UNIVERSITY OF MINNESOTA

CENTER FOR TRANSPORTATION STUDIES

INTELLIGENT TRANSPORTATION SYSTEMS INSTITUTE

Improving the Estimation of Travel Demand for Traffic Simulations: Part II

Final Report

Prepared by
Yao Wu
Gary Davis
David Levinson

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 II	December 2004
	6.

7. Author(s)	8. Performing Organization Report No.
Yao Wu, Gary Davis, David Levinson	

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
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

www.cts.umn.edu/pdf/CTS-04-11.pdf

16. Abstract (Limit: 200 words)

This report examined several methods for estimating Origin-Destination (OD) matrices for freeways using loop detector data. Least squares based methods were compared in terms of both off-line and on-line estimation. Simulated data and observed data were used for evaluating the static and recursive estimators. For off-line estimation, four fully constrained least squares methods were compared. The results showed that the variations of a constrained least squares approach produced more efficient estimates. For on-line estimation, two recursive least squares algorithms were examined. The first method extends Kalman Filtering to satisfy the natural constraints of the OD split parameters. The second was developed from sequential quadratic programming. These algorithms showed different capabilities to capture an abrupt change in the split parameters. Practical recommendations of the choice of different algorithms are given.

17. Document Analysis/Descriptors		18.Availability Statement
Algorithms Travel demand Traffic simulation	Origin-Destination (OD) Loop detector	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	95	

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

Final Report

Prepared by:
Yao Wu
Gary Davis
David Levinson
Department of Civil Engineering
University of Minnesota

December 2004

Center for Transportation Studies
University of Minnesota

CTS 04-11

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Executive Summary

This report examined several methods for estimating Origin-Destination (OD) matrices for freeways using loop detector data. Least squares based methods were compared in terms of both off-line and on-line estimation. Simulated data and observed data were used for evaluating the static and recursive estimators. For off-line estimation, four fully constrained least squares methods were compared. The results showed that the variations of a constrained least squares approach produced more efficient estimates. For on-line estimation, two recursive least squares algorithms were examined. The first method extends Kalman Filtering to satisfy the natural constraints of the OD split parameters. The second was developed from sequential quadratic programming. These algorithms showed different capabilities to capture an abrupt change in the split parameters. Practical recommendations of the choice of different algorithms are given.

CHAPTER 1. INTRODUCTION

Traffic congestion is an increasingly serious problem for many of the world's urban areas. In the United States, the number of automobiles and light trucks grew by 86% between 1970 and 1995, while the amount of passenger miles traveled grew by 49%, both of which contributed to worsening congestion on metropolitan freeways (Bureau of Transportation Statistics, 1997). The strategies for congestion relief generally fall under two classifications: supply strategies and demand strategies.

Supply strategies add to the system capacity, including the development of new or expanded infrastructure, so that the demand is better satisfied or the efficiency of the existing system is improved. Demand strategies aim to reduce or at least redistribute the travel demand. The measures in the category include increasing taxes and other transportation expenses such as congestion pricing, parking pricing, and promoting car-pooling and mass transit.

Intelligent Transportation Systems (ITS), which include Advanced Traffic Management Systems (ATMS), Advanced Traveler Information Systems (ATIS), and Automated Vehicle Control Systems (AVCS) are designed to make more efficient use of existing highway capacity by managing and controlling traffic flow with real-time traffic information. ITS strategies provide both supply and demand measures. The supply type of ITS include optimized signal operation using real-time measures of demand, incident detection and resolution, freeway management with ramp metering, and accident avoidance with variable message signs warning of upcoming conditions. Demand-type ITS measures include the provision of real-time traffic congestion information to support informed individual travel decisions (Papacostas and Prevedouros, 2001).

The successful implementation of ITS strategies not only depends on the availability of high-quality real-time information about traffic conditions, but also on prediction models in order to anticipate the response to the proposed traffic management actions. These practical models should be able to describe the interaction between travel demand and traffic flow phenomena. Since most traffic models use an origin-destination (O-D) matrix as the basic description of travel demand, it is necessary to generate estimates of OD matrices. Travel demand estimation is an essential input for all traffic management plans. Especially under emergency conditions such as accidents, travel demands are indispensable information for deciding how to re-route traffic. The advantage of using an OD matrix is that it not only indicates travel demand but also provides information about the direction of demand. Accurate and fast O-D trip table generation techniques are needed to implement on-line control strategies of ITS.

Traditional methods of acquiring OD matrices include license plate studies and a combination of home interview and roadside surveys. However, because these approaches are expensive, time-consuming, and labor demanding, they are excluded from wide-range application. Another disadvantage of traditional methods is that the obtained OD matrix is not dynamic and thus cannot be updated over time. This is not a desirable feature for on-line control applications.

In the recent two decades, because of the availability of the data collected by traffic surveillance systems, more attention has been given to estimating OD matrices from traffic counts that are readily available. It is hoped that the availability of time-series data of traffic counts will produce OD estimators that have desirable statistical

properties such as consistency, efficiency, and the ability to track changes in the OD patterns.

For general networks, because multiple routes connect each OD pair, the problem of OD estimation can be complicated. However, the problem is simplified when one considers linear networks, such as single intersections and freeway segments, where each origin and destination are connected by, at most, one route. Since urban freeways carry a large fraction of total urban travel, it is not surprising that estimation of freeway OD patterns has been receiving increased attention. The OD pattern can be inferred from the available time-series data of on-ramp, off-ramp, and mainline traffic counts (Yu and Davis, 1994).

The rest of the report is organized as follows. Chapter 2 provides a literature review of the existing OD estimation methods, followed by a description of a simulation and optimization method and its initial application on a simple network. Chapter 3 compares the performances of four least squares-based off-line estimation algorithms. Chapter 4 compares two on-line estimation methods. Chapter 5 then draws the conclusions and makes practical recommendations.

CHAPTER 2. Estimating OD matrices from traffic counts

2.1 Introduction

An Origin Destination (OD) matrix is a two dimensional array of elements whose values represent the travel demand between each given origin and destination. Travelers select routes connecting an origin to a destination and when they travel these routes, traffic volumes are generated on links of the network, which can be measured by detectors. In principle, traffic counts provide information about the underlying OD matrix that generated them, and it should be possible to estimate the OD matrix from a suitably rich set of counts.

The origins and destinations have somewhat different meanings under different scenarios. For a general network such as an urban area, it is usual to subdivide it into relatively homogeneous zones, using socioeconomic data and land use information. Therefore, for an urban area, the origins and destinations of an OD matrix are the traffic analysis zones. However, multiple routes exist from an origin zone to a destination zone and travelers have to choose one route connecting the origin and the destination. In the estimation procedure, this route choice has to be explicitly modeled, which complicates the problem. However, for a simple linear network, such as a freeway segment, the origins and destinations are the on-ramps and off-ramps respectively, and there is only one route connecting each OD pair. Therefore, the estimation of OD matrices for freeways is an especially simple case of the estimation for general networks. In this report, we focus on the estimation of OD matrices for freeways.

2.2 Categorization

Depending on the data availability, the existing OD estimation methods generally fall into two categories: under-specified and over-specified approaches.

2.2.1 Under-specification

This type of approach is most common when only one set of traffic counts is available. A simple network with a single set of observed link volumes is illustrated in Figure 2.1, in which zones 1 and 2 are origins and zones 3 and 4 are destinations. The numbers on the links represent traffic counts. As shown in Table 2.1, either of the two matrices can reproduce the observed link volumes, so the traffic volumes do not uniquely determine the OD matrix.

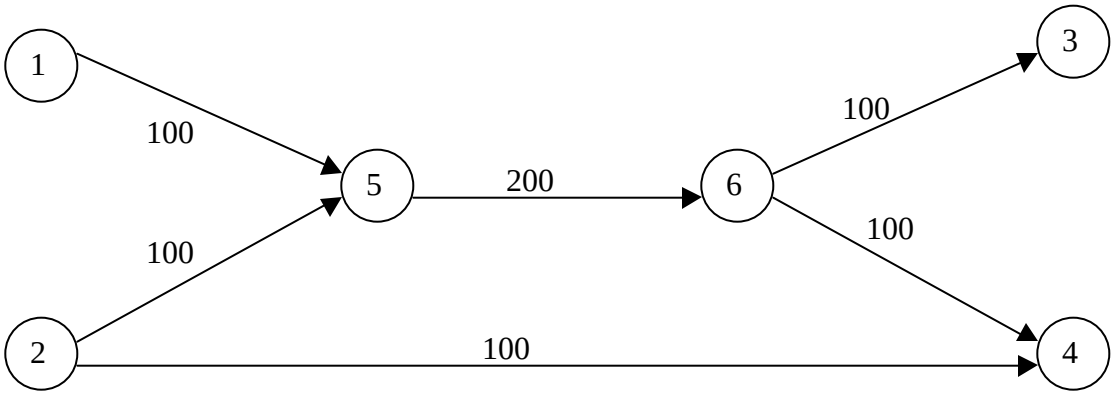


Figure 2.1 A simple network

O\D	3	4		O\D	3	4
1	0	100		1	50	50
2	100	100		2	50	150

Table 2.1 Two OD matrices

The explanation for more than one solution for this network is that the number of OD variables, which is four, is greater than the number of independent constraints, which is only three, as shown in (2.1), where x_{ij} represents the traffic volume from origin i to destination j. Therefore, the system is under-specified and there are actually many OD matrices that can exactly reproduce the observed traffic counts.

$$\begin{cases} x_{13} + x_{14} = 100 \\ x_{23} + x_{24} = 200 \\ x_{13} + x_{23} = 100 \end{cases} \tag{2.1}$$

For real networks such as the Twin Cities seven-county metropolitan area, similar problems exist. The number of Transportation Analysis Zones (TAZ) is 1165. Therefore, the dimension of the OD matrices to be estimated is 1165x1165, which is approximately 1.36 million. Unfortunately, the number of links is much smaller, of the order 10^4. A single set of traffic counts simply cannot produce a unique solution so that methodologies in this category need additional assumptions to find a solution. Generally, this involves first selecting a prior estimate for the OD matrix and then selecting as an estimate the matrix that reproduces the traffic counts while being closest in some sense to the prior estimate.

2.2.2 Over-specification

When time-series of traffic counts are available from automatic surveillance and control systems, we can have constraints for each time slice. If we assume that the OD parameters are strictly constant, then we can write equations such as (2.1) for each time interval. When the number of equations is larger than the number of parameters, the problem becomes over-determined, and generally no single matrix will exactly reproduce the traffic counts. However, a unique solution can be obtained by choosing the one that most closely matches the observations.

2.3 Review of Methodologies

2.3.1 Under-specification

The estimation of OD matrices from traffic counts dates back to the 1970s. The earliest methodologies use traffic counts from a single observation period as the basic information. However, as discussed above, since these counts are not sufficient to determine the matrix of OD flows, additional assumptions and *a priori* information are needed to lead to a unique solution. Willis and May (1981) give a review of these methods.

Gur and Turnquist (1979) formulated OD estimation as a nonlinear programming problem by trying to minimize the system travel time subject to constraints that observed travel times and link volumes correspond to those consistent with Wardrop's second principle. An iterative algorithm gave a unique solution for this problem, given an initial OD matrix and the travel time function for each link. However, in the absence of an efficient cooperation mechanism and the symmetric information among drivers, the assumption of the network equilibrium was dubious. In addition, the dependence on the initial trip matrix may lead to poor estimates if the initial matrix was poor.

Van Zuylen and Willumsen (1979) developed two procedures that are representative of the under-specification methods: information minimization and entropy maximization. The underlying rationale for the information minimization method is that one should select a matrix by adding as little information as possible when the available information from traffic counts is insufficient to determine the OD matrix. The entropy maximization method is based on the assumption that the most likely OD trip matrix has the greatest number of associated micro-states. For example, the number of ways to choose an OD matrix X_{ij} with a total number of trips N is:

$$W\{X_{ij}\} = \frac{N!}{\prod_{ij} X_{ij}} \tag{2.2}$$

and the maximum entropy estimate maximizes W. Although the result of this method has the same multi-proportional form as that of the information minimization method, the entropy-maximization demanded less computational effort and appeared to produce

estimates closer to the observed matrix. Instead of forcing the OD matrix to follow the gravity pattern as in Gur *et al.* (1978), these two methodologies make full use of the information contained in the traffic counts. They are particularly useful because no travel behavior assumption is made in these approaches. However, the need for an additional "prior" matrix to obtain reasonable results makes it necessary to collect a considerable amount of data besides traffic counts. Furthermore, since the prior information tended to dominate the results, the accuracy of the estimate depends on the choice of the initial solution.

Maher (1983) suggested a method based on Bayesian statistical inference. Instead of starting with a point estimate, he introduced a distribution over possible initial estimates, in order to represent the degree belief in these prior possibilities. A posterior distribution over the possibilities was then produced from the prior distribution and observations using Bayes Theorem. Although this proposed method allows flexibility in the degree of belief on the prior estimate, this value still needs to be chosen in practice. In addition, the assumption of the multivariate normal distribution only holds when the traffic volume is large enough, so that application to low-volume networks is limited.

Cascetta (1984) developed a generalized least squares estimator, or Atiken estimator of the OD matrix from "director or model" estimators and traffic counts. The director or model estimates here are essentially the same as the initial estimates in the above methods, and the estimator minimizes the distance to the starting estimates. Similar to the Bayes estimator proposed by Maher (1983), a dispersion matrix for the initial estimate should be identified prior to the estimation, which determines the accuracy of obtained estimates. However, the difference is that the Atiken estimator does not require distribution assumptions. Hendrickson and McNeil (1984) described a similar least squares estimator, but the dependence on initial estimates remained a problem.

2.3.2 Over-specified

In this section, the emphasis will be on static methods that are used for simple linear networks, such as intersections and freeways. The first publication on this subject was by Cremer and Keller (1983), and they used split parameters to represent an OD matrix. As shown in (2.3), the split parameter b_{ij} is the probability that a vehicle entering at origin i is destined for destination j. The linear traffic assignment model they proposed was originally applied to intersections. However, it can also be applied to freeway segments if the travel time for an OD pair ij is short compared to the duration of the counting interval. The predicted off-ramp count can be produced as:

$$\hat{y}_j(t) = \sum_i q_i(t) * b_{ij} \tag{2.3}$$

where $\hat{y}_j(t)$ is the predicted traffic count at off-ramp j during time interval t, j=1, . . . ,n;

$q_i(t)$ is the observed traffic count at on-ramp i during time interval t, i=1, . . . , m;

b_{ij} is the probability that a vehicle entering at i is destined for exit j.

The split parameters b_{ij} are also subject to the inequality and equality constraints.

$$0 \leq b_{ij} \leq 1 \text{ for all i, j} \tag{2.4a}$$

$$\sum_j b_{ij} = 1 \text{ for all i} \tag{2.4b}$$

A wide range of estimation techniques has been employed to solve this problem, such as parameter optimization techniques like least squares and constrained optimization and statistically based techniques like maximum likelihood estimation. Constrained Least Squares was suggested by Cremer and Keller (1983) and Nihan and Davis (1987). To illustrate this methodology, a freeway system is used as an example. It is assumed that OD variables are time-invariant. Because traffic counts for multiple time slices are available, the number of equations is greater than the number of variables. As a result, the problem is identified as over-specified. CLS estimates can be obtained by minimizing the sum of squared errors (2.5) subject to the constraints in equations (2.4a) and (2.4b).

$$f = \sum_t \sum_j [\hat{y}_j(t) - y_j(t)]^2 \tag{2.5}$$

The advantage of this method is that it leads to a unique solution that does not depend on the initial solution. This is a great improvement over the under-determined methodologies. However, it is not ideal for the estimation of OD matrices. Constrained least squares estimation originated from estimating turning proportions at an intersection, where an OD variable is equivalent to a turning proportion. In contrast, travel times between freeway origins and destinations vary both as functions of the distance and intervening traffic conditions, and can span several time intervals. As a result, off-ramp counts are always a mixture of the on-ramp counts from different time slices because of the platoon-dispersion effects.

Bell (1991) considered platoon dispersion effects. For a freeway, the exit volume is a mixture of the entry volumes because of the platoon dispersion effect. If all the vehicles can travel through the freeway in k intervals, then it is assumed that there are k OD matrices according to the k time intervals. For example, if the fastest vehicle reaches the exit within 1 interval and the slowest vehicle reaches the exit within k intervals, the predicted off-ramp count can be specified as:

$$\hat{y}_j = \sum_i b_{1_{ij}} q_i(t) + b_{2_{ij}} q_i(t-1) + \cdots + b_{k_{ij}} q_i(t+1-k) \tag{2.6}$$

$$0 \leq b_{k_{ij}} \leq 1 \tag{2.7a}$$

$$\sum_j \sum_k b_{k_{ij}} = 1 \qquad (2.7b)$$

where b_1, b_2, . . . , b_k are the split parameters for time interval t, t-1 and, . . . , t-k respectively.

The objective function we want to minimize is the same as that in the constrained least squares method except a discounting factor is taken into consideration. The parameters to be estimated are k OD matrices with the constraints specified in equations (2.7a) and (2.7b). Instead of estimating mn variables, Bell's method estimates k*mn variables.

2.4 Simulation and Optimization Method

2.4.1 Introduction

In practice, we may need to take into consideration travel time lags between OD pairs on freeway segments. One possible solution is to replace the simple linear traffic assignment model with a traffic flow simulator. A simulation and optimization method was initially proposed by Yu and Davis (1994). This methodology contained two main components: the simulator and the optimization routine. The simulator replaced the linear traffic model (2.3), and the optimization routine was designed to find an optimal solution that minimizes the objective function in equation (2.5). The simulator used in Yu and Davis (1994) was the Stochastic Macroscopic Simulator (STOMAC) and the optimization routine was a quasi-Newton optimization routine. In every iteration of the optimization routine, STOMAC generated traffic counts that were used to calculate the sum of squared errors. The optimization routine then changed the estimates so as to reduce the objective function value. This process ran iteratively until a convergence criterion was met. The performance of this Nonlinear Least Squares (NLS) method was compared with the other three methodologies based on the linear model (Ordinary Least Squares, Expectation-Maximization, and Constrained Approximate Maximum Likelihood) and it was concluded that NLS was the best choice.

2.4.2 Methodology

In this study, the simulation and optimization methodology was implemented as a first try. The focus is on estimating the OD split parameters for the freeway segments. The simulator employed was AIMSUN, and the optimization routine was the Neld-Mead Method.

AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network) is a microscopic simulator that can deal with different traffic networks. The behavior of each vehicle in the network is continuously modeled in the simulation period according to a car-following and lane-changing model. The simulator serves as a complicated traffic model, which can be used to predict the off-ramp counts, given a set

of on-ramp counts and an OD matrix. The objective function (2.8) is the weighted sum of the squared difference between the actual and simulated off-ramp counts. The weight assigned to an off-ramp is the inverse of the standard deviation of the observed counts of that off-ramp. Therefore, the off-ramps that have less traffic volumes are assigned larger weights.

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n} w_i (\hat{y}_{ij} - y_{ij})^2$$

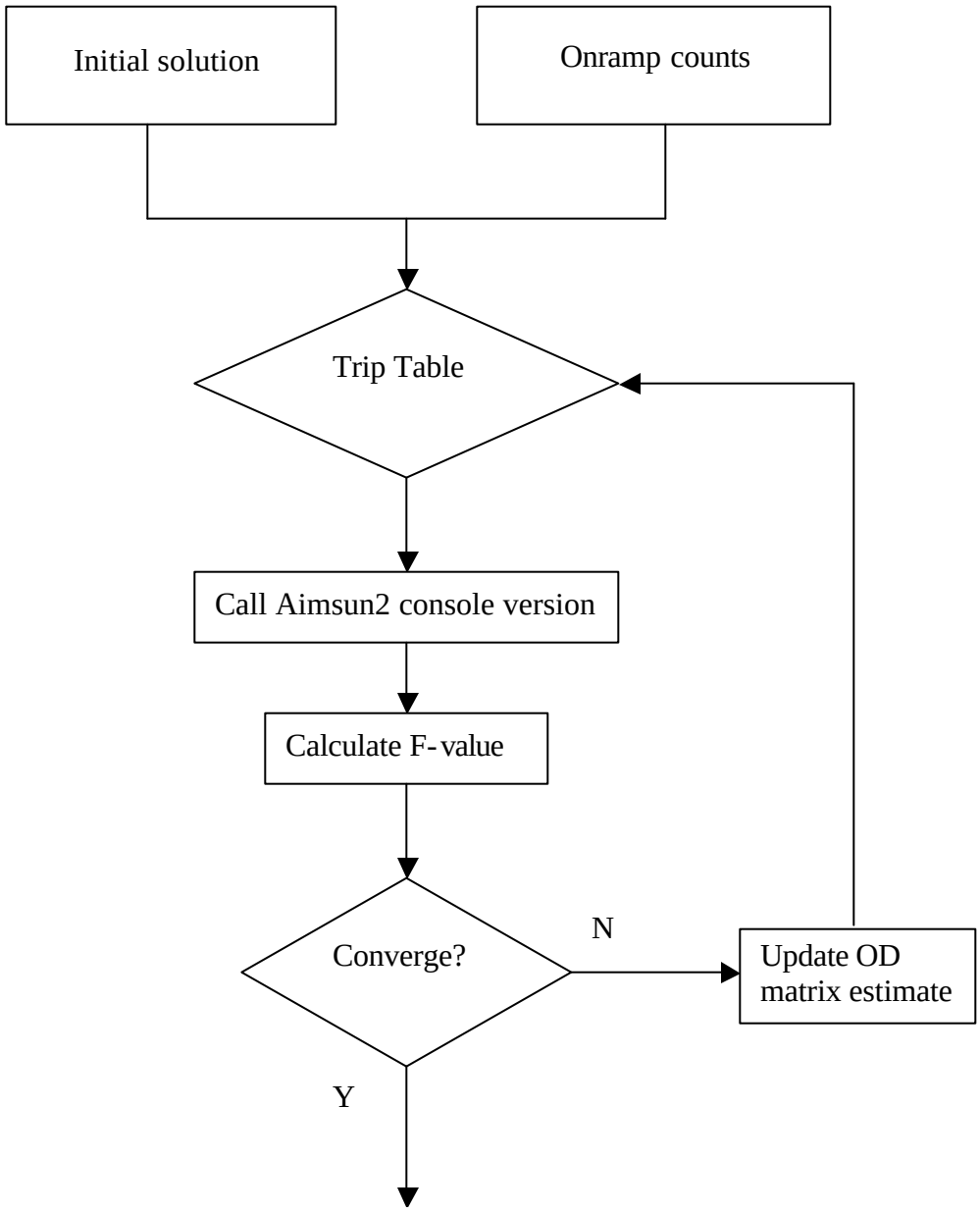
(2.8)

The Nelder-Mead method or the downhill simplex method is a non-gradient method for multidimensional minimization. It was employed to minimize the objective function.

For these tests, the OD matrix was assumed to be constant over time. In order to measure the effectiveness of the algorithm, a true OD matrix has to be available to be compared with the estimates. Since the true OD matrix for a real network is usually unknown, simulated data sets were used for the estimation.

The flow chart (Figure 2.2) describes the main steps of the implementation of this method.

1. Starting with an initial estimate and the onramp counts, these are converted to the trip table in a form that can be read by AIMSUN.
2. Then the AIMSUN console version is called to generate predicted off-ramp counts and then the objective function value (2.8) is calculated.
3. This value is compared to that of the previous run and if the convergence is achieved, then the solution of a locally optimal OD matrix is obtained. Otherwise, the Nelder-Mead updating routine is called, the OD matrix is updated, and the whole process is repeated .



O\D	1	2	…	n
1	b11	b12	…	b1n
2	0	b22	…	b2n
…	0	0	…	…
m	0	0	0	bmn

Estimated OD Matrix

Figure 2.2 Flow diagram for simulation and optimization algorithm

2.4.3 A Simple Test Network

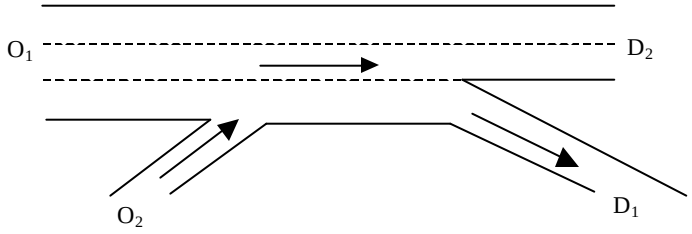


Figure 2.3 A Two-Origin Two-Destination Simulated Network

As shown in Figure 2.3, the test network has two origins and two destinations. A true OD matrix and three hours of five-minute on-ramp counts were assumed in advance, and are shown in Table 2.2. The network was coded into AIMSUN and off-ramp counts were generated using the assumed OD matrix and the on-ramp counts.

Like most numerical optimization methods, Nelder-Mead requires an initial estimate to start; three initial solutions were tested for this simulated network.

1. The equally split matrix assumes that all the destinations attract the same proportion of traffic.
2. The proportional OD matrix assumes that the OD split parameter is proportional to the traffic that the corresponding destination attracted.
3. The initial matrix is generated from an iterative method described in Willis and May (1981). This method adjusts the OD matrix proportional to the row and column sums alternatively until the inflows and outflows are balanced.

	OD-1	
True matrix	0.325	0.675
	0.25	0.75
	F-value	**0**

Initial solution 1	0.5	0.5	Initial solution 1	0.3214	0.6786
Start	0.5	0.5	End	0.3214	0.6786
				F-value	**1.799**
Initial solution 2	0.3242	0.6758	Initial solution 2	0.3245	0.6755
Start	0.3242	0.6758	End	0.3245	0.6755
				F-value	**3.317**
Initial solution 3	0.3175	0.6825	Initial solution 3	0.3165	0.6835
Start	0.4923	0.5077	End	0.4913	0.5087
				F-value	**1.571**

Table 2.2 OD estimates for the simulated network

Table 2.2 shows the results for this simulated network. Three different initial guesses of the matrix produced three different results. Although all the solutions have low objective function values, none of them is close enough to the true matrix. It might be explained by the possibility that the optimization method is not robust enough to locate the global optimum. Figure 2.4 depicts the surface of objective function values in the neighborhood of the true values of OD variables. The global optimum is hidden under the random up-and-downs so that it is very likely that the search direction stops at one local optimum.

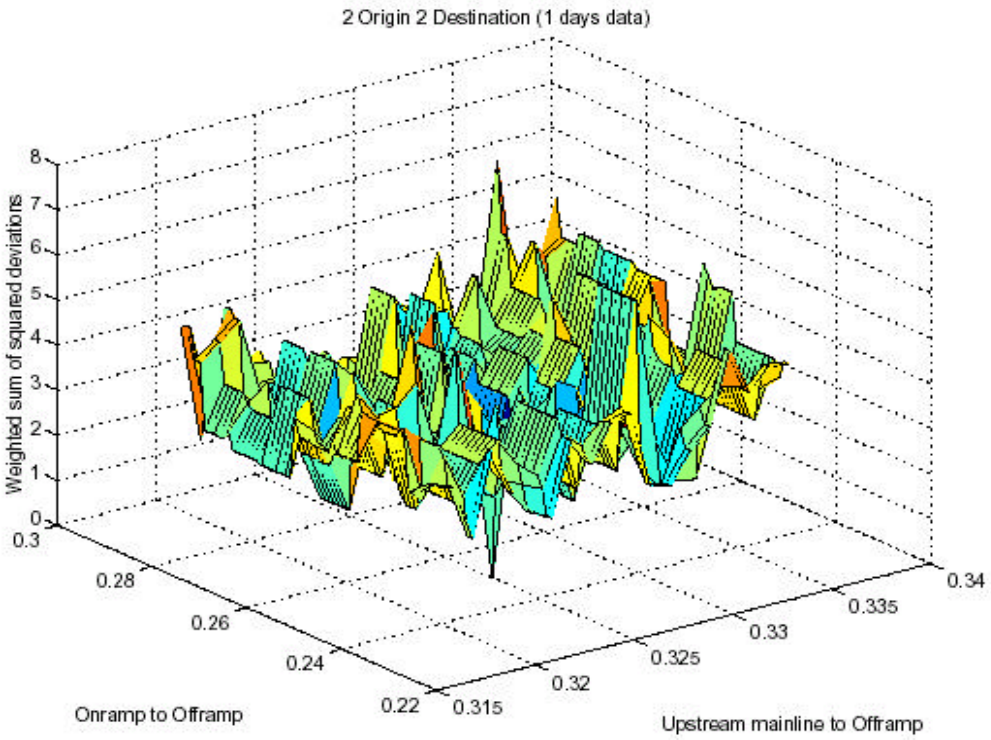


Figure 2.4 Objective Function Values Generated from AIMSUN

2.5 Advantage of the Linear Model

In the above section, we noticed that the irregularity of the objective function surface produced by the AIMSUN simulator imposes difficulties on the searching process for the global optimum. This results in the multiple local optimal solutions that approximate the observed off-ramp counts. Instead of attempting to find the global optimum from the simulation and optimization algorithm, an alternative would be to base the estimation on a better-behaved problem. Since the complication of the problem is introduced by the simulator, using the simple linear traffic model might turn out to be a reasonable approach.

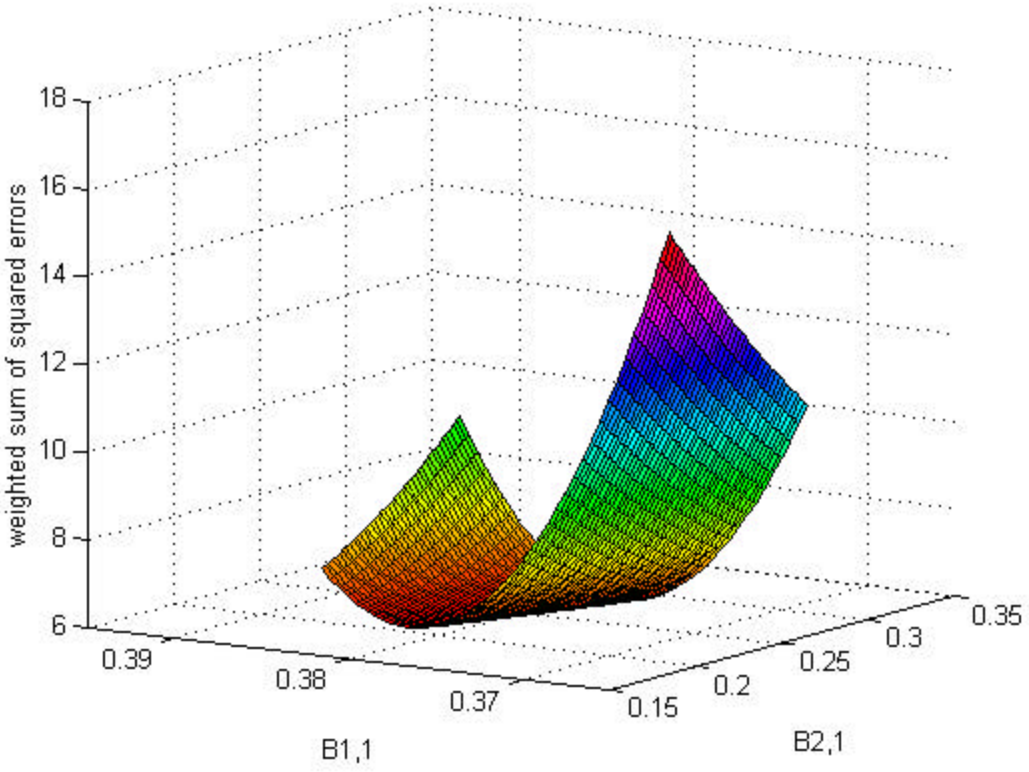


Figure 2.5 Objective Function Values Using the Linear Model

Figure 2.5 shows the surface of the objective function values of the same data set using least squares. This surface is smooth, so that the searching for the optimal point is straightforward. From optimization theory, if the objective function is strictly convex and the constraints set is convex, a unique solution exists for the optimization problem. Since the objective function (2.5) is quadratic and the constraints (2.4a and 2.4b) form a convex set, a unique OD estimate can be obtained for a given set of input and output counts. Because of this desirable property, least squares-based methods using the linear model should produce results that are easier to assess and we decided to choose them over the simulation and optimization algorithm.

CHAPTER 3. OFF-LINE ESTIMATION

3.1 Introduction

The off-line estimation of Origin-Destination matrices treats the matrices as constant over time. This chapter aims to evaluate the performance of different methods for estimating static OD matrices. As illustrated in Chapter 2, least squares-based methods using the linear traffic model and simulation and optimization were initially chosen as two candidates. However, unsuccessful experiments with the latter switched our attention to least squares methods. The rest of this chapter describes quantitative comparisons of four least squares-based methods using both simulated and actual data sets.

3.2 Description of Least Squares-Based Methods

3.2.1 Constrained Least Squares (CLS)

Assuming the linear traffic model, as in (3.1), the constrained least squares method solves the following nonlinear programming problem, where all the variables are as defined in Chapter 2.

Linear model: $$\hat{y}_j(t) = \sum_i q_i(t) b_{ij} \qquad (3.1)$$

Minimize: $$f = \sum_t \sum_j \left[\hat{y}_j(t) - y_j(t)\right]^2 \qquad (3.2)$$

Subject to: $$0 \le b_{ij} \le 1 \text{ for all i, j} \qquad (3.3a)$$

$$\sum_j b_{ij} = 1 \text{ for all i, j} \qquad (3.3b)$$

Constrained least squares can be solved by standard optimization algorithms. In this research, a Matlab built-in subroutine LSQLIN was used.

3.2.2 Weighted Constrained Least Squares (WCLS)

Instead of using equation (3.2) as the objective function, (3.4) adds weights on the squared errors. The weight of an off-ramp is the inverse of the square root of the average observed flow at that ramp so that more weights are assigned to the off-ramps with lower traffic volumes. Usually, traffic counts at the off-ramp are at least one order of magnitude smaller than those of the mainline. Therefore, mainline counts tend to be the only determining factor in measuring the closeness between the predicted and observed traffic volumes. A poor match could occur at the off-ramps. Greater weights for the off-ramps are expected to provide a closer match of traffic counts at all destinations.

$$f = \sum_i \sum_j w_j [\hat{y}_j(t) - y_j(t)]^2 \qquad (3.4)$$

$$w_j = 1/\sqrt{\bar{y}_j}$$

3.2.3 Constrained Least Squares with Time-Lagging (TCLS)

Since the simple linear traffic model ignores the potentially different travel times between OD pairs, a more realistic model should allow for time-lagging. Instead of calculating the predicted off-ramp counts from the on-ramp counts of the same time interval, they are treated as a mixture of on-ramp counts from two earlier time intervals when the exiting vehicles entered the freeway (Papageorgiou 1980).

$$\hat{y}_j(t) = \sum_i ((1-b_{ij})*b_{ij}*q_i(t-t_{ij}) + b_{ij}*b_{ij}*q_i(t-t_{ij}+1)) \qquad (3.5)$$

$$t_{ij} = Integer(tt_{ij}/T) + 1$$

$$b_{ij} = t_{ij} - tt_{ij}/T$$

where

tt_{ij} is the travel time between OD pair ij;

t_{ij} is the number of time intervals tt_{ij} occupies;

T is the duration of a counting interval;

b_{ij} is the proportion of traffic entering the freeway in the time interval $t - t_{ij} + 1$.

The implementation of this method requires travel times, which are usually obtained from speed data. However, single loop detectors do not measure speeds directly, but only lane occupancy and traffic flow. In this case, speeds can be estimated as follows.

The relationship between speed (V), density (K), and traffic flow (Q) is:

$$Q = KV \tag{3.6}$$

Density can be estimated from lane occupancy (O), which is defined as the proportion of time a short section of a roadway is occupied by vehicles.

$$O = \frac{\Delta T}{T} = \frac{(L + C)/V}{H} = (L + C) \times K \tag{3.7}$$

where ΔT is the duration of time that detector is occupied by vehicles;

T is total study time;
L is the average length of vehicles;
C is the length of the detector;
H is the vehicle headway.

Therefore, average speeds can be estimated as:

$$V = Q \times (L + C)/O \tag{3.8}$$

Coifman (2000) pointed out that in the condition of low flows, when the number of vehicles in a sample is small, the average vehicle length L can be skewed by long vehicles simply because they take longer to pass the detector. Under this condition, he suggested that the speed should be fixed as the free flow speed rather than using the above equation. Since the critical value of the occupancy is 0.10, if the lane occupancy falls below this value, traffic can be considered to be in a free flow condition. This rule is used in what follows.

3.2.4 DelftOD

DelftOD is a software package for dynamic OD matrix estimation, developed by Nanne van der Zijpp (1996) at Delft University in the Netherlands. The core of this method is a recursive least squares algorithm, which will be described in Chapter 4. However, if we fix the covariance-variance matrix of the random changes in the split parameters at zero, then this recursive algorithm is equivalent to an off-line algorithm, as long as the estimation period is long enough to ensure the convergence.

3.3 Description of data sets

3.3.1 A simple 2x2 network

As shown in Figure 2.3, the simple network has two origins and two destinations. For this network, two data sets were generated.

The first set was constructed so that the linear traffic model in equation (3.1) gives the expected values of the off-ramp counts. In this case, the CLS estimates of the OD matrix should be unbiased. This simulated data set was generated in two steps. First, the arrival rates at the mainline and on-ramp were simulated as Poisson random outcomes. Second, these arrival counts were then assigned to off-ramps as the outcomes of binomial distributions, and these were then summed to produce the exit volumes. This data generation procedure is consistent with the literature on this subject. Fifty days of data were simulated, each containing 36 five-minute counts. The chosen flow rates are 3600veh/hour for mainline and 180 veh/hour for the on-ramp and no congestion was observed during the whole simulation period on both the mainline and the on-ramp.

The second set uses AIMSUN to obtain exit counts. On-ramp counts were again sampled from a Poisson distribution. However, they were not assigned to off-ramps as binomial outcomes. The trip table was generated by multiplying the on-ramp counts with the assumed OD matrix and this was used to map the on-ramp flows to the off-ramp traffic. The exit counts were then generated by simulating traffic movement on the freeway. For AIMSUN simulation, the constant headway model was chosen for vehicles coming into the freeway, which minimizes the stochastic variations in the vehicle generation so that we get the same result for multiple simulations with the same on-ramp counts. Detectors were modeled at the on-ramps and off-ramps to collect the traffic counts. Since these detector counts were the traffic flows that actually came into and went out of the freeway during the simulation, they were used as the second data set. This data set also contained fifty days of three-hour counts.

3.3.2 Real network

TH-169 was chosen as an example of a real-freeway section. The primary reason for choosing it was that the network model was already built and calibrated for AIMSUN simulation.

As shown in Figure 3.1, the Northbound test section starts at the intersection between TH-169 and TH-55, and ends at 63rd Avenue. Figure 3.2 shows the detailed geometry. This segment is about 10.5 kilometers long with 11 on-ramps and ten off-ramps. Thus the OD matrix has 12 origins and 11 destinations. The number of nonzero individual OD elements is 76 and, after accounting for the equality constraints, the number of independent variables is 64.

Origin	1	2	3	4	5	6	7	8	9	10	11	12
Flow Rate (veh/hr)	2880	192	132	132	264	288	192	120	240	300	108	108

Table 3.1 Assumed Flow-rates for origins

For TH-169, three data sets have been tested. The first data set was generated from the linear traffic model assuming that the arrival counts are subject to Poisson distributions and the turning movements are outcomes of a multinomial distribution. The on-ramp counts were chosen such that the average for each origin was equal to the observed average for the corresponding ramp. Table 3.1 shows the assumed on-ramp flow rates. As with the two-origin two-destination case, the second data set was obtained using AIMSUN. Since no congestion was observed during the simulation period, so that queuing at on-ramps did not occur, the arrival volumes and entering volumes were the same. The third data set tested consisted of real traffic counts collected by loop detectors. The ramp counts were measured during the ramp-meter shutdown period on 23 weekdays in October and November, 2000 (October: 16-20, 23-25, 27, 30, and 31; November: 1-3, 6-10, 13, and 20-22). Only the morning peaks (7:00-10:00) were used in this study.

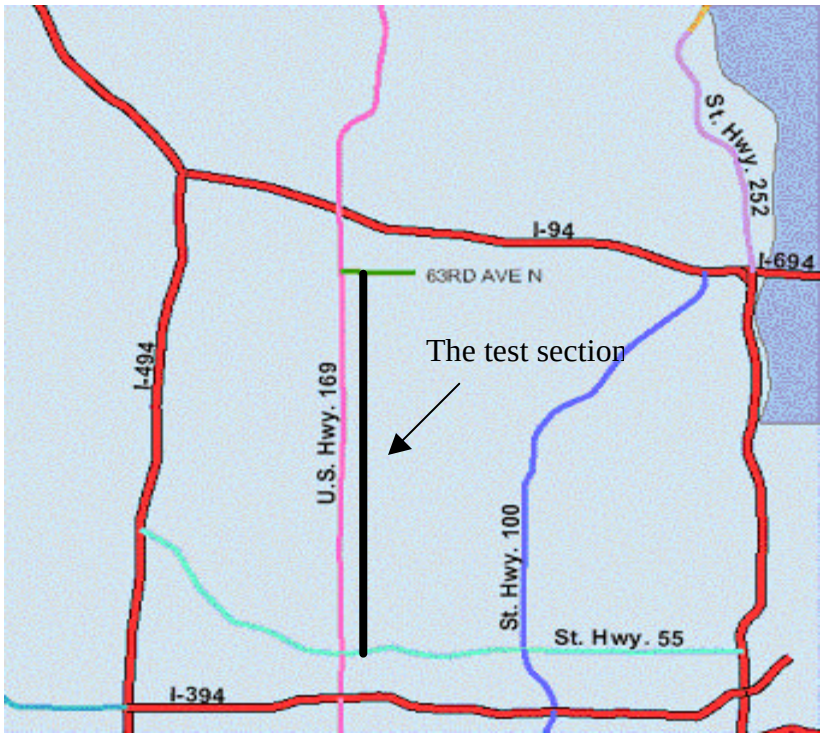


Figure 3.1 Real network of TH-169

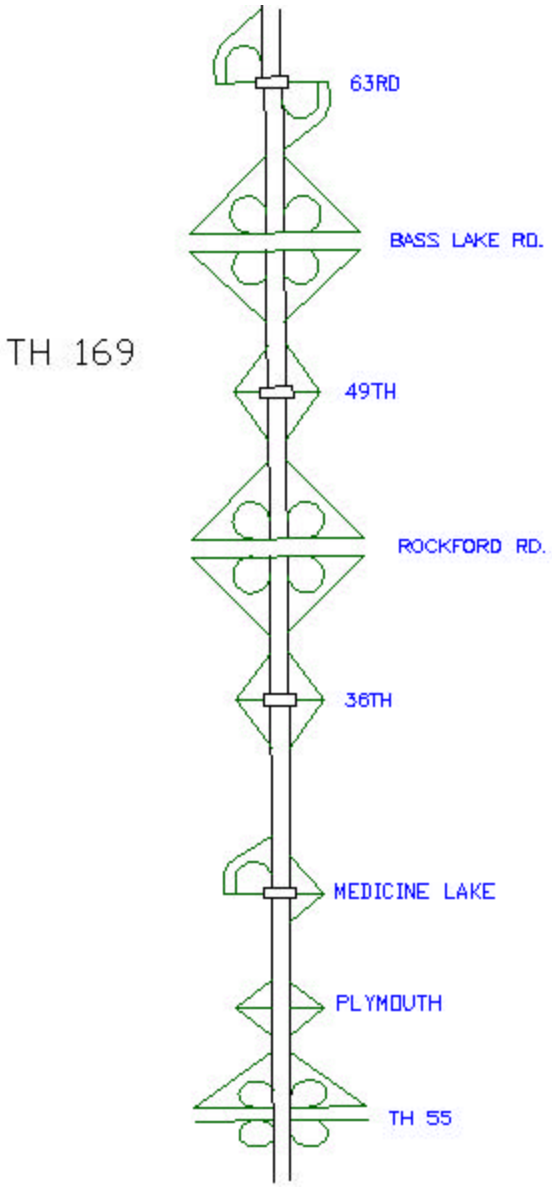


Figure 3.2 Geometry of the tested segment of TH-169

3.4 Evaluation Criteria

3.4.1 Measures of Bias and Comparative Efficiency

In statistics, a desirable estimator should have two properties: unbiased and minimum variance or efficiency. In other words, we would like to select the estimator that on average equals the true value and whose spread about this value is small. We are able to check both properties using the simulated data sets. However, because of the unavailability of the true OD matrix for the actual data, only efficiency can be evaluated.

$$MSE = \frac{1}{D}\sum_{t=1}^{D}\left(\hat{b}_{ij} - b_{ij}\right)^2 = (\bar{b}_{ij} - b_{ij})^2 + Var(\hat{b}_{ij}) \qquad (3.9)$$

where D is the number of days;

$\bar{b}_{ij}$ is the averaged value of the fifty estimates;

b_{ij} is the assumed true OD value;

$\hat{b}_{ij}$ is the estimated OD split parameter;

Var is the variance of an estimate and $Var(\hat{b}_{ij}) = \frac{1}{D}\sum_{t=1}^{D}(\hat{b}_{ij,t} - \bar{b}_{ij})^2$.

Mean Squared Error (MSE) is used to measure the average squared distance between the estimates and true values as shown in equation (3.9). The two terms on right-hand side of (3.9) are the measure of bias and efficiency respectively. Based on them, the following two criteria were developed.

$$RMSE^{bias} = \sqrt{\sum_i \sum_j (\bar{b}_{ij} - b_{ij})^2 \Big/ N} \qquad (3.10)$$

$$RMSE^{comparative\ \ efficiency} = \sqrt{\sum_i \sum_j Var(\hat{b}_{ij}) \Big/ N} \qquad (3.11)$$

where N is the total number of nonzero split parameters.

To evaluate the overall performance of an estimate, the bias and efficiency measures were combined:

$$RMSE^{Combined} = \sqrt{(RMSE^{bias})^2 + (RMSE^{comparative\ \ efficiency})^2} \qquad (3.12)$$

3.4.2 Measure of forecast uncertainty

Davis (1993) proposed a measure of forecast uncertainty based on the linear traffic model. The uncertainty of the predicted off-ramp counts contains two parts: one is demand uncertainty—the random distribution of vehicles to off-ramps; the second is parameter uncertainty—the variance due to uncertainty in the parameter estimates.

If we assume that the traffic is assigned as the outcome of a binomial distribution, the variance of the off-ramp counts when the OD matrix is known exactly is given by:

$$Var[y_j(t) \mid q(t), B] = \sum_i b_{ij}(1-b_{ij})q_i(t) \tag{3.13}$$

However, if the split parameters are estimated, they have their own variability from the estimation procedure. If $\mathbf{Q}_j$ denotes the covariance matrix of the estimates $(\hat{b}_{1j},\ldots,\hat{b}_{mj})^T$, the squared forecast error is given by (3.14) where the second term on the right-hand side of the equation evaluates parameter uncertainty.

$$s_j^2 = E[y_j(t) - \hat{y}_j(t)]^2 = \sum_i \hat{b}_{ij}(1-\hat{b}_{ij})q_i(t) + \mathbf{q}^T(t)\mathbf{Q}_j\mathbf{q}(t) \tag{3.14}$$

From practical point of view, it is easier to interpret the results in the unit of vehicles/hour, so the square roots of the two measures are taken as the evaluation criteria.

$$Demand\ \ Uncertainty_j(t) = \begin{cases} \sqrt{\sum_i b_{ij}(1-b_{ij})q_i(t)} & if\ \ \mathbf{B}\ \ is\ \ known; \\ \sqrt{\sum_i \hat{b}_{ij}(1-\hat{b}_{ij})q_i(t)} & if\ \ \mathbf{B}\ \ is\ \ unknown; \end{cases} \tag{3.15}$$

$$Parameter\ \ Uncertainty_j(t) = \sqrt{\mathbf{q}^T(t)\mathbf{Q}_j\mathbf{q}(t)} \tag{3.16}$$

In the report, the averaged values over time for the demand uncertainty and parameter uncertainty measures are reported.

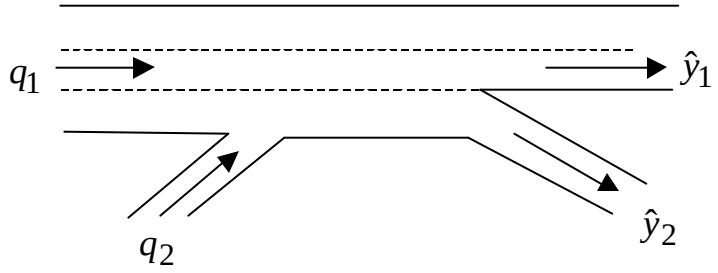


Figure 3.3 A simple network

A simple example will clarify the physical meaning of these uncertainty measures. Figure 3.3 shows a simple 2 by 2 network. To simplify the case, we only consider one time interval. Assume the entry volume is 200 vehicles for mainline and 40 vehicles for the onramp during the time interval. The estimated OD parameters and their standard deviations are displayed in Table 3.2.

Mean	1	2
1	0.375	0.625
2	0.225	0.775

S.D.	1	2
1	0.02	0.02
2	0.10	0.10

Table 3.2 An estimated OD matrix for the simple network

The demand uncertainty is the uncertainty in the process of traffic assignment. For destination 1, it is equal to

$$\sqrt{0.375(1-0.375)200 + 0.225(1-0.225)40} \cong 8veh .$$

The parameter uncertainty is due to the uncertainty in the OD matrix estimation. For destination 1, it is equal to $\sqrt{\begin{bmatrix}200 & 40\end{bmatrix}\begin{bmatrix}0.02^2 & 0\\ 0 & 0.10^2\end{bmatrix}\begin{bmatrix}200\\ 40\end{bmatrix}} \cong 23veh$.

$s_1 = \sqrt{8^2 + 23^2} = 25veh$, which is the prediction error of counts at destination 1 when using the estimated OD matrix in Table 3.2. In this case, the parameter uncertainty makes the larger contribution to it.

3.5 Results

3.5.1 The 2 by 2 network

Table 3.3 shows the results for the simple 2 by 2 simulated network. The assumed true OD matrix is on the top. The table lists the averaged estimates and their standard errors for the two fifty-day data sets.

As noted earlier, data set 1 was generated from the linear model so that the least square methods should give an unbiased estimate of the OD matrix. Results show this is the case, for both CLS and DelftOD. In all cases, approximate 95% confidence intervals, computed by adding and subtracting $2 \times$ standard deviation $/\sqrt{50}$ to the averages capture the true split parameters. It is observed that the standard deviations for origin 2 are an order of magnitude higher than those for origin 1. Note that traffic is assigned as the outcome of a binomial distribution (n, p), and p is estimated as $\hat{p} = y/n$, then $\mathrm{var}(\hat{p}) = p(1-p)/n$. The standard deviation of an OD parameter will be roughly inversely proportional to the square root of the number of entering vehicles corresponding to that parameter. In this case, the onramp flow rate is only 1/20 of that of

the mainline. Therefore, the standard deviation of mainline flow is anticipated to be much lower.

Data set 2 was generated using the AIMSUN simulator. The results for this data set show lower standard deviations than those from the first data set. The reason for less uncertainty is mainly due to the different data generation procedures. Rather than treating the OD flows as the outcomes of a binomial distribution, for AIMSUN, they were computed as binomial expected values, which eliminates some of stochastic variations.

TRUE	1	2
1	0.375	0.625
2	0.225	0.775

Data set 1

	CLS			DelftOD	
Average	1	2	Average	1	2
1	0.372	0.628	1	0.373	0.627
2	0.273	0.727	2	0.265	0.735

Standard Deviation	1	2	Standard Deviation	1	2
1	0.012	0.012	1	0.013	0.013
2	0.243	0.243	2	0.224	0.224

Data set 2

	CLS			DelftOD	
Average	1	2	Average	1	2
1	0.375	0.625	1	0.375	0.625
2	0.254	0.746	2	0.242	0.758

Standard Deviation	1	2	Standard Deviation	1	2
1	0.006	0.006	1	0.007	0.007
2	0.116	0.116	2	0.13	0.13

Table 3.3 Results of 2-Origin 2-Destination Network

Since the results for WCLS and TCLS were very similar to those of CLS and DelftOD, they were not reported. For this small network, the different algorithms do not show much variation in their performances, so the evaluation criteria are saved for the larger network.

3.5.2 TH-169

3.5.2.1 Data set 1

Three methods were compared for data set 1: DelftOD, weight constrained least squares (WCLS), and traditional constrained least squares (CLS). TCLS was not included in the comparison because the travel time was not involved in the data generation process. Tables A.1 to A.3 in Appendix A show the averages of the estimated split parameters from the fifty data sets and their standard deviations for the three methods respectively. All the estimates were approximately unbiased estimates for the true OD parameters.

	Bias	Comparative Efficiency	Combined
DelftOD	0.065	0.137	0.151
WCLS	0.063	0.134	0.149
CLS	0.034	0.142	0.146

Table 3.4 Measures of Bias and Efficiency for Data Set 1

Table 3.4 shows aggregate measures of bias and efficiency for the data generated using the linear traffic model. Estimates from CLS were on average closer to the true values, but on average had higher variability. However, all three estimates are unbiased and the magnitude of the difference in efficiency measures is small enough to be ignored. Overall speaking, the performances of the three methods are almost equivalent on this data set.

Table 3.5 shows the forecast uncertainty for the off-ramps. The covariance matrix of the estimates is diagonal assuming no cross-variance among the split parameters. For

example, $$Q_1 = \begin{bmatrix} \mathrm{var}(b_{1,1}) & 0 & \Lambda & 0 \\ 0 & \mathrm{var}(b_{2,1}) & O & M \\ M & O & O & 0 \\ 0 & \Lambda & 0 & \mathrm{var}(b_{12,1}) \end{bmatrix}.$$

Since the true OD matrix is known, equation (3.15) was used to calculate the demand uncertainty for every time interval given the on-ramp counts. The average was then computed and reported in Table 3.5. The parameter uncertainty was obtained from equation (3.16). For the upstream off-ramps 1-3 demand uncertainty exceeds parameter uncertainty, while for the remaining off-ramps the reverse is true. This is as expected since more OD parameters contribute to the flows farther downstream. Note also that the three estimation methods had similar level of parameter uncertainty.

Off-ramp	Demand Uncertainty (veh/hour)	Parameter Uncertainty (veh/hour)		
		DelftOD	WCLS	CLS
1	65	38	37	37
2	50	47	41	37
3	45	42	42	37
4	45	57	54	47
5	52	78	76	69
6	67	100	97	93
7	49	75	70	66
8	56	92	89	86
9	52	92	91	93
10	80	148	144	141
11	112	198	194	226

Table 3.5 Measures of Forecast Uncertainty for Data Set 1

3.5.2.2 Data Set 2

For this data set, the performances of all four methods were compared. The estimated matrices with the standard deviations of its elements are shown in Tables A.4 through A.7 in Appendix A. The bold-face split parameters are values whose 95% confidence intervals do not capture the true value. With a total of 76 parameters, an unbiased estimate could be expected to produce about four elements that significantly deviate from the true value at a confidence level of 95%. The numbers of bold-face values for DelftOD, TCLS, WCLS, and CLS are 12, 11, 7, and 4 respectively. This suggests that except for CLS, these methods tended to produce biased estimates.

	Bias	Comparative Efficiency	Combined
DelftOD	0.080	0.069	0.106
TCLS	0.101	0.068	0.122
WCLS	0.072	0.081	0.109
CLS	0.069	0.122	0.140

Table 3.6 Measures of Bias and Efficiency for Data Set 2

As displayed in Table 3.6, a distinguishable difference can be noticed in the measures of the performances on data set 2. The lowest deviation from the true values is observed on the CLS estimates and the lowest score on the efficiency measure is attributed to TCLS. The other three methods generated estimates that are more biased but at the mean time more efficient.

Because the tested segment is 6.5 miles long, the longest travel time should be more than one time interval—5 minutes. TCLS allows for the time-lagging, which is an

effective way to reduce the variability of the estimates. As shown in Table 3.6, the efficiency measure of DelftOD is as low as that of TCLS. For the overall measure of performance, DelftOD has the best score.

Table 3.7 displays the forecast uncertainties for data set 2. The greatest difference is at the mainline exit. Using the OD matrix estimated from TCLS compared to that from CLS, the prediction error for the mainline is reduced by approximately 120 vehicles per hour.

Off-ramp	Demand Uncertainty (veh/hour)	Parameter Uncertainty (veh/hour)			
		DelftOD	TCLS	WCLS	CLS
1	64	11	12	15	15
2	46	11	11	13	17
3	42	18	19	16	21
4	44	27	27	24	33
5	64	49	52	49	59
6	60	40	40	50	51
7	45	25	28	35	44
8	49	36	35	38	52
9	48	47	47	41	70
10	67	57	48	66	69
11	112	112	102	128	223

Table 3.7 Measures of forecast uncertainty for data set 2

3.5.2.3 Data set 3

Tables A.8 through A.11 show the estimates from the actual data. The estimated split parameters have reasonable values. The mainline exit generally attracts the largest share of the traffic from the same on-ramp.

	Comparative Efficiency
DelftOD	0.125
TCLS	0.135
WCLS	0.116
CLS	0.140

Table 3.8 Measures of efficiency for data set 3

Table 3.8 shows the measures of efficiency for data set 3. Since the true OD matrix is not available for the real data, the bias of the estimates cannot be measured. Consistent with data set 1 and 2, CLS result has more variation than those from other methods.

Table 3.9 displays the forecast uncertainty for data set 3. The demand uncertainty is essentially the same for all estimates. However, parameter uncertainty makes the difference. For the mainline, CLS introduced an extra uncertainty of 100 vehicles/hour in addition to approximately 180 vehicles/hour for the other methods.

Off-ramp	Demand Uncertainty (veh/hour)				Parameter Uncertainty (veh/hour)			
	DelftOD	TCLS	WCLS	CLS	DelftOD	TCLS	WCLS	CLS
1	28	28	28	28	50	43	36	35
2	30	30	30	30	54	62	45	44
3	17	17	17	17	43	55	51	51
4	14	13	14	14	80	76	67	65
5	18	17	17	18	98	106	96	91
6	12	12	12	13	66	73	59	62
7	23	23	23	24	93	104	74	70
8	14	14	14	14	62	73	82	77
9	18	18	18	18	82	81	84	92
10	13	13	13	13	72	76	68	78
11	84	83	86	87	180	184	174	276

Table 3.9 Measures of forecast uncertainty for data set 3

3.6 Conclusion

Using the simple linear traffic model, the least squares methods can produce a unique solution for the OD estimation problem. With both the simple network and real network, the estimates of all data sets are reasonable.

With the simple two-origin two-destination network, least squares-based methods generally showed good performance. The unique solution of each simulated data set is an unbiased estimate of the assumed true OD matrix. This result leads us to conclude that on a small network when travel time can be ignored, least squares-based methods are good approaches to estimate OD matrices. The lower standard errors in AIMSUN-generated data suggest that the accuracy of estimating split parameters depends on the accuracy of the data generation procedure.

With the larger real network TH-169, the results give us an insight in selecting the right algorithm for various purposes of OD estimation and the availability of data. For instance, with a large data set, if an unbiased matrix is desired, then the obvious choice would be CLS, as it consistently is the method that produced the estimate with lowest average bias. On the other hand, if the goal of OD estimation is to be used on ramp control, then an estimator with low variability might be preferred. WCLS,TCLS, and DelftOD are variations of CLS, but which allow for time-lagging or adding weights. The choice can be made on convenience or availability since the results suggest that none consistently outperformed the others.

CHAPTER 4. ON-LINE ESTIMATION

4.1 Introduction

In the previous chapter, time-invariant OD matrices were estimated from off-line estimation methodologies. This approach ignored the possibility of temporal variation in the OD parameters. The transportation system is dynamic. For instance, on the same freeway, the OD pattern in the morning can be significantly different from that in the afternoon. When the assumption of constant OD matrices no longer holds, detecting and responding to changes in OD matrices may be necessary. This chapter examines two methods for dynamic estimation of OD matrices, Recursive Least Squares via Kalman Filtering, and Recursive Sequential Quadratic Programming.

4.2 Description of methodologies

4.2.1 Recursive Least Squares via Kalman Filtering

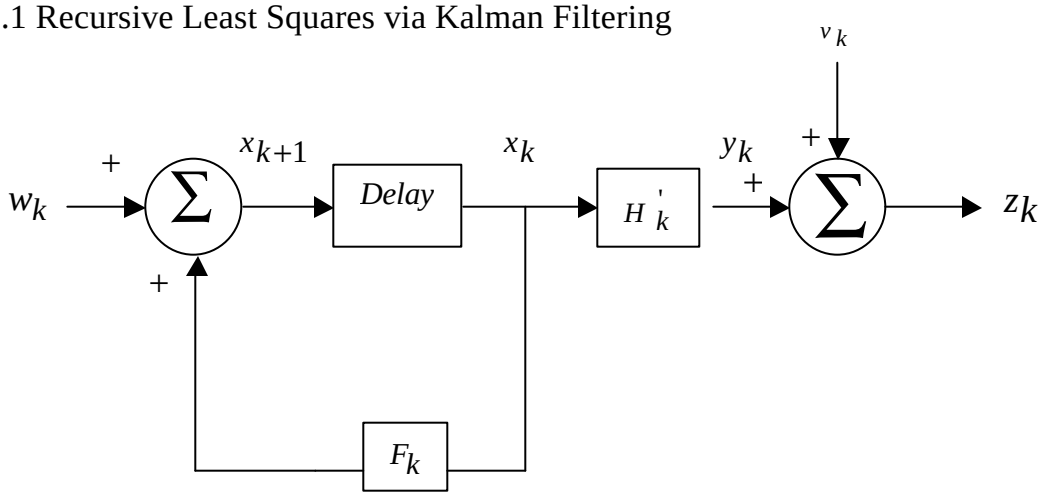


Figure 4.1 Finite-dimensional linear system serving as signal model

The theory of Kalman filtering was developed in the context of signal processing in the late 1950s and early 1960s. As depicted in figure 4.1, this finite-dimensional linear system is the prototype of the discrete-time systems (Anderson and Moore, 1979). This system can be described by a state evolution equation

$$\mathbf{x}_{k+1} = \mathbf{F}_k \mathbf{x}_k + \mathbf{w}_k \tag{4.1}$$

and an observation equation

$$\mathbf{z}_k = \mathbf{y}_k + \mathbf{v}_k = \mathbf{H}'_k \mathbf{x}_k + \mathbf{v}_k \tag{4.2}$$

where $\mathbf{x}_k$ is the system state at time k;

$\mathbf{y}_k$ is the system output;

$\mathbf{z}_k$ is the measured observation;

$\mathbf{F}_k$ is the state transition matrix;

$\mathbf{H}'_k$ is the measurement matrix;

$\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean noise.

A model for an OD matrix subject to random perturbations can be specified in the form of state-space equations.

$$\mathbf{b}_j(t+1) = \mathbf{b}_j(t) + \mathbf{w}_j(t) \tag{4.3}$$

$$y_j(t) = \mathbf{q}^T(t)\mathbf{b}_j(t) + v_j(t) \tag{4.4}$$

Suppose the variance covariance matrices for state variables and measurement error are R and r respectively. The optimal estimate of $b_j(t)$ given the sequence of observations $y_j(t)$, $y_j(t-1)$, $\cdots$, $y_j(1)$ and $q(t)$, $q(t-1)$, $\cdots$, $q(1)$ can be computed recursively via the Kalman filter.

$$\hat{\mathbf{b}}_j(t) = \hat{\mathbf{b}}_j(t-1) + \mathbf{K}_j(t)[y_j(t) - \mathbf{q}^T(t)\hat{\mathbf{b}}_j(t-1)] \tag{4.5}$$

$$\begin{aligned} \mathbf{K}_j(t) &= \frac{\mathbf{P}_j(t-1)\mathbf{q}(t)}{r_j(t) + \mathbf{q}^T(t)\mathbf{P}_j(t-1)\mathbf{q}(t)} \\ \mathbf{P}_j(t) &= \mathbf{P}_j(t-1) + \mathbf{R}_j - \frac{\mathbf{P}_j(t-1)\mathbf{q}(t)\mathbf{q}^T(t)\mathbf{P}_j(t-1)}{r_j(t) + \mathbf{q}^T(t)\mathbf{P}_j(t-1)\mathbf{q}(t)} \end{aligned} \tag{4.6}$$

As noted in Chapter 3, OD parameters should satisfy the equality and inequality constraints.

$$0 \le b_{ij} \le 1$$

$$\sum_j b_{ij} = 1$$

However, the satisfaction of the constraints are not guaranteed in Kalman filter (4.6). Therefore, we need some modifications for the traditional Kalman filter.

In literature, Nihan and Davis (1987) proposed normalization and projection procedures to impose the equality constraints. Van der Zijpp (1996) suggested another approach for constraint satisfaction, where the equality constraints are used as additional measurements in the Kalman filter. In our research, both procedures were adopted and combined in three steps for the constraint satisfaction.

1. Projection. Since the equality constraints are linear functions of the OD parameters, the additional measurement equation can be specified as follows.

$$\mathbf{1} = \mathbf{E}'\mathbf{bb}(t) \tag{4.7}$$

where $\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{m\times 1}$;

$$\mathbf{E}' = \begin{bmatrix} 1 & 0 & \Lambda & 0 & 1 & 0 & \Lambda & 0 & 1 & 0 & \Lambda & 0 & 1 & 0 & \Lambda & 0 \\ 0 & 1 & 0 & \Lambda & 0 & 1 & 0 & \Lambda & 0 & 1 & 0 & \Lambda & 0 & 1 & 0 & \Lambda \\ \mathrm{M} & & \mathrm{O} & & \mathrm{M} & & \mathrm{O} & & \mathrm{M} & & \mathrm{O} & & \mathrm{M} & & \mathrm{O} & \\ 0 & 0 & \Lambda & 1 & 0 & 0 & \Lambda & 1 & 0 & 0 & \Lambda & 1 & 0 & 0 & \Lambda & 1 \end{bmatrix}_{m\times mn}$$

;

$$\mathbf{bb}'(t) = \begin{bmatrix} b_{11}(t) & b_{21}(t) & \cdots & b_{m1}(t) & \cdots & b_{1n}(t) & b_{2n}(t) & \cdots & b_{mn}(t) \end{bmatrix}_{1\times mn}$$

.

If a zero matrix is assumed for the variance-covariance matrix of the measurement noise (i.e., the measurements are "perfect"), a Kalman Filter recursion can be applied to project the OD matrix onto the set satisfying the equality constraints. The estimates from (4.5) to (4.6) are updated as follows.

$$\begin{aligned} \hat{\mathbf{bb}}(t)^* &= \hat{\mathbf{bb}}(t) + \mathbf{K}(t)[\mathbf{1} - \mathbf{E}'\hat{\mathbf{bb}}(t)] \\ \mathbf{K}(t) &= \mathbf{P}(t)\mathbf{E}[\mathbf{E}'\mathbf{P}(t)\mathbf{E}]^{-1} \\ \mathbf{P}(t)^* &= \mathbf{P}(t) - \mathbf{P}(t)\mathbf{E}[\mathbf{E}'\mathbf{P}(t)\mathbf{E}]^{-1}\mathbf{E}'\mathbf{P}(t) \end{aligned} \tag{4.8}$$

where $\mathbf{bb}(t)^*$ is the updated estimates that satisfy the equality constraints.

2. Truncation. When the elements of $\hat{\mathbf{bb}}(t)^*$ violate the inequality constraints, we use a truncation approach.

$$\hat{b}_{ij}(t)^{**} = 0 \quad If \quad \hat{b}_{ij}(t)^{*} < 0$$

$$\hat{b}_{ij}(t)^{**} = 1 \quad If \quad \hat{b}_{ij}(t)^{*} > 1 \tag{4.9}$$

3. Normalization. Since truncation can then lead to the violation of the equality constraints, a normalization procedure proposed by Nihan and Davis (1987) was used to adjust the values of $\hat{b}_{ij}(t)^{**}$.

$$\hat{b}_{ij}(t)^{***} = \hat{b}_{ij}(t)^{**} / \sum_{j} \hat{b}_{ij}(t)^{**} \tag{4.10}$$

$b_{ij}(t)^{***}$ is the final estimated of the split parameter for OD pair ij at time interval t.

After the three steps, the constraints are guaranteed to be satisfied for all OD parameters.

In order to implement the Kalman filter algorithm, the choice of the variance-covariance matrices for error terms must be addressed. These matrices are assumed to be known in advance, but this is not the case in practice. In our experiments, the following values were used.

$$r_j(t) = \sqrt{\bar{y}_j(t)}\,, \; R = 0.0001I$$

where $\bar{y}_j(t)$ is the average of observed counts at off-ramp j.

4.2.2 Recursive Sequential Quadratic Programming

For the comparison, another recursive estimator of the OD matrix was developed based on the quadratic programming method used to solve the CLS problem in Chapter 3. A generic quadratic programming problem can be specified as

$$\begin{aligned} &Minimize \quad f = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ &subject \quad to \quad \mathbf{A}\mathbf{x} \le \mathbf{d} \end{aligned} \tag{4.11}$$

where $\mathbf{x}$ is a column vector of decision variables;

$\mathbf{H}$ is a positive definite matrix;

$\mathbf{c}$ is the first-derivative vector with respect to x;

$\mathbf{A}$ is a matrix of constraint coefficients;

$\mathbf{d}$ is a vector specifying the bounds of the constraints.

For the problem of OD estimation, the variables for optimization are the split parameters. With a change of notation, (4.11) can be rewritten as

$$\begin{aligned} &\textit{Minimize} \quad f = \frac{1}{2}\mathbf{b}\mathbf{b}^T(\mathbf{Q}^T\mathbf{Q})\mathbf{b}\mathbf{b} - (\mathbf{y}^T\mathbf{Q})\mathbf{b}\mathbf{b} \\ &\textit{subject to} \quad \sum_j b_{ij} \le 1;\ -\sum_j b_{ij} \le -1;\ b_{ij} \le 1;\ -b_{ij} \le -1 \end{aligned} \tag{4.12}$$

where

$$\mathbf{Q} = \begin{bmatrix} q_1(1) & \Lambda & q_m(1) & 0 & \Lambda & 0 & \Lambda & & & 0 \\ 0 & \Lambda & 0 & q_1(1) & \Lambda & q_m(1) & 0 & \Lambda & & 0 \\ 0 & & \Lambda & & & 0 & O & 0 & \Lambda & 0 \\ 0 & & & \Lambda & & & 0 & q_1(1) & \Lambda & q_m(1) \\ \Lambda & & \Lambda & & \Lambda & & \Lambda & & \Lambda & \\ & \Lambda & & \Lambda & & \Lambda & & \Lambda & & \Lambda \\ q_1(t) & \Lambda & q_m(t) & 0 & \Lambda & 0 & \Lambda & & & 0 \\ 0 & \Lambda & 0 & q_1(t) & \Lambda & q_m(t) & 0 & \Lambda & & \\ 0 & & \Lambda & & & 0 & O & 0 & \Lambda & 0 \\ 0 & & & \Lambda & & & 0 & q_1(t) & \Lambda & q_m(t) \end{bmatrix}$$

The algorithm for solving CLS in Chapter 3 is called the "active set method." The i^{th} constraint is said to be active if $\mathbf{A}_i^T\mathbf{x} = d_i$ and its inactive if $\mathbf{A}_i^T\mathbf{x} < d_i$. The constraint is said to be satisfied if it is active or inactive. If $\mathbf{A}_i^T\mathbf{x} > d_i$, the constraint is said to be violated. The active set includes all the constraints that are active.

Let k denote the iteration number, t_k denote the number of constraints in the working set, and I_k denote the set of indices of these constraints. $\mathbf{A}_k$, will denote the sub-matrix, containing the set of coefficients of the active constraints, while $\mathbf{z}_k$ denotes a basis for the subspace of vectors orthogonal to the rows of $\mathbf{A}_k$ (Gill, Murray and Wright, 1981).

The steps for solving off-line CLS problem (4.11) are illustrated in Figure 4.2:

1. If the conditions for optimality are satisfied at $\mathbf{x}_k$, the algorithm terminates with $\mathbf{x}_k$ as the solution. The conditions for optimality are $\mathbf{c} = \mathbf{A}_k{}^T\boldsymbol{\lambda}_k{}^*$ and $\boldsymbol{\lambda}_k{}^* \ge 0$, where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers.
2. Decide whether to continue minimizing in the current subspace or whether to delete a constraint from the working set. If a constraint is to be deleted go to step 6. If the same working set is retained, go on to step 3.

3. Compute the search direction $\mathbf{p}_k = \mathbf{z}_k{}^T(\mathbf{z}_k{}^T\mathbf{H}\mathbf{z}_k)^{-1}(-\mathbf{z}_k{}^T\mathbf{c})$.

4. Compute the maximum non-negative feasible step a_k along p_k towards the nearest constraint. $a_k = \min_i \left\{ abs\left(\frac{\mathbf{A}_{k,i}\mathbf{x}_k - d_i}{\mathbf{A}_{k,i}\mathbf{p}_k}\right) \right\}, i = 1, \cdots, ncon$

where ncon is the number of constraints in the active set $\mathbf{A}_k$.

5. If $a_k \le 1$, $\mathbf{x}_{k+1=}\mathbf{x}_k + a_k\mathbf{p}_k$, add the constraint to the working set.

6. Otherwise, let $a_k = 1$, $\mathbf{x}_{k+1=}\mathbf{x}_k + a_k\mathbf{p}_k$. Choose a constraint that is not satisfied to be deleted from the working set.

7. Go back to step 1.

The algorithm described above is designed for off-line estimation. However, with some modifications, it can be employed for on-line estimation. Let each iteration stand for a time interval, the recursive version of this algorithm updates **H** and **c** at every time interval using the observed volumes for the very time interval instead of using on-ramp counts and off-ramp counts of all time intervals. The recursion is realized by the following equations:

$$\mathbf{H}(t) = \mathbf{H}(t-1) + \frac{1}{t}[\mathbf{q}^T(t)\mathbf{q}(t) - \mathbf{H}(t-1)] \tag{4.13}$$

$$\mathbf{c}(t) = \mathbf{c}(t-1) + \frac{1}{t}[\mathbf{q}^T(t)\mathbf{y}(t) - \mathbf{c}(t-1)] \tag{4.14}$$

If $\mathbf{H}(t)$ and $\mathbf{c}(t)$ are substituted for the corresponding values of the OD matrix problem as specified in (4.12), at each time interval, the OD parameters are updated as in step 6 and a dynamic OD matrix can be obtained.

This recursive version of Sequential Quadratic Programming algorithm has the property that constraints are satisfied automatically at each iteration. Its performance on OD estimation will be compared with that of the recursive least squares via Kalman filter approach.

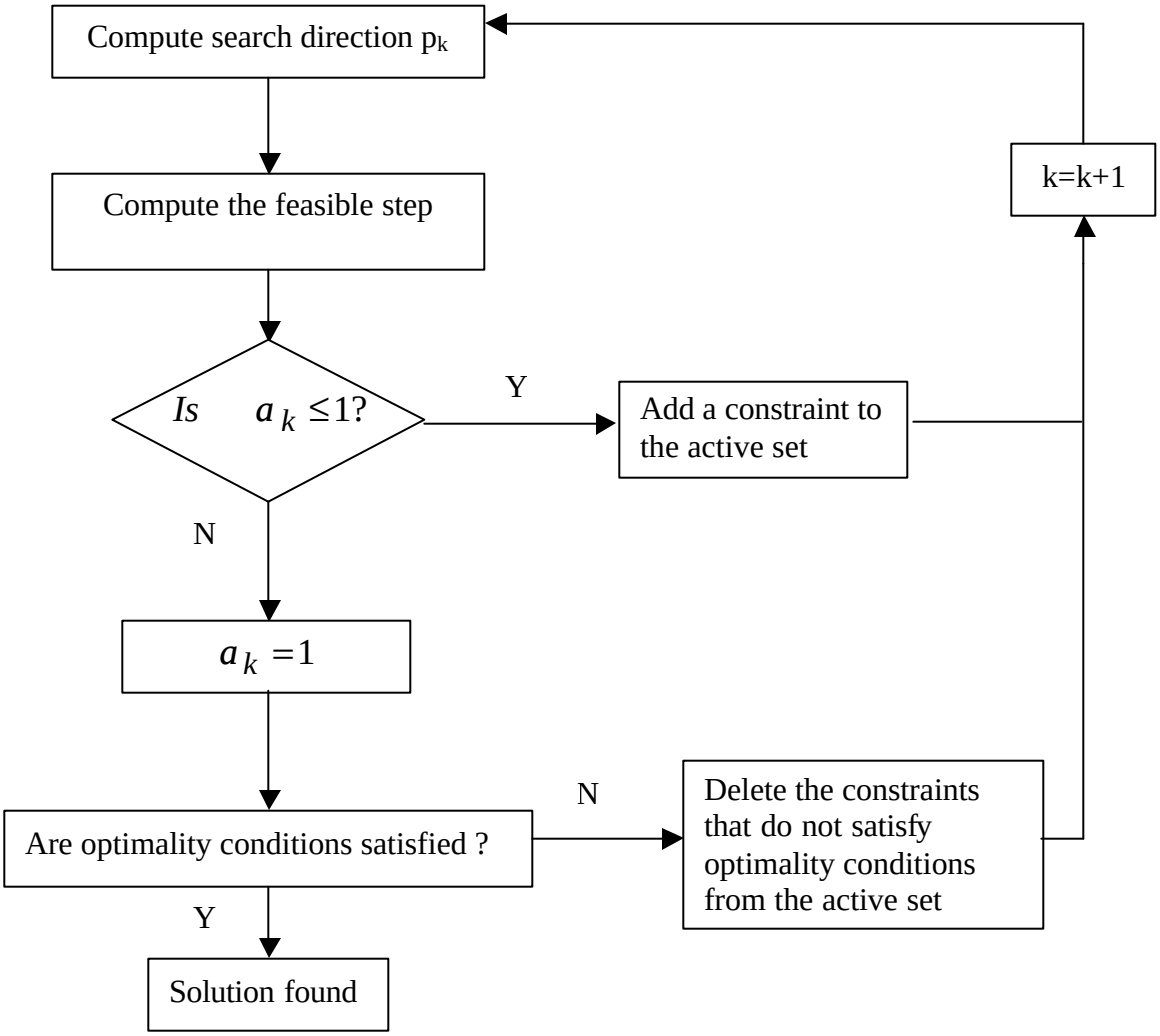


Figure 4.2 Flow diagram for active set algorithm

4.3 Description of data sets

4.3.1 The simple 2 by 2 network

For dynamic estimation, the ability of tracking the change in the OD matrix is the determining factor in evaluation of alternative algorithms. An OD matrix with an abrupt change was designed for this network. As shown in Table 4.1, a total period of six hours was assumed, with a jump happening at the beginning of the second three-hour period.

First three hours

	1	2
1	0.375	0.625
2	0.225	0.775

Second three hours

	1	2
1	0.225	0.775
2	0.375	0.625

Table 4.1 Assumed OD matrix with an abrupt change for simulated network

Similar to the off-line estimation, two data sets of five-minute on-ramp and off-ramp counts were generated. In addition to these, another two sets of one-minute counts were generated, denoted as data sets 3 and 4. The procedure of data generation was the same, and the only difference is the length of time interval. Table 4.2 summarizes the generation of these data sets.

Data Set	
1	5-minute data from linear traffic model
2	5-minute data from AIMSUN
3	1-minute data from linear traffic model
4	1-minute data from AIMSUN

Table 4.2 Description of data sets

4.3.2 TH-169 network

First three hours

	1	2	3	4	5	6	7	8	9	10	11
1	**0.14**	0.07	0.05	0.05	0.07	0.11	0.05	0.06	0.04	**0.14**	**0.25**
2	0.09	0.10	0.04	0.04	0.06	0.12	0.05	0.02	0.04	0.11	0.35
3	0.00	0.13	0.07	0.03	0.06	**0.07**	0.04	0.06	0.03	0.12	**0.40**
4	0.00	0.00	0.08	0.08	0.10	0.08	0.07	0.07	0.05	0.13	0.35
5	0.00	0.00	0.00	**0.10**	0.05	0.08	0.04	0.05	0.05	0.09	**0.55**
6	0.00	0.00	0.00	0.00	0.07	0.07	0.06	0.09	0.06	0.15	0.50
7	0.00	0.00	0.00	0.00	0.00	0.12	**0.07**	0.07	0.05	0.08	**0.62**
8	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.10	0.05	**0.20**	**0.60**
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.05	0.05	0.85
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.10	0.70
11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.90
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Second three hours

	1	2	3	4	5	6	7	8	9	10	11
1	**0.06**	0.07	0.05	0.05	0.07	0.11	0.05	0.06	0.04	**0.06**	**0.40**
2	0.09	0.10	0.04	0.04	0.06	0.12	0.05	0.02	0.04	0.11	0.35
3	0.00	0.13	0.07	0.03	0.06	**0.27**	0.04	0.06	0.03	0.12	**0.20**
4	0.00	0.00	0.08	0.08	0.10	0.08	0.07	0.07	0.05	0.13	0.35
5	0.00	0.00	0.00	**0.30**	0.05	0.08	0.04	0.05	0.05	0.09	**0.35**
6	0.00	0.00	0.00	0.00	0.07	0.07	0.06	0.09	0.06	0.15	0.50
7	0.00	0.00	0.00	0.00	0.00	0.12	**0.37**	0.07	0.05	0.08	**0.42**
8	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.10	0.05	**0.00**	**0.80**
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.05	0.05	0.85
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.10	0.70
11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.90
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Table 4.3 Assumed OD matrix with an abrupt change for TH-169

Similar to the simulated network, both the five-minute and one-minute time counts were generated for the segment of TH-169. Table 4.3 shows the assumed OD matrix for generating the data. The abrupt change takes place at the end of the first three hours. The bold-face numbers are the variables that change. A total of 11 cell values change, three of which are for flows coming from the mainline.

4.4 Results

4.4.1 The 2 by 2 network

4.4.1.1 Kalman filter results

Figure B.1 in Appendix B shows the results of Kalman filtering for the data generated from the linear traffic model. A time-series of estimated values and their 95% confidence intervals versus the true OD matrix are displayed. Data set 1 contains the five-minute data, and data set 3 has the corresponding one-minute data. Only the plots for $b_{1,1}$ and $b_{2,1}$ are presented, since the equality constraints determining the other two parameters once $b_{1,1}$ and $b_{2,1}$ are fixed.

As shown in Figure B.1 (a), for OD flows originated from the mainline, this algorithm detects the jump at the time interval 37, the estimated values decrease at a roughly quadratic rate and then converges to the true value after about 15 time intervals. However, for the flows from the on-ramp, the algorithm is not able to track the abrupt change. As in Figure B.1 (b), the estimated values remain at 0.35. Since the supposed on-ramp flow rate is 180 vehicles/hour, which is only 1/20 of the mainline, it appears that 72 data points do not provide enough information for the algorithm to detect the change.

In order to add data points, we generated the one-minute data so that the number of data points increases to 360. Figures B.1 (c) and (d) depict the results for the one-minute data. For flows from the mainline, the jump is well captured and the confidence interval is tightened. In the last time interval, the range between upper and lower bound is reduced from 0.04 to 0.02. For the on-ramp flows, the estimates also have tighter confidence intervals. In addition to that, the algorithm also shows a tendency to detect the change. As shown in Figure B.1 (d), the second half of the estimated values slowly converge to the true value, although the rate of convergence is still not as fast as that of $b_{1,1}$.

AIMSUN-generated data sets were examined as well. The results are displayed in Figure B.2. Similar to the linear traffic model, the jump in the OD parameters for the mainline origin is detected and again, because of the low volume on the on-ramp, the performance of the algorithm on those parameters is not desirable. In contrast with data set 3, changing to one-minute intervals in data set 4 does not help to track the abrupt change, although the confidence interval captures the true value. The one-minute data set includes more information at the expense of introducing more variability due to the reduced number of vehicles in one time interval. In this case, the presence of more information doesn't compensate the additional variability.

4.4.1.2 RSQP results

Figures B.3 and B.4 show the results of Recursive Sequential Quadratic Programming on data sets 1 through 4. Overall, they have the same pattern as the results for the Kalman filtering. For the parameter $b_{1,1}$, the algorithm is able to detect the change with all data sets. The introduction of one-minute data helps to get more reliable estimates but does not significantly improve the tracking behavior of the algorithm.

The RSQP algorithm differs from the Kalman filtering in that, instead of converging at approximately a quadratic rate once the jump was detected, it converges as an approximately linear rate. The rate of convergence is so slow that the confidence interval of estimates fails to capture the true value in the second half of the estimation period. This property of linear convergence reduces its ability to detect abrupt changes, compared to the Kalman filtering.

4.4.1.3 Comparison with the off-line results

Linear model							AIMSUN					
Off-line Estimation							Off-line Estimation					
Average	1	2	S.D.	1	2		Average	1	2	S.D.	1	2
1	0.372	0.628	1	0.012	0.012		1	0.375	0.625	1	0.006	0.006
2	0.273	0.727	2	0.243	0.243		2	0.254	0.746	2	0.116	0.116
Kalman Filter							Kalman Filter					
Data set 1							Data set 2					
Average	1	2	S.D.	1	2		Average	1	2	S.D.	1	2
1	0.368	0.632	1	0.012	0.012		1	0.374	0.626	1	0.007	0.007
2	0.355	0.645	2	0.213	0.213		2	0.243	0.757	2	0.127	0.127
Data set 3							Data set 4					
Average	1	2	S.D.	1	2		Average	1	2	S.D.	1	2
1	0.375	0.625	1	0.009	0.009		1	0.375	0.625	1	0.006	0.006
2	0.207	0.793	2	0.108	0.108		2	0.256	0.744	2	0.071	0.071
RSQP							RSQP					
Data set 1							Data set 2					
Average	1	2	S.D.	1	2		Average	1	2	S.D.	1	2
1	0.373	0.627	1	0.013	0.013		1	0.375	0.625	1	0.007	0.007
2	0.268	0.732	2	0.248	0.248		2	0.238	0.762	2	0.129	0.129
Data set 3							Data set 4					
Average	1	2	S.D.	1	2		Average	1	2	S.D.	1	2
1	0.376	0.624	1	0.006	0.006		1	0.375	0.625	1	0.005	0.005
2	0.190	0.810	2	0.117	0.117		2	0.296	0.704	2	0.087	0.087

Table 4.4 Comparison of on-line and off-line estimates

The estimates of the recursive algorithms at the end of the first three hours were compared with those from off-line CLS. As shown in Table 4.4, for all data sets, the recursive methods produced estimates that were not significantly different from the off-line estimates. They were able to converge to the off-line solution. Using one-minute data improved the reliability of estimates, as the standard deviation is reduced roughly by a half.

4.4.2 TH-169

Same experiments were carried using the TH-169 network. This segment of freeway is 6.5 miles long, so when generating one-minute data using the linear traffic model, the assumption that all vehicles can traverse the freeway in one time interval is unrealistic. Therefore, the one-minute data was only generated using AIMSUN.

4.4.2.1 Kalman filter results

Figures B.5 and B.6 show the results from Kalman filter with the five-minute data generated by the linear traffic model. All the 11 values that have been changed are graphed. The algorithm generally captures the change pattern for the parameters that representing the mainline incoming flows. However, for the other parameters, it only manages to show a slight trend to follow the patterns but not able to track it.

For the AIMSUN simulated data sets, only the results of six jumped parameters that have the typical behavior are shown. Results for the five-minute data generated using AIMSUN are almost the same. If the one-minute data is used, the confidence interval is tightened but for the lower-volume incoming flows, the detection of jump is still a problem. In addition to that, using the one-minute data appeared to introduce some systematic biases into the estimation. As for the parameter $b_{1,11}$, the estimated values are constantly higher than the values that generated the data. Since the travel time from the first origin to the last destination on this network is larger than one minute, time-lagging should be taken into consideration. Not allowing for time lags in the one-minute data is the most likely source that introduced the bias.

4.4.2.2 RSQP results

Figures B.9 through B.12 display the time-series plots for the RSQP estimation. This algorithm shows some ability to track an abrupt change, and generally the estimates are following the right direction. However, in terms of the proximity to the true value and the reliability of the estimates, the Kalman filter outperforms RSQP.

4.5 Conclusions

Recursive algorithms are able to detect the abrupt changes in the split parameters of mainline but not those with lower traffic volumes. Both Kalman Filter and RSQP performed well on the parameters origin in the mainline. The estimates from on-line algorithms converge to the off-line estimates as indicated by the results from the 2 by 2 network.

RLS via Kalman Filtering outperforms RSQP because of a faster convergence rate. The Kalman filter converges at an approximately quadratic rate. In contrast, RSQP has an approximately linear rate.

Additional information from reducing the length of the counting interval does not necessarily improve the estimates because of the additional biases introduced.

CHAPTER 5. CONCLUSIONS

This report describes research on estimating OD matrices for freeways from time-series on-ramp and off-ramp traffic counts. The simulation and optimization method was initially selected. However, when using AIMSUN as the prediction engine, the surface of objective function values is irregular, which poses difficulties for the optimization routine in locating the global optimal. The linear traffic model is then adopted to replace AIMSUN because of the tractable optimization problem it produces. Least squares-based methods using the linear model are able to generate unique estimates of the OD parameters and the results are easy to assess. Therefore, they are chosen over the simulation and optimization methods.

The performances of four least squares-based methods on estimating a static OD matrix are compared using both the simple two-origin two-destination network and the real TH-169 network. For the simple network, least squares-based methods produced unbiased estimate of the assumed true OD matrix. With the larger real network TH-169, traditional CLS consistently had the lowest bias but worst efficiency measure. In contrast, the other three methods generated slightly more biased but also more reliable estimates. The choice of the algorithm depends on the availability of data. For instance, with a large data set, if an unbiased estimate is desired, then the choice would be CLS. However, when the size of the data set is limited and the reliable OD estimator is desired, then one of the alternative variations of CLS (WCLS,TCLS, or DelftOD) should be chosen.

Recursive Least Squares via Kalman Filter and Recursive Sequential Quadratic Programming were tested for the on-line estimation of dynamic OD matrix. Both of them were able to detect the abrupt changes in the split parameters with high traffic volumes but not those with lower volumes. The Kalman Filter estimator converged at an approximately quadratic rate, which was faster than RSQP. One-minute data was tested in addition to five-minute data. The additional information from reducing the length of the counting interval tightens the confidence interval of the estimates, but does not necessarily improve the ability to respond to changes.

Travel time laggings were not accounted for in the on-line estimation methods. From our results, the additional information from using a shorter time interval introduced a bias when travel time lags became important. By taking into account the travel time, the additional information could be better utilized. The way to implement this would be to replace the measurement equation in Kalman Filter with an equation equivalent to that in TCLS.

The sequential quadratic programming approach showed an approximately linear convergence rate. The possibility of achieving a faster convergence rate should be investigated to refine this method.

In this report, we focused on freeways. In order to expand the methods to general networks, route-choice has to be considered. For on-line applications, the dynamic traffic assignment model would have to be incorporated into the estimation process.

References

1. Anderson B.D.O. and Moore J.B. (1979) Optimal Filtering.
2. Ashok K. (1996) Estimation and Prediction of Time-Dependent Origin-Destination Flow. Ph.D. Thesis, Massachusetts Institute of Technology.
3. Bell M. G.H. (1991) The real time estimation of origin-destination flows in the presence of platoon dispersion *Transportation Research B.,* Vol 25B, Nos. 2/3, pp. 115-125.
4. Bureau of Transportation Statistics, *Transportation Statistics Annual Report* 1997, BTS97-S-01, U.S. DOT, 1997.
5. Cascetta E. (1984) Estimation of trip matrices from traffic counts and survey data : a generalized least squares estimator. *Transportation Research Part B, v. 18B*, no. 4/5, pp. 289-299.
6. Cascetta E., Inaudi D. and Marquis G. (1993) Dynamic estimators of origin-destination matrices using traffic counts. *Transportation Science,* Vol.27, No.4.
7. Chang G. and Wu J. (1994) Recursive Estimation of Time-varying Origin-Destination Flows from Traffic Counts in Freeway Corridors. *Transportation Research B,* Vol. 28B, No.2, pp. 141-160.
8. Cremer M. (1983) Dynamic identification of flows from traffic counts at complex intersections. *Proceedings of the eighth International Symposium on Transportation and Traffic Theory* (V. F. Hurdle *et al*., Eds.). University of Toronto Press, Toronto, pp. 121-142.
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. 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.
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. Gill P.E., Murray W. and Wright M.H. (1981) Practical Optimization.
13. Hendrickson C. and McNeil S. (1984) Estimation of origin-destination matrices with constrained regression. *Transportation Research Record*, 976, pp. 25-32.
14. Kang J. (1995) Esimation of destination-specific traffic densities and identification of parameters on urban freeways using Markov models of traffic flow. Ph.D. Thesis, University of Minnesota.
15. Ljung L. and Söderström T. (1983) Theory and Pratic of Recursive Identification. MIT Press, Cambridge, MA.
16. Maher M.J. (1983) Inferences on trip matrices from observations on link volumes : a Bayesian statistical approach. *Transportation Research Part B*, v. 17B, no. 6, pp. 435-447.
17. 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.

18. 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.
19. 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.
20. Papacostas C.S. and Prevedouros P.D. (2001) *Transportation Engineering & Planning*.
21. Papageorgiou M. (1980) A new approach to time-of-day control based on a dynamic freeway traffic model, *Transportation Research Part B*, v. 14B, no. 4, pp. 349-360.
22. Press, W, H. Teukolsky, S, A. Vetterling, W, T. Flannery, B, P. *Numerical Recipes in FORTRAN*. Second Edition. Cambridge University Press, 1994.
23. Turnquist, M. and Gur, Y. (1979) Estimation of trip tables from observed link volumes, *Transportation Research Record*, 730.
24. Van der Zijpp N. (1996) Dynamic origin-destination matrix estimation on motorway networks. Ph.D. Thesis, Delft University of Technology.
25. Van Zuylen H. J. and Willumsen L.G. (1979) The most likely trip matrix estimate from traffic counts. *Transportation Res. B*, Vol. 14B, pp. 281-293.
26. Yu P. and Davis G.A. (1994) Estimating freeway origin-destination patterns using automation traffic counts. *Transportation Research Board*, 1457.

APPENDIX A: OFF-LINE ESTIMATION RESULTS

Table A.1 Data set 1 (DelftOD)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.133	0.065	0.047	0.039	0.054	0.098	0.039	0.045	0.023	0.121	0.336
2	0.114	0.100	0.043	0.068	0.106	0.093	0.084	0.077	0.085	0.082	0.149
3	0	0.139	0.108	0.047	0.126	0.086	0.042	0.086	0.066	0.139	0.162
4	0	0	0.109	0.141	0.105	0.094	0.087	0.130	0.077	0.142	0.117
5	0	0	0	0.090	0.075	0.077	0.082	0.055	0.067	0.109	0.447
6	0	0	0	0	0.090	0.093	0.082	0.093	0.079	0.158	0.405
7	0	0	0	0	0	0.201	0.058	0.108	0.088	0.121	0.423
8	0	0	0	0	0	0	0.088	0.111	0.105	0.236	0.460
9	0	0	0	0	0	0	0	0.091	0.079	0.089	0.741
10	0	0	0	0	0	0	0	0	0.206	0.107	0.686
11	0	0	0	0	0	0	0	0	0	0.173	0.827
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.010	0.013	0.010	0.014	0.021	0.023	0.018	0.021	0.020	0.035	0.048
2	0.149	0.124	0.087	0.095	0.148	0.132	0.100	0.104	0.121	0.111	0.204
3	0	0.176	0.139	0.083	0.154	0.126	0.068	0.116	0.117	0.184	0.215
4	0	0	0.133	0.141	0.119	0.137	0.128	0.172	0.126	0.203	0.183
5	0	0	0	0.102	0.095	0.095	0.102	0.065	0.079	0.160	0.229
6	0	0	0	0	0.090	0.129	0.088	0.101	0.096	0.147	0.203
7	0	0	0	0	0	0.193	0.088	0.126	0.106	0.173	0.257
8	0	0	0	0	0	0	0.134	0.145	0.150	0.232	0.233
9	0	0	0	0	0	0	0	0.106	0.103	0.134	0.183
10	0	0	0	0	0	0	0	0	0.123	0.148	0.177
11	0	0	0	0	0	0	0	0	0	0.214	0.214
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.2 Data set 1 (WCLS)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.133	0.065	0.047	0.039	0.053	0.098	0.040	0.046	0.025	0.121	0.332
2	0.111	0.095	0.043	0.069	0.106	0.089	0.082	0.073	0.087	0.092	0.153
3	0	0.137	0.115	0.047	0.125	0.088	0.036	0.084	0.065	0.139	0.165
4	0	0	0.102	0.140	0.113	0.108	0.079	0.123	0.073	0.130	0.130
5	0	0	0	0.089	0.072	0.073	0.080	0.056	0.061	0.113	0.455
6	0	0	0	0	0.103	0.094	0.079	0.086	0.073	0.162	0.403
7	0	0	0	0	0	0.191	0.060	0.105	0.091	0.115	0.439
8	0	0	0	0	0	0	0.086	0.111	0.112	0.234	0.457
9	0	0	0	0	0	0	0	0.090	0.084	0.088	0.738
10	0	0	0	0	0	0	0	0	0.197	0.103	0.700
11	0	0	0	0	0	0	0	0	0	0.170	0.830
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.010	0.011	0.010	0.013	0.020	0.022	0.016	0.020	0.020	0.034	0.046
2	0.147	0.115	0.085	0.093	0.145	0.133	0.100	0.101	0.126	0.118	0.196
3	0	0.168	0.142	0.075	0.157	0.125	0.063	0.106	0.113	0.180	0.210
4	0	0	0.125	0.146	0.127	0.149	0.111	0.161	0.117	0.197	0.188
5	0	0	0	0.096	0.098	0.086	0.102	0.073	0.073	0.163	0.243
6	0	0	0	0	0.097	0.123	0.088	0.097	0.089	0.155	0.206
7	0	0	0	0	0	0.191	0.089	0.126	0.110	0.163	0.262
8	0	0	0	0	0	0	0.123	0.142	0.148	0.226	0.225
9	0	0	0	0	0	0	0	0.110	0.108	0.133	0.178
10	0	0	0	0	0	0	0	0	0.114	0.142	0.164
11	0	0	0	0	0	0	0	0	0	0.207	0.207
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.3 Data set 1 (CLS)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.133	0.068	0.050	0.046	0.059	0.102	0.046	0.053	0.031	0.121	0.290
2	0.103	0.078	0.038	0.049	0.096	0.087	0.064	0.066	0.065	0.101	0.254
3	0	0.098	0.085	0.026	0.097	0.083	0.028	0.062	0.052	0.150	0.320
4	0	0	0.073	0.094	0.089	0.102	0.064	0.100	0.062	0.131	0.285
5	0	0	0	0.065	0.059	0.065	0.067	0.045	0.050	0.112	0.537
6	0	0	0	0	0.091	0.086	0.064	0.075	0.066	0.154	0.465
7	0	0	0	0	0	0.164	0.046	0.072	0.075	0.105	0.538
8	0	0	0	0	0	0	0.080	0.093	0.094	0.216	0.517
9	0	0	0	0	0	0	0	0.097	0.068	0.078	0.757
10	0	0	0	0	0	0	0	0	0.205	0.117	0.679
11	0	0	0	0	0	0	0	0	0	0.181	0.819
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.010	0.010	0.009	0.011	0.018	0.021	0.015	0.019	0.021	0.034	0.052
2	0.144	0.109	0.071	0.079	0.130	0.133	0.091	0.086	0.099	0.123	0.264
3	0	0.138	0.132	0.053	0.133	0.120	0.060	0.097	0.103	0.195	0.330
4	0	0	0.112	0.135	0.115	0.149	0.099	0.142	0.113	0.182	0.288
5	0	0	0	0.081	0.086	0.083	0.102	0.064	0.084	0.152	0.261
6	0	0	0	0	0.093	0.119	0.084	0.102	0.093	0.153	0.269
7	0	0	0	0	0	0.177	0.087	0.126	0.106	0.145	0.302
8	0	0	0	0	0	0	0.121	0.141	0.127	0.215	0.262
9	0	0	0	0	0	0	0	0.112	0.105	0.119	0.187
10	0	0	0	0	0	0	0	0	0.133	0.142	0.198
11	0	0	0	0	0	0	0	0	0	0.225	0.225
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.4 Data set 2 (DelftOD)

Mean	AIMSUN-DelftOD										
	1	2	3	4	5	6	7	8	9	10	11
1	0.130	0.061	0.046	0.043	0.076	0.094	0.044	0.049	0.025	**0.100**	**0.330**
2	0.025	**0.020**	0.036	0.024	0.064	0.042	0.034	0.032	0.036	0.073	**0.615**
3	0	**0.027**	0.031	0.038	0.058	0.042	0.032	0.038	0.045	0.070	0.619
4	0	0	0.083	**0.257**	0.193	0.036	0.033	0.054	0.029	0.075	0.240
5	0	0	0	**0.017**	0.132	**0.020**	0.021	0.047	0.038	0.076	0.649
6	0	0	0	0	**0.423**	0.027	0.018	**0.021**	0.017	**0.039**	0.453
7	0	0	0	0	0	0.112	0.088	0.041	0.104	0.064	0.591
8	0	0	0	0	0	0	0.031	0.046	0.038	0.069	0.815
9	0	0	0	0	0	0	0	0.044	0.039	0.052	0.866
10	0	0	0	0	0	0	0	0	**0.309**	0.048	0.642
11	0	0	0	0	0	0	0	0	0	0.104	0.896
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.003	0.003	0.005	0.007	0.012	0.010	0.005	0.008	0.011	0.013	0.026
2	0.037	0.034	0.034	0.036	0.070	0.041	0.038	0.038	0.043	0.067	0.134
3	0	0.031	0.035	0.050	0.055	0.054	0.036	0.040	0.060	0.072	0.142
4	0	0	0.057	0.088	0.132	0.060	0.051	0.076	0.048	0.090	0.187
5	0	0	0	0.034	0.079	0.032	0.029	0.047	0.048	0.072	0.152
6	0	0	0	0	0.059	0.035	0.023	0.022	0.023	0.048	0.086
7	0	0	0	0	0	0.082	0.056	0.045	0.075	0.057	0.153
8	0	0	0	0	0	0	0.045	0.065	0.050	0.095	0.148
9	0	0	0	0	0	0	0	0.049	0.058	0.066	0.124
10	0	0	0	0	0	0	0	0	0.047	0.050	0.064
11	0	0	0	0	0	0	0	0	0	0.101	0.101
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.5 Data set 2 (TCLS)

Mean

1	0.130	0.062	0.047	0.042	0.067	0.097	0.047	0.057	0.028	0.121	**0.301**
2	0.022	**0.016**	0.027	0.021	0.062	0.024	0.028	0.026	0.021	0.032	**0.720**
3	0	**0.019**	0.024	0.044	0.033	0.032	0.022	0.028	0.035	0.038	0.724
4	0	0	0.076	0.281	0.199	0.036	0.028	0.041	0.040	0.034	0.265
5	0	0	0	**0.014**	0.146	**0.012**	0.015	0.026	0.034	0.035	0.718
6	0	0	0	0	**0.497**	0.019	0.014	**0.013**	**0.010**	**0.015**	0.432
7	0	0	0	0	0	0.099	0.073	0.027	0.107	0.028	0.666
8	0	0	0	0	0	0	0.022	0.041	0.038	0.050	**0.849**
9	0	0	0	0	0	0	0	0.020	0.028	0.024	0.928
10	0	0	0	0	0	0	0	0	0.305	0.021	0.674
11	0	0	0	0	0	0	0	0	0	0.064	0.936
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.

1	0.003	0.003	0.005	0.006	0.012	0.010	0.006	0.008	0.011	0.012	0.023
2	0.047	0.035	0.041	0.041	0.080	0.050	0.039	0.047	0.042	0.059	0.133
3	0	0.028	0.045	0.065	0.058	0.065	0.036	0.046	0.064	0.070	0.169
4	0	0	0.058	0.106	0.135	0.061	0.052	0.057	0.061	0.068	0.186
5	0	0	0	0.028	0.085	0.027	0.026	0.044	0.041	0.049	0.121
6	0	0	0	0	0.072	0.034	0.031	0.026	0.019	0.038	0.096
7	0	0	0	0	0	0.081	0.061	0.048	0.082	0.043	0.149
8	0	0	0	0	0	0	0.035	0.060	0.059	0.086	0.120
9	0	0	0	0	0	0	0	0.038	0.045	0.044	0.084
10	0	0	0	0	0	0	0	0	0.055	0.043	0.072
11	0	0	0	0	0	0	0	0	0	0.096	0.096
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.6 Data set 2 (WCLS)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.131	0.062	0.047	0.042	0.065	**0.077**	0.036	**0.043**	0.026	**0.085**	**0.387**
2	0.036	0.022	0.030	0.049	0.084	0.069	0.056	0.065	0.031	0.091	0.466
3	0	**0.038**	0.042	0.065	0.102	0.073	0.048	0.058	0.056	0.094	0.425
4	0	0	0.075	0.186	0.090	0.074	0.046	0.054	0.048	0.094	0.332
5	0	0	0	0.031	0.175	0.062	0.035	0.046	0.033	0.095	0.522
6	0	0	0	0	**0.499**	0.062	0.029	0.035	0.028	0.061	0.286
7	0	0	0	0	0	0.182	0.112	0.053	0.066	0.095	0.491
8	0	0	0	0	0	0	0.054	0.070	0.053	0.127	0.696
9	0	0	0	0	0	0	0	0.049	0.046	0.091	0.814
10	0	0	0	0	0	0	0	0	**0.293**	0.049	0.659
11	0	0	0	0	0	0	0	0	0	0.099	0.901
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.004	0.003	0.004	0.005	0.012	0.012	0.008	0.008	0.009	0.014	0.027
2	0.057	0.043	0.030	0.059	0.067	0.071	0.051	0.068	0.042	0.099	0.188
3	0	0.043	0.047	0.062	0.082	0.072	0.046	0.064	0.072	0.075	0.209
4	0	0	0.053	0.072	0.079	0.069	0.062	0.069	0.058	0.093	0.206
5	0	0	0	0.034	0.078	0.056	0.039	0.042	0.042	0.078	0.161
6	0	0	0	0	0.075	0.054	0.034	0.038	0.039	0.064	0.126
7	0	0	0	0	0	0.097	0.066	0.048	0.052	0.087	0.209
8	0	0	0	0	0	0	0.063	0.070	0.065	0.123	0.185
9	0	0	0	0	0	0	0	0.043	0.047	0.077	0.121
10	0	0	0	0	0	0	0	0	0.046	0.059	0.082
11	0	0	0	0	0	0	0	0	0	0.099	0.099
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.7 Data set 2 (CLS)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.131	0.060	0.046	0.042	0.069	**0.081**	0.036	0.043	0.033	**0.089**	**0.370**
2	0.036	0.026	0.026	0.044	0.075	0.066	0.048	0.058	0.041	0.074	0.505
3	0	0.063	0.073	0.086	0.097	0.073	0.067	0.064	0.066	0.093	0.317
4	0	0	0.069	0.155	0.081	0.056	0.039	0.043	0.048	0.082	0.427
5	0	0	0	0.046	0.170	0.066	0.044	0.050	0.050	0.091	0.483
6	0	0	0	0	**0.470**	0.050	0.022	0.025	0.024	0.050	0.359
7	0	0	0	0	0	0.156	0.090	0.040	0.062	0.076	0.576
8	0	0	0	0	0	0	0.080	0.088	0.083	0.127	0.622
9	0	0	0	0	0	0	0	0.076	0.077	0.103	0.744
10	0	0	0	0	0	0	0	0	0.170	0.025	0.806
11	0	0	0	0	0	0	0	0	0	0.152	0.848
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.004	0.004	0.005	0.007	0.014	0.012	0.010	0.012	0.015	0.014	0.048
2	0.059	0.049	0.037	0.067	0.076	0.070	0.058	0.064	0.063	0.088	0.307
3	0	0.069	0.068	0.081	0.080	0.074	0.058	0.064	0.081	0.080	0.324
4	0	0	0.076	0.104	0.080	0.059	0.058	0.056	0.066	0.094	0.344
5	0	0	0	0.058	0.096	0.058	0.049	0.051	0.055	0.085	0.306
6	0	0	0	0	0.111	0.058	0.039	0.044	0.046	0.056	0.220
7	0	0	0	0	0	0.100	0.086	0.054	0.080	0.077	0.294
8	0	0	0	0	0	0	0.095	0.095	0.090	0.127	0.315
9	0	0	0	0	0	0	0	0.087	0.092	0.107	0.259
10	0	0	0	0	0	0	0	0	0.128	0.053	0.162
11	0	0	0	0	0	0	0	0	0	0.178	0.178
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.8 Data set 3 (DelftOD)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.131	0.135	0.061	0.029	0.046	0.022	0.063	0.030	0.037	0.007	0.439
2	0.114	0.207	0.091	0.024	0.158	0.094	0.043	0.078	0.069	0.028	0.096
3	0	0.038	0.099	0.121	0.161	0.301	0.000	0.110	0.000	0.000	0.167
4	0	0	0.194	0.124	0.103	0.224	0.013	0.142	0.030	0.026	0.141
5	0	0	0	0.361	0.072	0.068	0.056	0.094	0.011	0.039	0.298
6	0	0	0	0	0.074	0.026	0.201	0.046	0.128	0.080	0.444
7	0	0	0	0	0	0.000	0.371	0.030	0.101	0.199	0.301
8	0	0	0	0	0	0	0.006	0.057	0.000	0.036	0.901
9	0	0	0	0	0	0	0	0.014	0.118	0.085	0.783
10	0	0	0	0	0	0	0	0	0.114	0.212	0.674
11	0	0	0	0	0	0	0	0	0	0.021	0.979
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.013	0.013	0.010	0.018	0.027	0.015	0.021	0.012	0.016	0.011	0.038
2	0.174	0.171	0.075	0.047	0.164	0.115	0.070	0.107	0.100	0.052	0.172
3	0	0.163	0.129	0.126	0.177	0.189	0.000	0.096	0.000	0.000	0.213
4	0	0	0.167	0.165	0.170	0.220	0.042	0.127	0.066	0.076	0.177
5	0	0	0	0.211	0.110	0.075	0.070	0.123	0.023	0.084	0.261
6	0	0	0	0	0.106	0.047	0.178	0.075	0.102	0.084	0.252
7	0	0	0	0	0	0.000	0.224	0.049	0.120	0.143	0.294
8	0	0	0	0	0	0	0.024	0.095	0.000	0.084	0.129
9	0	0	0	0	0	0	0	0.042	0.162	0.113	0.183
10	0	0	0	0	0	0	0	0	0.109	0.114	0.177
11	0	0	0	0	0	0	0	0	0	0.050	0.050
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.9 Data set 1 (TCLS)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.131	0.135	0.061	0.025	0.046	0.017	0.062	0.026	0.041	0.007	0.450
2	0.094	0.195	0.104	0.043	0.127	0.102	0.041	0.103	0.084	0.008	0.099
3	0	0.026	0.112	0.135	0.168	0.333	0.005	0.116	0.001	0.003	0.100
4	0	0	0.151	0.09	0.078	0.249	0.033	0.124	0.031	0.038	0.207
5	0	0	0	0.418	0.08	0.084	0.043	0.083	0.012	0.036	0.243
6	0	0	0	0	0.09	0.024	0.25	0.066	0.107	0.09	0.372
7	0	0	0	0	0	0	0.319	0.059	0.086	0.215	0.321
8	0	0	0	0	0	0	0.033	0.055	0.003	0.022	0.886
9	0	0	0	0	0	0	0	0.012	0.097	0.079	0.812
10	0	0	0	0	0	0	0	0	0.114	0.223	0.663
11	0	0	0	0	0	0	0	0	0	0.022	0.978
12	0	0	0	0	0	0	0	0	0	0	1

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.011	0.015	0.013	0.016	0.029	0.013	0.024	0.015	0.014	0.008	0.038
2	0.152	0.218	0.14	0.093	0.179	0.142	0.093	0.121	0.165	0.028	0.170
3	0	0.108	0.132	0.168	0.211	0.212	0.018	0.144	0.003	0.015	0.193
4	0	0	0.182	0.137	0.157	0.277	0.096	0.133	0.089	0.079	0.279
5	0	0	0	0.199	0.11	0.108	0.068	0.111	0.027	0.076	0.210
6	0	0	0	0	0.124	0.044	0.206	0.105	0.123	0.108	0.299
7	0	0	0	0	0	0	0.221	0.105	0.115	0.169	0.295
8	0	0	0	0	0	0	0.076	0.071	0.01	0.056	0.127
9	0	0	0	0	0	0	0	0.038	0.132	0.129	0.181
10	0	0	0	0	0	0	0	0	0.108	0.129	0.193
11	0	0	0	0	0	0	0	0	0	0.057	0.057
12	0	0	0	0	0	0	0	0	0	0	0

Table A.10 Data set 3 (WCLS)

Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.134	0.141	0.061	0.029	0.051	0.020	0.065	0.021	0.042	0.008	0.428
2	0.076	0.128	0.101	0.046	0.079	0.113	0.018	0.081	0.042	0.027	0.289
3	0	0.014	0.111	0.107	0.157	0.245	0.001	0.150	0.031	0.019	0.165
4	0	0	0.162	0.105	0.095	0.187	0.006	0.117	0.023	0.018	0.288
5	0	0	0	0.375	0.060	0.103	0.034	0.087	0.015	0.030	0.297
6	0	0	0	0	0.076	0.036	0.214	0.068	0.105	0.113	0.389
7	0	0	0	0	0	0.000	0.345	0.069	0.064	0.217	0.306
8	0	0	0	0	0	0	0.066	0.098	0.014	0.012	0.810
9	0	0	0	0	0	0	0	0.028	0.099	0.065	0.808
10	0	0	0	0	0	0	0	0	0.130	0.196	0.674
11	0	0	0	0	0	0	0	0	0	0.008	0.992
12	0	0	0	0	0	0	0	0	0	0	1.000

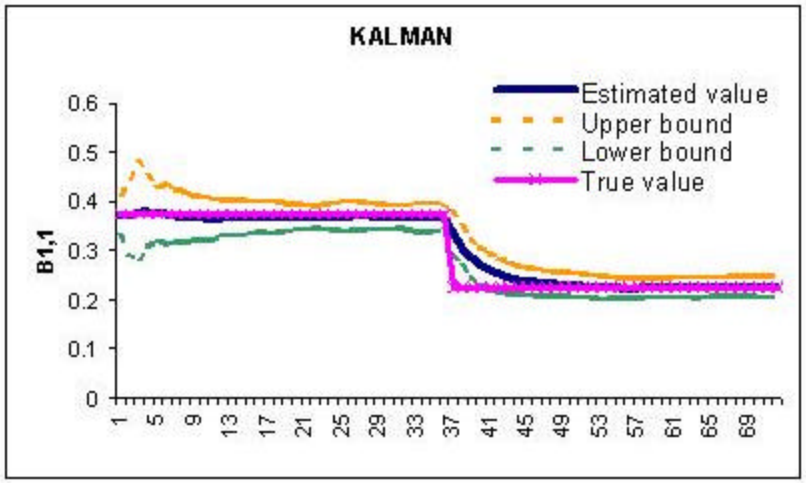
S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.010	0.011	0.013	0.012	0.028	0.011	0.018	0.020	0.020	0.011	0.041
2	0.118	0.169	0.105	0.064	0.119	0.120	0.065	0.115	0.082	0.065	0.264
3	0	0.045	0.121	0.109	0.169	0.189	0.005	0.149	0.089	0.041	0.204
4	0	0	0.173	0.160	0.102	0.176	0.018	0.147	0.045	0.047	0.244
5	0	0	0	0.194	0.104	0.098	0.048	0.111	0.029	0.053	0.215
6	0	0	0	0	0.108	0.043	0.124	0.092	0.121	0.131	0.218
7	0	0	0	0	0	0.000	0.166	0.092	0.084	0.118	0.208
8	0	0	0	0	0	0	0.094	0.103	0.037	0.036	0.172
9	0	0	0	0	0	0	0	0.048	0.127	0.082	0.152
10	0	0	0	0	0	0	0	0	0.101	0.103	0.149
11	0	0	0	0	0	0	0	0	0	0.016	0.016
12	0	0	0	0	0	0	0	0	0	0	0.000

Table A.11 Data set 3 (CLS)

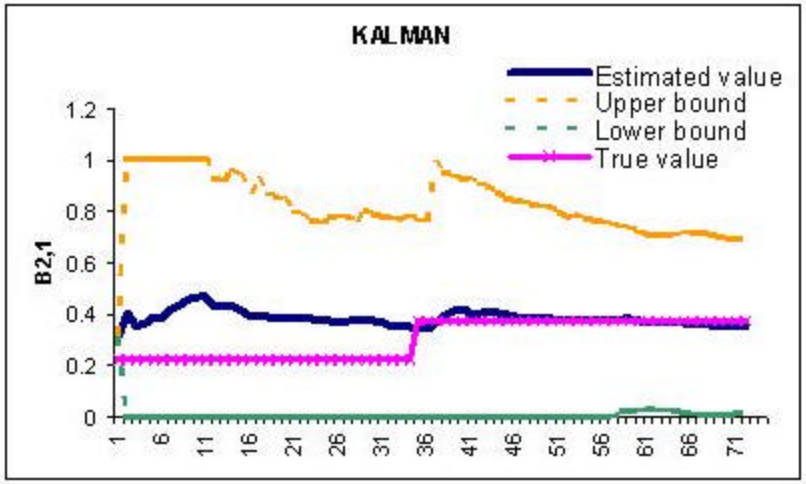
Mean	1	2	3	4	5	6	7	8	9	10	11
1	0.134	0.142	0.068	0.038	0.062	0.032	0.071	0.031	0.045	0.011	0.367
2	0.064	0.110	0.081	0.016	0.057	0.097	0.012	0.071	0.029	0.015	0.446
3	0	0.008	0.041	0.045	0.063	0.082	0.002	0.061	0.006	0.009	0.684
4	0	0	0.103	0.058	0.017	0.098	0.004	0.038	0.019	0.009	0.654
5	0	0	0	0.344	0.046	0.086	0.033	0.066	0.010	0.027	0.388
6	0	0	0	0	0.076	0.056	0.199	0.059	0.103	0.100	0.407
7	0	0	0	0	0	0.001	0.298	0.049	0.041	0.143	0.469
8	0	0	0	0	0	0	0.059	0.110	0.015	0.031	0.786
9	0	0	0	0	0	0	0	0.047	0.089	0.073	0.791
10	0	0	0	0	0	0	0	0	0.147	0.226	0.626
11	0	0	0	0	0	0	0	0	0	0.026	0.974
12	0	0	0	0	0	0	0	0	0	0	1.000

S.D.	1	2	3	4	5	6	7	8	9	10	11
1	0.010	0.011	0.013	0.014	0.028	0.013	0.016	0.019	0.021	0.013	0.067
2	0.108	0.158	0.120	0.033	0.090	0.123	0.039	0.101	0.092	0.037	0.282
3	0	0.022	0.071	0.070	0.133	0.111	0.008	0.122	0.025	0.030	0.312
4	0	0	0.187	0.141	0.051	0.154	0.019	0.073	0.065	0.027	0.439
5	0	0	0	0.180	0.080	0.112	0.055	0.100	0.030	0.060	0.290
6	0	0	0	0	0.112	0.073	0.118	0.071	0.117	0.116	0.311
7	0	0	0	0	0	0.005	0.181	0.083	0.060	0.126	0.334
8	0	0	0	0	0	0	0.081	0.158	0.042	0.066	0.263
9	0	0	0	0	0	0	0	0.083	0.161	0.136	0.336
10	0	0	0	0	0	0	0	0	0.139	0.122	0.247
11	0	0	0	0	0	0	0	0	0	0.096	0.096
12	0	0	0	0	0	0	0	0	0	0	0.000

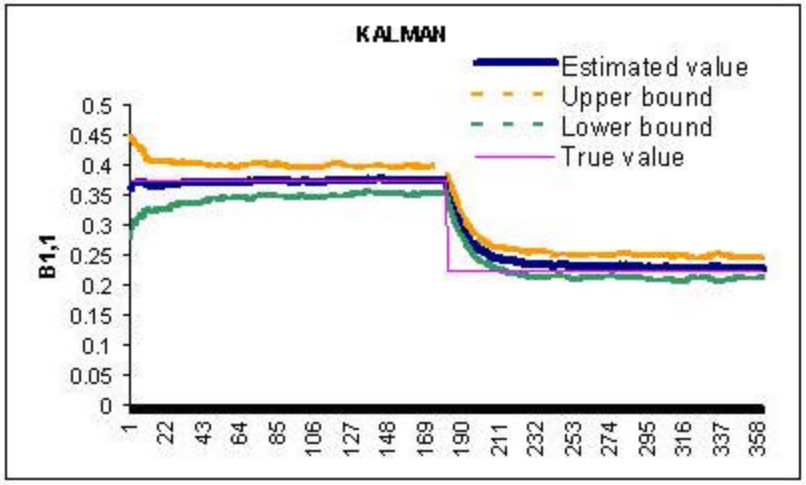
APPENDIX B ON-LINE ESTIMATION RESULTS



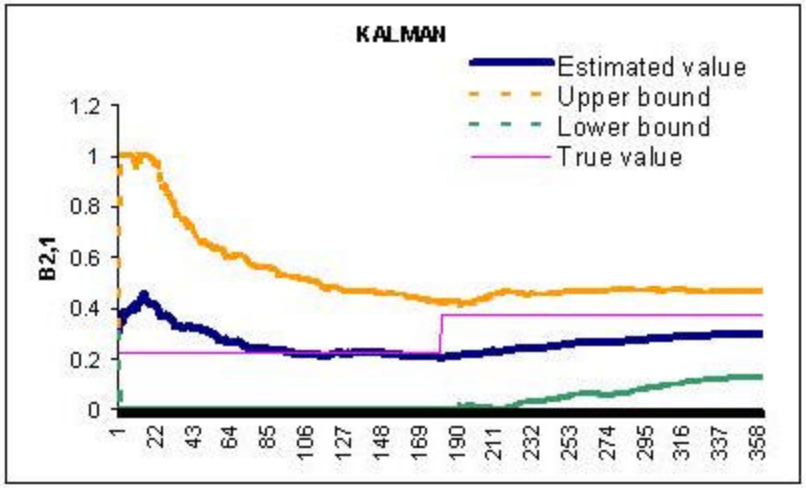
(a) Time-series plot of $b_{1,1}$ for data set 1



(b) Time-series plot of $b_{2,1}$ for data set 1

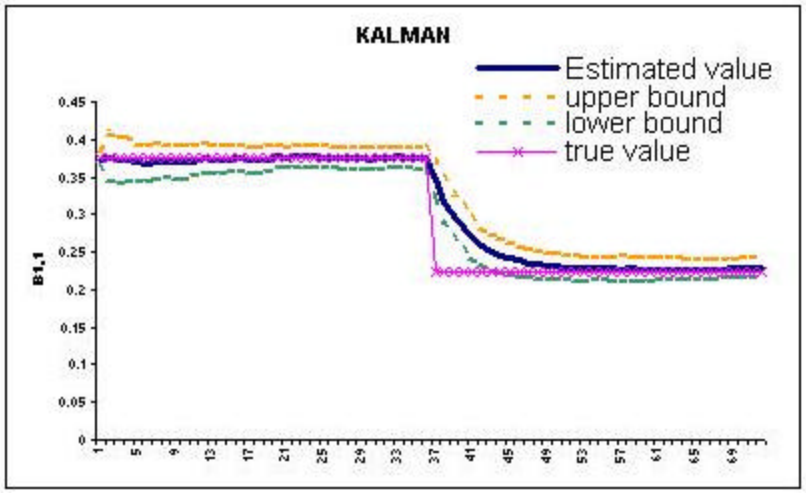


(c) Time-series plot of $b_{1,1}$ for data set 3

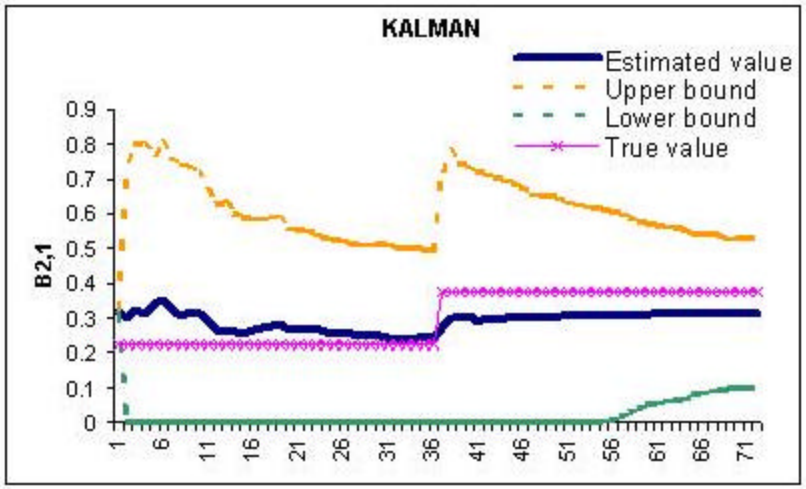


(d) Time-series plot of $b_{2,1}$ for data set 3

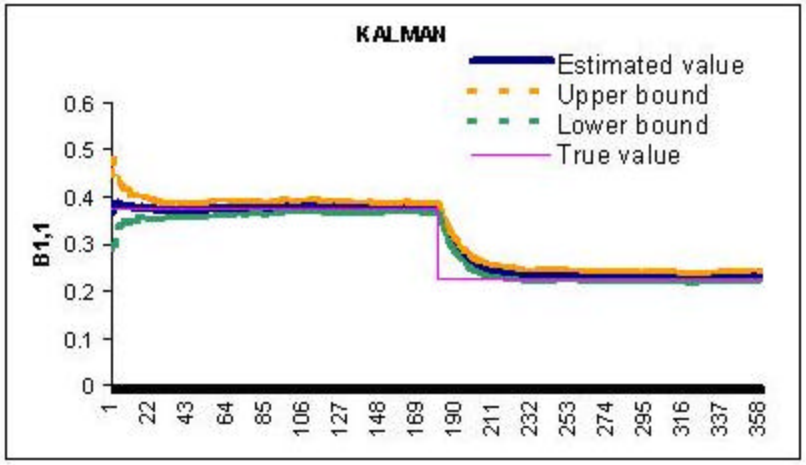
Figure B.1 Results of RLS for the simulated network with data sets 1 and 3



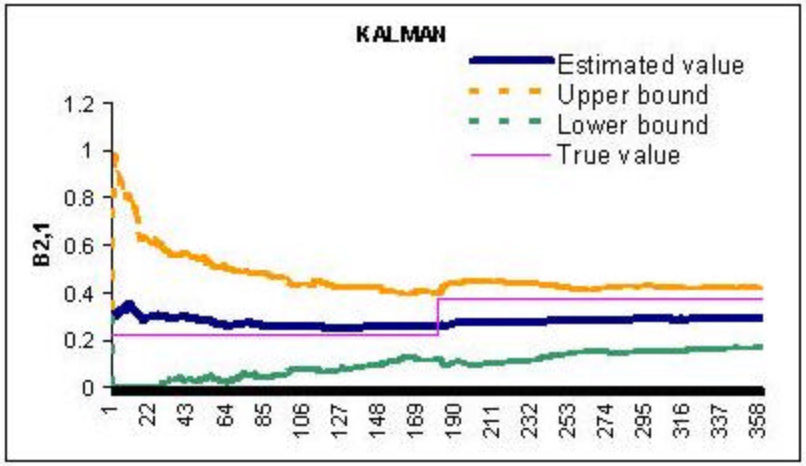
(a) Time-series plot of $b_{1,1}$ for data set 2



(b) Time-series plot of $b_{2,1}$ for data set 2

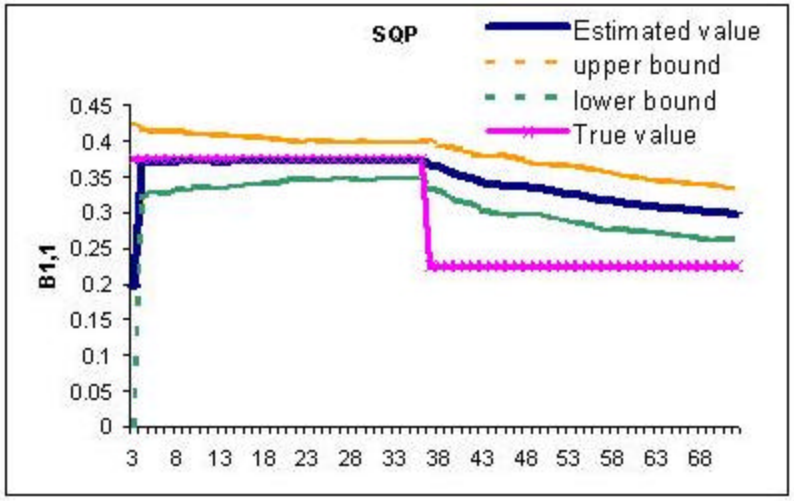


(c) Time-series plot of $b_{1,1}$ for data set 4

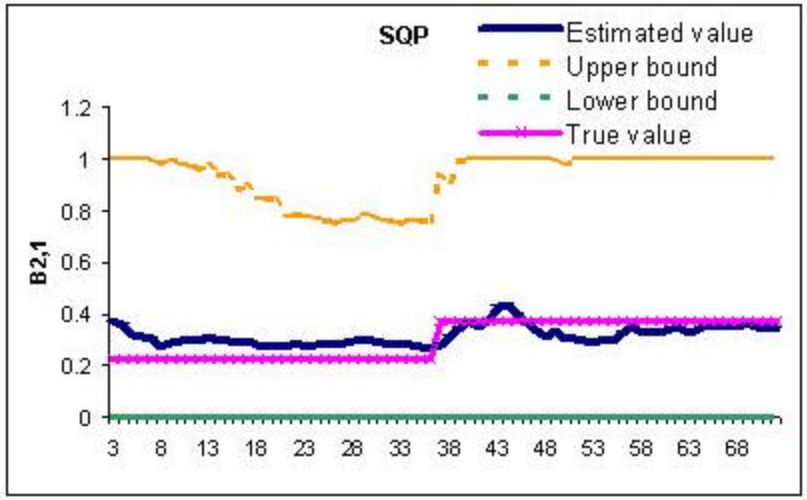


(d) Time-series plot of $b_{2,1}$ for data set 4

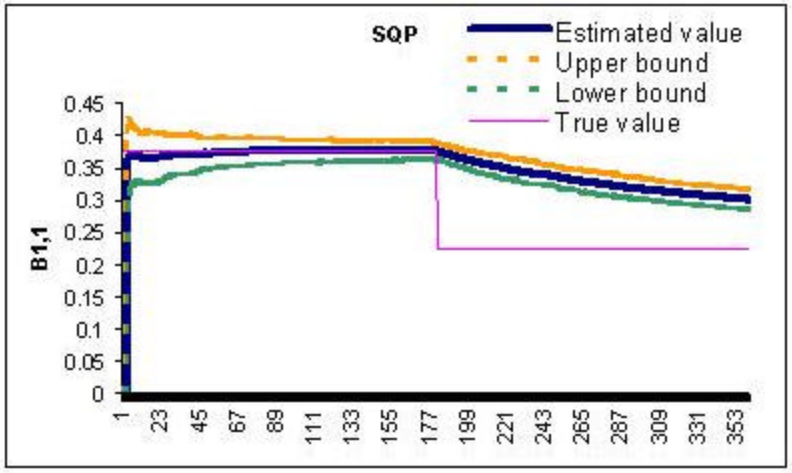
Figure B.2 Results of RLS for the simulated network with data sets 2 and 4



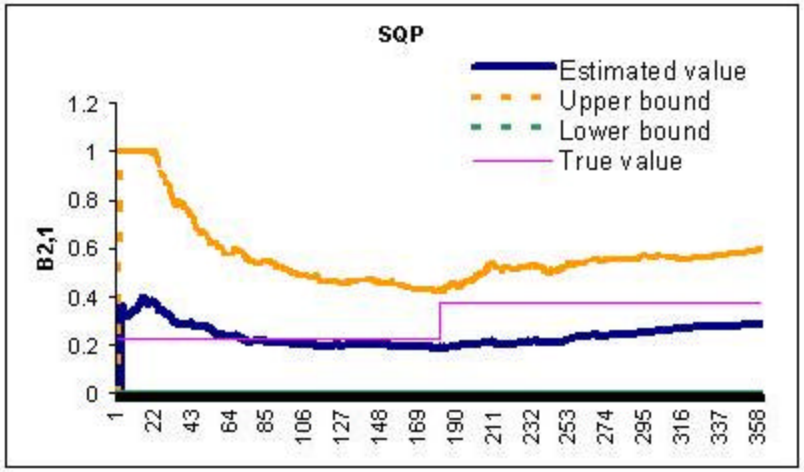
(a) Time-series plot of $b_{1,1}$ for data set 1



(b) Time-series plot of $b_{2,1}$ for data set 1

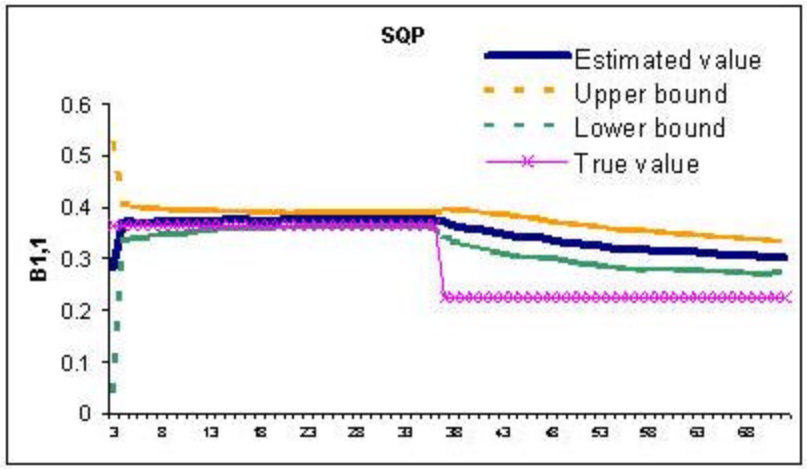


(c) Time-series plot of $b_{1,1}$ for data set 3

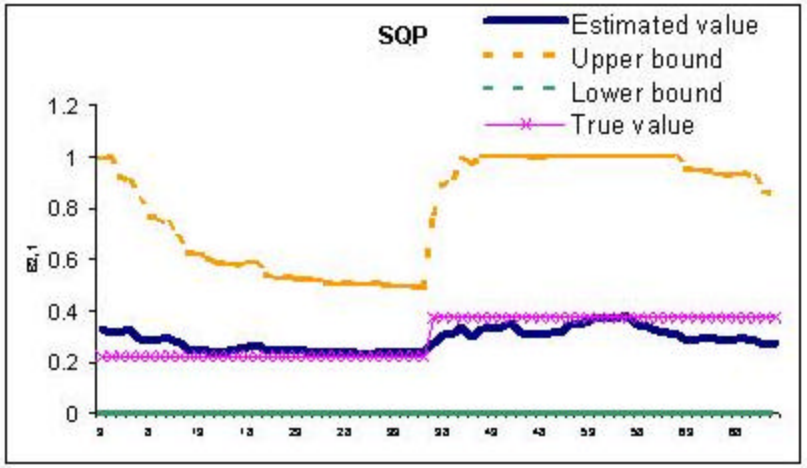


(d) Time-series plot of $b_{2,1}$ for data set 3

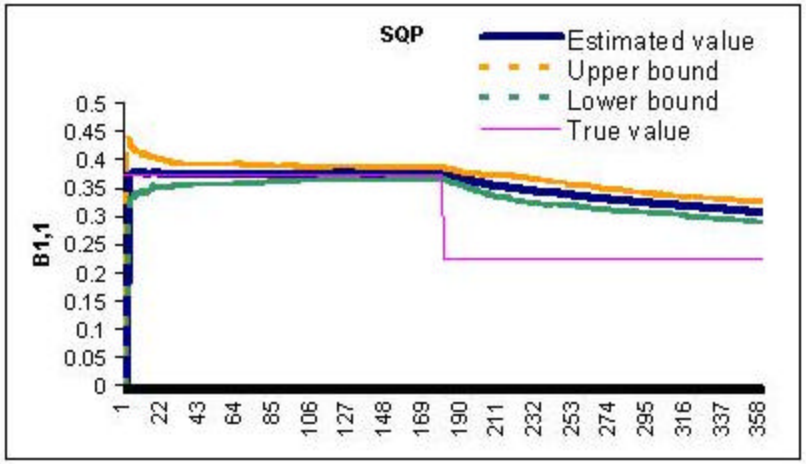
Figure B.3 Results of SQP for the simulated network with data sets 1 and 3



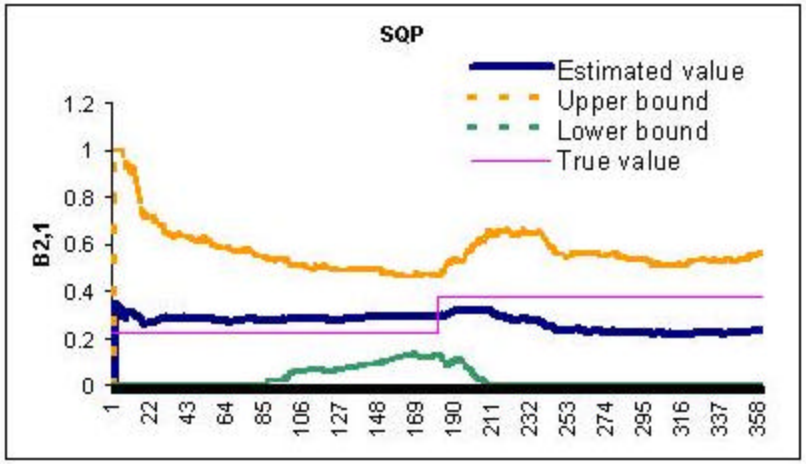
(a) Time-series plot of $b_{1,1}$ for data set 2



(b) Time-series plot of $b_{2,1}$ for data set 2



(c) Time-series plot of $b_{1,1}$ for data set 4



(d) Time-series plot of $b_{2,1}$ for data set 4

Figure B.4 Results of SQP for the simulated network with data sets 2 and 4

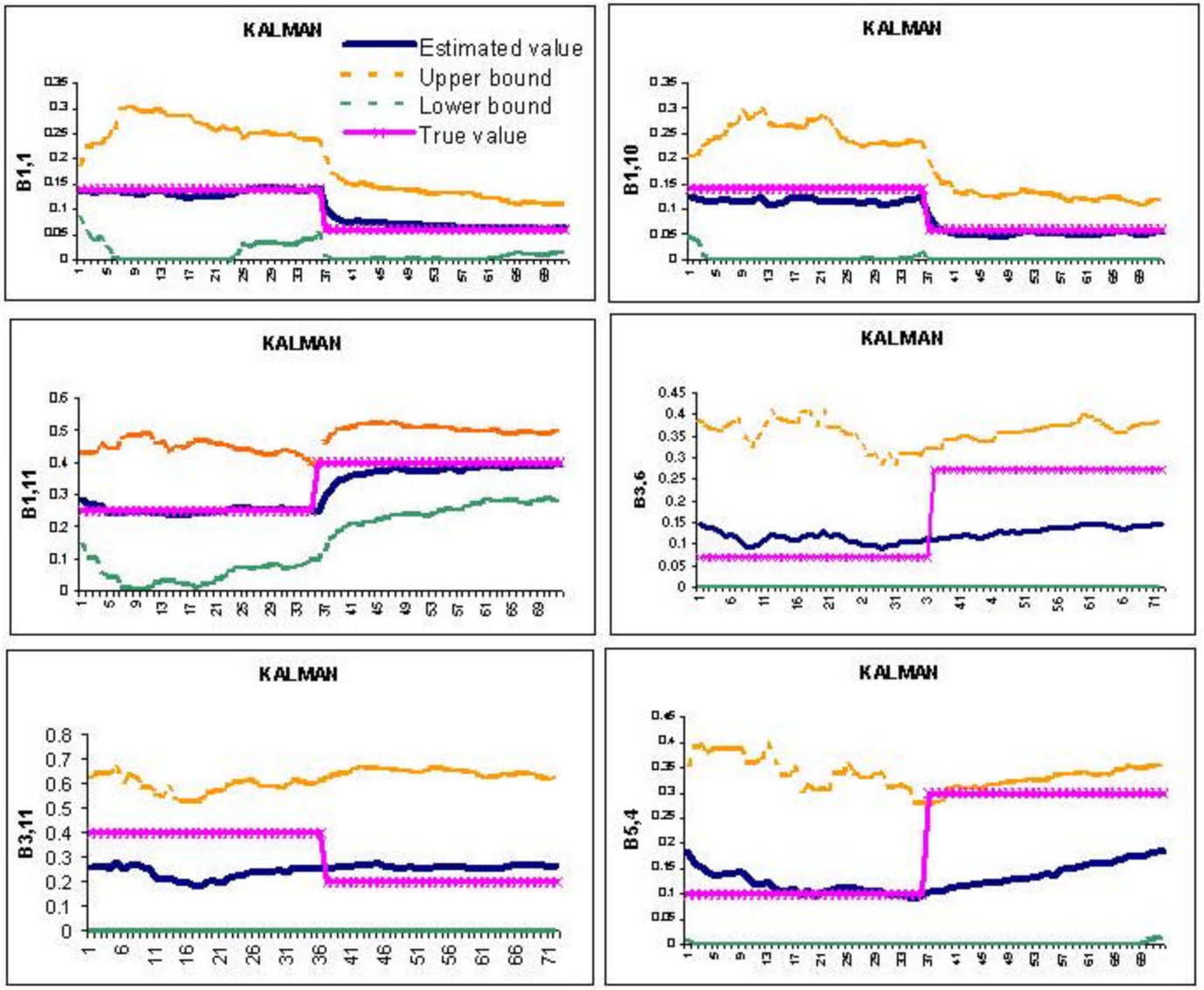


Figure B.5 Results of RLS for TH-169 with data set 1 (1)

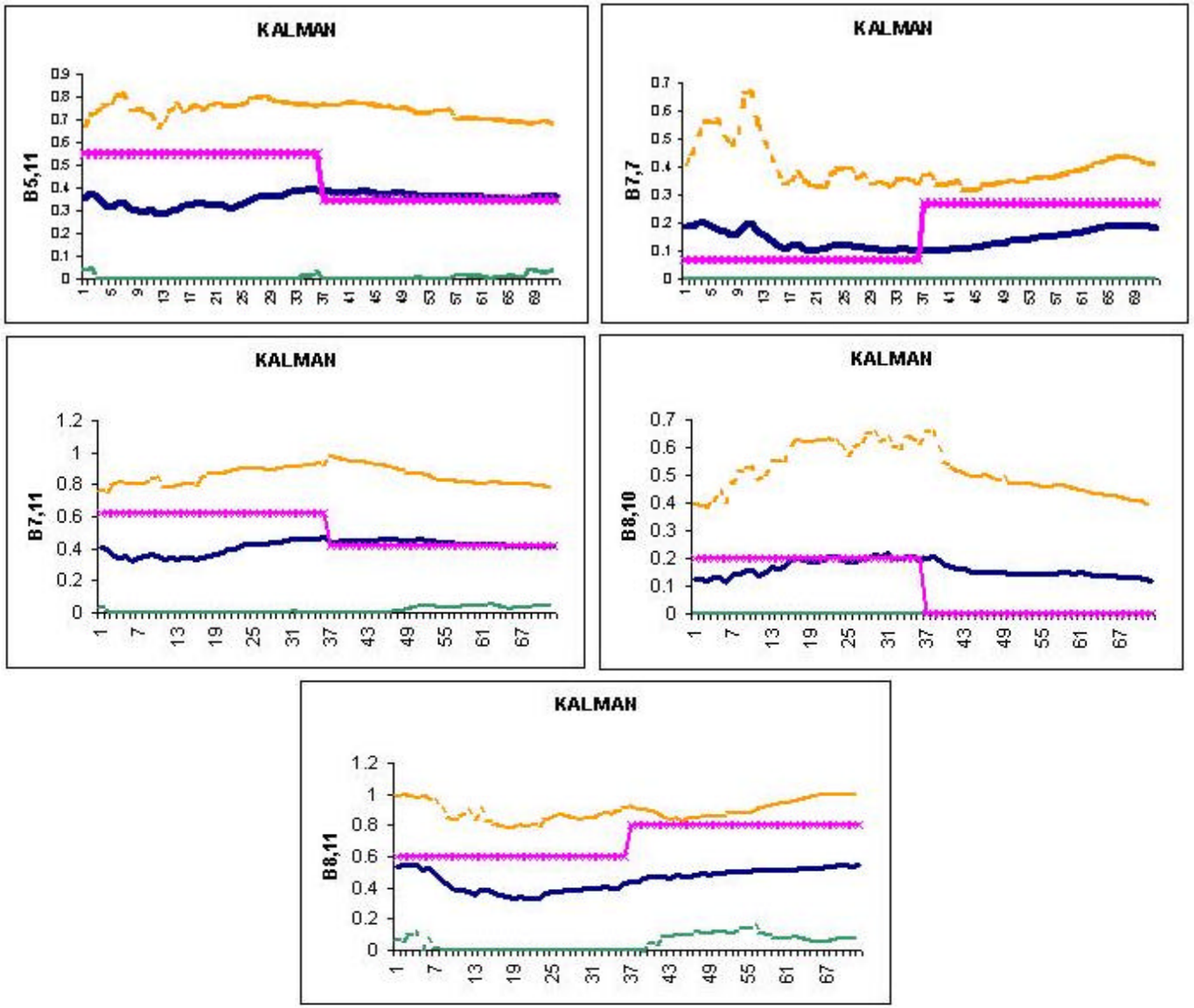


Figure B.6 Results of RLS for TH-169 with data set 1 (2)

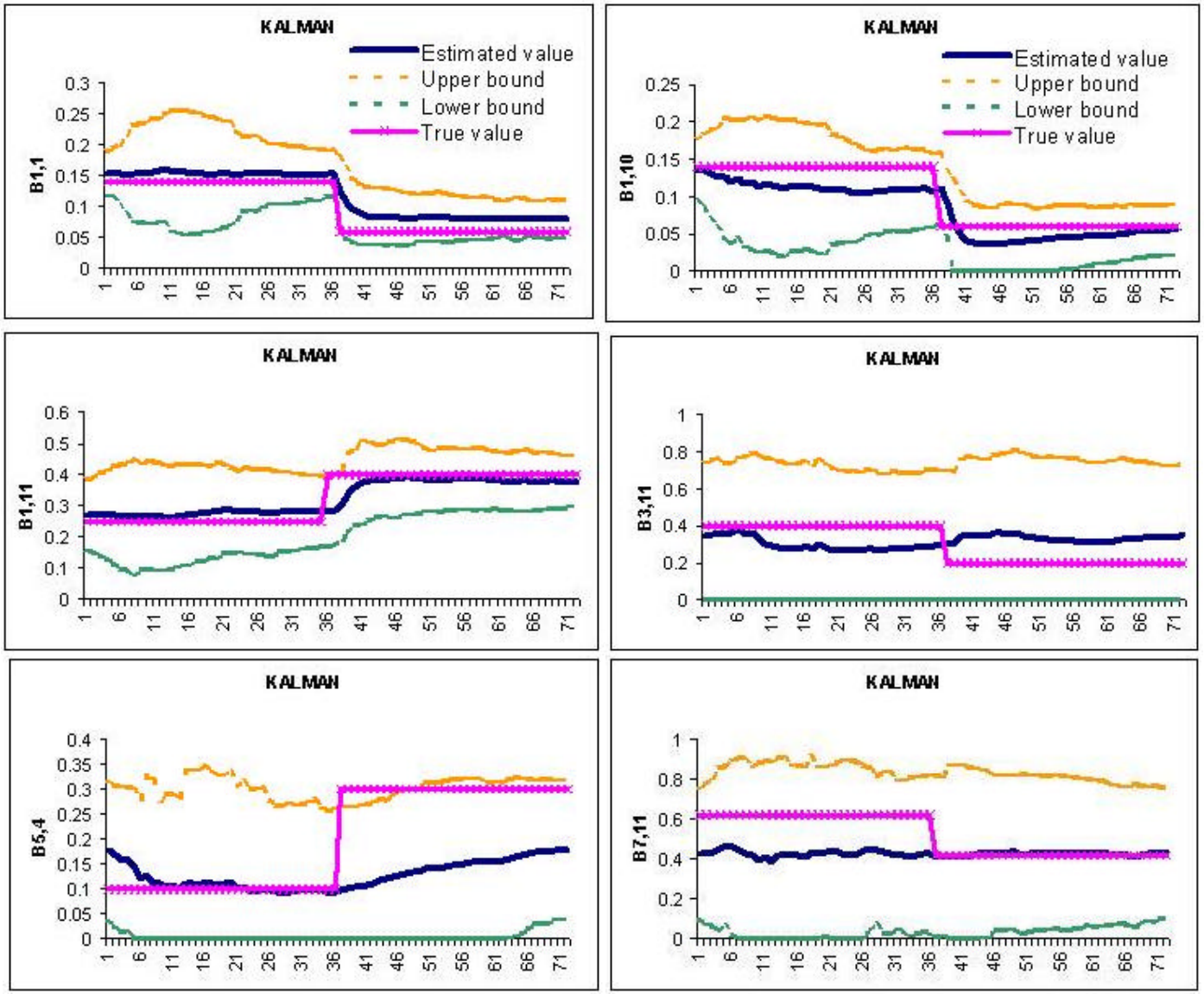


Figure B.7 Results of RLS for TH-169 with data set 2

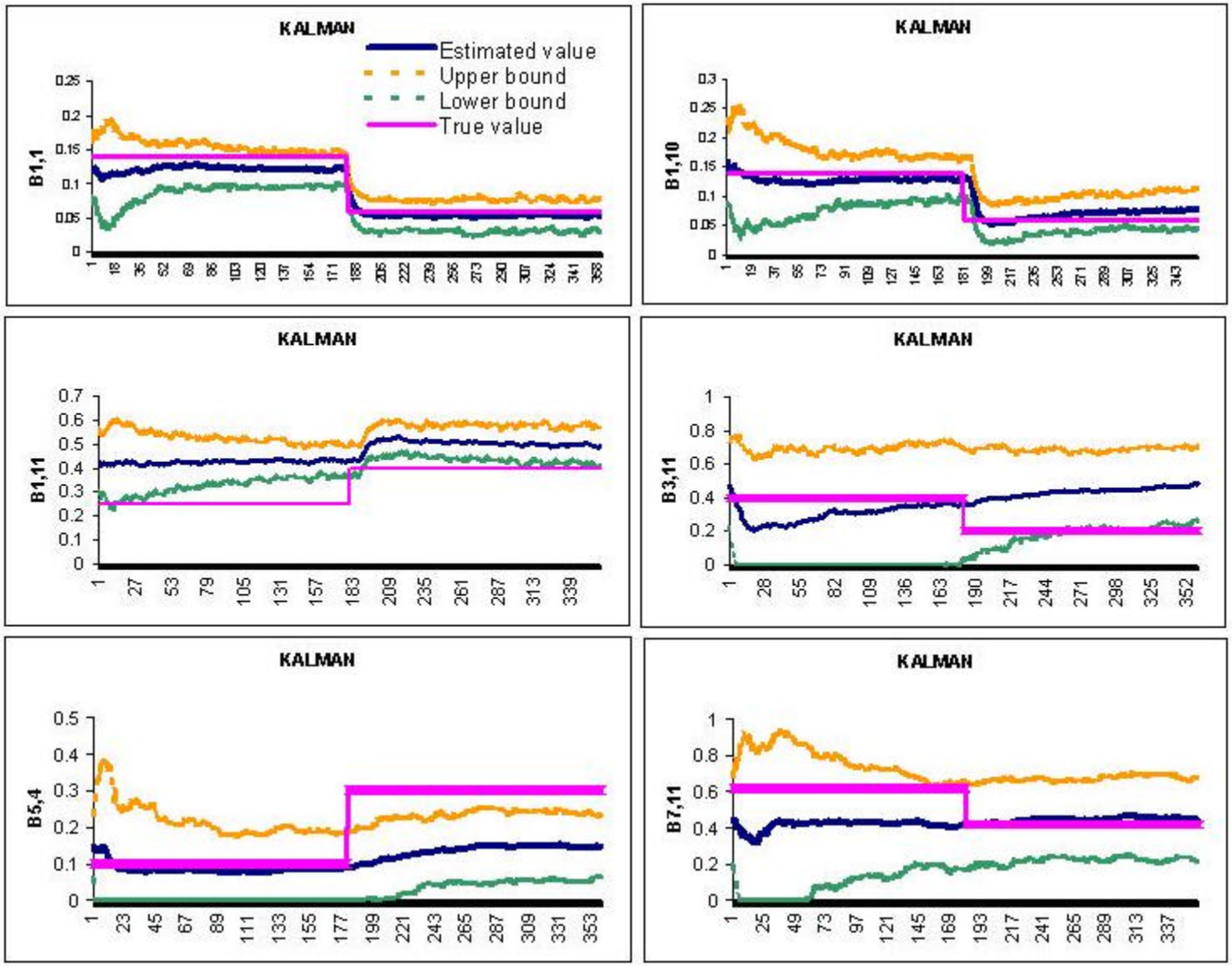


Figure B.8 Results of RLS for TH-169 with data set 4

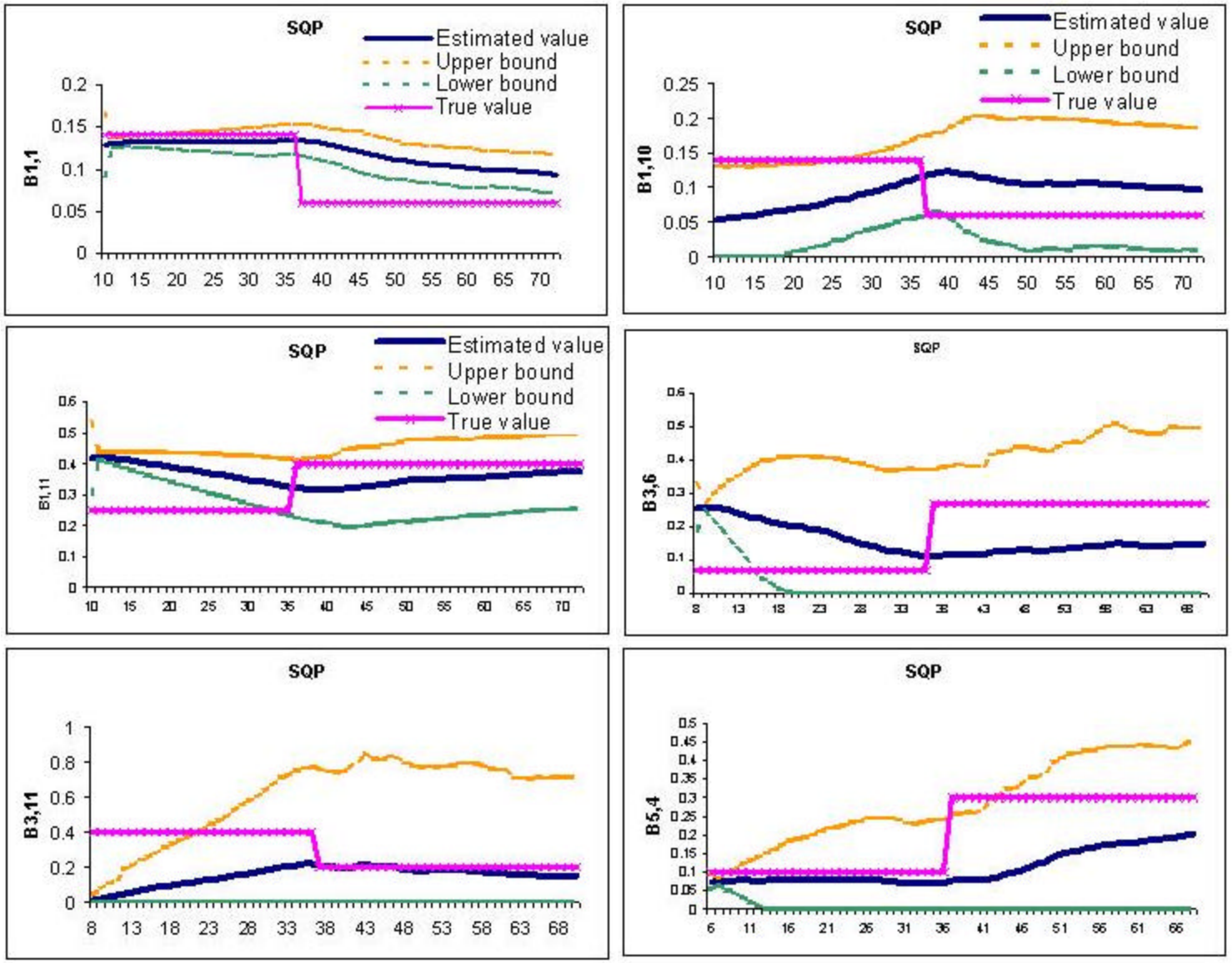


Figure B.9 Results of SQP for TH-169 on data set 1 (1)

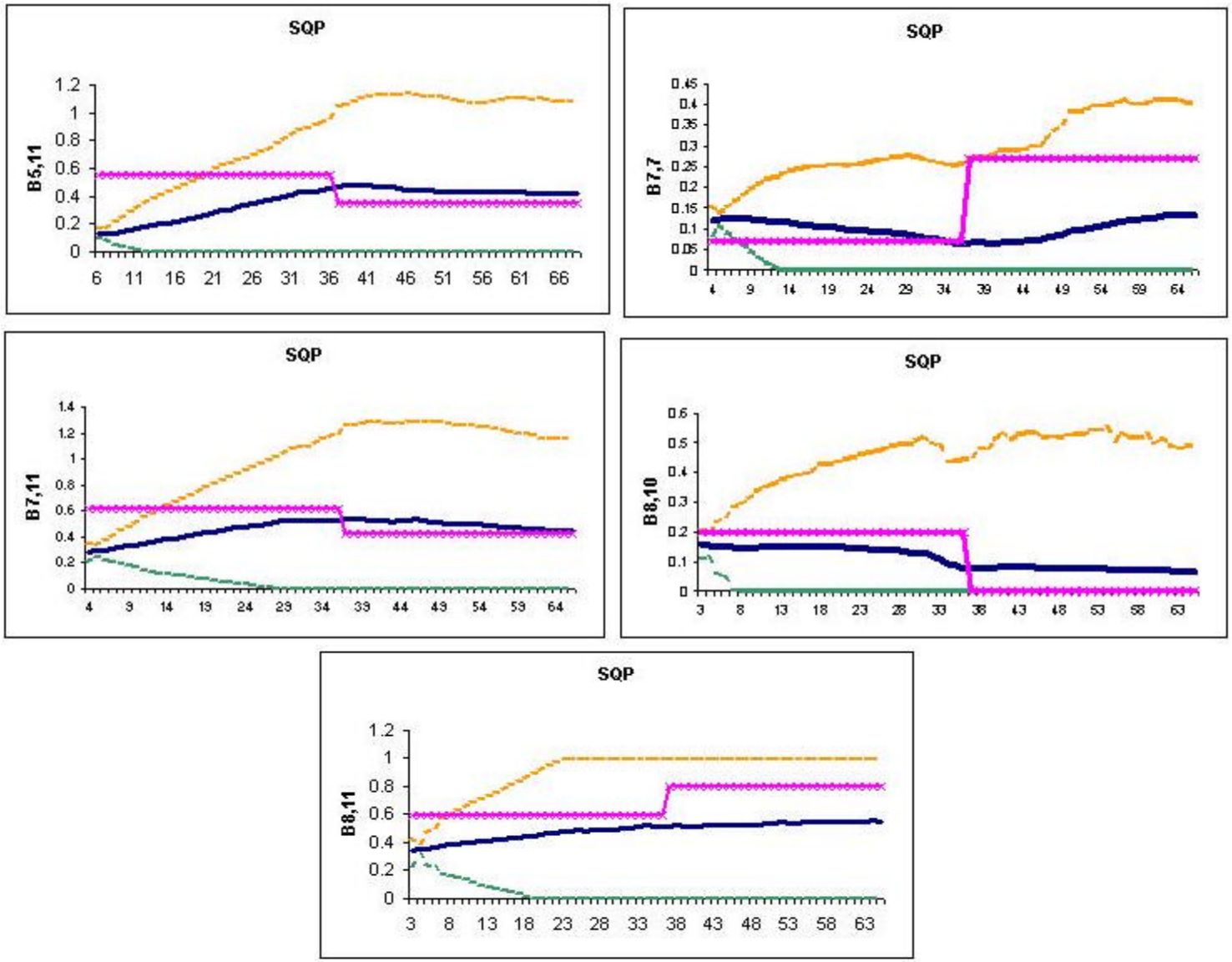


Figure B.10 Results of SQP for TH-169 on data set 1 (2)

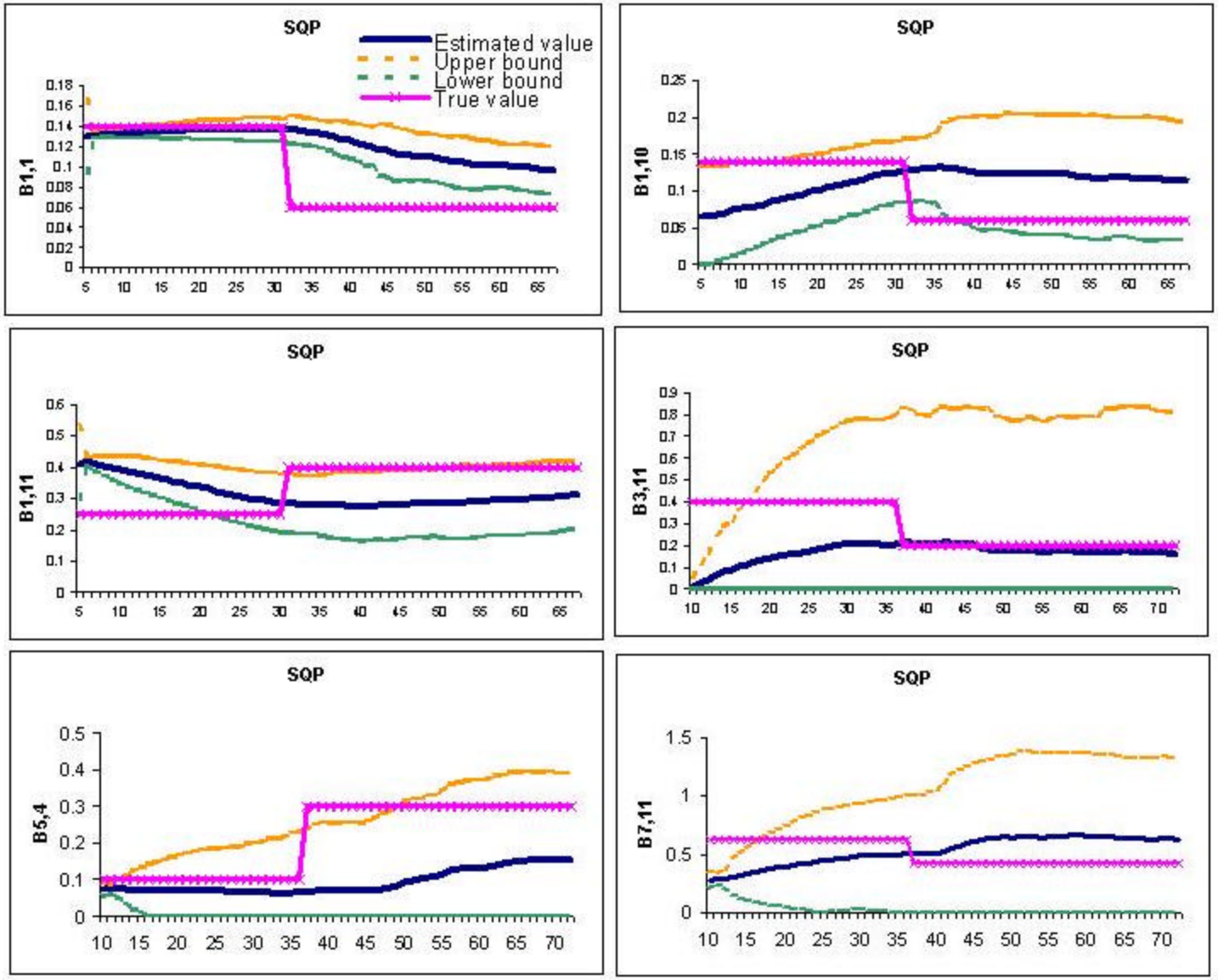


Figure B.11 Results for SQP on TH-169 with data set 2

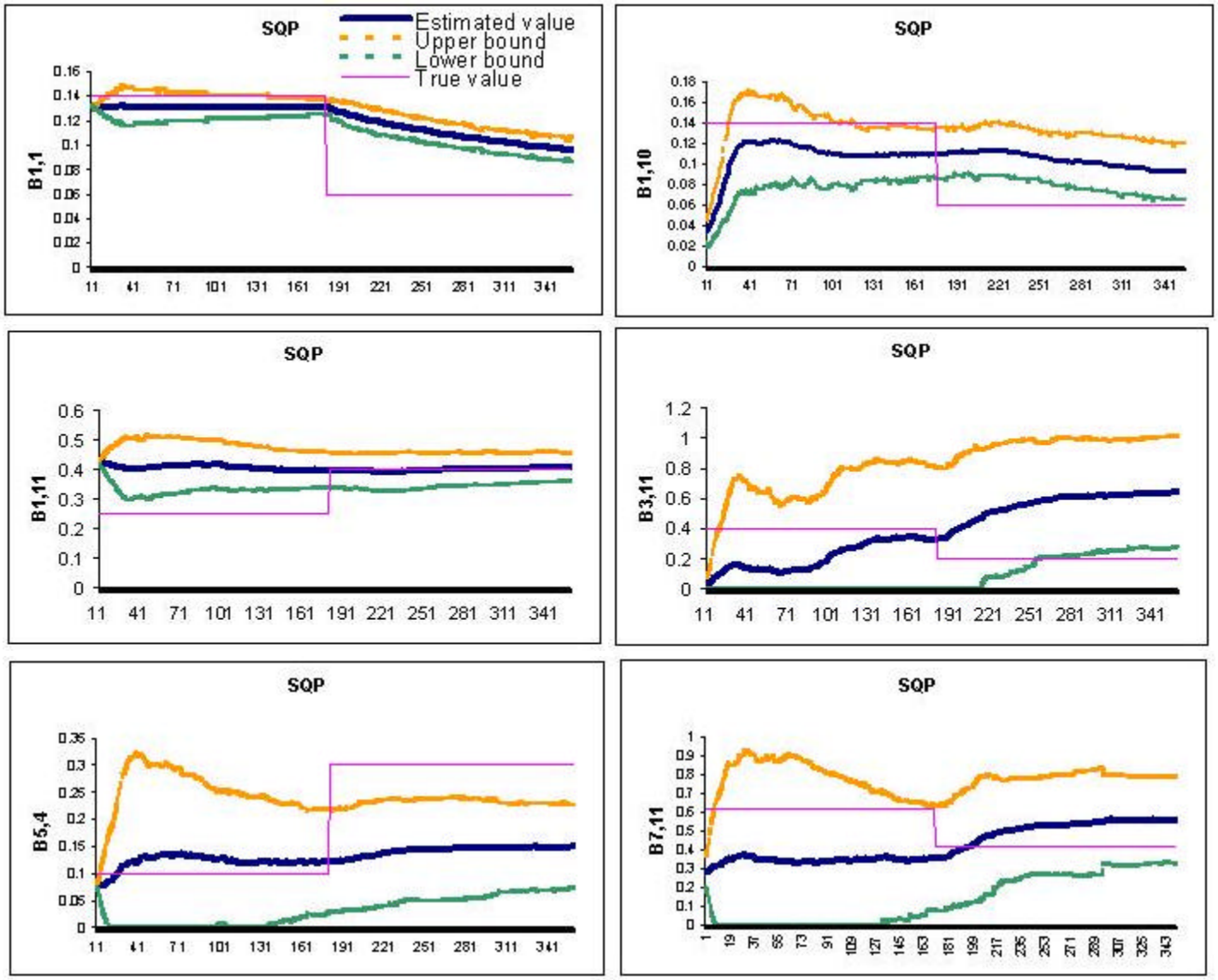


Figure B.12 Results for SQP on TH-169 with data set 4

APPENDIX C SOURCE CODE

A. CLS Matlab Script

```
clear;
load 'Q.dat';
load 'D.dat';
load 'aeq.dat';
load 'beq.dat';
load 'lb.dat';
load 'ub.dat';
load 'ini.dat';
ntime=36;
nor=12;
ndes=11;
nday=23;
for i=1:nor*ndes
  sumx(i)=0;
  sumxx(i)=0;
end

  fid1=fopen('X.dat','w');
  for t=1:nday
    for i=1:ntime*ndes
      for j=1:nor*ndes
        onr(i,j)=0;
      end
    end

   for k=1:ntime
      n=1;
      for i=(k-1)*ndes+1:k*ndes
        m=1;
        for j=(n-1)*nor+1:n*nor
          onr(i,j)=Q(k+(t-1)*ntime,m);
          m=m+1;
        end
        n=n+1;
      end
    end
        for i=1:ntime*ndes
          off(i,1)=D(i+(t-1)*ntime*ndes,1);
        end

        for j=1:ndes
          sumoff(j)=0;
          average(j)=0;
          for i=1:ntime
            sumoff(j)=sumoff(j)+off((i-1)*ndes+j);
          end
          average(j)=sumoff(j)/ntime;
        end

        for i=1:ndes
```

```
            for j=1:ntime
              weight((j-1)*ndes+i,(j-1)*ndes+i)=1/sqrt(average(i));
            end
          end

         on=onr;
         y=off;
A=[];
b=[];
[nineqcstr,numberOfVariables]=size(A);
[neqcstr,numberOfVariableseq]=size(aeq);
ncstr = nineqcstr + neqcstr;
X0=ini;
verbosity = 1;
caller = 'lsqlin';

[x,lambdaqp,exitflag,output]= ...
  qpsub(full(on),y,[full(aeq);full(A)],[beq;b],lb,ub,X0,neqcstr,verbosity,caller,ncstr,numberOfVariables);

for i=1:nor
   fprintf(fid1,'%6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f
%6.3f%6.3f\n',x(i),x(i+nor),x(i+nor*2),x(i+nor*3),x(i+nor*4),x(i+nor*5),x(i+nor*6),x(i+nor*7),x(i+nor*8),x(i
+nor*9),x(i+nor*10));
end

          for i=1:nor*ndes
            sumx(i)=sumx(i)+x(i);
            sumxx(i)=sumxx(i)+x(i).^2;
          end
  end

for i=1:132
  avg(i)=sumx(i)/nday;
  std(i)=sqrt((-nday*avg(i).^2+sumxx(i))/nday);
end
fprintf(fid1,'average \n');
for i=1:nor
  fprintf(fid1,'%6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f
%6.3f%6.3f\n',avg(i),avg(i+nor),avg(i+nor*2),avg(i+nor*3),avg(i+nor*4),avg(i+nor*5),avg(i+nor*6),avg(i+nor
*7),avg(i+nor*8),avg(i+nor*9),avg(i+nor*10));
          end
fprintf(fid1,'std \n');
          for i=1:nor
            fprintf(fid1,'%6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f
%6.3f\n',std(i),std(i+nor),std(i+nor*2),std(i+nor*3),std(i+nor*4),std(i+nor*5),std(i+nor*6),std(i+nor*7),std(i+n
or*8),std(i+nor*9),std(i+nor*10));
          end
fclose(fid1);
```

Inputs:

$Q_{ntime*m}$ is the onramp counts in the matrix form, where ntime is number of time intervals and m is the number of origins;

$D_{1*(ntime*n)}$ is the offramp counts in a column vector form, where n is the number of destinations;

aeq is the matrix that specifies the equality constraints, for TH-169 segment

$$aeq = \underbrace{\begin{bmatrix} 1 & 0 & \Lambda & 0 & O & & & & 1 & 0 & \Lambda & 0 \\ 0 & 1 & & M & & O & & & 0 & 1 & & M \\ M & & O & 0 & & & O & & M & & O & 0 \\ 0 & \Lambda & 0 & 1 & & & & O & 0 & \Lambda & 0 & 1 \end{bmatrix}}_{mn} \Big\} m$$

beq is a vector showing the equality constraints.

$$beq = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \Big\} m$$

lb is the lower bound for the inequality constraints;

ub is the upper bound for the inequality constraints.

B. Fortran code of data generation for TCLS

```
 Program station
 implicit none
 integer num
 parameter(num=26)
 integer i,j,k,n,tor(12,11,37),triptable(37*11,12*11),tt
 double precision occu(num*288),speed(num*288),vol(num*288),detecto
&r(num*288,4),id(num*288),count(num*288),length(21),zero,staspeed(2
&5,288),statspeed(25,288),section(21,288),time(21,288),odtime(12,11
&,288),t(12,11,39),beta(12,11,37),onramp(3744,3),qq(12,39),q(12,11,
&37),offramp(2880,3),off(11,37),offramp1(864,3),predict(407,1),pred
&ictoff(11,37),det(num*23*288,4)

 character*80 detfile,outfile,lfile,onfile,offfile,offfile1,ofile
 parameter(detfile='station.dat')
 parameter(outfile='outs.dat')
 parameter(lfile='length.dat')
 parameter(onfile='onramp.dat')
 parameter(offfile='offramp.dat')
 open(unit=1,file=detfile,status='old')
 open(unit=2,file=outfile,status='old')
 open(unit=3,file=lfile,status='old')

 call fread(num*288*23,4,det,1)
 call fread(21,1,length,3)
 zero=1.0d-8

 do tt=1,23
   do i=1,num*288
      do j=1,4
         detector(i,j)=det(i+num*288*(tt-1),j)
      end do
   end do

 do i=1,num*288
    id(i)=detector(i,1)
    count(i)=detector(i,2)
    occu(i)=detector(i,3)
    speed(i)=detector(i,4)
 end do
 close(1)
 close(3)

 do i=1,num-1,2
    do j=1,288
```

```
      if(count((i-1)*288+j)+count(i*288+j).gt.zero) then
      staspeed(i,j)=(count((i-1)*288+j)*speed((i-1)*288+j)+count(i
&*288+j)*speed(i*288+j))/(count((i-1)*288+j)+count(i*288+j))
      else
         staspeed(i,j)=60*1.609/3.6
      endif
    end do
  end do

  do j=1,288
    statspeed(1,j)=staspeed(1,j)
    statspeed(3,j)=staspeed(25,j)
    statspeed(5,j)=staspeed(23,j)
    statspeed(7,j)=staspeed(21,j)
    statspeed(9,j)=staspeed(19,j)
    statspeed(11,j)=staspeed(17,j)
    statspeed(13,j)=staspeed(15,j)
    statspeed(15,j)=staspeed(3,j)
    statspeed(17,j)=staspeed(13,j)
    statspeed(19,j)=staspeed(11,j)
    statspeed(21,j)=staspeed(9,j)
    statspeed(23,j)=staspeed(7,j)
    statspeed(25,j)=staspeed(5,j)
  end do
  k=0
  do i=1,num-1,2
    k=k+1
    do j=1,288
      statspeed(k,j)=statspeed(i,j)
    end do
  end do

  do j=1,288
    section(1,j)=statspeed(1,j)
    section(2,j)=statspeed(1,j)
    section(3,j)=(statspeed(1,j)+statspeed(2,j))/2
    section(4,j)=statspeed(2,j)
    section(5,j)=statspeed(3,j)
    section(6,j)=statspeed(4,j)
    section(7,j)=statspeed(5,j)
    section(8,j)=statspeed(6,j)
    section(9,j)=(statspeed(6,j)+statspeed(7,j))/2
    section(10,j)=statspeed(7,j)
    section(11,j)=(statspeed(7,j)+statspeed(8,j))/2
    section(12,j)=statspeed(8,j)
```

```
  section(13,j)=(statspeed(8,j)+statspeed(9,j))/2
  section(14,j)=statspeed(9,j)
  section(15,j)=statspeed(10,j)
  section(16,j)=statspeed(11,j)
  section(17,j)=(statspeed(11,j)+statspeed(12,j))/2
  section(18,j)=(statspeed(11,j)+statspeed(12,j))/2
  section(19,j)=statspeed(12,j)
  section(20,j)=statspeed(13,j)
  section(21,j)=statspeed(13,j)
end do

do i=1,21
   do j=1,288
    time(i,j)=length(i)/section(i,j)
   end do
end do

do i=1,12
  do j=1,11
    do k=1,288
      odtime(i,j,k)=0
    end do
  end do
end do

do k=1,288
   odtime(1,1,k)=time(1,k)
   odtime(2,1,k)=time(1,k)
end do

do j=2,11
   do k=1,288
      odtime(1,j,k)=time(2*j-2,k)+time(2*j-1,k)+odtime(1,j-1,k)
      odtime(2,j,k)=time(2*j-2,k)+time(2*j-1,k)+odtime(2,j-1,k)
   end do
end do

do i=3,12
   do k=1,288
      odtime(i,i-1,k)=time(2*i-3,k)
   end do
end do

do i=3,12
   do j=i,11
      do k=1,288
```

```
            odtime(i,j,k)=time(2*j-1,k)+time(2*j-2,k)+odtime(i,j-1,k)
          end do
        end do
      end do

      do i=1,12
        do j=1,11
          n=1
         do k=1,288
          if((k.ge.82).and.(k.le.120)) then
            t(i,j,n)=odtime(i,j,k)
            n=n+1
          endif
         end do
        end do
      end do

      do i=1,12
        do j=1,11
         do k=3,39
          tor(i,j,k-2)=int(t(i,j,k)/300)+1
          beta(i,j,k-2)=tor(i,j,k-2)-t(i,j,k)/300
         end do
        end do
      end do
c
c     onramp counts
c
      open(unit=4,file=onfile,status='old')
      call fread(3744,3,onramp,4)
      k=1
        do j=1,288
          if((j.ge.82).and.(j.le.120)) then
            qq(1,k)=onramp(j,2)+onramp(j+288,2)
            k=k+1
          endif
        end do

        do i=2,12
          k=1
          do j=i*288+1,(i+1)*288
            if(((j-i*288).ge.82).and.((j-i*288).le.120)) then
              qq(i,k)=onramp(j,2)
             k=k+1
            endif
          end do
```

```
        end do

c
c     calculate the matrix of onramp counts for time-varying CLS
c
      do i=1,12
        do j=1,11
          do k=1,37
            q(i,j,k)=(1-beta(i,j,k))*qq(i,(k+2-tor(i,j,k)))+beta(i,j,k
     &)*qq(i,(k+2-tor(i,j,k)+1))
          end do
        end do
      end do

      do k=1,37
        do j=1,11
          do i=1,12
            triptable((k-1)*11+j,(j-1)*12+i)=int(q(i,j,k))
          end do
        end do
      end do
c
c     offramp counts
c
         do i=1,10
           k=1
           do j=(i-1)*288+1,i*288
             if(((j-(i-1)*288).ge.84).and.((j-(i-1)*288).le.120)) then
               off(i,k)=offramp(j,2)
              k=k+1
             endif
           end do
         end do
c
c     calculate the vector of offramp counts for time-varying CLS
c
      open(unit=6,file=offfile1,status='old')
      call fread(864,3,offramp1,6)
      k=1
      do j=1,288
           if((j.ge.84).and.(j.le.120)) then
             off(11,k)=offramp1(j,2)+offramp1(j+288,2)+offramp1(j+288
     &*2,2)
             k=k+1
           endif
      end do
```

```
do i=2,37
  write(2,"(36f6.0)") (off(j,i),j=1,11)
end do
end do
end

subroutine fread(row,col,matrix,fnum)
integer row,col,fnum
double precision matrix(row,col),xirtam(col,row)
read(fnum,*) xirtam
call transp(xirtam,col,row,matrix)
end

subroutine transp(x,col,row,y)
integer row,col,i,j
double precision x(col,row), y(row,col)
do  i=1,row
   do  j=1,col
      y(i,j) = x(j,i)
end do
end do
end
```

C.Fortran code for Recursive Least Squares via Kalman Filtering

```
 Program Kalman
 implicit none
 integer i,j,ii,jj,s,t,nor,ntime,ndes,kk,m,n,ttt,nday
 parameter (nor=12,ndes=11,ntime=36,nday=50)
 integer index(nor)
 double precision onramp(ntime,nor),offramp(ntime,ndes)
 double precision a(ndes,ndes),aa(ndes,ndes),c(ndes,1),cc(ndes,1),f
&tol,R(nor*ndes,nor*ndes),sum(ndes)
 double precision b(nor,ndes),k(nor*ndes,ndes),p(nor*ndes,nor*ndes)
 double precision bb(nor*ndes,1),bbt(1,nor*ndes),q(nor*ndes,ndes),q
&t(ndes,nor*ndes),p0q(nor*ndes,ndes),qtp0(ndes,nor*ndes),pq(ndes,nd
&es),rr(nor*ndes,nor*ndes),kc(nor*ndes,1),y(ntime,ndes)
 double precision onra(ntime*ndes,nor*ndes),ft(nor,nor*ndes)
 double precision kplus(nor*ndes,nor),f(nor*ndes,nor),ftp(nor,nor*n
&des),ftpf(nor,nor),fpp(nor*ndes,nor),pp(nor,nor),rrplus(nor*ndes,n
&or*ndes),ftbb(nor,1),kcplus(nor*ndes,1),onrampt(ntime*nday,nor),of
&frampt(ntime*nday,ndes),average(nor,ndes,ntime),std(nor,ndes,ntime
&),sumbb(nor,ndes,ntime),sumsquarebb(nor,ndes,ntime),offrampp(ntime
&,ndes),sumerror(nday)
 character (40)  offile,onfile,outfile,inifile,eqfile
 parameter       (outfile='out.dat')
 parameter       (inifile='matrix169.dat')
 parameter       (offile='offramp.dat')
 parameter       (onfile='onramp.dat')
 parameter       (eqfile='aeq169.dat')

 open(unit=3,file=outfile,status='old')
 open(unit=1,file=offile,status='old')
 open(unit=2,file=onfile,status='old')
 open(unit=5,file=eqfile,status='old')
 call fread(ntime*nday,nor,onrampt,2)
 call fread(ntime*nday,ndes,offrampt,1)
 call fread(nor,nor*ndes,ft,5)
 close (2)
 close (1)
 close(5)

 do i=1,nor
   do j=1,ndes
     do t=1,ntime
       sumbb(i,j,t)=0.d+0
       sumsquarebb(i,j,t)=0.d+0
     end do
    end do
```

```
end do
ftol=1.d-8
do i=1,nor*ndes
    do j=1,nor
      f(i,j)=ft(j,i)
    end do
end do

do ttt=1,nday
open(unit=4,file=inifile,status='old')
call fread(nor,ndes,b,4)
close(4)
   do i=1,ntime
      do j=1,nor
        onramp(i,j)=onrampt(ntime*(ttt-1)+i,j)
      end do
      do j=1,ndes
        offramp(i,j)=offrampt(ntime*(ttt-1)+i,j)
      end do
    end do

do i=1,ntime*ndes
   do j=1,nor*ndes
     onra(i,j)=0.d+0
   end do
end do

do kk=1,ntime
   n=1
   do i=(kk-1)*ndes+1,kk*ndes
      m=1
      do j=(n-1)*nor+1,n*nor
        onra(i,j)=onramp(kk,m)
        m=m+1
      end do
      n=n+1
   end do
end do

do i=1,ntime
   do j=1,ndes
     y(i,j)=offramp(i,j)
   end do
end do
do t=1,ntime
   do i=1,ndes
```

```
          do j=1,nor*ndes
            q(j,i)=onra((t-1)*ndes+i,j)
            qt(i,j)=onra((t-1)*ndes+i,j)
          end do
        end do

          do i=1,nor
            bb(i+(j-1)*nor,1)=b(i,j)
            bbt(1,i+(j-1)*nor)=b(i,j)
          end do
c initial p

      if(t.eq.1) then
      do ii=1,nor*ndes
        do jj=1,nor*ndes
          p(ii,jj)=0.
          r(ii,jj)=0.
          if(ii.eq.jj) p(ii,jj)=1.d+0
          if(ii.eq.jj) r(ii,jj)=1.d-4
        end do
      end do
      endif

          call multiply(p,q,p0q,nor*ndes,nor*ndes,ndes)
          call multiply(qt,p,qtp0,ndes,nor*ndes,nor*ndes)
          call multiply(qtp0,q,pq,ndes,nor*ndes,ndes)
          do i=1,ndes
            do j=1,ndes
              if(i.eq.j) then
               a(i,j)=pq(i,j)+1.d+0
              else
              a(i,j)=pq(i,j)
              endif
            end do
          end do

          call inv(ndes,a,aa)
          call multiply(p0q,aa,k,nor*ndes,ndes,ndes)
          call multiply(k,qtp0,rr,nor*ndes,ndes,nor*ndes)
          do i=1,nor*ndes
            do j=1,nor*ndes
              p(i,j)=p(i,j)-rr(i,j)+r(i,j)
            end do
          end do

          call multiply(qt,bb,cc,ndes,nor*ndes,1)
```

```
do j=1,ndes
 c(j,1)=y(t,j)-cc(j,1)
end do

call multiply(k,c,kc,nor*ndes,ndes,1)
 do i=1,nor*ndes
    bb(i,1)=bb(i,1)+kc(i,1)
 end do

call multiply(ft,p,ftp,nor,nor*ndes,nor*ndes)
call multiply(ftp,f,ftpf,nor,ndes*nor,nor)
call INV(nor,ftpf,pp)
call multiply(f,pp,fpp,nor*ndes,nor,nor)
call multiply(p,fpp,kplus,nor*ndes,nor*ndes,nor)
call multiply(kplus,ftp,rrplus,nor*ndes,nor,nor*ndes)
call multiply(ft,bb,ftbb,nor,nor*ndes,1)
do i=1,nor
   ftbb(i,1)=1.d+0-ftbb(i,1)
end do

call multiply(kplus,ftbb,kcplus,nor*ndes,nor,1)
  do i=1,nor*ndes
     bb(i,1)=bb(i,1)+kcplus(i,1)
 end do

do i=1,nor*ndes
   do j=1,nor*ndes
      p(i,j)=p(i,j)-rrplus(i,j)
   end do
end do

 do j=1,ndes
    do i=1,j+1
       b(i,j)=bb((j-1)*nor+i,1)
       if(b(i,j).lt.(0.d+0)) b(i,j)=0.d+0
       if(b(i,j).gt.1.) b(i,j)=1.
    end do
 end do

do i=1,nor
sum(i)=0.
   do j=1,ndes
     sum(i)=sum(i)+b(i,j)
   end do
if(sum(i).eq.0) then
   do j=i-1,ndes
```

```
          b(i,j)=1.0d+0/(2+ndes-i)
        end do
        else
        if(abs(sum(i)-1).gt.ftol) then
          if(i.gt.1) then
          do j=i-1,ndes
            b(i,j)=b(i,j)/sum(i)
          end do
          else
          do j=1,ndes
            b(i,j)=b(i,j)/sum(i)
          end do
          endif
        endif
      endif
     end do
        b(nor,ndes)=1.
      do i=1,nor
        do j=1,ndes
          sumbb(i,j,t)=sumbb(i,j,t)+b(i,j)
          sumsquarebb(i,j,t)=sumsquarebb(i,j,t)+b(i,j)**2
        end do
      end do
  end do

   call multiply(onramp,b,offrampp,ntime,nor,ndes)
   do i=1,ntime
     do j=1,ndes
       sumerror(ttt)=sumerror(ttt)+((offramp(i,j)-offrampp(i,j))**2)
     end do
   end do
  end do
 do i=1,nor
   do j=1,ndes
     do t=1,ntime
     if((j+2).gt.i) then
       average(i,j,t)=sumbb(i,j,t)/nday
       std(i,j,t)=sqrt((sumsquarebb(i,j,t)-nday*(average(i,j,t)*
&*2))/nday)
     else
        average(i,j,t)=0.d+0
        std(i,j,t)=0.d+0
     endif
   end do
  end do
 end do
```

```
do i=1,nor
  do t=1,ntime
     write(3,"(11f8.4)") (average(i,j,t),j=1,ndes)
  end do
write(3,*) "
end do

do i=1,nor
  do t=1,ntime
     write(3,"(11f8.4)") (std(i,j,t),j=1,ndes)
  end do
 write(3,*) "
end do
close(3)
end

SUBROUTINE multiply(A,B,C,m,n,p)
implicit none
integer m,n,p,i,j,k
double precision A(m,n),B(n,p),C(m,p)

do i=1,m
   do k=1,p
        C(i,k)=0.d+0
      do j=1,n
        C(i,k)=C(i,k)+A(i,j)*B(j,k)
      end do
   end do
end do
end

  SUBROUTINE INV(K,A,C)
  Implicit none
  INTEGER K,I,J,L,K2
  double precisio n A(K,K),C(K,K),B(K,2*K),PIVOT,AIL

  DO 5 J=1,K
  DO 6 I=1,K
6   B(I,J)=A(I,J)
5   CONTINUE
  K2=K*2
  DO 7 J=1,K
  DO 8 I=1,K
  B(I,K+J)=0.0D+00
  IF(I.EQ.J) B(I,K+J)=1.0D+00
```

```
   8   CONTINUE
   7   CONTINUE
C       THE PIVOT OPERATION STARTS HERE
       DO 9 L=1,K
       PIVOT=B(L,L)
       DO 13 J=L,K2
  13   B(L,J)=B(L,J)/PIVOT
C       TO IMPROVE THE ROWS
       DO 14 I=1,K
       IF(I.EQ.L) GO TO 14
       AIL=B(I,L)
       DO 15 J=L,K2
  15   B(I,J)=B(I,J)-AIL*B(L,J)
  14   CONTINUE
   9   CONTINUE
       DO 45 I=1,K
       DO 46 J=1,K
  46   C(I,J)=B(I,K+J)
  45   CONTINUE
       RETURN
       END
C
C  fread - routine that reads the 2-d array from a file
C
      subroutine fread(row,col,matrix,fnum)
      integer row,col,fnum
      double precision matrix(row,col),xirtam(col,row)

      read(fnum,*) xirtam
      call transp(xirtam,col,row,matrix)
      end
C
C transp - returns the tranpose of a matrix.
C
      subroutine transp(x,col,row,y)
      integer row,col,i,j
      double precision x(col,row), y(row,col)

      do  i=1,row
        do  j=1,col
          y(i,j) = x(j,i)
      end do
      end do
      end
```

D. Matlab Script for Sequential Quadratic Programming

```
clear;
load 'Q.dat';
load 'D.dat';
load 'A.dat';
load 'B.dat';
load 'aeq.dat';
load 'beq.dat';
load 'lb.dat';
load 'ub.dat';
load 'ini.dat';
fid1=fopen('X.dat','w');
ntime=36;
nday=23;
nor=12;
ndes=11;

NewtonStep = 'Newton';
for i=1:nor
  for j=1:ndes
    sumbb(i,j)=0.;
    sumsquarebb(i,j)=0.;
    sumbb1(i,j)=0.;
    sumsquarebb1(i,j)=0.;
    for t=1:ntime
      sumbbx(i,j,t)=0;
      sumsquarebbx(i,j,t)=0.;
    end
  end
end

for ttt=1:nday
  fprintf(fid1,'%s %4d \n','ttt=',ttt);
[nineqcstr,numberOfVariables]=size(A);
[neqcstr,numberOfVariableseq]=size(aeq);
ncstr = nineqcstr + neqcstr;
eqix = 1:neqcstr;
lambda=zeros(ncstr,1);
aix=lambda;
indepInd = 1:ncstr;
A=[full(aeq);full(A)];
B=[beq;B];

normA = ones(ncstr,1);
normf=1;

errnorm = 0.01*sqrt(eps);
tolDep = 100*numberOfVariables*eps;
ACTSET=A(eqix,:);
ACTIND=eqix;
ACTCNT=neqcstr;
CIND=neqcstr+1;
neq=diag(ones(132,1),0);
X=ini;
```

```
simplex_iter=1;
cstr=A*X-B;

[QQ R]=qr((ACTSET)');
Z = QQ(:,neqcstr+1:numberOfVariables);

      for i=1:ntime
          for j=1:nor
            onramp(i,j)=Q(ntime*(ttt-1)+i,j);
          end
      end

      for i=1:ntime*ndes
        for j=1:nor*ndes
          onra(i,j)=0.d+0;
        end
      end

     for kk=1:ntime
       n=1;
       for i=(kk-1)*ndes+1:kk*ndes
          m=1;
          for j=(n-1)*nor+1:n*nor
            onra(i,j)=onramp(kk,m);
            m=m+1;
          end
          n=n+1;
       end
     end

%INITIAL H,c
for t=1:ntime
  q=onra((t-1)*ndes+1:(t-1)*ndes+ndes,:);
  y=D((ttt-1)*ntime*ndes+(t-1)*ndes+1:(ttt-1)*ntime*ndes+(t-1)*ndes+ndes);
  if t==1
    sum=q'*q;
    f=q'*y;
  else
    sum=sum+(1/t)*(q'*q-sum);
    f=f+(1/t)*(q'*y-f);
  end

  if det(sum)~=0
    break;
  end
end

t0=t;
H=sum;
oldind=0;

%MAIN ITERATION
while t < ntime
  t=t+1;
  q=onra((t-1)*ndes+1:(t-1)*ndes+ndes,:);
  y=D((ttt-1)*ntime*ndes+(t-1)*ndes+1:(ttt-1)*ntime*ndes+(t-1)*ndes+ndes);
```

```
f=f+(1/t)*(q'*y-f);
H=H+(1/t)*(q'*q-H);
c=-f;
 %from qpsub
  gf=H*X+c;
   %SD=-Z*((Z'*H*Z)\(Z'*gf));
   [SD, dirType] = compdir(Z,H,gf,numberOfVariables,c);
   GSD=A*SD;
   indf = find((GSD > errnorm * norm(SD))  &  ~aix);
 if isempty(indf) % No constraints to hit
    STEPMIN=1e16;
    dist=[]; ind2=[]; ind=[];
 else % Find distance to the nearest constraint
    dist = abs(cstr(indf)./GSD(indf));
    [STEPMIN,ind2] =  min(dist);
    ind2 = find(dist == STEPMIN);
    ind=indf(min(ind2));
 end

 delete_constr = 0;
 if ~isempty(indf)& isfinite(STEPMIN) % Hit a constraint
      if STEPMIN > 1  % Overstepped minimum; reset STEPMIN
         STEPMIN = 1;
        delete_constr = 1;
         ind=[];
      end
      X= X+STEPMIN*SD;

 else
   % did not hit a constraint
      STEPMIN = 1;   % Exact distance to the solution. Now delete constr.
      X = X + SD;
      delete_constr = 1;
 end

if delete_constr
    % Note: only reach here if a minimum in the current subspace found
       rlambda = -R\(QQ'*(H*X+c));
       actlambda = rlambda;
       actlambda(eqix) = abs(rlambda(eqix));
       indlam = find(actlambda < 0);

       if length(indlam)
       % Remove constraint
       lind = find(ACTIND == min(ACTIND(indlam)));
       lind=lind(1);
       ACTSET(lind,:) = [];
       aix(ACTIND(lind)) = 0;
       [QQ,R]=qrdelete(QQ,R,lind);
       ACTIND(lind) = [];
       ACTCNT = ACTCNT - 2;
              simplex_iter = 0;
              ind = 0;
             delete_constr = 0;
           else
```

```
            ACTCNT=ACTCNT-1;
     end
end

% Calculate gradient w.r.t objective at this point
gf=H*X+c;
% Update X and calculate constraints
cstr = A*X-B;
cstr(eqix) = abs(cstr(eqix));

if ind % Hit a constraint
   aix(ind)=1;
   ACTSET(CIND,:)=A(ind,:);
   ACTIND(CIND)=ind;
 %  CIND=CIND+1;
   [m,n]=size(ACTSET);
   [QQ,R] = qrinsert(QQ,R,CIND,A(ind,:)');
end

[m,n]=size(ACTSET);
 Z = QQ(:,m+1:n);
[QQ,R] = qr(ACTSET');
  ACTCNT=ACTCNT+1;
  CIND=ACTCNT+1;
if oldind
   aix(oldind) = 0;
end

  for i=1:nor
      for j=1:ndes
        xx(i,j)=X((j-1)*nor+i);
      end
  end

       for i=1:nor
          for j=1:ndes
            sumbbx(i,j,t)=sumbbx(i,j,t)+xx(i,j);
            sumsquarebbx(i,j,t)=sumsquarebbx(i,j,t)+xx(i,j).^2;
           end
       end
end %while t

    for i=1:nor
      for j=1:ndes
        xx(i,j)=X((j-1)*nor+i);
       end
    end
       for i=1:nor
          for j=1:ndes
            sumbb(i,j)=sumbb(i,j)+xx(i,j);
            sumsquarebb(i,j)=sumsquarebb(i,j)+xx(i,j).^2;
            end
       end
end %ttt

  for i=1:nor
```

```
        for j=1:ndes
            average1(i,j)=sumbb1(i,j)/nday;
            std1(i,j)=sqrt((sumsquarebb1(i,j)-nday*(average1(i,j).^2))/nday);
            for t=1:ntime
              avg(i,j,t)=sumbbx(i,j,t)/nday;
              std(i,j,t)=sqrt((sumsquarebbx(i,j,t)-nday*(avg(i,j,t).^2))/nday);
            end
        end
      end

        for i=1:nor
            for t=1:ntime
              fprintf(fid1,' %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f
\n',avg(i,1,t),avg(i,2,t),avg(i,3,t),avg(i,4,t),avg(i,5,t),avg(i,6,t),avg(i,7,t),avg(i,8,t),avg(i,9,t),avg(i,10,t),avg(i,11,t
));
            end
            fprintf(fid1,'\n');
          end

        for i=1:nor
        for t=1:ntime
          fprintf(fid1,' %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f
\n',std(i,1,t),std(i,2,t),std(i,3,t),std(i,4,t),std(i,5,t),std(i,6,t),std(i,7,t),std(i,8,t),std(i,9,t),std(i,10,t),std(i,11,t));
        end
        fprintf(fid1,'\n');
        end
         fclose(fid1);
```