Krylov space approximate Kalman filtering

Authors: Johnathan M. Bardsley, Albert Parker, Antti Solonen, & Marylesa Howard

Bardsley JM, Parker A, Solonen A, Howard M, "Krylov space approximate Kalman filtering," Numerical Linear Algebra with Applications, March 2013 20(2):171–184.

Krylov space approximate Kalman filtering

Johnathan M. Bardsley [1,*,†], Albert Parker [2], Antti Solonen [3] and Marylesa Howard [1]

[1]*Department of Mathematical Sciences, University of Montana, Missoula, MT, USA*
[2]*Center for Biofilm Engineering, Montana State University, Bozeman, MT, USA*
[3]*Department of Mathematics and Physics, Lappeenranta University of Technology, Lappeenranta & Finnish Meteorological Institute, Helsinki, Finland*

SUMMARY

The Kalman filter is a technique for estimating a time-varying state given a dynamical model for and indirect measurements of the state. It is used, for example, on the control problems associated with a variety of nav-igation systems. Even in the case of nonlinear state and/or measurement models, standard implementations require only linear algebra. However, for sufficiently large-scale problems, such as arise in weather forecast-ing and oceanography, the matrix inversion and storage requirements of the Kalman filter are prohibitive, and hence, approximations must be made. In this paper, we describe how the conjugate gradient iteration can be used within the Kalman filter for quadratic minimization, as well as for obtaining low-rank approx-imations of the covariance and inverse-covariance matrices required for its implementation. The approach requires that we exploit the connection between the conjugate gradient and Lanczos iterations. Copyright

KEY WORDS: Krylov subspace methods; conjugate gradient iteration; Lanczos iteration; Kalman filter; data assimilation; inverse problems

INTRODUCTION

In 1960, the Kalman filter (KF) was introduced by R. E. Kalman [1] as a statistically optimal method for recursively estimating a time-varying state, given a linear dynamical model, as well as indirect observations of the state.

The filter has been used extensively in application areas such as autonomous and assisted navigation. We are interested here in its use on large-scale examples, such as numerical weather forecasting, where standard formulations of KF, and its nonlinear analogue, the extended Kalman filter (EKF) [2], are computationally infeasible to implement.

Several variants of KF and EKF improve efficiency by projecting the state space onto a low-dimensional subspace; see, for example, [3–8]. This 'reduced-rank' approach is effective, provided the state is well represented on the subspace throughout the time window of the observations. How-ever, because the subspace is typically fixed in time, the dynamics of the system are often not correctly captured [9].

Another approach is to recast the filtering problem in variational form. In weather forecasting, for example, three-dimensional variational data assimilation (3D-Var) relies on a variational formulation of the Kalman filter [10], whereas the current state of the art is four-dimensional (4D)-Var [9,11], which utilizes a variational formulation of an initial value estimation problem [10,12,13] and has been shown to be identical to a Kalman smoother when the model is assumed to be perfect [14].

The quadratic minimization problems required for implementation of 3D-Var and 4D-Var are large scale (10^4–10^7 unknowns), and so efficient numerical optimization methods are needed. Similar to the 'reduced-rank' methods mentioned earlier, the partial orthogonal decomposition is used in [15] to reduce the dimensionality of the 4D-Var minimization problem. A more standard approach is to implement a preconditioned conjugate gradient (CG) method [8, 16–20].

In this paper, we propose the use of CG for quadratic minimization, as well as for the computation of 'low-rank' approximations of covariance, and inverse covariance, matrices. It is well known that when CG is applied to the minimization of

$$\phi(\mathbf{x}) = \frac{1}{2}\langle \mathbf{A}\mathbf{x}, \mathbf{x}\rangle - \langle \mathbf{b}, \mathbf{x}\rangle,$$

where $\mathbf{A}$ is symmetric, positive definite, and $n \times n$, and $\mathbf{x}$ and $\mathbf{b}$ are $n \times 1$, the minimizer, $\mathbf{A}^{-1}\mathbf{b}$, is obtained in, at most, n steps. Lesser known is the fact that CG can be used to efficiently build low-rank approximations of both $\mathbf{A}^{-1}$ and $\mathbf{A}$ using the close connection between the CG and Lanczos iterations. Using such low-rank approximations, we are able to obtain computationally efficient, low storage implementations of KF and of a variational formulation of KF (which we call VKF) that is similar to 3D-Var. The CG–Lanczos connection is also exploited in [19], where it is used instead to build preconditioners for CG iterations within 4D-Var.

Our overall approach is similar to that of [21, 22], the difference being that we use CG, rather than limited memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) [18, 23], for quadratic minimization, as well for constructing low storage covariance, and inverse covariance, approximations. The use of LBFGS within 3D and 4D-Var data assimilation was also explored in [8, 24].

The problem of using CG to build covariance, and inverse covariance, approximations has been studied extensively by Schneider and Willsky [25, 26]. These authors even suggested the use of their ideas in the context of the Kalman filter applied to an oceanography problem [27] but provided no mathematical detail regarding their implementation. In this paper, we introduce two different approximate Kalman filters, each of which employs CG for both quadratic minimization as well as for obtaining low-rank covariance and inverse covariance approximations.

The paper is organized as follows. In Section 2, we present both KF and VKF, and provide a general outline of the approximate filters, which we denote CG-KF and CG-VKF. In Section 3, we provide details on how to use CG to compute low-rank approximations of covariance and inverse covariance matrices and provide optimality results, as well as bounds on the error of these approximations. CG-KF and CG-VKF are tested on two numerical examples in Section 4, and we provide conclusions in Section 5.

2. KALMAN FILTERING METHODS

We begin our mathematical discussion by considering the following coupled system of discrete, linear, stochastic difference equations

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

$$\mathbf{y}_k = \mathbf{K}_k \mathbf{x}_k + \varepsilon_k^o. \tag{2}$$

In the first equation, $\mathbf{x}_k$ denotes the $n \times 1$ state vector of the system at time k; $\mathbf{M}_k$ is the $n \times n$ linear evolution operator; and ε_k^p is an $n \times 1$ random vector representing the prediction error and is assumed to characterize errors in the model and in the corresponding numerical approximations. In the second equation, $\mathbf{y}_k$ denotes the $m \times 1$ observed data vector; $\mathbf{K}_k$ is the $m \times n$ linear observation operator; and ε_k^o is an $m \times 1$ random vector representing the observation error. The error terms are assumed to be independent and normally distributed, with zero mean and with covariance matrices $\mathbf{C}_{\varepsilon_k^p}$ and $\mathbf{C}_{\varepsilon_k^0}$, respectively.

The task is to estimate the state $\mathbf{x}_k$ and its error covariance $\mathbf{C}_k$ at time point k given $\mathbf{y}_k$, $\mathbf{K}_k$, $\mathbf{C}_{\varepsilon_k^o}$, $\mathbf{M}_k$, $\mathbf{C}_{\varepsilon_k^p}$, and estimates $\mathbf{x}_{k-1}^{\text{est}}$ and $\mathbf{C}_{k-1}^{\text{est}}$ of the state and covariance at time point $k-1$.

The classical Kalman filter (KF) is the standard approach taken for such problems. It is optimal in the sense that it yields a minimum variance estimator [1] and has the following form.

Algorithm 1 (KF)
Select initial guess $\mathbf{x}_0^{\text{est}}$ and covariance $\mathbf{C}_0^{\text{est}}$ and set $k = 1$.

1. Compute the evolution model estimate and covariance:
   (a) compute $\mathbf{x}_k^p = \mathbf{M}_k \mathbf{x}_{k-1}^{\text{est}}$;
   (b) define $\mathbf{C}_k^p = \mathbf{M}_k \mathbf{C}_{k-1}^{\text{est}} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p}$.
2. Compute KF estimate and covariance:
   (a) define the Kalman Gain $\mathbf{G}_k = \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} (\mathbf{K}_k \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^0})^{-1}$;
   (b) compute the KF estimate $\mathbf{x}_k^{\text{est}} = \mathbf{x}_k^p + \mathbf{G}_k(\mathbf{y}_k - \mathbf{K}_k \mathbf{x}_k^p)$;
   (c) define the estimate covariance $\mathbf{C}_k^{\text{est}} = \mathbf{C}_k^p - \mathbf{G}_k \mathbf{K}_k \mathbf{C}_k^p$.
3. Update $k := k + 1$ and return to Step 1.

The cost of implementing the KF iteration is about $1/3(m^3)$ floating point operations (flops) for the computation of the Cholesky factorization of the matrix whose inverse appears in Step 2(a), as well as the system back-solves with this matrix required in Steps 2(b) and (c) [28]. Storage of, and multiplication by, full $n \times n$ covariance matrices is also required.

An equivalent variational formulation of KF follows from a sequential application of Bayes' Theorem. To see this, we recall Bayes' formula

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

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 posteriori 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 with normally distributed error, the function ℓ assumes the form

$$\ell(\mathbf{x}|\mathbf{y}_k) = \frac{1}{2}(\mathbf{y}_k - \mathbf{K}_k \mathbf{x})^{\mathrm{T}} \mathbf{C}_{\varepsilon_k^0}^{-1} (\mathbf{y}_k - \mathbf{K}_k \mathbf{x}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k^p)^{\mathrm{T}} (\mathbf{C}_k^p)^{-1} (\mathbf{x} - \mathbf{x}_k^p), \tag{5}$$

where $\mathbf{C}_{\varepsilon_k^0}$ and $\mathbf{C}_k^p$ are the covariance matrices of the measurement noise ε_k^o and of the prior $\mathbf{x}_k^p$, respectively.

The KF estimate $\mathbf{x}_k^{\text{est}}$ and its covariance $\mathbf{C}_k^{\text{est}}$ are precisely the minimizer and inverse Hessian of $\ell(\mathbf{x}|\mathbf{y}_k)$, respectively. This allows us to re-express the KF iteration in a variational form, which we denote the variational Kalman filter (VKF) [22].

Algorithm 2 (VKF)
Select initial guess $\mathbf{x}_0^{\text{est}}$ and covariance $\mathbf{C}_0^{\text{est}}$ and set $k = 1$.

1. Compute the evolution model estimate and covariance:
   (a) compute $\mathbf{x}_k^p = \mathbf{M}_k \mathbf{x}_{k-1}^{\text{est}}$;
   (b) define $\mathbf{C}_k^p = \mathbf{M}_k \mathbf{C}_{k-1}^{\text{est}} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p}$.
2. Compute VKF and covariance estimates:
   (a) compute the minimizer $\mathbf{x}_k^{\text{est}}$ and inverse Hessian $\mathbf{C}_k^{\text{est}}$ of

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

   to obtain $\mathbf{x}_k^{\text{est}}$ and $\mathbf{C}_k^{\text{est}}$.
3. Update $k := k + 1$ and return to Step 1.

The cost of implementing VKF is dominated by the computation of the minimizer of $\ell(\mathbf{x}|\mathbf{y}_k)$, whose definition requires the inverse of $\mathbf{C}_k^p$. The storage of $n \times n$ covariance matrices is also required. However, as we will see later, VKF is particularly amenable to computationally efficient approximations.

2.1. Extensions to nonlinear models

Nonlinear extensions of KF and VKF have been developed for the case when (1) and (2) are replaced by

$$\mathbf{x}_k = \mathcal{M}(\mathbf{x}_{k-1}) + \varepsilon_k^p, \tag{6}$$

$$\mathbf{y}_k = \mathcal{K}(\mathbf{x}_k) + \varepsilon_k^o, \tag{7}$$

where $\mathcal{M}$ and $\mathcal{K}$ are possibly nonlinear functions defined by the evolution and observation operators, respectively.

The best-known extension is the extended Kalman filter (EKF). EKF is obtained by the following modification of the KF algorithm: in Step 1, (a), use the nonlinear model $\mathbf{x}_k^p = \mathcal{M}(\mathbf{x}_{k-1}^{\text{est}})$ to compute the prior, but otherwise, use the following linearized approximations of $\mathcal{M}$ and $\mathcal{K}$:

$$\mathbf{M}_k = \frac{\partial \mathcal{M}(\mathbf{x}_{k-1}^{\text{est}})}{\partial \mathbf{x}} \quad \text{and} \quad \mathbf{K}_k = \frac{\partial \mathcal{K}(\mathbf{x}_k^p)}{\partial \mathbf{x}}. \tag{8}$$

Exactly the same changes can be made to VKF to incorporate nonlinear evolution and observation models. This is the approach we take for the nonlinear example in this paper.

The linearizations $\mathbf{M}_k$ and $\mathbf{K}_k$ in (8) can be computed or estimated in a number of ways. A common approach is to use finite differences. A more efficient approach—both computationally and in terms of storage—employs the adjoint and tangent linear codes defined by the numerical scheme(s) used in the solution of the evolution and/or the observation model. These codes are available in many important instances, for example, in weather forecasting [13]. The tangent code for the evolution and observation operators computes multiplication of a vector by $\mathbf{M}_k$ and $\mathbf{K}_k$, respectively; whereas the adjoint code for the evolution and observation operators computes multiplication of a vector by $\mathbf{M}_k^{\mathrm{T}}$ and $\mathbf{K}_k^{\mathrm{T}}$, respectively.

2.2. Efficient Kalman filter and variational Kalman filter algorithms using conjugate gradient

For large-scale problems, KF, EKF, and VKF can be prohibitively expensive to implement because of the need for the storage and inversion of large, dense matrices at every iteration. To address this challenge, we advocate the use of iterative methods within the filters for both the solution of linear systems (or quadratic minimization), as well as for obtaining low storage approximations of these matrices. In [21, 22], the LBFGS method was used for this purpose, whereas in this paper, we describe how to use the conjugate gradient (CG) method for efficient implementation of these filtering methods.

We assume that multiplication by the evolution and observation matrices $\mathbf{M}_k$ and $\mathbf{K}_k$ and their transposes are efficient, both in terms of storage and CPU time.

CG is a well-known iterative method for minimizing quadratic functions of the type

$$\phi(\mathbf{x}) = \frac{1}{2}\langle \mathbf{A}\mathbf{x}, \mathbf{x}\rangle - \langle \mathbf{b}, \mathbf{x}\rangle, \tag{9}$$

where $\mathbf{A}$ is an $n \times n$ symmetric positive-definite matrix and $\mathbf{b}$ is an $n \times 1$ vector [18]. After k iterations of CG, we obtain an approximate minimizer of ϕ, which we denote $\mathbf{x}_{\text{CG}}^k$. We will show later that the CG iteration history can also be used to create an approximation $\mathbf{B}_k$ of $\mathbf{A}^{-1}$ and that both $\mathbf{x}_{\text{CG}}^k$ and $\mathbf{B}_k$ are optimal approximations over a certain Krylov subspace. Moreover, we will show how to use the connection between CG and the Lanczos iterations to efficiently and stably compute both $\mathbf{B}_k$ and its pseudoinverse $\mathbf{B}_k^\dagger$, which approximates $\mathbf{A}$, also from CG iteration history.

In order to illustrate how CG is used within KF and VKF, we include a pseudocode for the approximate filtering methods.

Algorithm 3 (CG-KF)
Select initial guess $\mathbf{x}_0^{\text{est}}$ and lower storage initial covariance $(\mathbf{B}_0^{\#})^{\dagger} = \mathbf{C}_0^{\text{est}}$ and set $k = 1$.

1. Compute the evolution model estimate and covariance:
   (a) compute $\mathbf{x}_k^p = \mathbf{M}_k \mathbf{x}_{k-1}^{\text{est}}$;
   (b) define $\mathbf{C}_k^p = \mathbf{M}_k (\mathbf{B}_{k-1}^{\#})^{\dagger} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p}$.
2. Compute KF estimate and covariance:
   (a) apply CG to (9), with $\mathbf{A} = \mathbf{K}_k \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^o}$ and $\mathbf{b} = (\mathbf{y}_k - \mathbf{K}_k \mathbf{x}_k^p)$, to obtain $\mathbf{x}_k^*$, and low rank approximation $\mathbf{B}_k^*$ of $\mathbf{A}^{-1}$;
   (b) compute approximate KF estimate $\mathbf{x}_k^{\text{est}} = \mathbf{x}_k^p + \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} \mathbf{x}_k^*$;
   (c) apply CG to (9) with $\mathbf{A} = \mathbf{C}_k^p - \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} \mathbf{B}_k^* \mathbf{K}_k \mathbf{C}_k^p$ and $\mathbf{b} = \mathbf{v}$, where $\mathbf{v}$ is a white noise random vector, to obtain low-rank approximation $(\mathbf{B}_k^{\#})^{\dagger}$ of $\mathbf{A}$.
3. Update $k := k + 1$ and return to Step 1.

Note that CG-KF closely mimics KF (or Algorithm 1). Also, in Step 2(a), the matrix $\mathbf{A}$ is $m \times m$, where m is the dimension of the observation space. Thus, if m is small, the application of CG in Step 2(a) could be removed.

In Step 2(c), we choose $\mathbf{v}$ to be the random vector with entries -1 or 1 with equal probability, because this optimizes the accuracy of the covariance/inverse covariance approximations [29]. We have also tried taking $\mathbf{v} \sim N(\mathbf{0}, \mathbf{I})$, which works equally well. Note that, here, the covariance/inverse covariance approximation is of interest.

For VKF, we make similar modifications. However, only one application of CG per filter iteration is required, provided we assume that the approximation of the estimate covariance $\mathbf{B}_k^{\#} \approx \mathbf{C}_k^{\text{est}}$ can be written in square root form

$$\mathbf{B}_k^{\#} = \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}},$$

where $\mathbf{X}_k$ is defined as in Remark 6 later and is $n \times p$ with p the number of CG iterations. Then, using the matrix inversion lemma,

$$\begin{aligned}(\mathbf{C}_k^p)^{-1} &= ((\mathbf{M}_k \mathbf{X}_k)(\mathbf{M}_k \mathbf{X}_k)^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p})^{-1} \\ &= \mathbf{C}_{\varepsilon_k^p}^{-1} - \mathbf{C}_{\varepsilon_k^p}^{-1} \mathbf{M}_k \mathbf{X}_k (\mathbf{I} + \mathbf{X}_k^{\mathrm{T}} \mathbf{M}_k^{\mathrm{T}} \mathbf{C}_{\varepsilon_k^p}^{-1} \mathbf{M}_k \mathbf{X}_k)^{-1} \mathbf{X}_k^{\mathrm{T}} \mathbf{M}_k^{\mathrm{T}} \mathbf{C}_{\varepsilon_k^p}^{-1}.\end{aligned} \tag{10}$$

We assume that $\mathbf{C}_{\varepsilon_k^p}$ is diagonal and that p is small enough that the $p \times p$ matrix inverse required in (10) can be computed efficiently. Then multiplication by $(\mathbf{C}_k^p)^{-1}$ can be efficiently performed using (10).

Algorithm 4 (CG-VKF)
Select initial guess $\mathbf{x}_0^{\text{est}}$ and low-rank covariance approximation $\mathbf{B}_0^{\#} = \mathbf{X}_0 \mathbf{X}_0^{\mathrm{T}}$ of $\mathbf{C}_0^{\text{est}}$ and set $k = 1$.

1. Compute the evolution model estimate and covariance:
   (a) compute $\mathbf{x}_k^p = \mathbf{M}_k \mathbf{x}_{k-1}^{\text{est}}$;
   (b) define $(\mathbf{C}_k^p)^{-1}$ using (10).
2. Compute VKF and covariance estimates:
   (a) apply CG to the problem of minimizing

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

   which has the form (9) with $\mathbf{A} = \mathbf{K}_k^{\mathrm{T}} (\mathbf{C}_{\varepsilon_k^o})^{-1} \mathbf{K}_k + (\mathbf{C}_k^p)^{-1}$ and $\mathbf{b} = \mathbf{K}_k^{\mathrm{T}} (\mathbf{C}_{\varepsilon_k^o})^{-1} \mathbf{y}_k + (\mathbf{C}_k^p)^{-1} \mathbf{x}_k^p$, to obtain $\mathbf{x}_k^{\text{est}}$ and low-rank approximation $\mathbf{B}_k^{\#} = \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}$ of $\mathbf{A}^{-1}$.
3. Update $k := k + 1$ and return to Step 1.

We note that if p is too large, the matrix inversion lemma (10) will be expensive to compute. In this case, an auxiliary optimization, such as was done in Step 2(c) of CG-KF, can be used instead. However, in our experiments, an additional regularization term was needed with this approach.

Extensions of CG-KF and CG-VKF to the nonlinear cases (6) and (7) are exactly as for KF and VKF. Thus computation of the tangent linear and adjoint models are required, and in Step 1(a) of both algorithms, $\boldsymbol{x}_k^p$ is computed using $\mathbf{x}_k^p = \mathcal{M}(\mathbf{x}_{k-1}^{\text{est}})$.

The cost of implementing CG-KF and CG-VKF is dominated by the matrix-vector multiplies at a cost of about $2n^2$ (or $2m^2$) flops in each CG iteration [28]. Thus, as long as the number of CG iterations p is small relative to n, then CG-KF and CG-VKF will be cheaper to implement than KF and VKF. In addition, the covariance approximations used require only the storage of an $n \times p$ (rather than $n \times n$) matrix.

It remains to show how to construct the covariance and inverse covariance approximations ($\mathbf{B}_k^*$ and $(\mathbf{B}_k^\#)^\dagger$ in CG-KF and $\mathbf{B}_k^\#$ in CG-VKF) from CG iteration history. We do this in the next section.

3. KRYLOV SPACE APPROXIMATION OF $\mathbf{A}$ AND $\mathbf{A}^{-1}$

In this section, we provide necessary details for the implementation of CG within both CG-KF and CG-VKF. In particular, focusing on the general minimization problem (9), we present mathematical results regarding the efficient computation of both $\mathbf{B}_k^\dagger \approx \mathbf{A}$ and $\mathbf{B}_k \approx \mathbf{A}^{-1}$ from CG iterations. For this, we exploit the Lanczos iteration and its close connection to CG [30–32].

3.1. Conjugate gradient iteration

Given a symmetric, positive-definite $n \times n$ matrix $\mathbf{A}$, CG is a well-known iterative algorithm for solving the linear system $\mathbf{Ax} = \mathbf{b}$ or, equivalently, for minimizing a quadratic of the form (9) [33]. In order to establish necessary notation, we present the CG iteration now.

Algorithm 5
Given $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{x}^0$, let $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{p}^0 = \mathbf{r}^0$, and $k = 1$. Specify some stopping tolerance ϵ. Iterate:

1. $\gamma_{k-1} = \frac{\mathbf{r}^{(k-1)\mathrm{T}}\mathbf{r}^{k-1}}{\mathbf{p}^{(k-1)\mathrm{T}}\mathbf{A}\mathbf{p}^{k-1}}$ is the one-dimensional minimizer of ϕ in the direction $\mathbf{x}^{k-1} + \gamma\mathbf{p}^{k-1}$.
2. $\mathbf{x}^k = \mathbf{x}^{k-1} + \gamma_{k-1}\mathbf{p}^{k-1}$.
3. $\mathbf{r}^k = -\nabla_x\phi(\mathbf{x}^k) = \mathbf{b} - \mathbf{A}\mathbf{x}^k = \mathbf{r}^{k-1} - \gamma_{k-1}\mathbf{A}\mathbf{p}^{k-1}$ is the residual.
4. $\beta_k = -\frac{\mathbf{r}^{k\mathrm{T}}\mathbf{r}^k}{\mathbf{r}^{(k-1)\mathrm{T}}\mathbf{r}^{k-1}}$.
5. $\mathbf{p}^k = \mathbf{r}^k - \beta_k\mathbf{p}^{k-1}$ is the next conjugate search direction.
6. Quit if $||\mathbf{r}^k|| < \epsilon$. Else, set $k := k + 1$ and go to Step 1.

If we let $\mathbf{P}_k$ be the $n \times k$ matrix with $\{\mathbf{p}^i\}_{i=0}^{k-1}$ as columns and $\mathbf{P}_B$ be the $n \times (n-k)$ matrix with $\{\mathbf{p}^i\}_{i=k}^{n-1}$ as columns, then given what we know about CG (see, e.g., [31]),

$$\mathbf{D}_n = \begin{pmatrix} \mathbf{D}_k & 0 \\ 0 & \mathbf{D}_B \end{pmatrix} = \begin{pmatrix} \mathbf{P}_k^{\mathrm{T}}\mathbf{A}\mathbf{P}_k & 0 \\ 0 & \mathbf{P}_B^{\mathrm{T}}\mathbf{A}\mathbf{P}_B \end{pmatrix} = \mathbf{P}_n^{\mathrm{T}}\mathbf{A}\mathbf{P}_n$$

is an invertible diagonal matrix with entries $[\mathbf{D}_n]_{ii} = \mathbf{p}^{i\mathrm{T}}\mathbf{A}\mathbf{P}^i$. Thus,

$$\mathbf{A}^{-1} = \mathbf{P}_n\mathbf{D}_n^{-1}\mathbf{P}_n^{\mathrm{T}} = \mathbf{P}_k\mathbf{D}_k^{-1}\mathbf{P}_k^{\mathrm{T}} + \mathbf{P}_B\mathbf{D}_B^{-1}\mathbf{P}_B^{\mathrm{T}}.$$

For $k < n$, the k-rank approximation of $\mathbf{A}^{-1}$ produced by the CG algorithm is then

$$\mathbf{B}_k = \mathbf{P}_k\mathbf{D}_k^{-1}\mathbf{P}_k^{\mathrm{T}}. \tag{11}$$

Remark 6
We use (11) to define $\mathbf{B}_k$ within CG-KF, Step 2(a), and CG-VKF, Step 2(a). To minimize storage requirements, the matrix $\mathbf{X}_k = \mathbf{P}_k\mathbf{D}_k^{-1/2}$ is saved, which is $n \times p$, where p is the number of CG iterations.

3.2. Lanczos iteration

The Lanczos algorithm is an iterative method for solving the eigenvalue problem for large sparse matrices [34,35], and its performance in finite precision is well studied [32,36,37]. CG is equivalent to the Lanczos method for symmetric, positive-definite matrices [30–32].

In exact arithmetic, the Lanczos algorithm approximates the eigenpairs $(\lambda_i, \mathbf{w}^i)$ of a given $n \times n$ positive-definite matrix $\mathbf{A}$. In order to establish necessary notation, we provide the two-term recurrence version of the Lanczos algorithm due to Paige [32].

Algorithm 7
Given an initial vector $\tilde{\mathbf{v}}^0$, let $\mathbf{v}^0 = \frac{\tilde{\mathbf{v}}^0}{||\tilde{\mathbf{v}}^0||}$, $\alpha_0 = \mathbf{v}^{0\mathrm{T}}\mathbf{A}\mathbf{v}^0$, $\tilde{\mathbf{v}}^1 = \mathbf{A}\mathbf{v}^0 - \alpha_0\mathbf{v}^0$ and $k = 1$. Specify some stopping tolerance ϵ. Iterate:

1. $\eta_k = ||\tilde{\mathbf{v}}^k||$. Quit if $\eta_k < \epsilon$.
2. $\mathbf{v}^k = \frac{\tilde{\mathbf{v}}^k}{\eta_k}$ is a Lanczos vector.
3. $\mathbf{u}^k = \mathbf{A}\mathbf{v}^k - \eta_k\mathbf{v}^{k-1}$.
4. $\alpha_k = \mathbf{v}^{k\mathrm{T}}\mathbf{u}^k$.
5. $\tilde{\mathbf{v}}^{k+1} = \mathbf{u}^k - \alpha_k\mathbf{v}^k$.
6. $k := k + 1$, and go to Step 1.

If we define $\mathbf{V}_k$ to be the matrix with columns $\{\mathbf{v}_i\}_{i=0}^{k-1}$ and the tridiagonal Lanczos matrix $\mathbf{T}_k$ by

$$\mathbf{T}_k = \begin{pmatrix} \alpha_0 & \eta_1 & & & \\ \eta_1 & \alpha_1 & \eta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \eta_{k-2} & \alpha_{k-2} & \eta_{k-1} \\ & & & \eta_{k-1} & \alpha_{k-1} \end{pmatrix},$$

then [31,32]

$$\mathbf{T}_k = \mathbf{V}_k^{\mathrm{T}}\mathbf{A}\mathbf{V}_k,$$

which suggests the following low-rank approximation of $\mathbf{A}$:

$$\mathbf{B}_k^{\dagger} = \mathbf{V}_k\mathbf{T}_k\mathbf{V}_k^{\mathrm{T}}. \tag{12}$$

An efficient approximate eigenvalue decomposition for $\mathbf{A}$ can be efficiently obtained from (12). In particular, because $\mathbf{T}_k$ is tridiagonal, its spectral decomposition

$$\mathbf{T}_k = \mathbf{Y}_k\boldsymbol{\Theta}_k\mathbf{Y}_k^{\mathrm{T}}, \quad \boldsymbol{\Theta}_k = \mathrm{diag}(\theta_1^k, \ldots, \theta_k^k),$$

can be efficiently computed [31,36,38]. The Lanczos approximate eigen-decomposition of $\mathbf{A}$ is then given by

$$\mathbf{B}_k^{\dagger} = (\mathbf{V}_k\mathbf{Y}_k)\boldsymbol{\Theta}_k(\mathbf{V}_k\mathbf{Y}_k)^{\mathrm{T}}, \tag{13}$$

with the Ritz pairs $\{(\theta_i^k, \mathbf{V}_k\mathbf{y}_k^i)\}$ approximating k of the eigenpairs of $\mathbf{A}$.

Remark 8
We use covariance approximation (13) to define $(\mathbf{B}_k^{\#})^{\dagger}$ within CG-KF, Step 2(c). To minimize storage requirements, the $n \times p$ matrix $\mathbf{X}_k = \mathbf{V}_k\mathbf{Y}_k\boldsymbol{\Theta}_k^{1/2}$ is saved.

Remark 9
As an aside, we also note that the use of CG/Lanczos within CG-KF and CG-VKF when $\mathcal{K}$ and/or $\mathcal{M}$ are nonlinear is related to the work of Saad [39], where the (nonlinear) matrix exponential is approximated by a matrix generated using the Lanczos method.

3.3. Equivalence of the conjugate gradient and Lanczos approximations

To obtain $\mathbf{V}_k$ from CG iterations, which is what will be required in KF applications, we note that its columns can be obtained from the CG residuals via ([31,32], p. 50)

$$\mathbf{v}^i = (-1)^i \frac{\mathbf{r}^i}{||\mathbf{r}^i||}. \tag{14}$$

Moreover, it can be shown [32] that

$$\mathbf{T}_k^{-1} = \mathbf{R}_k \mathbf{D}_k^{-1} \mathbf{R}_k^{\mathrm{T}}, \quad \text{where} \quad \mathbf{R}_k = \mathbf{V}_k^{\mathrm{T}} \mathbf{P}_k. \tag{15}$$

and hence,

$$\mathbf{V}_k \mathbf{T}_k^{-1} \mathbf{V}_k^{\mathrm{T}} = \mathbf{P}_k \mathbf{D}_k^{-1} \mathbf{P}_k^{\mathrm{T}},$$

which shows that (11) and (12) are equivalent.

Remark 10
In our implementation, the covariance approximation (13) is obtained from CG iterations via (14) and (15), and the eigenvalue decomposition $\mathbf{T}_k^{-1} = \mathbf{Y}_k \mathbf{\Theta}_k^{-1} \mathbf{Y}_k^{\mathrm{T}}$.

3.4. Optimality and error bounds for matrix approximations

The fact that $\mathbf{T}_k = \mathbf{V}_k^{\mathrm{T}} \mathbf{A} \mathbf{V}_k$ shows that the Lanczos algorithm is a Raleigh–Ritz process, with $\mathbf{T}_k$ the minimizer of $\rho(\boldsymbol{\zeta}|\mathbf{V}_k) := ||\mathbf{A}\mathbf{V}_k - \mathbf{V}_k \boldsymbol{\zeta}||_2$ [36]. In other words,

$$\min_{\boldsymbol{\zeta} \in \mathbb{R}^{k \times k}} \rho(\boldsymbol{\zeta}|\mathbf{V}_k) = ||(\mathbf{A} - \mathbf{V}_k \mathbf{T}_k \mathbf{V}_k^{\mathrm{T}}) \mathbf{V}_k||_2,$$

which shows that $\mathbf{B}_k^{\dagger} = \mathbf{V}_k \mathbf{T}_k \mathbf{V}_k^{\mathrm{T}}$ is the best rank k approximation of $\mathbf{A}$ in range($\mathbf{V}_k$), the k-dimensional Krylov space also spanned by the CG conjugate directions. Moreover, when *CG* converges, this Krylov space contains the eigenspaces corresponding to the extreme and well-separated eigenvalues of $\mathbf{A}$ [32,36,37,40]. That is, $\mathbf{B}_k^{\dagger}$ is an accurate approximation of $\mathbf{A}$ (and $\mathbf{B}_k$ approximates $\mathbf{A}^{-1}$) in these eigenspaces.

Moreover, from Weyl's Theorem [36] and the triangle inequality, it can be shown that the norm of the error in the covariance approximation, $||\mathbf{A}^{-1} - \mathbf{B}_k||_2$, is at least as large as $1/\lambda_{k+1}$, the largest eigenvalue of $\mathbf{A}^{-1}$ not being estimated by the $\mathbf{T}_k^{-1}$, and it can become as large as this eigenvalue plus the error in the Lanczos estimates. Similarly, $||\mathbf{A} - \mathbf{B}_k^{\dagger}||_2$ is at least as large as $\lambda_{-(k+1)}$, the largest eigenvalue of $\mathbf{A}$ not being estimated by the Lanczos tridiagonal $\mathbf{T}_k$, and it can become as large as this eigenvalue plus the error in the Lanczos Ritz pairs.

3.5. The effect of finite precision

In finite precision, the CG search directions $\{\mathbf{p}^k\}$ lose conjugacy. Nevertheless, as long as 'local conjugacy' is maintained, the eigenvalues of $\mathbf{A}$ are not on the order of machine precision and the condition number of $\mathbf{A}$ is not too large, convergence $\mathbf{x}_{\mathrm{CG}}^k \to \mathbf{A}^{-1}\mathbf{b}$ is guaranteed [32].

On the other hand, loss of conjugacy of the search directions (which corresponds to loss of orthogonality of the Lanczos vectors $\{\mathbf{v}^k\}$) is detrimental to a Lanczos eigen-solver, resulting in only a few of the eigenvalues of $\mathbf{A}$ being estimated, the extreme and well-separated ones. It can be shown [32] that loss of conjugacy/orthogonality occurs at the same iteration k that a Ritz pair $(\theta_i^k, \mathbf{V}_k \mathbf{y}_k^i)$ converges. This can happen at the same iteration that CG converges, but it can also happen sooner. The upside is that by the time CG converges, all Lanczos eigenpair estimates have already converged, and so $\mathbf{B}_k$ is a good approximation to $\mathbf{A}^{-1}$ (and $\mathbf{B}_k^{\dagger}$ approximates $\mathbf{A}$) in the corresponding eigenspaces.

The ramifications for CG-KF and CG-VKF are that, without corrective measures, the covariance approximations are guaranteed to be accurate only in a small k-dimensional eigenspace of $\mathbf{A}$ (see

discussion in Section 3.4), where k is the iteration at which loss of conjugacy of the search directions occurs. In order to attain more accurate covariance approximations, corrective measures such as reorthogonalization [26, 32, 36], which is commonly used with Lanczos schemes, must be implemented. Depending on the distribution of the eigenvalues of $\mathbf{A}$, the expense in computational time and memory requirements can be as large as a Cholesky factorization, which, for the large-scale problems we consider, is prohibitive.

4. NUMERICAL EXAMPLES

In this section, we test CG-KF and CG-VKF on two examples.

4.1. An example with a large-scale linear evolution model

The first example is large dimensional and linear. We consider the following forced heat equation model

$$\frac{\partial x}{\partial t} = -\frac{\partial^2 x}{\partial u^2} - \frac{\partial^2 x}{\partial v^2} + \delta \exp\left[-\frac{(u-2/9)^2 + (v-2/9)^2}{\sigma^2}\right], \tag{16}$$

where x is a function of u and v over the domain $\Omega = \{(u, v) \mid 0 \leqslant u, v \leqslant 1\}$ and $\delta \geqslant 0$. We generate synthetic data using (16) with $\delta > 0$ and assume that the evolution model is given by (16) with $\delta = 0$, which gives a model bias. The problem can be made arbitrarily large scale via a sufficiently fine spatial discretization. However, the well-behaved nature of solutions of (16) calls for further experiments with a different test case, hence our second example in the next subsection.

We discretize the model (16) using a uniform $N \times N$ computational grid and the standard finite difference schemes of both the time and spatial derivatives. This gives the 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 (16) at each of the points of the computational grid. We define the observation matrix as $\mathbf{K}_k = \mathbf{K}$ for all k in (2), where $\mathbf{K}$ is the full weighting matrix that 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 and equally spaced at $n^2/64$ locations.

We first generate synthetic data using the stochastic equations

$$\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{f} + \varepsilon_k^p, \tag{17}$$

$$\mathbf{y}_{k+1} = \mathbf{K}\mathbf{x}_{k+1} + \varepsilon_k^o, \tag{18}$$

with $\varepsilon_k^o \sim N(\mathbf{0}, (0.8\sigma_{\text{obs}})^2\mathbf{I})$, $\varepsilon_k^p \sim N(\mathbf{0}, (0.5\sigma_{\text{ev}})^2\mathbf{I})$, $\delta = 3/4$ in (16), and where σ_{obs}^2 and σ_{ev}^2 are chosen so that the signal-to-noise ratios $\|\mathbf{K}\mathbf{x}_0\|^2/n^2\sigma_{\text{obs}}^2$ and $\|\mathbf{x}_0\|^2/n^2\sigma_{\text{ev}}^2$ are both 50. The initial condition used for the data generation is

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

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

For the implementation of our filtering algorithms, we assume the model

$$\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \varepsilon_k^p,$$
$$\mathbf{y}_{k+1} = \mathbf{K}\mathbf{x}_{k+1} + \varepsilon_k^o,$$

with $\varepsilon_k^p \sim N(\mathbf{0}, \sigma_{ev}^2 \mathbf{I})$, $\varepsilon_k^o \sim N(\mathbf{0}, \sigma_{obs}^2 \mathbf{I})$, and where $\mathbf{x}_0^{est} = \mathbf{0}$ and $\mathbf{C}_0^{est} = \mathbf{0}$ in Step 1 of the filter. Notice that the forcing function $\mathbf{f}$ is not contained in the evolution model, which adds a bias.

In our first example, we assume a 32×32 computational grid. For all implementations of CG within CG-KF and CG-VKF, the zero vector was the initial guess, and a stopping tolerance of $\|\mathbf{r}_k\| < 10^{-6}$ was used to signal CG convergence, where $\mathbf{r}_k$ is the residual at iteration k. The maximum number of CG iterations was set to 40.

In Figure 1, the root mean square errors $\sqrt{1/N \|\mathbf{x}_k - \mathbf{x}_k^{true}\|}$ are given for different methods. We compare KF, CG-KF, CG-VKF, and the algorithms LBFGS-KF from [21] and LBFGS-VKF from [22], which are the same as CG-KF and CG-VKF, except that instead of CG, the LBFGS method is used for quadratic optimization, and the LBFGS Hessian and inverse Hessian approximations are used for the approximation of the covariance/precision matrices. For LBFGS implementations, the same stopping tolerance and maximum number of iterations were used, and all vectors were stored for the covariance/precision matrix approximations.

The approximate methods produce results that are comparable with KF, with less computational cost (roughly five times faster in this case). The methods perform in a similar way, CG-VKF being slightly better and LBFGS-KF converging a bit slower than the others. The computational benefit of CG-VKF over the other methods is that only one CG optimization is required, because the matrix inversion lemma (10) is used for defining $(\mathbf{C}_k^p)^{-1}$.

As a second example, we run a much higher dimensional case with a 128×128 computational grid. In this case, the memory of a standard desktop computer is not enough to run KF. In Figure 2, we compare the CG and LBFGS implementations of the approximate filters. CG-based methods perform better in this test, and CG-VKF is, again, better than the others.

4.2. An Example with a small-scale, nonlinear evolution model

Our second example produces chaotic, unpredictable behavior but is not large scale. We consider the nonlinear Lorenz 1995 model introduced and analyzed in [41,42], given by

$$\frac{\partial x^i}{\partial t} = (x^{i+1} - x^{i-2})x^{i-1} - x^i + 8, \qquad i = 1, 2, \ldots, 40, \tag{19}$$

with periodic state space variables, that is, $x^{-1} = x^{n-1}$, $x^0 = x^n$ and $x^{n+1} = x^1$. In the present tests, we use the dimension $n = 40$. The model shares many characteristics with realistic atmospheric models (cf. [42]) and is often used as a test case for various weather forecasting schemes.

Next, we apply the filtering methods to the problem of estimating the state variables from data generated using the nonlinear, chaotic evolution model (19). The data was generated by integrating

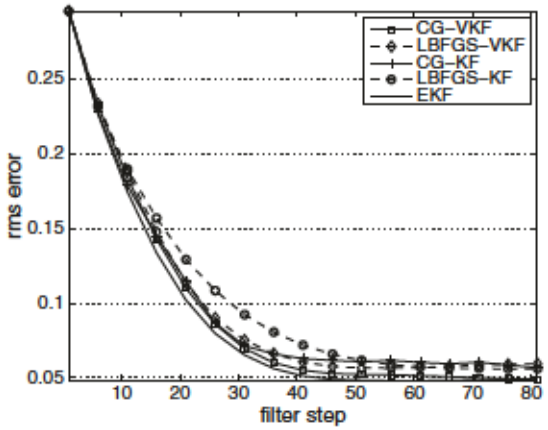

Figure 1. Root mean square error versus iteration for EKF, CG-KF, CG-VKF, LBFGS-KF, and LBFGS-VKF on a 32×32 computational grid.

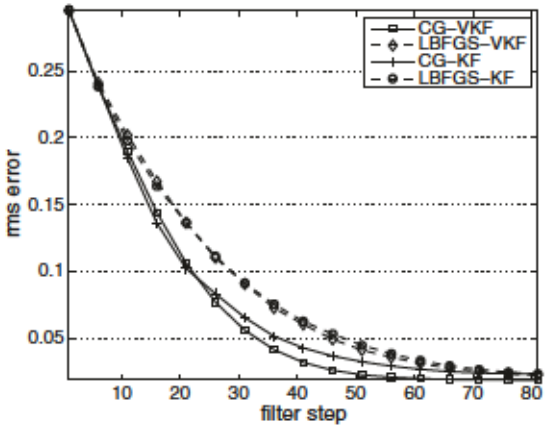

Figure 2. Root mean square error versus iteration for CG-KF, CG-VKF, LBFGS-KF, and LBFGS-VKF on a 128×128 computational grid.

the model using a fourth-order Runge–Kutta (RK4) method with time step $\Delta t = 0.025$. The discussion in [42] suggests that when using (19) as a test example for weather forecasting algorithms, the characteristic time scale is such that the mentioned Δt corresponds to 3 h. The 'truth' was generated by taking 42,920 time steps of the RK4 method, that is, 5365 days. The initial state for 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 three grid points in each set of five; that is, the observation matrix is $m \times n$, 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 Gaussian noise $N(\mathbf{0}, (0.15\,\sigma_{\text{clim}})^2\mathbf{I})$ where $\sigma_{\text{clim}} = 3.6414723$ is a standard deviation of the model state used in climatological simulations. The data generation codes, which were those used in the papers [21, 22] and were originally transcribed by us from the `scilab` codes written by the author of [43], are written in MATLAB (The MathWorks, Inc.).

For the application of EKF and VKF, we employ the coupled system

$$\mathbf{x}_{k+1} = \mathcal{M}(\mathbf{x}_k) + \varepsilon_k^p, \tag{20}$$

$$\mathbf{y}_{k+1} = \mathbf{K}\mathbf{x}_{k+1} + \varepsilon_k^o, \tag{21}$$

with $\varepsilon_k^p \sim N(\mathbf{0}, (0.05\,\sigma_{\text{clim}})^2\mathbf{I})$ and $\varepsilon_k^o \sim N(\mathbf{0}, (\sigma_{\text{clim}})^2\mathbf{I})$ and where $\mathcal{M}(\mathbf{x}_k)$ is obtained by taking two steps of the RK4 method applied to (19) from $\mathbf{x}_k$ with time step 0.025. Because $\mathcal{M}$ is a nonlinear function, EKF must be used. Because $\mathcal{K} := \mathbf{K}$ in (7) is linear, $\mathbf{K}_k = \mathbf{K}$ for all k in (8). However, a linearization of the nonlinear evolution function $\mathcal{M}$ is required. The computation of the tangent linear model $\mathbf{M}_k$ in (8) is performed by a routine in one of the `scilab` codes mentioned earlier, adopted for our use in MATLAB.

The initial guesses for the kF iterations were $\mathbf{x}_0^{\text{est}} = \mathbf{1}$ and $\mathbf{C}_0^{\text{est}} = \mathbf{I}$. In all implementations of CG within CG-KF and CG-VKF, iterations were stopped once $\|\mathbf{r}_k\| < 10^{-6}$, signaling CG convergence, where $\mathbf{r}_k$ is the residual at iteration k. The maximum number of CG iterations was set to 50. As in the previous example, the results were compared with the LBFGS versions [21, 22], and the same optimization settings were used for LBFGS iterations.

The results are given in Figures 3–5. CG-KF and LBFGS-KF are compared in terms of root mean square error in Figure 3, whereas CG-VKF and LBFGS-VKF are compared in Figure 4. In Figure 5, a comparison of the *forecast skill* of the methods is given, where forecast skill is defined as the mean

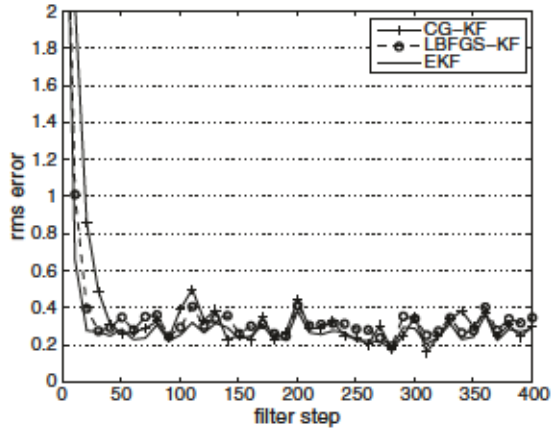

Figure 3. Root mean square error versus iteration for EKF, CG-KF, and LBFGS-KF.

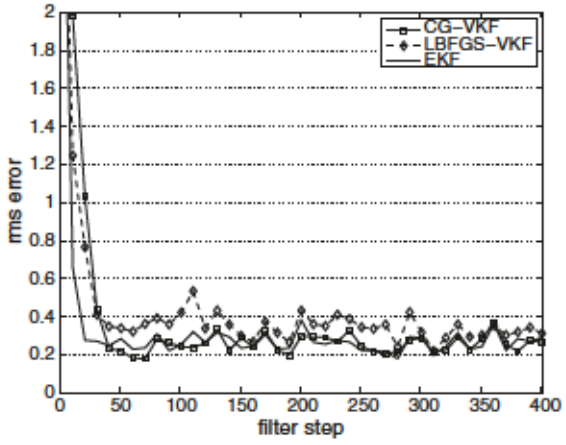

Figure 4. Root mean square error versus iteration for EKF, CG-VKF, and LBFGS-VKF.

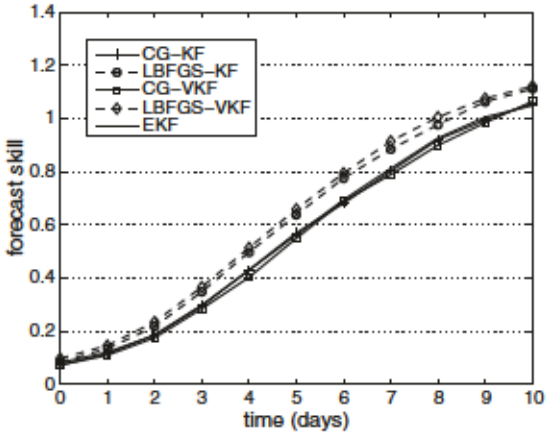

Figure 5. Forecast root mean square error versus iteration for EKF, CG-KF, CG-VKF, LBFGS-KF and LBFGS-VKF.

squared difference between the 'truth' and the forecast made with the model, scaled with σ_{clim} (see [21, 22] for details). The forecasts are started at every fourth filter step, starting from the 64th step (when all filters have converged). All methods track quite closely with the KF estimates after a sufficient number of iterations. However, CG implementations work better than the LBFGS versions, which is especially visible in the forecast skill in Figure 5.

5. CONCLUSIONS

For large-scale examples, such as arise in weather forecasting and oceanography, the Kalman filter can be prohibitively expensive to implement because it requires the solution of large linear systems and the storage of large, dense matrices. In this paper, we show how the conjugate gradient (CG) algorithm can be used for both the approximate solution of large linear systems, as well as for obtaining low-rank and low-storage approximations of matrices within KF.

Conjugate gradient is a standard iterative method for solving large, symmetric positive-definite linear systems. CG iteration history can also be used to obtain approximations of the coefficient matrix and its inverse. In p CG iterations, this inverse approximation will have rank p and requires the storage of only p n-vectors. To obtain the approximation of the coefficient matrix itself, the $p \times p$ tridiagonal Lanczos matrix $\mathbf{T}_k$ must be diagonalized, which can be efficiently done for p of small-to-moderate size. Multiplication by these approximate matrices is also efficient. We make use of these facts to define our approximate KF algorithms: CG-KF and CG-VKF.

More specifically, our implementation of CG-VKF made use of the CG approximation (11) once, whereas CG-KF made use of both the CG approximation (11) and the Lanczos approximation (13). The equivalence of (11) and (13) was made explicit and was used both in our numerical calculations of Lanczos from CG, as well as in the brief theoretical analysis of the covariance approximations contained in Section 3.4.

The resulting algorithms, which we have denoted the CG-KF and CG-VKF, are much more efficient to implement than KF for the two numerical examples considered here—one large scale and linear and the other medium scale and nonlinear—and provide results that are comparable. Moreover, in all of our tests, the CG-based approximate Kalman filters outperform the analogous limited memory BFGS (LBFGS) approximate Kalman filters, introduced in [21, 22].

ACKNOWLEDGEMENTS

This work was supported by the NSF under grant DMS-0915107.

REFERENCES

1. Kalman RE. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering, Series D* 1960; **82**:35–45.
2. Welch G, Bishop G. An introduction to the Kalman filter. *Technical Report 95-041*, University of North Carolina at Chapel Hill, Department of Computer Science, 1995.
3. Cane MA, Miller RN, Tang B, Hackert E, Busalacchi AJ. Mapping tropical pacific sea level: data assimilation via reduced state Kalman filter. *Journal of Geophysical Research* 1996; **101**:599–617.
4. Dee DP. Simplification of the Kalman filter for meteorological data assimilation. *Quarterly Journal of the Royal Meterological Society* 1990; **117**:365–384.
5. Fisher M. Development of a simplified Kalman filter. *Technical Memorandum 260*, European Center for Medium Range Weather Forecasting, 1998.
6. Gejadze IY, Dimet FXL, Shutyaev V. On analysis error covariances in variational data assimilation. *SIAM Journal on Scientific Computing* 2008; **30**:1847–1874.
7. Tian X, Xie Z, Dai A. An ensemble-based explicit four-dimensional variational assimilation method. *Journal of Geophysical Research* 2008; **113 D21124**:1–13.
8. Veersé F, Auroux D, Fisher M. Limited-memory BFGS diagonal predonditioners for a data assimilation problem in meteorology. *Optimization and Engineering* 2000; **1**:323–339.
9. Fisher M, Andersson E. Developments in 4D-Var and Kalman filtering. *Technical Memorandum 347*, European Center for Medium Range Weather Forecasting, 2001.
10. Lorenc AC. Modelling of error covariances by 4D-Var data assimilation. *Quarterly Journal of the Royal Meterological Society* 2003; **129**:3167–3182.

11. Rabier F, Järvinen H, Klinker E, Mahfouf J, Simmons A. The ECMWF operational implementation of four-dimensional variational assimilation. part 1: experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society* 2000; **126**:1143–1170.
12. Fisher M, Leutbecher M, Kelley G. On the equivalence between Kalman smoothing and weak-constraint four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meterological Society* 2005; **131**: 3235–3246.
13. LeDimet FX, Talagrand O. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus Series A* 1986; **38**:97–110.
14. Li Z, Navon M. Optimality of variational data assimilation and its relationship with the Kalman filter and smoother. *Quarterly Journal of the Royal Meterological Society* 2008; **127**:661–683.
15. Cao Y, Zhu J, Navon IM, Luo Z. A reduced order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids* 2007; **53**(10):1571–1583.
16. Fisher M. Minimization algorithms for variational data assimilation. *Proceedings of the ECMWF Seminar on Recent Developments in Numerical Methods for Atmospheric Modelling*, Reading, England, 1998; 364–385.
17. Fisher M, Nocedal J, Trémolet Y, Wright SJ. Data assimilation in weather forecasting: a case study in pre-constrained optimization. *Optimization and Engineering* 2009; **10**(3):1389–1420.
18. Nocedal J, Wright SJ. *Numerical Optimization*. Springer: New York, 2000.
19. Tshimanga J, Gratton S, Weaver AT, Sartenaer A. Limited-memory preconditioners, with application to incremental four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society* 2008; **134**:751–769.
20. Yang PCW, Navon IM. A new Hessian preconditioning method applied to variational data assimilation experiments using NASA general circulation models. *Monthly Weather Review* 1996; **124**:1000–1017.
21. Auvinen H, Bardsley JM, Haario H, Kauranne T. Large-scale Kalman filtering using the limited memory BFGS method. *Electronic Transactions in Numerical Analysis* 2010; **35**(9):217–233.
22. Auvinen H, Bardsley JM, Haario H, Kauranne T. The variational Kalman filter and an efficient implementation using limited memory BFGS. *International Journal for Numerical Methods in Fluids* 2010; **64**(3):314–335.
23. Nocedal J. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation* 1980; **35**(151): 773–782.
24. Veersé F. Variable-storage quasi-Newton operators for modeling error covariances. *Proceedings of the Third WMO International Symposium on Assimilation of Observations in Meteorology and Oceanography*, Québec City, Canada, 1999.
25. Schneider MK, Willsky AS. Krylov subspace estimation. *SIAM Journal on Scientific Computing* 2000; **22**(5): 1840–1864.
26. Schneider MK, Willsky AS. A Krylov subspace method for covariance approximation and simulation of random processes and fields. *Multidimensional Systems and Signal Processing* 2003; **14**(4):295–318.
27. Schneider MK, Willsky AS. Krylov subspace algorithms for space-time oceanography data assimilation. *IEEE International Geoscience and Remote Sensing Symposium,* Vol. 2, Honolulu, HI, USA, 2000; 727–729.
28. Watkins D. *Fundamentals of Matrix Computations*, 2nd edn. Wiley: New York, 2002.
29. Hutchinson MF. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics, Simulation and Computation* 1989; **19**(2):433450.
30. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, England, 1996.
31. Golub GH, Loan CFV. *Matrix Computations*, 3rd edn. The Johns Hopkins University Press: Baltimore, MD, USA, 1996.
32. Meurant G. *The Lanzcos and Conjugate Gradient Algorithms*. SIAM: Philadelphia, PA, USA, 2006.
33. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**:409–436.
34. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards* 1950; **45**:255–282.
35. Lanczos C. Solutions of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards* 1952; **49**:33–53.
36. Parlett B. *Symmetric Eigenvalue Problem*. Prentice Hall, 1980.
37. Saad Y. *Numerical Methods for Large Eigenvalue Problems*. SIAM: Philadelphia, PA, USA, 2011.
38. Cullum JK, Willoughby RA. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations Volume 1: Theory*. SIAM: Philadelphia, PA, USA, 2002.
39. Saad Y. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis* 1992; **29**:209–228.
40. Sleijpen G, Sluis AVD. Further results on the convergence behavior of conjugate gradients and Ritz values. *Linear Algebra and Its Applications* 1996; **246**:23–278.
41. Lorenz EN. Predictability: a problem partly solved. *Proceedings of the Seminar on Predicability,* European Center for Medium Range Weather Forecasting, Vol. 1, 1996; 1–18.
42. Lorenz EN, Emanuel KA. Optimal sites for supplementary weather observations: simulation with a small model. *Journal of Atmospheric Science* 1998; **55**:399–414.
43. Leutbecher M. A data assimilation tutorial based on the Lorenz-95 system, European Centre for Medium-Range Weather Forecasting, Web Tutorial, www.ecmwf.int/newsevents/training/lecture notes/pdf files/ASSIM/Tutorial.pdf.