Title

Development of A Path Flow Estimator for Inferring Steady-State and Time-Dependent Origin-Destination Trip Matrices

Authors
Zhang, Michael
Nie, Yu
Shen, Wei
et al.

Publication Date
2008-06-01

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Development of A Path Flow Estimator for Inferring Steady-State and Time-Dependent Origin-Destination Trip Matrices

**Michael Zhang, Yu Nie, Wei Shen, Ming S. Lee,
Sarawut Jansuwan, Piya Chootinan,
Surachet Pravinvongvuth, Anthony Chen, Will W. Recker**

California PATH Research Report
UCB-ITS-PRR-2008-10

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation, and the United States Department of Transportation, Federal Highway Administration.

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California. This report does not constitute a standard, specification, or regulation.

Final Report for Task Order 5502

June 2008

ISSN 1055-1425

Development of A Path Flow Estimator for Inferring Steady-State and Time-Dependent Origin-Destination Trip Matrices

Michael Zhang, Yu Nie and Wei Shen
Department of Civil & Environmental Engineering
University of California, Davis

Ming S. Lee, Sarawut Jansuwan, Piya Chootinan,
Surachet Pravinvongvuth and Anthony Chen
Department of Civil and Environmental Engineering
Utah State University

Will W. Recker
Department of Civil and Environmental Engineering
University of California, Irvine

Final Report for TO 5502

ABSTRACT

Reliable origin/destination (O-D) data are critical to many applications in transportation planning, design and operations. Because of the high costs of and challenges in obtaining reliable O-D trip matrices from surveys or other direct sampling methods, estimating O-D trip tables from a readily available data source, traffic counts, provides an attractive, economical alternative. This project investigates one such an estimation method and implements it in a user-friendly software tool called Visual PFE TD. The developed O-D estimation tool can be used to obtain both static and dynamic O-D trip tables for traffic simulation studies, project evaluations, and transportation planning in a more streamlined and less time-consuming manner. For example, it has been used to obtain an initial seed matrix for Paramics' O-D estimator to speed up the latter’s O-D estimation process.

A logit path flow estimator (LPFE) originally proposed by Michael Bell (1995) is adopted in this research for inferring both steady and time-dependent O-D trip tables. LPFE is chosen because: 1) it incorporates the logit-based route choice model while avoiding several difficulties encountered in the conventional bi-level formulation; 2) it avoids the difficult dynamic traffic assignment problem through decomposes the dynamic O-D estimation problem into a sequence of static problems, yet takes into account of queuing by linking the static problems across time with residual queues which can be carried over from one period to subsequent periods; and finally, 3) it has been validated in a number of scenarios as a potential tool to determine O-D flows and path travel times in various transportation networks.

In this research, we extended the original LPFE formulation and improved the efficiency of solution algorithms, implemented both steady-state and time-dependent LPFE in an object-oriented programming (OOP) framework, tested the performance of LPFE using synthetic data and quantify the accuracy and reliability of its O-D trip table estimates. We also developed Visual PFE and Visual PFE-TD, the graphic user interfaces (GUI) for both static and time-dependent LPFE.

Our test case studies show that LPFE is able to produce path flows and O-D travel demands that accurately match traffic counts under the logit traffic assignment assumption. We also found that information reflecting the spatial structure of travel

demands (e.g., a historical O-D table) is of great value to the improvement of the quality of O-D trip estimates, and that LPFE can still produce satisfying estimates even when traffic counts are only available on a small portion of links, as long as such structural information is maintained in the base O-D table.

EXCUTIVE SUMMARY

This final report documents the research effort for extending and implementing the time-dependent Logit Path Flow Estimator (TD-LPFE) and its software product: Visual PFE TD. TD-LPFE estimates time-dependent O-D trip matrices from traffic counts and historical O-D matrices, and Visual PFE TD provides a user friendly input-output interface to visualize, analyze and present the estimated trip tables and related statistics from TD-LPFE. In this research, we have

1. extended the LPFE to include measurement errors and historical O-D information
2. developed efficient, object oriented C++ codes for the TD-LPFE algorithm
3. investigated the effects of algorithm parameters, measurement errors, prior O-D information and network topology on the performance of LPFE and TD LPFE
4. developed a user-friendly graphical user interface for LPFE and TD LPFE
5. prepared a user manual and workbook for the developed software tool

The developed software tool can easily estimate O-D trip tables of large networks in a reasonable amount of time (in the order of minutes), and with Visual PFE TD the data preparation and results analysis times can also be significantly reduced.

The tool is most useful for planners and traffic engineers who deal with large networks with a coarse time-resolution (greater than 15 minutes), because of the way TD LPFE models traffic flow (it assumes trips complete within one time interval. This is also the main reason making TD LPFE computationally so efficient). The outputs from this tool may not be appropriate for high resolution simulation studies that require a time resolution less than 15 minutes. One can, however, combine this tool with O-D estimation tools that often come with the high fidelity simulation software (such as the O-D estimator in Paramics) to yield O-D trip tables with a higher time-resolution. Our experience indicates that with a seed O-D provided by TD-LPFE, Paramics's O-D estimator can obtain a good O-D trip table(s) in a much shorter time than with a seed O-D obtained from a planning model.

The above limitation of TD LPFE can only be removed through the incorporation of a finer traffic flow model that tracks the growth and decay of vehicular queues in a more realistic manner. But this would require a full-blown dynamic network loading model that destroys the simple modeling structure of LPFE, which could considerably

increase the computational complexity of the estimation problem. Nevertheless, such an approach is worth investigating because it can provide higher resolution trip matrices (e.g., O-D trip tables in every minute) that can directly be used in micro-simulation studies.

Where to make traffic measurements and how many locations to make them are also important topics that need to be addressed in future O-D estimation research. Our investigation indicates that if chosen properly, a small set of measurements can produce as good O-D demand estimates as a much larger set. It would therefore be of considerable economic interest if one can develop a selection procedure that optimizes the estimation results with a minimal set of measurements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1. Introduction

Reliable origin/destination trip data are critical to many applications in transportation planning, design and operations. In freeway corridor management, for example, time-of-day ramp metering algorithms require the knowledge of origin/destination flow fractions, and adaptive ramp control strategies often need to know origin/destination flow in order to distribute expected flow reductions from a bottleneck to various metered ramps upstream. Because "true" O-D data are rarely, if ever, directly obtainable in practice, their estimation from limited observations of traffic conditions on the network has been the subject of many research efforts. Traditionally, O-D trip matrices are derived from household or roadside surveys. However, survey-based approaches have two major limitations: they are labor intensive and costly to obtain, and they often fail to capture temporal demand pattern changes. The latter is particularly detrimental to devising effective real-time traffic management strategies for relieving traffic congestion. These limitations of the survey-based approach have spurred a stream of research that focused on inferring (static or dynamic) O-D matrices from traffic counts in parts of the network. This latter approach provides a faster and cheaper alternative to household or roadside surveys, and can produce estimates of time-dependent O-D trip tables. It can also incorporate historical O-D trip data from other sources such as surveys to improve its estimates.

Conventional O-D estimation methods concern only static matrices representing O-D trips made over a relatively long time period, within which traffic condition is assumed to be at steady-state. Accordingly, steady-state O-D matrices are estimated from average link traffic counts of that period. Static O-D data cannot provide the temporal variations in traffic demand within the designated analysis period, thereby not adequate for dynamic applications such as traffic simulation, integrated control systems etc. However, unlike its steady-state counterpart, the methods for estimating dynamic O-D data are still in its infancy. In part this is due to the difficulty in obtaining accurate historical dynamic O-D data. Yet to develop a reliable dynamic traffic assignment model is a more challenging, in some sense unresolved issue. Consequently, most existing dynamic estimation methods either focus on small networks to rule out route choices or assume that time-dependent travel times can be obtained from a surveillance system or from a simulation model to help “assign” traffic onto the network. Furthermore, the introduction of the time dimension adds computational complexity to the solution of the O-D estimation problem in real size networks.

In view of these limitations, a logit path flow estimator (LPFE) originally proposed by Bell (Bell & Shield 1995) is adopted in this research for inferring both steady and time-dependent O-D trip tables. LPFE is selected for several reasons. First, LPFE incorporates the logit-based route choice model but avoids the analytical and computational difficulties of pursuing equilibrium flow patterns by bi-level programming. Second, LPFE decomposes the dynamic O-D estimation problem into a sequence of static problems, and links these static problems across time with queues which can be carried on from one period to subsequent periods. This approximation does not require any dynamic network traffic flow/assignment model, but is still capable of capturing temporal demand fluctuations and one of the most important dynamic traffic phenomena, queuing. Last but not least, LPFE has been validated in a number of scenarios as a potential tool to determine O-D flows and path travel times in various transportation networks (Bell & Shield 1995, Bell, Shield, Busch & Kruse 1997, Bell & Grosso 1998).

In this research we

- extend the original LPFE formulation and improves the efficiency of its solution algorithms: Bell's original LPFE assumes that link travel times are independent of traffic volumes and all input data is free of measurement errors. Moreover, the model does not account for historical O-D data and/or section-related measurements derived from vehicle re-identification techniques (e.g., automatic vehicle location, video detectors). The extended LPFE formulation proposed in this research overcomes these limitations. Both the iterative balancing algorithm (IBA) and the method of successive averages (MSA) are employed to solve the extended LPFE. We combine these algorithms with a column-generation technique to avoid enumerating paths. Heuristics are applied to accelerate the identification of the optimal path set and to improve the overall convergence performance.

- implement both steady-state and time-dependent LPFE as well as the corresponding logit traffic assignment models: an object-oriented programming (OOP) framework is adopted to enhance the reusability and flexibility of computer codes. Two existing C++ class libraries developed at UC Davis, TNM and MAT, are used as a starting point of implementation. This reduces programming efforts because TNM contains well-defined network objects while MAT provides supports for programming both general and network-specific optimization problems. Our implementation consists of 17 subclasses derived from MAT.
- test and validate LPFE using synthetic and real data and quantify the accuracy and reliability of O-D trip table estimates: various scenarios are set up to examine the impact of 1) the choice of measurement locations, 2) additional inputs, such as historical O-D data and subpath information obtained from section-related measurements, 3) measurement errors, and 4) network topology. Based on numerical experiments, guidelines are drawn for the applicability and optimal setting of LPFE under different circumstances. Discussions on future research are also given.
- develop a graphic user interface (GUI) for LPFE: A LPFE GUI was built using Microsoft Visual Basic and commercially available software component. The end result, Visual PFE and Visual PFE-TD, is an integrated software suite that combines the Path Flow Estimator (PFE) with other software components to facilitate the estimation, visualization, and refinement of Origin-Destination (OD) trip tables with user-friendly Graphical User Interfaces (GUI).

This report is organized into six chapters. Chapter 2 reviews the steady-state and time-dependent O-D estimation methods. Chapter 3 introduces the mathematical formulations and solution algorithms of the original LPFE and its extension. An object-oriented programming framework of LPFE as well as implementation details is provided in Chapter 4 and Chapter 5 describes the Graphic User Interface (GUI) of LPFE. Numerical experiments are reported in Chapter 6. Finally, Chapter 7 provides conclusions and recommendations for future research.

Chapter 2. Related Literature

O-D matrix estimation methods were first developed in a static context. They estimate “average” O-D trip rates using the average traffic counts over a relatively long period (such as a morning or evening peak). Two problems dominate the research efforts in the static O-D estimation: modeling traveler's route choice behavior in the presence of congestion and ensuring a unique solution in a highly under-determined system. In the dynamic context, existing estimation methods can be roughly categorized into two classes based on the type of road topology they apply to: intersection-oriented and network-oriented. Intersection-oriented methods assume the complete information for all the origin (entry) and destination (exit) counts thus are often applied to handle isolated intersections or a fraction of freeways.

2.1 Notation and preview

Consider a transportation network $G(N, A)$, where N and A are the sets of nodes and links respectively. Let R and S represent the set of origins and destinations respectively, K_{rs} the set of paths joining an OD pair rs , and q_{rs} a positive traffic demand associated with O-D pair rs . Each directed link $a \in A$ is associated with a positive travel time $t_a(x_a)$ as a convex and non-decreasing function of link flow x_a . For any path $k \in K_{rs}$, its travel time is calculated as $c_k^{rs} = \sum_a t_a \delta_{rs}^{a,k}$ (where $\delta_{rs}^{a,k} = 1$ if path k uses link a , and 0 otherwise) and its flow is denoted by f_{rs}^k . For simplifying notation, we also use $\mathbf{x} = [x_1, x_2, ..., x_a, ..., x_n]$ to represent a vector of link flows. Similarly, $\mathbf{c}$ and $\mathbf{f}$ denote vector of path travel times and flows respectively.

It is known that a stable and unique link flow pattern $\mathbf{x}^*$ can be predicted from a traffic assignment model, given appropriate assumptions on the traveller's route choice behavior. For example, if travellers always try to minimize their own travel costs and have perfect information, the resulting stable flow pattern is a *user equilibrium* (UE) pattern. Thus a relationship between O-D trip table q and flow pattern x^* can be established as follows:

$$\sum_r \sum_s p_{rs}^a q_{rs} = x_a^*, \forall a \tag{2.1}$$

where p_{rs}^a denotes the proportion of trips between O-D pair rs using link a . In matrix notation, Equation 2.1 becomes

$$P\mathbf{q} = \mathbf{x}^* \tag{2.2}$$

P is termed as *assignment matrix* because it is determined from a traffic assignment. If we assume the relationship depicted in Equation 2.2 holds in a real world, $\mathbf{q}$ can be inferred from an observation of $\mathbf{x}^*$ (e.g., traffic counts measured at loop detectors), denoted as $\bar{\mathbf{x}}$. Note that we have

$$\bar{\mathbf{x}} = \mathbf{x}^* + \varepsilon \tag{2.3}$$

where ε is a vector representing measurement errors. This is the basic idea of O-D estimation from traffic counts.

Another way to relate observed link flows to O-D demands is using path flows as an intermediate. We have the following two relationships for an equilibrium flow pattern $\mathbf{x}^*$:

$$\sum_r \sum_s \sum_k f_{rs}^k \delta_{rs}^{a,k} = x_a^* \quad \forall a \in A, k \in K_{rs} \tag{2.4}$$

$$\sum_k f_{rs}^k = q_{rs} \quad \forall k \in K_{rs} \tag{2.5}$$

or in matrix notation

$$\Delta\mathbf{f} = \mathbf{x}^* \tag{2.6}$$

$$\mathbf{Mf} = \mathbf{q} \tag{2.7}$$

O-D estimation methods based on Equations 2.6 – 2.7 are often termed as path flow estimators (PFE).

2.2 Static O-D Estimation

According to ways in which an assignment matrix is determined, static O-D estimation methods can be classified into proportional-assignment methods and equilibrium-assignment methods. If traffic congestion in a network is minor, i.e., travel time is roughly independent of changes in traffic flow, the assignment matrix is assumed to be exogenously determined and obtained by means of some proportional assignment procedures, which can for example be a simple all-or-nothing (AON) assignment or a more advanced stochastic assignment. The earlier research efforts based on the proportional-assignment assumption were credited to Willumsen (1981), for introducing the entropy maximization approach, Van Zuylen (1980) for using the concept of minimizing information, and Cascetta (1984) for casting the problem in a generalized least squares framework. Many researchers, such as Bell (1983, 1991a), McNeil and Hendrickson (1985), Brenniger-Gthe et al. (1989), Lo et al. (1996), proposed improvements and/or conducted tests along the line of proportional assignment.

However, as congestion gets heavier in the network, the proportional-assignment assumption no longer holds. In this case the assignment matrix can no longer be determined exogenously. Instead, a traffic assignment model has to be incorporated to ensure that designated equilibrium conditions are satisfied for the estimated traffic flow pattern. In other words, the estimation of the assignment matrix itself is embedded into the O-D estimation process.

The first equilibrium-assignment O-D estimation model, due to Nguyen (1977), is to find an O-D matrix that regenerates the observed travel costs (hence observed traffic counts) when assigned onto the network in a user-optimal fashion. However, the solution of Nguyen's model may not be unique since the optimization problem is not strictly convex with respect to O-D demands. In other words, there exist many possible O-D trip matrices that produce the same set of link flows. Consequently, a priori information, often in the form of a target O-D table (e.g., an outdated O-D table from a survey), has to be used to lead to a unique solution. A great deal of research work has been focused on introducing a secondary optimization problem to incorporate such a priori information. The O-D estimation problem thus has a natural bilevel structure, where the secondary optimization program forms the upper level problem (leader) while the traffic assignment problem the lower level problem (follower). The objective of the secondary optimization is either minimizing the least squares distance between the estimated and target O-D matrices (Turnquist & Gur (1979), LeBlanc & Farhangian (1982); Sheffi(1985); Yang et al. (1992), Yang (1995)), or maximizing an entropy function that leads to the most likely O-D matrix (Jornsten & Nguyen (1980); Fisk & Boyce (1983); Nguyen (1984); Fisk (1988); Yang, Sasaki, Iida & Asakura (1992)). While the above formulations used deterministic user equilibrium conditions, Yang et al. (2001) extended the bilevel programming formulation to include the stochastic user equilibrium (SUE). The major drawbacks of these bilevel programming formulations, based on either UE or SUE principles, are the inherent difficulty of obtaining global optimum, and the computational inefficiency for large-scale networks (iteratively solve the traffic assignment problem).

Path flow estimator (PFE), in which path flows instead of O-D demands are treated as solution variables, was introduced in order to resolve the analytical and computational difficulties of the bilevel formulation. Sherali et al. (1994) proposed the first path flow estimator formulated as a linear programming problem. User-equilibrium route choice behavior is considered in the method. The path decomposition of O-D flows that reproduce the observed link flows are directly determined by a column generation procedure such that the resulting O-D matrix is the closest to a target one. Using a stochastic user-equilibrium assumption, Bell (1997) extended Sherali's model to a logit path flow estimator (LPFE) that seeks to maximize the path entropy and assign trips on least cost paths simultaneously. LPFE relaxes the assumption that all link counts should be available by associating each unmeasured link with an explicit capacity constraint. We shall explore the properties of LPFE in great detail in Chapter 3. Nie and Lee (2002) suggested replacing the column generation procedure of linear PFE (Sherali, Sivanandan & Hobeika 1994) by a K-shortest path ranking (KSPR) algorithm that determines UE paths (columns) exogenously. This scheme ``decouples" linear PFE into two simple and relatively separate components. Nie, Zhang and Recker (2005) incorporated this decoupled structure into a generalized least squares (GLS) framework to develop a GLS path flow estimator. The method is attractive due to its computational advantage and convenience of analyzing estimation errors. However, to determine UE paths the method requires that all link traffic counts be available and of sufficient accuracy, which is not often realizable in real world applications.

2.3 Intersection-Oriented Dynamic O-D Estimation Models

The idea of estimating “dynamic” O-D trip tables originated from a series of pioneering work of Cremer and Keller (1981, 1984, 1987), in which time-dependent traffic counts were first employed to transform the underdetermined static model to an over-determined dynamic model. The goal of this type of dynamic models are to identify the time-varying demand split parameters (or turning proportions) that represent O-D flows at intersections or along a small section of a freeway. In the Cremer-Keller intersection model, the sequence of O-D flows and exit flows are assumed to be dependent on the time-varying patterns of entry flows through a linear relationship. Different algorithms have been proposed to solve this intersection-oriented dynamic model. Among the competing methods are the nonrecursive approach, such as the cross-correlation method (Cremer & Keller 1987), the constrained ordinary least squares method (Cremer & Keller 1987, Sherali, Arora & Hobeika 1997), the iterative maximum likelihood technique (Nihan & Davis 1989), and the fixed point method (Nihan & Hamed 1992) on one hand, and the recursive approach, such as the method of recursive estimation (Cremer & Keller 1987), the recursive least squares (RLS) (Nihan & Davis 1987), and the Kalman filter (Cremer & Keller 1987, Nihan & Davis 1987) on the other hand. Nihan and Davis (1987) showed all recursive methods relate to a family of recursive prediction error (RPE) techniques and later Nihan and Davis (1989) compared RPE with the iterative maximum likelihood method.

Bell (1991b) made the first attempt to extend the Cremer-Nihan model on more general networks. Two approaches are proposed to permit the distribution of travel times to span a number of different intervals. The first approach, which is suggested to be suitable for single intersections and small networks, assumes that travel times follow geometrical distributions and makes use of the platoon dispersion model. The second approach makes no specific assumption about the form of the travel time distribution thus can be applied for larger networks such as freeway sections. Both approaches adopt a constrained weighed least squares method and are solved sequentially.

Chang and Wu (1994) generalized Bell's model to consider vehicle travel times between each O-D pairs more realistically. The decision parameters include not only the time-dependent O-D split parameters but also the dynamic assignment parameters. The latter is closely related to the time-dependent link travel time and introduced to define the proportion of the previous intervals' O-D flows that arrive at any given exits during some future interval. The introduction of dynamic assignment parameters significantly increases the number of state variables to be estimated. Thus, a simplification is made to assume that all vehicles that reach exit s during time t are distributed within intervals $t - n_{rs}^{t}$, where n_{rs}^{t} denotes the number of time intervals that the travel time between O-D pair rs at time t will cover (If n_{rs}^{t} is not an integer, another interval is added to remove rounding errors). To estimate dynamic travel time (hence generate proper assignment factors), the authors proposed a simple two-step method based on the simple speed-density-volume relationship. This further requires the information about mainline traffic counts to be available. An extended Kalman filtering procedure is used to solve the nonlinear dynamic system but no guarantee is made to satisfy the necessary constraints on the estimated variables. The travel time computation procedure in the Chang and Wu model seems to be too simplistic to be realistic for congested freeway corridors. Also, the assumption that travel time between any O-D pair will not span more than two intervals is very restrictive.

Wu and Chang (1996) further revised their earlier model to include constraints established through screenlines. A scrrenline is a hypothetical cut that divides the network into two parts. The basic idea of the modified model comes from the assumption that most vehicles from origin i , arriving at the screenline during interval t, should embark during some intervals knowable from travel time estimation. Consequently, the fundamental measurement equation relates the screenline flows to O-D flows and time lag factors. Since one can choose as many screenlines as possible for a given urban network, sufficient constraints can be obtained for a more reliable estimation. Later Chang and Tao (1996) extended the concept of screenline to cordon line for more general networks. The measurement equation used in this model is similar to that in Chang & Wu (1994) but the dynamic assignment parameter is assumed to be known. A Kalman filter is again used to estimate the state during each time interval and as their limited results have shown, up to 20% improvements have been observed over the basic scenario by using cordon lines. The Chang and Tao model does not address how the assignment matrix might be obtained, a question that any estimation model for general networks (which the proposed model claimed to be) cannot ignore. Constructing cordon lines arbitrarily raises another serious feasibility issue, namely, how the entry and exit flows for all these cordon lines should be observed.

2.4 Network-Oriented Dynamic O-D Estimation

Network-oriented dynamic O-D estimation methods focus on extending the modeling concept of static O-D estimation by assuming the knowledge of a *dynamic assignment matrix* and (often) historical dynamic O-D trip tables. A fundamental estimation relationship is of the similar form described by Equation (2.1), which reads:

$$\sum_r \sum_s \sum_t q_{rs}^t p_{rs,h}^{a,t} = x_{ah}^* \quad \forall r \in R, s \in S, t, h \in \{1,...,T\} \tag{2.8}$$

where $p_{rs,h}^{a,t}$ denotes the proportion of the demand q_{rs}^t occupying link a during time interval h. Unlike the steady-state relation defined by Equation (2.1), Equation(2.8) reflects the contributions of O-D flows embarking during prior intervals to link counts in later intervals. The determination of dynamic assignment factors $p_{rs,h}^{a,t}$ calls for a reliable dynamic traffic assignment (DTA) model in general. However, no existing dynamic O-D estimation method incorporates an assignment model as in static cases, mainly because a DTA model that is widely accepted does not exist. Instead, the dynamic assignment matrix need to be determined externally, often by assuming dynamic link travel times can be obtained from either simulations or observations.

The first known work along this line is due to Willumsen (1984), who proposed to extend the entropy maximization method to handle time-dependent traffic counts. Simulation-based models (e.g., CONTRAM and SATURN) were suggested to establish the relationship between time-varying O-D demands and link flows.

Okutani (1987) proposed a network-oriented dynamic estimation method based on Kalman filtering, which is suitable for on-line application. The measurement equation is defined in the same way as the fundamental equation 2.8. The transition equation given in Okutani's model is autoregressive, taking the following form:

$$q_{rs}^{t+1} = \sum_{l=t-p}^{t} \sum_{r'} \sum_{s'} f_{rs}^{r's'}(t,l) q_{r's'}^{l} + \varepsilon_{rs}^{t} \tag{2.9}$$

where p is the number of lagged O-D flows assumed to affect the O-D flows in interval $t + 1$, and $f_{rs}^{r's'}(t, l)$ reflects $q_{r's'}^{l}$ on q_{rs}^{t+1} . ϵ_{rs}^{t} is the random error. Okutani applies a standard linear Kalman filter to get optimal estimates for O-D flows sequentially. Okutani's original paper does not describe how dynamic assignment matrices might be obtained. Furthermore, as Ashok & Ben-Akiva (1993) argued, the use of autoregressive transition equation 2.9 makes it hard to “capture the complex structure of activities that result in the spatial and temporal pattern of trip making”.

Cascetta et al. (1993) suggested an optimization framework for estimating dynamic O-D flows, which can be viewed as an extension of classic static O-D estimators. Equation (2.8) is employed to describe the basic measurement relation and a target O-D matrix is assumed to get a unique solution. Cascetta et al's model takes the following form:

$$\min d_1(\mathbf{q}, \bar{\mathbf{q}}) + d_2(\mathbf{x}, \mathbf{x}^*)\mathbf{q} \geq \mathbf{0} \tag{2.10}$$

where $\bar{\mathbf{q}}$ is the target O-D flows, $\mathbf{x}$ and $\mathbf{x}^*$ represent the estimated (from Equation (2.8)) and observed link volumes, respectively. Different distance functions will lead to estimators $\hat{\mathbf{q}}$ (the optimal solution of (2.10)) with different statistical properties. For example, log-likelihood function and generalized least squares (GLS) function will respectively lead to a maximum likelihood estimator and an GLS estimator. Two different estimators are presented to address different applications. A simultaneous estimator, designed for off-line application, infers in one step the entire set of time-dependent O-D flows by using link traffic counts from all time intervals. On the other hand, a sequential estimator derives the O-D flows for a given time interval by using both previous O-D estimates and the current and previous traffic counts. The sequential estimator is suitable for on-line application because its computation time is relatively modest and it is able to make use of a priori estimated O-D matrices.

Cascetta et al. (1993) further showed that the dynamic assignment matrix could be determined by a two-step stochastic route choice model. In the first step, fractions of time-dependent paths are given by a discrete choice model. Second, path flows are mapped onto links using a dynamic network loading (DNL) procedure, hence producing a stochastic dynamic assignment matrix. However, the authors did not address how to determine a path set used in the discrete path choice model. Moreover, their DNL model still assumes the knowledge of the dynamic link travel times. That is, the time-dependent link travel times through the whole network have to be completely observable. If this is impossible, the authors noted that a DTA model may provide substitutes. As mentioned before, however, this is not yet feasible for large networks.

Ashok and Ben-Akiva (1993) proposed an improved version for Okutani's approach. Instead of using the Okutani's autoregressive specification for O-D flows (Equation (2.9)), Ashok and Ben-Akiva introduced the notion of deviation from target O-D flows and reformulate the transition equation as follows:

$$q_{rs}^{t+1} - \overline{q}_{rs}^{t+1} = \sum_{l=t-p}^{t} \sum_{r'} \sum_{s'} f_{rs}^{r's'}(t,l)(q_{r's'}^{l} - \overline{q}_{r's'}^{l}) + \varepsilon_{rs}^{t} \qquad (2.11)$$

Obviously, by imposing the target O-D flows (often coming from historical estimates), the estimates can avoid the risk of losing structural information about trip patterns. Based on the Kalman filter technique, the approach provides both real-time estimation and predictions. The determination of the assignment matrix in the Ashok and Ben-Akiva's model follows the idea from Cascetta et al. (1993), namely, either an entirely observable network or a DTA model has to be available.

In a subsequent work, Ashok (1996) indicated that the assignment matrix itself is an estimate whose random errors should not be ignored. This is because an assignment matrix is obtained from random variables such as link travel times and path fractions. It was shown that the estimator will become biased and inconsistent if this issue is not correctly addressed. In order to remedy the problem, a revised procedure is presented to take the stochastic assignment matrix into account. Treating all assignment factors as decision variables is not acceptable because it considerably increases the computational load. These authors thus provided a simplification in which only $t+p$ state equations are used. To determine the state augmentation required to transform this stochastic formulation to the standard transition and measurement equations, the method has to estimate each state variable as many times as the size of the lag when applying a Kalman filter procedure. Most recently, Ashok and Ben-Akiva (2000) suggested an alternative approach defining the state-vector by deviation of departure rates from each origin and the shares headed to each destination. Except its transition equation, this approach has a similar framework as those proposed previously by Ashok and Ben-Akiva (1993).

Ashok and Ben-Akiva (1993, 1996, 2000) reported encouraging results in several case studies and filed tests, and indicated that their method turned out to be robust. When attempting to implement Ashok-Ben-Akiva approaches for comparison purpose, however, Sherali et al. (2001) discovered that the matrix needed to be inverted in updating the Kalman gain matrix is always singular. As Sherali et al (2001) pointed out, the failure might be a result of violating some Kalman filter assumptions in the application. This observation called for special attention in checking data requirements and verifying standing assumptions when implementing Kalman filters in practice.

Sherali and Park (2001) proposed a dynamic path flow estimator, which describes the relationship between dynamic O-D flow and link flows as follows:

$$\sum_r \sum_s \sum_k \sum_t f_{rs}^{kt} \delta_{rs,h}^{a,kt} = x_{ah}^* \quad \forall a \in A, k \in K_{rs}, t, h \in \{1, \ldots, T\} \tag{2.12}$$

$$\sum_k f_{rs}^{kt} = q_{rs}^t \quad \forall k \in K_{rs}, t \in \{1, \ldots, T\} \tag{2.13}$$

where f_{rs}^{kt} is associated with the path flow departs along path k during time interval t, while $\delta_{rs,h}^{a,kt} = 1$ if link a is occupied during time interval h due to path flow f_{rs}^{kt}, and 0 otherwise.

Sherali et al.'s dynamic PFE is formulated as a constrained least squares (CLS) problem which seeks to determine a set of time-dependent shortest path flows that reproduce the observed link counts as closely as possible. Specifically, Sherali et al's CLS estimation model reads

$$\min \frac{1}{2} \sum_{t=1}^{T} \sum_{a \in A} [\sum_k \sum_t f_{rs}^{kt} \delta_{rs,h}^{a,kt} - \bar{x}_{ah}^*]^2 + \mu \sum_k \sum_t^{T} c_{rs}^{kt} f_{rs}^{kt} \tag{2.14}$$

subject to

$$f_{rs}^{kt} \geq 0 \quad \forall k \in K, t \tag{2.15}$$

where K denotes all paths existing between all O-D pairs. The second term in the objective function, a weighted total system cost, helps guide the solution toward more likely efficient paths. A path generation algorithm is devised to solve the optimization problem iteratively. The algorithm begins with solving a restricted master problem based on an initial choice of a set of O-D paths, then attempts to augment the master problem with paths from time-dependent shortest path search upon a time-space expanded network. The method will terminate and claim an optimum if no new time-dependent shortest path can be found. Dynamic link travel times are assumed to be known inputs thus not modelled in Sherali et al.'s work (2001). This is true for all existing network-oriented dynamic O-D estimation methods. Other limitations of this method include the requirement for a complete observation of all link flows (which is typically not available in reality), and its inability to take into account observation errors.

2.5 Summary

Although many approaches have been proposed to solve the O-D trip table estimation problem, the validity and applicability of most approaches to large-scale networks remain to be investigated. Particularly, existing dynamic O-D estimation methods are far from producing satisfactory results on real-sized networks. Above all, the determination of dynamic assignment matrices is separated from the O-D estimation process, thus the estimated flow pattern is not necessarily consistent with the network state that yielded the observed travel times.

Compared with existing dynamic O-D estimation methods, Bell's time-dependent LPFE (TDLPFE) seems much more tractable because it transforms the dynamic problem into static problems. TDLPFE is not a true “dynamic” model in the sense that all trips are assumed to complete within each time interval. However, the transition of queues across time periods makes it capable of capturing the queuing phenomenon, one of the most important characteristics of dynamic traffic. The research approach described in the following chapters thus use LPFE to develop a software tool for deriving both steady-state and time-dependent O-D trip tables to support various applications.

Chapter 3. LPFE: Formulation and Algorithm

The O-D estimation problem is closely interrelated with the traffic assignment problem, since outputs of one problem serve as inputs to the other. Thus, we shall first review the corresponding traffic assignment model before turning to logit path flow estimator.

3.1 Logit Traffic Assignment Model

Static traffic assignment models based on Wardrop's principles (1952) can be classified into deterministic and stochastic ones. The latter recognizes that travellers are unlikely to have perfect information about network conditions and stipulates that travelers choose the minimum cost path with certain probability. Stochastic traffic assignment models can further be classified as multinomial logit and multinomial probit assignment models. Probit models are theoretically sound but incur great computational obstacles that hinder its application, particularly for large networks: it requires either Monte Carlo techniques or path enumeration (e.g. Daganzo & Sheffi 1977, Powell & Sheffi 1982). On the other hand, the logit model is known to have some undesirable properties (such as the IIA property) but is much more tractable both analytically and computationally, therefore is the favored probability model used in stochastic traffic assignment.

3.1.1 The static case

The most widely used logit stochastic user equilibrium (SUE) model is due to Fisk (1980), who added a scaled path entropy term into the objective function of the Beckmann formulation of DUE (Deterministic User Equilibrium, Beckmann, McGuire & Winsten 1956). Fisk's model takes the following mathematical form:_

[FSUE]

$$\min \sum_a \int_0^{x_a} t_a(w)dw + \frac{1}{\theta} \sum_r \sum_s \sum_k (f_k^{rs}(\ln f_k^{rs} - 1)) \qquad (3.1)$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \quad \forall r \in R, s \in S, k \in K^{rs} \qquad (3.2)$$

$$\sum_r \sum_s \sum_k f_k^{rs} \delta_{a,k}^{rs} = x_a \quad \forall a \in A, k \in K^{rs}, r \in R, s \in S \qquad (3.3)$$

$$f_k^{rs} \geq 0 \quad \forall k \in K^{rs}, r \in R, s \in S \qquad (3.4)$$

where θ is called a dispersion parameter which represents traveller's sensitivity to path costs. It is shown that the optimality condition of this mathematical program is equivalent to the logit path choice model, i.e.,

$$P_k^{rs} = \frac{\exp(-\theta \pi_k^{rs})}{\sum_k \exp(-\theta \pi_k^{rs})}, \forall r \in R, s \in S, k \in K^{rs} \qquad (3.5)$$

where P_k^{rs} is the probability of travellers between O-D pair rs use path k, and π_k^{rs} is travel cost on path k. Note that in FSUE π_k^{rs} equals to the path travel time c_k^{rs}. Unlike its deterministic counterpart, the optimal SUE path solution is unique because the Hessian matrix is strict positive definite with respect to path flow.

Fisk's original logit model assumes that the link cost function $t_a(\cdot)$ is an increasing function of link flow. Because travel times given by such functions remain finite whenever link flows are finite, the model may predict unrealistically high link flows (sometimes well exceed the physical capacities of roads). This problem can be resolved by introducing explicit capacity constraints into the formulation, and the Lagrangian multipliers associated with these new constraints can be interpreted as queuing delay caused by insufficient road capacities. Bell (1997) took this approach but assumed that the link travel cost is constant and delay only exists on links operating at capacity. Bell's logit assignment model reads

$$\min \sum_{a} t_a x_a + \frac{1}{\theta} \sum_{r} \sum_{s} \sum_{k} (f_k^{rs}(\ln f_k^{rs} - 1)) \tag{3.6}$$

subject to Equations 3.2 to 3.4, and

$$x_a \leq C_a, \forall a \in A \tag{3.7}$$

Note that C_a denotes the capacity of link a and t_a represents a constant travel time (cost) on link a, which is independent of link flows. By using the link-path relationship 3.3, the model can be expressed in path flows only, as shown below

$$\min \sum_{r} \sum_{s} \sum_{k} f_k^{rs} c_k^{rs} + \frac{1}{\theta} \sum_{r} \sum_{s} \sum_{k} (f_k^{rs}(\ln f_k^{rs} - 1)) \tag{3.8}$$

subject to Equation 3.4, 3.2 , and

$$\sum_{r} \sum_{s} \sum_{k} f_k^{rs} \delta_{a,k}^{rs} \leq C_a, \forall a \in A \tag{3.9}$$

where c_k^{rs} is the travel time on path k. We can further simplify the model using matrix notation as follows:

[BSUE]

$$\min\langle \mathbf{f}, \mathbf{c} \rangle + \frac{1}{\theta} \langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle \tag{3.10}$$

subject to

$$\Delta \mathbf{f} \leq \mathbf{C} \tag{3.11}$$

$$\mathbf{Mf} = \mathbf{q} \tag{3.12}$$

$$\mathbf{f} \geq \mathbf{0} \tag{3.13}$$

where $\langle a, b \rangle = a^T b$, Δ and M are $m \times n$ and $l \times n$ matrices, respectively, with m, n and l are numbers of links, paths and O-D pairs respectively. Ignoring the nonnegative constraints, the optimality conditions of Bell's logit model can be written as

$$\frac{1}{\theta}\ln\mathbf{f} + \mathbf{c} + \langle\Delta, \mu\rangle - \langle\mathbf{M}, \omega\rangle = 0 \tag{3.14}$$

$$\mathbf{C} - \Delta\mathbf{f} \geq 0 \tag{3.15}$$

$$\langle\mu, \mathbf{C} - \Delta\mathbf{f}\rangle = 0 \tag{3.16}$$

$$\mu \geq 0 \tag{3.17}$$

$$\mathbf{q} - \mathbf{M}\mathbf{f} = 0 \tag{3.18}$$

Note that μ and ω are vectors of Lagrangian multipliers associated with constraints 3.11 and 3.12, respectively. Conditions 3.14 and 3.17 provides the logit path choice model as of 3.5. However, the path cost π_k^{rs} in 3.5 now has the following form

$$\pi_k^{rs} = \sum_a (t_a + \mu_a)\delta_{a,k}^{rs} \tag{3.19}$$

where μ_a is the Lagrangian multiplier, interpreted as the equilibrium queuing delay on link a.

3.1.2 The time-dependent case

Although the introduction of capacity constraints rules out link flows beyond capacities, it also reduces the feasible solution set that under some situations a feasible set may be empty, particularly in congested networks. An alternative is to model queues directly. The idea is to restrict link flow from exceeding the (weighted) sum of link capacity and queue, rather than just capacity itself. By not imposing any limit on queue length, one ensures that the feasible solution set is always non-empty, i.e., there is always at least one feasible solution. The formulation of this revised model reads,

[RBSUE]

$$\min\langle\mathbf{f}, \mathbf{c}\rangle + \frac{1}{\theta}\langle\mathbf{f}, \ln\mathbf{f} - \mathbf{I}\rangle + 0.5\langle\mathbf{v}, \mathbf{D}^{-1}\mathbf{v}\rangle \tag{3.20}$$

subject to 3.12 (nonnegative constraints are ignored) and

$$\Delta\mathbf{f} \leq \mathbf{C} + \mathbf{v} \tag{3.21}$$

where $\mathbf{v}$ is a vector of link queues, $\mathbf{D}$ is a diagonal matrix as follows:

$$\mathbf{D} = \begin{bmatrix} C_1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & C_m \end{bmatrix}$$

Note that the quadratic term is added into the objective function value in order to discourage the spreading of queues. RBSUE assumes that some traffic stays in queues at the end of the analysis period, which conflicts with the assumption that all trip be completed in steady-state assignment models. Nevertheless, as we shall see soon, the existence of such ``residual" queues makes it possible to model the carryover of congestion across time using steady-state assignment models. The optimality conditions of RBSUE include Equations 3.14, 3.17, 3.18, and

$$\mathbf{C} - \Delta\mathbf{f} + \mathbf{v} \geq 0 \tag{3.22}$$

$$\langle \mu, \mathbf{C} - \Delta\mathbf{f} + \mathbf{v} \rangle = 0 \tag{3.23}$$

$$\mathbf{D}^{-1}\mathbf{v} = \mu \tag{3.24}$$

Equation 3.24 simply states that the equilibrium queue is equal to the equilibrium delay times the capacity.

In the time-dependent case, a sequence of steady-state RBSUEs is used to approximate the temporal traffic evolution. Specifically, residual queues generated in previous periods must be processed in subsequent periods. The temporal overloading of links is captured because the effective link capacity may be reduced by residual queues from predecessor periods. Let $\mathbf{v}_{t-1}$ represent the vector of equilibrium queues generated at time $t-1$, we have the following time-dependent model

[TD-BSUE]

$$\min \langle \mathbf{f_t}, \mathbf{c_t} \rangle + \frac{1}{\theta} \langle \mathbf{f_t}, \ln \mathbf{f_t} - \mathbf{I} \rangle + 0.5 \langle \mathbf{v_t}, \mathbf{D}^{-1} \mathbf{v_t} \rangle \tag{3.25}$$

subject to

$$\Delta\mathbf{f_t} \leq \mathbf{C} + \mathbf{v_t} - \mathbf{v_{t-1}} \tag{3.26}$$

$$\mathbf{Mf_t} = \mathbf{q_t} \tag{3.27}$$

Each time period of a dynamic assignment problem is thus represented by a steady-state assignment problem as described by TD-BSUE, where the subscript t denotes the index of time period. The structure of TD-BSUE is similar to RBSUE except that residual queues from the last time period appears in the right hand side of Constraints 3.21. As mentioned, the role of $\mathbf{q}_{t-1}$ is to reduce link capacity in the current period. This approximation roughly captures the temporal flow propagation since the residual queues are processed before the new arrivals, and new queues are only formed when the reduced capacities are exceeded.

3.2 Logit Path Flow Estimator

Now we turn to the formulation of the logit path flow estimator (LPFE). LPFE derives path flows (hence an estimation of O-D demands) using a mathematical program similar to logit assignment models in structure. The major difference is that in logit assignment models path flows are so chosen that conservative conditions hold for each O-D pair, whereas in LPFE the selected path flows have to reproduce observed link flows.

3.2.1 The static case

Note that in reality only a subset of link flows can be measured. The observed and unobserved link set are denoted as A_o and A_u respectively, with $A = A_u + A_m$. Traffic counts are only available on links in A_m. For any link $a \in A_u$, the free flow link travel time and the link capacity are assumed to be known. Corresponding to the partition of link set, the vector of link capacities $\mathbf{C}$ can be divided into $\mathbf{C_o}$ and $\mathbf{C_u}$. Further, path-link incident matrix Δ can be divided into Δ_o and Δ_u.

The logit path flow estimator is formulated as a mathematical program as follows:

[LPFE]

$$\min\langle \mathbf{f}, \mathbf{c}\rangle + \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I}\rangle \tag{3.28}$$

subject to

$$\Delta_o \mathbf{f} = \bar{\mathbf{x}}_{\mathbf{o}} \tag{3.29}$$

$$\Delta_u \mathbf{f} \leq \mathbf{C_u} \tag{3.30}$$

where $\bar{\mathbf{x}}_{\mathbf{o}}$ is a vector of observed link flows. The KKT conditions of LPFE are

$$\frac{1}{\theta} \ln \mathbf{f} + \mathbf{c} + \langle \Delta_u, \mu \rangle - \langle \Delta_o, \lambda \rangle = 0 \tag{3.31}$$

$$\mathbf{C_u} - \Delta_u \mathbf{f} \geq 0 \tag{3.32}$$

$$\langle \mu, \mathbf{C_u} - \Delta_u \mathbf{f} \rangle = 0 \tag{3.33}$$

$$\mu \geq 0 \tag{3.34}$$

$$\Delta_o \mathbf{f} - \bar{\mathbf{x}}_{\mathbf{o}} = 0 \tag{3.35}$$

Again, the multipliers associated with capacity constraints, μ, can be interpreted as the equilibrium queuing delay. Moreover, the optimality condition 3.31 yields to the logit path choice Model 3.5. However, π_k^{rs} is defined differently using multipliers

$$\pi_k^{rs} = \sum_a t_a \delta_{a,k}^{rs} + \sum_{a_u} \mu_{a_u} \delta_{a_u,k}^{rs} - \sum_{a_o} \lambda_{a_o} \delta_{a_o,k}^{rs} \tag{3.36}$$

3.2.2 The time-dependent case

The time-dependent LPFE can be formulated using a similar approach described in Section 3.1.2, namely, residual queues are explicitly added into the objective function and capacity constraints such that congestion effects can be carried over from one time interval to others. At time t, TD-LPFE is formulated as

[TD-LPFE]

$$\min \langle \mathbf{f_t}, \mathbf{c_t} \rangle + \frac{1}{\theta} \langle \mathbf{f_t}, \ln \mathbf{f_t} - \mathbf{I} \rangle + 0.5 \langle \mathbf{v_u^t}, \mathbf{D_u^{-1}} \mathbf{v_u^t} \rangle \tag{3.37}$$

subject to

$$\Delta_u \mathbf{f_t} \leq \mathbf{C_u} + \mathbf{v_u^t} - \mathbf{v_u^{t-1}} \tag{3.38}$$

$$\Delta_o \mathbf{f_t} = \bar{\mathbf{x}}_\mathbf{o}^\mathbf{t} \qquad (3.39)$$[1]

where $\bar{\mathbf{x}}_\mathbf{o}^\mathbf{t}$ is the vector of observed link flows at time period t, $\mathbf{v}_\mathbf{u}^\mathbf{t}$ is the vector of queues on unobserved links at t. For each time period, TD-LPFE seeks a path flow pattern that satisfies the logit path choice model and replicates observed link flows simultaneously, while taking into account of the queues accumulated in previous time intervals. The entropy term encourages trips to spread across paths while the quadratic term discourages queuing.

Let μ_t and λ_t be the multipliers associated with constraints 3.38 and 3.39, the optimality conditions of TD-LPFE are expressed as

$$\frac{1}{\theta}\ln \mathbf{f_t} + \mathbf{c_t} + \langle \Delta_u, \mu_t \rangle - \langle \Delta_o, \lambda_t \rangle = 0 \qquad (3.40)$$

$$\mathbf{C_u} - \Delta_u \mathbf{f_t} + \mathbf{v}_\mathbf{u}^\mathbf{t} - \mathbf{v}_\mathbf{u}^\mathbf{t-1} \geq 0 \qquad (3.41)$$

$$\langle \mu_t, \mathbf{C_u} - \Delta_u \mathbf{f_t} + \mathbf{v}_\mathbf{u}^\mathbf{t} - \mathbf{v}_\mathbf{u}^\mathbf{t-1} \rangle = 0 \qquad (3.42)$$

$$\mu_t \geq 0 \qquad (3.43)$$

$$\Delta_o \mathbf{f_t} - \bar{\mathbf{x}}_\mathbf{o}^\mathbf{t} = 0 \qquad (3.44)$$

$$\mathbf{D}_\mathbf{u}^{-\mathbf{1}} \mathbf{v}_\mathbf{u}^\mathbf{t} = \mu_t \qquad (3.45)$$

3.2.3 Extended LPFE

Several extensions to the original LPFE are presented in this section. We shall use the static case to illustrate the major ideas.

First of all, the link travel time is assumed in the original LPFE to be the sum of a constant link traversal time and a queuing delay (which is positive so long as links operate at their capacities). This assumption accords with many real-world observations, particularly on arterial streets where intersection control dealy dominates the delay experienced by drivers. On freeways, however, queuing delay may not fully account for all congestion effects. It is thus useful to treat link traversal time as flow dependent

[1] This conservation equation assumes that all trips depart from an assignment interval also complete within that time interval.

through an appropriate link performance function. When considering flow-dependent link traversal times, the term related path costs in the objective function of LPFE is replaced with an integral expression as in Fisk's model:

$$\min \sum_a \int_0^{x_a} t_a(w)dw + \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle \tag{3.46}$$

subject to Equations 3.29, 3.30 and path-flow incidence relationship 3.3.

Using measurement constraints such as (3.29) implies extreme confidence on the quality of link observations. However, measurement errors are unavoidable in real applications. Ignoring these errors can of course lead to inaccurate estimation. More severely, inconsistent link flow patterns (e.g., flow conservation law might be violated due to such errors) can be introduced, thereby leading to possibly no feasible solution. The problem can be addressed in different ways. A commonly used approach is to consider link measurements in the objective function as follows:

$$\min \sum_a \int_0^{x_a} t_a(w)dw + \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle + \langle \Delta_o \mathbf{f} - \bar{\mathbf{x}}_\mathbf{o}, T^{-1}(\Delta_o \mathbf{f} - \bar{\mathbf{x}}_\mathbf{o}) \rangle \tag{3.47}$$

subject to Equations 3.30.

where T is the covariance matrix of link measurements. Although this approach incorporates measurement errors naturally, it may introduce numerical instabilities in the logit path choice model. Note that the exponential of link measurement mismatch ($\Delta_o \mathbf{f} - \bar{\mathbf{x}}_\mathbf{o}$) can lead to very large path flows. Such a problem arises often in the earlier stage in a numerical solution procedure. An alternative is to change constraints so that the link counts fall into a range of values rather than equal to an exact value, i.e.,

$$\min \sum_a \int_0^{x_a} t_a(w)dw + \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle \tag{3.48}$$

subject to Equation 3.30, and

$$\Delta_o \mathbf{f} \leq \langle \bar{\mathbf{x}}_\mathbf{o}, \mathbf{I}_o + \gamma_o \rangle \tag{3.49}$$

$$\Delta_o \mathbf{f} \geq \langle \bar{\mathbf{x}}_\mathbf{o}, \mathbf{I}_o - \delta_o \rangle \tag{3.50}$$

where $\mathbf{I}_o$, γ_o , and δ_o are all $|A_o| \times |A_o|$ diagonal matrices, and

$$\mathbf{I}_o = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix}, \gamma_o = \begin{bmatrix} \gamma_o^1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & \gamma_o^{|A_o|} \end{bmatrix}, \delta_o = \begin{bmatrix} \delta_o^1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & \delta_o^{|A_o|} \end{bmatrix}, \gamma_o \geq 0, \delta_o \geq 0$$

Note that $\gamma_o^{a_o}$ and $\delta_o^{a_o}$ reflect the confidence level of measurements on link $a_o \in A_o$. Obviously, a more reliable link measurement will constrain the observed flow within a smaller tolerance, while a less reliable measurement will allow a large error range.

The original LPFE does not exploit existing O-D information in its formulation. However, quite often a historical O-D table maintains useful structural information about O-D flow patterns that can greatly improve estimation quality. Moreover, the direct observation of O-D flows becomes increasingly feasible thanks to recent technological innovations (e.g., section-related measurement based on vehicle re-identification). Although such observations can only cover a small portion of O-D entries in most cases, the up-to-date information provided by them can still be of great help. To account for available O-D information, we assume that a partial O-D table $\mathbf{\bar{q}_o}$ exists, which is obtained from either a historical O-D table, or real-time section-related measurements. Recall that in assignment problems, O-D demands $\mathbf{q}$ are related to path flows $\mathbf{f}$ through conservation law

$$\mathbf{q} = \mathbf{Mf}$$

For a partially available O-D table, we have the following relationship

$$\mathbf{\bar{q}_o} = \mathbf{M_o f} \tag{3.51}$$

where $\mathbf{M_o}$ contains rows of M that correspond to observed O-D entries. Adding 3.51 into LPFE as constraints will enforce the estimated flow pattern to replicate these additional observations. However, given the O-D measurements may contain errors, it is better to introduce confidence levels as in link measurements. Thus, the following constraints will be considered:

$$\mathbf{M_o f} \leq \langle \mathbf{\bar{q}_o}, \mathbf{J}_o + \eta_o \rangle \tag{3.52}$$

$$\mathbf{M_o f} \geq \langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o - \varphi_o \rangle \tag{3.53}$$

Similarly, $\mathbf{J}_o$, η_o , and φ_o are $l_o \times l_o$ diagonal matrices (l_o is the number of observed O-D entries)

$$\mathbf{J}_o = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix}, \gamma_o = \begin{bmatrix} \eta_o^1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & \eta_o^{l_o} \end{bmatrix}, \varphi_o = \begin{bmatrix} \varphi_o^1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & \varphi_o^{l_o} \end{bmatrix}, \eta_o \geq 0, \varphi_o \geq 0$$

Finally, note that in the original LPFE, travel times on measured links contain no queuing delay because the capacity constraints of these links are not considered in estimation problems. This causes miscalculation of path costs and in turn the problem of path flow estimation. To address this issue, not only traffic counts, but also queuing delay needs to be measured. The observed link travel times (free flow travel time plus observed delay) should be considered as constant over the estimation process. Although measuring queuing delay directly is far more difficult than measuring traffic counts, it is possible, as Bell noted, to calculate the queuing delay by use of an appropriate delay formula[2]

Now we are ready to present the extended LPFE in its complete form:

<u>[E-LPFE]</u>

$$\min \sum_{a \in A_u} \int_0^{x_a} t_a(w)dw + \sum_{a \in A_o} \bar{t}_a x_a + \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle \tag{3.54}$$

subject to Equations 3.30, 3.49, 3.50, 3.52, and 3.53.

where $\bar{t}_a$ denotes the observed travel time on link a, which consist of a travel time component without delay and a delay component.

The Kuhn-Tucker conditions of this extended formulation are

$$\frac{1}{\theta} \ln \mathbf{f} + \bar{\mathbf{c}} + \langle \Delta_u, \mu \rangle + \langle \Delta_o, \lambda^+ \rangle - \langle \Delta_o, \lambda^- \rangle + \langle \Delta_o, v^+ \rangle - \langle \Delta_o, v^- \rangle = 0 \tag{3.55}$$

$$\mathbf{C_u} - \Delta_u \mathbf{f} \geq 0 \tag{3.56}$$

$$\langle \mu, \mathbf{C_u} - \Delta_u \mathbf{f} \rangle = 0 \tag{3.57}$$

[2] The Webster's formula was used for that purpose in Bell et al. (1997)

$$\langle \bar{\mathbf{x}}_{\mathbf{o}}, \mathbf{I}_o + \gamma_o \rangle - \Delta_o \mathbf{f} \geq 0 \tag{3.58}$$

$$\langle \lambda^+, \langle \bar{\mathbf{x}}_{\mathbf{o}}, \mathbf{I}_o + \gamma_o \rangle - \Delta_o \mathbf{f} \rangle = 0 \tag{3.59}$$

$$\langle \bar{\mathbf{x}}_{\mathbf{o}}, \mathbf{I}_o - \delta_o \rangle - \Delta_o \mathbf{f} \leq 0 \tag{3.60}$$

$$\langle \lambda^-, \langle \bar{\mathbf{x}}_{\mathbf{o}}, \mathbf{I}_o - \delta_o \rangle - \Delta_o \mathbf{f} \rangle = 0 \tag{3.61}$$

$$\langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o + \eta_o \rangle - \mathbf{M}_{\mathbf{o}} \mathbf{f} \geq 0 \tag{3.62}$$

$$\langle \nu^+, \langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o + \eta_o \rangle - \mathbf{M}_{\mathbf{o}} \mathbf{f} \rangle = 0 \tag{3.63}$$

$$\langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o - \varphi_o \rangle - \mathbf{M}_{\mathbf{o}} \mathbf{f} \leq 0 \tag{3.64}$$

$$\langle \nu^-, \langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o - \varphi_o \rangle - \mathbf{M}_{\mathbf{o}} \mathbf{f} \rangle = 0 \tag{3.65}$$

$$\mu \geq 0, \lambda^+ \geq 0, \lambda^- \geq 0, \nu^+ \geq 0, \nu^- \geq 0 \tag{3.66}$$

$\mu, \lambda^+, \lambda^-, \nu^+, \nu^-$ are multipliers associated with corresponding constraints, and $\bar{\mathbf{c}}$ are path travel times containing measured link travel times, i.e.,

$$\bar{c}_k^{rs} = \sum_{a \in A_o} \bar{t}_a \delta_{a,k}^{rs} + \sum_{a \in A_u} t_a(x_a) \delta_{a,k}^{rs} \tag{3.67}$$

Similarly, path costs used in the logit choice model take the following form

$$\bar{\pi}_k^{rs} = \bar{c}_k^{rs} + \sum_{a \in A_u} \mu_a \delta_{a,k}^{rs} + \sum_{a \in A_o} \lambda^+ \delta_{a,k}^{rs} - \sum_{a \in A_o} \lambda^- \delta_{a,k}^{rs} \tag{3.68}$$

The model extensions developed so far can be easily extended to the time-dependent case. For the sake of complexness, the formulation of the extended TD-LPFE is given below

[E-TD-LPFE]

$$\min \sum_{a \in A_u} \int_0^{t_a} t_a^t(w) dw + \sum_{a \in A_o} \bar{t}_a^t x_a^t + \frac{1}{\theta} \langle \mathbf{f_t}, \ln \mathbf{f_t} - \mathbf{I} \rangle + 0.5 \langle \mathbf{v_u^t}, \mathbf{D_u^{-1}} \mathbf{v_u^t} \rangle \tag{3.69}$$

subject to Equation 3.38 and

$$\Delta_o \mathbf{f^t} \le \langle \mathbf{\bar{x}_o^t}, \mathbf{I}_o + \gamma_o^t \rangle \tag{3.70}$$

$$\Delta_o \mathbf{f^t} \ge \langle \mathbf{\bar{x}_o^t}, \mathbf{I}_o - \delta_o^t \rangle \tag{3.71}$$

$$\mathbf{M_o f^t} \le \langle \mathbf{\bar{q}_o^t}, \mathbf{J}_o + \eta_o^t \rangle \tag{3.72}$$

$$\mathbf{M_o f^t} \ge \langle \mathbf{\bar{q}_o^t}, \mathbf{J}_o - \varphi_o^t \rangle \tag{3.73}$$

Again, the model E-TD-LPFE will be solved once for each time interval, and the residual queue $\mathbf{v_u^t}$ is updated based on the queue from the last period $\mathbf{v_u^{t-1}}$. For the first time interval ($t = 1$),it is assumed that $\mathbf{v_u^0} = 0$.

We close this section by mentioning that the total demand of an O-D entry is not fully observable even with recent surveillance techniques. Quite often, only a subset of vehicles has the proper equipment for the identification purpose. Consequently, what is actually available for estimation models is not the O-D flow itself, but the so-called O-D split fraction, i.e., proportions of traffic departing from an origin heading towards a certain destination. Thus, constraints 3.53 and 3.52 should be replaced by what describes such proportional relationship among O-D entries. Obviously, such relationship can be expressed in a linear equation, e.g.,

$$\mathbf{Sf} = 0$$

However, these constraints may pose numerical difficulties in applying algorithms like the iterative balancing algorithm because its right hand side is zero (remember $\ln 0$ has no definition). We leave this to further research.

3.3 Solution Algorithms

We only present algorithms for solving E-LPFE and E-TD-LPFE in this section. But the presented algorithm can solve logit assignment problems with minor modifications since the two problems have similar structure (above all, they are all based on the logit path choice model). We will first deal with a sub-problem of E-LPFE (E-LPFE-SUB), which assumes that all paths have been determined and all link travel times are fixed (flow independent). Then we will relax these assumptions by introducing a column generation procedure and a decent direction. Finally we tackle the time-dependent case.

3.3.1 Solve E-LPFE-SUB

When all used paths and link travel times are known, we obtain a sub-problem of E-LPFE, which reads

[E-LPFE-SUB]

$$\min\langle \mathbf{f}, \bar{\mathbf{c}} \rangle + \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle \tag{3.74}$$

subject to 3.30, 3.49, 3.50, 3.52, and 3.53.

Note that we use $\bar{\mathbf{c}}$ instead of $\mathbf{c}$ because some link travel times are measured values. There exist a number of descent direction algorithms for solving a nonlinear program like E-LPFE-SUB (e.g., the Frank-Wolfe algorithm). However, the complex constraint structure involved in E-LPFE-SUB makes it unappealing to use those classical procedures. The iterative balancing algorithm (IBA) offers a promising alternative by directly exploiting the Kuhn-Tucker conditions (Equations 3.55 to 3.66). The major steps of the algorithm are summarized below:

ALGORITHM IBA

Step 0 Initialization. Set $\mu = 0, \lambda^+ = 0, \lambda^- = 0, \nu^+ = 0, \nu^- = 0, \kappa = -\infty$.

Step 1 Iteratively check constraints and update path flows.

Step 1.1 Calculate path flow $\mathbf{f}$ according to Equation 3.55. Set $\tilde{\mathbf{x}} = \Delta\mathbf{f}, \tilde{\mathbf{q}} = \mathbf{M}\mathbf{f}$

Step 1.2 For each link $a \in A_u$, compute $e = \ln\tilde{x}_a - \ln C_a$

If $e > 0$, set $\mu_a = \mu_a + e$. Update $\mathbf{f}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$, if $\kappa < |e|$, set $\kappa = |e|$.

Step 1.3 For each link $a \in A_o$, compute $e = \ln\tilde{x}_a - \ln\bar{x}_a(1 + \gamma_a)$

If $e > 0$, set $\lambda_a^+ = \lambda_a^+ + e$. Update $\mathbf{f}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$, if $\kappa < |e|$, set $\kappa = |e|$.

Step 1.4 For each link $a \in A_o$, compute $e = \ln\tilde{x}_a - \ln\bar{x}_a(1 - \delta_a)$

If $e < 0$, set $\lambda_a^- = \lambda_a^- - e$. Update $\mathbf{f}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$, if $\kappa < |e|$, set $\kappa = |e|$.

Step 1.5 For each O-D pair rs for which the target O-D information exists

Compute $e = \ln\tilde{q}^{rs} - \ln\bar{q}^{rs}(1 + \eta_{rs})$

If $e > 0$, set $\nu_{rs}^+ = \nu_{rs}^+ + e$. Update $\mathbf{f}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$. If $\kappa < |e|$, set $\kappa = |e|$.

Step 1.6 For each O-D pair rs for which the target O-D information exists

Compute $e = \ln \tilde{q}^{rs} - \ln \bar{q}^{rs}(1 - \varphi_{rs})$

If $e < 0$, set $\nu_{rs}^{-} = \nu_{rs}^{-} - e$. Update $\mathbf{f}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$. If $\kappa < |e|$, set $\kappa = |e|$.

Step 2 Convergence test. If $\kappa < \epsilon$, stop; otherwise, return to Step1.

Apparently, IBA sequentially checks and fixes the violations of the Kuhn-Tucker conditions (3.56 to 3.66) while maintaining the relationship dictated by the logit path choice model (Equation 3.55).

Proposition 3.1 *If E-LPFE-SUB has a non-empty feasible set, algorithm IBA converges to its optimal solution.*

Proof: First, note that the Lagrangian associated with E-LPFE-SUB is

$$L(\mathbf{f}, \mu, \lambda^{+}, \lambda^{-}, \nu^{+}, \nu^{-}) = \langle \mathbf{f}, \bar{\mathbf{c}} \rangle + \frac{1}{\theta} \langle \ln \mathbf{f}, \ln \mathbf{f} - \mathbf{I} \rangle + \langle \Delta_u \mathbf{f} - \mathbf{C}, \mu \rangle + \langle \Delta_o \mathbf{f} - \langle I_o + \gamma_o, \bar{\mathbf{x}}_\mathbf{o} \rangle, \lambda^{+} \rangle$$

$$- \langle \Delta_o \mathbf{f} - \langle I_o - \delta_o, \bar{\mathbf{x}}_\mathbf{o} \rangle, \lambda^{-} \rangle + \langle \Delta_o \mathbf{f} - \langle J_o + \eta_o, \bar{\mathbf{q}}_\mathbf{o} \rangle, \nu^{+} \rangle - \langle \Delta_o \mathbf{f} - \langle J_o - \varphi_o, \bar{\mathbf{q}}_\mathbf{o} \rangle, \nu^{-} \rangle$$

Let $\mathbf{f}^{*}$ be a vector satisfying the first of the Kuhn-Tucker conditions 3.55, then we replace f in $L(\mathbf{f}, \mu, \lambda^{+}, \lambda^{-}, \nu^{+}, \nu^{-})$ with $\mathbf{f}^{*}$, which now is a function of Lagrangian multipliers. According to Saddle point theorem, the Lagrangian is minimized with respect to primal variables f, maximized with respect to dual variables (multipliers), thus we have

$$L(\mathbf{f}^{*}, \mu, \lambda^{+}, \lambda^{-}, \nu^{+}, \nu^{-}) \le L(\mathbf{f}^{*}, \mu^{*}, \lambda^{+*}, \lambda^{-*}, \nu^{+*}, \nu^{-*})$$

To show the convergence of the algorithm, we take derivatives of the Lagrangian with respect to μ

$$\frac{\partial L(\mathbf{f}^{*}, \mu, \lambda^{+}, \lambda^{-}, \nu^{+}, \nu^{-})}{\partial \mu}$$

$$= \langle \frac{1}{\theta} \ln \mathbf{f}^{*} + \bar{\mathbf{c}} + \langle \Delta_u, \mu \rangle + \langle \Delta_o, \lambda^{+} \rangle - \langle \Delta_o, \lambda^{-} \rangle + \langle \Delta_o, \nu^{+} \rangle - \langle \Delta_o, \nu^{-} \rangle, \frac{\partial \mathbf{f}^{*}}{\partial \mu} \rangle + \Delta_u \mathbf{f}^{*} - \mathbf{C}$$

Using the relationship 3.55, we obtain

$$\frac{\partial L(\mathbf{f}^*, \mu, \lambda^+, \lambda^-, \nu^+, \nu^-)}{\partial \mu} = \Delta_u \mathbf{f}^* - \mathbf{C}$$

For any link $a \in A_u$, if $\sum_r \sum_s \sum_k \delta_{a,k}^{rs} f_k^{rs} < C_a$, the constraint is inactive, so μ_a should remain equal to 0; Otherwise, the derivative of $L(\mathbf{f}^*, \mu, \lambda^+, \lambda^-, \nu^+, \nu^-)$ with respect to μ_a is positive. Thus, to maximize $L(\mathbf{f}^*, \mu, \lambda^+, \lambda^-, \nu^+, \nu^-)$, μ_a should increase. From 3.55 we know that the increase of μ_a will reduce the corresponding path flows, and hence the value $\sum_r \sum_s \sum_k \delta_{a,k}^{rs} f_k^{rs}$. Apparently, to increase $L(\mathbf{f}^*, \mu, \lambda^+, \lambda^-, \nu^+, \nu^-)$ as much as possible, we should increase μ_a such that $\sum_r \sum_s \sum_k \delta_{a,k}^{rs} f_k^{rs} = C_a$, but not beyond. It is easy to verify that the adjustment made in *Step1.2* of the IBA algorithm exactly achieves this. The similar argument can be applied to other constraints. As the algorithm always increases $L(\mathbf{f}^*, \mu, \lambda^+, \lambda^-, \nu^+, \nu^-)$ it converges provided there exists a feasible solution. This completes the proof.

However, in practice there are situations where E-LPFE-SUB does not have a feasible solution thus algorithm IBA would not converge. According to Bell (Bell et al. 1997), the major ones are

1. Traffic counts of one or more links belonging to A_o are not consistent with capacities of one or more links belong to A_u .
2. Traffic counts of some links belong to A_0 are mutually inconsistent.
3. Existing path set does not cover one or more counted links.

Such situations arise as a result of either errors in traffic counting, the inconsistence between steady-state assumption and dynamic traffic behavior, or an inappropriate path set. Algorithm IBA offers a natural way to identify the infeasibility of constraints. Note that the multipliers associated with the problematic constrains would be tending to infinity. To fix the incorrectness of the path set is relatively easy (it can be done by generating more paths or deleting those inappropriate). However, other causes of infeasibility are much difficult to address. Since in reality one may never obtain an input which is fully consistent, it is important to have the algorithm tolerate minor constraint inconsistencies. Another issue that Algorithm IBA does not address is the violation of slackness conditions like 3.57. Since the constraints are examined sequentially, it is possible that an active constraint becomes inactive when checking other constraints, which produces the violation of slackness conditions (while the constraint is inactive, the corresponding multipliers remain positive). We present a revised iterative balancing algorithm (RIBA) to resolve these problems.

ALGORITHM RIBA

```
Step 0  Initialization. Set $\mu = 0, \lambda^{+} = 0, \lambda^{-} = 0, v^{+} = 0, v^{-} = 0, \kappa = -\infty, \alpha = 1.0$ .
Step 1  Set $\kappa_0 = \kappa$ . Iteratively check constraints and update path flows.
    Step 1.1  Calculate path flow $\mathbf{f}$ according to Equation 3.55. Set $\tilde{\mathbf{x}} = \Delta\mathbf{f}, \tilde{\mathbf{q}} = \mathbf{Mf}$
    Step 1.2  For each link $a \in A_u$ , compute $e = \ln\tilde{x}_a - \ln C_a$
    If $e \geq 0$ , set $\mu_a = \mu_a + \alpha e$ ; otherwise, if $\mu > 0$ , set $\mu_a = \mu_a + e$ .
    Update $\mathbf{f}$ , $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$ , if $\kappa < |e|$ , set $\kappa = |e|$ .
    Step 1.3  For each link $a \in A_o$ , compute $e = \ln\tilde{x}_a - \ln\bar{x}_a(1 + \gamma_a)$
    If $e > 0$ , set $\lambda_a^{+} = \lambda_a^{+} + \alpha e$ ; otherwise, if $\lambda_a^{+} > 0$ , set $\lambda_a^{+} = \lambda_a^{+} + e$ .
    Update $\mathbf{f}$ , $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$ , if $\kappa < |e|$ , set $\kappa = |e|$ .
    Step 1.4  For each link $a \in A_o$ , compute $e = \ln\tilde{x}_a - \ln\bar{x}_a(1 - \delta_a)$
    If $e < 0$ , set $\lambda_a^{-} = \lambda_a^{-} - \alpha e$ ; otherwise, if $\lambda_a^{-} > 0$ , set $\lambda_a^{-} = \lambda_a^{-} - e$ .
    Update $\mathbf{f}$ , $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$ , if $\kappa < |e|$ , set $\kappa = |e|$ .
    Step 1.5  For each O-D pair $rs$ for which the target O-D information exists
    Compute $e = \ln\tilde{q}^{rs} - \ln\bar{q}^{rs}(1 + \eta_{rs})$
    If $e > 0$ , set $v_{rs}^{+} = v_{rs}^{+} + \alpha e$ ; otherwise, if $v_{rs}^{+} > 0$ , set $v_{rs}^{+} = v_{rs}^{+} + e$
    Update $\mathbf{f}$ , $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$ . If $\kappa < |e|$ , set $\kappa = |e|$ .
    Step 1.6  For each O-D pair $rs$ for which the target O-D information exists
    Compute $e = \ln\tilde{q}^{rs} - \ln\bar{q}^{rs}(1 - \varphi_{rs})$
    If $e < 0$ , set $v_{rs}^{-} = v_{rs}^{-} - \alpha e$ ; otherwise, if $v_{rs}^{-} > 0$ , set $v_{rs}^{-} = v_{rs}^{-} - e$ .
    Update $\mathbf{f}$ , $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$ . If $\kappa < |e|$ , set $\kappa = |e|$ .
Step 2  Convergence test
    Step 2.1  If $\kappa/\alpha < \epsilon$ , stop; otherwise, go to Step2.2.
```

Step 2.2 If $|(\kappa - \kappa_0)/\kappa_0| < \epsilon$, set $\alpha = \alpha\beta$. return to **Step**1.

Note that α is the step size while β is a positive scalar smaller than 1. α is set as 1 at the beginning, then reduced by applying β whenever the two consecutive iterations do not lead to sufficient decrease of κ, the convergence indicator. Further, when the violation of slackness conditions are detected, associated multipliers are reduced so as to reactivate the constraints.

3.3.2 A descent direction

In this section, we relax the assumption that all link travel times are fixed. Instead, the travel time on link $a \in A_u$ are treated as an increasing function of the link flow x_a. In this case, algorithm RIBA cannot directly obtain the optimal solution since a subset of travel times are flow dependent. However, the following proposition shows that RIBA can be used to provide a descent direction.

Proposition 3.2 *Assume that a path set is predetermined for E-LPFE. Let* $\mathbf{f^0}$ *be a feasible path flow pattern, and the associated path cost is* $\mathbf{c^0}$. *If* $\mathbf{f}^*$ *solves E-LPFE-SUB (in which* $\tilde{\mathbf{c}}$ *is replaced by* $\mathbf{c^0}$ *), then* $\mathbf{f}^* - \mathbf{f^0}$ *provides a descent direction for E-LPFE at* $\mathbf{f^0}$.

Proof: Let $z(\mathbf{f})$ denote the objective function of $E - LPFE$. To show $\mathbf{f}^* - \mathbf{f^0}$ is a descent direction, we need to show

$$\langle \nabla z(\mathbf{f^0}), \mathbf{f}^* - \mathbf{f^0} \rangle < 0$$

This equals to

$$\langle c^0 + \frac{1}{\theta} \ln \mathbf{f^0}, \mathbf{f}^* - \mathbf{f^0} \rangle < 0 \Rightarrow \langle \mathbf{c^0}, \mathbf{f}^* \rangle + \frac{1}{\theta} \langle \mathbf{f}^*, \ln \mathbf{f^0} \rangle < \langle \mathbf{c^0}, \mathbf{f^0} \rangle + \frac{1}{\theta} \langle \mathbf{f^0}, \ln \mathbf{f^0} \rangle \tag{3.75}$$

Now let $y(\mathbf{f}) = \langle \mathbf{f}, \ln \mathbf{f} \rangle$. $y(\mathbf{f})$ is a strictly convex function because its Hessian is always positive definite for any $\mathbf{f} > 0$. Thus, we have the following

$$y(\mathbf{f}^*) > y(\mathbf{f^0}) + \nabla y(\mathbf{f^0})(\mathbf{f}^* - \mathbf{f^0}) \Rightarrow \langle \mathbf{f}^*, \ln \mathbf{f}^* \rangle > \langle \mathbf{f}^*, \ln \mathbf{f^0} \rangle + \mathbf{f}^* - \mathbf{f^0}$$

When $\mathbf{f}^* \to \mathbf{f^0}$, we obtain $\langle \mathbf{f}^*, \ln \mathbf{f}^* \rangle > \langle \mathbf{f}^*, \ln \mathbf{f^0} \rangle$. Incorporate this relationship into 3.75, we have

$$\langle \mathbf{c^0}, \mathbf{f}^* \rangle + \frac{1}{\theta} \langle \mathbf{f}^*, \ln \mathbf{f}^* \rangle < \langle \mathbf{c^0}, \mathbf{f^0} \rangle + \frac{1}{\theta} \langle \mathbf{f^0}, \ln \mathbf{f^0} \rangle$$

This is apparently satisfied since $\mathbf{f}^*$ is an optimal solution of E-LPFE-SUB, which should achieve the minimum objective function value. End of proof.

Thus, the optimal solution to E-LPFE can be obtained by iteratively solving E-LPFE-SUB and moving along the resulted descent direction. We proceed to present a descent direction algorithm in which the Golden-section line search is used to find a suitable step size.

ALGORITHM DDAG[3]

Step 0 Initialization. Set iteration index $i = 0$, $x_a = 0 \forall a \in A, t_a = t_a(0)$, $\forall a \in A_u$.[4]

Call Algorithm RIBA to obtain $\mathbf{f^0}$.

Step 1 Find the descent direction. Update link travel times based on $\mathbf{f^i}$, then call Algorithm RIBA to obtain a new path flow pattern, denoted as $\mathbf{g^i}$. Set search direction $\mathbf{d^i} = \mathbf{g^i} - \mathbf{f^i}$.

Step 2 Find step size α .

Step 2.1 Initialization. Set $l = 0.0, b = G = (\sqrt{5} - 1)/2, \quad a = 1 - G, u = 1.0$. Compute $z_a = z(\mathbf{f^i} + a\mathbf{d^i}), z_b = z(\mathbf{f^i} + b\mathbf{d^i})$

Step 2.2 If $z_a < z_b$, $l = a, a = b, b = Ga + (1 - G)u$, else $u = b, b = a, a = Gb + (1 - G)l$. Compute z_a, z_b .

Step 2.3 If $(u - l) > (b - a)\epsilon_\alpha$, return *Step2.2*; otherwise if $z_a < z_b$, $\alpha = a$, else $\alpha = b$, go to **Step3** .

Step 3 Move along search direction. Set $\mathbf{f^{i+1}} = \mathbf{f^i} + \alpha\mathbf{d^i}$. If $|\mathbf{f^{i+1}} - \mathbf{f^i}| < \epsilon$, stop; otherwise, set *i* = *i* + *1*, return to **Step1**.

In Algorithm DDAG, we terminate the procedure when the two consecutive iterates are sufficiently close.

[3] Stands for Descent Direction Algorithm with Golden-section line search

[4] Travel times on links belong to A_o are measured thus not computed from link performance functions.

3.3.3 Column generation

So far a path set (hence the path-link incidence matrix Δ) is assumed to be predetermined. Ideally, all existing paths should be enumerated before hand. This is of course computationally prohibitive even for a network of modest size. In fact, a great portion of available paths would remain unused in a complex, congested network. Different strategies have been suggested to set up a ``proper'' path set, of which the most famous one is due to Dial (1971) , who proposed to reduce the original network into an acyclic one containing only “efficient” links. As we have noticed, however, a predetermined path set may introduce inconsistent constraints. Thus, it is desirable to generate paths iteratively only when they becomes needed, based on the diagnostic information obtained from previous iterations. This is known as *column generation* because the columns of path-link incidence matrix Δ are generated as the iterations progresses.

A shortest path algorithm is commonly used to generate new paths. However, the calculation of shortest paths is not necessarily based on travel times. In logit path flow estimators, for example, the paths that might help remove inconsistency of constraints should be given high priority. This can be achieved by arbitrarily decreasing/increasing link travel times so that the shortest path algorithm would be in favor/disfavor the choice of paths containing associated links. This idea is exhibited in the following column generation algorithm (CGA).

ALGORITHM CGA

Step 0 Initialization. Set iteration index $i = 0, x_a = 0 \forall a \in A, t_a = t_a(0), \forall a \in A_u$. Compute shortest paths for all O-D pairs, and build path-link incidence matrix Δ accordingly. Set K^i as the number of shortest paths.

Step 1 Solve the restricted master problem (RMP). Call Algorithm DDAG to get the current path and link solutions.

Step 2 Column generation

Step 2.1 Update link costs. For each link $a \in A_u$, $t_a = t_a(x_a^i) + \mu_a^i$.

For each link $a \in A_o$, if $\bar{x}_a(1-\delta_a) \le x_a \le \bar{x}_a(1+\gamma_a), t_a = \bar{t}_a$;

else if $x_a > \bar{x}_a(1+\gamma_a)$, $t_a = \bar{t}_a + \lambda^+$; else $t_a = \max(\bar{t}_a - \lambda^-, 0)$.

Step 2.2 Compute shortest paths for all O-D pairs based on updated link travel times $\mathbf{t}$, expand the matrix Δ with new paths.

Step 3 Convergence test. Set $i = i+1$, calculate the current number of shortest paths K^i . If $|(K^i - K^{i-1})/K^{i-1} < \epsilon|$, stop; otherwise, return to Step 1.

Algorithm CGA terminates when the number of new generated paths is small enough, implying that the matrix Δ is not changing much with the few new columns. In *Step2.1*, link travel times are modified by the corresponding multipliers. For unmeasured links, this means that the queuing delay is taken into consideration. On measured links, link costs are reduced by λ^- when the estimated link flow is less than the lower bound of the traffic count. Conceivably, this link is more likely to be selected by a shortest path algorithm in the next round since its cost may be significantly lowered. Note that we restrict the link travel times to be non-negative to avoid creating negative cycles that would cause the shortest path search to fail.

One could also use a K-shortest path algorithm to generate new paths. That is, in each iteration, not only the shortest path, but the second, third,..., and *Kth* shortest paths are generated for each O-D pair. Although the K-shortest path algorithm is more computationally demanding, it may accelerate the stabilization of Δ thereby improve the overall convergence speed of Algorithm CGA.

3.3.4 Time-dependent case

Now we turn to the solution algorithms for E-TD-LPFE. As in the steady case, we first assume that link travel times are fixed and the matrix Δ is given for each time interval t . Similarly, iterative balancing algorithm can be used to sequentially examine Kuhn-Tucker conditions. However, in this case, we need to deal with an additional optimal condition, the relationship between queue and delay (Equation 3.45). The revised iterative balancing algorithm for the time-dependent case(RIBATD) is given as below.

ALGORITHM RIBATD

```
Step 0  Initialization. Set $\lambda_t^+ = 0, \lambda_t^- = 0, \nu_t^+ = 0, \nu_t^- = 0, \kappa = -\infty, \alpha = 1.0$ . Set
        $\mathbf{v_t} = \max(0, \mathbf{v_{t-1}} + \mathbf{C} + \epsilon), \mu_t = \langle \mathbf{D^{-1}}, \mathbf{v_t} \rangle$
Step 1  Set $\kappa_0 = \kappa$. Iteratively check constraints and update path flows.
    Step 1.1  Calculate path flow $\mathbf{f_t}$ according to Equation 3.55. Set
        $\mathbf{\tilde{x}_t} = \Delta\mathbf{f_t}, \mathbf{\tilde{q}_t} = \mathbf{M}\mathbf{f_t}$
    Step 1.2  For each link $a \in A_u$, compute $s = 1 + C_a/(C_a + v_a^t - v_a^{t-1})$,
        $e = (\ln\tilde{x}_a^t - \ln(C_a + v_a^t - v_a^{t-1}))/s$
        If $e \geq 0$, set $\mu_a^t = \mu_a^t + e$; otherwise, if $\mu_a^t > 0$, set $\mu_a^t = \mu_a^t + e$.
        Update $\mathbf{f}$, $\mathbf{\tilde{x}}$ and $\mathbf{\tilde{q}}$. Set $v_a^t = C_a\mu_a^t$. If $\kappa < |e|$ , set $\kappa = |e|$.
    Step 1.3  For each link $a \in A_o$, compute $e = \ln\tilde{x}_a^t - \ln\bar{x}_a^t(1 + \gamma_a^t)$
        If $e > 0$, set $\lambda_{a,t}^+ = \lambda_{a,t}^+ + \alpha e$; otherwise, if $\lambda_{a,t}^+ > 0$, set
        $\lambda_{a,t}^+ = \lambda_{a,t}^+ + e$.
        Update $\mathbf{f_t}$, $\mathbf{\tilde{x}_t}$ and $\mathbf{\tilde{q}_t}$, if $\kappa < |e|$ , set $\kappa = |e|$.
    Step 1.4  For each link $a \in A_o$, compute $e = \ln\tilde{x}_a^t - \ln\bar{x}_a^t(1 - \delta_a^t)$
        If $e < 0$, set $\lambda_{a,t}^- = \lambda_{a,t}^- - \alpha e$; otherwise, if $\lambda_{a,t}^- > 0$, set $\lambda_{a,t}^- = \lambda_{a,t}^- - e$.
        Update $\mathbf{f_t}$, $\mathbf{\tilde{x}_t}$ and $\mathbf{\tilde{q}_t}$, if $\kappa < |e|$, set $\kappa = |e|$.
    Step 1.5  For each O-D pair $rs$ for which the target O-D information exists
        Compute $e = \ln\tilde{q}_t^{rs} - \ln\bar{q}_t^{rs}(1 + \eta_{rs}^t)$.
        If $e > 0$, set $\nu_{rs,t}^+ = \nu_{rs,t}^+ + \alpha e$; otherwise, if $\nu_{rs,t}^+ > 0$, set $\nu_{rs,t}^+ = \nu_{rs,t}^+ + e$
        Update $\mathbf{f_t}$, $\mathbf{\tilde{x}_t}$ and $\mathbf{\tilde{q}_t}$. If $\kappa < |e|$, set $\kappa = |e|$.
    Step 1.6  For each O-D pair $rs$ for which the target O-D information exists
        Compute $e = \ln\tilde{q}_t^{rs} - \ln\bar{q}_t^{rs}(1 - \varphi_{rs}^t)$
        If $e < 0$, set $\nu_{rs,t}^- = \nu_{rs,t}^- - \alpha e$; otherwise, if $\nu_{rs,t}^- > 0$, set $\nu_{rs,t}^- = \nu_{rs,t}^- - e$.
        Update $\mathbf{f_t}$, $\mathbf{\tilde{x}_t}$ and $\mathbf{\tilde{q}_t}$. If $\kappa < |e|$, set $\kappa = |e|$.
Step 2  Convergence test
    Step 2.1  If $\kappa/\alpha < \epsilon$, stop; otherwise, go to Step2.2.
    Step 2.2  If $|(\kappa - \kappa_0)/\kappa_0| < \epsilon$, set $\alpha = \alpha\beta$. return to Step1.
```

Note that RIBATD and RIBA are very similar (ignoring the time index t in RIBATD) except in **Step0** and *Step1.2*. In **Step0**, the queue in the current time period is initialized according to the queue left over from the last time interval. If the residual queue exceeds the capacity, the initial value of v_t should be such set that s in *Step1.2* is always positive.

Proposition 3.3 *Assuming that link travel times are fixed and the matrix* Δ *is given for each time interval* t*, algorithm RIBATD converges to the optimal solution of E-TD-LPFE provided a feasible solution exists.*

Proof: the arguments used in the proof of Proposition 3.1 can be applied here in general. We only need to show that the adjustment made in *Step1.2* of RIBATD ensures the increase of the Lagrangian. First, note that $e = \ln \tilde{x}_a^t - \ln(C_a + v_a^t - v_a^{t-1})$ is an over-adjustment because it would make the adjusted link flow, $[\tilde{x}]_a^t = C_a + v_a^t - v_a^{t-1}$. However, since v_a^t is increased accordingly, the constraint is inactive after the adjustment. The desirable update thus should be

$$[\mu_a^t] = \mu_a^t + \ln \tilde{x}_a^t - \ln(C_a + [v_a^t] - v_a^{t-1} \tag{3.76}$$

where $[v_a^t]$ denotes the queue length after the adjustment. However, it is not easy to directly determine $[\mu_a^t]$ since $[\mu_a^t]$ and $[v_a^t]$ are mutually dependent. The difficulty can be overcome by making use of the concaveness of the log function. Note that the following inequality holds

$$\ln(C_a + [v_a^t] - v_a^{t-1}) < \ln(C_a + v_a^t - v_a^{t-1}) + \frac{[v_a^t] - v_a^t}{C_a + v_a^t - v_a^{t-1}}$$

Thus, we can replace $-\ln(C_a + [v_a^t] - v_a^{t-1}$ in 3.76 by the right hand side of the above inequality. After some simple algebra, we obtain

$$[\mu_a^t] > \mu_a^t + (\ln \tilde{x}_a^t - \ln(C_a + v_a^t - v_a^{t-1})/(1 + \frac{C_a}{C_a + v_a^t - v_a^{t-1}})$$

The above inequality implies that the adjustment (the second term on the right hand side) is smaller than the maximum possible value. However, it effectively avoids over-adjustment and ensures convergence. Apparently, *Step1.2* of RIBATD employs this reduced adjustment. This completes the proof.

There exist other methods to update $[\mu_a^t]$ and $[v_a^t]$ according to Equation 3.76. The method of successive averages (MSA), for example, can be fitted as follows:

ALGORITHM MSATD

Step 0...

Step 1...

Step 1.2 For each link $a \in A_u$, if $C_a < v_a^{t-1} + \tilde{x}_a^t, [v_a^t] = v_a^{t-1} + \tilde{x}_a^t - C_a$; else, $[v_a^t] = 0$.
Compute $v_a^t = [v_a^t]/i + v_a^t(i-1)/i$, $e = u_a^t - v_a^t/C_a, u_a^t = v_a^t/C_a$.
Update $\mathbf{f}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{q}}$. If $\kappa < |e|$, set $\kappa = |e|$.

Step 2...

Since MSATD is the same as RIBATD except in *Step1.2,* we ignore the other steps in the above description to save space. The convergence of MSA is well known thus its proof is not repeated here.

Algorithm DDAG and CGA presented earlier can be applied to handle flow-dependent link travel times and path generation in the time-dependent case. The only difference is that in DDAG, either Algorithm RIBATD or MSATD (not RIBA) should be called to produce a sub-problem solution.

Finally, an algorithmic framework accommodating multiple time intervals is given as follows:

ALGORITHM MTI

Step 0 Initialization. Set $t = 0, \mathbf{v_t} = 0$.

Step 1 Solve the steady-state problem. Set $t = t + 1$, call Algorithm CGA to obtain $\mathbf{f_t}, \mathbf{v_t}$, compute estimated O-D matrix $\tilde{\mathbf{q}}_\mathbf{t}$ according to $\mathbf{f_t}$.

Step 2 if $t \geq T$, stop; otherwise, return **Step1**.

where T is the total number of time intervals.

Chapter 4. An Object-Oriented Implementation

The algorithms described in the last chapter are implemented using an object-oriented programming (OOP) design. Compared to the conventional programming approaches, an OOP design is preferable because

- OOP provides a clear modular structure to define abstract data types (objects) that encapsulate data and functions operating on it.
- OOP enhances the reusability of programs through the use of inheritance and polymorphism.
- OOP provides a good framework for code libraries where supplied software components can be easily adopted and modified by the programmer.

The reusability and extensibility of the OOP design make it easy to maintain and upgrade the developed PFE software tool as needs arise (e.g., when new formulations/algorithms need to be implemented). An OOP-based PFE tool also provides great convenience and flexibility for developing graphical user interfaces.

We shall first present the overall class hierarchy of our implementation and then go through the details of classes. Finally we will explain how the classes can be complied as a class library and discuss its potential usage.

4.1 Class Hierarchy

The hierarchy of the implemented classes is demonstrated in Figure 4.1. As shown, all subclasses are derived from the generic class `GEN_IA`, which provides a framework for implementing general iterative algorithms. Among others, `GEN_IA` imposes a loop structure that consists of three fundamental components: initialization, main operation and convergence test.

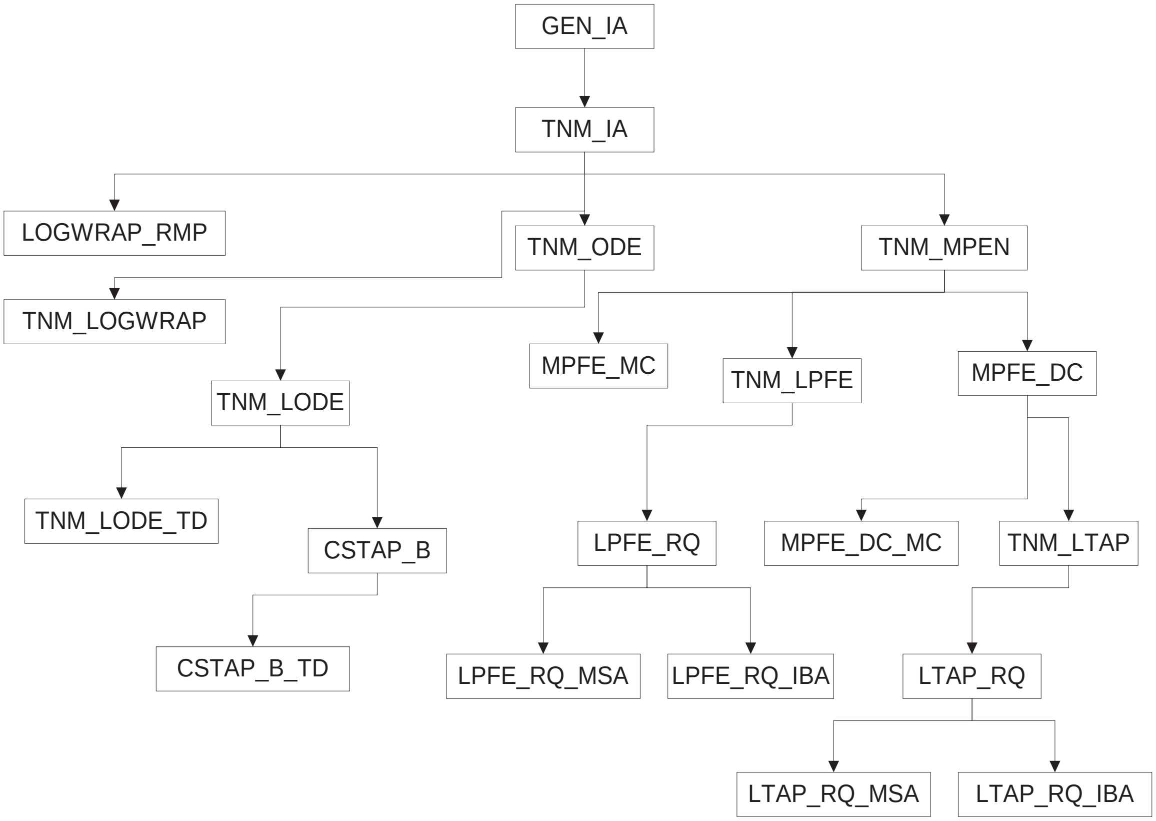


Figure 4.1 A class hierarchy tree

The first derived class of GEN_IA, TNM_IA, lays a foundation for developing network-related algorithms. The most important function of TNM_IA is to offer its subclasses an access to network objects defined in a C++ class library called TNM[1]. TNM defines both static and dynamic network objects and provides numerous functions to solve a variety of network problems, including shortest and K-shortest path problems, maximum flow problems, path enumeration, dynamic shortest path problems, stochastic routing problems, and dynamic network loading (simulation). This significantly reduces the programming task involved in this project, for many required functions (e.g., the calculation of shortest/K-shortest paths in the column generation algorithm) have been defined in TNM.

Subclass TNM_MPEN deals with the following mathematical program which can be regarded as a composite of logit traffic assignment models and path flow estimators,

[1] TNM is developed by Professor H. M. Zhang's research group at UC Davis in an independent research effort.

$$\min \frac{1}{\theta}\langle \mathbf{f}, \ln \mathbf{f} - \mathbf{I}\rangle + \sigma\langle \mathbf{c}, \mathbf{f}\rangle \tag{4.1}$$

subject to

$$\mathbf{Mf} = \mathbf{q} \quad \text{Flow conservation conditions} \tag{4.2}$$

$$\Delta\mathbf{f} \leq \mathbf{C} \quad \text{Capacity constraints} \tag{4.3}$$

$$\langle \bar{\mathbf{x}}_{\mathbf{o}}, \mathbf{I}_o - \delta_o\rangle \leq \Delta_o\mathbf{f} \leq \langle \bar{\mathbf{x}}_{\mathbf{o}}, \mathbf{I}_o + \gamma_o\rangle \quad \text{Link flow measurements} \tag{4.4}$$

$$\langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o - \varphi_o\rangle \leq \mathbf{M}_{\mathbf{o}}\mathbf{f} \leq \langle \bar{\mathbf{q}}_{\mathbf{o}}, \mathbf{J}_o + \eta_o\rangle \quad \text{O-D flow measurements} \tag{4.5}$$

Apparently, only a subset of constraints should be considered in a particular problem. Further, we may need to include travel costs in the objective function in one problem, but need not in another. The derived classes of TNM_MPEN aim at realizing such polymorphism. Specifically, TNM_MPEN itself does not consider constraints 4.2 and exclude travel costs in the object function ($\sigma = 0.0$). MPEN_MC uses the same constraints but consider travel costs. On the other branch, both MPEN_DC and MPEN_DC_MC include the flow conservation constraints, with $\sigma = 0.0$ in the former and 1.0 in the latter. Further, subclasses TNM_LPFE and TNM_LTAP explicitly set up the constraint structure for assignment and O-D estimation problems respectively, thus serve as base classes for the two types of problems. TNM_LPFE actually implements Algorithm RIBA for E-LPFE-SUB discussed in Section 3.3.1. Derived from TNM_LPFE, LPFE_RQ deals with the logit path flow estimator with residual queues, the formulation used in the time-dependent case. LPFE_RQ's two subclasses, LPFE_RQ_MSA LPFE_RQ_IBA, implement Algorithms MSATD and RIBATD respectively. A similar design applies to the subclass TNM_LTAP and its derived classes, which solve corresponding assignment problems.

Subclass LOGWRAP_RMP implements Algorithm DDAG to handle the flow-dependent link travel times. LOGWRAP_RMP is able to deal with most formulations discussed in the last chapter provided a correct sub problem solver is selected. For example, to solve E-LPFE-TD with the method of successive average, the sub problem solver of DDAG should be an LPFE_RQ_MSA object. TNM_LOGWRAP provides a general wrapper for performing column generation, and requires an object of `LOGWRAP_RMP` to solve the sub problem.

`TNM_ODE` is a base class for general O-D estimation problems. This subclass contains functions specific to O-D estimation problems, particularly input-output (I-O) functions such as reading measured link traffic counts from disk files and preparing estimation reports. Derived from `TNM_ODE`, `TNM_LODE` is further specified to target logit path flow estimators. `TNM_LODE` can be regarded as the outmost wrapper of LPFE objects, containing an object of `TNM_LOGWRAP` as an internal solver and providing necessary I-O supports. `TNM_LODE_TD` implements the framework described in Algorithm MTI for solving time-dependent LPFE, and overriding the I-O functions accordingly to meet the need of processing multiple time-intervals. Similarly, `CSTAP_B` and `CSTAP_B_TD` are outmost wrappers for steady-state and time-dependent logit traffic assignment problems (BSUE and TD-BSUE in Section 3.1.2), respectively.

4.2 Class Description

This section provides implementation details of all classes. Note that the class members of minor importance are not presented here.

CLASS GEN_IA PARENT CLASS: N/A

Data Members:

GEN_IA*subProblem - a pointer of a sub problem.

GEN_IA* auxProblem - a pointer of an auxiliary problem.

double OFV - objective function value.

int curIter - current iteration index.

double stepSize - current step size.

float cpuTime - consumed CPU time.

int ziggIter - the number of iterations of “no-improvement”. “No-improvement**”** means that an expected improvement (i.e., an adequate decrease of OFV) is not achieved.

int badIter - the number of iterations during which OFV changes opposite to the expected direction.

double convIndicator - an indicator of convergence

double ziggIndicator - an indicator of ``no-improvement".

int maxMainIter - maximum number of iterations permitted.

double convCriterion - a threshold value that is used to determine convergence.

double ziggCriterion - a threshold value that is used to determine no-improvement.

int maxZiggIter - maximum permitted number of no-improvement iterations.

int maxBaditer - maximum permitted number of bad iteration.

double lineSearchAccuracy - required accuracy in line search.

int maxLineSearchIter - maximum permitted number of line search

double stepSizeLB - the minimum permitted step size

Constructor

Void GEN_IA() - an empty default constructor

Return - none.

Attributes:

bool ReachMaxIter() - test if the maximum number of iterations is attained

Return - a boolean value

bool ReachMaxBad() - test if the maximum number of bad iterations is attained

Return - a boolean value

bool ReachAccuracy() - test if required convergence criterion is achieved

Return - a boolean value

bool ReachZigg() - test if no-improvement happens

Return - a boolean value

bool ReachMaxZigg() - test if maximum number of no-improvement iterations is attained a boolean value

bool ReachError() - test if an error is encountered, a boolean value

bool ReachUser() - test if the solution process is terminated externally (most likely by a terminal user)

Return - a boolean value

Bool ReachUser() - test if the solution process is terminated externally (most likely by a terminal user)

Return - a boolean value

Operations:

TERMFLAGS Solve() - the underlying operation. Call this function to solve the optimization program associated with the class.

Return - The termination reason of the solution process

int SolveSubProblem() - solve the sub problem. OFV will be automatically updated using the OFV obtained from sub problem.

Return - 0 if solving succeeds, 1 otherwise.

int SolveAuxProblem() - solve the auxiliary problem.

Return - 0 if solving succeeds, 1 otherwise.

int ExportPar(parFile) - write algorithmic parameters (e.g., maxMainIter, convCriterion etc.) into a file named parFile.alg

Return - 0 if succeeds, 1 otherwise

string parFile - a string specifying the output file name.

int ReadPar(parFile) - reading parameters from a disk file named parFile.alg

Return - if succeeds, 1 otherwise

string parFile - a string specifying the input file name.

void ImportDynamicMem(rhs) - **i**mport data members pertinent to the solution process (such as OFV, convIndicator etc.) from a given algorithm object.

Return **-** none.

GEN_IA *rhs **-** a pointer to a **GEN_IA** object.

intBisecSearch() **-** search step size using the bisection method.

Return **-** 0 if descent, 1 otherwise.

void GoldentSearch() **-** search step size using Golden method.

Return **-** none.

Overridables:

int Build(inFile, outFile, format) **-** build the algorithm object, including reading required input data, allocating memory and open I-O files. An algorithm object must be ``built'' before its Solve function can be called.

Return **-** 0 if building succeeds; a non-zero number otherwise.

string inFile **-** input file name.

string outFile **-** the output file name.

ENUM format **-** an enumeration type that describes input file format.

int Build(rhs) **-** build the algorithm object from an existing object. It is often used to build a sub problem according to its host object.0 if building succeeds; a non-zero number otherwise.

GEN_IA rhs **-** a pointer to an **GEN_IA** object.

void Initialize() **-** initialize the solution process. It must be overridden in derived classes.

Return **-** none

void ComputeOFV() **-** compute the objective function value, update OFV accordingly.

Return - none

void MainLoop() - define major operations of the solution process. It must be overridden in derived classes.

Return - none

bool Terminate() - test if the solution process should be terminated.

Return - a boolean value

void PreProcess() - this function is called before Initialize() in the solution process. It is used to accommodate some special building/initialization operations.

Return - none

void PostProcess() - this function is called after the solution process is terminated.

Return - a boolean value

int Report() - write solutions obtained from Solve() into disk files.

Return - 0 if succeeds; a non-zero value otherwise

int PrePareReport() - Open report files and write headers. It must be called before Report().

Return - 0 if succeeds; a non-zero value otherwise

int CloseReport() - Close report files.

Return - 0 if succeeds; a non-zero value otherwise

CLASS TNM_IA PARENT CLASS:: GEN_IA

Data Members:

TNM_SNET* network - a pointer of a `TNM_SNET` object. `TNM_SNET` is a static network class defined in **TNM**. This network object is initialized in the overridden

Build() function.

double costScalar **-** a positive scalar to re-scale link travel times.

ENUMl pf **-** an enumeration type of link performance functions.

Constructor:

void TNM_IA() **-** an empty default constructor.

Return - none.

CLASS TNM_MPEN PARENT CLASS: TNM_IA

Data Members:

double theta **-** The dispersion parameter θ in logit models.

double costCoef **-** The coefficient of travel cost term in the objective function (i.e., σ in Equation 4.1).

vector volConLink **-** a vector of pointers of **TNM_SLINK** (defined in **TNM**) objects associated with constraint 4.4.

vector capConLink **- a** vector of pointers of **TNM_SLINK** (defined in **TNM**) objects associated with constraint 4.3.

vector conDest **-** a vector of pointers of **TNM_SDEST** (defined in **TNM**) objects associated with constraint 4.2 or 4.5.

Constructor:

void TNM_MPEN(theta) **-** default constructor

Return **-** none

double theta **-** the dispersion parameter θ .

Attributes:

double GetTheta() **-** Get the dispersion parameter θ

Return - θ

Operations:

double **-**updateFlowPattern(link, shift)Update path and link flows based on the change of a multiplier made when checking a constraint associated with a link.shifted flows.

ENUM* link **-** a pointer of a **TNM_SLINK** (a static link class defined in **TNM**) object.

double shift **-** the change of the multiplier at the current iteration.

double UpdateFlowPattern(od, shift) **-** Update path and link flows based on the change of a multiplier made when checking a constraint associated with an OD pair.

Return - shifted flows.

ENUM* oda - pointer of a **TNM_SDEST**(a static O-D class defined in **TNM**) object.

double shift - the change of the multiplier at the current iteration.

double GetODIEMuChange(od) **-** Compute the required change of a multiplier associated with a constraint of O-D measurement.

Return - the change of the multiplier.

ENUM* oda - pointer of a **TNM_SDEST** object.

int SetTheta(theta) - Set the dispersion parameter θ

Return - 0 if succeeds, non-zero value otherwise.

double theta - the dispersion parameter

Overridables:

int InitPathFlow() - Compute path flows using the logit choice model based on the initial values of multipliers.

Return - 0 if succeeds, non-zero value otherwise.

double GetVolLinkMuChange(link) - Compute the required change of a multiplier associated with a constraint of link measurement

Return - the change of the multiplier

ENUM* link - a pointer of a **TNM_SLINK** object.

double GetCapLinkMuChange(link) - Compute the required change of a multiplier associated with a constraint of link capacity

Return - the change of the multiplier

ENUM* link - a pointer of a **TNM_SLINK** object.

CLASS MPEN_MC PARENT CLASS: TNM_MPEN

Constructor:

void MPEN_MC(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ .

CLASS MPEN_DC PARENT CLASS: TNM_MPEN

Constructor:

void MPEN_DC(theta) - default constructor

Return - none

double theta - the dispersion parameter θ .

Operations:

void UpdateODMultiplier() -Update path and link flows according to the multiplier associated with the flow conservation constraint.

Return - none

CLASS MPEN_DC_MCMPEN_DC

Constructor

void MPEN_DC_MC(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ .

TNM_LPFETNM_MPEN

Constructor:

void TNM_LPFE(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ .

CLASS LPFE_RQ PARENT CLASS: TNM_LPFE

Constructor:

void LPFE_RQ(theta) - default constructor, in which costCoef is set to 1.0

Return - none

double theta - the dispersion parameter θ .

CLASS LPFE_RQ_MSA PARENT CLASS: LPFE_RQ

Constructor:

void LPFE_RQ_MSA(theta) - default constructor, in which costCoef is set to 1.0.

Return **-** none

double theta - the dispersion parameter θ .

CLASS LPFE_RQ_IBA   PARENT CLASS: LPFE_RQ

Constructor

void MPEN_DC_MC(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ .

CLASS TNM_LTAP   PARENT CLASS: MPEN_DC

Constructor:

void TNM_LTAP(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ.

CLASS LTAP_RQ   PARENT CLASS: TNM_LTAP

Constructor

void LTAP_RQ - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ.

CLASS LTAP_RQ_MSA   PARENT CLASS: LTAP_RQ

Constructor

voidLTAP_RQ_MSA(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ .

CLASS LTAP_RQ_IBA PARENT CLASS: LTAP_RQ

Constructor:

void LTAP_RQ_MSA(theta) - default constructor, in which costCoef is set to 1.0.

Return - none

double theta - the dispersion parameter θ .

CLASS LOGWRAP_RMP PARENT CLASS: TNM_IA

Data Members:

bool isRQ - consider residual queue in the formulation or not

bool isAsn - consider assignment or O-D estimation

bool isMsLinkTime - consider measured link travel times or not

Constructor:

void LOGWRAP_RMP(mt, at, t) - default constructor.

Return - none

ENUM mt - the type of sub problem.

ENUM at - the type of solution algorithm (only applies to the time-dependent case).

double t - dispersion parameter.

Attributes:

bool GetAsn() - Check the type of its *subproblemtrue* if the sub problem is an assignment model.

bool GetRQ() - Check the type of its *subproblemtrue* if the sub problem considers residual queues.

bool IsMsLinkTime() - Check whether or not measured link times are considered

Return - a boolean value

CLASS TNM_LOGWRAP PARENT CLASS: TNM_IA

Data Members:

bool pathEnum - whether or not enumerate all paths.

int initPath - the number of shortest paths required to generate for each O-D pair in initialization.

int genPath - the number of shortest paths required to generate in each iteration.

int pathNum - the total number of paths generated so far.

Constructor:

void TNM_LOGWRAP(mt, at, t, ep) - default constructor.

Return - none

ENUM mt - the type of subproblem.

ENUM at - the type of solution algorithm (only applies to the time-dependent case).

double t - dispersion parameter.

bool ep - whether or not enumerate all paths.

Attributes:

bool GetPathEnum() - Check if the path enumeration is activated

Return - a boolean value.

Operations:

void SetInitPath(n) - Set the value of initPaththe number of paths.

void SetGenPath(n) - Set the value of genPaththe number of paths.

CLASS TNM_ODE PARENT CLASS: TNM_IA

Data Members:

bool m_useTarget - whether or not consider O-D measurements

bool m_useLinkError -whether or not consider link measurement errors

bool m_useTargetError - whether or not consider O-D measurement errors

bool m_useMsLinkTime - whether or not consider link travel time measurements.

vector msLink - a vector of pointers of **TNM_SLINK** objects associated with measured links.

vector umsLink - a vector of pointers of **TNM_SLINK** objects associated with unmeasured links.

vector dmdVector - a vector of pointers of **TNM_SDEST** objects associated with measured O-D pairs.

Constructor:

void TNM_ODE() - default constructor

Return - none

Operations:

void EnableTarget(t) - Set the value of m_useTarget

Return - none

bool t - true if allow to consider O-D measurements

void EnableLinkError(t) - Set the value of m_useLinkError

Return - none

bool t - true if allow to consider link measurement errors

void EnableTargetError(t) - Set the value of m_useTargetError

Return - none

bool t - true if allow to consider link travel time measurements

int GetLinkError(lerFile) - Read link measurement errors from a disk file

Return - 0 if succeeds, non-zero otherwise

ifstream lerFile - a file pointer

int GetTargetError(terFile) - Read O-D measurement errors from a disk file

Return - 0 if succeeds, non-zero otherwise

ifstream terFile - a file pointer

double RmsetLink() - compute root mean squares of the measured and estimated link flows

Return - calculated root mean square

double RmsetOD(type) - compute root mean squares of two specified vectors of O-D flows

Return - calculated root mean square

enum type - specify two vectors of O-D flows.

Overridables:

int GetMsLink(oFile) - Read link measurements from a disk file

Return - 0 if succeeds, non-zero otherwise

ifstream oFile - a file pointer

in tGetMsLinkTime(mFile) - Read measured link travel times from a disk file

Return - 0 if succeeds, non-zero otherwise

ifstream mFile - a file pointer

int GetTargetOD(tFile) - Read measured O-D flows (or historical O-D) from a disk file

Return - 0 if succeeds, non-zero otherwise

ifstream tFile - a file pointer

CLASS TNM_LODE PARENT CLASS: TNM_ODE

Data Members:

double tdDemand** - a two dimensional array storing the time-dependent O-D table.

double tdLinkVol** - a two dimensional array storing the time-dependent link flows.

int timeInterval - the number of time intervals of interest. = 1 for the steady-state case.

int curTime - current time interval.

Constructor:

void TNM_LODE(at, mt, ep, t) - default constructor.

Return - none

ENUM at - the type of solution algorithm (only applies to the time-dependent case).

ENUM mt - the type of sub problem.

bool ep - whether or not enumerate all paths.

double t - dispersion parameter.

Attributes:

int GetTimeInterval() - get the total number of time intervals.

Return - timeInterval

int GetCurTime() - get the current time intervalcurTime

Operations:

void EnableMsLinkTimer(m) - set the value of m_useMsLinkTimenone

bool m - true if allow to consider measured link travel times.

int SaveSol(t) - save the solution at time t into an intermediate binary file

Return - 0 if succeeds, non-zero value otherwise

bool t - the time index

int LoadSol(t) - load the solution at time t from a binary file into memory

Return - 0 if succeeds, non-zero value otherwise

bool t - the time index

CLASS TNM_LODE_TD PARENT CLASS: TNM_LODE

Data Members:

double tdTarDemand** **-** a two dimensional array storing the measured time-dependent O-D information**.**

double tdLinktime** **-** a two dimensional array storing the measured time-dependent link travel times.

Constructor:

void TNM_LODE_TD(at, ep, t) **-** default constructor.

Return - none

ENUM at **-** the type of solution algorithm.

bool ep **-** whether or not enumerate all paths.

double t **-** dispersion parameter.

CLASS CSTAP_B PARENT CLASS: TNM_LODE

Constructor:

void CSTAP_B(at, mt, ep, t) - default constructor.none

ENUM at **-** the type of solution algorithm (only applies to the time-dependent case)**.**

ENUM mt **-** the type of sub problem.

bool ep **-** whether or not enumerate all paths.

double t **-** dispersion parameter.

CLASS CSTAP_B_TD PARENT CLASS CSTAP_B

Constructor:

void CSTAP_B_TD(at, ep, t)- default constructor

ENUM at - the type of solution algorithm.

bool ep - whether or not enumerate all paths.

double t - dispersion parameter.

4.3 Class Library and Its Usage

All the twenty one classes can be complied as a static class library (mat.lib) to facilitate various applications. In this section, we first discuss how to make use of the class library at a programming level, then introduce two primitive user interfaces developed based on the library.

4.3.1 Use mat.lib in programming

With the class library prepared, it is easy to write a piece of code to solve either a logit assignment or an O-D estimation problem. Figure 4.2 provides an example for solving a steady-state LPFE.

First, the main interface mat.h has to be included to get access to class definitions, as shown in the first line of Figure 4.2. We also include the interface tnm.h for accessing another class library `TNM`, for some of constants and types defined in `TNM` may also be used. To solve the steady-state LPFE, we first define and create a corresponding algorithm object `TNM_LODE`. Note that four parameters are required to construct a `TNM_LODE` (See the class description for the meaning of these parameters). Lines 6 - 8 specify optional parameters, such as the type of link performance function (line 6). Default values would be used if no value is provided here. Then, the Build function is called to read files from disk files and allocate memory to handle data. This Build function requires various input files, depending on the problem type and the network file format (the details of the input files will be discussed in Appendix A). Once the algorithm is built successfully, the Solve() function is called to find the optimal solution, followed by three report functions which together prepare a solution report comprising of several ASCII files. Finally, the algorithm object is deleted in order to release memory.

The code shown in Figure 4.2 can actually be employed to solve almost all problems discussed in Chapter 3, provided line 5 is modified to fit in an appropriate object. Table 1 shows a guideline for the selection of objects. Clearly, the object-oriented design provides a neat programming interface because all complexities have been encapsulated within objects. This feature is particularly useful when developing user interfaces. In the following, we develop the interface for the console application..

Table 4.1 A guideline for the selection of algorithm objects

	Logit traffic assignment	Logit path flow estimator
Steady-state	CSTAP_B	TNM_LODE
Time-dependent	CSTAP_B_TD	TNM_LODE_TD

```
#include <mat.h>
#include <tnm.h>
int main()
{
   TNM_LODE *lode = new TNM_LODE(LOGIBA, true, false,0.1);
   lode->lpf = CSTLK;
   lode->EnableTarget();
   lode->EnableTargetError();
   if(lode->Build("c:\\in",  "c:\\out", NETFORT)!=0)
   {
        cout<<"\tFail to build an an algorithm object "<<endl;
        return 1;
   }
   lode->Solve();
   lode->PrepareReport();
   lode->Report();
   lode->CloseReport();
   delete lode;
   return 0;
}
```

Figure 4.2 A sample C++ code using mat.lib

4.3.2 A console interface

The console program named LPFE is originally designed to provide the general purpose PFE-GUI an external access to classes defined in mat.lib. As mentioned, the general purpose PFE-GUI is coded using Visual Basic (VB). Although it simplifies the GIS-related programming, the use of VB makes it a bit more difficult to directly communicate with mat.lib. The console interface bridges the gap. That is, PFE-GUI first offers a user-friendly way to prepare input files for the console program, then executes it to solve specified problems and finally visually displays outputs.

Although it is not convenient, the console program LPFE can also be independently used for testing purpose. The usage of LPFE is briefly summarized in the following.

- **LPFE ?, LPFE h or LPFE H**

Print help message. This command will also produce a file lpfe.hlp that explains how to set up parameters in lpfe.par.

- **LPFE n or LPFE N**

Create a default lpfe.par file. LPFE.par is a required input file that specifies important control parameters such as input and output file names, the dispersion factor, etc. It has to be placed in the same folder as LPFE.exe. Users need to specify the parameters in this file according to their own needs. Missing the file will lead to running error. Figure 4.3 shows an example lpfe.par file.

- **LPFE a or LPFE A**

Create a default algorithm parameter file (.alg). This file include parameters that affect algorithmic performance. Its details will be explained in Appendix A.

- **LPFE ASN or LPFE asn**

Perform logit traffic assignment. A hypothetical O-D estimation scenario can be produced based on assignment results. Specifically, the link flows obtained from assignment will be used to produce ``observed'' link traffic counts required for estimation problems.

- **LPFE EST, LPFE est or LPFE** Perform logit O-D estimation.

```
Input File Name: .\EXAMPLE\est\danet
Input file format: NETDANET2
Output File Name: .\EXAMPLE\est\danet
Retain iteration history: 1
Record link detail: 1
Record path detail: 1
Record path topology: 1
Speed flow function: BPRLK
dispersion parameter: 0.1
cost scalar: 1
algorithme type: IBA
Time dependent: 1
Enumerate all paths: 0
Allow residual queue: 1
Create an estimation scenario: 1
Consider target O-D demands: 1
Consider link observation errors: 1
Consider target O-D errors: 1
```

Figure 4.3 A example lpfe.par file

Chapter 5. Visual PFE – TD: A GIS-Based Decision Support System for Time-Dependent Origin-Destination Trip Table Estimation

5.1 What is Visual PFE-TD?

Visual PFE - TD is a special edition of Visual PFE, an integrated software suite that combines the Path Flow Estimator (PFE) with other software components to facilitate the estimation, visualization, and refinement of Origin-Destination (OD) trip tables with user-friendly Graphical User Interfaces (GUI).

We have already discussed the detailed algorithmic implementations of the Logit Path Flow Estimator (LPFE) and its console interface in Chapter 4, which operates under the DOS command prompt with no GUIs and no graphical presentation of the output data. To expand the capability of LPFE, this special edition of Visual PFE is created to integrate LPFE with components of Visual PFE.

With Visual PFE - TD, users can:

- Run LPFE with GUIs
- Convert the estimated OD tables to Microsoft Excel Files
- Change the colors and zoom levels of the OD table cells
- Interactively display and query OD desire lines
- Interactively display and query paths between any pair of OD
- Convert the PFE outputs to GIS files (ESRI shapefiles)
- Create thematic maps of network links and traffic analysis zones
- Generate diagnostic scatter plots
- Link the scatter plots to the network for identification of outliers
- Create and edit LPFE networks as text files

- Compare different scenarios

5.2 Key Features

Visual PFE - TD is developed with Microsoft Visual Basic.NET. It implements Microsoft's Multiple Document Interface (MDI) software architecture, within which multiple windows forms can be created and viewed within the main window. The main window is referred to as the MDI parent form and each sub-window opened within the MDI parent is called a MDI child form. With Visual PFE, a MDI child form can take on the form of a map, a table, a graphic plot, or a text document. The program functions are controlled using Graphical User Interfaces (GUI). When multiple result windows are opened in Visual PFE - TD, the titles of the MDI child windows are formatted in a way that results of the same network and time period can be identified. Figure 5.1 shows such an example, in which the titles of the OD table, scatter plot, path table, and report all include the name and the file path of the network (i.e., the title of the fist window).

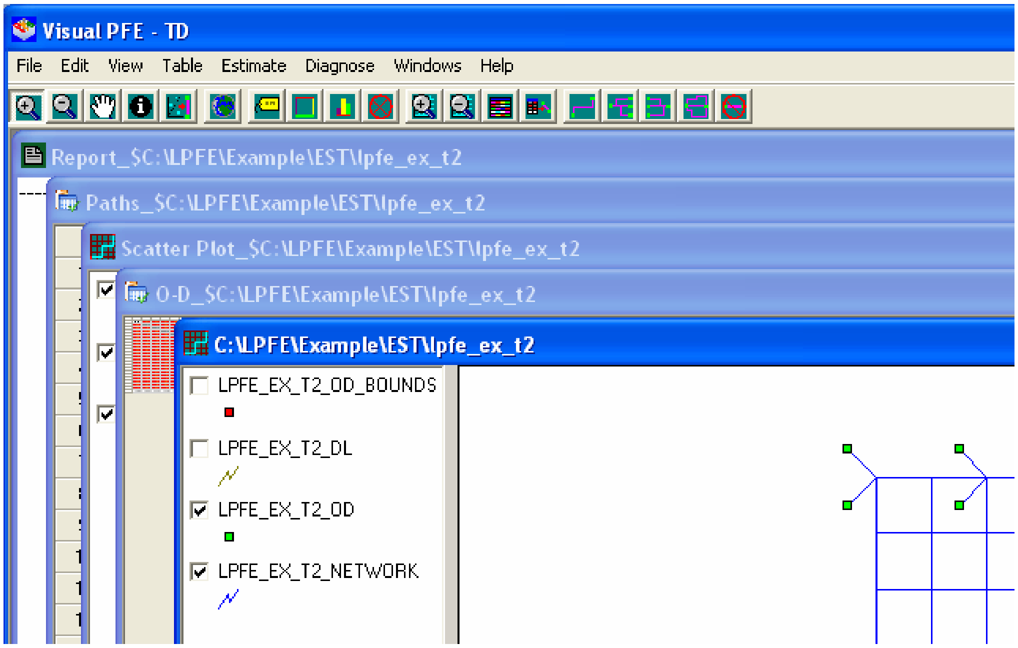
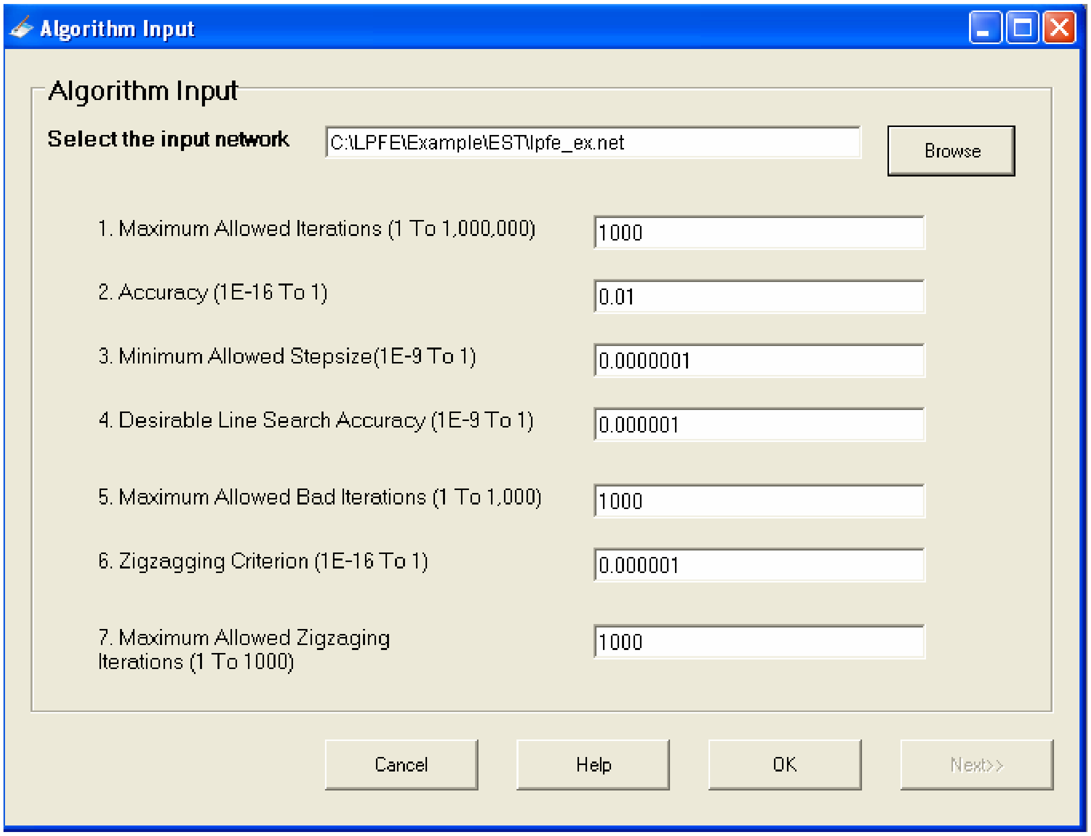

Visual PFE - TD
File
Edit
View
Table
Estimate
Diagnose
Windows
Help
Report_$C:\LPFE\Example\EST\lpfe_ex_t2
Paths_$C:\LPFE\Example\EST\lpfe_ex_t2
Scatter Plot_$C:\LPFE\Example\EST\lpfe_ex_t2
O-D_$C:\LPFE\Example\EST\lpfe_ex_t2
C:\LPFE\Example\EST\lpfe_ex_t2
☐ LPFE_EX_T2_OD_BOUNDS
☐ LPFE_EX_T2_DL
☑ LPFE_EX_T2_OD
☑ LPFE_EX_T2_NETWORK

Figure 5.1 Multiple Document Interface

5.2.1 Estimation GUI

The process of OD estimation with Visual PFE - TD begins with the Estimation GUI (see Figures 5.2 and 5.3). Users can use this GUI to set up a LPFE estimation, provided LPFE network data have been prepared. This GUI feeds the user inputs to the LPFE and the program returns the estimation results to the GUI once estimation is completed or specific problems are encountered during estimation. The original output files generated by the LPFE estimation program are all text-based. Visual PFE contains components that can convert the text files to the GIS and spreadsheet files.



Select the input network C:\LPFE\Example\EST\lpfe_ex.net

1. Maximum Allowed Iterations (1 To 1,000,000) 1000

2. Accuracy (1E-16 To 1) 0.01

3. Minimum Allowed Stepsize(1E-9 To 1) 0.0000001

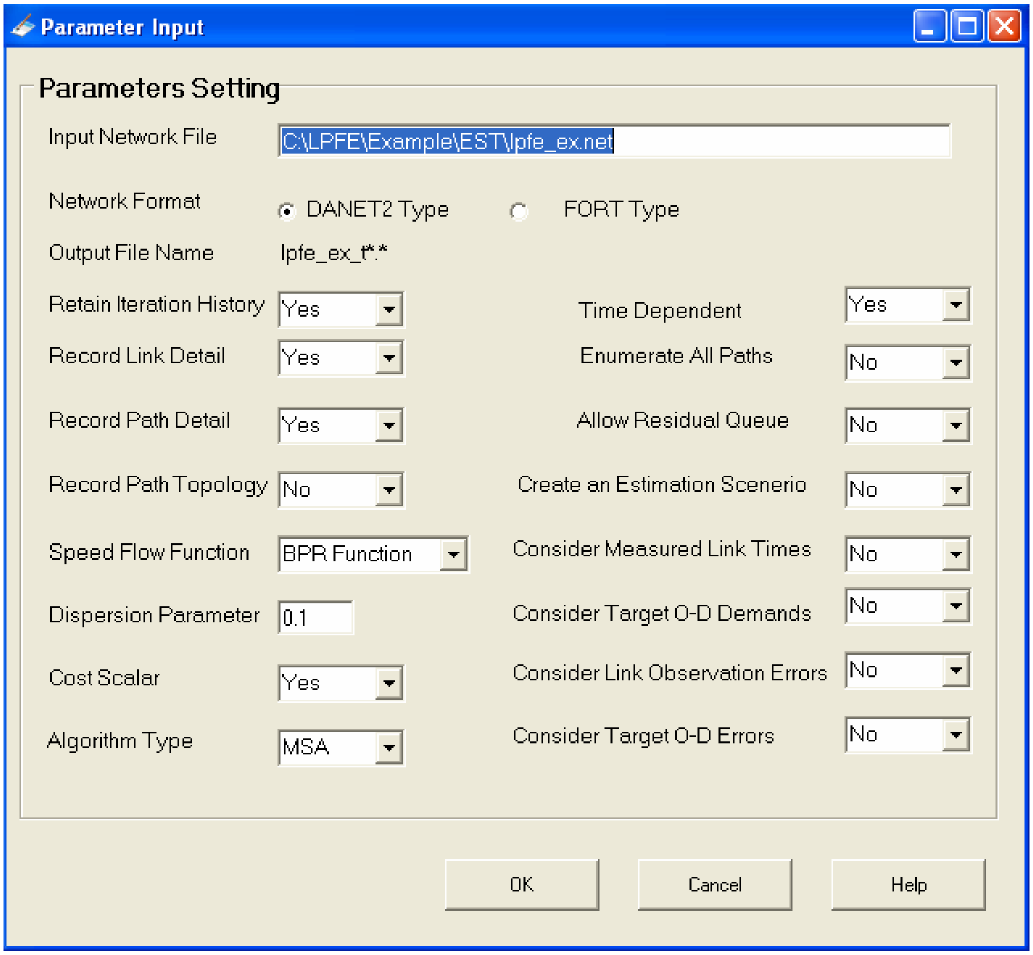
4. Desirable Line Search Accuracy (1E-9 To 1) 0.000001

5. Maximum Allowed Bad Iterations (1 To 1,000) 1000

6. Zigzagging Criterion (1E-16 To 1) 0.000001

7. Maximum Allowed Zigzaging Iterations (1 To 1000) 1000

Figure 5.2 Algorithm Input Window



Parameters Setting

Input Network File C:\LPFE\Example\EST\lpfe_ex.net

Network Format ● DANET2 Type ○ FORT Type

Output File Name lpfe_ex_t*.*

Retain Iteration History	Yes	Time Dependent	Yes
Record Link Detail	Yes	Enumerate All Paths	No
Record Path Detail	Yes	Allow Residual Queue	No
Record Path Topology	No	Create an Estimation Scenerio	No
Speed Flow Function	BPR Function	Consider Measured Link Times	No
Dispersion Parameter	0.1	Consider Target O-D Demands	No
Cost Scalar	Yes	Consider Link Observation Errors	No
Algorithm Type	MSA	Consider Target O-D Errors	No

Figure 5.3 Parameter Input Window

Once successfully estimated, Visual PFE-TD can present the outputs to the users with four types of windows forms:

- Network Maps
- OD and Path Tables
- Link Flow Scatter plots
- Estimation Reports

5.2.2 Network Maps

The Network Windows form is divided into the map and the legend areas (Figure 5.4). A layer can be made active by moving mouse cursor over and clicking at the corresponding title of the map legend. Labels, colors, and charts based on attributes can be applied to features of the active layer. Drawing order of the layers can also be changed by dragging the layer name up or down in the legend area.

A network map consists of four ESRI Shapefiles, each presented as a map layer:

- Network links: All of the link attributes are stored in this layer, including the observed and estimated link flows.
- Origins and destinations: The production and attraction of the origins and destinations are stored in this point layer.
- Desire lines: A desire line represents the flow between a pair of OD. The desire lines layer is hidden by default. Users can choose to display or hide them.
- Map boundary points: Two boundary points are included in the network map to create sufficient margins around the network such that all features of the network can be displayed within the map window.

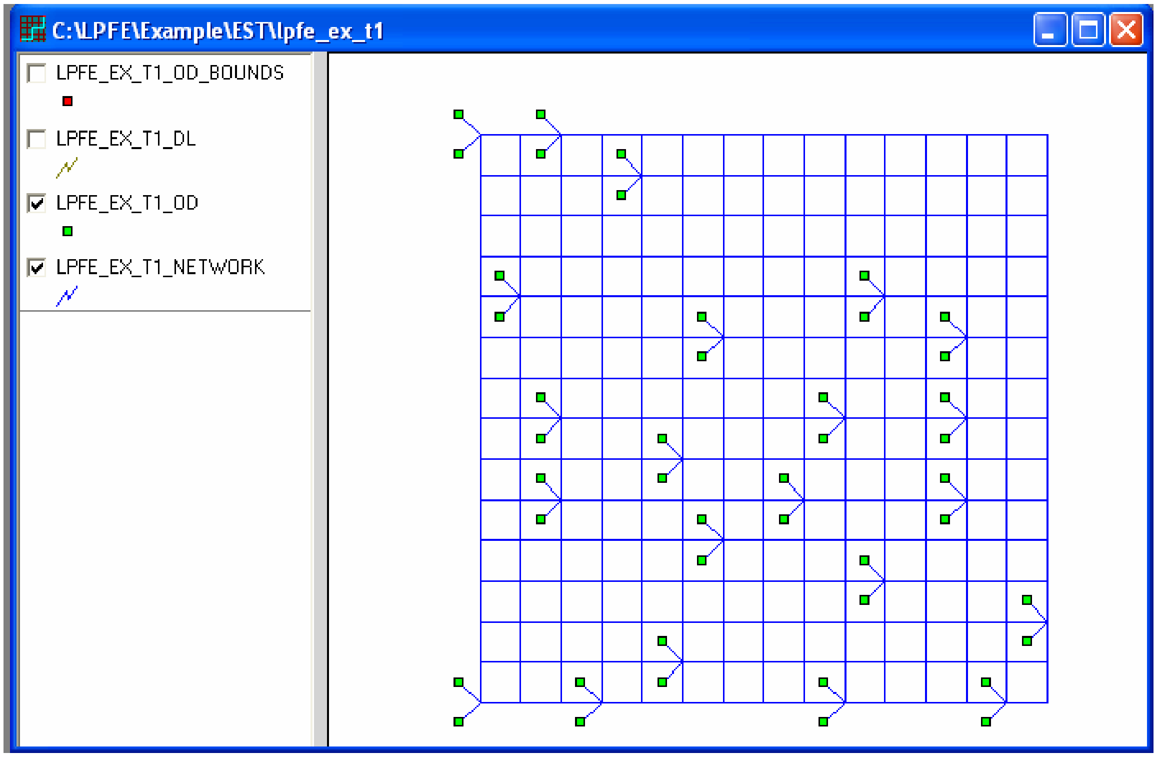
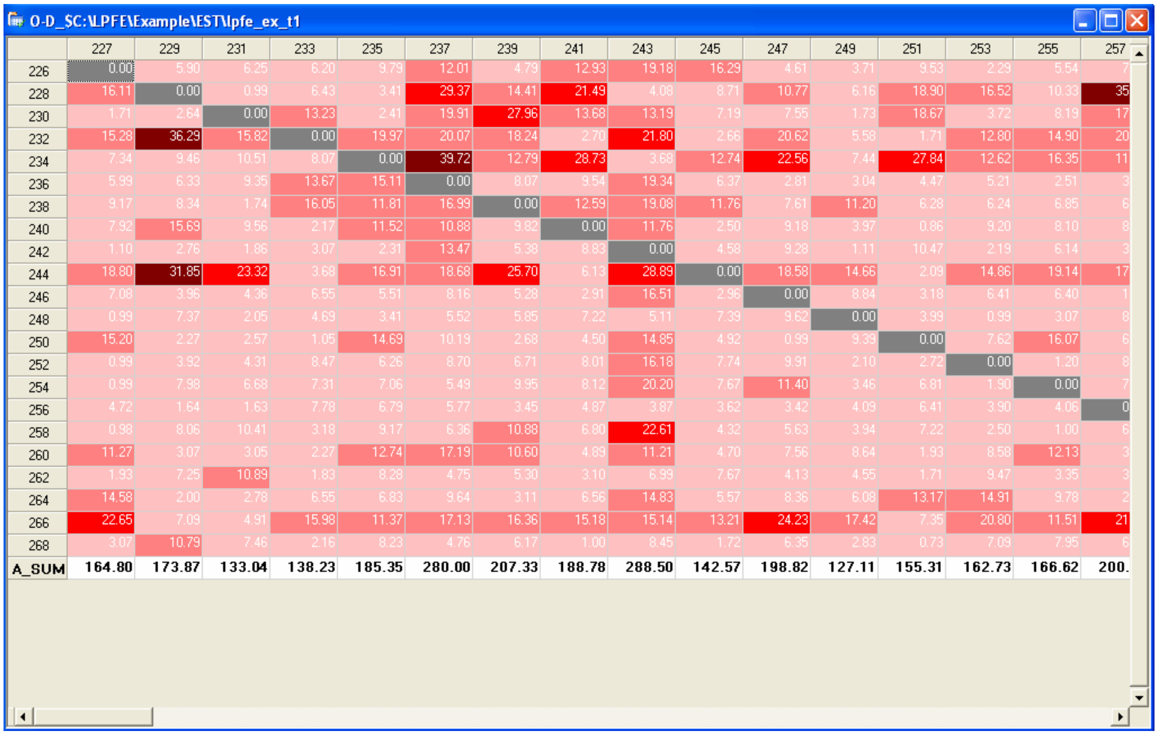


Figure 5.4 Network Map

5.2.3 OD and Path Tables

The OD table (see Figure 5.5) is presented as a spreadsheet. The color of each cell can be adjusted based on the value of the OD flow. The size of the cells can also be adjusted to facilitate visualization of the flow patterns over the entire OD table.



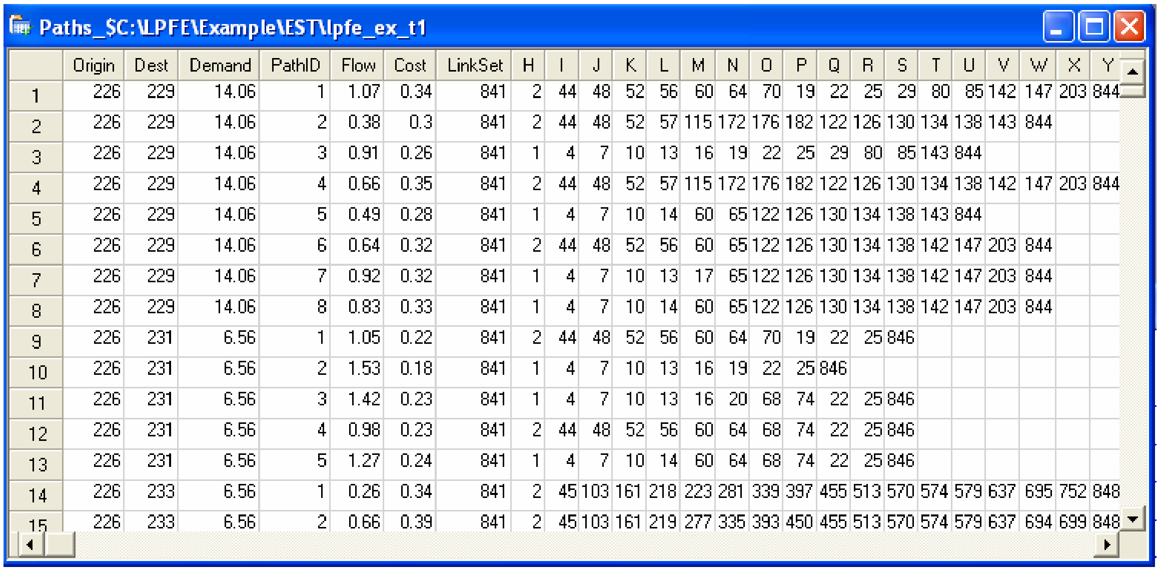
O-D_$C:\LPFE\Example\EST\lpfe_ex_t1

	227	229	231	233	235	237	239	241	243	245	247	249	251	253	255	257
226	0.00	5.90	6.25	6.20	9.79	12.01	4.79	12.93	19.18	16.29	4.61	3.71	9.53	2.29	5.54	7
228	16.11	0.00	0.99	6.43	3.41	29.37	14.41	21.49	4.08	8.71	10.77	6.16	18.90	16.52	10.33	35
230	1.71	2.64	0.00	13.23	2.41	19.91	27.96	13.68	13.19	7.19	7.55	1.73	18.67	3.72	8.19	17
232	15.28	36.29	15.82	0.00	19.97	20.07	18.24	2.70	21.80	2.66	20.62	5.58	1.71	12.80	14.90	20
234	7.34	9.46	10.51	8.07	0.00	39.72	12.79	28.73	3.68	12.74	22.56	7.44	27.84	12.62	16.35	11
236	5.99	6.33	9.35	13.67	15.11	0.00	8.07	9.54	19.34	6.37	2.81	3.04	4.47	5.21	2.51	3
238	9.17	8.34	1.74	16.05	11.81	16.99	0.00	12.59	19.08	11.76	7.61	11.20	6.28	6.24	6.85	6
240	7.92	15.69	9.56	2.17	11.52	10.88	9.82	0.00	11.76	2.50	9.18	3.97	0.86	9.20	8.10	8
242	1.10	2.76	1.86	3.07	2.31	13.47	5.38	8.83	0.00	4.58	9.28	1.11	10.47	2.19	6.14	3
244	18.80	31.85	23.32	3.68	16.91	18.68	25.70	6.13	28.89	0.00	18.58	14.66	2.09	14.86	19.14	17
246	7.08	3.96	4.36	6.55	5.51	8.16	5.28	2.91	16.51	2.96	0.00	8.84	3.18	6.41	6.40	1
248	0.99	7.37	2.05	4.69	3.41	5.52	5.85	7.22	5.11	7.39	9.62	0.00	3.99	0.99	3.07	8
250	15.20	2.27	2.57	1.05	14.69	10.19	2.68	4.50	14.85	4.92	0.99	9.39	0.00	7.62	16.07	6
252	0.99	3.92	4.31	8.47	6.26	8.70	6.71	8.01	16.18	7.74	9.91	2.10	2.72	0.00	1.20	8
254	0.99	7.98	6.68	7.31	7.06	5.49	9.95	8.12	20.20	7.67	11.40	3.46	6.81	1.90	0.00	7
256	4.72	1.64	1.63	7.78	6.79	5.77	3.45	4.87	3.87	3.62	3.42	4.09	6.41	3.90	4.06	0
258	0.98	8.06	10.41	3.18	9.17	6.36	10.88	6.80	22.61	4.32	5.63	3.94	7.22	2.50	1.00	6
260	11.27	3.07	3.05	2.27	12.74	17.19	10.60	4.89	11.21	4.70	7.56	8.64	1.93	8.58	12.13	3
262	1.93	7.25	10.89	1.83	8.28	4.75	5.30	3.10	6.99	7.67	4.13	4.55	1.71	9.47	3.35	3
264	14.58	2.00	2.78	6.55	6.83	9.64	3.11	6.56	14.83	5.57	8.36	6.08	13.17	14.91	9.78	2
266	22.65	7.09	4.91	15.98	11.37	17.13	16.36	15.18	15.14	13.21	24.23	17.42	7.35	20.80	11.51	21
268	3.07	10.79	7.46	2.16	8.23	4.76	6.17	1.00	8.45	1.72	6.35	2.83	0.73	7.09	7.95	6
A_SUM	164.80	173.87	133.04	138.23	185.35	280.00	207.33	188.78	288.50	142.57	198.82	127.11	155.31	162.73	166.62	200.

Figure 5.5 OD Table Window

For each cell in the OD table, the locations of the origin and destination and the magnitude of the OD flow can be graphically presented over the network map by initiating a dynamic linkage between the OD table and the desire line layer. That is, by highlighting a particular cell in the OD table, the desire line of that OD pair can be plotted over the network map with a band that varies by the magnitude of the OD flow (see Figure 5.5).

The path table is also implemented as a spreadsheet (Figure 5.6). A dynamic linkage exists between a path table and the corresponding network layer. The location of a path can be identified and highlighted over the network map by selecting a path in the Path table and initiating the dynamic linkage to the map.


Paths_$C:\LPFE\Example\EST\lpfe_ex_t1

	Origin	Dest	Demand	PathID	Flow	Cost	LinkSet	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	226	229	14.06	1	1.07	0.34	841	2	44	48	52	56	60	64	70	19	22	25	29	80	85	142	147	203	844
2	226	229	14.06	2	0.38	0.3	841	2	44	48	52	57	115	172	176	182	122	126	130	134	138	143	844		
3	226	229	14.06	3	0.91	0.26	841	1	4	7	10	13	16	19	22	25	29	80	85	143	844				
4	226	229	14.06	4	0.66	0.35	841	2	44	48	52	57	115	172	176	182	122	126	130	134	138	142	147	203	844
5	226	229	14.06	5	0.49	0.28	841	1	4	7	10	14	60	65	122	126	130	134	138	143	844				
6	226	229	14.06	6	0.64	0.32	841	2	44	48	52	56	60	65	122	126	130	134	138	142	147	203	844		
7	226	229	14.06	7	0.92	0.32	841	1	4	7	10	13	17	65	122	126	130	134	138	142	147	203	844		
8	226	229	14.06	8	0.83	0.33	841	1	4	7	10	14	60	65	122	126	130	134	138	142	147	203	844		
9	226	231	6.56	1	1.05	0.22	841	2	44	48	52	56	60	64	70	19	22	25	846						
10	226	231	6.56	2	1.53	0.18	841	1	4	7	10	13	16	19	22	25	846								
11	226	231	6.56	3	1.42	0.23	841	1	4	7	10	13	16	20	68	74	22	25	846						
12	226	231	6.56	4	0.98	0.23	841	2	44	48	52	56	60	64	68	74	22	25	846						
13	226	231	6.56	5	1.27	0.24	841	1	4	7	10	14	60	64	68	74	22	25	846						
14	226	233	6.56	1	0.26	0.34	841	2	45	103	161	218	223	281	339	397	455	513	570	574	579	637	695	752	848
15	226	233	6.56	2	0.66	0.39	841	2	45	103	161	219	277	335	393	450	455	513	570	574	579	637	694	699	848

Figure 5.6 Path Table Window

5.2.4 Link Flow Scatter Plots

One of the most common and important diagnostic graphs for the OD estimation problem is the observed-versus-estimated scatter plot of network link flows (see Figure 5.7). The scatter plot is implemented in Visual PFE-TD as a collection of map layers. The scatter plot point layer uses the observed value of a link as the X coordinate and the estimated value as the Y coordinate. A link layer is used to represent the two axes and other boundary lines delineating the levels of accuracy of the estimation.

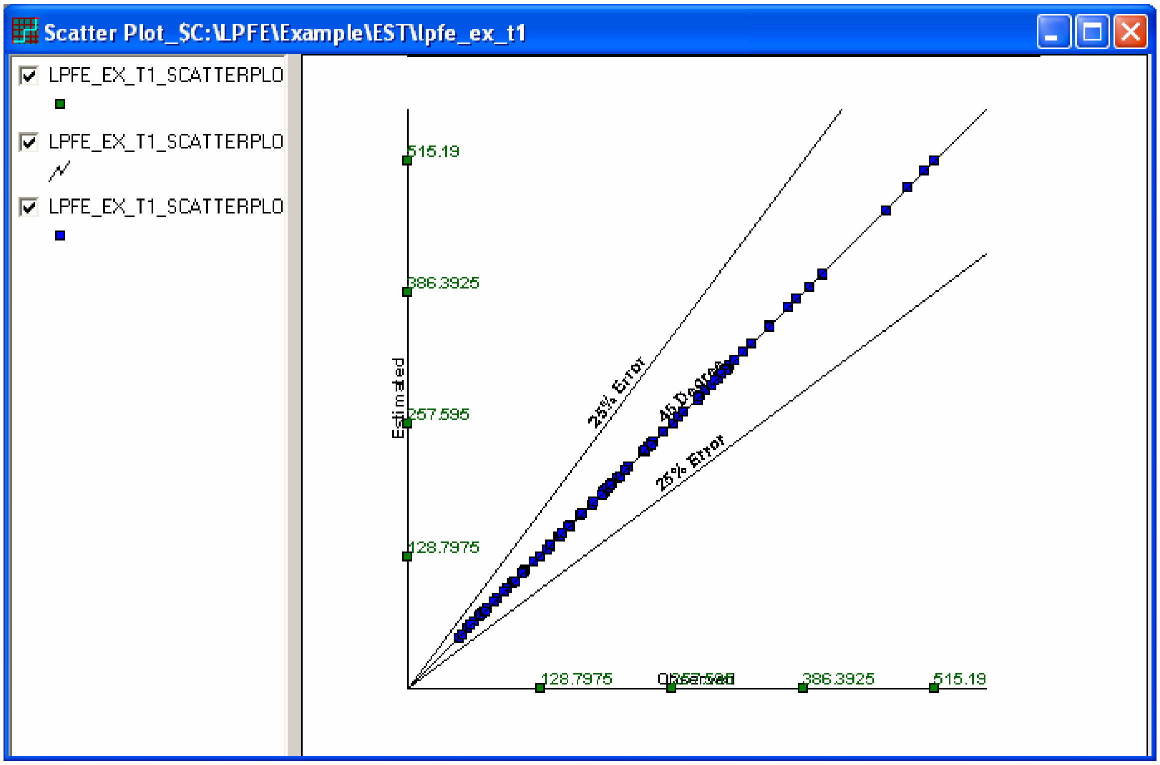
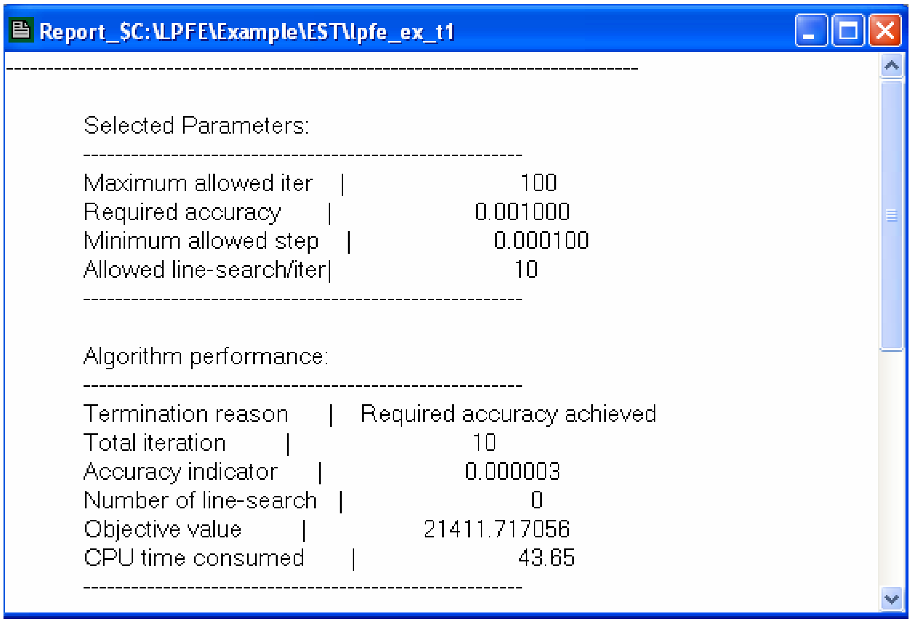


Figure 5.7 Scatter Plot Window

The scatter plot layer can be linked to the network map to identify the link location of a particular scatter plot point.

5.2.5 Report Document

The LPFE generates a summary report for every successful estimation. The estimation report can be viewed via the report document windows form (Figure 5.8). The document windows is also a text editor with all the regular text editing functions such as cut, copy, and paste.



```
--------------------------------------------------------------------------------

          Selected Parameters:
          --------------------------------------------------------
          Maximum allowed iter    |                    100
          Required accuracy       |               0.001000
          Minimum allowed step    |                 0.000100
          Allowed line-search/iter|                     10
          --------------------------------------------------------

          Algorithm performance:
          --------------------------------------------------------
          Termination reason      |   Required accuracy achieved
          Total iteration         |                    10
          Accuracy indicator      |               0.000003
          Number of line-search   |                      0
          Objective value         |           21411.717056
          CPU time consumed       |                  43.65
          --------------------------------------------------------
```

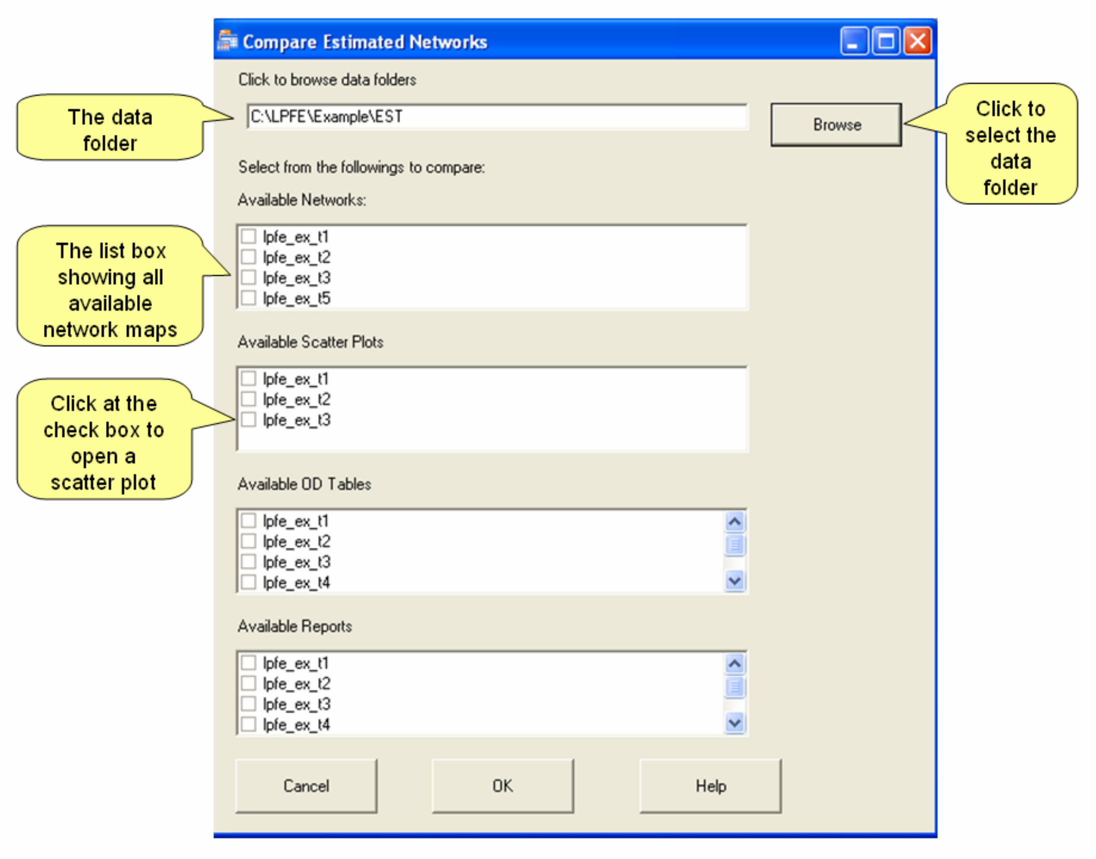
Figure 5.8 Report Window

5.2.6 Network Editing

After examining the results of a LPFE estimation, users can change attributes (e.g., the observed flows) of a network link and export the data back to the original LPFE data format for another estimation with improved results.

5.2.7 Scenario Comparison

If multiple estimation runs have been done for a network, all of the previous results can be opened and compared at the same time. The comparison is setup via the GUI in Figure 5.9.



Compare Estimated Networks

Click to browse data folders

C:\LPFE\Example\EST

Select from the followings to compare:

Available Networks:

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t5

Available Scatter Plots

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3

Available OD Tables

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t4

Available Reports

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t4

The data folder

The list box showing all available network maps

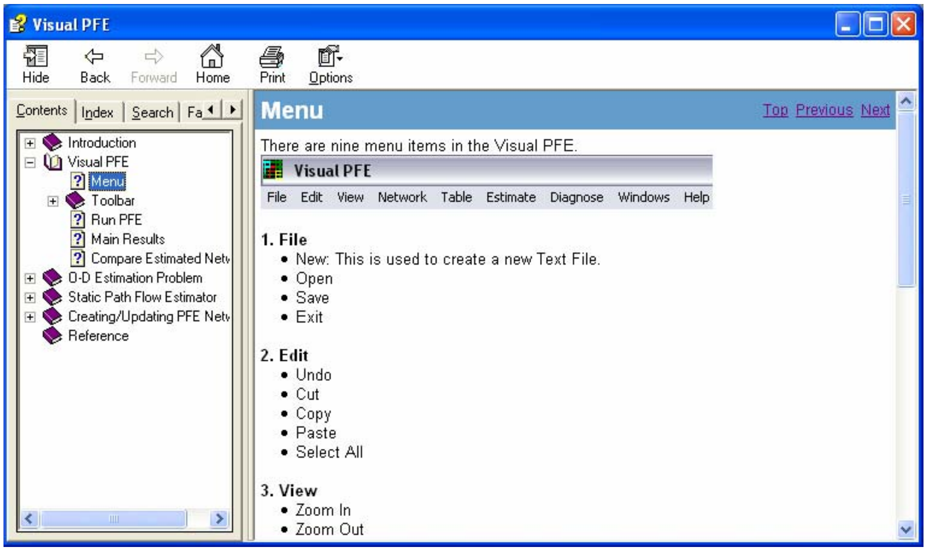
Click at the check box to open a scatter plot

Click to select the data folder

Figure 5.9 Scenario Comparison Window

5.2.8 Windows Help

A standard windows help document (Figure 5.10) is created and built in with Visual PFE-TD. Users can access instruction of particular tasks via the document.



Menu

There are nine menu items in the Visual PFE.

1. File
   - New: This is used to create a new Text File.
   - Open
   - Save
   - Exit

2. Edit
   - Undo
   - Cut
   - Copy
   - Paste
   - Select All

3. View
   - Zoom In
   - Zoom Out

Figure 5.10 Windows Help

5.3 Technical Specification

Visual PFE - TD integrates the LPFE program with three other software components. The architecture of the system is shown in Figure 5.11.

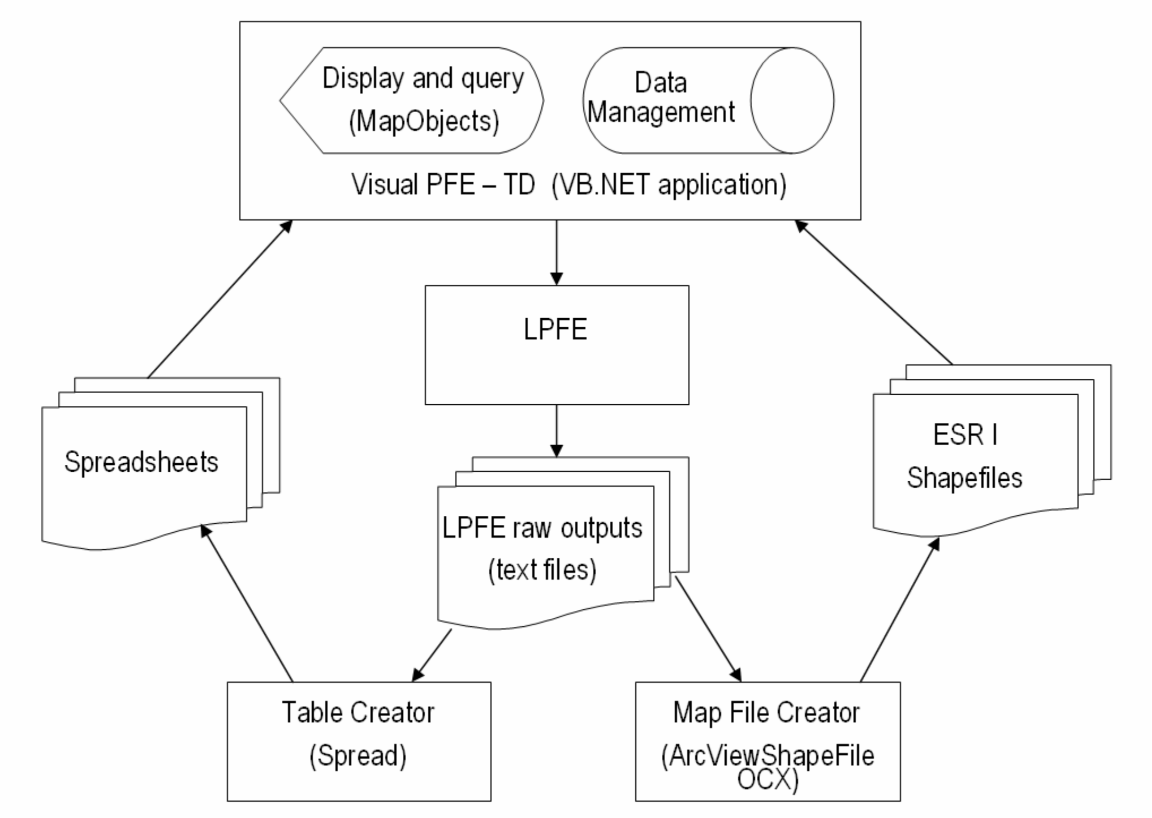


Figure 5.11 Visual PFE - TD Software Components

The three software components are:

- ArcViewShapeFile OCX
- ESRI MapObjects
- FarPoint Spread

5.3.1 LPFE

The LPFE program operates via the DOS command prompt. If an estimation run is successfully, the output files generated by the LPFE are in text files. If an estimation run encounters errors due to incompatible parameter setting, the LPFE program will quit and error messages will be issued by Visual PFE – TD.

5.3.2 The ArcViewShapefile Read/Write OCX

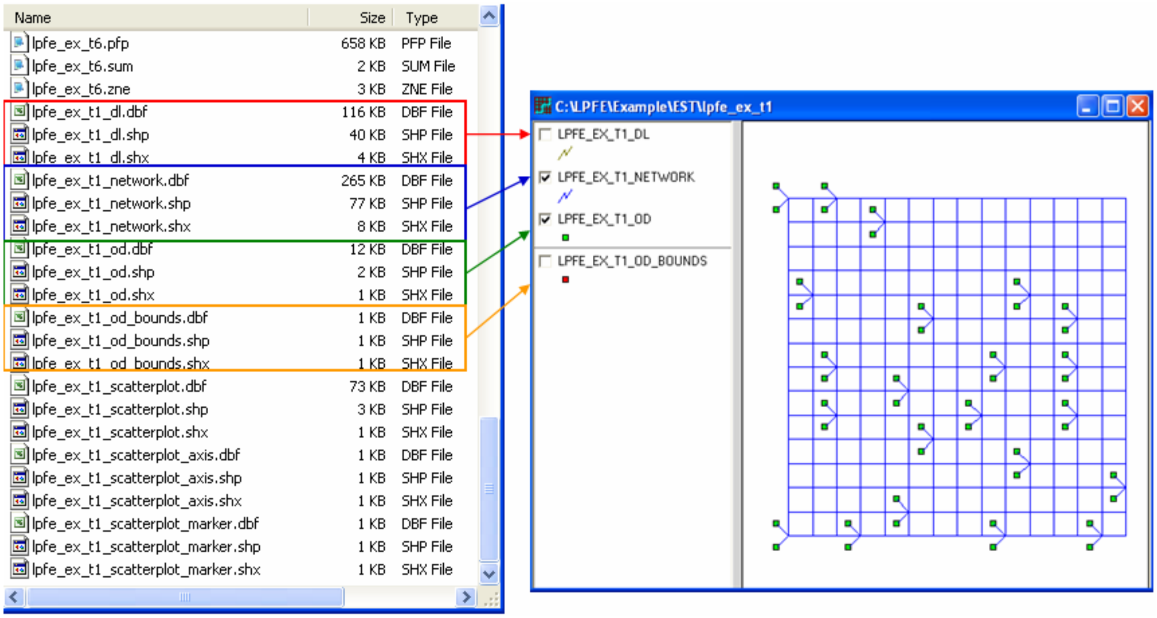
An OCX is an object-oriented software component conforming to Microsoft's ActiveX architecture. The OCX was created to easily read or write Arcview Shapefiles for data conversion purposes. The Shapefile format is a geographic data format published by ESRI. The ArcViewShapefile Read/Write OCX was created by Ross Pickard in Wellington, New Zealand and is a shareware made available for download from ESRI web site.

Visual PFE - TD uses the DLL version of the ArcViewShapeFile to convert the text files generated by LPFE to Shapefiles. The converted shapefiles can be displayed with the Visual PFE - TD map component and common GIS software. Table 5.1 shows the conversion between the LPFE text files to the shapefiles used by Visual PFE.

Table 5.1 LPFE Output Text Files to Shapefiles

LPFE Output File	Converted Shapefile
*.lfp	Network lines
*.zne	OD points
*.est	Desire lines
*.lfp	Scatter plot points

Visual PFE – TD converts the LPFE output text files to shapefiles. These shapefiles are stored in the same folder as the input network data. The shapefiles are used to visually depict the network, the origins, the destinations, and the desire lines. Figure 5.12 shows a side-by-side view of the list of shapefiles and the legend of the network map layers. It can be seen that three files (i.e., *.shp, *.shx, and *.dbf) are used to draw a layer (i.e., ESRI shapefile specification). The name of the layer is the same as that of the three shapefiles that make up the layer. Each layer name includes reference to the input network and the feature (i.e., network, od, desire lines, or boundary points) of the layer.



Name	Size	Type
lpfe_ex_t6.pfp	658 KB	PFP File
lpfe_ex_t6.sum	2 KB	SUM File
lpfe_ex_t6.zne	3 KB	ZNE File
lpfe_ex_t1_dl.dbf	116 KB	DBF File
lpfe_ex_t1_dl.shp	40 KB	SHP File
lpfe_ex_t1_dl.shx	4 KB	SHX File
lpfe_ex_t1_network.dbf	265 KB	DBF File
lpfe_ex_t1_network.shp	77 KB	SHP File
lpfe_ex_t1_network.shx	8 KB	SHX File
lpfe_ex_t1_od.dbf	12 KB	DBF File
lpfe_ex_t1_od.shp	2 KB	SHP File
lpfe_ex_t1_od.shx	1 KB	SHX File
lpfe_ex_t1_od_bounds.dbf	1 KB	DBF File
lpfe_ex_t1_od_bounds.shp	1 KB	SHP File
lpfe_ex_t1_od_bounds.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot.dbf	73 KB	DBF File
lpfe_ex_t1_scatterplot.shp	3 KB	SHP File
lpfe_ex_t1_scatterplot.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_axis.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_axis.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_axis.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_marker.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_marker.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_marker.shx	1 KB	SHX File

C:\LPFE\Example\EST\lpfe_ex_t1

□ LPFE_EX_T1_DL

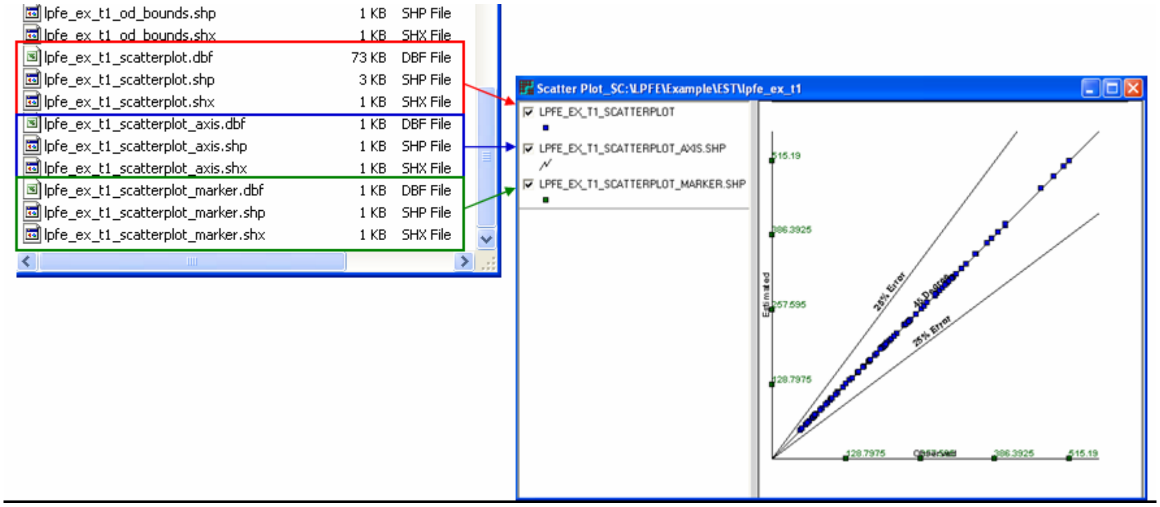
☑ LPFE_EX_T1_NETWORK

☑ LPFE_EX_T1_OD

□ LPFE_EX_T1_OD_BOUNDS

Figure 5.12 Shapefiles and Network Map Layers

Figure 5.13 shows a side-by-side view of the list of shapefiles and legend of the Scatter Plot layers.



Name	Size	Type
lpfe_ex_t1_od_bounds.shp	1 KB	SHP File
lpfe_ex_t1_od_bounds.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot.dbf	73 KB	DBF File
lpfe_ex_t1_scatterplot.shp	3 KB	SHP File
lpfe_ex_t1_scatterplot.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_axis.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_axis.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_axis.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_marker.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_marker.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_marker.shx	1 KB	SHX File

Scatter Plot_SC:\LPFE\Example\EST\lpfe_ex_t1

☑ LPFE_EX_T1_SCATTERPLOT

☑ LPFE_EX_T1_SCATTERPLOT_AXIS.SHP

☑ LPFE_EX_T1_SCATTERPLOT_MARKER.SHP

Figure 5.13 Shapefiles and the Scatter Plot Layers

5.3.3 ESRI MapObjects

MapObjects is a set of mapping software components that let users add GIS maps to software applications. MapObjects comprises an ActiveX control (OCX) called the Map control and a set of over forty-five ActiveX Automation objects. In Visual PFE, a map control is used to display the shapefiles created by the ArcViewShapeFile component. The map control facilitates all of the GIS functions for the converted shapefiles. Another OCX control is added to a map form to display the legends of different map layers. The legend control is internally connected to the map control. A layer can be made active through the legend control. On the active layer, labels and thematic mapping can be applied to features on the layer. For example, networks links can be varied by colors and width based on estimated link flow. Bar charts can be applied to display the varying magnitude of production and attraction at the origins and destinations.

5.3.4 FarPoint Spread

Spread by the FarPoint Technologies is a software component built to Microsoft's latest .NET software architecture. Spread is used to create spreadsheet applications. The strength of Spread is that it facilitates all of the typical spreadsheet functions with a dimension limitation (1 million columns by 1 million rows) that is unlikely to be violated by any OD tables. Visual PFE contains codes that process the PFE CSV outputs and formulate the spreadsheets stored and displayed with Spread. Once a spreadsheet is created in Spread, it can be saved as a Microsoft Excel file. Table 5.2 shows the association between the original PFE CSV outputs and the formatted Spread tables.

Table 5.2 LPFE Output Files to Spread Tables

LPFE Output Files	**Converted Spread Table**
*.est	OD table
*.pfp	Path table

The details of how to use the software are contained in the user manuals in the Appendix.

Chapter 6. Case studies

With the details of LPFE formulation and implementation discussed in previous chapters, this chapter presents a series of case studies to examine the performance characteristics of LPFE. We first describe a general evaluation framework in Section 5.1, followed by numerical experiments that are carefully designed to examine

- how the choice of algorithmic parameters affects performance?
- how different network topologies affect estimation performance?
- how measurement locations and additional inputs (such as partial O-D information) affect LPFE's and TD-LPFE's performance? and
- how input errors influence the quality of O-D estimates?

Note that most numerical studies presented in this chapter are devoted to the static problem, because the time-dependent LPFE is comprised of a sequence of static LPFEs and so findings derived from static studies naturally apply to the dynamic case. However, we shall discuss a dynamic example in the last section that highlights the effect of carryovers of residual queues across times.

6.1 Evaluation Framework

Theoretically, the evaluation of the LPFE's performance requires comparing the estimated O-D flows to the "true" O-D flows. In reality, however, obtaining "true" O-D data is extremely difficult. We use a combination of historical and synthetic O-D data in these cases studies, with an evaluation framework as shown in Figure 6.1.

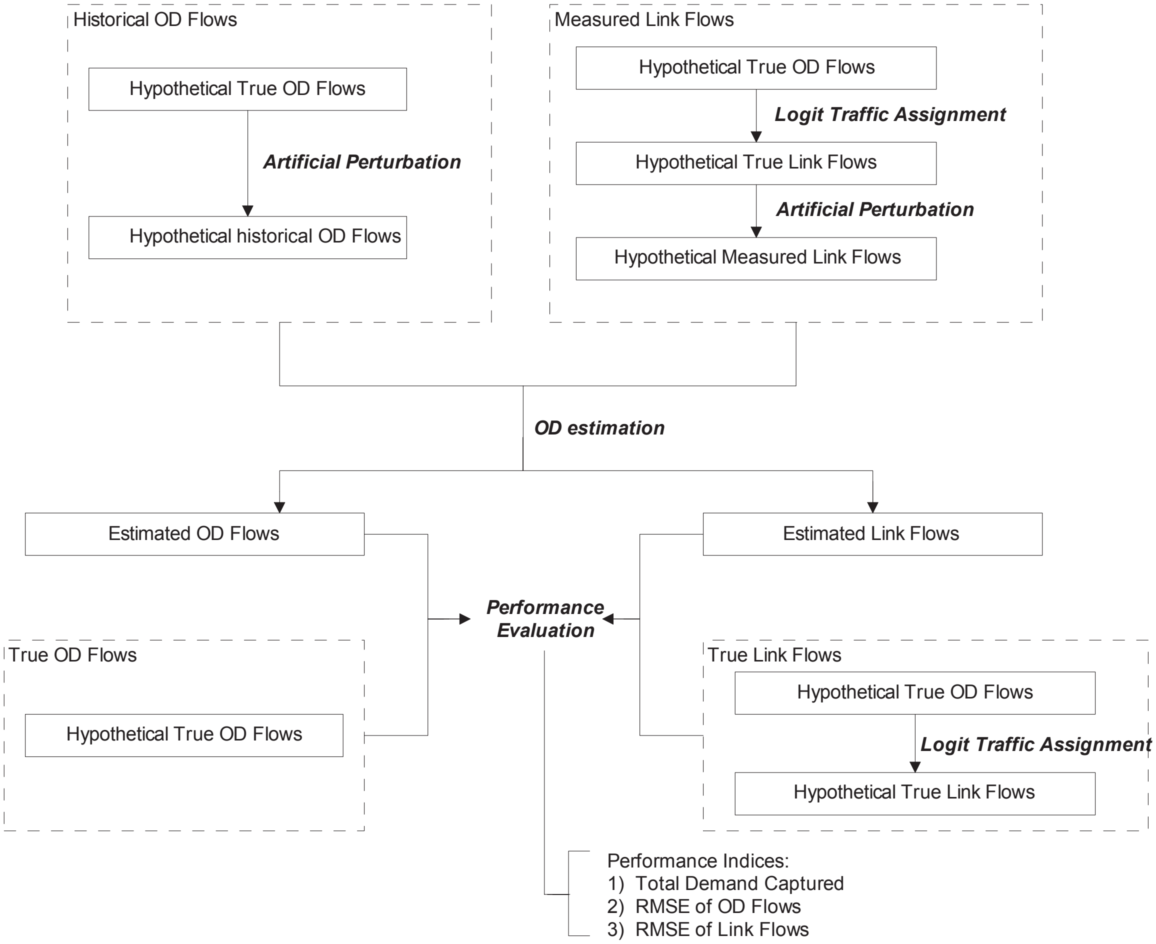


Figure 6.1 Evaluation framework

We assume that a “true” O-D trip table is known as a *priori*. This O-D table can be established, for example, from existing survey data. With this ``true" O-D table, link traffic counts are produced from the corresponding traffic assignment procedure (a logit traffic assignment in our case). Perturbations can be further introduced to emulate measurement errors. Similarly, one can obtain a ``historical" or ``observed" O-D table by applying a perturbation to the predetermined O-D table. Using these inputs we can evaluate the performance of LPFE by comparing the estimated O-D/link flows to those presumed to be known as true.

The measurements of effectiveness (MOE) adopted in our evaluation framework include: 1) TDC (total demand captured); 2) RMSE_OD (root mean square error of estimated OD flows); 3) RMSE_Link (root mean square error of estimated link flows). Mathematically, These MOEs can be expressed as:

$$\text{TDC (Total Demand Captured)} = \frac{\sum_{p=1}^{M} \bar{D}_p}{\sum_{p=1}^{M} D_p} \tag{6.1}$$

$$\text{RMSE\_OD (RMSE of O-D flows)} = \sqrt{\frac{\sum_{p=1}^{M} (\bar{D}_p - D_p)^2}{M}} \tag{6.2}$$

$$\text{RMSE\_Link (RMSE of link flows)} = \sqrt{\frac{\sum_{k=1}^{N} (\bar{X}_k - X_k)^2}{N}} \tag{6.3}$$

where,

$\bar{D}_p, p = 1, \ldots, M$ are the estimated O-D flows;

$D_p, p = 1, \ldots, M$ are the true O-D flows;

$\bar{X}_k, k = 1, \ldots, N$ are the estimated link flows;

$X_k, k = 1, \ldots, M$ are the true link flows.

TDC measures the percentage that the total true O-D flows are captured by estimated O-D flows. Thus, a *TDC* closer to 1 may suggest a better estimate. RMSE_OD depicts the average variation of the estimated O-D flows with respect to the ``true'' values. The smaller the RMSE_OD, the better the spatial structure of travel demands is captured. Similarly, the value of RMSE_Link measures the extent to which measured link flows are reproduced by estimated flows.

Finally, to simplify the analysis, all tested scenarios assume that link travel times are independent of link flows.

6.2 Algorithmic Settings

The performance of LPFE may be substantially affected by several algorithmic parameters. Thus, it is natural to ask what are the ``optimal'' parameters leading to the best performance. To answer this question, we explore the relationship between the parameters and the LPFE performance using numerical experiments. The major parameters in consideration include:

- maximum iteration number n. n restricts the maximum number of iterations for DDAG (the algorithm solving the E-LPFE-SUB problem);
- accuracy ϵ. ε defines the stopping criterion for both DDAG (the algorithm solving the e-LPFE-SUB problem) and CGA(the column generation algorithm). For narrative convenience, we shall denote the convergence indicator of the i th iteration of DDAG as k_i hereafter. Either algorithm will terminate if $k_i \leq \epsilon$;
- maximum bad iterations bn. bn represents the maximum number of iterations in which $k_i > k_{i-1}$;
- zigzagging criterion z and maximum zigzagging iterations zn. z defines the criterion to count an iteration as a zigzagging iteration, and zn restricts the maximum number of zigzagging iterations for DDAG. If $\frac{|k_i - k_{i-1}|}{k_{i-1}} < z$, the iteration will be treated as a zigzagging iteration.

A small network taken from Yang and Meng (Yang & Meng 1998) is adopted to examine the impact of these parameters. The network contains 9 nodes, 14 links, and 9 O-D pairs. Figure 6.2 shows the network topology and Figure 6.3 lists the O-D table. We use all link flows obtained from assignment as inputs. No additional O-D information is considered.

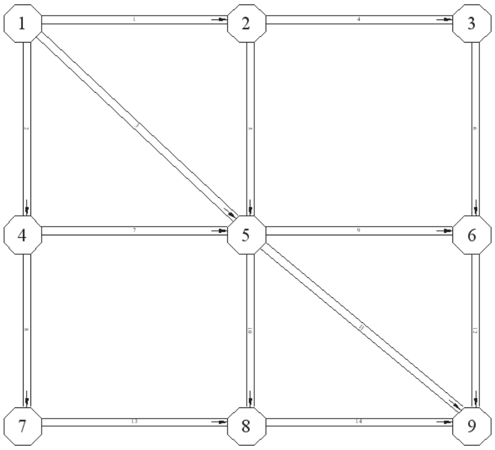
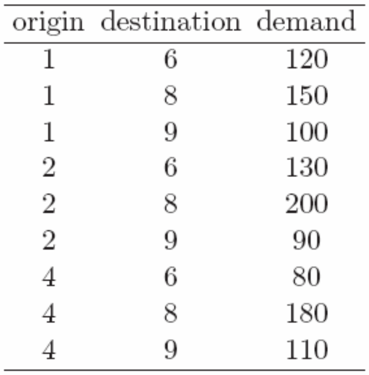


Figure 6.2 Yang and Meng's network



origin	destination	demand
1	6	120
1	8	150
1	9	100
2	6	130
2	8	200
2	9	90
4	6	80
4	8	180
4	9	110

Figure 6.3 O-D trip table

Figure 6.4- 6.7 show the values of RMSE_Link of various scenarios with different algorithmic settings. As shown in Figure 6.4, when the number of iteration is larger than 100, resulted RMSE_Link is very small and does not change much, implying that the observed link flow pattern is reproduced perfectly. Further, to achieve a satisfying link flow estimate requires that ϵ be set to be less than 0.01 (See Figure 6.5. The performance of LPFE seems not sensitive to the value of bn, as revealed in Figure 6.6. This is probably due to that this network contains a small number of constraints thus bad iterations are unlikely to appear. Figure 6.7 illustrates the impact of different combinations of z and zn. As we can see, for a fixed z, RMSE_Link decreases as zn increases. However, a very loose setting of z (say 0.5 or 1.0) can adversely affect LPFE's capability to reproduce measured link counts, no matter how many zigzagging iterations are allowed. In short, for this small network, satisfying results can be obtained using the following setting: $n \geq 100, \epsilon < 0.01, bn > 1, z < 0.1, zn > 5$.

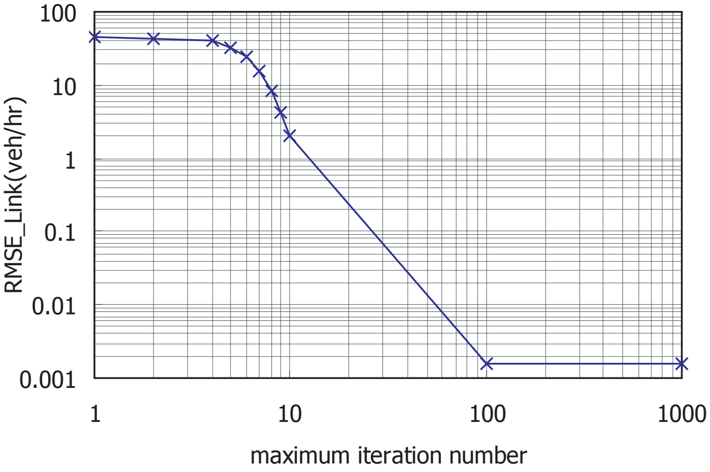


Figure 6.4 Scenarios with different n ($\varepsilon = 0.001, bn = 10, z = 0.001, zn = 10$)

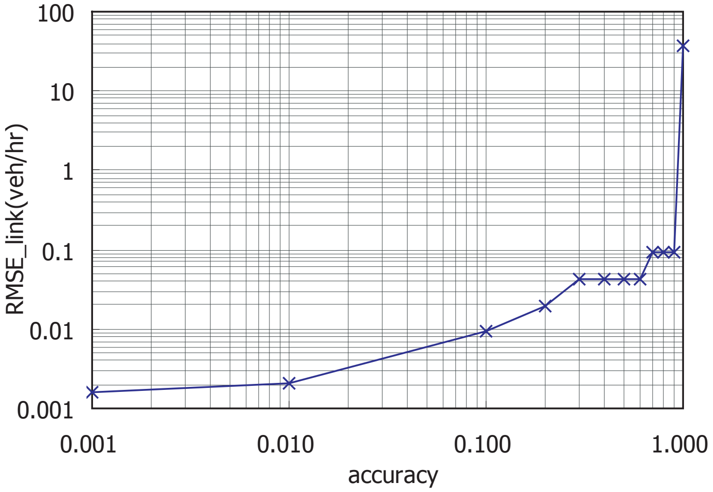


Figure 6.5 Scenarios with different ε $(n = 1000, bn = 10, z = 0.001, zn = 10)$

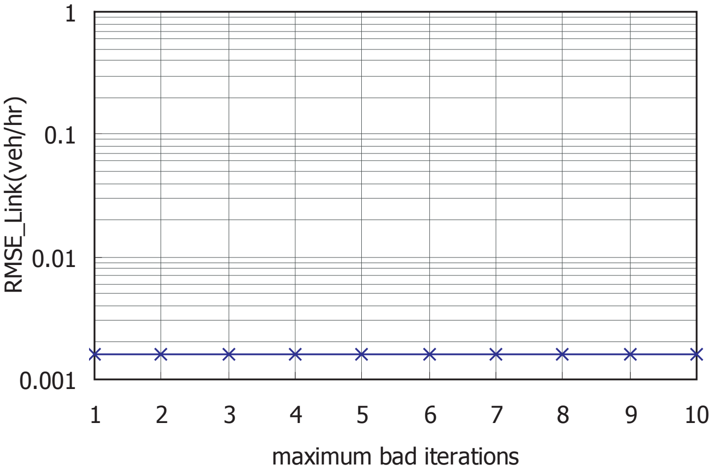


Figure 6.6 Scenarios with different bn $(n = 1000, \varepsilon = 0.001, z = 0.001, zn = 10)$

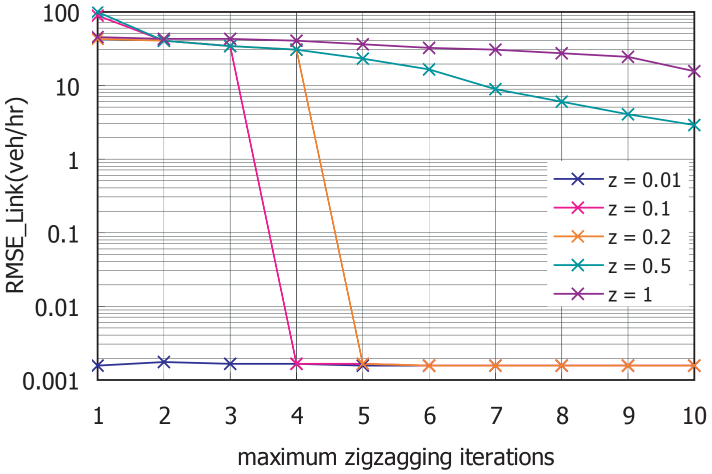
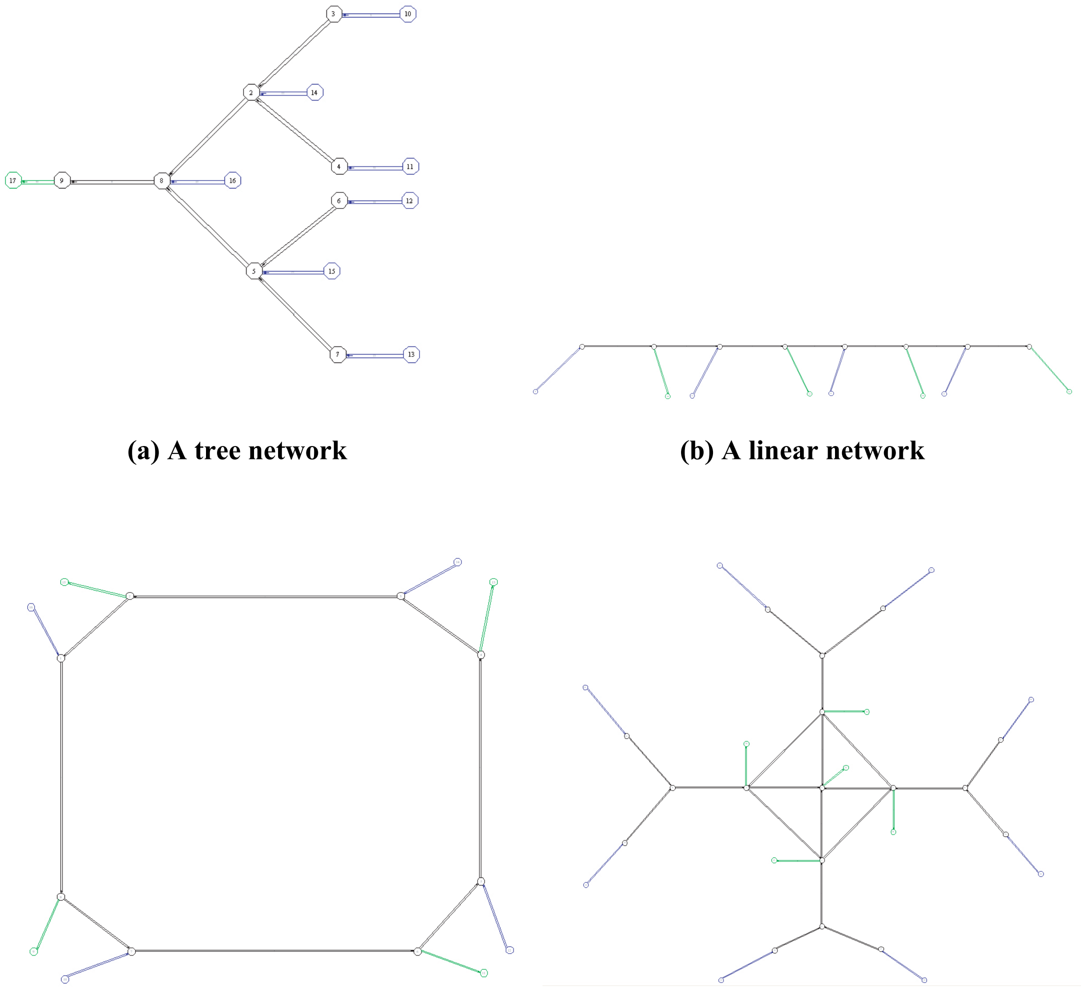


Figure 6.7 Scenarios with different z **&** zn $(n = 1000, \varepsilon = 0.001, bn = 10)$

Although the setting of algorithmic parameters is usually problem-specific, a general guideline can still be drawn. In most cases, if an estimation result is regarded as unsatisfying, users should so adjust the algorithmic parameters that LPFE would perform more DDAG iterations. To achieve this objective, one can try increase n, zn, bn, and/or reduce ϵ, z. In general, such adjustment helps produce a result better fulfilling the constraints. However, note that the CPU time increases rapidly with the number of iterations spent in the sub problem. Thus, users should exercise caution when making such parameter adjustments, particularly for large-scale problems.

6.3 The Impact of Network Topology

To study the impact of network topology on O-D estimation results, we focus on several basic network topologies. Most transportation networks can be considered as a combination of these basic types. The basic topology types in consideration include: *tree*, *linear*, *ring (a special linear type)*, and the combination of the above three (called a *derived* type in our discussion). Figure 5.8 illustrates these basic types.



(a) A tree network

(b) A linear network

(c) A ring network

(d) A derived network

Figure 6.8 Basic types of network topology

A *tree* network(Figure 6.8a) can be regarded as a typical network in mono-centric cities, where people live in the suburbs and work in a central business district(CBD). Usually, commuters have one primary path to travel from home to the work location. The *linear* network (Figure 6.8b) and the *ring* network (Figure 6.8c) are simplifications to corridor networks in multi-centric cities. In a linear-type network, the home places and work places of people locate along a *linear* arterial road, while in a ring network, major economic activities happen along a *ring* arterial road. Finally, a *derived* network is usually more complex and contains elements of the above three types of topologies. For example, the network in the same suburban neighborhood might be a tree-type network, where people from different local roads drive to the same freeway connecting the suburban neighborhood and the CBD area of the city. Within the CBD area, there might be either a linear road or ring road connecting all the freeways coming from different suburban neighborhoods. Again, we assume that all link flows are measured, and no observed/historical O-D information is available.

6.3.1 The case with a tree network

The tested tree network is shown in Figure 6.9, which consists of 16 nodes, 15 links, seven origins, and one destination. The number of OD pairs is seven. The true OD flows are listed in Figure 6.10. Figure 6.11 and Figure 6.12 visualize the estimated O-D profile and link flow profile. The red points in both plots represent the true values (O-D flow or link flow), while the blue points represent the estimated values (O-D flow or link flow). For a perfect estimation, in which true O-D and link flows are exactly reproduced by estimates, all the red points should overlap the blue points.

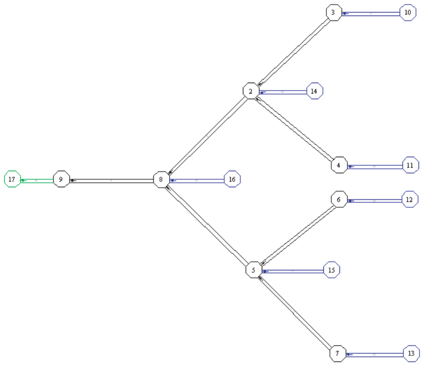
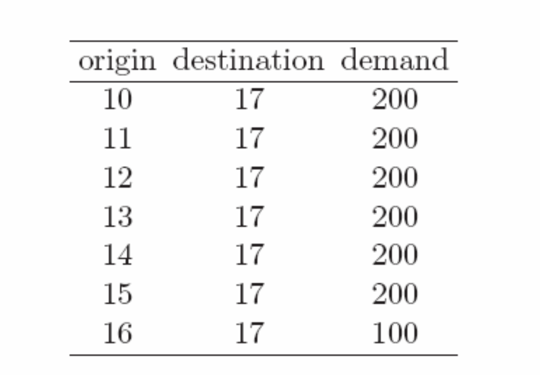


Figure 6.9 A tree network



origin	destination	demand
10	17	200
11	17	200
12	17	200
13	17	200
14	17	200
15	17	200
16	17	100

Figure 6.10 True O-D trip table

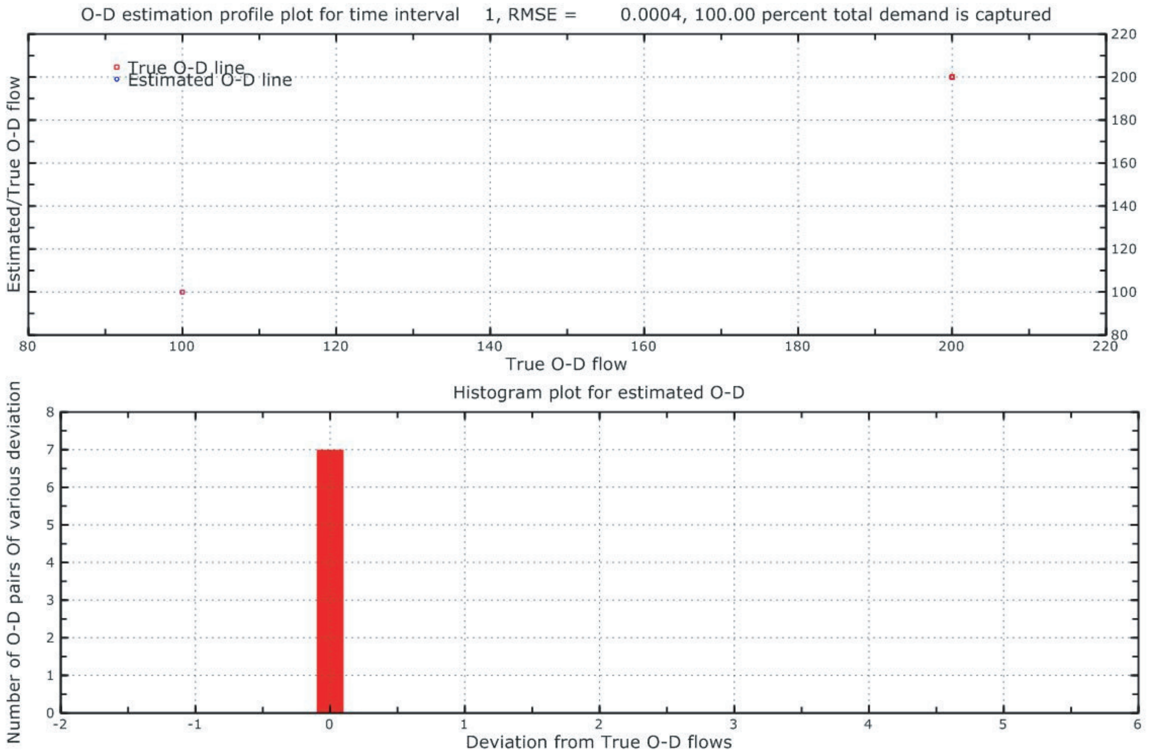


Figure 6.11 Estimated O-D flow profile for the tree network

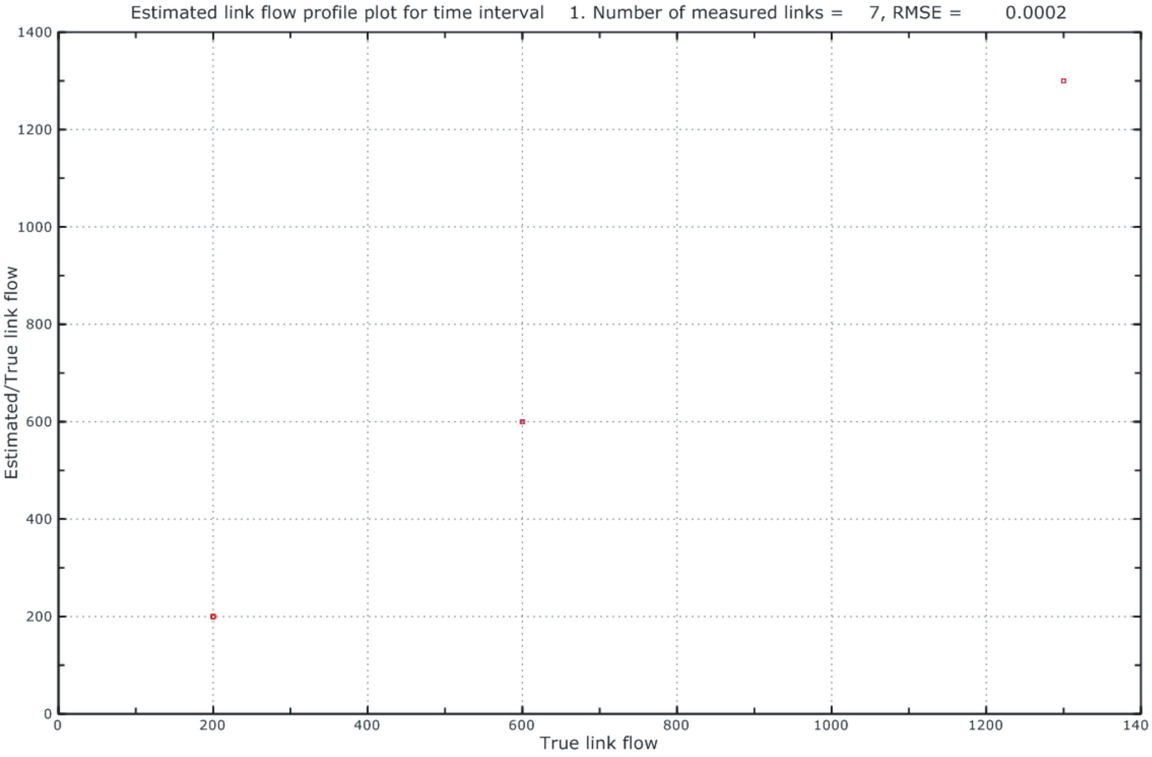


Figure 6.12 Estimated link flow profile for the tree network

From Figure 6.11 and Figure 6.12, we observe that both the estimated OD flows and the estimated link flows equal the true values, with TDC = 100%, RMSE_OD = 0, and RMSE_Link = 0. That is, a perfect estimation is obtained.

6.3.2 The case with a linear network

The tested *linear* network is shown in Figure 6.13. The network consists of 16 nodes (four origins, and four destinations), 15 links, and ten O-D pairs. The true O-D flows are listed in Table 6.14. The estimated O-D flow profile and link flow profile are shown in Figure 6.15 and Figure 6.16.

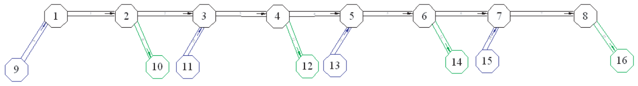
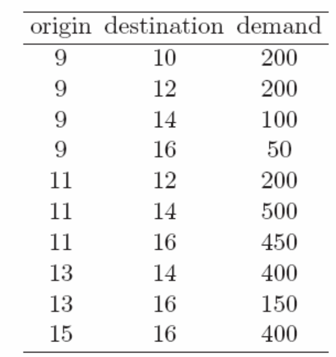


Figure 6.13 A linear network



origin	destination	demand
9	10	200
9	12	200
9	14	100
9	16	50
11	12	200
11	14	500
11	16	450
13	14	400
13	16	150
15	16	400

Figure 6.14 True O-D trip table

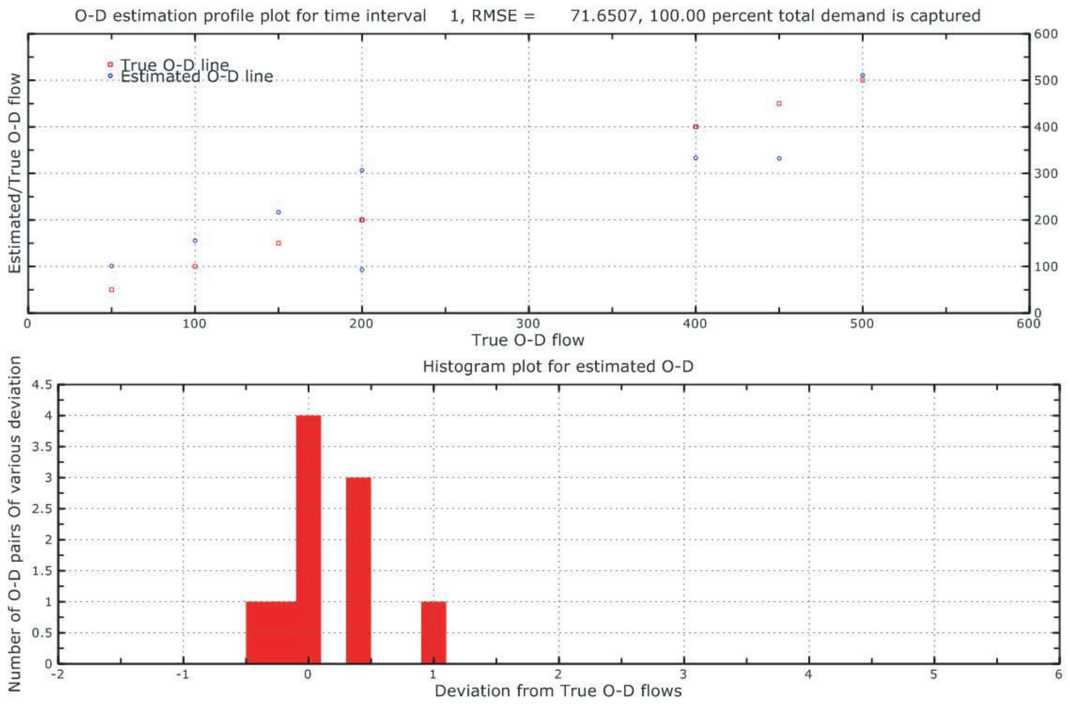


Figure 6.15 Estimated O-D flow profile for the linear network

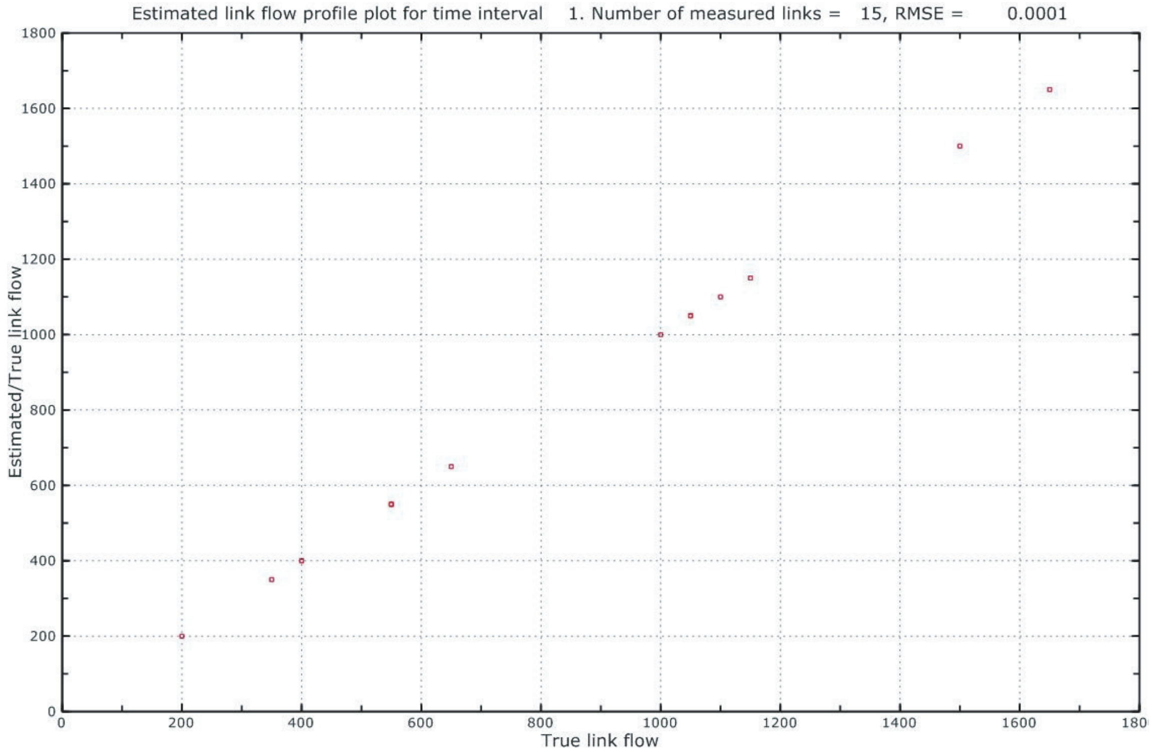


Figure 6.16 Estimated link flow profile for the linear network

Figure 6.16 shows that the estimated link flow profile replicates exactly the true link flow profile with RMSE_Link = 0. In addition, the total amount of demand is captured accurately too, with TDC = 100%. However, Figure 6.15 indicates that there are discrepancies between the estimated O-D flows and the true O-D flows, with RMSE_OD = 71.6507. This is expected because for a certain link flow profile in a *linear* network, there can be more than one path flow solutions corresponding to it. What LPFE predicts is just one of the path solutions, which may not be the true one. Such errors are of un intrinsic nature due to the under-determined property of O-D estimation problems.

6.3.3 The case with a ring network

The tested *ring* network is made up of 16 nodes (four origins, and four destinations), 16 links, and 16 O-D pairs, as illustrated in Figure 6.17. Figure 6.18 provides the true O-D flows. The estimated O-D flow profile and the link flow profile are shown in Figure 6.19 and Figure 6.20.

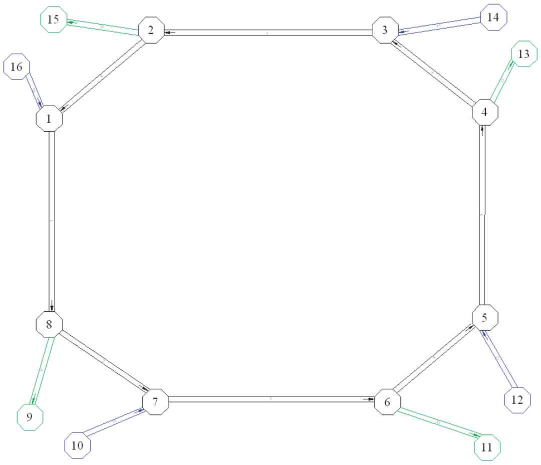


Figure 6.17 Ring network

origin	destination	demand	origin	destination	demand
10	11	40	14	11	140
10	13	20	14	13	200
10	15	125	14	15	300
10	9	34	14	9	200
12	11	50	16	11	100
12	13	100	16	13	50
12	15	130	16	15	20
12	9	140	16	9	80

Figure 6.18 True O-D trip table

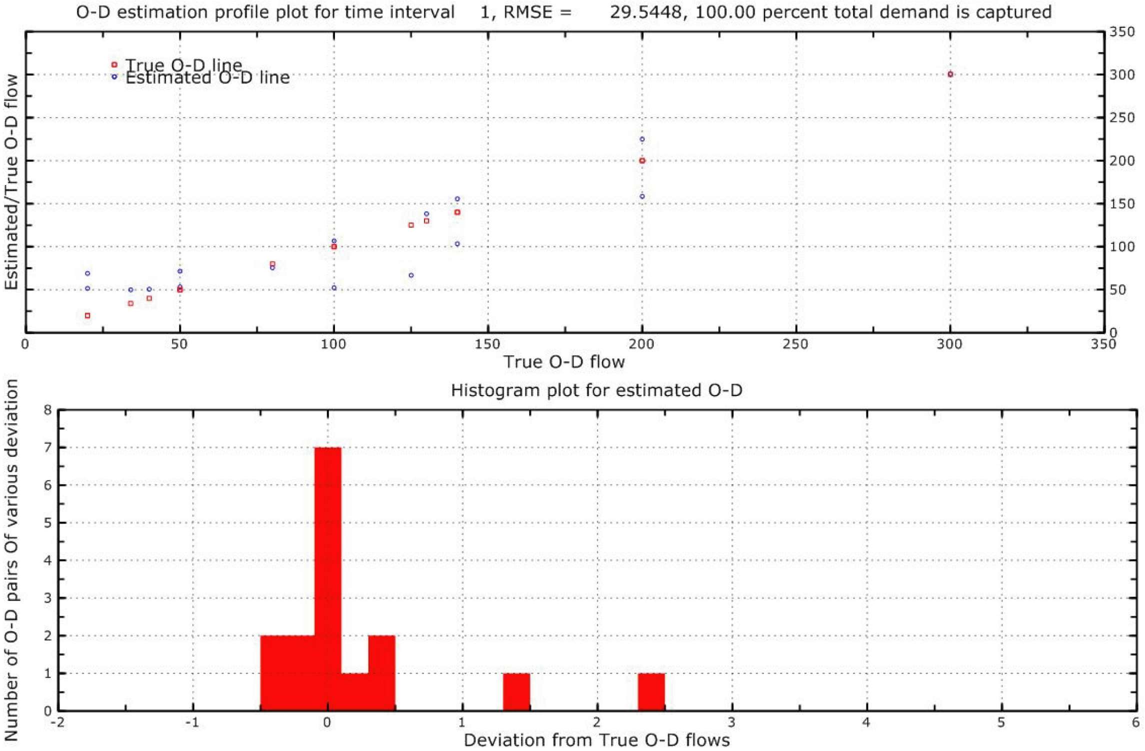


Figure 6.19 Estimated O-D profile for the ring network

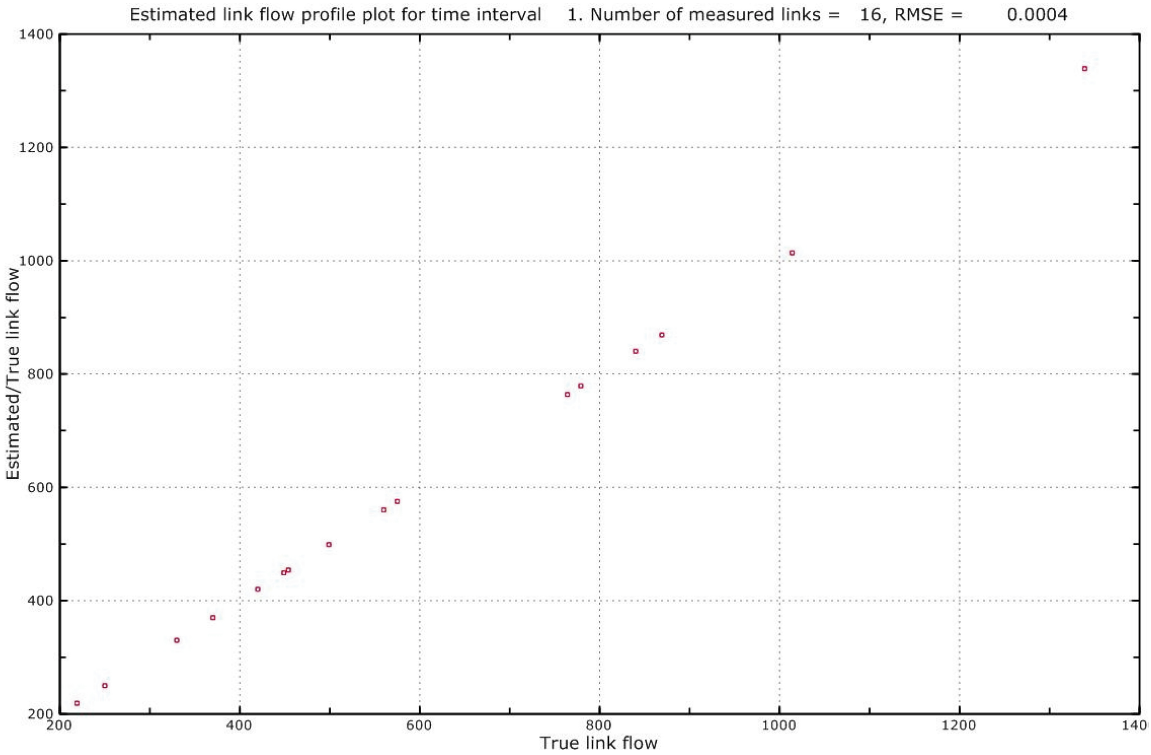
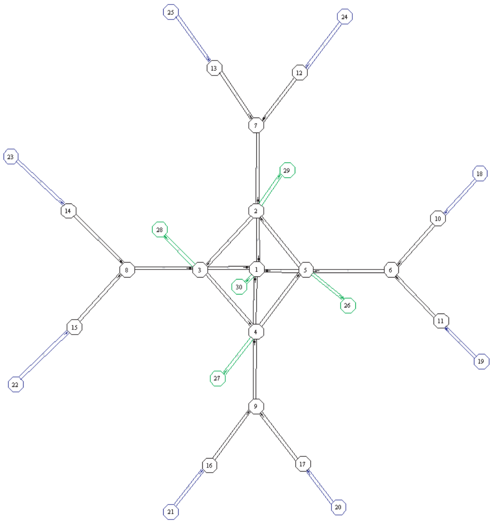


Figure 6.20 Estimated link flow profile for the ring network

Similar observations can be made for the *ring* network as the *linear* network, which is somewhat expected because a ring network is a special type of linear network. Both the link flow profile and the total amount of demand are predicted accurately, while estimated O-D flows does not perfectly match "true" O-D flows (with RMSE_OD = 29.5448). This is again due to the fact that the path flow solution in a *ring* network is usually not unique.

6.3.4 The case with a derived general network

The *derived* network we use for testing is shown in Figure 6.21. There are 30 nodes (eight origins, and five destinations), 33 links, and 40 O-D pairs in the network. The true O-D flows are provided in Figure 6.22. The estimated O-D flow profile and link flow profile are presented in Figure 6.23 and Figure 6.24.



origin	destination	demand	origin	destination	demand
18	30	200	22	30	150
18	26	30	22	26	35
18	27	50	22	27	25
18	28	20	22	28	25
18	29	50	22	29	20
19	30	150	23	30	150
19	26	25	23	26	15
19	27	40	23	27	20
19	28	30	23	28	20
19	29	60	23	29	25
20	30	100	24	30	150
20	26	10	24	26	20
20	27	20	24	27	15
20	28	25	24	28	20
20	29	20	24	29	25
21	30	100	25	30	200
21	26	20	25	26	15
21	27	25	25	27	20
21	28	20	25	28	25
21	29	30	25	29	30

Figure 6.21 A derived general network

Figure 6.22 True O-D trip table

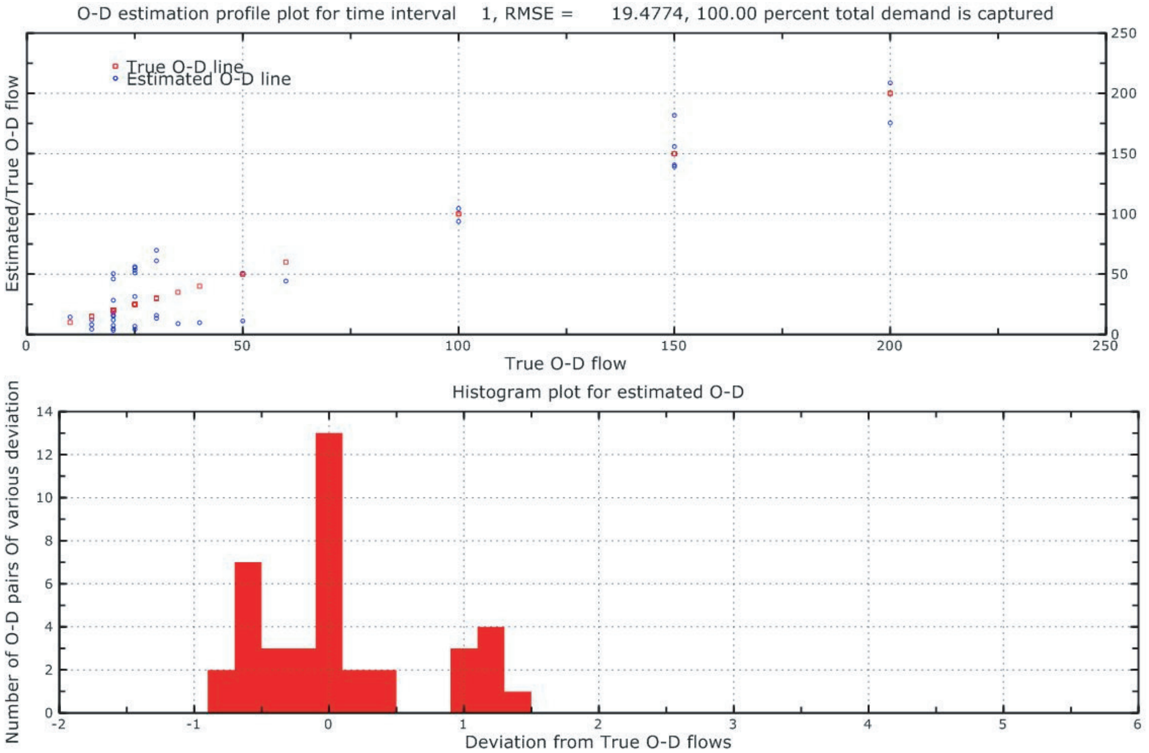


Figure 6.23 O-D estimation profile for the derived network

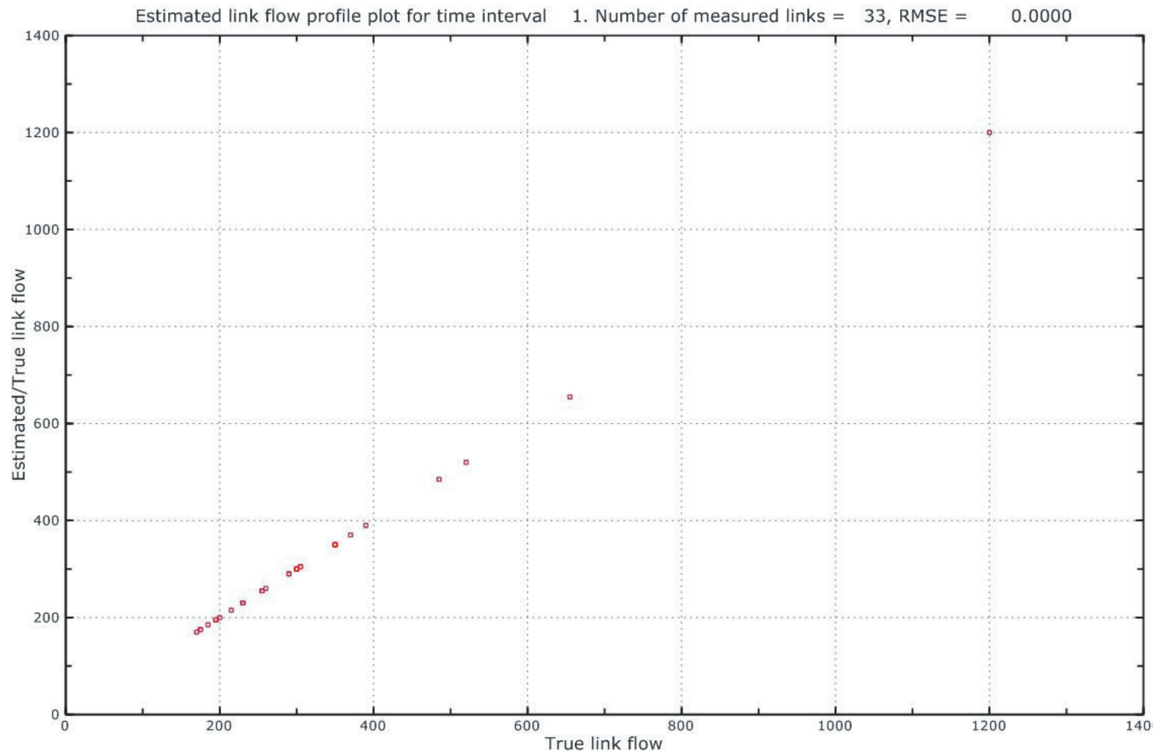


Figure 6.24 Estimated link flow profile for the derived network

As expected, for a more complex network, the link flow profile and the total amount of demand are still been accurately estimated, with RMSE_Link = 0, TDC = 100%. Due to the multiplicity of path flow solutions, deviations are observed between the estimated O-D flows and true O-D flows, with RMSE_OD = 19.4774.

In summary, for the basic network types we discussed above, LPFE can offer satisfying link flow estimates and accurately capture the total demands, provided all link flows are measured. The true O-D flows can not be reproduced exactly if there is more than one path solution, as for the *linear*, *ring*, and *derived* networks. Obviously, link traffic counts alone cannot guarantee estimation results of good quality in most cases. Additional information, such as partial O-D information, is needed in general, and this will be discussed in the next section.

6.4 The Impact of Additional Inputs

Starting from this section, we begin to work on a real network, the network of Dallas and Fort Worth in Texas. The Dallas-Fort Worth network (Figure 6.25) consists of 211 nodes (13 of which are origins and 13 of which are destinations), 481 links (26 of which are centroid connectors), and 140 O-D pairs. A two-hour static O-D trip table was obtained. When the static O-D estimation is performed, the hourly static O-D flows are treated as the true O-D flows. The static O-D trip table will further be decomposed to construct the hypothetical dynamic true O-D flows in the time-dependent case. For brevity, the O-D trip table is not listed here.

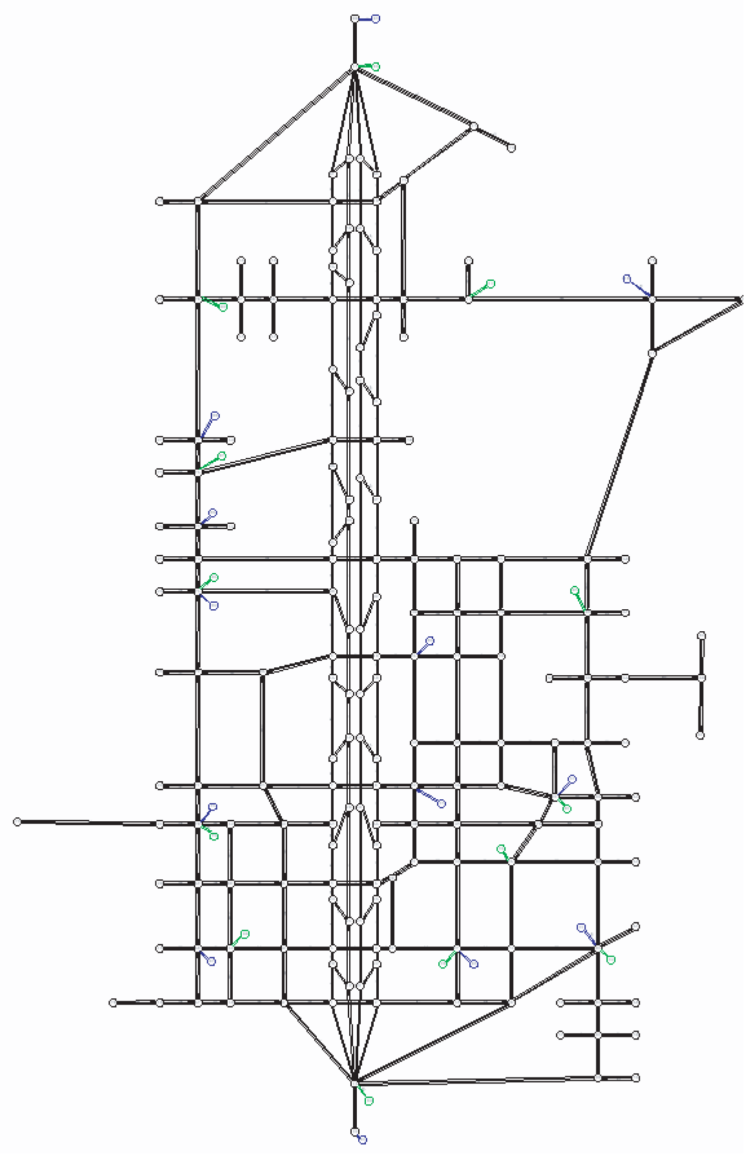


Figure 6.25 The Dallas-Fort Worth network

6.4.1 Effects of measurement location and the number of measurement locations on estimation performance

According to the observations made in Section 6.3, LPFE's capability of reproducing the true O-D flows depends on network topology. We first examine for this real network whether or not LPFE can provide reasonable results when inputs are ONLY measured link flows. The different sets of measurements in consideration include:

- flows on all links are measured (number of links: 267)
- flows on congested links are measured (number of links: 45)
- flows on efficient links are measured (number of links: 33)
- flows on freeway links are measured (number of links: 27)
- flows on centroid connectors and freeway links are measured (number of links: 53)

The set of all measured links consists of the links whose volume-capacity ratio is larger than 0.1. The set of congested links include all the links with volume-capacity ratio larger than 0.5. Efficient links form a minimal link set covering all the used path. This link set is designed to capture some information of the spatial structure of the demand. Freeway links are the links on the freeway connecting the two main OD pairs across the city. The centroid connectors are the links locating at the entrance/exit of each origin/destination.

TDC, RMSE_OD and RMSE_link of estimation results corresponding to different inputs are shown in Figure 6.26, Figure 6.27, and Figure 6.28, respectively.

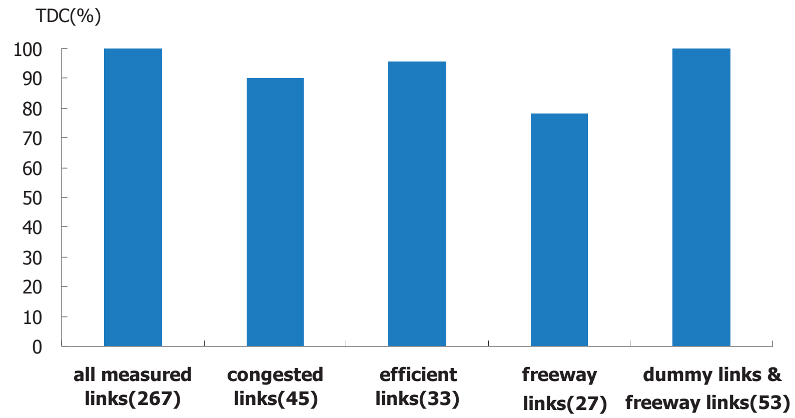


Figure 6.26 TDCs for different sets of measurements

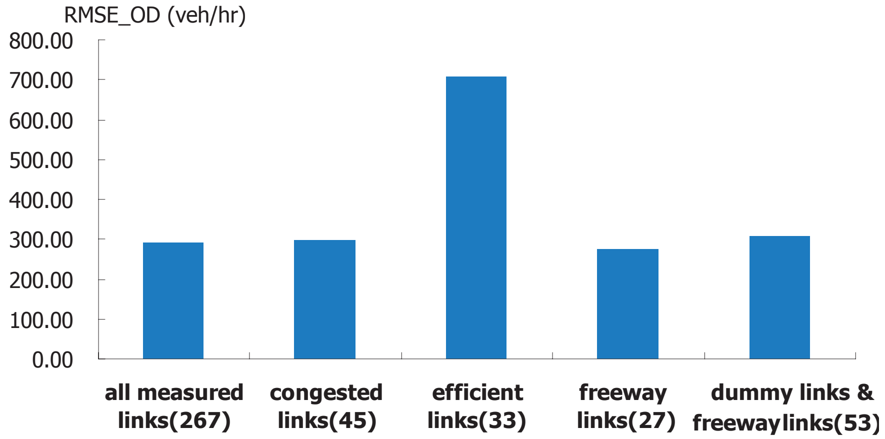


Figure 6.27 RMSE_ODs for different sets of measurements

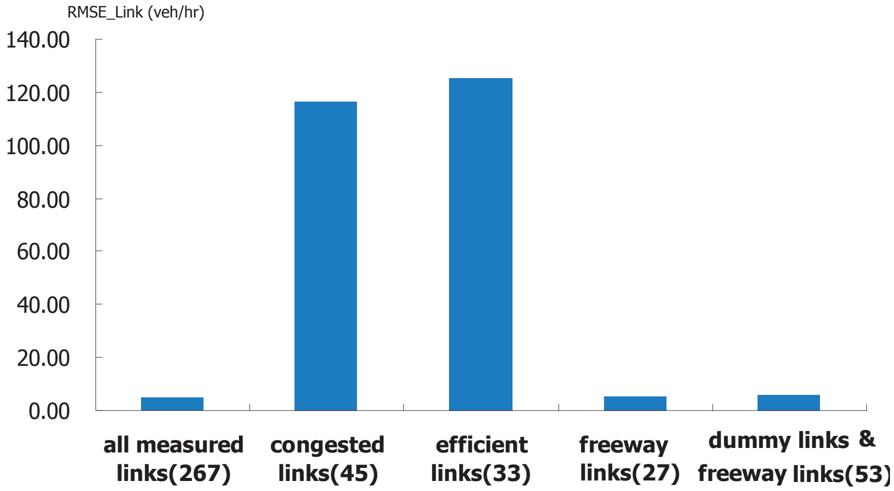


Figure 6.28 RMSE_Links for different sets of measurements

Figure 6.26 indicates that in most scenarios more than 90% of total demands are captured, except the scenario where only freeway links are measured (in which TDC = 78.15%). Interestingly, the total demand captured by the scenario with as much as 267 links (all measured links) is not very different from what captured by the scenario with only 33 links (efficient links), suggesting that LPFE can capture the total demand with a much smaller number of counting locations than what would be available.

Three out of the five scenarios, i.e., scenarios with input set of all measured links, freeway links, and centroid connectors & freeway links, replicate the link flows quite well with RMSE_Link no greater than 10veh/h. However, the RMSE_ODs for all the scenarios are rather high, compared to the average O-D flows (152veh/hr). This is again due to the existence of multiple path solutions corresponding to an observed link flow pattern. Moreover, the scenario of efficient links, which originally designed to capture spatial structure of the demand to some extent, perform the worst instead of the best in terms of RMSE_ODs.

To summarize, the scenario with input set of all measured links, and the scenario with input set of centroid connectors & freeway links seem to outperform the other three, when all the MOEs are taken into account.

6.4.2 The effects of historical O-D flows(50%)

In this section we introduce historical (observed) O-D data into inputs to improve the quality of estimates. The historical O-D trip table is generated by randomly introducing a perturbation with bounds $[-50\%, 50\%]$ to each O-D pair (RMSE_OD between this historical O-D trip table and the true O-D trip table is 110.41veh/hr). For comparison purpose, the same O-D trip table is used in combination with all the five link sets discussed in the last section. The MOEs of estimation results with and without the observed O-D trip table are shown in Figure 6.29, Figure 6.30, and Figure 6.31.

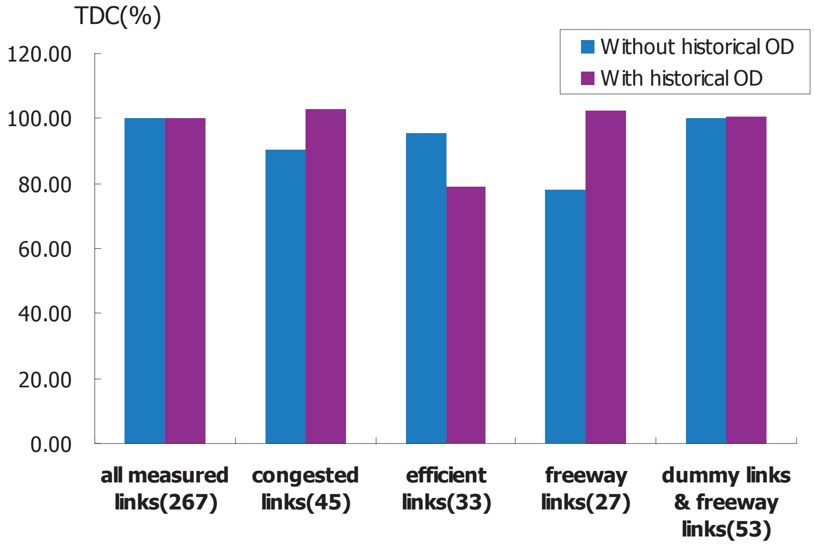


Figure 6.29 TDCs for different link sets with and without historical O-D data

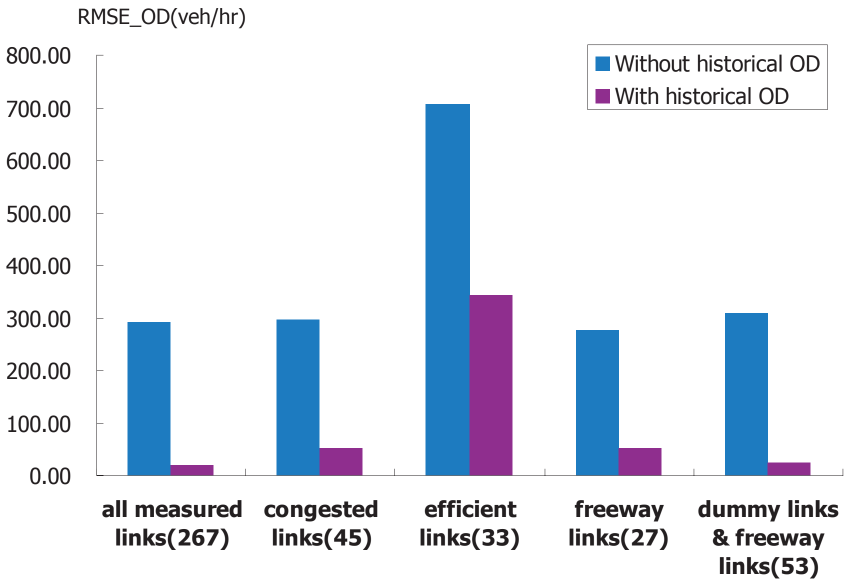


Figure 6.30 RMSE_ODs for different link sets with and without historical O-D data

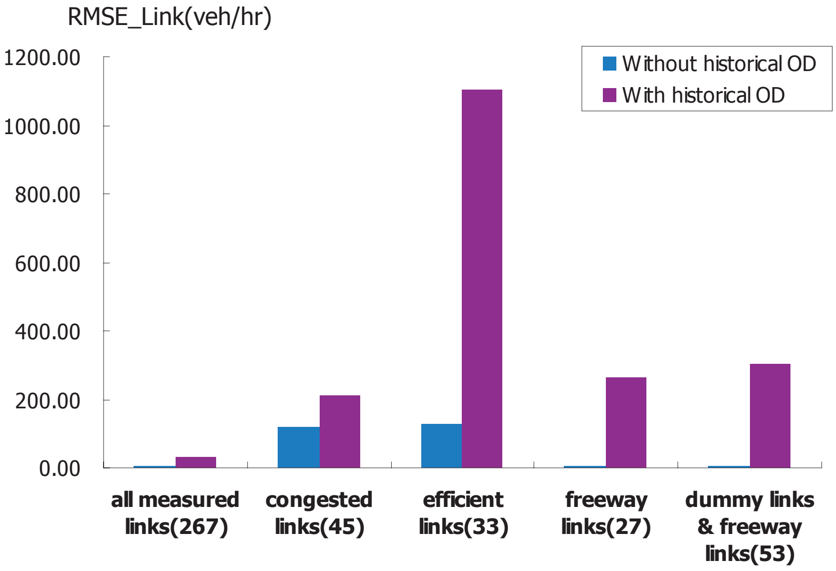


Figure 6.31 RMSE_Links for different link sets with and without historical O-D data

As indicated by Figure 6.29, by including historical O-D data into the input set, the TDCs of all the scenarios except the scenario with the input set of efficient links, are getting even closer to 100 %. Even for the scenario of freeway links only, which performed unsatisfactorily before (TDC = 78.15 %), the TDC becomes 102.27% with a disturbed historical O-D table.

RMSE_OD is reduced significantly too, with the included historical O-D information. For all scenarios except the one using efficient links, the RMSE_ODs are all below 60veh/hr. For the efficient link scenarios, the RMSE_OD also shows a dramatic drop from 706.89veh/hr to 343.23veh/hr (although it is still greater than the RMSE_OD of the historical O-D flows itself (110.41veh/hr).

The change of RMSE_Link in presence of O-D information is different: they increase instead of decrease in all cases. As shown in algorithm IBA for E-LPFE-SUB, the checking of O-D constraints always follows after that of link constraints. As a result, the algorithm might favor the consistency of O-D constraints to that of link constraints

In short, the estimation results of all scenarios are substantially improved when a historical O-D trip table is included as inputs. Note that the estimated result based on historical O-D trip table and *267* links (all measured links) does not show significant difference from the result obtained by the historical O-D trip table and *27* links (freeway links). In other words, for this network, we can get a pretty good estimation result by making use of a small set of the link counts as well as historical O-D flows. Note that the historical O-D trip table itself was subject to significant errors (RMSE_OD = 110.41veh/hr). Yet it does provide important information regarding to the spatial structure of O-D demands which greatly improves estimation results .

6.5 The effects of Measurement Errors

In reality, both link flow and O-D flow measurements are subject to errors which are usually not known exactly beforehand. For example, link traffic counts measured by detection devices might contain errors caused by calibration inaccuracy, failure of the device and so on. Further, a historical O-D trip table based on a travel diary survey done several years ago might be well out-of-date. Hence, the robustness of LPFE in presence of input errors is an important issue to address. We test the influence of measurement errors of O-D flows and link flows in subsection 6.5.1 and 6.5.2, respectively. All the tests use Dallas-Fort Worth network described in section 6.4.

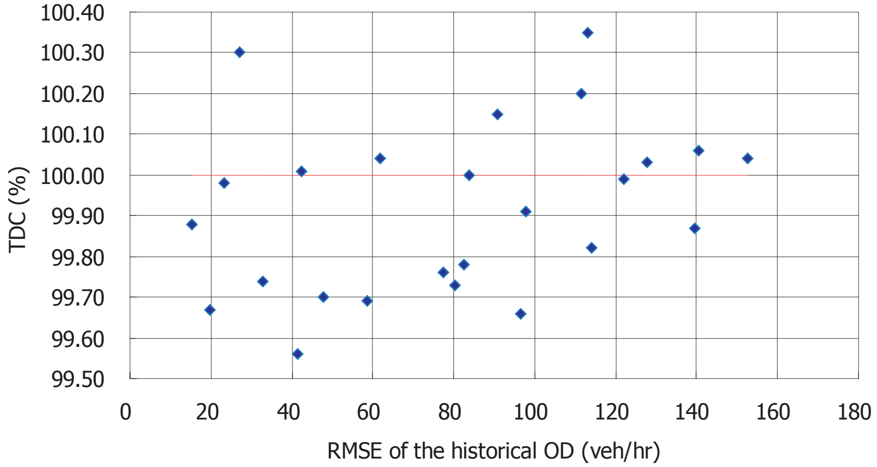
6.5.1 Measurement errors of O-D flows

Two different link flow sets, i.e., all links, and freeway links, are used to study the influence of measurement errors of historical O-D flows. For clarity link measurement errors are not considered here. For each link set, we set up 25 scenarios, each associated with a hypothetical historical O-D trip table with various levels of measurement errors. These O-D trip tables are generated by randomly applying a perturbation to (which is bounded from above) the "true" O-D flows. Five upper bounds, namely 10%, 20%, 30%, 40% and 50%, are considered for these perturbations, for each of which we generate five random O-D tables.

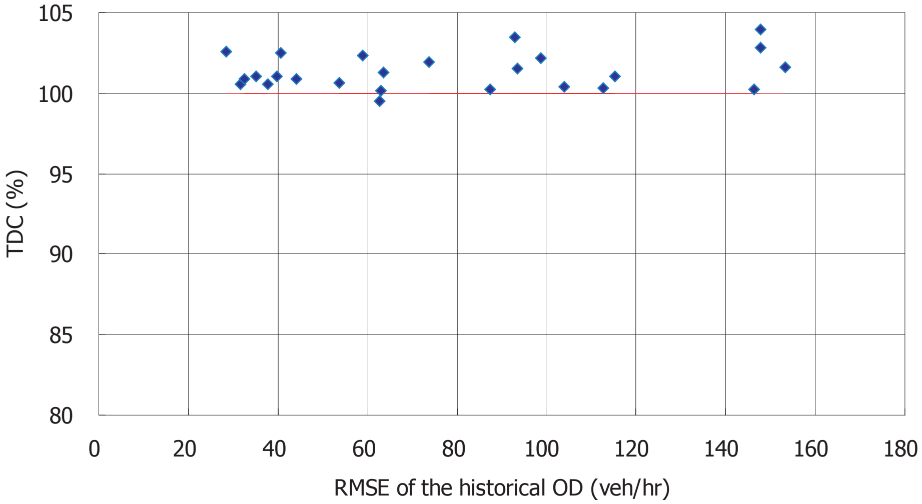
The RMSE_ODs of the hypothetical historical O-D trip tables of all the scenarios are given in Table 6.1, while MOEs are shown in Figures 6.32, 6.33, and 6.34. It can be seen from the figures that TDC is insensitive to the change of RMSE_ODs. Note that TDCs of all the scenarios fall into a narrow interval from 95% to 105%.

Table 6.1 RMSE_ODs for hypothetical historical O-D flow trip tables

uniform error	for the link set of all measured links veh/hr	for link set of freeway links veh/hr
±10%	15.33	40.51
	19.66	31.79
	42.26	37.85
	26.94	34.99
	23.41	28.46
±20%	47.88	62.81
	32.69	39.68
	82.43	62.98
	80.25	44.21
	83.85	32.51
±30%	127.84	87.39
	41.49	112.53
	111.35	92.90
	90.88	93.52
	114.17	73.62
±40%	121.93	147.85
	58.59	147.66
	61.82	53.68
	97.86	115.14
	96.46	98.61
±50%	113.11	63.44
	77.27	58.90
	140.35	146.29
	152.67	103.94
	139.43	153.42

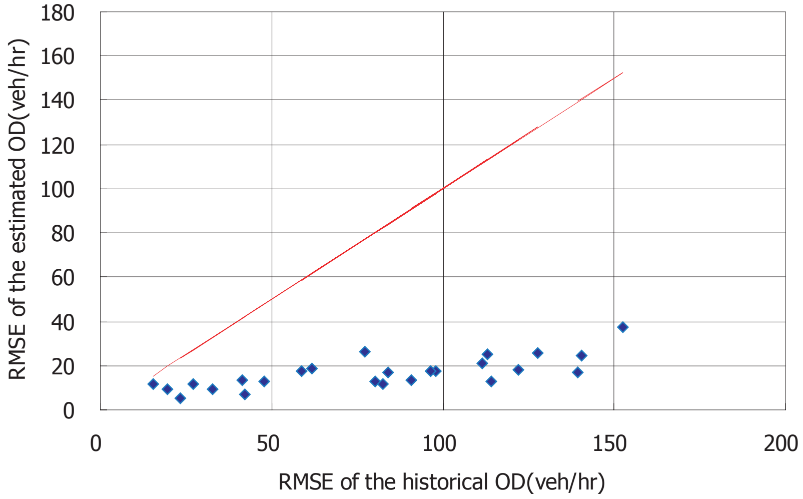


(a) all measured links (267) & historical O-D

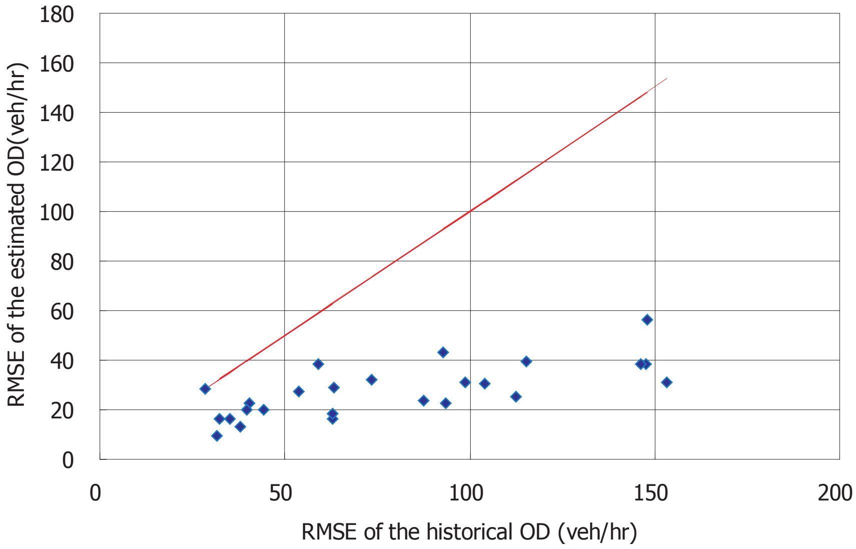


(b) freeway links (27) & historical O-D

Figure 6.32 TDCs for scenarios with different historical O-D flow errors

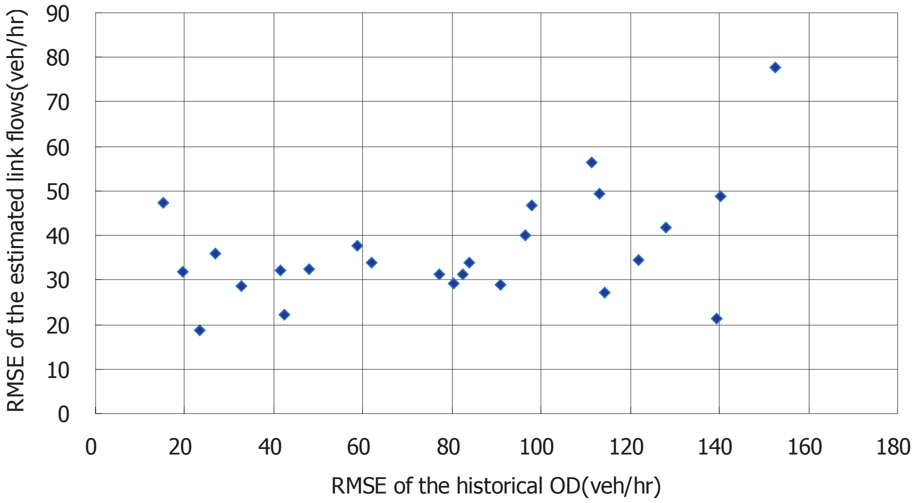


(a) all measured links (267) & historical O-D

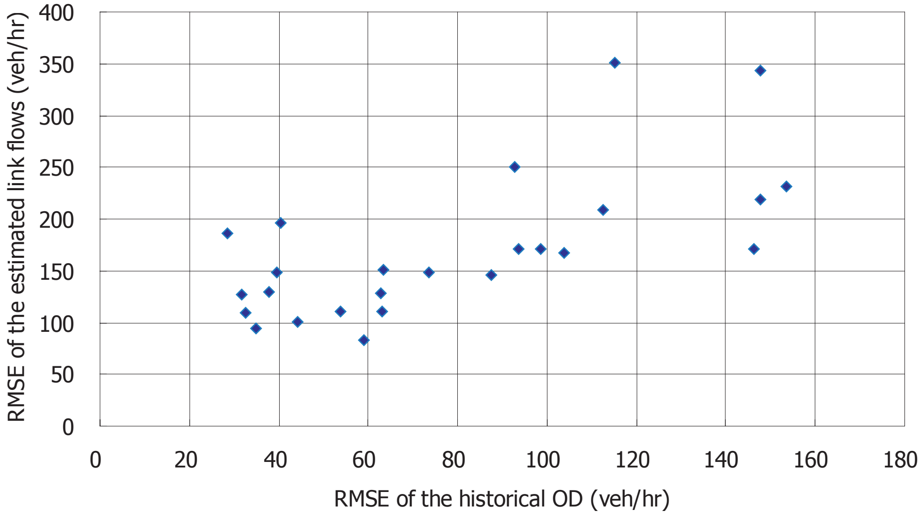


(b) freeway links (27) & historical O-D

Figure 6.33 RMSE_ODs for scenarios with different historical O-D flow errors



(a) all measured links (267) & historical O-D



(b) freeway links (27) & historical O-D

Figure 6.34 RMSE_Links for scenarios with different historical O-D flow errors

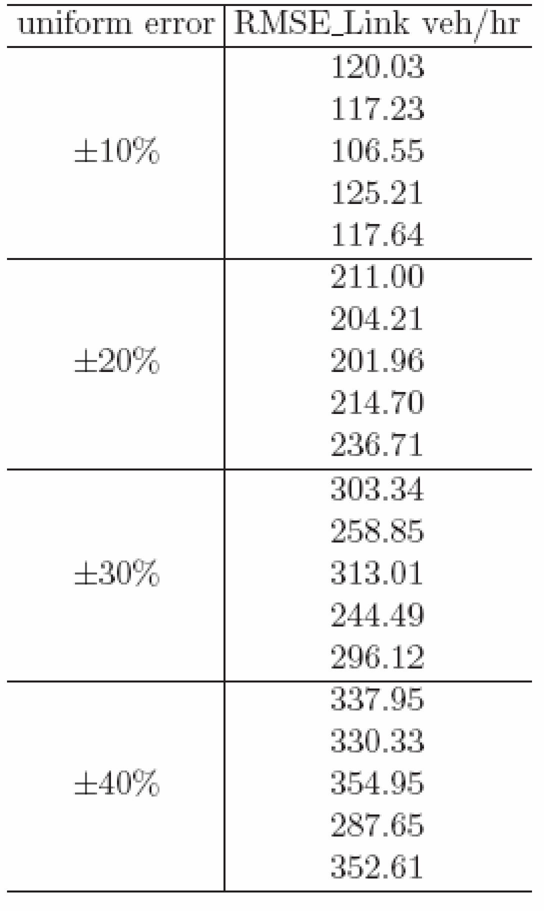
For RMSE_OD, the red line in Figure 6.34 marks the position where the RMSE_OD of the historical O-D flows equals the RMSE_OD of the estimated O-D flows. Thus a point below the red line means that the estimated O-D table is “closer” to the true O-D table than the target one. As we can see, for both sets, almost all the points lie below the red line, meaning that LPFE does improve the accuracy of the input O-D trip tables. Furthermore, when the RMSE_ODs of the historical O-D increase, the RMSE_ODs of the estimated O-D do not change very much (the RMSE_ODs of the estimated O-D is less than 60veh/hr in all cases). This seems to suggest that even a very coarse O-D table is still useful to improve estimation quality so long as it contains the structural pattern of the O-D demands to be estimated.

The RMSE_Links tend to slightly increase when the RMSE_ODs of the historical O-D trip table increase, especially for the input set of freeway links. Again, this might be due to that our algorithm always check link constraints before O-D constrains. However, compared to the magnitude of average link flows on freeway links (6104veh/hr), the RMSE_Links (the greatest one is 351veh/hr) are still acceptable.

6.5.2 Measurement errors of link flows

To study the influence of link measurement errors, 20 sets of hypothetical measured link flows are generated in a similar way as in the previous section: we assume all links are measured and only consider four upper bounds, i.e., $\pm 10\%, \pm 20\%, \pm 30\%, \pm 40\%$. RMSE_Links for all the scenarios are listed in Table. 6.2. The historical O-D trip table in these scenarios is the same as the one used in Section 6.4.2 OD(RMSE_OD = 110.41veh/hr). The MOEs are shown in Figures 6.35, 6.36, and 6.37.

Table 6.2 RMSE_Links for hypothetical measured link flows



uniform error	RMSE_Link veh/hr
±10%	120.03 117.23 106.55 125.21 117.64
±20%	211.00 204.21 201.96 214.70 236.71
±30%	303.34 258.85 313.01 244.49 296.12
±40%	337.95 330.33 354.95 287.65 352.61

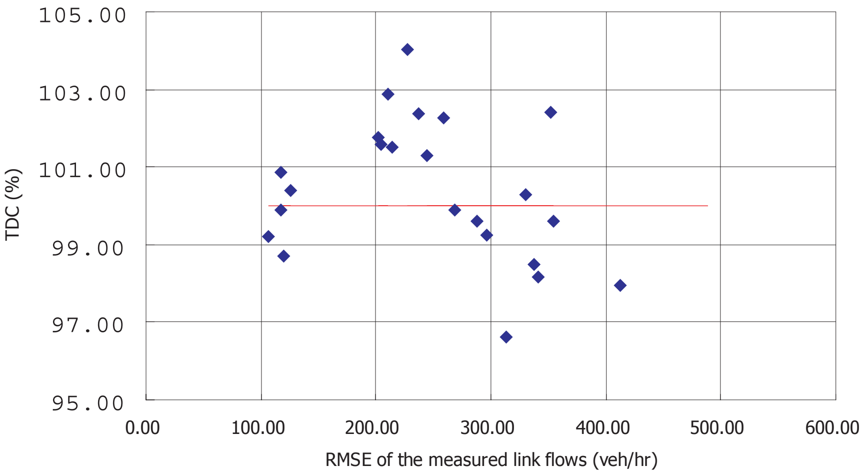


Figure 6.35 TDCs for scenarios with different link flow errors

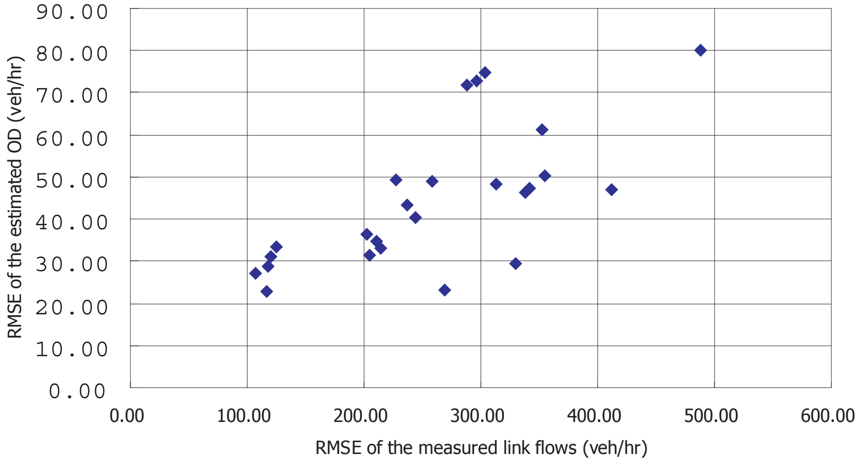


Figure 6.36 RMSE_ODs for scenarios with different link flow errors

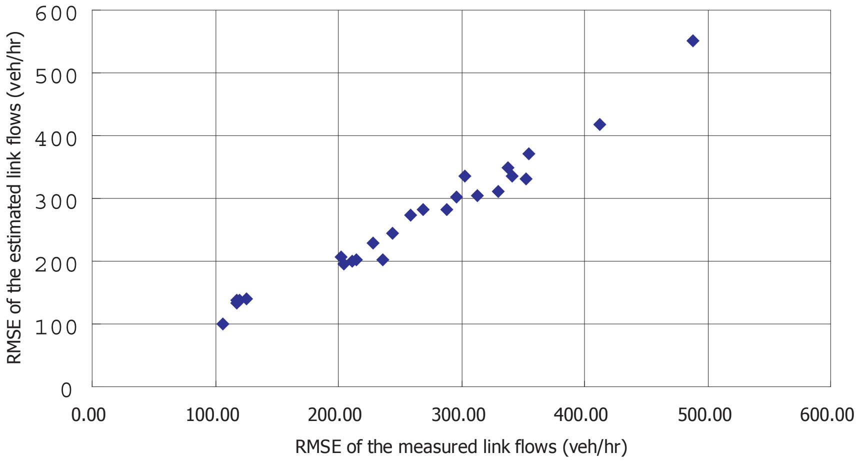


Figure 6.37 RMSE_Links for scenarios with different link flow errors

Figure 6.35 shows that TDC is not very sensitive to measurement errors of link flows. For all scenarios, the TDCs range between 95% to 105%.

Figures 6.36 and 6.37 show that both RMSE_OD and RMSE_Link increase as measurement errors in link counts increase. However, the RMSEs of estimated O-D flows in all scenarios are still lower than that of the historical O-D table, illustrating the robustness of LPFE in the presence of significant link measurement errors.

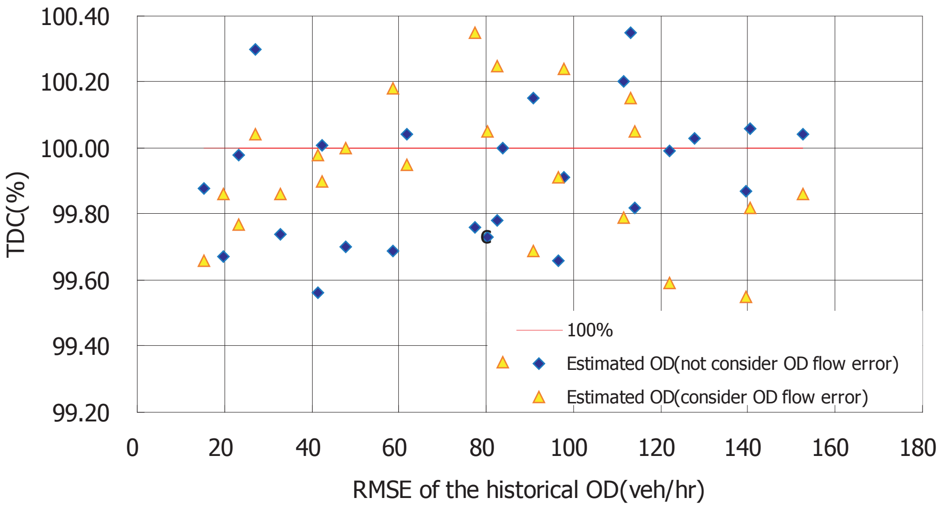
In summary, LPFE seems to be more sensitive to the link flow errors. When the input of link flows contain significant errors, LPFE may still capture the total demand well, but often fails to estimate O-D flows accurately. Thus, it is important to minimize link measurement errors in real applications. Fortunately, in practice it is much easier to obtain reasonably accurate link flow data than to obtain accurate historical O-D data.

6.6 Considering Measurement Errors in LPFE

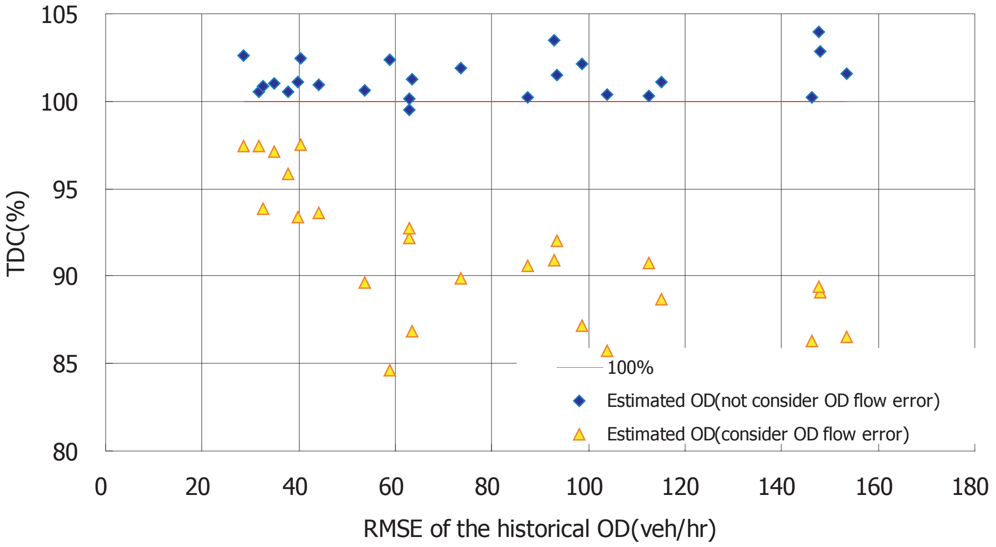
As shown in Chapter 2, LPFE is able to explicitly consider measurement errors of inputs by replacing the equality constraints with two sets of inequality constraints, which enlarges the feasible region such that it contains the “true” solution. This section will test whether or not this strategy helps provide a better solution.

6.6.1 Considering errors of the base O-D table

In this section, we rerun the 25 scenarios set up in Section 6.5.1, but address the O-D flow errors explicitly in LPFE. The MOEs of the estimation results are shown in Figures 6.38, 6.39, and 6.40. For comparison, the results when input errors are not explicitly considered are also plotted in those figures.

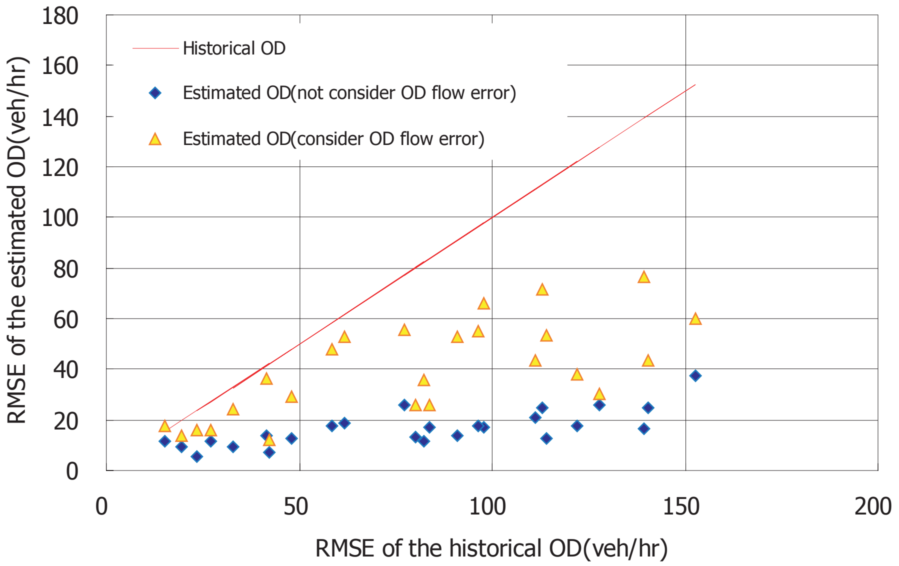


(a) all measured links (267) & historical OD

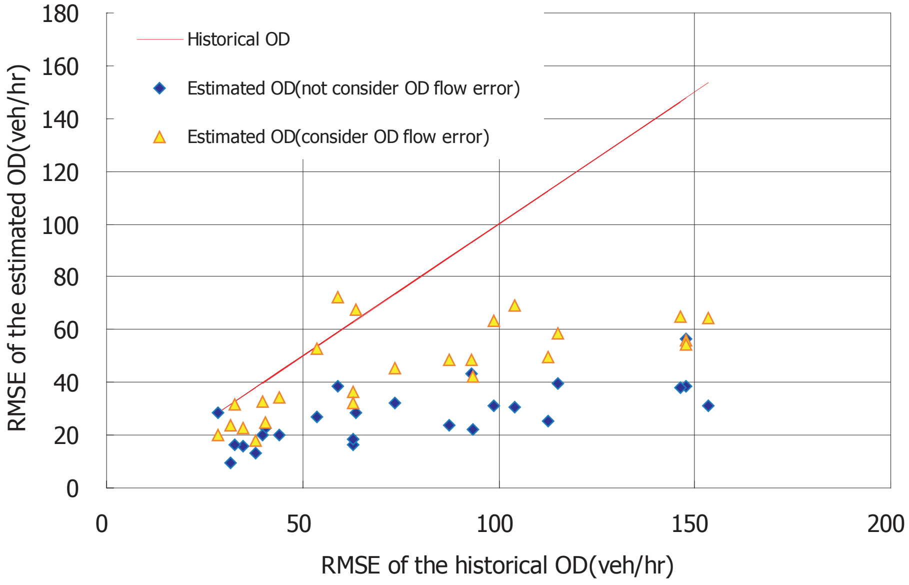


(b) freeway links (27) & historical OD

Figure 6.38 TDCs for scenarios with different settings on the consideration of O-D flow errors

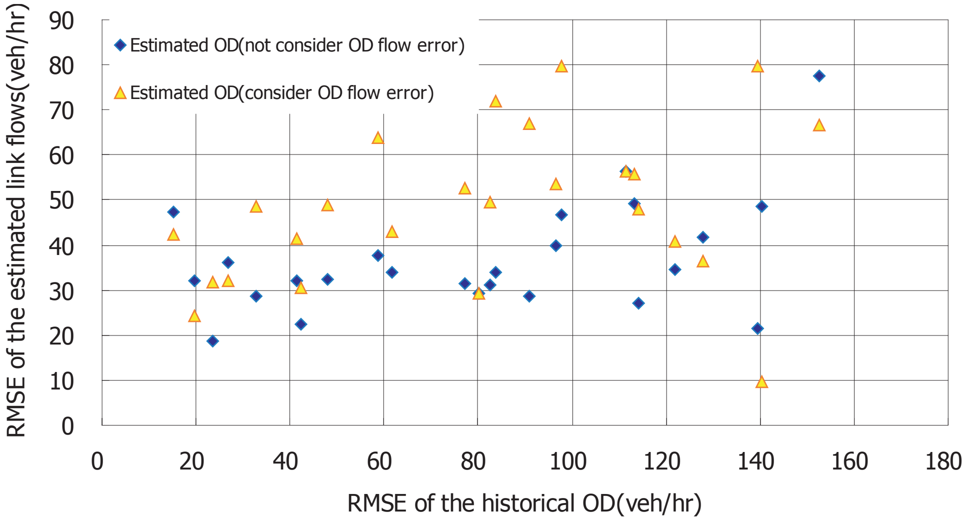


(a) all measured links (267) & historical OD

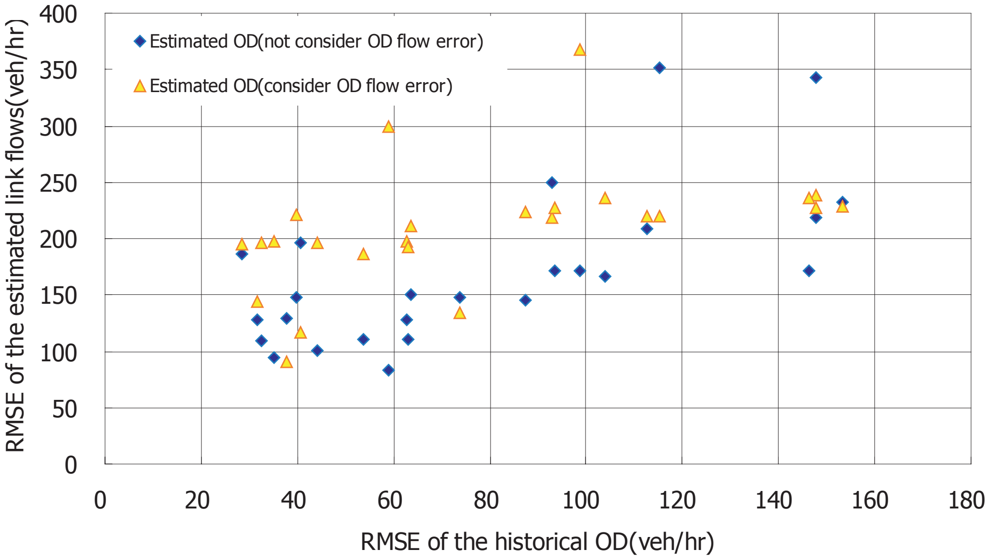


(b) freeway links (27) & historical OD

Figure 6.39 RMSE_ODs for scenarios with different settings on the consideration of O-D flow errors



(a) all measured links (267) & historical OD



(b) freeway links (27) & historical OD

Figure 6.40 RMSE_Links for scenarios with different settings on the consideration of O-D flow errors

In general these figures suggest that explicitly considering errors of input O-D trip rates produces worse (rather than better) estimates in terms of RMSE. This observation seems counterintuitive in the first glance. Recall that the errors of input O-D tables are considered in our formulation by confining estimated O-D flows within an interval centered at observed values (the interval length represents the confidence level). Although this strategy ensures that true OD values are contained within the feasible set of the optimization program, it enlarges the feasible region thus makes it harder to find the optimum. Moreover, our algorithm always fixes the infeasibility of a constraint by setting the estimated value to its boundary. Such adjustment may thus push estimates away from the "true" O-D trip rates since "true" values are in the interior of the feasible region instead of on its boundaries.

6.6.2 Considering errors of link measurements

Following similar logic, we in this subsection evaluate the effect of explicitly considering errors in link counts. The 20 scenarios used in Section 6.5.2 were rerun with the measurement errors of link flows considered. The MOEs of the estimation results are shown in Figures 6.41, 6.42, and 6.43.

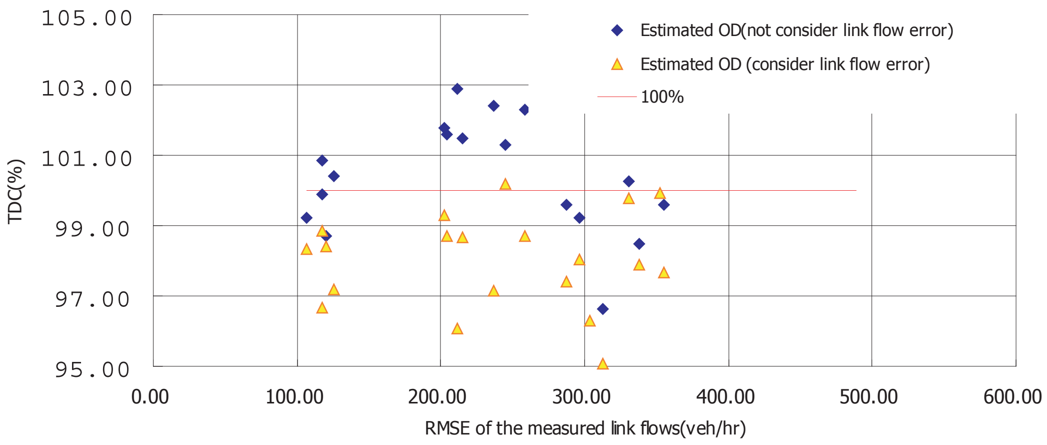


Figure 6.41 TDCs for scenarios with different settings on the consideration of link flow errors

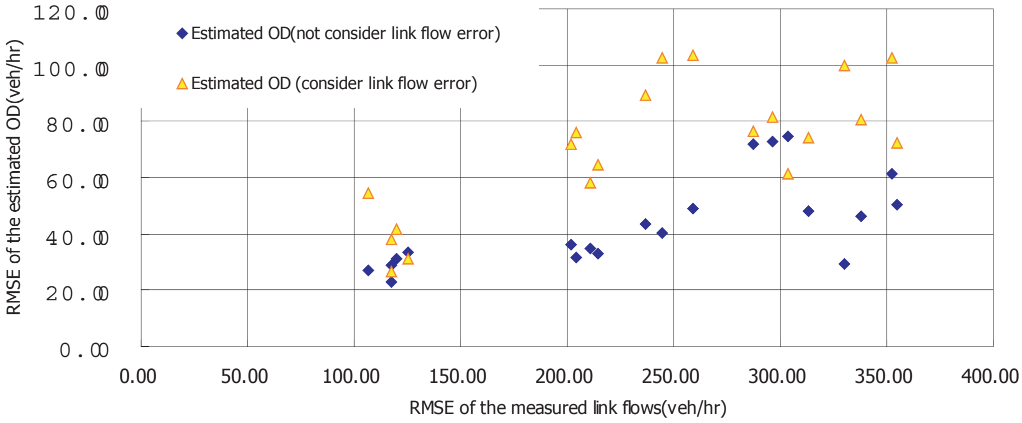


Figure 6.42 RMSE_ODs for scenarios with different settings on the consideration of link flow errors

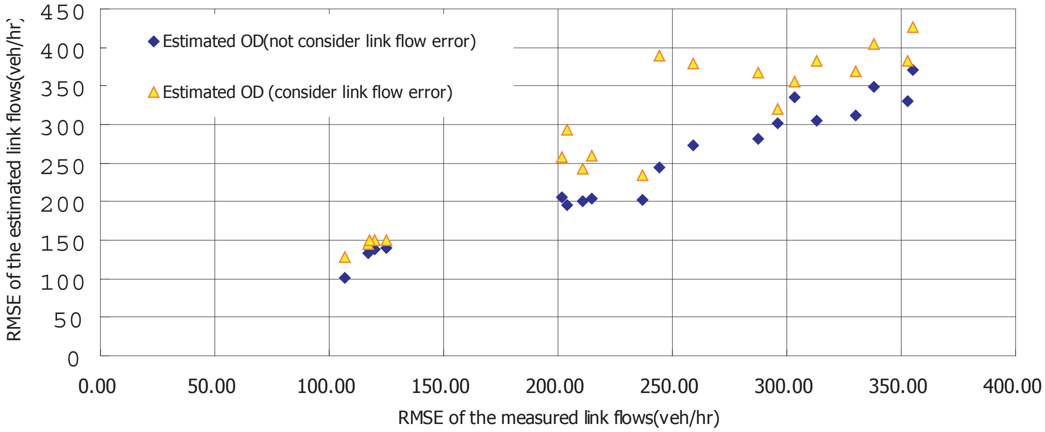


Figure 6.43 RMSE_Links for scenarios with different settings on the consideration of link flow errors

Similar observations can be made as in the last subsection, that the O-D estimation results are worse off when link measurement errors are explicitly taken into account. This is because, again, the algorithm always search the "corners" of a feasible region thus is likely to miss the "true" values lying in its interior.

6.7 The time-dependent case

The time-dependent LPFE models queuing by considering residual queues. Other than that, TD-LPFE is essentially the same as the static LPFE. In this section, we set up a time-dependent estimation scenario using the Dallas-Fort Worth network. The two-hour O-D demands used for static testing are now divided into 24 time intervals (each is 5-minute long). To illustrate the accumulation and dissipation of queues, these demands are assumed to be loaded onto network following a unimodal pattern as depicted in Figure 6.44.

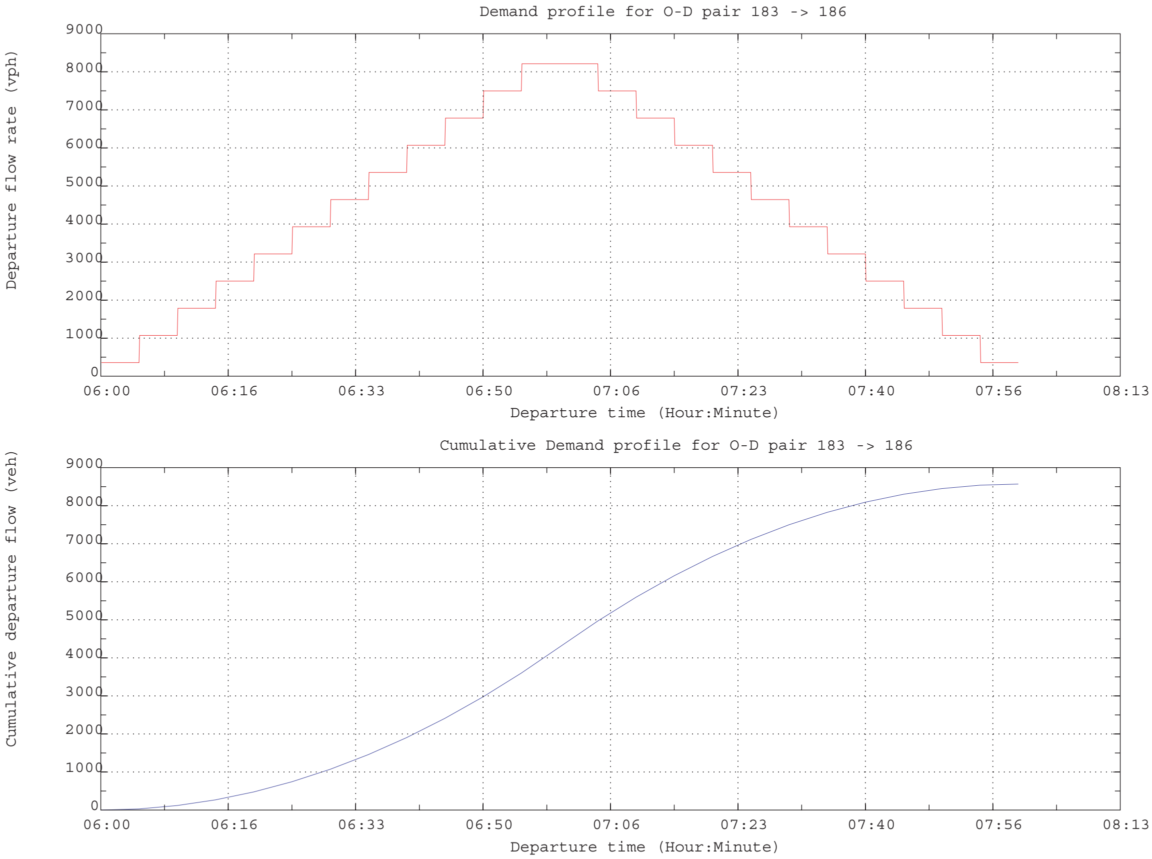


Figure 6.44 A time-dependent demand pattern

As before, we first solve a time-dependent logit traffic assignment model (TD_BSUE, c.f. Section 3.1.2) to produce time-dependent traffic counts. According to the observation made in Section 6.4.2, we only employ traffic counts on the freeway links and consider a historical O-D matrix which is generated by applying an up to 20% random perturbation over the "true" O-D matrix. Link RMSE at different time intervals are plotted in Figure 6.45. Clearly, it is easier for TD-LPFE to reproduce the observed link traffic counts when the demand level is low. The number of used paths is small in uncongested cases (as in the first few time intervals) and thus they are easy to be identified by a column generation procedure. When roads get congested, however, more paths were used and so the difficulty of satisfying all the constraints increases (i.e., it is easier to get a path set from column generation which differs than the "true" one). Of course, algorithmic parameters can be finely tuned to achieve a better match of link traffic counts, as described in Section 6.2. Such adjustment is not included in this experiment.

As reported in Figure 6.46, estimated O-D matrices at all time intervals shows significant improvements over the target matrices. A smaller RMSE implies that these estimated matrices are closer to the "true" matrices (statistically) compared to target matrices. Unlike the pattern of link RMSE, such improvements barely fluctuate across time. Further, a good portion of the total demand was captured in the estimation process (more than 92% in most cases), and the value of TDC keeps relatively stable at different time intervals.

Finally, Figure 6.47 shows the change of queue length with time on two selected links. As seen, the queue appears as temporal demands exceed road capacities (in this example, starting from time interval 9). These queues are passed to and processed in subsequent time intervals, and so we observed the gradual accumulation of queues on the links. When demand level drops back, the residual queues gradually shrink and at last completely disappear. Moreover, the evolution of queues on link 307 is reproduced quite accurately by TD-LPFE, as shown in Figure 6.47. However, on link 428, the mismatch between the estimated and "true" queue evolution pattern is significant. This is not surprising since link traffic counts are not perfectly reproduced from the estimation.

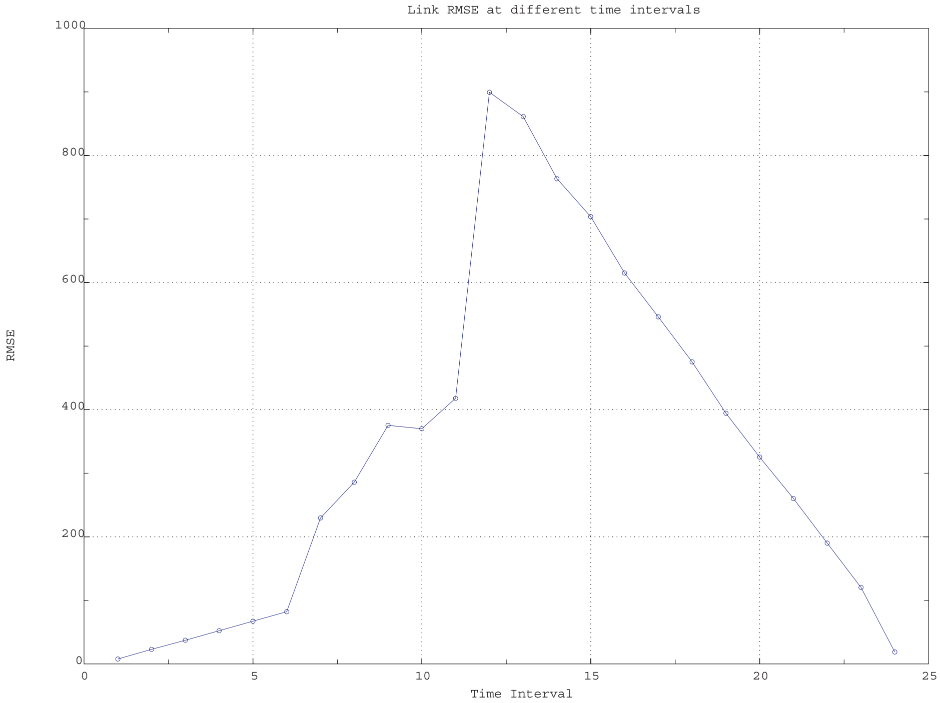


Figure 6.45 RMSE_Links at different time intervals

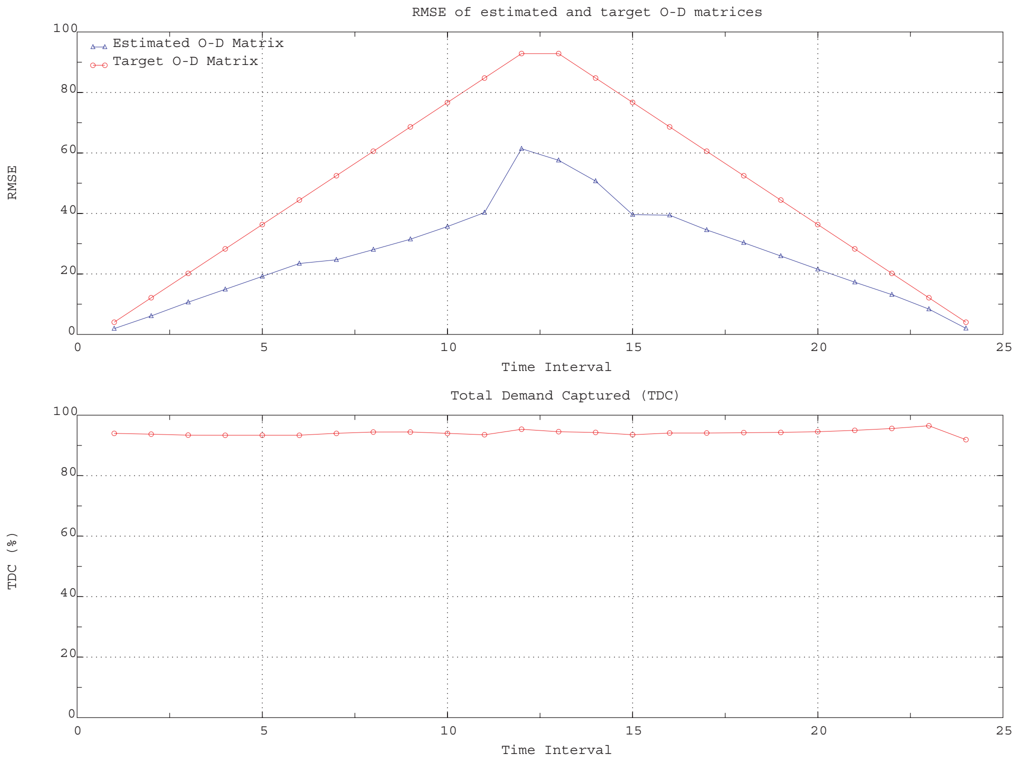


Figure 6.46 RMSE_ODs at different time intervals

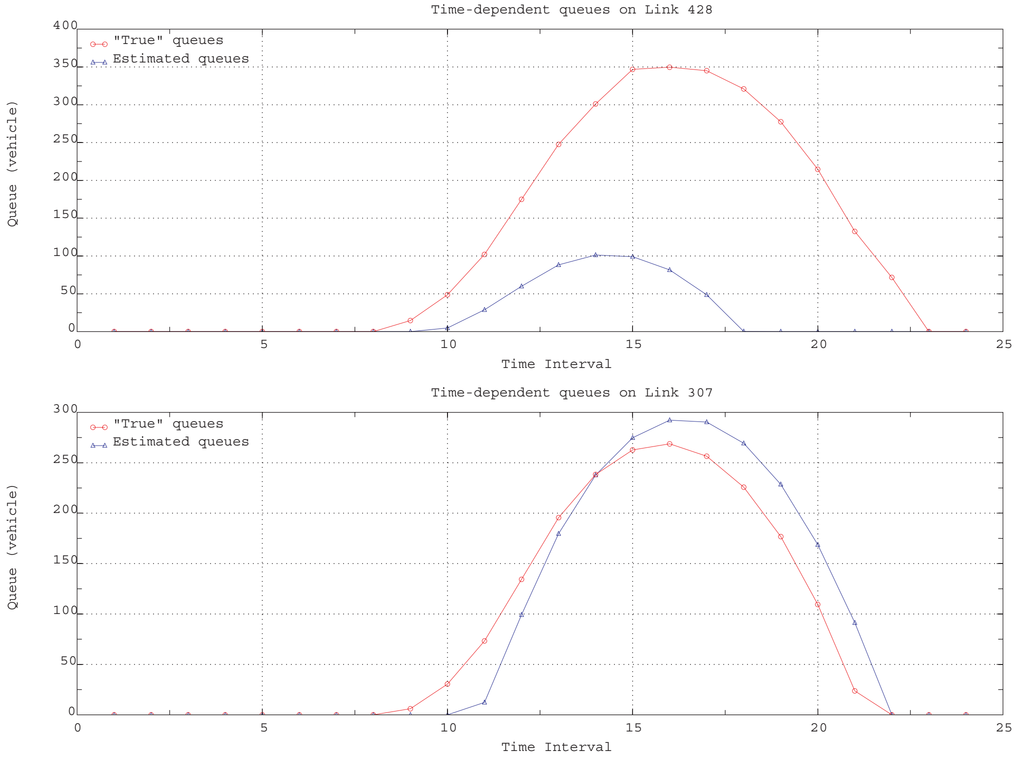


Figure 6.47 Accumulation and dissipation of queues on link 307 and 428

Chapter 7. Conclusions

Getting travel demands is the starting point of many transportation planning and operational applications. To supplement the traditional survey approaches, this research implements and tests a class of methods for estimating both steady-state and time-dependent travel demands from traffic surveillance data, which can be automatically collected at relatively low costs and provide up-to-date and time-dependent traffic information. The path flow estimator (PFE) considered in this research has several appealing features. First, it formulates the estimation problem as a one-level convex optimization problem and thus avoids the bi-level programming structure which requires to repeatedly solve traffic assignment problems and does not ensure global optimality and convergence. Second, PFE properly takes into account users' route choice behavior consistent with the stochastic user equilibrium. Finally, PFE provides a reasonable compromise to cope with traffic dynamics, i.e., decomposing a dynamic problem to a sequence of static problems through carrying queues over different times. This research presented several improvements over the original PFE and implemented the algorithms into a software tool with a friendly GIS-based user interface.

In this chapter, we summarize the major findings of this research, and provide some thoughts on the follow-up research work.

7.1 Major Findings Regarding the Performance of LPFE

Based on extensive numerical studies, we made the following observations and findings:

- Improper algorithmic parameter settings may lead to either poor estimation results or high computational cost. If default parameters do not produce satisfactory results, users are suggested try greater maximum number of iterations and maximum number of zigzagging iterations, combined with tighter accuracy and zigzagging requirements.

- For the basic network (typology) types, LPFE is capable of reproducing measured link flows and capturing total demands accurately, provided that all links are measured. For the *tree* network, the estimated O-D table can perfectly match the "true" O-D table. For *linear*, *ring* and *derived* networks, the estimated O-D table cannot reproduce the "true" O-D table mainly due to the existence of multiple path solutions. Since path solution is usually not unique in most networks, this type of error seems inevitable if only link flows are used as inputs.
- Information reflecting the spatial structure of travel demands (i.e., a historical or observed O-D table) greatly improves estimation quality. Even when traffic counts are available only on a small set of links (e.g., freeway links), LPFE can still produce satisfactory results if such structure information is well maintained in the input O-D table.
- The performance of LPFE is not sensitive to the errors contained in the input (historical) O-D table. However, as errors of the input O-D table increases, it becomes harder to reproduce the observed link traffic counts accurately. On the other hand, LPFE seems more sensitive to measurement errors in link counts. Nevertheless, LPFE can still make improvement over the input O-D table even if link measurements are subject to significant errors.
- Explicitly considering measurement errors does not produce better estimation results. This is mainly due to the fact that the algorithm always searches the corner points instead of the interior of the enlarged feasible region. Users therefore do not need to provide error bounds for the traffic counts.
- As shown in the time-dependent case, it becomes more difficult to accurately reproduce measured link traffic counts as a network gets more congested. However, the quality of estimated O-D trip rates is not substantially affected by congestion levels. Further, the formation and dissipation of queues are captured reasonably well by TD-LPFE.

7.2 Future Work

Our studies indicate that, besides the number of measurement locations, the location of the loop detectors and the network topology seem to substantially affect the quality of the estimates. Thus, some guidelines should be established to optimally pick measurement locations. Such a sensor location problem can be formulated in two ways, either seeking a deployment plan with a minimum number of sensors to achieve a given threshold of estimation quality, or given a number of sensors finding a deployment plan that optimize the estimates.

A major limitation of the implemented PFE is its capability of replicating realistic dynamic traffic phenomena. After all, time-dependent PFE is essentially a static model. The simplification employed in PFE, i.e. transforming a dynamic problem into a sequence of static problems, only works for relatively coarse time resolution (i.e., longer time intervals). Note that a trip is assumed to complete within the time interval it departs from the origin and this is true only when the corresponding time period is long enough. Only a true dynamic traffic flow model can remove this drawback. Nevertheless, in studies that need a finer time resolution, such as in a micro-simulation application, TD-LPFE can still be used to obtain an initial estimate of the time-dependent O-D trip tables, and use these estimates as the base O-D tables can speed up the convergence of the O-D estimators that come with the simulation tools. We have applied this procedure to the estimation of O-D tables for Paramics applications and obtained satisfactory results.

Bibliography

Ashok, K. (1996), Estimation and Prediction of Time-Dependent Origin-Destination Flows, Ph.d thesis, Massachusetts Institute of Technology, Cambridge, MA.

Ashok, K. & Ben-Akiva, M. (1993), `Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems', *In Proceedings of 12th International Symposium on the Theory of Traffic Flow and Transportation* pp. 465-484.

Ashok, K. & Ben-Akiva, M. (2000), `Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows', *Transportation Science* 34, 21-36.

Beckmann, M., McGuire, C. B. & Winsten, C. B. (1956), *Studies in the Economics of Transportation*, Yale University Press, New Haven, Connecticut.

Bell, M. G. H. (1983), `The estimation of an origin-destination matrix from traffic counts', *Transportation Science* 17, 198-217.

Bell, M. G. H. (1991*a*), `The estimation of origin-destination matrices by constrained generalized least squares', *Transportation Research* 25B, 13-22.

Bell, M. G. H. (1991*b*), `The real time estimation of origin-destination flows in the presence of platoon dispersion', *Transportation Research* 25B, 115-125.

Bell, M. G. H. & Grosso, S. (1998), `The path flow estimator as a network observer', *Traffic*

Engineering and Control pp. 540-549.

Bell, M. G. H. & Shield, C. M. (1995), `A log-linear model for path flow estimation', *Proceedings of the Fourth Internatial Conference on the Application of Advance technologies in Transporation Engineering, Capri* pp. 695-699.

Bell, M. G. H., Shield, C. M., Busch, F. & Kruse, K. (1997), `A stochastic user equilibrium path flow estimator', *Transportation Research* 5C, 197-210.

Brenninger-Gthe, M., Jrnsten, K. O. & Lundgren, J. T. (1989), `Estimation of origin-destination matrices from traffic counts using multi-objective programming formulations', *Transportation Research* 23B, 257-269.

Cascetta, E. (1984), `Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator', *Transportation Research* 18B, 289-299.

Cascetta, E., Inaudi, D. & Marquis, G. (1993), `Dynamic estimators of origin-destination matrices using traffic counts', *Transportation Science* 27, 363-373.

Chang, G.-L. & Tao, X. (1996), `Estimation of dynamic O-D distribution for urban network', *In Proceedings of 13th International Symposium on the Theory of Traffic Flow and Transportation* pp. 1-20.

Chang, G.-L. & Wu, J. (1994), `Recursive estimation of time-varying origin-destination flows from traffic counts in freeway corridors', *Transportation Research* 28B, 141-160.

Cremer, M. & Keller, H. (1981), `Dynamic identification of O-D flow from traffic counts at complex intersections', *In proceedings of 8th International Symposium on Transportation and Traffic Theory* .

Cremer, M. & Keller, H. (1984), `A systems dynamics approach to the estimation of entry and exit O-D flows', *In proceedings of 9th International Symposium on Transportation and Traffic Theory* .

Cremer, M. & Keller, H. (1987), `A new class of dynamic methods for the identification of origin-destination flows', *Transportation Research* 21B, 117-132.

Daganzo, C. F. & Sheffi, Y. (1977), `On stochastic models of traffic assignment', *Transportation Science* 11, 253-274.

Dial, R. B. (1971), `A probabilisitc multipath assignment model that obviates path enumeration', *Transportation Research* 5, 83-111.

Fisk, C. S. (1980), `Some developments in equilibrium traffic assignment', *Transportation Research* 14B, 243-255.

Fisk, C. S. (1988), `On combining maximum entropy trip matrix estimation with user optimal assignment', *Transportation Research* 22B, 66-79.

Fisk, C. S. & Boyce, D. E. (1983), `A note on trip matrix estimation from link traffic count data', *Transportation Research* 17B, 245-250.

Jornsten, K. & Nguyen, S. (1980), On the estimation of trip matrix from network data, Technical Report NITH-MAT-R79-36, Linkoping institute of technology, Linkoping, Sweden.

LeBlanc, L. & Farhangian, K. (1982), `Selection of a trip table which reproduces observed link flows', *Transportation Research* 22B, 83-88.

Lo, H. P., Zhang, N. & Lam, W. H. K. (1996), `Estimation of an origin-destination matrix with random link choice proportions: a statistical approach', *Transportation Research* 30B, 309-324.

McNeil, S. & Hendrickson, C. (1985), `A regression formulation of the matrix estimation problem', *Transportation Science* 19, 278-292.

Nguyen, S. (1977), Estimating an OD matrix from network data: a network equilibrium approach, Technical Report 60, University of Montreal.

Nguyen, S. (1984), *Transportation Planning Models*, Edited by M. Florian, Elsevier Science Publishers, Amsterdam, chapter Estimating origin-destination matrices from observed flows,pp. 363-380.

Nie, Y. & Lee, D.-H. (2002), `An uncoupled method for the equilibrium-based linear path flow estimator for origin-destination trip matrices', *Transportation Research Record* 1783, 72-79.

Nie, Y., Zhang, H. M. & Recker, W. W. (2005), `Inferring origin-destination trip matrices with a decoupled gls path flow estimator', *Transportation Research* 39B, 497-518.

Nihan, N. L. & Davis, G. A. (1987), `Recursive estimation of origin-destination matrices from input/output counts', *Transportation Research* 21B, 149-163.

Nihan, N. L. & Davis, G. A. (1989), `Application of prediction-error minimisation and maximum likelihood to estimate intersection o-d matrices from traffic counts', *Transportation Science* 23, 77-90.

Nihan, N. L. & Hamed, M. M. (1992), `Fixed-point approach to estimating freeway origin-destination matrices and the e®ect of erroneous data on estimate precision', *Transportation Research Record* 1357, 18-28.

Okutani, I. (1987), `The Kalman filtering approach in some transportation and traffic problems', *In Proceedings of 10th International Symposium on Transportation and Traffic Theory* pp. 397-416.

Powell, W. B. & Sheffi, Y. (1982), `The convergence of equilibrium with predetermined step sizes', *Transportation Science* 16, 45-55.

Sheffi, Y. (1985), *Urban Transportation Networks: Equilibrium Analysis with Mathematical*

Programming Methods, Prentice Hall, Englewood cli®s, NJ.

Sherali, H. D., Arora, N. & Hobeika, A. G. (1997), `Parameter optimization methods for estimating dynamic origin-destination trip tables', *Transportation Research* 31B, 141-157.

Sherali, H. D. & Park, T. (2001), `Estimation of dynamic origin-destination trip tables for a general network', *Transportation Research* 35B, 217-235.

Sherali, H. D., Sivanandan, R. & Hobeika, A. G. (1994), `A linear programming approach for synthesizing origin-destination trip tables from link traffic volumes', *Transportation*

Research 28B, 213-233. Turnquist, M. & Gur, Y. (1979), `Estimation of trip tables from observed link volumes', *Transportation Research Record* 730, 295-303.

Van-Zuylen, J. H. & Willumsen, L. G. (1980), `The most likely trip matrix estimated from traffic counts', *Transportation Research* 14B, 281-293.

Wardrop, J. G. (1952), `Some theoretical aspects of road traffic research', *Proceedings of the Institute of Civil Engineers, Part II* 1, 325-378.

Willumsen, K. G. (1984), `Estimating time-dependent trip matrices from traffic counts', *In Proceedings of the 9th International Symposium on Transportation and Traffic Theory* pp. 397-411.

Willumsen, L. G. (1981), `Simplified transport models based traffic counts', *Transportation* 10, 257-278.

Wu, J. & Chang, G.-L. (1996), `Estimation of time-varying origin-destination distributions with dynamic screenline flows', *Transportation Research* 30B, 277-290.

Yang, H. (1995), `Heuristic algorithms for the bilevel origin-destination matrix estimation problem', *Transportation Research* 29B, 231-242.

Yang, H. & Meng, Q. (1998), `Departure time, route choice and congestion toll in a queuing network with elastic demand', *Transportation Research* 32(4), 247-260.

Yang, H., Meng, Q. & Bell, M. G. H. (2001), `Simultaneous estimation of the origin-destination matrices and travel-cost coefficient for congested networks in a stochastic user equilibrium', *Transportation Science* 35, 107-123.

Yang, H., Sasaki, T., Iida, Y. & Asakura, Y. (1992), `Estimation of origin-destination matrices from traffic counts on congested networks', *Transportation Research* 26B, 417-433.

Appendix I. File formats

In most cases users will prepare input files through the graphical interface. However, the format of input files is introduced here for the reference purpose. All input files are named by a base plus a suffix. For narrative convenience we assume the base is “project” hereafter.

I.1 Input files for assignment

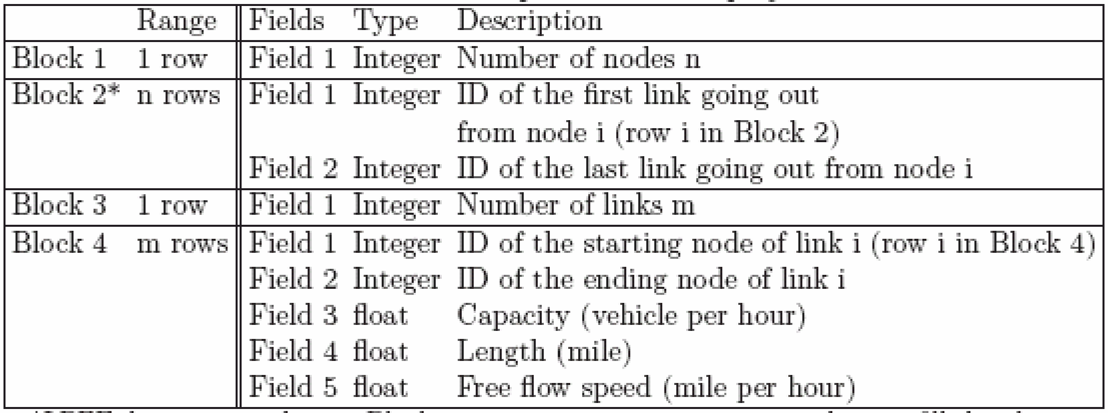
1. Network files. Two types of network format are accepted: FORT and DANET2.

   a FORT type network. In this format, a network is described by two files: project.1 and project.2

   (1) project.1 describes network topology as explained in Table I.1. An example is given in Figure I.1.

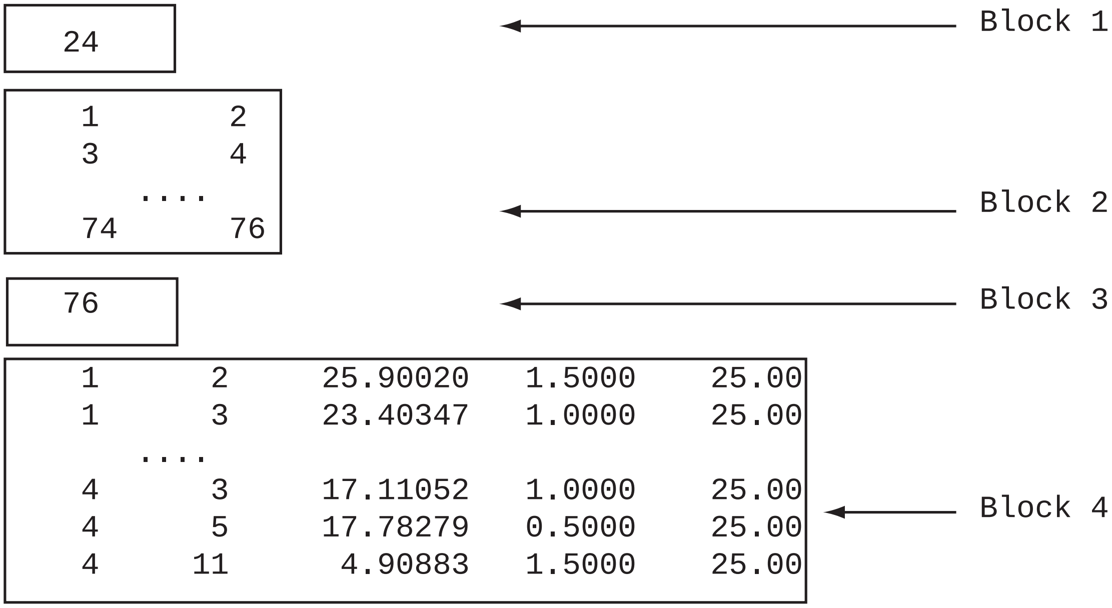
   (2) project.2 describes O-D information, as described in Table I.2. An example is shown in Figure I.2.

Table I.1 A description of the file project.1



	Range	Fields	Type	Description
Block 1	1 row	Field 1	Integer	Number of nodes n
Block 2*	n rows	Field 1	Integer	ID of the first link going out from node i (row i in Block 2)
		Field 2	Integer	ID of the last link going out from node i
Block 3	1 row	Field 1	Integer	Number of links m
Block 4	m rows	Field 1	Integer	ID of the starting node of link i (row i in Block 4)
		Field 2	Integer	ID of the ending node of link i
		Field 3	float	Capacity (vehicle per hour)
		Field 4	float	Length (mile)
		Field 5	float	Free flow speed (mile per hour)

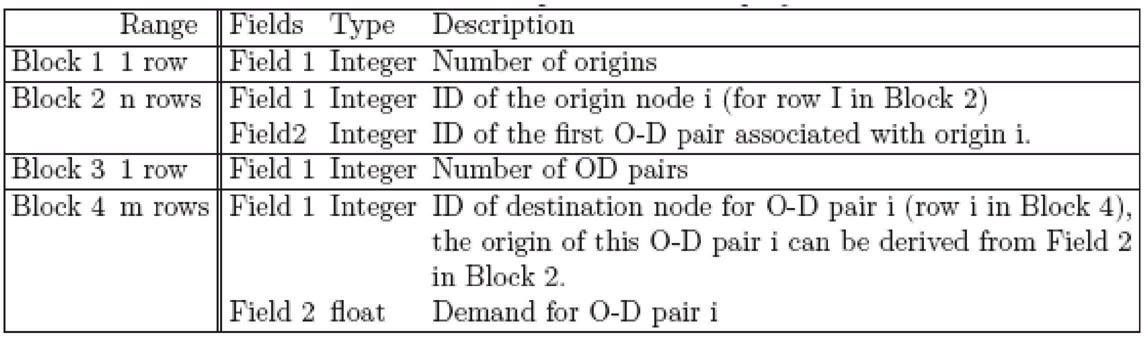
* LPFE does not use data in Block 2, so users can put any integers there to fill the places.



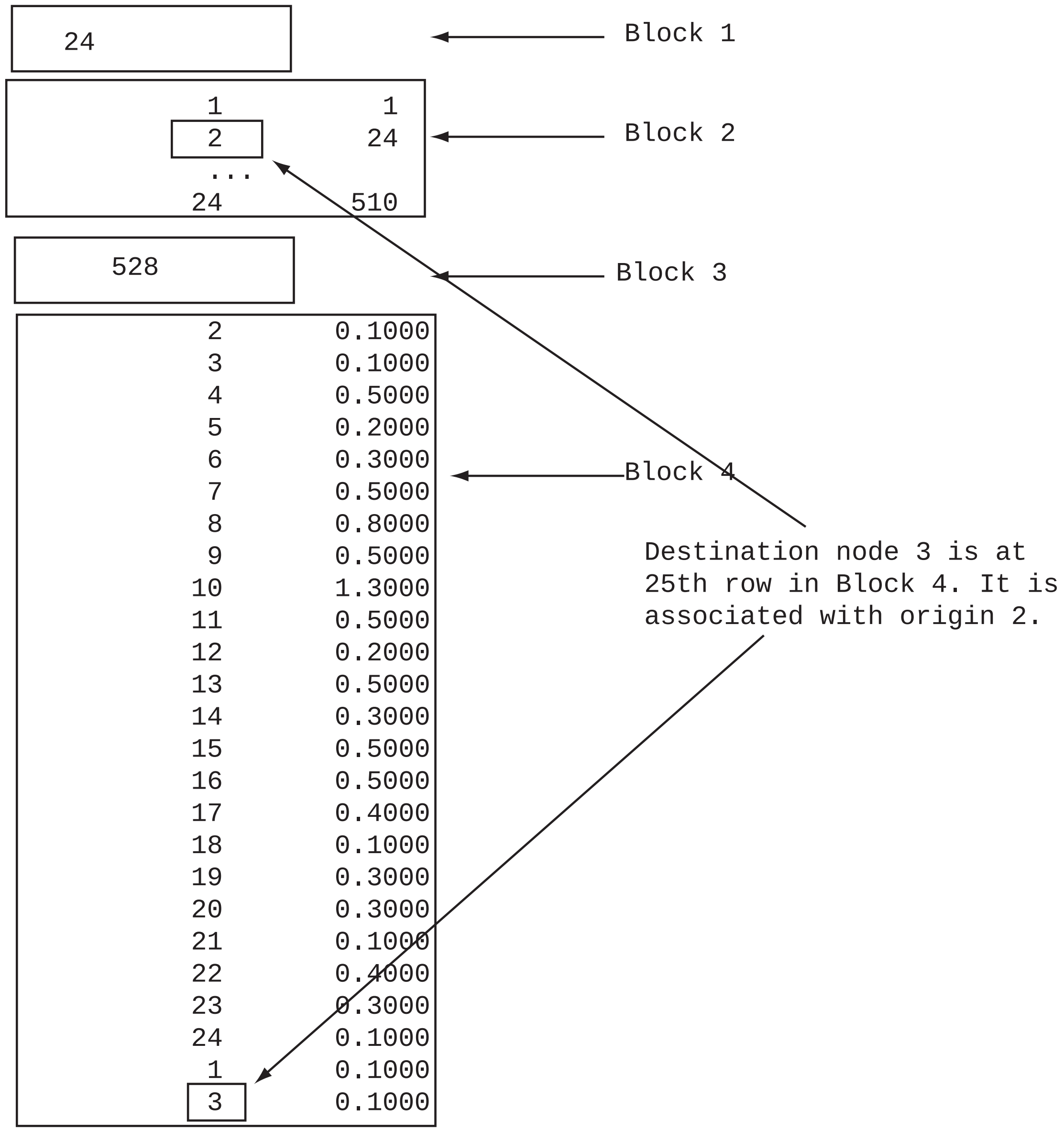
```
24

   1       2
   3       4
      ....
   74      76

76

   1      2     25.90020   1.5000    25.00
   1      3     23.40347   1.0000    25.00
      ....
   4      3     17.11052   1.0000    25.00
   4      5     17.78279   0.5000    25.00
   4     11      4.90883   1.5000    25.00
```

Figure I.1 An example of the file project.1

Table I.2 A description of the file project.2



	Range	Fields	Type	Description
Block 1	1 row	Field 1	Integer	Number of origins
Block 2	n rows	Field 1	Integer	ID of the origin node i (for row I in Block 2)
		Field2	Integer	ID of the first O-D pair associated with origin i.
Block 3	1 row	Field 1	Integer	Number of OD pairs
Block 4	m rows	Field 1	Integer	ID of destination node for O-D pair i (row i in Block 4), the origin of this O-D pair i can be derived from Field 2 in Block 2.
		Field 2	float	Demand for O-D pair i



```
24

        1         1
        2        24
          ...
       24       510

528

        2    0.1000
        3    0.1000
        4    0.5000
        5    0.2000
        6    0.3000
        7    0.5000
        8    0.8000
        9    0.5000
       10    1.3000
       11    0.5000
       12    0.2000
       13    0.5000
       14    0.3000
       15    0.5000
       16    0.5000
       17    0.4000
       18    0.1000
       19    0.3000
       20    0.3000
       21    0.1000
       22    0.4000
       23    0.3000
       24    0.1000
        1    0.1000
        3    0.1000
```

Destination node 3 is at 25th row in Block 4. It is associated with origin 2.

Figure I.2 An example of the file project.2

b DANET2 type network. This format is original designed to address the need of dynamic application, which describes a network by four files: project.net, project.lin, project.nod and project.odp.

(1) project.net provides basic parameters, as described in Table I.3. See Figure I.3 for an example.

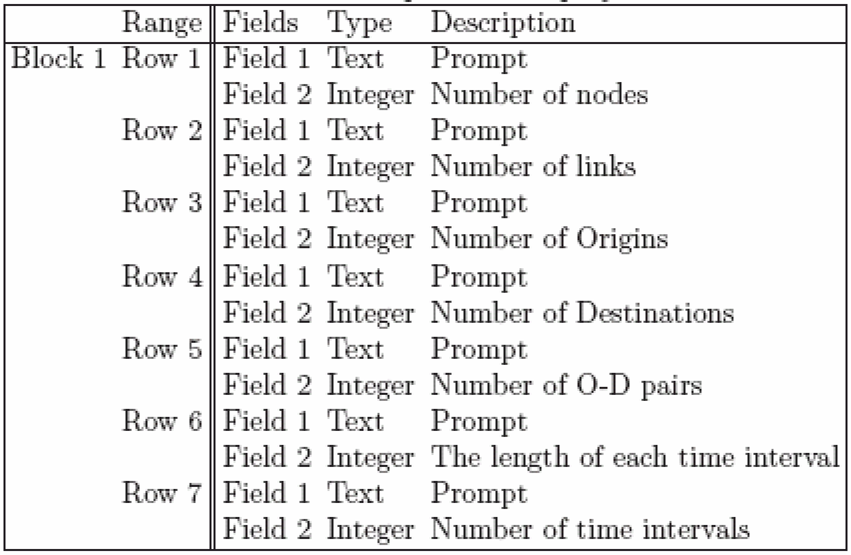
(2) project.lin describes link information, as described in Table I.4. See Figure I.4

for an example.

(3) project.nod describes node information, as described in Table I.5. See Figure I.5 for an example.

(4) project.odp describes O-D information, as described in Table I.6. See Figure I.6 for an example.

Table I.3 A description of file project.net

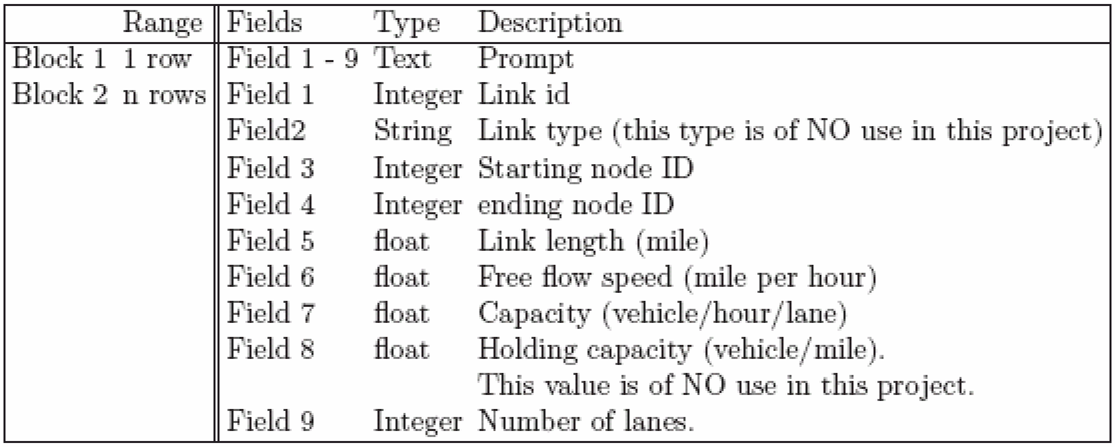


	Range	Fields	Type	Description
Block 1	Row 1	Field 1	Text	Prompt
		Field 2	Integer	Number of nodes
	Row 2	Field 1	Text	Prompt
		Field 2	Integer	Number of links
	Row 3	Field 1	Text	Prompt
		Field 2	Integer	Number of Origins
	Row 4	Field 1	Text	Prompt
		Field 2	Integer	Number of Destinations
	Row 5	Field 1	Text	Prompt
		Field 2	Integer	Number of O-D pairs
	Row 6	Field 1	Text	Prompt
		Field 2	Integer	The length of each time interval
	Row 7	Field 1	Text	Prompt
		Field 2	Integer	Number of time intervals

```
Number of nodes:                     130
Number of links:                     390
Number of origins:                     5
Number of destinations:               15
Number of OD Pairs:                  210
Unit Simulation Time (sec):          300
Assignment Horizon (in unit time):     6
```

Figure I.3 An example of the file project.net

Table I.4 A description of the file project.lin

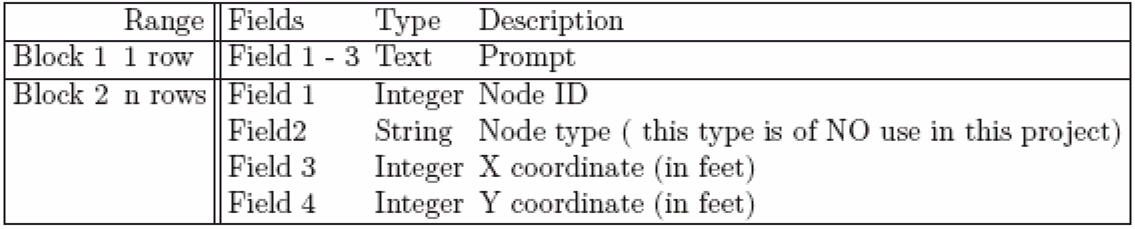


Range		Fields	Type	Description
Block 1	1 row	Field 1 - 9	Text	Prompt
Block 2	n rows	Field 1	Integer	Link id
		Field2	String	Link type (this type is of NO use in this project)
		Field 3	Integer	Starting node ID
		Field 4	Integer	ending node ID
		Field 5	float	Link length (mile)
		Field 6	float	Free flow speed (mile per hour)
		Field 7	float	Capacity (vehicle/hour/lane)
		Field 8	float	Holding capacity (vehicle/mile). This value is of NO use in this project.
		Field 9	Integer	Number of lanes.

```
ID   Type From   To LEN(m) FFS(m/h) Cap(v/h) RHOJ(v/m) Lane
 1  LWRLK    1    2      1       25     1323       171    1
 2  LWRLK    1   11      1       45     1560       164    2
 3  LWRLK    2    1      1       65     1976       175    2
 4  LWRLK    2    3      1       35     1394       175    2
 5  LWRLK    2   12      1       25     1240       176    2
 6  LWRLK    3    2      1       65     2252       178    4
 7  LWRLK    3    4      1       55     1859       176    2
 8  LWRLK    3   13      1       45     1629       175    1
 9  LWRLK    4    3      1       35     1394       171    2
10  LWRLK    4    5      1       55     1582       166    1
11  LWRLK    4   14      1       45     1473       173    3
12  LWRLK    5    4      1       55     1622       166    3
13  LWRLK    5    6      1       25     1142       180    2
14  LWRLK    5   15      1       45     1491       175    3
```

Figure I.4 An example of the file project.lin

Table I.5 A description of the file project.nod

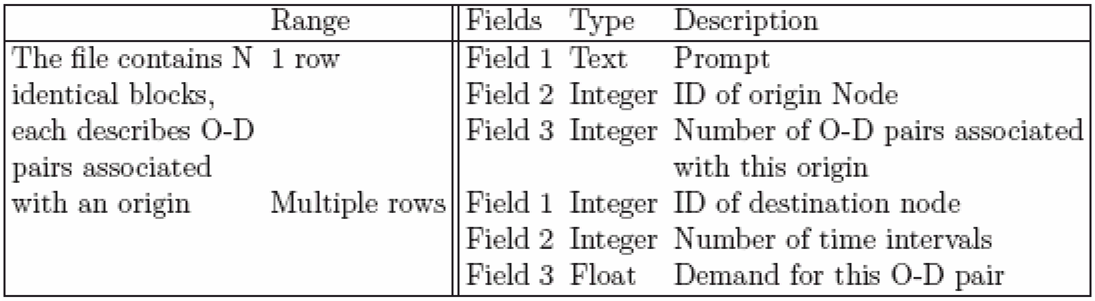


Range	Fields	Type	Description
Block 1 1 row	Field 1 - 3	Text	Prompt
Block 2 n rows	Field 1	Integer	Node ID
	Field2	String	Node type (this type is of NO use in this project)
	Field 3	Integer	X coordinate (in feet)
	Field 4	Integer	Y coordinate (in feet)

```
ID   Type  Xcord  Ycord
 1  FWJCT   5280   5280
 2  FWJCT  10560   5280
 3  FWJCT  15840   5280
 4  FWJCT  21120   5280
 5  FWJCT  26400   5280
 6  FWJCT  31680   5280
 7  FWJCT  36960   5280
 8  FWJCT  42240   5280
 9  FWJCT  47520   5280
10  FWJCT  52800   5280
11  FWJCT   5280  10560
12  FWJCT  10560  10560
13  FWJCT  15840  10560
```

Figure I.5 An example of the file project.nod

Table I.6 The description of the file project.odp



	Range	Fields	Type	Description
The file contains N identical blocks, each describes O-D pairs associated with an origin	1 row	Field 1	Text	Prompt
		Field 2	Integer	ID of origin Node
		Field 3	Integer	Number of O-D pairs associated with this origin
	Multiple rows	Field 1	Integer	ID of destination node
		Field 2	Integer	Number of time intervals
		Field 3	Float	Demand for this O-D pair

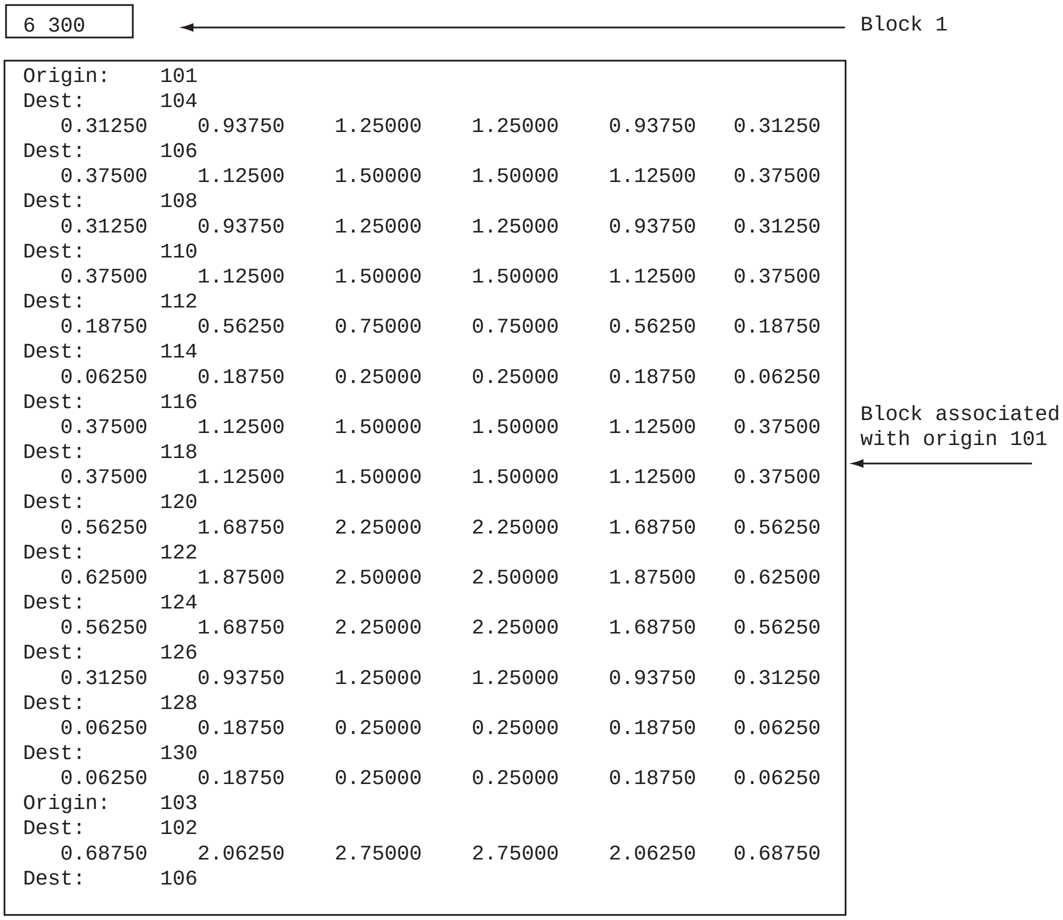
```
Origin:    101    14
           104     6     5.0000
           106     6     6.0000
           108     6     5.0000
           110     6     6.0000
           112     6     3.0000
           114     6     1.0000
           116     6     6.0000
           118     6     6.0000
           120     6     9.0000
           122     6    10.0000
           124     6     9.0000
           126     6     5.0000
           128     6     1.0000
           130     6     1.0000
Origin:    103    14
           102     6    11.0000
           106     6    19.0000
           108     6    15.0000
           110     6    19.0000
           112     6    11.0000
           114     6     3.0000
```

Figure I.6 An example of the file project.odp

2. Time-dependent demand files project.dmd. Figure I.7 provides an example while Table I.7 explains how to read it.

Table I.7 A description of the file project.dmd

	Range	Fields	Type	Description
Block 1	1 row	Field 1	Integer	Number of time intervals
		Field 2	Integer	The length of each time interval (in seconds)
There are multiple identical blocks after the first block, each corresponds to an origin	Row 1	Field 1	Text	Prompt
		Field 2	Integer	ID of origin node
	Row 2	Field 1	Text	Prompt
		Field 2	Integer	ID destination node
				Each field i corresponds to the demand of the O-D pair at the time interval i (they can be put in more than one row)



```
6 300
```

Block 1

```
Origin:    101
Dest:      104
   0.31250    0.93750    1.25000    1.25000    0.93750   0.31250
Dest:      106
   0.37500    1.12500    1.50000    1.50000    1.12500   0.37500
Dest:      108
   0.31250    0.93750    1.25000    1.25000    0.93750   0.31250
Dest:      110
   0.37500    1.12500    1.50000    1.50000    1.12500   0.37500
Dest:      112
   0.18750    0.56250    0.75000    0.75000    0.56250   0.18750
Dest:      114
   0.06250    0.18750    0.25000    0.25000    0.18750   0.06250
Dest:      116
   0.37500    1.12500    1.50000    1.50000    1.12500   0.37500
Dest:      118
   0.37500    1.12500    1.50000    1.50000    1.12500   0.37500
Dest:      120
   0.56250    1.68750    2.25000    2.25000    1.68750   0.56250
Dest:      122
   0.62500    1.87500    2.50000    2.50000    1.87500   0.62500
Dest:      124
   0.56250    1.68750    2.25000    2.25000    1.68750   0.56250
Dest:      126
   0.31250    0.93750    1.25000    1.25000    0.93750   0.31250
Dest:      128
   0.06250    0.18750    0.25000    0.25000    0.18750   0.06250
Dest:      130
   0.06250    0.18750    0.25000    0.25000    0.18750   0.06250
Origin:    103
Dest:      102
   0.68750    2.06250    2.75000    2.75000    2.06250   0.68750
Dest:      106
```

Block associated with origin 101

Figure I.7 An example of the file project.dmd

3. Algorithm parameter file project.alg. This input is optional. If ignored, default values will be used. Changing parameters in this file may significantly affect the computation results. To obtain a better converged solution (sometimes users may find the estimated link flows do not perfectly match observed values, or estimated O-D flows do not match target O-D flows. In these case, the solution is considered as not well converged.), a general resolution is to increase maximum allowed iterations (but in general no more 1,000), reduce accuracy, increase maximum allowed bad iterations, increase maximum allowed zigzagging iterations, or reducing zigzagging criterion. Note that the running time may grow up quickly as the above adjustments are made for large networks. Figure I.8 shows an example of the project.alg file.

```
Maximum allowed iterations(1~1000000): 100
Accuracy(1e-016~1): 0.001
Minimum allowed stepsize(>=1e-009): 0.0001
Desirable Line search accuracy(1e-009~1): 0.001
Maximum allowed line search per iteration(1~1000): 10
Maximum allowed bad iterations(1~1000): 5
Zigzagging criterion(1e-016~1): 0.01
Maximum allowed zigzagging iterations(1~1000): 5
```

Figure I.8 An example of the file project.alg

I.2 Input files for O-D estimation

1. Network files: just same as those required in Assignment. Note that O-D file is also needed to provide the structure of O-D matrix. Particularly, O-D demands specified O-D files (project.2 in FORT and project.odp in DANET2) will be regarded as "true values".
2. Time-dependent O-D file project.dmd. It is optional in O-D Estimation and used to provide true values of time-dependent O-D demands.
3. Algorithm parameter file project.alg, same as those required in Assignment.
4. Observation files, consisting of project.obs, project.mlt, project.tar, project.ler,

project.ter

(1) project.obs provides link traffic counts. This file is REQUIRED for any O-D estimation problem. See Figure I.9 for an example and Table I.8 for explanation.

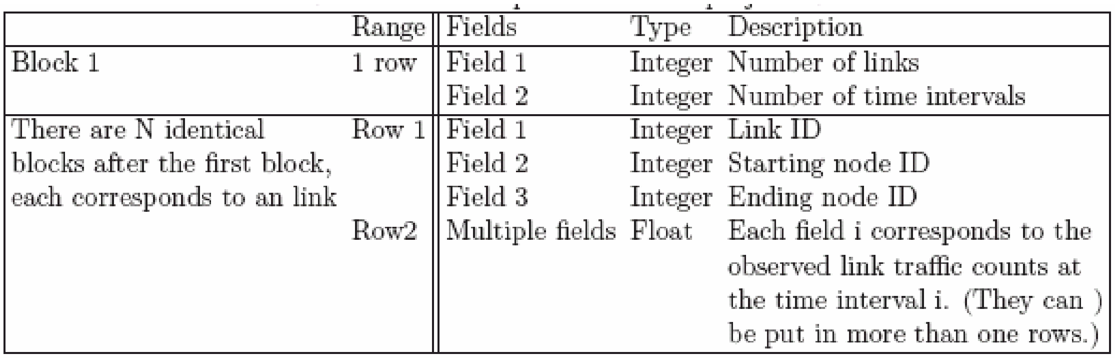
(2) project.mlt provides measured link travel times. This file is optional. If ignored, travel times on measured links may be computed from specified link performance function. The organization of this file is similar to project.obs thus not repeated. (the only difference is the traffic counts in .obs is replaced by measured travel times in .mlt.)

(3) project.tar provides either historical O-D demands or observed O-D data. This file is optional. See Figure I.10 for an example and Table I.9 for a description.

(4) project.ler specifies the error bounds for link traffic observations. This file is optional. See Table 10 for a description.

(5) project.ter specifies the error bounds for O-D observations. This file is optional. The HOMO format of NetName.ter is same as project.ler, thus its description is ignored here.

Table I.8 A description of the file project.obs



	Range	Fields	Type	Description
Block 1	1 row	Field 1	Integer	Number of links
		Field 2	Integer	Number of time intervals
There are N identical blocks after the first block, each corresponds to an link	Row 1	Field 1	Integer	Link ID
		Field 2	Integer	Starting node ID
		Field 3	Integer	Ending node ID
	Row2	Multiple fields	Float	Each field i corresponds to the observed link traffic counts at the time interval i. (They can) be put in more than one rows.)

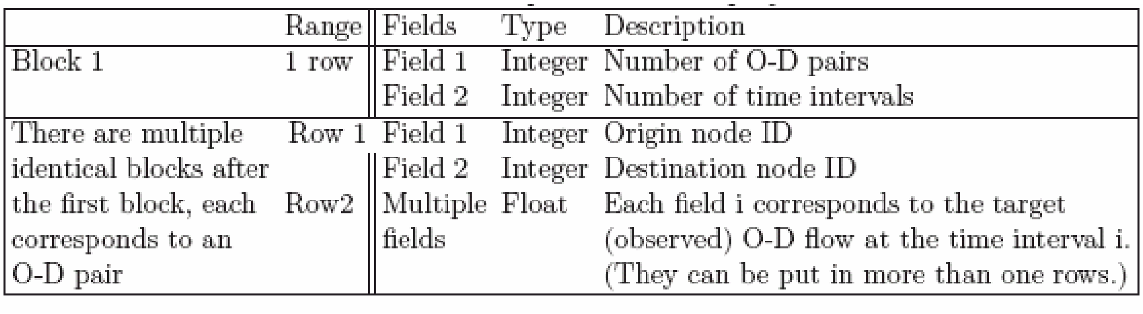
Block 1

Block associated with link 1

```
271     6
    1     1     2
     9.54      28.62      38.16      38.16      28.62      9.54
    2     1    11
    40.26     120.78     161.04     161.04     120.78      40.26
    3     2     1
    60.75     182.25        243        243     182.25      60.75
    4     2     3
    10.08      30.24      40.32      40.32      30.24      10.08
    6     3     2
   57.375    172.125      229.5      229.5    172.125     57.375
    7     3     4
    30.45      91.35      121.8      121.8      91.35      30.45
    8     3    13
  21.8875    57.4225      75.19      75.19    57.4225    21.8875
    9     4     3
    36.26     106.82      142.1      142.1     106.82      36.26
   10     4     5
   22.275     66.825       89.1       89.1     66.825     22.275
   11     4    14
     5.28      15.84      21.12      21.12      15.84       5.28
```

Figure I.9 An example of the file project.obs

Table I.9 A description of the file project.tar

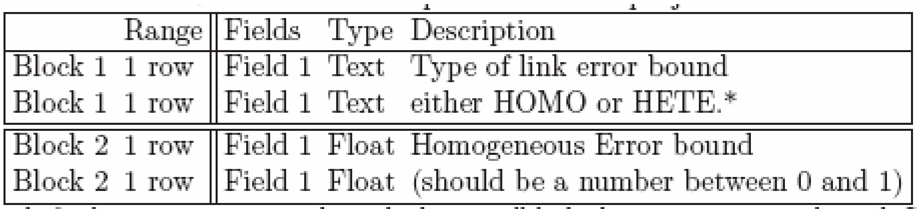


	Range	Fields	Type	Description
Block 1	1 row	Field 1	Integer	Number of O-D pairs
		Field 2	Integer	Number of time intervals
There are multiple identical blocks after the first block, each corresponds to an O-D pair	Row 1	Field 1	Integer	Origin node ID
		Field 2	Integer	Destination node ID
	Row2	Multiple fields	Float	Each field i corresponds to the target (observed) O-D flow at the time interval i. (They can be put in more than one rows.)

```
 210   6                                                   Block 1

   101   104
      3.75      11.25          15          15      11.25        3.75   Block assoicate
   101   106                                                            diwth O-D pair
       4.5       13.5          18          18       13.5         4.5   101 - 104
   101   108
      3.75      11.25          15          15      11.25        3.75
   101   110
       4.5       13.5          18          18       13.5         4.5
   101   112
      2.25       6.75           9           9       6.75        2.25
   101   114
      0.75       2.25           3           3       2.25        0.75
   101   116
       4.5       13.5          18          18       13.5         4.5
```

Figure I.10 An example of the file project.tar

Table I.10 A description of the file project.ler



	Range	Fields	Type	Description
Block 1	1 row	Field 1	Text	Type of link error bound
Block 1	1 row	Field 1	Text	either HOMO or HETE.*
Block 2	1 row	Field 1	Float	Homogeneous Error bound
Block 2	1 row	Field 1	Float	(should be a number between 0 and 1)

*HOMO stands for homogeneous error bound, that is, all links have a same error bound. HETE stands for heterogeneous, meaning that each link has different error bound. The current version does not support HETE error bound.

Appendix II

Visual PFE - TD®

Time Dependent

Visual PFE - TD® 1.0

User Manual

June 2006

Visual PFE and Visual PFE – TD are authored at the Utah State University. Copyright application for Visual PFE and Visual PFE-TD is currently in process. Please do not quote or reference the names without the authors' permission.

Table of Contents

List of Figures

List of Tables

Introduction

Welcome to a new dimension in OD estimation.

Visual PFE - TD is a special edition of Visual PFE, an integrated software suite that combines the Path Flow Estimator (PFE) with other software components to facilitate the estimation, visualization, and refinement of Origin-Destination (OD) trip tables with user-friendly Graphical User Interfaces (GUI).

The Logit Path Flow Estimator (LPFE) was developed at the Traffic Lab of University of California Davis. LPFE facilitates the estimation of time-dependent OD tables. LPFE operates under the DOS command prompt with no GUIs and no graphical presentation of the output data. To expand the capability of LPFE, this special edition of Visual PFE is created to integrate LPFE with components of Visual PFE.

With Visual PFE - TD, users can:

- Run LPFE with GUIs
- Convert the estimated OD tables to Microsoft Excel Files
- Change the colors and zoom levels of the OD table cells
- Interactively display and query OD desire lines
- Interactively display and query paths between any pair of OD
- Convert the PFE outputs to GIS files (ESRI shapefiles)
- Create thematic maps of network links and traffic analysis zones
- Generate diagnostic scatter plots
- Link the scatter plots to the network for identification of outliers
- Create and edit LPFE networks as text files
- Compare different scenarios

How to Use this Manual?

This manual is designed to explain various functions of the software, including instructions for software installation. Users of the software are referred to the *LPFE User Manual* for technical details about the LPFE program and how to prepare the input data. The *LPFE User Manual* can be found in the program folder of LPFE.

After installing the program, users are advised to go through the Introduction and the Graphical User Interfaces sections for knowledge of program functions.

To get an advanced knowledge of the software, users can go through the Technical Specification section of the manual to get an overview of the software components used to build the Visual PFE – TD.

Installing Visual PFE – TD

Before installing Visual PFE – TD, the LPFE program needs to be installed first. To install LPFE, run **LPFE_Setup.msi**. Follow the instructions of the set up program to complete the installation.

After LPFE is installed, in the folder where the program files are stored, there is a folder called Bin. The Bin folder contains the LPFE executable and other Dynamic Linking library (DLL) files. Before running LPFE from Visual PFE – TD, all of the files in the bin folder need to be copied and pasted to the data folder where the LPFE input network text files are stored.

To install Visual PFE, the computer needs to have the Microsoft .NET framework. Many of the Microsoft Windows programs are developed on the .NET framework. Therefore, for a computer running Microsoft Windows, it is likely that the .NET framework is already installed. If the users can not be sure if the computer has the .NET framework or not, an installation program **dotnetfx.exe** is included in the Visual PFE – TD installation CD. Running the program will install the .NET framework on the target computer.

After installing the .NET framework, users can proceed to run the Setup.exe program and follow the setup instruction to complete the installation.

Graphical User Interfaces

Visual PFE - TD is developed with Microsoft Visual Basic.NET. It implements Microsoft's Multiple Document Interface (MDI) software architecture, within which multiple windows forms can be created and viewed within the main window. The main window is referred to as the MDI parent form and each sub-window opened within the MDI parent is called a MDI child form. With Visual PFE, a MDI child form can take on the form of a map, a table, a graphic plot, or a text document. The program functions are controlled using Graphical User Interfaces (GUI).

Multiple Document Interface

When multiple result windows are opened in Visual PFE - TD, the titles of the MDI child windows are formatted in a way that results of the same network and time period can be identified. Figure 1 shows such an example, in which the titles of the OD table, scatter plot, path table, and report all include the name and the file path of the network (i.e., the title of the fist window).

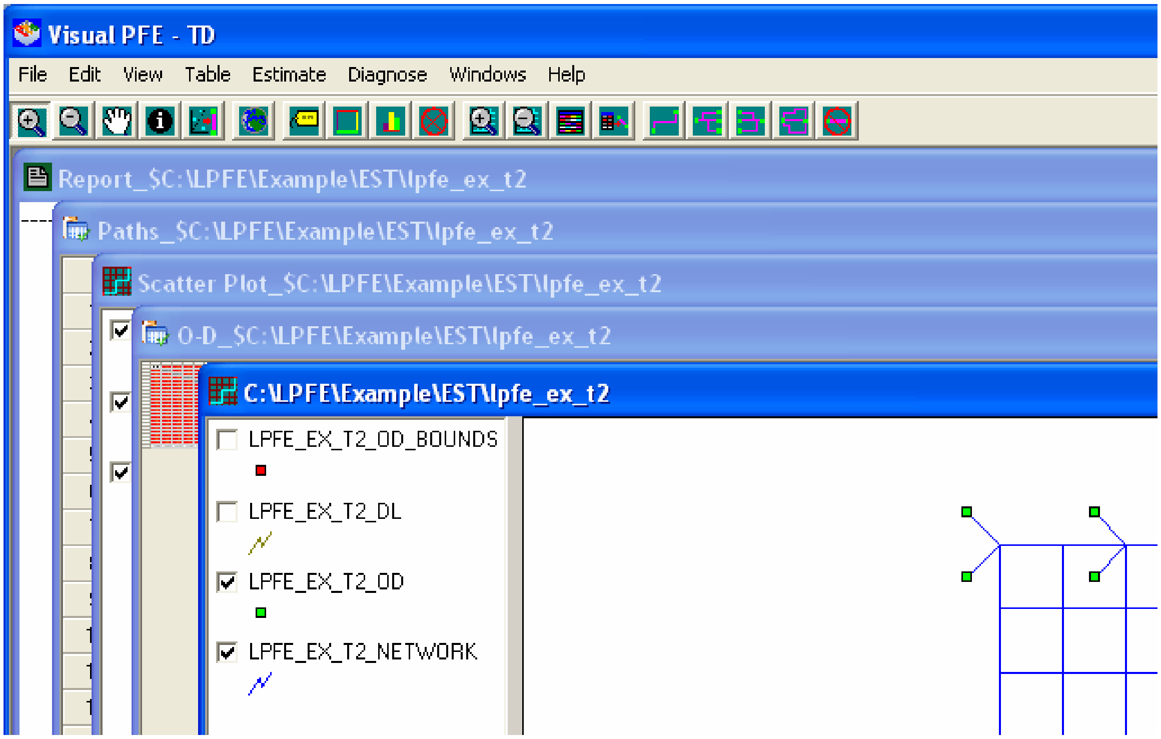

Visual PFE - TD
File Edit View Table Estimate Diagnose Windows Help
Report_$C:\LPFE\Example\EST\lpfe_ex_t2
Paths_$C:\LPFE\Example\EST\lpfe_ex_t2
Scatter Plot_$C:\LPFE\Example\EST\lpfe_ex_t2
O-D_$C:\LPFE\Example\EST\lpfe_ex_t2
C:\LPFE\Example\EST\lpfe_ex_t2
□ LPFE_EX_T2_OD_BOUNDS
□ LPFE_EX_T2_DL
☑ LPFE_EX_T2_OD
☑ LPFE_EX_T2_NETWORK

Figure 1 Multiple Document Interface

The Main Window

The Main Window is the MDI parent form of Visual PFE- TD. Figure 2 shows the program's main window.

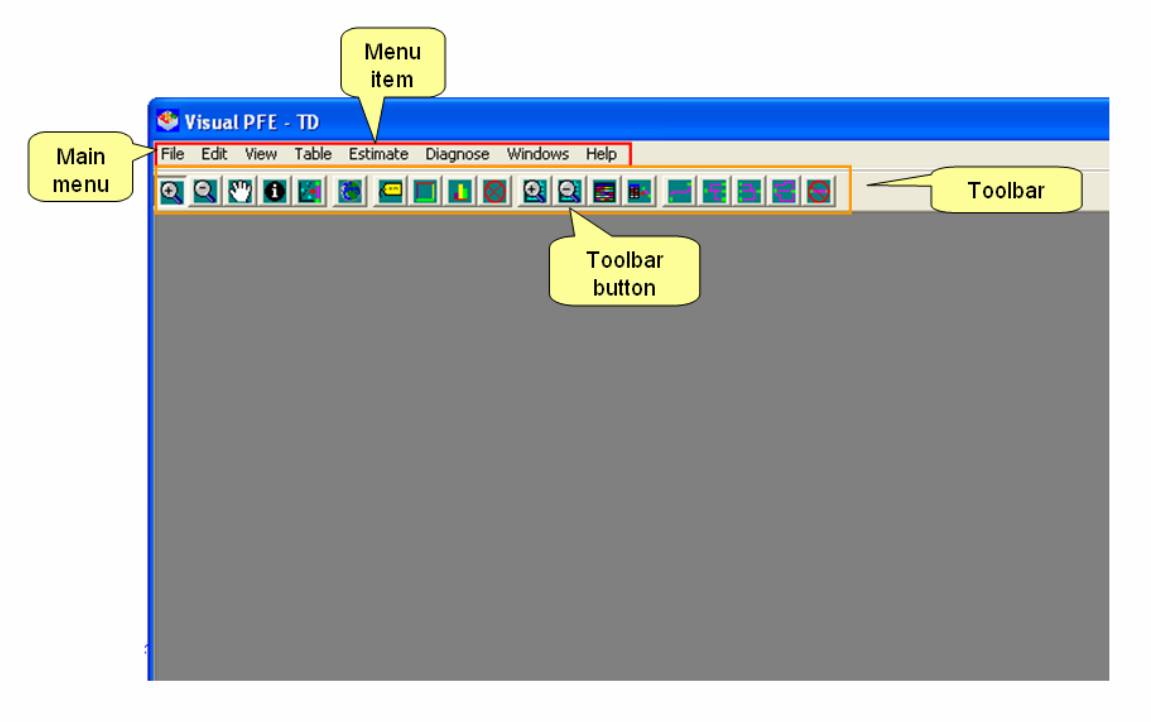


Figure 2 Main Window

Main Menu

There are eight menu items in the main menu of Visual PFE. Each menu item contains sub-items.

- File: Control functions related to creating, opening, and saving of individual MDI child forms.
  - New: Open a new text editor window, which can be used to create PFE network text files.
  - Open: Open an existing text file in a text document form.
  - Save As Excel: Save an OD table as a Microsoft Excel file.
  - Save As Text: Save the text file in the text document form.
  - Exit: Exit Visual PFE - TD.

- Edit: Control common functions of a text document form.
  - Undo: Undo changes to the text file.
  - Cut: Cut selected texts and place the content on the system clipboard.
  - Copy: Copy selected texts and place the content on the system clipboard.
  - Paste: Paste the texts contained on the system clipboard.
  - Select All: Select all texts in the text file.

- View: Control the navigation functions of a map window.
    - Zoom In: Zoom in
    - Zoom Out: Zoom out
    - Pan:
    - Full Scale

- Table: Control functions related to the table window.
    - Open Path Table: Open the path table pertaining to a PFE estimation.
    - Open OD Table: Open the path table pertaining to a PFE estimation.
    - OD Table Color: Adjust the individual colors of an OD table

- Estimate: Open the LPFE time dependent estimation GUI.

- Diagnose: Control functions designed to diagnose results of PFE estimation.
    - Statistics: Open the summary statistics of a PFE estimation with a text document window.
    - Graphs: Open a scatter plot (observed vs. estimated link flow) window of a PFE estimation.
    - Create Shapefiles: Create shapefiles from previously estimated LPFE results (e.g., *.est).
    - Compares: Open multiple MDI child forms pertaining to results of multiple PFE estimation runs (from different input sets).

- Windows: Organize opened MDI child windows.
    - Cascade: Cascade MDI child forms.
    - Tile: Tile MDI child forms.

- Help: Provide users with help to Visual PFE
    - Visual PFE – TD Help: Open the Visual PFE Windows HTML help file.
    - About Visual PFE - TD: Open the Visual PFE copyright form.

Toolbar

There are 19 toolbar buttons (Figure 3) in the Visual PFE - TD. The name of a toolbar button is shown when the mouse is placed over the button.

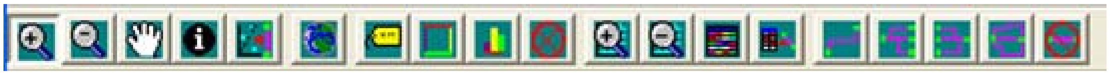


Figure 3 Toolbar Buttons

The first five buttons from left are toggle buttons. Only one button can be pressed down at a time. The toggle button that is pushed down will remain active until another toggle button is pressed down. The five toggle buttons are:

- Toolbar Button Zoom In : Zoom in a map window or a scatter plot window.
- Toolbar Button Zoom Out : Zoom out of a map window or a scatter plot window.
- Toolbar Button Pan : Pan a map window or a scatter plot window.
- Toolbar Button Identify : Click at a feature on the active layer of a map or scatter plot to show information of the feature.
- Toolbar Button Display Link : Relate a point in the scatter Plot to the corresponding link in the network. When this button is pressed down, click at a scatter plot point and the focus will be move to the network window with the corresponding network link highlighted.

Other than the five toggle buttons, rest of the buttons are push buttons. The function of a push button is activated after the button is pushed. After the function is completed, a push button returns inactive and does not stay pushed down. The push buttons are:

- Toolbar Button Full Scale : Return the scale of a map window or a scatter plot window to full scale.
- Toolbar Button Label Features : Activate the Label Setup Window to label features on the active layer of a map or a scatter plot.

- Toolbar Button Class Map : Activate the Class Map Setup Window to classify the features on the active layer of a map or a scatter plot using different colors and symbol sizes.

- Toolbar Button Bar Chart : Activate the Bar Chart Setup Window to create bar charts for the features on the active layer of a map or a scatter plot based on the selected values of the feature's attributes.

- Toolbar Button Clear Postings : Clear all postings (e.g., label, classes, and bar charts) added to the features of a map or a scatter plot.

- Toolbar Button Table Zoom In : Zoom in a table window.

- Toolbar Button Table Zoom Out : Zoom out a table window.

- Toolbar Button OD Color : Activate the OD Color Setup Window to adjust the colors of the OD table cells according to the values of the cells.

- Toolbar Button Display Desire Lines : Relate a cell in an OD table to the corresponding desire lines in the network. If the active cell of the OD table stays on an internal cell when the button is clicked, the focus will be move to the network window and the desire line corresponding to the OD pair will be highlighted. If a cell in the row sum or the column sum is the active cell, one-to-all or all-to-one desire lines will be highlighted. If the active cell of the OD table stays on the table total, all of the desire lines will be highlighted.

- Toolbar Button Single Path : Relate a record in the path table to the corresponding path in the network. When the button is clicked, the path corresponding to the active row of the Path Table will be highlighted.

- Toolbar Button One to All Paths : When the button is clicked, all the paths originating from the origin of the active row of the Path Table will be highlighted.

- Toolbar Button All to One Paths : When the button is clicked, all the paths ending at the destination of the active row of the Path Table will be highlighted.

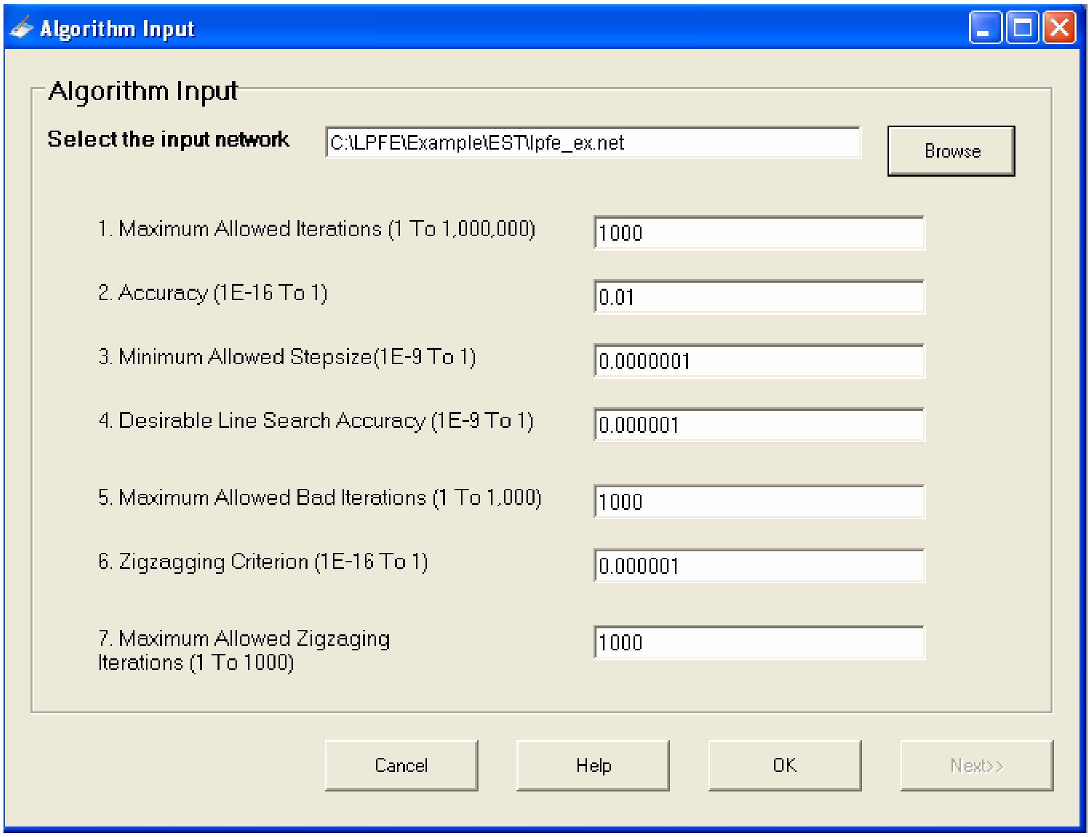
- Toolbar Button One to One Paths : When the button is clicked, all the paths beginning at the origin and ending at the destination of the active row of the Path Table will be highlighted.

- Toolbar Button Clear Track Lines : Clear highlighted lines (e.g., desire lines and paths).

Algorithm and Parameter Input Windows

To run the LPFE estimation program within Visual PFE – TD, two GUIs are created for the users to enter parameters of a LPFE estimation run. The first window is the Algorithm Input window (Figure 4) and the second is the Parameter Input Window (Figure 5). The first window can be activated by accessing the menu item Estimate/Time Dependent. After data on the Algorithm Input Window are properly entered, the Next button will be enabled (i.e., the LPFE input file *.alg is created successfully). If the parameters entered do not conform to the specification of the LPFE program, users will be asks to check the numbers on the form. For details about the parameters, please see LPFE User Manual.

Clicking at the Next button will launch the Parameter Input Window.

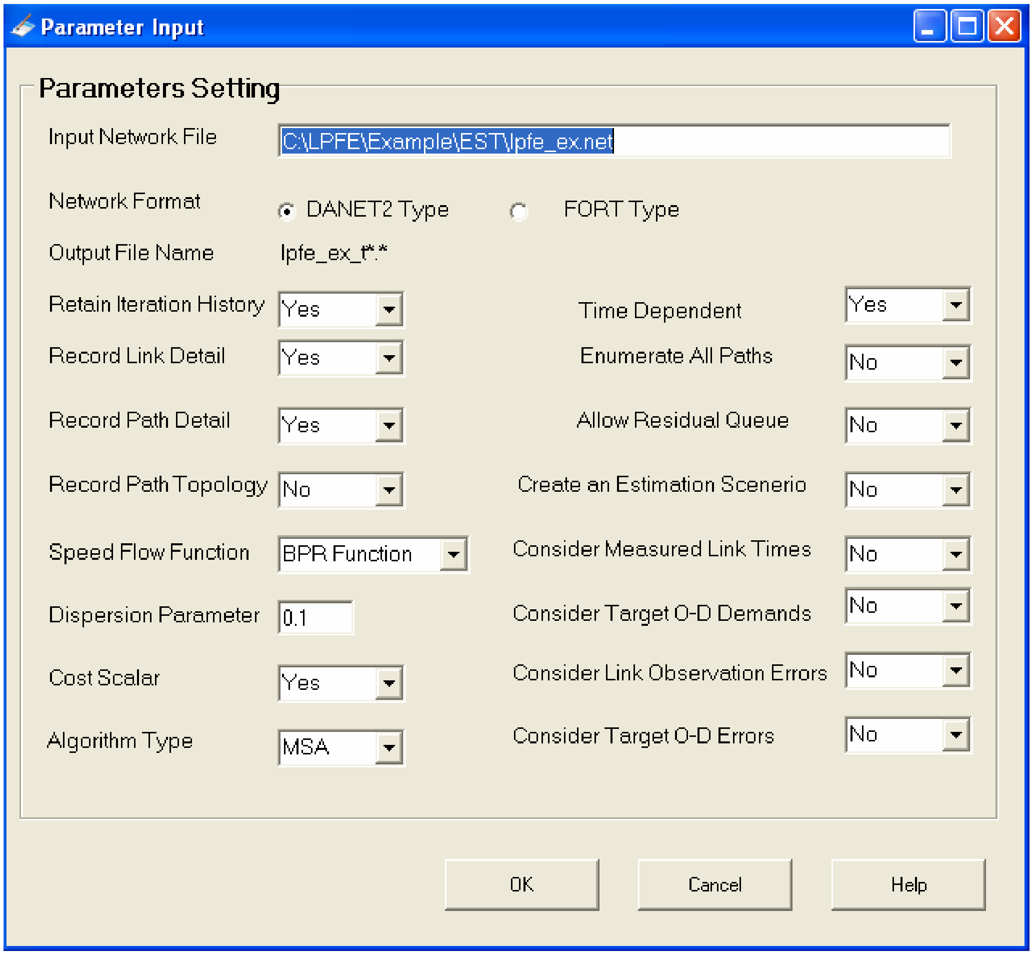


Algorithm Input

Select the input network: C:\LPFE\Example\EST\lpfe_ex.net

1. Maximum Allowed Iterations (1 To 1,000,000): 1000
2. Accuracy (1E-16 To 1): 0.01
3. Minimum Allowed Stepsize(1E-9 To 1): 0.0000001
4. Desirable Line Search Accuracy (1E-9 To 1): 0.000001
5. Maximum Allowed Bad Iterations (1 To 1,000): 1000
6. Zigzagging Criterion (1E-16 To 1): 0.000001
7. Maximum Allowed Zigzaging Iterations (1 To 1000): 1000

Figure 4 Algorithm Input Window



Parameters Setting

Input Network File: C:\LPFE\Example\EST\lpfe_ex.net

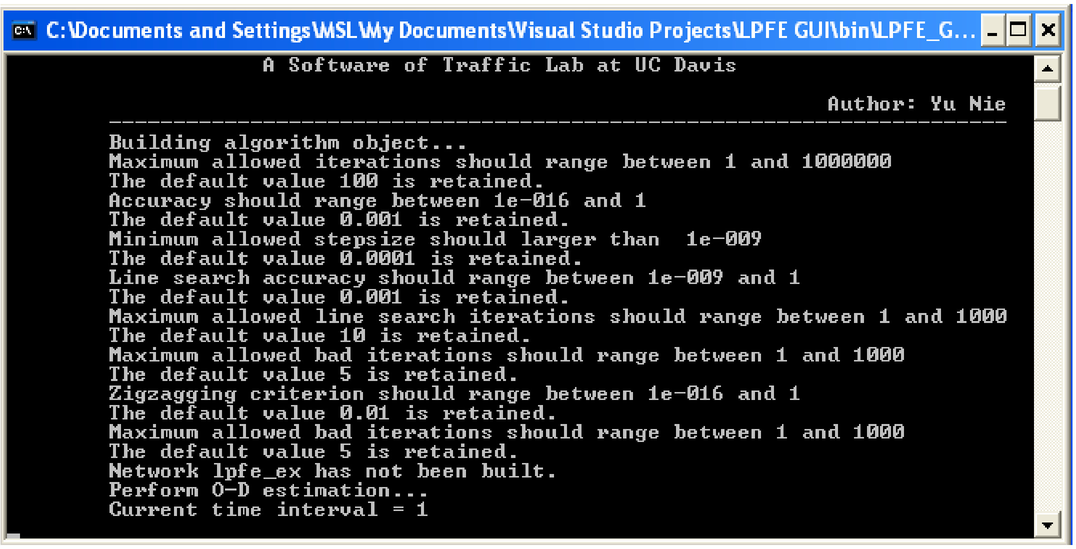
Network Format: ● DANET2 Type ○ FORT Type

Output File Name: lpfe_ex_t*.*

Retain Iteration History	Yes	Time Dependent	Yes
Record Link Detail	Yes	Enumerate All Paths	No
Record Path Detail	Yes	Allow Residual Queue	No
Record Path Topology	No	Create an Estimation Scenerio	No
Speed Flow Function	BPR Function	Consider Measured Link Times	No
Dispersion Parameter	0.1	Consider Target O-D Demands	No
Cost Scalar	Yes	Consider Link Observation Errors	No
Algorithm Type	MSA	Consider Target O-D Errors	No

Figure 5 Parameter Input Window

After parameters on the Parameter Input Window are properly entered, Visual PFE – TD will attempt to create another LPFE input file (i.e., the *.par file) after users click at OK. If the file is successfully created, the LPFE program will be launched and a DOS command window will appear (Figure 6). No user intervention is required when the program is executing. If the LPFE run is not successful, users will receive an error message prompting them to fix the LPFE network data or change the parameter values. For details about the LPFE network data and the parameters, please see *LPFE User Manual*.



C:\Documents and Settings\MSL\My Documents\Visual Studio Projects\LPFE GUI\bin\LPFE_G...

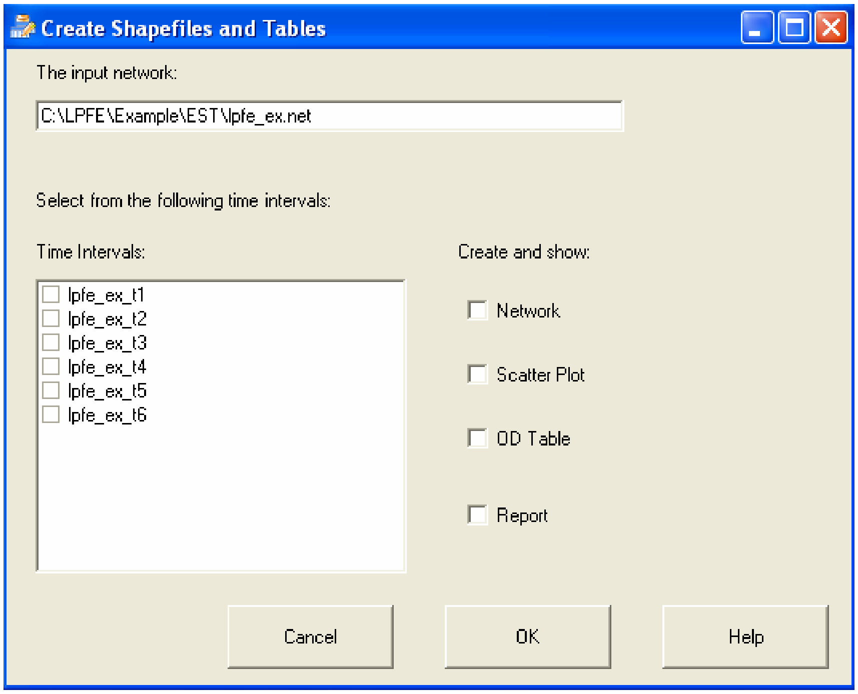
```
                    A Software of Traffic Lab at UC Davis

                                                              Author: Yu Nie
        ----------------------------------------------------------------------
        Building algorithm object...
        Maximum allowed iterations should range between 1 and 1000000
        The default value 100 is retained.
        Accuracy should range between 1e-016 and 1
        The default value 0.001 is retained.
        Minimum allowed stepsize should larger than  1e-009
        The default value 0.0001 is retained.
        Line search accuracy should range between 1e-009 and 1
        The default value 0.001 is retained.
        Maximum allowed line search iterations should range between 1 and 1000
        The default value 10 is retained.
        Maximum allowed bad iterations should range between 1 and 1000
        The default value 5 is retained.
        Zigzagging criterion should range between 1e-016 and 1
        The default value 0.01 is retained.
        Maximum allowed bad iterations should range between 1 and 1000
        The default value 5 is retained.
        Network lpfe_ex has not been built.
        Perform O-D estimation...
        Current time interval = 1
```

Figure 6 DOS Command Window

Create Shapefiles and Tables Window

A successful LPFE estimation will create output text files that will be converted by Visual PFE - TD for the visualization of the outputs. Each LPFE estimation produces OD and link flow estimates for multiple time periods. The Create Shapefiles and Tables Window (Figure 7) is created to let users choose which time periods and what kinds of results they want to see. Once the selection is made, multiple windows forms will be opened for the selected time periods (Figure 8).

The Create Shapefiles Window can be launched for previously estimated outputs, for which the shapefiles had not been created. To do this, go to the menu item Diagnose/Create Shapefiles and the Create Shapefiles Window will show up (Figure 9). Note that a Browse button is placed next to the input network path label. Clicking at the browse button will enable users to select the data folder in which previously estimated outputs are stored.



Create Shapefiles and Tables

The input network:

C:\LPFE\Example\EST\lpfe_ex.net

Select from the following time intervals:

Time Intervals:

□ lpfe_ex_t1
□ lpfe_ex_t2
□ lpfe_ex_t3
□ lpfe_ex_t4
□ lpfe_ex_t5
□ lpfe_ex_t6

Create and show:

□ Network

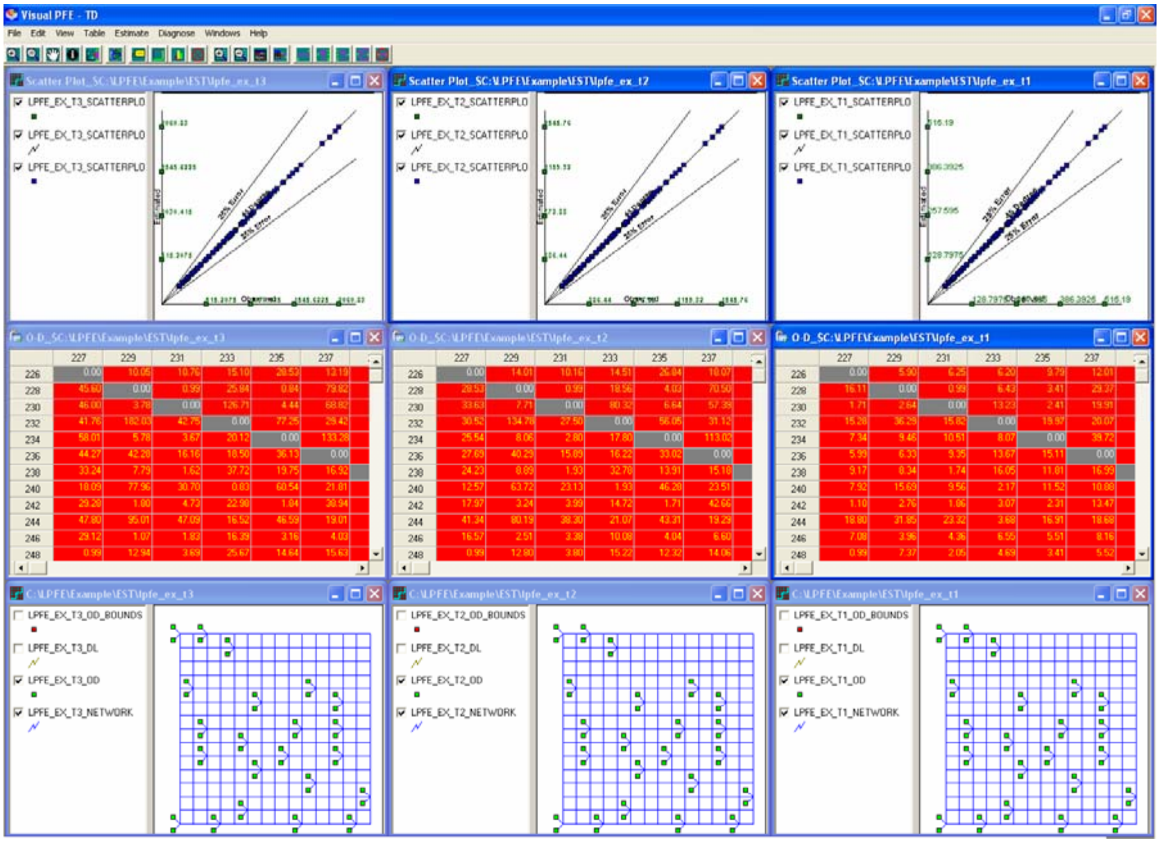
□ Scatter Plot

□ OD Table

□ Report

Cancel | OK | Help

Figure 7 Create Shapefiles and Tables Window



O-D_SC:\LPFE\Example\EST\lpfe_ex_t3

	227	229	231	233	235	237
226	0.00	10.05	10.76	15.10	28.53	13.19
228	45.60	0.00	0.99	25.84	0.84	79.82
230	46.00	3.78	0.00	126.71	4.44	68.82
232	41.76	182.03	42.75	0.00	77.25	29.42
234	58.01	5.78	3.67	20.12	0.00	133.28
236	44.27	42.28	16.16	18.50	36.13	0.00
238	33.24	7.79	1.62	37.72	19.75	16.92
240	18.09	77.96	30.70	0.83	60.54	21.81
242	29.28	1.00	4.73	22.90	1.04	38.94
244	47.80	95.01	47.09	16.52	46.59	19.01
246	29.12	1.07	1.83	16.39	3.16	4.03
248	0.99	12.94	3.69	25.67	14.64	15.63

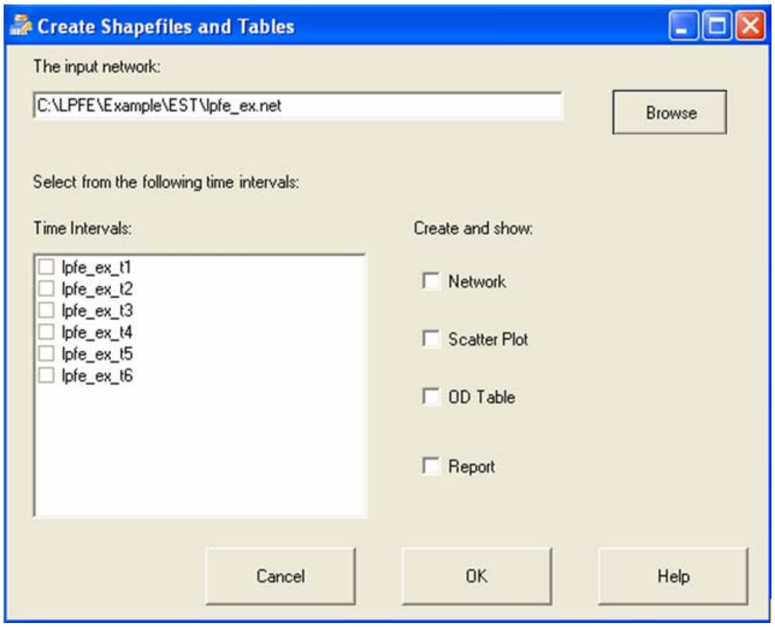
O-D_SC:\LPFE\Example\EST\lpfe_ex_t2

	227	229	231	233	235	237
226	0.00	14.01	10.16	14.51	26.84	18.07
228	28.53	0.00	0.99	18.56	4.03	70.50
230	33.63	7.71	0.00	80.32	6.64	57.39
232	30.52	134.78	27.50	0.00	56.05	31.12
234	25.54	8.06	2.80	17.80	0.00	113.02
236	27.69	40.29	15.89	16.22	33.02	0.00
238	24.23	8.89	1.93	32.78	13.91	15.18
240	12.57	63.72	23.13	1.93	46.28	23.51
242	17.97	3.24	3.99	14.72	1.71	42.66
244	41.34	80.19	38.30	21.07	43.31	19.29
246	16.57	2.51	3.38	10.08	4.04	6.60
248	0.99	12.80	3.80	15.22	12.32	14.06

O-D_SC:\LPFE\Example\EST\lpfe_ex_t1

	227	229	231	233	235	237
226	0.00	5.90	6.25	6.20	9.79	12.01
228	16.11	0.00	0.99	6.43	3.41	29.37
230	1.71	2.64	0.00	13.23	2.41	19.91
232	15.28	36.29	15.82	0.00	19.97	20.07
234	7.34	9.46	10.51	8.07	0.00	39.72
236	5.99	6.33	9.35	13.67	15.11	0.00
238	9.17	8.34	1.74	16.05	11.81	16.99
240	7.92	15.69	9.56	2.17	11.52	10.88
242	1.10	2.76	1.06	3.07	2.31	13.47
244	18.80	31.85	23.32	3.68	16.91	18.68
246	7.08	3.96	4.36	6.55	5.51	8.16
248	0.99	7.37	2.05	4.69	3.41	5.52

Figure 8 Main Window with Output Windows


Create Shapefiles and Tables

The input network:

C:\LPFE\Example\EST\lpfe_ex.net

Select from the following time intervals:

Time Intervals:

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t4
- □ lpfe_ex_t5
- □ lpfe_ex_t6

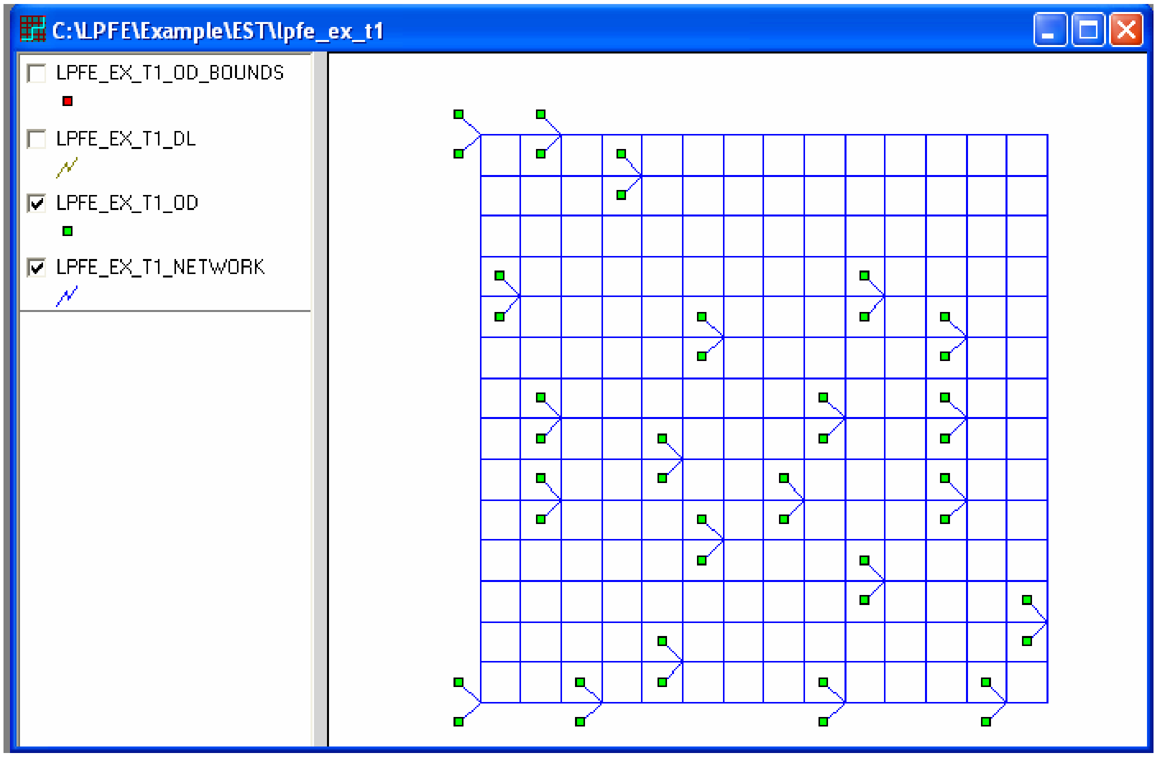
Create and show:

- □ Network
- □ Scatter Plot
- □ OD Table
- □ Report

Figure 9 Create Shapefiles Window for Previously Estimated Outputs

Network Map Window

The outputs of a LPFE estimation are converted to ESRI Shapefiles for visualization. A map window is shown in Figure 10. The title of a map window corresponds to the file name and path of the LPFE estimation output for one particular time period. For example, the map window in Figure 10 corresponds to the estimation result of time period 1 based on the input network file called **lpfe_ex**, which is stored in the folder of C:\LPFE\Example\EST\. For more details, see Technical Specification.



C:\LPFE\Example\EST\lpfe_ex_t1

□ LPFE_EX_T1_OD_BOUNDS 🟥

□ LPFE_EX_T1_DL

☑ LPFE_EX_T1_OD 🟩

☑ LPFE_EX_T1_NETWORK

Figure 10 Network Map Window

A network map consists of four ESRI Shapefiles, each presented as a map layer:

- Network links: All of the link attributes are stored in this layer, including the observed and estimated link flows.
- Origins and destinations: The production and attraction of the origins and destinations are stored in this point layer.
- Desire lines: A desire line represents the flow between a pair of OD. The desire lines layer is hidden by default. Users can choose to display or hide them.
- Map boundary points: Two boundary points are included in the network map to create sufficient margins around the network such that all features of the network can be displayed within the map window.

The Network Windows form is divided into the map and the legend areas. The names of the layers are displayed in the legend area. Note that a layer name is made up by prefixing the name of the estimation result (e.g., lpfe_ex_t1) with one of the words: Network, OD, DL, and OD_BOUNDS. See Technical Specification for details.

Basic functions of the map window are adjusted through the map legend:

- **Setting the active layer**: A layer can be made active by moving mouse cursor over and clicking at the corresponding title in the map legend. The legend of the active layer has a line underneath the layer name (see the legend of the network layer in Figure 10). Labels, colors, and charts based on attributes can only be applied to features of the active layer. When varying colors are applied to features of a layer by classes, the legend of the layer will show the colors and the corresponding values.

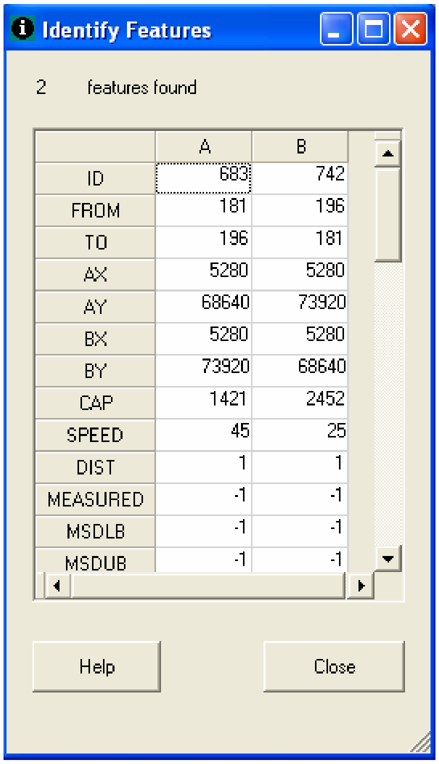
- **Making a layer invisible**: A layer can be made invisible by un-checking the check box right next to the layer's name in the legend area (i.e., the desire line and the boundary points are made invisible in the map window).

- **Changing the drawing order**: Drawing order of the layers can also be changed by dragging the layer name up or down in the legend area.

- **Adjusting the width of the legend**: When a layer's name is longer than the default legend width, the border between the map and the legend can be dragged and moved to accommodate the width of the legend.

Identify Features Window

Attributes of map features can be identified and displayed using the Identify toolbar button. To identify a particular feature in a layer, first make the layer active by clicking at the name of the layer in the legend, then press the tool button Identify. The mouse cursor, when placed above the map window, will change to the information symbol ℹ. Move the mouse cursor to the feature and click at it. The Identify Features Window will appear with attributes of the form displayed in a spreadsheet (Figure 11).



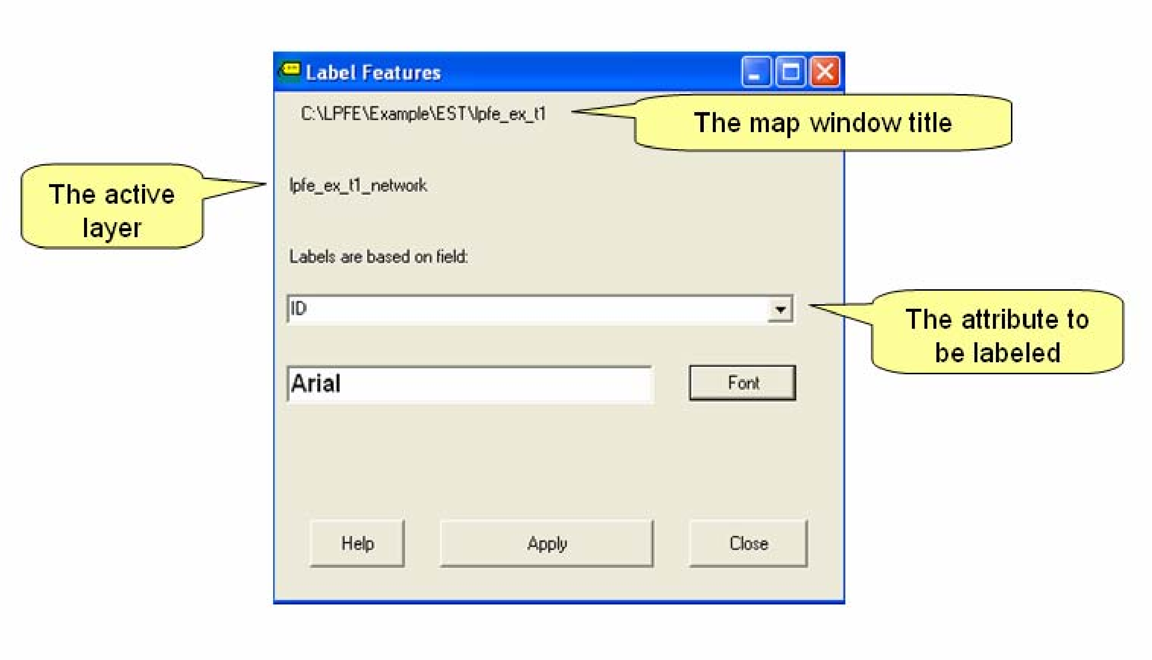
2 features found

	A	B
ID	683	742
FROM	181	196
TO	196	181
AX	5280	5280
AY	68640	73920
BX	5280	5280
BY	73920	68640
CAP	1421	2452
SPEED	45	25
DIST	1	1
MEASURED	-1	-1
MSDLB	-1	-1
MSDUB	-1	-1

Figure 11 Identify Features Window

Label Features Window

Features on a map layer can be labeled with attributes of the layer. For example, each link on the network layer can be labeled with link ID. Feature labeling is set up through the Label Feature Window (Figure 12). To activate the Label Feature Window, first make the layer active by clicking at the name of the layer in the legend, then press the tool button Label Features. Once the form appears, a particular attribute can be selected for labeling and the font for the label can also be adjusted using the Font button.



Label Features

C:\LPFE\Example\EST\lpfe_ex_t1

lpfe_ex_t1_network

Labels are based on field:

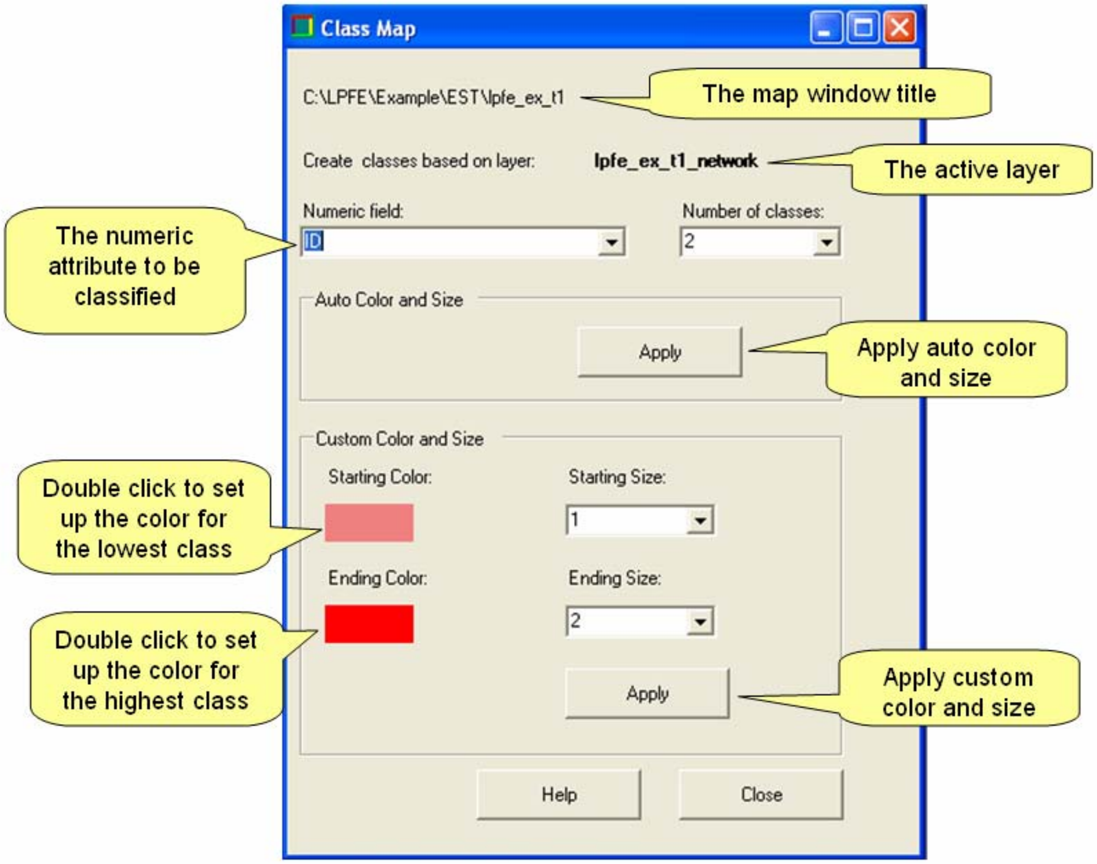
ID

Arial

Figure 12 Label Features Window

Class Map Window

Features on a map layer can be drawn with different colors and sizes based on values of a particular attribute of the layer. For example, links on the network layer can be grouped into five classes based on the values of estimated link flows. Each class of links will be drawn with the same color and width. Class mapping is set up through the Class Map Window (Figure 13). To activate the Class Map Window, first make the layer active by clicking at the name of the layer in the legend, then press the tool button Class Map. Once the form appears, a particular attribute and the number of classes can be selected. Users can choose to let the program determine the colors and sizes of the classes by applying auto color and size. Users can also set up the color and size by applying custom color and size. To do this, double click the color blocks to set up the color for the lowest and highest classes. Then, select the symbol sizes for the lowest and highest classes. After clicking the Apply (custom color and size) button, the program will create for all classes a graduate color and size transition from the lowest to highest classes.


Class Map

C:\LPFE\Example\EST\lpfe_ex_t1

Create classes based on layer: **lpfe_ex_t1_network**

Numeric field: ID

Number of classes: 2

Auto Color and Size

Custom Color and Size

Starting Color: 🟥

Starting Size: 1

Ending Color: 🟥

Ending Size: 2

The map window title

The active layer

The numeric attribute to be classified

Apply auto color and size

Double click to set up the color for the lowest class

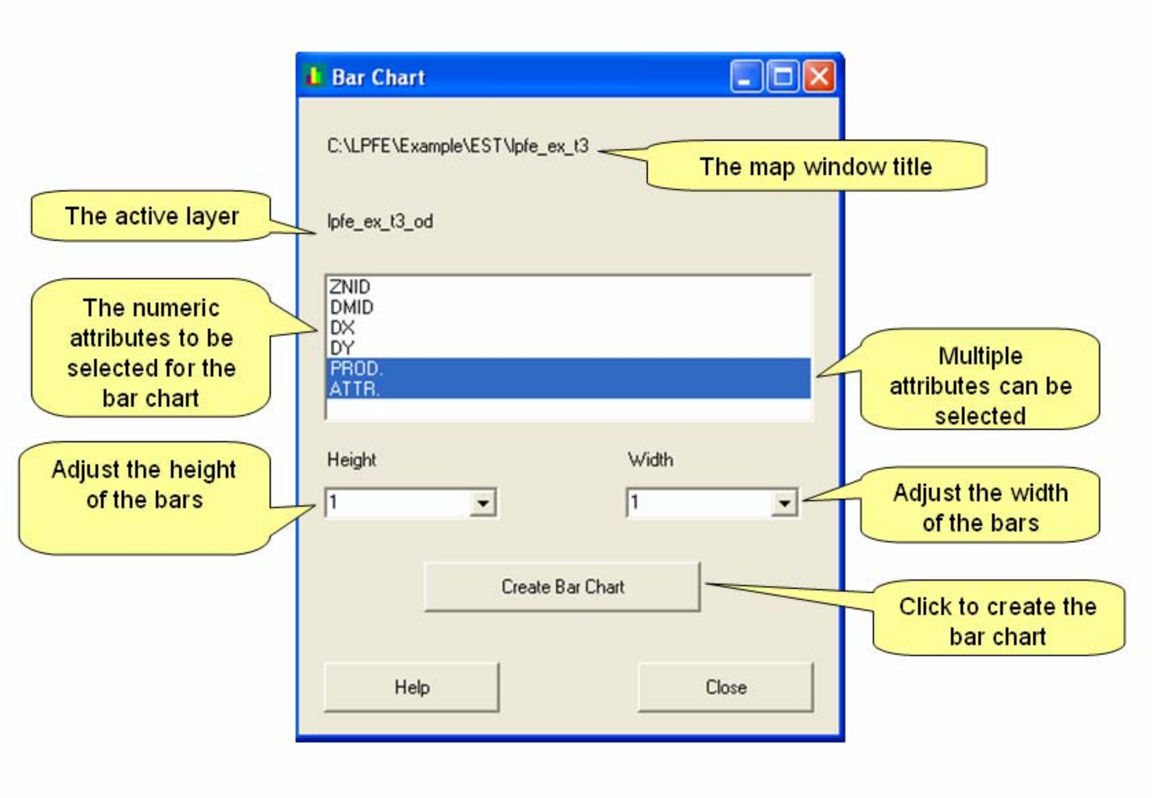
Double click to set up the color for the highest class

Apply custom color and size

Figure 13 Class Map Window

Bar Chart Window

Bar charts based on values of a particular attribute of a layer can be created using the Bar Char Window. Bar charts are best suited for visualizing the variation of production and attraction at origins and destinations. Note that bar charts are not designed for network links. Applying bar chats to network links will cause the links to become invisible (i.e., links are replaced by the bars). To activate the Bar Chart Window (Figure 14), first make the layer active by clicking at the name of the layer in the legend, then press the tool button Chart Map. Once the form appears, clicking at an attribute from the attribute list box will highlight the attribute. Clicking at an attribute twice will de-highlight the attribute. Multiple attributes can be selected at the same time. Each attribute will appear on the map as a bar of varying height (based on the value of the attribute). Users can choose to let the program determine the colors and sizes of the classes by applying auto color and size. Users can also set up the height and width of the bars by applying custom height and width. To do this, simply select the height and width from the two pull down boxes. After clicking the Create Bar Chart button, the program will create the bar chart accordingly.



C:\LPFE\Example\EST\lpfe_ex_t3

lpfe_ex_t3_od

ZNID
DMID
DX
DY
PROD.
ATTR.

Height: 1

Width: 1

The map window title

The active layer

The numeric attributes to be selected for the bar chart

Multiple attributes can be selected

Adjust the height of the bars

Adjust the width of the bars

Click to create the bar chart

Figure 14 Bar Chart Window

Scatter Plot Window

One of the most common and important diagnostic graphs for the OD estimation problem is the observed-versus-estimated scatter plot of network link flows. The scatter plot is implemented in Visual PFE - TD as a collection of map layers. The scatter plot point layer uses the observed value of a link as the X coordinate and the estimated value as the Y coordinate. A link layer is used to represent the two axes and other boundary lines delineating the levels of accuracy of the estimation.

The scatter plot points, the axes, and the markers are actually shapefiles and presented in a map window. Thus, functions of a scatter plot window are the same as a map window. A scatter plot window is shown in Figure 15. Note that the title of the Scatter Plot Window is created by prefixing the word "Scatter Plot_$" with the title of the corresponding network Map Window. The title enables the users to associate different output windows with the corresponding network. When preparing the LPFE network input network data, please make sure that the character "$" is not used to make up the file name. A network file with a file name containing the character "$" will cause the program to malfunction.

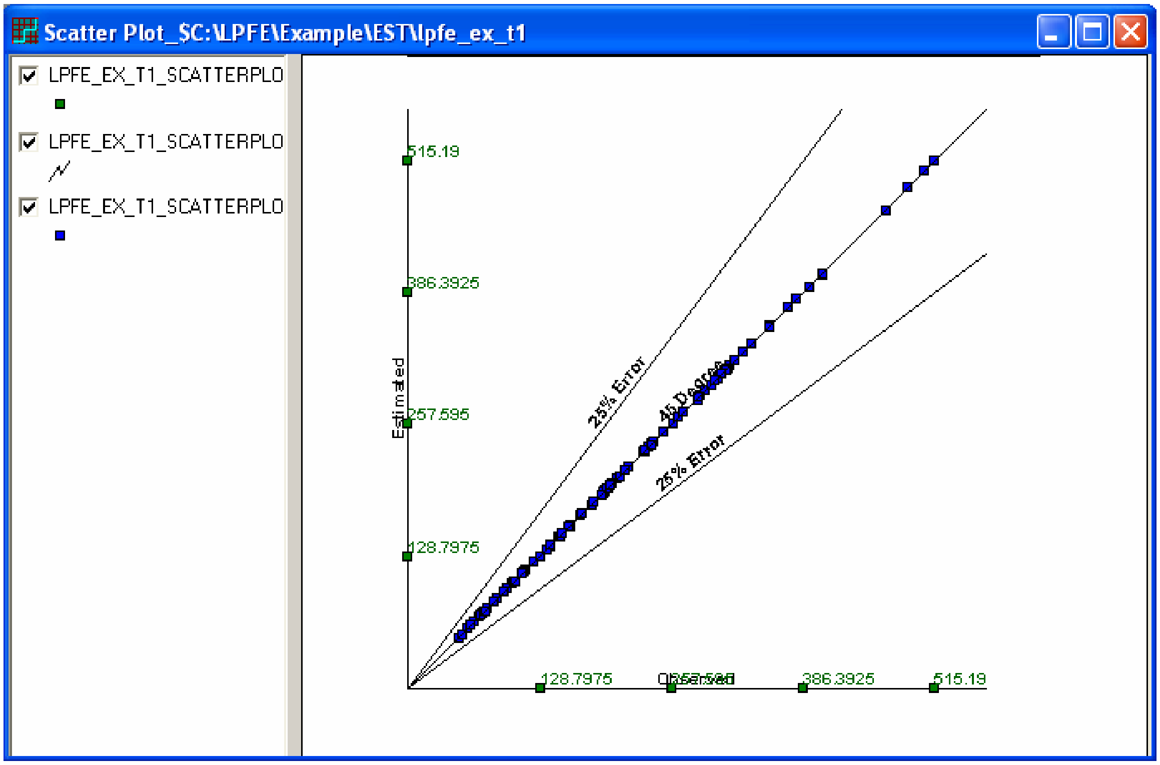


Figure 15 Scatter Plot Window

There are three ways to open a Scatter Plot Window:

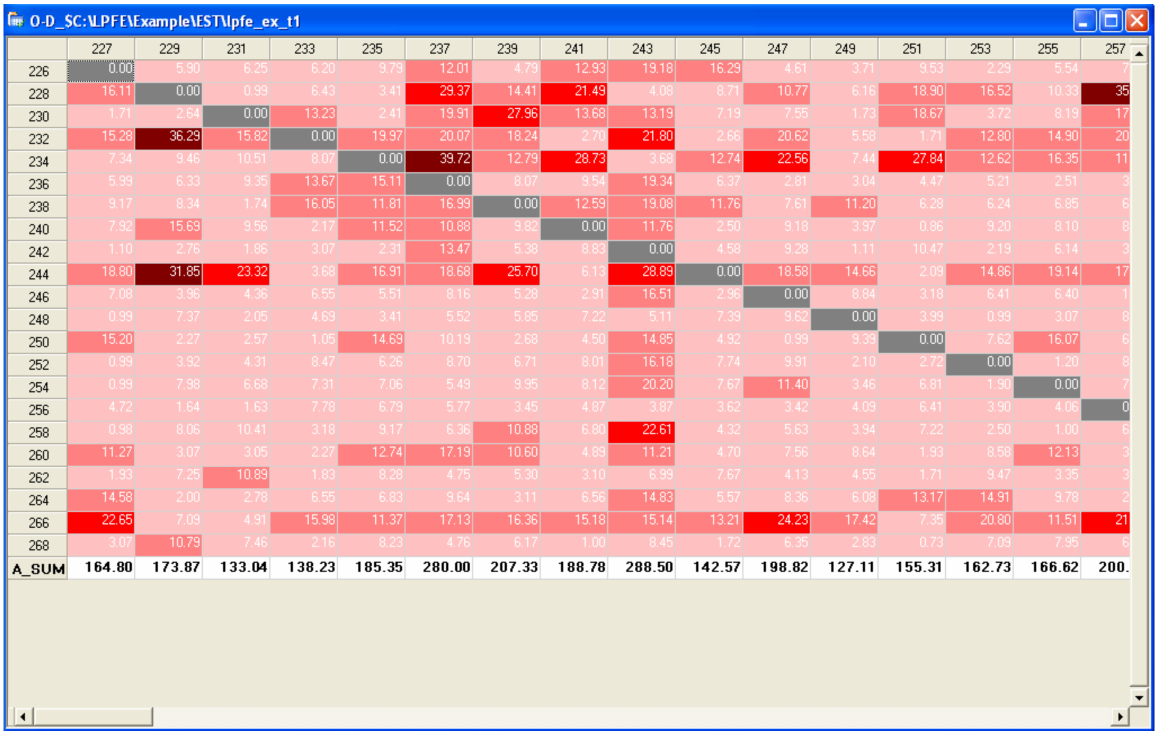
- Select the Scatter Plot option in the Create Shapefiles and Tables Window.
- When a network map is the active window, go to the menu item Diagnostic/Graph.
- Go to the menu item Diagnostic/Compare and select from the available scatter plots.

Link SP to Map

A unique function of the scatter plot is the dynamic linkage between a scatter plot point and the location of the corresponding link on the map. To use this function, when the network Map and the Scatter Plot windows are both opened, select the scatter plot point layer as the active layer. Click the Link SP to Map button, and the corresponding network link will be highlighted in the Map Window.

OD Table Window

The estimated OD flows between the origins and destinations are generated by LPFE as a text file (i.e., the *.est file). The Visual PFE- TD program converts the text file and formats it as a spreadsheet using the Spread software component (see Technical Specification). A resulting OD table is shown in Figure 16. Note that the title of the OD Table Window is also created by prefixing the word "O-D_$" with the title of the corresponding network Map Window.



O-D_$C:\LPFE\Example\EST\lpfe_ex_t1

	227	229	231	233	235	237	239	241	243	245	247	249	251	253	255	257
226	0.00	5.90	6.25	6.20	9.79	12.01	4.79	12.93	19.18	16.29	4.61	3.71	9.53	2.29	5.54	7
228	16.11	0.00	0.99	6.43	3.41	29.37	14.41	21.49	4.08	8.71	10.77	6.16	18.90	16.52	10.33	35
230	1.71	2.64	0.00	13.23	2.41	19.91	27.96	13.68	13.19	7.19	7.55	1.73	18.67	3.72	8.19	17
232	15.28	36.29	15.82	0.00	19.97	20.07	18.24	2.70	21.80	2.66	20.62	5.58	1.71	12.80	14.90	20
234	7.34	9.46	10.51	8.07	0.00	39.72	12.79	28.73	3.68	12.74	22.56	7.44	27.84	12.62	16.35	11
236	5.99	6.33	9.35	13.67	15.11	0.00	8.07	9.54	19.34	6.37	2.81	3.04	4.47	5.21	2.51	3
238	9.17	8.34	1.74	16.05	11.81	16.99	0.00	12.59	19.08	11.76	7.61	11.20	6.28	6.24	6.85	6
240	7.92	15.69	9.56	2.17	11.52	10.88	9.82	0.00	11.76	2.50	9.18	3.97	0.86	9.20	8.10	8
242	1.10	2.76	1.86	3.07	2.31	13.47	5.38	8.83	0.00	4.58	9.28	1.11	10.47	2.19	6.14	3
244	18.80	31.85	23.32	3.68	16.91	18.68	25.70	6.13	28.89	0.00	18.58	14.66	2.09	14.86	19.14	17
246	7.08	3.96	4.36	6.55	5.51	8.16	5.28	2.91	16.51	2.96	0.00	8.84	3.18	6.41	6.40	1
248	0.99	7.37	2.05	4.69	3.41	5.52	5.85	7.22	5.11	7.39	9.62	0.00	3.99	0.99	3.07	8
250	15.20	2.27	2.57	1.05	14.69	10.19	2.68	4.50	14.85	4.92	0.99	9.39	0.00	7.62	16.07	6
252	0.99	3.92	4.31	8.47	6.26	8.70	6.71	8.01	16.18	7.74	9.91	2.10	2.72	0.00	1.20	8
254	0.99	7.98	6.68	7.31	7.06	5.49	9.95	8.12	20.20	7.67	11.40	3.46	6.81	1.90	0.00	7
256	4.72	1.64	1.63	7.78	6.79	5.77	3.45	4.87	3.87	3.62	3.42	4.09	6.41	3.90	4.06	0
258	0.98	8.06	10.41	3.18	9.17	6.36	10.88	6.80	22.61	4.32	5.63	3.94	7.22	2.50	1.00	6
260	11.27	3.07	3.05	2.27	12.74	17.19	10.60	4.89	11.21	4.70	7.56	8.64	1.93	8.58	12.13	3
262	1.93	7.25	10.89	1.83	8.28	4.75	5.30	3.10	6.99	7.67	4.13	4.55	1.71	9.47	3.35	3
264	14.58	2.00	2.78	6.55	6.83	9.64	3.11	6.56	14.83	5.57	8.36	6.08	13.17	14.91	9.78	2
266	22.65	7.09	4.91	15.98	11.37	17.13	16.36	15.18	15.14	13.21	24.23	17.42	7.35	20.80	11.51	21
268	3.07	10.79	7.46	2.16	8.23	4.76	6.17	1.00	8.45	1.72	6.35	2.83	0.73	7.09	7.95	6
A_SUM	164.80	173.87	133.04	138.23	185.35	280.00	207.33	188.78	288.50	142.57	198.82	127.11	155.31	162.73	166.62	200.

Figure 16 OD Table Window

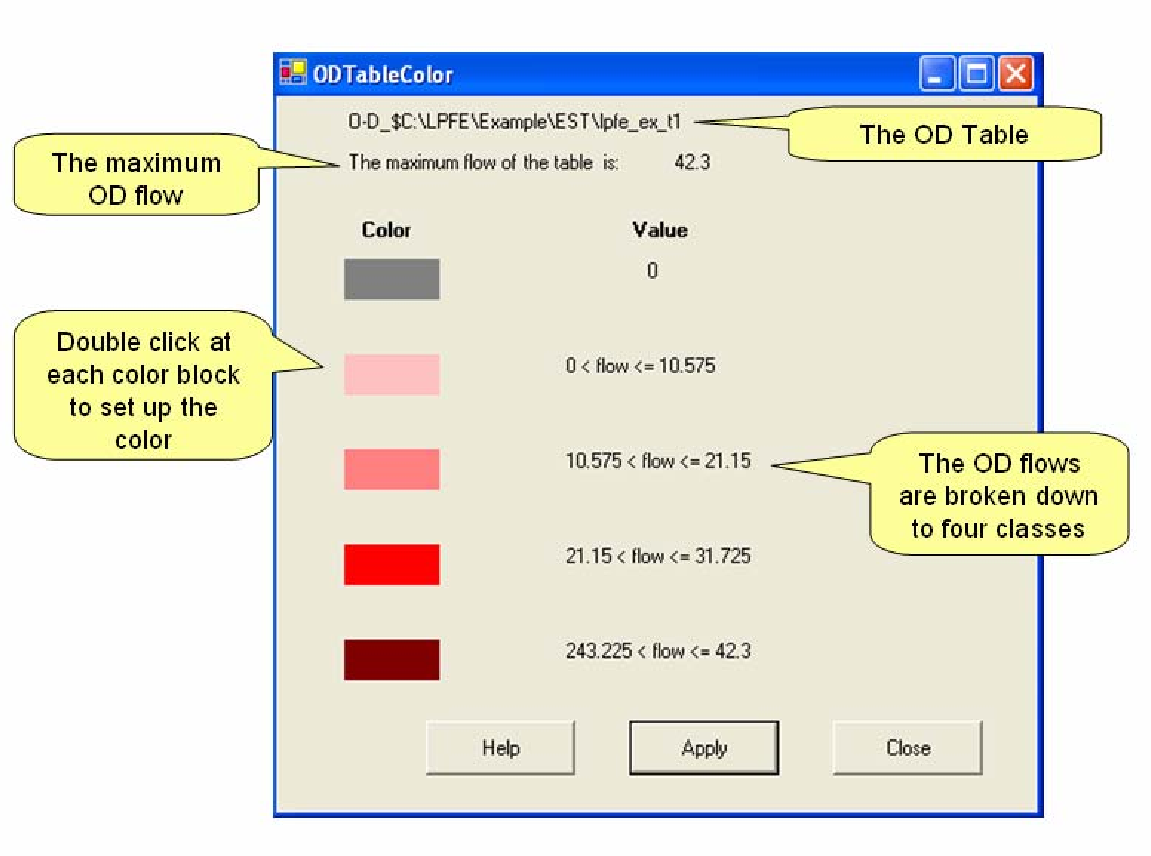
There are three ways to open an OD table:

- Select the OD Table option in the Create Shapefiles and Tables Window.
- When a network map is the active window, go to the menu item Tables/Open OD Table.
- Go to the menu item Diagnostic/Compare and select from the available OD Tables.

OD Color Window

The cell colors of the OD table can be adjusted using the OD Color Window (Figure 17). To open the OD Color Window, first make the OD table the active window, go to the menu item Table/OD Table Color or click at the toolbar button OD Color.

Once the OD Color Window is opened, the values of OD flows are broken down into four classes. Double click at each color block to set up the color. After clicking at the Apply button, the cell colors of the OD table will be adjusted accordingly.



ODTableColor

O-D_$C:\LPFE\Example\EST\lpfe_ex_t1

The maximum flow of the table is: 42.3

Color	Value
■	0
■	0 < flow <= 10.575
■	10.575 < flow <= 21.15
■	21.15 < flow <= 31.725
■	243.225 < flow <= 42.3

The OD Table

The maximum OD flow

Double click at each color block to set up the color

The OD flows are broken down to four classes

Figure 17 OD Color Window

Reduced OD Table View

Once the cell colors are adjusted, the OD table can be reduced to facilitate visualization of flow patterns of the entire table (Figure 18). The function is useful when comparing two OD tables of the same network.

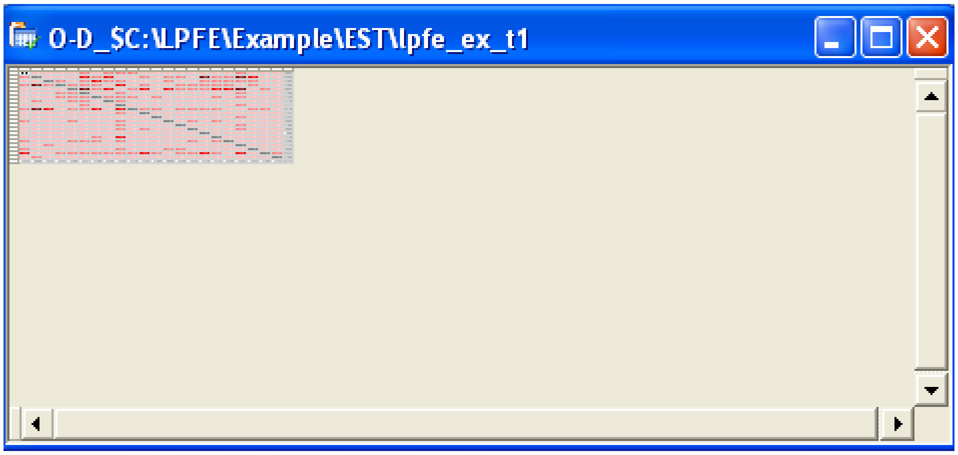


Figure 18 Reduced OD Table View

To reduce the OD table view, make the OD table the active window and click at the Table Zoom Out button. The view can be return back to normal by clicking at the Table Zoom In button.

Link OD Table to Map

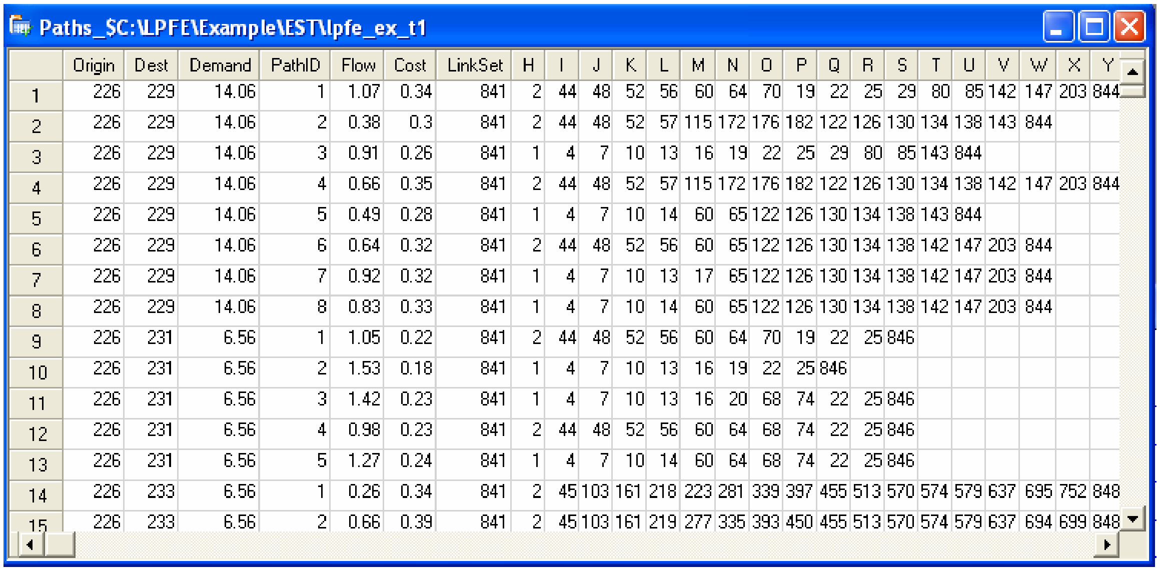
A unique function of the OD table is the dynamic linkage between the OD table and the location of the corresponding desire lines on the map. To use this function, when the network Map and the OD Table windows are both opened, click at a cell in the OD table to make it the active cell, then press down the Display Desire Line button. The corresponding desire line(s) will be highlighted in the Map Window. If the active cell of the OD table stays on an internal cell when the button is clicked, the focus will be move to the network map window and the desire line corresponding to the OD pair will be highlighted. If a cell in the row sum or the column sum is the active cell, one-to-all or all-to-one desire lines will be highlighted. If the active cell of the OD table stays on the table total, all of the desire lines will be highlighted.

Save an OD Table as Excel File

An OD table in Visual PFE – TD can be saved as a Microsoft Excel file. To use this function, when the OD Table window is opened, go to the menu item File/Save as Excel. The Save File As dialog box will open for users to enter a desired file name for the Excel File.

Path Table Window

The estimates of path flows between origins and destinations are also generated by LPFE as a text file (i.e., the *pfp file). Similar to the OD tables, the Visual PFE- TD program converts the path text file and formats it as a spreadsheet using the Spread software component (see Technical Specification). A resulting path table is shown in Figure 19. Note that the title of the Path Table Window is also created by prefixing the word “Paths_$” with the title of the corresponding network Map Window.


Paths_$C:\LPFE\Example\EST\lpfe_ex_t1

	Origin	Dest	Demand	PathID	Flow	Cost	LinkSet	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	226	229	14.06	1	1.07	0.34	841	2	44	48	52	56	60	64	70	19	22	25	29	80	85	142	147	203	844
2	226	229	14.06	2	0.38	0.3	841	2	44	48	52	57	115	172	176	182	122	126	130	134	138	143	844		
3	226	229	14.06	3	0.91	0.26	841	1	4	7	10	13	16	19	22	25	29	80	85	143	844				
4	226	229	14.06	4	0.66	0.35	841	2	44	48	52	57	115	172	176	182	122	126	130	134	138	142	147	203	844
5	226	229	14.06	5	0.49	0.28	841	1	4	7	10	14	60	65	122	126	130	134	138	143	844				
6	226	229	14.06	6	0.64	0.32	841	2	44	48	52	56	60	65	122	126	130	134	138	142	147	203	844		
7	226	229	14.06	7	0.92	0.32	841	1	4	7	10	13	17	65	122	126	130	134	138	142	147	203	844		
8	226	229	14.06	8	0.83	0.33	841	1	4	7	10	14	60	65	122	126	130	134	138	142	147	203	844		
9	226	231	6.56	1	1.05	0.22	841	2	44	48	52	56	60	64	70	19	22	25	846						
10	226	231	6.56	2	1.53	0.18	841	1	4	7	10	13	16	19	22	25	846								
11	226	231	6.56	3	1.42	0.23	841	1	4	7	10	13	16	20	68	74	22	25	846						
12	226	231	6.56	4	0.98	0.23	841	2	44	48	52	56	60	64	68	74	22	25	846						
13	226	231	6.56	5	1.27	0.24	841	1	4	7	10	14	60	64	68	74	22	25	846						
14	226	233	6.56	1	0.26	0.34	841	2	45	103	161	218	223	281	339	397	455	513	570	574	579	637	695	752	848
15	226	233	6.56	2	0.66	0.39	841	2	45	103	161	219	277	335	393	450	455	513	570	574	579	637	694	699	848

Figure 19 Path Table Window

To open a path table, when the corresponding network map is the active window, go to the menu item Tables/Open Path Table.

Display Paths

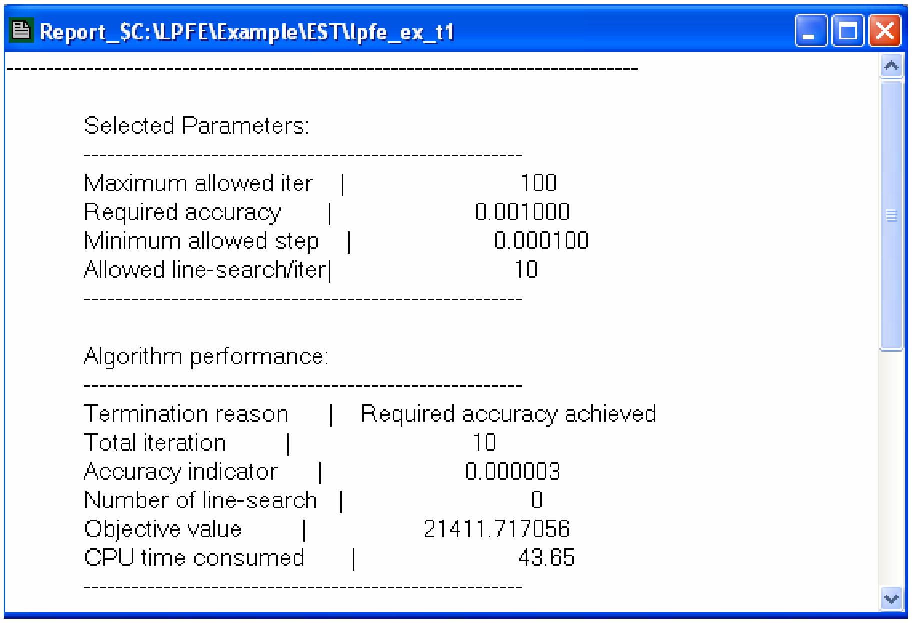
If users want to see the location of a particular path or paths, the Display Path toolbar buttons can be used. To use this function, when the network map and the path table windows are both opened, click at a cell in the path table to make it the active cell. There are four options to display paths:

- Toolbar Button Single Path: When the button is clicked, the path corresponding to the active row of the Path Table will be highlighted.
- Toolbar Button One to All Paths: When the button is clicked, all the paths originating from the origin of the active row of the Path Table will be highlighted.
- Toolbar Button All to One Paths: When the button is clicked, all the paths ending at the destination of the active row of the Path Table will be highlighted.

- Toolbar Button One to One Paths: When the button is clicked, all the paths beginning at the origin and ending at the destination of the active row of the Path Table will be highlighted.

Report Window

Every LPFE estimation generates a text file, documenting summary statistics of the estimation. The estimation statistics of the estimation can be viewed in Visual PFE – TD using the Report Window (Figure 20).



```
-------------------------------------------------------------------------------

          Selected Parameters:
          -------------------------------------------------------
          Maximum allowed iter    |                        100
          Required accuracy       |                   0.001000
          Minimum allowed step    |                     0.000100
          Allowed line-search/iter|                       10
          -------------------------------------------------------

          Algorithm performance:
          -------------------------------------------------------
          Termination reason      |   Required accuracy achieved
          Total iteration         |                     10
          Accuracy indicator      |                 0.000003
          Number of line-search   |                          0
          Objective value         |           21411.717056
          CPU time consumed       |                      43.65
          -------------------------------------------------------
```

Figure 20 Report Window

There are three ways to open a Report Window:

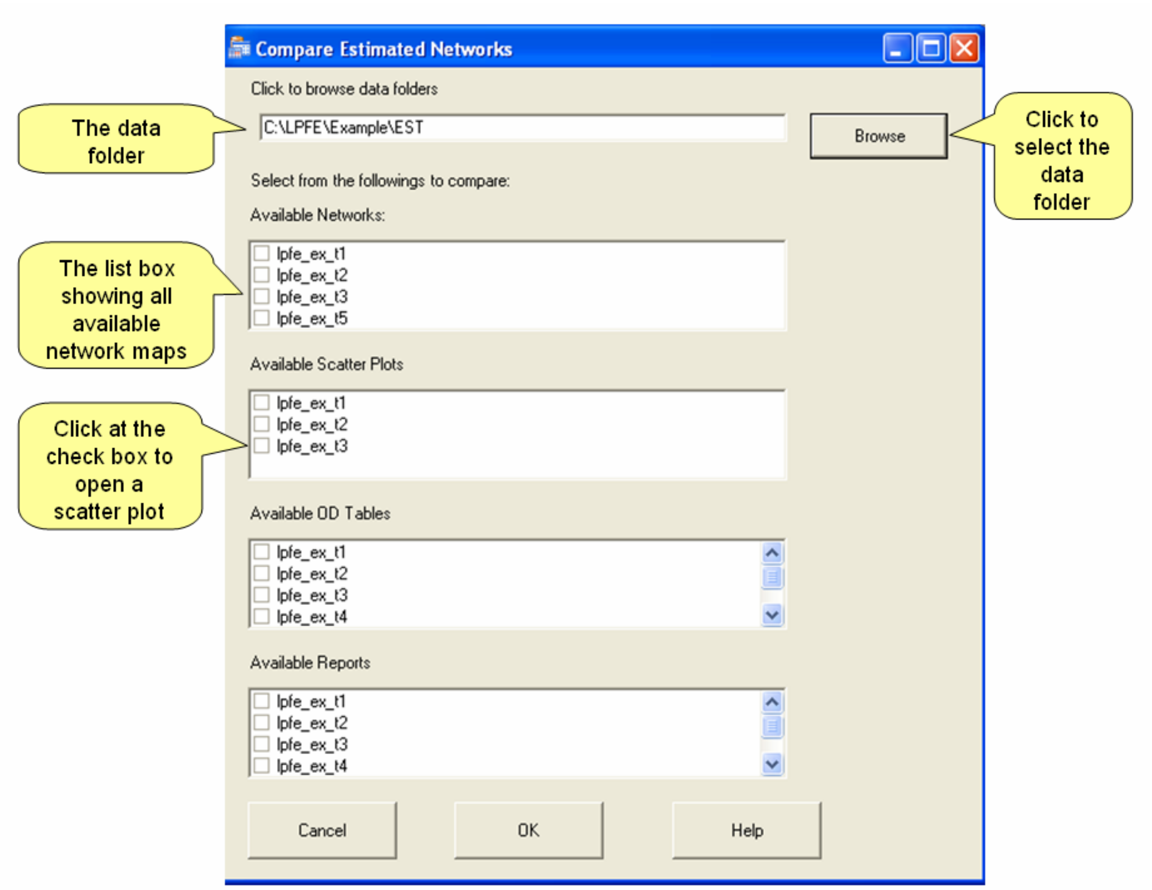
- Select the Report option in the Create Shapefiles and Tables Window.
- When a network map is the active window, go to the menu item Diagnostic/Statistics.
- Go to the menu item Diagnostic/Compare and select from the available reports.

Creating and Editing Text Files

A Report Window is actually a rich text box that can serve as a text editor. User can use the Report Window to edit or create LPFE input network data. To edit an existing text file, go to the menu item File/Open and select one of the LPFE input text files. To create a new file, go to the menu item File/New and a blank Report Window will be opened as a text editor. The menu item Edit contains common text editing tools such as Undo, Cut, Copy, Paste, and Select All. Once a new network data text file is created, the file can be saved by going to the menu item File/Save Text. The Save As dialog box will open for users to enter a desired file name for the text file.

Scenario Comparison Window

If multiple estimation runs have been done for a network, all of the previous results can be opened and compared at the same time. The comparison is setup via the GUI in Figure 21.



Compare Estimated Networks

Click to browse data folders

C:\LPFE\Example\EST

Select from the followings to compare:

Available Networks:

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t5

Available Scatter Plots

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3

Available OD Tables

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t4

Available Reports

- □ lpfe_ex_t1
- □ lpfe_ex_t2
- □ lpfe_ex_t3
- □ lpfe_ex_t4

The data folder

Click to select the data folder

The list box showing all available network maps

Click at the check box to open a scatter plot

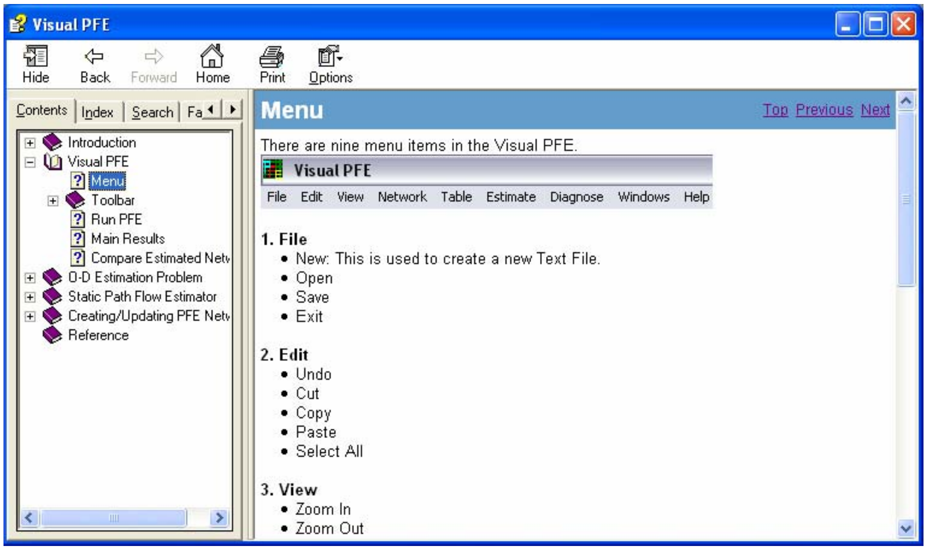
Figure 21 Scenario Comparison Window

To compare different results, click at the Browse button and go to the folder where all estimation results are stored. Once the data folder is selected, all available results will be shown in the corresponding list boxes. Select from the lists and click OK. All the selected result items will be open in corresponding windows.

Note that the available networks and scatter plots are shapefiles. Only those created in the Create Shapefiles and Tables Window can be open for comparison. If the network or scatter plots for a particular time period is never created, it will not be available in the list box. However, the OD tables and reports are text files and they are available for every time period of a LPFE estimation. The text files of the selected OD tables will be converted to a spreadsheet on the fly.

Windows Help

A standard windows help document (Figure 22) is created and built in with Visual PFE. Users can access instruction of particular tasks via the document. To access the Help document, go to the menu item Help/Visual PFE – TD Help. The Help document will be open for user to search for particular information. There are also Help buttons placed on various GUIs. Click at these buttons will also open a particular topic of the Help document.



Menu

There are nine menu items in the Visual PFE.

1. File
- New: This is used to create a new Text File.
- Open
- Save
- Exit

2. Edit
- Undo
- Cut
- Copy
- Paste
- Select All

3. View
- Zoom In
- Zoom Out

Figure 22 Windows Help

Technical Specification

Visual PFE - TD integrates the LPFE program with three other software components. The architecture of the system is shown in Figure 23.

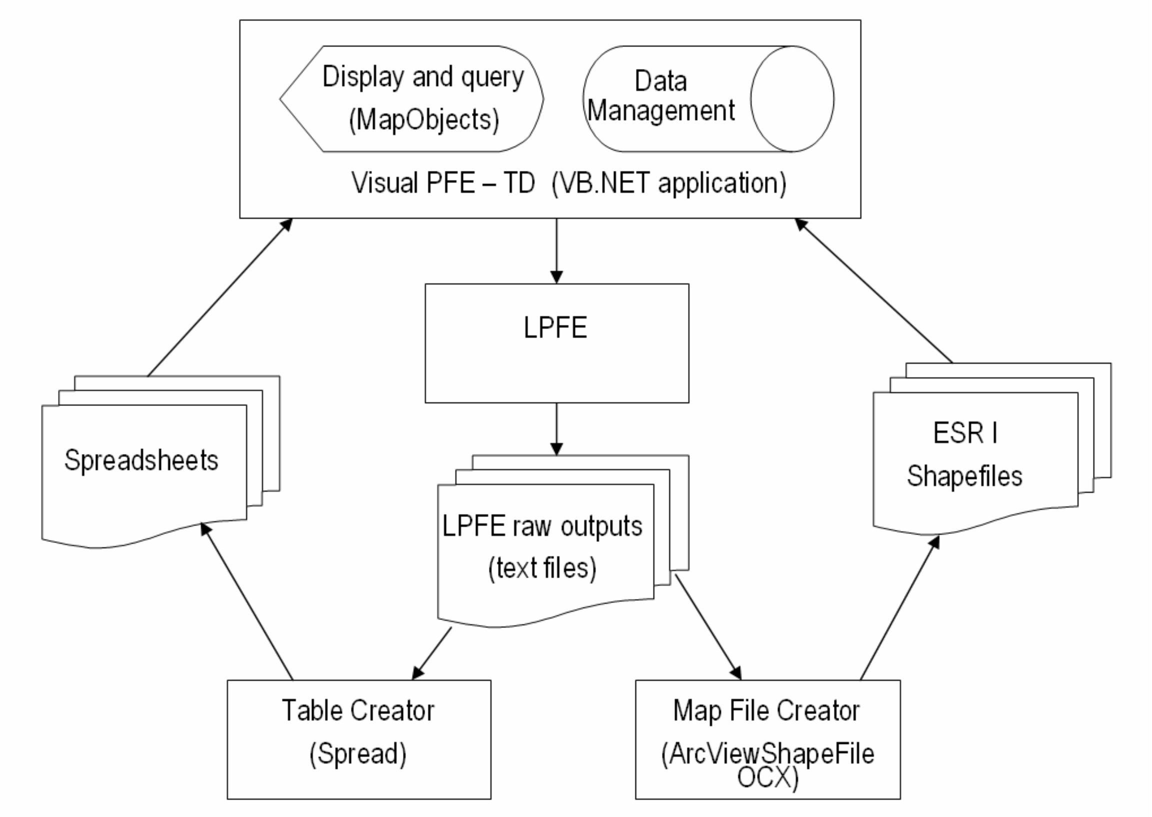


Figure 23 Visual PFE - TD Software Components

The three software components are:

- ArcViewShapeFile OCX
- ESRI MapObjects
- FarPoint Spread

LPFE

The LPFE program operates via the DOS command prompt. If an estimation run is successfully, the output files generated by the LPFE are in text files. If an estimation run encounters errors due to incompatible parameter setting, the LPFE program will quit and error messages will be issued by Visual PFE – TD.

The ArcViewShapefile Read/Write OCX

An OCX is an object-oriented software component conforming to Microsoft's ActiveX architecture. The OCX was created to easily read or write Arcview Shapefiles for data conversion purposes. The Shapefile format is a geographic data format published by ESRI. The ArcViewShapefile Read/Write OCX was created by Ross Pickard in Wellington, New Zealand and is a shareware made available for download from ESRI web site.

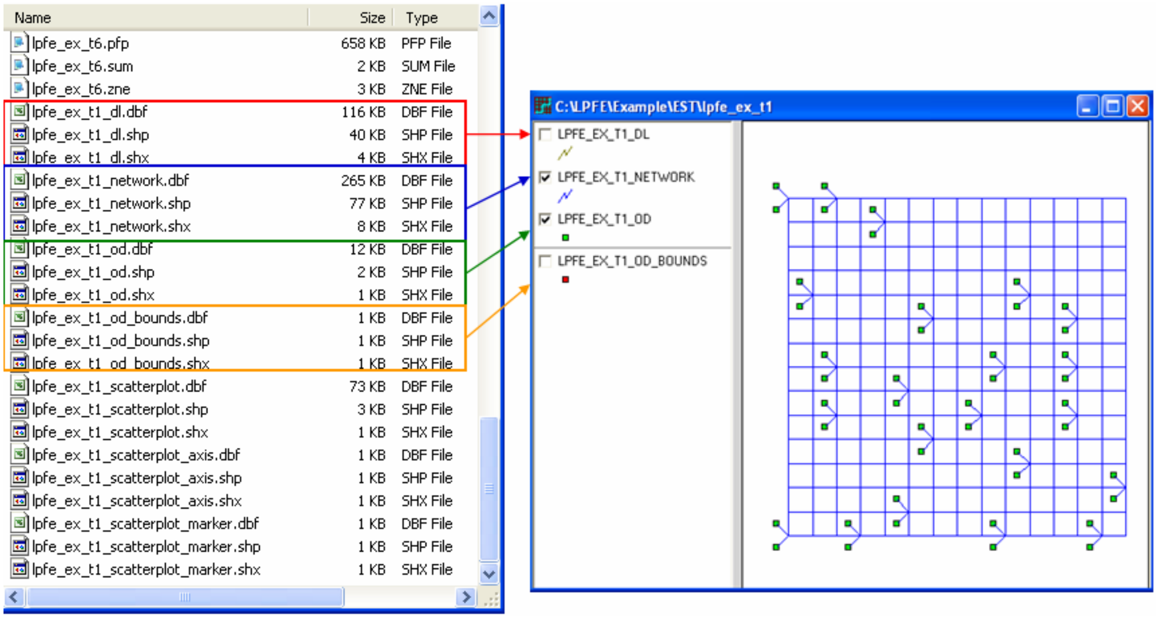
Visual PFE - TD uses the DLL version of the ArcViewShapeFile to convert the text files generated by LPFE to Shapefiles. The converted shapefiles can be displayed with the Visual PFE - TD map component and common GIS software. Table 1 shows the conversion between the LPFE text files to the shapefiles used by Visual PFE.

Table 1 LPFE Output Text Files to Shapefiles

LPFE Output File	Converted Shapefile
*.lfp	Network lines
*.zne	OD points
*.est	Desire lines
*.lfp	Scatter plot points

The Shapefiles of the Network Map and Scatter Plot Windows

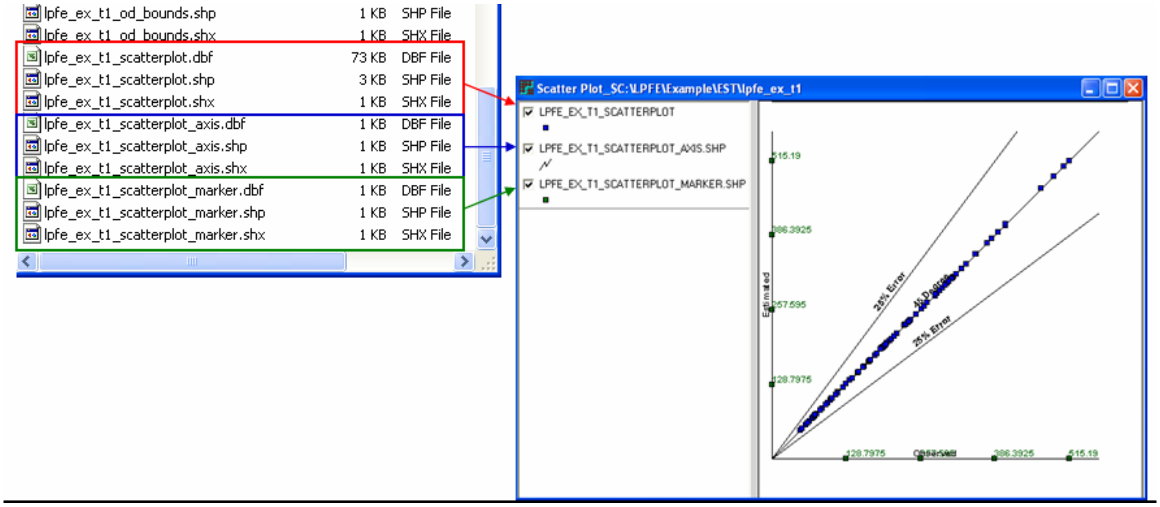
Visual PFE – TD converts the LPFE output text files to shapefiles. These shapefiles are stored in the same folder as the input network data. The shapefiles are used to visually depict the network, the origins, the destinations , and the desire lines. Figure 24 shows a side-by-side view of the list of shapefiles and the legend of the network map layers. It can be seen that three files (i.e., *.shp, *.shx, and *.dbf) are used to draw a layer (i.e., ESRI shapefile specification). The name of the layer is the same as that of the three shapefiles that make up the layer. Each layer name include reference to the input network and the feature (i.e., network, od, desire lines, or boundary points) of the layer.



Name	Size	Type
lpfe_ex_t6.pfp	658 KB	PFP File
lpfe_ex_t6.sum	2 KB	SUM File
lpfe_ex_t6.zne	3 KB	ZNE File
lpfe_ex_t1_dl.dbf	116 KB	DBF File
lpfe_ex_t1_dl.shp	40 KB	SHP File
lpfe_ex_t1_dl.shx	4 KB	SHX File
lpfe_ex_t1_network.dbf	265 KB	DBF File
lpfe_ex_t1_network.shp	77 KB	SHP File
lpfe_ex_t1_network.shx	8 KB	SHX File
lpfe_ex_t1_od.dbf	12 KB	DBF File
lpfe_ex_t1_od.shp	2 KB	SHP File
lpfe_ex_t1_od.shx	1 KB	SHX File
lpfe_ex_t1_od_bounds.dbf	1 KB	DBF File
lpfe_ex_t1_od_bounds.shp	1 KB	SHP File
lpfe_ex_t1_od_bounds.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot.dbf	73 KB	DBF File
lpfe_ex_t1_scatterplot.shp	3 KB	SHP File
lpfe_ex_t1_scatterplot.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_axis.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_axis.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_axis.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_marker.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_marker.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_marker.shx	1 KB	SHX File

Figure 24 Shapefiles and Network Map Layers

Figure 24 shows a side-by-side view of the list of shapefiles and legend of the Scatter Plot layers.



lpfe_ex_t1_od_bounds.shp	1 KB	SHP File
lpfe_ex_t1_od_bounds.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot.dbf	73 KB	DBF File
lpfe_ex_t1_scatterplot.shp	3 KB	SHP File
lpfe_ex_t1_scatterplot.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_axis.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_axis.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_axis.shx	1 KB	SHX File
lpfe_ex_t1_scatterplot_marker.dbf	1 KB	DBF File
lpfe_ex_t1_scatterplot_marker.shp	1 KB	SHP File
lpfe_ex_t1_scatterplot_marker.shx	1 KB	SHX File

Scatter Plot_SC:\LPFE\Example\EST\lpfe_ex_t1

☑ LPFE_EX_T1_SCATTERPLOT

☑ LPFE_EX_T1_SCATTERPLOT_AXIS.SHP

☑ LPFE_EX_T1_SCATTERPLOT_MARKER.SHP

Figure 25 Shapefiles and the Scatter Plot Layers

ESRI MapObjects

MapObjects is a set of mapping software components that let users add GIS maps to software applications. MapObjects comprises an ActiveX control (OCX) called the Map control and a set of over forty-five ActiveX Automation objects. In Visual PFE, a map control is used to display the shpafiles created by the ArcViewShapeFile component. The map control facilitates all of the GIS functions for the converted shapefiles. Another OCX control is added to a map form to display the legends of different map layers. The legend control is internally connected to the map control. A layer can be made active through the legend control. On the active layer, labels and thematic mapping can be applied to features on the layer. For example, networks links can be varied by colors and width based on estimated link flow. Bar charts can be applied to display the varying magnitude of production and attraction at the origins and destinations.

FarPoint Spread

Spread by the FarPoint Technologies is a software component built to Microsoft's latest .NET software architecture. Spread is used to create spreadsheet applications. The strength of Spread is that it facilitates all of the typical spreadsheet functions with a dimension limitation (1 million columns by 1 million rows) that is unlikely to be violated by any OD tables. Visual PFE contains codes that process the PFE CSV outputs and formulate the spreadsheets stored and displayed with Spread. Once a spreadsheet is created in Spread, it can be saved as a Microsoft Excel file. Table 2 shows the association between the original PFE CSV outputs and the formatted Spread tables.

Table 2 LPFE Output Files to Spread Tables

LPFE Output Files	Converted Spread Table
*.est	OD table
*.pfp	Path table

Appendix III

Visual PFE® 1.0

A Quick Start Tutorial

June 2006

Visual PFE and Visual PFE–TD are authored at the Utah State University. Copyright application for Visual PFE and Visual PFE-TD is currently in process. Please do not quote or reference the names without the authors' permission.

A Quick Start Tutorial

This tutorial is designed to give users the essential instruction to begin using Visual PFE for OD estimation in a timely fashion. The tutorial uses step-by-step examples to help users become familiar with the GUIs and the functions of the program. At the end, an example is provided to show users how to prepare the input network data.

In this tutorial, you will learn how to:

- Estimate OD trip tables for a particular network
- Navigate through the Map and Scatter Plot Windows
- Post on Features of a Layer
- Manage OD Tables
- Visualize Paths
- View Reports and Text Files
- Edit Networks
- Compare Previously Estimated Results
- Create Networks

Example 1: Estimate OD Tables for a Particular Network

In this example, you are given a PFE network named 9GRD, which consists of three separate text files: 9GRD.pfe, 9GRD.pfx, and 9GRD.pfp (You will learn how to create these three files in the final example).

How to Run a PFE Estimation

To estimate an OD table for the 9GRD network:

1. Open the Visual PFE program, select menu item **Estimate/Static** to open the Static PFE Estimation GUI (see Figure 1).

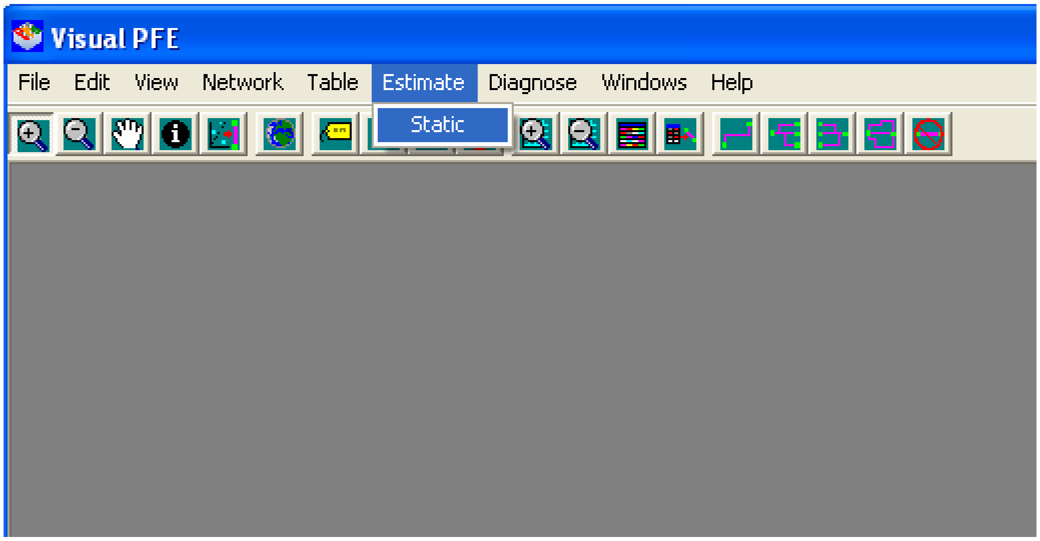
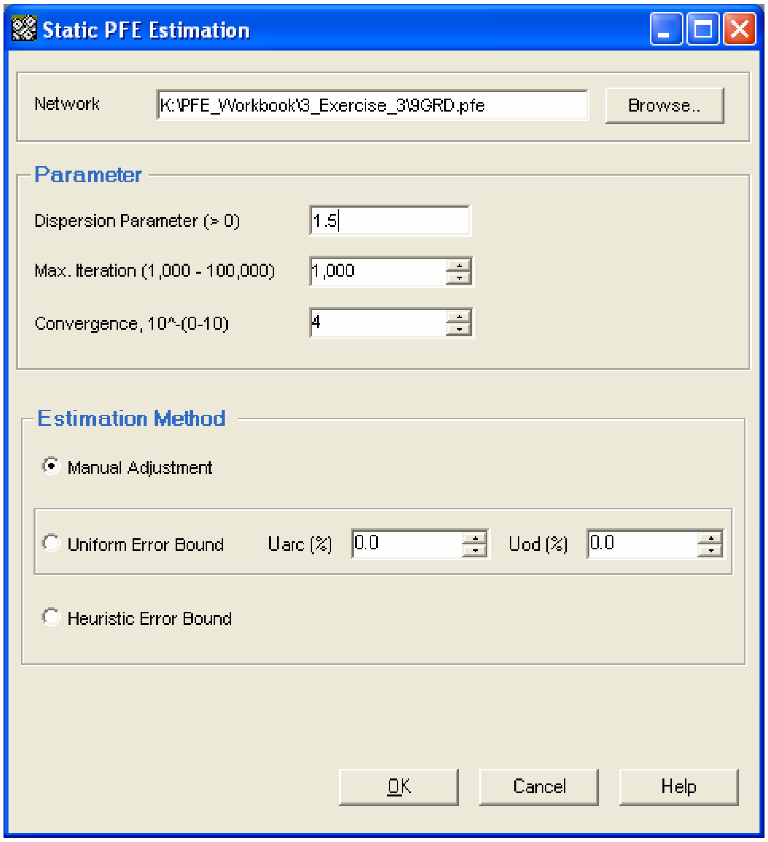


Figure 1 Estimate/Static

2. On the Estimation GUI, click at the **Browse** button to find the folder "*PFE_workbook\3_Exercise_3\9GRD.pfe*",
3. Enter "*1.50*" for **Dispersion Parameter**,
4. Select "*5,000*" for **Max. Iteration**,
5. Select "*4*" (⇒ 0.0001) for **Convergence level**,

6. Select **Manual Adjustment** (Figure 2) (to use measurement errors specified in the input files),
7. Click “**OK**” to run PFE estimation.



Static PFE Estimation

Network K:\PFE_Workbook\3_Exercise_3\9GRD.pfe

Parameter

Dispersion Parameter (> 0) 1.5

Max. Iteration (1,000 - 100,000) 1,000

Convergence, 10^-(0-10) 4

Estimation Method

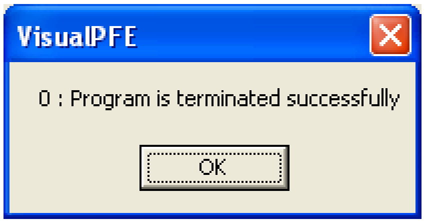
● Manual Adjustment

○ Uniform Error Bound Uarc (%) 0.0 Uod (%) 0.0

○ Heuristic Error Bound

Figure 2 Manual Adjustment

8. After the run is completed, the dialog box informing the status of the estimation appears (Figure 3).



VisualPFE

0 : Program is terminated successfully

Figure 3 Status of Estimation

Remarks:

If errors occur during the estimation, the dialog box will inform the users the specific errors and the suggested solution. At the end of this example, technical notes about the estimation methods, the parameters, and the potential error messages are provided for your reference. For complete coverage of the technical details on PFE, please refer to the *PFE Workbook*.

9. Visual PFE will display the network and estimated O-D trip table (Figure 4).

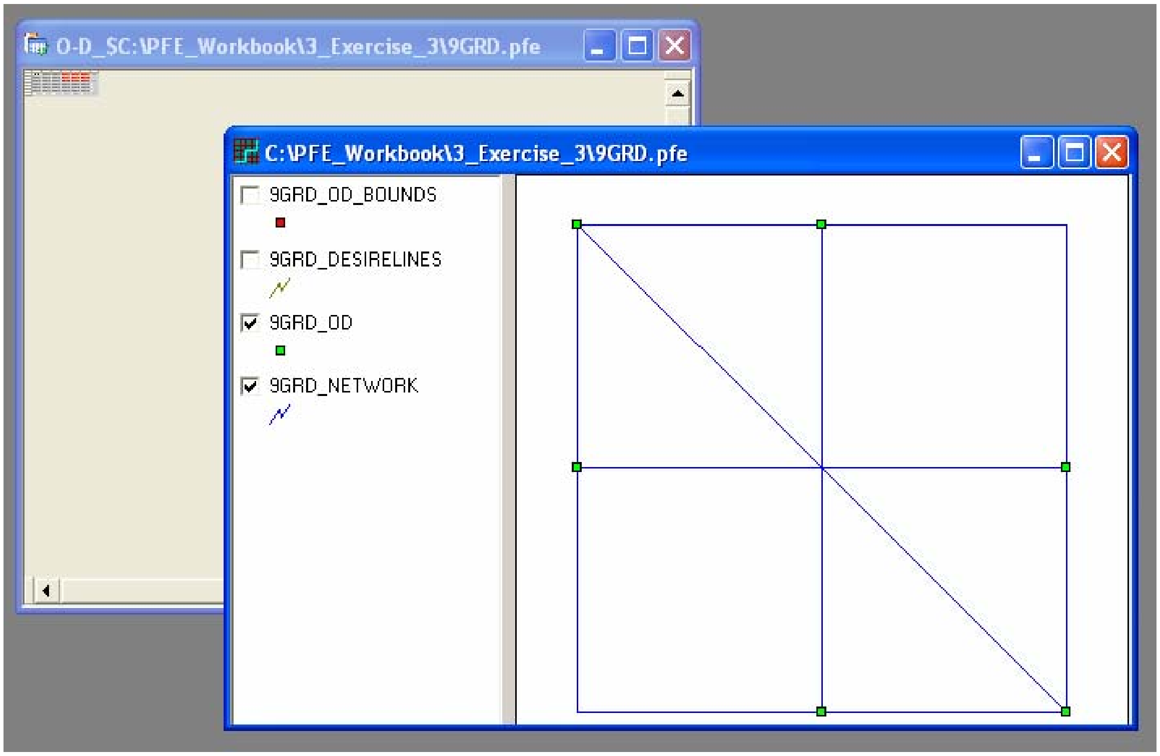
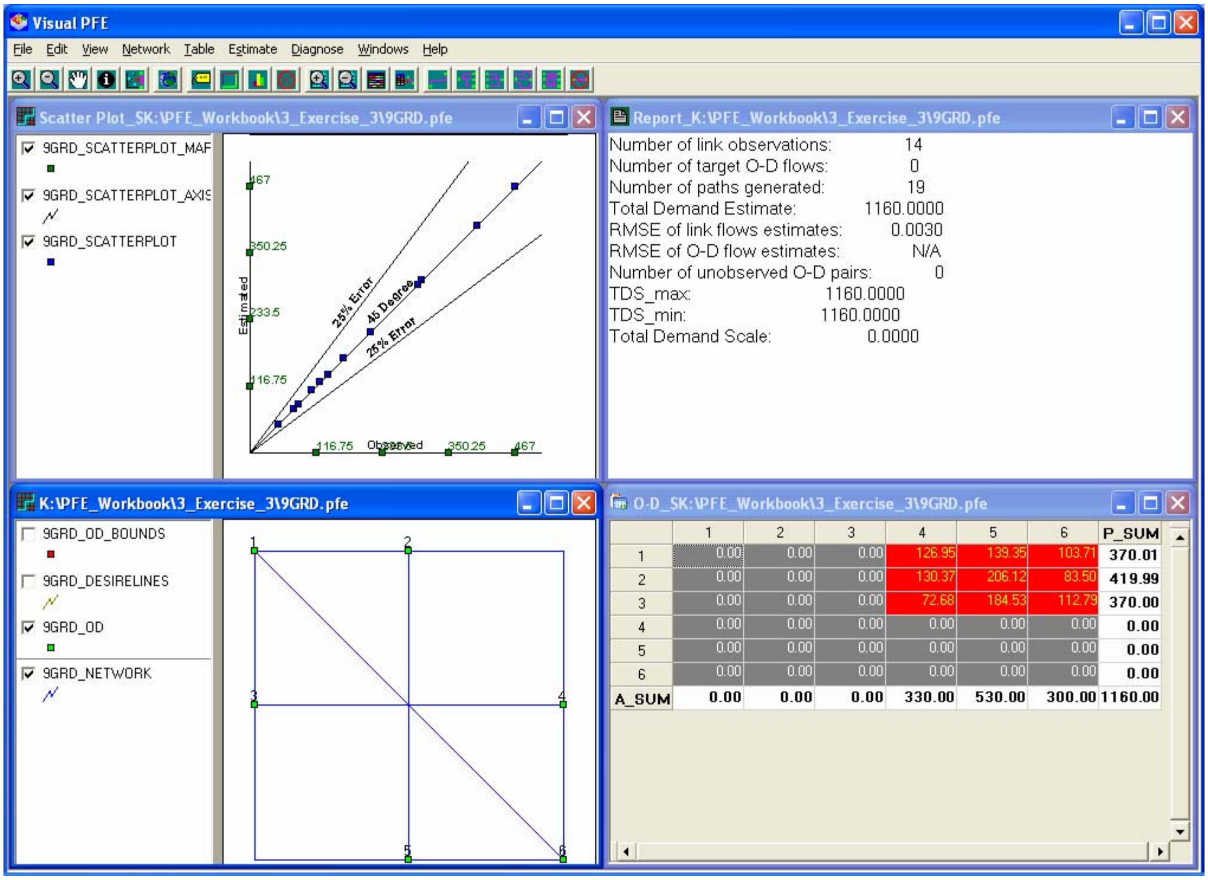

O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
□ 9GRD_OD_BOUNDS
□ 9GRD_DESIRELINES
☑ 9GRD_OD
☑ 9GRD_NETWORK

Figure 4 Network and Estimated O-D Trip Table

10. Open the scatter plot and the estimation report (Figure 5).
    a. Make the network map window the active window (click on the title bar of the map window),
    b. Select **Diagnose** >> **Statistics**,
    c. Select **Diagnose** >> **Graphs**,
    d. Select **Windows** >> **Tile**


Visual PFE
File Edit View Network Table Estimate Diagnose Windows Help
Scatter Plot_SK:\PFE_Workbook\3_Exercise_3\9GRD.pfe
9GRD_SCATTERPLOT_MAF
9GRD_SCATTERPLOT_AXIS
9GRD_SCATTERPLOT
467
350.25
233.5
116.75
Estimated
Observed
25% Error
45 Degree
25% Error
Report_K:\PFE_Workbook\3_Exercise_3\9GRD.pfe
K:\PFE_Workbook\3_Exercise_3\9GRD.pfe
9GRD_OD_BOUNDS
9GRD_DESIRELINES
9GRD_OD
9GRD_NETWORK
1
2
3
4
5
6
O-D_SK:\PFE_Workbook\3_Exercise_3\9GRD.pfe

```
Number of link observations:        14
Number of target O-D flows:          0
Number of paths generated:          19
Total Demand Estimate:          1160.0000
RMSE of link flows estimates:       0.0030
RMSE of O-D flow estimates:          N/A
Number of unobserved O-D pairs:       0
TDS_max:                   1160.0000
TDS_min:                   1160.0000
Total Demand Scale:           0.0000
```

	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

Figure 5 The Scatter Plot and the Estimation Report

By now, you have completed the estimation process for the given 9GRD network. All of the output components are generated and displayed in corresponding windows. After reviewing the notes about PFE parameters, you will learn functions of each individual output window in the next examples.

What are the Static PFE Estimation Parameters?

In order to perform the estimation using the static PFE, five input parameters are required.

1. Network: name of the main PFE network file (*.pfe) including full path of the directory (simply browse for location of the file).
2. Dispersion parameter: cost-sensitivity in the logit SUE model. It indicates how sensitive the road users are to the travel cost. A small value of "dispersion parameter" (e.g., between 0.01 and 1.00) is usually recommended to prevent the numerical problem of the algorithm. It is however important to calibrate this parameter to properly reflect the behavior of road users (before the estimation).
3. Max. Iteration: the maximum number of iterations allowed for the adjustment of path flows to match observed link volumes. The default value (recommended) is set at 1,000. Note that the algorithm could terminate before 1,000 iterations if it reaches convergence. The maximum iteration allowed to set is 100,000.
4. Convergence: the criterion used for terminating the PFE. The program will check for the maximum change of balancing factors. If the maximum change is less than the convergence criterion, the program will be terminated. The convergence value of 10^{-4} is recommended to obtain a reasonable solution within a reasonable computational time. A more strict convergence criterion (e.g., smaller value – $<10^{-4}$) can also be used, of course, with the expense of higher computational time.
5. Method to handle inconsistency: the method for handling the inconsistency of traffic data (e.g., traffic counts, etc.) in the estimation. In the static PFE, there are three options available:

   5.1. *Manual adjustment* allows the users to manually specify the error for each individual observation (i.e., percentage error) through the first PFE network file (*.pfe).

   5.2. *Uniform adjustment* uniformly specifics the same measurement error across all observations. Percentage error (e.g., 5%) needs to be specified by the user in the provided text box (see Figure 3). Noted that:

   - *The value of percentage error must be positive and could be greater than* 100 *percent if highly inconsistent data are expected. If this option is selected, the error bounds specified in the input file (*.pfe) will be disregarded.*
   - *Large measurement errors (e.g., >* 50 *percent) could be employed in the estimation to ensure the existence of feasible solution and fast convergence (i.e., ease to obtain the solution),*
   - *However, it was observed that an unreasonably large measurement error could lead to the underestimation of travel demand in the network (***Chen et al., 2004)**[1]. *A balance between solution quality and computational time requirement should be cautiously considered.*

[1] Chen, A., Chootinan, P., Recker, W., and Zhang, H.M. 2004. Development of a Path Flow Estimator for deriving steady-state and time-dependent origin-destination trip tables. California PATH Research Report, UCB-ITS-PRR-2004-29.

5.3. *Heuristic adjustment* utilizes the information (e.g., the difficulty of matching individual observations) obtained after the estimation process to automatically adjust (expand) the error bounds and re-estimates.

What are the PFE Return Codes?

A set of return codes has been developed to report possible sources of error and possible remedies for the execution of PFE.

Code	**Description**	**Solution**
0	Program is terminated successfully.	-
1	At least one of the required inputs is missing.	• *Check the existence of input files* (*input folder*)
2	Size of the working network exceeds capacity of the program; current capacity is 5,000 nodes, 10,000 links, 500 zones, and 250,000 OD pairs	-
3	Dispersion parameter (cost sensitivity) is too high, which is not suitable for the current unit of travel cost of the working network.	• *Consider the reduction of dispersion parameter*
4	Path building mechanism fails (e.g., impossible to build paths passing through some observed links).	• *Check the locations of problematic traffic counts*, • *Check network connectivity*, or • *Provide paths in the input to cover such observation*
5	The execution exceeds the maximum number of iterations specified by the user. By examining the trend of convergence, the algorithm seems to diverge.	• *Enlarge the measurement errors*
6	The execution exceeds the maximum number of iterations specified by the user. By examining the trend of convergence, the algorithm, however, seems to converge.	• *Enlarge the measurement errors*, or • *Increase the maximum number of iterations*

Remarks:

- *With the return codes 1, 2, and 3, PFE outputs will not be created.*
- *The estimation result is meaningful only when the program is terminated with the return code 0.*
- *With the return codes 4, 5 and 6, the outputs are however created so that the user can review the problem associated to the estimation through the map, scatter plot, etc.*

Example 2: Navigate through the Network Map and Scatter Plot Windows

In this example, we will begin start the example by working with the Map and Scatter Plot windows created in example 1.

How to Make a Window Active

1. Click at the title bar of the Network Map Window to make it the active window (see Figure 6). The map window should appear on top of every other opened window. See for yourself how the titles of the OD table, scatter plot, path table, and report all include the name and the file path of the network (i.e., the title of the fist window).

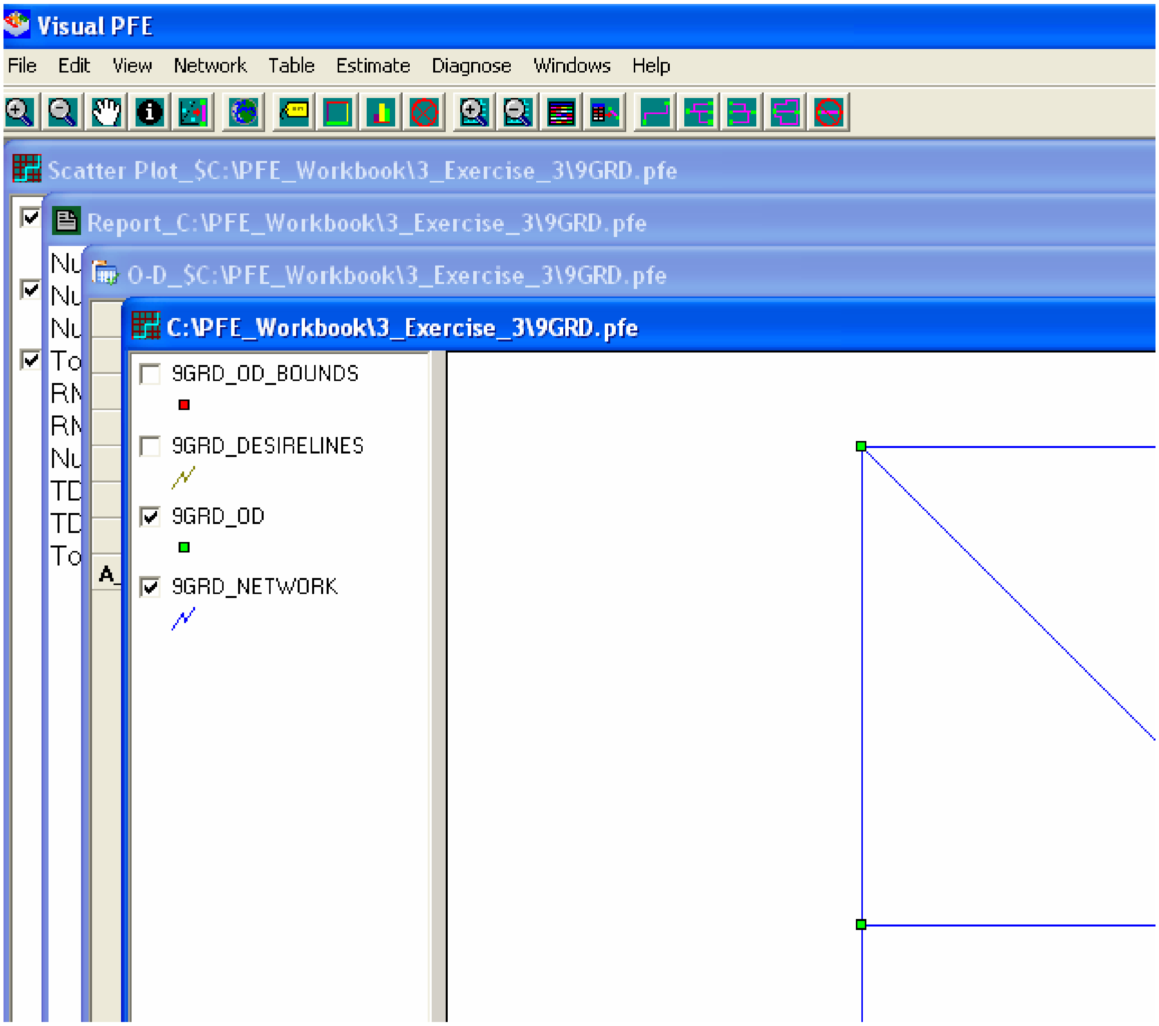

Visual PFE
File Edit View Network Table Estimate Diagnose Windows Help
Scatter Plot_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
Report_C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
□ 9GRD_OD_BOUNDS
□ 9GRD_DESIRELINES
☑ 9GRD_OD
☑ 9GRD_NETWORK

Figure 6 Active Window

How to Make a Layer Active

2. You can make the layer 9GRD_NETWORK active by clicking at the title of in the legend area. A thin line will appear underneath the layer name in the legend, indicating the current active layer (Figure 7).

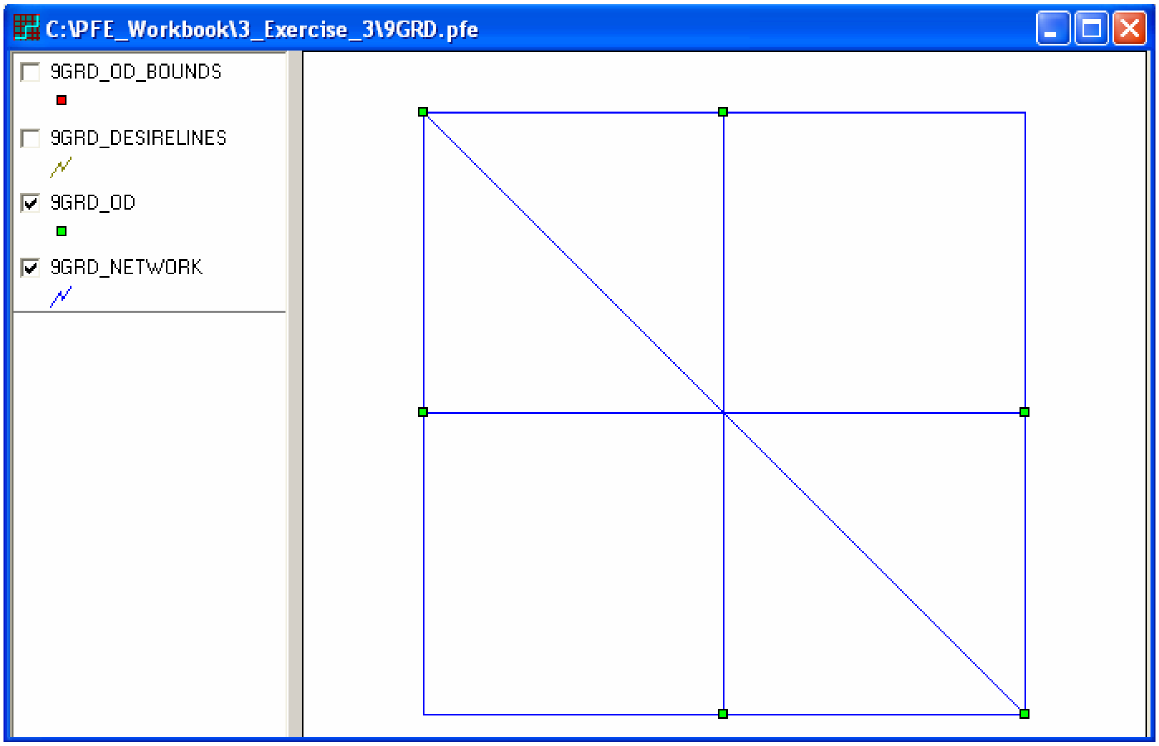
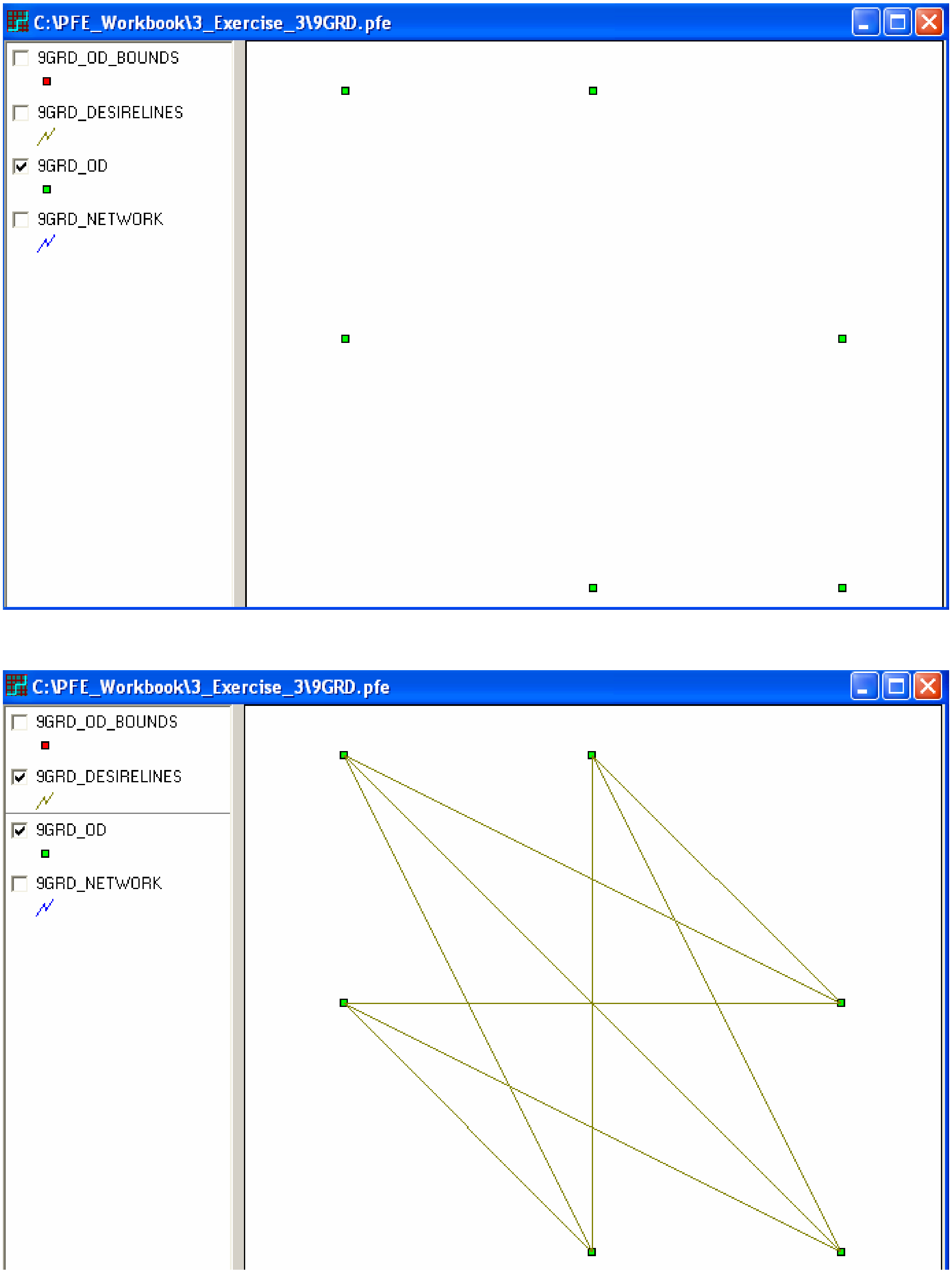


Figure 7 Legend and Active Layer

How to Make a Layer Invisible

3. You can make a layer invisible by un-checking the check box next to the name of the layer in the legend. For example, un-checking the legend of the 9GRD_NETWORK layer will make the layer invisible. Checking the legend of the 9GRD_DESIRELINES will make the desire lines visible. Once you see how this work, return the map to the original setting (make desire lines invisible and the network links visible) (Figure 8).



C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

☐ 9GRD_OD_BOUNDS 🟥

☐ 9GRD_DESIRELINES

☑ 9GRD_OD 🟩

☐ 9GRD_NETWORK

C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

☐ 9GRD_OD_BOUNDS 🟥

☑ 9GRD_DESIRELINES

☑ 9GRD_OD 🟩

☐ 9GRD_NETWORK

Figure 8 Invisible/Visible Layer

How to Increase the Legend Width

4. Sometimes you may have a map layer with a long file name that does not fit in the initial legend width. You can increase the width of the legend by clicking at the border bar between the legend and the map (the mouse cursor will turn into a east-west double arrow). Moving the border bar while holding the mouse button will move the border bar to the desired direction (Figure 9).

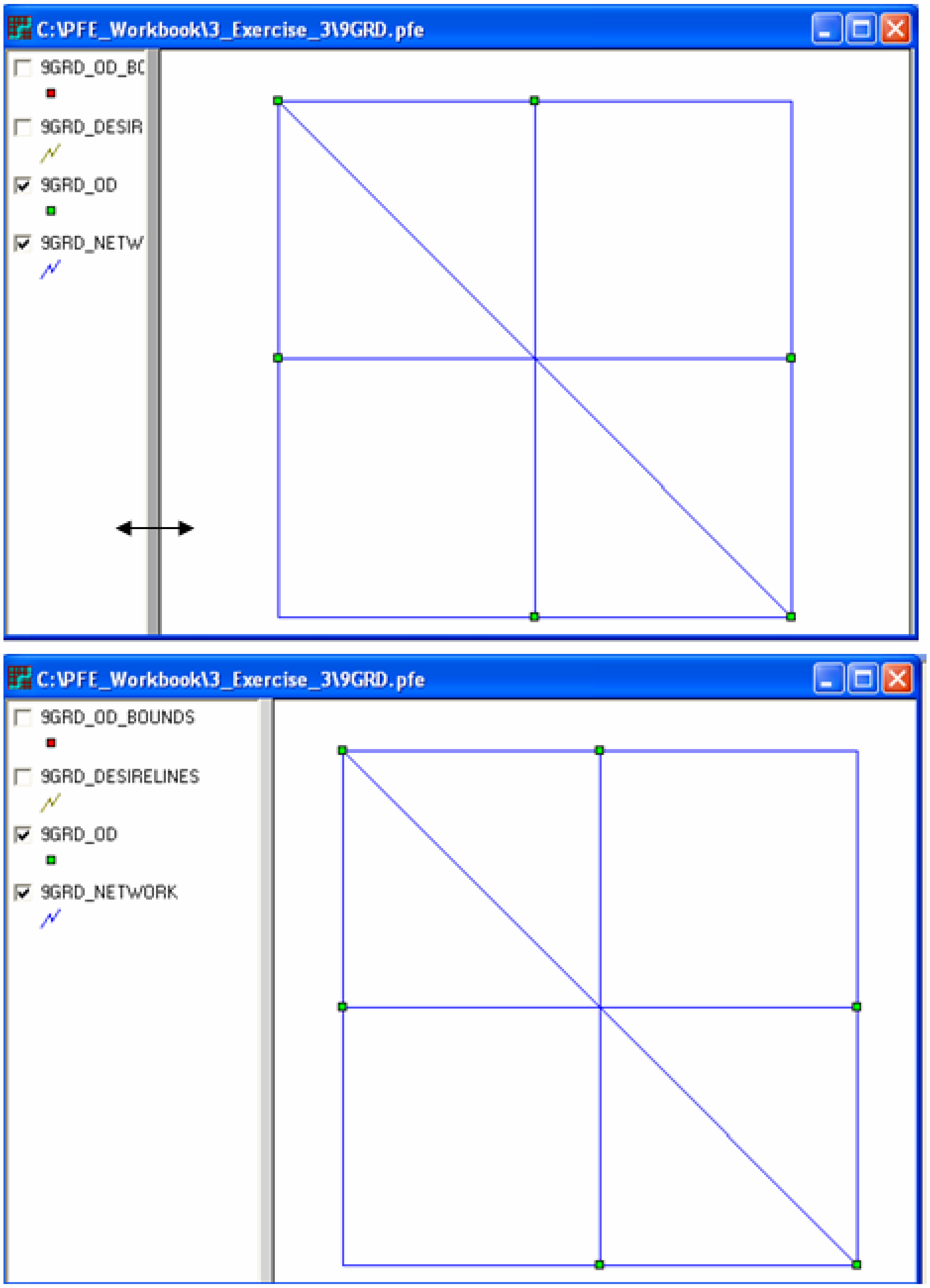
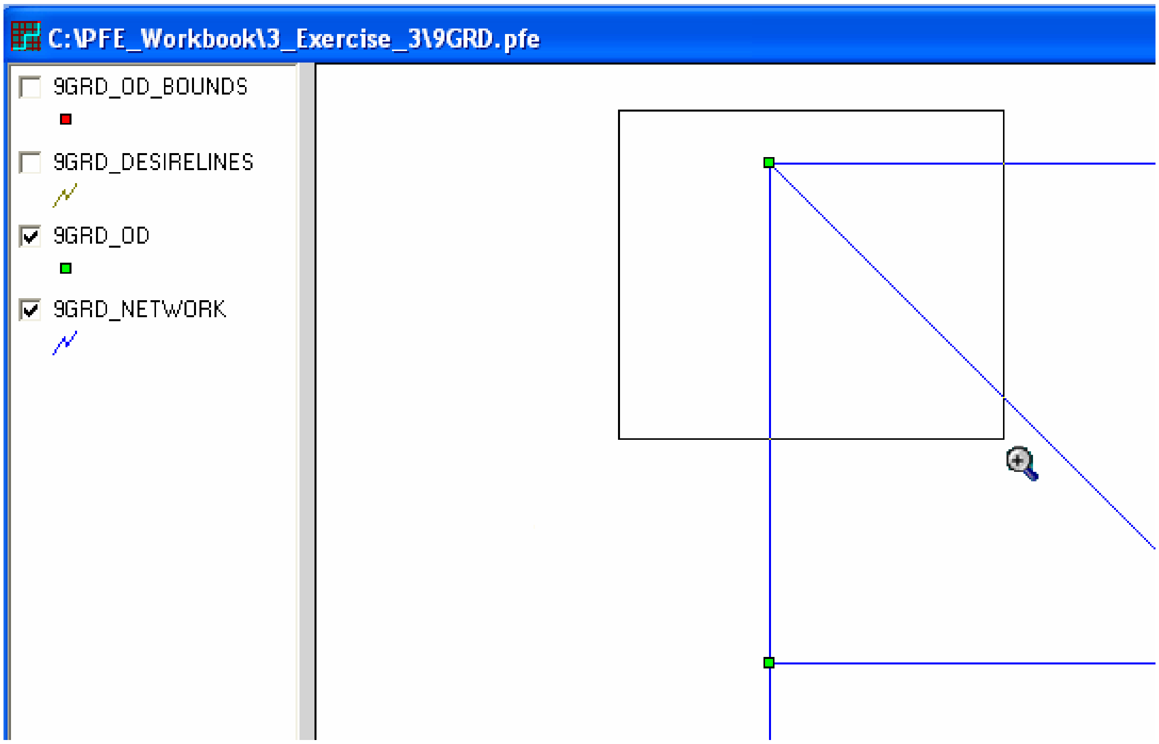


Figure 9 Increase of Legend Width

How to Navigate through the Network Map (Zoom In, Zoom Out, Pan, Full Scale, and Identify Features)

5. Click at the toolbar button Zoom In . Click at a point on the map where you want to zoom in (the mouse cursor will appear as a magnifying glass.) and drag a rectangle with the mouse cursor. The rectangle will be the extent to which the map will be zoomed in. Once you are satisfied with the extent of the rectangle, release the mouse button. The map will be zoom in accordingly (Figure 10).



C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

□ 9GRD_OD_BOUNDS

□ 9GRD_DESIRELINES

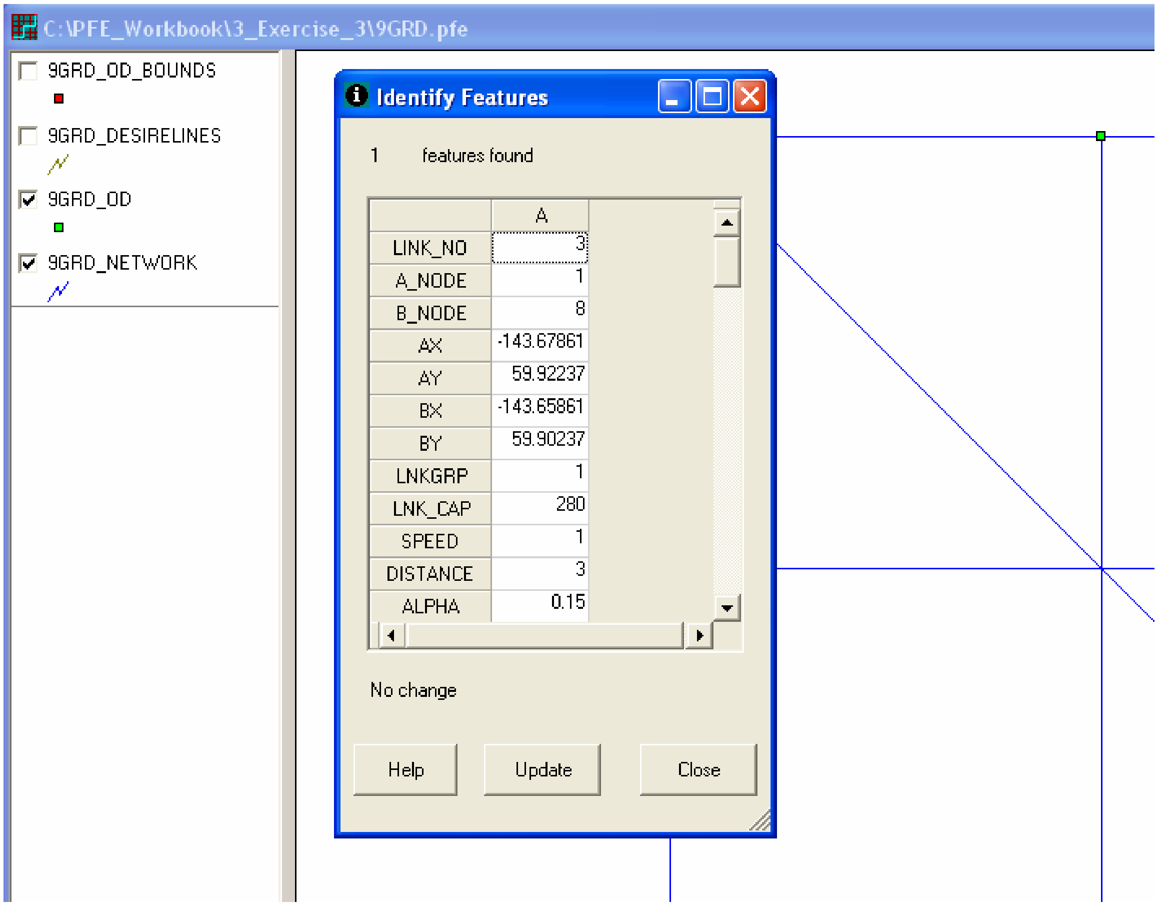
☑ 9GRD_OD

☑ 9GRD_NETWORK

Figure 10 Navigation through the Network Map

6. While the map is zoomed-in, you can move the displayed portion of the map to another spot by pressing down the toolbar button Pan . Clicking at the map then will turn the mouse cursor to a palm symbol. Moving the mouse cursor while holding down the mouse button will allow you to move the map window to another spot on the network.

7. To zoom out, click at the toolbar button Zoom Out . The map will be zoomed out by a certain proportion. If you are not satisfied with the scale, click at the button again will further reduce the map scale.
8. If you want to return the map to the original full scale of the network map, simply press the Full Scale button .
9. If you are interested in finding out the attribute of a particular map feature, you can use the toolbar button Identify ℹ. To do this, first make the feature layer the active layer by clicking at the name of the layer in the legend. A line underneath the layer name will appear in the legend. Press down the Identify button (the mouse cursor will turn into the information symbol ℹ. Click at a feature of the layer will open the Identify Feature Window, showing all the attributes of the feature identified (Figure 11).



C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

□ 9GRD_OD_BOUNDS

□ 9GRD_DESIRELINES

☑ 9GRD_OD

☑ 9GRD_NETWORK

Identify Features

1 features found

	A
LINK_NO	3
A_NODE	1
B_NODE	8
AX	-143.67861
AY	59.92237
BX	-143.65861
BY	59.90237
LNKGRP	1
LNK_CAP	280
SPEED	1
DISTANCE	3
ALPHA	0.15

No change

Figure 11 Identify Features

How to Navigate through the Scatter Plot Window (Display Link)

A scatter plot has the same form as a network map window. The navigation functions for the network map also apply to the scatter plot window. The only exception is that a scatter plot has one exclusive function, the Display Link toolbar button , which allows users to see the location of the link for which a scatter plot point represents. To use this function:

10. Make the scatter plot window the active window by clicking at the title bar of the scatter plot window.
11. Make the 9GRD_SCATTERPLOT layer the active layer by clicking at the name of the layer in the legend area.
12. Press down the Display Link button . Move the mouse cursor over the scatter plot and click at a particular scatter plot point.
13. The network map window will automatically appears on top and become the active window with the corresponding link highlighted (Figure 12).

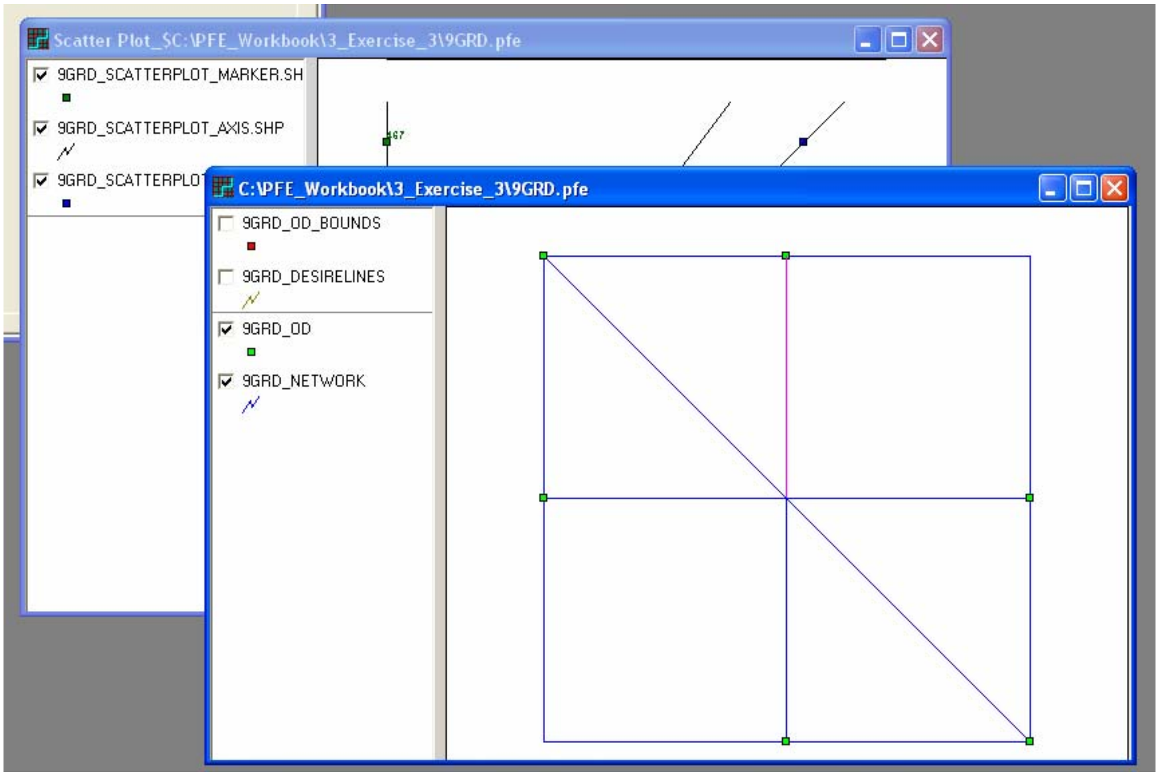

Scatter Plot_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
☑ 9GRD_SCATTERPLOT_MARKER.SH
☑ 9GRD_SCATTERPLOT_AXIS.SHP
☑ 9GRD_SCATTERPLO
C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
☐ 9GRD_OD_BOUNDS
☐ 9GRD_DESIRELINES
☑ 9GRD_OD
☑ 9GRD_NETWORK

Figure 12 Display Link Features

How to Clear Track Lines

14. To clear the magenta highlight from the map, when the network map window is the active window, press the Clear Track Lines button ⊘. The button also applies to the magenta highlights on desire lines and paths.

Example 3 Feature Postings

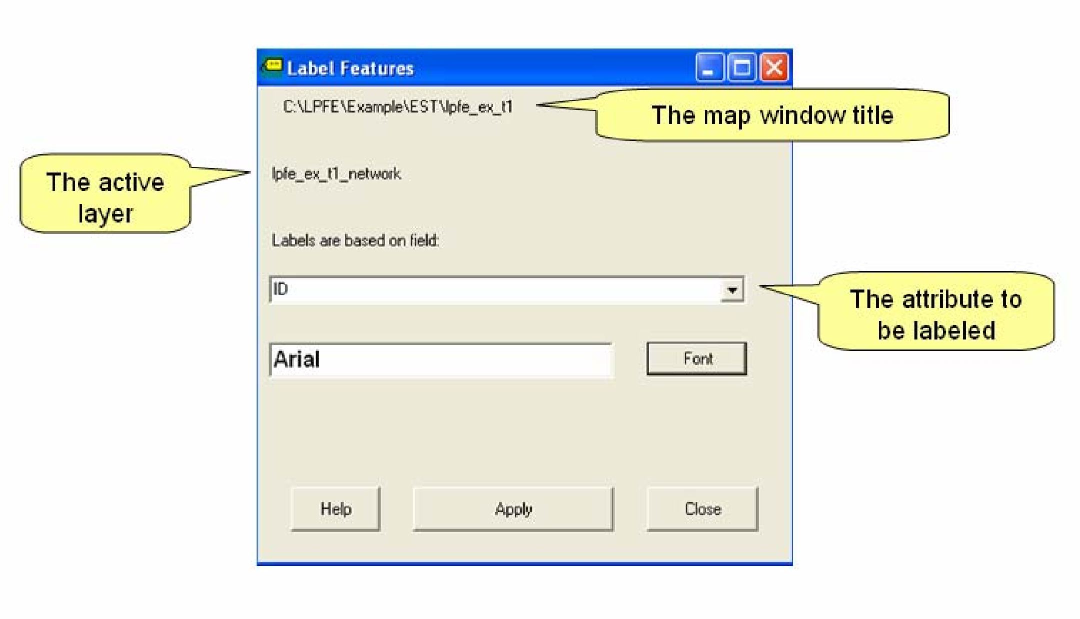
With Visual PFE, you can place three types of postings on features of a map layer based on the values of the features. The postings apply to both the network map window and the scatter plot windows. The three types of postings are:

- Label
- Class Map
- Bar Chart Map

Placing of all three postings require setting an active layer.

How to Label Features of a Layer

1. Make the layer of the features active
2. Press down the Label Feature button . The Label Feature Window will open (Figure 13).
3. Once the form appears, a particular attribute can be selected for labeling and the font for the label can also be adjusted using the Font button.
4. Click the Apply button the labels will be placed next to the features on a map (Figure 14).



C:\LPFE\Example\EST\lpfe_ex_t1

lpfe_ex_t1_network

Labels are based on field:

ID

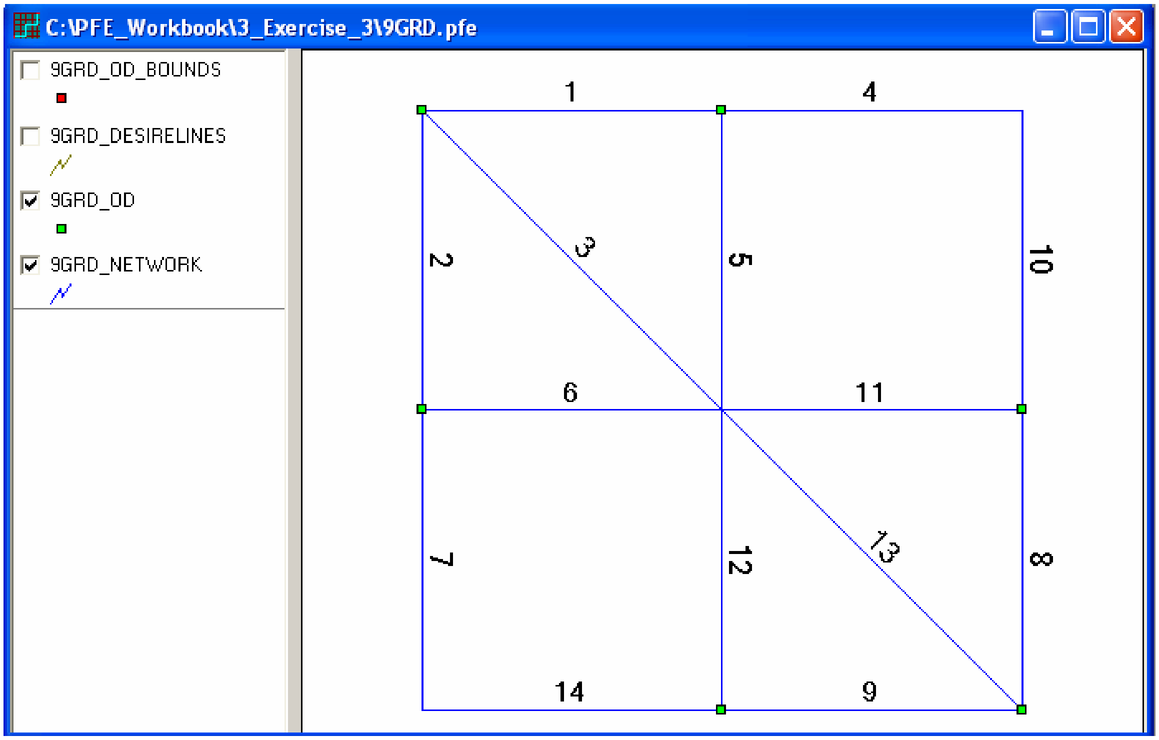
Arial

The map window title

The active layer

The attribute to be labeled

Figure 13 Label Features



C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

☐ 9GRD_OD_BOUNDS

☐ 9GRD_DESIRELINES

☑ 9GRD_OD

☑ 9GRD_NETWORK

Figure 14 Label of Link ID

How to Clear Labels

5. To clear the labels, press the Clear Postings button ⊗. The function applies to all three types of postings.

<u>Remarks</u>:

When a scatter plot shows up, the scatter plot axis layer and the marker layer are already labeled with the appropriate labels. Press down the Clear Postings button will clear these labels as well. To restore the original setting, just label the axis layer with the attribute "NAME" and the marker layer with "VALLUE" again.

After a particular posting is cleared from a map, the legend is reset. To place another posting, you need to remember to set the active layer again (Figure 15).

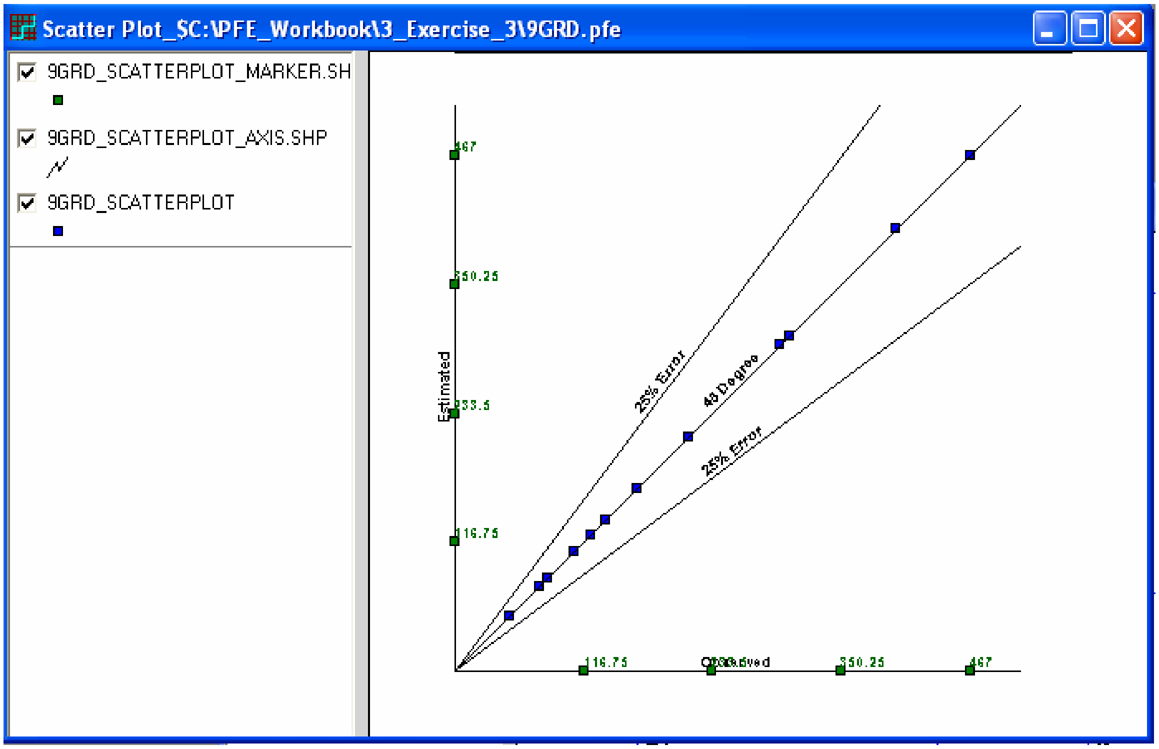
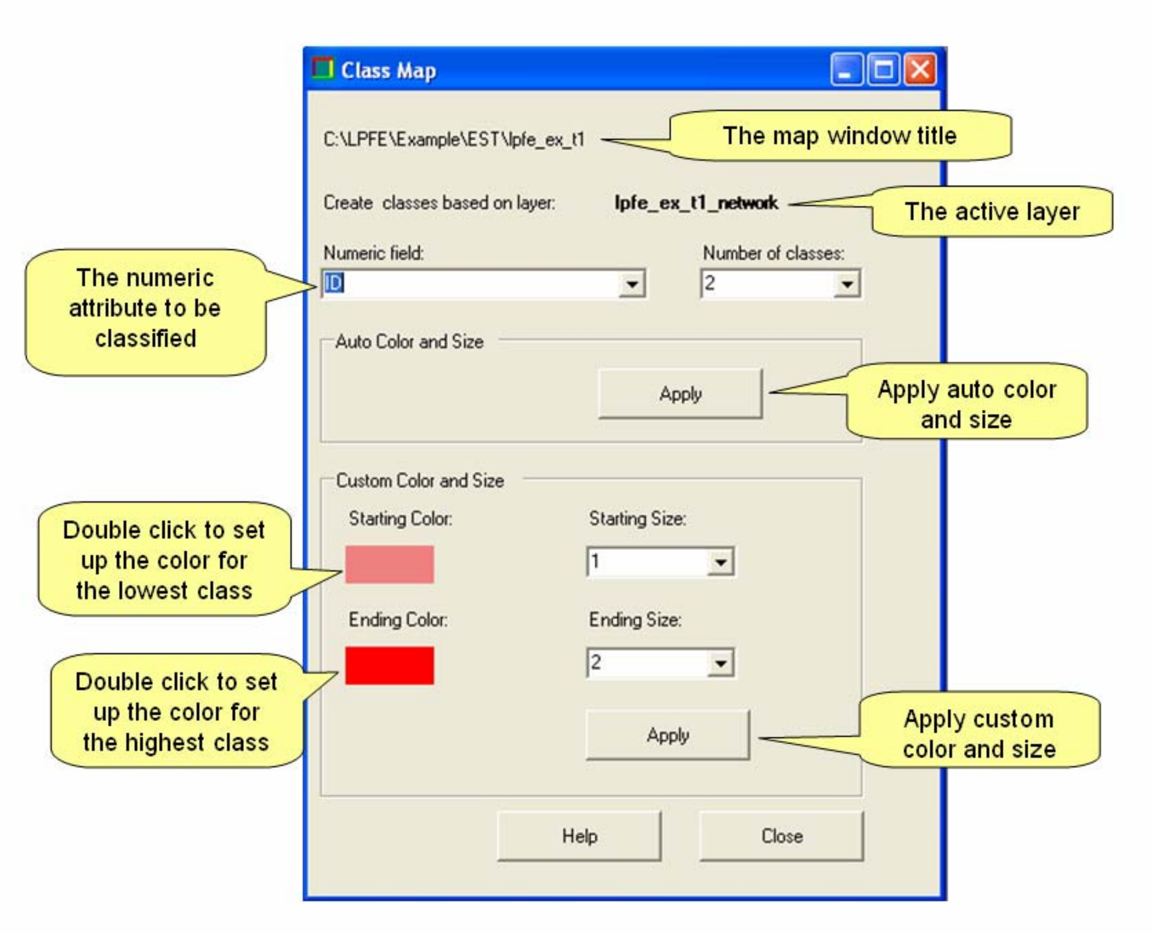

Scatter Plot_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
☑ 9GRD_SCATTERPLOT_MARKER.SH
☑ 9GRD_SCATTERPLOT_AXIS.SHP
☑ 9GRD_SCATTERPLOT
467
350.25
233.5
116.75
Estimated
25% Error
45 Degree
25% Error
116.75
233.5
Observed
350.25
467

Figure 15 Scatter Plot Axis Layer

How to Create a Class Map

6. Make the layer active by clicking at the name of the layer in the legend, then press the tool button Class Map. The Class Map Window will appear (Figure 16).



Class Map

C:\LPFE\Example\EST\lpfe_ex_t1

Create classes based on layer: **lpfe_ex_t1_network**

Numeric field: ID

Number of classes: 2

Starting Size: 1

Ending Size: 2

The map window title

The active layer

The numeric attribute to be classified

Apply auto color and size

Double click to set up the color for the lowest class

Double click to set up the color for the highest class

Apply custom color and size

Figure 16 Class Map Features

7. Once the form appears, select a particular attribute from the numeric fiend and select the number of classes.
8. To let the program determine the colors and sizes of the classes, click at the Apply button in the Auto Color and Size panel (Figure 17).

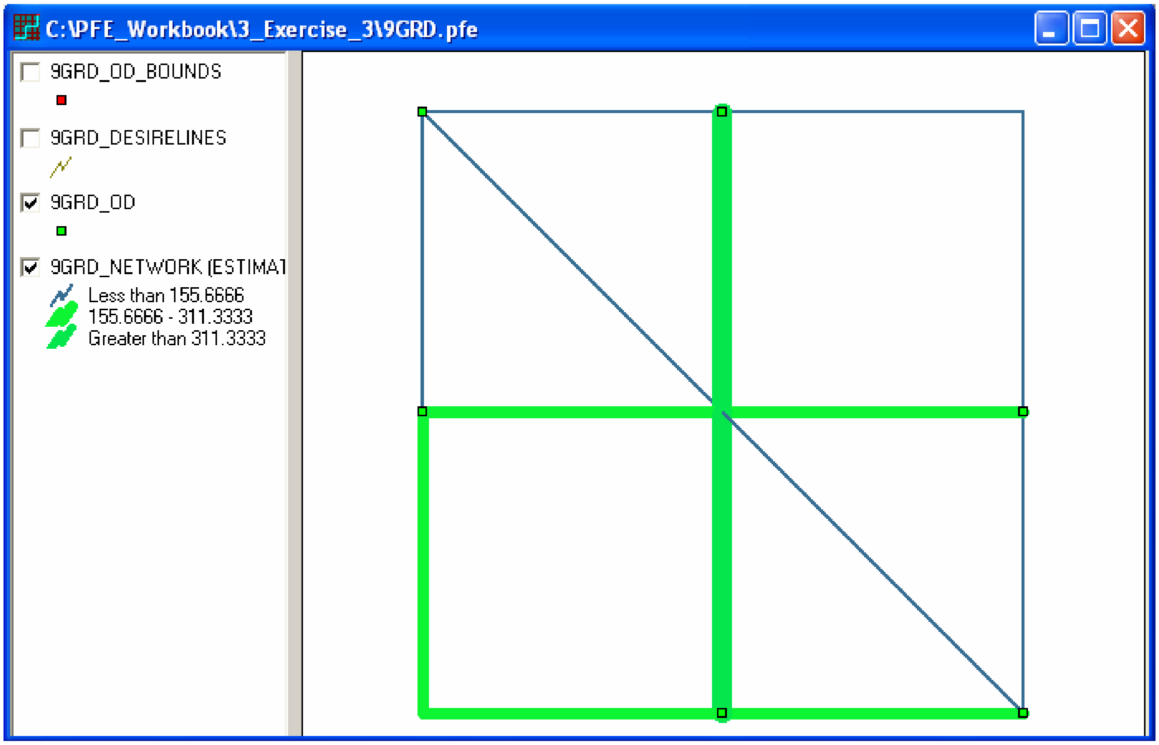


Figure 17 Auto Color and Size Class Map

9. To set up the color and size by applying custom color and size, double click at the color blocks to set up the colors for the lowest and highest classes. Then, select the symbol sizes for the lowest and highest classes. After clicking the Apply (Custom Color and Size) button, the program will create for all classes a graduate color and size transition from the lowest to highest classes (Figure 18).

<u>Remarks:</u>

After a class map is generated, the legend will be reset. To place another posting, you need to remember to set the active layer again.

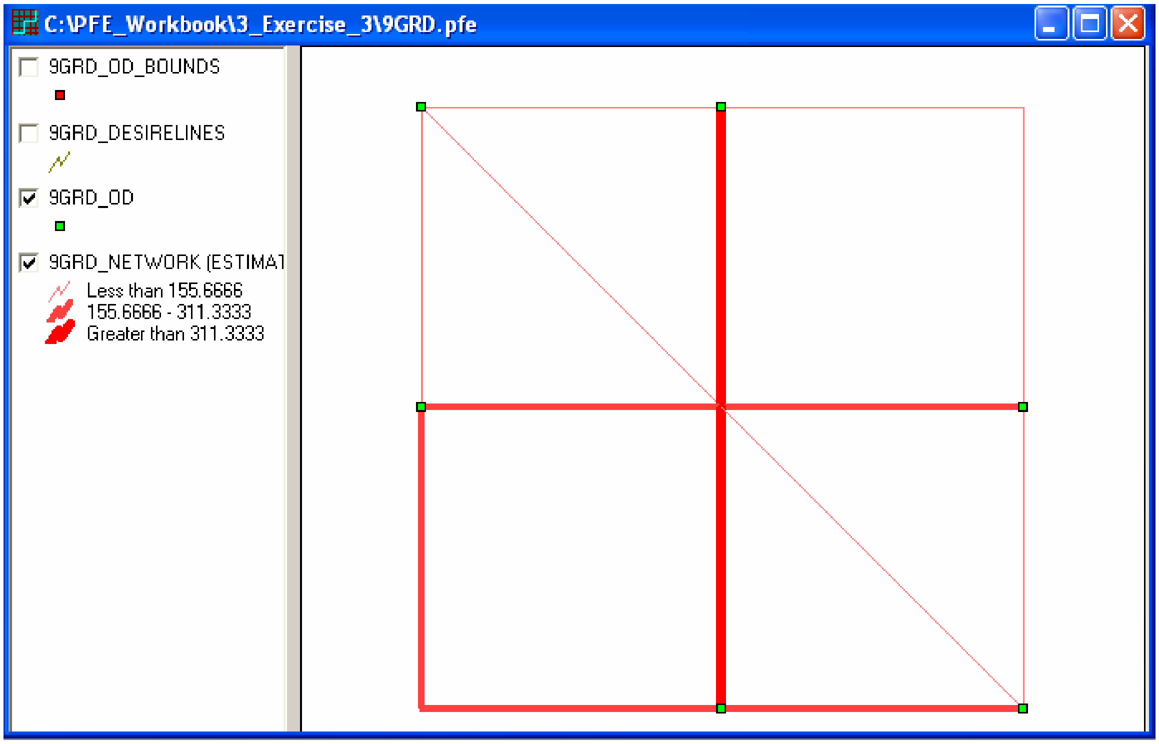


Figure 18 Custom Color and Size Class Map

How to Clear a Class Map

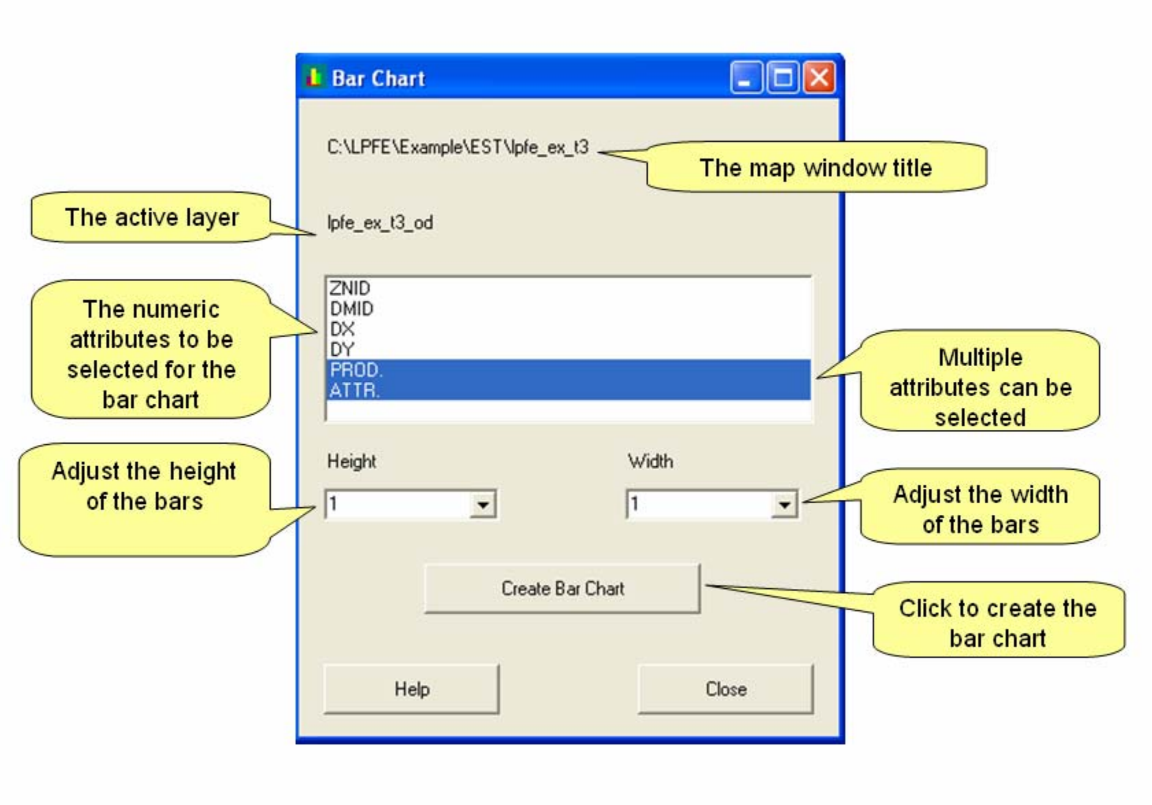
10. To clear the labels, press the Clear Postings button ⊗. The function applies to all three types of postings.

Remarks:

After a class map is cleared, the legend is reset. To place another posting, you need to remember tot set the active layer again.

How to Create a Bar Chart Map

11. Make the layer active by clicking at the name of the layer in the legend, then press the tool button Chart Map. The Bar Chart Window will appear (Figure 19).



C:\LPFE\Example\EST\lpfe_ex_t3

lpfe_ex_t3_od

Height: 1

Width: 1

The map window title

The active layer

The numeric attributes to be selected for the bar chart

Multiple attributes can be selected

Adjust the height of the bars

Adjust the width of the bars

Click to create the bar chart

Figure 19 Bar Chart Feature

12. Once the form appears, clicking at an attribute from the attribute list box to will select and highlight the attribute. Clicking at an attribute twice will de-select the attribute. Multiple attributes can be selected at the same time.
13. Select the height and width from the two pull down boxes.
14. Click at the Create Bar Chart button. The program will create the bar chart accordingly (Figure 20).

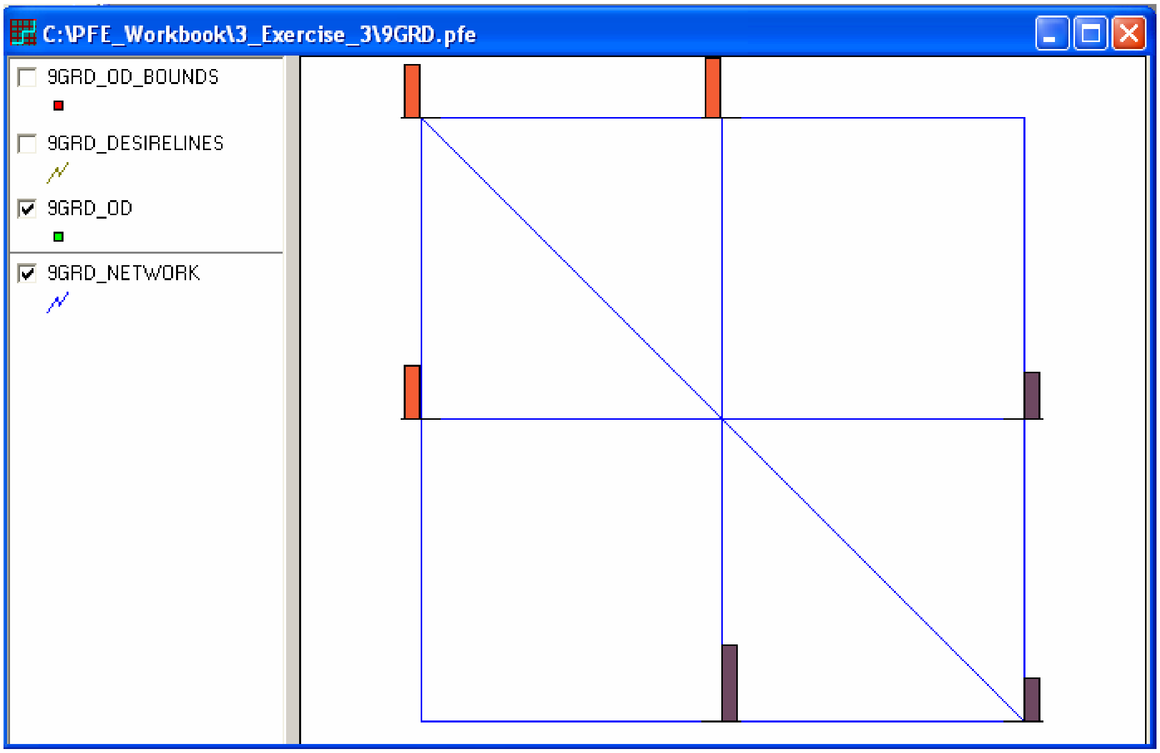


Figure 20 Bar Chart Window

How to Clear a Class Map

15. To clear the labels, press the Clear Postings button ⊗. The function applies to all three types of postings.

<u>Remarks:</u>

After a class map is cleared, the legend is reset. To place another posting, you need to remember tot set the active layer again.

Bar charts are best suited for visualizing the variation of production and attraction at origins and destinations. Note that bar charts are not designed for network links. Applying bar chats to network links will cause the map to be entirely filled with bars and the network links will become invisible (i.e., links are replaced by the bars).

Example 3 Manage OD Tables

After a successful PFE estimation, the OD table will be opened automatically by the program.

How to Open an OD Table

If a previously opened OD table window is now closed and the users wish to see it again, there are two ways to open an OD table:

1. When the network map is the active window, go to the menu item Tables/Open OD Table to open the corresponding OD table.
2. If no network window is opened, go to the menu item Diagnostic/Compare and select from the available OD Tables (Figure 21).

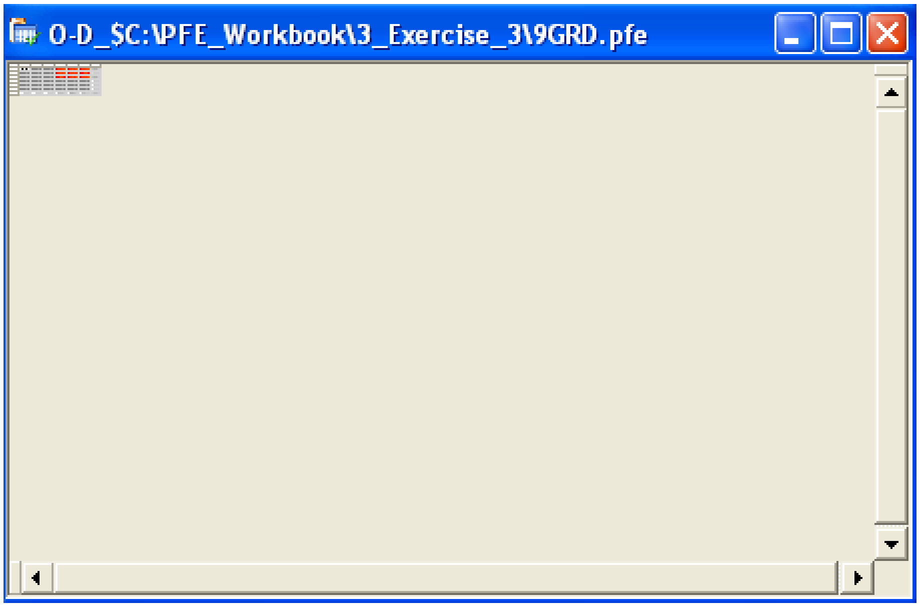


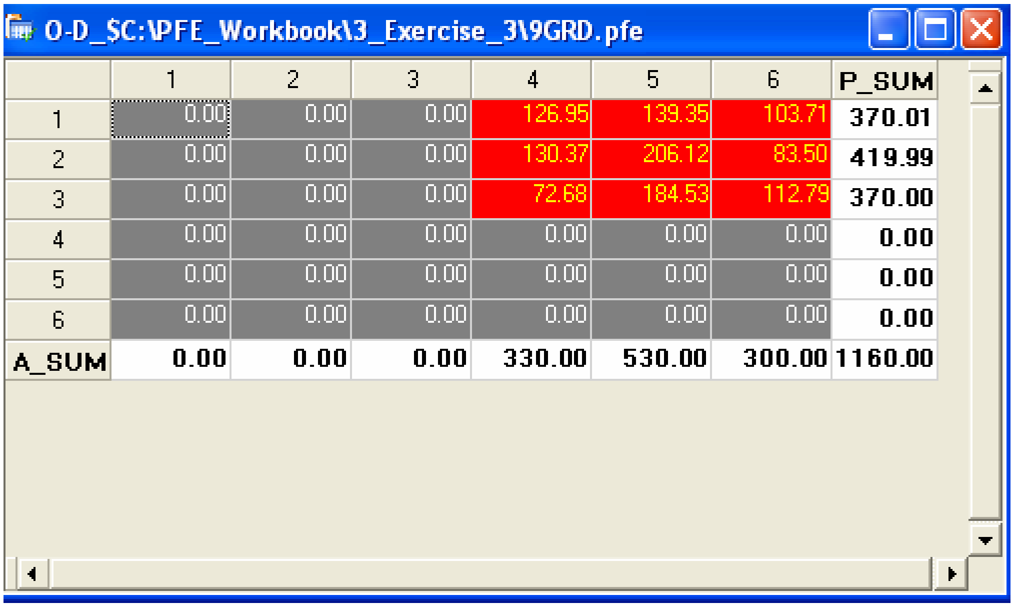
Figure 21 OD Table Window

Note that when the OD table is opened, the view is reduced so the color pattern of the entire table can be viewed without using the scrolling bars.

How to Enlarge the OD Table View

3. Make the OD table the active window.

4. Click at the toolbar button Table Zoom In . The OD Table will return to the normal view (Figure 22).



O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

Figure 22 The Enlarged OD Table

How to Reduce the OD Table View

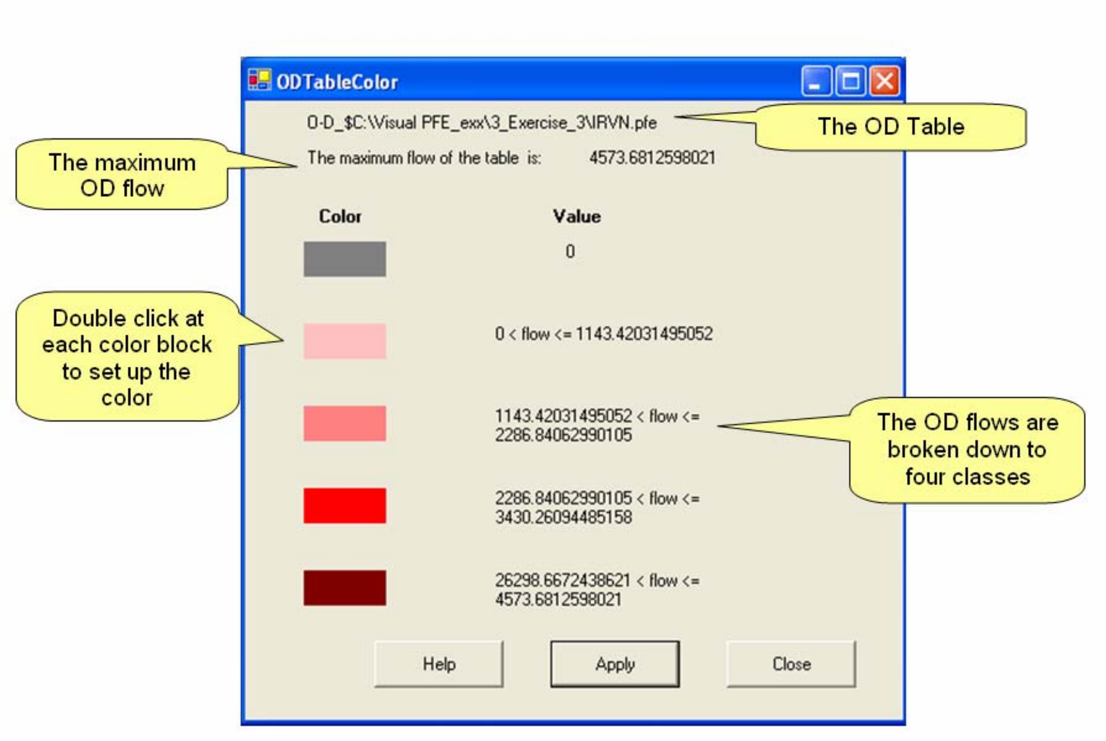
5. To reduce the view of the OD table again, simply press down the toolbar button Table Zoom Out . The OD table view will be reduced again.

The default color scheme for the OD table cells is gray (for cells = 0) and red for cells >0). Users can adjust the colors of the cells using the OD Color Window.

How to Adjust the Colors of the OD Cells

6. Make the OD table the active window

7. Go to the menu item Table/OD Table Color or click at the toolbar button OD Color . The OD Table Color Window will open (Figure 23).


ODTableColor
The OD Table
The maximum OD flow
Double click at each color block to set up the color
The OD flows are broken down to four classes
Help
Apply
Close

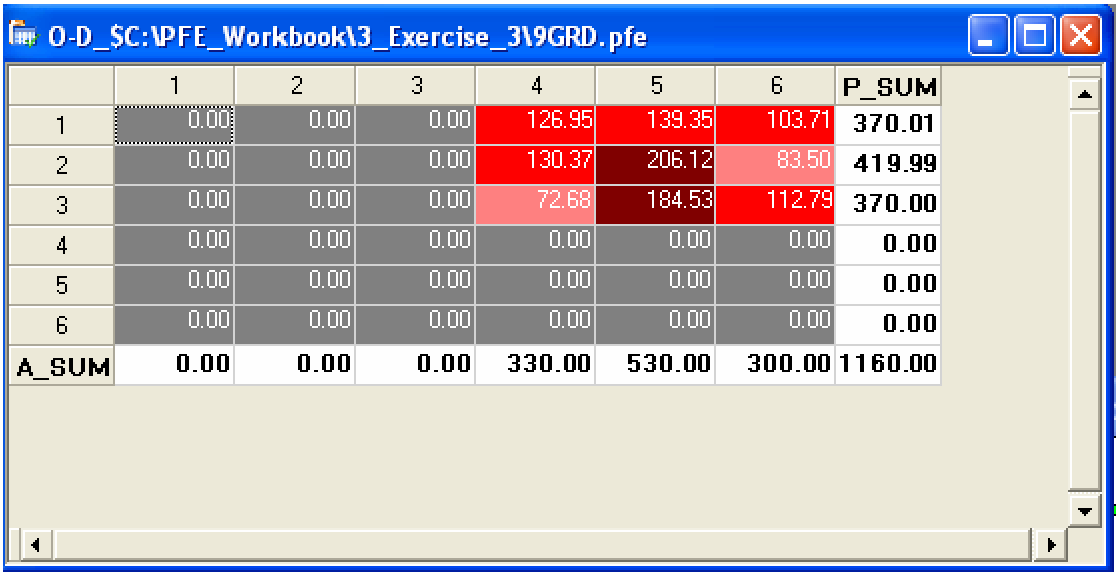
O-D_$C:\Visual PFE_exx\3_Exercise_3\IRVN.pfe

The maximum flow of the table is: 4573.6812598021

Color	Value
■	0
■	0 < flow <= 1143.42031495052
■	1143.42031495052 < flow <= 2286.84062990105
■	2286.84062990105 < flow <= 3430.26094485158
■	26298.6672438621 < flow <= 4573.6812598021

Figure 23 OD Table Color Feature

8. On the OD Color Window, the values of OD flows are broken down into four classes. Double click at each color block to set up the color.
9. Click at the Apply button. The cell colors of the OD table will be adjusted accordingly (Figure 24).



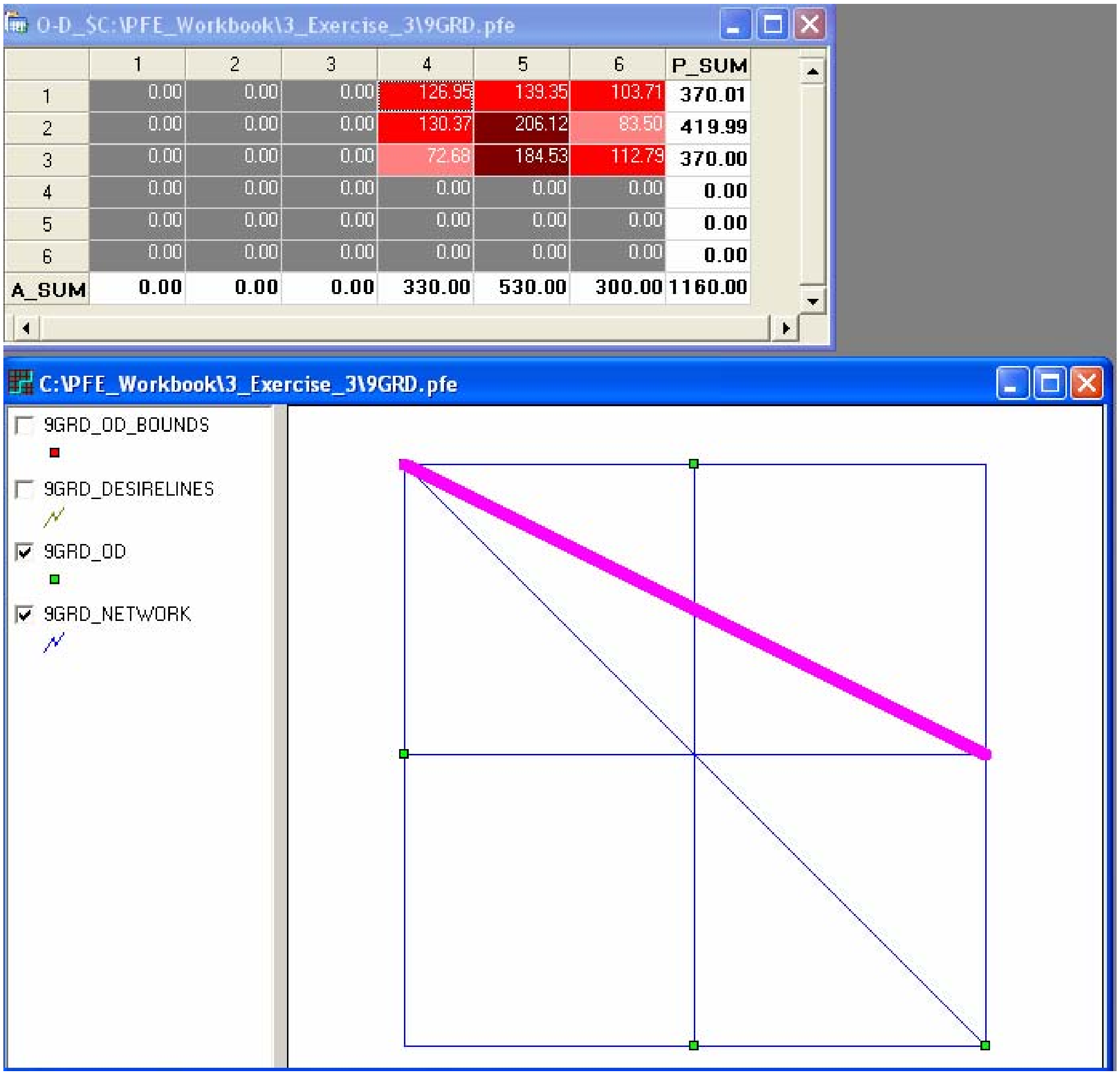
	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

Figure 24 The Adjusted OD Table Color

How to Display Desire Lines via the OD Table

The desire line layer is typically made invisible on the network map, because they tend to obstruct the view of the network links. Visual PFE is built with a unique function for users to view desire lines via the OD Table. To use this function:

10. Make sure both the network Map and the OD Table windows are both opened.
11. Click at a cell in the OD table to make it the active cell (the cell will be encircled with faint dash lines).
12. Press down the Display Desire Line toolbar button . The corresponding desire line(s) will be highlighted in the Map Window. If the active cell of the OD table stays on an internal cell when the button is clicked, the focus will be move to the network map window and the desire line corresponding to the OD pair will be highlighted (Figure 25).



O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

☐ 9GRD_OD_BOUNDS

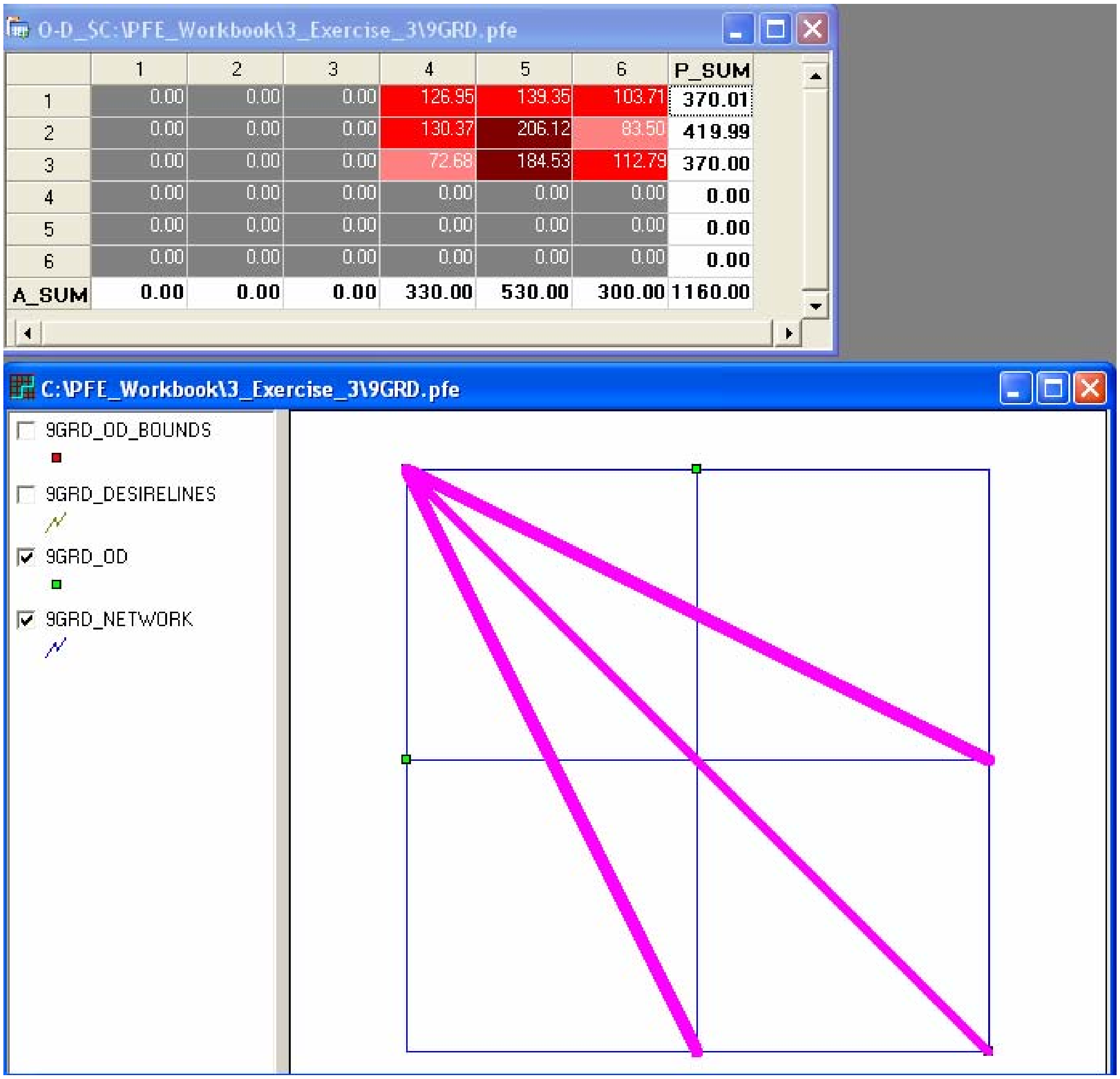
☐ 9GRD_DESIRELINES

☑ 9GRD_OD

☑ 9GRD_NETWORK

Figure 25 Display of Desire Line via OD Table

13. If a cell in the row sum or the column sum is the active cell, one-to-all or all-to-one desire lines will be highlighted (Figure 26).


O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

□ 9GRD_OD_BOUNDS

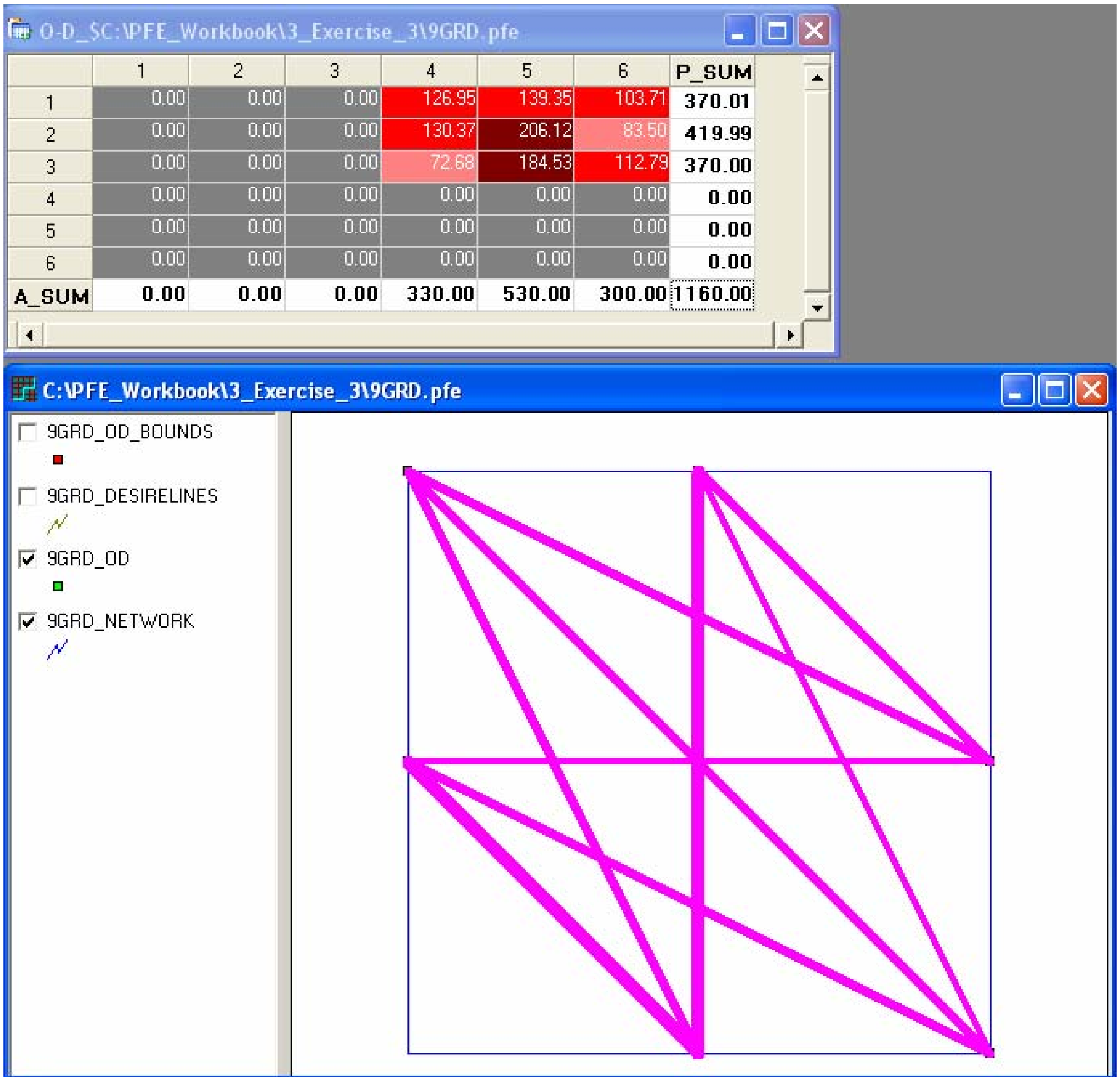
□ 9GRD_DESIRELINES

☑ 9GRD_OD

☑ 9GRD_NETWORK

Figure 26 Desire Lines from Row Sum or Column Sum

14. If the active cell of the OD table stays on the table total, all of the desire lines will be highlighted (Figure 27).



O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

☐ 9GRD_OD_BOUNDS

☐ 9GRD_DESIRELINES

☑ 9GRD_OD

☑ 9GRD_NETWORK

Figure 27 Desire Lines from Table Total

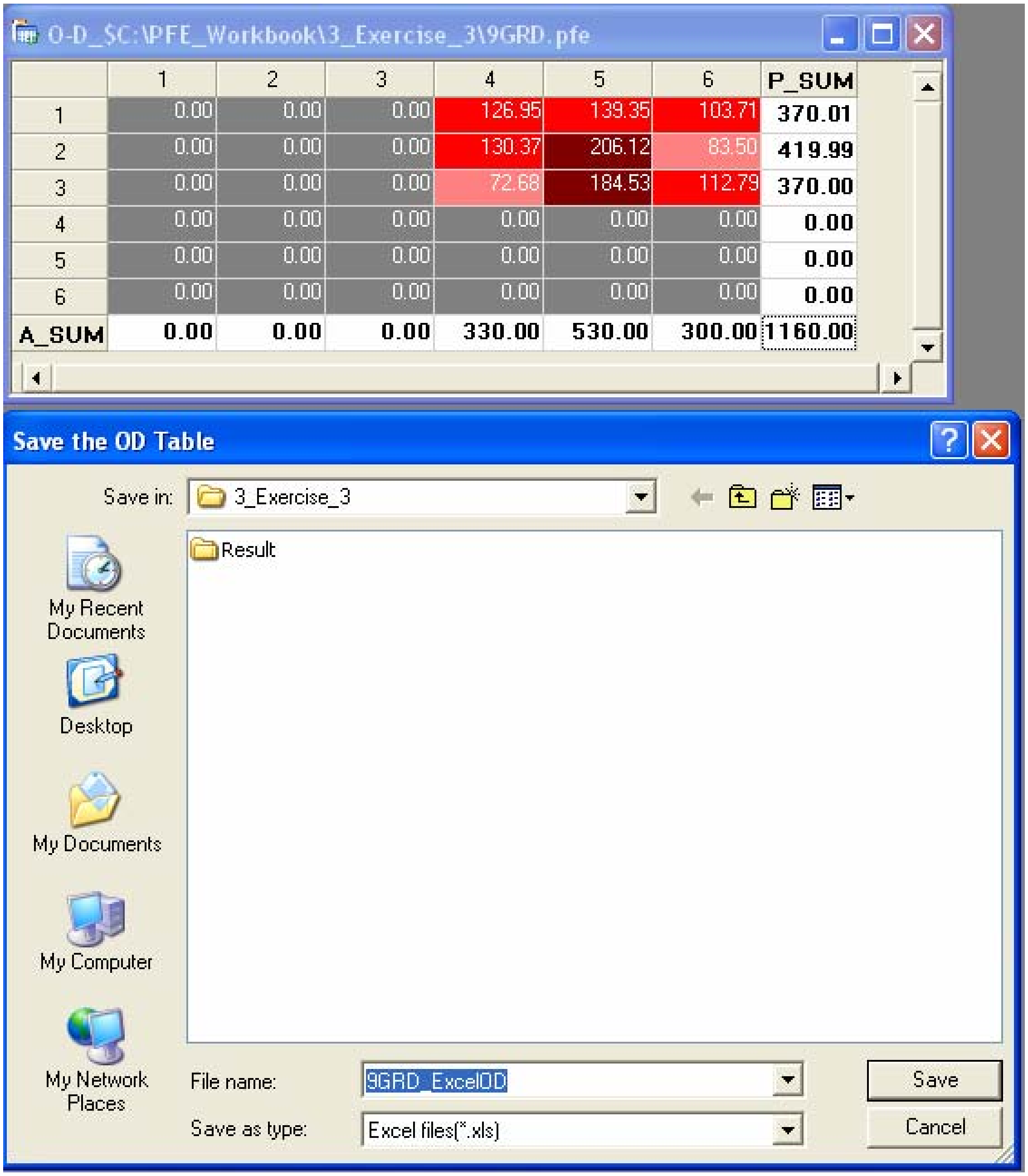
How to Clear the Desire Lines

15. To clear the magenta highlight from the map, when the network map window is the active window, press the Clear Track Lines button . The button also applies to the magenta highlights on links and paths.

The OD Table Window in Visual PFE is managed using the software component Spread. The table itself is a spreadsheet and can be saved as a Microsoft Excel file.

How to Save an OD Table as Excel File

16. Make the OD Table window the active window
17. Go to the menu item File/Save as Excel.
18. The Save File As dialog box will open for users to enter a desired file name for the Excel File (Figure 28).



O-D_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

	1	2	3	4	5	6	P_SUM
1	0.00	0.00	0.00	126.95	139.35	103.71	370.01
2	0.00	0.00	0.00	130.37	206.12	83.50	419.99
3	0.00	0.00	0.00	72.68	184.53	112.79	370.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A_SUM	0.00	0.00	0.00	330.00	530.00	300.00	1160.00

Save the OD Table

Save in: 3_Exercise_3

Result

File name: 9GRD_ExcelOD

Save as type: Excel files(*.xls)

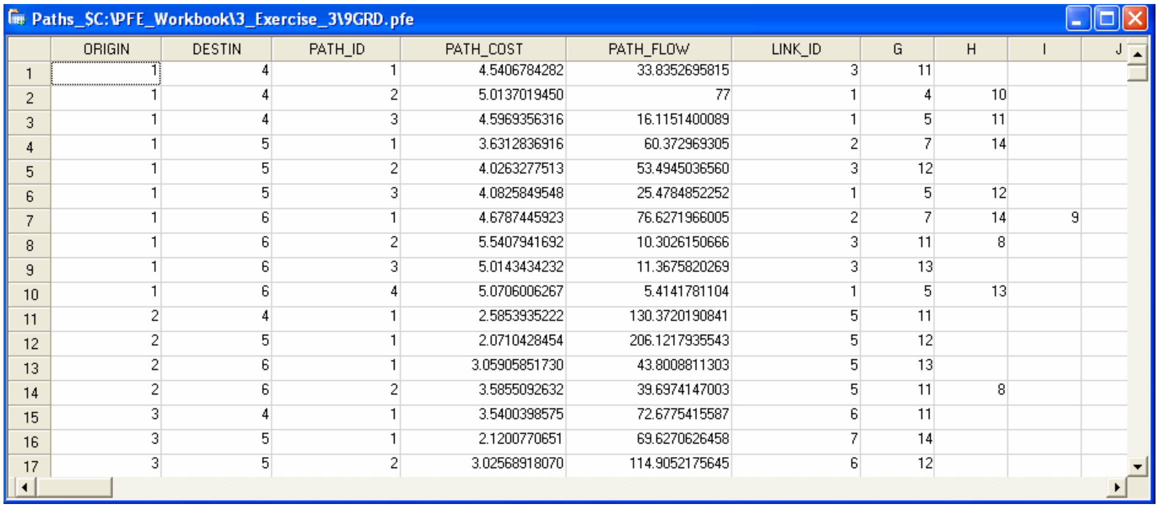
Figure 28 Save OD Table as Excel File

Example 5 Visualize Paths

The estimates of path flows between origins and destinations are also generated by PFE as a text file (i.e., the pthsum.csv file). Similar to the OD tables, the Visual PFE program can convert the path text file and formats it as a spreadsheet.

How to Open a Path Table

1. Make the corresponding network map the active window
2. Go to the menu item Tables/Open Path Table. The path will open accordingly (Figure 29).



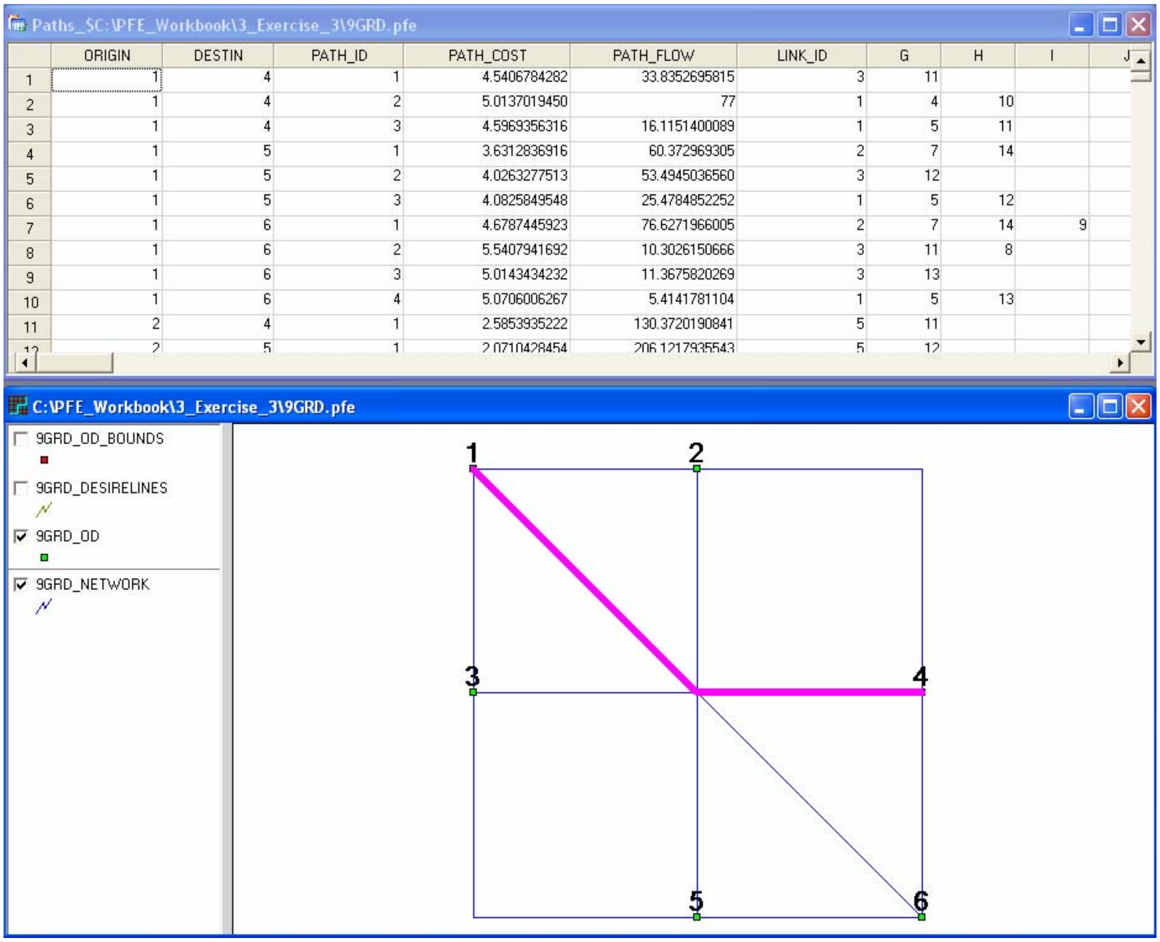
Paths_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

	ORIGIN	DESTIN	PATH_ID	PATH_COST	PATH_FLOW	LINK_ID	G	H	I	J
1	1	4	1	4.5406784282	33.8352695815	3	11			
2	1	4	2	5.0137019450	77	1	4	10		
3	1	4	3	4.5969356316	16.1151400089	1	5	11		
4	1	5	1	3.6312836916	60.372969305	2	7	14		
5	1	5	2	4.0263277513	53.4945036560	3	12			
6	1	5	3	4.0825849548	25.4784852252	1	5	12		
7	1	6	1	4.6787445923	76.6271966005	2	7	14	9	
8	1	6	2	5.5407941692	10.3026150666	3	11	8		
9	1	6	3	5.0143434232	11.3675820269	3	13			
10	1	6	4	5.0706006267	5.4141781104	1	5	13		
11	2	4	1	2.5853935222	130.3720190841	5	11			
12	2	5	1	2.0710428454	206.1217935543	5	12			
13	2	6	1	3.05905851730	43.8008811303	5	13			
14	2	6	2	3.5855092632	39.6974147003	5	11	8		
15	3	4	1	3.5400398575	72.6775415587	6	11			
16	3	5	1	2.1200770651	69.6270626458	7	14			
17	3	5	2	3.02568918070	114.9052175645	6	12			

Figure 29 Path Table

How to Display Paths

3. Make sure that both the network map and the path table windows are opened.
4. Make the path table the active window. Click at a cell in the path table to make it the active cell. There are four options to display paths:
    a. Toolbar Button Single Path : When the button is clicked, the path corresponding to the active row of the Path Table will be highlighted (Figure 30).



Paths_$C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
C:\PFE_Workbook\3_Exercise_3\9GRD.pfe
☐ 9GRD_OD_BOUNDS
☐ 9GRD_DESIRELINES
☑ 9GRD_OD
☑ 9GRD_NETWORK
1
2
3
4
5
6

	ORIGIN	DESTIN	PATH_ID	PATH_COST	PATH_FLOW	LINK_ID	G	H	I	J
1	1	4	1	4.5406784282	33.8352695815	3	11			
2	1	4	2	5.0137019450	77	1	4	10		
3	1	4	3	4.5969356316	16.1151400089	1	5	11		
4	1	5	1	3.6312836916	60.372969305	2	7	14		
5	1	5	2	4.0263277513	53.4945036560	3	12			
6	1	5	3	4.0825849548	25.4784852252	1	5	12		
7	1	6	1	4.6787445923	76.6271966005	2	7	14	9	
8	1	6	2	5.5407941692	10.3026150666	3	11	8		
9	1	6	3	5.0143434232	11.3675820269	3	13			
10	1	6	4	5.0706006267	5.4141781104	1	5	13		
11	2	4	1	2.5853935222	130.3720190841	5	11			
12	2	5	1	2.0710428454	206.1217935543	5	12			

Figure 30 Display of Single Path

b. Toolbar Button One to All Paths : When the button is clicked, all the paths originating from the origin of the active row of the Path Table will be highlighted (Figure 31).

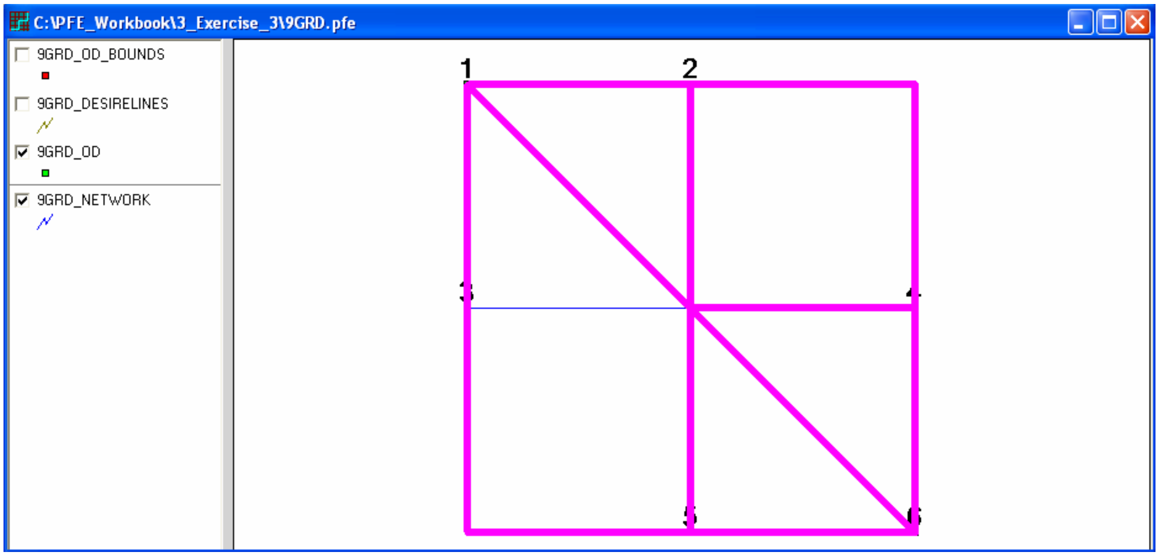


Figure 31Display of All Paths

c. Toolbar Button All to One Paths: When the button is clicked, all the paths ending at the destination of the active row of the Path Table will be highlighted (Figure 32).

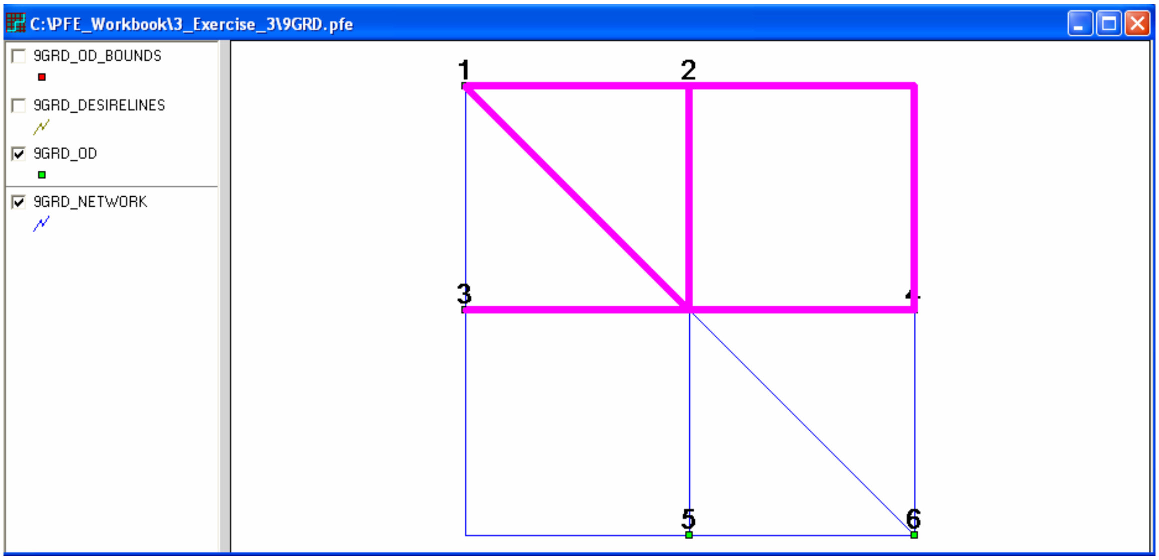


Figure 32 Display of All to One Paths

d. Toolbar Button One to One Paths: When the button is clicked, all the paths beginning at the origin and ending at the destination of the active row of the Path Table will be highlighted (Figure 33).

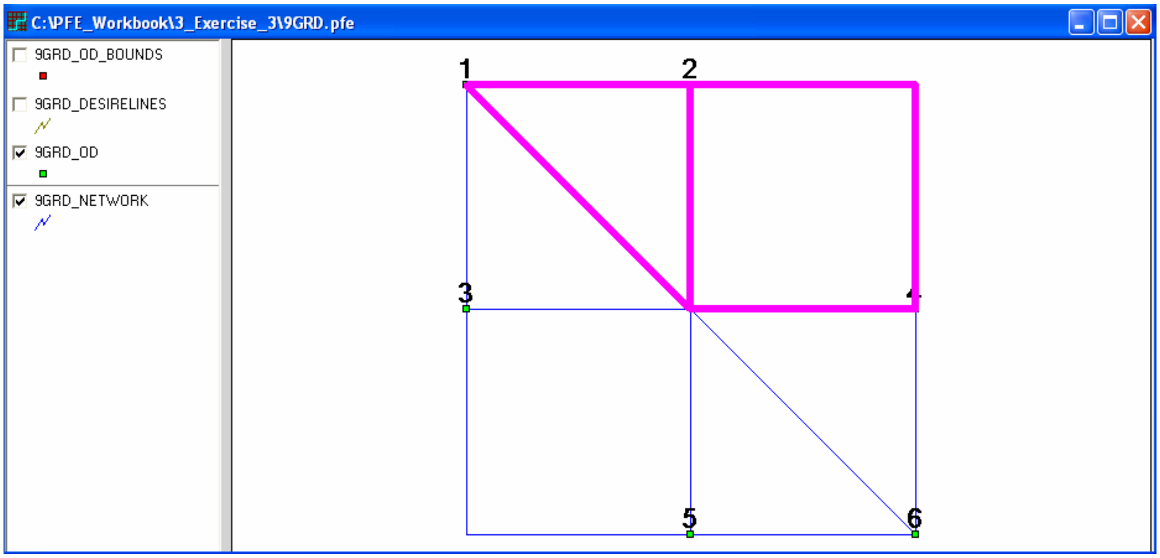


Figure 33 Display of One to One Paths

How to Clear the Paths

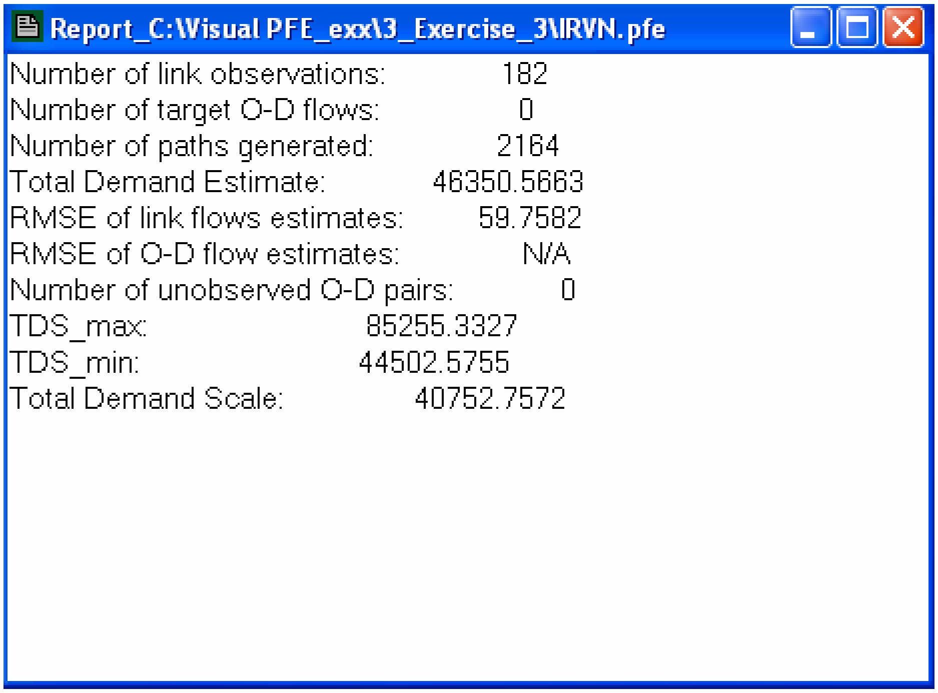
19. To clear the magenta highlight from the map, when the network map window is the active window, press the Clear Track Lines button . The button also applies to the magenta highlights on links and desire lines.

<u>Remarks:</u>

On a large network, the number of paths can be so substantial such that tracking links in all the paths becomes an extremely time consuming operation. The one-to-all and the all-to-one path tracking may actually fail when the number of links making up all the paths exceeds the limitation of the software. When this happens, users will be informed that the limitation is reached and be advised to use the one-to-one path option instead.

Example 6 View Report and Text Files

Every PFE estimation generates a text file, documenting summary statistics of the estimation. The estimation statistics of the estimation can be viewed in Visual PFE using the Report Window (Figure 34).



```
Number of link observations:            182
Number of target O-D flows:               0
Number of paths generated:             2164
Total Demand Estimate:           46350.5663
RMSE of link flows estimates:       59.7582
RMSE of O-D flow estimates:             N/A
Number of unobserved O-D pairs:           0
TDS_max:                     85255.3327
TDS_min:                     44502.5755
Total Demand Scale:              40752.7572
```

Figure 34 Report Window

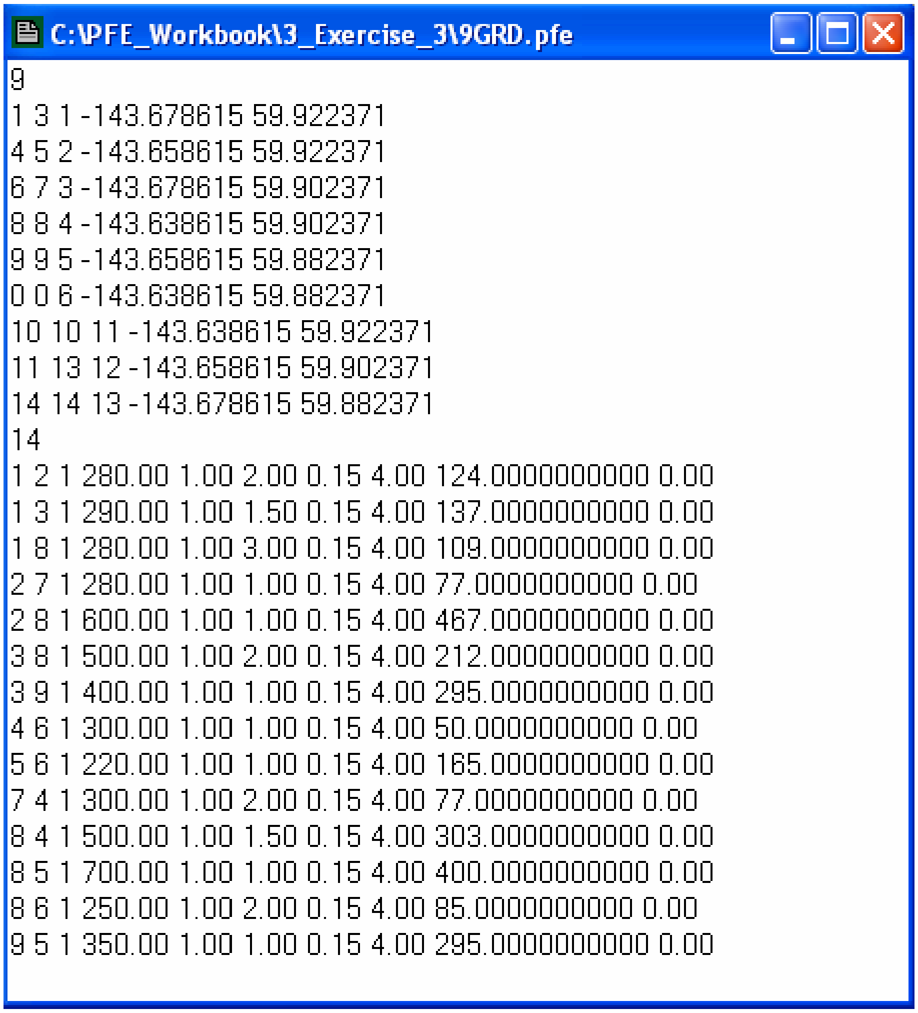
How to Open a Report Window

1. Make the network map the active window.
2. Go to the menu item Diagnostic/Statistics.

A Report Window is actually a rich text box that can serve as a text editor. User can use the Report Window to edit or create PFE input network data.

How to View and Edit Text Files

To edit an existing text file, go to the menu item File/Open and select one of the PFE input text files (e.g., 9GRD.pfe). The file will be opened with a report window (Figure 35).

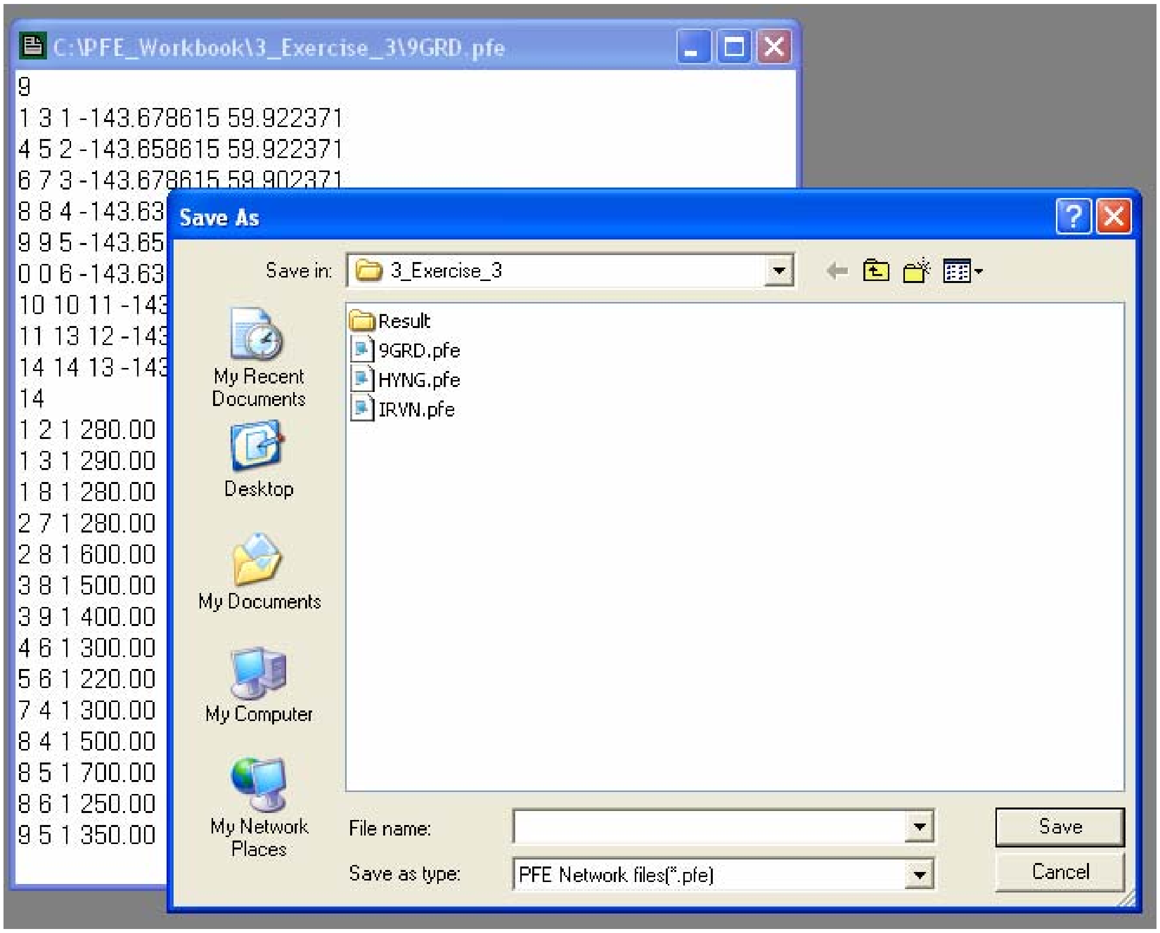


C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

```
9
1 3 1 -143.678615 59.922371
4 5 2 -143.658615 59.922371
6 7 3 -143.678615 59.902371
8 8 4 -143.638615 59.902371
9 9 5 -143.658615 59.882371
0 0 6 -143.638615 59.882371
10 10 11 -143.638615 59.922371
11 13 12 -143.658615 59.902371
14 14 13 -143.678615 59.882371
14
1 2 1 280.00 1.00 2.00 0.15 4.00 124.0000000000 0.00
1 3 1 290.00 1.00 1.50 0.15 4.00 137.0000000000 0.00
1 8 1 280.00 1.00 3.00 0.15 4.00 109.0000000000 0.00
2 7 1 280.00 1.00 1.00 0.15 4.00 77.0000000000 0.00
2 8 1 600.00 1.00 1.00 0.15 4.00 467.0000000000 0.00
3 8 1 500.00 1.00 2.00 0.15 4.00 212.0000000000 0.00
3 9 1 400.00 1.00 1.00 0.15 4.00 295.0000000000 0.00
4 6 1 300.00 1.00 1.00 0.15 4.00 50.0000000000 0.00
5 6 1 220.00 1.00 1.00 0.15 4.00 165.0000000000 0.00
7 4 1 300.00 1.00 2.00 0.15 4.00 77.0000000000 0.00
8 4 1 500.00 1.00 1.50 0.15 4.00 303.0000000000 0.00
8 5 1 700.00 1.00 1.00 0.15 4.00 400.0000000000 0.00
8 6 1 250.00 1.00 2.00 0.15 4.00 85.0000000000 0.00
9 5 1 350.00 1.00 1.00 0.15 4.00 295.0000000000 0.00
```

Figure 35 File/Open Text

3. Go to the menu item Edit and try all of the common text editing functions: Undo, Cut, Copy, Paste, and Select All.
4. When you are done with editing, go to the menu item File/Save Text (Figure 36).



C:\PFE_Workbook\3_Exercise_3\9GRD.pfe

```
9
1 3 1 -143.678615 59.922371
4 5 2 -143.658615 59.922371
6 7 3 -143.678615 59.902371
8 8 4 -143.63
9 9 5 -143.65
0 0 6 -143.63
10 10 11 -143
11 13 12 -143
14 14 13 -143
14
1 2 1 280.00
1 3 1 290.00
1 8 1 280.00
2 7 1 280.00
2 8 1 600.00
3 8 1 500.00
3 9 1 400.00
4 6 1 300.00
5 6 1 220.00
7 4 1 300.00
8 4 1 500.00
8 5 1 700.00
8 6 1 250.00
9 5 1 350.00
```

Save As

Save in: 3_Exercise_3

File name:

Save as type: PFE Network files(*.pfe)

Figure 36 File/Save Text

Remarks:

An input PFE network consists of three files (*.pfe, *.pfp, and *.pfx). Editing the network data as text files requires a thorough understanding of the data structure of all three files. The next example shows how to edit attributes of a network using the GUIs.

For more details on editing and creating PFE input networks, please see the *PFE Workbook*.

How to Create New Text Files

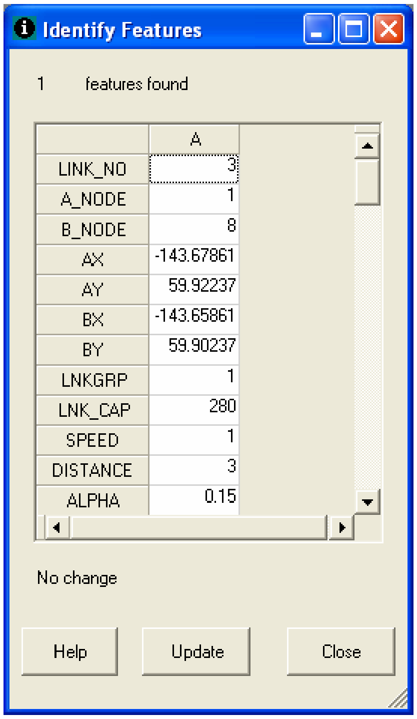
5. To create a new file, go to the menu item File/New and a blank Report Window will be opened as a text editor.

Example 7 Edit Networks

After examining the estimation results, users can change attributes (e.g., the observed flows) of the network and export the data back to the original PFE data format for another estimation with improved results.

How to Edit Network Attributes

1. Make the network map the active window
2. Make the “Network” layer the active layer
3. Click at the toolbar button Identify ℹ and move the mouse cursor over the map.
4. Select the link on which the attributes are to be changed



Identify Features

1 features found

	A
LINK_NO	3
A_NODE	1
B_NODE	8
AX	-143.67861
AY	59.92237
BX	-143.65861
BY	59.90237
LNKGRP	1
LNK_CAP	280
SPEED	1
DISTANCE	3
ALPHA	0.15

No change

Figure 37 Edit Network Attributes

5. Double click at a cell on the form to highlight the cell (Figure 37). Change the value of the highlighted cell.
6. Click at the “Update” button to register the value change.

7. After the Close button is clicked, users will be prompted for final confirmation of the change. Answer “Yes” will save the change to the database (i.e., the shapefile *.dbf will be changed). Otherwise, the change is discarded.
8. To export the network data (*.dbf) to a different set of PFE input files (*.pfe, *.pfp, and *.pfx), go to menu item Network/Export.
9. Save the edited network in a different name (e.g., 9GRDx.pfe).

<u>Remarks:</u>

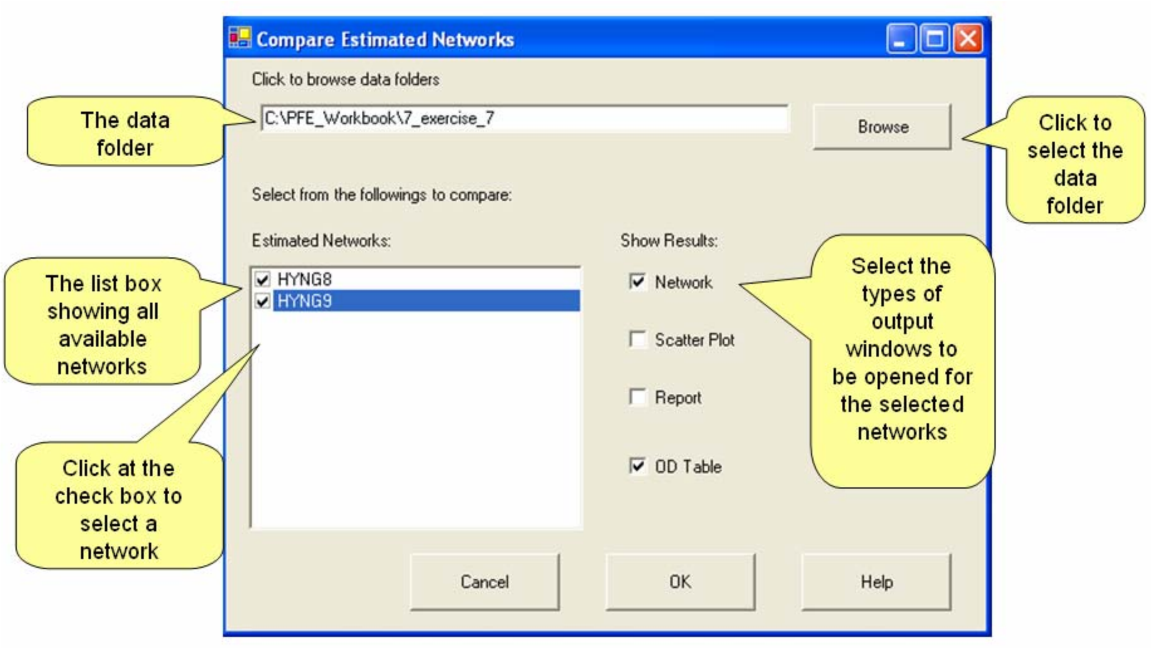
Open a File Explorer window and observe for yourself if all three PFE files are created in the data folder specified. You can also go to File/Open and open the newly exported *.pfe file and see if the change is correctly written to the file.

Example 8 Compare Scenarios

If multiple estimation runs have been done for a network, all of the previous results can be opened and compared at the same time. To compare different results:

How to Compare Previously Estimated Results

1. Make sure all of the previously opened output windows are now closed (to avoid file sharing violation).
2. Go to the menu item Diagnose/Compare. The Compare Estimated Networks Window will open (Figure 38).



Compare Estimated Networks

Click to browse data folders: C:\PFE_Workbook\7_exercise_7

Select from the followings to compare:

Estimated Networks:

- ☑ HYNG8
- ☑ HYNG9

Show Results:

- ☑ Network
- ☐ Scatter Plot
- ☐ Report
- ☑ OD Table

The data folder

Click to select the data folder

The list box showing all available networks

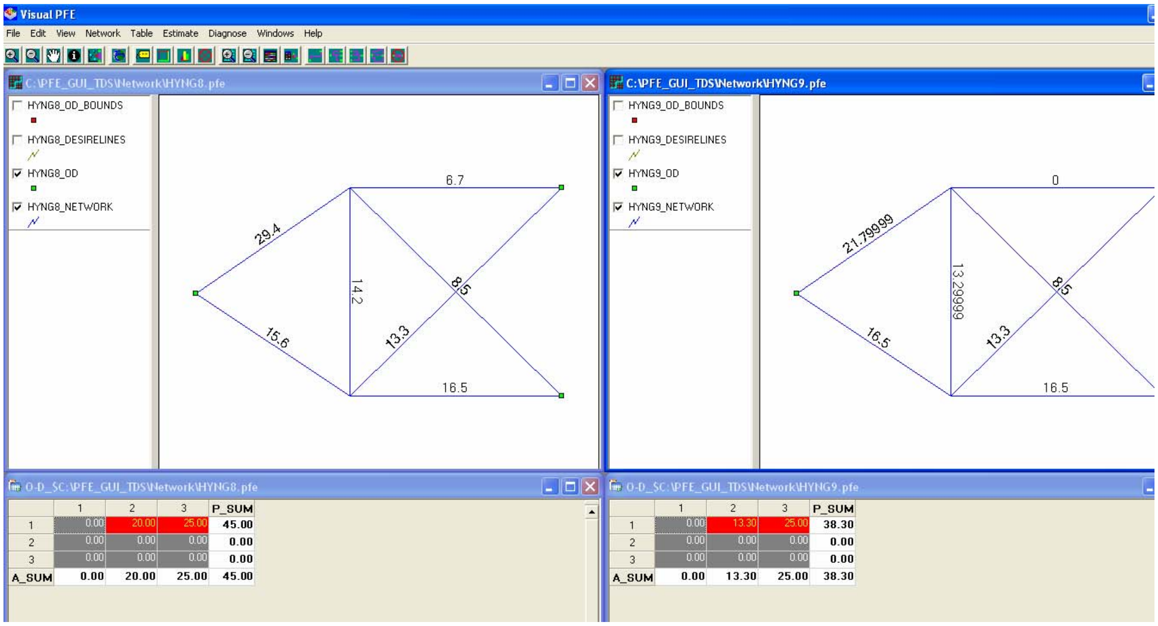
Select the types of output windows to be opened for the selected networks

Click at the check box to select a network

Figure 38 Compare Estimated Networks

3. On the window, click at the Browse button and go to the folder (e.g., the folder /PFE_Wookbook/7_exercise_7) where all estimation results are stored. Once the data folder is selected, all available results will be shown in the corresponding list boxes.
4. Select from the lists and check the types of results to be compared.

5. Click at the OK button. All the selected result items will be open in corresponding windows (Figure 39).



O-D_$C:\PFE_GUI_TDS\Network\HYNG8.pfe

	1	2	3	P_SUM
1	0.00	20.00	25.00	45.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
A_SUM	0.00	20.00	25.00	45.00

O-D_$C:\PFE_GUI_TDS\Network\HYNG9.pfe

	1	2	3	P_SUM
1	0.00	13.30	25.00	38.30
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
A_SUM	0.00	13.30	25.00	38.30

Figure 39 Compared Result

Example Create PFE Networks

Creating network input data for PFE requires an understanding of the data format. A detailed description of the data specification can be found in the User Manual of Visual PFE. Technical notes about the data files are provided at the end of this example for your reference. Visual PFE includes tools for users to build the input data from dBase files. The dBase data format (*.dbf) can be prepared through any software with database capability (e.g., Microsoft® Excel).

What Data are Required to Create PFE Networks?

Visual PFE network creating tool requires the following three data files to create PFE network.

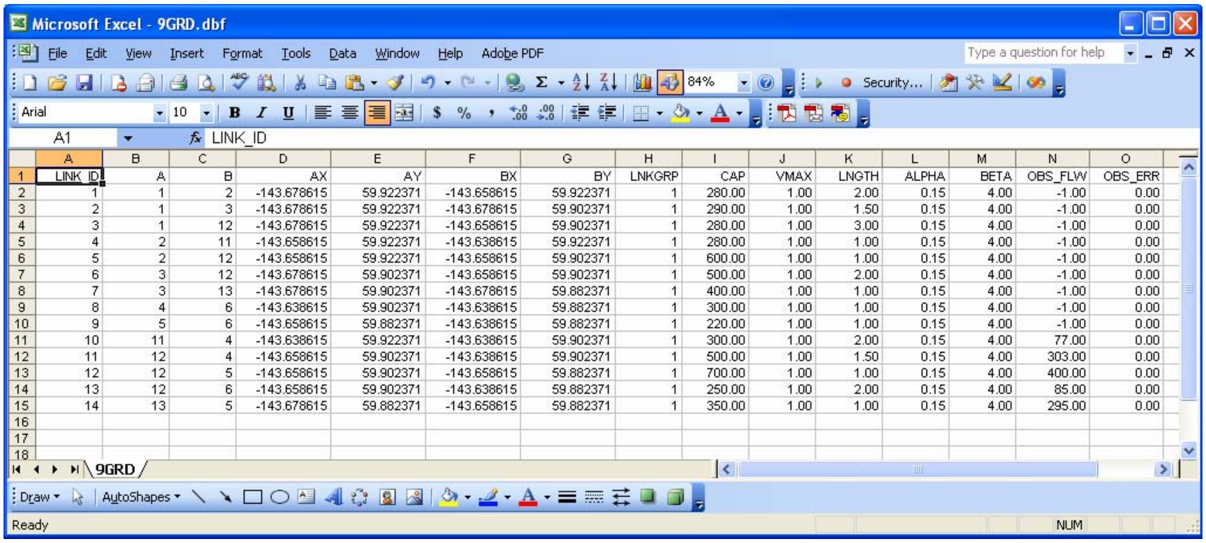
- Network topology file
- Trip table structure file (optional)

- Path-specific file (optional)

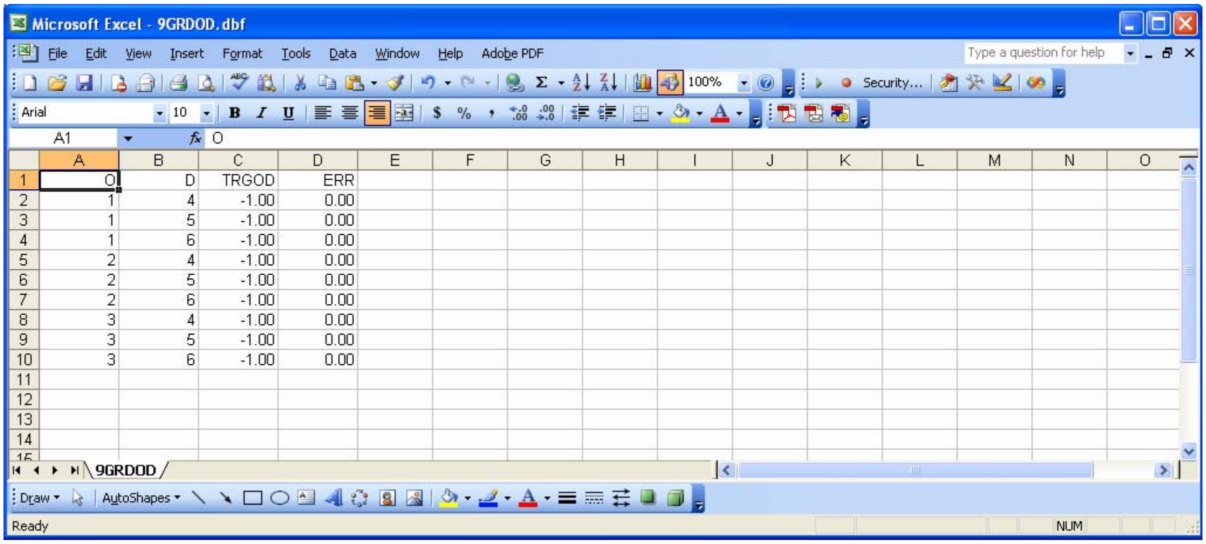
The three input files must be in the dBase data format (*.dbf). The first input file is mandatory while the other two files are optional. If the structure of trip table (i.e., the second file) is not provided, it will be determined through topology of the network. That is, all feasible (connected) pairs of origin and destination will be included. Similarly, if there is no specific path to be included, the path list in the third PFE network file (*.pfp) will be empty.

How to Prepare the Input dBase Data

1. Use a spreadsheet (Excel) to open the network topology file (9GRD.dbf), the trip table structure table (9GRDOD.dbf), and the user-specified paths (9GRD_Path.dbf) files in the folder …\PFE_workbook\2_Exercise_2\. Examine the structure of the input data tables (Figure 40).


Microsoft Excel - 9GRD.dbf
File Edit View Insert Format Tools Data Window Help Adobe PDF
Type a question for help
84%
Security...
Arial
10
A1
LINK_ID
9GRD
Draw
AutoShapes
Ready
NUM

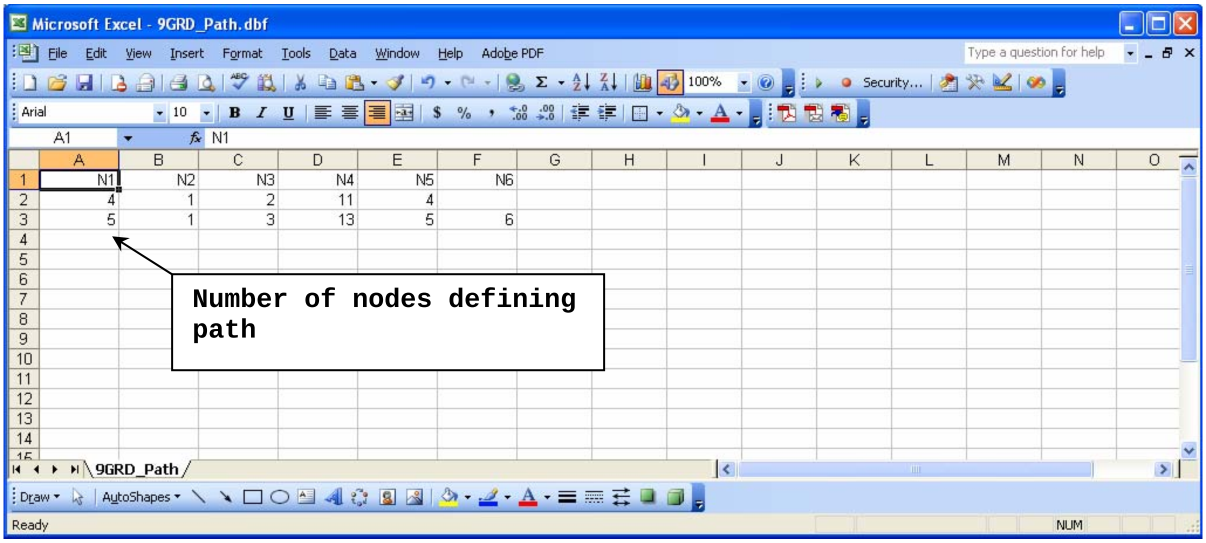
LINK_ID	A	B	AX	AY	BX	BY	LNKGRP	CAP	VMAX	LNGTH	ALPHA	BETA	OBS_FLW	OBS_ERR
1	1	2	-143.678615	59.922371	-143.658615	59.922371	1	280.00	1.00	2.00	0.15	4.00	-1.00	0.00
2	1	3	-143.678615	59.922371	-143.678615	59.902371	1	290.00	1.00	1.50	0.15	4.00	-1.00	0.00
3	1	12	-143.678615	59.922371	-143.658615	59.902371	1	280.00	1.00	3.00	0.15	4.00	-1.00	0.00
4	2	11	-143.658615	59.922371	-143.638615	59.922371	1	280.00	1.00	1.00	0.15	4.00	-1.00	0.00
5	2	12	-143.658615	59.922371	-143.658615	59.902371	1	600.00	1.00	1.00	0.15	4.00	-1.00	0.00
6	3	12	-143.678615	59.902371	-143.658615	59.902371	1	500.00	1.00	2.00	0.15	4.00	-1.00	0.00
7	3	13	-143.678615	59.902371	-143.678615	59.882371	1	400.00	1.00	1.00	0.15	4.00	-1.00	0.00
8	4	6	-143.638615	59.902371	-143.638615	59.882371	1	300.00	1.00	1.00	0.15	4.00	-1.00	0.00
9	5	6	-143.658615	59.882371	-143.638615	59.882371	1	220.00	1.00	1.00	0.15	4.00	-1.00	0.00
10	11	4	-143.638615	59.922371	-143.638615	59.902371	1	300.00	1.00	2.00	0.15	4.00	77.00	0.00
11	12	4	-143.658615	59.902371	-143.638615	59.902371	1	500.00	1.00	1.50	0.15	4.00	303.00	0.00
12	12	5	-143.658615	59.902371	-143.658615	59.882371	1	700.00	1.00	1.00	0.15	4.00	400.00	0.00
13	12	6	-143.658615	59.902371	-143.638615	59.882371	1	250.00	1.00	2.00	0.15	4.00	85.00	0.00
14	13	5	-143.678615	59.882371	-143.658615	59.882371	1	350.00	1.00	1.00	0.15	4.00	295.00	0.00


Microsoft Excel - 9GRDOD.dbf
File Edit View Insert Format Tools Data Window Help Adobe PDF
A1
O
9GRDOD
Ready
NUM

O	D	TRGOD	ERR
1	4	-1.00	0.00
1	5	-1.00	0.00
1	6	-1.00	0.00
2	4	-1.00	0.00
2	5	-1.00	0.00
2	6	-1.00	0.00
3	4	-1.00	0.00
3	5	-1.00	0.00
3	6	-1.00	0.00

Figure 40 Input dBase Data

Remark: Here, it is assumed that the target O-D volumes are unknown, so put –1.00 (minus one) as the target value. Only the structure of O-D trip table (to be estimated) as shown in Table 1 is known.


Microsoft Excel - 9GRD_Path.dbf
File Edit View Insert Format Tools Data Window Help Adobe PDF
A1
N1
9GRD_Path
Ready
NUM

N1	N2	N3	N4	N5	N6
4	1	2	11	4	
5	1	3	13	5	6

Number of nodes defining path

Figure 41 Input Path Data

Remark: Here, it is assumed that there are two paths that the user may particularly want to include into the estimation. The first path traverses nodes 1, 2, 11, 4 while the second path traverses nodes 1, 3, 13, 5, 6 (see Figure 41 and Figure 42).

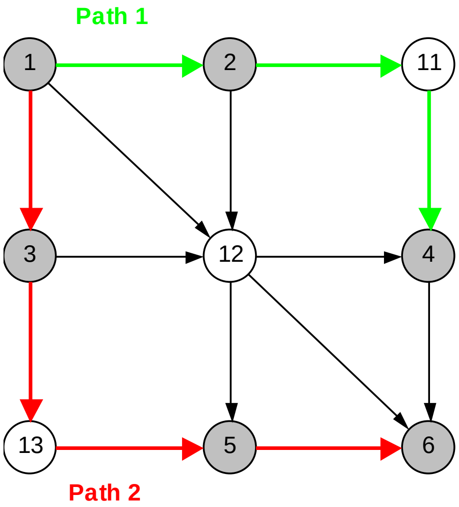
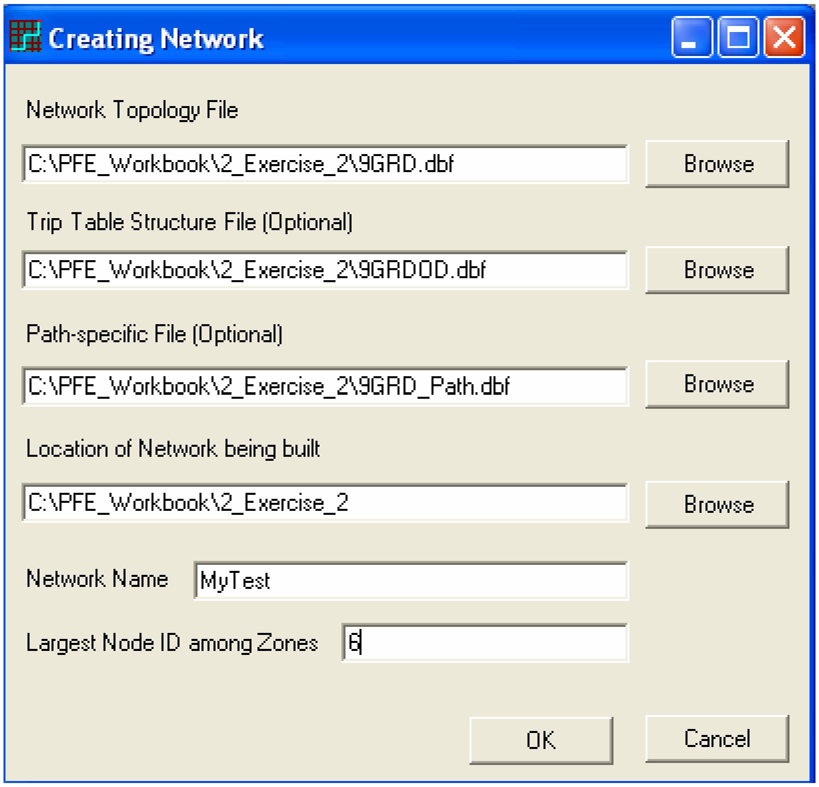


Figure 42 Paths in 9GRD Network

How to Create a PFE Network

2. Close the files and the spreadsheet once you are done reviewing them. Open Visual PFE, go to the menu item Network/Create. (see Figure 43) Enter the followings:
   - Network Topology File:"…\PFE_workbook\2_Exercise_2\9GRD.dbf"
   - Trip Table Structure File: "…\PFE_workbook\2_Exercise_2\9GRDOD.dbf"
   - User-specified Path File: "…\PFE_workbook\2_Exercise_2\9GRD_Path.dbf"
   - The largest node ID of TAZs: "6"
   - Name of network: "9GRD"
   - Location of network being created: "…\PFE_workbook\2_Exercise_2"



Network Topology File

C:\PFE_Workbook\2_Exercise_2\9GRD.dbf

Trip Table Structure File (Optional)

C:\PFE_Workbook\2_Exercise_2\9GRDOD.dbf

Path-specific File (Optional)

C:\PFE_Workbook\2_Exercise_2\9GRD_Path.dbf

Location of Network being built

C:\PFE_Workbook\2_Exercise_2

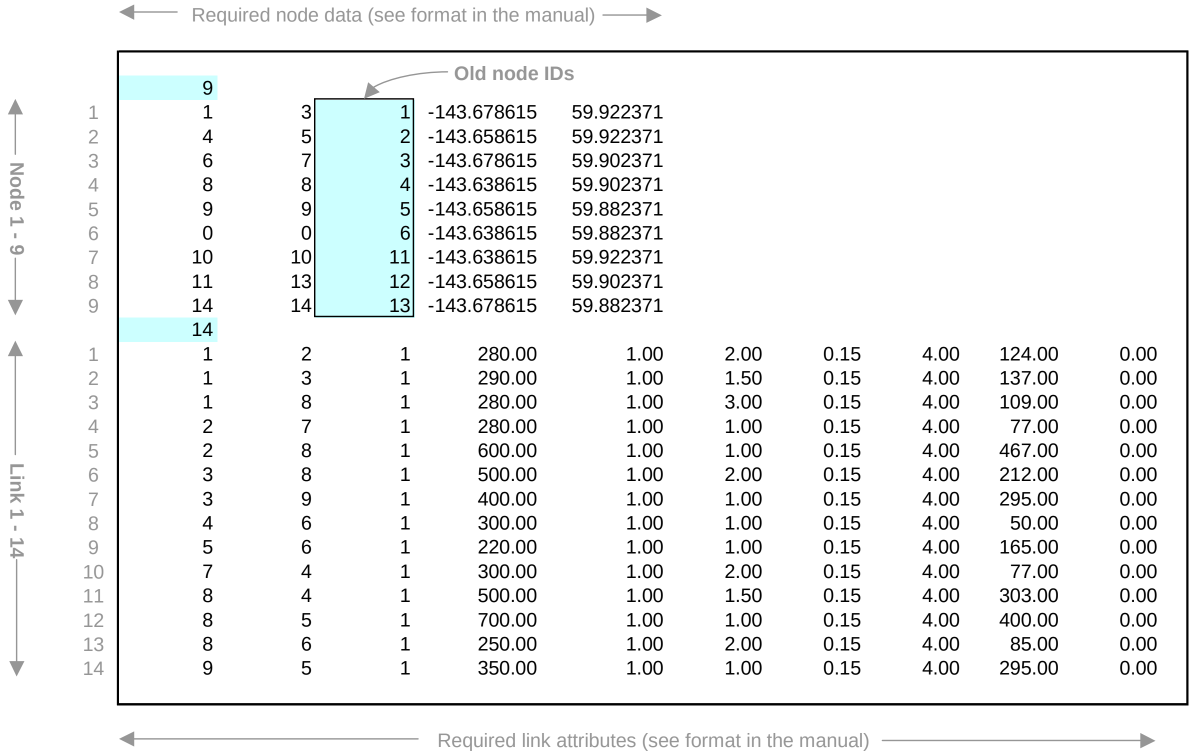
Network Name: MyTest

Largest Node ID among Zones: 6

Figure 43 Creat Network Feature

3. In Visual PFE, go to menu item File/Open and open the *.pfe, *.pfp, and *.pfx files you just created.

The *.pfe File


Required node data (see format in the manual)
Old node IDs
Node 1 - 9
Link 1 - 14
Required link attributes (see format in the manual)

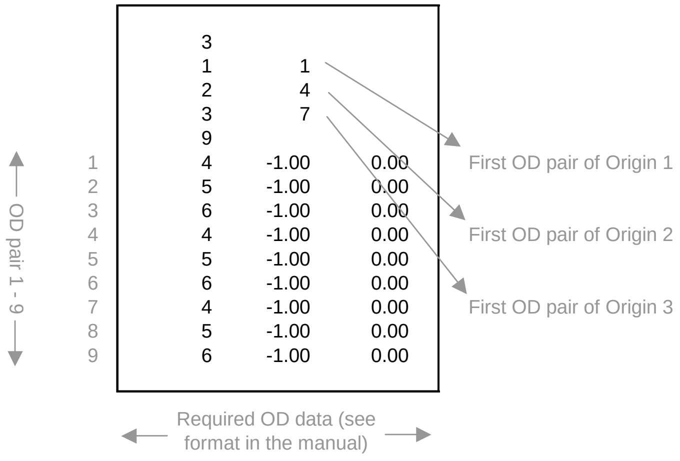
```
         9
1        1         3         1  -143.678615   59.922371
2        4         5         2  -143.658615   59.922371
3        6         7         3  -143.678615   59.902371
4        8         8         4  -143.638615   59.902371
5        9         9         5  -143.658615   59.882371
6        0         0         6  -143.638615   59.882371
7       10        10        11  -143.638615   59.922371
8       11        13        12  -143.658615   59.902371
9       14        14        13  -143.678615   59.882371
        14
1        1         2         1       280.00        1.00      2.00      0.15      4.00    124.00      0.00
2        1         3         1       290.00        1.00      1.50      0.15      4.00    137.00      0.00
3        1         8         1       280.00        1.00      3.00      0.15      4.00    109.00      0.00
4        2         7         1       280.00        1.00      1.00      0.15      4.00     77.00      0.00
5        2         8         1       600.00        1.00      1.00      0.15      4.00    467.00      0.00
6        3         8         1       500.00        1.00      2.00      0.15      4.00    212.00      0.00
7        3         9         1       400.00        1.00      1.00      0.15      4.00    295.00      0.00
8        4         6         1       300.00        1.00      1.00      0.15      4.00     50.00      0.00
9        5         6         1       220.00        1.00      1.00      0.15      4.00    165.00      0.00
10       7         4         1       300.00        1.00      2.00      0.15      4.00     77.00      0.00
11       8         4         1       500.00        1.00      1.50      0.15      4.00    303.00      0.00
12       8         5         1       700.00        1.00      1.00      0.15      4.00    400.00      0.00
13       8         6         1       250.00        1.00      2.00      0.15      4.00     85.00      0.00
14       9         5         1       350.00        1.00      1.00      0.15      4.00    295.00      0.00
```

Figure 44 PFE File

Remarks:

There are 9 nodes and 14 links as stated. IDs of intermediate nodes (nodes 11, 12, 13) were renumbered after the IDs of TAZs immediately. Nodes 11, 12, and 13 were renumbered to 7, 8, and 9 respectively. (see Figure 44)

The *.pfx (based on the trip table structure file entered)

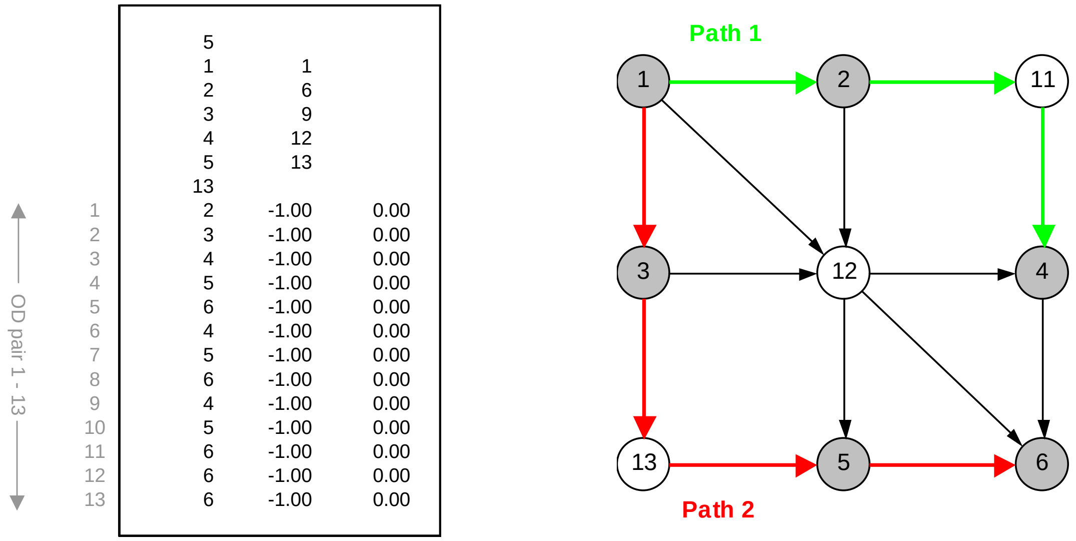

First OD pair of Origin 1
First OD pair of Origin 2
First OD pair of Origin 3
OD pair 1 - 9
Required OD data (see format in the manual)

```
        3
        1         1
        2         4
        3         7
        9
1       4     -1.00      0.00
2       5     -1.00      0.00
3       6     -1.00      0.00
4       4     -1.00      0.00
5       5     -1.00      0.00
6       6     -1.00      0.00
7       4     -1.00      0.00
8       5     -1.00      0.00
9       6     -1.00      0.00
```

Remarks:

If you left the trip table structure empty in the Create Network Window and the PFE determine the structure of the *.pfe file, the result will look like the one in the following figure.

***.pfx (determined through network topology)**

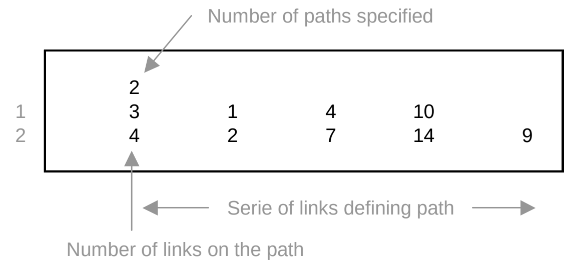


```
                5
                1          1
                2          6
                3          9
                4         12
                5         13
               13
       1        2      -1.00      0.00
       2        3      -1.00      0.00
       3        4      -1.00      0.00
       4        5      -1.00      0.00
       5        6      -1.00      0.00
       6        4      -1.00      0.00
       7        5      -1.00      0.00
       8        6      -1.00      0.00
       9        4      -1.00      0.00
      10        5      -1.00      0.00
      11        6      -1.00      0.00
      12        6      -1.00      0.00
      13        6      -1.00      0.00
```

Figure 45 PFX file and Paths on Network

Based on network topology, nodes 4 and 5 can also be the origin. The total number of origin nodes increases from 3 to 5. Similarly, the number of possible O-D pairs is 13 (compared to 9 O-D pairs specified in the 9GRDOD.dbf). (see Figure 45)

*.pfp (the path file of the created network)



```
2
3      1      4      10
4      2      7      14      9
```

Number of paths specified
1
2
Serie of links defining path
Number of links on the path

Remarks:

If there is no path specified by the user (e.g., leave the “User-specified Path” file blank), the 9GRD.pfp file will contain “0” as the number of paths specified, and the list of paths will be empty. Path 1-4-10, which is defined by a link sequence, is equivalent to Path 1->2->7(11)->4 defined by a node sequence. “7” is the new ID of node “11”. (see Figure 46)

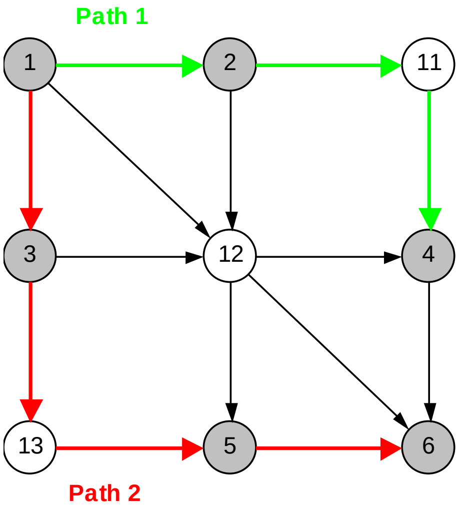


Figure 46 Paths Data

What are the PFE Input Data Files?

PFE requires three input files (*.pfe, *.pfx, and *.pfp) as listed below to perform the estimation.

- **pfe-file** – describes network topology (i.e., connectivity), contains link characteristics and observed link volumes,
- **pfx-file**– describes the skeleton (structure) of O-D trip table to be estimated as well as target O-D volumes (if available), and
- **pfp-file** – defines a certain set of routes that the user may particularly want to include into the estimation.

Common network data listed below are utilized to prepare these PFE input files.

Node-related information

- Node's ID
- X-coordinate of node
- Y-coordinate of node

Link-related information

- Starting node of link
- Ending node
- Functional class of link
  Remark: *the highest (maximum) functional class, defined by the user, must be reserved for centroid connectors. This is required for path-building purpose.*

- Link capacity
- Free-flow speed
- Link length
- Two parameters of BPR link cost function, α and β respectively

$$\text{travel time} = \text{free - flow travel time} \times \left(1.0 + \alpha \cdot \left(\frac{\mathit{link\ volume}}{\mathit{link\ capacity}} \right)^{\beta} \right)$$

- Observed link volume
- Measurement error of traffic count (e.g., percentage error – ±5%)

O-D related information

- ID of origin node
- ID of destination node

- Target O-D flow (if available)
- Confidence of target value (if target O-D flows are available)

User-specified path information

- List of paths
- Number of nodes defining path
- A sequence of nodes defining paths (from the origin to the destination)

Description of a Sample Network

The sample grid network (9GRD), depicted in Figure 1, consists of 9 nodes and 14 links. There are 6 traffic analysis zones (TAZs); all shaded nodes (nodes 1 through 6), and 9 O-D pairs. To synthesize observed traffic volumes, it is assumed that the true O-D trip table is available as shown below (Table 1). Network characteristics and synthesized link volumes are summarized in Table 2.

Table 1. True trip table for 9GRD network

From/To	4	5	6
1	120	150	100
2	130	200	90
3	80	180	110

Table 2. Link characteristics and observed volumes

A Node	B Node	Capacity	Speed	Length	Observed volume
1	2	280.00	1.00	2.00	124.00
1	3	290.00	1.00	1.50	137.00
1	12	280.00	1.00	3.00	109.00
2	11	280.00	1.00	1.00	77.00
2	12	600.00	1.00	1.00	467.00
3	12	500.00	1.00	2.00	212.00
3	13	400.00	1.00	1.00	295.00
4	6	300.00	1.00	1.00	50.00
5	6	220.00	1.00	1.00	165.00
11	4	300.00	1.00	2.00	77.00
12	4	500.00	1.00	1.50	303.00
12	5	700.00	1.00	1.00	400.00
12	6	250.00	1.00	2.00	85.00
13	5	350.00	1.00	1.00	295.00

Node and Link Numbering Scheme

Node numbering scheme must follow the requirement of Visual PFE as stated below.

TAZs (origins or destinations) "must" be numbered with a smaller ID compared to all other intermediate nodes (e.g., road intersections)

Similarly, links also need to be numbered according to a certain scheme (format) stated in the manual. By recognizing the fact that the formats required by Visual PFE are quite restrictive and laborious to prepare for real applications, "**network-creating tool**", being illustrated next, was also developed to assist the creation of PFE network. **The minimum requirement for this tool is that network nodes "must" be numbered according to the node-numbering scheme mentioned above**.

<u>Example</u>

Consider the sample network in Figure 47,

- All shaded nodes, which are either origins or destinations, are numbered from 1 to 6, which are smaller than IDs of all other intermediate nodes (nodes 11 through 13).
- The user may also be renumbered TAZs in many different ways (see below) as long as the required numbering scheme is satisfied.
- The O-D structure (see Table 1) **must** be consistent with node IDs defined by the user.

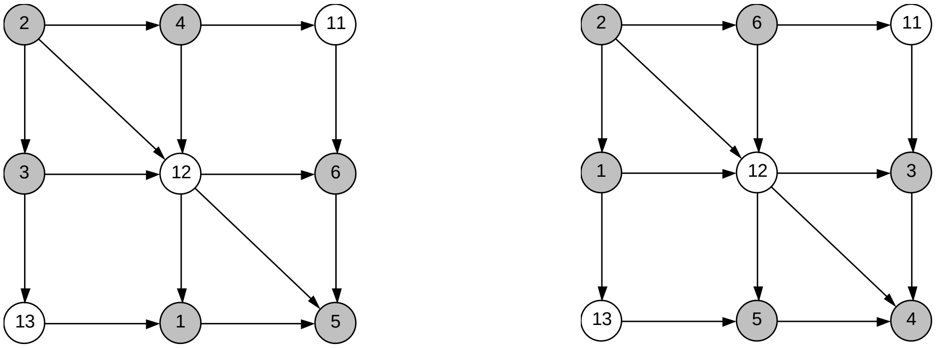


Figure 47 Example Network

<u>Remark</u>: *gap in the sequence of node IDs for TAZs should be avoided as much as possible*.