Electronic Transactions on Numerical Analysis.
Volume 39, pp. 271-285, 2012.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

VARIATIONAL ENSEMBLE KALMAN FILTERING USING LIMITED MEMORY BFGS*

ANTTI SOLONEN[†‡], HEIKKI HAARIO[‡], JANNE HAKKARAINEN[†], HARRI AUVINEN[‡], IDRISSA AMOUR[‡], AND TUOMO KAURANNE[‡]

Abstract. The extended Kalman filter (EKF) is one of the most used nonlinear state estimation methods. However, in large-scale problems, the CPU and memory requirements of EKF are prohibitively large. Recently, Auvinen et al. proposed a promising approximation to EKF called the variational Kalman filter (VKF). The implementation of VKF requires the tangent linear and adjoint codes for propagating error covariances in time. However, the trouble of building the codes can be circumvented by using ensemble filtering techniques, where an ensemble of states is propagated in time using the full nonlinear model, and the statistical information needed in EKF formulas is estimated from the ensemble. In this paper, we show how the VKF ideas can be used in the ensemble filtering context. Following VKF, we obtain the state estimate and its covariance by solving a minimization problem using the limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) method, which provides low-storage approximations to the state covariances. The resulting hybrid method, the variational ensemble Kalman filter (VEnKF), has several attractive features compared to existing ensemble methods. The model error and observation error covariances can be inserted directly into the minimization problem instead of randomly perturbing model states and observations as in the standard ensemble Kalman filter. New ensembles can be directly generated from the LBFGS covariance approximation without the need of a square root (Cholesky) matrix decomposition. The frequent resampling from the full state space circumvents the problem of ensemble in-breeding frequently associated with ensemble filters. Numerical examples are used to show that the proposed approach performs better than the standard ensemble Kalman filter, especially when the ensemble size is small.

Key words. Data assimilation, state estimation, Kalman filtering, ensemble filtering, LBFGS

AMS subject classifications. 65K10, 15A29

1. Introduction.

Since the introduction of the Kalman filter (KF) in [21] and its nonlinear extension, the extended Kalman filter (EKF), many approaches to overcome the problems with computational complexity present in KF and EKF have been proposed; see, e.g., [6, 9, 13]. In these approaches, the state vector is often projected onto a fixed, low-dimensional subspace. It is known that a fixed projection operator might not correctly capture the dynamics of a nonlinear system; see [14]. In particular, such reduced rank Kalman filters tend to suffer from covariance leakage; see [20].

In [2, 3], a low-storage variational approach to approximate KF and EKF was proposed called the variational Kalman filter (VKF). In VKF, the large matrices in KF formulas are replaced with a low-storage approximation provided by the quasi-Newton optimization method called limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS). The applicability of VKF is hindered by the requirement of tangent linear and adjoint codes for the evolution model, which require a considerable development effort separately for every model.

In ensemble filtering methods, the problems related to standard EKF (large matrices, need for tangent linear and adjoint codes) are circumvented by representing uncertainty in the model state as a number of samples instead of covariance matrices. Instead of moving the covariance in time using the linearized model, uncertainty is propagated simply by moving the ensemble members in time with the full nonlinear evolution model. The simplest version of this idea is the ensemble Kalman filter (EnKF), first proposed in [11], where the covariance matrices in the KF formulas are essentially replaced with sample statistics calculated from

* Received January 24, 2011. Accepted January 22, 2012. Published online on July 9, 2012. Recommended by P. Van Dooren.

[†]Lappeenranta University of Technology, P.O. Box 20, FI-53851 Lappeenranta, Finland (antti.solonen@gmail.com).

[‡]Finnish Meteorological Institute, P.O. Box 503, FI-00101 Helsinki, Finland.

the ensemble. However, EnKF suffers from some problems, e.g., sampling errors due to random perturbation of model state and observations and from ensemble in-breeding that results in a similar covariance leakage as that associated with reduced rank Kalman filters; see [23, 26, 28].

In this paper, we show how VKF ideas can be used in the ensemble filtering context to overcome some problems related to existing ensemble methods. In our approach, the model error and observation error covariances are inserted directly into the minimization problem instead of randomly perturbing model states and observations as in EnKF. New ensembles can be efficiently generated directly from the LBFGS covariance approximation without explicitly constructing the large covariance matrix. We call our hybrid approach the *variational ensemble Kalman filter (VEnKF)* and show by numerical examples that the hybrid method performs well compared to the standard EnKF, especially when the ensemble size is small.

The paper is organized as follows. In Section 2, we recall the basics of Kalman filtering and ensemble methods. We introduce the VEnKF algorithm in Section 3 and demonstrate its performance with numerical examples in Section 4. In Section 5, we discuss some specific topics related to our approach and the differences to existing ensemble filters. Section 6 concludes the paper.

2. Filtering methods.

In this section, we provide an overview of some existing Kalman filtering methods that are related to our approach. We start by recalling how the basic Kalman filter and some of its variants work and continue with an introduction to ensemble filtering methods.

2.1. Kalman filtering and variants.

The Kalman filter can be used to estimate the state $\mathbf{x}_k$ at discrete times k from observations $\mathbf{y}_k$, when the model and observation equations are linear:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{M}_k \mathbf{x}_{k-1} + \varepsilon_k^p \\ \mathbf{y}_k &= \mathbf{K}_k \mathbf{x}_k + \varepsilon_k^o. \end{aligned}$$

In the above system, $\mathbf{M}_k$ is the $d \times d$ evolution model and $\mathbf{K}_k$ is the $m \times d$ observation operator. The $d \times 1$ vector $\mathbf{x}_k$ represents the model state, and the observed data are denoted by the $m \times 1$ vector $\mathbf{y}_k$. The model error ε_k^p and the observation error ε_k^o are assumed to be normally distributed zero mean random variables with covariance matrices $\mathbf{C}_{\varepsilon_k^p}$ and $\mathbf{C}_{\varepsilon_k^o}$, respectively. The Kalman filter algorithm for estimating states and their error covariances can be written as follows.

The Kalman filter algorithm

1. Move the state estimate and covariance in time:
   (a) Compute $\mathbf{x}_k^p = \mathbf{M}_k \mathbf{x}_{k-1}^{\mathrm{est}}$.
   (b) Compute $\mathbf{C}_k^p = \mathbf{M}_k \mathbf{C}_{k-1}^{\mathrm{est}} \mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p}$.
2. Combine the prior with observations:
   (a) Compute the Kalman gain $\mathbf{G}_k = \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} (\mathbf{K}_k \mathbf{C}_k^p \mathbf{K}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^o})^{-1}$.
   (b) Compute the state estimate $\mathbf{x}_k^{\mathrm{est}} = \mathbf{x}_k^p + \mathbf{G}_k(\mathbf{y}_k - \mathbf{K}_k \mathbf{x}_k^p)$.
   (c) Compute the covariance estimate $\mathbf{C}_k^{\mathrm{est}} = \mathbf{C}_k^p - \mathbf{G}_k \mathbf{K}_k \mathbf{C}_k^p$.
3. Set $k \to k+1$ and go to step 1.

The extended Kalman filter directly uses the Kalman filter formulas in the nonlinear case by replacing the nonlinear model and observation operators with appropriate linearizations: $\mathbf{M}_k = \frac{\partial}{\partial \mathbf{x}} \mathcal{M}(\mathbf{x}_{k-1}^{\mathrm{est}})$ and $\mathbf{K}_k = \frac{\partial}{\partial \mathbf{x}} \mathcal{K}(\mathbf{x}_k^p)$.

In the variational formulation of the Kalman filter, the state estimation at step k is viewed as an optimization problem, where a quadratic function

$$l(\mathbf{x}|\mathbf{y}_k) = \frac{1}{2}\left(\mathbf{x}-\mathbf{x}_k^p\right)^{\mathrm{T}}\left(\mathbf{C}_k^p\right)^{-1}\left(\mathbf{x}-\mathbf{x}_k^p\right)+\frac{1}{2}\left(\mathbf{y}_k-\mathcal{K}(\mathbf{x})\right)^{\mathrm{T}}\left(\mathbf{C}_{\varepsilon_k^o}\right)^{-1}\left(\mathbf{y}_k-\mathcal{K}(\mathbf{x})\right) \tag{2.1}$$

is minimized with respect to $\mathbf{x}$. In the VKF method, introduced in [2], the minimization is done with the LBFGS optimization method, that produces both the state estimate and a low-storage approximation of the covariance (inverse hessian at the minimizer). In the VKF algorithm, the inverse of the prior covariance $\mathbf{C}_k^p$ is also approximated using LBFGS by setting up an auxiliary optimization problem

$$\underset{\mathbf{u}}{\operatorname{argmin}} \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{C}_k^p\mathbf{u}. \tag{2.2}$$

Thus, the LBFGS optimization routine provides low-storage approximation for both $(\mathbf{C}_k^p)^{-1}$ and $\mathbf{C}_k^{\mathrm{est}}$. All computations with the covariances can be carried out efficiently using the implicit low-storage representation without forming the full matrices. The VKF method is given as an algorithm below.

The variational Kalman filter algorithm

1. Move the state estimate and covariance in time:
   (a) Compute $\mathbf{x}_k^p = \mathbf{M}_k\mathbf{x}_{k-1}^{\mathrm{est}}$.
   (b) Define $\mathbf{C}_k^p = \mathbf{M}_k\mathbf{B}_{k-1}^{\#}\mathbf{M}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p}$.
   (c) Apply LBFGS to (2.2) to get an approximation $\mathbf{B}_k^*$ of $(\mathbf{C}_k^p)^{-1}$.
2. Combine the prior with observations:
   (a) Minimize expression (2.1) using LBFGS to get the state estimate $\mathbf{x}_k^{\mathrm{est}}$ and covariance estimate $\mathbf{B}_k^{\#}$.
3. Set $k \to k+1$ and go to step 1.

Note that while VKF can solve the storage problem related to EKF, it requires a way to evolve the covariance in time (step 1(b) in the algorithm above). Propagating the covariance using a direct linearization, as in EKF, is infeasible in high dimensions. In VKF, covariance propagation is done using tangent linear and adjoint codes that implement differentiation at the “code level”. This is a standard technique in variational data assimilation; see, e.g., [15, 22]. These codes must be prepared separately for every model and their construction is laborious, although automatic code generators have been recently developed; see, e.g., [8]. In the ensemble filters that we discuss next, tangent linear and adjoint codes are not needed.

2.2. Ensemble filtering. In ensemble filtering, the uncertainty in the state estimate $\mathbf{x}_k$ is represented as N samples, here denoted as $\mathbf{s}_k = (\mathbf{s}_{k,1}, \mathbf{s}_{k,2}, \ldots, \mathbf{s}_{k,N})$, instead of a covariance matrix. The first ensemble filtering method was the ensemble Kalman filter (EnKF) introduced in [11] and implemented in operational numerical weather prediction, e.g., in [19]. The ensemble Kalman filter essentially replaces the state covariance matrices in EKF with the sample covariance calculated from the ensemble. The sample covariance can be written as $\mathrm{Cov}(\mathbf{s}_k) = \mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}$, where

$$\mathbf{X}_k = \frac{1}{\sqrt{N-1}}\left((\mathbf{s}_{k,1}-\overline{\mathbf{s}_k}), (\mathbf{s}_{k,2}-\overline{\mathbf{s}_k}), \ldots, (\mathbf{s}_{k,N}-\overline{\mathbf{s}_k})\right).$$

The sample mean is denoted by $\overline{\mathbf{s}_k}$. Using our notation, the EnKF algorithm can be formulated as follows.

The ensemble Kalman filter algorithm

1. Move the state estimate and covariance in time:
   (a) Move ensemble forward and perturb members with model error:
   $\mathbf{s}^p_{k,i} = \mathcal{M}(\mathbf{s}^{\mathrm{est}}_{(k-1),i}) + \mathbf{e}^p_{k,i}, \; i = 1, \ldots, N.$
   (b) Calculate sample mean $\overline{\mathbf{s}_k}$ and covariance $\mathbf{C}^p_k = \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}$.
2. Combine the prior with observations:
   (a) Compute the Kalman gain $\mathbf{G}_k$.
   (b) Update ensemble members $\mathbf{s}^{\mathrm{est}}_{k,i} = \mathbf{s}^p_{k,i} + \mathbf{G}_k(\mathbf{y}_k - \mathbf{K}_k \mathbf{s}^p_{k,i} + \mathbf{e}^o_{k,i})$.
   (c) Calculate state estimate as the sample mean $\overline{\mathbf{s}^{\mathrm{est}}_{k,i}}$.

In the above algorithm, the vectors $\mathbf{e}^p_{k,i}$ and $\mathbf{e}^o_{k,i}$ are realizations of the model error and observation error distributions (Gaussians with covariances $\mathbf{C}_{\varepsilon^p_k}$ and $\mathbf{C}_{\varepsilon^o_k}$, respectively).

The ensemble Kalman filter is very simple to implement and it does not require tangent linear and adjoint codes. However, EnKF has various problems and numerous variants have been developed to overcome these issues; see, e.g., [1, 10, 19, 32]. In Section 5, we discuss these variants in light of the proposed VEnKF algorithm, which we introduce in the next section.

3. Variational ensemble Kalman filtering. Here we follow the VKF ideas and show how they can be implemented in the ensemble filtering context, resulting in a filter that we call the variational ensemble Kalman filter (VEnKF). As in VKF, the state estimation in VEnKF is based on minimizing the cost function in Equation (2.1). The prior covariance needed in the cost function is defined here as

$$\mathbf{C}^p_k = \mathrm{Cov}\left(\mathcal{M}(\mathbf{x}^{\mathrm{est}}_{k-1}) + \varepsilon^p_k\right) = \mathrm{Cov}\left(\mathcal{M}(\mathbf{x}^{\mathrm{est}}_{k-1})\right) + \mathrm{Cov}\left(\varepsilon^p_k\right) \approx \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon^p_k}. \tag{3.1}$$

Note that the above formula contains the common assumption that the model error and model response are uncorrelated. The same assumption is made in KF and EKF. In VEnKF, we calculate the sample covariance using the state estimate evolved from the previous time as the expectation instead of the sample mean used in EnKF. Thus, we define

$$\mathbf{X}_k = \frac{1}{\sqrt{N}}\left((\mathbf{s}_{k,1} - \mathbf{x}^p_k), (\mathbf{s}_{k,2} - \mathbf{x}^p_k), \ldots, (\mathbf{s}_{k,N} - \mathbf{x}^p_k)\right),$$

where $\mathbf{x}^p_k = \mathcal{M}(\mathbf{x}^{\mathrm{est}}_{k-1})$ and $\mathbf{s}_{k,i} = \mathcal{M}(\mathbf{s}^{\mathrm{est}}_{(k-1),i})$. Note that the ensemble members now do not contain random perturbations; the model error is included directly in Equation (3.1).

The inverse of the prior covariance $\mathbf{C}^p_k = \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon^p_k}$, needed when evaluating the cost function (2.1), can be obtained in two ways. Following the VKF derivation, we can approximate the inverse by applying LBFGS to the artificial optimization problem

$$\operatorname*{argmin}_{\mathbf{u}} \mathbf{u}^{\mathrm{T}}(\mathbf{X}_k \mathbf{X}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon^p_k})\mathbf{u}. \tag{3.2}$$

The sample covariance matrix in the above expression naturally does not have to be handled as a full matrix – in order to evaluate the cost function, we can just keep the covariance in "ensemble form" $\mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}$ and evaluate the cost function in the form $\mathbf{u}^{\mathrm{T}} \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}} \mathbf{u} + \mathbf{u}^{\mathrm{T}} \mathbf{C}_{\varepsilon^p_k} \mathbf{u}$. For the computation to remain efficient, we assume (as in VKF) that the model error covariance $\mathbf{C}_{\varepsilon^p_k}$ can be efficiently multiplied with a vector, which is the case, for instance, if the covariance is assumed to be diagonal. As a result of the above optimization, we obtain an LBFGS representation of the inverse of the prior covariance $(\mathbf{C}^p_k)^{-1}$. We can use the LBFGS representation to evaluate the first term when optimizing the cost function (2.1). For computing the matrix-vector product when the matrix is in the LBFGS form, there exists an efficient recursive algorithm; see Appendix A and [27] for details.

ETNA
Kent State University
http://etna.math.kent.edu

An attractive alternative way to proceed is to calculate the inverse of the prior covariance using the Sherman-Morrison-Woodbury (SMW) matrix inversion formula; see, e.g., [18]. The inverse of the prior covariance can be written as

$$
\begin{aligned}
(\mathbf{C}_k^p)^{-1} &= (\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \mathbf{C}_{\varepsilon_k^p})^{-1} \\
&= \mathbf{C}_{\varepsilon_k^p}^{-1} - \mathbf{C}_{\varepsilon_k^p}^{-1}\mathbf{X}_k(\mathbf{I} + \mathbf{X}_k^{\mathrm{T}}\mathbf{C}_{\varepsilon_k^p}^{-1}\mathbf{X}_k)^{-1}\mathbf{X}_k^{\mathrm{T}}\mathbf{C}_{\varepsilon_k^p}^{-1}.
\end{aligned}
\tag{3.3}
$$

This representation of the inverse can be directly inserted into Equation (2.1) when it it minimized. The computation of the quadratic expression $(\mathbf{x} - \mathbf{x}_k^p)^{\mathrm{T}}(\mathbf{C}_k^p)^{-1}(\mathbf{x} - \mathbf{x}_k^p)$ can be organized so that we do not have to store full matrices of size $d \times d$. With this formulation, the first LBFGS approximation can be avoided and the prior can be included 'exactly' in the second optimization. The application of the formula requires that the inverses of the model error covariances $\mathbf{C}_{\varepsilon_k^p}$ are available. If the model error is assumed to be constant (same for all k), this matrix inversion needs to be computed only once. In addition, the inversion of $\mathbf{I} + \mathbf{X}_k^{\mathrm{T}}\mathbf{C}_{\varepsilon_k^p}^{-1}\mathbf{X}_k$ needs to be computed at every step. However, this matrix is only of size $N \times N$, where N is the ensemble size, which is always very small compared to the dimension of the state space in large-scale applications. In the examples of this paper, we use the SMW formula for inverting the prior covariance.

When the LBFGS optimization is applied to minimize the function (2.1), we get a low-storage approximation for the covariance $\mathbf{C}_k^{\mathrm{est}}$. After that, we sample a new ensemble of state vectors from $\mathrm{N}(\mathbf{x}_k^{\mathrm{est}}, \mathbf{C}_k^{\mathrm{est}})$. Samples can be drawn efficiently, since the LBFGS representation for $\mathbf{C}_k^{\mathrm{est}}$ can be written in the form

$$
\mathbf{C}_k^{\mathrm{est}} = \mathbf{B}_0\mathbf{B}_0^{\mathrm{T}} + \sum_{i=1}^{n} \mathbf{b}_i\mathbf{b}_i^{\mathrm{T}},
$$

where $\mathbf{B}_0$ is a $d \times d$ matrix and $\mathbf{b}_i$ are $d \times 1$ vectors. From this representation one can produce a zero mean random vector $\mathbf{r} \sim N(\mathbf{0}, \mathbf{C}_k^{\mathrm{est}})$ simply by calculating

$$
\mathbf{r} = \mathbf{B}_0\mathbf{z} + \sum_{i=1}^{n} \omega_i\mathbf{b}_i,
$$

where $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ and $\omega_i \sim N(0, 1)$. The matrix $\mathbf{B}_0$ does not have to be stored explicitly, since the product $\mathbf{B}_0\mathbf{z}$ can be computed implicitly using the stored LBFGS vectors. See Appendix A for details about constructing $\mathbf{B}_0$ and $\mathbf{b}_i$.

Finally, we are ready to present our *variational ensemble Kalman filter* (VEnKF) as an algorithm. After setting the initial guesses for the state and its covariance to $\mathbf{x}_0^{\mathrm{est}}$ and $\mathbf{C}_0^{\mathrm{est}}$, respectively, and setting $k = 1$, we write our algorithm as follows:

The VEnKF algorithm

1. Move the ensemble forward and build the prior:
   (a) Compute prior center point $\mathbf{x}_k^p = \mathcal{M}(\mathbf{x}_{k-1}^{\mathrm{est}})$.
   (b) Compute prior ensemble $\mathbf{s}_{k,i}^p = \mathcal{M}(\mathbf{s}_{(k-1),i}^{\mathrm{est}})$, $i = 1, \ldots, N$.
   (c) Define $(\mathbf{C}_k^p)^{-1}$ using SMW formula (3.3)
   (**Alternatively:** apply LBFGS to (3.2) to get $(\mathbf{C}_k^p)^{-1}$).
2. Calculate the posterior estimate and generate the new ensemble:
   (a) Apply LBFGS to minimize (2.1) to get $\mathbf{x}_k^{\mathrm{est}}$ and $\mathbf{C}_k^{\mathrm{est}}$.
   (b) Sample new ensemble $\mathbf{s}_{k,i}^{\mathrm{est}} \sim N(\mathbf{x}_k^{\mathrm{est}}, \mathbf{C}_k^{\mathrm{est}})$; see Appendix A for details.
3. Set $k \to k + 1$ and go to step 1.

Since the optimization tasks are both quadratic, only a quadratic version of the LBFGS method is needed. The LBFGS algorithm is given in Appendix A, for a detailed analysis we refer to, e.g., [27].

The proposed VEnKF algorithm has several attractive features. First of all, it does not suffer from ensemble in-breeding that is often encountered with many ensemble filtering methods, since the new ensemble is re-generated at each step. The ensembles are generated from dynamically changing covariances that are not restricted to any fixed subspace as in reduced rank methods. The ensemble generation is performed directly by the low storage LBFGS construction without the need of any further matrix (Cholesky) decomposition. Moreover, the model error covariance term can be added explicitly to the optimization problem in step 1(c) of the algorithm, whereas in EnKF it is handled by randomly perturbing the prior ensemble with the model error. In addition, VEnKF uses the state estimate given by the optimizer as the expectation in the sample covariance calculations instead of the sample mean. Mainly due to these reasons, VEnKF has a better performance than EnKF in our examples, when the ensemble size is small (see Section 4). Note that VEnKF especially applies to high dimensional problems, since all calculations are performed using the LBFGS covariance representations without handling full covariance matrices.

It is worth noting that the covariance matrices produced by LBFGS (with a diagonal initial inverse Hessian) are full rank and VEnKF is therefore not a "reduced rank" method. Thus, the new ensemble members generated in VEnKF perturb also the directions of small eigenvalues. This is not the case for many other covariance approximation/optimization methods that could be used instead of LBFGS. For example, Lanczos and conjugate gradient methods operate on a low-dimensional subspace.

4. Numerical experiments. In this section, we demonstrate the performance of VEnKF with two synthetic examples, adopted from [2]. The first example is the well-known Lorenz 95 benchmark problem (a low-order nonlinear chaotic ODE system) that shares some characteristics with weather models. The second example is a linear system, where the dimension of the problem can be controlled by changing the discretization.

For comparing methods, we use the root mean square error (rms), written as

$$[\mathrm{rms}]_k = \sqrt{\frac{1}{d}||\mathbf{x}_k^{\mathrm{est}} - \mathbf{x}_k^{\mathrm{true}}||^2},$$

where $\mathbf{x}_k^{\mathrm{est}}$ is the filter estimate and $\mathbf{x}_k^{\mathrm{true}}$ is the truth used in the data generation at iteration k.

4.1. Lorenz 95. In this example, we consider the well-known nonlinear and chaotic Lorenz 95 model introduced in [24] and analyzed in [25]. The model shares many characteristics with realistic atmospheric models and it is often used as a low-order test case for weather forecasting schemes. We use a 40-dimensional version of the model, given as an ODE system

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + 8, \;\; i = 1, 2, \ldots, 40.$$

The state variables are periodic: $x_{-1} = x_{39}$, $x_0 = x_{40}$, and $x_{41} = x_1$. Out of the 40 model states, measurements are obtained from 24 states. We define the observation operator (following [2]) as $\mathcal{K}(\mathbf{x}) = \mathbf{K}\mathbf{x}$, where

$$[\mathbf{K}]_{rp} = \begin{cases} 1, & (r, p) \in \{(3j + i, 5j + i + 2)\} \\ 0, & \text{otherwise} \end{cases} \quad ,$$

where $i = 1, 2, 3$ and $j = 0, 1, \ldots, 7$. Thus, we observe the last three states in every set of five. To generate data, we add Gaussian noise to the model solution with zero mean and covariance $(0.15\sigma_{\text{clim}})^2\mathbf{I}$, where $\sigma_{\text{clim}} = 3.641$ ('climatological' standard deviation computed from long model simulations). In the filtering methods, we use $\mathbf{C}_{\varepsilon_k^p} = (0.05\sigma_{\text{clim}})^2\mathbf{I}$ as the model error covariance and $\mathbf{C}_{\varepsilon_k^o} = (0.15\sigma_{\text{clim}})^2\mathbf{I}$ as the observation error covariance. As initial guesses in the filtering, we use $\mathbf{x}_0^{\text{est}} = \mathbf{1}$ and $\mathbf{C}_0^{\text{est}} = \mathbf{I}$. For more details about the example, we refer to [2].

We run experiments with varying ensemble size N and varying number of LBFGS iterations. In Figure 4.1, we compare the performance of EKF, EnKF, and VEnKF with $N = (10, 15, 20, 40)$ in terms of the rms error. Since EnKF and VEnKF are stochastic methods, we display rms errors averaged over 10 repetitions. In VEnKF, the number of LBFGS iterations and the number of LBFGS vectors stored was the same as the ensemble size. From the results it is clear that VEnKF works better when the ensemble size is small. When the ensemble size gets large, the performances of VEnKF and EnKF approach each other.

In Figure 4.2, we compare the forecast skills given by different methods using the same ensemble sizes as above. The forecast skill is here defined as the mean squared difference between the "truth" and the forecast made with the model scaled with σ_{clim}; see [2] for details. Again, VEnKF performs better, especially when N is small. For instance, VEnKF with $N = 10$ performs equally well as EnKF with $N = 20$. Even with larger N, VEnKF is better on average.

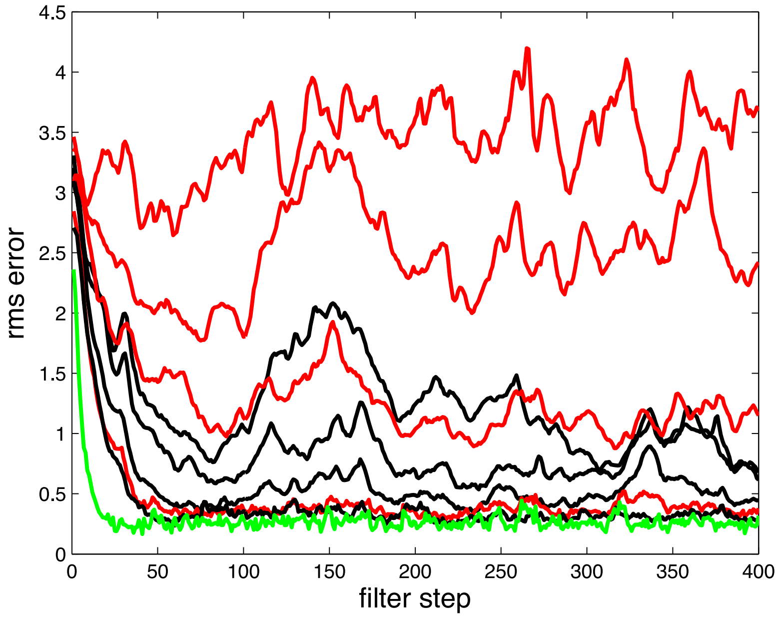

FIG. 4.1. *Comparison of EnKF (red), VEnKF (black), and EKF (green) with ensemble sizes* $N = (10, 15, 20, 40)$ *in the Lorenz 95 example. Increasing ensemble size leads to monotonically decreasing error levels for both EnKF and VEnKF.*

To further demonstrate the behavior of VEnKF, in Figure 4.3 we compare the rms errors (averaged over time) with varying ensemble sizes and varying number of the LBFGS iterations used. As a reference, we also plot the EnKF performance. One can see that 30 LBFGS iterations practically give an as good performance as 40 or 100 iterations, and EnKF starts to produce acceptable results when $N \geq 30$.

ETNA
Kent State University
http://etna.math.kent.edu

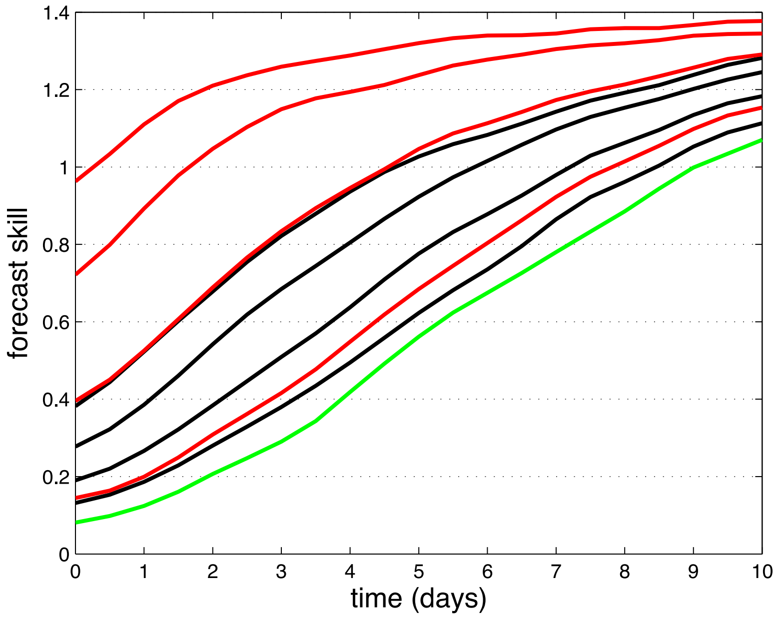

FIG. 4.2. *Forecast skill comparison of EnKF (red), VEnKF (black), and EKF (green) with ensemble sizes* $N = (10, 15, 20, 40)$ *in the Lorenz 95 example. Increasing ensemble size leads to monotonically increasing forecast skill for both EnKF and VEnKF.*

4.2. Heat equation. The purpose of this example, adopted from [2], is to demonstrate the behavior of VEnKF when the dimension is large. The example is linear, so we can directly compare the results with KF. However, as the dimension of the problem is increased, KF cannot be run anymore due to memory issues. Note that while this example does illustrate some computational aspects related to the methods, this system is well-behaved and we cannot conclude much about how the methods work in a high-dimensional chaotic case such as numerical weather prediction, for example.

The model describes heat propagation on a two-dimensional grid and is written as a PDE

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

where x is the temperature at coordinates u and v over the domain $\Omega = \{(u,v)|u,v \in [0,1]\}$. The last term in the equation is an external heat source, whose magnitude can be controlled with the parameter $\alpha \geq 0$.

We discretize the model using a uniform $S \times S$ grid. This leads to a linear forward model $\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{f}$, where $\mathbf{M}$ corresponds to the heat diffusion and $\mathbf{f}$ to the external forcing; see [2] for details. The dimension of the problem can be controlled by changing S. The observation operator $\mathbf{K}$ is defined as in [2]: the measured temperature is a weighted average of temperatures at neighboring points at $S^2/64$ evenly spaced locations.

The data are generated by adding normally distributed random noise to the model state and the corresponding response:

$$\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{f} + \mathrm{N}(\mathbf{0}, (0.5\sigma_{ev})^2\mathbf{I})$$
$$\mathbf{y}_{k+1} = \mathbf{K}\mathbf{x}_{k+1} + \mathrm{N}(\mathbf{0}, (0.8\sigma_{obs})^2\mathbf{I}).$$

In the data generation we use $\alpha = 0.75$ and choose σ_{ev} and σ_{obs} so that the signal to noise

ETNA
Kent State University
http://etna.math.kent.edu

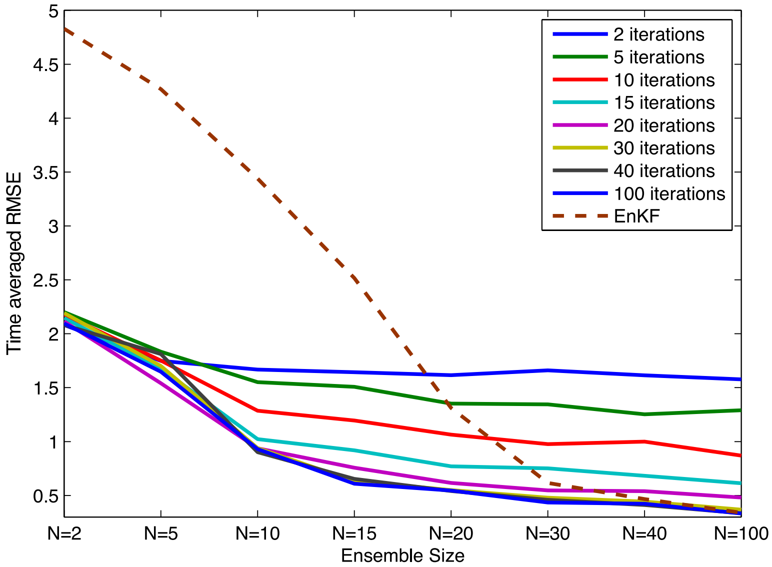

FIG. 4.3. *Rms error averaged over time for EnKF and for VEnKF with varying ensemble sizes and LBFGS iterations in the Lorenz 95 example.*

ratios at the initial condition defined by $\frac{||\mathbf{x}_0||^2}{S^2\sigma_{ev}^2}$ and $\frac{||\mathbf{K}\mathbf{x}_0||^2}{m^2\sigma_{obs}^2}$ are both 50. The initial condition for the data generation is

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

For the filtering we use a biased model, where the forcing term is dropped by setting $\alpha = 0$. The error covariances used for model and observations are $\sigma_{ev}^2\mathbf{I}$ and $\sigma_{obs}^2\mathbf{I}$, respectively. We start all filters from the initial guess $\mathbf{x}_0 = \mathbf{0}$. For ensemble filters, all members are initialized to the same value and for KF we set the initial covariance estimate to $\mathbf{C}_0^{est} = 0$.

In our first test, we take $S = 2^j$ and choose $j = 5$, which is the largest integer so that KF can still be computed on a standard desktop computer. Thus, the dimension of the initial test was $d = S^2 = 1024$. In Figure 4.4, we compare KF, VEnKF, and EnKF using ensemble sizes $N = (5, 10, 20, 50, 100)$ for VEnKF and EnKF. In VEnKF, we always take 20 LBFGS iterations and store 20 LBFGS vectors. The performance of VEnKF approaches that of KF as the ensemble size increases, but EnKF performs poorly: only with larger ensemble sizes we get acceptable results. We think that such a dramatic difference between VEnKF and EnKF is related to the handling of the measurement error in the method. In this case, we have a simple linear dynamic and a rather good guess about the model error, and just a plain "3D-Var" method, where the prior covariance is kept constant, performs rather well. This is the lower limit of the performance of VEnKF upon which we can improve by adding ensemble members. In EnKF, the model error can be added only by perturbing the (few) samples randomly, which can lead to large errors. In addition, in EnKF the state estimate is calculated as the sample mean, whereas VEnKF uses the MAP estimate. In this case this might produce the large errors in EnKF.

Next, we compared VEnKF to EnKF in a case, where the dimension is much higher ($j = 7, d = S^2 = 16384$) using the same ensemble sizes and the same LBFGS settings. In this case, KF cannot be used anymore due to memory issues. As it can be seen in Figure 4.5, the difference between EnKF and VEnKF is even more dramatic in this case: the EnKF performance is poor, whereas VEnKF is able to improve the simple 3D-Var results.

ETNA
Kent State University
http://etna.math.kent.edu

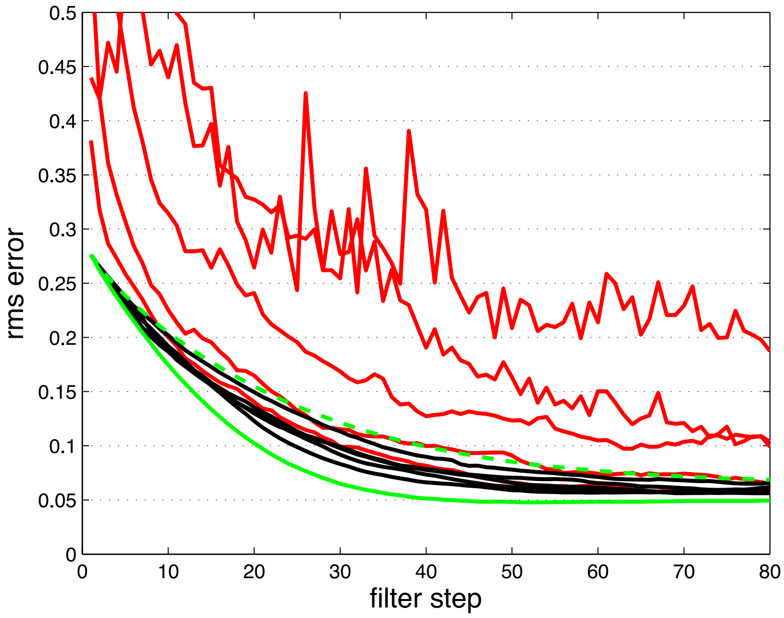

FIG. 4.4. *Performance comparison of KF (solid green), VEnKF (black), and EnKF (red) with different ensemble sizes in the case where $d = 1024$. The dashed green line is 3D-Var, where only the fixed model error covariance is used. Increasing ensemble size leads to monotonically decreasing error levels for both EnKF and VEnKF.*

5. Discussion. In the past decade, a wide literature about ensemble filtering has emerged. We start this section by reviewing some of it and discuss the existing approaches in light of our VEnKF method. Later, we discuss some specific topics related to VEnKF.

The standard EnKF is criticized in many papers because of the additional sampling errors brought in by randomly perturbing the observations. In so called square root ensemble filters (SRFs) this is not needed; see, e.g., [12] and the review of SRF methods given in [29]. Similar approaches include the ensemble adjustment Kalman filter (EAKF) and the ensemble transform Kalman filter (ETKF) given in [1] and [5], respectively. In SRF methods, the prior ensemble is deterministically transformed so that the posterior statistics match with the theory in the linear case (Kalman filter equations). This is done basically by writing the Kalman formulas for matrix "square roots" (symmetric decompositions) instead of covariance matrices in a manner that avoids forming full covariance matrices. This transformation is non-unique leading to various SRF formulations; see [29] for a comparison. One difficulty in SRF methods, as pointed out in [29], is the handling of the model error. A simple way is to include it by perturbing the ensemble (as in EnKF), but this potentially leads to sampling errors much in the same way as with perturbed observations in the standard EnKF. Dropping the model error altogether leads to underestimation of errors and possible divergence issues, and different "covariance inflation" mechanisms need to be developed for this purpose. In our VEnKF algorithm, both the model error and the observation error covariances are explicitly present in the minimized cost functions, and these problems do not appear here. Moreover, the square root filters operate, as for the standard EnKF, only in the subspace spanned by the ensemble vectors, whereas the VEnKF samples from the full state space.

Hybrid EnKF methods that incorporate features from both EnKF and variational methods (as VEnKF) have been found appealing in many studies. In [16], a hybrid approach is adopted, which combines elements of ensemble filtering and 3D-Var. However, the hybrid method uses perturbed observations and makes the perfect model assumption; model error

ETNA
Kent State University
http://etna.math.kent.edu

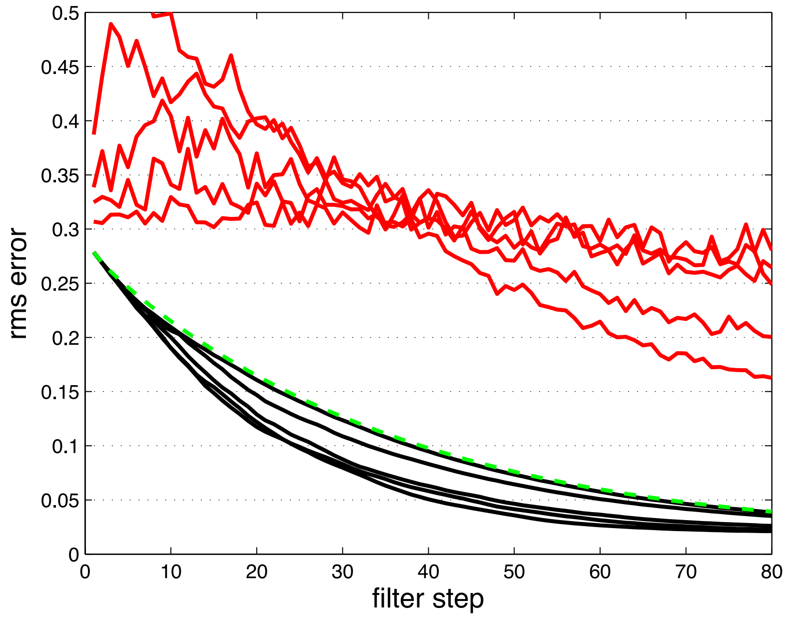

FIG. 4.5. *Performance comparison of VEnKF (black) and EnKF (red) with different ensemble sizes in the case where* $d = 16384$*. The dashed green line is 3D-Var, where only the fixed model error covariance is used. Increasing ensemble size leads to monotonically decreasing error levels for both EnKF and VEnKF.*

covariance cannot be easily incorporated. The prior covariance is defined as a linear combination of the sample covariance and the static model error covariance used in 3D-Var, which is rather ad-hoc and introduces a tuning parameter (the weight given for the sample covariance). Similar approaches are introduced in [10] and [30]. In [31], the hybrid approach is extended so that it can be used with the 4D-Var method, but the same problems remain.

Probably the closest method to our VEnKF approach is the maximum likelihood ensemble Filter (MLEF) proposed in [32]. In MLEF, an iterative optimization method is used to optimize a 3D-Var type of cost function, and the found optimum is used as the state estimate instead of the sample mean. However, the model error term is neglected and the method operates only in the ensemble subspace.

One criticism that is faced by all Kalman filter based methods are the Gaussian approximations: in all of the methods discussed so far in this paper, a Gaussian form is used for the prior. The ensemble methods are more nonlinear than EKF in the sense that the covariance information is propagated using the nonlinear model instead of the linearized model. Purely nonlinear filtering methods exist as well; see, e.g., [7] for a recent introduction to particle filters. Their benefit over the linear methods can be easily shown in small-dimensional cases, but they become infeasible in large systems. Some nonlinearity in the prior formulation can be induced, e.g., by representing the prior as a mixture of Gaussians or a kernel density estimate fitted to the ensemble; see, e.g., [1, 4] for some discussion. We note that different, nonlinear prior and likelihood formulations can be rather easily incorporated into the cost functions used in the proposed VEnKF method. The only complication is that the cost function is no longer quadratic and one has to use the full LBFGS algorithm instead of the quadratic version used in this paper and to worry about e.g., the Wolfe conditions in the optimization. Extending VEnKF to this type of nonlinear filtering is a topic for further research.

In the VEnKF algorithm, we sample new ensemble members at each iteration. Traditionally, sampling from a multivariate Gaussian distribution is done by performing a symmetric decomposition on the covariance matrix, $\mathbf{C} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$, for example using the Cholesky decomposition, and then producing random vectors as $\mathbf{Lz}$, where $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$. The work in [17] comes close to our approach as it employs the BFGS covariance approximation for the proposal distribution in the Metropolis MCMC algorithm, but using again the Cholesky decomposition. In high dimensions, performing these decompositions is infeasible, since we cannot even store the full covariance matrix. However, as shown in this paper, random sampling can be done directly using the LBFGS vectors. This is potentially useful in other contexts as well, where high dimensional random sampling is needed, for example in the field of inverse problems.

Although we have shown that VEnKF performs well, the approach has its downsides. In particular, the LBFGS optimization is sensitive to certain "parameters" and therefore VEnKF requires some case specific tuning. One tuning parameter is the preconditioner for the inverse Hessian used in the LBFGS optimization. In this paper, we use a heuristic given in [27] that eliminates this tuning parameter (see Appendix A for details). Other tuning parameters remain, such as the choice of the initial guess, the number of LBFGS vectors that we store, and the number of LBFGS iterations that we take. At present, we have not found any general way to define these parameters other than "trial and error".

Naturally, the performance of VEnKF depends on the accuracy of the LBFGS covariance approximations. In [2], the quality of LBFGS covariances was found to be good in low-dimensional numerical examples. However, the LBFGS performance in approximating covariance in high dimensions and realistic data assimilation problems remains a topic of future research.

6. Conclusions. In this paper, we propose a hybrid method called the variational ensemble Kalman filter (VEnKF) for high-dimensional data assimilation that combines elements from ensemble filtering and variational methods. VEnKF is based on the variational Kalman filter (VKF) method [2], where the memory issues related to EKF are solved by low-storage approximations of the state covariances obtained using the LBFGS optimization method. The proposed approach can solve some problems often encountered with ensemble methods, such as sampling errors due to random perturbation of states and observations and ensemble in-breeding. All of the calculations in VEnKF, noting especially the generation of new ensembles, can be made using the LBFGS covariance representation without handling full covariance matrices. Since VEnKF is an ensemble method, it does not need tangent linear and adjoint codes. We show with synthetic examples that the method can perform better than the standard ensemble Kalman filter. Testing the applicability of the method on real data assimilation problems remains a topic of future research.

Acknowledgements. The work was supported by the Centre of Excellence in Inverse Problems of the Academy of Finland.

Appendix A. In this appendix, we give details of some computational issues in VEnKF. First, we recall how the LBFGS optimization algorithm for quadratic minimization problems works. Then, we shown how random samples can be produced from the LBFGS covariance representation.

The LBFGS algorithm for minimizing a quadratic function $q(\mathbf{u}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{A}\mathbf{u}$, given an initial guess $\mathbf{u}_0$, reads as

LBFGS algorithm for quadratic problems

1. Choose an inverse Hessian approximation $\mathbf{H}_k^0$.
2. Compute the gradient $\mathbf{g}_k = \nabla q(\mathbf{u}_k) = \mathbf{A}\mathbf{u}_k$.

ETNA
Kent State University
http://etna.math.kent.edu

3. Compute a search direction $\mathbf{p}_k = \mathbf{H}_k \mathbf{g}_k$, where $\mathbf{H}_k$ is the LBFGS approximation of the inverse Hessian (details below).
4. Compute step size $\alpha_k = (\mathbf{g}_k^{\mathrm{T}} \mathbf{p}_k)/(\mathbf{p}_k^{\mathrm{T}} \mathbf{A} \mathbf{p}_k)$.
5. Set $\mathbf{u}_{k+1} = \mathbf{u}_k - \alpha_k \mathbf{p}_k$ and $k \to k+1$ and go to step 1.

The LBFGS algorithm uses the BFGS formula for approximating $\mathbf{H}_k$, which is recursively defined as

$$\mathbf{H}_{k+1} = \mathbf{V}_k^{\mathrm{T}} \mathbf{H}_k \mathbf{V}_k + \rho_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}},$$

where

$$\begin{aligned}
\rho_k &= 1/(\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k) \\
\mathbf{V}_k &= \mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^{\mathrm{T}} \\
\mathbf{s}_k &= \mathbf{u}_{k+1} - \mathbf{u}_k \\
\mathbf{y}_k &= \mathbf{g}_{k+1} - \mathbf{g}_k.
\end{aligned}$$

In LBFGS only a certain number n of the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ are used in the above formula. Writing out the recursive formula and storing only n most recent vectors leads to the following formula for the LBFGS inverse Hessian approximation:

$$\begin{aligned}
\mathbf{H}_k = {} & (\mathbf{V}_{k-1}^{\mathrm{T}} \dots \mathbf{V}_{k-n}^{\mathrm{T}}) \mathbf{H}_k^0 (\mathbf{V}_{k-n} \dots \mathbf{V}_{k-1}) \\
& + \rho_{k-n} (\mathbf{V}_{k-1}^{\mathrm{T}} \dots \mathbf{V}_{k-n+1}^{\mathrm{T}}) \mathbf{s}_{k-n} \mathbf{s}_{k-n}^{\mathrm{T}} (\mathbf{V}_{k-n+1} \dots \mathbf{V}_{k-1}) \\
& + \rho_{k-n+1} (\mathbf{V}_{k-1}^{\mathrm{T}} \dots \mathbf{V}_{k-n+2}^{\mathrm{T}}) \mathbf{s}_{k-n+1} \mathbf{s}_{k-n+1}^{\mathrm{T}} (\mathbf{V}_{k-n+2} \dots \mathbf{V}_{k-1}) \\
& + \dots \\
& + \rho_{k-1} \mathbf{s}_{k-1} \mathbf{s}_{k-1}^{\mathrm{T}}.
\end{aligned}$$

Note that in the LBFGS formulation, the initial inverse Hessian approximation $\mathbf{H}_k^0$ can vary from one iteration to another. One can choose, e.g., a fixed diagonal covariance $\mathbf{H}_k^0 = \gamma \mathbf{I}$, but then the issue of tuning γ emerges. We use a heuristic from [27] in which $\mathbf{H}_k^0 = \gamma_k \mathbf{I}$ and $\gamma_k = (\mathbf{s}_{k-1}^{\mathrm{T}} \mathbf{y}_{k-1})/(\mathbf{y}_{k-1}^{\mathrm{T}} \mathbf{y}_{k-1})$, which attempts to estimate the size of the covariance along the last search direction; see [27].

In our applications, we never want to calculate and store the full inverse Hessian, but to keep it in the above "vector form". There exists an efficient iterative algorithm for computing matrix-vector products with the inverse Hessian needed, e.g., when calculating the search direction in the LBFGS algorithm; see, e.g., [27] for details.

Assuming that the initial inverse Hessian can be decomposed into $\mathbf{H}_k^0 = \mathbf{L}_0 \mathbf{L}_0^{\mathrm{T}}$, the above LBFGS inverse Hessian formula can be written in the form

$$\mathbf{H}_k = \mathbf{B}_0 \mathbf{B}_0^{\mathrm{T}} + \sum_{i=1}^{n} \mathbf{b}_i \mathbf{b}_i^{\mathrm{T}},$$

where

$$\begin{aligned}
\mathbf{B}_0 &= (\mathbf{V}_{k-1}^{\mathrm{T}} \dots \mathbf{V}_{k-n}^{\mathrm{T}}) \mathbf{L}_0 \\
\mathbf{b}_1 &= \sqrt{\rho_{k-1}} \mathbf{s}_{k-1} \\
\mathbf{b}_i &= \sqrt{\rho_{k-i}} (\mathbf{V}_{k-1}^{\mathrm{T}} \dots \mathbf{V}_{k-i+1}^{\mathrm{T}}) \mathbf{s}_{k-i}, \quad i = 2, \dots, n.
\end{aligned}$$

Note that the square roots $\sqrt{\rho_i}$ can always be calculated, since in the LBFGS algorithm we choose the step length so that $\rho_i \geq 0$ for all i; see [27] for details. Thus, we can sample zero mean random variables from the covariance $\mathbf{H}_k$ by calculating

$$\mathbf{r} = \mathbf{B}_0\mathbf{z} + \sum_{i=1}^{n} \omega_i \mathbf{b}_i,$$

where $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ and $\omega_i \sim N(0, 1)$. It is simple to verify that $\mathrm{Cov}(\mathbf{r}) = \mathbf{H}_k$. The needed products can be calculated efficiently without storing full matrices of size $d \times d$. For the first product $\mathbf{q} = \mathbf{B}_0\mathbf{z}$, we can use the iteration

- Set $\mathbf{q} \to \mathbf{L}_0\mathbf{z}$.
- For $i = 1 \ldots n$
    1. $\rho_k = 1/(\mathbf{y}_k^{\mathrm{T}}\mathbf{s}_k)$.
    2. Set $\mathbf{q} \to \mathbf{q} - (\rho_k \mathbf{y}_k^{\mathrm{T}}\mathbf{q})\mathbf{s}_k$.
- Return $\mathbf{q}$.

For the other products $\omega_i\mathbf{b}_i$ we can first calculate the vectors $\mathbf{b}_i$ explicitly using a similar loop as above and then take sums of the vectors, weighted by random numbers ω_i.

REFERENCES

[1] J. L. ANDERSON, *An ensemble adjustment Kalman filter for data assimilation*, Mon. Wea. Rev., 129 (2001), pp. 2884–2903.

[2] H. AUVINEN, J. M. BARDSLEY, H. HAARIO, AND T. KAURANNE, *The variational Kalman filter and an efficient implementation using limited memory BFGS*, Internat. J. Numer. Methods Fluids, 64 (2009), pp. 314–335.

[3] ———, *Large-scale Kalman filtering using the limited memory BFGS method*, Electron. Trans. Numer. Anal., 35 (2009), pp. 217–233.
http://etna.math.kent.edu/vol.35.2009/pp217-233.dir

[4] T. BENGTSSON, C. SNYDER, AND D. NYCHKA, *Toward a nonlinear ensemble filter for high-dimensional systems*, J. Geophys. Res., 108 (2003), 8775 (10 pages).

[5] C. H. BISHOP, B. J. ETHERTON, AND S. J. MAJUMDAR, *Adaptive sampling with the ensemble transform Kalman filter. Part I: theoretical aspects*, Mon. Wea. Rev, 129 (2001), pp. 420–436.

[6] M. A. CANE, 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., 101 (1996), pp. 22599–22617.

[7] O. CAPPE, S. GODSILL, AND E. MOULINES, *An overview of existing methods and recent advances in sequential Monte Carlo*, Proc. IEEE, 95 (2007), pp. 899–924.

[8] Q. CHENG, J. CAO, B. WANG, AND H. ZHANG, *Adjoint code generator*, Sci. China Ser. F, 52 (2009), pp. 926–941.

[9] D. P. DEE, *Simplification of the Kalman filter for meteorological data assimilation*, Q. J. R. Meteorol. Soc., 117 (1990), pp. 365–384.

[10] B. J. ETHERTON AND C. H. BISHOP, *Resilience of hybrid ensemble/3DVAR analysis schemes to model error and ensemble covariance error*, Mon. Wea. Rev., 132 (2004), pp. 1065–1080.

[11] G. EVENSEN, *Sequential data assimilation with a non-linear quasi-geostrophic model using Monte Carlo methods to forecast error statistics*, J. Geophys. Res., 99 (1994), pp. 10143–10162.

[12] ———, *Sampling strategies and square root analysis schemes for the EnKF*, Ocean Dynam., 54 (2004), pp. 539–560.

[13] M. FISHER, *Development of a simplified Kalman filter*, ECMWF Technical Memorandum n. 260 (1998), European Centre for Medium-Range Weather Forecasts, Shinfield Park, Reading RG2-9AX, UK.

[14] M. FISHER AND E. ANDERSSON, *Developments in 4D-var and Kalman filtering*, ECMWF Technical Memorandum n. 347 (2001), European Centre for Medium-Range Weather Forecasts, Shinfield Park, Reading RG2-9AX, UK.

[15] R. GIERING AND T. KAMINSKI, *Recipes for adjoint code construction*, ACM Trans. Math. Software, 24 (1998), pp. 437–474.

[16] T. M. HAMILL AND C. SNYDER, *A hybrid ensemble Kalman filter-3D variational analysis scheme*, Mon. Wea. Rev., 128 (2000), pp. 2905–2919.

[17] K. M. HANSON AND G. CUNNINGHAM, *Posterior sampling with improved efficiency*, in Medical Imaging 1998: Image Processing, K. M. Hanson, ed., Proceedings of SPIE, 3338, SPIE, Bellingham WA, 1998, pp. 371–382.

[18] N. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[19] P. HOUTEKAMER AND H. L. MITCHELL, *Data assimilation using an ensemble Kalman filter technique*, Mon. Wea. Rev., 126 (1998), pp. 796–811.

[20] L. ISAKSEN, M. FISHER, E. ANDERSSON, AND J. BARKMEIJER, *The structure and realism of sensitivity perturbations and their interpretation as 'Key Analysis Errors'*, Q. J. R. Meteorol. Soc., 131 (2005), pp. 3053–3078.

[21] R. E. KALMAN, *A new approach to linear filtering and prediction problems*, Trans. ASME Ser. D J. Bas. Engrg., 82 (1960), pp. 35–45.

[22] F. X. LE DIMET AND O. TALAGRAND, *Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects*, Tellus Ser. A., 38 (1986), pp. 97–110.

[23] H. LI, E. KALNAY, AND T. MIYOSHI, *Simultaneous estimation of covariance inflation and observation errors within an ensemble Kalman filter*, Q. J. R. Meteorol. Soc., 135 (2009), pp. 523–533.

[24] E. N. LORENZ, *Predictability: a problem partly solved*, in Proceedings of the Seminar on Predictability, Vol. 1, T. Palmer, ed., ECMWF, Reading, UK, (1996), pp. 1–18.

[25] E. N. LORENZ AND K. A. EMANUEL, *Optimal sites for supplementary weather observations: simulation with a small model*, J. Atmospheric Sci., 55 (1998), pp. 399–414.

[26] Z. MENG AND F. ZHANG, *Tests of an ensemble Kalman filter for mesoscale and regional-scale data assimilation. Part II: imperfect model experiments*, Mon. Wea. Rev., 135 (2007), pp. 1403–1423.

[27] J. NOCEDAL AND S. WRIGHT, *Numerical Optimization*, Springer, Berlin, 1999.

[28] W. SACHER AND P. BARTELLO, *Sampling errors in ensemble Kalman filtering. Part I: theory*, Mon. Wea. Rev., 136 (2008), pp. 3035–3049.

[29] M. K. TIPPETT, J. L. ANDERSON, C. H. BISHOP, T. M. HAMILL, AND J. S. WHITAKER, *Ensemble square root filters*, Mon. Wea. Rev., 131 (2003), pp. 1485–1490.

[30] X. WANG, D. M. BARKER, C. SNYDER, AND T. M. HAMILL, *A hybrid ETKF-3DVAR data assimilation scheme for the WRF model. Part I: observing system simulation experiment*, Mon. Wea. Rev., 136 (2008), pp. 5116–5131.

[31] F. ZHANG, M. ZHANG, AND J. A. HANSEN, *Coupling ensemble Kalman filter with four-dimensional data assimilation*, Adv. Atmospher. Sci., 26 (2009), pp. 1–8.

[32] M. ZUPANSKI, *Maximum likelihood ensemble filter: theoretical aspect*, Mon. Wea. Rev., 133 (2005), pp. 1710–1726.