2018-10

Sparsity-Based Kalman Filters for Data Assimilation

Kang, Wei; Xu, Liang

ArXiv

Kang, Wei, and Liang Xu. "Sparsity-Based Kalman Filters for Data Assimilation." arXiv preprint arXiv:1810.04236 (2018).
http://hdl.handle.net/10945/65305

Sparsity-Based Kalman Filters for Data Assimilation[*]

Wei Kang[†] and Liang Xu[‡]

October 11, 2018

Abstract

Several variations of the Kalman filter algorithm, such as the extended Kalman filter (EKF) and the unscented Kalman filter (UKF), are widely used in science and engineering applications. In this paper, we introduce two algorithms of sparsity-based Kalman filters, namely the sparse UKF and the progressive EKF. The filters are designed specifically for problems with very high dimensions. Different from various types of ensemble Kalman filters (EnKFs) in which the error covariance is approximated using a set of dense ensemble vectors, the algorithms developed in this paper are based on sparse matrix approximations of error covariance. The new algorithms enjoy several advantages. The error covariance has full rank without being limited by a set of ensembles. In addition to the estimated states, the algorithms provide updated error covariance for the next assimilation cycle. The sparsity of error covariance significantly reduces the required memory size for the numerical computation. In addition, the granularity of the sparse error covariance can be adjusted to optimize the parallelization of the algorithms.

1 INTRODUCTION

For dynamical systems with numerical models, data assimilation is an estimation process of combining observational data with a numerical model to obtain an estimate of the system's state. Data assimilation is essential to numerical weather prediction (NWP). The state estimate can be considered as an interpolation of the sparse observational data; and it is used as the initial condition for the numerical forecast process. If the dimension is relatively low and the data set is small, various linear and nonlinear estimators can be found in the literature that have optimal or suboptimal performances. However, to assimilate big data sets with models that have high dimensions, such as those in operational NWP systems with tens of millions of variables, achieving reliable state estimates and error probability distributions is a challenging problem that have been studied for decades with a huge literature.

[*]This work was supported in part by U.S. Naval Research Laboratory - Monterey, CA

[†]Wei Kang is with Faculty of Applied Mathematics, Naval Postgraduate School, Monterey, CA, USA, `wkang@nps.edu`

[‡]Liang Xu is with Naval Research Laboratory, Monterey, CA, USA, `liang.xu@nrlmry.navy.mil`

There are two categories of methods widely used in NWP, namely variational methods and the ensemble Kalman filter (EnKF) [8, 3]. The former is based on a weighted least-square optimization, such as the four dimensional variational data assimilation (4D-Var) in a fixed time window or the three dimensional version (3D-Var) that excludes the time variable. The EnKF algorithm is based on the Kalman filter except that the error covariance is approximated using a set of state ensembles. 4D-Var methods are used in operational NWP systems by many meteorological centers. While it serves as an effective method of data assimilation, 4D-Var algorithms have difficulty to explicitly track the evolution of error covariance within its estimation process due to outrageous computational costs and input/output (I/O) loads required by matrices of extremely high dimensions. EnKF, on the other hand, updates information about the error covariance in the form of ensembles. However, the number of ensemble states is significantly smaller than the number of state variables. As a result, the rank deficiency of error covariance tends to deteriorate the integrity of the estimation process unless remedies to the algorithm, such as localization and covariance inflation, are applied.

Different types of Kalman filters have been developed and widely used in science and engineering applications, such as the EnKF, the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). In this paper, we introduce two algorithms of sparsity-based Kalman fillters, namely the sparse UKF and the progressive EKF. The goal of the work is to explore innovative ideas that take the advantage of the (almost) sparsity structure of matrices so that analysis and error covariance can be updated effectively and efficiently without the drawback of rank deficiency. The granularity of subproblems for the purpose of algorithm parallelization is also emphasized in the method. The filters are developed specifically for problems with very high dimensions. Different from EnKFs in which the error covariance is represented using a set of dense ensemble vectors, the new algorithms in this paper are based on a sparse but full rank matrix as an approximation of the error covariance. This is made possible because of two assumptions: (a) the error covariance is approximately a sparse matrix; (b) the system model is component based, i.e. the state vectors are divided into components that can be computed independently in parallel.

2 Sparse UKF

Consider a dynamical system model in which the state variable is $x(t)$, where $t = 1, 2, 3, \cdots$ represents time steps. The value of observation at $t = k$ is denoted by $y(k)$. The system model is defined as follows,

$$\begin{aligned} x(k) &= \mathcal{M}(x(k-1)) + \eta_{k-1}, & x(k), \eta_{k-1} &\in \mathbb{R}^n, \\ y(k) &= \mathcal{H}(x(k)) + \delta_k, & y_k, \delta_k &\in \mathbb{R}^m, \end{aligned} \tag{1}$$

where η_{k-1} is a random variable representing the model error. Its covariance is Q. The observational error, δ_k, has a covariance R. In data assimilation, the goal is to estimate the value of $x(k)$ given the observations $y(1), y(2), \cdots, y(k)$ and the model (1). If (1) is linear and if all random variables are Gaussian, then the Kalman filter is an optimal state estimator. For nonlinear systems with non-Gaussian random errors, various types of Kalman filters exist in the literature with successful applications in science and engineering. If a system has a very high dimension, the conventional form of Kalman filter based on a dense error covariance is not applicable. In this section, we introduce an algorithm that is a variation of UKF for problems with approximately sparse error covariances.

In a sparse matrix/vector, most entries are zeros. In this paper, we use an underbar to emphasize that a vector or matrix is sparse, for instance $\underline{P}$ and $\underline{x}$. A sparse vector is associated with an index set, denoted by $\mathcal{I}$, consisting of the indices of nonzero entries. For a sparse matrix, columns may have different numbers of nonzero entries. The largest such number is denoted by N_{sp}. In sparsity-based algorithms, a full model evaluation is not always necessary. Using a *component-based* model can significantly reduce the computational load. In the notation, a component-based model has three inputs: the sparse state variable, its index, and the index of the output state. More specifically,

$$\underline{x}(k) = \mathcal{M}(\underline{x}(k-1); \mathcal{I}_1; \mathcal{I}_2), \tag{2}$$

where $\mathcal{I}_1$ is the index set of the sparse vector $\underline{x}(k-1)$ and $\mathcal{I}_2$ is the index set of $\underline{x}(k)$. The model evaluates only the entries with indices in $\mathcal{I}_2$, setting all other entries as zeros. Note that $\underline{x}(k)$ is different from the full state variable $x(k)$ because the later is, in general, a dense vector with mostly nonzero entries. Therefore, it is important to specify the index set $\mathcal{I}_2$ of the sparse vector $\underline{x}(k)$ to be evaluated using a component-based model. For simplicity, we often omit $\mathcal{I}_1$ in the notation, i.e.

$$\underline{x}(k) = \mathcal{M}(\underline{x}(k-1); \mathcal{I}), \tag{3}$$

where $\mathcal{I}$ is the same as $\mathcal{I}_2$ in (2). Algebraic operations between sparse vectors, such as addition and dot product, are defined in the same way as dense vectors. Thus, we may conduct operations between sparse vectors and dense vectors, such as adding a sparse vector to a dense vector $\underline{x} + y$ as long as both vectors have the same dimension. A new operation, called merging, between a sparse vector and a dense vector is defined as follows,

$$z = \underline{x} \triangleright y, \quad \begin{cases} i\text{th component of } z = i\text{th component of } x, & \text{if } i \in \mathcal{I}. \\ i\text{th component of } z = i\text{th component of } y, & \text{if } i \notin \mathcal{I}. \end{cases} \tag{4}$$

If an operation has an underbar, it means that the evaluation is carried out only at a given index set. For instance, given two sparse matrices $\underline{\mathrm{A}}$ and $\underline{\mathrm{B}}$, then $\underline{\mathrm{A}} * \underline{\mathrm{B}}$ is a different matrix from $\underline{\underline{A} * \underline{B}}$. The former is the conventional matrix multiplication between two sparse matrices; the later is a matrix multiplication in which the entries in a given index set are evaluated and all other entries are set to be zeros. Other operations, such as $\underline{\sqrt{\underline{P}}}$, are defined similarly. A summary of notations is summarized in the following table.

2.1 The UKF

The unscented Kalman filter has been increasingly popular in engineering applications since its introduction about twenty years ago [4, 5]. In a UKF algorithm, the error covariance is propagated with the dynamics using a set of vectors, or σ-points denoted by x^σ. Their definition is given in (5)-(6). The σ-points are computed at each time step using a square root of the error covariance. In most UKF applications, σ-points are computed using either Cholesky factorization or matrix diagonalizations. In the notation, a variable with a superscript 'a', such as x^a, represents the *analysis* value of the variable, i.e., the updated value based on observations. A variable with a superscript 'b', such as y^b, represents the background, i.e., the propagated value of analysis using

Notation	Definition	Notation	Definition
x	state variable	y	observation variable
$\mathcal{M}$	model function	$\mathcal{H}$	observation operator
n	state space dimension	$t = 1, 2...$	(discrete) time variable
x_i^σ	σ-point at $t = k - 1$		
x_i^b	background - state vector	y_i^b	output of observation operator $\mathcal{H}(x_i^b)$
$\bar{x}^b$	average of x_i^b	$\bar{y}^b$	average of y_i^b
P^b	background - error covariance		
x^a	analysis - state vector	P^a	analysis - error covariance

Table 1: Notations

the system model. The algorithm is summarized as follows. At $t = k - 1$, suppose we have the analysis and error covariance as well as its square root

$$\begin{array}{l} x^a(k-1),\ P^a(k-1), \\ X^a(k-1) = \sqrt{(n+\kappa)P^a(k-1)}, \end{array} \tag{5}$$

where κ is a scaling factor for the fine tuning of the higher order moments of the approximation error [4]. How to tune the value of κ for a sparsity-based UKF is an open problem that needs further study. In this paper, $\kappa = 0$ is used in all examples. A set of σ-points is generated as follows,

$$\begin{array}{ll} x_0^\sigma(k-1) = x^a(k-1), & \\ x_i^\sigma(k-1) = x^a(k-1) + X_i^a(k-1), & 1 \le i \le n, \\ x_i^\sigma(k-1) = x^a(k-1) - X_i^a(k-1), & n+1 \le i \le 2n. \end{array} \tag{6}$$

The next step is to propagate the σ-points, which represent the background at $t = k$. For simplicity, the index 'k' of all variables in the kth time-step is omitted.

$$\begin{array}{lll} x_i^b = \mathcal{M}(x_i^\sigma(k-1)), & y_i^b = \mathcal{H}(x_i^b), & 0 \le i \le 2n, \\ \bar{x}^b = \displaystyle\sum_{i=0}^{2n} w_i x_i^b, & \bar{y}^b = \displaystyle\sum_{i=0}^{2n} w_i y_i^b, & \end{array} \tag{7}$$

where the weights are defined as follows

$$w_0 = \frac{\kappa}{n+\kappa},\ w_i = \frac{1}{2(n+\kappa)}, \tag{8}$$

for $i = 1, 2, \cdots, 2n$. Define the variations

$$X_i^b = x_i^b - \bar{x}^b,\ Y_i^b = y_i^b - \bar{y}^b. \tag{9}$$

The background covariances are

$$\begin{aligned} P^b &= \sum_{i=0}^{2n} w_i X_i^b (X_i^b)^T + Q, \\ P_{xy} &= \sum_{i=0}^{2n} w_i X_i^b (Y_i^b)^T, \\ P_{yy} &= \sum_{i=0}^{2n} w_i Y_i^b (Y_i^b)^T + R. \end{aligned} \tag{10}$$

The Kalman gain, K, satisfies the following equation,

$$KP_{yy} = P_{xy}. \tag{11}$$

The analysis is updated as follows

$$\begin{aligned} x^a &= \bar{x}^b + K(y_o - \bar{y}^b), \\ P^a &= P^b - K(P_{xy})^T. \end{aligned} \tag{12}$$

where y_o is the observation at $t = k$. This completes one iteration of the filter. For the next step, $t = k + 1$, go back to (5) replacing the analysis by the updated value of x^a and P^a.

2.2 Sparse UKF

The square root factorization of a matrix is not unique. For large and sparse matrices, various algorithms and their implementations on different computing platforms have been studied for many years. The literature can be traced back to the early days of electronic computers [2]. In the case of Cholesky factorization, the square root of a sparse matrix is still sparse, although the computation may require larger memory than the original matrix [1, 7].

A dense error covariance is intractable in computation for global models used in NWP. In the proposed approach, we assume that P and $\sqrt{P}$ are approximately sparse. In the algorithm, they are replaced by their sparse approximations, $\underline{P}$ and $\sqrt{\underline{P}}$. The indices of nonzero entries in the ith column are denoted by $\mathcal{I}_i$ and $\mathcal{I}_i^\sigma$, respectively. The sparsity index set of the forecast of σ-points, i.e. the background, are denoted by $\mathcal{I}^b$. Becasue $\underline{P}$ and $\sqrt{\underline{P}}$ are approximations of P and $\sqrt{P}$, the sparsity patterns do not have to be exact. In fact, in the example of Lorenz-96 model presented in the next section, we assume that $\mathcal{I}_i$, $\mathcal{I}_i^\sigma$ and $\mathcal{I}^b$ equal to each other although $\sqrt{\underline{P}}$ may have a different sparsity pattern from that of $\underline{P}$.

Algorithm I (Sparse UKF)
Given the initial analysis,

$$x^a(k-1), \ \underline{P}^a(k-1). \tag{13}$$

Step 1. σ-points and forecast

$$\underline{X}^a(k-1) = \underline{\sqrt{(n+\kappa)\underline{P}^a(k-1)}}, \text{ sparsity index set } \mathcal{I}^\sigma \tag{14}$$

and

$$\begin{aligned}
x_0^b &= \mathcal{M}(x^a(k-1)), & y_0^b &= \mathcal{H}(x_0^b), & \\
\underline{x}_i^b &= \mathcal{M}(x^a(k-1) + \underline{X}_i^a(k-1); \mathcal{I}_i^b), & y_i^b &= \mathcal{H}(\underline{x}_i^b \triangleright x_0^b), & 1 \le i \le n. \\
\underline{x}_i^b &= \mathcal{M}(x^a(k-1) - \underline{X}_i^a(k-1); \mathcal{I}_i^b), & y_i^b &= \mathcal{H}(\underline{x}_i^b \triangleright x_0^b), & n+1 \le i \le 2n. \\
\bar{x}^b &= \sum_{i=0}^{2n} w_i(\underline{x}_i^b \triangleright x_0^b), & \bar{y}^b &= \sum_{i=0}^{2n} w_i y_i^b &
\end{aligned} \tag{15}$$

Step 2. Background covariances

$$\begin{aligned}
\underline{P}^b &= \sum_{i=0}^{2n} w_i \underline{(\underline{x}_i^b \triangleright x_0^b - \bar{x})(\underline{x}_i^b \triangleright x_0^b - \bar{x})^T} + \underline{Q}, \quad \text{sparsity index set } \mathcal{I}, \\
P_{xy} &= \sum_{i=0}^{2n} w_i(\underline{x}_i^b \triangleright x_0^b - \bar{x})(y_i^b - \bar{y})^T, \\
P_{yy} &= \sum_{i=0}^{2n} w_i(y_i^b - \bar{y})(y_i^b - \bar{y})^T + R.
\end{aligned} \tag{16}$$

Step 3. Kalman gain and analysis

$$\begin{aligned}
KP_{yy} &= P_{xy}, \\
x^a &= \bar{x}^b + K(y_o - \bar{y}^b), \\
\underline{P}^a &= \underline{P}^b - \underline{K(P_{xy})^T} + \gamma I, \quad \text{sparsity index set } \mathcal{I}.
\end{aligned} \tag{17}$$

In (16), we assume $\underline{x}_0^b = x_0^b$. The constant term γI in (17) is a diagonal matrix. The value of γ is selected so that P^a is positive definite. The positive definiteness is guaranteed if γ is larger than $|\lambda_{\min}|$, where $|\lambda_{\min}|$ is the smallest negative eigenvalue of

$$\underline{P}^b - \underline{K(P_{xy})^T}. \tag{18}$$

If the updated covariance matrix is positive definite, then $\gamma = 0$. In all examples, the value of γ is adaptively changed in every cycle depending on the smallest negative eigenvalue of (18)

The sparse UKF algorithm is based on the assumption that P^a can be approximated by a sparse matrix $\underline{P}^a$. Although the σ-points in the algorithm play a similar role as that of ensembles in EnKF, using sparse UKF one can avoid the problem of rank deficiency. For systems with very high dimensions, the number of ensemble members used in an EnKF is much smaller than the dimension. As shown in Figure 1, the narrow and tall matrix of ensemble vectors makes EnKF fundamentally a rank deficient approach. In contrast, the block diagonal matrix $\underline{P}^a$ shown in Figure 1 as a sparse approximation of P^a has full rank. If N_{sp} of $\underline{P}^a$ is an integer close to the ensemble size of an EnKF, then the memory required by $\underline{P}^a$ is smaller than that by the ensemble matrix because of the symmetry of error covariance. The required computational load in Step 1 is extremely high if full state vectors are computed. Thanks to the sparsity, we only need to compute the entries with indices in $\mathcal{I}^b$. For a sparse UKF to be successful for problems with very high dimensions, it is critical to have component-based numerical models so that subsets of entries defined by $\mathcal{I}^b$ are computed in parallel; and most entries of state vectors are not evaluated at all.

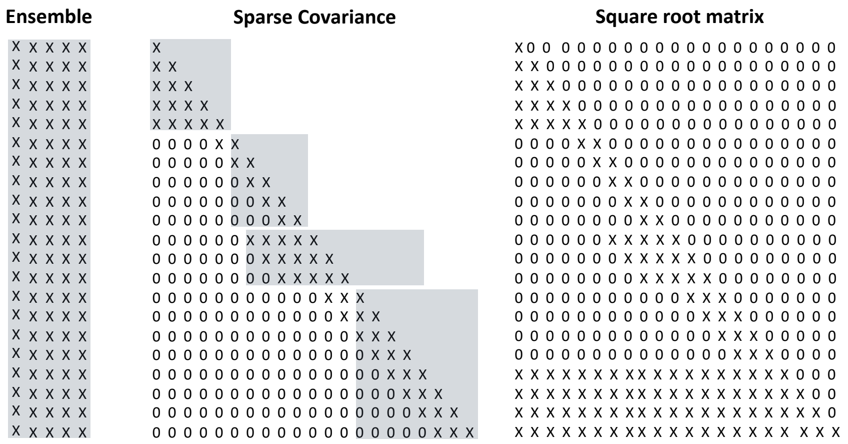

Figure 1: Patterns of ensemble vectors and sparse error covariances

In addition each component-based computation requires a part of the state vector only. It saves the computer I/O load.

Memory and (I/O) requirements are big factors influencing the efficiency of computational algorithms. Because an error covariance and its diagonal blocks are symmetric, the memory and I/O usage can be reduced by almost a half for subproblems with symmetry. The granularity of a computational algorithm has considerable impact on its efficiency in parallel computing. By granularity control we mean that one can divided a high dimensional problem into subproblems of desired dimensions. As shown in Figure 1, the error covariance and its square root consist of sparse blocks or sparse columns. This is different from the EnKF in which state vectors in an ensemble are dense. In a sparse approximation of P^a, the number of nonzero entries, a parameter similar to that used for distance-based localization methods, can be easily changed in $\underline{P^a}$ so that the error covariance and its square root can be grouped into smaller blocks of different sizes for parallel computation.

2.3 Lorenz-96 model

In this section, we use a Lorenz-96 model that was first introduced in [6] to test the performance of the sparse UKF. Consder

$$\begin{aligned}\frac{dx_i}{dt} &= (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \quad i = 1, 2, \cdots, n,\\ x_{n+1} &= x_1,\\ n &= 40,\\ \Delta t &= 0.025,\\ F &= 8.\end{aligned} \tag{19}$$

The system has chaotic trajectories as shown in Figure 2, a plot of $x_1(t), x_2(t), x_3(t)$. The simulations are conducted based on a 4th-order Runge-Kutta discretization. The trajectories are used

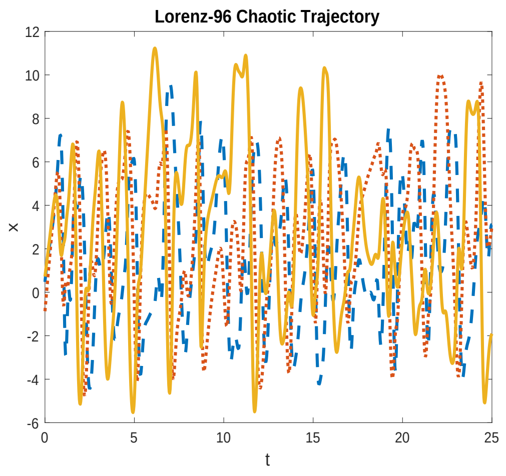

Figure 2: A chaotic trajectory of the Lorenz-96 model, x_1(solid), x_2(dash), x_3(dot).

as the ground truth. The sparsity pattern for $\underline{P}^a$ and $\sqrt{\underline{P}^a}$ are assumed to be centered along the diagonal line with a fix length. The total number of nonzero entries in each column is N_{sp}. We would like to point out that the sparse matrices are approximations of the true error covariance and its square root. The sparsity pattern of $\sqrt{\underline{P}^a}$ is, in fact, different from that of $\underline{P}^a$. In the approximation, however, we ignore the difference and use the same sparsity pattern for both. As a result, the memory required by $\sqrt{\underline{P}^a}$ is reduced almost by half. This idea works fine for the Lorenz-96 model. However, a systematic way of choosing the sparsity pattern for $\sqrt{\underline{P}^a}$ based on given $\underline{P}^a$ is an open problem that needs further study.

The numerical experimentation is based on $N = 1000$ uniformly distributed random initial states in $[-1\ 1]$. The time step size is $\Delta t = 0.025$. The total number of time steps for each simulation is $N_t = 4000$. The number of observations at any given time is $m = 20$, i.e. every other state variable is measured. The observational error has the Gaussian distribution with a covariance $R = I$, the identity matrix. The initial background error covariance is $P^b(0) = 0.2I$. The following RSME is used to measure the accuracy of estimation

$$RSME = \sqrt{\frac{1}{n(N_t+1)} \sum_{k=0}^{N_t} ||x^a(k) - x^{truth}(k)||_2^2}. \tag{20}$$

For comparison, an EnKF is applied to the same data set. The localization radius is $\rho = 4$ and the inflation factor is $\sqrt{1.08}$. A full scale UKF based on dense error covariance is also applied as the best estimator for comparison. As an indication of computational load, the number of entries to be computed in each algorithm is shown in Table 2. The boxplot of simulation results is shown in Figure 3. To summarize, the sparse UKF has considerably smaller error variation than that of the EnKF. This is expected because the new approach avoids the problem of rank deficiency. The medians of estimation errors are also smaller than that of EnKF. In the cases of $N_{sp} = 7$ and 11, the memory size required by the sparse error covariance is smaller than the memory size needed to store the ensemble vectors in the EnKF if $N_{ens} = 10$. The Cholesky factorization, while maintaining the sparsity property, may require additional memory. In this example of Lorenz-96, we use the same sparsity pattern for both $\sqrt{\underline{P}^a}$ and $\underline{P}^a$. This assumption simplifies the algorithm and reduces the memory and I/O requirements. The computational load, in the number of entry evaluations, is increased for sparse UKF because of the number of σ-points is $2n$. Reducing the number of entries being evaluated and testing the impact of Cholesky factorization on the efficiency of UKF is ongoing research that is not addressed in this paper.

Filter	Size	Entries EVAL	Error Median	Error Mean	Error STD
EnKF	$N_{ens} = 10$	400	0.3462	1.0741	1.0652
S-UKF	$N_{sp} = 7$	600	0.3061	0.3067	0.0071
S-UKF	$N_{sp} = 11$	920	0.2691	0.2691	0.0048

Table 2: Summary of simulation results

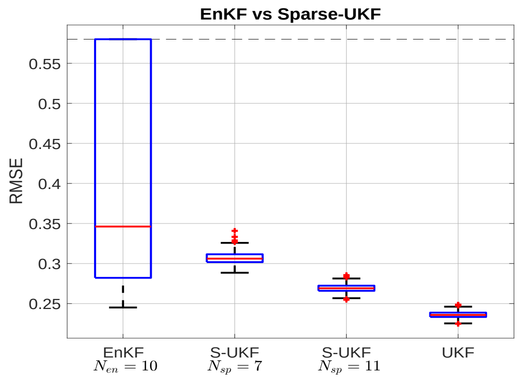

Figure 3: Boxplot of RMSE

3 Progressive EKF

In a sparse UKF, the σ-points are computed by taking a square root of the error covariance, such as the Cholesky factorization. The resulting σ-points are sparse. However, this process may require additional memory and computation. In this section, we propose a progressive algorithm of approximating error covariance without taking square roots.

3.1 Basic ideas

The main assumption for this algorithm is the following progressive relationship

$$M_{k-1}P^a(k-1)M_{k-1}^T = P^a(k-1) + \Delta P^b, \tag{21}$$

where ΔP^b is assumed to be small. In (21), M_{k-1} is the Jacobian of $\mathcal{M}$ at $x^a(k-1)$. Similarly, the Jacobian of $\mathcal{H}$ is H_k. To estimate ΔP^b, assume

$$M_{k-1} \quad = I + \Delta M_{k-1}. \tag{22}$$

where we assume that ΔM_{k-1} is small. If the system model is based on the discretization of a differential equation with a small time step size, then

$$\mathcal{M}(x(k-1)) = x(k-1) + O(\Delta t^{\alpha}), \quad \alpha \geq 1. \tag{23}$$

The Jacobian of $O_{k-1}(\Delta t^{\alpha})$ in space variables is expected to have small value if Δt is small, which makes (22) a reasonable assumption. Then we have

$$\begin{aligned}
&M_{k-1}P^a(k-1)M_{k-1}^T \\
&= (I + \Delta M_{k-1})P^a(k-1)(I + \Delta M_{k-1}^T) \\
&= P^a(k-1) + \Delta M_{k-1}P^a(k-1) + (\Delta M_{k-1}P^a(k-1))^T + \Delta M_{k-1}P^a(k-1)\Delta M_{k-1}^T \\
&\approx P^a(k-1) + \Delta M_{k-1}P^a(k-1) + (\Delta M_{k-1}P^a(k-1))^T .
\end{aligned} \tag{24}$$

This is in consistent with (21). It can be computed using a tangent linear model. Or it can be approximated using the dynamical model

$$\begin{aligned}
&M_{k-1}P^a(k-1)M_{k-1}^T \\
&= (I + \Delta M_{k-1})P^a(k-1)(I + \Delta M_{k-1}^T) \\
&\approx (\mathcal{M}(x(k-1) + \delta P^a(k-1)) - \mathcal{M}(x(k-1)))/\delta \\
&+ (\mathcal{M}(x(k-1) + \delta P^a(k-1)) - \mathcal{M}(x(k-1)))^T/\delta - P^a.
\end{aligned} \tag{25}$$

where $\delta > 0$ is the step size of a finite difference approximation of $\Delta M_{k-1}P^a$. Its value should be determined depending on the numerical model and its linearization. In (25), a vector and matrix summation is a new matrix resulting from adding the vector to every column in the matrix. Applying an operator to a matrix is to apply the operator to every column in the matrix.

3.2 Progressive EKF

The column vectors in the matrices in (24) and (25) are sparse. However, the number of column vectors equals n, which can be as high as $10^6 - 10^7$ for some atmospheric models. Applying a full model to all the vectors is impractical because of the high computational and I/O loads. On the other hand, if we approximate the error covariance using a given sparsity, only a small portion of the entries in each column vector is to be evaluated. Evaluating the entire state vector is unnecessary. This is the reason we need a component-based model. Then the algorithm of progressive EKF is summarized as follows.

Algorithm II (Progressive EKF)
Given the initial analysis at $t = k-1$,

$$x^a(k-1) \text{ and } \underline{P}^a(k-1). \tag{26}$$

Step 1. Forecast

$$\begin{aligned}
x^b &= \mathcal{M}(x^a(k-1)), \\
y^b &= \mathcal{H}(x^b).
\end{aligned} \tag{27}$$

Step 2. Background error covariance

$$\begin{aligned}\underline{P}^b = &\underline{\left(\mathcal{M}\left(x^a(k-1)+\delta \underline{P}^a(k-1),\mathcal{I}\right)-x^b\right)}/\delta \\ &+\underline{\left(\mathcal{M}\left(x^a(k-1)+\delta \underline{P}^a(k-1),\mathcal{I}\right)-x^b\right)^T}/\delta - \underline{P}^a + Q.\end{aligned} \tag{28}$$

Step 3. Kalman gain and analysis

$$\begin{aligned}K &= \underline{P}^b H_k^T (H_k \underline{P}^b H_k^T + R)^{-1},\\ x^a &= x^b + K(y_o - y^b),\\ \underline{P}^a &= (I - KH_k)\underline{P}^b.\end{aligned} \tag{29}$$

Different from the sparse UKF, this algorithm avoids the computation of matrix square roots. However, the algorithm requires that ΔP^b in (21) can be approximated effectively. From (23), the method is expected to work better for a small time step-size. If Δt is large, ΔM_{k-1} in (22) may not be small enough. A remedy is to use a refined step-size in an inner-loop computation. More specifically, the discrete model is a discretization of a continuous-time model. The discrete time moment $k-1$ corresponds to the continuous time moment $(k-1)\Delta t$. We refine the step size by dividing the time interval into n_p subintervals. In our examples, we choose $n_p = 2$. The refined time steps are

$$(k-1)\Delta t, (k-1)\Delta t + \frac{\Delta t}{n_p}, \cdots, (k-1)\Delta t + s\frac{\Delta t}{n_p}, \cdots, k\Delta t, \qquad 0 \le s \le n_p \tag{30}$$

For the inner loop, one can compute a sequence of backgrounds, $\tilde{x}^b(s)$.

$$\begin{aligned}t_s &= (k-1)\Delta t + s\frac{\Delta t}{n_p},\\ \tilde{x}^b(s) &= \tilde{\mathcal{M}}_{t_s}(x^a(k-1)), \quad s = 1, 2, \cdots, n_p.\end{aligned} \tag{31}$$

where $\tilde{\mathcal{M}}_{t_s}$ represents the refined model function in the time interval from $t = (k-1)\Delta t$ to $t = t_s$. In Step 2, repeat (28) n_p times along the sequence of background states, $\tilde{x}^b(s)$, without adding Q until the last round. This refined Step 2 increases the computational load, while improving the accuracy of the progressive estimation.

3.3 Examples

In the following, we apply the progressive EKF to the Lorenz-96 model using the same parameters as in (19). The error covariance is approximated using sparsity matrices with $N_{sp} = 7, 11, 17$. For $N_{sp} = 11$, we tested the idea of refining step-size using $n_p = 1$ and $n_p = 2$. The results are shown in Figure 4 and summarized in Table 3. Comparing to EnKF, the error variations of the progressive EKF are uniformly and significantly smaller. If $N_{sp} = 7$, which is smaller than the ensemble size $N_{ens} = 10$, the median value of estimation error is larger than that of the EnKF. The median error for $N_{sp} = 11$ is comparable to that of the EnKF. If a refined step-size in (31) is applied, for instance $n_p = 2$, the median estimation error is further reduced. Comparing to the performance of the sparse UKF, the error variations are similar. However, the estimation error of the sparse UKF has a smaller median in all cases. For example, to achieve a similar performance as the sparse UKF when $N_{sp} = 11$, one has to use a larger sparsity index $N_{sp} = 17$ for the progressive EKF. The computational load of the progressive EKF, in terms of the component evaluation, is comparable to the EnKF, as shown in Table 3.

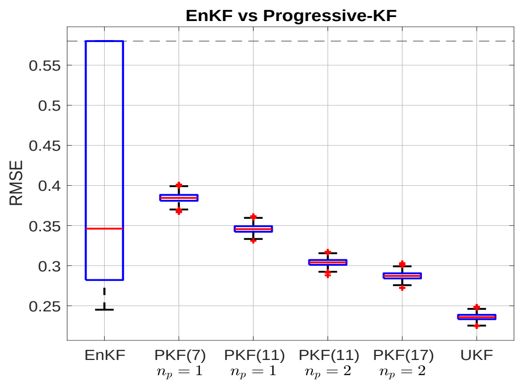

Figure 4: Boxplot of RMSE. For Prograssive-KF, $N_{sp} = 7$, 11, and 17

Filter	Size	Entries EVAL	Error Median	Error Mean	Error STD
EnKF	$N_{ens} = 10$	400	0.3462	1.0741	1.0652
P-KF	$N_{sp} = 7$ $N_p = 1$	320	0.3845	0.3846	0.0055
P-KF	$N_{sp} = 11$ $N_p = 1$	480	0.3455	0.3458	0.0050
P-KF	$N_{sp} = 11$ $N_p = 2$	480x2	0.3041	0.3041	0.0044
P-KF	$N_{sp} = 17$ $N_p = 3$	720x2	0.2872	0.2873	0.0046

Table 3: Summary of simulation results

4 Conclusions

Two algorithms of Kalman Filters based on sparse error covariances are introduced. They are tested using the Lorenz-96 model with 40 state variables and chaotic trajectories. Both algorithms share the same basic idea: the error covariance is approximated using a sparse matrix. Thanks to the sparsity, the required memory size is significantly reduced. The symmetry of the error covariance can potentially reduce the I/O load. The analysis error covariance can be updated as a sparse matrix in each cycle using a deterministic process, either a square root matrix or a progressive algorithm. The updated sparse matrix is then used as the background error covariance for the next cycle. Relative to the EnKF, the main advantage of the proposed methods is that the estimation process do not need an ensemble; and the error covariance has a full rank. The algorithms do not suffer issues of rank deficiency as in EnKFs. As a result, the variation of analysis error is constantly small in all examples. Techniques of localization and covariance inflation are unnecessary. Relative to 4D-Var methods, the proposed algorithms are mostly parallel. They provide not only the state estimate but also the analysis error covariance. For the purpose of scalability, we suggest that the proposed methods are applied with component-based numerical models. From the examples, the sparse UKF has better accuracy than the progressive EKF. If the computational load of taking square roots of sparse matrices is affordable, then the sparse UKF is the approach of our choice. On the other hand, the progressive EKF is a simple algorithm that avoids taking square roots of large matrices, provided that the progressive approximation of error covariance is adequately accurate. Although most conclusions drawn in this paper are based on simulations using the Lorenz-96 model, the algorithms are developed for general applications. Testing the methods using different types of system models is a main topic of our future work.

References

[1] T. A. Davis, Direct Methods for Sparse Linear Systems, SIAM, 2006.

[2] T. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, A survey of direct methods for sparse linear systems, Acta Numerica, Vol. 25, 2016, pp. 383-566.

[3] P. L. Houtekamer and F. Zhang, Review of the Ensemble Kalman Filter for Atmospheric Data Assimilation, Monthly Weather Review, Vol. 144, 2016, pp. 4489 - 4532.

[4] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators, *IEEE Trans. Automatic Control*, Vol. 45 (3), 2000, pp. 477-482.

[5] S. J. Julier and J. K. Uhlmann, Unscented filtering and nonlinear estimation, *Proceedings of the IEEE,* Vol. 92, No. 3, 2004, pp.

[6] E. Lorenz, Predictability – A problem partly solved. Seminar on Predictability, Vol. I, ECMWF, 1996.

[7] E. Rozin and S. Toledo, Locality of reference in sparse Cholesky factorization methods, Electronic Transactions on Numerical Analysis, Vol. 21, 2005, pp. 81-106.

[8] L. Xu, R. Rosmond, and R. Daley, Development of NAVDAS-AR: Formulation and initial tests of the linear problem, Tellus, 57A, 2005, pp. 546-559.