Algorithm 900: A Discrete Time Kalman Filter Package for Large Scale Problems

GERMÁN A. TORRES
Universidad Nacional de Córdoba

Data assimilation is the process of feeding a partially unknown prediction model with available information from observations, with the objective of correcting and improving the modeled results. One of the most important mathematical tools to perform data assimilation is the Kalman filter. This is essentially a predictor-corrector algorithm that is optimal in the sense of minimizing the trace of the covariance matrix of the errors. Unfortunately, the computational cost of applying the filter to large scale problems is enormous, and the programming of the filter is highly dependent on the model and the format of the data involved. The first objective of this article is to present a set of Fortran 90 modules that implement the reduced rank square root versions of the Kalman filter, adapted for the assimilation of a very large number of variables. The second objective is to present a Kalman filter implementation whose code is independent of both the model and observations and is easy to use. A detailed description of the algorithms, structure, parallelization is given along with examples of using the package to solve practical problems.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: *Time series analysis*; G.4 [**Mathematical Software**]: *Algorithm design and analysis*; J.2 [**Physical Sciences and Engineering**]: *Earth and atmospheric sciences*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Large scale problems, data assimilation, Kalman filter

ACM Reference Format:

Torres, G. A. 2010. Algorithm 900: A discrete time Kalman filter package for large scale problems. ACM Trans. Math. Softw. 37, 1, Article 11 (January 2010), 16 pages.
DOI = 10.1145/1644001.1644012 http://doi.acm.org/10.1145/1644001.1644012

1. INTRODUCTION

The Kalman filter is a set of mathematical equations that combine information from a model output and observations to produce a better estimation

This work was carried out with the aid of a grant from the Inter-American Institute for Global Change Research (IAI) CRN II 2017, which is supported by the US National Science Foundation (Grant GEO-0452325) and with funds from the Secretaría de Ciencia, Tecnología e Innovación Productiva—SECYT (Grant A06/STICAMSUD/02).

Author's address: Facultad de Matemática, Astronomía y Física, Medina Allende s/n, Ciudad Universitaria, 5000, Córdoba, Argentina; email: torres@famaf.unc.edu.ar.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2010 ACM 0098-3500/2010/01-ART11 $10.00

DOI 10.1145/1644001.1644012 http://doi.acm.org/10.1145/1644001.1644012

(or analysis) of the dynamical system. Essentially, it implements a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance under some hypotheses. The state and the covariance matrix of forecast errors are predicted (using the model), and, if observation data is available, a correction step is performed (see Welch and Bishop [2001]).

The filter is named after Rudolph E. Kalman, who in 1960 published his paper describing a recursive solution to the discrete-data linear filtering problem [Kalman 1960]. The Kalman filter has been extensively applied in motion prediction (see Azuma and Bishop [1994]), parameter estimation (see Charalambous and Hibey [2001] and Annan et al. [2005]), navigation and global positioning systems (see Kim and Iltis [2002]), improvement of species concentrations in chemical transport models (see Zhang et al. [1997, 1999], van Loon et al. [2000], Flemming et al. [2001], El Serafy et al. [2002], and Segers [2002]), image treatment (see Ertürk [2002] and Kuo et al. [2002]), improvement of oceanographic models (see Allen et al. [2002] and Hoteit et al. [2004]), etc.

The size of the problem may restrict the application of the Kalman filter. For example, in a chemistry transport model of 20 species concentrations in a grid of $50 \times 50 \times 20$ cells, the state vector would be order $n = 10^6$ and we would need to operate on square matrices of order n. The most expensive part of the Kalman algorithm is the prediction of the covariance matrix of forecast errors, where we have to apply the tangent linear model $2n$ times. In complex models, a time step could take a few seconds making a standard Kalman filter step prohibitively expensive to implement. However, new techniques have been developed for large scale problems like reduced rank square root methods and ensemble methods based in Monte-Carlo estimations (see Brown and Gaston [1995], Chin et al. [1995], Pham et al. [1998], Segers et al. [2000], Hoteit et al. [2001, 2002], Evensen [2003], Asif [2004], Hoteit and Pham [2004], Treebushny and Madsen [2005], Chen et al. [2005] and Hanea et al. [2005]). According to the application, covariance matrices may have a sparse structure which could simplify some array operations and speed up execution (e.g., for observations the covariance matrices may be treated as diagonal). The user must decide whether or not to use reduced rank methods according to the complexity of the model.

Another common problem is related to implementation difficulties. Every time the filter is applied, the model needs to be called many times. Therefore, the model source code has to be rearranged. Noise has to be added to generate samples. An interface is needed between the incoming observations and the data produced by the model. After the assimilation step the model has to restart with the newly generated analysis, and the complete implementation is full of subtle relationships between the model and the observations. One major obstacle is that models are like black boxes, where the user changes configuration files and not the code itself. The source code for the model should not be changed unless the user is an expert.

This article proposes a modular assimilation environment in Fortran 90. It means that observations, model and assimilation are separated from one another. For example, if the observation stations change location, there is no need to modify the main code, but only a module related to locations. If we need to change the model, there is no need to transform all the code, but only the module related to the model. If we want to change the Kalman filter being used, a change in the module related to the assimilation will suffice. This will allow the user to make minimum changes to the code supporting the modeling system (model, data formats, libraries, configuration files) in order to avoid introducing coding bugs. The choice of the language was made because many large scale models are written either in Fortran 77 or in Fortran 90 (even when it is possible to mix languages, some users prefer not to so). The modules can handle single and double precision and use the BLAS/LAPACK libraries for the matrix operations. For the parallel implementation the BLACS, SCALAPACK and MPI libraries are used.

This article also proposes the implementation of a reduced rank square root ensemble Kalman filter taking advantage of the facilities in Fortran 90 for matrix-vector manipulation which is used extensively in 2D and 3D models. An example of a 3D system is presented and comparisons between the model solution and the true and assimilated solutions are given. Notice that the assimilation modules include other versions of the filter, like the complete extended Kalman filter and the ensemble Kalman filter (useful for small and medium scale problems), and the RRSQRT Kalman filter (useful for large scale problems).

Several other packages are available that implement the Kalman filter, for example,

(1) STSA (The Time Series Analysis Toolbox for O-Matrix [STSA]): This toolbox is a collection of O-Matrix functions for performing time series and statistics related analysis and visualization. It has capabilities for ARMA and ARFIMA, Bayesian, non-linear and spectral analysis related models. Time series filtering functions and spectral analysis functions are provided. Random number generators are included for both time series, and general statistical analysis. It is a commercial package using the O-Matrix language.
(2) BFL (Bayesian Filtering Library [BFL]): This library provides an application independent framework for inference in Dynamic Bayesian Networks, that is, recursive information processing and estimation algorithms based on Bayes' rule, such as Extended Kalman filters, particle filter, etc. It is written in C++.
(3) KALMTOOL [KALMTOOL]: It is a set of MATLAB tools for state estimation for nonlinear systems. The toolbox contains functions for Extended Kalman filtering as well as for two new filters called the DD1 and DD2 filters. The toolbox specifically addresses the problem of not having observations available at all sampling instants.
(4) Bayes++ Bayesian Filter Classes [BAYES++]: Bayes++ is an open source library of C++ classes. These classes represent and implement a wide variety of numerical algorithms for Bayesian Filtering of discrete systems. The classes provide tested and consistent numerical methods and the class hierarchy explicitly represents the variety of filtering algorithms and system model types.

(5) COSTA [COSTA]: It is an open source project designed to provide a free toolbox for data assimilation for models conforming to the COSTA interface. The current available implementations are: data assimilation methods (ensemble, reduced rank square root, ensemble square root and COFFEE methods) and parameter estimation methods (simplex, conjugate gradient and LBFGS methods). This package is prepared for parallel computing.
(6) PALM [Buis et al. 2003]: This project aims to provide a general structure for a modular implementation of a data assimilation system where the assimilation algorithm is split up into independent units. The package uses C, Fortran and MPI for portability. Today, PALM software is used in an operational way in the French Operational oceanography project MERCATOR.
(7) ESMF (Earth System Modeling Framework [ESMF], Collins et al. [2005]; Hill et al. [2004]): Developed by the team centered at NCAR, ESMF is an open source software for building climate, numerical weather prediction, data assimilation, and other Earth science software applications. Some of the features of this package are: Fortran 90 and (partial) C/C++ interfaces, portability, MPI and OpenMP support, a large set of test, infrastructure and superstructure for coupling and building Earth system components.

Some of the packages cited above have the disadvantage that they are either commercial or programmed in a high-level language. However, this list also presents a number of highly sophisticated systems that provide features such as parallelism, interfaces with other languages, modularity and portability.

The aim of the package presented in this article is to provide parallelism, portability, and modularity, but especially ease of use. The user only needs to code the basic features related to the model and observations (e.g., setting the number of observations, covariance matrices, model propagation, etc). Derived types could have been used but the idea was to keep the programs at the maximum level of simplicity for the user. The parallel version is just an additional module, and the user does not need to worry about communicators, array distribution or parallelization strategies.

This article is structured as follows: In Section 2, a mathematical background is presented along with a brief explanation of the implemented methods. Section 3 is devoted to explaining several versions of the Kalman filter. Section 4 refers to the design of the package, Section 5 shows a sample application and, finally, Section 6 is dedicated to the conclusions.

2. MATHEMATICAL BACKGROUND

Let us define the following entities:

$$n \doteq \text{dimension of the model state (variables we want to assimilate)} \quad (1)$$
$$p \doteq \text{number of observations (usually } p \ll n) \quad (2)$$
$$\mathbf{x^t} \doteq \text{true state, } \mathbf{x^t} \in \mathrm{R}^n \quad (3)$$
$$\mathbf{x^f} \doteq \text{background (or forecast) model state, } \mathbf{x^f} \in \mathrm{R}^n \quad (4)$$
$$\mathbf{x^a} \doteq \text{analysis model state, } \mathbf{x^a} \in \mathrm{R}^n \quad (5)$$
$$\mathbf{y} \doteq \text{vector of observations, } \mathbf{y} \in \mathrm{R}^p \quad (6)$$

$$H \doteq \text{observation operator}, H : \mathrm{R}^n \longrightarrow \mathrm{R}^p \tag{7}$$
$$\mathbf{H} \doteq \text{tangent observation operator}, \mathbf{H} : \mathrm{R}^n \longrightarrow \mathrm{R}^p \tag{8}$$
$$\mathbf{R} \doteq \text{observation error covariance matrix}, \mathbf{R} \in \mathrm{R}^{p\times p} \tag{9}$$
$$\mathbf{P^f} \doteq \text{background (or forecast) error covariance matrix}, \mathbf{P^f} \in \mathrm{R}^{n\times n} \tag{10}$$
$$\mathbf{P^a} \doteq \text{analysis error covariance matrix}, \mathbf{P^a} \in \mathrm{R}^{n\times n} \tag{11}$$
$$\overline{\mathbf{x}} \doteq \text{expected value of } \mathbf{x} \tag{12}$$

The pair $\left(\mathbf{x^f}, \mathbf{P^f}\right)$ defines the previous knowledge (or background) of the system state with an estimation of the error. It can be obtained, for example, via a model. The pair $\left(\mathbf{y}, \mathbf{R}\right)$ provides the observations with an estimation of the observation errors. These can be obtained, for example, from measurement stations. The objective is to generate a new pair $\left(\mathbf{x^a}, \mathbf{P^a}\right)$ where the analysis $\mathbf{x^a}$ is as close as possible to the true state in the root mean square sense.

Assuming that we have

—*nontrivial errors*: $\mathbf{P^f}$ and $\mathbf{R}$ are positive definite matrices,

—*unbiased errors*: the expectation of the background and observation errors are zero, that is, $\overline{\mathbf{x^f} - \mathbf{x^t}} = \overline{\mathbf{y} - H\left(\mathbf{x^t}\right)} = 0$,

—*uncorrelated errors*: observation and background errors are mutually uncorrelated, that is, $\overline{\left(\mathbf{x^f} - \mathbf{x^t}\right)\left(\mathbf{y} - H\left(\mathbf{x^t}\right)\right)^T} = 0$,

then it can be proved (see Bouttier and Courtier [2002]) that the analysis defined by corrections to the background, which depends linearly on background observation departures and has a minimum variance estimate, is

$$\mathbf{x^a} = \mathbf{x^f} + \mathbf{K}\left(\mathbf{y} - H\left(\mathbf{x^f}\right)\right), \tag{13}$$

where $\mathbf{K}$, called the gain matrix, is defined by

$$\mathbf{K} = \mathbf{P^f}\mathbf{H}^T\left(\mathbf{H}\mathbf{P^f}\mathbf{H}^T + \mathbf{R}\right)^{-1}. \tag{14}$$

This is called the BLUE (Best Linear Unbiased Estimator).

The covariance matrix for analysis errors is given (for any $\mathbf{K}$) by

$$\mathbf{P^a} = \left(\mathbf{I} - \mathbf{KH}\right)\mathbf{P^f}\left(\mathbf{I} - \mathbf{KH}\right)^T + \mathbf{KRK}^T, \tag{15}$$

and, if $\mathbf{K}$ is the optimal least-squares gain, the expression is reduced to

$$\mathbf{P^a} = \left(\mathbf{I} - \mathbf{KH}\right)\mathbf{P^f}. \tag{16}$$

In practice, the user does not know $\mathbf{K}$ exactly and the formulation (16) can lead to an erroneous gain matrix, so it is convenient to work with the analysis error covariance matrix defined by (15).

The problem of finding the pair $\left(\mathbf{x^a}, \mathbf{P^a}\right)$ is equivalent to the minimization of the following functional:

$$J\left(\mathbf{x}\right) = \left(\mathbf{x} - \mathbf{x^f}\right)^T \mathbf{P^f}^{-1}\left(\mathbf{x} - \mathbf{x^f}\right) + \left(\mathbf{y} - H\left(\mathbf{x}\right)\right)^T \mathbf{R}^{-1}\left(\mathbf{y} - H\left(\mathbf{x}\right)\right). \tag{17}$$

For the evolution in time of the whole system, we need a model to propagate the state vector and the forecast error covariance matrix or we can apply the

BLUE, and get a better approximation of the true state. Suppose that we have a model $M_{l\rightarrow l+1} : \mathrm{R}^n \longrightarrow \mathrm{R}^n$ that takes the state vector forward from time step l to time step $l+1$, and also assume that the tangent linear model $\mathbf{M}_{l\rightarrow l+1} : \mathrm{R}^n \longrightarrow \mathrm{R}^n$ is available. Assuming the hypotheses for the BLUE, and

—*forecast errors*: The model error $M_{l\rightarrow l+1}\left(\mathbf{x}_l^{\mathbf{t}}\right) - \mathbf{x}_{l+1}^{\mathbf{t}}$ is unbiased with known model error covariance matrix $\mathbf{Q}_l$,

—*uncorrelated analysis and model errors*: The analysis error $\mathbf{x}_l^{\mathbf{a}} - \mathbf{x}_l^{\mathbf{t}}$ and the model error $M_{l\rightarrow l+1}\left(\mathbf{x}_l^{\mathbf{t}}\right) - \mathbf{x}_{l+1}^{\mathbf{t}}$ are mutually uncorrelated,

we can prove (see Bouttier and Courtier [2002]) that the optimal way (in the least square sense) to assimilate the observations sequentially is given by the Kalman filter algorithm

$$\mathbf{x}_{l+1}^{\mathbf{f}} = M_{l\rightarrow l+1}\left(\mathbf{x}_l^{\mathbf{a}}\right), \tag{18}$$

$$\mathbf{P}_{l+1}^{\mathbf{f}} = \mathbf{M}_{l\rightarrow l+1}\mathbf{P}_l^{\mathbf{a}}\mathbf{M}_{l\rightarrow l+1}^T + \mathbf{Q}_l, \tag{19}$$

$$\mathbf{K}_{l+1} = \mathbf{P}_{l+1}^{\mathbf{f}}\mathbf{H}_{l+1}^T\left(\mathbf{H}_{l+1}\mathbf{P}_{l+1}^{\mathbf{f}}\mathbf{H}_{l+1}^T + \mathbf{R}_{l+1}\right)^{-1}, \tag{20}$$

$$\mathbf{x}_{l+1}^{\mathbf{a}} = \mathbf{x}_{l+1}^{\mathbf{f}} + \mathbf{K}_{l+1}\left[\mathbf{y}_{l+1} - H_{l+1}\left(\mathbf{x}_{l+1}^{\mathbf{f}}\right)\right], \tag{21}$$

$$\begin{aligned}\mathbf{P}_{l+1}^{\mathbf{a}} &= \left(\mathbf{I} - \mathbf{K}_{l+1}\mathbf{H}_{l+1}\right)\mathbf{P}_{l+1}^{\mathbf{f}}\left(\mathbf{I} - \mathbf{K}_{l+1}\mathbf{H}_{l+1}\right)^T + \\ &\quad + \mathbf{K}_{l+1}\mathbf{R}_{l+1}\mathbf{K}_{l+1}^T.\end{aligned} \tag{22}$$

Here the subscripts represent time evolution.

Equations (18) and (19) are the prediction part of the filter, while Eqs. (20)–(22) are the correction applied to minimize the variance of the analysis. As stated, the filter can be applied to small and medium problems, but it does not scale well to larger problems because

—*Storage*: For atmospheric applications, we can have $n \approx 10^6$ and a full error covariance matrix of size $n \times n$ may be in excess of a teraword (see the assimilation system at the Data Assimilation Office [Lyster et al. 2003] as an example).

—*Too Many Model Evaluations*: Equation (19) requires $2n$ evaluations of the tangent linear model. In some cases, the tangent version of the model is obtained by an automatic differentiation package, or it is computed using two evaluations of the model. Therefore, the propagation of the forecast error covariance matrix is too expensive.

—*Matrix-Vector Manipulation*: Normally, large scale models represent the system state as 3D matrices, so a transformation matrix→vector is needed to apply one filter step, and then a transformation vector→matrix to continue the propagation in time.

—*Non-linearities*: The extended Kalman filter (18)-(22) linearizes the model, but this linearization has shown to be invalid in a number of applications (see Evensen [1994, 1997]).

The next section explains the main algorithms that are used in the assimilation modules.

3. VERSIONS OF THE KALMAN FILTER ALGORITHM

3.1 EKF (Extended Kalman Filter)

The algorithm (18)–(22) without the time subscripts (for simplicitiy) may be written as

$$\begin{aligned}
\mathbf{x^f} &= M\left(\mathbf{x^a}\right), && (23)\\
\mathbf{P^f} &= \mathbf{MP^aM}^T + \mathbf{Q}, && (24)\\
\mathbf{K} &= \mathbf{P^fH}^T\left(\mathbf{HP^fH}^T + \mathbf{R}\right)^{-1}, && (25)\\
\mathbf{x^a} &= \mathbf{x^f} + \mathbf{K}\left(\mathbf{y} - H\left(\mathbf{x^f}\right)\right), && (26)\\
\mathbf{P^a} &= \left(\mathbf{I} - \mathbf{KH}\right)\mathbf{P^f}\left(\mathbf{I} - \mathbf{KH}\right)^T + \mathbf{KRK}^T. && (27)
\end{aligned}$$

3.2 RRSQRTKF (Reduced Rank Square Root Kalman Filter)

The covariance matrices have good properties being both symmetric and (semi) positive definite. Therefore, they can be factorized and square roots computed (for example, via Cholesky decomposition, or SVD). Whereas in some contexts the square root of a matrix P means $P^{1/2}$, we will say that a matrix S is a square root of a matrix P if $P = SS^T$. Compared with standard Kalman filtering algorithms, square root algorithms are known for their superior numerical properties (see Bierman [1977] and Kaminski et al. [1971]). "They are also more numerically robust than non-square-root forms because they are less susceptible to rounding errors and prevent the error covariance matrices from becoming negative definite" (see Brown and Gaston [1995]). Paige [1985] suggests general representations of covariance matrices in linear filtering in which the covariance and information matrices are implicitly defined. He also develops numerically reliable algorithms (see also Kourouklis [1977]).

Stability problems can be reduced using the square root form of covariance matrices, but there may still be storage and time difficulties. The solution for this is to use a square root covariance matrix with less columns. That is, given a covariance matrix $\mathbf{P}$

$$\mathbf{P} = \mathbf{SS}^T, \quad \mathbf{S} \in \mathrm{R}^{n\times n} \qquad \longrightarrow \qquad \mathbf{P} \approx \mathbf{SS}^T, \quad \mathbf{S} \in \mathrm{R}^{n\times m}, \tag{28}$$

where $m \ll n$ and m is usually referred to as the number of modes. With this formulation, we still have the symmetry and semipositive definiteness of the covariance matrices.

Define

$$\begin{aligned}
m_a &\doteq \text{number of modes of the analysis error covariance matrix} && (29)\\
m_f &\doteq \text{number of modes of the forecast error covariance matrix} && (30)\\
m_q &\doteq \text{number of modes of the model error covariance matrix} && (31)\\
m_r &\doteq \text{number of modes of the observation error covariance matrix,} && (32)\\
\mathbf{S^a} &\doteq \text{square root covariance matrix of analysis errors, } \mathbf{S^a} \in \mathrm{R}^{n\times m_a} && (33)\\
\mathbf{S^f} &\doteq \text{square root covariance matrix of forecast errors, } \mathbf{S^f} \in \mathrm{R}^{n\times m_f} && (34)
\end{aligned}$$

$$\mathbf{S^m} \doteq \text{square root covariance matrix of model errors, } \mathbf{S^m} \in \mathrm{R}^{n\times m_q} \tag{35}$$
$$\mathbf{S^o} \doteq \text{square root covariance matrix of observation errors, } \mathbf{S^o} \in \mathrm{R}^{n\times m_r} \tag{36}$$

Then, covariance matrices in the Kalman filter algorithm (23)–(27) can be transformed in terms of the reduced rank square root as follows

$$\begin{aligned}\mathbf{S^f}\mathbf{S^f}^T &\approx \mathbf{P^f} \approx \mathbf{MS^a}\mathbf{S^a}^T\mathbf{M}^T + \mathbf{S^m}\mathbf{S^m}^T = \left[\mathbf{MS^a} \mid \mathbf{S^m}\right]\left[\mathbf{MS^a} \mid \mathbf{S^m}\right]^T \Longrightarrow \\ \mathbf{S^f} &\approx \left[\mathbf{MS^a} \mid \mathbf{S^m}\right],\end{aligned} \tag{37}$$

$$\begin{aligned}\mathbf{K} &= \mathbf{P^f}\mathbf{H}^T\left(\mathbf{HP^fH}^T + \mathbf{R}\right)^{-1} \approx \mathbf{S^f}\mathbf{S^f}^T\mathbf{H}^T\left(\mathbf{HS^f}\mathbf{S^f}^T\mathbf{H}^T + \mathbf{S^o}\mathbf{S^o}^T\right)^{-1} \Longrightarrow \\ \mathbf{K} &\approx \mathbf{S^f}\left(\mathbf{HS^f}\right)^T\left(\left[\mathbf{HS^f} \mid \mathbf{S^o}\right]\left[\mathbf{HS^f} \mid \mathbf{S^o}\right]^T\right)^{-1},\end{aligned} \tag{38}$$

$$\begin{aligned}\mathbf{S^a}\mathbf{S^a}^T &\approx \left(\mathbf{I} - \mathbf{KH}\right)\mathbf{P^f}\left(\mathbf{I} - \mathbf{KH}\right)^T + \mathbf{KRK}^T \\ &\approx \left(\mathbf{I} - \mathbf{KH}\right)\mathbf{S^f}\mathbf{S^f}^T\left(\mathbf{I} - \mathbf{KH}\right)^T + \mathbf{KS^o}\mathbf{S^o}^T\mathbf{K}^T \\ &= \left[\left(\mathbf{I} - \mathbf{KH}\right)\mathbf{S^f} \mid \mathbf{KS^o}\right]\left[\left(\mathbf{I} - \mathbf{KH}\right)\mathbf{S^f} \mid \mathbf{KS^o}\right]^T \Longrightarrow \\ \mathbf{S^a} &\approx \left[\left(\mathbf{I} - \mathbf{KH}\right)\mathbf{S^f} \mid \mathbf{KS^o}\right]\end{aligned} \tag{39}$$

Note that

(1) the number of forecast modes m_f is $m_a + m_q$,
(2) after the assimilation step, the square root of the analysis error covariance matrix (39) has a larger number of columns, viz $m_a + m_q + m_r$, and a truncation strategy is needed in order to be able to continue with the algorithm (in Treebushny and Madsen [2003] a procedure based on the Lanczos decomposition algorithm is used, in van Loon and Heemink [1997], Segers et al. [2000], and Segers [2002] a procedure based in the SVD is explained, in Hoteit and Pham [2004] a reduced-order extended Kalman filter is proposed).
(3) generally, the number of observations p is much less than the dimension of the state space n. In this case, we should take $m_r = p$ in order to avoid loss of information.

Finally, the RRSQRTKF algorithm is:

$$\mathbf{x^f} = M\left(\mathbf{x^a}\right), \tag{40}$$
$$\mathbf{S^f} = \left[\mathbf{MS^a} \mid \mathbf{S^m}\right], \tag{41}$$
$$\mathbf{K} = \mathbf{S^f}\left(\mathbf{HS^f}\right)^T\left(\left[\mathbf{HS^f} \mid \mathbf{S^o}\right]\left[\mathbf{HS^f} \mid \mathbf{S^o}\right]^T\right)^{-1}, \tag{42}$$
$$\mathbf{x^a} = \mathbf{x^f} + \mathbf{K}\left[\mathbf{y} - H\left(\mathbf{x^f}\right)\right], \tag{43}$$
$$\mathbf{S^a} = \left[\left(\mathbf{I} - \mathbf{KH}\right)\mathbf{S^f} \mid \mathbf{KS^o}\right], \tag{44}$$
$$\mathbf{S^a} \leftarrow \text{reduce } \mathbf{S^a} \text{ to } m_a \text{ columns}, \tag{45}$$

3.3 ENKF (ENsemble Kalman Filter)

The idea of the ensemble Kalman filter is to represent the error statistics using an ensemble of model states. Therefore, instead of forecasting the analysis error covariance matrix using the tangent linear model, the model states are propagated and the covariance matrix is recovered from them. It is very easy to implement, and there is no need to propagate full covariance matrices, but only a few model states (or modes) that contain the information about the system and its statistics. It captures nonlinearities of the model (see Evensen [1994, 2003]) and a tangent linear model is not necessary (and sometimes not available). The only problem is that the error in the Monte Carlo sampling decreases proportionally to $1/\sqrt{N}$ where N is the number of modes. The ENKF has been applied succesfully to several models (in Allen et al. [2002] it is used in the European Regional Seas Ecosystem Model ERSEM, Segers [2002] shows an implementation in the LOTOS model and Annan et al. [2005] use it as an efficient method for parameter estimation and ensemble forecasting in climate modelling is developed).

The ENKF algorithm explained in Evensen [2003] is:

$$\text{Generate (only the first time): } \xi^{\mathbf{a}i} \in \mathcal{N}\left(\mathbf{x^a}, \mathbf{P^a}\right), \quad i = 1:m, \tag{46}$$

$$\text{Propagate: } \xi^{\mathbf{f}i} = M\left(\xi^{\mathbf{a}i}\right) + \eta^i, \quad \eta^i \in \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right), \quad i = 1:m, \tag{47}$$

$$\text{Estimate: } \mathbf{x^f} = \frac{1}{m}\sum_{i=1}^{m} \xi^{\mathbf{f}i}, \tag{48}$$

$$\text{Estimate: } \mathbf{P^f} = \frac{1}{m-1}\sum_{i=1}^{m} \left(\xi^{\mathbf{f}i} - \mathbf{x^f}\right)\left(\xi^{\mathbf{f}i} - \mathbf{x^f}\right)^T, \tag{49}$$

$$\text{Gain matrix: } \mathbf{K} = \mathbf{P^f}\mathbf{H}^T\left(\mathbf{H}\mathbf{P^f}\mathbf{H}^T + \mathbf{R}\right)^{-1}, \tag{50}$$

$$\text{Correct: } \xi^{\mathbf{a}i} = \xi^{\mathbf{f}i} + \mathbf{K}\left(\mathbf{y}^i - H\left(\xi^{\mathbf{f}i}\right)\right), \quad \mathbf{y}^i \in \mathcal{N}\left(\mathbf{y}, \mathbf{R}\right), \quad i = 1:m, \tag{51}$$

$$\text{Estimate: } \mathbf{x^a} = \frac{1}{m}\sum_{i=1}^{m} \xi^{\mathbf{a}i}, \tag{52}$$

$$\text{Estimate: } \mathbf{P^a} = \frac{1}{m-1}\sum_{i=1}^{m} \left(\xi^{\mathbf{a}i} - \mathbf{x^a}\right)\left(\xi^{\mathbf{a}i} - \mathbf{x^a}\right)^T, \tag{53}$$

3.4 RRSQRTENKF (Reduced Rank Square Root ENsemble Kalman Filter)

This is a version of the Ensemble Kalman filter using the square root of covariance matrices. This method is called EnSR (see Whitaker and Hamill [2002] and Tippett et al. [2003]), but for the rest of this article it will be referred to as RRSQRTENKF. The algorithm (46)-(53) becomes

$$\text{Generate (only the first time): } \xi^{\mathbf{a}i} \in \mathcal{N}\left(\mathbf{x^a}, \mathbf{S^a}\mathbf{S^a}^T\right), \quad i = 1:m, \tag{54}$$

$$\text{Propagate: } \xi^{\mathbf{f}i} = M\left(\xi^{\mathbf{a}i}\right) + \eta^i, \quad \eta^i \in \mathcal{N}\left(\mathbf{0}, \mathbf{S^m}\mathbf{S^m}^T\right), \quad i = 1:m, \tag{55}$$

$$\text{Estimate: } \mathbf{x^f} = \frac{1}{m}\sum_{i=1}^{m} \xi^{\mathbf{f}i}, \tag{56}$$

$$\text{Estimate: } \mathbf{S^f} = \frac{1}{\sqrt{m-1}} \begin{pmatrix} \vdots & & \vdots \\ \xi^{\mathbf{f}1} - \mathbf{x^f} & \cdots & \xi^{\mathbf{f}m} - \mathbf{x^f} \\ \vdots & & \vdots \end{pmatrix}, \tag{57}$$

$$\text{Gain matrix: } \mathbf{K} = \mathbf{S^f}\left(\mathbf{HS^f}\right)^T \left(\left[\mathbf{HS^f} \mid \mathbf{S^o}\right]\left[\mathbf{HS^f} \mid \mathbf{S^o}\right]^T\right)^{-1}, \tag{58}$$

$$\text{Correct: } \xi^{\mathbf{a}i} = \xi^{\mathbf{f}i} + \mathbf{K}\left(\mathbf{y}^i - H\left(\xi^{\mathbf{f}i}\right)\right),$$
$$\mathbf{y}^i \in \mathcal{N}\left(\mathbf{y}, \mathbf{S^o S^o}^T\right), \quad i = 1:m, \tag{59}$$

$$\text{Estimate: } \mathbf{x^a} = \frac{1}{m}\sum_{i=1}^{m} \xi^{\mathbf{a}i}, \tag{60}$$

$$\text{Estimate: } \mathbf{S^a} = \frac{1}{\sqrt{m-1}} \begin{pmatrix} \vdots & & \vdots \\ \xi^{\mathbf{a}1} - \mathbf{x^a} & \cdots & \xi^{\mathbf{a}m} - \mathbf{x^a} \\ \vdots & & \vdots \end{pmatrix}, \tag{61}$$

Notice that both ENKF and RRSQRTENKF require the generation of random vectors with a prescribed distribution. This can be done every time step, but it is better to let the ensemble evolve according to the model and the observations. From the numerical experiments performed it is clear that ENKF and RRSQRTENKF require some time before the ensemble members represent the dynamical system well.

We also note that the user should choose the number of columns of the reduced rank square root covariance matrices equal to the ensemble size. If this is not the case, a reduction step, as in (45), needs to be added.

4. DESIGN

4.1 Overview

We have developed a set of Fortran 90 modules that implement the Kalman filter adapted for large scale problems. The methods implemented are the Extended Kalman filter (identified as EKF), the Reduced Rank Square Root filter (identified as RRSQRTKF), the Ensemble Kalman filter (identified as ENKF) and the Reduced Rank Square Root Ensemble filter (identified as RRSQRTENKF). Each method and each necessary assimilation task (for example, model and observations) is coded in modules that can be replaced according to the application.

A list of capabilities is

—*Modularity*: As mentioned before, we have separated all the assimilation tasks in the implementation. This means that observations, model and assimilation are different entities. For example, if the observation stations change location, there is no need to modify the main code, but only a module related to location. If we need to change the model, there is no need to transform all the code, but only the module related to the model. If we want to change the Kalman filter version, a change in the module related to the assimilation will suffice. This allows the user to make minimum changes to the code implementing modelling system (model, data formats, libraries, configuration files) in order to minimize the risk of introducing programming bugs.

—*Simplicity*: There are no derived types defined in the code. It is clear that derived types are a useful language tool, but in this case the intention was to determine the global variables and the specific functions associated with the assimilation. Users are encouraged to introduce new abstract types and restrict the complexity they need to those modules that have to be edited, rather than adjust their code to an existing structure. The software has been written for users with some programming experience as well as expert coders.

—*Language*: The modules are programmed in Fortran 90. The choice of the language was made because many large scale models already exist in either Fortran 77 or Fortran 90. Only standard Fortran 90 ([Adams et al. 1992]) has been used and the code has been succesfully compiled and executed with the Intel Fortran Compiler, Portland Fortran Compiler, GNU Fortran and g95. For parallelization MPI was used (compiled with the Intel Fortran Compiler, Portland Fortran Compiler, GNU Fortran and g95 respectively).

—*Precision*: The modules can use either single and double precision. Some models have their outputs in single precision, others in double precision, so this is a useful feature. The switch between precisions requires a change to a single parameter and recompilation.

—*Parallelism*: Repeated tasks like propagating states (or applying the observation operator, or the tangent observation operator, or the tangent model) are parallelized using the master-slave strategy and MPI (Message Passing Interface). In this case the master sends a set of independent tasks to the slave processors. Once a task is completed, the slave acquires a new task from the master. Linear algebra operations are performed using BLACS (Basic Linear Algebra Communication Subprograms) and SCALAPACK (Scalable LAPACK). The global matrices are distributed in a process grid and operations are performed in each process over local matrices. Finally, the global matrices are rebuilt from local pieces. The parallelism is included in a module and the user just needs to call the parallel routines implemented thereby avoiding the use of explicit communicators and data distribution.

5. SAMPLE APPLICATION

We illustrate the use of our software to solve a problem to assimilate CO concentrations in the area of Santiago de Chile. The user manual that accompanies the software contains a number of other illustrative examples as well as programming details.

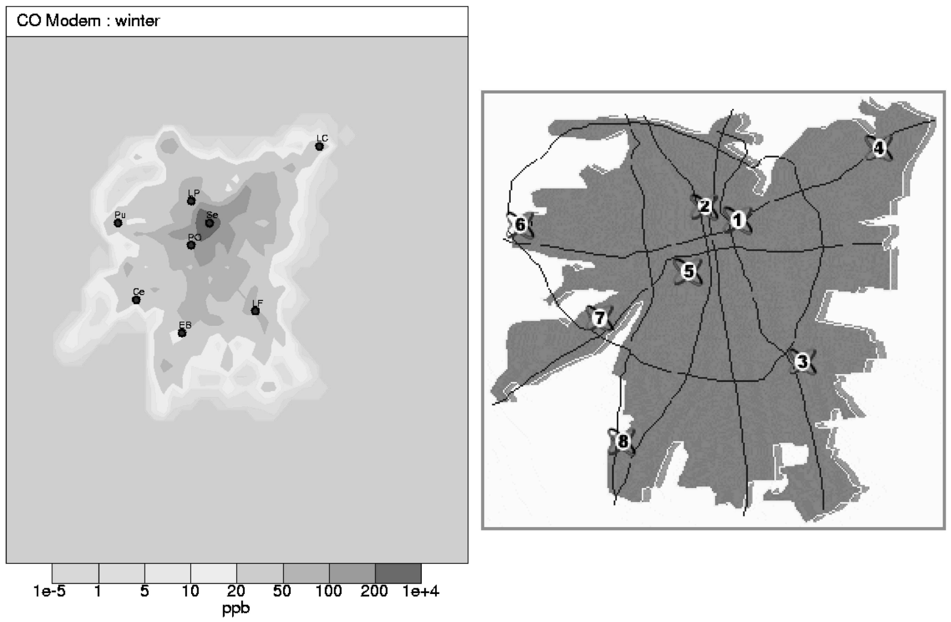

Fig. 1. Emissions generated by MODEM and stations.

5.1 Assimilation of CO

The implementation used the MATCH model [MATCH]. The version of the filter applied was the Reduced Rank Square Root Kalman filter with 50 samples.

A square grid of order 41 is considered in the horizontal domain with 16 levels in the vertical direction leading to a space state of dimension 26896. For the initialization phase, the model was run for a period of three days with an atmosphere free of CO. The initial state vector is set to the last output of the MATCH initial run, adding an error of 100%. The simulation period was 13 days starting at June 17th, 1999, performing an analysis every 3 hours (the time step for observations). The meteorological fields were generated using the HIRLAM model [HIRLAM] with a resolution of 0.01 degrees ($\approx$ 1 km.) and a 1 hour time resolution.

The CO emissions were generated by MODEM [MODEM] (see Figure 1).

There are eight monitoring stations located at different positions in Santiago. The observations are taken from the measuring stations at surface level, at 3 hours intervals, when the analysis step is performed. In Figure 1 we show the eight monitoring stations and the domain of simulation: (1) Seminario, (2) Independencia-Recoleta and (5) Parque O'Higgins are in Santiago city center; (3) La Florida covers the east and south area; (4) Las Condes - Vitacura monitors the north-east sector; (6) Pudahuel-Cerro Navia and (7) Cerrillos register measurements at the west side of the city; and (8) El Bosque is located at the south. The error in the observations was set to 30% of the reported value.

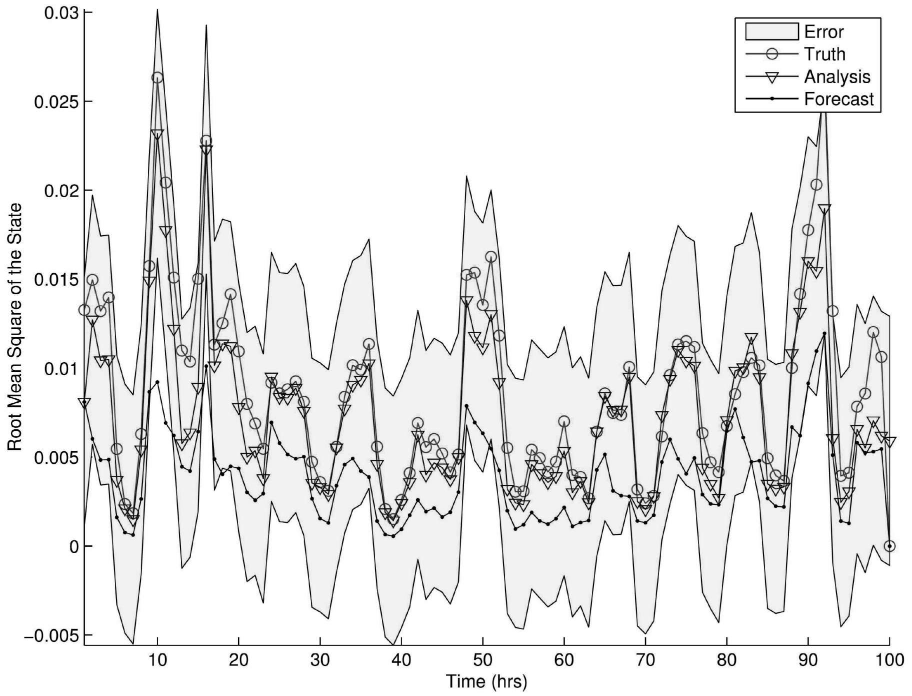

Fig. 2. Results of the CO assimilation.

After 100 hours of the simulation running assimilation, we obtain the results shown in Figure 2, where we also present a comparison of the model, observations, truth and assimilation.

6. CONCLUSIONS

Due to the constant growth of the efficiency and speed of the computers investigations related to modelization have received a new impulse. New models and strategies have been proposed which require increasing computer power. Therefore, new ways of improving forecasts have been implemented, for example, the discrete time Kalman filter. Over the past decades scientists have realized that only improving a model is not enough to get good forecasts, because there are always errors in the data. We can have a perfect model, but if the data is in error we cannot do anything about it. That is why information coming from observations must be used to correct the model. The idea behind this work is to provide a platform where researchers (from oceanography, climatology, etc.) can make use of the Kalman filter in their prediction models in an easy way, changing their source codes as little as possible in order to add assimilation material.

The package is oriented to large scale problems, although some versions of the Kalman filters without simplifications are also implemented. The modules are organized to separate tasks; model, observations and assimilation are considered as different so that each may be changed with only local changes to the code. This provides a cleaner implementation as well as one that minimizes the risk of introducing errors.

Parallelism is implemented using MPI using a master-slave strategy. Linear algebra operations are optimized using the BLAS, LAPACK, PBLAS and SCALAPACK libraries.

The treatment of vectors and matrices takes advantage of Fortran 90 features. This aspect is very important because most of the models in 3D use 3D matrices to represent the state of a domain, but the filter needs the representation of the state as a vector, so an efficient way of transforming between matrices and vectors is required.

The assimilation libraries can be inserted in any model if observations are available. There are versions of the filter that do not require the tangent linear model (which is sometimes not available) but can capture the strong nonlinearities present in many problems, for example, air pollution. These libraries do not deal with setting the covariance matrices because this strongly depends on both the model and the received observations. The most difficult part of the assimilation is how to set the covariance matrices of model and observation errors. The user must know the modeling system in detail in order to obtain an effective assimilation, and may need to improve some parts of the code to gain efficiency for a particular problem.

Numerical tests in 0D, 1D, 2D and 3D have been implemented with satisfactory results in all the versions of the filter. In 0D, solving the ordinary differential equation as in the example provided in the user's manual, one can see how efficient the filter is in reducing the uncertainties. We note that the full Kalman filter produces discontinuities in the assimilated solution, and these occur when an assimilation step is performed. The filters based on Monte Carlo methods (ENKF and RRSQRTENKF) produce smoother solutions. From the experiments we can see that the ensemble needs a period of time to obtain a good representation of the system state. This also can be seen in 0D. For large scale problems it is impossible to apply the full filter. The two 3D problems presented in the examples (see the user's manual) illustrate the use of the package in real applications. In the first case we were able to assimilate CO concentrations in the model MATCH, and in the second case we were able to assimilate O3 concentrations in the POLAIR3D model. In the last case a detailed description is provided on how to implement the model.

We believe that our libraries represent a helpful tool for the modeler, and a good starting point for tuning the variables involved in the filter.

ACKNOWLEDGMENTS

We want to thank to the referees and the editors for their helpful suggestions to improve the submission of this article, as well as to Ing. Karina Buchhorsts and the team at CMM (Center of Mathematical Modeling) in Santiago de Chile for their help in running the assimilation experiments.

REFERENCES

Adams, J., Brainerd, W., and Martin, J. 1992. *Fortran 90 Handbook: Complete Ansi/Iso Reference*. Intertext Publications.

ALLEN, J., EKNES, M., AND EVENSEN, G. 2002. An ensemble Kalman filter with a complex marine ecosystem model: Hindcasting phytoplankton in the cretan sea. *Ann. Geophysi. 20*, 1–13.

ANNAN, J., HARGREAVES, J., EDWARDS, N., AND MARSH, R. 2005. Parameters estimation in an intermediate complexity earth system model using an ensemble Kalman filter. *Ocean Model. 8*, 135–154.

ASIF, A. 2004. Fast implementations of the Kalman-Bucy filter for satellite data assimilation. *IEEE Signal Process. Lett. 11*, 2.

AZUMA, R. AND BISHOP, G. 1994. Improving static and dynamic registration in an optimcal see-through HMD. *Computer Graphics—SIGGRAPH 94 Conference Proceedings*, ACM, New York, 197–204.

BAYES++. http://bayesclasses.sourceforge.net/Bayes++.html.

BFL. http://www.orocos.org/bfl.

BIERMAN, G. 1977. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York.

BOUTTIER, F. AND COURTIER, P. 2002. *Data Assimilation Concepts and Methods*. Meteorological training course lecture series ECMWF.

BROWN, D. AND GASTON, F. 1995. The design of parallel square-root covariance Kalman filters using algorithm engineering. *VLSI Journal 20*, 101–119.

BUIS, S., DECLAT, D., GONDET, E., MASSART, S., MOREL, T., AND THUAL, O. 2003. PALM: A dynamic parallel coupler for data assimilation. *Geophy. Res. Abst. 5*, 05476.

CHARALAMBOUS, C. AND HIBEY, J. 2001. Exact filters for Newton-Raphson parameter estimation algorithms for continuous-time partially observed stochastic systems. *Syst. Cont. Lett. 42*, 101–115.

CHEN, T., MORRIS, J., AND MARTIN, E. 2005. Particle filters for state and parameter estimation in batch processes. *J. Proc. Cont. 15*, 665–673.

CHIN, T., KARL, W., AND WILLSKY, A. 1995. A distributed and iterative method for square root filtering in space-time estimation. *Automatica 31*, 1, 67–82.

COLLINS, N., THEURICH, G., DELUCA, C., SUAREZ, M., TRAYANOV, A., BALAJI, V., LI, P., YANG, W., HILL, C., AND DA SILVA, A. 2005. Design and implementation of components in the earth system modeling framework. *Int. J. High Perf. Comput. Appli. 19*, 341–350.

COSTA. http://www.costapse.org/.

EL SERAFY, G., VAN DER A., R., ESKES, H., AND KELDER, H. 2002. Assimilation of 3D ozone field in global chemistry-transport models using Kalman filter. *Adv. Space Res. 30*, 11, 2473–2478.

ERTÜRK, S. 2002. Real-time digital image stabilization using Kalman filters. *Real-Time Imag. 8*, 317–328.

ESMF. http://www.esmf.ucar.edu/.

EVENSEN, G. 1994. Sequential data assimilation with a non-linear quasi-geostrophic model using Monte Carlo methods to forecast errors statistics. *J. Geophys. Res. 99 C5*, 10143–10162.

EVENSEN, G. 1997. Advanced data assimilation for strongly non-linear dynamics. *Mon. Weather Rev. 125*, 1342–1354.

EVENSEN, G. 2003. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics 53*, 343–367.

FLEMMING, J., REIMER, E., AND STERN, R. 2001. Long term evaluation of the ozone forecast by an Eulerian model. *Phys. Chem. Earth (B) 26*, 10, 775–779.

HANEA, R., VELDERS, G., AND HEEMINK, A. 2005. Comparison of different hybrid Kalman filter algorithms for a large scale chemistry transport model. *Geophys. Res. Abstracts 7*, 00108.

HILL, C., DELUCA, C., BALAJI, V., SUAREZ, M., AND DA SILVA, A. 2004. Architecture of the earth system modeling framework. *Comput. Sci. Eng. 6*, 18–28.

HIRLAM. http://hirlam.org/.

HOTEIT, I. AND PHAM, D. 2004. An adaptively reduced-order extended Kalman filter for data assimilation in the tropical pacific. *J. Marine Syst. 45*, 173–188.

HOTEIT, I., PHAM, D., AND BLUM, J. 2001. A semi-evolutive partially local filter for data assimilation. *Marine Pollut. Bull. 43*, 7-12, 164–174.

HOTEIT, I., PHAM, D., AND BLUM, J. 2002. A simplified reduced order Kalman filtering and application to altimetric data assimilation in tropical pacific. *J. Maryne Syst. 36*, 101–127.

Hoteit, I., Triantafyllou, G., and Petihakis, G. 2004. Towards a data assimilation system for the Cretan Sea ecosystem using a simplified Kalman fsilter. *J. Marine Syst. 45*, 159–171.

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

KALMTOOL. http://www.iau.dtu.dk/research/control/kalmtool.html.

Kaminski, P., Bryson, A., and Schmidt, S. 1971. Discrete square root filtering: A survey of current techniques. *IEEE Trans. Autom. Cont. AC-16*, 727–736.

Kim, S. and Iltis, R. 2002. Performance comparison of particle and extended Kalman filter algorithms for GPS C/A code tracking and interference rejection. In *Proceedings of the Conference on Information Sciences and Systems.* Princeton Univ. Princeton, NJ.

Kourouklis, S. 1977. Implementing a general and numerically stable aproach to the Kalman filtering problem. Master's thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada.

Kuo, C., Chao, C., and Hsieh, C. 2002. An efficient motion estimation algorithm for video coding using Kalman filter. *Real-Time Imag. 8*, 253–264.

Lyster, P., Guo, J., Clune, T., and Larson, J. 2003. The computational complexity and parallel scalability of atmospheric data assimilation algorithms. NASA Data Assimilation Office (DAO), Goddard Laboratory for Atmospheres.

MATCH. http://www.smhi.se/sgn0106/if/FoUl/en/models/match/match.html.

MODEM. http://www.sectra.cl/contenido/metodologia/transporte_medioambiente/estimacion_emisiones_fuentes_moviles_modem.asp.

Paige, C. 1985. Covariance matrix representation in linear filtering. *In "Linear Algebra and Its Role in Systems Theory"*, AMS Publications, Providence RI 47, 309–321.

Pham, D., Verron, J., and Roubaud, M. 1998. A singular evolutive extended Kalman filter for data assimilation in oceanography. *J. Marine Syst. 16*, 3-4, 323–340.

Segers, A. 2002. *Data Assimilation in Atmospheric Chemistry Models Using Kalman Filtering*. Delft University Press.

Segers, A., Heemink, A., Verlaan, M., and van Loon, M. 2000. A modified RRSQRT-filter for assimilating data in atmospheric chemistry models. *Environmental Modelling & Software 15*, 663–671.

STSA. http://www.omatrix.com/stsa.html.

Tippett, M. K., Anderson, J. L., Bishop, C. H., Hamill, C. H., and Whitaker, J. S. 2003. Ensemble square root filters. *Mon. Weather Rev. 131*, 1485–1490.

Treebushny, D. and Madsen, H. 2003. A new reduced rank square root Kalman filter technique for data assimilation in large scale modelling systems. *Geophys. Res. Abs. 5*, 12800.

Treebushny, D. and Madsen, H. 2005. On the construction of a reduced rank square-root Kalman filter for efficient uncertainty propagation. *Future Gen. Comput. Syst. 21*, 1047–1055.

van Loon, M., Builtjes, P., and Segers, A. 2000. Data assimilation of ozone in the atmospheric transport chemistry model LOTOS. *Environ. Model. Softw. 15*, 603–609.

van Loon, M. and Heemink, A. 1997. Kalman filtering for non linear atmospheric chemistry models: First experiences. Tech. Rep. MAS-R9711, CWI, Amsterdam, The Netherlands.

Welch, G. and Bishop, G. 2001. *An Introduction to the Kalman Filter*. SIGGRAPH 2001 - Course 8, ACM, New York.

Whitaker, J. S. and Hamill, T. M. 2002. Ensemble data assimilation without perturbed observations. *Mon. Weather Rev. 130*, 1913–1924.

Zhang, X., Heemink, A., and Eijkeren, J. 1997. Data assimilation in transport models. *Appl. Math. Model. 21*, 1, 2–14.

Zhang, X., Heemink, A., Janssen, L., Janssen, P., and Sauter, F. 1999. A computationally efficient Kalman smoother for the evaluation of the CH_4 budget in Europe. *Appl. Math. Model. 23*, 109–129.

Received December 2006; revised August 2008; accepted April 2009