An efficient implementation of the ensemble Kalman filter based on an iterative Sherman–Morrison formula

Elias D. Nino Ruiz · Adrian Sandu · Jeffrey Anderson

Received: 15 July 2013 / Accepted: 10 February 2014 / Published online: 25 February 2014
© Springer Science+Business Media New York 2014

Abstract We present a practical implementation of the ensemble Kalman filter (EnKF) based on an iterative Sherman–Morrison formula. The new direct method exploits the special structure of the ensemble-estimated error covariance matrices in order to efficiently solve the linear systems involved in the analysis step of the EnKF. The computational complexity of the proposed implementation is equivalent to that of the best EnKF implementations available in the literature when the number of observations is much larger than the number of ensemble members, as typically is case in practice. Moreover, the proposed method provides the best theoretical complexity when it is compared to generic formulations of matrix inversion based on the Sherman–Morrison formula. The stability analysis of the proposed method is carried out and a pivoting strategy is discussed in order to reduce the accumulation of round-off errors without increasing the computational effort. A parallel implementation is discussed as well. Computational experiments carried out using an oceanic quasi-geostrophic model reveal that the proposed algorithm yields the same accuracy as other EnKF implementations, but scales better with regard to the number of observations.

Keywords Ensemble Kalman filter · Matrix inversion · Sherman–Morrison formula · Matrix decomposition

1 Introduction

The ensemble Kalman filter (EnKF) is a well-established, sequential Monte Carlo method to estimate the state and parameters of non-linear, large dynamical models (Evensen 2009) such as those found in atmospheric (Ott et al. 2004), oil reservoir (Evensen 2009b; Haugen et al. 2008), and oceanic (Alves and Robert 2005) simulations. The popularity of EnKF owes to its simple conceptual formulation and its relative ease implementation (Evensen 2009a). EnKF represents the error statistics by an ensemble of model states, and the evolution of error statistics is obtained implicitly via the time evolution of the ensemble during the forecast step. In the analysis step, information from the model and the measurements is combined in order to obtain an improved estimate of the true vector state. This process is repeated over the observed time period. In typical data assimilation applications, the dimension of state space (number of variables) ranges between $O(10^{7})$ and $O(10^{9})$, and the dimension of the observation space between $O(10^{5})$ and $O(10^{7})$. Consequently, the dimension of the linear systems solved during the analysis step is very large, and the computational cost considerable. In order to address this challenge we propose an efficient implementation of the EnKF analysis step based on an iterative application of the Sherman–Morrison formula.
The paper is structured as follows. Section 2 discusses the conceptual formulation of the EnKF and several efficient implementations available in the literature. Section 3 presents the novel implementation of the EnKF based on iterative Sherman–Morrison formula, in which the special structure of the measurements error covariance matrix is exploited. Computational cost and stability analyses are carried out for this approach, and pivoting and parallelization ideas are discussed. Section 4 reports numerical results of the proposed algorithm applied to the quasi-geostrophic model. Future work on parallelization is discussed in Sect. 5. Conclusions are drawn in Sect. 6.

2 Formulation of the EnKF

EnKF consists of two steps: the forecast and the analysis. An EnKF cycle starts with the matrix \( X^B \in \mathbb{R}^{n \times N} \) whose columns \( x^B_i \in \mathbb{R}^{n \times 1} \) form an ensemble of model states, all corresponding to the same model time \( t_{current} \):

\[
X^B = \left( x^B_1, x^B_2, \ldots, x^B_N \right) \in \mathbb{R}^{n \times N}.
\]

Typically \( x^B_i \) is an ensemble of model forecasts. Here \( n \) is the size of the model state vector, and \( N \) is the number of ensemble members. Each ensemble member \( x^B \) differs from the true state of the system \( x^{true} \in \mathbb{R}^{n \times 1} \), and we denote by \( \xi_i \in \mathbb{R}^{n \times 1} \) the corresponding error. The statistics of the ensemble of states is consistent with the background probability distribution.

The ensemble mean \( \bar{x}^B \in \mathbb{R}^{n \times 1} \) and the ensemble covariance matrix \( P^B \in \mathbb{R}^{n \times n} \) can be written as follows:

\[
x^B = \frac{1}{N} \sum_{i=1}^{N} x^B_i = \frac{1}{N} \left( X^B \cdot 1_{N \times 1} \right) \in \mathbb{R}^{n \times 1}, \tag{1a}
\]
\[
\bar{x}^B = \bar{x}^B \otimes 1^T_{N \times 1} \in \mathbb{R}^{n \times N}, \tag{1b}
\]
\[
P^B = \frac{1}{N-1} \cdot \left( X^B - \bar{x}^B \right) \cdot \left( X^B - \bar{x}^B \right)^T + Q \in \mathbb{R}^{n \times n}. \tag{1c}
\]

Here \( 1_{N \times 1} \in \mathbb{R}^{N \times 1} \) is a vector whose entries are all equal one. \( Q \) is the covariance matrix of model errors. In the typical case where \( x^B \) is an ensemble of model forecasts, the explicit addition of the matrix \( Q \) to the covariance formula is not necessary. Instead, the effect of model errors can be accounted for by adding random vectors \( \xi_i \sim \mathcal{N}(0, Q) \) to model states: \( x^B_i \leftarrow x^B_i + \xi_i \). Prior to any measurement, the forecast step provides the best estimation to the true vector state \( x^{true} \) (Suárez et al. 2012).

The vector of observations \( y \in \mathbb{R}^{m \times 1} \) is available at \( t_{current} \), where \( m \) is the number of data points. The observations are related to the model state by the relation

\[
y = H x + v
\]

where \( H \in \mathbb{R}^{m \times n} \) is the observation operator which maps the model space state into the observed space, and \( v \sim \mathcal{N}(0, R) \) is a vector of observation errors, accounting for both instrument and representativeness errors.

In order to account for observation errors one forms the matrix \( Y \in \mathbb{R}^{m \times N} \) whose columns \( y_i \in \mathbb{R}^{m \times 1} \) are perturbed measurements (Kovalenko et al. 2011):

\[
Y = (y + v_1, y + v_2, \ldots, y + v_N) = (y_1, y_2, \ldots, y_N) \in \mathbb{R}^{m \times N}.
\]

The vectors \( u_i \in \mathbb{R}^{m \times 1} \) represent errors in the data, and are drawn from a normal distribution \( v_i \sim \mathcal{N}(0, R) \), where \( R \) is the data error covariance matrix.

The EnKF analysis step produces an ensemble of improved estimates (analyses) \( X^A \in \mathbb{R}^{n \times N} \) by applying the Kalman filter to each of the background ensemble members:

\[
X^A = X^B + K \cdot \left( Y - H \cdot X^B \right) \in \mathbb{R}^{n \times N}, \tag{2a}
\]
\[
K = P^B \cdot H^T \cdot \left( H \cdot P^B \cdot H^T + R \right)^{-1} \in \mathbb{R}^{n \times m}, \tag{2b}
\]

where the matrix \( K \in \mathbb{R}^{n \times m} \) is the Kalman gain and quantifies the contribution of the background–observations difference to the analysis.

The EnKF forecast step uses the dynamical model operator \( M \) to evolve each member of the ensemble \( X^A \) from the current time \( t_{current} \) to the next time \( t_{next} \) where observations are available:

\[
X^B(t_{next}) = M_{t_{current} \rightarrow t_{next}} \left( X^A(t_{current}) \right) \in \mathbb{R}^{n \times N}. \tag{3}
\]

The forecast ensemble \( X^B \) is the background for the new EnKF cycle at \( t_{next} \). The analysis and forecast steps are repeated.

2.1 Efficient implementations of the analysis step

From Eqs. (2a), (2b) the analysis step can be written as

\[
X^A = X^B + P^B \cdot H^T \cdot Z, \tag{4}
\]

where \( Z \in \mathbb{R}^{m \times N} \) is the solution of the following linear system:

\[
\left( H \cdot P^B \cdot H^T + R \right) Z = \left( Y - H \cdot X^B \right) \in \mathbb{R}^{m \times N}. \tag{5}
\]

A direct solution of this linear system can be obtained using the Cholesky decomposition for matrix inversion (Lin and More 1999; Schnabel and Eskow 1990; Xia and Gu 2010). While this is a numerically stable and accurate approach (Gill et al. 1996; Meinguet 1983; Stewart and
Doore 1997), its application to (5) leads to the following complexity (Mandel 2006) of the analysis step:

\[ \mathcal{O} \left( m^3 + m^2 \cdot N + m \cdot N^2 + n \cdot N^2 \right). \]  

(6)

This is an acceptable complexity for a large number of degrees of freedom \( m \), but not for a large number of observations \( n \). An alternative is to solve (5), and the overall analysis step, using Singular Value Decomposition (SVD) based methods. Those methods exploit the special structure of the data error covariance matrix \( \mathbf{R} \), which is often (block) diagonal and can be easily factorized:

\[ \mathbf{R} = \text{diag} \left( \mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{n_{\text{block}}} \right), \]

with \( \mathbf{R}_k \in \mathbb{R}^{N_k \times N_k} \), where \( 1 \leq k \leq N_{\text{block}} \). \( N_{\text{block}} \) is the number of blocks in the matrix \( \mathbf{R} \) and:

\[ m = \sum_{k=1}^{N_{\text{block}}} N_k. \]

The matrix \( \mathbf{R} \) is a covariance matrix, and in practice it is always positive definite.

The observation operator \( \mathbf{H} \in \mathbb{R}^{m \times n} \) is sparse or can be applied efficiently to a state vector. Then, when \( \mathbf{R} \) is diagonal, we can express the system matrix (5) as follows:

\[ \hat{\mathbf{S}} = \left( \mathbf{X}^B - \mathbf{X}^B \right) \in \mathbb{R}^{n \times N}, \]

\[ \mathbf{W} = \sqrt{\mathbf{R}} \left[ \frac{1}{N-1} \sqrt{\mathbf{R}^{-1}} \mathbf{H} \hat{\mathbf{S}} \left( \mathbf{H} \hat{\mathbf{S}} \right)^T \sqrt{\mathbf{R}^{-1}} + \mathbf{I} \right] \sqrt{\mathbf{R}}. \]

(7)

Employ the singular value decomposition

\[ \sqrt{\mathbf{R}^{-1}} \cdot \mathbf{H} \cdot \hat{\mathbf{S}} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T \in \mathbb{R}^{m \times N}, \]

(8)

where \( \mathbf{U} \in \mathbb{R}^{m \times m} \) and \( \mathbf{V} \in \mathbb{R}^{N \times N} \) are orthogonal square matrices, and \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N) \in \mathbb{R}^{m \times N} \) is the diagonal matrix holding the singular values of \( \sqrt{\mathbf{R}^{-1}} \cdot \mathbf{H} \cdot \hat{\mathbf{S}} \in \mathbb{R}^{m \times N} \). The linear system (5) can be written as follows (Mandel 2006):

\[ \left[ \sqrt{\mathbf{R}} \mathbf{U} \left( \frac{\Sigma^2}{N-1} + 1 \right) \mathbf{U}^T \sqrt{\mathbf{R}} \right] \mathbf{Z} = \left( \mathbf{Y} - \mathbf{H} \mathbf{X}^B \right), \]

which yields the solution:

\[ \mathbf{Z} = \sqrt{\mathbf{R}}^{-1} \mathbf{U} \text{diag} \left\{ \left( \frac{\sigma_i^2}{N-1} + 1 \right)^{-1} \right\} \mathbf{U}^T \sqrt{\mathbf{R}}^{-1} \cdot \left( \mathbf{Y} - \mathbf{H} \mathbf{X}^B \right). \]

The overall complexity of the analysis step

\[ \mathcal{O} \left( N^2 \cdot m + N^3 + n \cdot N^2 \right) \]

(9)

is suitable for large \( n \) and \( m \), assuming \( N \) remains small. Many algorithms in the literature employ SVD (9) for the solution of the linear system (5) (Evensen 2009a). The analysis step is written in terms of the solution (9) in order to minimize the number of matrix computations. Due to this, the solution of the linear system and the improvement of the forecast ensemble are performed as a single step. The ensemble adjustment Kalman filter (EAKF) and the ensemble transform Kalman filter (ETKF) are based on this idea (Tippett et al. 2003). Other efficient implementations of the ensemble Kalman filter make use of SVD decompositions in order to derive pseudo-inverses, furthermore; these algorithms compute the inverse in the \( N \)-dimensional ensemble space rather than \( m \)-dimensional measurement space. Thus, in practice, when \( m \gg N \), those algorithms exhibit a good performance. All these methods have the overall complexity (number of long operations) given in (9).

A different approach is to employ iterative methods for solving the linear system (5), for instance the conjugate gradient method (Reid 1972; Golub and OLeary 1989; Cohen 1972; Eisenstat 1981) for \( N \) right-hand sides. However, each iteration costs \( \mathcal{O}(N^2 \cdot m) \), therefore iterative methods do not seem to be competitive for the solution of (5).

The well-established EnKF implementations presented above employ a Cholesky or SVD decomposition, which require considerable computational effort. The next section discusses an efficient implementation of the EnKF which does not require any decomposition prior to the solution of the linear system (5).

3 Iterative implementation of the EnKF analysis step

We make the assumptions (Tippett et al. 2003; Mandel 2006) that, in practice:

- The data error covariance matrix \( \mathbf{R} \) has a simple structure (e.g., is block diagonal).
- The observation operator \( \mathbf{H} \) is sparse or can be applied efficiently.
- The variables \( m \) and \( n \) are very large.
- In many real applications of the EnKF \( m \gg N \), and the number of variables ranges between \( \mathcal{O}(10^7) \) and \( \mathcal{O}(10^9) \).

Taking into account the previous assumptions, we now derive the implementation of the EnKF. We define the matrix of member deviations \( \mathbf{S} \in \mathbb{R}^{n \times N} \) as follows:

\[ \mathbf{S} = \frac{1}{\sqrt{N-1}} \cdot \left( \mathbf{x}_1^B - \bar{x}^B, \mathbf{x}_2^B - \bar{x}^B, \ldots, \mathbf{x}_N^B - \bar{x}^B \right) \in \mathbb{R}^{n \times N}, \]

(10)

which allows to write the ensemble covariance matrix as

\[ \mathbf{P}^B = \mathbf{S} \cdot \mathbf{S}^T \in \mathbb{R}^{n \times n}. \]

(11)

By replacing Eq. (11) in (5), the linear system solved during the analysis step is written as follows:
\(D = Y - H \cdot X^R \in \mathbb{R}^{m \times N},\)
\(V = H \cdot S \in \mathbb{R}^{m \times N},\)
\((R + V \cdot V^T) \cdot Z = D \in \mathbb{R}^{m \times N}.\)  \hfill (12a)
\hfill (12b)
\hfill (12c)

Note that
\[
W = R + V \cdot V^T = R + \sum_{i=1}^{N} v_i \cdot v_i^T
\]
can be computed recursively via the sequence of matrices
\(W^{(k)} \in \mathbb{R}^{m \times m}\) with \(W^{(0)} = R,
\]
\(W^{(k)} = W^{(k-1)} + v_k \cdot v_k^T, \quad k = 1, \ldots, N.\)  \hfill (13)

By replacing Eq. (13) in (12c) we obtain:
\[
(W^{(N-1)} + v_N \cdot v_N^T) \cdot Z = D.
\]  \hfill (14)

The linear system (14) can be solved by making use again of the Sherman–Morrison formula (Fraser 2008):
\[
(A + L \cdot M \cdot N)^{-1} = A^{-1} - A^{-1} L \left( M^{-1} + N A^{-1} L \right)^{-1} N A^{-1}
\]
with \(A = [W^{(N-1)}]^{-1},\)
\(L = v_{N\text{ens}},\)
\(M = 1\) and \(N = v_{N\text{ens}}^T.\)

The solution of (14) is computed as follows:
\[
Z = F^{(N)} - g^{(N)} \left( 1 + v_{N\text{ens}}^T g^{(N)} \right)^{-1} v_{N\text{ens}}^T F^{(N)} \in \mathbb{R}^{m \times N}
\]  \hfill (15)

where \(F^{(N)} \in \mathbb{R}^{m \times N}\) and \(g^{(N)} \in \mathbb{R}^{m \times N}\) are given by the solution of the following linear systems:
\[
W^{(N-1)} \cdot F^{(N)} = D \in \mathbb{R}^{m \times N},
\]  \hfill (16a)
\[
W^{(N-1)} \cdot g^{(N)} = v_N \in \mathbb{R}^{m \times 1}.\]  \hfill (16b)

Note that (16a) can be written as follows:
\[
W^{(N-1)} \cdot f_i^{(N)} = d_i \in \mathbb{R}^{m \times 1}, \quad 1 \leq i \leq N,
\]  \hfill (17)

where \(f_i^{(N)} \in \mathbb{R}^{m \times 1}\) and \(d_i \in \mathbb{R}^{m \times 1}\) are the i-th columns of the matrices \(F^{(N)}\) and \(D,\) respectively. Following (15), the i-th column of the matrix \(Z\) is given by:
\[
z_i = f_i^{(N)} - g^{(N)} \left( 1 + v_{N\text{ens}}^T g^{(N)} \right)^{-1} v_{N\text{ens}}^T f_i^{(N)} \in \mathbb{R}^{m \times 1}.
\]  \hfill (18)

By Eq. (18), the computation of \(Z\) involves the solution of the linear systems (16b) and (17). We apply the Sherman–Morrison formula again. The solution of the linear system (17) can be obtained as follows:
\[
f_i^{(N)} = f_i^{(N-1)} - g^{(N-1)} \cdot \left( 1 + v_{N\text{ens}}^T \cdot g^{(N-1)} \right)^{-1} \cdot v_{N\text{ens}}^T \cdot f_i^{(N-1)},
\]  \hfill (19)

where \(f_i^{(N-1)} \in \mathbb{R}^{m \times 1}\) and \(g^{(N-1)} \in \mathbb{R}^{m \times 1}\) are the solutions of the following linear systems, respectively:
\[
W^{(N-2)} \cdot f_i^{(N-1)} = d_i \in \mathbb{R}^{m \times 1},
\]
\[
W^{(N-2)} \cdot g^{(N-1)} = v_{N-1} \in \mathbb{R}^{m \times 1}.
\]

The linear system (16b) can be solved similarly. Note that the solution of each linear system involves the computation of two new linear systems, derived from the matrix sequence (13). Each of the new linear systems can be solved by applying recursively the Sherman–Morrison formula. For simplicity, we denote by \(f\) and \(g\) the solutions of the new linear systems in each recursively application of the Sherman–Morrison formula. Thus, for any column \(x\) of either matrix \(D\) or matrix \(V,\) we have that:
\[
\left[ W^{(k)} \right]^{-1} x = \left[ W^{(k-1)} \right]^{-1} x - \left[ W^{(k-2)} \right]^{-1} v_k
\]
\[
\left( 1 + v_k^T \left[ W^{(k-1)} \right]^{-1} v_k \right)^{-1} \left[ W^{(k-1)} \right]^{-1} x \in \mathbb{R}^{m \times 1},
\]
\[
\left[ W^{(k-1)} \right]^{-1} x = \left[ W^{(k-2)} \right]^{-1} x - \left[ W^{(k-2)} \right]^{-1} v_k
\]
\[
\left( 1 + v_k^T \left[ W^{(k-1)} \right]^{-1} v_k \right)^{-1} \left[ W^{(k-1)} \right]^{-1} x \in \mathbb{R}^{m \times 1},
\]

with \(\left[ W^{(0)} \right]^{-1} x = R^{-1} x\) and \(\left[ W^{(0)} \right]^{-1} v_1 = R^{-1} v_1.\) We note that:

- The computation of \(\left[ W^{(k)} \right]^{-1} \cdot x\) involves the solution of the linear systems \(W^{(k-1)} \cdot f = x\) and \(W^{(k-1)} \cdot g = v_k.\)
(13), the base case is the linear system $R^{-1} \cdot x$ in which the matrix $R$ is (block) diagonal.

From the previous analysis we derive a recursive Sherman–Morrison formula as follows. Define

$$S(x, k) = \begin{cases} z = R^{-1} x, & k = 0, \\ f = S(x, k - 1); & 1 \leq k \leq N, \\ g = S(x, k - 1), & 1 \leq k \leq N, \\ z = f - g (1 + v_k^T v_k)^{-1} v_k^T f; \end{cases}$$

where $x \in \mathbb{R}^{m \times 1}$. The columns of $Z$ are computed as follows:

$$z_i = S(d_i, N) \in \mathbb{R}^{m \times 1}, \quad 1 \leq i \leq N.$$  

The recursive computations performed by $S(\bullet)$ can be represented as a tree in which the solution $z \in \mathbb{R}^{m \times 1}$ of each node depends on the computations of its left ($f \in \mathbb{R}^{m \times 1}$) and right ($g \in \mathbb{R}^{m \times 1}$) children (i.e., on the solutions of two linear systems). Figure 1 illustrates the derivation of linear systems in order to solve $W^{(N)} \cdot z = d$ for $N = 3$ and $d \in \{d_1, d_2, d_3\}, \quad d \in \mathbb{R}^{m \times 1}.$

We see that $S(\bullet)$ solves multiple times identical linear systems. For instance, the repeated computations performed in order to solve $W^{(3)} \cdot z = d$ are represented as dashed nodes in Fig. 1. In this case, the linear system $R \cdot g = v_1$ is solved four times in the last level. The total number of linear systems to solve is $O(N \cdot 2^N)$, i.e., it increases exponentially with regard to the number of ensemble members if identical computations are not avoided. The next subsection discusses how to achieve this and obtain an efficient implementation of the recursive Sherman–Morrison formula.

3.1 An iterative Sherman–Morrison formula for matrix inversion

In order to avoid identical computations in Fig. 1 we can solve the linear systems from the last level of the tree up to the root level. The key idea is to compute only once the common computations per level.

We denote by $U \in \mathbb{R}^{m \times N}$ and $Z \in \mathbb{R}^{m \times N}$ the matrices holding partial results of the computations with regard to $V$ and $D$, respectively. Then, level 0 can be computed as follows:

$$Z^{(0)} = R^{-1} \cdot D,$$  

$$U^{(0)} = R^{-1} \cdot V$$  

(20a)  

(20b)

and level 1 $\leq k \leq N$

$$h^{(k)} = (1 + v_k^T u^{(k-1)}_k)^{-1} u^{(k-1)}_k,$$  

$$Z^{(k)} = Z^{(k-1)} - h^{(k)} \cdot (v_k^T \cdot Z^{(k-1)}),$$  

$$u^{(k)}_i = u^{(k-1)}_i - h^{(k)} \cdot (v_k^T \cdot u^{(k-1)}_i),$$

where $k + 1 \leq i \leq N$ and $u^{(k)}_i$ is the $i$-th column of matrix $U^{(k)}$. Note that, at iteration $k$, $h^{(k)}$ holds the common computation $x u^{(k)}_k - 1 = (W^{(k)} - 1) v_k$. Moreover, the columns $1 \leq i \leq k$ of matrix $U^{(k)}$ are not updated since they are not required for the next computations.

The computations performed by this iteration in order to solve $W^{(N)} \cdot z = d$ for $N = 3$ and $d \in \{d_1, d_2, d_3\}, \quad d \in \mathbb{R}^{m \times 1}$ are shown in Fig. 2.

Some key features of the iteration are highlighted next.

- The number of iterations is $N$.
- At level 0 matrices $Z^{(0)}$ and $U^{(0)}$ are computed according to (20). Notice, these computations does not imply a significant computational effort since they depend on the inverse of $R$.
- The matrix $W^{(k)} \in \mathbb{R}^{m \times m}$ is never stored in memory. It can be represented implicitly by matrix $V \in \mathbb{R}^{m \times N}$. This implicit representation realizes considerable memory savings, especially when $m \gg N$.
- At iteration $k$, only the columns $u^{(k)}_i$ with $k < i \leq N$ are updated.
Since the matrix $\mathbf{R}$ has a simple structure its inverse is easy to obtain. In the case of $\mathbf{R}$ (block) diagonal:

$$\mathbf{R}^{-1} = \text{diag} \left( \mathbf{R}^{-1\_1}, \mathbf{R}^{-1\_2}, \ldots, \mathbf{R}^{-1\_\text{block}} \right) \in \mathbb{R}^{m \times m},$$

and in general, under the assumptions done ($\mathbf{R}$ is easy to decompose), the computations $\mathbf{Z}^{(0)} = \mathbf{R}^{-1} \cdot \mathbf{D} \in \mathbb{R}^{m \times N}$ and $\mathbf{U}^{(0)} = \mathbf{R}^{-1} \cdot \mathbf{V} \in \mathbb{R}^{m \times N}$ can be performed with no more than $O(N^2 \cdot m)$ long operations.

Putting it all together, we define the iterative Sherman–Morrison formula $\mathbf{S}_\alpha (\mathbf{R}, \mathbf{V}, \mathbf{D})$ as follows:

- **Step 1.** Compute the matrices $\mathbf{Z}^{(0)} \in \mathbb{R}^{m \times N}$ and $\mathbf{U}^{(0)} \in \mathbb{R}^{m \times N}$ as follows:

$$\mathbf{Z}^{(0)} = \mathbf{R}^{-1} \cdot \mathbf{D}, \quad \mathbf{U}^{(0)} = \mathbf{R}^{-1} \cdot \mathbf{V}, \quad (21a)$$

where $\mathbf{R}^{-1}$ is computed according to its special structure.

- **Step 2.** For $k = 1$ to $N$ compute:

$$\mathbf{h}^{(k)} = \left( 1 + \mathbf{v}_k^T \cdot \mathbf{u}_k \right)^{-1} \mathbf{u}_k \in \mathbb{R}^{m \times 1}, \quad (22a)$$

$$\mathbf{Z}^{(k)} = \mathbf{Z}^{(k-1)} - \mathbf{h}^{(k)} \cdot \left( \mathbf{v}_k^T \cdot \mathbf{Z}^{(k-1)} \right) \in \mathbb{R}^{m \times N}, \quad (22b)$$

$$\mathbf{u}_{i}^{(k)} = \mathbf{u}_{i}^{(k-1)} - \mathbf{h}^{(k)} \cdot \left( \mathbf{v}_i^T \cdot \mathbf{u}_{i}^{(k-1)} \right) \in \mathbb{R}^{m \times 1}, \quad (22c)$$

for $k + 1 \leq i \leq N$.

We now use the iterative Sherman–Morrison formula in the analysis step to obtain an efficient implementation of the Ensemble Kalman filter (SMEnKF). This filter is as follows. The background ensemble states $\mathbf{X}^{B}$ are obtained from the forecast step (3), the ensemble mean $\mathbf{X}^{B}$ is given by (1a), and the ensemble deviations form the mean $\mathbf{S}$ are given by (10). The analysis is obtained as follows:

$$\mathbf{D} = \mathbf{Y} - \mathcal{H} \left( \mathbf{X}^{B} \right) \in \mathbb{R}^{m \times N},$$

$$\mathbf{V} = \mathcal{H} \left( \mathbf{S} \right) \in \mathbb{R}^{m \times N},$$

$$\mathbf{Z} = \mathcal{S}_\alpha \left( \mathbf{R}, \mathbf{V}, \mathbf{D} \right) \in \mathbb{R}^{m \times N},$$

$$\mathbf{X}^{A} = \mathbf{X}^{B} + \mathbf{S} \cdot \mathbf{V}^T \cdot \mathbf{Z} \in \mathbb{R}^{n \times N},$$

where the function $\mathcal{H} (\mathbf{G}) \in \mathbb{R}^{m \times N}$ is an efficient implementation of the observation operator applied to several state vectors, represented by $\mathbf{G} \in \mathbb{R}^{n \times N}$.

### 3.1.1 Inflation aspects

Inflation increases periodically the ensemble spread, such as to compensate for the small ensemble size, to simulate the existence of model errors, and to avoid filter divergence (Li et al. 2009). All the inflation techniques applied in traditional EnKF can be used, virtually without modification, in the context of SMEnKF. For example, after the forecast step, one can increase the spread of the ensemble

$$x_i \leftarrow x_i^{B} + \alpha \left( x_i - x_i^{B} \right), \quad 1 \leq i \leq N,$$

such as the ensemble covariance $\mathbf{P}^{B}$ is increased by a factor $\alpha^2$ (Wu et al. 2011).

### 3.1.2 Localization aspects

Using (2b), the analysis step can be written as follows:

$$\mathbf{X}^{A} = X^{B} + \mathbf{\Delta X}^{B},$$

$$\mathbf{\Delta X}^{B} = \mathbf{S} \cdot \mathbf{V}^T \cdot \mathbf{Z} = \mathbf{P}^{B} \cdot \mathbf{H}^T \cdot \left( \mathbf{H} \cdot \mathbf{P}^{B} \cdot \mathbf{H}^T + \mathbf{R} \right)^{-1} \mathbf{D} \quad (23)$$

$$\approx \mathbf{P}^{L}_{\ell} \cdot \mathbf{H}^T \cdot \left( \mathbf{H} \cdot \mathbf{P}^{L} \cdot \mathbf{H}^T + \mathbf{R} \right)^{-1} \mathbf{D}. \quad (24)$$

Localization techniques are explained in detail in Anderson (2007). Localization replaces the ensemble based $\mathbf{P}^{B}$ by a matrix $\mathbf{P}^{L}_{\ell} = \rho \circ \mathbf{P}^{B}$ in (23), where $\rho$ is a localization matrix and $\circ$ represents the Schur product.

Clearly localization in the form (24) requires the full covariance matrix, and cannot be applied in the context of the iterative Sherman–Morrison implementation. Applying SMEnKF with a single data point $\mathbf{y}_{i}$ leads to a correction $\mathbf{\Delta X}^{B}_{[i]}$, which can be localized by multiplication with a diagonal matrix $\hat{\Delta}_{[i]}$ that scales down components with the negative exponential of their distance to the observation $i$ location, and sets them to zero if outside the radius of influence:

$$\mathbf{\Delta X}^{B}_{[i]} = \hat{\Delta}_{[i]} \cdot \mathbf{S} \cdot \mathbf{V}^T \cdot \mathbf{Z}_{[i]}.$$  

This can be applied in succession for all data points to obtain a fully localized solution.

We discuss next a general approach to perform partial localization. Let $x_i$ be an individual component of the state vector and $y_j$ an individual observation. Define the impact factor $\delta_{i,j} \in [0, 1]$ of the information in $y_j$ on the state point $x_i$. For example, one can use a correlation length, and a radius about the measurement location outside which the impact factor is zero. Define the influence matrix $\Delta = (\delta_{i,j}) \in \mathbb{R}^{n \times n}$, and replace (24) with the following partial localization formula

$$\mathbf{\Delta X}^{B}_{[i]} \approx \mathbf{P}^{L}_{\ell} \cdot \mathbf{H}^T \cdot \left( \mathbf{H} \cdot \mathbf{P}^{L} \cdot \mathbf{H}^T + \mathbf{R} \right)^{-1} \mathbf{d}_{\ell} \in \mathbb{R}^{n \times N}$$

$$= \Delta \circ \left( \mathbf{S} \cdot \mathbf{V}^T \right) \cdot \mathbf{Z}.$$  

The $(i, \ell)$-th entry contains the $i$-th component the correction vector for the $\ell$-th ensemble member and reads

$$\mathbf{\Delta X}^{B}_{i,\ell} = \sum_{k=1}^{N} \mathbf{S}_{i,k} \sum_{j=1}^{m} \delta_{i,j} \mathbf{V}_{k,j} \mathbf{Z}_{j,\ell}. \quad (25)$$
for \(1 \leq i \leq n\) and \(1 \leq \ell \leq N\). The components of the correction matrix (25) are independent of one another, and can be evaluated in parallel after the system solution \(Z\) has been computed.

3.2 Computational complexity

In the complexity analysis of the iterative Sherman–Morrison formula we count only the long operations (multiplications and divisions). Moreover, as discussed before, we make the assumptions presented in Tippett et al. (2003), Mandel (2006), namely, the data error covariance matrix \(R \in \mathbb{R}^{m \times m}\) is inexpensive to decompose, and the observation operator \(H\) can be applied efficiently to any vector. We now analyze each step of the iterative Sherman–Morrison formula when \(R\) is diagonal, the extension to nondiagonal data error covariance matrices is immediate.

In the first step (21) each row \(i\) of matrices \(D \in \mathbb{R}^{m \times N}\) and \(V \in \mathbb{R}^{m \times N}\) is divided by the corresponding component \(r_i \in \mathbb{R}^m\) in order to obtain \(Z^{(0)} \in \mathbb{R}^{m \times N}\) and \(U^{(0)} \in \mathbb{R}^{m \times N}\) respectively. This yields to \(m \cdot N\) number of long operation for each matrix, therefore:

\[
T_{\text{step1}}(N, m) = 2 \cdot m \cdot N. \tag{26}
\]

In the second step (22) we compute the vector \(h^{(k)} \in \mathbb{R}^m\) (22a), and the matrices \(Z^{(k)} \in \mathbb{R}^{m \times N}\) (22b) and \(U^{(k)} \in \mathbb{R}^{m \times N}\) (22c). The number of long operations for each of one are as follows:

\[
h^{(k)} = \frac{u^{k-1}}{1 + v^T_k v^{k-1}},
\]

\[
Z^{(k)} = Z^{(k-1)} - h^{(k)} \cdot \left( \frac{v_k^T \cdot Z^{(k-1)}}{m} \right),
\]

\[
u^{(k)} = u^{(k-1)} - h^{(k)} \cdot \left( \frac{v_k^T \cdot u^{(k-1)}}{m} \right).
\]

Since the second step (22) is performed \(N\) times, the number of long operations can be expressed as:

\[
T_{\text{step2}}(N, m) = \sum_{k=1}^{N} \left( 2 \cdot m + \frac{N}{Z^{(k)}} + \sum_{j=1}^{k-1} \sum_{j=1}^{N} (2 \cdot m) \right) u^{(k)}
\]

\[
= 2 \cdot N \cdot m + 2 \cdot N^2 \cdot m.
\]

Consequently, from (26), (27), we have

\[
T_{\text{SMF}}(N, m) = 2 \cdot m \cdot N + 3 \cdot N^2 \cdot m + N \cdot m
\]

\[
\leq 3 \cdot (N^2 \cdot m + N \cdot m),
\]

which yields a complexity of

\[
O \left( N^2 \cdot m \right). \tag{28}
\]

Note that when \(R\) is not diagonal, under the assumptions done, the computations (21) of \(Z^{(0)}\) and \(U^{(0)}\) can be efficiently performed in \(O(m \cdot N^2)\) long operations; the overall effort becomes \(3 \cdot (N^2 \cdot m + N^2 \cdot m)\). This leads to the same complexity (28) for \(R\) diagonal, block diagonal, or in general easy to decompose.

The overall complexity of the analysis step for the SMEEnKF

\[
X^A = X^B + S \cdot \sqrt{Z},
\]

is:

\[
O \left( N^2 \cdot m + N^2 \cdot n \right). \tag{29}
\]

The complexity of the SMEEnKF is equivalent to the upper bounds on the methods described in Tippett et al. (2003), as detailed in the Table 1. The term \(N^3\) does not appear in the upper-bound of the proposed method even when \(R\) is not diagonal. This implies that SMEEnKF scales better than the other approaches when the size of the ensemble is increased.

Maponi (2007) proposed a general approach based on the Sherman–Morrison formula to solve linear systems. The application of this generic algorithm to (14) leads to an

| Analysis method | Computational cost |
|-----------------|--------------------|
| Direct (Tippett et al. 2003) | \(O(N^2 \cdot m + N^3 + N^2 \cdot n)\) |
| Serial (Anderson and Collins 2007) | \(O(N \cdot m + N \cdot m \cdot n)\) |
| (For each observation) | |
| ETKF (Anderson 2001) | \(O(N^2 \cdot m + N^3 + N^2 \cdot n)\) |
| EAKF (Anderson 2001) | \(O(N^2 \cdot m + N^3 + N^2 \cdot n)\) |
| SMEEnKF | \(O(N^2 \cdot m + N^2 \cdot n)\) |

The costs are functions of the ensemble size \(N\), number of observations \(m\) and state dimension \(n\).
increased computational cost as the special structure of the system (and special structure of $R$) are not exploited. The generic algorithm applied to EnKF analysis

$$W^{(N)} z_i = d_i \in \mathbb{R}^m,$$

for $1 \leq i \leq N$, (30) uses the decomposition [Maponi (2007), Remark 1]:

$$W^{(N)} = W^{(0)} + \sum_{i=1}^{m} u_i \cdot v_i^T,$$

where $W^{(0)} = \text{diag}(w_1, w_2, \ldots, w_m) \in \mathbb{R}^{m \times m}$ is a diagonal matrix holding the diagonal entries of $W^{(N)}$, $u_i$ is the i-th column of $U = W^{(N)} - W^{(0)} \in \mathbb{R}^{m \times m}$ and $v_i = e_i$ is the i-th element of the canonical basis in $\mathbb{R}^m$. Thus, according to [Maponi (2007), Corollary 4], each linear system (30) can be solved with $O(m^3)$ long operations, leading to a total of

$$O\left(N \cdot m^3\right).$$

Therefore the computational cost of the analysis step is:

$$O\left(N \cdot m^3 + N \cdot n\right),$$

which is larger than the computational cost of the SMEnKF when $m \gg N$. Moreover, according to [Maponi (2007), Theorem 3], when $m \gg N$, the solution of linear system (30) can be computed with no more than $O(N^2 \cdot m + N^2)$ long operations. The resulting computational cost of the analysis step is:

$$O\left(N \cdot m^3 + N \cdot m^3 + N \cdot n\right),$$

which is similar to the computational costs of the ETKF and EAKF methods when $m \gg N$. In this case, it is unclear how to construct the matrix $U \in \mathbb{R}^{m \times N}$ according to Maponi’s method; $U$ can not be chosen as we propose since $W^{(0)}$ must be diagonal and $V$ differs from our definition in (12b). In addition, Maponi’s algorithm requires the explicit representation in memory of the matrix $W^{(N)}$, which, in practice, is $O(10^2 \times 10^7)$ dimensional. In contradistinction, $W^{(N)}$ is not required explicitly in memory by our iterative Sherman–Morrison formula.

Lastly, in [Maponi (2007), Section 2] the author shows that the intermediate matrices $W^{(k)}$ is his method can be singular. In order to deal with this, Maponi proposes in [Maponi (2007), Algorithm 4] a method which makes use of partial pivoting in order to avoid singular matrices in $W^{(k)}$. The stability conditions are discussed in [Maponi (2007), Theorem 12] where the author shows that [Maponi (2007), Algorithm 4] can be carried out since (in our notation)

$$\begin{align*} I + V^T R^{-1} U^{(N)} &= R^{-1} W^{(N)} \in \mathbb{R}^{m \times m}, \end{align*}$$

(32)

is always positive definite, where the identity matrix $I$ is consistent with the dimension. Note that (32) holds only for $N = m$ in Maponi (2007). Due to this, the computational cost remains bounded by (31). Recent EnKF implementations based of Maponi’s method (Godinez and Moulton 2012) do not consider any stability conditions. In the next section we perform a stability analysis of the proposed iterative Sherman–Morrison formula.

3.3 Stability analysis

The solution of the linear system (5) by the iterative Sherman–Morrison formula yields the next sequence of matrices during the computation of $[W^{(N)}]^{-1}$:

$$\begin{align*} [W^{(0)}]^{-1} &= R^{-1} \\
[W^{(1)}]^{-1} &= R^{-1} - \frac{1}{\gamma_1} u_{1}^0 \cdot v_1^T \cdot R^{-1} \\
&= \left(I - \frac{1}{\gamma_1} u_{1}^0 \cdot v_1^T\right) [W^{(0)}]^{-1} \\
&\vdots \\
[W^{(k)}]^{-1} &= \left(I - \frac{1}{\gamma_k} u_k^{k-1} \cdot v_k^T\right) [W^{(k-1)}]^{-1} \\
\end{align*}$$

where

$$\gamma_k = 1 + v_k^T \cdot u_k^{k-1} \in \mathbb{R}, \text{ for } 1 \leq k \leq N.$$

The following situations may affect the proposed method:

- If any step produces $\gamma_k = 0$, then subsequent steps cannot proceed.
- Round-off errors can be considerably amplified if $\gamma_k \approx 0$ (numerical instability).
- If any matrix $W^{(k)}$ in the sequence:

$$\{W^{(0)}, W^{(1)}, \ldots, W^{(N)}\},$$

is singular, the algorithm cannot proceed.

We now show that the positive definiteness of the covariance matrix $R$ is a sufficient in order to guarantee the stability of the iterative Sherman–Morrison formula.

**Theorem 1** Assume that $R$ is positive definite with $\xi^T R \xi \geq \alpha \| \xi \|^2$ for any $\xi \in \mathbb{R}^m$. Then all matrices $W^{(k)}$ are positive definite with $\xi^T W^{(k)} \xi \geq \alpha \| \xi \|^2$ for any $\xi \in \mathbb{R}^m$.

**Proof** First, $W^{(0)} = R$ is positive definite. Next, we proceed by finite induction and assume that $W^{(k-1)}$ is positive definite with $\xi^T W^{(k-1)} \xi \geq \alpha \| \xi \|^2$. From (14) we have that:

$$W^{(k)} = W^{(k-1)} + v_k \cdot v_k^T.$$
and therefore \( \mathbf{W}^{(k)} \) is also positive definite:

\[
\xi^T \mathbf{W}^{(k)} \xi = \xi^T \mathbf{W}^{(k-1)} \xi + \left( \xi^T \mathbf{v}_k \right)^2 \geq \alpha \| \xi \|^2 \quad \forall \xi \in \mathbb{R}^m.
\]

Theorem 2  Assume that \( \mathbf{R} \) is positive definite. The sequence of values \( \gamma_k \) generated by the algorithm are strictly greater than one for all \( 1 \leq k \leq N \).

Proof  By the iterative Sherman–Morrison formula, the common computations \( (\mathbf{u}_i^{(k)})_{k+1} \) are given by:

\[
\mathbf{u}_i^{(0)} = R^{-1} \cdot \mathbf{v}_1 = \left[ \mathbf{W}^{(0)} \right]^{-1} \cdot \mathbf{v}_1,
\]

\[
\mathbf{u}_i^{(1)} = \left( I - \frac{1}{\gamma_1} \cdot \mathbf{u}_i^{(0)} \cdot \mathbf{v}_1^T \right) \cdot \left[ \mathbf{W}^{(0)} \right]^{-1} \cdot \mathbf{v}_2
\]

\[
\vdots
\]

\[
\mathbf{u}_k^{(k)} = \left( I - \frac{1}{\gamma_k} \cdot \mathbf{u}_k^{(k-1)} \cdot \mathbf{v}_k^T \right) \cdot \left[ \mathbf{W}^{(k-1)} \right]^{-1} \cdot \mathbf{v}_{k+1}
\]

Since \( \mathbf{W}^{(k-1)} \) is positive definite we have:

\[
\gamma_k = 1 + \mathbf{v}_k^T \cdot \mathbf{u}_k^{(k-1)} = 1 + \mathbf{v}_k^T \cdot \left[ \mathbf{W}^{(k-1)} \right]^{-1} \cdot \mathbf{v}_k > 1,
\]

consequently \( \gamma_k > 1 \) for all \( 1 \leq k \leq N - 1 \).

Theorem 1 leads directly to the following

Corollary 1  Assume that \( \mathbf{R} \) is positive definite. At iteration \( k \), the linear system:

\[
\mathbf{W}^{(k)} \cdot \mathbf{Z}^{(k)} = \mathbf{D},
\]

has a unique solution, for \( 1 \leq k \leq N \).

3.4 Pivoting

Theorem 2 shows that \( \gamma_k \) values cannot be near zero. Due to this, we expect that the round-off errors will not increase considerably during an iteration of the iterative Sherman–Morrison formula since:

\[
\frac{1}{\gamma_k} \in (0, 1).
\]

The following pivoting strategy can be (optionally) applied in order to further decrease round-off error accumulation. It consists of interchanging the columns of matrices \( \mathbf{V} \) and \( \mathbf{U}^{(k-1)} \) such that the pair \((\mathbf{v}_j, \mathbf{u}_j^{(k-1)})\) maximizes \( \gamma_k \). Formally, at iteration \( k \), prior the matrix computations (22), we look for a column index \( i_k \) such that:

\[
i_k = \arg \max_i \left\{ 1 + \mathbf{v}_i^T \cdot \mathbf{u}_i^{(k-1)} \right\}, \quad k \leq i \leq N.
\]  

(33)

and then, the columns \( i_k \) and \( i_k \) are interchanged in matrices \( \mathbf{V} \) and \( \mathbf{U}^{(k-1)} \).

The iterative Sherman–Morrison formula with pivoting gives the next computational cost:

\[
T_{\text{SMF}}^{\text{Piv}}(N, m) = 3 \cdot (N^2 \cdot m + N \cdot m) + \sum_{k=1}^{N} \sum_{i=1}^{m} m,
\]

which yields to:

\[
T_{\text{SMF}}^{\text{Piv}}(N, m) = O \left( N^2 \cdot m \right),
\]

from which we can conclude that seeking the maximum value of \( \gamma_k \) according to (33) does not increase the computational cost of the iterative Sherman–Morrison formula. Consequently, the overall complexity in the analysis step of SMEnKF remains bounded by (29).

4 Experimental results

In this section several computation tests are conducted in order to assess the accuracy and running time of the SMEnKF.

4.1 Experimental setting

The SMEnKF as well as the EnKF implementations based on Cholesky (EnKFCH) and SVD (EnKFSVD) are coded in Fortran 90. The LAPACK and BLAS libraries (Anderson et al. 1990) are utilized in order to obtain efficient implementations of the Cholesky and SVD decompositions, as well as of other computations in SMEnKF as follows:

- The matrix \( \mathbf{W} \in \mathbb{R}^{m \times m} \) is built using DSYTR function as follows:

\[
\mathbf{W}^{(N)} = \alpha \cdot \mathbf{V} \cdot \mathbf{V}^T + \beta \cdot \mathbf{R} \in \mathbb{R}^{m \times m},
\]

with \( \alpha = 1/(N - 1) \) and \( \beta = 1 \).

- The functions DPOTRF and DPOTRI are used to compute the Cholesky decomposition of matrix \( \mathbf{W} \). Only the upper triangular of \( \mathbf{W}^{(N)} \) is stored.

- The SVD decomposition is performed by the DGESVD function. Only the first \( N \) pair of eigenvectors/values are computed. The right eigenvectors are not computed since they are not required.

- In the SMEnKF, the update

\[
\mathbf{Z}^{(k)} = \mathbf{Z}^{(k-1)} - \alpha \cdot x \cdot y^T.
\]
is performed making use of the DGGER function with \( \alpha = 1/k \), \( \mathbf{x} = \mathbf{u}_k \text{ }^{k-1} \), and \( \mathbf{y} = \mathbf{y}_k^T \cdot \mathbf{z}^{(k-1)} \). \( \mathbf{U}^{(k)} \) is updated in a similar way.

In order to measure the quality of the solutions we employ the following performance metrics. The elapsed time (ET) measures the overall simulation time for a method \( \ast \). This metric is defined as

\[
\text{ET}(\ast) = \text{Forecast}_\ast + \text{Analysis}_\ast
\]

(34)

where Forecast\( _\ast \) and Analysis\( _\ast \) are the overall running times for the forecast and analysis steps respectively.

The root mean square error (RMSE) is defined as

\[
\varepsilon(\ast) = \text{RMSE} = \sqrt{\frac{1}{N_{\text{steps}}} \left( \sum_{t=1}^{N_{\text{steps}}} \text{RSE}_t^2 \right)}
\]

where \( N_{\text{steps}} \) is the number of time steps and \( \text{RSE}_t \) is the root square error at time \( t \) defined as follows:

\[
\text{RSE}_t = \sqrt{\frac{1}{n} \left( (\mathbf{x}_t^{\text{true}} - \mathbf{x}_t^A)^T \cdot (\mathbf{x}_t^{\text{true}} - \mathbf{x}_t^A) \right)}
\]

where \( \mathbf{x}_t^{\text{true}} \) is the true vector state at time \( t \), and \( \mathbf{x}_t^A \) is the analysis ensemble mean at time \( t \). As can be seen the RMSE measures in average the distance between a reference solution \( (\mathbf{x}_t^{\text{true}}) \) and the given solution \( (\mathbf{x}_t^A) \).

The EnKF implementations are tested using a quasi-geostrophic (QG) model Carton and Baraille (1994) which defines the model operator \( (\mathcal{M}) \) in the EnKF experiments. To compare the performance of different EnKF implementations we measure the elapsed times and the accuracy of analyses for different values of \( m \) and \( N \).

### 4.2 Quasi-geostrophic model

The Earth’s ocean has a complex flow system influenced by the rotation of the Earth, the density stratification due to temperature and salinity, as well as other factors. QG is a simple model which mimics the behavior of the ocean. It is defined by the following partial differential equations (Carton and Baraille 1994):

\[
\frac{\partial \Delta \Psi}{\partial t} + r_n \frac{\partial \Psi}{\partial x} + \beta \frac{\partial \Psi}{\partial y} = -r_b \Psi + r_h \Delta \Psi - r_g \Delta^2 \Psi + \sin(2\pi y) \frac{\text{External Force}}{\text{(35)}}
\]

where \( \Psi \) is the stream function, \( \Delta \) is the Laplacian operator

\[
\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}
\]

\( F \) is the Froud number, \( r_n \) represents the Rossby number, \( r_h \) is the horizontal friction and \( r_g \) is the biharmonic horizontal friction and \( x \) and \( y \) represent the horizontal and vertical spatial coordinates. Note that the stream function and the potential vorticity are related via the Laplacian operator. This elliptic property reflects the assumption that the flow is geostrophically balanced in the horizontal direction, and hydrostatically balanced in the vertical direction.

The QG experiment studies the behavior of EnKF implementations when \( m \gg N \) as is usually the case in practice. We consider three different grids, denoted \( Q_{D_1 \times D_2} \), where the number of horizontal and vertical grid points in space are \( D_1 \) and \( D_2 \), respectively. Specifically, we employ in experiments \( Q_{33 \times 33} \) (small instance), \( Q_{65 \times 65} \) (medium instance) and \( Q_{129 \times 129} \) (large instance). The horizontal and vertical dimensions of the grid are denoted by \( L_x \) and \( L_y \) respectively. For all the instances the parameters \( r_h = 10^{-6} \), \( r_g = 10^{-7} \), \( r_g = 2^{-12} \), \( \beta = 1.0 \) and \( r = 10^{-5} \) are fixed. The other parameters values are summarized in Table 2.

| Instance     | \( L_x \) | \( L_y \) | \( D_1 \) | \( D_2 \) |
|--------------|-----------|-----------|-----------|-----------|
| \( Q_{33 \times 33} \) | 0.4       | 0.4       | 33         | 33         |
| \( Q_{65 \times 65} \) | 1.0       | 1.0       | 65         | 65         |
| \( Q_{129 \times 129} \) | 1.0       | 1.0       | 129        | 129        |

\( L_x \) and \( L_y \) represent the horizontal and vertical grid sizes, and \( N \) and \( M \) are the number of horizontal and vertical grid points, respectively.

\( \Delta \Psi \) is the potential vorticity,

\[
\mathcal{J}(\Psi, \Delta \Psi) = \frac{\partial \Psi}{\partial x} \cdot \Delta \Psi - \frac{\partial \Psi}{\partial y} \cdot \frac{\partial \Delta \Psi}{\partial x} + \frac{\partial \Delta \Psi}{\partial y}
\]

– There are 1,200 time steps, each one representing 1.27 days in the ocean.
– The vorticity of the ocean at each grid point provides a component of the vector state.
– The computation of the stream function is done through the solution of the Helmholtz equation (Otto and Larsson 1999).
– Homogeneous Dirichlet boundary conditions are assumed. Due to this, the boundaries of the grid are not mapped into the state vector, and \( n = (D_1 - 2) \cdot (D_2 - 2) \).
– The initial ensemble members are

\[
\mathbf{x}_i^0 = \mathbf{x}_i^{\text{true}} + \varepsilon_i^0 \cdot \left( \frac{1}{n} \cdot \sum_{k=1}^{n} \mathbf{x}_k^{\text{true}} \right) \in \mathbb{R}^{n \times 1}
\]
for $1 \leq i \leq N$, where $e^B$ is drawn from a normal distribution with zero mean and covariance matrix $B_0 = \sigma_B^2 \cdot I \in \mathbb{R}^{n \times n}$. For testing purposes, three values are assumed for the standard deviation of background errors $\sigma_B \in \{0.05, 0.1, 0.15\}$.

- The number of observation per simulation, for each size $(n)$ of the model state, is defined as follows:

$$m = P_{\text{obs}} \cdot n,$$

where $P_{\text{obs}}$ is the percentage of components observed from the model state. We consider $P_{\text{obs}} \in \{50, 70, 90\\}$.

- Measurements are taken every 10 time units and they are constructed as follows:

$$y_i = Hx_{\text{true}} + \nu_i \in \mathbb{R}^{m \times 1},$$

where $\nu_i \sim N(0, R)$ with $R = \sigma_O^2 \cdot I \in \mathbb{R}^{m \times m}$ with $\sigma_O = 0.01$.

- For the time evolution of the model, zero boundary conditions are assumed and the boundaries are not included onto the ensemble representation. Due to this, the dimension of the vector state $n = (N - 2) \cdot (M - 2)$.

- For each instance we consider simulations with $N \in \{20, 60, 100\}$ ensemble members. The number of ensemble members is one to two orders of magnitude smaller than the total number of observations.

The RSME values for analysis errors for the $Q_{33 \times 33}$, $Q_{65 \times 65}$ and $Q_{129 \times 129}$ instances are shown in Tables 6, 7, and 8, respectively. The results depend on the number of ensemble members $(N)$, the number of observations $(m)$, and the deviation of the initial ensemble mean $(\sigma_B)$. The RSME quantifies errors in the stream function $\Psi$. In terms of accuracy there is no significant difference between different EnKF implementations. As expected, when the error in the initial ensemble is increased, the accuracy in the analysis decreases. The error does not show an exponential growth, even when the number of components in the model state $(n)$ is much larger than the number of ensemble members (e.g., for the $Q_{129 \times 129}$ instance). When the number of ensemble members is increased, the analysis error is decreased. This is illustrated by the snapshots of the $Q_{33 \times 33}$ simulation over 1200 time steps presented in Fig. 4. There, we can clearly see that the ensemble of size 100 provides a better estimation ($\hat{x}^{100}$) than the true state of the model ($x^{\text{true}}$) than the ensembles of sizes 20 and 60. Additionally, the number of observations plays an important role in the estimation of the true model state when the size of the vector state is much larger than the number of ensemble members (Tables 3, 4, 5).

The ET values for the $Q_{33 \times 33}$, $Q_{65 \times 65}$ and $Q_{129 \times 129}$ instances are shown in Tables 6, 7 and 8, respectively. The time is expressed in seconds (s) if it is below 30 minutes, and otherwise is expressed in minutes (min) and hours (h). The EnKCH shows good performance when the number of observations is small. From Table 3 (the blocks where the number of observations are 480, 672 and 864) we see that...
Fig. 4  Snapshots of the $Q_{33 \times 33}$ simulation for $N = 20, 60$ and $100$ members, at the time steps $t = 0, 239, 478, 717, 956$ and $1,195$ (out of 1,200). As expected, when the number of ensemble members is increased the estimation of the true state ($x^{\text{true}}$) is improved (the RMSE is decreased).

The EnKFSVD shows a good performance for the $Q_{33 \times 33}$ and $Q_{65 \times 65}$ instances. We note that, for those instances, when the number of ensemble members is the largest tested ($N = 100$), the EnKFSVD performs relatively better than the SMEEnKF implementation. We say relatively because the difference is just given by seconds. This is expected since both methods provide similar computational complexities. However, it is possible that these two methods scale different with regard to the dimension of the model, the number of observations and the size of the ensemble.

The SMEEnKF is sensible to the number of ensemble members when the condition $m \gg N$ is not fully satisfied. For instance, consider the instances $Q_{33 \times 33}$ and $Q_{65 \times 65}$. There, the SMEEnKF performs well when the number of observations and ensemble members are relatively close. This obeys
to the computational efforts described in Table 1. We note that the term $N^3$ does not appear in the computational cost of the proposed implementation but it appears on the EnKF-FSVD bound (9). Thus, when the number of observations is not much greater than the number of ensemble members, the SMEnKF provides better performance than the EnKFSVD. However, for those scenarios, when the size of the ensemble is increased and the number of observations is hold, the SMEnKF performance is altered considerably. Recall that, the computational effort of the iterative Sherman–Morrison formula is given by $O(N^2 \cdot m)$, therefore we can expect low performances when $m$ is not much greater than $N$. On the other hand, when the number of observations is several order of magnitude larger than the number of ensemble members (i.e. in $QG_{129\times 129}$ is approximately 2), the SMEnKF provides the best performance. For instance, in some cases, the SMEnKF is 2.12 min faster than the EnKFSVD. This magnitude is large enough considering the size of the model and the similar theoretical complexity for both methods. Then, even when both methods provide the same theoretical computational cost, they do not scale similar, the SMEnKF is less sensitive to increments in the number of observations than the EnKFSVD when $m \gg N$, which is the relevant case in practice. Thus, the results of this test leads to conclude that the SMEnKF is not sensitive to the increase in the number of observations when this is considerably larger than the number of ensemble members, making it attractive for implementation with large-scale observational systems.

5 Future work: parallel implementation

In this section we discuss an efficient parallel implementation of the iterative Sherman–Morrison formula. Since the

| $N$ | $m$ | $\sigma_B$ | SMEnKF (s) | EnKFC (s) | EnKFSVD (s) |
|-----|-----|------------|------------|-----------|-------------|
| 20  | 480 | 0.05       | 15.2       | 33.4      | 17.2        |
| 1.984 | 0.05 | 60.0       | 44.9 min   | 68.9      |
| 0.10 | 60.2 | 33.4 min   | 67.9      |
| 0.15 | 60.7 | 45.4 min   | 68.5      |
| 672 | 0.05 | 15.6       | 61.9       | 17.0      | 33           |
| 2,778 | 0.05 | 61.5       | 1.8 h      | 67.8      |
| 0.10 | 61.3 | 1.3 h      | 70.2      |
| 0.15 | 61.0 | 1.4 h      | 68.5      |
| 864 | 0.05 | 15.9       | 113.7      | 17.7      | 3            |
| 3,572 | 0.05 | 62.4       | 3.8 h      | 70.3      |
| 0.10 | 62.1 | 3.3 h      | 71.0      |
| 0.15 | 62.1 | 3.0 h      | 70.4      |
| 60  | 480 | 0.05       | 40.4       | 57.8      | 40.9        |
| 1,984 | 0.05 | 161.4      | 45.0 min   | 168.3     |
| 0.10 | 161.2 | 55.3 min   | 166.5     |
| 0.15 | 162.3 | 51.9 min   | 167.8     |
| 672 | 0.05 | 42.3       | 90.3       | 43.4      | 3            |
| 2,778 | 0.05 | 170.4      | 2.4 h      | 172.8     |
| 0.10 | 170.2 | 1.9 h      | 175.4     |
| 0.15 | 169.9 | 2.5 h      | 174.6     |
| 864 | 0.05 | 44.3       | 150.6      | 44.4      | 3            |
| 3,572 | 0.05 | 179.2      | 4.1 h      | 180.7     |
| 0.10 | 178.2 | 2.9 h      | 181.5     |
| 0.15 | 177.5 | 4.3 h      | 179.9     |
| 100 | 480 | 0.05       | 70.0       | 83.2      | 69.1        |
| 1,984 | 0.05 | 281.8      | 52.4 min   | 279.3     |
| 0.10 | 281.5 | 40.1 min   | 280.4     |
| 0.15 | 280.8 | 58.4 min   | 280.5     |
| 672 | 0.05 | 75.5       | 118.9      | 72.5      | 2            |
| 2,778 | 0.05 | 303.2      | 1.7 h      | 295.6     |
| 0.10 | 303.4 | 1.7 h      | 294.8     |
| 0.15 | 303.3 | 2.9 h      | 295.5     |
| 864 | 0.05 | 80.9       | 209.1      | 76.7      | 4            |
| 3,572 | 0.05 | 327.9      | 4.8 h      | 313.4     |
| 0.10 | 327.2 | 4.0 h      | 313.4     |
| 0.15 | 327.3 | 5.1 h      | 315.7     |

Different numbers of ensemble members and numbers of observations are considered
algorithm (21)–(22) can be applied individually to each column of the matrices $Z^{(0)} \in \mathbb{R}^{m \times N}$ and $U^{(0)} \in \mathbb{R}^{m \times N}$, there are $2N$ computations that can be performed in parallel. We define the matrix $G^{(0)} \in \mathbb{R}^{m \times 2N}$ holding the columns of $V \in \mathbb{R}^{m \times N}$ and $D \in \mathbb{R}^{m \times N}$ as follows:

$$G^{(0)} = [V, D] = [v_1, v_2, \ldots, v_N, d_1, d_2, \ldots, d_N],$$
$$= [g^{(0)}_1, \ldots, g^{(0)}_N, \tilde{g}^{(0)}_{N+1}, \ldots, \tilde{g}^{(0)}_{2N}],$$

Let $N_{\text{proc}}^0$ be the number of available processors at the initial time. The number of operations per processor is

$$C_p^0 = \frac{2 \cdot N}{N_{\text{proc}}^0}.$$

The matrix (36) can be written as

$$G^{(0)} = [B_1^{(0)}, B_2^{(0)}, \ldots, B_{N_{\text{proc}}^0}^{(0)}],$$

where the blocks $B_i^{(0)} \in \mathbb{R}^{m \times C_p^0}$ are

$$B_i^{(0)} = \begin{bmatrix} g^{(0)}_{(i-1)C_p^0+1} & \cdots & g^{(0)}_{(i-1)C_p^0+2} & \cdots & g^{(0)}_{iC_p^0} \end{bmatrix},$$

for $1 \leq i \leq N_{\text{proc}}^0$. The parallel, first step (21) of the iterative Sherman–Morrison formula is implemented as an update over the blocks:

$$B_i^{(1)} = R^{-1} \cdot B_i^{(0)}, B_i^{(1)} \in \mathbb{R}^{m \times C_p^0}, \text{ for all } 1 \leq i \leq N_{\text{proc}}^0,$$

which yields

$$G^{(1)} = \begin{bmatrix} R^{-1} \cdot B_1^{(0)}, R^{-1} \cdot B_2^{(0)}, \ldots, R^{-1} \cdot B_{N_{\text{proc}}^0}^{(0)} \end{bmatrix} = \begin{bmatrix} B_1^{(1)}, B_2^{(1)}, \ldots, B_{N_{\text{proc}}^0}^{(1)} \end{bmatrix}.$$
The second step (22) of the iterative Sherman–Morrison formula consists of a sequence of updates applied to the matrices $Z^{(0)}$ and $U^{(0)}$. Such matrices are represented by the columns of matrix $G^{(1)}$. Thus, consider the computation of level one, each column of the matrix $G^{(1)}$ can be updated as follows:

$g_i^{(2)} = g_i^{(1)} - g_i^{(1)} \left(1 + v_i^T \cdot g_i^{(1)}\right)^{-1} \left(v_i^T \cdot g_i^{(1)}\right) \in \mathbb{R}^{m \times 1}$,

for $2 \leq i \leq 2 \cdot N$. Similarly to the first step, the computations can be grouped in blocks

$$
\begin{bmatrix}
G_1^{(1)} & \cdots & G_N^{(1)} & G_{N+1}^{(1)} & \cdots & G_{2N}^{(1)} \\
\end{bmatrix}
= \begin{bmatrix}
G_1^{(0)} & \cdots & G_N^{(0)} & G_{N+1}^{(0)} & \cdots & G_{2N}^{(0)} \\
\end{bmatrix}
$$

(37)

Table 7 Analysis RMSE for different EnKF implementations applied to the $Q_{64 \times 64}$ instance

| $N$ | $m$ | $\sigma_B$ | SMEEnKF | EnKFC | EnKFSVD |
|-----|-----|------------|---------|-------|---------|
| 20  | 1,984 | 0.05      | $2.38 \times 10^{-4}$ | $2.38 \times 10^{-4}$ | $2.38 \times 10^{-4}$ |
|     |      | 0.10      | $4.77 \times 10^{-4}$ | $4.77 \times 10^{-4}$ | $4.77 \times 10^{-4}$ |
|     |      | 0.15      | $7.16 \times 10^{-4}$ | $7.16 \times 10^{-4}$ | $7.16 \times 10^{-4}$ |
| 2,778 |      | 0.05      | $2.38 \times 10^{-4}$ | $2.38 \times 10^{-4}$ | $2.38 \times 10^{-4}$ |
|     |      | 0.10      | $4.77 \times 10^{-4}$ | $4.77 \times 10^{-4}$ | $4.77 \times 10^{-4}$ |
|     |      | 0.15      | $7.15 \times 10^{-4}$ | $7.15 \times 10^{-4}$ | $7.15 \times 10^{-4}$ |
| 3,572 |      | 0.05      | $2.38 \times 10^{-4}$ | $2.38 \times 10^{-4}$ | $2.38 \times 10^{-4}$ |
|     |      | 0.10      | $4.77 \times 10^{-4}$ | $4.77 \times 10^{-4}$ | $4.77 \times 10^{-4}$ |
|     |      | 0.15      | $7.15 \times 10^{-4}$ | $7.15 \times 10^{-4}$ | $7.15 \times 10^{-4}$ |
| 60  | 1,984 | 0.05      | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ |
|     |      | 0.10      | $4.69 \times 10^{-4}$ | $4.69 \times 10^{-4}$ | $4.69 \times 10^{-4}$ |
|     |      | 0.15      | $7.04 \times 10^{-4}$ | $7.04 \times 10^{-4}$ | $7.04 \times 10^{-4}$ |
| 2,778 |      | 0.05      | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ |
|     |      | 0.10      | $4.68 \times 10^{-4}$ | $4.68 \times 10^{-4}$ | $4.68 \times 10^{-4}$ |
|     |      | 0.15      | $7.03 \times 10^{-4}$ | $7.03 \times 10^{-4}$ | $7.03 \times 10^{-4}$ |
| 3,572 |      | 0.05      | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ |
|     |      | 0.10      | $4.68 \times 10^{-4}$ | $4.68 \times 10^{-4}$ | $4.68 \times 10^{-4}$ |
|     |      | 0.15      | $7.03 \times 10^{-4}$ | $7.03 \times 10^{-4}$ | $7.03 \times 10^{-4}$ |
| 100 | 1,984 | 0.05      | $2.37 \times 10^{-4}$ | $2.37 \times 10^{-4}$ | $2.37 \times 10^{-4}$ |
|     |      | 0.10      | $4.74 \times 10^{-4}$ | $4.74 \times 10^{-4}$ | $4.74 \times 10^{-4}$ |
|     |      | 0.15      | $7.10 \times 10^{-4}$ | $7.10 \times 10^{-4}$ | $7.10 \times 10^{-4}$ |
| 2,778 |      | 0.05      | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ | $2.34 \times 10^{-4}$ |
|     |      | 0.10      | $4.68 \times 10^{-4}$ | $4.68 \times 10^{-4}$ | $4.68 \times 10^{-4}$ |
|     |      | 0.15      | $7.02 \times 10^{-4}$ | $7.02 \times 10^{-4}$ | $7.02 \times 10^{-4}$ |
| 3,572 |      | 0.05      | $2.32 \times 10^{-4}$ | $2.32 \times 10^{-4}$ | $2.32 \times 10^{-4}$ |
|     |      | 0.10      | $4.64 \times 10^{-4}$ | $4.64 \times 10^{-4}$ | $4.64 \times 10^{-4}$ |
|     |      | 0.15      | $6.97 \times 10^{-4}$ | $6.97 \times 10^{-4}$ | $6.97 \times 10^{-4}$ |

All methods give similar results

$B_1^{(1)} = \begin{bmatrix}
g_{(i-1)C_1}^{(1)} & \cdots & g_{iC_1}^{(1)} & \cdots & g_{(i+1)C_1}^{(1)} & \cdots & g_{(C_1+1)C_1}^{(1)} \\
\end{bmatrix} \in \mathbb{R}^{m \times C_1}$,

for $1 \leq i \leq N^{1}_{\text{proc}}$, and distributed over the processors:

$B_1^{(2)} = B_1^{(1)} - g_1^{(1)} \cdot \left(1 + v_1^T \cdot g_1^{(1)}\right)^{-1} \left(v_1^T \cdot B_1^{(1)}\right)$,

for all $1 \leq i \leq N^{1}_{\text{proc}}$. Note that $g_1^{(1)} (u_1^{(0)})$ is not updated since it is not required in subsequent computations. Thus, for the matrix $G^{(2)}$

$G^{(2)} = \begin{bmatrix}
g_1^{(2)} & g_2^{(2)} & \cdots & g_N^{(2)} \\
\end{bmatrix}$,

the next common computation is $g_2^{(2)} (u_2^{(1)})$, and for the same reasons, this vector is not updated.

In general, at time step $t$, $1 \leq t \leq N$, the first $t$ columns of the matrix $G^{(1)}$ are not included in the update process:

$G^{(t)} = \begin{bmatrix}
g_1^{(t)} & g_2^{(2)} & \cdots & g_{t-1}^{(t-1)} & g_t^{(t)} & g_{t+1}^{(t)} & \cdots & g_N^{(t)} \\
\end{bmatrix}$,
The parallel computation of (22) at time step \( t \) is performed as follows:

- Compute the number of computation units (columns of matrix \( G^{(t)} \in \mathbb{R}^{m \times N} \)) per processor:

\[
C_p^{(t)} = \frac{2 \cdot N - t}{N_{\text{proc}}}.
\]

- Perform the update in parallel over the blocks:

\[
B_i^{(t)} = B_i^{(t)} - g_i^{(t)} \cdot \left(1 + v_i^T \cdot g_i^{(t)} \right)^{-1} \cdot \left(v_i^T \cdot B_i^{(t)} \right),
\]

for all \( 1 \leq i \leq N_{\text{proc}} \), where

\[
B_i^{(t)} = \left[g_i^{(1)} \cdot g_i^{(1+2)} \cdot \ldots \cdot g_i^{(N_{\text{proc}})} \right].
\]

This parallel implementation of the iterative Sherman–Morrison formula leads to the complexity:

\[
T_{\text{PAR}}^{SMF} (m, N) = O \left(C_0^{(t)}, m \right) + \sum_{t=1}^{N} O \left(C_0^{(t)}, m \right).
\]

Notice, when the number of processors at time \( 0 \leq t \leq N \) is \( N_{\text{proc}} = 2N - t \) then \( C_p^{(t)} = 1 \). Hence, the corresponding computational cost of the analysis step is bounded by:

\[
O(m \cdot N + n \cdot N),
\]

therefore, when the number of observations is large enough relative to the number of ensemble members, this parallel approach of the iterative Sherman–Morrison formula exhibits a linear behavior, making this implementation attractive.

6 Conclusions

We discuss an implementation of the EnKF based on an iterative application of the Sherman–Morrison formula. The algorithm exploits the special structure of the background error covariance matrix projected onto the observation space. The computational complexity of the new approach is equivalent to that of the best EnKF formulations available in the literature. A sufficient condition for the stability of the proposed method is the non-singularity of the data error covariance matrix, which is typically the case in practice. In addition, a pivoting strategy is developed in order to reduce round-off error propagation without increasing the computational effort of the proposed method. The computational cost of this algorithm provides a better theoretical performance than other generic formulations of matrix inversion based on the Sherman–Morrison formula available in the literature. To assess the accuracy and performance of the proposed implementation a tests have been carried out using a quasi-geostrophic model. All EnKF implementations tested (EnKFC, EnKFSV, SMEnKF) provide virtually identical analyses. However, the proposed Sherman–Morrison approach scales better than the others with regard to the number of observations. The parallel version of the new algorithm has a theoretical complexity that grows only linearly with the number of observations, and is therefore well suited for implementation in large scale data assimilation systems.

Acknowledgments This work has been supported in part by NSF through awards NSF OCI-867094397, NSF CCF-0916493, NSF DMS-0915047, NSF CMII-1130667, NSF CCF-1218454, AFOSR FA9550-12-1-0293-DEF, AFOSR 12-2640-06, and by the Computational Science Laboratory at Virginia Tech.

References

Alves, O., Robert, C.: Tropical Pacific Ocean model error covariances from Monte Carlo simulations. Q. J. R. Meteorol. Soc. 131, 3643–3658 (2005)
Anderson, J.L.: An ensemble adjustment Kalman filter for data assimilation. Mon. Weather Rev. 129(12), 2884–2903 (2001)
Anderson, J.L.: Exploring the need for localization in ensemble data assimilation using a hierarchical ensemble filter. Physica D Nonlinear Phenomena 230(1–2), 99–111 (2007)
Anderson, J.L., Collins, N.: Scalable implementations of ensemble filter algorithms for data assimilation. J. Atmos. Ocean. Technol. 24(8), 1452–1463 (2007)
Angerson, E., Bai, Z., Dongarra, J., McKenney, A., Du Croz, J., Hammarling, S., Demmel, J., Bischof, C., Sorensen, D.: Lapack: A portable linear algebra library for high-performance computers. In: Proceedings of Supercomputing 1990. pp. 2–11 (1990). doi: 10.1109/SUPERC.1990.129995
Carton, X., Baraille, R.: Data assimilation in quasi-geostrophic ocean models. In: Proceedings of OCEANS 94. Oceans Engineering for Today’s Technology and Tomorrow’s Preservation. vol. 3, pp. III/337–III/346 (1994). doi:10.1109/OCEANS.1994.364221
Cohen, A.: Rate of convergence of several conjugate gradient algorithms. SIAM J. Numer. Anal. 9(2), 248–259 (1972)
Eisenstat, S.: Efficient implementation of a class of preconditioned conjugate gradient methods. SIAM J. Sci. Stat. Comput. 2(1), 1–4 (1981)
Evensen, G.: Chap. 14. Data Assimilation: The Ensemble Kalman Filter, 2nd edn. Springer, New York (2009)
Evensen, G.: Estimation in an oil reservoir simulator. Data Assimilation, pp. 263–272. Springer, Berlin (2009)
Evensen, G.: The ensemble Kalman filter for combined state and parameter estimation. Control Syst. IEEE 29(3), 83–104 (2009)
Fraser, A.M.: Hidden Markov models and dynamical systems. SIAM (2008). doi: 10.1137/1.9780898717747
Gill, P., Saunders, M., Shinnerl, J.: On the stability of Cholesky factorization for symmetric quasidfinite systems. SIAM J. Matrix Anal. Appl. 17(1), 35–46 (1996)
Godinez, H., Moulton, J.: An efficient matrix-free algorithm for the ensemble Kalman filter. Comput. Geosci. 16(3), 565–575 (2012). doi:10.1007/s10596-011-9268-9
Golub, G., OLeary, D.: Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. SIAM Rev. 31(1), 50–102 (1989)
Haugen, V., Naevdal, G., Natvik, L.J., Evensen, G., Berg, A., Flornes, K.: History matching using the ensemble Kalman filter on a north sea field case. SPE J. 13, 382–391 (2008)

Kovalenko, A., Mannseth, T., Naevdal, G.: Error estimate for the ensemble Kalman filter analysis step. SIAM J. Matrix Anal. Appl. 32(4), 1275–1287 (2011)

Li, H., KALNAY, E., Miyoshi, T.: Simultaneous estimation of covariance inflation and observation errors within an ensemble Kalman filter. Q. J. R. Meteorol. Soc. 135(639), 523–533 (2009). doi:10.1002/qj.371

Lin, C., More, J.: Incomplete Cholesky factorizations with limited memory. SIAM J. Sci. Comput. 21(1), 24–45 (1999). doi:10.1137/S1064827597327334

Mandel, J.: Efficient implementation of the ensemble Kalman filter. University of Colorado at Denver and Health Sciences Center, Technical Report (2006)

Maponi, P.: The solution of linear systems by using the Sherman-Morrison formula. Linear Algebra Appl. 420(2–3), 276–294 (2007). doi:10.1016/j.laa.2006.07.007

Meinguet, J.: Refined error analyses of Cholesky factorization. SIAM J. Numer. Anal. 20(6), 1243–1250 (1983)

Ott, E., Hunt, B.R., Szunyogh, I., Zimin, A.V., Kostelich, E.J., Corazza, M., Kalnay, E., Patil, D.J., Yorke, J.A.: A local ensemble Kalman filter for atmospheric data assimilation. Tellus A 56(5), 415–428 (2004)

Otto, K., Larsson, E.: Iterative solution of the Helmholtz equation by a second-order method. SIAM J. Matrix Anal. Appl. 21(1), 209–229 (1999). doi:10.1137/S0895479897316588

Reid, J.: The use of conjugate gradients for systems of linear equations possessing property A. SIAM J. Numer. Anal. 9(2), 325–332 (1972)

Schnabel, R., Eskow, E.: A new modified Cholesky factorization. SIAM J. Sci. Stat. Comput. 11(6), 1136–1158 (1990). doi:10.1137/0911064

Stewart, M., Van Dooren, P.: Stability issues in the factorization of structured matrices. SIAM J. Matrix Anal. Appl. 18(1), 104–118 (1997)

Suarez, A., Heather Dawn, R., Dustan, W., Coniglio, M.: Comparison of ensemble Kalman filter-based forecasts to traditional ensemble and deterministic forecasts for a case study of banded snow. Weather Forecast. 27, 85–105 (2012)

Tippett, M.K., Anderson, J.L., Bishop, C.H., Hamill, T.M., Whitaker, J.S.: Ensemble square root filters. Mon. Weather Rev. 131(7), 1485–1490 (2003)

Wu, G., Zheng, X., Li, Y.: Inflation adjustment on error covariance matrix of ensemble Kalman filter. In: International Conference on Multimedia Technology (ICMT), 2011. pp. 2160–2163 (2011)

Xia, J., Gu, M.: Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices. SIAM J. Matrix Anal. Appl. 31(5), 2899–2920 (2010). doi:10.1137/090750500