Efficient Data Assimilation for Spatiotemporal Chaos: a Local Ensemble Transform Kalman Filter

Brian R. Hunt

Institute for Physical Science and Technology and Department of Mathematics

University of Maryland, College Park MD 20742

Eric J. Kostelich

Department of Mathematics and Statistics

Arizona State University, Tempe AZ 85287-1804

Istvan Szunyogh

Institute for Physical Science and Technology and Department of Atmospheric and Oceanic Science

University of Maryland, College Park MD 20742

12 December 2006

Abstract

Data assimilation is an iterative approach to the problem of estimating the state of a dynamical system using both current and past observations of the system together with a model for the system's time evolution. Rather than solving the problem from scratch each time new observations become available, one uses the model to "forecast" the current state, using a prior state estimate (which incorporates information from past data) as the initial condition, then uses current data to correct the prior forecast to a current state estimate. This Bayesian approach is most effective when the uncertainty in both the observations and in the state estimate, as it evolves over time, are accurately quantified. In this article, we describe a practical method for data assimilation in large, spatiotemporally chaotic systems. The method is a type of "ensemble Kalman filter", in which the state estimate and its approximate uncertainty are represented at any given time by an ensemble of system states. We discuss both the mathematical basis of this approach and its implementation; our primary emphasis is on ease of use and computational speed rather than improving accuracy over previously published approaches to ensemble

Kalman filtering. We include some numerical results demonstrating the efficiency and accuracy of our implementation for assimilating real atmospheric data with the global forecast model used by the U.S. National Weather Service.

1 Introduction

Forecasting a physical system generally requires both a model for the time evolution of the system and an estimate of the current state of the system. In some applications, the state of the system can be measured directly with high accuracy. In other applications, such as weather forecasting, direct measurement of the global system state is not feasible. Instead, the state must be inferred from available data. While a reasonable state estimate based on current data may be possible, in general one can obtain a better estimate by using both current and past data. "Data assimilation" provides such an estimate on an ongoing basis, iteratively alternating between a forecast step and a state estimation step; the latter step is often called the "analysis". The analysis step combines information from current data and from a prior short-term forecast (which is based on past data), producing a current state estimate. This estimate is used to initialize the next short-term forecast, which is subsequently used in the next analysis, and so on. The data assimilation procedure is itself a dynamical system driven by the physical system, and the practical problem is to achieve good "synchronization" [40] between the two systems.

Data assimilation is widely used to study and forecast geophysical systems [13, 28]. The analysis step is generally a statistical procedure (specifically, a Bayesian maximum likelihood estimate) involving a prior (or "background") estimate of the current state based on past data, and current data (or "observations") that are used to improve the state estimate. This procedure requires quantification of the uncertainty in both the background state and the observations. While quantifying the observation uncertainty can be a nontrivial problem, in this article we consider that problem to be solved, and instead concentrate on the problem of quantifying the background uncertainty.

There are two main factors that create background uncertainty. One is the uncertainty in the initial conditions from the previous analysis, which produces the background state via a short-term forecast. The other is "model error", the unknown discrepancy between the model dynamics and actual system dynamics. Quantifying the uncertainty due to model error is a challenging problem, and while this problem generally cannot be ignored in practice, we discuss only crude ways of accounting for it in this article. For the time being, let us consider an idealized "perfect model" scenario, in which there is no model error.

The main purpose of this article is to describe a practical framework for data assimilation that is both relatively easy to implement and computationally efficient, even for large, spatiotemporally chaotic systems. (By "spatiotemporally chaotic" we mean a spatially extended system that exhibits

temporally chaotic behavior with weak long-range spatial correlations.) The emphasis here is on methodology that scales well to high-dimensional systems and large numbers of observations, rather than on what would be optimal given unlimited computational resources. Ideally, one would keep track of a probability distribution of system states, propagating the distribution using the Fokker-Planck-Kolmogorov equation during the forecast step. While this approach provides a theoretical basis for the methods used in practice [25], it would be computationally expensive even for a low-dimensional system and is not at all feasible for a high-dimensional system. Instead one can use a Monte Carlo approach, using a large ensemble of system states to approximate the distribution (see [6] for an overview), or a parametric approach like the Kalman filter [26, 27], which assumes Gaussian distributions and tracks their mean and covariance. (The latter approach was derived originally for linear problems, but serves as a reasonable approximation for nonlinear problems when the uncertainties remain sufficiently small.)

The methodology of this article is based on the Ensemble Kalman Filter [7, 8, 9], which has elements of both approaches: it uses the Gaussian approximation and follows the time evolution of the mean and covariance by propagating an ensemble of states. The ensemble can be reasonably small relative to other Monte Carlo methods because it is used only to parametrize the distribution, not to sample it thoroughly. The ensemble should be large enough to approximately span the space of possible system states at a given time, because the analysis essentially determines which linear combination of the ensemble members forms the best estimate of the current state, given the current observations.

Many variations on the Ensemble Kalman Filter have been published in the geophysical literature, and this article draws ideas from a number of them [1, 2, 4, 17, 20, 21, 30, 36, 37, 45, 48]. These articles in turn draw ideas both from earlier work on geophysical data assimilation and from the engineering and mathematics literature on nonlinear filtering. For the most part, we limit our citations to ensemble-based articles rather than attempt to trace all ideas to their original sources. We call the method described here a Local Ensemble Transform Kalman Filter (LETKF), because it is most closely related to the Local Ensemble Kalman Filter [36, 37] and the Ensemble Transform Kalman Filter [4]. Indeed, it can produce analyses that are equivalent to the LEKF in a more efficient manner that is formally similar to the ETKF. While this article does not describe a fundamentally new method for data assimilation, it proposes a significant refinement of previously published approaches that combines formal simplicity with the flexibility to adapt to a variety of applications.

In Section 2, we start by posing a general problem about which trajectory of a dynamical system "best fits" a time series of data; this problem is solved exactly for linear problems by the Kalman filter and approximately for nonlinear problems by ensemble Kalman filters. Next we derive the Kalman filter equations as a guide for what follows. Then we discuss ensemble Kalman filters

in general and the issue of "localization", which is important for applications to spatiotemporally chaotic systems. Finally, we develop the basic LETKF equations, which provide a framework for data assimilation that allows a system-dependent localization strategy to be developed and tuned. We discuss also several options for "covariance inflation" to compensate for the effects of model error and the deficiencies due to small sample size and linear approximation that are inherent to ensemble Kalman filters.

In Section 3, we give step-by-step instructions for efficient implementation of the approach developed in Section 2 and discuss options for further improving computational speed in certain cases. Then in Section 4, we present a generalization that allows observations gathered at different times to be assimilated simultaneously in a natural way. In Section 5, we present preliminary results using a global atmospheric forecast model with real observations; these results compare favorably with the data assimilation method used by the National Weather Service, and demonstrate the feasibility of the LETKF algorithm for large models and data sets. Section 6 is a brief conclusion. The notation in this article is based largely on that proposed in [24], with some elements from [37].

2 Mathematical Formulation

Consider a system governed by the ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = F(t,\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is an m-dimensional vector representing the state of the system at a given time. Suppose we are given a set of (noisy) observations of the system made at various times, and we want to determine which trajectory $\{\mathbf{x}(t)\}$ of (1) "best" fits the observations. For any given t, this trajectory gives an estimate of the system state at time t.

To formulate this problem mathematically, we need to define "best fit" in this context. Let us assume that the observations are the result of measuring quantities that depend on the system state in a known way, with Gaussian measurement errors. In other words, an observation at time t_j is a triple $(\mathbf{y}_j^o, H_j, \mathbf{R}_j)$, where $\mathbf{y}_j^o$ is a vector of observed values, and H_j and $\mathbf{R}_j$ describe the relationship between $\mathbf{y}_j^o$ and $\mathbf{x}(t_j)$:

$$\mathbf{y}_j^o = H_j(\mathbf{x}(t_j)) + \boldsymbol{\varepsilon}_j,$$

where $\boldsymbol{\varepsilon}_j$ is a Gaussian random variable with mean $\mathbf{0}$ and covariance matrix $\mathbf{R}_j$. Notice that we are assuming a perfect model here: the observations are based on a trajectory of (1), and our problem is simply to infer which trajectory produced the observations. In a real application, the observations come from a trajectory of the physical system for which (1) is only a model. So a more realistic (but more complicated) problem would be to determine a pseudo-trajectory of (1), or a trajectory of an

associated stochastic differential equation, that best fits the observations. Formulating this problem mathematically then requires some assumptions about the size and nature of the model error. We use the perfect model problem as motivation and defer the consideration of model error until later.

Given our assumptions about the observations, we can formulate a maximum likelihood estimate for the trajectory of (1) that best fits the observations at times $t_1 < t_2 < \cdots < t_n$. The likelihood of a trajectory $\mathbf{x}(t)$ is proportional to

$$\prod_{j=1}^{n} \exp\left(-\frac{1}{2}[\mathbf{y}_j^o - H_j(\mathbf{x}(t_j))]^T \mathbf{R}_j^{-1} [\mathbf{y}_j^o - H_j(\mathbf{x}(t_j))] \right).$$

The most likely trajectory is the one that maximizes this expression, or equivalently minimizes the "cost function"

$$J^o(\{\mathbf{x}(t)\}) = \sum_{j=1}^{n} [\mathbf{y}_j^o - H_j(\mathbf{x}(t_j))]^T \mathbf{R}_j^{-1} [\mathbf{y}_j^o - H_j(\mathbf{x}(t_j))]. \tag{2}$$

Thus, the "most likely" trajectory is also the one that best fits the observations in a least square sense.

Notice that (2) expresses the cost J^o as a function of the trajectory $\{\mathbf{x}(t)\}$. To minimize the cost, it is more convenient to write J^o as a function of the system state at a particular time t. Let $M_{t,t'}$ be the map that propagates a solution of (1) from time t to time t'.[1] Then

$$J_t^o(\mathbf{x}) = \sum_{j=1}^{n} [\mathbf{y}_j^o - H_j(M_{t,t_j}(\mathbf{x}))]^T \mathbf{R}_j^{-1} [\mathbf{y}_j^o - H_j(M_{t,t_j}(\mathbf{x}))] \tag{3}$$

expresses the cost in terms of the system state $\mathbf{x}$ at time t. Thus to estimate the state at time t, we attempt to minimize J_t^o.

For a nonlinear model, there is no guarantee that a unique minimizer exists. And even if it does, evaluating J_t^o is apt to be computationally expensive, and minimizing it may be impractical. But if both the model and the observation operators H_j are linear, the minimization is quite tractable, because J_t^o is then quadratic. Furthermore, instead of performing the minimization from scratch at each successive time t_n, one can compute the minimizer by an iterative method, namely the Kalman filter [26, 27], which we now describe in the perfect model scenario. This method forms the basis for the approach we will use in the nonlinear scenario.

2.1 Linear Scenario: the Kalman Filter

In the linear scenario, we can write $M_{t,t'}(\mathbf{x}) = \mathbf{M}_{t,t'}\mathbf{x}$ and $H_j(\mathbf{x}) = \mathbf{H}_j\mathbf{x}$ where $\mathbf{M}_{t,t'}$ and $\mathbf{H}_j$ are matrices. Using the terminology from the introduction, we now describe how to perform a forecast

[1] In the derivations that follow, we allow t' to be less than t, though in practice integrating (1) backward in time may be problematic — for example, if (1) represents a discretization of a dissipative partial differential equation. Our use of $M_{t,t'}$ for $t' < t$ is entirely expository; the methodology we develop will not require backward integration of (1).

step from time t_{n-1} to time t_n followed by an analysis step at time t_n, in such a way that if we start with the most likely system state, in the sense described above, given the observations up to time t_{n-1}, we end up with the most likely state given the observations up to time t_n. The forecast step propagates the solution from time t_{n-1} to time t_n, and the analysis step combines the information provided by the observations at time t_n with the propagated information from the prior observations. This iterative approach requires that we keep track of not only the most likely state, but also its uncertainty, in the sense described below. (Of course, the fact that the Kalman filter computes the uncertainty in its state estimate may be viewed as a virtue.)

Suppose the analysis at time t_{n-1} has produced a state estimate $\overline{\mathbf{x}}^a_{n-1}$ and an associated covariance matrix $\mathbf{P}^a_{n-1}$. In probabilistic terms, $\overline{\mathbf{x}}^a_{n-1}$ and $\mathbf{P}^a_{n-1}$ represent the mean and covariance of a Gaussian probability distribution that represents the relative likelihood of the possible system states given the observations from time t_1 to t_{n-1}. Algebraically, what we assume is that for some constant c,

$$\sum_{j=1}^{n-1} [\mathbf{y}^o_j - \mathbf{H}_j \mathbf{M}_{t_{n-1},t_j} \mathbf{x}]^T \mathbf{R}_j^{-1} [\mathbf{y}^o_j - \mathbf{H}_j \mathbf{M}_{t_{n-1},t_j} \mathbf{x}] = [\mathbf{x} - \overline{\mathbf{x}}^a_{n-1}]^T (\mathbf{P}^a_{n-1})^{-1} [\mathbf{x} - \overline{\mathbf{x}}^a_{n-1}] + c. \tag{4}$$

In other words, the analysis at time t_{n-1} has "completed the square" to express the part of the quadratic cost function $J^o_{t_{n-1}}$ that depends on the observations up to that time as a single quadratic form plus a constant. The Kalman filter determines $\overline{\mathbf{x}}^a_n$ and $\mathbf{P}^a_n$ such that an analogous equation holds at time t_n.

First we propagate the analysis state estimate $\overline{\mathbf{x}}^a_{n-1}$ and its covariance $\mathbf{P}^a_{n-1}$ using the forecast model to produce a background state estimate $\overline{\mathbf{x}}^b_n$ and covariance matrix $\mathbf{P}^b_n$ for the next analysis:

$$\overline{\mathbf{x}}^b_n = \mathbf{M}_{t_{n-1},t_n} \overline{\mathbf{x}}^a_{n-1}, \tag{5}$$

$$\mathbf{P}^b_n = \mathbf{M}_{t_{n-1},t_n} \mathbf{P}^a_{n-1} \mathbf{M}^T_{t_{n-1},t_n}. \tag{6}$$

Under a linear model, a Gaussian distribution of states at one time propagates to a Gaussian distribution at any other time, and the equations above describe how the model propagates the mean and covariance of such a distribution. (Usually, the Kalman filter adds a constant matrix to the right side of (6) to represent additional uncertainty due to model error.)

Next, we want to rewrite the cost function $J^o_{t_n}$ given by (3) in terms of the background state estimate and the observations at time t_n. (This step is often formulated as applying Bayes' rule to the corresponding probability density functions.) In (4), $\mathbf{x}$ represents a hypothetical system state at time t_{n-1}. In our expression for $J^o_{t_n}$, we want $\mathbf{x}$ to represent instead a hypothetical system state at time t_n, so we first replace $\mathbf{x}$ by $\mathbf{M}_{t_n,t_{n-1}} \mathbf{x} = \mathbf{M}^{-1}_{t_{n-1},t_n} \mathbf{x}$ in (4). Then using (5) and (6) yields

$$\sum_{j=1}^{n-1} [\mathbf{y}^o_j - \mathbf{H}_j \mathbf{M}_{t_n,t_j} \mathbf{x}]^T \mathbf{R}_j^{-1} [\mathbf{y}^o_j - \mathbf{H}_j \mathbf{M}_{t_n,t_j} \mathbf{x}] = [\mathbf{x} - \overline{\mathbf{x}}^b_n]^T (\mathbf{P}^b_n)^{-1} [\mathbf{x} - \mathbf{x}^b_n] + c.$$

It follows that

$$J^o_{t_n}(\mathbf{x}) = [\mathbf{x} - \overline{\mathbf{x}}^b_n]^T (\mathbf{P}^b_n)^{-1} [\mathbf{x} - \overline{\mathbf{x}}^b_n] + [\mathbf{y}^o_n - \mathbf{H}_n \mathbf{x}]^T \mathbf{R}_n^{-1} [\mathbf{y}^o_n - \mathbf{H}_n \mathbf{x}] + c. \tag{7}$$

To complete the data assimilation cycle, we determine the state estimate $\overline{\mathbf{x}}^a_n$ and its covariance $\mathbf{P}^a_n$ so that

$$J^o_{t_n}(\mathbf{x}) = [\mathbf{x} - \overline{\mathbf{x}}^a_n]^T (\mathbf{P}^a_n)^{-1} [\mathbf{x} - \overline{\mathbf{x}}^a_n] + c'$$

for some constant c'. Equating the terms of degree 2 in $\mathbf{x}$, we get

$$\mathbf{P}^a_n = \left[(\mathbf{P}^b_n)^{-1} + \mathbf{H}_n^T \mathbf{R}_n^{-1} \mathbf{H}_n \right]^{-1}. \tag{8}$$

Equating the terms of degree 1, we get

$$\overline{\mathbf{x}}^a_n = \mathbf{P}^a_n \left[(\mathbf{P}^b_n)^{-1} \overline{\mathbf{x}}^b_n + \mathbf{H}_n^T \mathbf{R}_n^{-1} \mathbf{y}^o_n \right]. \tag{9}$$

Notice that when the model state is observed directly, $\mathbf{H}_n$ is the identity matrix, and equation (9) expresses the analysis state estimate as a weighted average of the background state estimate and the observations, weighted according to the inverse covariance of each.

Equations (8) and (9) can be written in many different but equivalent forms, and it will be useful later to rewrite both of them now. Using (8) to eliminate $(\mathbf{P}^b_n)^{-1}$ from (9) yields

$$\overline{\mathbf{x}}^a_n = \overline{\mathbf{x}}^b_n + \mathbf{P}^a_n \mathbf{H}_n^T \mathbf{R}_n^{-1} (\mathbf{y}^o_n - \mathbf{H}_n \overline{\mathbf{x}}^b_n). \tag{10}$$

The matrix $\mathbf{P}^a_n \mathbf{H}_n^T \mathbf{R}_n^{-1}$ is called the "Kalman gain". It multiplies the difference between the observations at time t_n and the values predicted by the background state estimate to yield the increment between the background and analysis state estimates. Next, multiplying (8) on the right by $(\mathbf{P}^b_n)^{-1} \mathbf{P}^b_n$ and combining the inverses yields

$$\mathbf{P}^a_n = (\mathbf{I} + \mathbf{P}^b_n \mathbf{H}_n^T \mathbf{R}_n^{-1} \mathbf{H}_n)^{-1} \mathbf{P}^b_n. \tag{11}$$

This expression provides a more efficient way than (8) to compute $\mathbf{P}^a_n$, since it does not require inverting $\mathbf{P}^b_n$.

Initialization. The derivation above of the Kalman filter avoids the issue of how to initialize the iteration. To solve the best fit problem we originally posed, we should make no assumptions about the system state prior to the analysis at time t_1. Formally we can regard the background covariance $\mathbf{P}^b_1$ to be infinite, and for $n = 1$ use (8) and (9) with $(\mathbf{P}^b_1)^{-1} = \mathbf{0}$. This works if there are enough observations at time t_1 to determine (aside from the measurement errors) the system state; that is, if $\mathbf{H}_1$ has rank equal to the number of model variables m. The analysis then determines $\overline{\mathbf{x}}^a_1$ in the appropriate least-square sense. However, if there are not enough observations, then the matrix to be

inverted in (8) does not have full rank. To avoid this difficulty, one can assume a prior background distribution at time t_1, with $\mathbf{P}_1^b$ reasonably large but finite. This adds a small quadratic term to the cost function being minimized, but with sufficient observations over time, the effect of this term on the analysis at time t_n decreases in significance as n increases.

2.2 Nonlinear Scenario: Ensemble Kalman Filtering

Many approaches to data assimilation for nonlinear problems are based on the Kalman filter, or at least on minimizing a cost function similar to (7). At a minimum, a nonlinear model forces a change in the forecast equations (5) and (6), while nonlinear observation operators H_n force a change in the analysis equations (10) and (11). The extended Kalman filter (see, for example, [25]) computes $\overline{\mathbf{x}}_n^b = M_{t_{n-1},t_n}(\overline{\mathbf{x}}_{n-1}^a)$ using the nonlinear model, but computes $\mathbf{P}_n^b$ using the linearization $\mathbf{M}_{t_{n-1},t_n}$ of M_{t_{n-1},t_n} around $\overline{\mathbf{x}}_{n-1}^a$. The analysis then uses the linearization $\mathbf{H}_n$ of H_n around $\overline{\mathbf{x}}_n^b$. This approach is problematic for complex, high-dimensional models such as a global weather model for (at least) two reasons. First, it is not easy to linearize such a model. Second, when the number of model variables m is several million, computations involving the $m \times m$ covariance matrices are very expensive.

Approaches used in operational weather forecasting generally eliminate, for pragmatic reasons, the time iteration of the Kalman filter. The U.S. National Weather Service performs data assimilation every 6 hours using the "3D-Var" method [32, 38], in which the background covariance $\mathbf{P}_n^b$ in (7) is replaced by a constant matrix $\mathbf{B}$ representing typical uncertainty in a 6-hour forecast. This simplification allows the analysis to be formulated in a manner that precomputes the most expensive matrix operations, so that they do not have to be repeated at each time t_n. The 3D-Var cost function also allows a nonlinear observation operator H_n, and is minimized numerically to produce the analysis state estimate $\mathbf{x}_n^a$.

The "4D-Var" method [31, 42] used by the European Centre for Medium-Range Weather Forecasts uses a cost function that includes a constant-covariance background term as in 3D-Var, together with a sum like (2) accounting for the observations collected over a 12-hour time window. Again the cost function is minimized numerically; this procedure is computationally intensive, because computing the gradient of the 4D-Var cost function requires integrating both the nonlinear model and its linearization over the 12-hour window, and this procedure is repeated until a satisfactory approximation to the minimum is found.

The key idea of ensemble Kalman filtering [7, 9] is to choose at time t_{n-1} an ensemble of initial conditions whose spread around $\overline{\mathbf{x}}_{n-1}^a$ characterizes the analysis covariance $\mathbf{P}_{n-1}^a$, propagate each ensemble member using the nonlinear model, and compute $\mathbf{P}_n^b$ based on the resulting ensemble at time t_n. Thus like the extended Kalman filter, the (approximate) uncertainty in the state estimate is propagated from one analysis to the next, unlike 3D-Var (which does not propagate the uncertainty at all) or 4D-Var (which propagates it only with the time window over which the cost function is

minimized). Furthermore, ensemble Kalman filters do this without requiring a linearized model. On the other hand, 4D-Var (with "weak constraint") allows a wide variety of model error terms to be incorporated into the cost function.

In spite of their differences, though, we emphasize that in the absence of computational limitations, 4D-Var and ensemble Kalman filtering should be able to produce similar results because they both seek to minimize the same type of cost function. Indeed, in a perfect model scenario [11], we obtained similar results with both methods when we used a sufficiently long time window for 4D-Var and when we used enough ensemble members and performed the analysis sufficiently frequently in a 4D version (described in Section 4 of this article) of our LETKF. In atmospheric data assimilation, ensemble Kalman filtering has not yet equaled the best results using 4D-Var, but it has begun to achieve results that compare favorably with operational 3D-Var results [22, 46, 34]; see also Section 5.

Perhaps the most important difference between ensemble Kalman filtering and the other methods described above is that the former quantifies uncertainty only in the space spanned by the ensemble. Assuming that computational resources restrict the number of ensemble members k to be much smaller than the number of model variables m, this can be a severe limitation. On the other hand, if this limitation can be overcome (see the section on "Localization" below), then the analysis can be performed in a much lower-dimensional space (k versus m). Thus, ensemble Kalman filtering has the potential to be more computationally efficient than the other methods. Indeed, the main point of this article is to describe how to do ensemble Kalman filtering efficiently without sacrificing accuracy.

Notation. We start with an ensemble $\{\mathbf{x}_{n-1}^{a(i)} : i = 1, 2, \ldots, k\}$ of m-dimensional model state vectors at time t_{n-1}. One approach would be to let one of the ensemble members represent the best estimate of the system state, but here we assume the ensemble to be chosen so that its average represents the analysis state estimate. We evolve each ensemble member according to the nonlinear model to obtain a background ensemble $\{\mathbf{x}_n^{b(i)} : i = 1, 2, \ldots, k\}$ at time t_n:

$$\mathbf{x}_n^{b(i)} = M_{t_{n-1},t_n}(\mathbf{x}_{n-1}^{a(i)}).$$

For the rest of this article, we will discuss what to do at the analysis time t_n, and so we now drop the subscript n. Thus, for example, H and $\mathbf{R}$ will represent respectively the observation operator and the observation error covariance matrix at the analysis time. Let ℓ be the number of scalar observations used in the analysis.

For the background state estimate and its covariance, we use the sample mean and covariance of the background ensemble:

$$\overline{\mathbf{x}}^b = k^{-1} \sum_{i=1}^{k} \mathbf{x}^{b(i)},$$

$$\mathbf{P}^b = (k-1)^{-1} \sum_{i=1}^{k} (\mathbf{x}^{b(i)} - \overline{\mathbf{x}}^b)(\mathbf{x}^{b(i)} - \overline{\mathbf{x}}^b)^T = (k-1)^{-1} \mathbf{X}^b (\mathbf{X}^b)^T, \tag{12}$$

where $\mathbf{X}^b$ is the $m \times k$ matrix whose ith column is $\mathbf{x}^{b(i)} - \overline{\mathbf{x}}^b$. (Notice that the rank of $\mathbf{P}^b$ is equal to the rank of $\mathbf{X}^b$, which is at most $k-1$ because the sum of its columns is $\mathbf{0}$. Thus, the ensemble size limits the rank of the background covariance matrix.) The analysis must determine not only an state estimate $\overline{\mathbf{x}}^a$ and covariance $\mathbf{P}^a$, but also an ensemble $\{\mathbf{x}^{a(i)} : i = 1, 2, \ldots, k\}$ with the appropriate sample mean and covariance:

$$\overline{\mathbf{x}}^a = k^{-1} \sum_{i=1}^{k} \mathbf{x}^{a(i)},$$

$$\mathbf{P}^a = (k-1)^{-1} \sum_{i=1}^{k} (\mathbf{x}^{a(i)} - \overline{\mathbf{x}}^a)(\mathbf{x}^{a(i)} - \overline{\mathbf{x}}^a)^T = (k-1)^{-1} \mathbf{X}^a (\mathbf{X}^a)^T, \tag{13}$$

where $\mathbf{X}^a$ is the $m \times k$ matrix whose ith column is $\mathbf{x}^{a(i)} - \overline{\mathbf{x}}^a$.

In Section 2.3, we will describe how to determine $\overline{\mathbf{x}}^a$ and $\mathbf{P}^a$ for a (possibly) nonlinear observation operator H in a way that agrees with the Kalman filter equations (10) and (11) in the case that H is linear.

Choice of analysis ensemble. Once $\overline{\mathbf{x}}^a$ and $\mathbf{P}^a$ are specified, there are still many possible choices of an analysis ensemble (or equivalently, a matrix $\mathbf{X}^a$ that satisfies (13) and the sum of whose columns is zero). Many ensemble Kalman filters have been proposed, and one of the main differences among them is how the analysis ensemble is chosen. The simplest approach is to apply the Kalman filter update (10) separately to each background ensemble member (rather than their mean) to get the corresponding analysis ensemble member. However, this results in an analysis ensemble whose sample covariance is smaller than the analysis covariance $\mathbf{P}^a$ given by (11), unless the observations are artificially perturbed so that each ensemble member is updated using different random realization of the perturbed observations [5, 20]. Ensemble square-root filters [1, 45, 4, 43, 36, 37] instead use more involved but deterministic algorithms to generate an analysis ensemble with the desired sample mean and covariance. As such, their analyses coincide exactly with the Kalman filter equations in the linear scenario of the previous section. We will use this deterministic approach in Section 2.3.

Localization. Another important issue in ensemble Kalman filtering of spatiotemporally chaotic systems is spatial localization. If the ensemble has k members, then the background covariance matrix $\mathbf{P}^b$ given by (12) describes nonzero uncertainty only in the (at most) k-dimensional subspace spanned by the ensemble, and a global analysis will allow adjustments to the system state only in this subspace. If the system is high-dimensionally unstable — more precisely, if it has more than k positive Lyapunov exponents — then forecast errors will grow in directions not accounted for by

the ensemble, and these errors will not be corrected by the analysis. On the other hand, in a sufficiently small local region, the system may behave like a low-dimensionally unstable system driven by the dynamics in neighboring regions; such behavior was observed for a global weather model in [39, 35]. Performing the analysis locally requires the ensemble to represent uncertainty in only the local unstable space. By allowing the local analyses to choose different linear combinations of the ensemble members in different regions, the global analysis is not confined to the k-dimensional ensemble space and instead explores a much higher-dimensional space [12, 36, 37]. Another explanation for the necessity of localization for spatiotemporally chaotic systems is that the limited sample size provided by an ensemble will produce spurious correlations between distant locations in the background covariance matrix $\mathbf{P}^b$ [20, 17]. Unless they are suppressed, these spurious correlations will cause observations from one location to affect, in an essentially random manner, the analysis in locations an arbitrarily large distance away. If the system has a characteristic "correlation distance", then the analysis should ignore ensemble correlations over much larger distances. In addition to providing better results in many cases, localization allows the analysis to be done more efficiently as a parallel computation [30, 36, 37].

Localization is generally done either explicitly, considering only the observations from a region surrounding the location of the analysis [29, 20, 30, 1, 36, 37], or implicitly, by multiplying the entries in $\mathbf{P}^b$ by a distance-dependent function that decays to zero beyond a certain distance, so that observations do not affect the model state beyond that distance [21, 17, 45]. We will follow the explicit approach here, doing a separate analysis for each spatial grid point of the model. (Our use of "grid point" assumes the model to be a discretization of a partial differential equation, or otherwise to be defined on a lattice, but the method is also applicable to systems with other geometries.) The choice of which observations to use for each grid point is up to the user of the method, and a good choice will depend both on the particular system being modeled and on the size of the ensemble (more ensemble members generally allow more distant observations to be used gainfully). It is important, however, that most of the observations used in the analysis at a particular grid point also be used in the analysis at neighboring grid points. This ensures that the analysis ensemble does not change suddenly from one grid point to the next. For an atmospheric model, a reasonable approach is to use observations within a cylinder of a given radius and height centered at the analysis grid point and to determine empirically which values of the radius and height work best. At its simplest, the method we describe gives all of the chosen observations equal influence on the analysis, but we will also describe how to make their influence decay gradually toward zero as their distance from the analysis location increases.

Initial background ensemble. A common method for generating a background ensemble to use at the first analysis time is to run the model for a while and to select model states at different ran-

domly chosen times. The intent of this method is for the initial background ensemble to be sampled from a climatological distribution. This is a reasonable choice for the background distribution when no prior observations are available.

2.3 LETKF: A Local Ensemble Transform Kalman Filter

We now describe an efficient means of performing the analysis that transforms a background ensemble $\{\mathbf{x}^{b(i)} : i = 1,2,\ldots,k\}$ into an appropriate analysis ensemble $\{\mathbf{x}^{a(i)} : i = 1,2,\ldots,k\}$, using the notation defined above. We assume that the number of ensemble members k is smaller than both the number of model variables m and the number of observations ℓ,[2] even when localization has reduced the effective values of m and ℓ considerably compared to a global analysis. (In this section we assume that the choice of observations to use for the local analysis has been performed already, and consider $\mathbf{y}^o$, H, and $\mathbf{R}$ to be truncated to these observations; as such, correlations between errors in the chosen observations and errors in other observations are ignored.) Most of the analysis takes place in a k-dimensional space, with as few operations as possible in the model and observation spaces.

Formally, we want the analysis mean $\overline{\mathbf{x}}^a$ to minimize the Kalman filter cost function (7), modified to allow for a nonlinear observation operator H:

$$J(\mathbf{x}) = (\mathbf{x} - \overline{\mathbf{x}}^b)^T (\mathbf{P}^b)^{-1} (\mathbf{x} - \overline{\mathbf{x}}^b) + [\mathbf{y}^o - H(\mathbf{x})]^T \mathbf{R}^{-1} [\mathbf{y}^o - H(\mathbf{x})]. \tag{14}$$

However, the $m \times m$ background covariance matrix $\mathbf{P}^b = (k-1)^{-1}\mathbf{X}^b(\mathbf{X}^b)^T$ has rank at most $k-1$, and is therefore not invertible. Nonetheless, as a symmetric matrix, it is one-to-one on its column space S, which is also the column space of $\mathbf{X}^b$, or in other words the space spanned by the background ensemble perturbations. So in this sense, $(\mathbf{P}^b)^{-1}$ is well-defined on S. Then J is also well-defined for $\mathbf{x} - \overline{\mathbf{x}}^b$ in S, and the minimization can be carried out in this subspace.[3] As we have said, this reduced dimensionality is an advantage from the point of view of efficiency, though the restriction of the analysis mean to S is sure to be detrimental if k is too small.

To perform the analysis on S, we must choose an appropriate coordinate system. A natural approach is to use the singular vectors of $\mathbf{X}^b$ (the eigenvectors of $\mathbf{P}^b$) to form a basis for S [1, 36, 37]. Here we avoid this step by using instead the columns of $\mathbf{X}^b$ to span S, as in [4]. One conceptual difficulty in this approach is that the sum of these columns is zero, so they are necessarily linearly

[2] This assumption is only expository; our algorithm does not require an upper bound on k, but it is less efficient than doing the analysis in model space if $m < k$ or in observation space if $\ell < k$.

[3] Considerably more general cost functions can be used, at the expense of having to perform the minimization numerically in the ensemble space S. Since this space is relatively low-dimensional, this approach is still feasible even for high-dimensional systems. In [18] we use a non-quadratic background term within the LETKF framework, and [48] uses a similar approach to allow a non-quadratic observation term.

dependent. We could assume the first $k-1$ columns to be independent and use them as a basis, but this assumption is unnecessary and clutters the resulting equations. Instead, we regard $\mathbf{X}^b$ as a linear transformation from a k-dimensional space $\tilde{S}$ onto S, and perform the analysis in $\tilde{S}$. Let $\mathbf{w}$ denote a vector in $\tilde{S}$; then $\mathbf{X}^b\mathbf{w}$ belongs to the space S spanned by the background ensemble perturbations, and $\mathbf{x} = \overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w}$ is the corresponding model state.

Notice that if $\mathbf{w}$ is a Gaussian random vector with mean $\mathbf{0}$ and covariance $(k-1)^{-1}\mathbf{I}$, then $\mathbf{x} = \overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w}$ is Gaussian with mean $\overline{\mathbf{x}}^b$ and covariance $\mathbf{P}^b = (k-1)^{-1}\mathbf{X}^b(\mathbf{X}^b)^T$. This motivates the cost function

$$\tilde{J}(\mathbf{w}) = (k-1)\mathbf{w}^T\mathbf{w} + [\mathbf{y}^o - H(\overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w})]^T\mathbf{R}^{-1}[\mathbf{y}^o - H(\overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w})] \tag{15}$$

on $\tilde{S}$. In particular, we claim that if $\overline{\mathbf{w}}^a$ minimizes $\tilde{J}$, then $\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^b + \mathbf{X}^b\overline{\mathbf{w}}^a$ minimizes the cost function J. Substituting the change of variables formula into (14) and using (12) yields the identity

$$\tilde{J}(\mathbf{w}) = (k-1)\mathbf{w}^T(\mathbf{I} - (\mathbf{X}^b)^T[\mathbf{X}^b(\mathbf{X}^b)^T]^{-1}\mathbf{X}^b)\mathbf{w} + J(\overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w}). \tag{16}$$

The matrix $\mathbf{I} - (\mathbf{X}^b)^T[\mathbf{X}^b(\mathbf{X}^b)^T]^{-1}\mathbf{X}^b$ is the orthogonal projection onto the null space N of $\mathbf{X}^b$. (Generally N will be one-dimensional, spanned by the vector $(1,1,\ldots,1)^T$, but it could be higher-dimensional.) Thus, the first term on the right side of (16) depends only on the component of $\mathbf{w}$ in N, while the second term depends only on its component in the space orthogonal to N (which is in one-to-one correspondence with S under $\mathbf{X}^b$). Thus if $\overline{\mathbf{w}}^a$ minimizes $\tilde{J}$, then it must be orthogonal to N, and the corresponding vector $\overline{\mathbf{x}}^a$ minimizes J.

A cost function equivalent to (15) appears in [33]. More generally, implementations of 3D-Var and 4D-Var commonly use a preconditioning step that expresses the cost function in a form similar to (15).

Nonlinear observations. The most accurate way to allow for a nonlinear observation operator H would be to numerically minimize $\tilde{J}$ in the k-dimensional space $\tilde{S}$, as in [48]. If H is sufficiently nonlinear, then $\tilde{J}$ could have multiple minima, but a numerical minimization using $\mathbf{w} = \mathbf{0}$ (corresponding to $\mathbf{x} = \overline{\mathbf{x}}^b$) as an initial guess would still be a reasonable approach. Having determined $\overline{\mathbf{w}}^a$ in this manner, one would compute the analysis covariance $\tilde{\mathbf{P}}^a$ in $\tilde{S}$ from the second partial derivatives of $\tilde{J}$ at $\overline{\mathbf{w}}^a$, then use $\mathbf{X}^b$ to transform the analysis results into the model space, as below. But to formulate the analysis more explicitly, we now linearize H about the background ensemble mean $\overline{\mathbf{x}}^b$. Of course, if H is linear then we will find the minimum of $\tilde{J}$ exactly. And if the spread of the background ensemble is not too large, the linearization should be a decent approximation, similar to the approximation we have already made that a linear combination of background ensemble members is also a plausible background model state.

Since we only need to evaluate H in the ensemble space (or equivalently to evaluate $H(\overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w})$ for $\mathbf{w}$ in $\tilde{S}$), the simplest way to linearize H is to apply it to each of the ensemble members $\mathbf{x}^{b(i)}$ and interpolate. To this end, we define an ensemble $\mathbf{y}^{b(i)}$ of background observation vectors by

$$\mathbf{y}^{b(i)} = H(\mathbf{x}^{b(i)}). \tag{17}$$

We define also their mean $\overline{\mathbf{y}}^b$, and the $\ell \times k$ matrix $\mathbf{Y}^b$ whose ith column is $\mathbf{y}^{b(i)} - \overline{\mathbf{y}}^b$. We then make the linear approximation

$$H(\overline{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w}) \approx \overline{\mathbf{y}}^b + \mathbf{Y}^b\mathbf{w}. \tag{18}$$

The same approximation is used in, for example, [21], and is equivalent to the joint state-observation space method in [1].

Analysis. The linear approximation we have just made yields the quadratic cost function

$$\tilde{J}^*(\mathbf{w}) = (k-1)\mathbf{w}^T\mathbf{w} + [\mathbf{y}^o - \overline{\mathbf{y}}^b - \mathbf{Y}^b\mathbf{w}]^T\mathbf{R}^{-1}[\mathbf{y}^o - \overline{\mathbf{y}}^b - \mathbf{Y}^b\mathbf{w}]. \tag{19}$$

This cost function is in the form of the Kalman filter cost function (7), using the background mean $\overline{\mathbf{w}}^b = \mathbf{0}$ and background covariance $\tilde{\mathbf{P}}^b = (k-1)^{-1}\mathbf{I}$, with $\mathbf{Y}^b$ playing the role of the observation operator. The analogues of the analysis equations (10) and (11) are then

$$\overline{\mathbf{w}}^a = \tilde{\mathbf{P}}^a(\mathbf{Y}^b)^T\mathbf{R}^{-1}(\mathbf{y}^o - \overline{\mathbf{y}}^b), \tag{20}$$

$$\tilde{\mathbf{P}}^a = [(k-1)\mathbf{I} + (\mathbf{Y}^b)^T\mathbf{R}^{-1}\mathbf{Y}^b]^{-1}. \tag{21}$$

In model space, the analysis mean and covariance are then

$$\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^b + \mathbf{X}^b\overline{\mathbf{w}}^a, \tag{22}$$

$$\mathbf{P}^a = \mathbf{X}^b\tilde{\mathbf{P}}^a(\mathbf{X}^b)^T. \tag{23}$$

To initialize the ensemble forecast that will produce the background for the next analysis, we must choose an analysis ensemble whose sample mean and covariance are equal to $\overline{\mathbf{x}}^a$ and $\mathbf{P}^a$. As mentioned above, this amounts to choosing a matrix $\mathbf{X}^a$ so that the sum of its columns is zero and (13) holds. Then one can form the analysis ensemble by adding $\overline{\mathbf{x}}^a$ to each of the columns of $\mathbf{X}^a$.

Symmetric square root. Our choice of analysis ensemble is described by $\mathbf{X}^a = \mathbf{X}^b\mathbf{W}^a$, where

$$\mathbf{W}^a = [(k-1)\tilde{\mathbf{P}}^a]^{1/2} \tag{24}$$

and by the $1/2$ power of a symmetric matrix we mean its symmetric square root. Then $\tilde{\mathbf{P}}^a = (k-1)^{-1}\mathbf{W}^a(\mathbf{W}^a)^T$, and (13) follows from (23). The use of the symmetric square root to determine $\mathbf{W}^a$ from $\tilde{\mathbf{P}}^a$ (as compared to, for example, a Cholesky factorization, or the choice described

in [4]), is important for two main reasons. First, as we will see below, it ensures that the sum of the columns of $\mathbf{X}^a$ is zero, so that the analysis ensemble has the correct sample mean (this is also shown for the symmetric square root in [44]). Second, it ensures that $\mathbf{W}^a$ depends continuously on $\tilde{\mathbf{P}}^a$; while this may be a desirable property in general, it is crucial in a local analysis scheme, so that neighboring grid points with slightly different matrices $\tilde{\mathbf{P}}^a$ do not yield very different analysis ensembles. Another potentially desirable property of the symmetric square root is that it minimizes the (mean-square) distance between $\mathbf{W}^a$ and the identity matrix, so that the analysis ensemble perturbations are in this sense as close as possible to the background ensemble perturbations subject to the constraint on their sample covariance [36, 37].

To see that the sum of the columns of $\mathbf{X}^a$ is zero, we express this condition as $\mathbf{X}^a\mathbf{v} = \mathbf{0}$, where $\mathbf{v}$ is a column vector of k ones: $\mathbf{v} = (1,1,\ldots,1)^T$. Notice that by (21), $\mathbf{v}$ is an eigenvector of $\tilde{\mathbf{P}}^a$ with eigenvalue $(k-1)^{-1}$:

$$(\tilde{\mathbf{P}}^a)^{-1}\mathbf{v} = [(k-1)\mathbf{I} + (\mathbf{Y}^b)^T\mathbf{R}^{-1}\mathbf{Y}^b]\mathbf{v} = (k-1)\mathbf{v},$$

because the sum of the columns of $\mathbf{Y}^b$ is zero. Then by (24), $\mathbf{v}$ is also an eigenvector of $\mathbf{W}^a$ with eigenvalue 1. Since the sum of the columns of $\mathbf{X}^b$ is zero, $\mathbf{X}^a\mathbf{v} = \mathbf{X}^b\mathbf{W}^a\mathbf{v} = \mathbf{X}^b\mathbf{v} = \mathbf{0}$ as desired.

Finally, notice that we can form the analysis ensemble first in $\tilde{S}$ by adding $\bar{\mathbf{w}}^a$ to each of the columns of $\mathbf{W}^a$; let $\{\mathbf{w}^{a(i)}\}$ be the columns of the resulting matrix. These "weight" vectors specify what linear combinations of the background ensemble perturbations to add to the background mean to obtain the analysis ensemble in model space:

$$\mathbf{x}^{a(i)} = \bar{\mathbf{x}}^b + \mathbf{X}^b\mathbf{w}^{a(i)}. \tag{25}$$

Local implementation. Notice that once the background ensemble has been used to form $\bar{\mathbf{y}}^b$ and $\mathbf{Y}^b$, it is no longer needed in the analysis, except in (25) to translate the results from $\tilde{S}$ to model space. This point is useful to keep in mind when implementing a local filter that computes a separate analysis for each model grid point. In principle, one should form a global background observation ensemble $\mathbf{y}^{b(i)}_{[g]}$ from the global background vectors, though in practice this can be done locally when the global observation operator $H_{[g]}$ uses local interpolation. After the background observation ensemble is formed, the analyses at different grid points are completely independent of each other and can be computed in parallel. The observations chosen for a given local analysis dictate which coordinates of $\mathbf{y}^{b(i)}_{[g]}$ are used to form the local background observation ensemble $\mathbf{y}^{b(i)}$, and the analysis in $\tilde{S}$ produces the local weight vectors $\{\mathbf{w}^{a(i)}\}$. Computing the analysis ensemble $\{\mathbf{x}^{a(i)}\}$ for the analysis grid point using (25) then requires only using the background model states at that grid point.

As long as the sets of observations used for a pair of neighboring grid points overlap heavily, the local weight vectors $\{\mathbf{w}^{a(i)}\}$ for the two grid points are similar. In a region over which the weight

vectors do not change much, the analysis ensemble members are approximately linear combinations of the background ensemble members, and thus should represent reasonably "physical" initial conditions for the forecast model. However, if the model requires of its initial conditions high-order smoothness and/or precise conformance to an conservation law, it may be necessary to post-process the analysis ensemble members to smooth them and/or project them onto the manifold determined by the conserved quantities before using them as initial conditions (this procedure is often called "balancing" in geophysical data assimilation).

In other localization approaches [21, 17, 45], the influence of an observation at a particular point on the analysis at a particular model grid point decays smoothly to zero as the distance between the two points increases. A similar effect can be achieved here by multiplying the entries in the inverse observation error covariance matrix $\mathbf{R}^{-1}$ by a factor that decays from one to zero as the distance of the observations from the analysis grid point increases. This "smoothed localization" corresponds to gradually increasing the uncertainty assigned to the observations until beyond a certain distance they have infinite uncertainty and therefore no influence on the analysis.

Covariance inflation. In practice, an ensemble Kalman filter that adheres strictly to the Kalman filter equations (10) and (11) may fail to synchronize with the "true" system trajectory that produces the observations. One reason for this is model error, but even with a perfect model, the filter tends to underestimate the uncertainty in its state estimate [45]. Regardless of the cause, underestimating the uncertainty leads to overconfidence in the background state estimate, and, hence, the analysis underweights the observations. If the discrepancy becomes too large over time, the observations are essentially ignored by the analysis, and the dynamics of the data assimilation system become decoupled from the truth.

Generally this tendency is countered by an *ad hoc* procedure (with at least one tunable parameter) that inflates either the background covariance or the analysis covariance during each data assimilation cycle. (Doing this is analogous to adding a model error covariance term to the right side of (6), as is usually done in the Kalman filter.) One "hybrid" approach adds a multiple of the background covariance matrix $\mathbf{B}$ from the 3D-Var method to the ensemble background covariance prior to the analysis [16]. "Multiplicative inflation" [2, 17] instead multiplies the background covariance matrix (or equivalently, the perturbations of the background ensemble members from their mean) by a constant factor larger than one. "Additive inflation" adds a small multiple of the identity matrix to either the background covariance or the analysis covariance during each cycle [36, 37]. Finally, if one chooses the analysis ensemble in such a way that each member has a corresponding member of the background ensemble, then one can inflate the analysis ensemble by "relaxation" toward the background ensemble: replacing each analysis perturbation from the mean by a weighted average of itself and the corresponding background perturbation [47].

Within the framework described in this article, the hybrid approach is not feasible because it requires the analysis to consider uncertainty outside the space spanned by the background ensemble. However, once the analysis ensemble is formed, one could develop a means of inflating it in directions (derived from the 3D-Var background covariance matrix $\mathbf{B}$ or otherwise) outside the ensemble space so that uncertainty in these directions is reflected in the background ensemble at the next analysis step. Additive inflation is feasible, but requires substantial additional computation to determine the adjustment necessary in the k-dimensional space $\tilde{S}$ that corresponds to adding a multiple of the identity matrix to the model space covariance $\mathbf{P}^b$ or $\mathbf{P}^a$. Relaxation is simple to implement, and is most efficiently done in $\tilde{S}$ by replacing $\mathbf{W}^a$ with a weighted average of it and the identity matrix.

Multiplicative inflation can be performed most easily on the analysis ensemble by multiplying $\mathbf{W}^a$ by an appropriate factor (namely $\sqrt{\rho}$, if one wants to multiply the analysis covariance by ρ). To perform multiplicative inflation on the background ensemble instead, one can multiply $\mathbf{X}^b$ by such a factor, and adjust the background ensemble $\{\mathbf{x}^{b(i)}\}$ accordingly before applying the observation operator H to form the background observation ensemble $\{\mathbf{y}^{b(i)}\}$. A more efficient approach, which is equivalent if H is linear, is simply to replace $(k-1)\mathbf{I}$ by $(k-1)\mathbf{I}/\rho$ in (21), since $(k-1)\mathbf{I}$ is the inverse of the background covariance matrix $\tilde{\mathbf{P}}^b$ in the k-dimensional space $\tilde{S}$. One can check that this has the same effect on the analysis mean $\overline{\mathbf{x}}^a$ and covariance $\mathbf{P}^a$ as multiplying $\mathbf{X}^b$ and $\mathbf{Y}^b$ by $\sqrt{\rho}$. If ρ is close to one, this is a good approximation to inflating the background ensemble before applying the observation operator even when this operator is nonlinear.

Multiplicative inflation of the background covariance can be thought of as applying a discount factor to the influence of past observations on the current analysis. Since this discount factor is applied during each analysis, the cumulative effect is that the influence of an observation on future analyses decays exponentially with time. The inflation factor determines the time scale over which observations have a significant influence on the analysis. Other methods of covariance inflation have a similar effect, causing observations from sufficiently far in the past essentially to be ignored. Thus, covariance inflation localizes the analysis in time. This effect is especially desirable in the presence of model error, because then the model can only reliably propagate information provided by the observations for a limited period of time.

3 Efficient Computation of the Analysis

Here is a step-by-step description of how to perform the analysis described in the previous section, designed for efficiency both in ease of implementation and in the amount of computation and memory usage. Of course there are some trade-offs between these objectives, so in each step we first describe the simplest approach and then in some cases mention alternate approaches and possible gains in computational efficiency. We also give a rough accounting of the computational complexity

of each step, and at the end discuss the overall computational complexity. After that, we describe an approach that in some cases will produce a significantly faster analysis, at the expense of more memory usage and more difficult implementation, by reorganizing some of the steps. As before, we use "grid point" in this section to mean a spatial location in the forecast model, whether or not the model is actually based on a grid geometry; we use "array" to mean a vector or matrix. The use of "columns" and "rows" below is for exposition only; one should of course store arrays in whatever manner is most efficient for one's computing environment.

The inputs to the analysis are a background ensemble of $m_{[g]}$-dimensional model state vectors $\{\mathbf{x}^{b(i)}_{[g]} : i = 1, 2, \ldots, k\}$, a function $H_{[g]}$ from the $m_{[g]}$-dimensional model space to the $\ell_{[g]}$-dimensional observation space, an $\ell_{[g]}$-dimensional vector $\mathbf{y}^{o}_{[g]}$ of observations, and an $\ell_{[g]} \times \ell_{[g]}$ observation error covariance matrix $\mathbf{R}_{[g]}$. The subscript g here signifies that these inputs reflect the global model state and all available observations, from which a local subset should be chosen for each local analysis. How to choose which observations to use is entirely up to the user of this method, but a reasonable general approach is to choose those observations made within a certain distance of the grid point at which one is doing the local analysis and determine empirically which value of the cutoff distance produces the "best" results. If one deems localization to be unnecessary in a particular application, then one can ignore the distinction between local and global, and skip Steps 3 and 9 below.

Steps 1 and 2 are essentially global operations, but may be done locally in a parallel implementation. Steps 3–8 should be performed separately for each local analysis (generally this means for each grid point, but see the parenthetical comment at the end of Step 3). Step 9 simply combines the results of the local analyses to form a global analysis ensemble $\{\mathbf{x}^{a(i)}_{[g]}\}$, which is the final output of the analysis.

1. *Apply $H_{[g]}$ to each $\mathbf{x}^{b(i)}_{[g]}$ to form the global background observation ensemble $\{\mathbf{y}^{b(i)}_{[g]}\}$, and average the latter vectors to get the $\ell_{[g]}$-dimensional column vector $\overline{\mathbf{y}}^{b}_{[g]}$. Subtract this vector from each $\{\mathbf{y}^{b(i)}_{[g]}\}$ to form the columns of the $\ell_{[g]} \times k$ matrix $\mathbf{Y}^{b}_{[g]}$.* (This subtraction can be done "in place", since the vectors $\{\mathbf{y}^{b(i)}_{[g]}\}$ are no longer needed.) This requires k applications of H, plus $2k\ell_{[g]}$ (floating-point) operations. If H is an interpolation operator that requires only a few model variables to compute each observation variable, then the total number of operations for this step is proportional to $k\ell_{[g]}$ times the average number of model variables required to compute each scalar observation.

2. *Average the vectors $\{\mathbf{x}^{b(i)}_{[g]}\}$ to get the $m_{[g]}$-dimensional vector $\overline{\mathbf{x}}^{b}_{[g]}$, and subtract this vector from each $\mathbf{x}^{b(i)}_{[g]}$ to form the columns of the $m_{[g]} \times k$ matrix $\mathbf{X}^{b}_{[g]}$.* (Again the subtraction can be done "in place"; the vectors $\{\mathbf{x}^{b(i)}_{[g]}\}$ are no longer needed). This step requires a total of $2km_{[g]}$ operations. (If H is linear, one can equivalently perform Step 2 before Step 1, and obtain $\overline{\mathbf{y}}^{b}_{[g]}$

and $\mathbf{Y}^b_{[g]}$ by applying H to $\vec{\mathbf{x}}^b_{[g]}$ and $\mathbf{X}^b_{[g]}$.)

3. This step selects the necessary data for a given grid point (whether it is better to form the local arrays described below explicitly or select them later as needed from the global arrays depends on one's implementation). *Select the rows of $\vec{\mathbf{x}}^b_{[g]}$ and $\mathbf{X}^b_{[g]}$ corresponding to the given grid point, forming their local counterparts: the m-dimensional vector $\vec{\mathbf{x}}^b$ and the $m \times k$ matrix $\mathbf{X}^b$, which will be used in Step 8. Likewise, select the rows of $\vec{\mathbf{y}}^b_{[g]}$ and $\mathbf{Y}^b_{[g]}$ corresponding to the observations chosen for the analysis at the given grid point, forming the ℓ-dimensional vector $\vec{\mathbf{y}}^b$ and the $\ell \times k$ matrix $\mathbf{Y}^b$. Select the corresponding rows of $\mathbf{y}^o_{[g]}$ and rows and columns of $\mathbf{R}_{[g]}$ to form the ℓ-dimensional vector $\mathbf{y}^o$ and the $\ell \times \ell$ matrix $\mathbf{R}$.* (For a high-resolution model, it may be reasonable to use the same set of observations for multiple grid points, in which case one should select here the rows of $\mathbf{X}^b_{[g]}$ and $\vec{\mathbf{x}}^b_{[g]}$ corresponding to all of these grid points.)

4. *Compute the $k \times \ell$ matrix $\mathbf{C} = (\mathbf{Y}^b)^T \mathbf{R}^{-1}$.* If desired, one can multiply entries of $\mathbf{R}^{-1}$ or $\mathbf{C}$ corresponding to a given observation by a factor less than one to decrease (or greater than one to increase) its influence on the analysis. (For example, one can use a multiplier that depends on distance from the analysis grid point to discount observations near the edge of the local region from which they are selected; this will smooth the spatial influence of observations, as described in Section 2.3 under "Local Implementation".) Since this is the only step in which $\mathbf{R}$ is used, it may be most efficient to compute $\mathbf{C}$ by solving the linear system $\mathbf{R}\mathbf{C}^T = \mathbf{Y}^b$ rather than inverting $\mathbf{R}$. In some applications, $\mathbf{R}$ may be diagonal, but in others $\mathbf{R}$ will be block diagonal with each block representing a group of correlated observations. As long as the size of each block is relatively small, inverting $\mathbf{R}$ or solving the linear system above will not be computationally expensive. Furthermore, many or all of the blocks that make up $\mathbf{R}$ may be unchanged from one analysis time to the next, so that their inverses need not be recomputed each time. Based on these considerations, the number of operations required (at each grid point) for this step in a typical application should be proportional to $k\ell$, multiplied by a factor related to the typical block size of $\mathbf{R}$.

5. *Compute the $k \times k$ matrix $\tilde{\mathbf{P}}^a = \left[(k-1)\mathbf{I}/\rho + \mathbf{C}\mathbf{Y}^b\right]^{-1}$, as in (21).* Here $\rho > 1$ is a multiplicative covariance inflation factor, as described at the end of the previous section. Though trying some of the other approaches described there may be fruitful, a reasonable general approach is to start with $\rho > 1$ and increase it gradually until one finds a value that is optimal according to some measure of analysis quality. Multiplying $\mathbf{C}$ and $\mathbf{Y}^b$ requires less than $2k^2\ell$ operations, while the number of operations needed to invert the $k \times k$ matrix is proportional to k^3.

6. *Compute the $k \times k$ matrix $\mathbf{W}^a = [(k-1)\tilde{\mathbf{P}}^a]^{1/2}$, as in (24).* Again the number of operations required is proportional to k^3; it may be most efficient to compute the eigenvalues and eigen-

vectors of $\left[(k-1)\mathbf{I}/\rho + \mathbf{C}\mathbf{Y}^b\right]$ in the previous step and then use them to compute both $\tilde{\mathbf{P}}^a$ and $\mathbf{W}^a$.

7. *Compute the k-dimensional vector* $\bar{\mathbf{w}}^a = \tilde{\mathbf{P}}^a\mathbf{C}(\mathbf{y}^o - \bar{\mathbf{y}}^b)$*, as in (20), and add it to each column of* $\mathbf{W}^a$*, forming a* $k \times k$ *matrix whose columns are the analysis vectors* $\{\mathbf{w}^{a(i)}\}$. Computing the formula for $\bar{\mathbf{w}}^a$ from right-to-left, the total number of operations required for this step is less than $3k(\ell + k)$.

8. *Multiply* $\mathbf{X}^b$ *by each* $\mathbf{w}^{a(i)}$ *and add* $\bar{\mathbf{x}}^b$ *to get the analysis ensemble members* $\{\mathbf{x}^{a(i)}\}$ *at the analysis grid point, as in (25).* This requires $2k^2m$ operations.

9. *After performing Steps 3–8 for each grid point, the outputs of Step 8 form the global analysis ensemble* $\{\mathbf{x}^{a(i)}_{[g]}\}$.

We now summarize the overall computational complexity of the algorithm described above. If p is the number local analyses performed (equal to the number of grid points in the most basic approach), then notice that $pm = m_{[g]}$, while $p\bar{\ell} = q\ell_{[g]}$, where $\bar{\ell}$ is the average number of observations used in a local analysis and q is the average number of local analyses in which a particular observation is used. If $\bar{\ell}$ is large compared to k and m, then the most computationally expensive step is either Step 5, requiring approximately $2k^2p\bar{\ell} = 2k^2q\ell_{[g]}$ operations over all the local analyses, or Step 4, whose overall number of operations is proportional to $kp\bar{\ell} = kq\ell_{[g]}$, but with a proportionality constant dependent on the correlation structure of $\mathbf{R}_{[g]}$. In any case, as long as the typical number of correlated observations in a block of $\mathbf{R}_{[g]}$ remains constant, the overall computation time grows at most linearly with the total number $\ell_{[g]}$ of observations. It also grows at most linearly with the total number $m_{[g]}$ of model variables; if $m_{[g]}$ is large enough compared to $\ell[g]$, then the most expensive step is Step 8, with $2k^2m_{[g]}$ overall operations. The terms in the computation time that grow with the number of observations or number of model variables are at most quadratic in the number k of ensemble members. However, for a sufficiently large ensemble, the matrix operations in Steps 5 and 6 that take of order k^3 operations per local analysis, or k^3p operations overall, will become significant.

In Section 5, we present some numerical results for which we find the computation time indeed grows roughly quadratically with k, linearly with q, and sublinearly with $\ell_{[g]}$.

Batch processing of observations. Some of the steps above have a q-fold redundancy, in that computations involving a given observation are repeated over an average of q different local analyses. For a general observation error covariance matrix $\mathbf{R}_{[g]}$ this redundancy may be unavoidable, but it can be avoided as described below if the global observations can be partitioned into local groups (or "batches") numbered $1, 2, \ldots$ that meet the following conditions. First, all of the observations in

a given batch must be used in the exact same subset of the local analyses. Second, observations in different batches must have uncorrelated errors, so that each batch j corresponds to a block $\mathbf{R}_j$ in a block diagonal decomposition of $\mathbf{R}_{[g]}$. (These conditions can always met if $\mathbf{R}_{[g]}$ is diagonal, by making each batch consist of a single observation. However, as explained below, for efficiency one should make the batches as large as possible while still meeting the first condition.) Then at Step 3, instead of selecting (overlapping) submatrices of $\overline{\mathbf{y}}^b_{[g]}$, $\mathbf{Y}^b_{[g]}$, $\mathbf{y}^o_{[g]}$, and $\mathbf{R}_{[g]}$, for each grid point, let $\overline{\mathbf{y}}^b_j$, $\mathbf{Y}^b_j$, $\mathbf{y}^o_j$, represent the rows corresponding to the observations in batch j, and do the following for each batch. Compute and store the $k \times k$ matrix $\mathbf{C}_j\mathbf{Y}^b_j$ and the k-dimensional vector $\mathbf{C}_j(\mathbf{y}^o_j - \overline{\mathbf{y}}^b_j)$, where $\mathbf{C}_j = (\mathbf{Y}^b_j)^T\mathbf{R}_j^{-1}$ as in Step 4. (This can be done separately for each batch, in parallel, and the total number of operations is roughly $2k^2\ell_{[g]}$.) Then do Steps 5–8 separately for each local analysis; when $\mathbf{C}\mathbf{Y}^b$ and $\mathbf{C}(\mathbf{y}^o - \overline{\mathbf{y}}^b)$ are required in Steps 5 and 7, compute them by summing the corresponding arrays $\mathbf{C}_j\mathbf{Y}^b_j$ and $\mathbf{C}_j(\mathbf{y}^o_j - \overline{\mathbf{y}}^b_j)$ over the batches j of observations that are used in the local analysis. To avoid redundant addition in these steps, batches that are used in exactly the same subset of the local analyses should be combined into a single batch. The total number of operations required by the summations over batches roughly k^2ps, where s is the average number of batches used in each local analysis. Both this and the $2k^2\ell_{[g]}$ operations described before are smaller than the roughly $2k^2p\bar{\ell} = 2k^2q\ell_{[g]}$ operations they combine to replace.

This approach has similarities with the "sequential" approach of [21] and [45], in which observations are divided into uncorrelated batches and a separate analysis is done for each batch; the analysis is done in the observation space whose dimension is the number of observations in a batch. However, in the sequential approach, the analysis ensemble for one batch of observations is used as the background ensemble for the next batch of observations. Since batches with disjoint local regions of influence can be analyzed separately, some parallelization is possible, though the LETKF approach described above is more easily distributed over a large number of processors. For a serial implementation, either approach may be faster depending on the application and the ensemble size.

4 Asynchronous Observations: 4D-LETKF

In theory, one can perform a new analysis each time new observations are made. In practice, this is a good approach if observations are made at regular and sufficiently infrequent time intervals. However, in many applications, such as weather forecasting, observations are much too frequent for this approach. Imagine, for example, a 6-hour interval between analyses, like at the National Weather Service. Since weather can change significantly over such a time interval, it is important to consider observations taken at intermediate times in a more sophisticated manner than to pretend that they occur at the analysis time (or to simply ignore them). Operational versions of 3D-Var and 4D-Var (see Section 2.2) do take into account the timing of the observations, and one of the primary

strengths of 4D-Var is that it does so in a precise manner, by considering which forecast model trajectory best fits the observations over a given time interval (together with assumed background statistics at the start of this interval).

We have seen that the analysis step in an ensemble Kalman filter considers model states that are linear combinations of the background ensemble states at the analysis time, and compares these model states to observations taken at the analysis time. Similarly, we can consider approximate model trajectories that are linear combinations of the background ensemble trajectories over an interval of time, and compare these approximate trajectories with the observations taken over that time interval. Instead of asking which model trajectory best fits the observations, we ask which linear combination of the background ensemble trajectories best fits the observations. As before, this is relatively a low-dimensional optimization problem that is much more computationally tractable than the full nonlinear problem.

This approach is similar to that of an ensemble Kalman smoother [10, 8], but over a much shorter time interval. As compared to a "filter", which estimates the state of a system at time t using observations made up to time t, a "smoother" estimates the system state at time t using observations made before and after time t. Over a long time interval, one must generally take a more sophisticated approach to smoothing than to simply consider linear combinations of an ensemble of trajectories generated over the entire interval, both because the trajectories may diverge enough that linear combinations of them will not approximate model trajectories, and because in the presence of model error there may be no model trajectory that fits the observations over the entire interval. Over a sufficiently short time interval however, the approximation of true system trajectories by linear combinations of model trajectories with similar initial conditions is quite reasonable.

While this approach to assimilating asynchronous observations is suitable for any ensemble Kalman filter [23], it is particularly simple to implement in the LETKF framework. We call this extension 4D-LETKF; see [19] for an alternate derivation of this algorithm.

To be more concrete, suppose that we have observations $(\tau_j, \mathbf{y}^o_{\tau_j})$ taken at various times τ_j since the previous analysis. Let H_{τ_j} be the observation operator for time τ_j and let $\mathbf{R}_{\tau_j}$ be the error covariance matrix for these observations. In Section 2.3, we mapped a vector $\mathbf{w}$ in the k-dimensional space $\tilde{S}$ into observation space using the formula $\overline{\mathbf{y}}^b + \mathbf{Y}^b\mathbf{w}$, where the background observation mean $\overline{\mathbf{y}}^b$ and perturbation matrix $\mathbf{Y}^b$ were formed by applying the observation operator H to the background ensemble at the analysis time. So now, for each time τ_j, we apply H_{τ_j} to the background ensemble at time τ_j, calling the mean of the resulting vectors $\overline{\mathbf{y}}^b_{\tau_j}$ and forming their differences from the mean into the matrix $\mathbf{Y}^b_{\tau_j}$.

We now form a combined observation vector $\mathbf{y}^o$ by concatenating (vertically) the (column) vectors $\mathbf{y}^o_{\tau_j}$, and similarly by vertical concatenation of the vectors $\overline{\mathbf{y}}^b_{\tau_j}$ and matrices $\mathbf{Y}^b_{\tau_j}$ respectively, we form the combined background observation mean $\overline{\mathbf{y}}^b$ and perturbation matrix $\mathbf{Y}^b$. We form the cor-

responding error covariance matrix $\mathbf{R}$ as a block diagonal matrix with blocks $\mathbf{R}_{\tau_j}$ (this assumes that observations taken at different times have uncorrelated errors, though such correlations if present could be included in $\mathbf{R}$).

Given this notation, we can then use the same analysis equations as in the previous sections, which are based on minimizing the cost function $\tilde{J}^*$ given by (19). (We could instead write down the appropriate analogue to (15) and minimize the resulting nonlinear cost function $\tilde{J}$; this would be no harder in principle than in the case of synchronous observations.) Referring to Section 3, the only change is in Step 1, which one should perform for each observation time τ_j (using the background ensemble and observation operator for that time) and then concatenate the results as described above. Step 2 still only needs to be done at the analysis time, since its output is used only in Step 8 to form the analysis ensemble in model space. All of the intermediate steps work exactly the same, in terms of the output of Step 1.

In practice, the model will be integrated with a discrete time step that in general will not coincide with the observation times τ_j. One should either interpolate the background ensemble trajectories to the observation times, or simply round the observation times off to the nearest model integration time. In either case, one must either store the background ensemble trajectories until the analysis time, or perform Step 1 of Section 3 during the model integration and store its results. The latter approach will require less storage if the number of observations per model integration time step is less than the number of model variables.

One can perform localization in the same manner as with synchronous observations, but it may be advantageous to take into account the timing of the observations when deciding which of them to use in a given local analysis. For example, due to spatial propagation in the model dynamics, one may wish to include earlier observations from a greater distance than later observations. On the other hand, earlier observations may be less useful than observations closer to the analysis time due to model error; it may help then to decrease the influence of the earlier observations as described in Step 4 of Section 3.

5 Numerical Experiments with Real Atmospheric Observations

We have implemented the 4D-LETKF algorithm, as described in Sections 3 and 4, with the operational Global Forecast System (GFS) model [14] of the U.S. National Centers for Environmental Prediction (NCEP). This model is used (currently at higher resolution than we describe below) for National Weather Service forecasts. Previously we have published results using this model with the LEKF algorithm of [36, 37], in a perfect model scenario (with simulated observations) [41]. Using the same parameters for the LETKF algorithm, we have obtained results very similar to those in [41], which we do not repeat here; with LETKF, the data assimilation steps run 3 to 5 times as fast

as with LEKF.

Here, we present some preliminary results obtained using the 4D-LETKF algorithm with the same model and real atmospheric observations collected in January and February 2004, and compare them with results from the NCEP Spectral Statistical Interpolation (SSI) [15], a state-of-the-art implementation of 3D-Var. Further results will appear in a future publication. Our data set includes all operationally assimilated observations except for satellite radiances and measurements of atmospheric humidity. Observations include vertical sounding profiles of temperature and wind by weather balloons, surface pressure observations by land and sea stations, temperature and wind reports by commercial aircraft, and wind vectors derived from satellite based observation of clouds.

For all of the results below, we assimilate observations every 6 hours, and we use a model resolution of T62 (a 192×94 longitude-latitude grid) with 28 vertical levels, for a total of about $500,000$ points. In our 4D-LETKF implementation, for each grid point, we selected observations from within a $h \times h \times v$ subset of the model grid, centered at the analysis grid point, with the vertical height v varying (depending on the vertical level) from 1 to 7 grid points as in [41], and the horizontal width h held constant for each experiment at either 5 or 7 grid points. The number of ensemble members k we use in each experiment is either 40 or 80. In all 4D-LETKF experiments we used a spatially-dependent multiplicative covariance inflation factor ρ, which we taper from 1.15 at the surface to 1.1 at the top of the model atmosphere in the Southern Hemisphere, and from 1.25 at the surface to 1.15 at the top in the Northern Hemisphere (between 30°S and 30°N latitudes, we linearly interpolate between these values).

5.1 Analysis Quality

In this section, we compare the analyses from our 4D-LETKF implementation, using $k = 40$ ensemble members and a 7×7 horizontal grid ($h = 7$) for each local region, and from the NCEP SSI, using the same model resolution and the same observational data set for its global analysis (we call this the "benchmark" SSI analysis). To estimate the analysis error for a given state variable, we compute the spatial RMS difference between its analysis and the operational high-resolution SSI analysis computed by NCEP (we call this the "verification" SSI analysis). While this verification technique favors the benchmark SSI analysis, which is obtained with the same data assimilation method, it can provide useful information in regions where the 4D-LETKF and benchmark SSI analyses exclude a large portion of the observations assimilated into the verifying SSI analysis. Such a region is the Southern Hemisphere, where satellite radiances are known to have a strong positive impact on the quality of the analysis.

We initialize 4D-LETKF with a random ensemble of physically plausible global states at midnight on 1 January 2004. Specifically, we take each initial ensemble member from an operational NCEP analysis at a randomly chosen time between 15 January and 31 March 2004. The 4D-LETKF

analyses start to synchronize with the observations after a few days. To exclude from our comparison the transient error due to initialization of 4D-LETKF, we average all estimated errors over the month of February 2004 only.

Figure 1 shows the estimated analysis error of each method for temperature in the Southern Hemisphere extratropics (20°S to 90°S latitudes) as a function of atmospheric pressure. The 4D-LETKF analysis is more accurate than the benchmark SSI at all except near the surface, where the two methods are quite similar in accuracy. The advantage of the 4D-LETKF analysis is especially large in the upper atmosphere, where observations are extremely sparse. Figure 2 makes the same comparison between the 48-hour forecasts generated from the respective analyses, again verified against the operational high-resolution SSI analysis at the appropriate time. We see that the 4D-LETKF forecasts are also more accurate than those from the background SSI analysis, especially in the upper atmosphere.

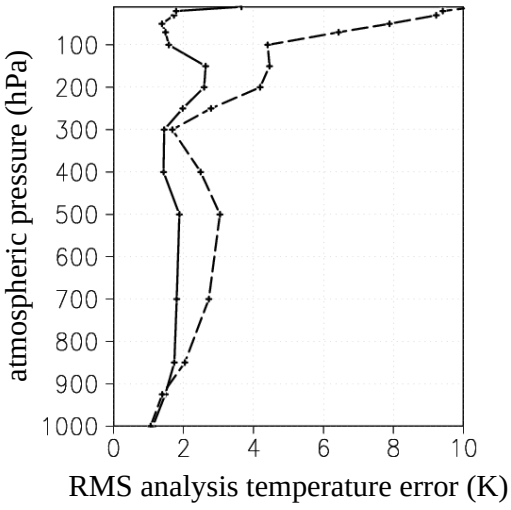

Figure 1: Vertical profile of the estimated analysis temperature error, in degrees Kelvin, for the 4D-LETKF (solid) and benchmark SSI (dashed) analyses in the Southern Hemisphere extratropics. The atmospheric height is indicated by pressure, in hectopascal. The estimate of the error is obtained by calculating the root-mean-square difference between each analysis and the verifying SSI analysis for latitudes between 20°S and 90°S and averaging over all the analysis times in February 2004.

For wind in the Southern Hemisphere and temperature and wind in the Northern Hemisphere, the RMS analysis and forecast errors for the 4D-LETKF and benchmark SSI are more similar to each other, though in all cases the 4D-LETKF results are significantly better in the highest part of the atmosphere (0 hPa to 100 hPa).

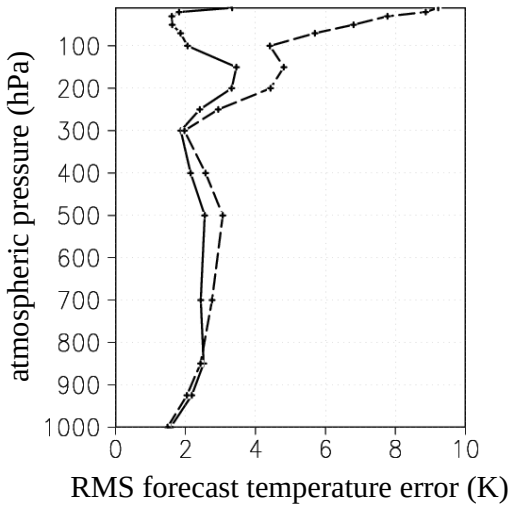

Figure 2: Estimated 48-hour forecast temperature error versus atmospheric pressure for the 4D-LETKF (solid) and benchmark SSI (dashed) methods in the Southern Hemisphere extratropics. The estimated forecast error is computed in the same way as the estimated analysis error shown in Figure 1.

We emphasize that these results are obtained with modest tuning of the 4D-LETKF parameters, and we expect further significant improvements from a more thorough exploration of the algorithm's parameter space, as well as a more sophisticated approach to model error, such as the adaptive bias-correction technique of [3].

Qualitatively similar results with the same model and a similar data set are reported in [46], using both an alternate implementation of the 4D-LETKF algorithm (with a different covariance inflation approach) and a related method based on the Ensemble Square-Root Filter of [45]. The latter method was slightly more accurate in the Northern Hemisphere and slightly less accurate in the Southern Hemisphere, and both methods were more accurate than the corresponding benchmark SSI, when verified both against the operational high-resolution SSI analysis and against observations. See also [22] and [34] for comparisons of ensemble Kalman filter results to those of other operational 3D-Var methods, using forecast models from the Meteorological Service of Canada and the Japan Meteorological Agency, respectively; the latter article also implements the LETKF approach.

5.2 Computational Speed

In this section, we present and discuss timing results from several representative analyses of the 4D-LETKF experiment using the GFS described above, with different numbers of observations. In

addition, we vary the number of ensemble members ($k = 40$ or 80) and the horizontal width of the local region ($h = 5$ or 7 grid points in both latitude and longitude). Though we use a parallel implementation, we report in Table 1 below the total CPU time used on a Linux cluster of forty 3.2 GHz Intel Xeon processors. The actual run time is many times faster; with the larger local region ($h = 7$) the analysis takes about 6 minutes on our cluster with $k = 40$ ensemble members, and 18 minutes with $k = 80$. Thus, the results shown in Figures 1 and 2 can be obtained in an operational setting that allots only a few minutes for each analysis. Furthermore, because the observations are very nonuniformly distributed spatially, we expect to be able to reduce the parallel run time considerably by balancing the load more evenly between processors. We will report details of our parallel implementation in a future publication.

Table 1 shows the total CPU time in seconds for 4 different 4D-LETKF parameter sets at each of 4 different analysis times. Different numbers of observations are available for each analysis time, with about 50% more at 1200 GMT than at 0600 GMT. The computation time generally grows with the number of observations, though not by as large a factor. Referring to the discussion immediately following Steps 1 to 9 in Section 3, this indicates that the matrix multiplication portion of Step 5 that requires on the order of $k^2 q \ell_{[g]}$ total floating point operations is a significant component of the computation time, but that other parts of the computation are significant too. (Recall that $\ell_{[g]}$ is the global number of observations and q is the average number of analyses in which each observation is used, which in this implementation is roughly the average number of grid points per local region.) As h increases from 5 to 7, the value of q approximately doubles, and so does the computation time. And as k increases from 40 to 80, the computation time grows by a factor of 4 to 5, indicating that the time is roughly quadratic in k but suggesting that terms that are cubic in k are becoming significant.

analysis time	0600 GMT	1800 GMT	0000 GMT	1200 GMT
# observations	159,947	193,877	236,168	245,850
$k = 40$, $h = 5$	945	945	1244	1142
$k = 40$, $h = 7$	1846	2076	2105	2200
$k = 80$, $h = 5$	4465	4453	5124	5010
$k = 80$, $h = 7$	9250	10631	10463	10943

Table 1: Total CPU time in seconds (on 3.2GHz Intel Xeon processors) for various analyses with 4D-LETKF using the GFS with approximately $500,000$ model grid points. Columns represent different analysis times, arranged in increasing order of the number of observations assimilated. Rows represent different values of the ensemble size k and horizontal localization width h. Notice that even on a single processor, all of these analyses can be done in less than real time.

Indeed, examining the CPU time spent in various subroutines on different processors confirms that most of the time is spent in Steps 5 and 6, and that in local analyses where observations are dense, the matrix multiplication in Step 5 dominates the computation time, while in local analyses where observations are sparse, the matrix inverse and square root in Steps 5 and 6 dominate. We find that the latter operations take more time in the analyses with the larger local region; this suggests that the iterate eigenvalue routine we use takes longer to run in cases when the presence of more observations causes $\tilde{\mathbf{P}}^a$ to be further from a multiple of the identity matrix. There is also some computational overhead not accounted for in Section 3 whose contribution to the computation time is not negligible, in particular determining which observations to use for each local analysis.

Overall, our timing results indicate that with a model and data set of this size, a substantially larger ensemble size than we currently use may be problematic, but that our implementation of 4D-LETKF should be able to assimilate more observations with at most linear growth in the computation time. Furthermore, though we do not vary the number of model variables in Table 1, our examination of the time spent performing each of the steps from Section 3 suggests that we can increase the model resolution significantly without having much effect on the analysis computation time (though the time spent running the model would of course increase accordingly).

6 Summary and Acknowledgments

In this article, we have described a general framework for data assimilation in spatiotemporally chaotic systems using an ensemble Kalman filter that in its basic version (Section 3) is relatively efficient and simple to implement. In a particular application, one may be able to improve accuracy by experimenting with different approaches to localization (see the discussion in Sections 2.2 and 2.3), covariance inflation (see the end of Section 2.3), and/or asynchronous observations (Section 4). For very large systems and/or when maximum efficiency is important, one should consider carefully the comments about implementation in Section 3 (and at the end of Section 4, if applicable). One can also apply this method to low-dimensional chaotic systems, without using localization.

In Section 5, we presented preliminary results for a relatively straightforward implementation of the LETKF approach with real atmospheric data and an operational global forecast model. Our results demonstrate that this implementation can produce results of operational quality within a few minutes on a parallel computer of reasonable size. The efficiency of the basic algorithm provides many opportunities to improve the quality of the results with the variations discussed and referred to in this article.

Many colleagues have contributed to this article in various ways. We thank M. Cornick, E. Fertig, J. Harlim, H. Li, J. Liu, T. Miyoshi, and J. Whitaker for sharing their thoughts and experiences implementing and testing the LETKF algorithm in a variety of scenarios. We also thank C. Bishop,

T. Hamill, K. Ide, E. Kalnay, E. Ott, D. Patil, T. Sauer, J. Yorke, M. Zupanski, and the anonymous reviewers for their generous input. This feedback resulted in many improvements to the exposition in this article and to our implementation of the algorithm. We thank Y. Song, Z. Toth, and R. Wobus for providing us with the observations and benchmark analyses used in Section 5, and we thank G. Gyarmati for developing software to read the observations on our computers. This research was supported by grants from NOAA/THORPEX, the J. S. McDonnell Foundation, and the National Science Foundation (grant #ATM034225). The second author gratefully acknowledges support from the NSF Interdisciplinary Grants in the Mathematical Sciences program (grant #DMS0408012).

References

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

[2] J. L. Anderson & S. L. Anderson, A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts, Mon. Wea. Rev. **127**, 2741–2758 (1999).

[3] S.-J. Baek, B. R. Hunt, E. Kalnay, E. Ott, I. Szunyogh, Local ensemble Kalman filtering in the presence of model bias, Tellus A **58**, 293–306 (2006).

[4] C. H. Bishop, B. J. Etherton, S. J. Majumdar, Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects, Mon. Wea. Rev. **129**, 420–436 (2001).

[5] G. Burgers, P. J. van Leeuwen, G. Evensen, Analysis scheme in the ensemble Kalman filter, Mon. Wea. Rev. **126**, 1719–1724 (1998).

[6] A. Doucet, N. de Freitas, N. Gordon, eds., *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001.

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

[8] G. Evensen, The ensemble Kalman filter: theoretical formulation and practical implementation, Ocean Dynam. **53**, 343–367 (2003).

[9] G. Evensen, *Data Assimilation: the Ensemble Kalman Filter*, Springer (2006).

[10] G. Evensen & P. J. van Leeuwen, An ensemble Kalman smoother for nonlinear dynamics, Mon. Wea. Rev. **128**, 1852–1867 (2000).

[11] E. J. Fertig, J. Harlim, B. R. Hunt, A comparative study of 4D-VAR and a 4D ensemble Kalman filter: perfect model simulations with Lorenz-96, Tellus A **59** (2007), in press.

[12] I. Fukumori, A partitioned Kalman filter and smoother, Mon. Wea. Rev. **130**, 1370–1383 (2002).

[13] M. Ghil & P. Malanotte-Rizzoli, Data assimilation in meteorology and oceanography, Adv. Geophys. **33**, 141–266 (1991).

[14] Global Climate and Weather Modeling Branch (Environmental Modeling Center, Camp Springs MD), The GFS Atmospheric Model, NCEP Office Note **442** (2003), `http://www.emc.ncep.noaa.gov/officenotes/FullTOC.html`

[15] Global Climate and Weather Modeling Branch (Environmental Modeling Center, Camp Springs MD), SSI Analysis System 2004, NCEP Office Note **443** (2004), `http://www.emc.ncep.noaa.gov/officenotes/FullTOC.html`

[16] T. M. Hamill & C. Snyder, A hybrid ensemble Kalman filter–3D variational analysis scheme Mon. Wea. Rev. **128** 2905–2919 (2000).

[17] T. M. Hamill, J. S. Whitaker, and C. Snyder, Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter, Mon. Wea. Rev. **129**, 2776–2790 (2001).

[18] J. Harlim & B. R. Hunt, A non-Gaussian ensemble filter for assimilating infrequent noisy observations, Tellus A, to appear.

[19] J. Harlim & B. R. Hunt, Four-dimensional local ensemble transform Kalman filter: numerical experiments with a global circulation model, preprint.

[20] P. L. Houtekamer & H. L. Mitchell, Data assimilation using an ensemble Kalman filter technique, Mon. Wea. Rev. **126**, 796–811 (1998).

[21] P. L. Houtekamer & H. L. Mitchell, A sequential ensemble Kalman filter for atmospheric data assimilation, Mon. Wea. Rev. **129**, 123–137 (2001).

[22] P. L. Houtekamer, H. L. Mitchell, G. Pellerin, M. Buehner, M. Charron, L. Spacek, B. Hansen, Atmospheric data assimilation with an ensemble Kalman filter: results with real observations, Mon. Wea. Rev. **133**, 604–620 (2005).

[23] B. R. Hunt, E. Kalnay, E. J. Kostelich, E. Ott, D. J. Patil, T. Sauer, I. Szunyogh, J. A. Yorke, A. V. Zimin, Four-dimensional ensemble Kalman filtering, Tellus A **56**, 273–277 (2004).

[24] K. Ide, P. Courtier, M. Ghil, A. C. Lorenc, Unified notation for data assimilation: operational, sequential, and variational, J. Meteo. Soc. Japan **75**, 181–189 (1997).

[25] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press (1970).

[26] R. E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME, Ser. D: J. Basic Eng. **82**, 35–45 (1960).

[27] R. E. Kalman & R. S. Bucy, New results in linear filtering and prediction theory, Trans. ASME, Ser. D: J. Basic Eng. **83**, 95–108 (1961).

[28] E. Kalnay, *Atmospheric Modeling, Data Assimilation and Predictability*, Cambridge Univ. Press (2002).

[29] E. Kalnay & Z. Toth, Removing growing errors in the analysis, Proc. of the 10th Conf. on Numerical Weather Prediction, Amer. Meteo. Soc., Portland, Oregon (1994).

[30] C. L. Keppenne, Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter, Mon. Wea. Rev. **128**, 1971–1981 (2000).

[31] F.-X. Le Dimet & O. Talagrand, Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects, Tellus A **38**, 97–110 (1986).

[32] A. C. Lorenc, A global three-dimensional multivariate statistical interpolation scheme, Mon. Wea. Rev. **109**, 701–721 (1981).

[33] A. C. Lorenc, The potential of the ensemble Kalman filter for NWP—a comparison with 4D-Var, Quart. J. Roy. Meteo. Soc. **129**, 3183–3203 (2003).

[34] T. Miyoshi & S. Yamane, Local ensemble transform Kalman filtering with an AGCM and a T159/L48 resolution, preprint.

[35] M. Oczkowski, I. Szunyogh, D. J. Patil, Mechanisms for the development of locally low-dimensional atmospheric dynamics, J. Atmos. Sci. **62**, 1135–1156 (2005).

[36] E. Ott, B. R. Hunt, I. Szunyogh, M. Corazza, E. Kalnay, D. J. Patil, J. A. Yorke, A. V. Zimin, E. J. Kostelich, Exploiting local low dimensionality of the atmospheric dynamics for efficient ensemble Kalman filtering, `arXiv:physics/0203058v3` (2002).

[37] E. Ott, B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corazza, E. Kalnay, D. J. Patil, J. A. Yorke, A local ensemble Kalman filter for atmospheric data assimilation, Tellus A **56**, 415–428 (2004).

[38] D. F. Parrish & J. C. Derber, The National Meteorological Center's spectral statistical-interpolation analysis system, Mon. Wea. Rev. **120**, 1747–1763 (1992).

[39] D. J. Patil, B. R. Hunt, E. Kalnay, J. A. Yorke, E. Ott, Local low dimensionality of atmospheric dynamics, Phys. Rev. Lett. **86**, 5878–5881 (2001).

[40] L. M. Pecora & T. L. Carroll, Synchronization in chaotic systems, Phys. Rev. Lett. 64, 821–824 (1990).

[41] I. Szunyogh, E. J. Kostelich, G. Gyarmati, D. J. Patil, B. R. Hunt, E. Kalnay, E. Ott, J. A. Yorke, Assessing a local ensemble Kalman filter: perfect model experiments with the National Centers for Environmental Prediction global model, Tellus A **57**, 528–545 (2005).

[42] O. Talagrand & P. Courtier, Variational assimilation of meteorological observations with the adjoint vorticity equation I: theory, Quart. J. Roy. Meteo. Soc. **113**, 1311–1328 (1987).

[43] M. K. Tippett, J. L. Anderson, C. H. Bishop, T. M. Hamill, J. S. Whitaker, Ensemble square-root filters, Mon. Wea. Rev. **131**, 1485–1490 (2003).

[44] X. Wang, C. H. Bishop, S. J. Julier, Which is better, an ensemble of positive-negative pairs or a centered spherical simplex ensemble?, Mon. Wea. Rev. **132**, 1590–1605 (2004).

[45] J. S. Whitaker, T. M. Hamill, Ensemble data assimilation without perturbed observations, Mon. Wea. Rev. **130**, 1913–1924 (2002).

[46] J. S. Whitaker, T. M. Hamill, X. Wei, Y. Song, Z. Toth, Ensemble data assimilation with the NCEP global forecast system, preprint.

[47] F. Zhang, C. Snyder, J. Sun, Impacts of initial estimate and observation availability on convective-scale data assimilation with an ensemble Kalman filter, Mon. Wea. Rev. **132**, 1238–1253 (2004).

[48] M. Zupanski, Maximum likelihood ensemble filter: theoretical aspects, Mon. Wea. Rev. **133**, 1710–1726 (2005).