of travel behavior also. In addition, there is a need to include an assignment technique. However, for one-route networks (freeways, rail transit, and pipelines) there is no need for an assumption about routing.

The problem

The problem of determining the OD parameters from the traffic counts can be formulated as follows.

$$\sum_i b_{ij} Q_i = O_j \qquad (2.1)$$

$$\sum_j b_{ij} = 1.0 \qquad (2.2)$$

Where,

b_{ij} = proportion of trips from i to j;

Q_i = on-ramp counts (origin flows);

O_j = the off-ramp (destination flows).

Here Q_i and O_j can be observed. Equations (2.1) and (2.2) are representations of flow conservation. Using these equations, one could estimate the OD parameters b_{ij}, but there are not enough equations as variables and there is an *under-specification* problem. This issue was addressed in Robillard (1974). Since there are multiple solutions, the methods focused on estimating a "plausible" or "most likely" OD matrix.
One of the good techniques was the EM (*Entropy Maximization*). This method can be explained with the same example as described in Wills and May (1981). Consider a small freeway section with the flows at the ramps as depicted in the Figure 2.1.

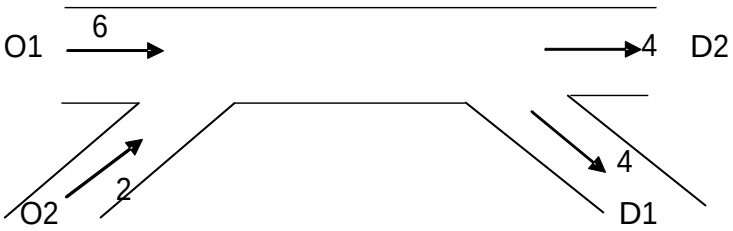

Figure 2.1 – Sample Freeway Section

The problem of the under specification can be clearly seen here.

	D1	D2	sum
O1	4	2	6
O2	0	2	2
sum	4	4	8

Table 2.1 – Possible Trip Table 1

	D1	D2	sum
O1	3	3	6
O2	1	1	2
sum	4	4	8

Table 2.2 – Possible Trip Table 2

	D1	D2	sum
O1	2	4	6
O2	2	0	2
sum	4	4	8

Table 2.3 - Possible Trip Table 3

There are only two independent equations, $T_{11} + T_{12} = 6$ and $T_{21} + T_{22} = 2$ and four unknowns and different trip table tables that satisfy the same equations. As seen in Table 2.1, 2.2, and 2.3. Now under each scenario there is a different combination of vehicles. A combination is defined as a mapping of each vehicle from the origin to the destination. Table 2.4 shows one such possible mapping for trip table shown in Table 2.1.

O1	Destination	O2	Destination
Veh 1	D1	Veh 1	D2
Veh 2	D1	Veh 2	D2
Veh 3	D2		
Veh 4	D2		
Veh 5	D2		
Veh 6	D2		

Table 2.4 – A Sample Mapping Pattern for Table 2.1

By altering the destinations of each vehicle another combination can be generated. Mathematically, using combinatorics, the number of these patterns can be determined. The maximum number of possible states for each system is given in Equation 2.3.

$$E = \frac{\prod_{i} Q_i!}{\prod_{i,j} T_{ij}!} \tag{2.3}$$

The *entropy* (E) is defined as the number of available maps. The EM method tries to estimate the OD table that has the maximum entropy. Van Zuylen and Willumsen (1979) discuss two methods of estimation using the EM method and the Information Minimization approach. The main drawback of the EM method is that it relies on the starting estimate and so there is this need for a good starting solution. Speiss (1987) approached the problem by the Maximum Likelihood Estimation. He proposes a model of the same type, however the starting solution is not optional but an essential part of the estimation. He defines the partially observed OD table cells as Poisson variables with unknown means, and attempts to estimate the underlying means.

As a parallel concept, the idea of posing the estimation processes as a *linear/non-linear programming program* was also being developed. Turnquist and Gur (1979) proposed to define the problem as a constrained optimization problem. They concluded in their research that a good start ensured a good result. Martin and Bell (1992) also proposed a network-programming problem for turning movement estimation at intersections.

The advent of the continual surveillance techniques resulted in the *time series data* for the detector counts. Using these counts, one could estimate an OD matrix (Static method) or track a time-varying OD matrix (Dynamic method). Nihan and Davis (1987) and Nihan and Davis (1991) are examples of the former while Cremer and Keller (1987) and Ashok (1996) are examples of the latter. The first two methods were statistical

models to estimate the central tendency of the OD estimates while the other two used a Kalman filter-based method of tracking the time varying OD matrix.

The general drawback observed by the static methods was that they were very effective for intersections but not as effective in the case of the freeways. Davis (1993) posits the breakdown of the methods occurs because the travel time between the origin and destination was not just a function of the distance but also of the intermediate traffic conditions. In addition, the congestion effects also lead to the traffic exiting the off-ramp to be a mixture of traffic entering the freeway from prior intervals. Davis concluded that there was a need to incorporate a traffic flow model into the estimation process.

Davis (1993) argued that the freeway traffic flow model can be considered a Markov population model. The freeway was broken down into Markovian compartments with vehicles making random exits based on current compartment population. The probability of exit under certain assumptions was related to the space mean speed. Using the number of exiting vehicles in each section, density was calculated. Now, with a speed-density relationship the speed was tracked dynamically. This was the framework for STOMAC (STOchastic MACroscopic simulator).

Davis and Yu (1994) used STOMAC to compare four different estimation methodologies. Two of the methods used, EM (Expectation Maximization) and CAML (Constrained Approximate Maximum Likelihood) were viewed as quasi-ML methods that preserve the simplicity of the linear model and are more efficient than the OLS, but have bias in their estimates. The last technique, NLS (Non-linear Least Squares), differed in that it included the travel time between the origin and destination pair, unlike the OLS, EM, and CAML that ignored that term. In this technique, the STOMAC was embedded into the minimization routine. A starting estimate of the OD matrix was fed into the minimization routine, STOMAC calculated the error sum of squares, and this process was iterated until an optimal OD matrix was estimated. They used 50 different data sets. When compared the statistical properties of the estimates, the NLS estimates were the most efficient and unbiased. Davis and Kang (1994) had a different version of the traffic flow model and the results were similar. Davis's (1993) approach was an application of the above method.

Most of the above methods have some form of linear models that approximate the relationship between the entry and exit volumes. Davis and Yu (1994) indicated that the non-linear model (traffic flow model) outperforms the other methods (based on linear models). After the overview of all the developed techniques, the *OD estimation approach* that had a *traffic simulator* embedded in the *minimization process* seems to be the most attractive choice. This research will further that idea by using an improved traffic flow model and a robust optimization process.

The method adopted in this research enhances the method in Davis and Yu (1994). This research will use a microscopic traffic simulator, AIMSUN, and a state-of-the-art optimization routine, MINOS. The appeal of the method of using a simulator embedded in a search routine is its simplicity. The computation time related to the method will restrict this method to be an offline OD proportions estimation process. Since the method estimates one OD matrix, it will be a static OD estimation process.

Chapter 3 – Methodology

3.1 Introduction

Traffic on a network can be pictured as a *system* where vehicles arrive randomly and traverse certain links of the network and exit the system after a certain time period. This movement of the vehicle from an *origin* (entry point) to a *destination* (exit location) is defined as a *trip*. The arrivals follow a random process and the routes taken are a function of the driver's preferences. The travel times on the chosen route are a function of the traffic conditions. The traffic conditions are a result of the different choices that the drivers make and the actual driving characteristics (traffic flow model). The choice-making process and the actual driving characteristics represent the travel behavior.

If every vehicle in the system is tracked and their start and the end points collected, the aggregated information can be represented using a *trip-table*, an elegant representation of the travel demand. This table represents the underlying *traffic pattern* for the network. In addition to the traffic pattern, if the travel behavior is known, the traffic conditions on the network can be reproduced because they are a direct result of the traffic pattern and travel behavior.

Extending this idea to freeway traffic, the inputs to the system would be a traffic flow model and an OD matrix. On a freeway, there is only one route for every possible trip, so there is no route selection process. The traffic conditions on a freeway are characterized by the on-ramp counts, off-ramp counts and speeds. If we have an appropriate traffic flow model and the OD matrix, the traffic conditions can be reproduced.

If one of the inputs in the system is unknown, but the outputs and other inputs known, the unknown input can be estimated by matching a set of outputs corresponding to a set of inputs to the actual conditions. The OD estimation problem is an example of such a case. The OD matrix is unknown but the traffic conditions—the counts, speeds, and density are known. If an appropriate traffic model is used, the OD matrix can be estimated by trying to reproduce the traffic conditions on the freeway. In other words, when using the traffic flow model, a search for the OD matrix is done in the feasible space of OD matrices and a particular matrix chosen based on its ability to reproduce the traffic conditions. Hence the *OD matrix estimation process* can be defined as an *optimization problem* that searches for the optimal OD matrix that minimizes the deviations of the predicted and the actual traffic conditions.

3.2 The Minimization Problem

An OD matrix is always defined over a *time interval*. The travel demand for a region or a network is defined over a specific time interval, for example, three hours, one day, a week etc. The OD matrix changes with the time scale and region. For a given site (freeway), the OD matrix is time varying. The morning and evening peak have certain characteristics. Now, this underlying *traffic pattern* can be estimated as a time-varying estimate set over a certain time period or one estimate for a shorter time period. The OD estimation is focused on the peak periods because of higher traffic volumes.

An OD matrix can be represented in different forms. The standard form is the trip table, where every cell entry T_{ij} is the number of trips made from origin i to destination j. It can also be represented as a percentage matrix, where every cell b_{ij} is the percentage of trips originating at origin i that will end up at destination j. The latter definition is chosen for reasons explained in the following section. The trip table is the product of the productions at the origins (on-ramp counts) and the percentage OD matrix. Now, using this trip table and a traffic flow model, the traffic conditions can be predicted.

An OD matrix is generally specified for a large geographical region. The area is divided into zones and the OD matrix has trips to and from these zone centroids. For a freeway, the origins are the on-ramps and the destinations are the off-ramps. On most freeways, the OD matrix is upper triangular as the downstream on-ramps cannot feed upstream off-ramps. Also the first origin will be the upstream mainline and the last destination will be the downstream mainline. The input for this system would the on-ramp counts and the percentage OD matrix and the traffic conditions that could be matched would be the off-ramp/mainline counts.

If it is assumed that this percentage OD matrix is constant over the peak period, the OD estimation problem can be defined as the search for that optimal matrix that minimizes the deviations from the actual off-ramp counts. The OD matrix, when defined as a percentage matrix, has to satisfy the constraints that the row sums have to add to 1.0, implying that the sum of trips originating from an on-ramp have to match the on-ramp counts. Therefore the OD estimation problem can be defined as a *linearly-constrained minimization* problem.

In this setup, the upstream mainline and the downstream mainline are treated as the first on-ramp and the last off-ramp respectively. Hence the OD estimation searches for that optimal OD matrix that best matches the off-ramp counts including the downstream mainline. The downstream mainline counts are typically an order or two higher in magnitude than the off-ramp counts. In order to avoid the minimization process from being dominated by the downstream mainline, the sum of the error terms are weighted based on their magnitude. Using the inverse of the standard deviations of the counts as the weighting terms scales the variances equally and removes the domination of the downstream mainline counts. The Non Linear Programming Problem (NLP) can be formally defined as

NLP: Minimize $$\sum_{j}\sum_{t} w_j (O_{tj} - \hat{O}_{tj})^2$$

Subject to $$\sum_{i} b_{ij} = 1.0$$

$$0.0 \le b_{ij} \le 1.0$$

Where,

O_{tj} - Actual off-ramp counts at ramp j in time slice t;

$\hat{O}_{tj}$ – Predicted off-ramp counts at ramp j in time slice t;

w_j – the weight for the ramp j = inverse of standard deviation of O_{tj} ;

i – Origin index;

j – Destination index;

t – Time index.

The solution to the above NLP is the estimate of the OD matrix B that matches the actual off-ramp counts with the greatest accuracy.

3.3. The Time Invariant OD Matrix

The trip table is constantly changing over every time slice because the inputs (on-ramp counts) are time varying. The justification of the assumption of a time invariant OD matrix needs to be addressed. Can the OD matrix be time invariant? This section justifies the assumption of a time invariant OD matrix resulting in a time variant trip table.

Consider Figure 3.1. It is a schematic that explains the underlying process that relates the on-ramp counts and the off-ramp counts. At a very abstract level (Level 1), the whole process can be assumed to be a *Data generation mechanism* that takes the on-ramp counts as inputs and gives the off-ramp counts as the outputs. This process can then be further broken down at Level 2, which involves the creation of the trip table and a *Traffic Flow mechanism.* At the lowest level, the traffic flow process can be broken down as a process that takes in the trip table and calculates the routes and the choice making process and then assigns the trips to the network and propagates the vehicles through the network. This is a conceptual model of the actual process that relates the on-ramp counts and the off-ramp counts.

The OD estimation process involves calculation of the trip table from the observed on-ramp and off-ramp counts. For the best performance of the method, the process as defined in Level 3 must be replicated. The real world process cannot be exactly reproduced because of its complex nature and so a satisfactory approximation is required. The level of satisfaction is related to the need for the approximation and its simplicity. Therefore in the OD estimation process, approximations to the above processes are used.

Approximation to a process is done by making certain assumptions about the process based on the available information and the knowledge at hand. Any assumption made should follow the *principle of parsimony* or *Occam's razor*. It is a logical principle attributed to a medieval philosopher, William of Occam, which states that while trying to explain a phenomenon one must always choose the simplest explanation, one that calls for the smallest leaps of logic.

As shown in Figure 3.1, there are two components that are approximated in the *data generation mechanism*. The first is related to the creation of the trip table from the on-ramp counts and the second is the *traffic flow model.* The latter has been approximated with a microscopic traffic simulator. The first step that relates to the

creation of trip tables from on-ramp counts is approximated by using a time-invariant percentage OD matrix and the on-ramp counts. The choice of the approximation is related to its simplicity and appeal to the intuitive sense of the process and is in line with the above-mentioned principle of parsimony.

As described in Chapter 2, the OD estimation process in one time slice has an identifiability problem as there are more unknowns than equations. Therefore, even over additional time slices, if the assumption is made that the OD matrix is different for each time slice it leads to the same problem. To solve the problem we assume there is a time invariant OD matrix over some sub-set of the multiple time slices. Applying *Occam's razor,* the simplest assumption of the OD matrix being constant over all the time slices is adopted in this research. The following section describes the theory behind such an approximation.

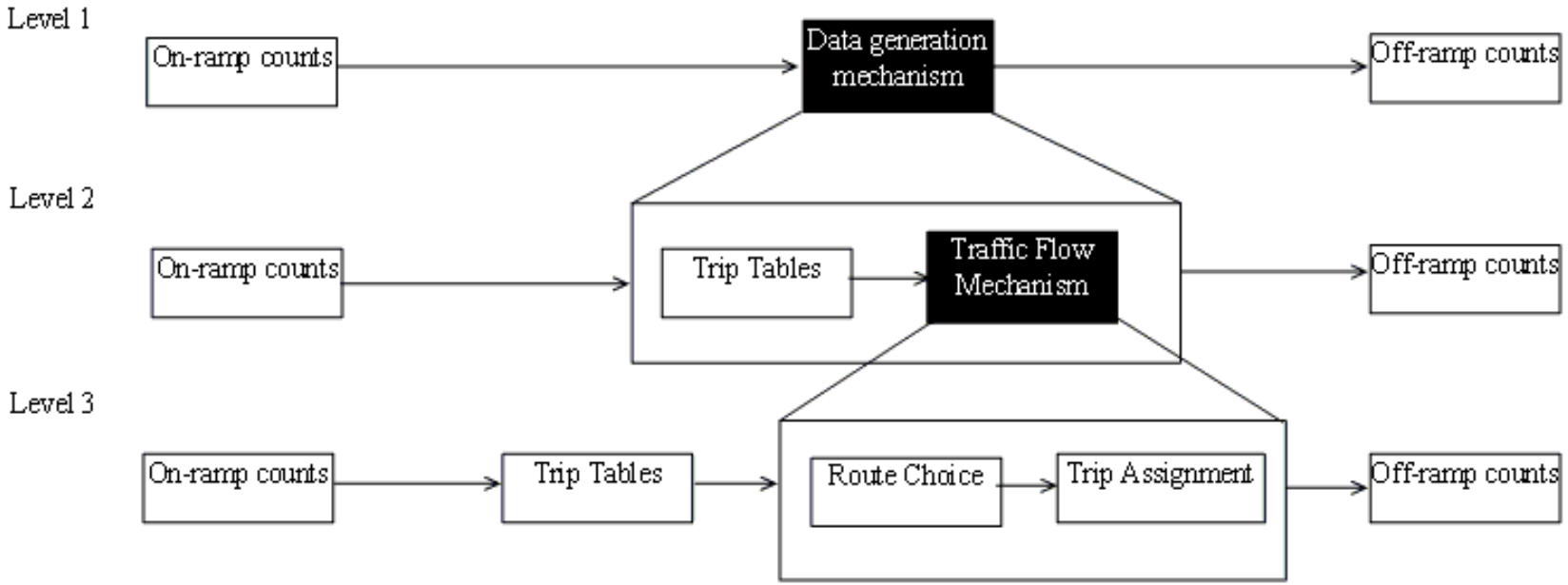

Figure 3.1 Data Mechanics

Before proceeding with the discussion, some concepts on multinomial probability distribution are reviewed. The multinomial distribution is the extension of the binomial distribution. Basically, each random experiment has multiple outcomes, say m. Each individual outcome X_i (i = 1, 2... m) has an associated probability p_i with it. In other words, if the same experiment were to be repeated N times, the N outcomes could be any combination of the m possible outcomes and, if there are n_i observations of outcome X_i, then the joint probability distribution would be as given in equation (3.1).

$$P(X_1, X_2 K X_m; p_1, p_2 K p_m; N) = \frac{N!}{n_1! n_2! \Lambda n_m!} p_1^{n_1} p_2^{n2} \Lambda p_m^{n_m} \qquad (3.1)$$

This idea can be extended to an OD matrix. Consider one row, k, in the OD matrix. The cell entries b_{kj} that represent the percentage of trips from on-ramp k to off-ramp j can be interpreted as the probability that a trip originating at k ends up at j. Now, when a vehicle enters a freeway at on-ramp k the OD matrix entry b_{kj} corresponds to the probability that the vehicle will end up in destination j. In this setup, there are as many outcomes as there are destinations and the associated probabilities for each of the outcomes are the OD matrix row entries b_{kj}.

Consider the following experiment. At any given time interval, for every arriving vehicle at on-ramp k, using the multinomial probabilities given by the OD matrix row entries b_{kj}, a destination is assigned. Based on this assignment, the trip table cell entries T_{kj} are updated. The number of trials for this experiment is the on-ramp count Q_j. The same experiment is repeated with all the rows of the OD matrix. As a result, a trip table for that time interval is generated.

The same set of experiments can be repeated over all the time slices using the same OD matrix and the time varying on-ramp counts. Since the on-ramp counts serve as the number of trials for each individual experiment and they are time variant, the resulting trip table is also time variant.

Using the above argument, the time sliced trip-tables can be visualized as outcomes corresponding to multiple experiments, using the OD matrix as the multinomial probabilities and the on-ramp counts as the number of trials. Therefore, the time varying trip table can be explained as the random outcome of an experiment using a set of fixed multinomial probabilities and time varying number of trials. Thus, using the definition of the OD matrix as a percentage matrix, the idea of a time invariant OD matrix is posited. Since this research focuses on estimation of these percentage OD matrices, all further references to an OD matrix will correspond to this definition, unlike the traditional trip table.

3.4 Issues with Estimation

The main issue with estimation is *identifiability*. The classic example is trying to solve 10 equations with 15 unknowns. Clearly, the solution does not exist as there is not enough information in the system to estimate the unknowns. The OD matrix problem in one time slice has the same nature (as discussed in the literature review). There are always more unknowns than the equations. However, by adding observations, the system is *over-specified*. Since the OD estimation problem has been set up as an over-specified system, it is expected to have enough information to estimate the OD matrix.

Once it has been established that the solution exists and that it can be estimated, the question of whether the current estimation method can arrive at the estimate needs to be addressed. In this setup, the method is a search over the feasible space using a traffic flow model. The two essential components are the traffic flow model and the search routine.

The traffic flow model takes the OD matrix and the on-ramp counts and generates the off-ramp counts. It can be pictured as a map between these inputs (on-ramp counts) and the outputs (off-ramp counts). Mathematically, it can be defined as a map between the OD matrix and on-ramp counts to the off-ramp counts. If this map is one-to-one, then it assures a unique solution to the minimization problem. Also, the traffic flow model needs to be satisfactory in reproducing the traffic conditions and the map must be well-behaved for search routine to converge on a solution.

Given that the map and the traffic flow model have the required properties, the OD matrix can be estimated if the search routine is efficient and robust. Typically most methods are gradient-based and use the Newtonian methods. Based on the nature of the objective function (convex, concave or mixed), there will be local and global optimal solutions. Most gradient-based methods result in local optimal solutions and are functions of the starting solution. Hence, the robustness in arriving at the global optimal solution irrespective of the starting solution is a function of the nature of the objective function.

3.5 The Method

Based on the above sections, the methodology can be defined as a process of using a traffic flow model and a search routine to match the off-ramp counts for a freeway section over the peak period in order to arrive at a time invariant OD matrix estimate that corresponds to the underlying multinomial probabilities.

The off-ramp counts are an aggregate effect of people's choices and driving characteristics. If the traffic flow model can capture these effectively, the confidence in the OD estimate will be higher. AIMSUN is a proven traffic simulator and, since it is accessible and is easy to use, it is chosen as the traffic flow model.

MINOS is a state-of-the-art optimization search routine that can solve a variety of NLP. Since this is a proven, efficient search routine and is accessible, it is chosen as the search routine.

If the OD estimation results in a set of estimates none in which have an exact match, the choice of the best solution could be made based on the closeness to the actual underlying OD matrix. In most cases, the OD matrix is unknown. This problem can be overcome by using a simulated data set. First, an OD matrix and on-ramp counts are assumed and the traffic is simulated in AIMSUN and the off-ramp counts, are observed. Using these simulated off-ramp counts and the assumed on-ramp counts the OD matrix is estimated. If there are multiple estimates, the best solution can be chosen with reference to the assumed OD matrix.

The OD matrix is evaluated based on its ability to reproduce the off-ramp counts. But the usefulness of the estimate is related to its ability to reproduce the traffic characteristics of the system as a whole. Typically, performance is evaluated using the system-wide Measure of Effectiveness (MOEs), like Total Travel, Total Travel Time, Average Speed and Total delay. Hence, the OD estimates will be evaluated not only by

their ability to reproduce the off-ramp counts and the initial OD matrix, but also these system- wide MOE's. This is an important requirement, because in most cases the underlying OD matrix is unknown but these statistics can be collected. Therefore, the performance of the estimates can be evaluated even though the underlying OD matrix is unknown.

Multiple starting solutions (seeds) will be used to overcome the local minima problems as MINOS is a gradient-based algorithm. A seed generation process is also added to the OD estimation method to provide different starting solutions. Chapter 5 describes the different seed generation methods used. The methodology is schematically shown in Figure 3.2.

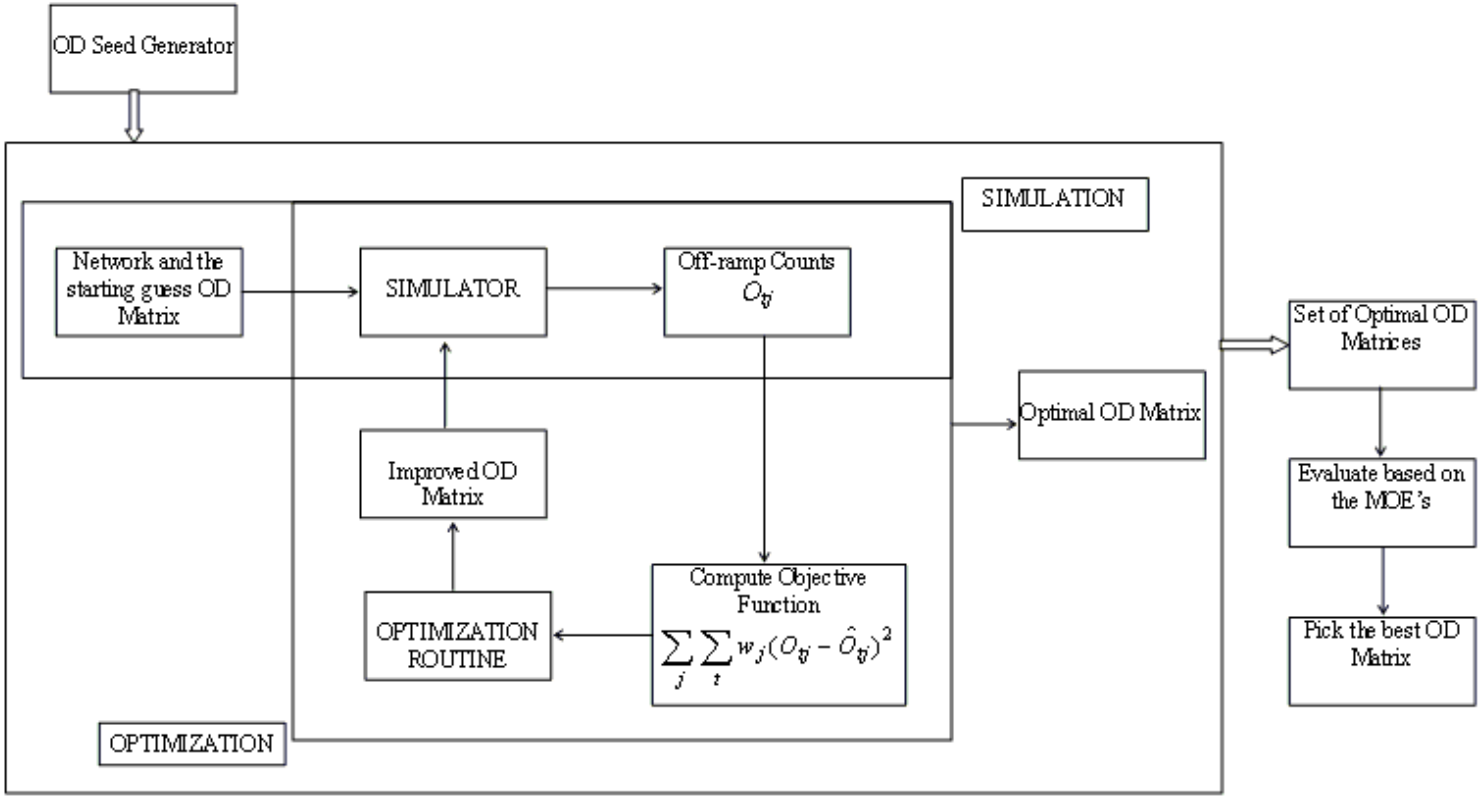

Figure 3.2 - Methodology

Chapter 4 – Implementation

The OD estimation method as described in Figure 3.2 has two working components, the simulator and the optimization routine. The method has been implemented as a FORTRAN77 program, which interfaces these two components to complete the OD estimation process. This chapter is a discussion of the features of these two components and their interface.

4.1 Simulation Component – AIMSUN

AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-urban Networks) is a microscopic traffic simulator that models the behavior of each vehicle in the network, using several vehicle behavior models such as car-following and lane-changing models. AIMSUN uses elements whose states change discretely and continuously over the simulated period. Some of the elements with short continuous variation of states are vehicles and detectors and some of the elements with discrete variation of states are traffic signals and entrance points.

The input data required by the AIMSUN is composed of three categories: Network Description, Traffic Control Plans, and Traffic Demand Data. Network Description contains information about network geometry, layout of sections, and junctions and location of detectors. Traffic Control Plans are different types of traffic controls like traffic signals, give-way signs and ramp metering. Traffic demand data can defined in two different ways in AIMSUN—by the traffic flows at the sections or by an OD matrix.

The Traffic Network Model in AIMSUN consists of a set of road sections connected by nodes. These sections and nodes may be connected to centroids, which can be considered as sources and sinks of traffic. Therefore the network can be coded using sections and nodes. Using these basic elements—sections, centroids, and nodes—a network can be built in AIMSUN. There is an easy-to-use GUI–TEDI (graphical Traffic Editor) that is used to build the networks. For the freeway sections built in this research, *centroids* are *dedicated* to each on-ramp and off-ramp, serving as either origins or destinations for convenience, although the same centroid can be used as a source and sink.

AIMSUN provides different tools for modeling real time traffic controls. It is capable of modeling traffic signals, give-way signs, and ramp metering. In this research for the freeway sections, *no control* was implemented. Depending on the available form of traffic demand data, two types of simulations are possible using AIMSUN: one is based on input traffic flows and turning percentages and the other is based on OD matrices.

The traffic conditions to be simulated defined by an OD matrix should be for each time slice and for each vehicle type. This is generated using an external program that creates the trip table in the AIMSUN-readable format for the simulation. Also, a suitable headway model for generating the vehicles must be specified. In this research since there

are multiple simulations with the same data set, to reduce the stochastic variations in the model, the *constant headway* model is used.

Once a vehicle is generated, the assignment of the vehicle to the objects connected to the centroid can be done in two ways: (1) probabilistic or (2) path to destination-dependent. In the probabilistic approach, the user specifies a proportion of vehicles taking each one of the possible objects connected to the centroid. In the destination-dependant approach, the system decides to which object each vehicle must be assigned. For freeway sections built in this research, there is a centroid for every on-ramp and off-ramp. Hence, both the above-mentioned approaches lead to the same result.

When the simulation is based on OD matrices and route or paths, it is called the Route-Based simulation model. In this model, vehicles are fed into the network according to the demand data defined as an OD matrix and they drive along the network following a certain path in order to reach their destination. There are two modes of Route-Based simulation—Fixed and Variable—depending on whether or not new routes are to be calculated periodically during the simulation. For freeway sections, there is only one route from each on-ramp to off-ramp so there is no need for a route choice model. The data related to a network is stored as a folder by the name of the network file. The specific information of the network is saved in ASCII format. This feature was exploited to build the interface between AIMSUN and MINOS.

4.2 The Optimization Component – MINOS

4.2.1 Introduction

A typical minimization problem can be represented as

Minimize F(x)
Subject to $Ax < b$

Where the *objective function* F(x) depends on variables x that are being minimized and the constraints on x, defined by the constants A and b. The dimension of x and b, n and m respectively, define the size of the problem, i.e., the number of free variables and the number of the constraints that are involved in the problem. In addition, the functional form of the objective function also defines the solution procedure. In theory, the nature of the problem is *linear* or *non-linear* (based on objective function), *constrained* or *unconstrained* (based on whether A is defined), and, based on the form of A, it is *bounded* or *unbounded.*

If the dimension of the problem is small, for example, 1 or 2, the technique of graphically solving the problem can be used, as the objective function and the constraints will be lines, curves, or surfaces and the minima can be visually located. However, if the objective function is complex, it will be a relatively hard task.

Traditionally the gradients help to get a feel for the shape of the function and can be used as a guide to the minimization process. It is known that at minima/maxima, the gradients vanish and the sign of the second derivative determines the nature of that point. Most optimization techniques are gradient-based because, once a point where the gradient vanishes is located, the optimal point is also known. The gradient-based methods are numerous and vary with the type of the problem. However, the gradient-based methods rely on the existence of the gradient, and their efficiencies are higher if the objective function is smooth. Hence, if the nature of the objective function is not suitable for a gradient-based algorithm, a non-gradient based method is needed.

The non-gradient based methods are fewer in numbers than the gradient-based methods but vary in nature significantly. The simplest of nongradient-based methods is the grid search technique, but it is time consuming. The more advanced nongradient-based search techniques are Nelder-Mead, Genetic Algorithm, and Simulated Annealing. As dimensions of the problem (n, m) get larger, matrices are used to represent the information efficiently. In such cases the Jacobian and the Hessian, the matrix representation of the higher order derivatives, are used. Therefore, in all large-scale optimization problems, matrices are an essential part of the data handling.

If the objective function and the constraints were linear, the simplex algorithm can be used. This is an efficient technique to solve linear problems. However, for the non-linear cases there are multiple methods like the reduced gradient, sequential quadratic programming, or a quasi-Newton method to solve the problem.

Most of the methods start at a given point (a feasible point). Then in the n-dimensional space, pick a direction of descent using the gradients in the n-dimensional space. The methods do a line-search along the chosen direction and terminate either at a bound or a constraint. At this new point, the process of choosing a descent direction and then a line search are repeated, until a convergence criterion is satisfied. The essential part of the optimization is the gradient, which will be completely and accurately defined if the actual functional form of the objective function is known. In most cases, the gradient cannot be defined explicitly and so the numerical approximations are used to compute the gradient of the objective function. This can be equated to tweaking each of the n-coordinates of the starting point by a small amount and then computing the gradient with respect to that parameter. In other words, the gradient is approximated as shown in equation (4.1).

$$F'(x) = \underset{\Delta x \to 0}{Lim} \frac{F(x+\Delta x) - F(x)}{\Delta x} \approx \frac{F(x+\Delta x) - F(x)}{\Delta x} \tag{4.1}$$

Where,
x –the vector of current coordinates
Δx –the change in the current coordinates.

The change is made to one variable at a time. The premise is that Δx is small enough to approximate the limit. Therefore, the step-size Δx is very crucial as the numerical approximations can be different based on the nature of F(x).

Given the gamut of methods, the choice of one over the other is based on preference related to access, knowledge, usage etc. MINOS is one of the state-of-the-art

programs for minimization of large scale problems. It was selected since it was accessible and easy to use it was selected. The algorithm implemented in MINOS for minimization of linearly constrained problems is the reduced gradient algorithm. In the following section the salient features of MINOS are discussed and the implementation of the algorithm is explained.

4.2.2 MINOS

MINOS is a FORTRAN-based computer system designed to solve large-scale optimization expressed in the following standard form

NLP:	Minimize	$F(x) + c^T x + d^T y$
	Subject to	$f(x) + A_1 y = b_1$
		$A_2(x) + A_3 y = b_2$
		$l < x, y < u$

Where,
Vectors – c, d, b_1, b_2, l and u and matrices A_1, A_2 and A_3 are constants, F(x) is a smooth function and f(x) is a vector of smooth functions. Ideally the first derivatives need to be provided by the user for F(x) and f(x), otherwise MINOS numerically estimates them.

The objective function as defined here has linear and non-linear variables. In addition, the constraints can also have non-linearities based on form of f(x). Variables l and u represent the bounds on the free variables. Hence the NLP can be *linear, non-linear*, *bounded, unbounded, constrained,* or *unconstrained* based on the forms of the functions described above. MINOS uses *simplex* for solving linear problems. If the objective function has non-linearities and it is linearly constrained, it uses *reduced-gradient* algorithm in conjunction with the *quasi-Newton* algorithm. Finally if there are non-linear constraints, it uses *projected Lagrangian* algorithm.

4.2.3 MINOS Files

This section describes the source files of MINOS. There are 14 FORTRAN files and two input files. The three files that are of importance to the OD estimation program are: Mi05funs.f, Minos.spc, and Minos.mps. These are briefly described here. Appendix A and B describes an example to elaborate the input files for setting up the problem in MINOS.

Mi0Funs.F – The Objective Function

Mi0Funs.F is the file that has the routine defining the objective function in the optimization program. It returns the objective function value for a given value of the free parameters, i.e., the routine returns a number – the value of F(x) for a given value of x.
In the OD estimation process, minimizing the *weighted sum of the squared errors* is defined as the objective. Therefore the subroutine has to return the weighted sum of

squared errors between the actual off-ramp counts and the predicted off-ramp counts corresponding to the current value of the OD matrix. The form of the objective function is given in equation (4.2).

$$F = \sum_{t} \sum_{j} w_j \left(\hat{O}_{tj} - O_{tj} \right)^2 \qquad (4.2)$$

Where,
F – The objective function
w_j – The weight for off-ramp j
$\hat{O}_{tj}/O_{tj}$ – The predicted/actual off-ramp counts

Therefore in each function evaluation, the current OD matrix is used to build a time sliced trip table which is used to simulate traffic in AIMSUN following which the predicted off-ramp counts are extracted to compute the objective function. The routine sets up the input files for AIMSUN, calls AIMSUN, extracts the data, computes the objective function, and returns F.

In addition, X the independent variables vector is a linear array, but the OD matrix is a two-dimensional. So the matrix needs to be stacked into an array. Typically, the OD matrix is an upper triangular matrix as downstream on-ramps cannot feed upstream off-ramps. Therefore, if NOR is the number of origins and NDES the number of destinations, the dimension of X, n is not NOR*NDES but smaller. Therefore the allowed interchanges need to be tracked. Also, since the objective function cannot be explicitly defined, the gradients are numerically estimated.

Minos.spc – SPECS File

This is the specification file that defines the run time parameters. If no parameters are defined the default values are used. A full list of the specs files definitions and the default values are given in the MINOS user manual. Typically, the key parameters that need to be defined in the specs file are the following: *Minimize* (nature of problem); *Nonlinear Variables* (number of parameters); *Super Basics limit* (feature of MINOS); *Derivative Level* (definition of gradients – none/partial/full); *Function Precision* (related to the step size); *Optimality Tolerance* (related to the exit conditions); and *Iterations Limit* (max number of iterations). The Specs file is a text file and can be edited in any text editor.

MPS file – The Data File

This is the user-defined file that defines the name of the variables and constraints, the linear constraints, lower and upper bounds, and the starting values. It is a text file and can be edited in any text editor. In contrast to the free-form of the Specs file, this is a fixed-form file. The entries are to be confined within the specific columns to be meaningful. In the OD estimation program, this file is written before the estimation starts by the program using the information from the starting OD matrix.

4.2.4 Reduced Gradient Method

The gradient is defined as the direction of the change in the function for a change in the free variable. Typically, if there are n free variables, the gradient vector has n components, one for each of the free variables. In a constrained optimization, there is a smaller feasible space than the unconstrained problem. If this additional information of the constraints can be included in the gradients, it reduces the form and this new form is called the *reduced gradient.*

Consider the example shown below,

NLP: Minimize F(x)
Subject to Ax = b
x " 0

If x is split into 2 components v and u then the constraints can be re-written as

Bv + Cu = b (4.3)

Differentiating equation (4.3)

⇒ Bdv + Cdu = 0
(4.4)

$\therefore dv = -B^{-1}Cdu$ (4.5)

So, using equation (4.5) the gradient with respect to u using the chain rule is

$$\frac{dF}{d\mathrm{u}} = \frac{\partial F}{\partial u} + \frac{\partial F}{\partial v}\frac{\partial v}{\partial u} = \frac{\partial F}{\partial u} - B^{-1}C\frac{\partial F}{\partial v} \quad (4.6)$$

This is called the reduced gradient. In matrix notation, if g(x) is the gradient of F(x) then the reduced gradient can be written as $\left[-B^{-1}C \quad I\right] \; g(x)$ and the corresponding step in the optimization can be defined as $\left[-B^{-1}C \quad I\right] \; d\mathrm{u}$. Using the reduced gradient a search direction along w is determined and the step size is given by feasibility conditions. Once the free variables w change is identified, the change to the dependent variables v can be calculated. The OD estimation problem has been setup as a linearly-constrained minimization NLP. Hence the reduced gradient method can be used.

The simplex algorithm is an efficient method of minimizing linear objective functions with linear constraints. The variables are split into two components—basic and non-basic variables—with the non-basic set to the lower bounds and the basic variables assigned values to retain feasibility. Such a solution is called a basic solution. The constraint matrix A is split into a square matrix B whose columns are drawn from A and, as the algorithm proceeds, different columns are replaced until the objective function

cannot be improved. Extending this idea to minimizing non-linear objective functions with linear constraints, an algorithm was developed by Murtagh and Saunders (1978) and is implemented in MINOS. They split the variables as superbasic, basic, and non-basic. The non-basic variables are set to the lower bounds, the superbasic variables are the free variables that govern the optimization, and the basic variables take on values to assure feasibility. Hence the constraint matrix A is split into 3 matrices – B (square), S and N.
The objective of all algorithms is to arrive at an equation to determine the step size from a given feasible point towards the optimal point. The optimal point is the minima/maxima and is a stationary point as the gradient vanishes. The following section describes the method to determine the step size.
At a given feasible point x, if a step Δx is defined, the new point $x + \Delta x$ will be a stationary point if the new step is also on the plane defined by the current set of constraints. The current set of active constraints are defined by fixing the non-basic variables to the lower bounds, hence if the point is stationary, then the change to these has to be null.

$$\Delta x_N = 0 \tag{4.7}$$

Also, the constraint equation

$$Ax = b \tag{4.8}$$

Differentiating equation (4.8)

$$A\Delta x = 0 \tag{4.9}$$

Using the partitions in (4.9)

$$\therefore B\Delta x_B + S\Delta x_S + N\Delta x_N = 0 \tag{4.10}$$

Using equation (4.7) in (4.10)

$$\Delta x_B = -B^{-1}S\Delta x_S$$

$$\Rightarrow \Delta x = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \Delta x_S \tag{4.11}$$

Again, as in the example, using chain rule the gradient with respect to x_s is defined as

$$\frac{dF}{dx_S} = \frac{\partial F}{\partial x_S} + \frac{\partial F}{\partial x_B}\frac{\partial x_B}{\partial x_S} + \frac{\partial F}{\partial x_N}\frac{\partial x_N}{\partial x_S} \tag{4.12}$$

Using (4.11) in (4.12) we get,

$$\Rightarrow \frac{dF}{dx_{\mathrm{S}}} = \frac{\partial F}{\partial x_{\mathrm{S}}} - B^{-1}S\frac{\partial F}{\partial x_{\mathrm{B}}} + 0\frac{\partial F}{\partial x_{\mathrm{N}}}$$

$$\Rightarrow g_A = \frac{dF}{dx_{\mathrm{S}}} = \begin{bmatrix} -B^{-1}S & I & 0 \end{bmatrix} g(x) \tag{4.13}$$

Where, g(x) is the gradient and g_A is the *reduced gradient*.
Now, consider the Taylor Approximation of the function

$$F(x+\Delta x) = F(x) + g(x)^{T}\Delta x + \_\ \Delta x^{T} G(x+\alpha\Delta x)\Delta x \tag{4.14}$$

Where, g(x) and G(x) are the Jacobian and the Hessian respectively.

Now, differentiating equation (4.14) and setting the gradients to zero, gives an equation to calculate a step size such that a stationary point can be located.

$$\begin{bmatrix} g_B \\ g_S \\ g_N \end{bmatrix} + G\begin{bmatrix} \Delta x_B \\ \Delta x_S \\ \Delta x_N \end{bmatrix} = 0 \tag{4.15}$$

Now, multiplying equation (4.15) with $\begin{bmatrix} -B^{-1}S & I & 0 \end{bmatrix}$ we get,

$$\begin{bmatrix} -B^{-1}S & I & 0 \end{bmatrix} G \begin{bmatrix} \Delta x_B \\ \Delta x_S \\ \Delta x_N \end{bmatrix} = - \begin{bmatrix} -B^{-1}S & I & 0 \end{bmatrix} g \tag{4.16}$$

Using (4.11) and (4.13) in (4.16) we get,

$$\begin{bmatrix} -B^{-1}S & I & 0 \end{bmatrix} G \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \Delta x_S = - g_A$$

This equation gives the Newton step size Δx_S for the algorithm. The term on the left side is the reduced Hessian. The vector $\begin{bmatrix} -B^{-1}S & I & 0 \end{bmatrix}$ is denoted as Z.

The implementation of the algorithm is a bit more complicated as it calls for efficient methods of handling the data and matrices. The *basic* variables matrix B is factorized into a lower and upper triangular matrix (B = LU). Also the reduced Hessian (Z^TGZ) as defined above is never calculated and a quasi-Newton approximation R^TR is used. The computation of the reduced gradient is done in two stages. The equation $B^T\pi=g_B$ is solved and the resulting π is used to calculate $g_A = g_S - S^T\pi$. The steps of the algorithm as described in Beck, Lasdon and Engquist (1983) are in Appendix C.

4.3 The Interface

The OD estimation method is an optimization problem with an embedded simulator. The preceding sections explained the structure and working of the simulator – AIMSUN and the optimization program – MINOS. This section describes the interface, the means to facilitate the information exchange between these components. The process of building the interface consists of identifying the information exchanged, setting up the means to complete the exchange, and ensure that it is simple and transparent.

The OD estimation process is a search in the feasible space of OD matrices by MINOS using the objective function value and the estimates of the gradient. The objective function is evaluated using the sub-routine Funobj described in section 4.2.3. Each evaluation includes using the current estimate of the OD matrix for a simulation run in AIMSUN and calculating the objective function using the predicted counts from that run and the actual counts.

The major difference between MINOS and AIMSUN is that MINOS is in source code form in FORTRAN77 whereas AIMSUN is an executable. The interface building would be easier if both were in source code forms. Based on the current configuration, it can be setup in two forms. The interface could be built as third-party mediating information between the components or as a built-in feature in MINOS. This is depicted in Figure 4.1. The interface could be an external program that can be called by Funobj to run AIMSUN and it returns the counts (third-party type) or a FORTRAN77 code inside Funobj that calls AIMSUN and extracts the counts (built-in type). The built-in type of interface is more elegant and has a lower overhead in terms of code. Hence it was chosen and the information interchange between AIMSUN and MINOS was setup inside the Funobj subroutine inside MINOS.

AIMSUN needs a network and a time sliced trip table for a simulation run. MINOS needs the actual counts and the predicted counts corresponding to the current estimate of the OD matrix. Hence the information exchanged between the two components is the current OD estimate from MINOS to AIMSUN and the predicted counts from AIMSUN to MINOS.

Third-Party Type Built-in Type

Fig 4.1 – Interface Types

AIMSUN stores the relevant information of a network and the counts from a simulation run in ASCII text format. The interface sends information into AIMSUN by writing into these files and extracts the counts from these files to pass into MINOS. The typical steps in the execution inside Funobj are the following:

1. Write the current OD matrix into a file.
2. Read the OD matrix and on-ramp counts and generate the trip tables.
3. Write the trip table for every time slice into AIMSUN readable format.
4. Call AIMSUN.
5. Read the AIMSUN detector files and extract the off-ramp counts.
6. Calculate the objective function.

Step 1 and Step 5 are simple file writing- and reading-processes. Taking the current estimate of the OD matrix and multiplying it with the time slice on-ramp (input) counts creates the time sliced trip table, which is done in step 2. The resulting trip table needs to be written into a specific format for AIMSUN to read. The important point to note is that the elements of the OD matrix are in real numbers but the on-ramp counts and the trip table elements are integers. Hence there is a rounding operation in the product between these matrices. Steps 2 and 3 are implemented in a separate subroutine trip table that reads the OD matrix, creates the time sliced trip table, and writes into the needed format for AIMSUN. The call to AIMSUN is done from within MINOS by transferring control to the system and running the simulator and then returning the control to the program. The final step of extracting the predicted counts is done using a separate subroutine "getoff." The steps are described in detail in Figure 4.2.

The files required for the OD estimation program can be classified into categories, one related to the simulator and the other to the optimization program. The latter requires 14 MINOS source code files, three user-defined source code files (to setup the problem, write the trip table and read the counts), which are compiled and linked into one executable file. In addition, there are the data files needed for the information interchange related to the counts and starting OD solutions. The simulation component is associated with a folder that has the network built using TEDI, the AIMSUN console version and the scenario file to run the simulation.

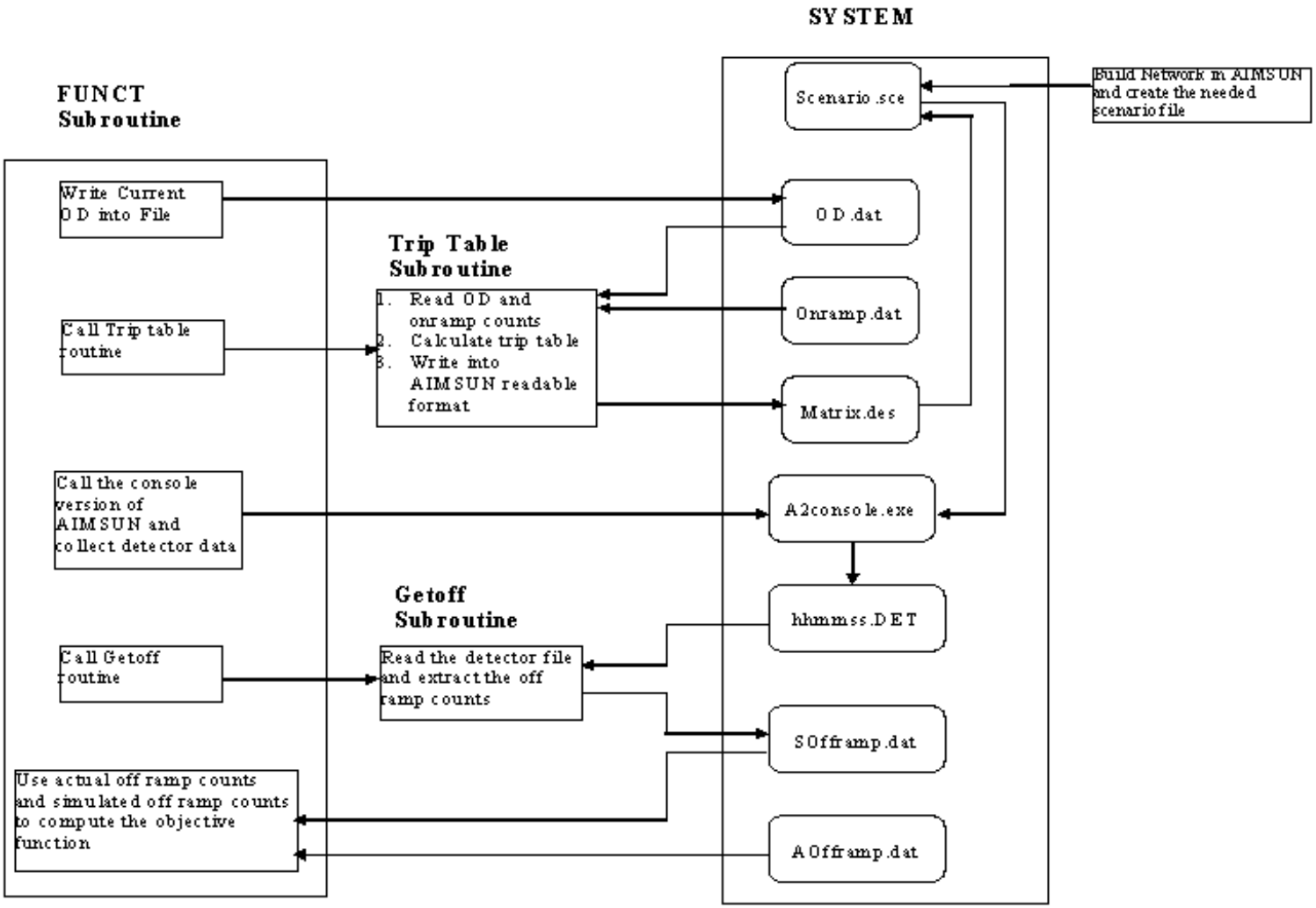

Figure 4.2 Interface

Chapter 5 – Generation of the Initial Solutions

The OD estimation problem has been posed as an optimization problem aimed at minimizing the weighted sum of squares of the deviation of the predicted off-ramp counts to the actual off-ramp counts. The method being used to search over the solution space is the reduced gradient algorithm as implemented in MINOS. Typically the search can start at any feasible point and proceed from there to finding the optimal point. Most search routines start at one of the bounds and search from there onwards. The efficiency of the search is a function of the nature of the problem, the algorithm, and the starting point.

Considering that the first two factors are held constant, a faster result can be expected if a start is made in the vicinity of the optimal solution. It is important to note that if the algorithm is robust, even a bad start will result in the optimal solution. However, the speed of the convergence will be higher if we can start closer to the optimal solution. Hence it becomes crucial to make a good guess of the solution using some techniques and then let the method search the space for the optimal solution from that point.

The starting solution (seed) is generally an "educated guess" of the solution. To make that guess, the available information on the problem must be used. In the OD estimation problem, the detector data – on-ramp counts, off-ramp counts, and mainline counts—are readily available and can be used. In addition, the travel time and trip length details can be extracted from the geometry. Using this information, five different methods have been described in this chapter to estimate the starting solution for the minimization problem. The reason for the choice of these techniques is based on simplicity, access, and ease of implementation.

5.1 The Equally Split OD Matrix

This is the simplest method of seed generation. As the name suggests, it assumes that all destinations are equally likely; hence it assigns proportions to all possible destinations equally. As an illustration, consider the example freeway section as shown below in Figure 5.1.

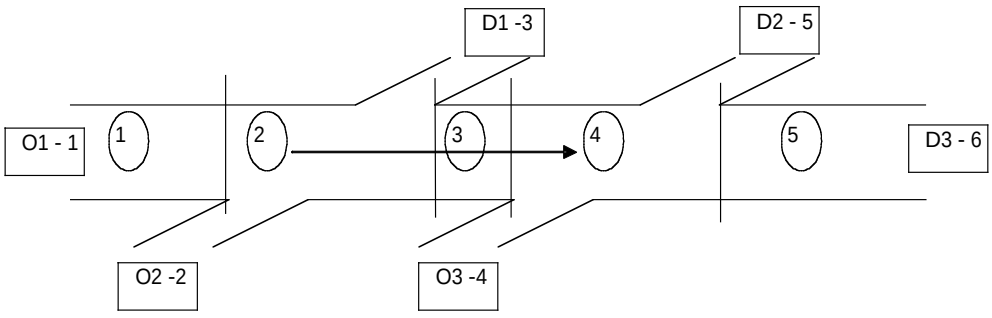

Figure 5.1—Example Freeway

In this case there are three origins, three destinations, and eight possible interchanges. Now this method concludes that D1, D2 and D3 are equally likely destinations for trips originating at O1 and O2, so the OD proportions will be 1/3 (33.33%). And, for O3 since it has only two possible destinations (D2 and D3), the proportions will each be _ (50%). Following the convention in the OD estimation program, the impossible interchanges are represented with a -1, an indication that no trips originating at the corresponding origin can go to the corresponding destination. Using these rules, the OD matrix generated is shown in Table 5.1.

O/D	D1	D2	D3
O1	1/3	1/3	1/3
O2	1/3	1/3	1/3
O3	-1.0	_	_

Table 5.1 – The Equally Split OD Matrix

5.2 Proportional OD Matrix

This is the most commonly used and oldest method to estimate an OD matrix. It is based on the concept that the attraction at any destination is a function of the number of trips that end at that destination. In other words, the higher the number of trips ending at a destination, higher the proportion of trips it attracts from all the origins. Hence if we have three possible destinations, then the proportions are assigned based on the total number of trips each of them attracts. A feature of this estimate is that it is independent of the number of origins. This is explained using the same example as shown in Figure 5.1 and the data in Table 5.2.

S. No	Item	Value
1	Section Lengths (in meters)	200, 300, 50, 250, 400
2	On-ramp Counts	375, 25 and 100
3	Off-ramp Counts	30, 70 and 380
4	Mainline Counts	370, 390, 350, 440 and 370

Table 5.2 – Freeway Example data

Now, starting with row 1 in the OD matrix, Origin 1 can feed all the three possible destinations which have attractions – 30, 70, and 380. Hence the elements in the row are 30/(30+70+380), 70/(30+70+380) and 380/(30+70+380). Similarly, the elements in row 2 and row 3 are estimated. The OD matrix with the intermediate calculations is shown in Table 5.3.

O\D	D1	D2	D3
O1	=30/(30+70+380) = 0.063	=70/(30+70+380) = 0.146	=380/(30+70+380) = 0.791
O2	=30/(30+70+380) = 0.063	=70/(30+70+380) = 0.146	=380/(30+70+380) = 0.791
O3	-1.0	=70/(70+380) = 0.156	=380/(70+380) = 0.844

Table 5.3 – The Proportional OD matrix

5.3 Iterative Method

This is a method that has been adopted from Wills and May (1981). It is based on an iterative proportional fitting algorithm developed by Deming and Stephan (1940). It can be viewed as a hybrid proportional assignment technique that balances the inflows with the outflows. The algorithm iteratively adjusts the cells of the OD matrix proportional to the row and column sum until convergence is reached. The steps of the algorithm as seen in that publication are given below.

Step 0

Set k = 0

$T_{ij}^{(0)} = 1$ for all possible interchanges
0 for all impossible interchanges

Step 1

Set $T_{ij}^{(2k+1)} = \frac{O_i^{'}}{\sum_j T_{ij}^{(2k)}} T_{ij}^{(2k)}$ For all i, j

Where, O'_i is the observed volume at point i adjusted for all known demands from i.

Step 2

Set $T_{ij}^{(2k+2)} = \frac{D_j^{'}}{\sum_i T_{ij}^{(2k+1)}} T_{ij}^{(2k+1)}$ For all i, j

Where, D'_j is the observed exiting volume at point j adjusted for all known trips that end at j.

Step 3

If $| T_{ij}^{(2k+2)} - T_{ij}^{(2k)} | < \delta$ for all i, j then STOP

Else set k = k + 1 and go to Step 1.

To illustrate the working of the algorithm, the same example as described in Figure 5.1 and data in Table 5.2 are used. There are three matrices to track during the iterations and also the final OD matrix is a trip-interchange matrix rather than a percentage matrix. The intermediate steps are shown and the final OD matrix is shown in Table 5.4.

Iteration – 1

OD 1			OD 2			OD 3		
1	1	1	125	125	125	29.29	49.71	269.8
1	1	1	8.33	8.33	8.33	1.95	3.31	17.99
0	1	1	0	50	50	0	19.8	107.9

Iteration – 2

OD 1			OD 2			OD 3		
125	125	125	29.29	49.71	269.8	25.13	40.45	315.5
8.33	8.33	8.33	1.95	3.31	17.99	7.290	5.280	10.79
0	50	50	0	19.8	107.9	0.000	32.40	63.30

Iteration – 3

OD 1			OD 2			OD 3		
25.13	40.45	315.5	25.13	40.45	315.5	25.13	40.45	315.5
7.290	5.280	10.79	7.290	5.280	10.79	7.290	5.280	10.79
0.000	32.40	63.30	0.000	32.40	63.30	0.000	32.40	63.30

O/D	D1	D2	D3
O1	0.0659	0.1061	0.8278
O2	0.3120	0.2262	0.4616
O3	-1.0	0.3385	0.6615

Table 5.4 – The Iterative OD matrix estimate

5.4 The Gravity Model

The gravity model has been one of the oldest trip distribution methods. Although the gravity model is a macroscopic model, it can be extended to freeways to determine the proportion of trips getting off at each ramp. The main parameter in the Gravity model is

the impedance function. The model for the impedance function was proposed by Nancy Nihan and it incorporated the impedance function proposed by Voorhees. This is used as it is stated in Wills and May (1981). It was a model based on the Gamma distribution. It is related to the concept that the probability of very long and very short trips is low on the freeway. The gamma function has a shape that is similar to this assumption. The model that she proposed was the following:

$$F_{ij} = \frac{\beta^{\alpha}}{\Gamma(\alpha)} d_{ij}^{(\alpha-1)} e^{-\beta d_{ij}} \qquad (5.1)$$

Where,

F_{ij} is the travel propensity factor between ramp i and j.
α = shape factor $\cong$ 1.5
β = size parameter = α /avg. trip length
d_{ij} = distance between pair (i, j)
avg. trip length = _ = (1/T)*Σ_k (Link length)*(Link Volume)
T = sum of all trips generated

The cell entries in the OD matrix are defined as

$$T_{ij} = \frac{b_j F_{ij}}{\sum_j b_j F_{ij}} Q_i \qquad (5.2)$$

Where,

T_{ij} = trip interchange between pair (i, j)
b_j = balance factor from iterations
Q_i = production at i
D_j = attraction at j

Subject to the constraint $\Sigma_i T_{ij} = D_j$

In the implementation of the algorithm, the balancing factor was ignored. This was done as a simplifying step because only a starting solution was needed rather than an accurate estimate. Again using the same example in Figure 5.1 and the data from Table 5.2 this method is elucidated. The α is assumed to the average trip length from the geometry = (370*200+390*300+350*50+440*250+370*400)/(375+25+100) = 933m. The last parameter of the gravity model $\beta = \alpha/933 = 0.0016$. Using these parameters and the distance matrix (Table 5.5), the impedance factors are estimated and shown in Table 5.6.

O/D	D1	D2	D3
O1	500	800	1200
O2	300	600	1000
O3	0	250	650

Table 5.5 – The Distance matrix

O/D	D1	D2	D3
O1	0.0008325	0.0008122	0.0005050
O2	0.0005701	0.0008692	0.0006671
O3	0	0.0004650	0.0008686

Table 5.6 – The Impedance matrix

The resulting OD matrix is given in Table 5.7.

O/D	D1	D2	D3
O1	0.1793	0.3328	0.4878
O2	0.1093	0.3170	0.5736
O3	-1.0	0.1850	0.8150

Table 5.7 – The Gravity Model OD matrix

5.5 Turning Percentage

This is the most intuitive method of estimating an OD matrix for a freeway section. The method, as the name suggests, is based on the turning percentages. The underlying idea is that at any given off-ramp, the turning percentage is independent of the trip origin. Hence by tracking the turning percentages in each section, we can back-calculate the OD matrix percentages. The last section has the turning percentage = 100% as it is the final destination. The method is illustrated using the example from Figure 5.1 and Table 5.2.
The turning percentage in each section is first calculated. A section is defined as a portion of the freeway that is between on-ramps and off-ramps such that the upstream end of the section is an on-ramp and the downstream end is an off-ramp. Using this convention, there are five sections. The turning percentages in them are as follows: 0, 7.69, 0, 15.9, and 100. The OD cells are assigned by starting at every origin and proceeding downstream assigning the OD cells with the turning percentages at each off-ramp and tracking the exited vehicles. The resulting OD matrix calculations are shown below in Table 5.8.

O\D	D1	D2	D3
O1	0.0769	=0.159*(1-0.0769) = 0.1468	=1-0.159-0.0769 = 0.7763
O2	0.0769	=0.159*(1-0.0769) = 0.1468	=1-0.159-0.0769 = 0.7763
O3	-1.0	0.159	=1.0-0.159 = 0.841

Table 5.8 Turning Percentage OD matrix

5.6 Implementation

The seeds are OD matrix estimates from the data for the freeway section for a given time interval. In the OD estimation program there is data for every five minutes over three hours. Hence, in the actual seed generation process, the cumulative data is used for making the estimates. In other words, the data used for the seed generation (Table 5.2) is the aggregated value of the data from all the time intervals.

All the above methods have been implemented in the form of programs written in FORTRAN77. The choice of using FORTRAN77 is used because MINOS is in FORTRAN77 and so, for compatibility and uniformity, the language was chosen over a later version like F90 or C/C++. They serve as an external input to the OD estimation programs. The seeds are generated and fed into the OD estimation program and OD matrix estimates are obtained.

Chapter 6 – Test sites and Results

This chapter describes the test sites, the data, the different experiments conducted on these sites and their results. Two test sites are used for the OD estimation method. The first site is a one on-ramp, one off-ramp freeway section primarily used to test the program. The second site is a real freeway section, TH-169.

6.1 Case 1 – 2 Origin, 2 Destination

Figure 6.1 shows the first test site for the OD estimation method. It is a hypothetical freeway section that has one on-ramp and one off-ramp, two origins and two destinations. The OD estimation method has four parameters to estimate. It must be noted that there are only two free variables, as there are two constraints.

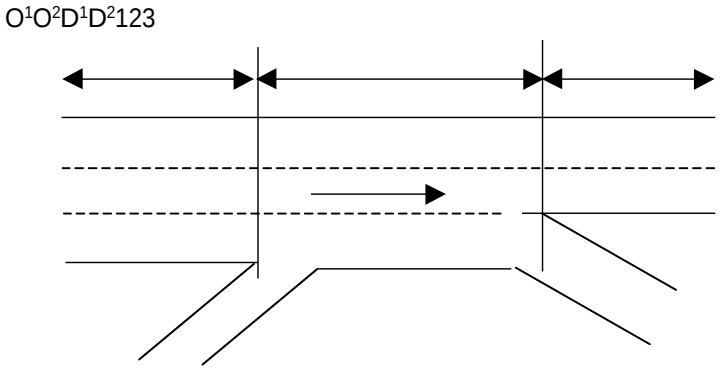

Figure 6.1 - First Test Site

The section parameters are shown in Table 6.1.

section #	Length (m)	#lanes	v-max (km/h)	Capacity (vplph)
1	488	3	116	2100
2	559	3	116	2100
3	356	2	116	2100
Total	1403			

Table 6.1 – Section Parameters

6.1.1 Data

Since this is a hypothetical freeway section, the data set was generated as follows. An OD matrix and some on-ramp counts were assumed. The site coded in AIMSUN, traffic is simulated, and the off-ramp counts are measured. Now these measured off-ramp counts and the assumed on-ramp counts are the data set used to estimate an OD matrix. The off-

counts are collected every five minutes and the simulation period is three hours (representing the peak period). This process was used to generate traffic data for five different days. The data set used for the estimation is shown in Appendix D. The data generation process is shown in Figure 6.2.

The idea to use AIMSUN to construct the data set had the advantage of removing the specification error in the estimation because the model that generated the data will be the same model used in the OD estimation. So the error in the estimate introduced due to the discrepancy in the two models is removed, unlike with real data where AIMSUN is an approximation to the real world.

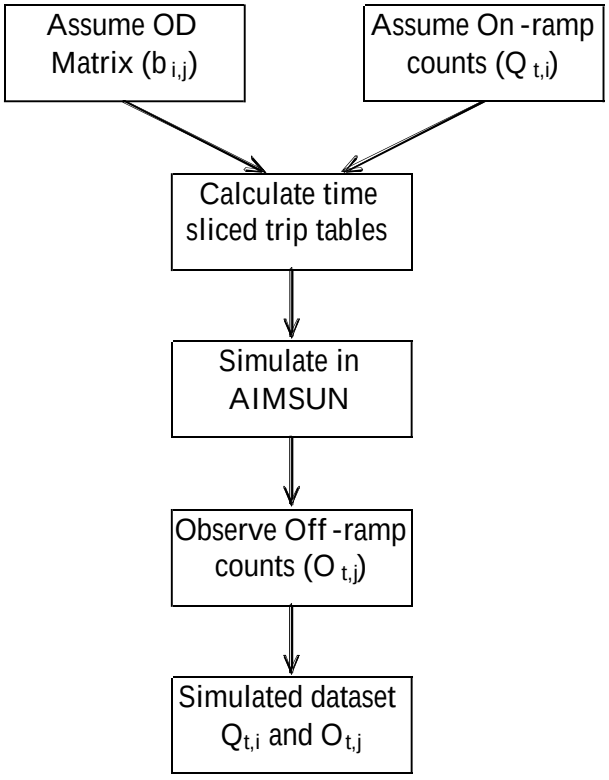

Figure 6.2 – Data generation process (simulated data set)

6.1.2 Results

The results of the OD estimation are shown in Table 6.2, which has the actual OD matrix, three starting solutions and their corresponding final estimates, the starting and final values of the weighted sum of squared errors, and the R^2 for the two destinations with respect to each of the three final solutions.

The first observation is that the OD estimation method has been implemented "bug-free" and that the method works, a conclusion from the convergence in the objective function. The results indicate that the final OD estimates are different from each other and the assumed OD matrix.

Actual OD

	d1	d2
o1	0.3250	0.6750
o2	0.2500	0.7500

Seed			Solution		
1					
	d1	d2		d1	d2
o1	0.5000	0.5000	o1	0.3097	0.6904
o2	0.5000	0.5000	o2	0.5000	0.5000
2					
	d1	d2		d1	d2
o1	0.3220	0.6780	o1	0.3235	0.6765
o2	0.3220	0.6780	o2	0.4051	0.5949
3					
	d1	d2		d1	d2
o1	0.5646	0.4354	o1	0.3199	0.6801
o2	0.4646	0.5354	o2	0.4367	0.5633

Weighted Sum of Squares

	Start	End
seed		
1	253.67	2.43
2	3.10	2.34
3	310.25	2.39

R-squared

	D1	D2
seed		
1	0.9535	0.9856
2	0.9617	0.9766
3	0.9565	0.9775

Table 6.2 – OD Estimates, 1-day 3hr simulation

However, they all have low objective function values and also high r-squares, an indication that, in terms of reproducing the counts, the OD estimates perform similarly. The issue of multiple solutions needs to be addressed. The logical step that followed was to investigate the nature of the objective function to explain the above results. Before proceeding to investigate, another experiment was conducted using all five days of data generated.

6.1.3 Multiple Days

The idea of using multiple days for the estimation process is related to the assumption that the inability of the method to estimate the OD matrix is lack of sufficient information (identifiability). Hence by providing the additional day's data, maybe there is some additional information in the system to help identify the OD matrix. The results from this case are shown in Table 6.3. The format of the data in Table 6.3 is the same as in Table 6.2.

Actual	OD	d1	d2
	o1	0.3250	0.6750
	o2	0.2500	0.7500

Seed			Solution		
1					
	d1	d2		d1	d2
o1	0.5000	0.5000	o1	0.3152	0.6848
o2	0.5000	0.5000	o2	0.5000	0.5000
2					
	d1	d2		d1	d2
o1	0.3220	0.6780	o1	0.3262	0.6738
o2	0.3220	0.6780	o2	0.3234	0.6766
3					
	d1	d2		d1	d2
o1	0.5646	0.4354	o1	0.3183	0.6817
o2	0.4646	0.5354	o2	0.4211	0.5789

Weighted Sum of Squares

	Start	End
seed		
1	1342.17	13.78
2	19.63	14.54
3	1664.08	14.30

R-squared

	D1	D2
seed		
1	0.9477	0.9749
2	0.9441	0.9792
3	0.9441	0.9762

Table 6.3 – OD Estimates, 3-days 3hr simulation

Again, as seen in Table 6.2, the OD matrix estimates are all different from each other but reproduce the counts well, as indicated from the high r-squares and the low objective function values. An interesting point to note is that, in both cases (Table 6.2 and Table 6.3), the downstream mainline (D2) is matched better than the off-ramp (D1). Also, the first row of the OD matrix (the upstream mainline) is estimated better than the second row (on-ramp). An important observation is that the mainline tends to dominate the estimation process.

6.1.4 Nature of the Objective Function

The advantage of this site is that there are only two independent variables (four free variables and two linear constraints). This opens the possibility of plotting the objective function across the feasible space in three dimensions.

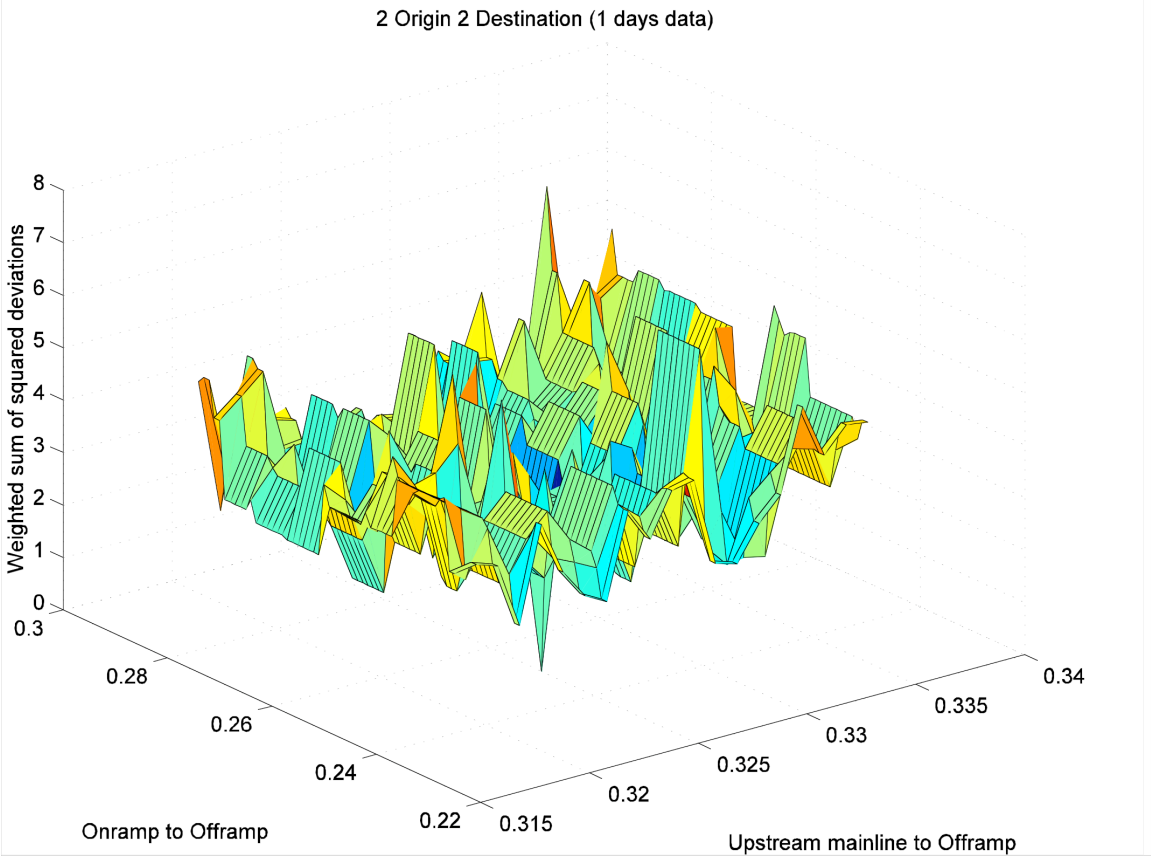

Figure 6.3 – Objective Function 1-Day

This plot will give insight into the nature of the objective function and help explain the performance of MINOS in minimizing the objective function from the shape/surface of the objective function over which MINOS searches to find the optimal OD matrix. So the objective function was calculated for a range of values around the actual value of the OD matrix using one, two, three, four, and all five days of data. The plots are shown below in Figures 6.3, 6.4, 6.5, 6.6, and 6.7 respectively.

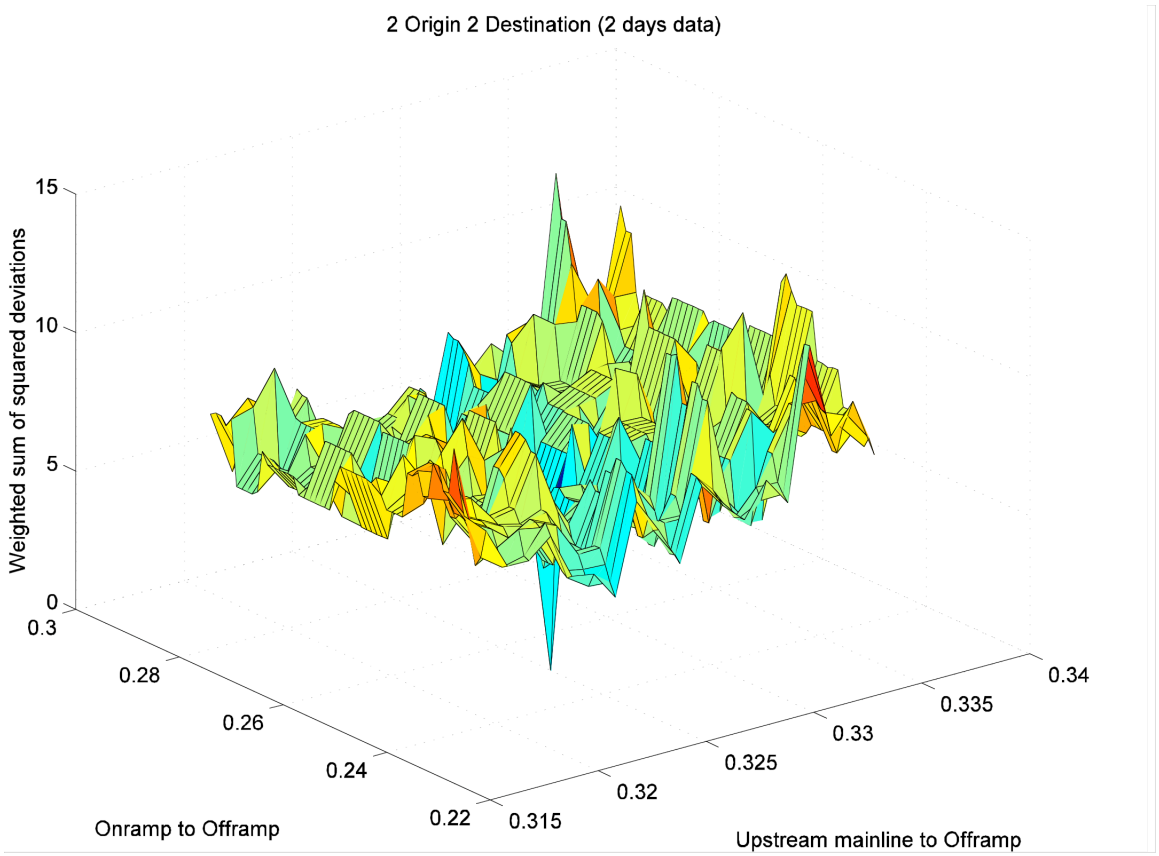

Figure 6.4 – Objective Function 2-Days

MINOS is a gradient-based method and only promises local optima. In addition, since MINOS has to numerically estimate the gradients, the search method is not as efficient as compared to the case where the gradients can be defined explicitly. It is observed that the plots are very uneven and spiky. The optimal point is the lowest point on all the plots that corresponds to the actual OD matrix. The valley leading to that point is very steep. Also, as the additional days of data are added the objective function looks less noisy.

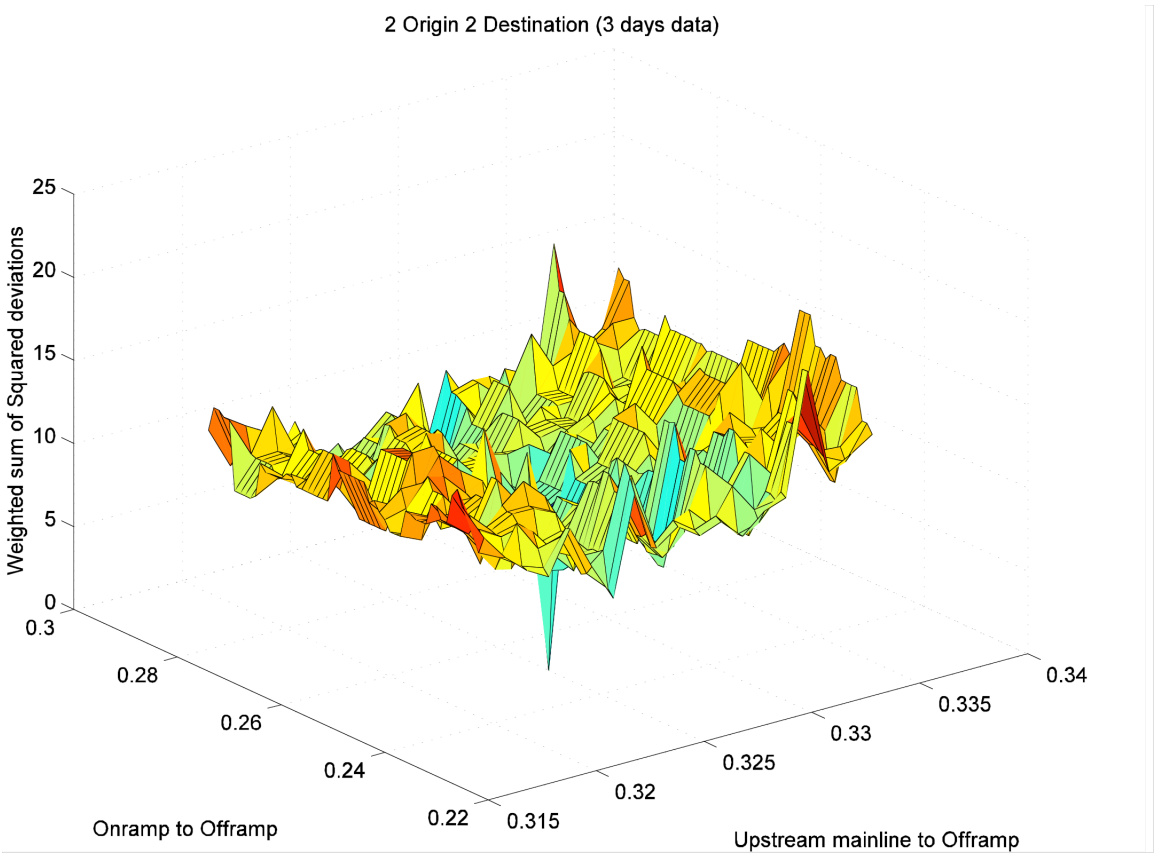

Figure 6.5 – Objective Function 3-Days

The objective function is the map between the OD matrix and the off-ramp counts. The OD matrix is the input and the output is the off-ramp counts, which feature in the objective function. The map as described above is a function of AIMSUN. Hence the shape of the objective function is related to the way AIMSUN takes in an OD matrix and generates the off-ramp counts. The general spiky nature of the objective function indicates a discontinuous map. Also there seems to be a many-to-one map, i.e., there is more than one OD matrix that has the same objective function value. This explains why we end up with different OD estimates but similar levels of performance with matching the counts.

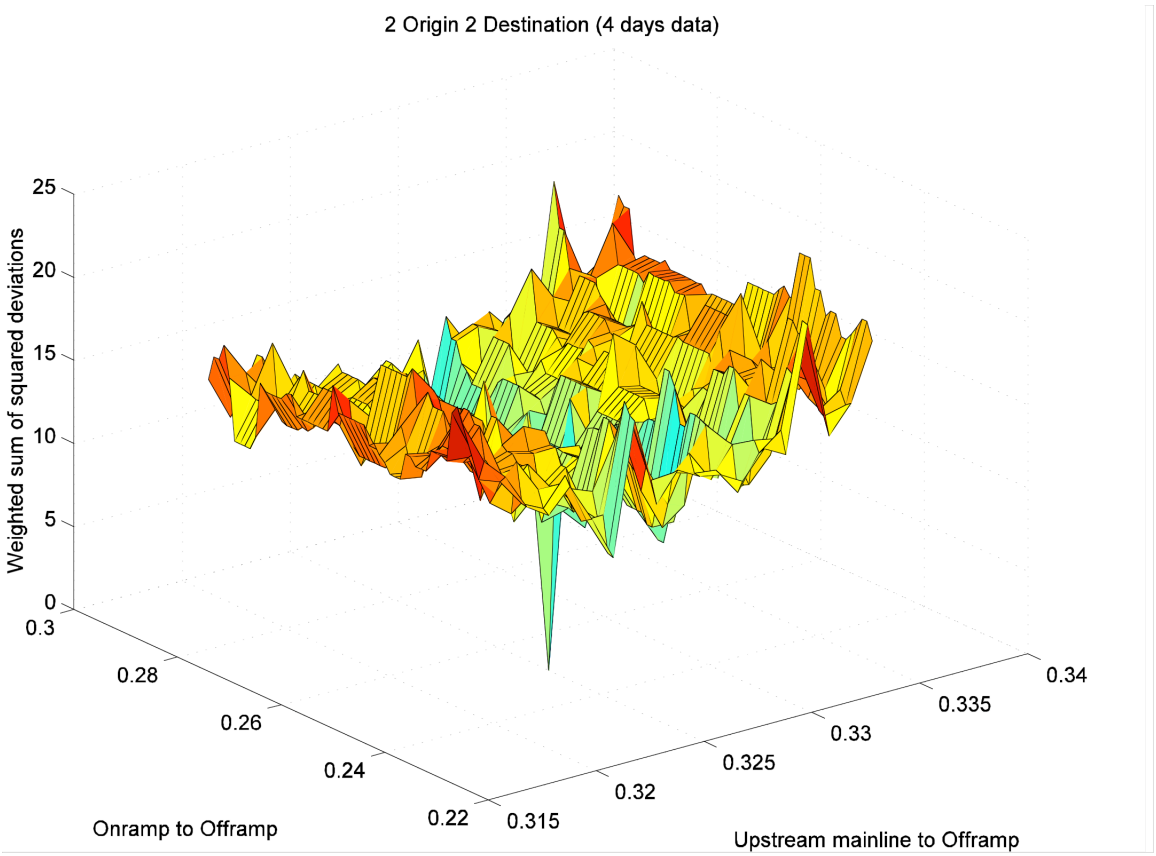

Figure 6.6 – Objective Function 4-days

The gradual smoothing or reduction in the noise of the objective function would make the searching process of MINOS efficient. However, the smoothing did not have a significant influence on the results. Therefore, it was concluded that the information given by the additional days was not sufficient.

If two days of data were identical, then the additional information will be zero and if they are as different as they can get, the information added will be maximum. Hence a reasonable mix of the two will be ideal, as more information is needed for the OD estimation. Also, the dominant effect of the mainline in the OD estimation is observed. So, it is hypothesized that if there was some way to reduce its effect on the estimation, it might be useful. Hence another data set was constructed using these ideas.

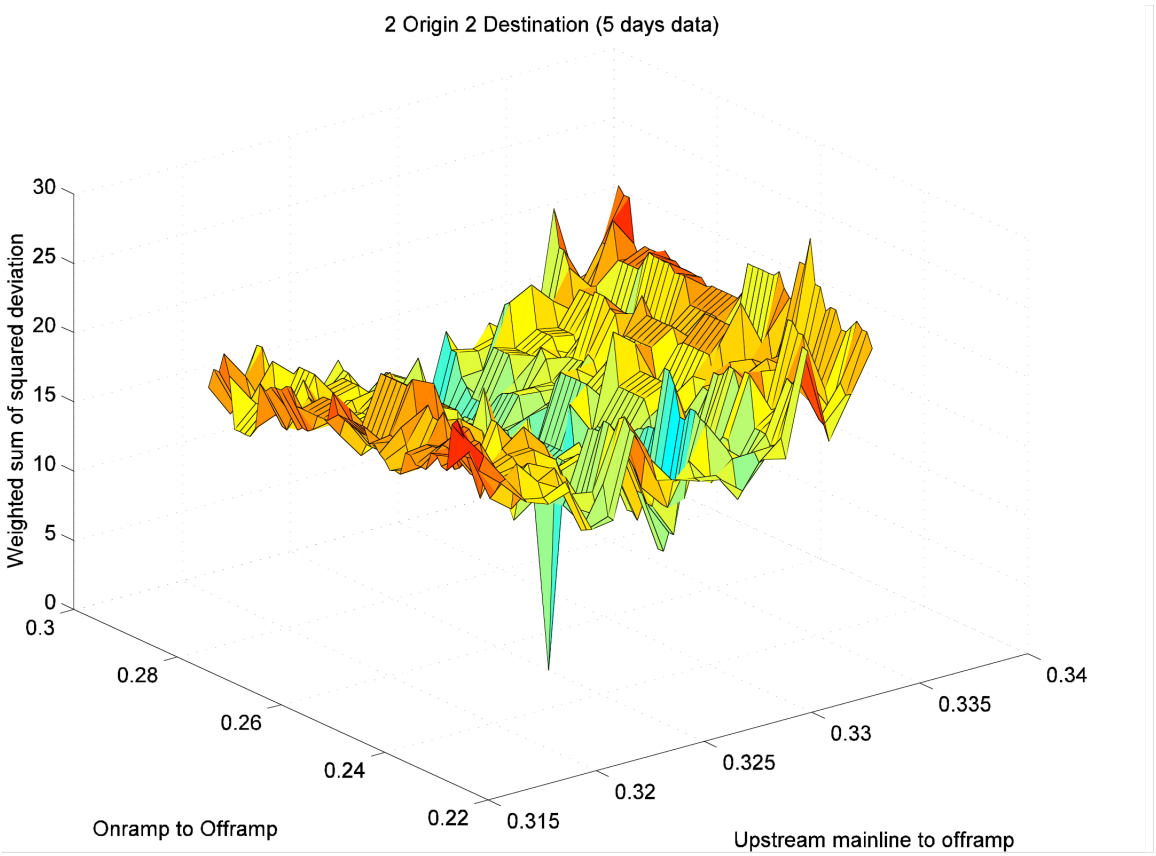

Figure 6.7 – Objective Function 5-Days

6.1.5 Radical Data Set

The dominance of the mainline counts in the OD estimation was observed. Hence it was decided to alter the effect of the mainline in the data set. If it is imagined that the on-ramps were operated one at a time by shutting off the other ramps and the mainline, the actual OD matrix can be estimated accurately. But, this is a practical impossibility. An approximation to this would be to assume on-ramp volumes equal in magnitude to the mainline counts, which is termed as a “bizarre” day. Using these on-ramp counts from such a “bizarre” day and a regular day, an OD matrix could be estimated. The data set used is in Appendix E. The results from this experiment are shown in Table 6.4

Actual OD

	d1	d2
o1	0.3250	0.6750
o2	0.2500	0.7500

Seed 1

	d1	d2
o1	0.5000	0.5000
o2	0.5000	0.5000

Solution

	d1	d2
o1	0.3332	0.6669
o2	0.1834	0.8166

Seed 2

	d1	d2
o1	0.3288	0.6712
o2	0.3288	0.6712

Solution

	d1	d2
o1	0.3135	0.6865
o2	0.3241	0.6759

Seed 3

	d1	d2
o1	0.2827	0.7173
o2	0.4548	0.5452

Solution

	d1	d2
o1	0.3265	0.6735
o2	0.2488	0.7512

Weighted Sum of Squares

seed	Start	End
1	849.37	4.90
2	7.61	4.45
3	17.44	3.45

R-squared

seed	D1	D2
1	0.9851	0.9959
2	0.9890	0.9943
3	0.9849	0.9929

Table 6.4 – OD Estimates, Radical data set

Again, as in the above cases, the different OD matrices replicate counts with the same levels of accuracy. However, seed-3 has estimated the OD matrix accurately, an indication that the data set was such that the OD matrix could be estimated accurately. However, there is a chance that this estimate was purely accidental. Since not all the seeds converged to the solution, there is not enough evidence to believe that this assures the correct solution.

In conclusion, the experiments with this site have helped in checking the setup for the OD estimation process, given good insight into the nature of the objective function, brought to light some inherent features in this OD estimation method using MINOS and AIMSUN, and provided some indications to the existence of multiple solutions and an identifiability problem.

6.2. Case 2 – TH-169

The real site chosen for the OD estimation was TH-169. It was chosen because the network was previously built and calibrated in AIMSUN. The chosen section is Northbound, starting at the intersection with TH-55 and up to the intersection with 63rd Avenue, just south of the intersection with I-94. It is about 6.5 miles long and has 11 on-ramps and 10 off-ramps. So the OD matrix has 12 origins and 11 destinations. There are 76 non-zero entries in the OD matrix. The section parameters of the freeway are given in Appendix F. A schematic of the location of the test site is shown in Figure 6.8.

6.2.1 Data

The real data used for the estimation process was collected during the ramp-meter shutdown period, so that the simulation was done without ramp metering. The data was the five minute detector counts from November 1, 2 and 3, 2000 for the morning peak

between 7 a.m. and 10 a.m. The drawback in this case is that the actual OD matrix is not known, so there is no reference to compare the estimates and pick the best. The only guidelines that are available are the ramp counts, their r-squares, and percentage deviations.

Figure 6.8 – Test Site TH-169

However, if a simulated data set is created as shown in Figure 6.2 and then an OD matrix estimated, the properties of the estimates can be evaluated better. Also, the system-wide MOE's like Total Travel and Total Travel Time can also be measured and will indicate the ability of the OD matrix to reproduce the system characteristics. But, using a simulated data set discounts AIMSUN and removes the specification error so the results with the simulated data set are expected to be better than the real data set.

6.2.2 Simulated Data Set Results

The simulated data set is given in Appendix G. The results from the OD estimation are shown in Table 6.5. The actual OD estimates are also given in Appendix G. Table 6.5 has the average absolute percentage deviation for the off-ramp counts using the OD estimates

and the actual off-ramp counts. Also the r-squares for the ramps are given. The 'F-value' is the final objective function value and the Sq.dev is the sum of the square of the errors. The system-wide MOE's are shown in Table 6.6 and 6.7. Solutions 2, 3 and 5 correspond to seeds 2, 3, and 5 respectively.

Solution 5

Fvalue	7.46
Sq.dev	2209

Ramp	% deviation	R-squared
D1	0.56	0.9927
D2	1.91	0.9742
D3	0.65	0.9794
D4	0.00	1.0000
D5	1.12	0.9883
D6	0.41	0.9941
D7	0.93	0.9932
D8	2.17	0.9721
D9	2.17	0.9328
D10	1.12	0.9768
D11	0.00	0.9921

Solution 2

Fvalue	131.84
Sq.dev	21740

Ramp	% deviation	R-squared
D1	3.20	0.9345
D2	4.08	0.9477
D3	17.61	0.8815
D4	8.54	0.8058
D5	4.61	0.9202
D6	5.76	0.9297
D7	8.05	0.9298
D8	8.70	0.8152
D9	13.53	0.9207
D10	18.81	0.8970
D11	6.01	0.9802

Solution 3

Fvalue	9.09
Sq.dev	2377

Ramp	% deviation	R-squared
D1	1.94	0.9827
D2	1.92	0.9830
D3	0.96	0.9688
D4	1.07	0.9852
D5	2.07	0.9819
D6	1.18	0.9848
D7	1.43	0.9905
D8	1.45	0.9785
D9	1.78	0.9706
D10	2.46	0.9637
D11	1.25	0.9919

Table 6.5 – Results, TH-169 Simulated Data Set

The estimates accurately reproduce the off-ramp counts, and all the final OD estimates have a low magnitude of error from the actual OD matrix and can be thought of as closed solutions. Also, these estimates reproduce the system wide MOE's very well.

System wide MOE's	unit	Actual
Mean Flow	veh/hr	6942
Mean Speed	km/hr	76.2
Mean Delay	sec/veh	9
Mean # Stops	per veh	0.1
Total Travel	km	112792.7
Total Travel Time	hours	1480.2
Total Delay	hours	63
Total # of Stops		3638

Table 6.6 – System wide MOE's for the simulated data set

System wide MOE's	Solution 5	% deviation	Solution 2	% deviation	Solution 3	% deviation
Mean Flow	6938	-0.06	6949	0.10	6939	-0.04
Mean Speed	76	-0.26	74.3	-2.49	76.4	0.26
Mean Delay	10	11.11	14	55.56	9	0.00
Mean # Stops	0.1	0.00	0.1	0.00	0.1	0.00
Total Travel	112778.7	-0.01	115292.6	2.22	112332.6	-0.41
Total Travel Time	1483.9	0.25	1551.7	4.83	1470.3	-0.67
Total Delay	65.8	4.44	110.7	75.71	55.5	-11.90
Total # of Stops	4706	29.36	8573	135.65	4682	28.70

Table 6.7 – System Wide MOE's for the Solutions

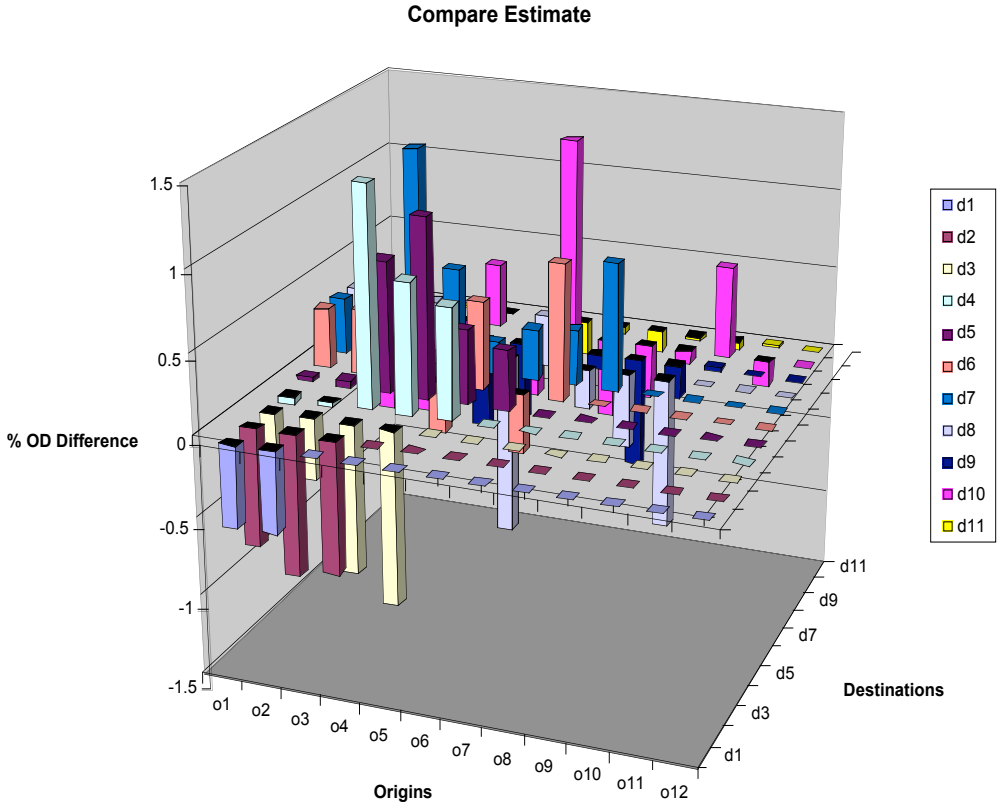

Figure 6.9 – OD Estimate Comparison (Solution from Seed 5 vs. Assumed)

To illustrate the accuracy of the OD matrix estimated, the percentage deviation in the OD matrix entries between the one estimate (solution 5) and the assumed OD matrix is shown in Figure 6.9. Figure 6.9 has one column for each OD matrix cell entry, representing the percentage error between the estimated and actual OD matrix.

The important point to note about this figure is that the observed range of error is between ± 2%, a fairly high range of accuracy. Hence this experiment has generated expected outcomes—good reproduction of the counts and the system properties. All the solutions seem to be close to each other and the assumed OD matrix. This indicates that the method as set up has performed well. This is because the data was generated using

AIMSUN and so has discounted the errors that will be normally introduced by using AIMSUN to approximate the data-generating method. As a conclusion, the estimates seem to converge to the actual OD and are very good in terms of reproducing the counts and the system-wide statistics.

6.2.3 Real Data Results

The real data set used for the OD estimation process is given in Appendix H. The results from the run using one-day data are given in Table 6.8. The warm-up run was done to account for the discrepancy that occurs when the simulation starts and the system is empty, but in reality there are cars in the system. Hence by skipping the first three time slices, a time period over which the system has near-reality operating conditions, the OD estimation process is conducted for the remaining 33 time slices rather than the 36 time slices as in the first case.

The results are not as good as the simulated data set. The low r-squares are due to the inherent nature of the counts. But the percentage deviation and the sum of the squared errors are good measures of the performance of the estimates. Again, as in the previous cases, the mainline is matched best. The final objective function values are an order of magnitude higher than the simulated data set results. The only similarity in the two experiments is that the OD estimate corresponding to seed 5 (turning percentage based) was the best estimate.

Real data 1 day 1-Nov-00 7 am-10 am		
Solution 5		
Fvalue	260.63	
Sq.dev	21960	
Ramp	% deviatio	R-squared
D1	14.58	0.8150
D2	22.51	0.4790
D3	15.81	0.1777
D4	27.06	0.2319
D5	24.43	0.0330
D6	32.23	0.0078
D7	18.91	0.6044
D8	20.49	0.1666
D9	29.44	0.4666
D10	28.28	0.5371
D11	6.03	0.8710
	Overall	0.9800

Real data 1 day with warm up time 1-Nov-00 7:15 am-10 am		
Solution 5		
Fvalue	243.36	
Sq.dev	22626	
Ramp	% deviatio	R-squared
D1	15.41	0.8040
D2	26.66	0.5220
D3	15.65	0.2325
D4	24.98	0.2479
D5	19.06	0.1058
D6	30.39	0.0081
D7	24.00	0.6077
D8	21.49	0.1777
D9	36.86	0.5235
D10	32.86	0.5086
D11	6.65	0.8927
	Overall	0.9800

Table 6.8 – Results, TH-169 Real Data

The inability to better the estimates was attributed to lack of information in the system. At an abstract level, if there is not enough information in the system, the parameters cannot be identified. So, some modifications were proposed to the problem along the lines of adding information into the system. The experiments with the first test site in terms of improving the estimates by adding additional data into the system was attempted to this real data set. Unlike in the simulated case, wherein the data set could be modified, the real data used for the estimation cannot be changed.

As a first modification, the additional information into the system was the multiple days' data. Two other modifications were proposed. Based on the observation that the mainline proportions were being estimated accurately, the OD estimation process was decomposed as a two-phase optimization. The first phase was the OD estimation of all the OD cell entries and, after the optimal solution was obtained as a second phase, the mainline elements were treated as constants and the rest of the OD elements were re-estimated. The last modification was to add the OD matrix into the objective function. In other words, in addition to a term with the weighted sum of squared errors of the off-ramps, a term that measured the squared error of the current OD matrix with the starting OD matrix was added. This can be interpreted as forcing MINOS to search in a feasible space not very far from the starting solution, under the pretext that there is a good starting solution. Mathematically, the objective function is represented as follows.

$$F_{multidays} = \sum_{d}\sum_{j}\sum_{t} w_{d,j}(O_{d,t,j} - \hat{O}_{d,t,j})^2$$

$$F_{new} = \sum_{d}\sum_{j}\sum_{t} w_{d,j}(O_{d,t,j} - \hat{O}_{d,t,j})^2 + N_{days} * N_{time} * N_{des} \sum_{i}\sum_{j} (b_{i,j} - b0_{i,j})^2$$

Where,

$F_{multidays}$ –the objective function with multiple days.
F_{new} –the new objective function with OD matrix included in it.
i, j, t, d – indices for origins, destinations, time slices, and days respectively.
$w_{d,j}$ – weight for destination j on day d.
$O_{d,t,j}$, $\hat{O}_{d,t,j}$ - observed and predicted counts at off-ramp j in time slice t on day d.
$b_{i,j}$, $b0_{i,j}$ – current and starting OD matrix entries from on-ramp i to off-ramp j.
N_{days}, N_{time}, *and* N_{des} – number of days, time slices, and destinations respectively.

The objective function is the same for the two-step optimization, just that in the second phase, the first row of the OD matrix $b_{1,j}$ are treated as constants. The second term in F_{new} has only one term for all the time slices and days, but the first term has multiple terms corresponding to each time slice in each day for each destination. So, to scale the terms equally, the second term is scaled up by the product of N_{days}, N_{time},and N_{des}. The related results are in Table 6.9. The estimated OD matrices are in Appendix H.

There are some marginal improvements in terms of the reduction of the objective function value. However, the ability to match the ramps has not improved significantly.

This inability to get good estimates can be attributed to a combination of the mismatch between reality and AIMSUN, bad data, and the identifiability issue in the estimation process. Evidence to support all the above has been seen in the previous experiments.

3-days Nov 1,2,3 7 am-10 am Solution 5			New Objective Function 3-days Nov 1,2,3 7 am-10 am Solution 5			2-step Optimization 3-days Nov 1,2,3 7 am-10 am Solution 5		
Fvalue	825.42		Fvalue	811.56		Fvalue	806.64	
Sq.dev	75247		Sq.dev	77924		Sq.dev	74742	
Ramp	% deviatio	R-squared	Ramp	% deviatio	R-squared	Ramp	% deviatio	R-squared
D1	17.25	0.5781	D1	19.21	0.5695	D1	17.45	0.5737
D2	20.35	0.4912	D2	22.56	0.4934	D2	20.68	0.4820
D3	20.28	0.2497	D3	21.27	0.2502	D3	19.12	0.2448
D4	23.32	0.1151	D4	24.98	0.1022	D4	24.00	0.1155
D5	32.29	0.1330	D5	34.38	0.1211	D5	32.42	0.1089
D6	32.09	0.0246	D6	30.91	0.0075	D6	28.39	0.0317
D7	17.85	0.6519	D7	19.07	0.6425	D7	17.95	0.6503
D8	21.60	0.1652	D8	22.15	0.1556	D8	21.83	0.1666
D9	27.93	0.5001	D9	28.06	0.5270	D9	28.12	0.4897
D10	40.84	0.4943	D10	37.43	0.5078	D10	36.76	0.5144
D11	7.20	0.8082	D11	7.24	0.8133	D11	7.19	0.8073
	Overall	0.9773		Overall	0.9777		Overall	0.9806

Table 6.9 – Results, TH-169 Real Data with Modifications

Chapter 7 –Hypothetical Grid Network

Networks have a better information base in the sense that in addition to the numerous link counts there is a possibility of collecting the turning movement counts also. This additional information is expected to improve the OD estimation process. Also as mentioned earlier, the OD estimate is more consequential to the network than freeway sections, as simulation without an OD matrix is impossible in a network but possible for freeways. To start with, we build a hypothetical grid type network as shown in Figure 7.1

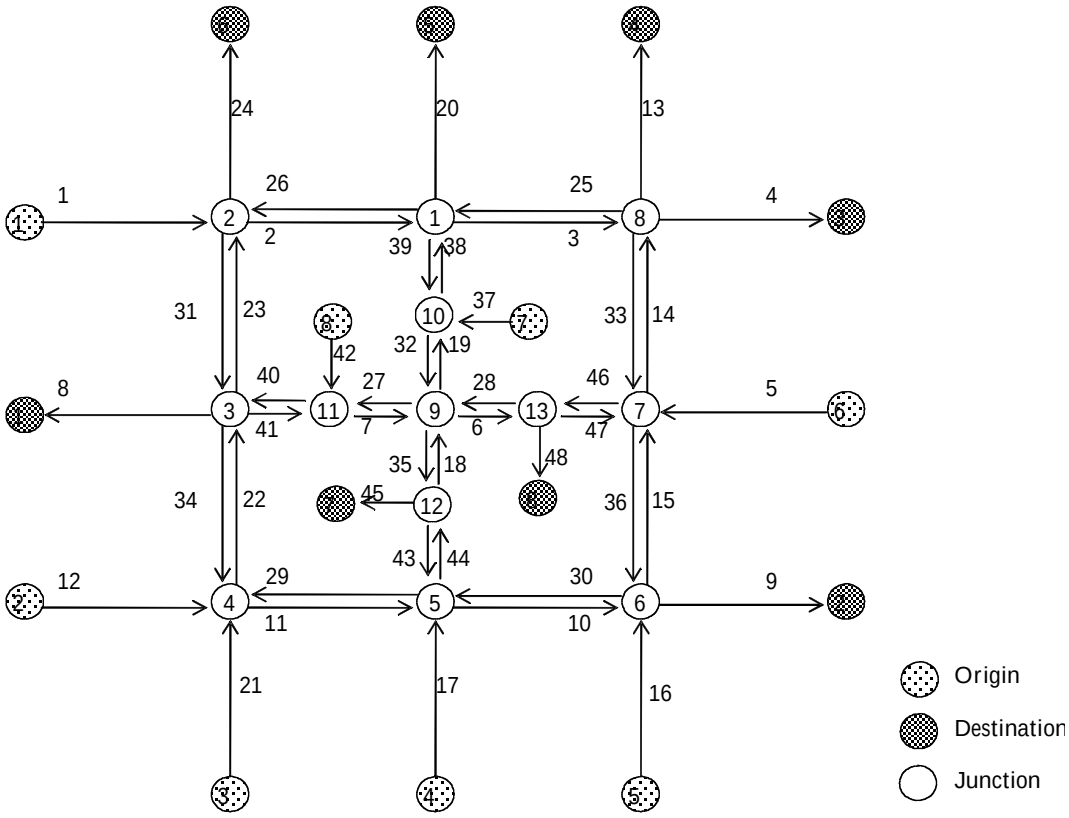

Figure 7.1 – Hypothetical Grid Network

The network has eight origins and eight destinations (in all 64 possible interchanges). There are four lights and one stop sign. Also, there are 16 one-way and 32 two-way links and 13 junctions in the network. As before, input counts and an OD matrix are assumed and simulating within AIMSUN generates the data set. In this case, the observed data comprises of the turning volumes at each section and the link counts every 15 minutes over a three-hour simulation period. Now the OD estimation process is done trying to match the turning volumes rather than the link counts. The data set is given in Appendix I. The important difference between the freeway and the network simulation is the route choice model. AIMSUN has the capability to model dynamic and static route choice models. Also, from the geometry, as the inputs vary, there is a shifting of trips

between equally likely alternate routes. The following scenarios are used for the OD estimation.

The scenarios get progressively complex and the final is the closest to reality.
1. One time slice and fixed route choice.
2. One time slice and dynamic route choice.
3. Multiple time slices, fixed route choice, same trip table per time slice.
4. Multiple time slices, dynamic route choice, same trip table per time slice.
5. Multiple time slices, fixed route choice, different trip table per time slice.
6. Multiple time slices, dynamic route choice, different trip table per time slice.

New Grid 15min slice match turning volumes

System wide MOE's	unit	Actual	Solution 1	% deviation	Solution 2	% deviation
Mean Flow	veh/hr	2388	2388	0.00	2412	-1.01
Mean Speed	km/hr	24.5	25	-2.04	24.8	-1.22
Mean Delay	sec/veh	112	111	0.89	105	6.25
Mean # Stops	per veh	5.9	5.8	1.69	5.9	0.00
Total Travel	km	162.3	158.8	2.16	155.1	4.44
Total Travel Time	hours	6.6	6.4	3.03	6.3	4.55
Total Delay	hours	18.6	18.4	1.08	17.6	5.38
Total # of Stops		3522	3463	1.68	3558	-1.02

Table 7.1 – Results Hypothetical Grid 1 Time Slice

The results pertinent to step one are given in Table 7.1. As of now, only Step one has been completed and the rest are being worked on (one time slice = 15 min).

The best OD estimate (Solution 2, Seed 1) is shown as a comparison to the assumed OD matrix in terms of the percentage deviation from the cell entry in Figure 7.2. The plot in Figure 7.2 is between the simulated and actual link counts. All the points seem to be around the y=x line. Also, the OD matrix estimated is good in terms of matching the turning volumes, the assumed OD matrix, and the system wide MOE's. Hence, the experiment has given us satisfactory results with the one time slice case. The OD estimation process on a network remains to be fully developed but the initial results are encouraging and promising.

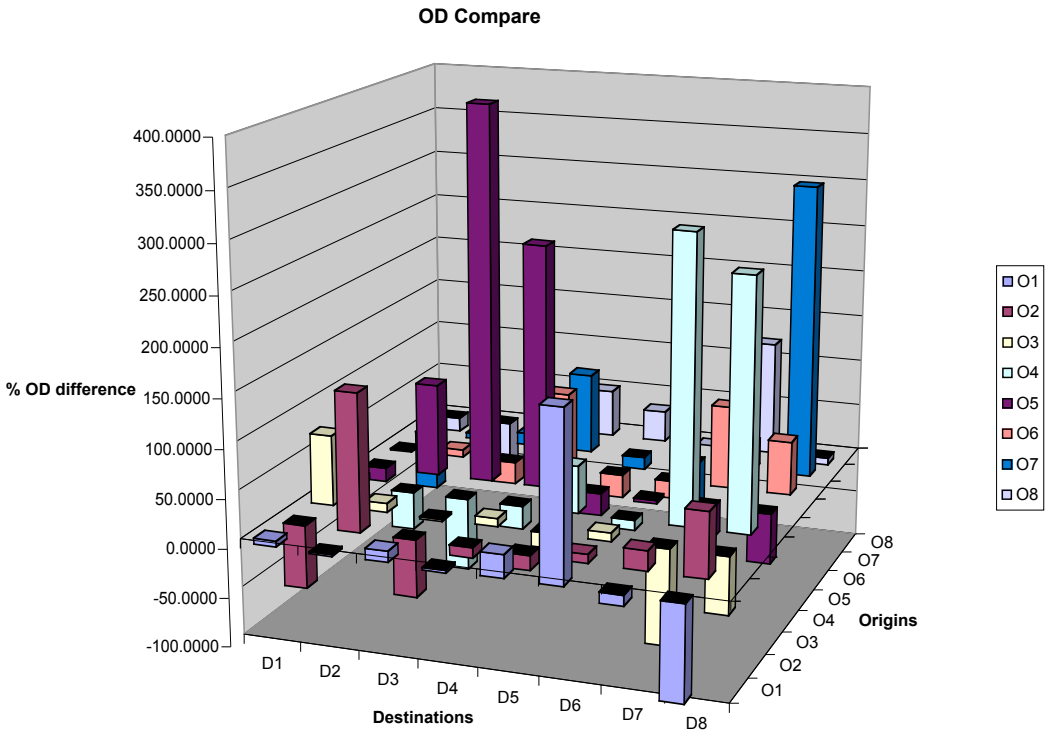

Figure 7.2 – OD Estimate Comparison

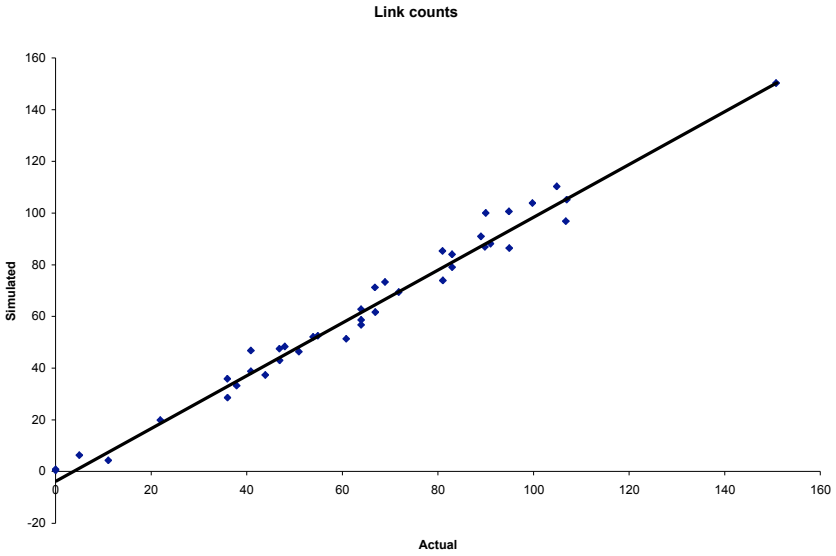

Figure 7.3 – Link Counts Comparison

Chapter 8 – Conclusions

An *offline method* to estimate a *static OD proportions* matrix for a *freeway section* over the *peak period* has been proposed. Most OD estimation methods have some form of a linear model to represent the relation between the on-ramp and the off-ramp counts and Davis and Yu (1994) showed that the approach having a traffic flow model embedded in the estimation process outperformed the linear models. This method enhances that approach. The appeal in the method is its simplicity. The problem has been defined as an optimization process with an embedded simulator that tries to find an optimal OD matrix that minimizes the weighted sum of the squared deviations of the off-ramp counts. The method is a combination of simulation and optimization. The simulation component is the microscopic simulator AIMSUN and the optimization routine is MINOS.

The method does not need a prior estimate of the OD matrix unlike most offline methods. As a part of the estimation process, using the time series of counts, estimates of the OD matrix are made using five different methods, and starting solutions (seeds) are generated. These are used to start the search in the optimization process. This method can also be interpreted as an efficient updating scheme of the starting OD estimate. Hence, this can also be used to improve any prior estimate.

Experiments were conducted on two test sites. The first test site was an imaginary test section of a freeway. The data set was simulated and the general observations on the results are as follows. The starting solutions converged on different OD matrices that had comparable performance with respect to reproducing the counts, an indication of a many-to-one map between OD matrices and the objective function. The mainline proportions were matched best. The plots of the objective function over the space near the optimal point gave very useful insight into the non-linearity and spiky nature of the map between the OD matrix and off-ramp counts as generated by AIMSUN. The experiments with the "bizarre" day data set indicated the possibility of an identifiability issue.

The second test site was TH-169. This had 76 non-zero entries that needed to be estimated. This is a sizeable increase in dimension over the network handled in Davis and Yu (1994). Two sets of experiments were conducted on this site using a real and a simulated data set. The simulated data set resulted in very good estimates that matched the counts and the system statistics very well, and the final solutions were close to each other and to the true OD matrix. The real data set, on the other hand, did not produce equivalent results. New modifications to the method were proposed and the results did not improve significantly.

The experiments alluded to the following issues. The map between the OD matrix and the off-ramp counts is very spiky and the surface is very uneven. Also the plot indicated the possibility of a many-to-one map between the OD matrix and the objective function. The experiment on the "bizarre" day indicated an identifiabilty issue. The inability to improve the estimates using real data to match the ramp counts as well as the simulated data set could be a mix of all the above causes and also data discrepancy. The most important observation is the performance in the simulated data set. As discussed in Chapter 3, the process relating the on-ramp and off-ramp counts can be approximated as a data-generation scheme. If a microscopic traffic simulator can approximate that process reasonably well, the OD matrix can be estimated accurately, which is evident from the

simulated data set wherein the microscopic traffic simulator is the process that generated the data set.

In conclusion, this offline method has great appeal due to its simplicity. Also the method does not need an *a priori* start solution: multiple starting solutions are generated using the data set. The performance improves as the ability of AIMSUN to match the actual model that generated the data set is higher. Also, the data collected from the detectors must be error free. Interesting findings relating the mainline dominance and identifiability have been found from the experiments on the first test site. The modifications to the traditional objective function have been proposed and the two-step optimization seems promising. Finally, the use of a micro-simulation makes the evaluation of the estimates better because counts as well as other system statistics can be compared.

Future work can be done on enhancing the performance of the microscopic traffic simulator with better calibration. The new modifications to the traditional objective function, the two-step optimization, and the new objective function need to be investigated. Finally, the identifiability issue of insufficient information in the off-ramp counts can be investigated by experimenting with alternate sites that have additional information, like a small network with turning movement volumes.

References

1. May, D. A. and Willis E. A. (1981) Deriving Origin-Destination information from routinely collected traffic counts – Vol. I, research report UCB-ITS-RR-81-8.
2. Robillard P. (1974) Estimating the O-D matrix from observed link volumes, *Transportation Research* Vol. 9, pp. 123-128.
3. Van Zuylen, J. H. and Willumsen, L, G. (1979) The most likely trip matrix estimated from traffic counts, *Transportation Research Part B*, Vol. 14B, pp. 281-293.
4. Speiss, H. (1987) A maximum likelihood model for estimating Origin-Destination matrices. Transportation Research Part B, Vol. 21B, No.5, 1987, pp. 395-412.
5. Turnquist, M. and Gur, Y. (1979) Estimation of trip tables from observed link volumes, *Transportation Research Record 730*, 1979.
6. Martin, P. and Bell, M. (1992) Network programming to derive the turning movements from link flows, *TRB Annual Meeting presentation 1992*.
7. Nihan, L. N. and Davis, G. (1987) Recursive estimation of Origin-Destination matrices from input/output counts. Transportation Research Part B, Vol. 21B, No.2, 1987, pp. 149-163.
8. Nihan, L. N. and Davis, G. (1991) Stochastic process approach to the estimation of Origin-Destination parameters from time-series of traffic counts, *Transportation Research Record 1328,* pp. 36-42.
9. Cremer, M. and Keller, H. (1987) A new class of dynamic methods for identification of Origin-Destination flows. Transportation Research Part B, Vol. 21B, No.2, 1987, pp. 117-132.
10. Ashok, K. (1996) Estimation and prediction of time-dependent Origin-Destination flows. Ph.D. Thesis. M.I.T
11. Davis, G. (1993) A statistical theory for estimation of origin-destination parameters from time-series of traffic counts. Transportation and Traffic Theory C.F Daganzo (Editor), 1993.
12. Yu, P. and Davis, G. (1994) Estimating the Freeway Origin-destination patterns using automation traffic counts, *Transportation Research Record 1457*, 1994.
13. Kang, J. and Davis, G. (1994) Estimating the destination-specific densities on urban freeways for advanced traffic management, *Transportation Research Record 1457*, 1994, pp. 143-148.
14. Davis, G. (1993) Estimating the freeway demand patterns and impact of uncertainty on-ramp controls. Journal of Transportation Engineering, Vol. 119 No. 4, July/August 1993, pp. 489-503.
15. Davis, G. (1994) Dynamic estimation of origin-destination patterns in Freeways, Final report MN/RC – 94/18
16. Kang, J. (1995) Estimation of destination-specific traffic densities and identification of parameters on urban freeways using Markov models of traffic flow, Ph.D. Thesis, University of Minnesota.
17. Abrahamsson, T. (1998) Estimation of Origin-Destination matrices using traffic counts – A literature survey. Interim Report, IRA-98-021.
18. Hazelton, L. M. (2000) Estimation of Origin-Destination matrices from link flows on uncongested networks. Transportation Research Part B, Vol. 34 (2000) pp. 549-566.

19. Cascetta, E. and Postorino, N. M. (2001) Fixed-point approaches to estimation of O/D matrices using traffic counts on congested networks. Transportation Science, Vol. 35, No.2, May 2001, pp. 134-147.
20. Transport Simulation Systems (2001) *Aimsun version 4.0 – User Manual,* April 2001.
21. Murtagh, B. A. and Saunders, M. A. (1978) Large-scale linearly constrained optimization. *Mathematical Programming,* Vol. 14, 1978, pp. 41-72.
22. Murtagh, B. and Saunders, M. (1983) *MINOS 5.5 User's Guide*.
23. Beck, P. Lasdon, L. and Engquist, M. (1983). *A Reduced gradient algorithm for nonlinear network problems,* ACM Transactions on Mathematical software, Vol. 9, No. 1, March 1983, pp. 57-70.
24. Nemhauser, G. L. Rinnoy Kan, A. H. G. and Todd, M. J. (Editors). *Handbook in Operations Research and Management Science, Volume 1, Optimization.* Elsevier Science Publishers B.V., 1989.
25. Press, W. H. Teukolsky, S. A. Vetterling, W. T. Flannery, B. P. *Numerical Recipes in FORTRAN*. Second Edition. Cambridge University Press, 1994.
26. H.J. Greenberg. *Mathematical Programming Glossary*. World Wide Web, http://www.cudenver.edu/~hgreenbe/glossary/glossary.html, 1996-2000
27. Deming, W. E. and Stephan, F. F. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. Annal of Mathematical Statistics 11, 1940. pp 427-444.

APPENDICES

Appendix A – Example MPS file

Consider the following network to estimate and OD matrix.

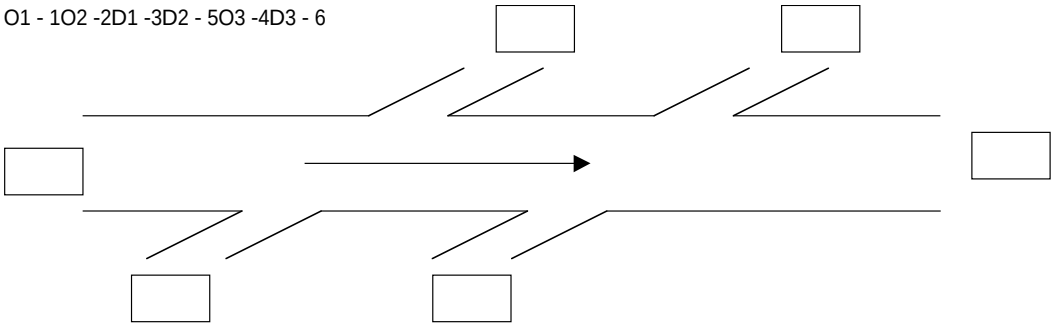

Figure A1 – Example freeway

The following section describes the relevant parameters and the format for the MPS and SPECS file. It has 3 origins and 3 destinations. A Typical OD matrix would look like Table A1.

O/D	D1	D2	D3
O1	0.063	0.146	0.791
O2	0.063	0.146	0.791
O3	0.0	0.156	0.844

Table A1 – Sample OD matrix for example freeway

The MPS File

There are only eight feasible interchanges (O3 _ D1 not possible) represented by the variables named as X001 to X008 in the order enumerating from Origin1 proceeding to all destinations and then the Origin 2 and so forth. The OD matrix is stacked into one single array. There are three constraints, corresponding to each of the three origins, stating that the sum of the proportions of the trips have to add up to 1.0 named as ORI001, ORI002, and ORI003. They are equalities denoted by the key word E and the RHS and the Coefficients are 1.0. This is represented in the RHS and the COLUMNS sections. Logically these values are bounded by 0.0 and 1.0. Hence, the lower and upper bounds for all the eight variables LO and UP are 0.0 and 1.0. In addition, the initial values are defined for the variables in the INITIAL bounds section. The file is as shown below.

NAME

```
ROWS
E  ORI001
E  ORI002
E  ORI003
COLUMNS
   X001      ORI001     1.0000000
   X002      ORI001     1.0000000
   X003      ORI001     1.0000000
   X004      ORI002     1.0000000
   X005      ORI002     1.0000000
   X006      ORI002     1.0000000
   X007      ORI003     1.0000000
   X008      ORI003     1.0000000
RHS
   DEMANDS   ORI001     1.0000000
   DEMANDS   ORI002     1.0000000
   DEMANDS   ORI003     1.0000000
BOUNDS
 LO BOUND1   X001       0.0000000
 UP BOUND1   X001       1.0000000
 LO BOUND1   X002       0.0000000
 UP BOUND1   X002       1.0000000
 LO BOUND1   X003       0.0000000
 UP BOUND1   X003       1.0000000
 LO BOUND1   X004       0.0000000
 UP BOUND1   X004       1.0000000
 LO BOUND1   X005       0.0000000
 UP BOUND1   X005       1.0000000
 LO BOUND1   X006       0.0000000
 UP BOUND1   X006       1.0000000
 LO BOUND1   X007       0.0000000
 UP BOUND1   X007       1.0000000
 LO BOUND1   X008       0.0000000
 UP BOUND1   X008       1.0000000
 FX INITIAL  X001       0.063
 FX INITIAL  X002       0.146
 FX INITIAL  X003       0.791
 FX INITIAL  X004       0.063
 FX INITIAL  X005       0.146
 FX INITIAL  X006       0.791
 FX INITIAL  X007       0.156
 FX INITIAL  X008       0.844
ENDATA
```

Appendix B – Example SPECS File

The Specs File

The choice of Function Precision affects the step size in the calculation of the numerical gradients in the forward differencing and central differencing stages. They are (Function Precision)$^{1/2}$ and (Function Precision)$^{1/3}$. In addition, since the OD proportions are in the order of magnitude around 0.5, the changes are expected in the second decimal place, hence the ideal choice for Function Precision is around 10^{-4}. The choice of the optimality tolerance is based on a trial and error type, however, as the tolerance is increased from the default value of 1.0e-06, the line search is being relaxed. The iteration limit is set based on some trial runs. It determines the termination condition if the optimal is not yet found.

```
Begin OD-estimation
Minimize
Objective = Funobj
   Nonlinear variables        8
   Super basics limit         10
   Derivative Level           2
   Function Precision         1.0E-03
   Optimality Tolerance       1.0e-02
   MPS file                   10
  Iterations limit            30
End OD-estimation
```

Appendix C – Steps of the Reduced Gradient Algorithm as implemented in MINOS

The following section is the steps of the algorithm as seen in *(23)*.

Let $\bar{x} = (\bar{x}_B, \bar{x}_S, \bar{x}_N)$ be the current (feasible) value of x

1. Step 1

1.1 Evaluate $\partial F / \partial x_B$ at the current point

1.2 Calculate π and g_A

2. If certain optimality tests on the reduced problem using the current set of tolerances are met, go to step 7. (Test for convergence in the current subspace.)

3. Compute $\lambda = g_N - N^T \pi$

4. If λ_i " 0 when $\{x_N\}_i$ is at a lower bound and λ_i ! 0 when $\{x_N\}_i$ is at an upper bound continue, go to step 6.

5. If the optimality tolerances on the reduced problem are 'tight' stop (Kuhn-Tucker conditions are satisfied).
Replace the loose tolerances with the tight ones and go to step 7.

6. Add one or more *non-basic* variables to the super basic set.

7. Calculate the search direction vector d_S and the maximum step length α_S for the *superbasic* variables. The scalar α_S is the largest value such that $\bar{x}_S + \alpha d_S$ satisfies the bounds on x_S.

8. Calculate the search direction vector d_B and the maximum step length α_B for the *basic* variables. The vector d_B is computed solving
$Bd_B = -Sd_S$

9. Line search: Let $\alpha 1 = \min \{\alpha_S, \alpha_B\}$ and $d = (d_B, d_S, 0)$. Find an approximate solution α^* to the one-dimensional optimization problem,
Minimize $f(\bar{x} + \alpha d)$
Subject to $0 ! \alpha ! \alpha 1$

10. Replace the current point with $\bar{x} + \alpha^* d$. If $\alpha^* < \alpha 1$ then go to Step 1 (no new constraint was encountered so we remain in current subspace).

11. If a *basic* variable has reached a bound, make it *non-basic* and replace it with a *superbasic*. If a *superbasic* has hit a bound, make it *non-basic*. Go to Step 1.

The two tolerances in the algorithm are TOLRG and TOLDJ. The first is used for the check for convergence in the current subspace (Step 1). It is compared with the current value of the reduced gradient, and if it is smaller, then the optimization moves into a new sub-space or stays in the same sub-space. The other tolerance is the equivalent to determining the entering variable into the basis in the simplex algorithm. If there aren't any, then the current solution is deemed optimal.

Appendix D – Data and results for First Test Site

Assumed OD Matrix

	D1	D2
O1	0.325	0.675
O2	0.25	0.75

On-ramp Counts

	Day1		Day2		Day3		Day4		Day5	
time slice	O1	O2	O1	O2	O1	O2	O1	O2	O1	O2
1	158	2	172	7	175	7	150	8	152	7
2	198	1	183	2	202	0	185	1	197	1
3	255	3	277	4	257	4	244	5	243	5
4	264	3	281	4	283	1	231	3	261	6
5	284	7	305	15	301	14	316	13	303	8
6	344	4	354	5	352	7	381	5	394	6
7	379	10	380	12	340	14	356	22	388	12
8	393	9	383	14	400	10	405	12	436	12
9	439	1	431	1	425	1	438	0	463	0
10	518	2	477	2	511	2	483	0	485	1
11	452	27	416	26	419	23	420	20	408	14
12	389	11	442	17	402	13	421	11	389	11
13	429	14	467	16	420	11	402	5	408	9
14	461	11	475	14	505	13	461	16	471	11
15	408	39	412	41	395	38	422	46	399	35
16	402	11	426	7	400	17	426	11	410	12
17	354	33	366	31	376	42	346	26	361	40
18	397	13	400	13	418	9	420	7	394	14
19	411	8	386	11	415	17	388	9	406	11
20	324	18	332	19	367	24	356	14	346	30
21	378	26	408	27	404	22	375	17	418	19
22	370	13	387	20	375	13	393	12	376	10
23	394	35	375	33	380	27	377	33	369	34
24	424	13	379	13	385	12	407	11	382	16
25	483	4	415	2	444	5	495	6	511	9
26	390	19	390	24	382	16	350	11	396	12
27	445	18	431	12	395	22	454	13	418	24
28	404	36	388	30	399	32	395	24	386	40
29	470	8	470	5	462	9	482	3	414	1
30	412	7	409	15	372	15	409	8	401	10
31	425	42	404	35	418	39	469	38	416	32
32	435	31	427	34	412	25	437	32	448	32
33	415	9	451	13	421	13	417	14	401	12
34	448	17	483	14	482	12	450	18	461	11
35	377	23	400	32	382	29	388	38	386	24
36	340	42	334	38	352	48	378	46	387	44

Off-ramp Counts

	Day1		Day2		Day3		Day4		Day5	
time slice	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
1	43	87	49	96	49	98	43	85	43	86
2	63	129	59	125	66	134	59	122	62	128
3	81	166	86	177	82	167	76	159	77	161
4	86	178	91	192	91	189	77	161	85	178
5	92	194	96	211	100	209	100	208	98	205
6	109	226	119	237	114	237	122	252	124	257
7	123	256	122	259	116	239	121	256	129	271
8	130	269	130	270	129	270	134	278	139	288
9	139	292	139	288	138	285	143	295	147	300
10	148	298	147	285	139	301	148	301	151	304
11	150	314	135	304	161	318	139	299	147	321
12	151	329	143	310	138	302	144	297	137	287
13	145	307	150	326	143	298	139	293	135	279
14	147	301	155	315	145	307	146	300	148	312
15	143	302	150	315	152	319	137	317	144	303
16	143	305	157	322	140	303	153	288	137	288
17	126	266	132	289	125	278	127	296	124	268
18	128	272	135	283	143	282	139	282	125	281
19	138	287	129	272	142	301	133	275	146	288
20	109	243	114	245	132	277	121	255	121	258
21	134	266	123	279	118	285	121	259	129	285
22	125	261	148	280	146	271	130	272	133	275
23	116	276	128	267	123	267	131	271	131	265
24	154	297	126	277	135	275	130	281	125	275
25	147	292	140	281	140	292	148	311	150	309
26	148	323	131	276	135	281	142	291	150	306
27	141	295	142	297	128	277	138	289	141	282
28	146	310	126	290	137	280	132	294	137	314
29	148	306	155	302	146	294	149	303	141	289
30	147	311	138	297	138	302	152	302	131	278
31	142	289	137	286	121	296	141	300	136	297
32	137	320	140	311	150	298	154	340	138	303
33	156	309	159	291	146	308	145	304	155	305
34	131	297	143	323	149	299	143	301	146	299
35	143	296	150	323	142	315	152	303	140	304
36	128	255	137	288	127	264	131	288	132	283

OD Matrix Estimate (Seed 1) OD Matrix Estimate (Seed 2)

	d1	d2
o1	0.3097	0.6904
o2	0.5000	0.5000

	d1	d2
o1	0.3235	0.6765
o2	0.4051	0.5949

OD Matrix Estimate (Seed 3)

	d1	d2
o1	0.3199	0.6801
o2	0.4367	0.5633

D2

Appendix E – Radical Data Set and Results for First Test Site

	On-ramp counts				Off-ramp counts			
	Day1		Day2		Day1		Day2	
time slice	O1	O2	O1	O2	D1	D2	D1	D2
1	164	7	82	45	46	92	32	72
2	191	1	96	45	61	127	41	96
3	254	5	127	45	79	165	49	116
4	263	3	132	45	87	180	55	122
5	308	13	154	45	99	207	59	134
6	356	5	178	45	115	241	68	152
7	380	13	190	45	124	259	71	158
8	409	13	205	45	135	282	78	171
9	448	1	224	45	135	285	81	182
10	474	1	237	45	149	318	87	191
11	429	23	215	45	155	316	83	183
12	396	13	198	45	135	289	77	171
13	425	10	213	45	144	295	80	175
14	468	14	234	45	144	304	86	188
15	408	39	204	45	142	306	78	178
16	415	10	208	45	143	299	79	173
17	359	33	180	45	132	272	73	160
18	405	13	203	45	135	286	75	169
19	393	11	197	45	136	273	75	167
20	344	20	172	45	119	255	68	154
21	393	21	197	45	127	274	73	161
22	383	13	192	45	128	270	74	168
23	388	34	194	45	129	276	74	163
24	416	14	208	45	140	291	76	168
25	466	5	233	45	150	301	86	189
26	377	15	189	45	139	294	76	171
27	420	18	210	45	133	294	78	174
28	398	30	199	45	141	286	77	169
29	454	5	227	45	150	311	82	181
30	400	12	200	45	138	287	78	172
31	414	36	207	45	134	290	74	171
32	425	28	213	45	130	289	82	177
33	430	11	215	45	163	307	79	178
34	465	17	233	45	147	324	86	186
35	390	31	195	45	141	309	80	174
36	354	42	177	45	126	263	70	162

OD Matrix Estimates

Seed 1

	d1	d2
o1	0.3332	0.6669
o2	0.1834	0.8166

Seed 2

	d1	d2
o1	0.3135	0.6865
o2	0.3241	0.6759

Seed 3

	d1	d2
o1	0.3265	0.6735
o2	0.2488	0.7512

Appendix F – Th-169 Site Section Parameters

section#	lanes	length(m)
1	2	98
2	3	129
3	2	200
4	3	343
5	2	670
6	2	1028
7	2	688
8	2	1013
9	2	562
10	3	538
11	2	331
12	3	202
13	2	323
14	3	460
15	2	590
16	2	830
17	2	371
18	3	130
19	2	327
20	2	684
21	2	167
22	3	655

Appendix G – Th-169 Site Simulated Data Set and Results

Assumed OD matrix

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
O1	0.077	0.085	0.045	0.044	0.067	0.035	0.069	0.036	0.045	0.039	0.459
O2	0.077	0.085	0.045	0.044	0.067	0.035	0.069	0.036	0.045	0.039	0.459
O3	0	0.092	0.049	0.047	0.072	0.038	0.074	0.039	0.049	0.042	0.497
O4	0	0	0.054	0.052	0.079	0.042	0.082	0.043	0.054	0.046	0.547
O5	0	0	0	0.055	0.084	0.044	0.087	0.046	0.057	0.049	0.579
O6	0	0	0	0	0.089	0.047	0.092	0.048	0.06	0.051	0.613
O7	0	0	0	0	0	0.051	0.101	0.053	0.066	0.057	0.672
O8	0	0	0	0	0	0	0.106	0.056	0.07	0.06	0.709
O9	0	0	0	0	0	0	0	0.063	0.078	0.067	0.793
O10	0	0	0	0	0	0	0	0	0.083	0.071	0.846
O11	0	0	0	0	0	0	0	0	0	0.078	0.922
O12	0	0	0	0	0	0	0	0	0	0	1

On-ramp counts

time slice	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12
1	247	22	26	18	31	69	26	20	33	31	18	23
2	251	30	11	20	71	41	22	15	46	60	12	23
3	233	27	15	33	53	38	38	20	61	45	15	23
4	265	22	15	15	69	64	33	22	45	60	30	26
5	260	42	30	20	46	50	42	20	41	58	12	26
6	314	41	15	27	50	58	45	18	53	86	18	18
7	272	23	27	27	53	65	49	15	65	56	12	27
8	263	38	26	8	49	64	41	22	88	84	22	23
9	291	53	26	22	38	71	53	15	58	107	12	26
10	323	42	22	18	61	99	26	20	58	106	23	35
11	303	33	15	26	58	69	50	33	76	73	18	33
12	293	56	23	31	53	83	65	26	56	64	35	22
13	320	38	20	15	41	58	53	26	49	71	15	22
14	287	41	23	18	31	46	35	15	35	58	22	18
15	272	45	23	33	49	50	46	22	31	35	20	20
16	300	27	18	15	50	45	38	33	42	49	18	15
17	278	42	26	27	33	50	38	18	30	60	22	11
18	267	31	27	30	38	56	38	18	30	53	23	15
19	263	38	23	8	22	46	26	23	26	49	7	26
20	255	27	30	20	33	33	23	18	26	27	15	18
21	224	35	20	12	38	41	35	20	31	38	18	18
22	227	12	22	22	38	42	42	18	41	31	15	20
23	227	45	26	26	45	49	22	18	35	46	8	11
24	273	53	23	15	46	42	15	12	31	33	20	15
25	235	18	20	15	30	31	20	26	42	42	27	8
26	201	26	23	15	26	42	20	15	23	30	7	11
27	203	18	22	5	33	46	26	38	27	33	20	8
28	200	38	22	22	30	30	26	11	38	33	12	5
29	209	35	15	23	26	38	15	15	18	23	15	8
30	192	38	22	27	35	53	23	38	23	30	11	15
31	192	27	27	20	38	18	15	12	38	26	18	5
32	192	15	15	18	31	31	27	15	22	15	18	5
33	202	22	33	23	31	23	26	30	31	30	20	7
34	176	27	23	38	38	38	26	20	15	22	30	7
35	198	41	20	15	33	18	20	12	42	22	27	12
36	186	22	22	22	38	31	15	27	42	26	26	8

Off-ramp counts

time slice	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
1	20	22	10	6	11	5	10	5	9	5	82
2	21	24	14	18	30	17	35	18	28	24	263
3	20	25	13	17	31	18	36	23	31	27	355
4	22	25	15	18	33	17	38	21	32	28	382
5	23	28	15	18	30	20	39	25	35	32	388
6	27	31	17	19	33	18	41	22	34	32	389
7	23	28	17	20	37	21	44	26	38	33	420
8	23	26	15	18	33	20	41	26	38	33	428
9	25	32	17	20	35	20	42	27	41	34	452
10	29	33	18	19	40	23	45	24	35	37	430
11	26	31	19	21	38	22	48	25	42	31	435
12	26	31	16	19	38	24	47	33	41	40	467
13	29	32	19	21	36	22	49	26	41	36	447
14	25	30	18	20	33	22	41	32	44	41	433
15	24	28	16	18	33	18	42	23	32	34	436
16	25	31	18	21	34	20	41	23	34	31	433
17	24	29	16	18	32	20	40	25	36	34	393
18	23	29	16	18	34	18	40	21	31	30	367
19	23	27	16	17	30	18	36	25	31	31	364
20	22	27	16	18	29	16	33	19	28	23	313
21	20	25	13	16	30	17	37	20	28	25	312
22	18	22	14	16	26	17	32	21	26	25	323
23	20	24	13	15	27	16	34	20	30	25	307
24	24	29	15	16	31	17	34	20	26	23	310
25	20	26	16	19	31	18	36	23	30	28	328
26	18	21	13	13	25	15	31	16	26	26	294
27	17	21	12	14	25	14	31	18	24	22	271
28	18	22	12	14	24	13	29	19	27	25	274
29	18	22	13	14	25	14	32	15	23	21	250
30	19	21	13	15	27	14	31	16	21	20	261
31	17	20	13	14	25	14	30	21	29	24	285
32	16	18	12	13	25	14	28	16	21	19	245
33	18	22	12	13	21	13	28	16	23	21	255
34	16	19	12	15	25	14	30	16	23	23	260
35	18	22	13	14	24	14	29	18	23	22	279
36	16	20	12	15	22	13	29	19	23	24	268

OD estimate from Seed 2

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.0798	0.0889	0.0336	0.0479	0.0815	0.0429	0.0762	0.0300	0.0299	0.0103	0.4791
o2	0.0463	0.0486	0.0530	0.0375	0.0392	0.0014	0.0672	0.0510	0.0253	0.0526	0.5779
o3	0	0.0551	0.0341	0.0173	0.0769	0.0135	0.0836	0.0534	0.0235	0.0505	0.5922
o4	0	0	0.0518	0.0308	0.0717	0.0278	0.0646	0.0190	0.0681	0.0440	0.6222
o5	0	0	0	0.0288	0.0704	0.0279	0.0444	0.0551	0.0841	0.0539	0.6353
o6	0	0	0	0	0.0504	0.0535	0.0657	0.0530	0.0441	0.0675	0.6658
o7	0	0	0	0	0	0.0399	0.0793	0.0485	0.0596	0.0720	0.7006
o8	0	0	0	0	0	0	0.0977	0.0571	0.0494	0.0637	0.7321
o9	0	0	0	0	0	0	0	0.0613	0.0722	0.0569	0.8096
o10	0	0	0	0	0	0	0	0	0.0807	0.0774	0.8419
o11	0	0	0	0	0	0	0	0	0	0.0749	0.9252
o12	0	0	0	0	0	0	0	0	0	0	1.00

OD estimate from Seed 3

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.0780	0.0860	0.0447	0.0441	0.0678	0.0343	0.0698	0.0360	0.0452	0.0389	0.4552
o2	0.0779	0.0857	0.0454	0.0445	0.0674	0.0353	0.0694	0.0362	0.0448	0.0388	0.4546
o3	0	0.0930	0.0492	0.0482	0.0731	0.0383	0.0752	0.0393	0.0486	0.0420	0.4930
o4	0	0	0.0543	0.0532	0.0806	0.0422	0.0829	0.0433	0.0536	0.0463	0.5436
o5	0	0	0	0.0563	0.0852	0.0446	0.0877	0.0458	0.0567	0.0490	0.5748
o6	0	0	0	0	0.0903	0.0473	0.0929	0.0485	0.0600	0.0519	0.6090
o7	0	0	0	0	0	0.0520	0.1021	0.0533	0.0660	0.0571	0.6695
o8	0	0	0	0	0	0	0.1077	0.0563	0.0696	0.0602	0.7062
o9	0	0	0	0	0	0	0	0.0631	0.0780	0.0675	0.7915
o10	0	0	0	0	0	0	0	0	0.0833	0.0721	0.8447
o11	0	0	0	0	0	0	0	0	0	0.0786	0.9214
o12	0	0	0	0	0	0	0	0	0	0	1.00

OD estimate from Seed 5

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.0766	0.0844	0.0448	0.0440	0.0670	0.0351	0.0692	0.0361	0.0450	0.0387	0.4589
o2	0.0766	0.0843	0.0448	0.0440	0.0670	0.0351	0.0693	0.0361	0.0451	0.0388	0.4590
o3	-1	0.0913	0.0486	0.0476	0.0726	0.0381	0.0750	0.0391	0.0488	0.0420	0.4971
o4	-1	-1	0.0534	0.0524	0.0799	0.0419	0.0825	0.0431	0.0537	0.0462	0.5470
o5	-1	-1	-1	0.0554	0.0844	0.0442	0.0872	0.0455	0.0567	0.0488	0.5778
o6	-1	-1	-1	-1	0.0893	0.0468	0.0923	0.0482	0.0601	0.0517	0.6117
o7	-1	-1	-1	-1	-1	0.0514	0.1014	0.0529	0.0659	0.0567	0.6717
o8	-1	-1	-1	-1	-1	-1	0.1068	0.0557	0.0695	0.0598	0.7081
o9	-1	-1	-1	-1	-1	-1	-1	0.0624	0.0778	0.0670	0.7928
o10	-1	-1	-1	-1	-1	-1	-1	-1	0.0830	0.0714	0.8456
o11	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.0779	0.9221
o12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1.00

Appendix H – Th-169 Site Real Data Set and Results

On-ramp counts (3 days)
Day 1 (Nov 1 2002, 7am – 10am)

	day 1											
time	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12
1	243	11	13	9	16	36	13	10	17	16	9	12
2	247	15	5	10	37	21	11	7	24	31	6	12
3	229	14	7	17	28	19	20	10	32	23	8	12
4	261	11	8	8	36	33	17	11	23	31	15	13
5	257	22	15	10	24	26	22	10	21	30	6	13
6	310	21	7	14	26	30	23	9	28	45	9	9
7	268	12	14	14	28	34	25	7	34	29	6	14
8	259	19	13	4	25	33	21	11	46	44	11	12
9	287	27	13	11	20	37	28	7	30	56	6	13
10	318	22	11	9	32	52	13	10	30	55	12	18
11	299	17	8	13	30	36	26	17	40	38	9	17
12	289	29	12	16	27	43	34	13	29	33	18	11
13	316	20	10	8	21	30	28	13	25	37	7	11
14	284	21	12	9	16	24	18	8	18	30	11	9
15	268	23	12	17	25	26	24	11	16	18	10	10
16	296	14	9	7	26	23	20	17	22	25	9	8
17	274	22	13	14	17	26	20	9	15	31	11	5
18	263	16	14	15	20	29	20	9	15	28	12	7
19	259	20	12	4	11	24	13	12	13	25	3	13
20	252	14	15	10	17	17	12	9	13	14	8	9
21	221	18	10	6	19	21	18	10	16	19	9	9
22	224	6	11	11	19	22	22	9	21	16	8	10
23	223	23	13	13	23	25	11	9	18	24	4	5
24	269	27	12	8	24	22	8	6	16	17	10	8
25	231	9	10	8	15	16	10	13	22	22	14	4
26	198	13	12	8	13	22	10	7	12	15	3	5
27	200	9	11	2	17	24	13	19	14	17	10	4
28	197	20	11	11	15	15	13	5	19	17	6	2
29	206	18	8	12	13	19	8	8	9	12	8	4
30	189	20	11	14	18	28	12	20	12	15	5	7
31	189	14	14	10	20	9	8	6	19	13	9	2
32	189	7	7	9	16	16	14	8	11	8	9	2
33	199	11	17	12	16	12	13	15	16	15	10	3
34	174	14	12	19	20	19	13	10	8	11	15	3
35	195	21	10	8	17	9	10	6	22	11	14	6
36	183	11	11	11	20	16	7	14	22	13	13	4

Day 2 (Nov 2 2002, 7am – 10am)

	day 2											
time	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12
1	220	11	8	12	13	38	16	14	15	28	5	11
2	222	8	16	4	33	13	18	12	29	25	7	16
3	228	16	10	8	31	36	17	12	29	28	19	12
4	298	10	10	12	30	32	22	11	19	33	9	18
5	266	12	12	12	31	19	21	14	20	36	5	7
6	313	18	12	11	24	24	39	10	30	32	9	11
7	288	15	12	12	33	31	20	16	27	21	2	18
8	274	15	11	12	19	40	21	14	31	43	17	14
9	313	14	12	8	27	36	27	13	37	51	8	9
10	298	16	12	13	31	37	24	13	35	46	14	15
11	269	27	10	12	31	36	28	17	34	44	9	16
12	296	26	13	12	27	43	29	22	28	29	10	15
13	296	18	15	5	22	34	31	12	28	34	10	15
14	295	20	10	6	23	25	15	11	20	33	9	3
15	273	18	13	9	19	24	16	9	17	23	3	8
16	265	20	12	13	27	23	19	13	21	27	10	10
17	240	13	13	9	20	30	11	13	16	14	7	12
18	243	16	18	9	21	31	12	3	23	27	9	12
19	229	17	11	17	10	20	10	15	21	15	6	6
20	223	8	16	14	7	19	17	5	15	10	11	9
21	259	12	7	12	21	27	7	7	16	14	6	11
22	239	16	12	7	21	29	15	5	13	22	4	10
23	229	15	11	13	34	18	10	12	10	21	9	12
24	215	13	11	15	18	22	16	13	27	24	11	9
25	222	15	11	5	24	22	10	10	17	22	15	4
26	173	12	11	15	17	9	16	13	17	19	10	6
27	182	22	8	10	18	12	12	8	18	19	10	2
28	184	14	10	16	17	21	7	11	17	13	9	7
29	203	15	7	12	20	12	11	7	6	16	7	9
30	223	21	8	13	6	16	11	18	16	15	11	6
31	202	14	11	7	17	16	10	8	20	12	6	9
32	214	14	9	10	21	15	15	8	17	15	13	7
33	198	17	12	11	11	14	14	5	17	3	6	7
34	229	16	7	13	16	16	12	14	13	14	12	5
35	207	13	15	14	21	8	8	12	14	10	8	1
36	207	6	11	11	15	17	9	11	13	14	15	3

Day 3 (Nov 3 2002, 7am – 10am)

	day 3											
time	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12
1	240	14	12	6	21	23	13	11	15	23	9	5
2	243	9	10	5	37	22	5	9	20	31	10	10
3	238	14	6	11	25	21	21	10	27	24	11	13
4	240	12	5	8	39	28	16	12	26	16	10	10
5	297	16	12	8	35	27	19	12	30	30	9	8
6	282	11	11	8	28	29	20	3	26	36	7	10
7	270	16	13	8	23	29	23	15	33	34	4	18
8	300	16	11	10	14	33	20	12	29	31	13	9
9	318	15	16	16	33	28	32	8	27	29	11	12
10	291	16	7	17	35	32	29	21	33	31	6	11
11	285	25	12	18	23	44	21	13	32	44	11	19
12	312	20	12	4	39	50	34	14	28	21	12	6
13	272	14	7	18	24	28	21	14	23	33	6	7
14	300	21	13	9	22	22	21	5	21	28	9	16
15	245	18	14	15	24	23	14	12	16	29	9	4
16	279	13	13	16	15	14	15	12	16	17	5	6
17	278	13	10	16	16	32	10	16	21	25	5	20
18	264	18	14	14	31	25	12	12	15	17	3	8
19	267	15	11	9	20	24	13	11	26	8	5	11
20	257	10	18	4	26	26	11	5	21	10	7	13
21	223	9	11	9	19	23	15	13	18	15	5	10
22	206	16	14	17	20	22	14	11	20	17	7	7
23	212	18	14	10	19	10	13	16	13	12	6	9
24	218	19	10	8	28	27	15	10	23	15	5	11
25	227	17	17	12	26	22	13	11	21	20	11	8
26	205	19	8	12	18	20	11	16	25	19	13	3
27	230	13	17	10	18	14	7	11	13	8	7	8
28	229	15	9	14	24	22	13	10	15	19	12	3
29	187	18	6	12	12	15	8	12	10	13	13	8
30	203	7	15	10	22	20	5	23	15	12	4	6
31	181	17	8	20	15	18	8	18	16	15	6	4
32	206	9	13	10	14	16	9	11	14	11	8	4
33	200	11	7	20	19	9	15	19	12	15	10	7
34	171	7	10	12	19	15	17	7	21	13	10	6
35	183	21	17	11	22	20	9	13	16	13	10	14
36	197	10	16	12	28	10	12	12	13	16	7	2

Off-ramp counts (3 days)
Day 1 (Nov 1 2002, 7am – 10am)

time	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
1	31	30	22	28	45	18	32	14	16	15	179
2	35	34	22	27	29	12	32	16	25	19	187
3	24	39	15	34	12	13	24	14	14	23	203
4	32	27	18	28	26	11	36	16	16	23	204
5	35	27	19	27	21	14	39	12	24	27	230
6	40	33	24	23	28	11	39	17	18	14	228
7	36	44	23	26	34	18	55	23	46	28	230
8	43	32	17	25	22	12	30	19	30	25	245
9	45	41	24	17	26	9	37	20	16	25	257
10	41	39	19	16	27	24	34	23	28	24	263
11	48	39	23	17	42	15	50	14	34	34	305
12	38	44	21	24	35	18	46	14	23	25	285
13	52	51	21	25	30	13	41	22	21	16	240
14	51	67	26	18	37	13	41	19	31	19	215
15	44	59	21	21	35	9	29	19	17	17	207
16	44	64	26	14	27	10	30	16	19	21	194
17	40	46	25	22	28	20	27	11	21	21	203
18	40	59	18	11	32	17	30	12	24	20	192
19	40	52	12	11	19	7	35	16	30	18	200
20	34	40	17	15	36	22	16	14	21	6	183
21	27	45	12	8	24	21	23	15	22	10	184
22	36	30	20	16	46	19	28	10	25	13	183
23	27	21	20	22	22	18	19	17	14	9	162
24	37	40	20	22	21	26	36	17	13	14	167
25	29	40	20	25	29	17	22	21	14	16	180
26	25	26	19	18	58	22	23	18	9	9	162
27	27	20	16	15	22	20	19	10	15	9	151
28	16	35	18	18	26	15	22	13	17	9	165
29	20	24	15	18	28	12	23	12	11	7	166
30	17	25	23	11	25	17	25	13	19	13	160
31	26	28	19	12	27	19	23	9	17	8	171
32	19	30	13	12	19	13	19	22	8	12	134
33	22	22	22	10	20	10	17	8	7	12	172
34	21	21	24	17	18	7	25	16	12	19	172
35	27	19	14	11	37	8	18	13	7	11	162
36	22	22	15	14	18	15	20	17	15	7	173

Day 2 (Nov 2 2002, 7am – 10am)

time	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
1	28	29	20	26	47	17	48	16	19	16	184
2	29	35	16	29	26	10	19	12	19	17	179
3	28	33	13	28	34	12	36	14	22	21	204
4	30	29	20	32	24	17	31	21	13	16	209
5	37	37	12	23	38	12	37	18	26	20	225
6	39	37	17	25	21	11	36	22	36	23	222
7	42	36	33	26	36	9	48	13	33	22	279
8	32	33	27	17	24	18	35	19	25	23	269
9	44	40	22	21	37	14	33	14	23	20	226
10	49	40	26	24	51	15	44	14	35	28	305
11	30	41	16	19	34	22	34	30	26	19	272
12	48	44	23	24	31	15	47	23	33	27	236
13	33	69	33	20	43	21	45	13	24	29	256
14	49	54	25	18	22	14	42	19	22	19	206
15	46	53	21	13	46	12	33	14	29	19	177
16	47	43	27	21	46	16	26	16	19	17	180
17	44	54	17	13	28	12	25	16	28	18	213
18	41	43	12	17	20	16	25	17	21	19	161
19	42	46	12	14	28	14	24	14	27	16	171
20	30	47	18	16	18	13	24	19	20	13	181
21	33	45	23	17	21	17	25	13	12	9	152
22	31	30	26	15	26	28	29	15	23	13	178
23	36	32	21	15	24	23	27	12	23	5	177
24	21	39	26	22	33	12	28	17	16	11	187
25	28	34	15	18	33	24	19	22	15	8	184
26	24	23	16	17	30	11	18	16	12	7	173
27	24	28	7	18	16	10	13	14	11	7	180
28	32	19	15	15	39	8	28	12	18	5	140
29	20	30	11	26	29	20	22	12	10	6	157
30	32	23	19	13	20	13	23	12	7	14	148
31	15	21	20	18	40	10	20	12	8	15	187
32	32	24	14	16	31	19	22	19	7	8	185
33	31	23	17	22	10	10	17	23	10	13	153
34	9	38	17	18	20	13	11	13	10	8	175
35	19	26	16	24	24	14	27	23	17	14	171
36	25	27	22	18	35	11	24	18	17	10	142

Day 3 (Nov 3 2002, 7am – 10am)

time	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
1	36	30	13	30	19	12	32	18	20	25	168
2	26	31	18	31	28	4	36	10	19	15	175
3	24	26	16	22	22	13	26	4	19	22	209
4	31	31	20	35	16	12	41	17	17	18	215
5	31	39	15	31	19	9	34	11	27	14	220
6	27	35	39	22	37	17	38	17	35	20	223
7	30	52	25	17	28	8	36	16	15	12	237
8	30	45	18	15	36	8	41	15	21	13	239
9	37	37	25	28	27	14	40	22	33	23	245
10	35	38	16	27	28	18	41	28	19	28	282
11	38	40	20	22	35	19	39	17	22	28	268
12	44	53	22	18	34	24	35	20	31	20	240
13	36	50	20	22	30	14	42	31	28	21	260
14	44	53	26	20	24	8	38	15	28	25	212
15	40	49	25	18	37	15	32	18	25	12	209
16	41	46	19	8	16	13	34	13	20	19	160
17	47	30	26	19	18	17	31	12	24	22	184
18	44	52	21	21	25	19	24	19	18	19	211
19	38	48	12	20	26	18	28	16	17	20	197
20	40	45	24	21	6	18	33	18	21	10	188
21	38	33	18	8	6	18	21	16	22	14	177
22	27	22	14	10	7	13	26	16	23	8	166
23	33	32	22	19	13	18	25	8	11	6	194
24	28	34	18	17	14	21	24	18	15	6	168
25	21	37	27	15	15	16	25	20	15	11	184
26	25	31	17	17	17	17	20	13	20	13	192
27	29	24	20	20	12	14	15	18	16	3	165
28	21	32	21	17	21	19	21	17	11	15	175
29	20	19	17	12	27	16	28	18	26	15	166
30	26	22	15	13	13	13	19	13	9	9	185
31	24	17	23	17	25	20	19	16	13	9	162
32	22	22	8	17	19	21	17	14	7	14	174
33	26	26	14	18	13	11	12	19	12	9	169
34	21	25	15	14	23	7	19	15	13	11	183
35	16	11	13	14	16	18	18	18	15	7	178
36	30	26	21	16	21	15	25	27	10	13	150

OD estimate – 1 day data

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.1380	0.1414	0.0641	0.0517	0.0515	0.0388	0.0861	0.0000	0.0281	0.0000	0.4003
o2	0.0224	0.0629	0.0793	0.0658	0.1850	0.0490	0.0501	0.2530	0.0586	0.0318	0.1422
o3	0	0.0546	0.0770	0.0951	0.2531	0.0677	0.0770	0.0620	0.0539	0.0747	0.1851
o4	0	0	0.0821	0.0910	0.2068	0.0737	0.1106	0.0817	0.0521	0.0943	0.2076
o5	0	0	0	0.1259	0.2014	0.0568	0.0488	0.0413	0.0620	0.0803	0.3836
o6	0	0	0	0	0.1418	0.0000	0.0738	0.1050	0.1074	0.1378	0.4343
o7	0	0	0	0	0	0.0642	0.0391	0.1076	0.0445	0.1167	0.6279
o8	0	0	0	0	0	0	0.0744	0.0506	0.0733	0.0906	0.7112
o9	0	0	0	0	0	0	0	0.1229	0.0796	0.0535	0.7440
o10	0	0	0	0	0	0	0	0	0.0770	0.0883	0.8348
o11	0	0	0	0	0	0	0	0	0	0.1003	0.8997
o12	0	0	0	0	0	0	0	0	0	0	1

OD estimate – 1 day data (with warm-up)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.1312	0.1489	0.0687	0.0469	0.0864	0.0296	0.0986	0.0307	0.0692	0.0316	0.2583
o2	0.1075	0.1088	0.0688	0.0342	0.0307	0.0597	0.0710	0.0486	0.0000	0.0540	0.4167
o3	0	0.1348	0.0445	0.0455	0.0853	0.0554	0.0611	0.0253	0.0268	0.0194	0.5020
o4	0	0	0.0647	0.0780	0.0748	0.0371	0.0350	0.0799	0.0172	0.0455	0.5679
o5	0	0	0	0.1106	0.0841	0.0336	0.0685	0.0588	0.0000	0.0305	0.6139
o6	0	0	0	0	0.0875	0.0442	0.0513	0.0703	0.0414	0.0425	0.6629
o7	0	0	0	0	0	0.0553	0.1168	0.0445	0.0213	0.0290	0.7332
o8	0	0	0	0	0	0	0.0694	0.0715	0.0593	0.0431	0.7567
o9	0	0	0	0	0	0	0	0.0576	0.0548	0.0587	0.8289
o10	0	0	0	0	0	0	0	0	0.0572	0.0708	0.8721
o11	0	0	0	0	0	0	0	0	0	0.0659	0.9341
o12	0	0	0	0	0	0	0	0	0	0	1

OD estimate – 3 days data

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.1218	0.1356	0.0667	0.0483	0.0388	0.0305	0.0710	0.0273	0.0355	0.0137	0.4109
o2	0.0754	0.0660	0.0900	0.0796	0.1888	0.0681	0.0826	0.0646	0.0611	0.0741	0.1498
o3	0	0.0535	0.0767	0.1004	0.2040	0.0773	0.0855	0.0703	0.0582	0.0729	0.2011
o4	0	0	0.0755	0.0955	0.2013	0.0772	0.0839	0.0662	0.0604	0.0842	0.2558
o5	0	0	0	0.1035	0.1746	0.0906	0.0807	0.0546	0.0570	0.0828	0.3563
o6	0	0	0	0	0.1827	0.0775	0.0858	0.0429	0.0556	0.0918	0.4637
o7	0	0	0	0	0	0.0572	0.1015	0.0630	0.0525	0.0870	0.6388
o8	0	0	0	0	0	0	0.0773	0.0628	0.0615	0.0888	0.7097
o9	0	0	0	0	0	0	0	0.0608	0.0583	0.0780	0.8030
o10	0	0	0	0	0	0	0	0	0.0589	0.0764	0.8646
o11	0	0	0	0	0	0	0	0	0	0.0764	0.9297
o12	0	0	0	0	0	0	0	0	0	0	1

OD estimate – 3 days data (New Objective Function)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.1333	0.1438	0.0700	0.0558	0.0434	0.0285	0.0774	0.0291	0.0358	0.0119	0.3711
o2	0.0746	0.0654	0.0825	0.0847	0.1853	0.0756	0.0845	0.0635	0.0619	0.0802	0.1418
o3	0	0.0582	0.0796	0.0947	0.2023	0.0791	0.0837	0.0681	0.0529	0.0847	0.1967
o4	0	0	0.0757	0.0960	0.2012	0.0750	0.0847	0.0699	0.0571	0.0846	0.2557
o5	0	0	0	0.0983	0.1782	0.0848	0.0915	0.0596	0.0630	0.0740	0.3505
o6	0	0	0	0	0.1856	0.0673	0.0925	0.0566	0.0564	0.0888	0.4528
o7	0	0	0	0	0	0.0570	0.0976	0.0666	0.0541	0.0875	0.6371
o8	0	0	0	0	0	0	0.0818	0.0644	0.0559	0.0879	0.7100
o9	0	0	0	0	0	0	0	0.0594	0.0614	0.0764	0.8028
o10	0	0	0	0	0	0	0	0	0.0579	0.0742	0.8678
o11	0	0	0	0	0	0	0	0	0	0.0777	0.9223
o12	0	0	0	0	0	0	0	0	0	0	1

OD estimate – 3 days data (2-step optimization)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
o1	0.1252	0.1364	0.0636	0.0545	0.0377	0.0225	0.0740	0.0304	0.0352	0.0096	0.4109
o2	0.0755	0.0664	0.0898	0.0797	0.1889	0.0667	0.0835	0.0648	0.0612	0.0737	0.1498
o3	0	0.0538	0.0771	0.1008	0.2016	0.0777	0.0858	0.0703	0.0586	0.0733	0.2011
o4	0	0	0.0755	0.0955	0.2013	0.0768	0.0839	0.0662	0.0608	0.0842	0.2558
o5	0	0	0	0.1056	0.1762	0.0891	0.0807	0.0534	0.0576	0.0811	0.3563
o6	0	0	0	0	0.1829	0.0766	0.0869	0.0431	0.0559	0.0909	0.4637
o7	0	0	0	0	0	0.05647	0.1016	0.0627	0.0531	0.0872	0.6388
o8	0	0	0	0	0	0	0.0771	0.0629	0.0605	0.0898	0.7097
o9	0	0	0	0	0	0	0	0.060304	0.0593	0.0774	0.8030
o10	0	0	0	0	0	0	0	0	0.0597	0.0757	0.8646
o11	0	0	0	0	0	0	0	0	0	0.0703	0.9297
o12	0	0	0	0	0	0	0	0	0	0	1

APPENDIX I – External Sub-Routines Used in OD Estimation Program

The initial subroutine to set-up the program in MINOS

```
C
C initial - initializes the data
C calculates bfix, actual off-ramp counts and the weights
C opens a file to pass comments during the running (unit=3)
C
subroutine initial()
      implicit            double precision (a-h,o-z)
      integer             nor,ndes,ntime,ndays,nod,i,j,k,callno
      parameter           (nor=12,ndes=11,ntime=36,ndays=1)
      double precision    b(nor,ndes),bb(nor*ndes),bfix(nor,ndes)
      double precision    aoffrc(ndays,ntime,ndes),onrc(ndays,ntime,nor)
      double precision    bl(nor*ndes),bu(nor*ndes)
      character*(*)       paths,ftype
      parameter           (paths='paths.dat',ftype='f8.4')
      character*80        odfile,offile,outfile,mpsfile,scefile,olist,dlist
      character*80        onfile,countsfile,ttfile,oldfile,simoffile
      double precision    avgoff(ndays,ndes),sumoff(ndays,ndes)
      double precision    weight(ndays,ndes),stdevoff(ndays,ndes)
      double precision    varoff(ndays,ndes),ssoff(ndays,ndes),coefft
      parameter           (coefft=1.0d+0)
      character*200       fmttype

      common /matrix/bfix
      common /off/aoffrc
      common /weights/weight
      common /callid/callno
      common /pathod/odfile
      common /pathsce/scefile
      common /pathtt/onfile,olist,dlist,ttfile
      common /pathget/countsfile
      common /pathsimoff/simoffile

      open(unit=1,file=paths,status='old')
      callgetpaths(1,odfile,oldfile,offile,outfile,mpsfile,scefile,olist
          &t,dlist,onfile,ttfile,countsfile,simoffile)
      close(1)
C
C unit=3 is the file to send stuff to track the pgm
C
      open(unit=3,file=outfile,status='old')
C
```

```
C get the OD matrix and the offramp counts
C

      open(unit=1,file=oldfile,status='old')
      open(unit=2,file=offile,status='old')
      call read2d(1,b,nor,ndes)
      call read3d(2,aoffrc,ndays,ntime,ndes)
      close(1)
      close(2)
C
C unstack matrix and send useful entries
C
      nod = 0
      do 20 i=1,nor
do 30 j=1,ndes
                                                  if ((b(i,j).eq.-1.0d+0)) then
bfix(i,j) = 0.0d+0
                                                  else if (b(i,j).eq.1.0d+0) then
bfix(i,j) = 1.0d+0
           else
           bfix(i,j) = -1.0d+0
           nod = nod + 1
           bb(nod) = b(i,j)
           end if
      30    continue
      20 continue
      write(3,*) 'Inside initial'
      write(3,*) 'the # of independent var =',nod
      write(3,*) 'the OD matrix '
      do 40 i=1,nor
        write(3,*) (b(i,j),j=1,ndes)
      40 continue
      write(3,*) 'the bfix matrix '
      do 50 i=1,nor
        write(3,*) (bfix(i,j),j=1,ndes)
      50 continue
      do 55 i=1,nod
           bl(i) = 0.0d+0
           bu(i) = 1.0d+0
      55 continue
C
C write the needed info into the MPS file
C
      open(unit=4,file=mpsfile,status='old')
      write(4,'(a4)') 'NAME'
      write(4,'(a4)') 'ROWS'
```

```
      do 60 i=1,nor
if (bfix(i,ndes).lt.0.0d+0) then
                                                write(4,'(t2,a1,t5,a3,i3.3)') 'E','ORI',i
end if
      60 continue
      write(4,'(a7)') 'COLUMNS'
      k=0
      do 65 i=1,nor
do 70 j=1,ndes
                                                if (bfix(i,j).lt.0.0d+0) then
k=k+1
write(4,'(t5,a1,i3.3,t15,a3,i3.3,t25,f11.7)') 'X',k,'ORI
           &',i,coefft
end if
   70   continue
   65 continue

      write(4,'(a3)') 'RHS'
      do 75 i=1,nor
        if (bfix(i,(ndes-1)).lt.0.0d+0) write(4,'(t5,a7,t15,a3,i3.3,t25
     &,f11.7)') 'DEMANDS','ORI',i,coefft
   75 continue

      write(4,'(a6)') 'BOUNDS'
      do 85 i=1,nod
        write(4,'(t2,a2,t5,a6,t15,a1,i3.3,t25,f11.7)') 'LO','BOUND1','X'
     &,i,bl(i)
        write(4,'(t2,a2,t5,a6,t15,a1,i3.3,t25,f11.7)') 'UP','BOUND1','X'
     &,i,bu(i)
   85 continue
      do 90 i=1,nod
       write(4,'(t2,a2,t5,a7,t15,a1,i3.3,t25,f11.7)') 'FX','INITIAL','X'
     &,i,bb(i)
   90 continue
      write(4,'(a6)') 'ENDATA'
      close(4)
C
C  calculate the weights from the off-ramp counts
C
      do 93 i=1,ndays
      do 95 j=1,ndes
      sumoff(i,j) = 0.0d+0
      avgoff(i,j) = 0.0d+0
      do 100 k=1,ntime
      sumoff(i,j)=sumoff(i,j)+aoffrc(i,k,j)
  100     continue
```

```
        avgoff(i,j)=sumoff(i,j)/ntime
  95    continue
  93 continue

     do 103 i=1,ndays
       do 105 j=1,ndes
         ssoff(i,j) = 0.0d+0
         varoff(i,j) = 0.0d+0
         stdevoff(i,j) = 0.0d+0
         weight(i,j) = 0.0d+0
         do 110 k=1,ntime
           ssoff(i,j)=ssoff(i,j)+((aoffrc(i,k,j)-avgoff(i,j))**2)
  110     continue
         varoff(i,j) = ssoff(i,j)/(ntime-1)
         stdevoff(i,j) = sqrt(varoff(i,j))
         weight(i,j) = 1.0d+0/stdevoff(i,j)
  105    continue
 103 continue

     write(3,*) 'the weights are'
     do 115 i=1,ndays
          write(3,120) 'Day ',i,' :'
          call retfmt(ftype,ndes,fmttype)
          write(3,fmt=fmttype) (weight(i,j),j=1,ndes)
 115 continue
 120 format(a,i1,a,$)
     write(3,*) 'Exiting Initial'
     callno=1
 end
 C
 C
 subroutine read2d(fnum,matrix,row,col)
     integer              fnum,row,col,i,j,k
     double precision     matrix(row,col)
     character*10         comment

     read(fnum,*) comment
     do 200 j=1,row
     read(fnum,*) (matrix(j,k),k=1,col)

 200  continue
 end

 subroutine read3d(fnum,matrix,ht,row,col)

     integer              fnum,row,ht,col,i,j,k
     double precision     matrix(ht,row,col)
     character*10         comment
```

```
      do 300 i=1,ht
      read(fnum,*) comment
      do 310 j=1,row
      read(fnum,*) (matrix(i,j,k),k=1,col)
310   continue
300   continue
end

subroutine getpaths(fnum,odfile,oldfile,offile,outfile,mpsfile,sce
     &file,olist,dlist,onfile,ttfile,countsfile,simoffile)
      integer           fnum
      character*80      odfile,offile,outfile,mpsfile,comment,scefile
      character*80      olist,dlist,onfile,ttfile,countsfile,oldfile
      character*80      simoffile

      read(fnum,*) comment
      read(fnum,*) odfile
      read(fnum,*) oldfile
      read(fnum,*) onfile
      read(fnum,*) offile
      read(fnum,*) simoffile
      read(fnum,*) outfile
      read(fnum,*) mpsfile
      read(fnum,*) scefile
      read(fnum,*) olist
      read(fnum,*) dlist
      read(fnum,*) ttfile
      read(fnum,*) countsfile
end
C
C
subroutine retfmt(ftype,nrep,fmttype)
      integer           nrep,i
      character*(*)     fmttype,ftype

      fmttype=ftype//')'
      do 400 i=1,nrep
if (i.ne.nrep) fmttype = ftype//','//fmttype
400 continue
          fmttype='('//fmttype
end
The subroutine to calculate the time sliced trip table and write to AIMSUN readable
      format
C
C subroutine that reads the od matrix and creates the trip table
```

```
C and writes into the aimsun readable format file Matrix.des
C
subroutine ttable(ndays,nor,ndes,ntime,day)
      implicit            double precision (a-h,o-z)
      integer             nor,ndes,ntime,ndays,day
      double precision    b(nor,ndes)
      double precision    onrc(ndays,ntime,nor)
      integer             tt(ntime,nor,ndes)
      integer             origin(nor),destination(ndes)
      character*80        dfile,odfile,ttfile,olist,dlist
      integer             i,j,k

      common /pathod/odfile
      common /pathtt/dfile,olist,dlist,ttfile

      open(unit=1,file=odfile,status='old')
      open(unit=2,file=dfile,status='old')
      open(unit=14,file=olist,status='old')
      open(unit=5,file=dlist,status='old')
      call read2d(1,b,nor,ndes)
      call read3d(2,onrc,ndays,ntime,nor)
      call get(nor,origin,14)
      call get(ndes,destination,5)
      close(1)
      close(2)
      close(14)
      close(5)

      call ttcal(b,onrc,ndays,nor,ndes,ntime,tt,day)
      open(unit=7,file=ttfile,status='old')
      call fwrite(tt,ntime,nor,ndes,origin,destination,7)
      close(7)

end
C
C ttcal - calculates the trip table from the OD matrix
C and the on-ramp counts
C
subroutine ttcal(b,onrc,ndays,nor,ndes,ntime,tt,day)
      integer nor,ndes,ntime,i,j,k,ndays,day
      double precision  b(nor,ndes), onrc(ndays,ntime,nor)
      integer           tt(ntime,nor,ndes)
          do 100 i=1,ntime
do 110 j=1,nor
                                                  do 120 k=1,ndes
if (b(j,k).gt.0.0d+0) then
```

```
tt(i,j,k) = nint(b(j,k)*onrc(day,i,j))
else
tt(i,j,k) = 0
end if
 120       continue
 110     continue
100  continue
end

C
C fwrite - writes the trip table entries into the AISMSUN
C readable format file Matrix.des
C
subroutine fwrite(tt,ntime,nor,ndes,origin,destination,fnum)
      integer          fnum,nor,ndes,ntime,i,j,k
      integer          tt(ntime,nor,ndes)
      integer          start,end,delt
      integer          origin(nor),destination(ndes)
      parameter          (start=0,end=180,delt=5)

      integer nveh,nvehcl,npoll,nguide,ndrive
      parameter (nveh=1,nvehcl=0,npoll=0,nguide=0,ndrive=1)
      real length(4),width(4),maxds(4),maxacc(4),nordec(4),maxdec(4)
      real speeda(4),mindis(4),givtim(4),guiacc(4)
      real fuelp(7)
      real crutol
      parameter (crutol = 0.0)

      data fuelp/7*0.0/
      data length/3*4.0,0.0/
      data width/3*2.0,0.0/
      data maxds/3*80.0,0.0/
      data maxacc/3*3.0,0.0/
      data nordec/3*4.0,0.0/
      data maxdec/3*6.0,0.0/
      data speeda/3*1.0,0.0/
      data mindis/3*1.0,0.0/
      data givtim/3*30.0,0.0/
      data guiacc/3*1.0,0.0/
      write(fnum,'(a)')'* ODMatrix description, file name D:\\temp\\new_
      &site\\odm\\Matrix.des'
      write(fnum,'(a)') '* Version'
      write(fnum,'(a)') '@3.100000'
      write(fnum,'(a)') '*  Number of vehicle types'
      write(fnum,'(I1.1)') nveh
      write(fnum,'(a)') '*  Vehicle Type Name'
```

```
      write(fnum,'(a)') '@car'
      write(fnum,'(a)')'*    number of vehicle type classes it pertains'
      write(fnum,'(t6,i1.1)') nvehcl
      write(fnum,'(a)') '*    fuel-consumption parameters'
      write(fnum,'(t4,7f7.3)') fuelp
      write(fnum,'(a)') '*    number of pollutants'
      write(fnum,'(t6,i1.1)') npoll
      write(fnum,'(a)') '*      guided vehicles'
      write(fnum,'(t6,f7.3)') nguide
      write(fnum,'(a)') '*      number of driver types'
      write(fnum,'(t8,i1.1)') ndrive
      write(fnum,'(a)') '     100.000'
      write(fnum,'(a)') '*  length (min, max, mean, deviation)'
      write(fnum,'(t2,4f7.3)') length
      write(fnum,'(a)') '*  width (min, max, mean, deviation)'
      write(fnum,'(t2,4f7.3)') width
      write(fnum,'(a)') '*  maximum desired speed (min, max, mean,deviat
     &ion)'
      write(fnum,'(t2,4f8.3)') maxds
      write(fnum,'(a)') '*  maximum acceleration (min, max, mean, deviat
     &ion)'
      write(fnum,'(t2,4f7.3)') maxacc
      write(fnum,'(a)') '*  normal deceleration (min, max, mean, deviati
     &on)'
      write(fnum,'(t2,4f7.3)') nordec
      write(fnum,'(a)') '*  maximum deceleration (min, max, mean, deviat
     &ion)'
      write(fnum,'(t2,4f7.3)') maxdec
      write(fnum,'(a)')'*  speed acceptance (min, max, mean, deviation)'
      write(fnum,'(t2,4f7.3)') speeda
      write(fnum,'(a)')'*  minimum distance between vehicles (min, max,
     &mean, deviation)'
      write(fnum,'(t2,4f7.3)') mindis
      write(fnum,'(a)') '*  give-way time (min, max, mean, deviation)'
      write(fnum,'(t2,4f8.3)') givtim
      write(fnum,'(a)') '*  guidance acceptance (min, max, mean, deviati
     &on)'
      write(fnum,'(t2,4f7.3)') guiacc
      write(fnum,'(a)') '* cruising tolerance'
      write(fnum,'(f7.5)') crutol
      write(fnum,'(a)')'* time begin, time end, nb.fixed intervals'
      write(fnum,'(i4,i4,i4)') start,end,ntime
      do 300 i=1,ntime
      write(fnum,*) delt
  300 continue
      write(fnum,'(a)') '* number of statements'
```

```
      write(fnum,*) (nor*ndes)

      do 310 i=1,nor
      do 320 j=1,ndes
      write(fnum,'(a)')'* statement ------------------------------'
      write(fnum,'(a)')'* idcentroids: origin, dest.  nbvehmods'
      write(fnum,'(i4,i4,i4)') origin(i),destination(j),nveh
      write(fnum,'(a)') '@car'
      write(fnum,'(a)') '* time function type'
      write(fnum,*) '0'
      do 330 k=1,ntime
      write(fnum,*) tt(k,i,j)
330   continue
320   continue
310 continue
end
C
C get - reads an array from the file
C
subroutine get(row,array,fnum)
      character*15    comment
      integer         row,fnum,i
      integer         array(row)

      read(fnum,'(a)') comment
      do 500 i=1,row
      read(fnum,*) array(i)
500 continue
end
```

```
Subroutine to get the off-ramp counts
subroutine retoff(ndes,ntime)
      integer      ntime,ndes,i,j
      integer      offrc(ntime,ndes)
      real         soffrc(ntime,ndes)
      character    ftype*(*),fmttype*200
      parameter    (ftype='f6.1')
```

```
      call getoff(offrc,ntime,ndes)
      do 130 i=1,ntime
      do 135 j=1,ndes
      soffrc(i,j) = 1.0*offrc(i,j)
  135   continue
        call retfmt(ftype,ndes,fmttype)
        write(19,fmt=fmttype) (soffrc(i,j),j=1,ndes)
130 continue
end

subroutine getoff(off,ntime,ndes)
      integer          ntime,ndes,start,end,delta
      integer          off(ntime,ndes)
      integer          i,j,hour,min
      character*(*)  pathname
      parameter      (pathname='c:\\thesis\\progra~1\\169thesis\\odm\\Aim
      &sun\\',start=0,end=180)
      character       fname*12,buffer*80,full*80
      character*32   counts(ndes)

      delta=(end-start)/ntime
      do 100 i=1,ntime
           hour = int(i*delta/60) + int(start/60)
           min = mod((mod(start,60)+delta*i),60)
           write(fname,'(i2.2,a,i2.2,a)') hour,'h',min,'m00.det'
           full = pathname//fname
           open(unit=1,file=full,status='old')
           do 110 j=1,27
           read(1,'(a)') buffer
  110   continue
        do 120 j=1,ndes
          read(1,'(a)') counts(j)
          read(counts(j)(29:),'(bn,i3)') off(i,j)
  120   continue
        close(1)
100 continue
end
```

APPENDIX J – Seed generation program

Main program

```
program seeds
implicit            double precision(a-h,o-z)
integer             nor,ndes,ntime,ndays,nsec
double precision    precision
character*(*)       paramfile,outfile
parameter           (paramfile='169parameter.dat')
parameter           (outfile='169seeds.out')

open(unit=1,file=paramfile,status='old')
call readparam(1,nor,ndes,ntime,ndays,nsec,precision)
close(1)
open(unit=20,file=outfile,status='old')
write(20,*) 'nor =',nor
write(20,*) 'ndes =',ndes
write(20,*) 'ndays =',ndays
write(20,*) 'nsec =',nsec
write(20,*) 'ntime =',ntime
write(20,*) 'precision =',precision
call generate(nor,ndes,ntime,ndays,nsec,precision)
write(20,*) 'exiting pgm'
close(20)
end
```

Associated subroutines

```
subroutine generate(nor,ndes,ntime,ndays,nsec,precision)
    integer             nor,ndes,ntime,ndays,nsec
    double precision    precision
    integer             origin(nor),destination(ndes)
    character*(*)       seed1,seed2,seed3,seed4,seed5,seed6
    parameter           (seed1='169od1.dat',seed2='169od2.dat')
    parameter           (seed3='169od3.dat',seed4='169od4.dat')
    parameter           (seed6='169od6.dat')

    open(unit=2,file=seed1,status='old')
    open(unit=3,file=seed2,status='old')
    open(unit=4,file=seed3,status='old')
    open(unit=5,file=seed4,status='old')
    open(unit=8,file=seed6,status='old')

    call seedone(2,nor,ndes,precision)
    close(2)
    call seedtwo(3,nor,ndes,ntime,ndays,precision)
    close(3)
    call seedthree(4,nor,ndes,ntime,ndays,precision)
```

```
      close(4)
      call seedfour(5,nor,ndes,ntime,ndays,nsec,precision)
      close(5)
      call seedsix(8,nor,ndes,ntime,ndays,nsec,precision)
      close(8)

      write(20,*) 'exiting generate...'
end
subroutine seedone(fnum,nor,ndes,prec)
      integer             nor,ndes,fnum
      integer             origin(nor),destination(ndes)
      integer             i,j,sum(nor)
      double precision    od1(nor,ndes),prec
      character*(*)       olist,dlist
      parameter           (olist='169ori.dat',dlist='169des.dat')

      write(20,*) 'inside seed one'
      open(unit=11,file=olist,status='old')
      open(unit=12,file=dlist,status='old')
      call read1d(11,origin,nor)
      call read1d(12,destination,ndes)
      close(11)
      close(12)

      do 105 i=1,nor
      sum(i) = 0
      do 110 j=1,ndes
if (origin(i).lt.destination(j)) then
                                                sum(i) = sum(i) + 1
end if
110    continue
105 continue

          do 115 i=1,nor
do 120 j=1,ndes
          if (origin(i).lt.destination(j)) then
      od1(i,j) = 1.0d+0/sum(i)
      call precise(od1(i,j),prec,od1(i,j))
else
                                                od1(i,j) = -1.0d+0
end if
120    continue
115 continue

      call checkOD(od1,nor,ndes)
      call writeod(fnum,od1,nor,ndes)
```

```
      write(20,*) 'exiting seed one'
end
subroutine seedtwo(fnum,nor,ndes,ntime,ndays,prec)
      integer                nor,ndes,ntime,ndays,fnum
      double precision    od2(nor,ndes),prec
      integer                origin(nor),destination(ndes)
      double precision    offramp(ndays,ntime,ndes)
      double precision    sumo(nor),sumd(ndes)
      character*(*)        olist,dlist,offile
      parameter             (olist='169ori.dat',dlist='169des.dat')
      parameter             (offile='169offramp.dat')


      write(20,*) 'inside seed two'
      open(unit=21,file=olist,status='old')
      open(unit=22,file=dlist,status='old')
      open(unit=23,file=offile,status='old')

      call read1d(21,origin,nor)
      call read1d(22,destination,ndes)
      call read3d(23,offramp,ndays,ntime,ndes)

      close(21)
      close(22)
      close(23)

      do 200 i=1,ndes
      sumd(i) = 0.0d+0
      do 210 j=1,ndays
      do 220 k=1,ntime
      sumd(i) = sumd(i) + offramp(j,k,i)
  220       continue
  210    continue
200 continue
      do 230 i=1,nor
      sumo(i) = 0.0d+0
      do 240 j=1,ndes
if (origin(i).le.destination(j)) then
                                                sumo(i) = sumo(i) + sumd(j)
end if
240    continue
230 continue

          do 250 i=1,nor
do 260 j=1,ndes
```

```
          if (origin(i).lt.destination(j)) then
      od2(i,j) = 1.0d+0*sumd(j)/sumo(i)
      call precise(od2(i,j),prec,od2(i,j))
else
                                                od2(i,j) = -1.0d+0
end if
260    continue
250 continue

      call checkOD(od2,nor,ndes)
      call writeod(fnum,od2,nor,ndes)
      write(20,*) 'exiting seed two'
end
subroutine seedthree(fnum,nor,ndes,ntime,ndays,prec)
      integer              fnum,nor,ndes,ntime,ndays
      integer              origin(nor),destination(ndes)
      double precision  onramp(ndays,ntime,nor),offramp(ndays,ntime,ndes)
      double precision  od31(nor,ndes),od32(nor,ndes),od33(nor,ndes)
      double precision  diffod(nor,ndes),maxdiff,tol,sumtij
      double precision  prod(nor),att(ndes),prec
      integer              i,j,k,iter
      character*(*)      olist,dlist,onfile,offile
      parameter          (olist='169ori.dat',dlist='169des.dat')
      parameter          (onfile='169onramp.dat',offile='169offramp.dat')


      write(20,*) 'inside seed three'
      iter = 0
      open(unit=31,file=olist,status='old')
      open(unit=32,file=dlist,status='old')
      open(unit=33,file=onfile,status='old')
      open(unit=34,file=offile,status='old')

      call read1d(31,origin,nor)
      call read1d(32,destination,ndes)
      call read3d(33,onramp,ndays,ntime,nor)
      call read3d(34,offramp,ndays,ntime,ndes)

      close(31)
      close(32)
      close(33)
      close(34)

      tol = 1.0d+0
      maxdiff = 100.0d+0
      call initialize(nor,ndes,origin,destination,od31)
```

```
      call getsums(nor,ndes,ndays,ntime,onramp,offramp,prod,att)

      write(20,*) 'prod is...'
      write(20,*) (prod(i),i=1,nor)
      write(20,*) 'att is...'
      write(20,*) (att(i),i=1,ndes)

340 if (maxdiff.gt.tol) then
          do 345 i=1,nor
do 350 j=1,ndes
      sumtij = 0.0d+0
      do 355 k=1,ndes
      sumtij = sumtij + od31(i,k)
 355          continue
            od32(i,j) = prod(i)*od31(i,j)/sumtij
 350       continue
 345     continue

         do 360 i=1,nor
           do 365 j=1,ndes
             sumtij = 0.0d+0
             do 370 k=1,nor
               sumtij = sumtij + od32(k,j)
 370           continue
             od33(i,j) = att(j)*od32(i,j)/sumtij
 365       continue
 360     continue

         maxdiff = 0.0d+0
         do 375 i=1,nor
           do 380 j=1,ndes
             diffod(i,j) = abs(od33(i,j)-od31(i,j))
             if (maxdiff.le.diffod(i,j)) then
      maxdiff = diffod(i,j)
      end if
 380       continue
 375     continue

         do 385 i=1,nor
           do 390 j=1,ndes
             od31(i,j) = od33(i,j)
 390       continue
 385     continue

         iter = iter +1
```

```
        write(20,*) 'iter =',iter
        write(20,*) 'post maxdiff =',maxdiff

        goto 340
      end if
      call convert(nor,ndes,od31,prec)
      call checkOD(od31,nor,ndes)
      call writeod(fnum,od31,nor,ndes)
      write(20,*) 'exiting seed three...'
      end

subroutine initialize(nor,ndes,origin,destination,od)

      integer         nor,ndes
      integer         origin(nor),destination(ndes)
      double precision od(nor,ndes)
      integer         i,j

          do 300 i=1,nor
do 301 j=1,ndes
                                                  if (origin(i).lt.destination(j)) then
od(i,j) = 1.0d+0
                                                  else
od(i,j) = 0.0d+0
                                                  end if
301    continue
300 continue
end

subroutine getsums(nor,ndes,ndays,ntime,onramp,offramp,prod,att)

      integer              nor,ndes,ntime,ndays
      double precision     onramp(ndays,ntime,nor),offramp(ndays,ntime,ndes)
      double precision     prod(nor),att(ndes)
      integer              i,j,k
      double precision     orisum,dessum,sumdiff

      orisum = 0.0d+0
      dessum = 0.0d+0

          do 310 i=1,ndays
do 311 j=1,ntime
          do 312 k=1,nor
      prod(k) = prod(k) + onramp(i,j,k)
      orisum = orisum + onramp(i,j,k)
  312      continue
  311    continue
310 continue
```

```
          do 313 i=1,ndays
do 314 j=1,ntime
          do 315 k=1,ndes
     att(k) = att(k) + offramp(i,j,k)
     dessum = dessum + offramp(i,j,k)
 315      continue
 314    continue
313 continue

     write(20,*) 'inside getsums'
     write(20,*) 'sumo =',orisum,'sumd =',dessum

sumdiff = orisum - dessum


     if (sumdiff.ne.0.0d+0) then
       do 316 i=1,ndes
          att(i) = att(i)*(1.0d+0 + sumdiff/dessum)
 316    continue
     end if

     write(20,*) 'exiting getsums...'

end
subroutine convert(nor,ndes,od,prec)
     integer              nor,ndes
     double precision   od(nor,ndes),sum(nor),prec

     do 321 i=1,nor
     sum(i) = 0.0d+0
     do 322 j=1,ndes
     sum(i) = sum(i) + od(i,j)

322    continue
321 continue

     do 323 i=1,nor

     do 324 j=1,ndes

     od(i,j) = od(i,j)/sum(i)
     call precise(od(i,j),prec,od(i,j))
324    continue
323 continue

          do 325 i=1,nor
do 326 j=1,ndes
                                        if (od(i,j).eq.0.0d+0) then
od(i,j) = -1.0d+0
                                        end if
```

```
326    continue
325 continue
end

subroutine seedfour(fnum,nor,ndes,ntime,ndays,nsec,prec)

      integer               nor,ndes,ntime,ndays,nsec,fnum
      double precision  od4(nor,ndes),prec
      double precision  factor(nor,ndes),distance(nor,ndes),avgtrip
      integer               i,j,k

      write(20,*) 'inside fourth'
      call calavg(ndays,ntime,nsec,nor,avgtrip)
      write(20,*) 'The avg trip length =',avgtrip

      call caldistance(nor,ndes,nsec,distance)
      write(20,*) 'The distance matrix'
      call writeod(20,distance,nor,ndes)

      call calfactor(nor,ndes,avgtrip,distance,factor)
      write(20,*) 'The factor matrix'
      call writeod(20,factor,nor,ndes)

      call calod(nor,ndes,factor,od4,prec)
      call checkOD(od4,nor,ndes)
      call writeod(fnum,od4,nor,ndes)
end
subroutine calod(nor,ndes,factor,odm,prec)
      integer               nor,ndes
      double precision  factor(nor,ndes),odm(nor,ndes),sum(nor),prec
      integer               i,j
      do 410 i=1,nor
           sum(i) = 0.0d+0
           do 415 j=1,ndes
           odm(i,j)=0.0d+0
           sum(i) = sum(i) + factor(i,j)
415   continue
410 continue

      do 420 i=1,nor

      do 425 j=1,ndes

      odm(i,j) = factor(i,j)/sum(i)
      call precise(odm(i,j),prec,odm(i,j))
425    continue
420 continue
```

```
        do 430 i=1,nor
do 435 j=1,ndes
                                                    if (odm(i,j).eq.0.0d+0) then
odm(i,j) = -1.0d+0
                                                    end if
435    continue
430 continue
end

subroutine calavg(ndays,ntime,nsec,nor,avgtrip)
    integer             ndays,ntime,nsec,nor
    double precision    mainline(ndays,ntime,nsec),section(nsec,2)
    double precision    onrc(ndays,ntime,nor)
    double precision    avgtrip,sumtrip,sumppl
    integer             i,j,k,m
    character*(*)       mainfile,onfile,secfile
    parameter           (mainfile='169main.dat',onfile='169onramp.dat')
    parameter           (secfile='169sec.dat')


    open(unit=41,file=onfile,status='old')
    open(unit=42,file=mainfile,status='old')
    open(unit=43,file=secfile,status='old')
    call read3d(41,onrc,ndays,ntime,nor)
    call read3d(42,mainline,ndays,ntime,nsec)
    call read2d(43,section,nsec,2)
    close(41)
    close(42)
    close(43)

    avgtrip = 0.0d+0
    sumtrip = 0.0d+0
    sumppl = 0.0d+0
    do 450 i=1,ndays
        do 451 j=1,ntime
        do 452 k=1,nsec
        sumtrip = sumtrip+mainline(i,j,k)*section(k,2)
452     continue
      do 453 m=1,nor
       sumppl = sumppl + onrc(i,j,m)
453     continue
    451  continue
    450 continue
avgtrip = sumtrip/sumppl
end
```

```
subroutine caldistance(nor,ndes,nsec,distance)
      integer              nor,ndes,nsec
      double precision  section(nsec,2),distance(nor,ndes)
      integer             origin(nor),destination(ndes)
      integer             i,j,k
      character*(*)     ofile,dfile,secfile
      parameter         (ofile='169origin.dat',dfile='169destination.dat')
      parameter         (secfile='169sec.dat')

      open(unit=30,file=ofile,status='old')
      open(unit=31,file=dfile,status='old')
      open(unit=32,file=secfile,status='old')
      call read1d(30,origin,nor)
      call read1d(31,destination,ndes)
      call read2d(32,section,nsec,2)
      close(30)
      close(31)
      close(32)

      write(20,*) 'origin file'
      write(20,*) (origin(i),i=1,nor)
      write(20,*) 'destination file'
      write(20,*) (destination(i),i=1,ndes)


      do 460 i=1,nor
      do 461 j=1,ndes
      distance(i,j) = 0.0d+0
      if (origin(i).le.destination(j)) then
do 462 k=origin(i),destination(j)
                                          distance(i,j) = distance(i,j)+section(k,2)
462         continue
          end if
461  continue
460 continue

      write(20,*) 'section parameters'
      write(20,*) (section(k,2),k=1,nsec)
end
subroutine calfactor(nor,ndes,avgtrip,distance,factor)
      integer              nor,ndes
      double precision     term1,term2,term3,avgtrip
      double precision     factor(nor,ndes)
      real                 alpha,beta
      double precision     distance(nor,ndes)
```

```
      alpha = 3.0
      beta = alpha/avgtrip
      term1 = exp(alpha*log(beta))/exp(gammln(alpha))
      write(20,'(a,e12.6)') 'term1 = ', term1
      do 470 i=1,nor
      do 471 j=1,ndes
          term2 = exp((alpha-1)*log(real(distance(i,j))))
          term3 = exp(real(distance(i,j))*(-1.0)*beta)
          factor(i,j) = term1*term2*term3
471    continue
470 continue
end
```

Gammln – subroutine taken from Numerical Recipes in Fortran77.

```
subroutine seedsix(fnum,nor,ndes,ntime,ndays,nsec,prec)

      integer              nor,ndes,nsec,ntime,ndays,fnum
      double precision     onramp(ndays,ntime,nor),prec
      double precision     mainline(ndays,ntime,nsec)
      double precision     offramp(ndays,ntime,ndes)
      double precision     turnper(ndays,ntime,nsec),od6(nor,ndes)
      double precision     avgtp(nsec), sum, avgonramp(nor),tempavg(nor)
      integer              origin(nor),destination(ndes)
      character*(*)        onfile,offile,mainfile,olist,dlist
      parameter            (olist='169origin.dat',dlist='169destination.dat')
      parameter            (onfile='169onramp.dat',offile='169offramp.dat')
      parameter            (mainfile='169main.dat')
      integer              i,j,k,secid


      write(20,*) 'inside seed six'
      open(unit=61,file=onfile,status='old')
      open(unit=62,file=offile,status='old')
      open(unit=63,file=mainfile,status='old')
      open(unit=64,file=olist,status='old')
      open(unit=65,file=dlist,status='old')

      call read3d(61,onramp,ndays,ntime,nor)
      call read3d(62,offramp,ndays,ntime,ndes)
      call read3d(63,mainline,ndays,ntime,nsec)
      call read1d(64,origin,nor)
      call read1d(65,destination,ndes)

      close(61)
      close(62)
      close(63)
      close(64)
```

```
      close(65)

      do 603 i=1,ndays
      do 605 j=1,ntime
      do 610 k=1,nsec
      turnper(i,j,k) = 0.0d+0
610   continue
605   continue
603 continue

      do 612 i=1,ndays
      do 615 j=1,ntime
      do 620 k=1,ndes
      secid = destination(k)
      turnper(i,j,secid) = 1.0d+0*offramp(i,j,k)/mainline(i,j,secid)
620   continue
615   continue
612 continue

      do 621 k=1,nsec
      sum = 0.0d+0
      do 622 i=1,ndays
      do 623 j=1,ntime
      sum = sum + turnper(i,j,k)
623    continue
622   continue
          avgtp(k) = sum/(ntime*ndays)
621 continue
          avgtp(nsec) = 1.0d+0
      write(20,*) 'the tp'
      write(20,*) (avgtp(i),i=1,nsec)
      do 624 i=1,nor
      sum=0.0d+0
      do 625 j=1,ndays
      do 626 k=1,ntime
      sum = sum + onramp(j,k,i)
  626       continue
  625    continue
        avgonramp(i) = sum/(ntime*ndays)
        tempavg(i) = avgonramp(i)
624 continue
          do 630 i=1,nor
do 640 j=1,ndes
          if (origin(i).le.destination(j)) then
      od6(i,j) = avgtp(destination(j))*tempavg(i)
```

```
    tempavg(i) = tempavg(i) - od6(i,j)
else
                                                od6(i,j) = -1.0d+0
end if
640    continue
630 continue

    do 635 i=1,nor
    do 636 j=1,ndes
    sum = 0.0d+0
    if (od6(i,j).ne.-1.0d+0) then
    od6(i,j) = od6(i,j)/avgonramp(i)
    call precise(od6(i,j),prec,od6(i,j))
    end if
636  continue
635 continue

    call checkOD(od6,nor,ndes)
    call writeod(fnum,od6,nor,ndes)
    write(20,*) 'exiting seed6'
end

subroutine readparam(fnum,nor,ndes,ntime,ndays,nsec,precision)
    integer              nor,ndes,ntime,ndays,nsec,fnum
    double precision   precision
    character*80       comment

    read(fnum,*) comment
    read(fnum,*) comment
    read(fnum,*) nor
    read(fnum,*) ndes
    read(fnum,*) ntime
    read(fnum,*) ndays
    read(fnum,*) nsec
    read(fnum,*) precision
end
subroutine read1d(fnum,array,row)
    integer           fnum,row,i
    integer           array(row)
    character*80   comment

    read(fnum,*) comment
    do 21 i=1,row
    read(fnum,*) array(i)
21  continue
```

```
end

subroutine read3d(fnum,matrix,ht,row,col)

      integer              fnum,row,ht,col,i,j,k
      double precision  matrix(ht,row,col)
      character*10       comment

      do 30 i=1,ht
      read(fnum,*) comment
      do 31 j=1,row
      read(fnum,*) (matrix(i,j,k),k=1,col)

31   continue
30  continue
end

subroutine writeod(fnum,od,nor,ndes)

      integer              fnum,nor,ndes
      double precision  od(nor,ndes)
      integer               i,j

      do 40 i=1,nor

      write(fnum,*) (od(i,j),j=1,ndes)

40  continue
end

subroutine write1d(fnum,array,row)

      integer          fnum,row,i
      integer          array(row)
      character*80  comment
      do 50 i=1,row
      write(fnum,*) array(i)

50  continue
end

subroutine write3d(fnum,matrix,ht,row,col)

      double precision  fnum,row,ht,col,i,j,k
      integer               matrix(ht,row,col)

      do 60 i=1,ht
      write(fnum,*) 'ht =',ht
      do 61 j=1,row
      write(fnum,*) (matrix(i,j,k),k=1,col)

61    continue
```

```
60  continue
end

subroutine read2d(fnum,matrix,row,col)
      integer               fnum,row,col,i,j,k
      double precision  matrix(row,col)
      character*10        comment

           read(fnum,*) comment
do 62 j=1,row
                                                  read(fnum,*) (matrix(j,k),k=1,col)
62   continue
end

subroutine precise(num,prec,precnum)
      double precision  num,prec,precnum
      precnum = num - mod(num,prec)
end
subroutine checkOD(od,nor,ndes)
      integer               nor,ndes,i,j
      double precision   od(nor,ndes)
      double precision   sum(nor)

      do 70 i=1,nor
      sum(i) = 0.0d+0
      do 80 j=1,ndes-1
if (od(i,j).gt.0.0d+0) then
                                                  sum(i) = sum(i)+od(i,j)
end if
  80    continue
        od(i,ndes) = 1.0d+0 - sum(i)
      70 continue
end
```