An efficient algorithm for real-time estimation and prediction of dynamic OD tables

M. Bierlaire and F. Crittin

April 3, 2002

Abstract

The problem of estimating and predicting Origin-Destination (OD) tables is known to be important and difficult. In the specific context of Intelligent Transportation Systems (ITS), the dynamic nature of the problem and the real-time requirements make it even more intricate.

We consider here a least-square modeling approach for solving the OD estimation and prediction problem, which seems to offer convenient and flexible algorithms. The dynamic nature of the problem is represented by an auto-regressive process, capturing the serial correlations of the state variables. Our formulation is inspired from Cascetta et al. (1993) and Ashok and Ben-Akiva (1993). We compare the Kalman filter algorithm to LSQR, an iterative algorithm proposed by Paige and Saunders (1982) for the solution of large-scale least-squares problems. LSQR explicitly exploits matrix sparsity, allowing to consider larger problems, likely to occur in real applications.

We show that the LSQR algorithm significantly decreases the computation effort needed by the Kalman filter approach for large-scale problems. We also provide a theoretical number of flops for both algorithms, in order to predict which algorithm will perform better on a specific instance of the problem.

1 Introduction

The development of Intelligent Transportation Systems (ITS) has considerably changed the field of transportation modeling during the past ten years. Indeed, the potential of these systems requires from transportation modelers the ability to explicitly capture the interaction between travelers and ITS, between demand and supply.

With regard to the demand aspects, the estimation and prediction of OD tables has become an important element of Dynamic Traffic Management Systems (DTMS) (Ashok and Ben-Akiva, 1993, Bierlaire et al., 2000, Ben-Akiva et al., 2002). The main difficulty of the problem is due to the following characteristics:

1. The dynamic nature of the process must be captured in the modeling framework.

2. Only indirect measurements of OD flows can be obtained through link flows. Therefore, the estimation problem is intrinsically under-determined for non trivial problems, as there are usually more unknowns than the number of observations.

3. Due to real-time requirements of DTMS, current and future OD flows must be available at any point in time, based on the most up-to-date data. Then, as time proceeds and more data becomes available, the solution must be updated to reflect the evolution of the network conditions.

Several approaches have been proposed in the literature to model the dynamic nature of demand. van der Zijpp (1996) proposes an approach based on time-space trajectories, Chang and Wu (1994) use a random walk model, Okutani (1987) describes the dynamic through an auto-regressive formulation capturing serial formulation across OD flows of subsequent time intervals. Ashok and Ben-Akiva (1993) propose an auto-regressive process as well, but based on the deviations between actual and historical OD flows.

Overcoming the under-determination of the estimation problem has been also captured in various ways. For static OD estimation, concepts like gravity (Casey, 1955), entropy (Wilson, 1970, Willumsen, 1981) and information theory (Van Zuylen and Willumsen, 1980) have been proposed. However, the use of an a priori OD table, derived from surveys or from previous studies is the most common way to overcome the under-determination. For dynamic OD estimation, the a priori table may be obtained from historical database, from a one-step prediction of a table estimated for the previous time-interval, or even using probe vehicles data (see Ashok, 1996 for more details).

The Kalman filter algorithm (Kalman, 1960) has been widely proposed to accommodate the real-time requirements (Okutani and Stephanades, 1984, Ashok and Ben-Akiva, 1993, Ashok and Ben-Akiva, 2000, Chang and Wu, 1994, van der Zijpp and Hammerslag, 1994). This algorithm solves a least-square problem in an incremental fashion, allowing to update the solution when additional data is available.

In this paper, we derive a least-square model, combining the formulation proposed by Cascetta et al. (1993) and Ashok and Ben-Akiva (1993). The state variables are the deviations between historical and actual OD flows. The main motivation is to indirectly take into account all experiences gained over many prior estimation, and accumulated in the historical data. Moreover it also gives statistical stability as the deviations can be more realistically assumed to be normally distributed with zero mean.

The Kalman filter algorithm has been implemented in the DynaMIT system (Antoniou et al., 1997, Ben-Akiva et al., 2002, Ben-Akiva et al., 2001). The main drawback of the Kalman filter algorithm appears to be its inability to handle large-scale problems. Indeed, even if efficient implementations are used (Chui and Chen, 1991), the analytical computation of the normal equations and the variance propagation require intensive linear algebra computation. Moreover, the sparsity of the least-square problem is not exploited by the algorithm, and a lot of fill-in is taking place. Another limitation of the Kalman filter is its constant numerical complexity. When traffic conditions are normal, or when the time intervals are short, the auto-regressive process can provide a pretty accurate estimate of the OD table. The Kalman filter algorithm always consumes the same amount of computational resources, irrespectively of the quality of the a priori matrix.

It is important to make the distinction between the *model formulation* and the *solution algorithm*. Usually, the model formulation is motivated by the use of the Kalman filter algorithm (e.g. Ashok, 1996), and the name *Kalman filter* refers to both the model and the algorithm. We consider Kalman filtering as an incremental algorithm to solve a least-square problem in a real-time context (Bertsekas, 1995). The use of a least-square approach to solve the dynamic OD estimation problem has been originally proposed by Cascetta et al. (1993). In this paper, we build on their modeling framework by (i) exploiting Ashok and Ben-Akiva, 1993 proposal of using deviations as state variables, and an auto-regressive model combined with historical data to obtain an a priori OD table, and (ii) providing an efficient algorithm to solve the problem in real-time.

We propose to use the LSQR algorithm (Paige and Saunders, 1982), analytically equivalent to a conjugate gradient method, requiring only matrix-vector products and, therefore, explicitly accounting for the problem's sparsity. In order to avoid to compute the variance propagation, which produces a great deal of fill-in in the matrices, all OD tables, for all time intervals within the considered horizon, must be included in the state vector. The associated model therefore grows with time, and may become intractable. This is not acceptable for systems supposed to run continuously in time, *i.e.*with a virtually infinite horizon. Therefore, we include only a limited number of past time intervals in the estimation process. We assume that the estimators and associated variance-covariance matrices for previous time intervals are given. This assumption is reasonable for all practical purposes, as the impact of new data on old OD tables becomes insignificant with time.

In theory, LSQR converges in n iterations, where n is the number of variables to estimate. In the case where the actual deviations are sufficiently well predicted by the auto-regressive process, the iterative nature of LSQR makes it converge in a few iterations, significantly decreasing the computational burden. The Kalman filter algorithm, based on a direct method, has a constant computational cost and, therefore, does not exploit such advantage.

2 Least-square formulation of the model

The model presented here is directly derived from Ashok and Ben-Akiva (1993). We consider an analysis period divided into equal intervals $h = 1, \dots, N$. The network is modeled by a directed graph $(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of links. Origin-Destination (OD) pairs form a subset of $\mathcal{N} \times \mathcal{N}$ of cardinality n_{OD}. We denote by $x_h \in \mathbb{R}^{n_{OD}}$ the actual OD table capturing all trips departing during time interval h, and by x_h^H the associated historical OD table. The vector of deviations is denoted by $\partial x_h = x_h - x_h^H$. We assume that n_ℓ links from $\mathcal{L}$ are equipped with sensors able to count the number of vehicles during a given time interval. We note $y_{\ell h}$ the number of vehicles crossing sensor ℓ during time interval h, and $y_h \in \mathbb{R}^{n_\ell}$ the vector gathering all such counts. The model is composed of the *transition equations*, capturing the dynamic of the system, and the *measurement equation*, mapping the state variables onto the data.

The transition equations are based on an auto-regressive process on the OD flows deviations, which provides a preliminary estimate of the OD flow. They are given by:

$$\partial x_h = \sum_{p=h-q'}^{h-1} f_h^p \partial x_p + w_h, \tag{1}$$

where f_h^p, a $n_{OD} \times n_{OD}$ matrix, represents the contribution of ∂x_p to ∂x_h, q' is the number of former time intervals influencing ∂x_h and w_h is a vector of random variables capturing the error. Note that f_h^p are usually sparse in most practical applications. Namely, f_h^p is often computed from linear regression models for each OD pair. In that case, f_h^p matrices are diagonal, as correlation across ODs are therefore ignored. We make the following assumptions on w_h:

- $E[w_h] = 0$,
- $E[w_h w_t'] = Q_h \delta_{ht}$, where Q_h is a $(n_{OD} \times n_{OD})$ variance-covariance matrix, and δ_{ht} is the Kronecker symbol.

The measurement equations capture the relationship between the state variables (OD deviations), and the measurements (sensor data):

$$y_h = \sum_{p=h-p'}^{h} a_h^p x_p + v_h, \tag{2}$$

where $y_h \in \mathbb{R}^{n_\ell}$ contains the sensor data for time interval h, a_h^p is a $n_\ell \times n_{OD}$ matrix, called the assignment matrix, mapping OD flows departing during interval p to link flows observed during interval h. It captures network topology, route choice assumptions and travel time. These matrices are usually sparse, as it is not common that all OD flows use all sensors on the network, at every departure time interval. Finally, p' is the maximum number of time intervals needed to travel between any OD pair and v_h is a vector of random variables capturing the error measurement on sensor data during time interval h. We make the following assumptions on v_h:

- $E[v_h] = 0$
- $E[v_h v_t'] = R_h \delta_{ht}$, where R_h is an $n_\ell \times n_\ell$ variance-covariance matrix.

The assignment matrices a_h^p captures three aspects of the transportation system: the network topology (link-path incidence), the route choice model and the travel time across the network. Clearly, in congested networks, the matrices depend on the prevailing traffic conditions. Unfortunately, incorporating that dependence into the model significantly complicates the OD estimation problem. Therefore, most approaches assume that they are given for the OD estimation, and iterate between OD estimation and traffic assignment until some sort of convergence is reached. It is namely the approach suggested by the DynaMIT system (Ben-Akiva et al., 2001). Recent techniques, based on bilevel optimization problems, include explicitly traffic equilibrium conditions in the model (Florian and Chen, 1993, Barceló and Casas, 1999). Such issues are out of the scope of this paper, which focused on algorithmic enhancements.

Equation (2) is not based on deviations. Therefore, we prefer the following equivalent formulation:

$$\partial y_h = \sum_{p=h-p'}^{h} a_h^p \partial x_p + v_h, \tag{3}$$

where $\partial y_h = y_h - \sum_{p=h-p'}^{h} a_h^p x_p^H$.

We present now the least-square formulation of the real-time dynamic OD estimation and prediction problem. The size of the problem depends on data availability. We assume that sensor data is available for time intervals 1 to k. The least-square formulation is given by

$$\min_X \sum_{h=1}^{k} \|\Omega_h^{-1} C_h^N X - \Omega_h^{-1} z_h\|_2^2 + \sum_{h=k+1}^{N} \|\Omega_h^{-1} C_h^N X\|_2^2, \tag{4}$$

where

$$X = \begin{pmatrix} \partial x_1 \\ \vdots \\ \partial x_{N-1} \\ \partial x_N \end{pmatrix}, \tag{5}$$

and

$$z_h = \begin{pmatrix} 0_{n_{OD}\times 1} \\ \partial y_h \end{pmatrix}, \tag{6}$$

$$\Theta_h = \Omega_h \Omega_h^T = \begin{pmatrix} Q_h & 0 \\ 0 & R_h \end{pmatrix} = \begin{pmatrix} P_h P_h^T & 0 \\ 0 & S_h S_h^T \end{pmatrix}, \tag{7}$$

and

$$\begin{aligned} C_h^N &= \begin{pmatrix} 0 & \cdots & 0 & -f_h^{h-q'} & \cdots & \cdots & \cdots & -f_h^{h-1} & I & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & a_h^{h-p'} & \cdots & a_h^{h-1} & a_h^h & 0 & \cdots & 0 \end{pmatrix} \\ &= \begin{pmatrix} C_h^u \\ C_h^d \end{pmatrix} \end{aligned} \tag{8}$$

if $q' > p'$, and

$$\begin{aligned} C_h^N &= \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 & -f_h^{h-q'} & \cdots & -f_h^{h-1} & I & 0 & \cdots & 0 \\ 0 & \cdots & 0 & a_h^{h-p'} & \cdots & \cdots & \cdots & a_h^{h-1} & a_h^h & 0 & \cdots & 0 \end{pmatrix} \\ &= \begin{pmatrix} C_h^u \\ C_h^d \end{pmatrix} \end{aligned} \tag{9}$$

if $q' \leq p'$. Note that negative values of $h - p'$ and $h - q'$ are meaningless, and associated matrices are just ignored in the formulation. In general, we will denote by C_h^m, $k \leq m \leq N$, the $n_{OD} + n_\ell \times mn_{OD}$ matrix obtained from C_h^N by dropping the appropriate number of zeros on the right.

This (huge) least-square problem has Nn_{OD} unknowns and $Nn_{OD} + kn_\ell$ equations. It captures both estimation and prediction of the OD tables. Indeed, for any time interval h within the horizon, the solution of (4) provides an estimation of the OD tables up to interval k, and a prediction of OD tables for intervals $k+1$ to N.

It is important to note here that matrices f_h^p in (1) and a_h^p in (3) are very sparse for most realistic problems. The solution algorithms must exploit this sparsity in order to be able to handle large-scale problems.

From a practical viewpoint, problem (4) may be intractable when the number of state variables $n_{OD}N$ is large. However, the estimation and prediction problems can be treated separately. For the OD estimation problem, the structure of matrices C_h^N defined by (8) and (9) is such that only $n_{OD}s'$ state variables are actually updated for the OD estimation at each time interval, where $s' = \max(p', q')$. This must obviously be exploited in the implementation of any algorithm. Once the estimated OD tables are available, the predicted OD tables are obtained by a direct application of the auto-regressive process.

If $n_{OD}s'$ is still too large for a specific algorithm, the problem size must be reduced even more. This is achieved by keeping the state variables $\partial x_{k-s'}$, $\ldots$, $\partial x_{k-\tau-1}$ constant, and updating only $\partial x_{k-\tau}$, $\ldots$, ∂x_k, for a given τ such that $0 \leq \tau \leq s'$. This procedure has been adopted for the OD estimation and prediction model implemented in DynaMIT (Antoniou et al., 1997, Ben-Akiva et al., 2002, Ben-Akiva et al., 2001), with $\tau = 0$. Note that the procedure does not bias the results if all vehicles are observed during one of the time intervals k, $k-1, \ldots k-\tau$. It means that the sensors must be sufficiently close to each origin in the network, so that each vehicle can be observed during the first τ time intervals of its trip.

3 Solution algorithms

We present here two solution algorithms. The Kalman filter algorithm (Kalman, 1960) is designed to update the solution of a least-square problem in a real-time context, as more data is made available. We show that applying the Kalman filter algorithm to our least-square formulation leads to the exact same algorithm as Ashok and Ben-Akiva (1993). Then, we consider the LSQR algorithm, proposed by Paige and Saunders (1982), in order to exploit (i) the sparsity of the problem and (ii) the a priori solution as provided by the auto-regressive process.

3.1 Kalman Filter

The Kalman filter algorithm solves (4) in an iterative way. The algorithm for a general incremental least-square problem is described by Bertsekas (1995). We assume that the problem has been solved up to time interval $k-1$, with solution X_{k-1} and variance-covariance matrix H_{k-1}. The update of these quantities is made through a two stage process. The first stage incorporates the transition equation to obtain $\hat{X}_k$ and $\hat{H}_k$, while the second incorporates the measurement equation to obtain X_k and H_k. However, in order to obtain an efficient formulation, the special structure of the problem must be exploited, as described below. Incorporating the transition equation is equivalent to solve the following problem:

$$\min_X \left\| \left(\begin{array}{c|c} P_k^{-1} & 0 \\ \hline 0 & (\Omega_{k-1}^{tot})^{-1} \end{array} \right) \left(\begin{array}{c|c} -F_{k-1} & I \\ \hline C_{k-1}^{tot} & 0 \end{array} \right) X - \left(\begin{array}{c} 0 \\ (\Omega_{k-1}^{tot})^{-1} z_{k-1}^{tot} \end{array} \right) \right\|^2 \tag{10}$$

where

$$F_{k-1} = \begin{pmatrix} 0 \cdots 0 & f_k^{k-q'} \cdots f_k^{k-1} \end{pmatrix} \in \mathbb{R}^{n_{OD} \times (k-1) n_{OD}}, \tag{11}$$

and

$$C_{k-1}^{tot} = \begin{pmatrix} C_1^{k-1} \\ \vdots \\ C_{k-2}^{k-1} \\ C_{k-1}^{k-1} \end{pmatrix}, z_{k-1}^{tot} = \begin{pmatrix} z_1 \\ \vdots \\ z_{k-2} \\ z_{k-1} \end{pmatrix}, \Omega_{k-1}^{tot} = \begin{pmatrix} \Omega_1 & & 0 \\ & \ddots & \\ 0 & & \Omega_{k-1} \end{pmatrix}. \tag{12}$$

The dimensions of these matrices are reported in Table 5 in the appendix. Note that the lower part of (10) gathers the $k-1$ first terms of (4), and that the terms corresponding to the prediction problem have been dropped. The solution of (10), obtained from the normal equations (see Section 6 for details), is

$$\hat{X}_k = \begin{pmatrix} I \\ F_{k-1} \end{pmatrix} X_{k-1} \tag{13}$$

with variance-covariance matrix

$$\hat{H}_k = \left(\begin{array}{c|c} H_{k-1} & H_{k-1} F_{k-1}^T \\ \hline F_{k-1} H_{k-1} & F_{k-1} H_{k-1} F_{k-1}^T + Q_k \end{array} \right). \tag{14}$$

The measurement equation is incorporated now as follows, again based on Bertsekas (1995).

$$\begin{aligned} H_k &= \hat{H}_k + (C_k^d)^T R_k^{-1} C_k^d, & (15)\\ X_k &= \hat{X}_k + H_k^{-1}(C_k^d)^T (R_k^{-1}\partial y_k - R_k^{-1} C_k^d \hat{X}_k) & (16)\end{aligned}$$

From the Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1996), we have that

$$H_k^{-1} = \left(I - K_k C_k^d\right) \hat{H}_k^{-1}, \qquad (17)$$

where

$$K_k = \hat{H}_k^{-1}(C_k^d)^T \left(R_k + C_k^d \hat{H}_k^{-1}(C_k^d)^T\right)^{-1}. \qquad (18)$$

Using the development derived in Section 6, we obtain

$$X_k = \hat{X}_k + K_k(\partial y_k - C_k^d \hat{X}_k). \qquad (19)$$

Note that equations (13), (14), (17), (18) and (19) are equivalent to the algorithm proposed by Ashok and Ben-Akiva (1993). This result is important, as it proves that our approach of the problem is actually equivalent to theirs.

3.2 LSQR

LSQR is an iterative method for solving the least-square problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 \qquad (20)$$

when A is large and sparse. Proposed by Paige and Saunders (1982), it is analytically equivalent to the conjugate gradient method, which is iterative by nature. Its convergence is theoretically achieved within at most n iterations. LSQR, based on two bi-diagonalization procedures, generates a sequence of x_k such that the associated sequence of residual's norms monotonically decreases. It exhibits better numerical properties than the conjugate gradient method, especially when A is ill-conditioned. A key property of this algorithm is that the matrix A is used only to compute products of the form Ax or $A^T y$, where x and y are vectors of appropriate dimensions, which is particularly attractive for large sparse problems. Indeed, A does not need to be explicitly constructed and stored, which is a particularly appealing feature for solving (4), given its specific structure.

LSQR is detailed by Paige and Saunders (1982). It is not designed for real-time applications, and is designed to start from 0. However, it can be adapted to solve real-time problems. First, its iterative nature allows for an easy update of a previous estimate $\bar{x}$. Defining $y = x - \bar{x}$, (20) can be written as

$$\min_{y \in \mathbb{R}^n} \|Ay - (b - A\bar{x})\|_2^2. \tag{21}$$

We denote by

$$x^* = \mathrm{LSQR}(A, b, \bar{x}) = \bar{x} + \mathrm{argmin}_{y \in \mathbb{R}^n} \|Ay - (b - A\bar{x})\|_2^2. \tag{22}$$

Second, it can be applied in a real time context as follows.

Initialize When no sensor data is available, historical OD tables are the best estimates. Therefore, we set $X_0 = 0$, that is $\partial x_h = 0$, $h = 1, \ldots, N$ and $k = 0$.

For $k = 1, \ldots, N$ At each interval k, we incorporate more sensor data, and update the estimated and predicted OD tables accordingly as follows

$$X_k = \mathrm{LSQR}\left(\sum_{h=1}^{k} \Omega_h^{-1} C_h^k, \sum_{h=1}^{k} \Omega_h^{-1} z_h, X_{k-1}\right). \tag{23}$$

Contrarily to LSQR, the Kalman filter algorithm is incremental by nature. At each time interval k, it involves only the matrices C_k and Ω_k, and the vector z_k, while LSQR involves matrices from all previous time intervals as well (see (23)). Consequently, the size of the problem grows with time, and LSQR does not look like an appealing candidate for real-time applications at first glance. This is probably one of the reasons why the Kalman filter algorithm has been widely proposed for real-time applications in the literature. On the other hand, the Kalman filter ignores and destroys the sparsity of the matrices (see (13), (14), (17), (18) and (19)). In order for LSQR to be applied in a real-time context, the number of terms in (23) must be kept constant. Therefore, we propose to replace (23) by

$$X_k = \mathrm{LSQR}\left(\sum_{h=k-r'}^{k} \Omega_h^{-1} C_h^k, \sum_{h=k-r'}^{k} \Omega_h^{-1} z_h, X_{k-1}\right), \tag{24}$$

where r' must be greater or equal to s'. The choice of r' is a trade-off between accuracy of the solution, and computation burden. Indeed, ignoring the terms corresponding to time intervals 1 to $k - r' - 1$ slightly biases the solution. Actually, it is equivalent to ignore the estimation error of those time intervals, by not propagating the variance-covariance matrix.

Note that the bias can be reduced, while keeping the problem's sparsity, by propagating only the variance of the estimators. This would keep the variance-covariance matrix diagonal. However, we do not investigate this possibility. Indeed, it appears from the experiments we have conducted (see Section 4) that the bias associated with (24) is not significant. Finally, the OD prediction problem is not directly solved by LSQR, as it simply amounts to applying the auto-regressive process to the estimated deviations.

3.3 Theoretical comparison

We compare here the numerical complexity of both algorithms. Following Golub and Van Loan (1996), we count the number of floating point operations (flops) associated with each algorithm. Note that if A is a $m \times n$ matrix, and B is a $n \times p$ matrix, the product AB takes $2mnp$ flops. If C is an invertible matrix of dimension n, computing its inverse takes $2n^3$ flops.

An upper bound on the total number of flops to perform p iterations of the LSQR algorithm is

$$2Cu + 6\overline{m} + 4\overline{n} + p(2Cu + 6\overline{m} + 10\overline{n} + 25), \tag{25}$$

where Cu is the number of flops required to compute the matrix-vector products $\Omega_k^{-1} C_k X$ and $(\Omega_k^{-1} C_k)^T X$, $\overline{m} = (r'+1)(n_{OD} + n_\ell)$ and $\overline{n} = (s'+1+r')n_{OD}$.

We denote by d_A the density of a sparse matrix A, that is the number of nonzero entries divided by the total number of entries. We have that

$$d_{C_k} = \frac{(r'+1)n_{OD} + q'(r'+1)n_{OD}^2 d_f + (p'+1)(r'+1)n_\ell n_{OD} d_a}{(s'+1+r')n_{OD}(r'+1)(n_{OD}+n_\ell)}, \tag{26}$$

and

$$d_{\Omega_k^{-1}} = \frac{(r'+1)(n_{OD}^2 d_{Q^{-1}} + n_\ell^2 d_{R^{-1}})}{(r'+1)^2(n_{OD}+n_\ell)^2}, \tag{27}$$

where d_f is an upper bound on the density of matrices f_h^p defined in (1), and d_a is an upper bound on the density of assignment matrices a_h^p defined in (2). $d_{Q^{-1}}$ and $d_{R^{-1}}$ are similarly defined. Consequently,

$$\begin{aligned} Cu &= 2(s'+1+r')(r'+1)(n_\ell + n_{OD})n_{OD} d_{C_k} \\ &\quad + 2(r'+1)^2(n_{OD}+n_\ell)^2 d_{\Omega_k^{-1}}. \end{aligned} \tag{28}$$

For the Kalman filter algorithm, we assume that the result of the product of two sparse matrices is dense to obtain the number of flops for each equation.

Eq. (13) $n_{OD} + q' d_f n_{OD}^2$,

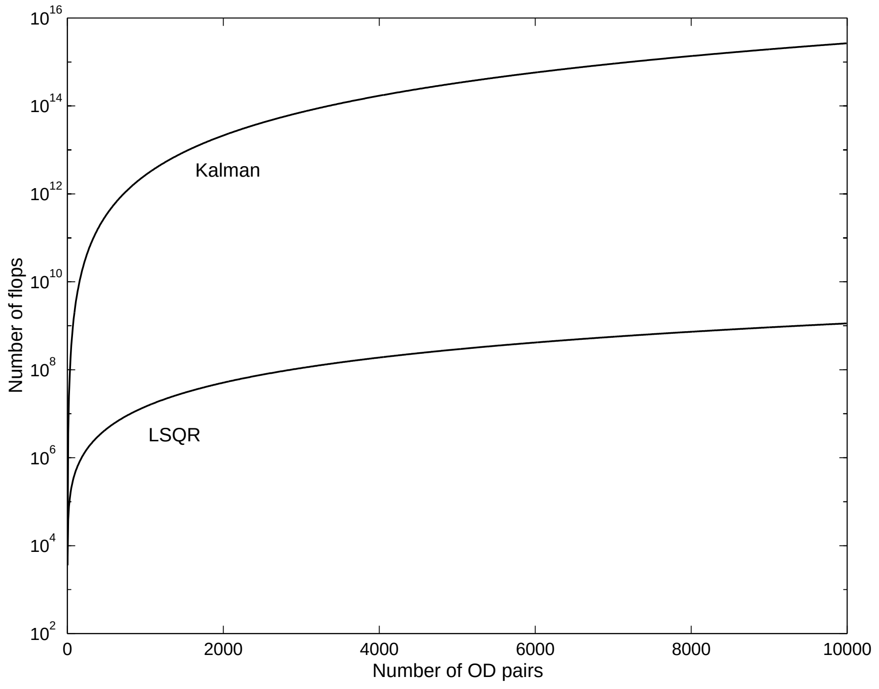


Figure 1: Kalman-LSQR flops comparison

Eq. (14) $2(q')^3 n_{OD}^3 d_f + n_{OD}^2 d_Q$,

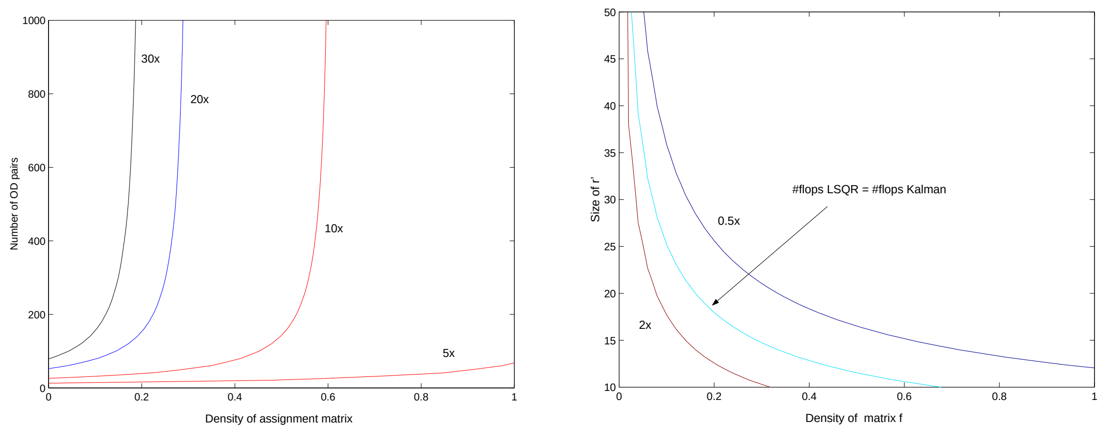
Eq. (17) $(s'+1)n_{OD}(2n_\ell d_a + 2(s'+1)^2 n_{OD}^2 + 1)$,

Eq. (18) $4n_\ell(s'+1)^2 n_{OD}^2 d_a + n_\ell^2 d_R + 2n_\ell^{3‘} + 2(s'+1)n_{OD}n_\ell^2$,

Eq. (19) $2(s'+1)n_{OD}(1 + (s'+1)n_{OD}d_a)$.

We illustrate these formulas in Figure 1, where the (logarithm of the) number of flops for each algorithm is plotted as a function of the number of ODs. We assume that $n_\ell = n_{OD}/10$, the variance-covariance matrices and the transition matrix are diagonal, and that $p' = r' = 10$ and $q' = 9$. The density of the assignment matrix is $d_a = 5\%$.

In Figure 2, we analyze scenarios where the relative performance of both algorithms is given. In Figure 2(a), we assume that $n_\ell = n_{OD}/10$, the variance-covariance matrices and the transition matrix are diagonal, and that $p' = r' = 10$ and $q' = 9$. The plotted curves represent combinations of values for n_{OD} and d_a such that the LSQR algorithm is 5, 10, 20 and 30 times (resp.) faster than the Kalman filter algorithm. In Figure 2(b), we have $n_{OD} = 1000$, $n_\ell = 100$ and $d_a = 2\%$. The plotted curves represent combinations of values for r' and



(a) n_{OD} vs. d_a (b) r' vs. d_f

Figure 2: Kalman-LSQR flops equivalence

d_f such that the LSQR algorithm performs as well, twice faster ($2\times$) and twice slower ($0.5\times$) (resp.) than the Kalman filter algorithm.

It clearly appears from these plots that LSQR is much more efficient than Kalman filter when sparse matrices are involved, which often occurs in practice. Interestingly, when the matrix sparsity is important (low values of d_f is Figure 2(b)), the r' parameter introduced to simplify the model in (24) can be set to a high value, while keeping the performance gain significant.

We complete the theoretical analysis by a simplification of the flops counting formulas, in order to obtain a level of magnitude. For this purpose, we assume that $n_{OD} = n^2$ and $n_\ell = \delta n$, where n is the number of nodes in the network, and δ is the average degree of the nodes. It is pessimistic, as all nodes are supposed to be both origins and destinations, and all links are assumed to be equipped with sensors.

If n is large, the dominant term is

$$4(p+1)(r'+1)n^4(q'd_f + d_{Q^{-1}}). \tag{29}$$

The sparsity of the auto-regressive process is therefore critical for the performance of the LSQR algorithm. A similar analysis for the Kalman filter algorithm leads to the following dominant term:

$$2(s'+1)^3 n^6(2d_f + 1). \tag{30}$$

We deduce from (29) and (30) that if the number of LSQR iterations p is such that

$$p \leq \frac{(s'+1)^3(2d_f+1)}{2(r'+1)(q'd_f+d_{Q^{-1}})}n^2, \tag{31}$$

than LSQR is more efficient than Kalman filter. In a real-time context, the number of LSQR iterations are usually low as the starting point is close to the solution when traffic conditions are more or less stable. In the worst case, LSQR theoretically performs $(r'+1)n^2$ iterations. In order for LSQR to be better, the density of the transition matrices has to be such that

$$d_f \leq \frac{(s'+1)^3-(r'+1)^2 d_{Q^{-1}}}{(r'+1)^2 q' - 2(s'+1)^3}. \tag{32}$$

Again, we observe that high values of r' are acceptable if the density of the transition matrices is low.

When both the transition matrices f_h^p and associated variance-covariance matrices are diagonal, the density of the assignment matrix becomes the dominant parameter in the flops computation. Indeed, assuming that $d_f = 1/n^2$ and $d_{Q^{-1}} = 1/n^2$, the dominant term for Cu in the number of flops (25) is

$$4(p+1)(r'+1)p'\delta n^3 d_a. \tag{33}$$

In that case, the complexity of LSQR depends on the density of the assignment matrix, and not any more on the density of the transition equations.

Considering again the worst case where LSQR performs $(r'+1)n^2$ iterations, the following condition must be verified in order for LSQR to be more efficient that the Kalman filter algorithm.

$$d_a \leq \frac{(s'+1)^3}{2(r'+1)^2 p'\delta}n. \tag{34}$$

Note that for large values of n and reasonable values of r', the density of the assignment matrix is irrelevant, and LSQR is systematically better than Kalman filter. This is illustrated in Figure 2(a).

4 Numerical Comparisons

We provide now an empirical comparison of the Kalman filter and LSQR algorithms to solve (4). Both algorithms have been implemented in Matlab (The Mathworks Inc., 1994), using the sparse matrices structure. The implementation of the Kalman filter algorithm that we use for these numerical comparisons actually uses less number of flops than the theoretical number estimated in Section 3.3. First, we use synthetic data in order to illustrate the accuracy of both approaches, when the "true" OD tables are known by the analyst. The objective is to illustrate the impact on the limitation of the number of terms in (24). Then, we present two case studies to compare the computational performance of the algorithms: a medium-scale model for the Central Artery/Third Harbor Tunnel (CA/T) network in Boston (Ma), and a large-scale model in Irvine (Ca).

4.1 Synthetic Data

We consider the network depicted in Figure 3, with three OD pairs {(1,5), (1,6), (2,6)}, and a time horizon of $N = 15$ time intervals of T minutes each. For the sake of simplicity, we assume for each link a travel time of T minutes and an infinite capacity. Consequently, $p' = 3$. We also assume that $q' = 2$ and, therefore $s' = 3$.

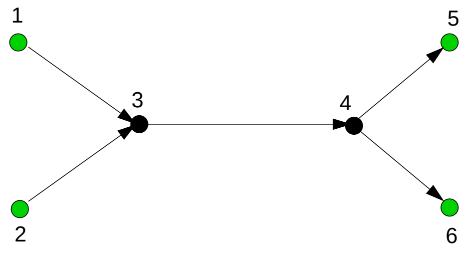


Figure 3: Simple Network

The "true" OD flows for pairs $(1,5)$ and $(1,6)$ are given in Table 1, where the unit is a number of cars per time interval (T minutes). The flows for OD pair $(2,6)$ are twice these values. Note that time intervals -4 to 0 are used to warm up the simulation and to avoid starting with an empty network.

Time int.	OD	Time int.	OD	Time int.	OD	Time int.	OD
-4	36	1	30	6	12	11	42
-3	12	2	48	7	12	12	24
-2	30	3	12	8	60	13	42
-1	12	4	18	9	42	14	12
0	36	5	36	10	66	15	18

Table 1: "True" demand for the simple network

The "true" assignment matrices are defined by

$$a_h^h = \begin{bmatrix} 1/3 & 1/3 & 0 \\ 0 & 0 & 1/3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad a_h^{h-1} = \begin{bmatrix} 1/3 & 1/3 & 0 \\ 0 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$a_h^{h-2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 \end{bmatrix} \quad a_h^{h-3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 \end{bmatrix}$$

The historical OD tables have been obtained by a random perturbation of the true OD. The link flows resulting from the assignment of the true OD tables have also been perturbed to obtain the sensor data. The auto-regressive process is such that $f_h^p = I$, for $p = h-3, h-2, h-1$. The variance-covariance matrices Q_k are diagonal, with the OD flows of the last time interval on the diagonal. Variance-covariance matrices R_k are diagonal with variance arbitrarily set to 1.

The relative error on estimated OD flows, that is

$$\frac{\|X_{\text{true}} - X_{\text{estimated}}\|}{\|X_{\text{estimated}}\|}, \tag{35}$$

obtained by each algorithm is illustrated on Figure 4. It appears, as expected, that the Kalman filter algorithm is more accurate than the LSQR algorithm from the 5th time interval on, as $r' + 1 = 4$. It is particularly noticeable that the difference remains almost constant (as it depends mainly on r') and is negligible.

4.2 Case-studies

DynaMIT is a state-of-the-art, real-time computer system for traffic estimation, prediction, and generation of traveler information and route guidance. It supports the operation of Advanced Traveler Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS) at Traffic Management Centers. DynaMIT is the result of about 10 years of intense research and development at the Intelligent Transportation Systems Program of the Massachusetts Institute of Technology (for description and details, see Ben-Akiva et al., 2002 and Bottom et al., 1999). DynaMIT's OD estimation and prediction

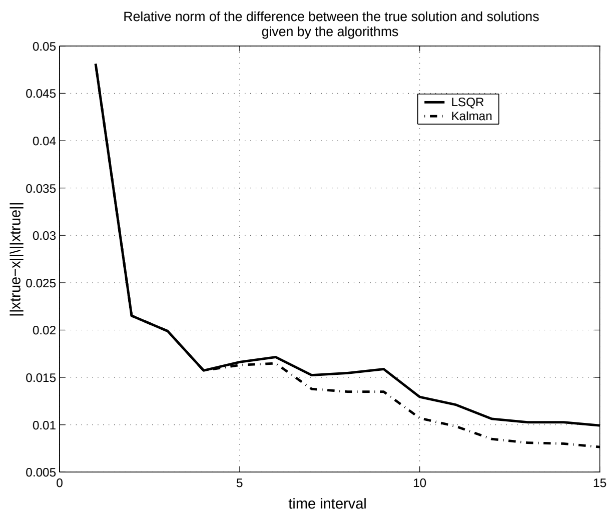


Figure 4: Error in the solution given by the algorithms

algorithm (Antoniou et al., 1997) is a Kalman filter algorithm directly derived from Ashok and Ben-Akiva (1993). In this paper, we have used DynaMIT to obtain assignment matrices for both case studies. We have also compared the results obtained with DynaMIT with those obtained with Matlab, in order to verify the algorithms implementation.

The first network is the Central Artery/Third Harbor Tunnel network, currently under construction (see Figure 5). It is a medium-scale network, with 211 links and 183 nodes. We consider a scenario with five origins and two destinations for a total of 10 OD pairs, and 35 link counts. We simulate 60 minutes during the morning from 7:00am to 8:00am. This simulation period is divided into 15 minutes time intervals. The results are described in Section 4.2.1.

The second network contains the major highways I-5, I-405 and CA-133 around Irvine, Ca. It contains also arterial roads in a triangular area defined by I-5, I-405 and Jeffrey road (see Figure 6). It is a large-scale network, with 618 links, 296 nodes and 627 OD pairs. We simulate 60 minutes during the morning from 7:15am to 8:15am. This simulation period is divided into 15 minutes time intervals. The results are described in Section 4.3. Irvine Network data comes from a traffic management center in Irvine, California.

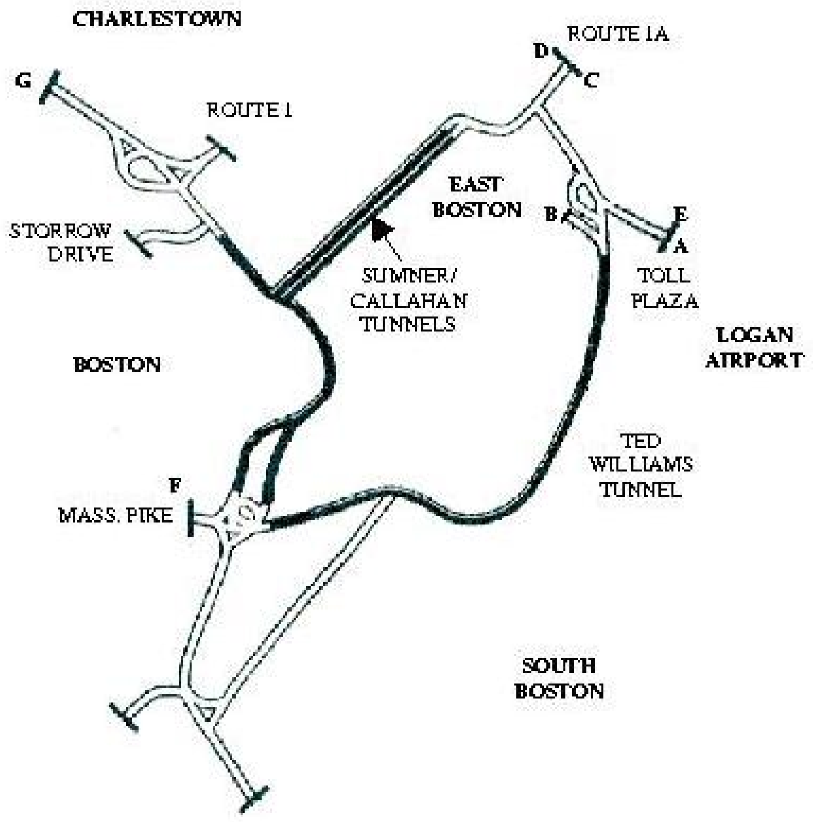


Figure 5: Central Artery/Third Harbor Tunnel network

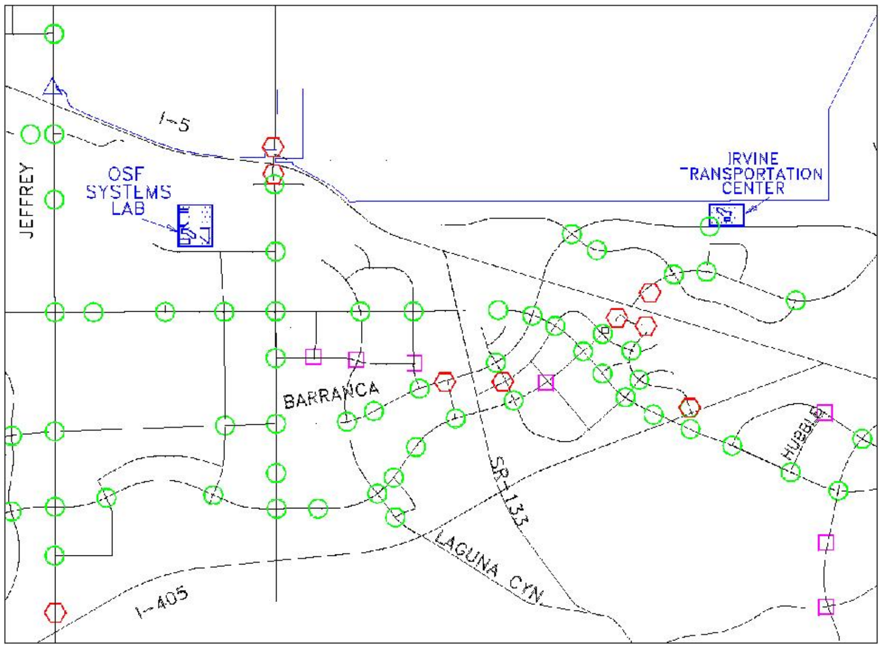


Figure 6: Irvine network

4.2.1 Central Artery network

We solve the OD estimation problem for the CA/T network with $p' = 0$ and $q' = r' = 1$. In table 2, we report (i) the average number of flops per time interval for the Kalman filter algorithm, as reported by Matlab, (ii) the same information for LSQR algorithm, (iii) the relative root mean squared error (RRMSE), that is

$$\frac{\sqrt{n_\ell \sum_{k=1}^{N} \sum_{j=1}^{n_{OD}} \left((\partial x_k)_j^{\text{Kalman}} - (\partial x_k)_j^{\text{LSQR}} \right)^2}}{\sum_{k=1}^{N} \sum_{j=1}^{n_{OD}} |((\partial x_k)_j^{\text{Kalman}}|} \tag{36}$$

and the relative mean error (RME), that is

$$\frac{\sum_{k=1}^{N} \sum_{j=1}^{n_{OD}} \left| (\partial x_k)_j^{\text{Kalman}} - (\partial x_k)_j^{\text{LSQR}} \right|}{\sum_{k=1}^{N} \sum_{j=1}^{n_{OD}} |((\partial x_k)_j^{\text{Kalman}}|}, \tag{37}$$

for various values of the ATOL parameter, ATOL being the tolerance on the normalized least-squares residual used as a stopping criterion for the LSQR algorithm (see Paige and Saunders, 1982).

In appears clearly from Table 2 that the empirical results are consistent with the theoretical analysis, and that LSQR significantly outperforms the Kalman

ATOL	flops Kalman	flops LSQR	RRMSE	RME
10^{-2}	38800	4100	0.979	0.766
10^{-3}	38800	5753	0.076	0.054
10^{-4}	38800	5949	0.044	0.037
10^{-5}	38800	6471	0.044	0.034

Table 2: Comparison for the CA/T network

filter algorithm. This instance, where LSQR is 6 times better than Kalman in the worst case (ATOL=10^{-5}) is representative of other experiments on problems of similar characteristics. LSQR also allows a trade-off between the results accuracy and computational burden. This is often critical for real-time applications. Setting the ATOL parameter to 10^{-2} produces an algorithm almost 10 times faster than Kalman, with a reasonable reduction of accuracy.

As a final note, the theoretical number of flops for LSQR (see Section 3.3) is about 10000, and for Kalman is about 160000. The discrepancy between theoretical and actual numbers of flops is due to the simplifying assumptions used in Section 3.3.

4.3 Irvine Results

We solve the OD estimation problem for the Irvine network with $p' = 0$ and $q' = r' = 1$. In table 3, we report (i) the average number of flops per time interval for the Kalman filter algorithm, as reported by Matlab, (ii) the same information for LSQR algorithm, (iii) the RRMSE (36) and (iv) the MSE (37).

ATOL	flops Kalman	flops LSQR	RRMSE	RME
10^{-2}	$7.38\ 10^7$	$5.42\ 10^5$	0.4262	0.3020
10^{-3}	$7.38\ 10^7$	$1.30\ 10^6$	0.3713	0.2380
10^{-4}	$7.38\ 10^7$	$2.34\ 10^6$	0.1435	0.1149
10^{-5}	$7.38\ 10^7$	$3.23\ 10^6$	0.1425	0.1146

Table 3: Comparison for the Irvine network

The LSQR algorithm solves the problem from 23 times (ATOL=10^{-5}) to 136 times (ATOL=10^{-2}) faster than Kalman. As predicted by the theoretical analysis, the advantage of using LSQR becomes more significant when the size of the problem increases.

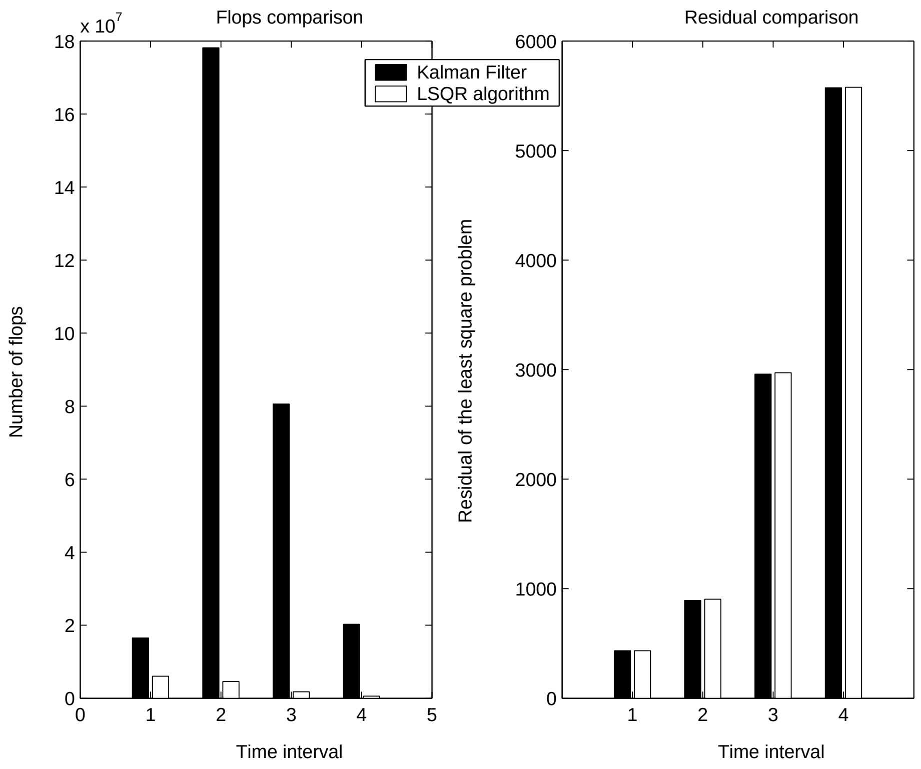


Figure 7: Results for the Irvine network

Figure 7 shows for each time interval and for each algorithm, the number of flops on the left and the value of the residual on the right, *i.e.* $\|\Omega_h^{-1}C_hX_h - \Omega_h^{-1}z_h\|$. It clearly demonstrates the superiority of LSQR method in large scale case, with a similar quality of the result.

Note that the theoretical number of flops for LSQR is about 9 10^6, and for Kalman is 4 10^9. Again, the discrepancy between theoretical and actual number of flops in Table 3 is due to the simplifying assumptions used in Section 3.3.

Note that we prefer to mention flops instead of computation time to decrease the impact of a specific computer hardware, MATLAB's overhead, memory access and implementation. However, these times provide us with the same qualitative conclusions. As an example, Table 4 reports the run time of each algorithm on Irvine case study. These times have been obtained by running MATLAB on a Dell PowerEdge 6300 with 4 Intel Pentium II, 500 Mhz, 1 Gb memory.

Time Interval	1	2	3	4
Kalman Filter	1.03	5.69	2.35	1.27
LSQR algorithm	1.66	1.49	0.73	0.24

Table 4: CPU Time computation in seconds for the Irvine network

Finally, we have run the Irvine case study with $p' = r' = 4$ and $q' = 3$. In that case, we were not able to solve it with the Kalman filter algorithm, which exhausted the available memory in Matlab. The LSQR algorithm has been able to solve the problem in about $6.7\ 10^7$ flops. Note that this is less than the number of flops reported for the Kalman filter algorithm in Table 3.

5 Conclusion

In this paper, we have proposed a least-square formulation of the real-time dynamic OD estimation and prediction problem, based on a combination of the approaches by Cascetta et al. (1993) and Ashok and Ben-Akiva (1993). In order to emphasize the model's validity, we have shown that applying the Kalman filter algorithm as presented by Bertsekas (1995) leads to the exact same algorithm as Ashok and Ben-Akiva (1993).

The proposed formulation enables to directly use algorithm LSQR to solve the problem. Proposed by Paige and Saunders (1982), LSQR is a numerically robust conjugate gradient algorithm designed to solve large-scale sparse least-square problems. Because it is not designed for real-time applications, we have imposed a simplification to maintain the size of the problem constant over time. This simplification amounts not to propagate the variance-covariance matrix of old estimated matrices.

Both the theoretical estimation of the flops and empirical comparisons on real data exhibit a significantly better performance for the LSQR algorithm, in the presence of sparse matrices. We have also shown that the model simplification has a limited impact on the quality of the solution.

The Kalman filter approach based on the normal equations cannot afford large-scale problems, as it involves the multiplication and inversion of very large matrices. The computational complexity of the LSQR algorithm is based only on its ability to multiply a large matrix by a vector. If the large matrix is very sparse, as it is often the case in practice, such a procedure can be implemented efficiently. Also, the iterative nature of LSQR allows, contrarily to Kalman, to exploit previous estimates when the traffic conditions are stable, performing less iterations to converge to the solution.

A direct extension of the algorithm presented in this paper is obtained when the OD deviations are constrained by lower and upper bounds. One important motivation is to avoid negative OD flows when the deviations are added to the historical values. The bound-constrained LSQR algorithm proposed by Bierlaire et al. (1991) can be considered in that case.

We emphasize that the least-square formulation adopted in this paper is well adapted when additional data can be considered in order to improve the quality of the estimated OD. For example, license plate data collected in parking lot (Bierlaire and Toint, 1995) or probe vehicles data based on GPS, ETC or cellular phone technologies (Smith et al., 2001). Contrarily to the Kalman filter algorithm, which requires to re-derive the equations, the LSQR algorithm can be used as is, with the extended formulation.

Finally, the ideas proposed in this paper are not restricted to the specific model formulation proposed by Cascetta et al. (1993) and Ashok and Ben-Akiva (1993). Our decision to adopt this model is to provide a formal and unbiased comparison between the algorithms in a given context. But our ideas can be applied to any model that aims to solve a real-time problem with a Kalman filter approach.

6 Appendix

In this appendix, we provide some technical derivation for the Kalman filter algorithm. First, we recall how (10) can be obtained from the normal equation. The solution of a least square problem

$$\min_x \|Ax - b\|^2 \tag{38}$$

is obtained from the normal equation $A^T A x = A^T b$. The solution is

$$(A^T A)^{-1} A^T b, \tag{39}$$

with variance-covariance

$$(A^T A)^{-1}. \tag{40}$$

The normal equation for (10) is

$$\begin{aligned}
&\left(\begin{array}{c|c} -F_{k-1}^T & (C_{k-1}^{tot})^T \\ \hline I & 0 \end{array}\right)\left(\begin{array}{c|c} Q_k^{-1} & 0 \\ \hline 0 & \Theta_{k-1}^{-1} \end{array}\right)\left(\begin{array}{c|c} -F_{k-1} & I \\ \hline (C_{k-1}^{tot}) & 0 \end{array}\right) X = \\
&\left(\begin{array}{c|c} -F_{k-1}^T & (C_{k-1}^{tot})^T \\ \hline I & 0 \end{array}\right)\left(\begin{array}{c|c} Q_k^{-1} & 0 \\ \hline 0 & \Theta_{k-1}^{-1} \end{array}\right)\begin{pmatrix} 0 \\ z_{k-1}^{tot} \end{pmatrix}.
\end{aligned} \tag{41}$$

From (40), we have that

$$\hat{H}_k^{-1} = \left(\begin{array}{c|c} -F_{k-1}^T & (C_{k-1}^{tot})^T \\ \hline I & 0 \end{array}\right)\left(\begin{array}{c|c} Q_k^{-1} & 0 \\ \hline 0 & \Theta_{k-1}^{-1} \end{array}\right)\left(\begin{array}{c|c} -F_{k-1} & I \\ \hline (C_{k-1}^{tot}) & 0 \end{array}\right). \tag{42}$$

A direct multiplication of (14) and (42) shows that $\hat{H}_k \hat{H}_k^{-1} = I$. The solution (13) is obtained from (39) and (14). We have

$$\begin{aligned}
&\left(\begin{array}{c|c} H_{k-1} & H_{k-1}F_{k-1}^T \\ \hline F_{k-1}H_{k-1} & F_{k-1}H_{k-1}F_{k-1}^T + Q_k \end{array}\right)\left(\begin{array}{c|c} -F_{k-1}^T & (C_{k-1}^{tot})^T \\ \hline I & 0 \end{array}\right)\left(\begin{array}{c|c} Q_k^{-1} & 0 \\ \hline 0 & \Theta_{k-1}^{-1} \end{array}\right)\begin{pmatrix} 0 \\ z_{k-1}^{tot} \end{pmatrix} \\
&= \left(\begin{array}{c|c} -H_{k-1}F_{k-1}^T + H_{k-1}F_{k-1}^T & H_{k-1}(C_{k-1}^{tot})^T \\ \hline -F_{k-1}H_{k-1}F_{k-1}^T + F_{k-1}H_{k-1}F_{k-1}^T + Q_k & F_{k-1}H_{k-1}(C_{k-1}^{tot})^T \end{array}\right)\begin{pmatrix} 0 \\ \Theta_{k-1}^{-1} z_{k-1}^{tot} \end{pmatrix} \\
&= \left(\begin{array}{c|c} 0 & H_{k-1}(C_{k-1}^{tot})^T \\ \hline Q_k & F_{k-1}H_{k-1}(C_{k-1}^{tot})^T \end{array}\right)\begin{pmatrix} 0 \\ \Theta_{k-1}^{-1} z_{k-1}^{tot} \end{pmatrix} \\
&= \begin{pmatrix} H_{k-1}(C_{k-1}^{tot})^T \Theta_{k-1}^{-1} z_{k-1}^{tot} \\ F_{k-1}H_{k-1}(C_{k-1}^{tot})^T \Theta_{k-1}^{-1} z_{k-1}^{tot} \end{pmatrix} = \begin{pmatrix} I \\ F_{k-1} \end{pmatrix} H_{k-1}(C_{k-1}^{tot})^T \Theta_{k-1}^{-1} z_{k-1}^{tot}.
\end{aligned}$$

Eq.	Matrix	Rows	Columns
(10)	P_k	n_{OD}	n_{OD}
(10)	Ω^{tot}_{k-1}	$n_{OD}+n_\ell$	$n_{OD}+n_\ell$
(10)	F_{k-1}	n_{OD}	$(k-1)n_{OD}$
(10)	C^{tot}_{k-1}	$(k-1)(n_{OD}+n_\ell)$	$(k-1)n_{OD}$
(10)	X_{k-1}	$(k-1)n_{OD}$	1
(10)	z^{tot}_{k-1}	$(k-1)(n_{OD}+n_\ell)$	1
(13)	$\hat{X}_k$	kn_{OD}	1
(14)	H_{k-1}	$(k-1)n_{OD}$	$(k-1)n_{OD}$
(14)	$\hat{H}_k$	kn_{OD}	kn_{OD}
(18)	K_k	kn_{OD}	n_ℓ
(18)	R_k	n_ℓ	n_ℓ

Table 5: Matrices dimensions

We finally obtain (13) by noting that

$$X_{k-1} = H_{k-1}(C^{tot}_{k-1})^T\Theta^{-1}_{k-1}z^{tot}_{k-1}.$$

To show (19), we use (17) in (16),

$$X_k = \hat{X}_k + \left(I - K_kC^d_k\right)\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}\left(\partial y_k - C^d_k\hat{X}_k\right). \tag{43}$$

Denoting $g_k = \left(\partial y_k - C^d_k\hat{X}_k\right)$, we show that

$$\begin{aligned}\left(I - K_kC^d_k\right)\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k &= \hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k - K_kC^d_k\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k \\ &= K_kg_k.\end{aligned} \tag{44}$$

Indeed

$$\begin{aligned}&K_kC^d_k\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k \\ =\ &\hat{H}_k^{-1}(C^d_k)^T\left(R_k + C^d_k\hat{H}_k^{-1}(C^d_k)^T\right)^{-1}C^d_k\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k \\ =\ &\hat{H}_k^{-1}(C^d_k)^T\left(R_k + C^d_k\hat{H}_k^{-1}(C^d_k)^T\right)^{-1}\left(C^d_k\hat{H}_k^{-1}(C^d_k)^T + R_k\right)R_k^{-1}g_k \\ &-\hat{H}_k^{-1}(C^d_k)^T\left(R_k + C^d_k\hat{H}_k^{-1}(C^d_k)^T\right)^{-1}R_kR_k^{-1}g_k \\ =\ &\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k - \hat{H}_k^{-1}(C^d_k)^T\left(R_k + C^d_k\hat{H}_k^{-1}(C^d_k)^T\right)^{-1}g_k \\ =\ &\hat{H}_k^{-1}(C^d_k)^TR_k^{-1}g_k - K_kg_k, \quad \text{from (18).}\end{aligned}$$

We conclude the appendix by providing in Table 5 the dimensions of the matrices appearing in Section 3.1.

References

Antoniou, C., Ben-Akiva, M., Bierlaire, M. and Mishalani, R. (1997). Demand simulation for dynamic traffic assignment, *Proceedings of the 8th IFAC Symposium on Transportation Systems*, Chania, Greece.

Ashok, K. (1996). *Estimation and Prediction of Time-Dependent Origin-Destination Flows*, PhD thesis, Massachusetts Institute of Technology, Cambridge, Ma.

Ashok, K. and Ben-Akiva, M. (1993). Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems, *in* C. Daganzo (ed.), *Transportation and Traffic Theory*, Elsevier Science Publishing Company Inc. Proceedings of the 12th ISTTT.

Ashok, K. and Ben-Akiva, M. (2000). Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows, *Transportation Science* **34**(1): 21–36.

Barceló, J. and Casas, J. (1999). The use of neural networks for short-term prediction of traffic demand, *in* A. Ceder (ed.), *Transportation and Traffic Theory. Proceedings of the 14th ISTTT*, Pergamon, pp. 419–443.

Ben-Akiva, M., Bierlaire, M., Burton, D., Koutsopoulos, H. and Mishalani, R. (2001). Network state estimation and prediction for real-time transportation management applications, *Networks and Spatial Economics* **1**(3/4): 293–318.

Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H. N. and Mishalani, R. (2002). Real-time simulation of traffic demand-supply interactions within DynaMIT, *in* M. Gendreau and P. Marcotte (eds), *Transportation and network analysis: current trends. Miscellenea in honor of Michael Florian*, Kluwer Academic Publishers, Boston/Dordrecht/London.

Bertsekas, D. P. (1995). *Nonlinear Programming*, Athena Scientific, Belmont.

Bierlaire, M., Mishalani, R. and Ben-Akiva, M. (2000). General framework for dynamic demand simulation, *Technical Report RO-000223*, ROSO-DMA-EPFL Swiss Institute of Technology, CH-1015 Lausanne.

Bierlaire, M. and Toint, P. L. (1995). MEUSE: an origin-destination estimator that exploits structure, *Transportation Research B* **29**(1): 47–60.

Bierlaire, M., Toint, P. L. and Tuyttens, D. (1991). On iterative algorithms for linear least squares problems with bound constraints, *Linear Algebra and its Applications* **143**: 111–143.

Bottom, J., Ben-Akiva, M., Bierlaire, M., Chabini, I., Koutsopoulos, H. and Yang, Q. (1999). Investigation of route guidance generation issues by simulation with DynaMIT, *in* A. Ceder (ed.), *Transportation and Traffic Theory. Proceedings of the 14th ISTTT*, Pergamon, pp. 577–600.

Cascetta, E., Inaudi, D. and Marquis, G. (1993). Dynamic estimators of origin-destination matrices using traffic counts, *Transportation Science* **27**(4): 363–373.

Casey, H. J. (1955). Applications to traffic engineering of the law of retail gravitation, *Traffic Quaterly* **9**(1): 23–35.

Chang, G. L. and Wu, J. (1994). Recursive estimation of time-varying od flows from traffic counts in freeway corridors, *Transportation Research B* **28**.

Chui, C. and Chen, G. (1991). *Kalman Filtering with Real-Time Applications*, Spinger-Verlag.

Florian, M. and Chen, Y. (1993). A coordinate descent method for the bilevel OD matrix adjutment problem, *IFORS conference*, Lisbon.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*, third edn, Johns Hopkins University Press, Baltimore and London.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems, *J. of Basic Eng., Trans. ASME, Series D* **82**(1): 33–45.

Okutani, I. (1987). The Kalman filtering approaches in some transportation and traffic problems, *in* N. H. Gartner and N. H. M. Wilson (eds), *Transportation and traffic theory*, Elsevier, New-York, pp. 397–416. Proceedings of the Tenth International Symposium on Transportation and Traffic Theory, July 8–10 1987, MIT, Cambridge, Massachusetts.

Okutani, I. and Stephanades, Y. (1984). Dynamic prediction of traffic volume through kalman filtering theory, *Transportation Research* **18**.

Paige, C. C. and Saunders, M. A. (1982). LSQR: an algorithm for sparse linear equations and sparse least squares, *ACM Transactions on Mathematical Software* **8**: 43–71.

Smith, B. L., Pack, M. L., Lovell, D. J. and Sermons, M. W. (2001). Transportation management applications of anonymous mobile call sampling, *Proceedings of the 11th Annual Meeting of ITS America, Miami, FL.*

The Mathworks Inc. (1994). Matlab optimization toolbox, The Mathworks Inc.

van der Zijpp, N. (1996). *Dynamic Origin-Destination Matrix Estimation on Motorway Networks.*, PhD thesis, Dpt of Civil Rnginnering, Delft University of Technology.

van der Zijpp, N. and Hammerslag, R. (1994). Improved kalman filtering approach for estimating origin-destination matrices for freeway corridors, *Transportation Research Record* **1443**.

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

Willumsen, L. G. (1981). *An entropy maximising model for estimating trip matrices from traffic counts*, PhD thesis, University of Leeds.

Wilson, A. G. (1970). *Entropy in urban and regional modelling*, Pion, London.