DYNAMIC ESTIMATION OF ORIGIN-DESTINATION TRIP-TABLES FROM REAL-TIME TRAFFIC VOLUMES USING PARAMETER OPTIMIZATION METHODS

by

Namita Arora

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of requirements for the degree of
Master of Science

in
Industrial and Systems Engineering

APPROVED:

Hanif D. Sherali

Dr. H. D. Sherali (Chairman)

Hobeika

Dr. A. G. Hobeika

Subhash Sarin

Dr. S. C. Sarin

October 1993
Blacksburg, Virginia

DYNAMIC ESTIMATION OF ORIGIN-DESTINATION TRIP-TABLES FROM REAL-TIME TRAFFIC VOLUMES USING PARAMETER OPTIMIZATION METHODS

by

Namita Arora

Committee Chairman: Dr. Hanif D. Sherali

Industrial & Systems Engineering

ABSTRACT

The motivation behind this research is the need for a real-time implementable, yet accurate, procedure for estimating an origin-destination (O-D) trip-table based on entering and exiting traffic volume data for a given freeway section. These tables help in on-line control of traffic facilities, and consequently, are of significant use in alleviating traffic congestion. The dynamism of the approach captures the variations in the traffic counts with time which are in turn used to predict user travel patterns.

Two models have been developed for this problem, one based on a least squares estimation and the other based on an l_1 norm approach. Two projected conjugate gradient schemes are investigated for solving the constrained least squares problem, and an interior point affine scaling algorithm that is applied to the dual problem is explored for solving the l_1 estimation linear programming problem. Computational results are presented on a set of test problems involving the determination of O-D trip tables for both intersection and

freeway scenarios in order to demonstrate the viability of the proposed methods. These results exhibit that, unlike as reported in the literature based on previous efforts, properly designed parameter optimization methods can indeed provide accurate estimates in a real-time implementation. Hence, this research presents a competitive alternative to the iterative statistical techniques that have been heretofore used because of their real-time processing capabilities, despite their inherent inaccuracies. We hope that the proposed technology enhances existing methods for constructing O-D trip-tables from traffic counts.

ACKNOWLEDGMENTS

The author wishes to express her sincere thanks and gratitude to her advisor Dr. H. D. Sherali, for proposing this research, and for his constant encouragement and continued advice. He offered timely suggestions and guidance throughout the course of the work. During association with him, the author was exposed to several topics which are in the forefront of research activities in optimization applications to transportation. The author expresses her gratitude to Dr. A. G. Hobeika, who agreed to serve on the advisory committee. This research would not have reached fruition without his input, understanding and thoughtful remarks, sustained advice and the opportunity to work at the University Center for Transportation Research. The author wishes to thank Dr. S. C. Sarin for serving on the committee and for his advice and encouragement. The author expresses her thanks to General Motors Corporation for a graduate research fellowship which supported her during the course of this research and enabled completion of this work.

The author wishes to express appreciation for her thoughtful, caring and supportive husband Mr. Sanjay Arora. He has been a constant source of inspiration and strength for her. The author is also grateful and expresses her sincere thanks to her parents and parents-in-law for their support and encouragement.

Thanks are due to Dr. R. Sivanandan, who has specialized in the area of O-D trip-table estimation, for his help and guidance from time to time. The author also acknowledges and wishes to thank all her friends and well-wishers for their help and cooperation.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background

When a traffic congestion causing event, like a major accident, a stalled car, or a chemical spill occurs, the basic functions of the transportation network in the region are placed in jeopardy. Under such situations, affected links in an area are cordoned off to traffic by authorities, and this results in the closure of streets or in a reduction of their traffic carrying capabilities, disabling the road network from meeting the traffic demands. In turn, this induces traffic chaos, delays, and in many instances, accidents.

A major question that arises in such instances is how to minimize traffic chaos and detours. The answer to this question lies in the development of effective and efficient diversion strategies in real-time, including a better management of the existing infrastructure. This research effort is intended to contribute to a particular aspect of alleviating congestion through traffic diversion strategies. A crucial information in developing a successful diversion plan is the knowledge of destinations of travelers to be diverted. For this the traffic engineers and planners need information on turning movements and through volumes on freeways and street intersections for better traffic management. Origin-Destination matrices are an essential part of this information. An Origin-Destination (O-D) trip-table is a two-dimensional matrix of elements whose cell values represent the number of trips made between various O-D zone pairs in a given region. The information contained in an O-D trip table was conventionally established through extensive surveys of the travelers. These survey techniques are expensive, time-consuming and labor intensive. This necessitates obtaining more realistic trip-tables through cheaper and quicker means, and is the motivating factor leading to the development of theoretical approaches for synthesizing trip-tables using available information in the form of link traffic counts. The central idea in all these approaches is to use the information available in the data on link volumes to derive a trip-table that will replicate these observed link volumes as closely as possible.

1.2 O-D Estimation Procedures

The existing methods of O-D trip-table estimation can be distinctly classified into two sets of approaches:

- Static approaches
- Dynamic approaches

The O-D estimation procedures in which variations in O-D patterns over time are not taken into account, fall into the category of static approaches. The trip-tables are estimated for short time periods, ideally one hour. This is intended to capture the trip destinations during the time while the incident lasts. Alternatively, assuming steady state conditions, the observed link volumes during the hour preceding the incident can be used in the estimation process. *A priori* information and assumptions, such as the specification of prior trip tables, are needed to obtain satisfactory estimates. The inadequacy of the static models to deal with network performance under time varying demands has been pointed out by a number of researchers.

The dynamic methods address the dynamics of traffic flow. These methods utilize the information made available from the time sequences of changes in the traffic volumes and do not require more sites of data collection. In such problems the goal is to determine or estimate the proportions of traffic from each entrance going into an exit. These proportions are referred to by various names such as, turning proportions, split parameters, split ratios etc. These approaches do not require any *a priori* information as in the case of static models. This is an advantage of these models over the static models, since well-founded *a priori* information is often not available or is incomplete, and when substituted by more or less arbitrary assumptions, leads to poor estimates. The dynamic approaches are merely based on time sequences of traffic counts and require only moderate computational effort in most cases. They are hence very suitable for on-line applications and control. The history of Dynamic O-D estimation dates back to the early 1980's. During the recent years there have been significant advances in the area, but the need for a real-time O-D estimator still persists.

1.3 O-D Estimation in the IVHS Scenario

Intelligent Vehicle/ Highway System has attracted the attention of the transportation community, due to its potential for enhancing safety, mobility, and economic development, and for reducing energy costs and environmental impacts. These technologies will also contribute to reducing traffic congestion through advanced traffic management strategies. Improved mobility as well as improved convenience and comfort for all surface transportation users are the primary goals of IVHS. Five IVHS categories have been identified within the U.S., viz.

- Advanced Traffic Management Systems (ATMS)
- Advanced Traveler Information Systems (ATIS)
- Commercial Vehicle Operations (CVO)
- Advanced Vehicle Control Systems (AVCS)
- Advanced Public Transportation Systems (APTS)

Advanced Traffic Management Systems (ATMS) employ innovative technologies and integrate new and existing traffic management and control systems in order to be responsive to dynamic traffic conditions while servicing all modes of transportation. The key features of ATMS are subsystem integration and real-time control adjustments that account for traffic fluctuations.

Traffic management systems apply traffic engineering technologies to bring order and efficiency to the movement of highway vehicles. This is achieved through the use of vehicle detectors, communications, computers, and ramp signals to meter the flow of vehicles entering a freeway. The nature of dynamic O-D trip table estimators is such that the trip-tables generated act as tools for predicting traffic patterns for a particular traffic facility.

The following primary characteristics have been identified for ATMS.

- Collection of real-time traffic data.
- Reaction to changes in traffic flow with timely traffic management strategies.
- Area-wide surveillance and detection systems.
- Integration of the management of various functions, including transportation, demand management, freeway ramp metering, automated (electronic) toll collection, and arterial signal control.
- Collaborative action on the part of transportation management agencies and jurisdictions in order to optimize the strategies available to improve traffic flow.

- Rapid response incident management strategies.

O-D trip tables contribute to almost all of the characteristics mentioned above.

1.4 The Present Research

"Parameter Optimization" is one of the existing approaches for estimating the O-D trip-table split parameters for a freeway intersection system. The problem is posed as a constrained least squares. The O-D trip-tables estimated as a result of solving this problem have traditionally been found to be very accurate. However, the solution techniques that have been suggested involve an unacceptable level of computational effort.

In the present research a constrained least squares problem is formulated. The purpose is to solve this problem in real-time for on-line implementation. The two solution techniques employed to solve this problem are both conjugate gradient schemes. Conjugate gradient methods are conjugate direction or gradient deflection methods that lie somewhere in-between the methods of steepest descent and Newton's method. The principal advantage of these methods is that they do not require storage of any matrices as in Newton's method, and are globally convergent.

Given a starting point or iterate in terms of the split parameters, the anti-gradient to the objective function is computed at this point. Gradient based algorithms have slow convergence properties, and are prone to zigzagging. To alleviate this situation, the anti-gradient direction is deflected using an appropriate scheme such that the new direction points more favorably toward an optimal limiting solution. The deflected direction is projected onto the nullspace of the active constraints. Alternatively, the anti-gradient can be projected onto the feasible region and the projected anti-gradient deflected toward a favorable direction. In either case, a line search (in closed-form) is then conducted in order to determine the next iterate.

We have also developed an l_1 estimation model for this problem. Such a model is more appropriate than a least squares approach whenever there exist outliers in the data. Moreover, since this model can be equivalently transformed to a linear programming problem, it can easily accommodate certain other types of constraints that provide more reliable split parameter estimates. In any case, it is of interest to compare algorithmic effort and solution accuracy or quality using a least squares versus an l_1 estimation approach. The latter approach leads to a linear programming formulation as mentioned above, but one for which the dual has a special structure that is highly amenable to the affine scaling interior point algorithm. Hence, this is the approach that we explore in this thesis.

1.5 Organization of the Thesis

This thesis is organized as follows. Chapter 2 presents a review of the existing dynamic approaches the estimation of O-D trip-tables. Chapters 3 and 4 describe proposed models and solution techniques for the constrained least squares and the l_1 estimation approaches, respectively. Chapter 5 contains test results and comparisons from computational runs made on data simulated for a freeway section or an intersection. Finally Chapter 6 presents a brief discussion of the results along with recommendations for future research.

CHAPTER 2
DYNAMIC APPROACHES FOR O-D ESTIMATION: LITERATURE REVIEW

2.1 Introduction

The problem of predicting traffic flows in real-time has been viewed from several different angles in recent years. A number of different modeling approaches exist. In this review we distinguish between two main groups under the category of Dynamic Approaches for O-D estimation. The classification used is given below.

Types of models:

The existing models can be classified as follows:

1. ***Statistical Models.***
   a) Non-recursive
   b) Recursive
   c) Kalman Filtering
2. ***Parameter Optimization Models.***

Different model variations have been proposed in the sub-classes of the above categorization. The details of these models are provided in the following sections.

2.2 Statistical Models

The statistical models can be classified into three major subgroups, *viz*.,

- Non-recursive models
- Recursive models and,

- Kalman Filtering Approaches

Under this classification, Cremer and Keller (1981) adopted an approach that utilizes information available from processing the time sequences of changes in traffic volume through complex intersections.

Problem Definition:

An intersection with m entrances and n exits is considered. The following variables are introduced to describe the time-dependent distribution of traffic flow.

$\tau_j =$ Average travel time a vehicle needs to pass from any entrance to exit j

$T =$ Sampling interval defining a suitable discretized measurement period unit.

$q_i(k)=$ Traffic volume entering the ith entrance during the kth time interval, $t \in [kT,(k+1)T)$, $i=1,\ldots,m$

$y_j(k)=$ Traffic volume leaving via the jth exit during the shifted kth time interval, $t \in [kT+\tau_j,(k+1)T+\tau_j)$, $j=1,\ldots,n$

$f_{ij}(k)=$ That part of $q_i(k)$ which leaves the intersection through exit j during the shifted kth interval

$b_{ij}(k)=$ Split parameter, the portion of $q_i(k)$ which constitutes $f_{ij}(k)$,

i.e., $b_{ij}(k)=\dfrac{f_{ij}(k)}{q_i(k)}$

The introduced variables are interconnected by the following equations, which take the form of constraints that restrict the split parameter estimates.

$$f_{ij}(k)=q_i(k)b_{ij}(k) \qquad [1a]$$

$$0 \leq b_{ij}(k) \leq 1, \qquad \forall \qquad i,j,k \tag{1b}$$

$$\sum_{j=1}^{n} b_{ij}(k) = 1, \qquad \forall \qquad i,k \tag{1c}$$

$$b_{ii}(k) = 0, \qquad \forall \qquad i = 1,\ldots \min(m,n) \tag{1d}$$

$$y_j(k) = \sum_{i=1}^{m} f_{ij}(k) = \sum_{i=1}^{m} q_i(k) b_{ij}(k), \qquad \forall \qquad j,k \tag{1e}$$

Equation [1e] can be written in matrix form as,

$y^t(k) = q^t(k)B(k)$

where,

$y^t(k)$= $1 \times n$ row vector of elements $y_j(k)$

$q^t(k)$= $1 \times m$ row vector of elements $q_i(k)$

$B(k)$= $m \times n$ matrix of the elements $b_{ij}(k)$

2.2.1 Non-Recursive estimation approaches

Cremer and Keller (1983), developed the method of ***Cross Correlation Matrices.*** In reality, the splitting of the entry flows into the O-D flows varies from time interval to time interval randomly about a certain mean. Hence, the mean deviations, $\Delta q(k)$ of the entering volumes, $\Delta y(k)$ of the exiting volumes, and $\Delta b_{ij}(k)$ of the split parameters, are introduced.

Define,

$$\bar{q} = \frac{1}{K} \sum_{k=1}^{K} q(k)$$

$$\bar{y} = \frac{1}{K} \sum_{k=1}^{K} y(k)$$

$$\overline{B} = \frac{1}{K}\sum_{k=1}^{K} B(k)$$

as the mean values of $q(k)$, $y(k)$ and $B(k)$, respectively, over K sampling periods.

Under the above assumption, the following equations become inconsistent with real measurements $\Delta y(k)$ and $\Delta q(k)$ when $k > m$:

$$\Delta y(k) = \Delta q^t(k)\overline{B} + q^t(k)\Delta B(k) - \frac{1}{K}\sum_{l=1}^{K}\Delta q^t(l)\Delta B(l) \qquad [2.1]$$

An inconsistent solution $\hat{B}$ that minimizes the square of the error term:

$$q^t(k)\Delta B(k) - \frac{1}{K}\sum_{l=1}^{K}\Delta q^t(l)\Delta B(l)$$

in equation [2.1] is sought, which serves as an estimate for the real mean $\overline{B}$.

This is done as follows. Define,

$$Q = \begin{bmatrix} \Delta q^t(1) \\ \Delta q^t(2) \\ . \\ . \\ \Delta q^t(K) \end{bmatrix}, \qquad Y = \begin{bmatrix} \Delta y^t(1) \\ \Delta y^t(2) \\ . \\ . \\ \Delta y^t(K) \end{bmatrix} \qquad [2.2]$$

Then the least squares solution to [2.1] is given by,

$$\hat{B} = (Q^t Q)^{-1} Q^t Y \qquad [2.3]$$

where

$$Q^t Q = \sum_{k=1}^{K} \Delta q(k)\Delta q^t(k) = K\Phi_{qq} \qquad [2.4]$$

$$Q^t Y = \sum_{k=1}^{K} \Delta q(k) \Delta y^t(k) = K\Phi_{qy} \qquad [2.5]$$

and where, Φ_{qq} and Φ_{qy} are the finite interval cross-correlation matrices correlating the sequences $\Delta q(k)$ with itself and $\Delta q(k)$ with $\Delta y(k)$, respectively, over the observation periods k=1,....K. Hence, substituting equations [2.4] and [2.5] in [2.3], the following estimate is obtained.

$$\hat{B} = \Phi_{qq}^{-1} \Phi_{qy} \qquad [2.6]$$

Loosely interpreted, an estimate for the split parameter $\overline{b_{ij}}$ is computed by picking out that part of the leaving sequence $y_j(k)$ which is correlated with the sequence $q_j(k)$ of arrival counts at port i and relating it to the autocorrelation function of $q_j(k)$.

The authors concluded that better estimates could be expected for longer observation periods. However, when the mean split patterns $\overline{B}$ vary very slowly, the consideration of long observation periods becomes detrimental to the estimation procedure. It may be pointed out that this method does not satisfy the constraints [1b], [1c], and [1d]. Also, in the process of estimation, the inversion of an *mxm* matrix (for a facility with *m* entrances) becomes relatively expensive in a real-time implementation.

Nihan and Davis (1989) establish the validity of the ***Maximum Likelihood (ML) estimator*** when a time series of counts for each separate turning movement is available, and they employ the Expectation Maximization (EM) algorithm to obtain ML estimates when only counts of entering and exiting volumes are available.

These off-line estimators are developed under the following assumptions.

- all data for all times $t=1,\ldots,N$ are available for processing
- $B(t)=B,\ t=1,\ldots,N.$

Assuming now that input counts $q_j(t)$ are generated by random variables that are independent across time, the likelihood of observation $x_{ij}(t)$ is given by

$$L_i\{x_{ij}(t), b_{ij}, j=1,\ldots,n\} = \prod_t \{[\frac{q_i(t)!}{\Pi_j x_{ij}(t)!}][\prod_j (b_{ij})^{x_{ij}(t)}]\} \qquad [2.7]$$

Using the first order optimality conditions, the following maximum likelihood estimator is obtained:

$$\hat{b}_{ij} = \frac{\sum_t x_{ij}(t)}{\sum_t q_i(t)}, \qquad i=1,\ldots,m, \qquad j=1,\ldots,n \qquad [2.8]$$

When only entering counts $q(t)$ and exiting counts $y(t)$ are known, the EM algorithm handles the problem by starting with an initial estimate $\hat{B}$ and then estimating conditional expectations using

$$\sum_t \hat{x}_{ij}(t) = E\{\Sigma_t x_{ij}(t) | B, y(t), t=1,\ldots,N\} \qquad [2.9]$$

The value of $\hat{B}$ is reestimated by iterating between equation [2.9] and equation [2.8] until convergence is achieved.

The EM algorithm produces estimates with significant biases. However, these estimates are attractive due to the statistical efficiency properties of the maximum likelihood estimators. These estimates exhibit lower variances as compared with the authors' OLS estimates, described in the following sections. The ML estimates also satisfy the constraints [1b], [1c], [1d].

2.2.2 Recursive estimation approaches

The recursive estimation algorithms use for each time interval, the additional information contained within the new measurements $q_j(k)$ and $y_j(k)$ at each step, to generate a successively improved estimate $\hat{b}_{ij}$ for each of the unknown parameters $b_{ij}(k)$.

The deviations, $\Delta y(k)$ and $\Delta q(k)$, of counts from their mean values as obtained from an arbitrary interval, are considered. At the beginning of the estimation, the formula uses for the first step an estimate for the split parameter matrix, $\hat{B}(0)$, which may be taken from preceding investigations, or may even be selected arbitrarily. The general *k*th step is then carried out using the prediction

$$\Delta \hat{y}(k) = \Delta q'(k)\hat{B}(k-1) \qquad [2.10]$$

which is then substituted into the following recursive scheme,

$$\hat{b}_{ij}(k) = \hat{b}_{ij}(k-1) + \gamma\Delta q_i(k)[\Delta y_j(k) - \Delta \hat{y}_j(k)], \qquad [2.11]$$

Here, γ is a gain factor satisfying

$$0 < \gamma < \tfrac{2}{m\sigma_q^2}$$

where,

$$\sigma_q^2 = \max_i[E\{\Delta q_i^2(k)\}]. \qquad [2.12]$$

The recursive formula given by equation [2.8] yields bias-free estimates of the split parameter, which converges to the real mean values $\overline{b_{ij}}$. Also, computationally, the effort needed for this method turns out to be the lowest as compared with cross-correlation and parameter optimization methods. However, the quality of the estimates is worsened. A major drawback of the present method is that the constraints [1b], [1c], and [1d] are not automatically satisfied. These constraints need to be enforced later by additional normalization steps after calculating each of the new estimates.

Nancy and Nihan (1989) present recursive approaches based on minimizing the error between the predicted and observed output counts. The symbols used by them have the usual interpretation, as mentioned earlier, except for the following:

$b_{ij}(t)$= the probability that a vehicle enters at i and exits at j, during interval t.

$x_{ij}(t)$= the number of vehicles entering at i and exiting at j during interval t.

The expected value for each turning movement is given by

$$E\{x_{ij}(t)\} = b_{ij}(t)q_i(t) \qquad [2.13]$$

A large family of recursive estimation techniques are based on attempting to minimize the squared error between the observations made during interval t and the predicted values of these observations, given the parameter estimates from time $(t-1)$. These are known as the ***Recursive Prediction Error*** **(RPE)** techniques. The RPE techniques for the present problem take the following form:

$$\hat{b}_j(t) = \hat{b}_j(t-1) + g_j(t)[y_j(t) - q^t(t)\hat{b}_j(t-1)], \qquad j = 1,\ldots,n \qquad [2.14]$$

where, $g_j(t)$ is the gain vector which determines the particular RPE algorithm being used.

Three variants of the RPE estimators have been developed:

a) A Stochastic Gradient Algorithm.

b) Recursive Least-squares Estimator.

c) Tracking Recursive Least-squares Estimator derived via a Kalman Filter.

a) Stochastic Gradient Algorithm

Cremer and Keller (1983) are credited for being the first ones to apply the RPE algorithm to the O-D estimation problem for an intersection. The recursive equation for this algorithm is given by

$$\hat{b}_j(t) = \hat{b}_j(t-1) + G_j q(t)[y_j(t) - q^t(t)\hat{b}_j(t-1)], \qquad j = 1,\dots,n \qquad [2.15]$$

where, G_j are *mxm* diagonal matrices of predetermined gain terms. This is a steepest descent type of procedure.

b) Ordinary Least-squares Estimator

The OLS algorithm has a recursive form of the following type:

$$\begin{aligned} \hat{b}_{ij}(t) &= \hat{b}_j(t-1) + K_j(t) + [y_j(t) - q^t(t)\hat{b}_j(t-1)] \\ K_j(t) &= \frac{P_j(t-1)q(t)}{1 + q^t(t)P_j(t-1)q(t)} \\ P_j(t) &= P_j(t-1) - \frac{P_j(t-1)q(t)q^t(t)P_j(t-1)}{1 + q^t(t)P_j(t-1)q(t)} \end{aligned} \qquad [2.16]$$

The OLS algorithm, also sometimes referred to as the Recursive Least Squares (RLS) Algorithm, converges faster than the simple gradient based algorithm. The above OLS estimator does not guarantee the satisfaction of constraints [1b], [1c], and [1d].

c) Tracking Recursive Least-squares Estimator

An advantage possessed by recursive algorithms over off-line algorithms is that recursive algorithms can be modified so as to track parameters which show slow stepwise variation across time. Tracking is achieved by preventing the gain term $g_j(t)$ from going to 0 as t increases. Cremer and Keller (1987) were the first to note this possibility. For gradient algorithms, the gain term G_j is a prespecified constant. For RLS algorithms, a method that employs a Kalman Filter as introduced by both Nihan and Davis (1987) and Cremer and Keller (1987) is discussed.

Assuming that the columns of $B(t)$ vary with time according to a random process, the application of a Kalman Filter to estimate the system state $b_j(t)$ yields the following recursion:

$$\hat{b}_j(t) = \hat{b}_j(t-1) + K_j(t)[y_j(t) - q^t(t)\hat{b}_j(t-1)]$$
$$K_j(t) = \frac{P_j(t-1)q(t)}{r_j(t) + q^t(t)P_j(t-1)q(t)} \qquad [2.17]$$
$$P_j(t) = P_j(t-1) + R_j - \frac{P_j(t-1)q(t)q^t(t)P_j(t-1)}{r_j(t) + q^t(t)P_j(t-1)q(t)}$$

The above scheme again does not satisfy [1b], [1c], [1d].

Nihan and Davis (1987) present a modification of the RLS estimators [2.16] and [2.17] to incorporate the above mentioned constraints. This modification involves two main steps:

(1) Truncation:

For each j, compute

$$\hat{b}_j(t) = b_j(t-1) + \beta_j(t)[y_j(t) - q^t(t)\hat{b}(t-1)] \qquad [2.18]$$

where, β_j are scalars computed to satisfy [1b].

(2) Modify each row i of $\hat{\beta}(t)$ by using one of the following:

a) Normalization

$$\hat{b}_{ij}(t) = \frac{\hat{b}_{ij}(t)}{\sum_j \hat{b}_{ij}(t)} \qquad \text{or,} \qquad [2.19]$$

b) Projection

$$\hat{b}_{ij}(t) = \hat{b}_{ij}(t) + \frac{1.0 - \sum_j \hat{b}_{ij}(t)}{n} \qquad [2.20]$$

d) Kalman Filtering:

Cremer and Keller (1981) also give a recursive estimation scheme that is designed using a Kalman Filter. The problem is reformulated as follows:

A subsystem of O-D flows adding up to the jth exit flow is considered. Hence,

$$y_j(k) = \sum_{i=1}^{m} q_i(k) b_{ij}(k) = q^t(k) b_j(k) \qquad [2.21]$$

where $b_j(k)$ is defined to be the state vector of the jth subsystem. A dynamic transition of this state vector is described by the following state equation:

$$b_j(k+1) = b_j(k) + w(k) \qquad [2.22]$$

where $w(k)$ is an input vector of random white noise terms with zero means and known covariances.

The state variables are given by the jth exit's error flow measurements. These are related to $y_j(k)$ by equation [2.21] and may contain random terms $v(k)$, again, with zero means and known covariances. The measurement output equation then becomes,

$$y_j(k) = q^t(k) b_j(k) + v(k) \qquad [2.23]$$

The Kalman Filter generates bias free estimates only if the system model is completely observable, which is equivalently stated as,

$$\operatorname{rank}[q(k-K), q(k-K+1), q(k-K+2), \ldots, q(k)] = n \qquad [2.24]$$

The estimation of the unknown split parameters is performed by the following set of equations which establish the Kalman Filter:

$$d(k) = [P(k-1)+W]q(k)[q^t(k)(P(k-1)+W)q(k)+r]^{-1} \qquad [2.25]$$

$$P(k) = [I - d(k)q^t(k)][P(k-1)+W] \qquad [2.26]$$

where $d(k)$ is the filter gain, and $P(k)$ is the estimation error covariance matrix. The estimates b_{ij} are found using the following equation:

$$\hat{b}_j(k) = \hat{b}_j(k-1) + d(k)[y_j(k) - q^t(k)\hat{b}_j(k-1)] \qquad [2.27]$$

If the assumptions for the Kalman filter are fulfilled, the above filter equations compute bias-free, minimum variance estimates. For a 4-arm intersection the computational time required for these estimates was about 2 secs on a HP 1000 computer. But in the present case, the assumptions are not completely met since the constraints [1b], [1c] and [1d] have been ignored. Hence the estimates obtained may not necessarily be accurate.

Bell (1991) presented a new outlook to the problem of O-D estimation. He based his approach on the assumption that the time taken by a vehicle to traverse a junction or a network follows a certain distribution which spans a number of time intervals. This is different from the other approaches mentioned in the sense that the travel times are restricted to a fixed number of time intervals or are small as compared to the size of the interval chosen.

Two approaches were presented for the estimation of O-D flows under the above assumptions. The first one is applicable when the travel times are approximately geometrically distributed, and uses the recurrence model of platoon dispersion. The second approach makes no assumption about a specific distribution of travel times, but requires an estimation of additional parameters.

The first model is applicable only to single intersections, or small networks, since it assumes geometrically distributed travel times. The second method is suitable for motorway networks, but requires more parameters to be specified.

Both the preceding approaches use a constrained recursive least squares algorithm for estimation. The constraints [1b], [1c], and [1d] are partially satisfied.

2.3 Parameter Optimization Models

Cremer and Keller (1987) examined an estimation procedure that would use conditions [1b], [1c], and [1d] as additional specifications for the solution, and arrived at the method of constrained optimization for the estimation process. A parameter optimization model was formulated as a reference model, which was then tuned to give a best match with the real system.

Adopting the same notation as in Cremer and Keller (1981), the following deterministic model for a traffic facility is set up:

$$\hat{y}^t(k) = q^t(k)\hat{B} \qquad [2.28]$$

Given $q_i(k)$ and $y_j(k)$ over a period of K sampling intervals, a matrix $\hat{B}$ of estimates for the split parameters is sought that minimizes the sum of errors between the exit flows of the model and the measured exit flows. Toward this end, the following model is formulated:

$$\underset{\hat{B}}{Minimize} \quad \frac{1}{K}\sum_{k=1}^{K}\left\| y(k) - \hat{y}(k) \right\|^2 \qquad [2.29]$$

subject to:

$$\bar{y}^t = \bar{q}^t \bar{B} \qquad [2.30]$$

$$\sum_{j=1}^{n} b_{ij}(k) = 1 \qquad \forall \quad i,k \qquad [2.31]$$

$$0 \le b_{ij}(k) \le 1 \qquad \forall \quad i,j,k \qquad [2.32]$$

$$b_{ii}(k) = 0 \qquad \forall \quad i = 1,\dots \min(m,n) \qquad [2.33]$$

where,

$\bar{B}$= estimate for the unknown mean split matrix

$\bar{y}$= mean deviation of the exit volume j as predicted

Note: The contraints [2.30] are only an optional set of constraints. A relaxation of the above problem can be formulated by relaxing these constraints.

By the above formulation, the problem of estimating the split parameters $\overline{b_{ij}}(k)$ and hence the partial flows $\overline{f_{ij}}$ is posed as a ***parameter optimization*** problem with constraints. The Complex Algorithm of Box (1965) was used to solve this problem.

The above model estimates the average value of the split parameter matrix fairly well. The quality of the estimates improves with the length K of the optimization period. For a four arm intersection the computation of a set of new estimates of the split parameter is reported to have taken almost 1 minute on a HP 1000 computer. The computational effort needed for this method is relatively more than that required by the foregoing statistical approaches due to the iterative nature of the optimization routine. This procedure has an advantage over the method of cross-correlation matrices, in the sense that all the constraints are satisfied, and even additional conditions may be incorporated into the model as necessary. Due to the extent of computational effort required, this method is more suited for off-line applications and planning purposes.

CHAPTER 3

CONJUGATE GRADIENT ALGORITHMS FOR THE LEAST SQUARES O-D ESTIMATION PARAMETER OPTIMIZATION MODEL

In this chapter, we present an alternative to the parameter optimization approach of Cremer and Keller (1987). Recall that the solution obtained by them was very accurate but was not suitable for real-time O-D estimation due to the relatively high computational effort required for their procedure.

Following the same formulation as Cremer and Keller (1987), two new solution techniques are suggested for this problem. These procedures are based on conjugate gradient schemes using exact line searches in closed form. In the first scheme, given any iterate, the anti-gradient is computed and is deflected via a suitable scheme. The deflected direction is first projected onto the feasible region, and then an exact line search is conducted in closed-form along this resultant direction in order to determine the next iterate. This automatically takes care of the imposed constraints on the split parameters, and hence overcomes a major drawback that was present in most of the existing methods, as pointed out in Chapter 2. As an alternative, another conjugate gradient scheme is suggested in which the anti-gradient is first projected onto the feasible region and then the projected gradient is deflected. An exact line-search is then performed in closed form, as done in the first scheme.

3.1 Problem Formulation

A traffic facility (an intersection or a linear freeway section) with m entrances and n exits is considered. For each entrance i=1,...,m define

J_i = {j: exit j is permissible for users entering at entrance i},

and conversely, for each exit j=1,...,n define

I_j = {i: entrance i admits users that can take exit j}

Then, the constrained least squares estimation problem can be formulated as follows:

LSQ: Minimize $$f(\beta) = \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{n}[\sum_{i\in I_j} q_i(k-\tau_{ij})\beta_{ij} - y_j(k)]^2 \qquad [3.1]$$

$$\sum_{j\in J_i}\beta_{ij} = 1 \qquad \forall \qquad i = 1,\ldots m \qquad [3.2]$$

$$\beta_{ij} \geq 0 \qquad \forall \quad j\in J_i, \quad i = 1,\ldots m \qquad [3.3]$$

where

$i =$ 1,...,m : Index for entrances

$j =$ 1,...,n : Index for exits

K = Total number of time periods

T = Sampling interval defining a suitable discretized measurement period unit

τ_{ij}= Average number of time intervals that a vehicle needs to traverse from entrance i to exit j

$q_i(k)$= Traffic volume entering the ith entrance during the kth time interval $t\in[kT,(k+1)T)$, i=1,...,m (Note that we assume the availability of data for periods $1-\max\tau_{ij},\ldots,K-\min\tau_{ij}, j\in J_i$, for each $i = 1,\ldots,m$.)

$y_j(k)$= Traffic volume leaving via the jth exit during the kth time interval, $t\in[kT,(k+1)T)$, j=1,...,n

$\beta_{ij}=$ Split parameter, the portion of $q_i(k)$ that leaves through exit j

3.2 Initial Solution

An initial feasible solution to the problem **LSQ** is obtained by inexactly solving a relaxation of the problem. Specifically, an inexact solution is sought to the optimality conditions for the unconstrained problem by setting the gradient of the objective function to zero. This is done as follows.

The gradient to the objective function at any point β is given by

$$(g)_{rs} = (\nabla f(\beta))_{rs} = \left.\frac{\partial f}{\partial \beta_{rs}}\right|_{\beta} = \sum_k \varepsilon_s(k) q_r(k-\tau_{rs}), \quad \forall \quad r \in I_s, \quad s=1,\ldots,n \qquad [3.4]$$

where

$$\varepsilon_j(k) = \sum_{i \in I_j} [q_i(k-\tau_{ij})\beta_{ij} - y_j(k)] \quad \forall \quad j,k \qquad [3.5]$$

Setting the gradient to zero we get,

$$\sum_k \varepsilon_s(k) q_r(k-\tau_{rs}) = 0 \quad \forall \quad r \in I_s, \quad s=1,\ldots,n \qquad [3.6]$$

Substituting the value of $\varepsilon_s(k)$ from [3.5] we get,

$$\sum_k [\sum_{i \in I_s} q_i(k-\tau_{is})\beta_{is} - y_s(k)] q_r(k-\tau_{rs}) = 0 \quad \forall \quad r \in I_s, \quad s=1,\ldots,n \qquad [3.7]$$

This yields the system of equations,

$$\sum_{i \in I_s} [\sum_k q_i(k-\tau_{is}) q_r(k-\tau_{rs})] \beta_{is} = \sum_k q_r(k-\tau_{rs}) y_s(k) \quad \forall \quad r \in I_s, \quad s=1,\ldots,n$$

Note that for each exit s, the foregoing system yields a seperable set of equations in β_{rs}, $r \in I_s$, which can be solved for this latter set of parameters. In order to conserve effort, we approximated this solution by examining the equation for any given (r,s), where $r \in I_s$,

$s \in \{1,\ldots,n\}$, and solving for β_{rs} from that equation assuming that $\beta_{is} = \beta_{rs} \forall\ i \in I_s$. This yields,

$$\beta_{rs} = \frac{\sum_{k=1}^{K} y_s(k) q_r(k-\tau_{rs})}{\sum_{i \in I_s} [\sum_{k=1}^{K} q_i(k-\tau_{is}) q_r(k-\tau_{rs})]} \quad \forall \quad r \in I_s, \quad s = 1,\ldots,n \qquad [3.8]$$

Since we have a constrained problem the resulting split parameter vector obtained via [3.8] is projected onto the constraint set using the scheme given in section 3.2.1 below. The projected solution β^0 obtained is a feasible solution and may be used as a starting solution (**INIT-1**).

This solution can be further modified to obtain an advanced initial starting solution (**INIT-2**) as follows. Examine the solution β^0 obtained above and find,

$$j_i \equiv \arg\max\{\beta^0_{ij} : j \in J_i\} \quad \forall\ i = 1,\ldots,m \qquad [3.9]$$

Now, suppose that we write $\beta_{rj_r} = 1 - \sum_{\substack{s \in J_r \\ s \neq j_r}} \beta_{rs} \ \forall\ r = 1,\ldots,m$ from [3.2], and we accordingly consider $f(\beta)$ as being a function of $\beta' \equiv \{\beta_{rs}, s \in J_r, s \neq j_r, \forall\ r = 1,\ldots,m\}$ alone. Call this function $F(\beta')$. Then, we can find a minimum of f over [3.2] alone by solving for β via the system given by $\nabla F(\beta') = 0$ along with [3.2], where $\nabla F(\beta')$ denotes the gradient of F with respect to β'. This yields the following system of simultaneous equations:

$$\frac{\partial f}{\partial \beta_{rs}} - \frac{\partial f}{\partial \beta_{rj_r}} = 0 \quad \forall \quad s \in J_r,\ s \neq j_r\ r = 1,\ldots,m \qquad [3.10]$$

$$\sum_{s \in J_r} \beta_{rs} = 1 \quad \forall \quad r = 1,\ldots,m \qquad [3.11]$$

The above system can be expanded as follows:

$$\sum_{i\in I_s}[\sum_{k=1}^{K} q_i(k-\tau_{is})q_r(k-\tau_{rs})]\beta_{is} - \sum_{\substack{i\in I_{j_r}\\ i\neq r}}[\sum_{k=1}^{K} q_i(k-\tau_{ij_r})q_r(k-\tau_{rj_r})]\beta_{ij_r} - [\sum_{k=1}^{K} q_r^2(k-\tau_{rj_r})\beta_{rj_r}$$

$$= \sum_{k=1}^{K}[y_s(k)q_r(k-\tau_{rs}) - y_{j_r}(k)q_r(k-\tau_{rj_r})] \;\; \forall \;\; s\in J_r,\; s\neq j,\; r=1,\ldots,m \quad [3.12]$$

$$\sum_{s\in J_r}\beta_{rs} = 1 \;\; \forall \;\; r=1,\ldots,m \quad [3.13]$$

If the solution β obtained as a result of solving the system of equations [3.12]-[3.13] satisfies the non-negativity constraints [3.3], then an optimal solution is at hand and the algorithm can be terminated. Otherwise, we project β onto the feasible region using the scheme of section 3.2.1, to yield an initial feasible solution, β^0, to the problem **LSQ**.

3.2.1 Projection Scheme

The solution, $\bar{\beta}$, to the unconstrained problem is projected onto the region defined by the constraints [3.2] and [3.3] using the following scheme. The structure of our problem is highly amenable to the projection scheme suggested by Sherali and Shetty (1980) and Bitran and Hax (1976). This scheme, as applied to the present problem, is as follows:

Define,

$$\boldsymbol{\beta} = \{\beta: \sum_{j\in J_i}\beta_{ij} = 1, \;\; \forall \;\; i=1,\ldots m, \;\; \beta_{ij}\geq 0, \;\; j\in J_i, \;\; \forall \;\; i=1,\ldots m\} \quad [3.14]$$

We need to project $\bar{\beta}$ onto $\boldsymbol{\beta}$ in order to determine the initial solution β^0. That is, we need to find,

$$\beta^0 = P_{\boldsymbol{\beta}}(\bar{\beta}) \equiv \arg\min\{\|\beta-\bar{\beta}\|: \;\; \beta\in\boldsymbol{\beta}\} \quad [3.15]$$

This projection operation can be done separately for each entrance i. Hence for any i, the stepwise scheme is as follows:

Initialization:

Set the iteration counter t=0 and let,

$$\gamma^0 = (\bar{\beta}_{ij}, \ \ j \in J_i) \tag{3.16}$$

$$M_0 = J_i \tag{3.17}$$

Step 1:

At any iteration t, given γ^t and M_t, compute

$$\rho_t = [1 - \sum_{j \in M_t} \gamma_j^t] \Big/ |M_t| \ \text{ and} \tag{3.18}$$

$$\overline{\gamma}^t = \gamma^t + \rho_t u_t \ \ \text{where} \tag{3.19}$$

u_t= vector of $|M_t|$ elements each equal to 1. Hence, $\sum_{j \in M_t} \overline{\gamma}_j^t = 1$

If $\overline{\gamma}^t \geq 0$, then terminate with the solution

$$\beta_{ij}^0 = \begin{cases} \overline{\gamma}_j^t & \text{if} \quad j \in M_t \\ 0 & \text{if} \quad j \notin M_t \end{cases} \tag{3.20}$$

Else, proceed to Step 2.

Step 2:

Define M_{t+1} and γ^{t+1} as follows:

$$M_{t+1} = \{ j \in M_t : \ \overline{\gamma}_j^t > 0 \} \ \ \text{and} \tag{3.21}$$

$$\gamma^{t+1} = \{ \gamma_j^{t+1}, \ \ j \in M_{t+1} \}. \tag{3.22}$$

Increment t by one and return to Step 1.
(Note that $M_{t+1} \neq \phi$.)

3.3 Solution Techniques for solving LSQ

The least squares problem **LSQ** is solved using one of the two solution techniques called **LSQst-1** and **LSQst-2.** A detailed description of these two schemes is given in Sections 3.3.1 and 3.3.2, respectively.

3.3.1 Scheme LSQst-1

This algorithm commences with the initial solution β^0 generated as in Section 3.2.1. The initial direction of motion is set to the anti-gradient of the objective function at β^0. The prescribed direction of motion is projected onto the feasible region and an exact line-search is then conducted to determine the revised iterate. On subsequent iterations, the anti-gradient direction is deflected using one of the schemes suggested in subsection 3.3.1.1(ii) below, and this resulting direction is then projected onto the feasible region. Note that finitely often, say every $\sum_{i=1}^{m}|J_i|$ iterations (the dimension of the vector β), the adopted direction can be taken as the projected anti-gradient direction in order to ensure convergence; see Bazaraa, *et al.* (1993). Likewise, each time a new subspace of active constraints is encountered, the algorithm can either reset to the anti-gradient direction, or else, it can preserve prior information and adopt an appropriate deflected direction. Both these types of strategies have been investigated.

In any case, given a direction d_t, which might be the anti-gradient or the deflected anti-gradient, this can be projected onto the set of feasible directions as follows:
Define,

$$J_{i0}^l = \{j \in J_i : \beta_{ij}^l = 0\} \qquad \forall\, i = 1,\dots,m. \qquad [3.23]$$

and

$$D = \{d : \sum_{j \in J_i} d_{ij} = 0 \ \forall\, i = 1,\dots,m \text{ and } d_{ij} \geq 0 \ \forall\, j \in J_{i0}^l,\, i = 1,\dots,m\} \qquad [3.24]$$

The projected direction is given by,

$$d_{l_D} = P_D(d_l) = \text{argmin } \{\|d - d_l\| : d \in D\} \qquad [3.25]$$

This projection can be done for each index i separately, by the nature of the constraints in [3.24], using a procedure described in subsection 3.3.1.1(iv) below. If the projected direction $d_{l_D} = 0$ and $d_l = -g_l$ where g_l is the gradient of the objective function at β^l, then we can stop with β^l as the optimal solution since then, β^l is a KKT solution to the convex program **LSQ** (see Bazaraa *et al.*, 1993). If $d_{l_D} = 0$, and $d_l \neq -g_l$, then we set $d_l = -g_l$ and project this new direction using the scheme of subsection 3.3.1.1(iv). On the other hand, if $d_{l_D} \neq 0$, then d_{l_D} is an improving feasible direction provided that $d_{l_D} \cdot g_l < 0$. If this is not the case, we reset d_l to $-g_l$ and repeat this step. An optimal step-length λ_l along this direction can be found by minimizing $f(\beta^l + \lambda d_{l_D})$ over $\lambda \in (0, l_{\max}]$ where $\lambda_{\max}$ is the maximum step length that can be taken along d_{l_D} and yet maintain feasibility to the constraints of the problem **LSQ**.

The new iterate β^{l+1} is then found using,

$$\beta^{l+1} = \beta^l + \lambda_l d_{l_D} \qquad [3.26]$$

Incrementing l by one, this process can now be repeated until either $d_{l_D} = 0$ is obtained when $d_l = -g_l$, or else a suitable termination criteria is satisfied based on the rate of improvement in objective value over some given number of consecutive iterations. More specifically, in addition to specifying a termination tolerance value on the norm of the (projected) gradient at any iteration, another termination criteria is introduced based on the improvement in objective value. If the objective function value fails to improve by at least 1% over each of four consecutive iterations, the algorithm is terminated. A solution close to the optimum is most likely assured in such cases because gradient based algorithms tend to take relatively small step-lengths, slowing down the improvement in objective value per iteration, when the solution approaches an optimum. Note that a near optimal solution is sufficient in the present application of the research due to the predictive nature of the split parameter matrix.

A detailed statement of the algorithmic steps for the above procedure is given below. But first, we present some preliminary computations that are used in this procedure.

3.3.1.1 Some preliminaries

(i) Computation of the gradient g_l at iteration *l*:

At any iteration *l*, given the iterate β^l, define $\varepsilon_j^l(k)$ as the error at exit *j* at time period *k* as follows:

$$\varepsilon_j^l(k) = \sum_{i \in I_j} [q_i(k - \tau_{ij})\beta_{ij}^l - y_j(k)] \quad \forall \quad j,k \qquad [3.27]$$

so that

$$f(\beta^l) = \frac{1}{2} \sum_k \sum_j [\varepsilon_j^l(k)]^2 \qquad [3.28]$$

Also, denote $g_l = \nabla f(\beta^l)$ as the gradient of the objective function *f* at $\beta = \beta^l$. Note that the (*r,s*)th component of g_l is given by:

$$(g_l)_{rs} = (\nabla f(\beta^l))_{rs} = \left.\frac{\partial f}{\partial \beta_{rs}}\right|_{\beta=\beta^l} = \sum_k \varepsilon_s^l(k) q_r(k - \tau_{rs}), \quad \forall \quad r \in I_s, \quad s = 1,\ldots,n \quad [3.29]$$

(ii) Deflection strategies for determining the direction of motion d_l:

The direction of motion d_l is selected by deflecting the anti-gradient $-g_l$ via one of the several schemes suggested below:

1. $$d_l = -g_l + \psi_l d_{l-1}, \quad \text{where } d_0 = -g_0 \qquad [3.30]$$

$$\psi_l = \frac{\|g_l\|}{\|d_{l-1}\|}, \qquad l \geq 1 \qquad \text{(Sherali and Ulular, 1989)} \qquad [3.31]$$

2. $$d_l = -g_l + \psi_l d_{l-1}, \quad \text{where } d_0 = -g_0 \qquad [3.32]$$

$$\psi_l = \frac{g_l \cdot q_l - \left(1/\lambda_{l-1} \right) g_l \cdot d_{l-1}^a}{d_{l-1} \cdot q_l}, \quad \text{and where} \qquad [3.33]$$

$$q_l = g_l - g_{l-1} \qquad [3.34]$$

$$d_{l-1}^a = \beta^l - \beta^{l-1} \qquad \text{(Sherali and Ulular, 1990)} \qquad [3.35]$$

3. $$d_l = -g_l + \psi_l d_{l-1}, \quad \text{where } d_0 = -g_0 \qquad [3.36]$$

$$\psi_l = \frac{\|g_l\|^2}{\|g_{l-1}\|^2}, \qquad \text{(Fletcher and Reeves', 1964)} \qquad [3.37]$$

4. $$d_l = -g_l + \psi_l d_{l-1}^a \quad \text{where } d_0 = -g_0 \qquad [3.38]$$

$$\psi_l = \frac{\|g_l\|}{\|d_{l-1}^a\|}, \qquad l \geq 1 \qquad [3.39]$$

$$d_{l-1}^a = \beta^l - \beta^{l-1} \qquad \text{(Sherali and Ulular, 1989)} \qquad [3.40]$$

(iii) Computation of Step-lengths

Given an improving, feasible direction d_{l_D} at a solution β^l, the optimal step-length λ_l is found by first determining a solution λ^* to the system

Set $$\frac{df(\beta^l + \lambda d_{l_D})}{d\lambda} = 0 \qquad [3.41]$$

where

$$f\ (\beta^l + \lambda d_{l_D}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{n}\left[\sum_{i \in I_j} q_i(k-\tau_{ij})(\beta_{ij}^l + \lambda(d_{l_D})_{ij}) - y_j(k)\right]^2 \qquad [3.42]$$

Define the error term at the given lth iteration as,

$$E_j^l(k) = \sum_{i \in I_j} q_i(k-\tau_{ij})(\beta_{ij}^l + \lambda(d_{l_D})_{ij}) - y_j(k) \quad \forall\ j,k \qquad [3.43]$$

Then

$$\frac{df\ (\beta^l + \lambda d_{l_D})}{d\lambda} = \sum_{k=1}^{K}\sum_{j=1}^{n} E_j^l(k)\left[\sum_{i \in I_j} q_i(k-\tau_{ij})(d_{l_D})_{ij}\right] \qquad [3.44]$$

Substituting for the value of $E_j^l(k)$ from [3.43], we get,

$$\frac{df(\beta^l + \lambda d_{l_D})}{d\lambda} = \sum_{k=1}^{K}\sum_{j=1}^{n}\{\sum_{i \in I_j}[q_i(k-\tau_{ij})\beta_{ij}^l + \lambda q_i(k-\tau_{ij})(d_{l_D})_{ij}] - y_j(k)\}[\sum_{i \in I_j} q_i(k-\tau_{ij})(d_{l_D})_{ij}]$$

[3.45]

Now, define

$$\delta_j^l(k) = \sum_{i \in I_j} q_i(k-\tau_{ij})(d_{l_D})_{ij} \qquad [3.46]$$

Then, recalling the definition of $\varepsilon_j^l(k)$ from [3.4] and using [3.45], equation [3.41] becomes

$$\sum_{k=1}^{K}\sum_{j=1}^{n}\{\varepsilon_j^l(k) + \lambda\delta_j^l(k)\}\delta_j^l(k) = 0$$

Solving for λ, we get,

$$\lambda = \lambda^* \equiv \frac{-\sum_{k=1}^{K}\sum_{j=1}^{n} \delta_j^l(k)\varepsilon_j^l(k)}{\sum_{k=1}^{K}\sum_{j=1}^{n}[\delta_j^l(k)]^2} \qquad [3.47]$$

where the denominator is non-zero since d_{l_D} is a descent direction and **LSQ** has an unconstrained minimum. Also, note that

$$\frac{d^2 f}{d\lambda^2} = \sum_{k=1}^{K}\sum_{j=1}^{n}[\delta_j^l(k)]^2 > 0$$

Hence, an unconstrained global minimum to the function f along the direction d_{l_D} occurs at $\lambda = \lambda^*$. The constrained optimal step length λ_l is obtained from,

$\lambda_l = \min\{\lambda^*, \lambda_{\max}\}$

where the maximum feasible step length $\lambda_{\max}$ is given by

$$\lambda_{\max} = \min_{i=1,\dots m}\left\{\min_{j \in J_i}\{-\frac{\beta_{ij}^l}{(d_{l_D})}: \ (d_{l_D})_{ij} < 0\}\right\} \qquad [3.48]$$

(Note that since $d_{l_D} \in D$ as defined in [3.25] and that $d_{l_D} \neq 0$, $\lambda_{\max}$ is well defined and lies in the open interval $(0, \infty)$.)

(iv) Scheme for Determining Projected Search Directions

The following procedure can be applied for determining projections that are required in scheme **LSQst-1** as well as in scheme **LSQst-2** described later in subsection 3.3.2. When using this projection strategy for **LSQst-1**, we set the vector p_l being projected onto the feasible direction set D to the vector d_l,. The projected direction p_{l_D} then represents d_{l_D}. When using the scheme for **LSQst-2**, we set $p_l = -g_l$, and hence obtain the projected anti-gradient as $g_{l_{proj}} = p_{l_D}$.

Initialization:

Set the iteration counter t=0.

Define,

$$\gamma^0 = ((p_l)_{ij}, j \in J_i) \tag{3.49}$$

$$M_0 = J_i \tag{3.50}$$

Step 1:

Given γ^t and M_t compute,

$$\rho_t = \left[- \sum_{j \in J_i} \gamma_j^t \right] \Big/ |M_t| \tag{3.51}$$

$$\overline{\gamma}^t = \gamma^t + \rho_t u_t \tag{3.52}$$

where u_t = vector of $|M_t|$ ones.

Hence, $\sum_{j \in J_i} \overline{\gamma}_j^t = 0$. If $\overline{\gamma}_j^t \geq 0 \;\; \forall \; j \in J_{i0} \cap M_t$, then terminate with the projected direction given by,

$$(p_{l_D})_{ij} = \begin{bmatrix} \overline{\gamma}_j^t & \text{if } j \in M_t \\ 0 & \text{if } j \notin M_t \end{bmatrix} \tag{3.53}$$

Else, go to Step 2.

Step 2:

Set

$$M_{t+1} = M_t - \{j \in J_{i0} : \overline{\gamma}_j^t < 0\} \tag{3.54}$$

$$\gamma^{t+1} = \{\overline{\gamma}_j^t : j \in M_{t+1}\} \tag{3.55}$$

$t \leftarrow t+1$

and return to Step 1.

3.3.1.2 Algorithmic Steps for LSQst-1

Initialization:

(i) Pick a termination tolerance $\varepsilon_0 > 0$ (Recommended value: $\varepsilon_0 = 10^{-4}$.)

(ii) Pick values for:

ff_{max} = an upper limit on the number of consecutive iterations for which the algorithm is run before terminating it while the objective function value does not improve by at least 1% per iteration

rst_{max} = parameter for resetting the direction of motion

(Recommended values: $ff_{max} = 4$, $rst_{max} = \sum_{i=1}^{m} |J_i|$.)

(iii) Set ff=0, where ff is a counter for the number of most recent consecutive iterations for which the objective function value did not improve by at least 1% per iteration,

rst=1, where rst is the counter for resetting the direction of motion to the anti-gradient direction atleast every rst_{max} iterations,

$nrst$=0, where $nrst$ keeps a count of the total number of times the direction of motion is reset to the anti-gradient using any of the resetting criterion and set,

l=0, where l is the iteration counter for the algorithm.

(iv) Determine a starting solution β^0 as in Section 3.2, and compute $f_0 = f(\beta^0)$ and $g_0 = \nabla f(\beta^0)$.

(v) If $\|g_0\| < \varepsilon_0$ then stop with the current solution ,β^0, as a near optimal solution.

(This termination criteria is denoted as **TRMc-1a**.)

Step 1(a):

Let $d_l = -g_l$, increment $nrst \leftarrow nrst + 1$, and set rst=1.

Proceed to Step 1(c).

Step 1(b):

Determine the deflected direction d_l using one of the schemes given in subsection 3.3.1.1(ii).

Step 1(c):

Find $d_{l_D} = P_D(d_l)$ where P_D is the operator that projects onto the feasible direction space D as defined in [3.25].

Step 2(a):

(Termination Criteria: TRMc-2): If $\|d_{l_D}\| < \varepsilon_0$ then if $d_l = -g_l$, stop with β^l as a near optimal solution.

(Reset Criteria: RSTc-1): Otherwise, if $\|d_{l_D}\| < \varepsilon_0$ and $d_l \neq -g_l$, then return to Step 1(a).

Step 2(b):

(Reset Criteria RSTc-2): If $\|d_{l_D}\| \geq \varepsilon_0$, but $d_{l_D} \cdot g_l \geq -\varepsilon_0$, then return to Step 1(a).

Otherwise, compute the step-length $\lambda_l = \min\{\lambda^*, \lambda_{\max}\}$, where

$$\lambda^{*} \equiv \frac{-\sum_{k=1}^{K}\sum_{j=1}^{n} \delta_j^l(k)\varepsilon_j^l(k)}{\sum_{k=1}^{K}\sum_{j=1}^{n} [\delta_j^l(k)]^2}$$

and,

$$\lambda_{\max} = \min_{i=1,\dots m} \left\{ \min_{j \in J_i} \{-\frac{\beta_{ij}^l}{(d_{l_D})}: \ (d_{l_D})_{ij} < 0\} \right\}$$

and where $\delta_j^l(k)$ and $\varepsilon_j^l(k)$ are as defined in equations [3.46] and [3.4], respectively

Step 3:

Find the new iterate $\beta^{l+1} = \beta^l + \lambda_l d_{l_D}$.

Step 4:

Compute f_{l+1} as in [3.1]

and $g_{l+1} \equiv \nabla f(\beta^{l+1})$ as in subsection 3.3.1.1(i).

(Termination criteria TRMc-1b): If $\|g_{l+1}\| < \varepsilon_0$, terminate the algorithm with the current solution as a near optimal solution. Else, proceed to Step 5(a).

Step 5(a):

If $f_{l+1} \not\leq 0.99 f_l$, then set $ff \leftarrow ff+1$ and proceed to Step 5(b).

Else, set ff=0 and go to Step 6(a).

Step 5(b): (Termination criteria TRMc-3)

If $ff \geq ff_{\max}$ then stop and declare the current solution as being near optimal.

Else, proceed to Step 6(a).

Step 6(a): (Reset criteria RSTc-3)

If $\lambda_l = \lambda_{\max}$, then increment $l \leftarrow l+1$ and return to Step 1(a).

Else, proceed to Step 6(b).

Step 6(b): (Reset criteria RSTc-4)

If $rst < rst_{\max}$, go to Step 7.

Else, increment $l \leftarrow l+1$ and return to Step 1(a).

Step 7:

Increment $l \leftarrow l+1$ and,

$rst \leftarrow rst+1$ and return to Step 1(b).

3.3.2 Scheme LSQst-2

The second proposed algorithm also commences with the initial solution β^0 as generated in subsection 3.2.1. However as opposed to projecting the deflected direction of motion d_l, onto the set of feasible directions, the procedure **LSQst-2** begins each iteration by projecting the anti-gradient of the objective function onto the feasible direction set, D defined in [3.25]. This is done by using the projection scheme given in subsection 3.3.1.1(iv) with $p_l = -g_l$. This produces the projected anti-gradient, $g_{l_{proj}}$,as the vector p_{l_D}, and represents

$$g_{l_{proj}} = P_D(-g_l) = \text{argmin } \{\|d-(-g_l)\| : \text{d} \in D\} \qquad [3.56]$$

When using the scheme for **LSQst-2**, set $p_l = -g_l$ S=G, and hence obtain the projected anti-gradient as $g_{l_{proj}} = p_{l_G}$.

If $g_{l_{proj}} \equiv 0$ then the algorithm is terminated with the current solution as being optimal. If $g_{l_{proj}} \neq 0$ then while initializing the procedure, or resetting according to some criterion, we use the direction of motion as $d_l = g_{l_{proj}}$. An exact line-search is then performed along this direction to obtain the next iterate. At other subsequent iterations

l+1, given that the previous implemented direction d_l is still a feasible direction (else a resetting to the projection of the anti-gradient is performed), the direction of motion $d_{(l+1)}$ is found by deflecting the negative projected gradient $g_{(l+1)_{proj}}$ using this previous direction of motion d_l according to one of the schemes given in subsection 3.3.1.1(ii). This scheme conforms with the application of conjugate gradient methods on projected subspaces and would be related to second-order reduced gradient methods; see Bazaraa *et al.*, (1993).

We now present a statement of the algorithmic steps for the procedure **LSQst-2.**

3.3.2.1 Algorithmic Steps for LSQst-2

Initialization:

(i) Pick a termination tolerance $\varepsilon_0 > 0$ (Recommended value: $\varepsilon_0 = 10^{-4}$.)

(ii) Pick values for:

$ff_{\max}$ = an upper limit on the number of consecutive iterations for which the algorithm is run before terminating it while the objective function value does not improve by at least 1% per iteration

$rst_{\max}$ = parameter for resetting the direction of motion

(Recommended values: $ff_{\max} = 4$, $rst_{\max} = \sum_{i=1}^{m} |J_i|$.)

(iii) Set ff=0, where ff is a counter for the number of most recent consecutive iterations for which the objective function value did not improve by at least 1% per iteration,

rst=1, where rst is the counter for resetting the direction of motion to the anti-gradient direction atleast every $rst_{\max}$ iterations, *nrst*=0, where *nrst* keeps a count of the total number of times the direction of motion is reset to the anti-gradient using any of the resetting criterion and set,

l=0, where *l* is the iteration counter for the algorithm.

(iv) Determine a starting solution β^0 as in Section 3.2, and compute $f_0 = f(\beta^0)$ and $g_0 = \nabla f(\beta^0)$.

(v) Project $-g_0$ using the scheme of subsection 3.3.1.1(iv) onto the set *D*, as defined in [3.56], to get $g_{0_{proj}}$.

(vi) If $\left\| g_{0_{proj}} \right\| < \varepsilon_0$ then stop with the current solution ,β^0, as a near optimal solution. (This termination criteria is denoted as **TRMc-1a**.)

Step 1(a):

Let $d_l = g_{l_{proj}}$, increment $nrst \leftarrow nrst + 1$, and set *rst*=1.

Proceed to Step 2(a).

Step 1(b):

Determine the deflected direction d_l using one of the schemes given in subsection 3.3.1.1 (ii).

Step 2(a):

(Termination Criteria TRMc-2): If $\|d_l\| \le \varepsilon_0$ then if $d_l = g_{l_{proj}}$, stop with β^l as a near optimal solution. Otherwise,

(Reset Criteria RSTc-1): Otherwise if $\|d_l\| < \varepsilon_0$ and $d_l \neq g_{l_{proj}}$ then return to Step 1(a).

Step 2(b):

(Reset Criteria RSTc-2): If $\|d_l\| \geq \varepsilon_0$ but $d_l \cdot g_{l_{proj}} \geq -\varepsilon_0$, then return to Step 1(a).

Otherwise, compute the step-length $\lambda_l = \min\{\lambda^*, \lambda_{\max}\}$, where

$$\lambda^* \equiv \frac{-\sum_{k=1}^{K}\sum_{j=1}^{n}\delta_j^l(k)\varepsilon_j^l(k)}{\sum_{k=1}^{K}\sum_{j=1}^{n}[\delta_j^l(k)]^2}$$

and,

$$\lambda_{\max} = \min_{i=1,\ldots m}\left\{\min_{j\in J_i}\{-\frac{\beta_{ij}^l}{(d_{l_{act}})}: \ (d_{l_{act}})_{ij} < 0\}\right\}$$

and where $\delta_j^l(k)$ and $\varepsilon_j^l(k)$ are as defined in equations [3.46] and [3.4], respectively

Step 3:

Find the new iterate $\beta^{l+1} = \beta^l + \lambda_l d_l$.

Step 4:

Compute f_{l+1} as in [3.1]

and $g_{l+1} \equiv \nabla f(\beta^{l+1})$ as in subsection 3.3.1.1(i).

Find $g_{(l+1)_{proj}} = P_D(-g_{(l+1)})$ as defined in [3.56] where P_D is the operator that projects onto the set D, using the projection scheme of subsection 3.3.1.1 (iv).

(Termination criteria TRMc-1b): If $\left\|g_{(l+1)_{proj}}\right\| < \varepsilon_0$, then terminate the algorithm with the current solution as being a near optimal solution, else, proceed to Step 5(a).

Step 5(a):

If $f_{l+1} \nleq 0.99 f_l$, then set $ff \leftarrow ff+1$ and proceed to Step 5(b).

Else, set ff=0 and go to Step 6(a).

Step 5(b): (Termination criteria TRMc-3)

If $ff \geq ff_{max}$ then stop and declare the current solution as being near optimal.

Else, proceed to Step 6(a).

Step 6(a): (Resetting criteria RSTc-3)

If $\lambda_l = \lambda_{max}$ then increment $l \leftarrow l+1$ and return to Step 1(a).

Else, proceed to Step 6(b).

Step 6(b): (Resetting criteria RSTc-4)

If $rst < rst_{max}$, go to Step 7.

Else, increment $l \leftarrow l+1$ and return to Step 1(a).

Step 7:

Increment $l \leftarrow l+1$ and,

$rst \leftarrow rst+1$ and return to Step 1(b).

CHAPTER 4

AN L_1 ESTIMATION PROCEDURE FOR DETERMINING O-D TRIP-TABLES USING AN AFFINE SCALING ALGORITHM

In this chapter, a new *linear programming model* (LP) is presented for estimating an O-D trip-table using traffic volume counts by posing the problem as an L_1 estimation problem. Such a model is more attractive than a least squares approach in the presence of outliers in the input data. Moreover, this model is readily able to accomodate additional side constraints that lend a further control over the determined split parameter estimates. It is also of interest to note that the dual to this LP has a special structure. The Affine Scaling variant of Karmarkar's interior point algorithm (see Vanderbei *et al.*, 1986, Barnes, 1986 and Sherali *et al.*, 1988), specialized for the L_1 estimation problem via a bounded variable dual formulation as in Sherali *et al.*, (1988) is used to solve this dual problem by exploiting this special structure. A commercial simplex based routine is also used to solve this dual problem as a point of comparison.

4.1 Problem Formulation

A traffic facility (an intersection or a linear freeway section) with m entrances and n exits is considered. For each entrance i=1,...,m define

J_i = {j: exit j is permissible for users entering at entrance i},

and conversely, for each exit j=1,...,n define

I_j = {i: exit j is permissible for users entering at entrance i}

Then, the L_1 estimation problem for computing the split parameters β_{ij} can be formulated as follows:

LP: Minimize $$\sum_{k=1}^{K}\sum_{j=1}^{n}\left|\sum_{i\in I_j} q_i(k-\tau_{ij})\beta_{ij} - y_j(k)\right| \qquad [4.1]$$

subject to

$$\sum_{j\in J_i}\beta_{ij} = 1 \qquad \forall \quad i = 1,\ldots,m \qquad [4.2]$$

$$\sum_{i\in I_j}\beta_{ij}\bar{q}_{ij} = \bar{y}_j \qquad \forall \quad j = 1,\ldots,n \qquad [4.3]$$

$$\beta_{ij} \geq 0 \qquad \forall \quad j\in J_i, \quad i = 1,\ldots,m \qquad [4.4]$$

where

$i =$ $1,\ldots,m$: Index for entrances

$j =$ $1,\ldots,n$: Index for exits

K = Total number of time periods

T = Sampling interval defining a suitable discretized measurement period unit

τ_{ij} = Average number of time intervals that a vehicle needs to traverse from entrance i to exit j

$q_i(k)$= Traffic volume entering the ith entrance during the kth time interval $t\in[kT,(k+1)T)$, i=1,...,m (Note that we assume the availability of data for periods $1-\max\ \tau_{ij},\ldots,K-\min\ \tau_{ij}, j\in J_i$, for each $i = 1,\ldots,m$.)

$y_j(k)$= Traffic volume leaving via the jth exit during the kth time interval, $t\in[kT,(k+1)T)$, j=1,...,n

β_{ij}= Split parameter, the portion of $q_i(k)$ that leaves through exit j

$$\bar{q}_{ij} = \frac{1}{K}\sum_{k=1}^{K} q_i(k-\tau_{ij}) \text{ and}$$

$$\bar{y}_j = \frac{1}{K}\sum_{k=1}^{K} y_j(k)$$

Note that in contrast with the least squares formulation, we have included herein a set of restrictions [4.3] which enforce that on the average, the split parameters must be exact. Such linear constraints are readily accommodated within a linear programming approach.

The above L_1 estimation problem can be equivalently formulated as a linear programming problem as follows.

LP: Minimize $$\sum_{k=1}^{K}\sum_{j=1}^{n}(\delta_{kj}^{+}+\delta_{kj}^{-}) \qquad [4.5]$$

subject to

$$\sum_{i\in I_j} q_i(k-\tau_{ij})\beta_{ij}-(\delta_{kj}^{+}-\delta_{kj}^{-})=y_j(k) \quad \forall \quad k=1,\ldots,K,\ j=1,\ldots,n \qquad [4.6]$$

$$\sum_{j\in J_i}\beta_{ij}=1 \quad \forall \quad i=1,\ldots,m \qquad [4.7]$$

$$\sum_{i\in I_j}\beta_{ij}\bar{q}_{ij}=\bar{y}_j \quad \forall \quad j=1,\ldots,n \qquad [4.8]$$

$$\beta_{ij}\geq 0 \quad \forall \quad j\in J_i,\quad i=1,..,m \qquad [4.9]$$

$$\delta_{kj}^{+},\delta_{kj}^{-}\geq 0 \quad \forall \quad k=1,..,K,\quad j=1,\ldots,n \qquad [4.10]$$

The dual to problem **LP** can be written as follows:

D: Maximize $$\sum_{k=1}^{K}\sum_{j=1}^{n}y_j(k)w_{jk}^{(1)}+\sum_{i=1}^{m}w_i^{(2)}+\sum_{j=1}^{n}\bar{y}_jw_j^{(3)} \qquad [4.11]$$

subject to

$$\sum_{k=1}^{K}q_i(k-\tau_{ij})w_{jk}^{(1)}+w_i^{(2)}+\bar{q}_{ij}w_j^{(3)}+w_{ij}^{(4)}=0 \quad \forall \quad j\in J_i,\quad i=1,\ldots,m, \qquad [4.12]$$

$$-1 \le w_{jk}^{(1)} \le 1 \quad \forall \quad j = 1,\ldots,n, \quad k = 1,\ldots,K \qquad [4.13]$$

$$w_i^{(2)} \text{ unrestricted} \quad \forall \quad i = 1,\ldots,m \qquad [4.14]$$

$$w_j^{(3)} \text{ unrestricted} \quad \forall \quad j = 1,\ldots,n \qquad [4.15]$$

$$w_{ij}^{(4)} \ge 0 \quad \forall \quad j \in J_i \quad i = 1,\ldots,m \qquad [4.16]$$

Let W represent the entire set of dual variables, so that,

$$W \equiv \{w_{jk}^{(1)} \forall j = 1,\ldots,n, k = 1,\ldots,K, w_i^{(2)} \forall i = 1,\ldots,m, w_j^{(3)} \forall j = 1,\ldots,n, w_{ij}^{(4)} \forall j \in J_i, i = 1,\ldots,m\}$$

[4.17]

Denote the dimension of the vector W by L. Partition the index set $\{1,2,\ldots,L\}$ into sets L_1, L_2, L_3, and L_4 such that,

the vector $\{W_l, \quad l \in L_1\}$ represents $\{w_{jk}^{(1)} \quad \forall \quad j = 1,\ldots,n, \quad k = 1,\ldots,K\}$ [4.18]

the vector $\{W_l, \quad l \in L_2\}$ represents $\{w_i^{(2)} \quad \forall \quad i = 1,..,m\}$ [4.19]

the vector $\{W_l, \quad l \in L_3\}$ represents $\{w_j^{(3)} \quad \forall \quad j = 1,\ldots,n\}$ [4.20]

the vector $\{W_l, \quad l \in L_4\}$ represents $\{w_{ij}^{(4)} \quad \forall \quad j \in J_i, \quad i = 1,\ldots,m\}$ [4.21]

Then at any iteration t, let $W^t = (W_l^t, \quad l \in L_1 \cup L_2 \cup L_3 \cup L_4)$ be a dual feasible solution where the variables W_l^t satisfy the following constraints, *in addition to* [4.12]:

$$-1 \le W_l^t \le 1 \quad \forall \quad l \in L_1 \qquad [4.22]$$

$$W_l^t \text{ unrestricted} \quad \forall \quad l \in L_2 \qquad [4.23]$$

$$W_l^t \text{ unrestricted} \quad \forall \quad l \in L_3 \qquad [4.24]$$

$$W_l^t \ge 0 \quad \forall \quad l \in L_4 \qquad [4.25]$$

Define $D^t = \text{diag}\{D_l^t, \quad l = 1,\ldots,L\}$ [4.26]

where

$$D_l^t = \begin{cases} \min\{(1+W_l^t),(1-W_l^t)\} & \forall \quad l \in L_1 \\ 1 \quad \forall \quad l \in L_2 \\ 1 \quad \forall \quad l \in L_3 \\ W_l^t \quad \forall \quad l \in L_4 \end{cases} \qquad [4.27]$$

Also, let $\delta(W^t) = \{\delta_l(W_l^t) \quad \forall \quad l = 1,\ldots,L\}$ [4.28]

where

$$\delta_l(W_l^t) = \begin{cases} -2D_l^t W_l^t / [1-(W_l^t)^2] & \forall \quad l \in L_1 \\ 0 \quad \forall \quad l \in L_2 \\ 0 \quad \forall \quad l \in L_3 \\ 1 \quad \forall \quad l \in L_4 \end{cases} \qquad [4.29]$$

The associated Barrier Problem (BP) (see Sherali *et al.*, 1988) can then be formulated as follows, where μ is an infinitesimal perturbation:

$$\text{BP: Maximize} \sum_{k=1}^{K}\sum_{j=1}^{n} y_j(k) w_{jk}^{(1)} + \sum_{i=1}^{m} w_i^{(2)} + \sum_{j=1}^{n} \bar{y}_j w_j^{(3)} + \mu\Big[\sum_{k=1}^{K}\sum_{j=1}^{n} \log(1-(w_{jk}^{(1)})^2) + \sum_{i=1}^{m}\sum_{j \in J_i} \log w_{ij}^{(4)}\Big] \qquad [4.30]$$

subject to

$$\sum_{k=1}^{K} q_i(k-\tau_{ij}) w_{jk}^{(1)} + w_i^{(2)} + \bar{q}_{ij} w_j^{(3)} + w_{ij}^{(4)} = 0 \quad \forall \quad j \in J_i \quad i = 1,\ldots,m, \qquad [4.31]$$

Let β represent the vector of dual variables associated with BP. Then the problem BP can be solved using the following algorithm. (For μ sufficiently small, this will recover an optimum to D within any specified accuracy tolerance; see Sherali *et al.*, 1988.)

4.2 Outline of the algorithm

Given a dual feasible iterate W^t having $-1 < W_l^t < 1 \ \forall \ l \in L_1$ and $W_l^t > 0 \ \forall \ l \in L_4$, the variables are scaled using the transformation matrix D^t defined by [4.27] and the associated scaled Barrier problem BP is constructed. The gradient to the objective function in problem BP is projected onto the nullspace of the constraints. The new iterate is found by taking a suitable step-length λ from the current point in the scaled W-space in the direction of the projected gradient. A least squares solution to the Karush-Kuhn-Tucker system gives the dual variables associated with problem BP. Using the inverse transformation, yields the new iterate in the original W- space. The algorithm is terminated when the projected gradient comes within a tolerance limit ε of zero.

To present the algorithm more conveniently, let us restate the dual problem D defined by [4.11]-[4.16] in the form

D: Maximize $\{CW\!: AW = 0$, and $-1 \le w_{jk}^{(1)} \le 1 \ \forall \ k = 1,\ldots,K, j = 1,\ldots,n, w_i^{(2)}$ unrestricted

$\forall \ i = 1,\ldots,m,\ w_j^{(3)}$ unrestricted $\forall \ j = 1,\ldots,n,\ w_{ij}^{(4)} \ge 0 \ \forall \ j \in J_i,\ i = 1,\ldots,m\}$

where

$$C = \left[\left\{y_j(k) \ \forall \ k = 1,\ldots,K, j = 1,\ldots,n,\ 1 \ \forall \ i = 1,\ldots,m,\ \bar{y}_j \ \forall j = 1,\ldots,n,\ 0 \ \forall j \in J_i,\ i = 1,\ldots,m\right\}\right] \qquad [4.32]$$

and where $A = [Q, E, \bar{Q}, I]$ with Q, E, $\bar{Q}$ and I being submatrices defined as follows:

Let $r = \sum_{i=1}^{m} |J_i|$ denote the number of rows in [4.12], and let these rows be arranged in m blocks, having $|J_i|$ rows in block i, for each i=1,...,m. Also, arrange the components of

$w_{jk}^{(1)}$ according to $\begin{bmatrix} w_{1.}^{(1)} \\ \vdots \\ w_{n.}^{(1)} \end{bmatrix}$ where $w_{j.}^{(1)} = \begin{bmatrix} w_{j1}^{(1)} \\ \vdots \\ w_{jK}^{(1)} \end{bmatrix}$ $\forall\ j = 1,\dots,n$, and arrange the components of

$w_{ij}^{(4)}$ as $\begin{bmatrix} w_{1.}^{(4)} \\ \vdots \\ w_{m.}^{(4)} \end{bmatrix}$ where $w_{i.}^{(4)} = [w_{ij}^{(4)}, j \in J_i]$ has $|J_i|$ rows for each i=1,...,m.

Then, we have the following:

(i) $Q = \begin{bmatrix} Q_1 \\ \vdots \\ Q_m \end{bmatrix}$ where each Q_i is a $|J_i| \times nK$ submatrix having rows corresponding to each $j \in J_i$ with components in this row divided into n blocks of the form

$[0,\dots,0,\{q_i(k-\tau_{ij}), k = 1,\dots,K\},0,\dots,0]$

where the non-zero blocks correspond to the particular position j.

(ii) $E = \begin{bmatrix} E_1 \\ \vdots \\ E_m \end{bmatrix}$ where each E_i is a $|J_i| \times m$ submatrix with each row being the unit vector e_i.

(iii) $\overline{Q} = \begin{bmatrix} \overline{Q}_1 \\ \vdots \\ \overline{Q}_m \end{bmatrix}$ where each $\overline{Q}_i$ is a $|J_i| \times n$ submatrix having rows of the form $\overline{q}_{ij} e_j$ corresponding to each $j \in J_i$.

(iv) I is an $r \times r$ identity matrix.

4.2.1 Algorithmic Steps

Initialization:

i) Set the iteration counter t=1.

ii) Initialize the dual variables,

$$w_{jk}^{(1)} = 0 \quad \forall \quad j = 1,\ldots,n, \quad k = 1,\ldots,K,$$

$$w_{i}^{(2)} = -1 \quad \forall \quad i = 1,\ldots,m,$$

$$w_{j}^{(3)} = 0 \quad \forall \quad j = 1,\ldots,n,$$

$$w_{ij}^{(4)} = 1 \quad \forall \quad i = 1,\ldots,m, \quad j \in J_i,$$

Note that this solution is feasible to the dual problem **D.**

Step 1:

Using A and C defined above, find the dual estimate β^t, the Lagrangian gradient vector θ^t for BP, the projected gradient p^t, and the direction of motion d^t in the W-space as follows, where prime denotes the transpose operation.

$$\beta^t = (A(D^t)^2 A')^{-1} AD^t[D^t C' + \mu\delta(W_t)] \qquad [4.33]$$

$$\theta^t = C' - A'\beta^t + \mu(D^t)^{-1}\delta(W_t) \qquad [4.34]$$

$$p^t = D^t\theta^t \qquad [4.35]$$

$$d^t = D^t p^t \qquad [4.36]$$

Step 2 (Termination test):

If $\|p^t\| \leq \varepsilon$ where ε is some termination tolerance, then stop with β^t as a near optimal solution to **LP**. Else, proceed to Step 3.

(Note that depending on the tolerance ε, β^t might only be near feasible. In particular, feasibility to [4.7] and [4.9] can be assured by projecting this solution β^t onto the set β defined in [3.9], via the projection scheme of Section 3.2.1.)

Step 3 (New iterate):

i) Compute the step length λ_t using the following equation

$$\lambda_t = \min \ \{u\lambda_{\max}, \frac{(1-u^2)^2}{4\mu}\} \quad \text{where } u \in [0.97, 0.99] \qquad [4.37]$$

$$\text{and where } 1/\lambda_{\max} = \text{maximum}\left[\max_{l \in L_1}(\max\{\frac{d_l^t}{(1-w_l^t)}, \frac{-d_l^t}{(1+w_l^t)}\}), \max_{l \in L_4}(\frac{-d_l^t}{w_l^t})\right] \qquad [4.38]$$

ii) Compute the new iterate as $W^{t+1} = W^t + \lambda_t d^t$. [4.39]

iii) Set $t \leftarrow t+1$, and re-iterate by returning to Step 1.

Comment:

A relaxation of the problem **LP** can be formulated by disregarding the optional constraints [4.3]. The dual to this new problem, **LPr**, will be smaller in size than D. It will have n fewer variables although the dual constraints will remain unaffected. It may be noted that this formulation will then be comparable to the least squares formulation **(LSQ)** of the problem. The computational effort required to solve the dual to **LPr** may not be very different from that required to solve D, but it would be of interest to compare the solution obtained from **LPr** with the **LSQ** solutions. These two dual problems are also being solved using a commercial simplex based algorithm.

CHAPTER 5
MODEL TESTS AND RESULTS

5.1 Introduction

Two parameter optimization models have been developed in the present research. The efficiency of these models in estimating the split parameter matrix was judged by making test runs on three main types of data sets. These data sets were generated by simulating the real traffic conditions on a small freeway section or an intersection. The three criteria used to study the efficiency and robustness of these models are: 1) total run-time, 2) closeness of the solution obtained to the actual optimum and, 3) the number of iterations needed before optimality is declared.

5.2 Computer Resources for the Test Runs

The optimization methods for estimating the split parameter matrix involve complex iterative algorithms. In the present research, the solution techniques used to solve the constrained least squares formulations **LSQst-1** and **LSQst-2**, and the L_1 norm formulation **LP**, require matrix manipulation types of operations. Matlab 4.0 is a software package known for its computational efficiency in performing matrix operations. This package was used on a 486/DX2-66 personal computer to perform the tests. AMPL, which is a modeling language for mathematical programming problems, was also employed to obtain the optimal solutions to the test problems. AMPL links to the optimization package MINOS for solving linear and nonlinear mathematical programs. For problems that are nonlinear in objective but linear in the constraints, MINOS employs a reduced gradient approach. An iteration attempts to reduce the objective within the subspace of basic and superbasic variables, employing a quasi-Newton algorithm to select a search direction. An inexact line search is performed to determine the prescribed step-length. AMPL sends MINOS the information it needs to compute the values and partial derivatives of all nonlinear terms in the objective function and constraints. The method of "automatic differentiation" is employed to compute the derivatives efficiently and exactly. For linear programs, MINOS reduces to the primal simplex algorithm.

5.3 Sample Freeway Sections and Intersections

Three main categories of the test problems were generated according to the following scenarios.

Case I: Travel time between each entrance-exit pair is zero (case of an intersection).

Case II: Travel time between each entrance-exit pair is nonzero and all exits are accessible from each entrance (case of a small freeway section).

Case III: Travel time between each entrance-exit pair is nonzero and not all exits are necessarily accessible from each entrance (case of a closed ramp on the freeway or a 'no left turn' scenario for an intersection).

A traffic facility with 3 entrances and 3 exits is considered. Such a geometry is representative of both a small freeway section or an intersection with some one-way streets. Tests were made for $K = 60$, 80, 100 and 120 time periods for each of the three cases mentioned above.

5.4 Test Problems

The following parameter values were used for generating the test problems.

Cycle time = 50 seconds

Number of entrances = 3

Number of exits = 3

Total entry volume at entrance 1 = 6000 vehicles
Total entry volume at entrance 2 = 6000 vehicles
Total entry volume at entrance 3 = 1000 vehicles

Case I: Travel time on each link is zero

Table 5.1: Travel time matrix for Case I

exit *j* / entrance *i*	1	2	3
1	0	0	0
2	0	0	0
3	0	0	0

Case II: Travel time on each link is nonzero and all exits are accessible from each entrance

Table 5.2: Travel time matrix for Case II

exit *j* / entrance *i*	1	2	3
1	1	3	5
2	1	1	3
3	3	1	1

Case III: Travel time on each link is nonzero and Exit 1 is not accessible from Entrance 3

Table 5.3: Travel time matrix for Case III (negative values indicate non-existent routes)

exit *j* / entrance *i*	1	2	3
1	1	3	5
2	1	1	3
3	-1	1	1

5.5 Test Results

Tests were made on the two least squares solution methods, *viz.*, **LSQst-1** and **LSQst-2,** and the linear programming model **LP**, using the data generated from the test problems formulated above. Besides these, a relaxation of the model **LP** in which the constraint [4.3] was relaxed, was also tested. This formulation is referred to as **LPr** in the following sections.

The results obtained from test runs made on **LSQst-1** and **LSQst-2** are given below in Section 5.5.1. Section 5.5.2 presents a comparison between results obtained from the test runs on model **LP** and its relaxed form **LPr**.

5.5.1 Results Obtained from Test Runs on LSQst-1 and LSQst-2

Both the algorithms, **LSQst-1** and **LSQst-2**, were tested using the initial starting solutions **INIT-1** and **INIT-2.** The algorithm **LSQst-2** performed extremely well under both situations. The solutions obtained in both the cases exactly matched the optimum obtained using AMPL. When tested using **INIT-2,** the computational time required by **LSQst-2,** increased marginally as compared to tests with initial start **INIT-1**. **LSQst-1** performed well using **INIT-1** and did not exhibit significant improvement when tested with **INIT-2.**

A comparison of the initial objective values using initial solutions obtained from **INIT-1** and **INIT-2** is given in Table 5.4. It may be noted that **INIT-2** gives a more advanced start to the algorithm.

All four schemes mentioned in Section 3.3.1.1(ii) for finding the deflected direction were tested using both the conjugate gradient solution techniques. It was found that strategy 2 gave the most favorable results when used with **LSQst-1**, as might be expected from the quadratic nature of the objective function. Note that this strategy due to Sherali and Ulular, (1990) reduces to Hestenes and Steifel's, (1952) choice of selecting deflection parameters under exact line searches. It was also observed that strategy 3 gave better results in comparison to strategy 2, when used with **LSQst-2**. In actuality, since we are working with optimizing a quadratic objective function along with exact line searches over projected subspaces in this procedure, strategy 3 which is Fletcher and Reeves', (1964) is equivalent to Hestenes and Steifel's (1952) and Sherali and Ulular's (1990) formulas. However, it has a simpler computational form Hence, in the test runs, we have used these strategies in each case, respectively, for finding the deflected direction. The effect of the parameters governing travel time delays and number of time periods on the computational efficiency of the approaches, the accuracy of the solutions obtained, and the number of iterations required, have also been recorded.

The results obtained from solving LSQ using AMPL were very promising. In almost all cases the computational time required to attain optimality was between 1-3 seconds, and the number of iterations required ranged from 12 to 15.

Table 5.4: Objective Function values for initial starting solutions obtained using INIT-1 and INIT-2 for Cases I, II and III for *K*=60, 80, 100, & 120

	Case I	Case II	Case III
***K*=60**			
Initial Objective Value (INIT-1)	1.3532×10^4	1.2730×10^4	3.7447×10^4
Initial Objective Value (INIT-2)	49.7874	223.1970	442.1393
***K*=80**			
Initial Objective Value (INIT-1)	2.0763×10^4	1.2978×10^4	3.8339×10^4
Initial Objective Value (INIT-2)	60.2107	228.6839	432.8570
***K*=100**			
Initial Objective Value (INIT-1)	1.3644×10^4	1.3185×10^4	3.5779×10^4
Initial Objective Value (INIT-2)	60.4798	224.1850	415.4939
***K*=120**			
Initial Objective Value (INIT-1)	1.3648×10^4	2.0191×10^4	3.0832×10^4
Initial Objective Value (INIT-2)	62.1190	226.3594	467.5126

The following tables show a comparison of the estimated objective value ($f_{estimated}$), the computational time and the number of iterations required by each of the models for K^{*} = 60, 80, 100 and 120 time periods, respectively. The optimal objective value (f_{opt}) for each case, as obtained from AMPL, is also given.

Table 5.5: **Comparison of estimated objective values, computational time, and number of iterations for $K = 60$, for LSQst-1 using INIT-1 and LSQst-2 using (INIT-1, INIT-2)**

	LSQ st-1			LSQ st-2		
	Case I	**Case II**	**Case III**	**Case I**	**Case II**	**Case III**
f_{opt}	47.9663	51.4006	59.4959	47.9663	51.4006	59.4959
$f_{estimated}$	47.9663	52.0300	59.5114	47.9663 47.9663	51.4007 51.4007	59.4960 59.4960
Computational Time (sec)	2.75	7.58	8.19	2.85 3.24	7.30 8.47	5.71 6.09
Number of Iterations	3	7	8	(3, 3)	(7, 8)	(9, 10)
Resetting Counter (*nrst*)	1	1	1	(1, 1)	(2, 2)	(1,1)
Termination Criteria	TRMc-1b	TRMc-3	TRMc-3	TRMc-1b	TRMc-3	TRMc-3

Table 5.6: **Comparison of estimated objective values, computational time, and number of iterations for $K = 80$, for LSQst-1 using INIT-1 and LSQst-2 using (INIT-1, INIT-2)**

	LSQ st-1			LSQ st-2		
	Case I	**Case II**	**Case III**	**Case I**	**Case II**	**Case III**
f_{opt}	56.7943	57.1540	76.4173	56.7943	57.1540	76.4173
$f_{estimated}$	56.7943	57.5572	76.4333	56.7943 56.7943	57.2785 57.2785	76.4173 76.4173
Computational Time (sec)	3.84	9.61	10.55	3.90 3.73	9.28 10.22	7.42 7.91
Number of Iterations	3	7	8	(3, 3)	(7, 7)	(5, 5)
Resetting Counter (*nrst*)	1	1	1	(1, 1)	(2, 2)	(1, 1)
Termination Criteria	TRMc-1b	TRMc-3	TRMc-3	TRMc-1b	TRMc-3	TRMc-3

Table 5.7: **Comparison of estimated objective values, computational time, and number of iterations for $K = 100$, for LSQst-1 using INIT-1 and LSQst-2 using (INIT-1, INIT-2)**

	LSQ st-1			LSQ st-2		
	Case I	**Case II**	**Case III**	**Case I**	**Case II**	**Case III**
f_{opt}	56.2789	56.8650	76.3525	56.2789	56.8650	76.3525
$f_{estimated}$	56.2789	57.2227	76.3550	56.2789 56.2789	56.8650 56.8650	76.3526 76.3526
Computational Time (sec)	4.45	11.70	13.13	6.43 6.98	12.08 13.45	8.84 9.44
Number of Iterations	3	7	8	(3, 3)	(7, 8)	(5, 5)
Resetting Counter (*nrst*)	1	1	1	(1, 1)	(2, 2)	(1, 1)
Termination Criteria	TRMc-1b	TRMc-3	TRMc-3	TRMc-1b	TRMc-3	TRMc-3

Table 5.8: **Comparison of estimated objective values, computational time, and number of iterations for $K = 100$, for LSQst-1 using INIT-1 and LSQst-2 using (INIT-1, INIT-2)**

	LSQ st-1			LSQ st-2		
	Case I	**Case II**	**Case III**	**Case I**	**Case II**	**Case III**
f_{opt}	57.1667	60.0901	79.9260	57.1667	60.0901	79.9260
$f_{estimated}$	57.1667	60.4355	79.9262	57.1667 57.1667	60.0902 60.0902	79.9260 79.9260
Computational Time (sec)	7.75	13.73	13.73	7.36 8.18	13.67 16.04	10.60 11.37
Number of Iterations	3	7	7	(3, 3)	(7, 7)	(5, 5)
Resetting Counter (*nrst*)	1	1	1	(1, 1)	(2, 2)	(1, 1)
Termination Criteria	TRMc-1b	TRMc-3	TRMc-3	TRMc-1b	TRMc-3	TRMc-3

Tables 5.9-5.12 present the optimal and the estimated split parameter matrices for Cases I, II and III for K = 60, 80, 100 and 120 time periods, for each of the two least squares solution procedures.

Table 5.9 (a): Case I: Optimal and estimated split parameter matrices for K = 60, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit *j*	1	2	3
entrance *i*			
1	0.2224	0.4312	0.3465
2	0.2034	0.3964	0.4002
3	0.3959	0.0520	0.5521

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit *j*	1	2	3
entrance *i*			
1	0.2224	0.4312	0.3465
2	0.2034	0.3964	0.4002
3	0.3959	0.0520	0.5521

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.2224	0.4312	0.3465
2	0.2034	0.3964	0.4002
3	0.3959	0.0520	0.5521

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.2224	0.4312	0.3465
2	0.2034	0.3964	0.4002
3	0.3959	0.0520	0.5521

Table 5.9 (b):Case II: Optimal and estimated split parameter matrices for K = 60, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2386	0.4937	0.2678
2	0.2054	0.3208	0.4738
3	0.2752	0.1487	0.5762

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2408	0.4743	0.2849
2	0.2040	0.3166	0.4794
3	0.2744	0.2927	0.4329

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2386	0.4937	0.2678
2	0.2054	0.3208	0.4738
3	0.2752	0.1487	0.5762

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2386	0.4937	0.2678
2	0.2054	0.3208	0.4738
3	0.2752	0.1487	0.5762

Table 5.9 (c):Case III: Optimal and estimated split parameter matrices for K=60, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2517	0.4928	0.2493
2	0.2376	0.2606	0.5017
3	0.0000	0.4941	0.5058

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2576	0.4926	0.2498
2	0.2378	0.2604	0.5018
3	0.0000	0.4980	0.5020

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2577	0.4929	0.2494
2	0.2376	0.2606	0.5018
3	0.0000	0.4942	0.5058

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2577	0.4929	0.2494
2	0.2376	0.2606	0.5018
3	0.0000	0.4942	0.5058

Table 5.10 (a): Case I: Optimal and estimated Split parameter matrices for $K = 80$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2906	0.4489	0.2605
2	0.1596	0.3753	0.4652
3	0.2705	0.0327	0.6968

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2906	0.4489	0.2605
2	0.1596	0.3753	0.4652
3	0.2705	0.0327	0.6968

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2906	0.4489	0.2605
2	0.1596	0.3753	0.4652
3	0.2705	0.0327	0.6968

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2906	0.4489	0.2605
2	0.1596	0.3753	0.4652
3	0.2705	0.0327	0.6968

Table 5.10 (b):Case II: Optimal and estimated split parameter matrices for $K = 80$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2703	0.4978	0.2318
2	0.1856	0.3014	0.5130
3	0.2256	0.1886	0.5858

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2745	0.5018	0.2237
2	0.1781	0.3107	0.5112
3	0.2502	0.1029	0.6469

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2703	0.4979	0.2318
2	0.1856	0.3014	0.5130
3	0.2256	0.1886	0.5858

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2703	0.4979	0.2318
2	0.1856	0.3014	0.5130
3	0.2256	0.1886	0.5858

Table 5.10 (c): Case III: Optimal and estimated Split parameter matrices for $K = 80$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2568	0.4979	0.2452
2	0.2387	0.2636	0.4977
3	0.0000	0.4843	0.5157

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2569	0.4953	0.2462
2	0.2386	0.2637	0.4977
3	0.0000	0.4880	0.5120

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2568	0.4979	0.2452
2	0.2387	0.2636	0.4977
3	0.0000	0.4843	0.5157

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2568	0.4979	0.2452
2	0.2387	0.2636	0.4977
3	0.0000	0.4843	0.5157

Table 5.11 (a): Case I: Optimal and estimated split parameter matrices for $K = 100$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit *j*	1	2	3
entrance *i*			
1	0.2957	0.4530	0.2513
2	0.1761	0.3621	0.4618
3	0.1556	0.0747	0.7697

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit *j*	1	2	3
entrance *i*			
1	0.2957	0.4530	0.2513
2	0.1761	0.3621	0.4618
3	0.1556	0.0747	0.7697

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.2957	0.4530	0.2513
2	0.1761	0.3621	0.4618
3	0.1556	0.0747	0.7697

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit *j*	1	2	4
entry *i*			
1	0.2957	0.4530	0.2513
2	0.1761	0.3621	0.4618
3	0.1556	0.0747	0.7697

Table 5.11 (b): Case II: Optimal and estimated split parameter matrices for K= 100, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit *j*	1	2	3
entrance *i*			
1	0.2762	0.4992	0.2246
2	0.1796	0.2933	0.5270
3	0.2453	0.2131	0.5416

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit *j*	1	2	3
entrance *i*			
1	0.2665	0.4911	0.244
2	0.1853	0.2857	0.5291
3	0.2723	0.3138	0.4138

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.2762	0.4992	0.2246
2	0.1796	0.2933	0.5270
3	0.2453	0.2131	0.5416

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.2762	0.4992	0.2246
2	0.1796	0.2933	0.5270
3	0.2453	0.2131	0.5416

Table 5.11 (c): Case III: Optimal and estimated split parameter matrices for $K = 100$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2622	0.4874	0.2503
2	0.2371	0.2661	0.4968
3	0.0000	0.4897	0.5103

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2622	0.4872	0.2506
2	0.2371	0.2660	0.4969
3	0.0000	0.4916	0.5084

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2622	0.4875	0.2503
2	0.2371	0.2661	0.4968
3	0.0000	0.4897	0.5103

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2622	0.4875	0.2503
2	0.2371	0.2661	0.4968
3	0.0000	0.4897	0.5103

Table 5.12 (a): Case I: Optimal and estimated split parameter matrices for $K = 120$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit *j*	1	2	3
entrance *i*			
1	0.3038	0.4673	0.2288
2	0.1544	0.3572	0.4884
3	0.2492	0.0175	0.7332

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit *j*	1	2	3
entrance *i*			
1	0.3038	0.4673	0.2288
2	0.1544	0.3572	0.4884
3	0.2492	0.0175	0.7332

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.3039	0.4673	0.2288
2	0.1544	0.3572	0.4884
3	0.2492	0.0179	0.7329

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit *j*	1	2	3
entry *i*			
1	0.3038	0.4673	0.2288
2	0.1544	0.3572	0.4884
3	0.2492	0.0175	0.7332

Table 5.12 (b): Case II: Optimal and estimated split parameter matrices for $K = 120$, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2714	0.4978	0.2307
2	0.1904	0.2929	0.5166
3	0.2188	0.2300	0.5512

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2672	0.4835	0.2494
2	0.1875	0.2942	0.5182
3	0.2640	0.3129	0.4231

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2714	0.4978	0.2307
2	0.1904	0.2930	0.5166
3	0.2188	0.2300	0.5512

Estimated Split Parameter Matrix Using INIT-2 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2714	0.4978	0.2307
2	0.1904	0.2930	0.5166
3	0.2188	0.2300	0.5512

Table 5.12 (c): Case III: Optimal and estimated split parameter matrices for K = 120, for LSQst-1 and LSQst-2

Optimum Split Parameter Matrix (AMPL)			
exit j	1	2	3
entrance i			
1	0.2570	0.4946	0.2483
2	0.2388	0.2611	0.5000
3	0.0000	0.5133	0.4867

Esimated Split Parameter Matrix Using INIT-1 for LSQst-1			
exit j	1	2	3
entrance i			
1	0.2570	0.4947	0.2484
2	0.2389	0.2612	0.5001
3	0.0000	0.5133	0.4867

Estimated Split Parameter Matrix Using INIT-1 for LSQst-2			
exit j	1	2	3
entry i			
1	0.2570	0.4946	0.2483
2	0.2388	0.2611	0.5000
3	0.0000	0.5133	0.4867

Estimated Split Parameter Matrix Using INIT-2 for LSQst-1			
exit j	1	2	3
entry i			
1	0.2570	0.4946	0.2483
2	0.2388	0.2611	0.5000
3	0.0000	0.5133	0.4867

5.5.2 Results Obtained from Test Runs on LP and LPr

The effect of the parameters governing travel time delays and number of time periods on the computational efficiency of the approaches, the accuracy of the solutions obtained, and the number of iterations required, was observed.

The duals to problems **LP** and **LPr**, respectively, were also solved using AMPL. Although the number of iterations required to solve each problem ranged between 200 and 300, the computational time required was between 3-6 seconds for both **LP** and **LPr**.

The following tables show a comparison of the estimated primal objective value ($f_{estimated}$), the computational time, and the number of iterations required by each of the models **LP** and **LPr**, for K = 60, 80, 100 and 120 time periods, respectively. Here, $f_{estimated}$ is the objective value obtained from **LP** and **LPr** when optimality is attained and is referred to as '$f_{estimated}$ **(LP)**' in the tables. The corresponding optimal **(LP)** objective values (f_{opt}), as obtained from AMPL, are also given. These are referred to as 'f_{opt} **(LP)**' in the tables. Since the formulation of **LPr** is comparable with the least squares formulation of the problem, we have also computed the average least squares errors for the split parameter matrices obtained from solving **LPr**, in each of the four test problem sizes. For each test problem, these least squares estimates, $f_{estimated}$ **(NLP)**, along with the corresponding optimal nonlinear objective function value, f_{opt} **(NLP)**, as obtained from AMPL, are presented in Tables 5.13-5.16.

It may be noted that since a student version of AMPL was being used, we were unable to solve the problems **LP** and **LPr** for K= 100 and 120 time periods. The sizes of the matrices generated in solving these particular problems exceeded the maximum allowable matrix size for this scaled down version of AMPL. We can, however, obtain these results using a professional version of the software.

Table 5.13: Comparison of estimated objective values, computational time, and number of iterations for K = 60, for LP and LPr

	LP			LPr		
	Case I	Case II	Case III	Case I	Case II	Case III
f_{opt} (NLP)				47.9663	51.4006	59.4959
$f_{estimated}$ (NLP)				51.0794	52.6972	62.2696
f_{opt} (LP)	6.8410	6.8068	7.4891	6.8100	6.7191	7.3907
$f_{estimated}$ (LP)	6.8447	6.8068	7.5683	7.0053	6.7253	7.4349
Computational Time (secs)	16.65	19.22	17.58	21.42	19.11	16.43
Number of Iterations	31	34	32	40	35	30

Table 5.14: Comparison of estimated objective values, computational time, and number of iterations for K = 80, for LP and LPr

	LP			LPr		
	Case I	Case II	Case III	Case I	Case II	Case III
f_{opt} (NLP)				56.7943	57.1540	76.4173
$f_{estimated}$ (NLP)				58.1647	59.0514	81.6774
f_{opt} (LP)	7.4455	7.3209	8.2118	7.4153	7.2372	8.1807
$f_{estimated}$ (LP)	7.4507	7.4540	8.3808	7.4354	7.2410	8.3062
Computational Time (secs)	29.82	34.22	35.53	29.28	36.75	27.19
Number of Iterations	39	43	46	38	47	35

Table 5.15: **Comparison of estimated objective values, computational time, and number of iterations for K = 100, for LP and LPr**

	LP			LPr		
	Case I	Case II	Case III	Case I	Case II	Case III
f_{opt} (NLP)				56.2789	56.8650	76.3525
$f_{estimated}$ (NLP)				56.9284	59.2258	78.4092
f_{opt} (LP)	NA	NA	NA	NA	NA	NA
$f_{estimated}$ (LP)	7.3866	7.2797	8.3120	7.3761	7.2333	8.2040
Computational Time (secs)	27.96	61.25	30.10	30.54	62.07	25.59
Number of Iterations	26	58	28	29	60	24

Table 5.16: **Comparison of estimated objective values, computational time, and number of iterations for K = 120, for LP and LPr**

	LP			LPr		
	Case I	Case II	Case III	Case I	Case II	Case III
f_{opt} (NLP)				57.1667	60.0901	79.9260
$f_{estimated}$ (NLP)				58.0788	61.5147	84.0472
f_{opt} (LP)	NA	NA	NA	NA	NA	NA
$f_{estimated}$ (LP)	7.4210	7.4326	8.4430	7.4221	7.4502	8.3949
Computational Time (secs)	30.32	79.25	60.42	44.55	88.21	44.44
Number of Iterations	22	58	45	33	66	33

Tables 5.17-5.20 show the optimal split parameter matrices, as obtained from AMPL and a comparison between the estimated split parameter matrices obtained from solving **LP** and **LPr**, respectively. These results are obtained from test runs using data sets generated for Cases I, II and III for K= 60, 80, 100 and 120 time periods. The AMPL results are not available for K=100 and 120 time periods due to the limitations on permissible matrix sizes in the student version of AMPL.

Table 5.17(a): Case I: Optimal and estimated split parameter matrices for K = 60, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	0.1894	0.4650	0.3455
2	0.2495	0.3679	0.3825
3	0.3125	0.0276	0.6599
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2160	0.4573	0.3266
2	0.2210	0.3802	0.3989
3	0.3275	0.0000	0.6725

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	0.1838	0.4569	03593
2	0.2445	0.3846	0.3709
3	0.3448	0.0000	0.6555
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.1772	0.4615	0.3613
2	0.2368	0.3947	0.3685
3	0.3865	0.0000	0.6135

Table 5.17(b): Case II: Optimal and estimated split parameter matrices for K=60, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	0.2127	0.5000	0.2873
2	0.2657	0.3092	0.4254
3	0.0710	0.1759	0.7530
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2127	0.5000	0.2873
2	0.2654	0.3092	0.4254
3	0.0710	0.1759	0.7530

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	0.2295	0.5000	0.2705
2	0.2364	0.3155	0.4480
3	0.4762	0.1815	0.7422
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2458	0.5000	0.2542
2	0.2227	0.3156	0.4617
3	0.0685	0.1816	0.7500

Table 5.17(c): Case III: Optimal and estimated split parameter matrices for K = 60. for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	0.2634	0.4703	0.2663
2	0.2354	0.2662	0.4984
3	0.0128	0.5165	0.4706
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2634	0.4918	0.2447
2	0.2355	0.2465	0.5181
3	0.0000	0.5227	0.4773

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	0.2730	0.4819	0.2450
2	0.2347	0.2654	0.4998
3	0.0408	0.4804	0.4787
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2697	0.4852	0.2450
2	0.2350	0.2652	0.4998
3	0.0000	0.5213	0.4787

Table 5.18(a): Case I: Optimal and estimated split parameter matrices for $K = 80$, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	0.2473	0.5152	0.2375
2	0.1994	0.3104	0.4902
3	0.2888	0.0271	0.6841
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2413	0.5168	0.2419
2	0.1987	0.3126	0.4887
3	0.3289	0.0044	0.6667

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	0.2667	0.4871	0.2462
2	0.1655	0.3487	0.4858
3	0.3365	0.0000	0.6635
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2903	0.4677	0.2420
2	0.1396	0.3738	0.4865
3	0.3288	0.0000	0.6712

Table 5.18(b): Case II: Optimal and estimated split parameter matrices for $K = 80$, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	0.2586	0.5259	0.2155
2	0.2143	0.2614	0.5242
3	0.1191	0.2548	0.6261
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2027	0.5818	0.2155
2	0.2904	0.2123	0.4973
3	0.0000	0.2056	0.7944

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	0.2586	0.5259	0.2155
2	0.2117	0.2791	0.5091
3	0.0643	0.1940	0.7416
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2586	0.5259	0.2155
2	0.2119	0.2753	0.5128
3	0.0624	0.2283	0.7093

Table 5.18(c): Case III: Optimal and estimated split parameter matrices for K = 80. for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	0.2639	0.4802	0.2558
2	0.2354	0.2658	0.4987
3	0.0028	0.5029	0.4943
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2639	0.4818	0.2543
2	0.2354	0.2724	0.4922
3	0.0000	0.4467	0.5533

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	0.2694	0.4878	0.2428
2	0.2350	0.2658	0.4991
3	0.0000	0.4733	0.5267
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2802	0.4762	0.2436
2	0.2342	0.2657	0.5001
3	0.0000	0.5276	0.4724

Table 5.19(a): Case I: Optimal and estimated split parameter matrices for K = 100, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2942	0.5061	0.1997
2	0.1898	0.2868	0.5234
3	0.0778	0.2114	0.7108

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2802	0.4999	0.2200
2	0.1972	0.2973	0.5055
3	0.0989	0.1931	0.7083

Table 5.19(b): Case II: Optimal and estimated split parameter matrices for K= 100

Optimum Split Parameter Matrix for LP (AMPL)			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2466	0.5391	0.2143
2	0.2350	0.2442	0.5208
3	0.0860	0.2641	0.6499

Optimum Split Parameter Matrix for LPr (AMPL)			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2688	0.5178	0.2143
2	0.1929	0.2749	0.5322
3	0.1141	0.2413	0.6447

Table 5.19(c): Case III: Optimal and estimated split parameter matrices for K = 100, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2726	0.4735	0.2539
2	0.2286	0.2730	0.4984
3	0.0000	0.4796	0.5204

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2697	0.4735	0.2568
2	0.2350	0.2661	0.4990
3	0.0000	0.5214	0.4786

Table 5.20(a): Case I: Optimal and estimated split parameter matrices for K = 120, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2747	0.5087	0.2166
2	0.1899	0.2933	0.5168
3	0.2070	0.1594	0.6336

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2668	0.5134	0.2198
2	0.2006	0.2844	0.5149
3	0.1755	0.1947	0.6298

Table 5.20(b): Case II: Optimal and estimated split parameter matrices for K = 120, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2729	0.4669	0.2603
2	0.2088	0.2983	0.4929
3	0.0941	0.3932	0.5127

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2766	0.4574	0.2660
2	0.2154	0.3027	0.4820
3	0.0211	0.4465	0.5325

Table 5.20(c): Case III: Optimal and estimated split parameter matrices for K = 120, for LP and LPr

Optimum Split Parameter Matrix for LP using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LP			
exit j	1	2	3
entry i			
1	0.2644	0.4876	0.2480
2	0.2354	0.2554	0.5092
3	0.0000	0.5251	0.4749

Optimum Split Parameter Matrix for LPr using AMPL			
exit j	1	2	3
entry i			
1	NA	NA	NA
2	NA	NA	NA
3	NA	NA	NA
Estimated Split Parameter Matrix Using LPr			
exit j	1	2	3
entry i			
1	0.2694	0.4904	0.2402
2	0.2350	0.2539	0.5111
3	0.0000	0.5247	0.4753

CHAPTER 6
CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH

Two parameter optimization models have been proposed in the present research. The first one is a constrained least squares problem, and the second an L_1 estimation problem, posed as a linear programming problem. Two projected conjugate gradient schemes are suggested for solving the least squares problem. In both these schemes, deflected anti-gradient directions are employed and exact line searches are performed in closed form over projected subspaces. The solution to the linear programming problem is found by solving the specially structured dual to this problem using an affine scaling interior point algorithm, as well as by using a commercial simplex algorithm package.

In the following sections we present a discussion of the test results, and our conclusions and recommendations for future research in this area.

6.1 Conclusions

The results obtained from the tests runs on the two solution techniques for solving the constrained least squares problems are very promising. For the simple case of an intersection, where the travel time from one arm to another arm of the intersection is considered zero, the split parameter estimates obtained using the suggested solution techniques were an exact replication of the optimal split parameter matrices obtained from solving the problem using AMPL. For test problem sizes ranging between 60 to 120 time periods, the corresponding O-D trip-tables were generated within 3-6 seconds. It was observed that the value of the least squares error between the actual exiting volumes and the estimated model exiting volumes became larger as the number of time periods was increased or if travel time delays were introduced. If an O-D trip-table can be estimated within the time duration of one time period, or in other words, if the O-D matrix is updated each time period, the procedure used for generating the O-D matrix can be implemented for on-line traffic control. It is important to note here that to be able to capture the variations in traffic patterns over time, the cycle times considered should be small. The computational times needed by the proposed conjugate gradient schemes for accurately estimating the split parameter matrices in the presence of travel time delays ranged between 7-14 seconds. This computation time, although more than the time required by the 'zero-travel-time' case, is small enough and appreciably less than 50 seconds, which is the cycle time for the simulated test problems. This renders the procedures suitable for implementation in real-time strategies.

It was observed that both **LSQst-1** and **LSQst-2** performed equally well for all three types of data sets. However, **LSQst-2** always produced a solution of quality (final objective value) at least as good as that produced by **LSQst-1**, while consuming only marginally more effort. Hence, overall we can recommend **LSQst-2** for implementation. The number of iterations needed to reach a near optimal solution was small (between 3-8 iterations in case of both **INIT-1** and **INIT-2** starting solutions). This reflects on the computational efficiency of the suggested procedures. The starting solution technique **INIT-2** is recommended because it gives a better initial starting value to the algorithm at the cost of only a marginal increase in computational effort.

The quality of the split parameter estimates obtained as a result of solving the duals to **LP** and **LPr** was very accurate. The estimated least absolute error between the actual exiting volumes and the exiting volumes of the model ranged between 6.7 and 7.5. In each case the split parameter estimates were computed in about 1 minute. This computational time is almost equal to the cycle time considered for generating the data sets for our test problems, insuring a new O-D trip-table almost every time period. However, with a professional simplex based routine applied to the dual problem (AMPL), these times were reduced to a range between 1-3 secs. Hence, the suggested linear programming model can also be used for on-line traffic control purposes, but with an appropriate solution procedure choice or time period duration value.

6.2 Comparison with Existing Parameter Optimization Techniques

The existing literature emphasizes the accuracy of the estimates obtained using parameter optimization methods. But it also points out that due to the extent of computational effort required to solve these problems, the time needed to derive the O-D matrices is impractically high. As stated by Cremer and Keller (1987), the statistical methods require less computational time due to their recursive nature, bearing an advantage over the parameter optimization methods. A computational time of 1 sec by the recursive scheme, 2 secs by the Kalman Filter method and about 4 secs by the correlation method of Cremer and Keller (1987), is considered acceptable for real-time implementation, according to the existing literature. For the recursive scheme and the Kalman Filter method the above mentioned times are the times taken to obtain new estimates recursively *every time period* and for the correlation method it is the time taken for obtaining new estimates *every 20 time periods* using information of only K=20 past time periods. The recursive scheme and the Kalman Filter method were able to produce replications of the real split parameters only after 20 time intervals. These times are reported from test runs on an HP-1000 computer. The traffic facility tested is a 4-arm intersection (number of entrances=4 and number of exits=4) considering a cycle time of 70 seconds. Also, it is important to mention at this point that no results have been reported on data incorporating travel time delays.

In the present research we have overcome the computational inefficiency of the parameter optimization methods by applying fast converging conjugate gradient based algorithms to the problem. Our formulations also incorporate the travel time delays that were almost ignored in the other research efforts in this area. This is an important factor which makes our methods applicable to linear freeway sections. Cremer and Keller, (1987) have stated that their formulations can incorporate travel-time delays on the cost of increased computational effort, although, no computational results have been displayed on this account.

Preliminary computations and comparisons with the existing parameter optimization procedures indicate that the proposed projected conjugate gradient schemes perform significantly better than the existing methods used for solving the constrained least squares formulation for the O-D trip-table estimation problem. The linear programming formulation gave results that were comparable with the existing methods. We used an interior point affine scaling algorithm to solve the dual to the L_1 estimation. Preliminary tests were done on the linear programming formulation using AMPL on a 486/DX2-66 personal computer, which gave extremely favorable results in less than 3 secs. The algorithms used by AMPL to solve linear programs are simplex based algorithms.

6.3 Recommendations for Implementation

Noting the accuracy of the results and the low computational time required to estimate the split parameters, we recommend implementing the two projected conjugate gradient based algorithms for real-time O-D trip-table estimation, and in particular, algorithm **LSQst-2** using **INIT-2** for finiding the initial feasible solution. This approach would be particularly useful for predicting traffic variations on linear freeway sections, enabling control of traffic at an entrance ramp itself (using traffic signals or Variable Message Signs (VMS)), referred to as *Ramp Metering* in the IVHS community. The regulation of traffic and the prescription of diversion strategies in the case of an incident on a ramp would become faster and more user oriented.

6.4 Recommendations for Future Research

Tests on real traffic data need to be conducted in order to fully validate the model. Algorithmic and computational improvements are needed to further reduce the computational time required by the linear programming formulation. This is a challenging avenue for further research since no such formulation is reported in the literature on dynamic methods of O-D estimation, so far, although linear programming methods for static O-D estimation problems do exist. It would also be interesting to test the model using travel times, varying over the number of time periods considered, instead of fixing these delays to constant values. This would simulate the realistic case of transition from non-peak hour traffic to peak-hour and back to non-peak hour. During this period, the travel times between an entrance-exit pair may gradually increase (due to traffic congestion) and then drop back to a lower value. Such a scenario can be simulated by giving a Normal distribution to the travel time delays over the number of time periods considered.

The next logical extension of the model would be the application of dynamic O-D estimation to larger complex networks. For this, decomposition strategies or heuristics are needed that would link together smaller sections of the network (similar to small freeway sections), for which the O-D matrices can be estimated using the procedures suggested in the present research, in order to formulate a pattern of movement over the entire network. These strategies should be designed to allow all computations to be performed in real-time.

BIBLIOGRAPHY

1. Barnes, E. R., (1986), "A Variation of Karmarkar's Algorithm for Solving Linear Programming Problems," *Mathematical Programming,* 36, pp. 123-134.

2. Bazaraa, M. S., Jarvis, J. J., Sherali, H. D., (1990), "*Linear Programming and Network Flows*", John Wiley & Sons.

3. Bazaraa, M. S., Sherali, H. D., Shetty, C. M., (1993), "*Nonlinear Programming: Theory and Algorithms*", second edition, John Wiley & Sons.

4. Bell, Michael G. H., (1991), "The Real-Time Estimation of Origin-Destination Flows in the Presence of Platoon Dispersion", *Transportation Research,* 25 B(2/3), pp 115-125.

5. Bitran, G., and Hax, A., (1976), "On the Solution of Convex Knapsack Problems with Bounded Variables", *Proceedings of IXth International Symposium on Mathematical Programming,* Budapest, pp 357-367.

6. Box, M. J., (1965), "A New Method of Constrained Optimization and a Comparison with Other methods", *Comput. Journal*, 8, pp 42-52.

7. Choi, G. and Sherali, H. D., (1993), "Subgradient Deflection Strategies and a New Variable Target Value Method", working paper, Dept. of Industrial & Systems Engineering, VPI & SU, Blacksburg, Va. 24061-0118.

8. Cremer, M. and Keller, H., (1981), "Dynamic Identification of Flows from Traffic Counts at Complex Intersections", *Eighth International Symposium on Transportation and Traffic Theory*, pp 121-142.

9. Cremer, M. and Keller, H., (1984), "A Systems Dynamic Approach to the Estimation of Entry and Exit O-D Flows", *Ninth International Symposium on Transportation and Traffic Theory*, pp 431-450.

10. Cremer, M. and Keller, H., (1987), "A New Class of Dynamic Methods for the Identification of Origin-Destination Flows", *Transportation Research* - B, 25 B(2/3), pp 117-132.

11. Fletcher, R., and Reeves, C., (1964), " Function Minimization by Conjugate Gradients", *Computer Journal* 7, pp 149-154, 1964.

12. Fourier, R., Gay, D. M. and Kernighan, B. W., (1993), "*AMPL A Modeling Language for Mathematical Programming*", The Scientific Press.

13 Hestenes, M. R., and Steifel, E., (1952), "Methods of Conjugate Gradients for Solving Linear Systems", *J. Research National Bureau of Standards*, 49, pp 409-436.

13. Sherali, H. D. and Shetty, C. M., (1980), "On the Generation of Deep Disjunctive Cutting Planes", *Naval Research Logistics Quarterly*, September 1980, 27(30), pp 453-475.

14. Sherali, H. D., Skarpness, B. O. and Kim Buyong, (1988), "An Assumption-Free Convergence Analysis for a Perturbation of the Scaling Algorithm for Linear Programs, with Application to the L_1 Estimation Problem", *Naval Research Logistics Quarterly*, 35(5), pp 473-492

15. Sherali, H. D. and Ulular, O., (1989), "A Primal-Dual Conjugate Gradient Algorithm for Specially Structured Linear and Convex Problems", *Applied Mathematics and Optimization*, 20, pp 193-221,

16. Sherali, H. D. and Ulular, O., (1990), "Conjugate Gradient Methods Using Quasi-Newton Updates with Inexact Line Searches", *Journal of Analysis and Applications*, 150(2), pp 359-337.

17. Nihan, N. L. and Davis, G. A., (1987), "Recursive Estimation of O-D Matrices from Input/ Output Counts", *Transportation Research* -B, 21-B(2), pp 149-163.

18. Nihan, N. L. and Davis, G. A., (1989), "Application of Prediction-Error Minimization and Maximum Likelihood to Estimate Intersection O-D Matrices from Traffic Counts". *Transportation Science*, 23(2), pp 77-90.

19. Vanderbei, R. J., Meketon, M. S., and Freedman, B. A., (1986),"A Modification of Karmarkar's Linear Programming Algorithm," *Algorithmica*, 1 (4), pp. 395-408.

20. Willumsen, L. G., (1984), "Estimating Time-Dependent Trip Matrices from Traffic Counts", *Ninth International Symposium on Transportation and Traffic Theory*, pp 397-411.

VITA

Namita Arora was born on September 4, 1964 in Uttar Pradesh state in India. She obtained her Bachelor of Science degree in Mathematics in 1984 from Meerut University, India. She then joined the University of Roorkee, India to pursue her Master's degree in Applied Mathematics and was awarded the Master of Science degree in 1986. She gained over two years of research and teaching experience before coming to the United States in 1990. She joined the Master's program in Industrial and Systems Engineering at the Virginia Polytechnic Institute and State University, in Fall 1991, to specialize in Operations Research. She received the General Motors fellowship for graduate research in 1992 and was employed at the University Center for Transportation Research at Virginia Tech as a graduate research assistant. She completed all requirements for the M. S. degree in Fall 1993.

Namita Arora