INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **15**:1 *Prepared using* fldauth.cls *[Version: 2002/09/18 v1.01]*

The Variational Kalman filter and an efficient implementation using limited memory BFGS

H. Auvinen[1,*], J. M. Bardsley[2], H. Haario[1] and T. Kauranne[1]

[1] *Department of Mathematics and Physics*
Lappeenranta University of Technology
Lappeenranta, Finland
[2] *Department of Mathematical Sciences*
University of Montana
Missoula, Montana 59812, USA

SUMMARY

The standard formulations of the Kalman filter (KF) and extended Kalman filter (EKF) require storing and multiplication of matrices of size $n \times n$, where n is the size of the state space, and the inversion of matrices of size $m \times m$, where m is the size of the observation space. For large dimensions implementation issues arise. In this paper we introduce a Variational Kalman Filter (VKF) method to provide a low storage approximation of KF/EKF methods. In stead of using the KF formulae, we solve the underlying maximum a posteriori optimization problem using the limited memory BFGS (LBFGS) method. Moreover, the LBFGS optimization method is used to obtain a low storage approximation of state estimate covariances and prediction error covariances. A detailed description of the VKF method with LBFGS is given. The methodology is tested on linear and nonlinear test examples. Our simulations indicate that the approach yields results that are comparable with those obtained using KF and EKF, respectively, and can be used on much larger scale problems.

KEY WORDS: Kalman filter, Bayesian inversion, large-scale optimization

1. Introduction

Several variants of the Kalman filter (KF) and the extended Kalman filter (EKF) have been proposed to reduce the computational computational complexity for large dimensional problems. The Reduced Rank Kalman Filter or Reduced Order extended Kalman filter (see, e.g., [14], [4], [16]) project the dynamical state vector of the model onto a lower dimensional subspace. The success of the approach depends on a judicious choice of the reduction operator. Moreover, since the reduction operator is typically fixed in time, the dynamics of the system may not be correctly captured; see [6] for more details.

In [1], we showed how high dimensional KF and EKF may be approximatively carried out using the limited memory BFGS (LBFGS) optimization algorithm.

*Correspondence to: H. Auvinen, Department of Mathematics and Physics, Lappeenranta University of Technology, email: harri.auvinen@lut.fi

The resulting methods were effective and exhibited low storage and computational cost characteristics. In this paper, we introduce an alternative approximation, the Variational Kalman Filter (VKF), for KF and EKF. In the variational approach, we solve an equivalent maximum a posteriori optimization problem using LBFGS, which replaces the explicit computation and use of the Kalman gain matrix, in order to obtain state estimates and covariance approximations. Further, the LBFGS method is used for matrix inversion, in order to propagate the state estimate covariance information in time.

The paper is organized as follows. We introduce the notations used in this paper in Section 2, with discussion of the Kalman filter. In Section 3, we present our methods to approximate the Kalman Filter and Fixed lag Kalman smoother. To test and compare these methods, we present results from a number of numerical experiments in Section 4, and we end with conclusions in Section 5.

2. The Kalman Filter

The Kalman filter is given as the coupled system of discrete, linear, stochastic difference equations

$$\mathbf{x}_k = \mathbf{M}_k \mathbf{x}_{k-1} + \boldsymbol{\varepsilon}_k^p, \tag{1}$$

$$\mathbf{y}_k = \mathbf{K}_k \mathbf{x}_k + \boldsymbol{\varepsilon}_k^o, \tag{2}$$

In the first equation, $\mathbf{x}_k$ denotes the $n \times 1$ state of the system at time k; $\mathbf{M}_k$ is the $n \times n$ linear evolution operator; and $\boldsymbol{\varepsilon}_k^p$ is a $n \times 1$ random vector known as the prediction error and is assumed to characterize errors in the model and corresponding numerical approximations. In the second equation, $\mathbf{y}_k$ denotes the $m \times 1$ observed data; $\mathbf{K}_k$ is the $m \times n$ linear observation operator; and $\boldsymbol{\varepsilon}_k^o$ is an $m \times 1$ random vector known as the observation error. The error terms are supposed to be independent and Normally distributed, with zero means and with covariance matrixes $\mathbf{C}_{\boldsymbol{\varepsilon}_k^p}$ and $\mathbf{C}_{\boldsymbol{\varepsilon}_k^o}$, respectively.

The literature contains various derivations of the Kalman filter. Here, we recall a derivation of the filter directly from Bayes' Law. We do this in order to fix the notations and to emphasize the connection between the filter and the minimization of a least squares problem with quadratic penalty. The Bayes' formula is given as

$$p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}) = \frac{p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x}) p_{\mathbf{x}}(\mathbf{x})}{p_{\mathbf{y}}(\mathbf{y})}, \tag{3}$$

where $\mathbf{x}$ is the vector of unknowns, $\mathbf{y}$ the measurements, $p_{\mathbf{x}}$ denotes the prior density, and $p_{\mathbf{y}|\mathbf{x}}$ is the density of the likelihood function. The maximum a posterior (MAP) estimate is obtained by maximizing (3). Equivalently, one can minimize

$$\ell(\mathbf{x}|\mathbf{y}) := -\log p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x}) - \log p_{\mathbf{x}}(\mathbf{x}). \tag{4}$$

For the linear model (2) at time k, the function ℓ assumes the form

$$\ell(\mathbf{x}|\mathbf{y}_k) = \frac{1}{2}(\mathbf{y}_k - \mathbf{K}_k\mathbf{x})^{\mathrm{T}} \mathbf{C}_{\boldsymbol{\varepsilon}_k^o}^{-1} (\mathbf{y}_k - \mathbf{K}_k\mathbf{x}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k^p)^{\mathrm{T}} (\mathbf{C}_k^p)^{-1} (\mathbf{x} - \mathbf{x}_k^p), \tag{5}$$

where $\mathbf{C}_{\boldsymbol{\varepsilon}_k^o}$ and $\mathbf{C}_k^p$ are the covariance matrices of the measurement noise $\boldsymbol{\varepsilon}_k^o$ and of the prior $\mathbf{x}_k^p$, respectively. The minimizer $\mathbf{x}_k^{est}$ of the above expression together with

its error covariance $\mathbf{C}_k^{est}$ (the inverse Hessian matrix of ℓ) can be written as

$$\begin{aligned} \mathbf{x}_k^{est} &= \mathbf{x}_k^p + \mathbf{G}_k(\mathbf{y}_k - \mathbf{K}_k\mathbf{x}_k^p), & (6)\\ \mathbf{C}_k^{est} &= \mathbf{C}_k^p - \mathbf{G}_k\mathbf{K}_k\mathbf{C}_k^p, & (7) \end{aligned}$$

where

$$\mathbf{G}_k = \mathbf{C}_k^p\mathbf{K}_k^{\mathrm{T}}(\mathbf{K}_k\mathbf{C}_k^p\mathbf{K}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^o})^{-1}. \quad (8)$$

The above formulae (6),(7),(8) for the MAP estimate of (3) form an essential part of the usual formulation of the Kalman filter. To complete, we only need to specify the prior terms $\mathbf{x}_k^p$, $\mathbf{C}_k^p$. In Bayesian terms, equation (2) provides the likelihood function, while (1) gives the prior. By assuming that process noise $\boldsymbol{\varepsilon}_k^p$ and the state estimate $\mathbf{x}_{k-1}^{est}$ are independent we can derive the covariance of the prior $\mathbf{C}_k^p$ from (1):

$$\mathbf{C}_k^p = \mathrm{cov}(\mathbf{M}_k\mathbf{x}_{k-1}^{est} + \boldsymbol{\varepsilon}_k^p) = \mathbf{M}_k\mathrm{cov}(\mathbf{x}_{k-1}^{est})\mathbf{M}_k^{\mathrm{T}} + \mathrm{cov}(\boldsymbol{\varepsilon}_k^p) \quad (9)$$

Thus we get the prior and the corresponding prior covariance matrix as

$$\begin{aligned} \mathbf{x}_k^p &= \mathbf{M}_k\mathbf{x}_{k-1}^{est}, & (10)\\ \mathbf{C}_k^p &= \mathbf{M}_k\mathbf{C}_{k-1}^{est}\mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p}. & (11) \end{aligned}$$

The formulae (6),(7),(8) combined with the expressions (10), (11) for the prior now comprise the Kalman filter algorithm.

The Kalman Filter algorithm

Step 0: Select initial guess $\mathbf{x}_0^{est}$ and covariance $\mathbf{C}_0^{est}$, and set $k = 1$.
Step 1: Compute the evolution model estimate and covariance:
(i) Compute $\mathbf{x}_k^p = \mathbf{M}_k\mathbf{x}_{k-1}^{est}$;
(ii) Compute $\mathbf{C}_k^p = \mathbf{M}_k\mathbf{C}_{k-1}^{est}\mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p}$.
Step 2: Compute Kalman filter estimate and covariance:
(i) Compute the Kalman Gain $\mathbf{G}_k = \mathbf{C}_k^p\mathbf{K}_k^{\mathrm{T}}(\mathbf{K}_k\mathbf{C}_k^p\mathbf{K}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^o})^{-1}$;
(ii) Compute the Kalman filter estimate $\mathbf{x}_k^{est} = \mathbf{x}_k^p + \mathbf{G}_k(\mathbf{y}_k - \mathbf{K}_k\mathbf{x}_k^p)$;
(iii) Compute the estimate covariance $\mathbf{C}_k^{est} = \mathbf{C}_k^p - \mathbf{G}_k\mathbf{K}_k\mathbf{C}_k^p$.
Step 3: Update $k := k + 1$ and return to Step 1.

A nonlinear extension of KF is obtained when (1), (2) are replaced by

$$\begin{aligned} \mathbf{x}_k &= \mathcal{M}(\mathbf{x}_{k-1}) + \boldsymbol{\varepsilon}_k^p, & (12)\\ \mathbf{y}_k &= \mathcal{K}(\mathbf{x}_k) + \boldsymbol{\varepsilon}_k^o, & (13) \end{aligned}$$

where $\mathcal{M}$ and $\mathcal{K}$ are possibly nonlinear functions. EKF is obtained by the following modification of the KF algorithm: in Step 1, (i) use the nonlinear model $\mathbf{x}_k^p = \mathcal{M}(\mathbf{x}_k^{est})$ to compute the prior, but employ the linearized approximations,

$$\mathbf{M}_k = \frac{\partial\mathcal{M}(\mathbf{x}_{k-1}^{est})}{\partial\mathbf{x}}, \quad \text{and} \quad \mathbf{K}_k = \frac{\partial\mathcal{K}(\mathbf{x}_k^p)}{\partial\mathbf{x}}, \quad (14)$$

for the covariance calculations, and otherwise employ the same formulae as above.

We note that $\mathbf{M}_k$ and $\mathbf{K}_k$ can be computed or estimated in a number of ways. For example, the numerical scheme that is used in the solution of either the evolution or the observation model defines a tangent linear code (see, e.g., [8]), which can be used to compute (14). A common, but also more computationally expensive, approach is to use finite differences to approximate (14).

3. The Variational Kalman filter method

In a general assimilation problem, where n is the size of state space and m is the size of the observation space, the standard formulation of the Kalman filter (KF) and extended Kalman filter (EKF) require the storage and multiplication of $n \times n$ matrices and the inversion of $m \times m$ matrices. Thus the computational cost of KF/EKF increases rapidly as the size of the problem becomes large. Due to this fact there are several interesting assimilation problems, where the standard formulation of KF or EKF is impractical to implement.

The idea in the variational approach is to provide an approximation of KF and EKF using the limited memory BFGS method. In particular, we directly minimize the expression in (4) in order to avoid the computational and storage issues involved with the use of the Kalman filter matrix formulae.

We will make the reasonable assumption that multiplication by the evolution and observation matrices $\mathbf{M}_k$ and $\mathbf{K}_k$ and by the covariance matrices $\mathbf{C}_{\boldsymbol{\varepsilon}_k^p}$ and $\mathbf{C}_{\boldsymbol{\varepsilon}_k^o}$ is efficient, both in terms of storage and CPU time. Additional computational challenges arise for sufficiently large n due to the storage requirements for $\mathbf{C}_k^{est}$, which becomes a full matrix as the iterations proceed. The same is also true for $\mathbf{C}_k^p$, however, given that

$$\mathbf{C}_k^p = \mathbf{M}_k \mathbf{C}_{k-1}^{est} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p}, \tag{15}$$

storage issues are restricted to those for $\mathbf{C}_k^{est}$.

These difficulties can be efficiently handled by using the LBFGS method. In particular, consider the quadratic function

$$q(\mathbf{u}) = \frac{1}{2}\langle \mathbf{A}\mathbf{u}, \mathbf{u}\rangle - \langle \mathbf{b}, \mathbf{u}\rangle, \tag{16}$$

where $\mathbf{A}$ is an $n \times n$ symmetric positive definite matrix and $\mathbf{b}$ is an $n \times 1$ vector. The LBFGS method applied to the problem of minimizing q with respect to $\mathbf{u}$ can be used to obtain both an estimate of the minimizer of q, as well as a low storage estimate of $\mathbf{A}$ and $\mathbf{A}^{-1}$, both of which will be employed in our approach, (see the Appendix for the details of the LBFGS algorithm for a quadratic cost function, as well as the limited memory formulas for $\mathbf{A}$ and $\mathbf{A}^{-1}$).

Let us first introduce the Variational Kalman filter (VKF) method in case of a linear evolution model $\mathbf{M}$ and observation model $\mathbf{K}$. In this case, we iteratively minimize (5) by the LFBGS algorithm. As a result of the optimization, we obtain both an estimate of $\mathbf{x}_k^{est}$, as well as the corresponding limited memory approximation of state estimate covariance $\mathbf{C}_k^{est}$.

A central question is the next step: how to propagate the covariance of the state estimate, $\mathbf{C}_{k-1}^{est}$, so that it well-approximates the prior matrix $\mathbf{C}_k^p$ in (5) for the next time point k. In the above manner, we use LBFGS to approximate

$$(\mathbf{C}_k^p)^{-1} = (\mathbf{M}_k \mathbf{C}_{k-1}^{est} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p})^{-1}. \tag{17}$$

During this task we use the previously computed LBFGS estimate of $\mathbf{C}_{k-1}^{est}$.

We summarize the linear Variational Kalman filter algorithm as follows:

The linear Variational Kalman Filter algorithm

Step 0: Select initial guess $\mathbf{x}_0^{est}$ and covariance $\mathbf{C}_0^{est}$, and set $k = 1$.
Step 1: Compute the evolution model estimate and covariance:

(i) Compute $\mathbf{x}_k^p = \mathbf{M}_k \mathbf{x}_{k-1}^{est}$;
(ii) Approximate $(\mathbf{C}_k^p)^{-1} = (\mathbf{M}_k \mathbf{C}_{k-1}^{est} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p})^{-1}$ by using LBFGS method;

Step 2: Compute Variational Kalman filter and covariance estimates:
(i) Minimize $\ell(\mathbf{x}|\mathbf{y}_k) = (\mathbf{y}_k - \mathbf{K}_k\mathbf{x})^{\mathrm{T}}(\mathbf{C}_{\boldsymbol{\varepsilon}_k^o})^{-1}(\mathbf{y}_k - \mathbf{K}_k\mathbf{x}) + (\mathbf{x} - \mathbf{x}_k^p)^{\mathrm{T}}(\mathbf{C}_k^p)^{-1}(\mathbf{x} - \mathbf{x}_k^p)$ by using LBFGS method;
(ii) Store the result of the minimization as an estimate $\mathbf{x}_k^{est}$;
(iii) Store the limited memory approximation of $\mathbf{C}_k^{est}$;

Step 3: Update $k := k + 1$ and return to Step 1.

Note that in Step 1, (ii) and Step 2, (i) the optimizations are quadratic and therefore only quadratic LBFGS is needed, see appendix for details. For practical applications, a scaling of the initial inverse Hessian is needed in order to obtain accurate results. This can be handled by choosing, e.g., $\mathbf{B}_0^{-1} = \beta\mathbf{I}$. In the numerical examples in section 4, the constant β is chosen so that $\beta\mathbf{I}$ well-approximates the diagonal of the covariance matrix of interest.

3.0.1. The nonlinear Variational Kalman filter method As in the case of EKF, we need a linearization to propagate the covariance information from one observation time to the next. However, the direct linearization as in EKF is impractical for large dimensions. Rather, in the case of a non-linear evolution model we should use the adjoint operator, if available, in (17). Furthermore, if the adjoint operator is coded in an implicit form, *i.e.* in the software of the model, we get full benefits from a limited memory presentation of $\mathbf{C}^{est}$. This, indeed, is the situation in many operational codes for weather forecasting.

Supposing that the linearization $\mathbf{M}_k$ of $\mathcal{M}_k$ is available, we can summarize the nonlinear VKF method in the following algorithm:

The nonlinear Variational Kalman Filter algorithm

Step 0: Select initial guess $\mathbf{x}_0^{est}$ and covariance $\mathbf{C}_0^{est}$, and set $k = 1$.

Step 1: Compute the evolution model estimate and covariance:
(i) Compute $\mathbf{x}_k^p = \mathcal{M}_k(\mathbf{x}_{k-1}^{est})$;
(ii) Approximate $(\mathbf{C}_k^p)^{-1} = (\mathbf{M}_k \mathbf{C}_{k-1}^{est} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p})^{-1}$ by using LBFGS method;

Step 2: Compute Variational Kalman filter estimate and covariance:
(i) Minimize $\ell(\mathbf{x}|\mathbf{y}) = (\mathbf{y} - \mathcal{K}_k(\mathbf{x}))^{\mathrm{T}}(\mathbf{C}_{\boldsymbol{\varepsilon}_k^o})^{-1}(\mathbf{y} - \mathcal{K}_k(\mathbf{x})) + (\mathbf{x} - \mathbf{x}_k^p)^{\mathrm{T}}(\mathbf{C}_k^p)^{-1}(\mathbf{x} - \mathbf{x}_k^p)$ by using LBFGS method;
(ii) Store the result of the minimization as an estimate $\mathbf{x}_k^{est}$;
(iii) Store the limited memory approximation of $\mathbf{C}_k^{est}$;

Step 3: Update $k := k + 1$ and return to Step 1.

Especially, if the *tangent linear* $\mathbf{M}_k^{tl}$ and corresponding *adjoint code* $\mathbf{M}_k^*$ [8] are available for the evolution model $\mathcal{M}$, Step 1, (ii) can be written as:

Step 1: Compute the evolution model estimate and covariance:
(ii) Minimize $(\mathbf{C}_k^p)^{-1} = (\mathbf{M}_k^{tl} \mathbf{C}_{k-1}^{est} \mathbf{M}_k^* + \mathbf{C}_{\boldsymbol{\varepsilon}_k^p})^{-1}$ by using LBFGS method;

This feature of the method is one of the major advantages compared to EKF, since the time consuming linearization of the evolution model can be avoided. The traditional linearization requires n evolution model function calls, but inside VKF the required number of tangent linear and adjoint code evaluations is around 15-40, which is far less than n in large scale problem.

Similar possibilities exist for use the tangent linear code $\mathbf{K}_k^{tl}$ of the observation model $\mathcal{K}$ in Step 2, (i):

Step 2: Compute the evolution model estimate and covariance:
(i) Minimize $\ell(\mathbf{x}|\mathbf{y}) = (\mathbf{y}-\mathbf{K}_k^{tl}\mathbf{x})^{\mathrm{T}}(\mathbf{C}_{\boldsymbol{\varepsilon}_k^o})^{-1}(\mathbf{y}-\mathbf{K}_k^{tl}\mathbf{x})+(\mathbf{x}-\mathbf{x}_k^p)^{\mathrm{T}}(\mathbf{C}_k^p)^{-1}(\mathbf{x}-\mathbf{x}_k^p)$;

Note that the tangent linear code $\mathbf{K}_k^{tl}$ is impractical for the EKF, since in the equation (8) $\mathbf{K}_k$ operates for full matrice and therefore the computational advantage of using $\mathbf{K}_k^{tl}$ instead is lost. We also note that in Step 2 a quadratic LBFGS algorithm can be used.

3.1. The Variational Kalman Smoother method

Next we introduce a Variational Kalman smoother (VKS) method, which can be used afterwards to smooth the results of the Variational Kalman filter. The idea is to simulate a fixed-lag Kalman smoother (FLKS) method and take full benefit from the limited memory covariance approximation from of the VKF method. In general, the post-processing improves the quality of the VKF results.

The VKF method provides an estimate $\mathbf{x}_k^{est}$ and a corresponding limited memory approximation of the covariance matrix $\mathbf{C}_k^{est}$ after each time step k. In VKS, we use these results from the previous $[k_0, k_0{+}1, ..., k]$ time steps, where the parameter $k_0 = k-lag$ determines the length of the time interval. In case of linear evolution model $\mathbf{M}_k$, we couple the results together by using the following cost function:

$$J(\mathbf{x}_{k_0}) = \sum_{t=k_0}^{k} (\mathbf{M}_t\mathbf{x}_{k_0} - \mathbf{x}_t^{est})^{\mathrm{T}}(\mathbf{C}_t^{est})^{-1}(\mathbf{M}_t\mathbf{x}_{k_0} - \mathbf{x}_t^{est}), \tag{18}$$

where $\mathbf{M}_t(\mathbf{x}_{k_0})$ is a model trajectory from $\mathbf{x}_{k_0}$. The minimization of the cost function is done by using the 4d-Var method, see [5], [8].

In the nonlinear case, the evolution model $\mathcal{M}_t$ is used instead of $\mathbf{M}_t$ in the cost function formulation (18). Furthermore the gradient of (18) can be computed efficiently by using the adjoint of the evolution model, but in principle, the linearization of $\mathcal{M}_t$ can be used again as well. Since the smoothing process improves the accuracy of the estimate at time k_0, it is possible to outperform EKF in retrospective analysis.

During VKF iterations, the inverse Hessian limited memory BFGS formula is used to represent $\mathbf{C}_k^{est}$. In the VKS cost function (18) we instead need the inverse of $\mathbf{C}_t^{est}$. In practice this detail is handled by using the direct Hessian limited memory BFGS formula, see, e.g., [12] and the appendix. The direct Hessian limited memory BFGS formula provides needed $(\mathbf{C}_t^{est})^{-1}$, but is not recursive.

4. Numerical Experiments

In this section, we present numerical results that justify the approach set forth in the previous section, which we will call the Variational Kalman filter (VKF). We apply the method to the same examples as used in [1]. The first is sufficiently large-scale that the use of the VKF is justified. In particular, we assume the following forced heat equation evolution model

$$\frac{\partial x}{\partial t} = -\frac{\partial^2 x}{\partial u^2} - \frac{\partial^2 x}{\partial v^2} + \alpha \exp\left[-\frac{(u-2/9)^2 + (v-2/9)^2}{\sigma^2}\right], \tag{19}$$

where x is a function of u and v over the domain $\Omega = \{(u, v) \mid 0 \leq u, v \leq 1\}$ and $\alpha \geq 0$. In our experiment, we will generate synthetic data using (19) with $\alpha > 0$ and assume that the evolution model is given by (19) with $\alpha = 0$, which gives a model bias. The problem can be made as large-scale as one wants via the choice of a sufficiently fine discretization of the domain Ω. However, the well-behaved nature of solutions of (19), in particular the fact that its solution tend to a steady state, makes further experiments with a different test case a necessity.

For this reason, we also test our method on a second example, which produces chaotic solutions, and hence has unpredictable behaviour. In particular, we consider the simple non-linear Lorenz'95 model introduced and analyzed in [10, 11] and which is given by

$$\frac{\partial x^i}{\partial t} = (x^{i+1} - x^{i-2})x^{i-1} - x^i + 8, \qquad i = 1, 2, \ldots, 40, \tag{20}$$

with periodic state space variables, i.e. $x^{-1} = x^{n-1}$, $x^0 = x^n$ and $x^{n+1} = x^1$, $n = 40$. Then (20) is a chaotic dynamical system (cf. [11]), which is desirable for testing purposes. As the model is computationally light and shares many characteristics with realistic atmospheric models (cf. [11]), it is commonly used for testing different data analysis schemes for weather forecasting.

4.1. An Example with a Large-Scale Linear Evolution Model

We perform our first experiments using model (19) using a uniform $n \times n$ computational grid and the standard finite difference discretization of both the time and spatial derivatives, which yields the following time stepping equation $\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{f}$, where $\mathbf{M} = \mathbf{I} - \Delta t\mathbf{L}$. Here $\mathbf{L}$ is given by the standard finite difference discretization of the two-dimensional Laplacian operator with homogeneous Dirichlet boundary conditions, Δt is chosen to guarantee stability, and $\mathbf{f}$ is the constant vector determined by the evaluation of the forcing term in (19) at each of the points of the computational grid. We define $\mathbf{K}_k = \mathbf{K}$ for all k in (2), where $\mathbf{K}$ is the full weighting matrix, which has the following grid representation

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Such an observation matrix could model, for example, an array of square heat sensors on the bottom of a metal plate that have dimension $2/n \times 2/n$ with the edges aligned with the grid lines.

In our first test, we generate synthetic data using the linear stochastic equations

$$\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{f} + N(\mathbf{0}, (0.5\sigma_{\text{ev}})^2\mathbf{I}), \tag{21}$$

$$\mathbf{y}_{k+1} = \mathbf{K}\mathbf{x}_{k+1} + N(\mathbf{0}, (0.8\sigma_{\text{obs}})^2\mathbf{I}), \tag{22}$$

with $\alpha = 3/4$ in (19) and where σ^2_{ev} and σ^2_{obs} are chosen so that the signal to noise ratios, defined by $\|\mathbf{x}_0\|^2/n^2\sigma^2_{\text{ev}}$ and $\|\mathbf{K}\mathbf{x}_0\|^2/n^2\sigma^2_{\text{obs}}$ respectively, are both 50. The initial condition used for the data generation was

$$[\mathbf{x}_0]_{ij} = \exp[-((u_i - 1/2)^2 + (v_j - 1/2)^2)],$$

where (u_i, v_j) is the ijth grid point.

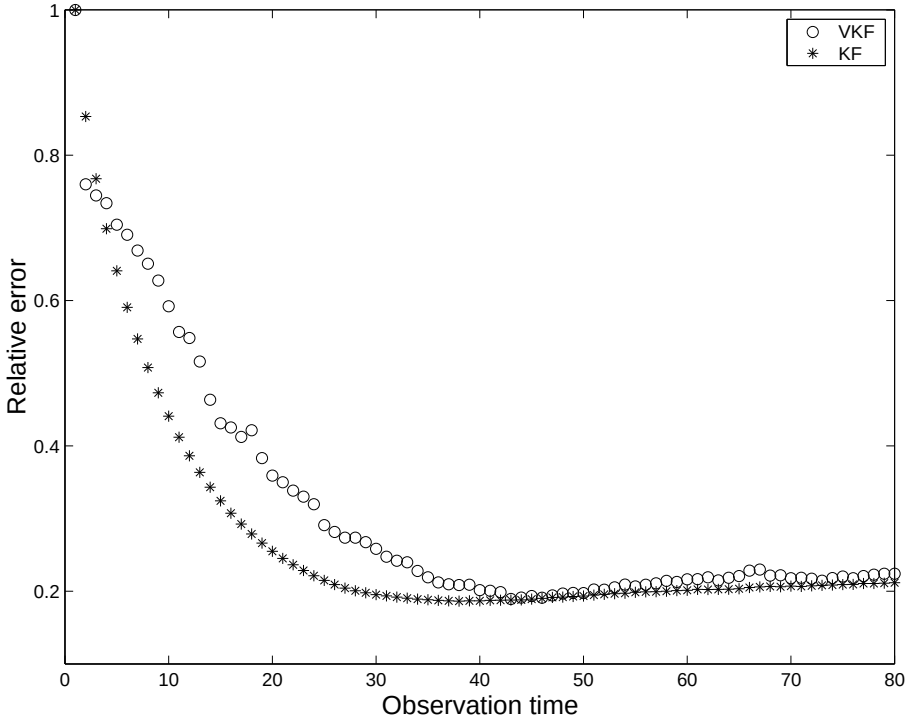

Figure 1. Relative error curves for KF ($*$) and VKF (o). The horizontal axis represents the time of the observations.

In our implementation of KF, we used the biased models

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{M}\mathbf{x}_k + N(\mathbf{0}, \sigma_{\text{ev}}^2\mathbf{I}), \\ \mathbf{y}_{k+1} &= \mathbf{K}\mathbf{x}_{k+1} + N(\mathbf{0}, \sigma_{\text{obs}}^2\mathbf{I}), \end{aligned}$$

with initial conditions $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{C}_0^{\text{est}} = 0.001\mathbf{I}$ in Step 0 of the filter. We compare the results obtained with VKF and KF, where $n = 2^j$ with j taken to be the largest positive integer so that memory issues do not arise in the MATLAB implementation for the standard KF. For the computer on which the simulations were done (a laptop with 2G RAM memory and a 2.13 GHz processor) the largest such j was 5, making $n = 32$. We note that in our implementation of the LBFGS method, we have chosen to take only 10 LBFGS iterations with 9 saved vectors. The purpose of this test is to show that the results obtained with VKF are comparable to those obtained with KF. To do this, we present a plot in Figure 1 of the relative error vector, which has kth component

$$[\mathbf{relative\_error}]_k := \frac{\|\mathbf{x}_k^{\text{est}} - \mathbf{x}_k\|}{\|\mathbf{x}_k\|},$$

for both the Variational Kalman Filter and for the standard Kalman Filter. We see that results obtained using the two approaches yield similar, though not identical, relative error curves. Inside the VKF method, the initial inverse Hessian parameters during approximation of $\mathbf{C}_k^{est}$ and $(\mathbf{C}_k^p)^{-1}$ were $\mathbf{B}_0^{-1} = \mathbf{I}$ and $\mathbf{B}_0^{-1} = 4000\mathbf{I}$, respectively for all k. At the begin of the filtering period KF provides more accurate results, but later the difference decreaces. Both curves eventually begin to increase

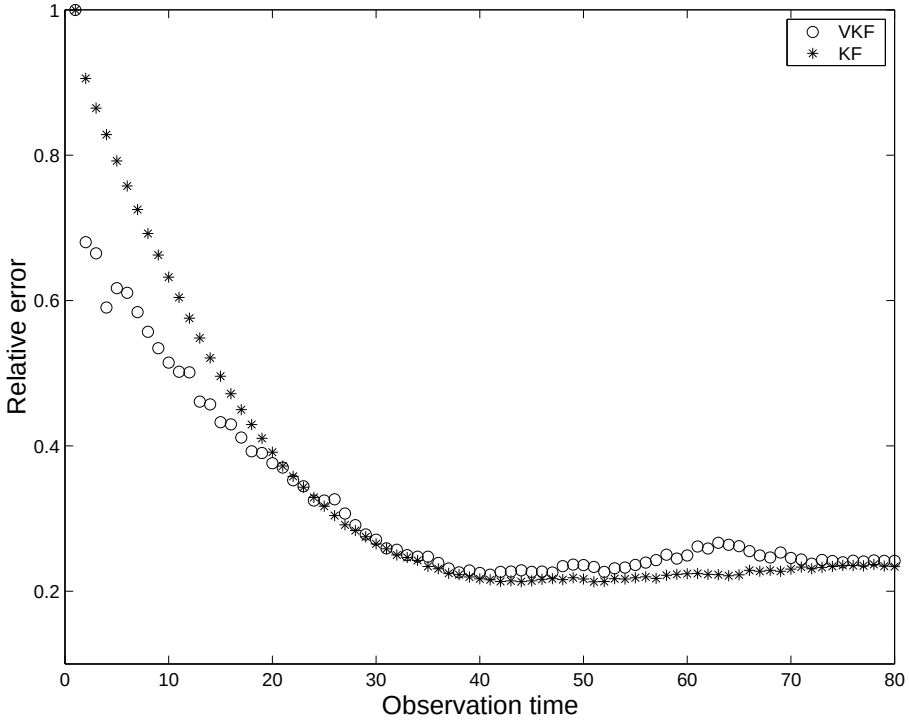

Figure 2. Relative error curves for KF (∗) and VKF (o). The horizontal axis represents the observation time.

once the forcing term, which is not used in the state space model in KF, has a prominent effect on the data; in early iterations, it is overwhelmed by the diffused initial temperature. We also mention that in the large number of test runs that we did using this large scale model, our implementation of the VKF was on average about 10 times faster than was the standard KF.

For a thorough comparison, we perform the same test using different values for σ^2_{ev} and σ^2_{obs}, namely, so that the signal to noise ratios mentioned above are both 10. The same initial inverse Hessian parameters were used inside the VKF method as in the previous test. The relative error curves in Figure 2 result. In this case, VKF provides better results at the begin of the filtering period. This might be due to a regularization effect, implicitly implemented via the the use of a truncated LBFGS algorithm.

Finally, we choose σ^2_{ev} and σ^2_{obs} as in the original experiment, but take $\alpha = 2$, which has the effect of making the state space model that is used within VKF and KF less accurate. In this case the initial inverse Hessian parameters were $\mathbf{B}_0^{-1} = \mathbf{I}$ for $\mathbf{C}_k^{est}$ and $\mathbf{B}_0^{-1} = 2000\mathbf{I}$ for $(\mathbf{C}_k^p)^{-1}$. When this is done, we obtain the solution curves appearing in Figure 3. Thus it seems that as the underlying evolution becomes less accurate, while the noise level remains moderately low, KF provides better results

In order to show that satisfactory results can also be obtained for much larger scale problems, we take $j = 8$ which gives $n = 256$. The number of unknowns in this problem is then 65536. We take all other parameter values to be those of the original experiment. Note, however, that the stability condition of the time stepping scheme

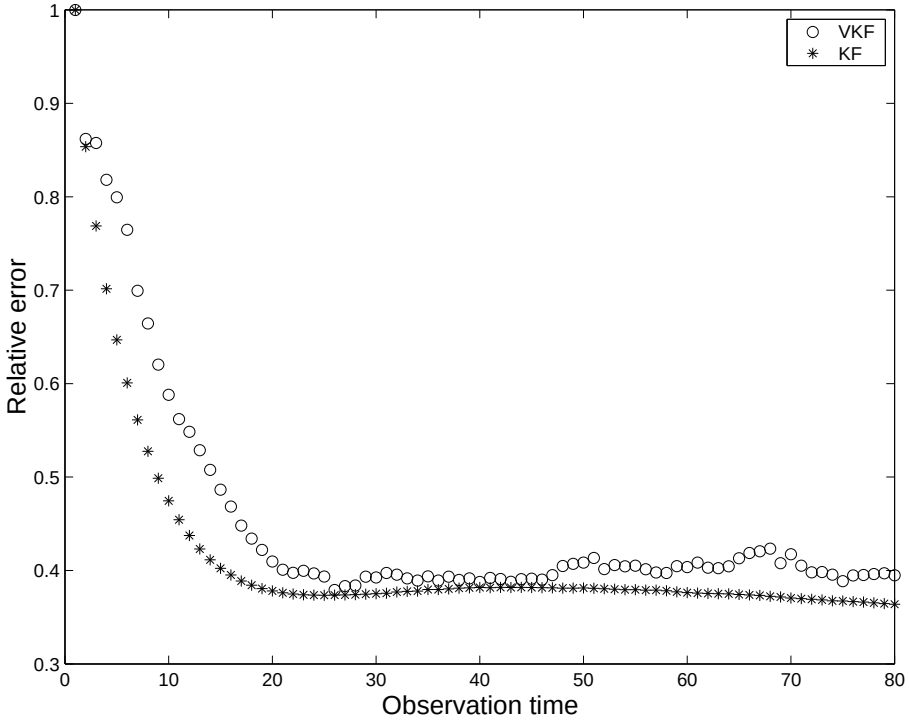

Figure 3. Relative error curves for KF (∗) and VKF (o). The horizontal axis represents the observation time.

requires a much smaller time step for this problem. A relative error plot similar to those in the previous example is given in Figure 4. We do not include an error curve for the Kalman filter because memory issues prevent it on our computer for either $n = 128$ or $n = 256$. However, in this case we compare results with LBFGS-KF method [1].

4.2. An Example with a Small-Scale, Nonlinear Evolution Model

In our next example, we apply EKF, VKF and VKS to the problem of estimating the state variables from data generated using the nonlinear, chaotic evolution model (20). To generate the data, a time integration of the model was first performed using a fourth order Runge-Kutta (RK4) method with time-step $\Delta t = 0.025$. Analysis in [11] suggests that when (20) is used as a test example for weather forecasting data assimilation algorithms, the characteristic time scale is such that the above Δt corresponds to 3 hours, which we will use in what follows. It is also noted in [11] that for $\Delta t \leq 0.5$, the RK4 method is stable. The "true data" was generated by taking 42920 time steps of the RK4 method, which corresponds to 5365 days. The initial state at the beginning of the data generation was $x^{20} = 8 + 0.008$ and $x^i = 8$ for all $i \neq 20$.

The observed data is then computed using this true data. In particular, after a 365 day long initial period, the true data is observed at every other time step and at the last 3 grid points in each set of 5; that is, the observation matrix is $m \times n$

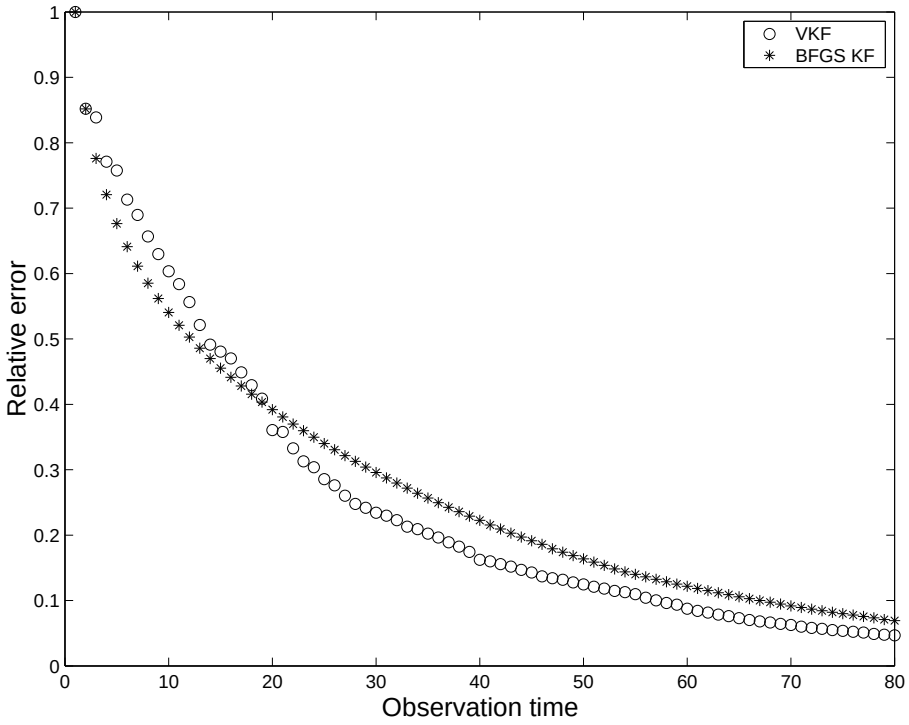

Figure 4. Relative error curves for LBFGS-KF ($*$) and VKF (o). The horizontal axis represents the observation time.

with nonzero entries

$$[\mathbf{K}]_{rs} = \begin{cases} 1 & (r,s) \in \{(3j+i, 5j+i+2) \mid i = 1,2,3,\ j = 0,1,\ldots,7\}, \\ 0 & \text{otherwise.} \end{cases}$$

The observation error is simulated using the Gaussian random vector $N(\mathbf{0}, (0.15\,\sigma_{\text{clim}})^2\mathbf{I})$ where σ_{clim} is a standard deviation of the model state used in climatological simulations, $\sigma_{\text{clim}} := 3.6414723$. The data generation codes were written in MATLAB and were transcribed by us from the `scilab` codes written by the author of [9].

In our application of EKF and VKF, we assume the coupled stochastic system

$$\mathbf{x}_{k+1} = \mathcal{M}(\mathbf{x}_k) + N(\mathbf{0}, (0.05\,\sigma_{\text{clim}})^2\mathbf{I}), \tag{23}$$

$$\mathbf{y}_{k+1} = \mathbf{K}\mathbf{x}_{k+1} + N(\mathbf{0}, (0.15\,\sigma_{\text{clim}})^2\mathbf{I}), \tag{24}$$

where $\mathcal{M}(\mathbf{x}_k)$ is obtained by taking two steps of the RK4 method applied to (20) from $\mathbf{x}_k$ with time-step 0.025. We note that if the noise term is removed from (23) and the above initial condition is used, our data generation scheme results.

Due to the fact that $\mathcal{M}$ is a nonlinear function, EKF must be used (see equations (12) and (13)). Since $\mathcal{K} := \mathbf{K}$ in (13) is linear, $\mathbf{K}_k = \mathbf{K}$ for all k in (14). However a linearization of the nonlinear evolution function $\mathcal{M}$ is required. Fortunately, the computation of $\mathbf{M}_k$ in (14) is performed by a routine in one of the `scilab` codes mentioned above and that we have adapted for our use in MATLAB.

The initial condition used in implementation of both the EKF and VKF is defined by $[\mathbf{x}_{t0}]_i = [\mathbf{x}_{t0}^{\text{true}}]_i + N(0, (0.3\,\sigma_{\text{clim}})^2)$ for all i, and the initial covariance was taken

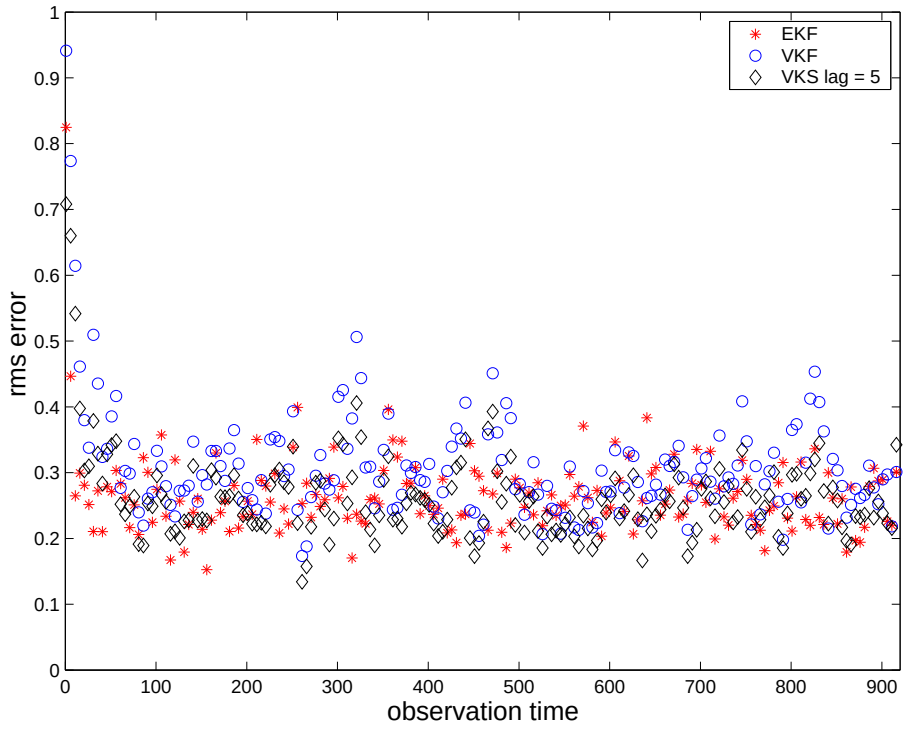

Figure 5. Plot of residual mean square error for VKF (o), EKF ($*$) and VKS ($\diamond$) applied to (20).

to be $\mathbf{C}_0^{est} = (0.13\,\sigma_{\rm clim})^2\mathbf{I}$. In our implementation of the LBFGS method within VKF, we computed 15 iterations with 14 saved vectors and the initial inverse Hessian parameters were $\mathbf{B}_0^{-1} = 0.15\mathbf{I}$ for $\mathbf{C}_k^{est}$ and $\mathbf{B}_0^{-1} = 10\mathbf{I}$ for $(\mathbf{C}_k^p)^{-1}$.

In order to analyze the the accuracy of the state estimates $\mathbf{x}_k^{est}$ obtained by EKF, VKF and VKS we plot the vector with components

$$[\mathbf{rms}]_k \quad = \quad \sqrt{\frac{1}{40}\|\mathbf{x}_k^{est} - \mathbf{x}_k^{true}\|^2} \tag{25}$$

in Figure 5. We can see that the all three methods yield comparable results.

In order to compare the forecasting abilities of the two approaches, we compute the following forecast statistics at every 8th observation. Take $j \in \mathcal{I} := \{8i \mid i = 1, 2, \ldots, 100\}$ and define

$$[\mathbf{forcast\_error}_j]_i \quad = \quad \frac{1}{40}\|\mathcal{M}_{4i}(\mathbf{x}_j^{est}) - \mathbf{x}_{j+4i}^{true}\|^2, \quad i = 1, \ldots, 20, \tag{26}$$

where $\mathcal{M}_n$ denotes a forward integration of the model by n time steps with the RK4 method. Thus this vector gives a measure of forecast accuracy given by the respective filter estimate up to 80 time steps, or 10 days out. This allows us to define the forecast skill vector

$$[\mathbf{forecast\_skill}]_i = \frac{1}{\sigma_{\rm clim}}\sqrt{\frac{1}{100}\sum_{j\in\mathcal{I}}[\mathbf{forecast\_error}_j]_i}, \quad i = 1, \ldots, 20, \tag{27}$$

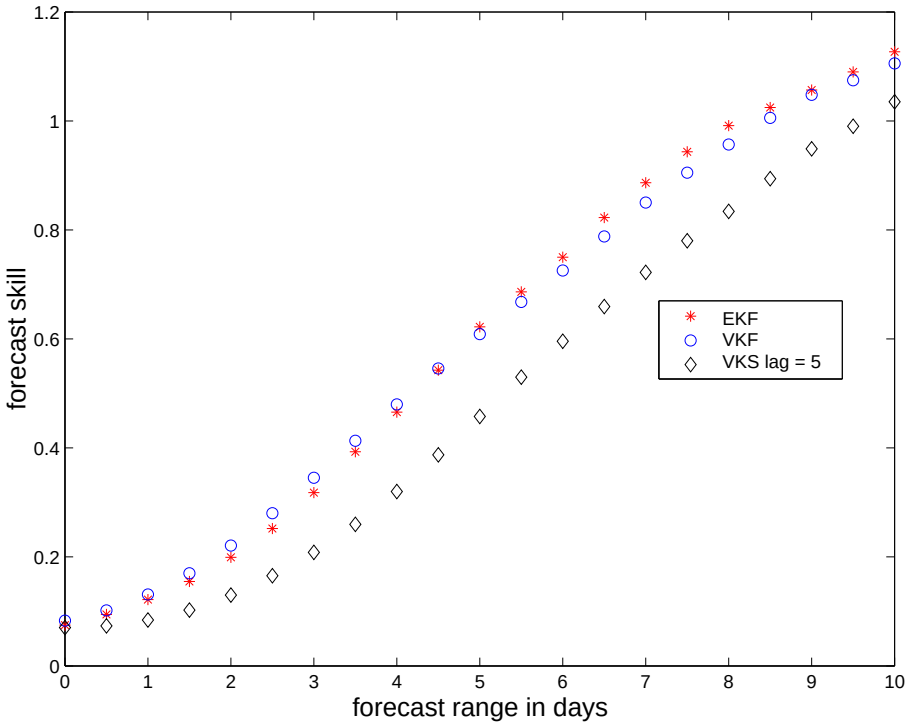

Figure 6. Plot of forecast skill vector for VKF (o) and EKF (∗) applied to (20). This plot also contains a retrospective forecast skill of the VKS (⋄) method.

which is plotted in Figure 6. The results show that the forecasting skill of the EKF and VKF is very similar, which suggests that on the whole, the quality of the VKF estimates is as high as those obtained using EKF. The forecast of VKS method is computed $lag = 5$ observation times afterwards and therefore it actually is not a forecast, but it demonstrates the improvement of the accuracy of retrospective analysis. Figure 6 also illustrates the fact that the Lorenz 95 model (20) is truly chaotic.

In the test cases considered here, a linear or linearized model matrix $\mathbf{M}_k$ has been available. This is not true in many important examples. In numerical weather forecasting, however, a tangent linear code is available that provides a means of computing the matrix vector product $\mathbf{M}_k\mathbf{x}$.

5. Conclusions

The standard implementations of KF and EKF become exceedingly time and memory consuming as the dimension of the underlying state space increases. Several variants of KF and EKF have been proposed to reduce the dimension of the system, thus making implementation in high dimensions possible. The Reduced Rank Kalman Filter or Reduced Order extended Kalman filter (see, e.g., [14], [4], [16]) project the dynamical state vector of the model onto a lower dimensional subspace. The success of this approach depends on a judicious choice of the reduction

operator. Moreover, since the reduction operator is typically fixed in time, they can suffer from "covariance leaks" [6]. A typical cause to this is that a nonlinear system does not generally leave any fixed linear subspace invariant.

In this paper, we propose the use of the limited memory BFGS (LBFGS) minimization method in order to circumvent the computational complexity and memory issues of standard KF and EKF. In particular, we replace the $n \times n$, where n is the dimension of the state space, covariance matrices within KF and EKF with low storage approximations obtain using LBFGS. The large-scale matrix inversions required in KF and EKF implementations are also approximated using LBFGS. We call the resulting method the Variational Kalman filter (VKF). In order to test these methods, we consider two test cases: a large-scale linear and a small scale nonlinear. The VKF is applied in the large-scale linear case and is shown to be effective. In fact, our method exceeds the speed of standard KF by an order of magnitude, and yields comparable results when both methods can be applied. Furthermore, it can be used on much larger scale problems. In the nonlinear, small scale case, VKF and VKS are implemented and are also shown to give results that are comparable to those obtained using standard EKF. We believe that these results suggest that our approach deserves further consideration. We note that in truly high-dimensional non-linear cases we need a tangent linear and corresponding adjoint code of the evolution model in order to get full benefits of the VKF method. In many important application fields, for example in numerical weather forecasting, such codes already are available both for the evolution model and observation model.

6. Acknowledgements

This work was supported by the Academy of Finland (application number 213476, Finnish programme for Centre of Excellence in research 2006–2011), with Tekes funding decision 40084/06). We are thankful for M. Leutbecher for providing us with the Scilab version of the Lorenz95 code, that served as starting point for the Matlab model implementation. Finally, the second author would like to thank both the University of Montana and the University of Helsinki for their support during his stay in Finland, where this work was, in part, undertaken.

7. Appendix

For completeness, we present the LBFGS method for a quadratic minimization and the limited memory formulations for the Hessian and inverse Hessian matrices. The LBFGS Minimization Algorithm for $q(\mathbf{u}) = \frac{1}{2}\langle \mathbf{A}\mathbf{u}, \mathbf{u}\rangle - \langle \mathbf{b}, \mathbf{u}\rangle$ reads as

$\nu := 0$;
$\mathbf{u}_0 :=$ initial guess;
$\mathbf{B}_0^{-1} :=$ initial inverse Hessian approximation;
begin quasi-Newton iterations
  $\mathbf{g}_\nu := \nabla q(\mathbf{u}_\nu) = \mathbf{A}\mathbf{u}_\nu - \mathbf{b}$;
  $\mathbf{B}_\nu^{-1} :=$ LBFGS approximation to $\mathbf{A}^{-1}$;
  $\mathbf{v}_\nu = \mathbf{B}_\nu^{-1}\mathbf{g}_\nu$;
  $\tau_\nu = \langle \mathbf{g}_\nu, \mathbf{v}_\nu\rangle / \langle \mathbf{v}_\nu, \mathbf{A}\mathbf{v}_\nu\rangle$;

$\mathbf{u}_{\nu+1} := \mathbf{u}_\nu - \tau_\nu \mathbf{v}_\nu$;
end quasi-Newton iterations

7.1. The Limited Memory Approximation for $\mathbf{A}^{-1}$

The BFGS matrix $\mathbf{B}_\nu^{-1}$ is computed using recursion

$$\mathbf{B}_{\nu+1}^{-1} = \mathbf{V}_\nu^{\mathrm{T}} \mathbf{B}_\nu^{-1} \mathbf{V}_\nu + \rho_\nu \mathbf{s}_\nu \mathbf{s}_\nu^{\mathrm{T}},$$

where

$$\begin{aligned} \mathbf{s}_\nu &:= \mathbf{u}_{\nu+1} - \mathbf{u}_\nu, \\ \mathbf{d}_\nu &:= \nabla q(\mathbf{u}_{\nu+1}) - \nabla q(\mathbf{u}_\nu), \\ \rho_\nu &:= 1/\mathbf{d}_\nu^{\mathrm{T}} \mathbf{s}_\nu, \\ \mathbf{V}_\nu &:= \mathbf{I} - \rho_\nu \mathbf{d}_\nu \mathbf{s}_\nu^{\mathrm{T}}. \end{aligned}$$

However, for large-scale problems the storage of the full matrix $\mathbf{B}_\nu^{-1}$ is infeasible, which motivates the limited storage version of the algorithm. At iteration ν, suppose that the j vector pairs $\{\mathbf{s}_i, \mathbf{d}_i\}_{i=\nu-j}^{\nu-1}$ are stored. Then we the LBFGS approximation of the inverse Hessian is given by

$$\begin{aligned} \mathbf{B}_\nu^{-1} &= (\mathbf{V}_{\nu-1}^{\mathrm{T}} \cdots \mathbf{V}_{\nu-j}^{\mathrm{T}}) \mathbf{B}_0^{-1} (\mathbf{V}_{\nu-j} \cdots \mathbf{V}_{\nu-1}) \\ &+ \rho_{\nu-j} (\mathbf{V}_{\nu-1}^{\mathrm{T}} \cdots \mathbf{V}_{\nu-j+1}^{\mathrm{T}}) \mathbf{s}_{\nu-j} \mathbf{s}_{\nu-j}^{\mathrm{T}} (\mathbf{V}_{\nu-j+1} \cdots \mathbf{V}_{\nu-1}) \\ &+ \rho_{\nu-j+1} (\mathbf{V}_{\nu-1}^{\mathrm{T}} \cdots \mathbf{V}_{\nu-j+2}^{\mathrm{T}}) \mathbf{s}_{\nu-j+1} \mathbf{s}_{\nu-j+1}^{\mathrm{T}} (\mathbf{V}_{\nu-j+2} \cdots \mathbf{V}_{\nu-1}) \\ &+ \vdots \\ &+ \rho_{\nu-1} \mathbf{s}_{\nu-1} \mathbf{s}_{\nu-1}^{\mathrm{T}}. \end{aligned} \tag{28}$$

Assuming exact arithmetic and that $j = n$, we have that $\mathbf{u}_\nu$ converges to the unique minimizer of q in at most n iterations, and if n iterations are performed $\mathbf{B}_{n+1}^{-1} = \mathbf{A}^{-1}$ [12]. In the implementation in this paper, however, $j << n$ and LBFGS iterations are stopped once a prespecified maximum number of iterations or gradient norm stopping tolerance is reached.

7.2. A Low Storage Approximation of $\mathbf{A}$

The required formulas are given in [3], and take the following form. Let

$$\mathbf{S}_\nu = [\mathbf{s}_{\nu-j}, \ldots, \mathbf{s}_{\nu-1}], \qquad \mathbf{D}_\nu = [\mathbf{d}_{\nu-j}, \ldots, \mathbf{d}_{\nu-1}],$$

then

$$\mathbf{B}_\nu = \xi_\nu \mathbf{I} - [\xi_\nu \mathbf{S}_\nu \quad \mathbf{D}_\nu] \begin{bmatrix} \xi_\nu \mathbf{S}_\nu^{\mathrm{T}} \mathbf{S}_\nu & \mathbf{L}_\nu \\ \mathbf{L}_\nu^{\mathrm{T}} & -\mathbf{D}_\nu \end{bmatrix}^{-1} \begin{bmatrix} \xi_\nu \mathbf{S}_\nu^{\mathrm{T}} \\ \mathbf{D}^{\mathrm{T}} \end{bmatrix}, \tag{29}$$

where $\mathbf{L}_\nu$ and $\mathbf{D}_\nu$ are the $j \times j$ matrices

$$(\mathbf{L}_\nu)_{i,j} = \begin{cases} \mathbf{s}_{\nu-j-1+i}^{\mathrm{T}} \mathbf{d}_{\nu-j-1+j} & \text{if } i > j, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$\mathbf{D}_\nu = \mathrm{diag}(\mathbf{s}_{\nu-j}^{\mathrm{T}} \mathbf{d}_{\nu-j}, \ldots, \mathbf{s}_{\nu-1}^{\mathrm{T}} \mathbf{d}_{\nu-1}).$$

We note that when $\xi_\nu = 1$ for all ν in (29), we have an exact equality between $\mathbf{B}_\nu$ in (29) and (28). However, we have found that a more accurate Hessian approximation is obtained if, following [12], we use the scaling $\xi_\nu = \mathbf{d}_{\nu-1}^{\mathrm{T}} \mathbf{d}_{\nu-1} / \mathbf{s}_{\nu-1}^{\mathrm{T}} \mathbf{d}_{\nu-1}$ instead.

We note that the middle matrix in (29) has size $2j \times 2j$, which is of reasonable size provided j is not too large, and its inversion can be carried out efficiently using a Cholesky factorization that exploits the structure of the matrix (for details see [3]).

REFERENCES

1. H. Auvinen, J. M. Bardsley, H. Haario, and T. Kauranne, *Large-Scale Kalman Filtering Using the Limited Memory BFGS Method*. Submitted 2008 to ETNA.
2. Harri Auvinen, Heikki Haario and Tuomo Kauranne, *Optimal approximation of Kalman filtering with a temporally local 4D-Var in operational weather forecasting*, Proceedings of the 11th ECMWF Workshop on Use of High-Performance Computing in Meteorology (W. Zwiefelhofer, G. Mozdzynski (eds.)), World Scientific 2005.
3. Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel, *Representations of Quasi-Newton Matrices and Their Use in Limited Memory Methods*, Mathematical Programming, Volume 63, Numbers 1-3, January, 1994, pp. 129-156.
4. Cane M.A., R.N.Miller, B. Tang, E.C. Hackert and A.J. Busalacchi, *Mapping tropical Pacific sea level: data assimilation via reduced state kalman filter*. J. Geophys. Res., Vol. 101, pp. 599–617, 1996.
5. J. Derber: A variational continuous assimilation technique. *Mon. Weather Rev.* **117** 2437-2446 (1989).
6. Michael Fisher and Erik Andersson, *Developments in 4D-Var and Kalman Filtering*, ECMWF Technical Memorandum 347, 2001.
7. R. E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME – Journal of Basic Engineering, 82 (Series D), 1960, pp. 35-45.
8. F.-X. LeDimet and O. Talagrand: Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus A series,* Vol. 38, pp. 97-110 (1986).
9. Martin Leutbecher, *A data assimilation tutorial based on the Lorenz-95 system*, European Centre for Medium-Range Weather Forecasts Web Tutorial, `www.ecmwf.int/newsevents/training/lecture_notes/pdf_files/ASSIM/Tutorial.pdf`
10. E. N. Lorenz, *Predictability: A problem partly solved*, Proc. Seminar on Predicability, Vol. 1, ECMWF, Reading, Berkshire, UK, pp. 1-18, 1996.
11. E. N. Lorenz and K. A. Emanuel, *Optimal Sites for Supplementary Weather Observations: Simulation with a Small Model*, Journal of Atmospheric Science, 1998, pp. 399-414.
12. Jorge Nocedal and Stephen Wright, *Numerical Optimization*, Springer 1999.
13. Clive D. Rodgers, *Inverse Problems for atmospheric sounding: Theory and Practice*, World Scientific, 2000.
14. Dee D.P., *Simplification of the Kalman filter for meteorological data assimilation* Quart. J. Roy. Meteor. Soc., vol. 117, pp. 365–384, 1990.
15. Curtis R. Vogel, *Computational Methods for Inverse Problems*, SIAM 2002.
16. Voutilainen A., T. Pyhälahti, K. Kallio, J. Pulliainen, H. Haario, J. Kaipio) *A filtering Approach for Estimating Lake Water Quality from Remote Sensing data* . Int. J. Appl. Earth Observation & Geoinformation, ISSN 0303-2434, Vol 9, 1, 50-64, February 2007.