EnKF-C user guide

version 0.97

Pavel Sakov

June 19, 2014 – May 8, 2015

Contents

License

Introduction

EnKF-C aims to provide a light-weight generic framework for off-line data assimilation (DA) into large-scale layered geophysical models with the ensemble Kalman filter (EnKF). Here "light-weight" has higher priority than "generic"; that is, the code is not designed to cover every virtual possibility for the sake of it, but rather to be expandable in practical (from the author's point of view) situations. Following are its other main features:

- coded in C for GNU/Linux platform;
- can conduct DA either in EnKF or ensemble optimal interpolation (EnOI) mode;
- permits multiple model grids.

To make the code simpler, EnKF-C assumes that the model is layered and model states are stored in NetCDF format. EnKF-C is supposed to run with localisation only.

EnKF-C is available from `https://code.google.com/p/enkf-c`. This user guide is a part of the EnKF-C package. It is also available from `http://arxiv.org/abs/1410.1233`.

The user guide has two main sections. Section 1 overviews the basics of the EnKF; section 2 provides technical description of EnKF-C.

Chapter 1

EnKF

1.1 Kalman filter

The Kalman filter (KF) is the underlying concept behind the EnKF. It is rather simple if formulated as the recursive least squares.

Consider first a global (in time) nonlinear minimisation problem

$$\{\mathbf{x}_i^a\}_{i=1}^k = \arg\min_{\{\mathbf{x}_i\}_{i=1}^k} \mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_k), \tag{1.1}$$

$$\begin{aligned}\mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_k) = {} & (\mathbf{x}_1 - \mathbf{x}_1^f)^{\mathrm{T}} (\mathbf{P}_1^f)^{-1} (\mathbf{x}_1 - \mathbf{x}_1^f) \\ & + \sum_{i=1}^{k} \left[\mathbf{y}_i - \mathcal{H}_i(\mathbf{x}_i)\right]^{\mathrm{T}} (\mathbf{R}_i)^{-1} \left[\mathbf{y}_i - \mathcal{H}_i(\mathbf{x}_i)\right] \\ & + \sum_{i=2}^{k} \left[\mathbf{x}_i - \mathcal{M}_i(\mathbf{x}_{i-1})\right]^{\mathrm{T}} (\mathbf{Q}_i)^{-1} \left[\mathbf{x}_i - \mathcal{M}_i(\mathbf{x}_{i-1})\right]. \end{aligned} \tag{1.2}$$

Here $\{\mathbf{x}_i^a\}_{i=1}^k$ is a set of k state vectors that minimise the cost function (1.2); indices $i = 1, \ldots, k$ correspond to a sequence of DA cycles, so that $\mathbf{x}_1$ is the estimated model state at the first cycle and $\mathbf{x}_k$ is the estimated model state at the last cycle; $\mathbf{y}_i$ are observation vectors related to the model state by a nonlinear observation operator $\mathcal{H}_i(\mathbf{x})$; $\mathcal{M}_i(\mathbf{x})$ is a nonlinear model operator relating the model states $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$; $\mathbf{P}_1^f$ is the initial state error covariance; $\mathbf{R}_i$ is the observation error covariance for $\mathbf{y}_i$; and $\mathbf{Q}_i$ is the model error covariance for $\mathcal{M}_i$; $(\cdot)^{\mathrm{T}}$ denotes matrix transposition.

The minimisation problem (1.1, 1.2) is, generally, very complicated, but, luckily, has an exact solution in the *linear* case; moreover, this solution is recursive. Namely, assume that $\mathcal{M}$ and $\mathcal{H}$ are affine:

$$\mathcal{M}_i(\mathbf{x}^{(1)}) - \mathcal{M}_i(\mathbf{x}^{(2)}) = \mathbf{M}_i\,(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}), \tag{1.3a}$$

$$\mathcal{H}_i(\mathbf{x}^{(1)}) - \mathcal{H}_i(\mathbf{x}^{(2)}) = \mathbf{H}_i\,(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}), \tag{1.3b}$$

where $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$ are arbitrary model states, and $\mathbf{M}_i$, $\mathbf{H}_i$ = Const. Then the cost function (1.2)

becomes quadratic and can be factorised as follows:

$$\mathcal{L}_k(\mathbf{x}_1,\ldots,\mathbf{x}_k) = (\mathbf{x}_k - \mathbf{x}_k^f)^{\mathrm{T}}(\mathbf{P}_k^f)^{-1}(\mathbf{x}_k - \mathbf{x}_k^f) + \tilde{\mathcal{L}}_{k-1}(\mathbf{x}_1,\ldots,\mathbf{x}_{k-1}),$$

so that

$$\min_{\{\mathbf{x}_i\}_{i=1}^{k-1}} \mathcal{L}_k(\mathbf{x}_1,\ldots,\mathbf{x}_k) = (\mathbf{x}_k - \mathbf{x}_k^f)^{\mathrm{T}}(\mathbf{P}_k^f)^{-1}(\mathbf{x}_k - \mathbf{x}_k^f) + \text{Const.}$$

(*Proposition*) Then

$$\min_{\{\mathbf{x}_i\}_{i=1}^{k}} \mathcal{L}_{k+1}(\mathbf{x}_1,\ldots,\mathbf{x}_k,\mathbf{x}_{k+1}) = (\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^f)^{\mathrm{T}}(\mathbf{P}_{k+1}^f)^{-1}(\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^f) + \text{Const},$$

where

$$\mathbf{x}_{k+1}^f = \mathcal{M}_k(\mathbf{x}_k^a), \tag{1.4a}$$
$$\mathbf{P}_{k+1}^f = \mathbf{M}_k\mathbf{P}_k^a(\mathbf{M}_k)^{\mathrm{T}} + \mathbf{Q}_k, \tag{1.4b}$$

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \mathbf{K}_k\left[\mathbf{y}_k - \mathcal{H}(\mathbf{x}_k^f)\right], \tag{1.5a}$$
$$\mathbf{P}_k^a = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f, \tag{1.5b}$$

where

$$\mathbf{K}_k \equiv \mathbf{P}_k^f(\mathbf{H}_k)^{\mathrm{T}}\left[\mathbf{H}_k\mathbf{P}_k^f(\mathbf{H}_k)^{\mathrm{T}} + \mathbf{R}_k\right]^{-1}. \tag{1.5c}$$

This solution is known as the Kalman filter (KF, Kalman, 1960). Equations (1.4) describe advancing the system in time and represent the stage commonly called "forecast", while equations (1.5) describe assimilation of observations and represent the stage called "analysis". The superscripts f and a are used hereafter to refer to the forecast and analysis variables, correspondingly. The forecast and analysis model state estimates $\mathbf{x}^f$ and $\mathbf{x}^a$ are commonly called (simply) forecast and analysis. Matrix $\mathbf{K}$ is called Kalman gain.

There are a few things to be noted about the KF:

1. The state $\mathbf{X}$ of the DA system (SDAS) is represented by the estimated model state vector and model state error covariance:

   $$\mathbf{X} = \{\mathbf{x}, \mathbf{P}\}. \tag{1.6}$$

   This means that in any moment of time all previous information about the system is encrypted into the current (forecast or analysis) SDAS.

2. The KF provides solution for the *last* analysis, corresponding to $\mathbf{x}_k^a$ in (1.1) (or, with a minor re-formulation, to the last forecast); finding the full (global in time) solution requires application of the Kalman *smoother* (KS). Both the KF and KS can be derived by a re-factorisation of the positive definite quadratic form (1.2).

3. Because the SDAS represents a (part of a) solution of the global least squares problem, it does not depend on the order in which observations are assimilated or on their grouping.

4. Ditto, the SDAS does not depend on a linear non-singular transform of the model state in the sense that the forward and inverse transforms commute with the evolution of the DA system.

5. Solution (1.4, 1.5) can be *used* in a nonlinear case by approximating

$$\mathbf{M}_i \leftarrow \nabla \mathcal{M}_i(\mathbf{x}^f_{i-1}),$$
$$\mathbf{H}_i \leftarrow \nabla \mathcal{H}_i(\mathbf{x}^f_i),$$

in which case it is called the extended Kalman filter (EKF).

1.2 EnKF

The canonic form of the KF (1.4, 1.5) is not necessarily the most convenient or suitable one in practice. The corresponding algorithms can be prone to loosing the positive definiteness of the state error covariance $\mathbf{P}$ due to round-up errors; and more importantly, explicit use of $\mathbf{P}$ makes these algorithms non-scalable in regard to the model state dimension.

Both these immediate problems can be addressed with the ensemble Kalman filter, or the EnKF. In the EnKF the SDAS is carried by an ensemble of model states $\mathbf{E}$, which can be split into ensemble mean and ensemble anomalies:

$$\mathbf{X} = \{\mathbf{E}\} = \{\mathbf{x}, \mathbf{A}\}. \tag{1.7}$$

It is related to the SDAS of the KF (1.6) as follows:

$$\mathbf{x} = \frac{1}{m}\mathbf{E}\mathbf{1}, \tag{1.8a}$$
$$\mathbf{P} = \frac{1}{m-1}\mathbf{A}\mathbf{A}^{\mathrm{T}}, \tag{1.8b}$$
$$\mathbf{A} \equiv \mathbf{E} - \mathbf{x}\mathbf{1}^{\mathrm{T}}, \tag{1.8c}$$

where $\mathbf{1}$ is a vector with all elements equal to 1. The above means that the model state estimate is given by the ensemble mean, while the model state error covariance estimate is implicitly represented by the ensemble anomalies $\mathbf{A}$ via the factorisation (1.8b).

The EnKF is linearly scalable in regard to the state vector dimension, and in a number of ways is a more natural and extendable representation of the KF than the canonic form (1.4, 1.5). In particular, the forecast stage of the EnKF involves only propagation of each ensemble member:

$$\mathbf{E}^f_i = \mathcal{M}_i(\mathbf{E}^a_{i-1}). \tag{1.9}$$

This is a remarkably simple equation compared to the KF forecast equations (1.4), even though the model error still needs to be handled in some way. One option is to include stochastic model error into the model operator in (1.9). (This would make it different to the model operator in the KF.) Another option is to use the multiplicative inflation. The third option is to mimic the treatment of model error in the KF, although this brings back problems with scalability in regard to the state size.

At the analysis stage one has to update the ensemble mean and ensemble anomalies to match (1.5). This involves handling ensemble as a whole, which is different to the forecast stage, when each ensemble member is propagated individually.

Note that factorisation (1.8b) is not unique: if $\mathbf{A}$ satisfies (1.8b), then $\tilde{\mathbf{A}} = \mathbf{AU}$, where $\mathbf{U}$ is an arbitrary orthonormal matrix $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$, also satisfies (1.8b). However, $\tilde{\mathbf{A}}$ should not only factorise $\mathbf{P}$, but also remain an ensemble anomalies matrix, $\tilde{\mathbf{A}}\mathbf{1} = \mathbf{0}$. This requires an additional constraint $\mathbf{U1} = \mathbf{1}$. Summarising, if $\mathbf{E} = \mathbf{x1}^{\mathrm{T}} + \mathbf{A}$ is an ensemble that satisfies (1.8), then ensemble

$$\tilde{\mathbf{E}} = \mathbf{x1}^{\mathrm{T}} + \mathbf{AU}^p, \quad \mathbf{U}^p : \ \mathbf{U}^p(\mathbf{U}^p)^{\mathrm{T}} = \mathbf{I}, \ \mathbf{U}^p\mathbf{1} = \mathbf{1} \tag{1.10}$$

also satisfies (1.8). If $\mathbf{E}$ is full rank (i.e. $\mathrm{rank}(\mathbf{E}) = \min(m, n)$, where n is the state dimension), then each unique $\mathbf{U}^p$ generates a unique ensemble, and (1.10) describes all possible ensembles matching a given SDAS of the KF. Such transformation of the ensemble is called ensemble *redrawing*. In the linear case (i.e. for affine model and observation operators) redrawing of the ensemble in the EnKF does not affect evolution of the underlying KF; and conversely, in the nonlinear case the redrawing does indeed affect evolution of the underlying KF.

1.3 EnKF analysis

In this section we will give a brief overview of solutions for the EnKF analysis, and then describe the particular schemes used in EnKF-C.

1.3.1 Overview

In the "baseline" EnKF (full-rank ensemble, no localisation) the analysed SDAS matches that of the KF, although the algebraic side is indeed different. The update of the ensemble mean is generally straightforward, in accordance with that in the KF (1.5a). The details may depend on the chosen algorithm to achieve better numerical efficiency (see sec. 1.3.3).

The update of ensemble anomalies can be done either via a right-multiplied or left-multiplied (or post/pre-multiplied) transform of the ensemble anomalies:

$$\mathbf{A}^a = \mathbf{T}_L \, \mathbf{A}^f, \tag{1.11}$$

or

$$\mathbf{A}^a = \mathbf{A}^f \, \mathbf{T}_R. \tag{1.12}$$

$\mathbf{T}_L$ and $\mathbf{T}_R$ are referred to hereafter as left-multiplied and right-multiplied ensemble transform matrices (ETMs), respectively. Note that for a full rank ensemble $\mathbf{T}_R$ has to satisfy $\mathbf{T}_R\mathbf{1} = \mathbf{1}$. It follows from (1.10) that if $\mathbf{T}_R$ is a particular solution for the right-multiplied ETM, then (for a full rank ensemble) any other solution can be written as

$$\tilde{\mathbf{T}}_R = \mathbf{T}_R\mathbf{U}^p, \quad \mathbf{U}^p : \ \mathbf{U}^p(\mathbf{U}^p)^{\mathrm{T}} = \mathbf{I}, \ \mathbf{U}^p\mathbf{1} = \mathbf{1}. \tag{1.13}$$

Similarly, the analysis increment can be represented as a linear combination of the forecast ensemble anomalies:

$$\mathbf{x}^a = \mathbf{x}^f + \mathbf{A}^f \mathbf{w}. \tag{1.14}$$

Equations (1.12) and (1.14) can be combined into a single transform of the ensemble:

$$\mathbf{E}^a = \mathbf{E}^f \mathbf{X}_5, \tag{1.15}$$

$$\mathbf{X}_5 = \frac{1}{m}\mathbf{1}\mathbf{1}^{\mathrm{T}} + \left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^{\mathrm{T}}\right)\left(\mathbf{w}\mathbf{1}^{\mathrm{T}} + \mathbf{T}_R\right) = \frac{1}{m}\mathbf{1}\mathbf{1}^{\mathrm{T}} + \mathbf{w}\mathbf{1}^{\mathrm{T}} + \left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^{\mathrm{T}}\right)\mathbf{T}_R, \tag{1.16}$$

as $\mathbf{1}^{\mathrm{T}}\mathbf{w} = 0$. For a number of common schemes, including the ETKF and DEnKF, $\mathbf{1}^{\mathrm{T}}\mathbf{T}_R = \mathbf{1}^{\mathrm{T}}$, so that

$$\mathbf{X}_5 = \mathbf{w}\mathbf{1}^{\mathrm{T}} + \mathbf{T}_R. \tag{1.17}$$

The designation $\mathbf{X}_5$ is used for historic reasons, following Evensen (2003).

1.3.2 Some schemes

As follows from the previous section, there are multiple solutions for the ETM that match the KF covariance update equation (1.5b); however the particular solutions may have different properties in practice due to the DAS nonlinearity, their algorithmic convenience, or their robustness in suboptimal conditions. This section provides some background for the schemes used in EnKF-C:

- ETKF;
- DEnKF.

ETKF

It is easy to show using the definition of $\mathbf{K}$ (1.5c) and matrix shift lemma (1.19) that

$$(\mathbf{I} - \mathbf{K}\mathbf{H})\,\mathbf{P}^f = (\mathbf{I} - \mathbf{K}\mathbf{H})^{1/2}\,\mathbf{P}^f\,(\mathbf{I} - \mathbf{K}\mathbf{H})^{\mathrm{T}/2},$$

which yields the following solution for the left-multiplied ETM:

$$\mathbf{T}_L = (\mathbf{I} - \mathbf{K}\mathbf{H})^{1/2} \tag{1.18}$$

(Sakov and Oke, 2008b), that is

$$\mathbf{A}^a = (\mathbf{I} - \mathbf{K}\mathbf{H})^{1/2}\mathbf{A}^f. \tag{1.18a}$$

Hereafter by $\mathbf{X}^{1/2}$ we denote the unique positive definite square root from a positive definite (generally, non-symmetric) matrix $\mathbf{X}$, defined as $\mathbf{X}^{1/2} = \mathbf{V}\mathbf{L}^{1/2}\mathbf{V}^{-1}$, where $\mathbf{X} = \mathbf{V}\mathbf{L}\mathbf{V}^{-1}$ is the eigenvalue decomposition of $\mathbf{X}$. By "matrix shift lemma" we refer to the following identity:

$$\mathcal{F}(\mathbf{A}\mathbf{B})\,\mathbf{A} = \mathbf{A}\,\mathcal{F}(\mathbf{B}\mathbf{A}), \tag{1.19}$$

where $\mathcal{F}$ is an arbitrary function expandable into Taylor series. Rewriting (1.18a) as

$$\mathbf{A}^a = \left[\mathbf{I} - \frac{1}{m-1}\mathbf{A}^f(\mathbf{H}\mathbf{A}^f)^{\mathrm{T}}(\mathbf{H}\mathbf{P}^f\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\right]^{1/2}\mathbf{A}^f$$

and using the matrix shift lemma, we obtain:

$$\mathbf{A}^a = \mathbf{A}^f\left[\mathbf{I} - \frac{1}{m-1}(\mathbf{H}\mathbf{A}^f)^{\mathrm{T}}(\mathbf{H}\mathbf{P}^f\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{A}^f\right]^{1/2}$$

which yields the corresponding to (1.18) right-multiplied ETM:

$$\mathbf{T}_R = \left[\mathbf{I} - \frac{1}{m-1}(\mathbf{H}\mathbf{A}^f)^{\mathrm{T}}(\mathbf{H}\mathbf{P}^f\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{A}^f\right]^{1/2} \tag{1.20}$$

(Evensen, 2004). Applying the matrix inversion lemma

$$(\mathbf{A} + \mathbf{U}\mathbf{L}\mathbf{V})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{L}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}, \tag{1.21}$$

(1.18) can be transformed to:

$$\mathbf{T}_L = (\mathbf{I} + \mathbf{P}^f\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})^{-1/2} \tag{1.22}$$

(Sakov and Bertino, 2011); and applying the matrix shift lemma yields the corresponding right-multiplied ETM:

$$\mathbf{T}_R = \left[\mathbf{I} + \frac{1}{m-1}(\mathbf{H}\mathbf{A}^f)^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{A}^f\right]^{-1/2}, \tag{1.23}$$

also known as the ensemble transform Kalman filter, or ETKF (Bishop et al., 2001).

Historic reference. Another (and probably the first) solution for $\mathbf{T}_R$ equivalent to (1.20) and (1.23) was found by Andrews (1968):

$$\mathbf{T}_R = \mathbf{I} - \frac{1}{m-1}(\mathbf{H}\mathbf{A}^f)^{\mathrm{T}}\mathbf{M}^{-1/2}\left(\mathbf{M}^{1/2} + \mathbf{R}^{1/2}\right)^{-1}\mathbf{H}\mathbf{A}^f, \tag{1.24}$$

where $\mathbf{M} \equiv \mathbf{H}\mathbf{P}^f\mathbf{H}^{\mathrm{T}} + \mathbf{R}$.

Equations (1.20, 1.23, 1.24) yield algebraically different expressions for the (unique) symmetric right-multiplied solution. Apart from being the only symmetric solution, it also represents the minimum distance solution for the ensemble anomalies: its ensemble of analysed anomalies is closer to the ensemble of forecast anomalies with the inverse forecast (or analysis) covariance as the metric than any other ensemble of analysed anomalies given by (1.13) (Ott et al., 2003, rev. 2005). This means that in the above sense the symmetric right-multiplied solution preserves the identities of ensemble members during analysis in the best possible way.

Note that while the left-multiplied solutions (1.18, 1.22) correspond to the symmetric right-multiplied solution, they are not symmetric.

In a typical DAS with a large scale model one can expect $m = 100$, $p = 10^3 - 10^7$, $n = 10^6 - 10^9$; that is

$$m \ll p \ll n. \tag{1.25}$$

Therefore, considering the size of ETMs ($n \times n$ for left-multiplied ETMs and $m \times m$ for right-multiplied ETMs), only right-multiplied solutions are suitable for use with large scale models. The ETKF solution (1.23) represents the most popular option due to its simple form and numerical effectiveness: for a diagonal $\mathbf{R}$, it only requires to calculate inverse square root of a symmetric $m \times m$ matrix. Also, along with the left-multiplied solution (1.22), it generally has better numerical properties than solutions (1.18) and (1.20) due to the fact that the inverse square root in it is calculated from the sum of a positive definite and a positive semi-definite matrices.

DEnKF

Assuming that $\mathbf{KH}$ is small in some sense, one can approximate solution (1.18) by expanding it into Taylor series about $\mathbf{I}$ and keeping the first two terms of the expansion:

$$\mathbf{T}_L = \mathbf{I} - \frac{1}{2}\mathbf{KH}. \tag{1.26}$$

This approximation is known as the deterministic Kalman filter, or DEnKF (Sakov and Oke, 2008a). It has a simple interpretation of using half of the Kalman gain for updating the ensemble anomalies; but apart from that the DEnKF often represents a good practical choice due to its algorithmic convenience and good performance in suboptimal situations. The DEnKF is the default scheme in EnKF-C.

1.3.3 Some numerical considerations

Instead of using the forecast ensemble observation anomalies $\mathbf{HA}^f$ and innovation $\mathbf{y} - \mathcal{H}(\mathbf{x}^f)$ it is convenient to use their standardised versions:

$$\mathbf{s} = \mathbf{R}^{-1/2}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}^f)\right]/\sqrt{m-1}, \tag{1.27}$$

$$\mathbf{S} = \mathbf{R}^{-1/2}\mathbf{HA}^f/\sqrt{m-1}. \tag{1.28}$$

Then

$$\mathbf{w} = \mathbf{G}\mathbf{s}; \tag{1.29}$$

for the ETKF

$$\mathbf{T}_R = (\mathbf{I} + \mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1/2}, \tag{1.30}$$

and for the DEnKF

$$\mathbf{T}_R = \mathbf{I} - \frac{1}{2}\mathbf{GS}, \tag{1.31}$$

where

$$\mathbf{G} \equiv (\mathbf{I} + \mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathrm{T}}, \tag{1.32}$$

$$= \mathbf{S}^{\mathrm{T}}(\mathbf{I} + \mathbf{S}\mathbf{S}^{\mathrm{T}})^{-1}. \tag{1.33}$$

Here (1.32) involves inversion of an $m \times m$ matrix, while (1.33) involves inversion of a $p \times p$ matrix. Therefore, in the DEnKF it is possible to calculate $\mathbf{w}$ and $\mathbf{T}$ using a single inversion of either a $p \times p$ or $m \times m$ matrix, depending on the relation between the number of observations and the ensemble size. In contrast, the ETKF (1.30) requires calculation of the inverse square root of an $m \times m$ matrix. Then, one can use expression (1.32) for $\mathbf{G}$ and calculate both inversion in it and inverse square root in (1.30) from the same singular value decomposition (SVD). This makes the DEnKF somewhat more numerically effective because, firstly, one can exploit situations when $p < m$ to invert a matrix of lower dimension and, secondly, it requires only matrix inversion, which can be done via Cholesky decomposition instead of SVD.

1.4 Localisation

Localisation is a necessary attribute of the EnKF systems with large-scale models, aimed at overcoming the rank deficiency of the ensemble. It can also be seen as aimed at reducing spurious long range correlations occurring due to the finite size of the ensemble; or at limiting the impact of distant observations because of the unreliability of the corresponding covariances.

There are two common localisation methods for the EnKF – covariance localisation (CL, Hamill and Whitaker, 2001; Houtekamer and Mitchell, 2001), also known as covariance filtering, and local analysis (LA, Evensen, 2003; Ott et al., 2003, rev. 2005). Although CL may have advantages in certain situations (non-local observations, "strong" assimilation), in practice the two methods produce similar results (Sakov and Bertino, 2011). For algorithmic reasons EnKF-C uses LA.

Instead of calculating the global ensemble transform $\mathbf{X}_5$, LA involves calculating local ensemble transforms $\overset{i}{\mathbf{X}}_5$ for each element i of the state vector. This is done using local normalised ensemble observation anomalies $\overset{i}{\mathbf{S}}$ and local normalised innovation $\overset{i}{\mathbf{s}}$, obtained by tapering global $\mathbf{S}$ and $\mathbf{s}$:

$$\overset{i}{\mathbf{s}} \equiv \mathbf{s} \circ \overset{i}{\mathbf{f}}, \tag{1.34a}$$

$$\overset{i}{\mathbf{S}} \equiv \mathbf{S} \circ (\overset{i}{\mathbf{f}}\,\mathbf{1}^{\mathrm{T}}), \tag{1.34b}$$

where $\mathbf{A} \circ \mathbf{B}$ denotes by-element, or Hadamard, or Schur product of matrices $\mathbf{A}$ and $\mathbf{B}$. We consider non-adaptive localisation only, when the vector of taper coefficients $\mathbf{f}$ does not depend on the SDAS or observations. Typically, the taper coefficient for observation o is a function of locations of the state element i (denoted as $\overset{i}{\mathbf{r}}$) and observation o (denoted as $\overset{\{o\}}{\mathbf{r}}$): $\overset{i}{\mathbf{f}}_o = g(\overset{i}{\mathbf{r}}, \overset{\{o\}}{\mathbf{r}})$, where g is the taper function. In layered geophysical models g is often assumed to depend only on horizontal distance between these locations:

$$\overset{i}{\mathbf{f}}_o = g(|\overset{i}{\boldsymbol{\rho}} - \overset{\{o\}}{\boldsymbol{\rho}}|), \tag{1.35}$$

or on combination of horizontal and vertical distances, e.g.: $\overset{i}{f}_o = g_{xy}(|\overset{i}{\boldsymbol{\rho}} - \overset{\{o\}}{\boldsymbol{\rho}}|) g_z(|\overset{i}{z} - \overset{\{o\}}{z}|)$, where $\mathbf{r} = (\boldsymbol{\rho}, z)$, and $\boldsymbol{\rho} = (x, y)$. In the case (1.35) for a given set of observations the local ensemble transform $\overset{i}{\mathbf{X}}_5$ depends only on horizontal grid coordinates of the state element $\mathbf{x}_i$ and can be used for updating all state elements with the same horizontal grid coordinates. It is currently the only option in EnKF-C.

Smooth taper functions have advantage over non-smooth functions (such as the boxcar, or step

function) because they maintain the spatial continuity of the analysis. EnKF-C uses the popular polynomial taper function by Gaspari and Cohn (1999), which has a number of nice properties.

1.5 Asynchronous DA

Observations assimilated at each cycle in the KF are assumed to be made simultaneously at the time of assimilation. In such cases observations and DA method are referred to as *synchronous*. In reality, observations assimilated at a given cycle are made over some period of time called "data assimilation window" (DAW). If the DA method accounts for the time of observations, observations and DA method are referred to as *asynchronous*.

The EnKF can be naturally extended for asynchronous DA. Let us consider the minimisation problem (1.1, 1.2) in the case of perfect model $\mathbf{Q} = 0$. It becomes

$$\mathbf{x}_1^a = \arg\min\ \mathcal{L}(\mathbf{x}_1), \tag{1.36}$$

$$\mathcal{L}(\mathbf{x}_1) = (\mathbf{x}_1 - \mathbf{x}_1^f)^{\mathrm{T}}(\mathbf{P}_1^f)^{-1}(\mathbf{x}_1 - \mathbf{x}_1^f) + \sum_{i=1}^{k} \left[\mathbf{y}_i - \mathcal{H}_i(\mathbf{x}_i)\right]^{\mathrm{T}} (\mathbf{R}_i)^{-1} \left[\mathbf{y}_i - \mathcal{H}_i(\mathbf{x}_i)\right], \tag{1.37}$$

$$\mathbf{x}_{i+1} = \mathcal{M}(\mathbf{x}_i), \quad i = 1, \ldots, k-1. \tag{1.38}$$

Compared to the original problem, the dimensionality of the solution is much reduced due to relations (1.38), which mean that the model state at any time can be found by propagating the initial state: $\mathbf{x}_2 = \mathcal{M}_2(\mathbf{x}_1),\ \mathbf{x}_3 = \mathcal{M}_3 \circ \mathcal{M}_2(\mathbf{x}_1), \ldots$. The cost function (1.37) can then be written as:

$$\mathcal{L}(\mathbf{x}_1) = (\mathbf{x}_1 - \mathbf{x}_1^f)^{\mathrm{T}}(\mathbf{P}^f)^{-1}(\mathbf{x}_1 - \mathbf{x}_1^f) + \left[\mathbf{y} - \mathcal{H} \circ \mathcal{M}(\mathbf{x}_1)\right]^{\mathrm{T}} \mathbf{R}^{-1} \left[\mathbf{y} - \mathcal{H} \circ \mathcal{M}(\mathbf{x}_1)\right], \tag{1.39}$$

where observations $\mathbf{y}$ represent the augmented observation vector: $\mathbf{y} = [\mathbf{y}_1^{\mathrm{T}}, \ldots, \mathbf{y}_k^{\mathrm{T}}]^{\mathrm{T}}$, $\mathbf{R}$ is the corresponding observation error covariance, and operator $\mathcal{H} \circ \mathcal{M}(\mathbf{x}_1)$ is assumed to project the initial state $\mathbf{x}_1$ to observation space. Note that usually only $\mathcal{H}$ is a function that depends on assimilated observations; now $\mathcal{M}$ also depends on observations, propagating the initial state to the time of each observation. It is also possible to interpret $\mathcal{M}(\mathbf{x})$ as the *trajectory* starting from $\mathbf{x}$, while $\mathcal{H}$ maps it to observations.

Apart from the operator $\mathcal{H} \circ \mathcal{M}(\mathbf{x}_1)$, the cost function (1.39) has the same form as that for a single DA cycle with synchronous observations. Consequently, in the linear case (1.3) one can use solutions for $\mathbf{w}$ and $\mathbf{T}$ from section 1.3.3, subject to extending definitions of $\mathbf{s}$ and $\mathbf{S}$ as follows:

$$\mathbf{s} = \mathbf{R}^{-1/2} \left[\mathbf{y} - \mathcal{H} \circ \mathcal{M}(\mathbf{x}_1^f)\right] / \sqrt{m-1}, \tag{1.40}$$

$$\mathbf{S} = \mathbf{R}^{-1/2}\, \mathbf{H} \circ \mathbf{M}\, \mathbf{A}_1^f / \sqrt{m-1}, \tag{1.41}$$

where $\mathbf{H} \circ \mathbf{M}$ is the tangent linear operator of $\mathcal{H} \circ \mathcal{M}$ about $\mathbf{x}_1^f$. This means that to account for the time of observations in the EnKF one simply needs calculate innovation and forecast ensemble observation anomalies using ensemble at the time of each observation. There are no specific restrictions on $\mathbf{R}$, so that in theory observation errors can be correlated in time.

The minimisation problem (1.36),(1.39) implies assimilation time $t = t_1$; however, in the linear case the standardised innovation and ensemble observation anomalies in form (1.40,1.41) represent

objects invariant to assimilation time: the reference to $\mathbf{x}_1$ is only present to define the operator $\mathcal{H} \circ \mathcal{M}$. Consequently, the ensemble transform $\mathbf{X}_5$ calculated from $\mathbf{s}$ and $\mathbf{S}$ can be applied to ensemble at any particular time to yield (the same) analysed trajectories for the ensemble members: $\mathcal{M}(\mathbf{E})\,\mathbf{X}_5 = \mathcal{M}(\mathbf{E}\,\mathbf{X}_5)$. This time invariance of ensemble transforms can be used to update ensemble back in time using observations from future cycles without the need in backward model (Evensen and van Leeuwen, 2000).

Note. The background term $(\mathbf{x}_1 - \mathbf{x}_1^f)^{\mathrm{T}}(\mathbf{P}^f)^{-1}(\mathbf{x}_1 - \mathbf{x}_1^f)$ in (1.39) can be seen as accumulating the previous history of the system rather than characterising the initial uncertainty in the global problem. In this case it is natural to anchor it to the previous analysis:

$$\mathcal{L}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^f)^{\mathrm{T}}(\mathbf{P}^f)^{-1}(\mathbf{x} - \mathbf{x}^f) + [\mathbf{y} - \mathcal{H} \circ \mathcal{M}(\mathbf{x})]^{\mathrm{T}}\,\mathbf{R}^{-1}\,[\mathbf{y} - \mathcal{H} \circ \mathcal{M}(\mathbf{x})]\,. \tag{1.42}$$

Here $\mathbf{x}^f$ is the analysed state at the end of the previous cycle, considered as the forecast for the current cycle, and $\mathbf{P}^f$ is the corresponding state error covariance. Minimising $\mathcal{L}$ yields the analysed initial model state $\mathbf{x}^a$, which in turn yields the analysed trajectory; the analysed state error covariance is defined so that the analysed background term absorbs the observation term:

$$\mathbf{x}^a = \arg\min\ \mathcal{L}(\mathbf{x}),$$

$$\mathbf{P}^a :\ \mathcal{L}(\mathbf{x}^a) = (\mathbf{x}^a - \mathbf{x}^f)^{\mathrm{T}}(\mathbf{P}^a)^{-1}(\mathbf{x}^a - \mathbf{x}^f) + \mathrm{Const.}$$

This framework is a natural extension of the problem (1.1) in the linear, perfect-model case to continuous time; and is convenient for iterative minimisation.

1.6 EnOI

The EnOI, or ensemble optimal interpolation (Evensen, 2003), can be defined as the EnKF with a static or, more generally, pre-defined, ensemble. It can be summarised as follows:

$$\mathbf{x}_i^b = \mathcal{M}_i(\mathbf{x}_{i-1}^a), \tag{1.43}$$

$$\mathbf{x}_i^a = \mathbf{x}_i^b + \mathbf{A}^b\,\mathbf{w}_i, \tag{1.44}$$

where $\mathbf{x}^b$ is the forecast model state estimate referred to as background, and $\mathbf{A}^b$ is an ensemble of static, or background, anomalies; the corresponding state error covariance $\mathbf{P}^b$ is also often referred to as the background covariance.

The main incentive for using the EnOI is its low computational cost due to the integration of only one instance of the model. Despite of the similarity with the EnKF, the EnOI is a rather different concept, as there is no global in time cost function associated with it. Conceptually the EnOI is closer to 3D-Var, as both methods use static (anisotropic, multivariate) covariance. It is an improvement on the optimal interpolation, which typically uses isotropic, homogeneous and univariate covariance.

In contrast to the EnKF, due to the use of a static ensemble the EnOI avoids potential problems related to the ensemble spread; but at the same time it does critically depend on the ensemble, while the EnKF with a stochastic model typically "forgets" the initial ensemble over time.

The EnOI can account for the time of observations by calculating innovation using forecast at observation time, as in (1.40). This approach is commonly known as "first guess at appropriate time", or FGAT.

Chapter 2

EnKF-C

2.1 Overview

EnKF-C conducts data assimilation in three stages: *prep*, *calc* and *update*.

prep preprocesses observations so that they are ready for DA. It has the following work flow:

- read original observations and fill each observation into a uniform structure *measurement*;
- write observations to file `observations-orig.nc`;
- combine observations into superobservations;
- write super observations to `observations.nc`.

calc calculates ensemble transforms for updating the forecast ensemble of model states (EnKF) or the background model state (EnOI) in the following stages:

- read observations from `observations.nc`;
- calculate ensemble of forecast observations $\mathbf{HE}^f$ (EnKF) or ensemble of forecast observation anomalies $\mathbf{HA}^f$ and background observation estimates $\mathbf{Hx}^f$ (EnOI);
- for each horizontal grid cell calculate local ensemble transforms $\mathbf{X}_5$ (EnKF) or background update coefficients $\mathbf{w}$ (EnOI);
- save these transforms to `X5.nc` (EnKF) or `w.nc` (EnOI);
- calculate and report forecast and analysis innovation statistics.

update updates the ensemble (EnKF) or the background (EnOI) by using transforms calculated by *calc*.

2.2 Starting up: example 1

It may be a good idea to start getting familiar with the system by running the example in `examples/1`. The example has been put up based on the runs of regional EnKF and EnOI re-

analysis systems for Tasman Sea developed by Bureau of Meteorology. It allows one to conduct a single assimilation for 23 December 2007 (day 6565 since 1 January 1990) with either EnKF or EnOI. To reduce the size of the system, the model state has been stripped down to two vertical levels and 100×100 horizontal grid. Due to its size (almost 80 MB) the data for this example is available for download separately from the EnKF-C code – see `examples/1/README` for details.

2.3 Parameter files

EnKF-C requires 5 parameter files to run:

- main parameter file;
- model parameter file;
- grid parameter file;
- observation types parameter file;
- and observation metadata file.

Examples of these parameter files can be found in `examples/1`. Running EnKF-C binaries with `--describe-prm-format` in the command line provides information on the parameter file formats.

2.3.1 Main parameter file

The main parameter file specifies the main parameters of DA and 4 other parameter files. Its format is described by running `enkf_prep`, `enlf_calc` or `enk_update` with option `--describe-prm-format`:

```
>./bin/enkf_prep --describe-prm-format

  Main parameter file format:

    MODE            = { ENKF | ENOI }
  [ SCHEME          = { DENKF* | ETKF | EnKF-N } ]
  [ TARGET          = { ANALYSIS* | INCREMENT } ]
    MODEL           = <model prm file>
    GRID            = <grid prm file>
    OBSTYPES        = <obs. types prm file>
    OBS             = <obs. data prm file>
    DATE            = <julian day of analysis>
    ENSDIR          = <ensemble directory>
    BGDIR           = <background directory>                (MODE = ENOI)
  [ KFACTOR         = <kfactor> ]                           (1*)
  [ RFACTOR         = <rfactor> ]                           (1*)
    ...
    LOCRAD          = <locrad>
  [ STRIDE          = <stride> ]                            (1*)
  [ SOBSTRIDE       = <stride> ]                            (1*)
  [ FIELDBUFFERSIZE = <fieldbuffersize> ]                   (1*)
  [ INFLATION       = <inflation> [ SPREADLIMITED* | PLAIN ] (1*)
    ...
  [ REGION          = <name> { <lon1> <lon2> <lat1> <lat2> } ]
```

```
  ...
[ POINTLOG        { <i> <j> } ]
  ...
[ EXITACTION      = { BACKTRACE* | SEGFAULT } ]
[ BADBATCHES      = <obstype> <max. bias> <max. mad> <min # obs.> ]
  ...

Notes:
  1. { ... | ... | ... } denotes the list of possible choices
  2. [ ... ] denotes an optional input
  3. ( ... ) is a note
  4. * denotes the default value
  5. < ... > denotes a description of an entry
  6. ... denotes repeating the previous item an arbitrary number of times
```

2.3.2 Model parameter file

The model parameter file mainly describes the composition of the state vector by listing the model variables and specifying the associated grids.

```
>./bin/enkf_prep --describe-prm-format model

  Model parameter file format:

    NAME        = <name>

    VAR         = <name>
    GRID        = <name>
  [ INFLATION   = <value> ]

  [ <more of the above blocks> ]
```

Each model variable is described in a block started by the entry for the variable name. The inflation magnitude for a variable, if specified, overrides the common value set in the main parameter file.

EnKF-C permits using multiple model grids. In case of using multiple grids each model variable must be associated with one of the grids defined in the grid parameter file.

2.3.3 Grid parameter file

Grid parameter file describes grids used for model variables. Each grid is described in a section started by the grid name entry and contains the grid name, grid data file, and names of the dimensions and coordinates in the grid data file. It also contains variable names for the depth and for number of layers in a vertical column (z grids) or land mask (sigma grids):

```
>./bin/enkf_prep --describe-prm-format grid

  Grid parameter file format for z-model:
```

```
    NAME             = <name>
    VTYPE            = { z | sigma }
    DATA             = <data file name>
    XDIMNAME         = <x dimension name>
    YDIMNAME         = <y dimension name>
    ZDIMNAME         = <z dimension name>
    XVARNAME         = <x variable name>
    YVARNAME         = <y variable name>
    ZVARNAME         = <z variable name>
    DEPTHVARNAME     = <depth variable name>
    NUMLEVELSVARNAME = <# of levels variable name> (z)
    MASKVARNAME      = <land mask variable name> (sigma)

  [ <more of the above blocks> ]
```

The code is supposed to identify the type of the horizontal grid used, while the type of the vertical grid has to be specified explicitly.

At the moment, EnKF-C supports 3 possible types of horizontal grids:

- equidistant rectangular grids aligned with physical coordinates;
- non-equidistant rectangular grids aligned with physical coordinates;
- quadrilateral simply connected grids (via libgu).

For rectangular grids the code tries to determine and handle periodicity in X or Y directions.

The split between the main, model and grid parameter files is not final and may change in future.

2.3.4 Observation types parameter file

Observation types are the interface that connects model and observations. They are specified in a separate parameter file. Each observation type is described in a separate section identified by the entry `NAME`. Apart from the type name, the section must contain the associated model variable; whether observations of this type are surface or volume; the associated observation operator; the allowed range. The optional parameters include the R-factor for the type (sec. 2.8) and spatial limits on corresponding observations. For example:

```
>./bin/enkf_prep --describe-prm-format obstypes

  Observation types parameter file format:

    NAME      = <name>
    VAR       = <model variable name>
  [ VAR2      = <model variable name> ]
    ISSURFACE = { yes | no }
  [ OFFSET    = <file name> <variable name> ]    (none*)
    HFUNCTION = <H function name>
  [ ASYNC     = <time interval> ]                (synchronous*)
  [ RFACTOR   = <rfactor> ]                      (1*)
  [ MINVALUE  = <minimal allowed value> ]        (-inf*)
```

```
  [ MAXVALUE  = <maximal allowed value> ]        (+inf*)
  [ XMIN      = <minimal allowed X coordinate> ] (-inf*)
  [ XMAX      = <maximal allowed X coordinate> ] (+inf*)
  [ YMIN      = <minimal allowed Y coordinate> ] (-inf*)
  [ YMAX      = <maximal allowed Y coordinate> ] (+inf*)
  [ ZMIN      = <minimal allowed Z coordinate> ] (-inf*)
  [ ZMAX      = <maximal allowed Z coordinate> ] (+inf*)

  [ <more of the above blocks> ]
```

The `OFFSET` entry may be used for adding the known model bias to observations, for example, to specify the mean dynamic topography (MDT) when assimilating sea level anomaly (SLA) observations.

2.3.5 Observation metadata file

Observation metadata file specifies observations to be assimilated. EnKF-C has a simple policy in this regard: if a file is listed in the observation data file, then observations from this file are assimilated. This allows one using custom observation windows for particular observation types, instruments etc., specifying details on the script level.

The observation parameter file contains an arbitrary number of sections identified by entries `PRODUCT`. Each section specifies the observation type, input files, reader and, possibly, observation error:

```
./bin/enkf_prep --describe-prm-format obsmeta

  Observation meta data file format:

    PRODUCT   = <product>
    READER    = <reader>
    TYPE      = <observation type>
    FILE      = <data file wildcard>
    ...
  [ ERROR_STD = { <value> | <data file> } [ EQ* | PL | MU | MI | MA ] ]
    ...

  [ <more of the above blocks> ]
```

Observation files can be defined using wildcards "*" and "?". Missing a file is reported in the log and is not considered to be a fatal error. The available readers are listed by the variable `allreaders` defined in `prep/allreaders.c`.

The last line in the example above specifies the observation error. It can contain either a number or a file name. In the case of entering the file name there also should be another entry in the same line specifying the name of the variable to be read. The variable should have the same dimension (2D or 3D) as the associated observation type as described by the variable `ISSURFACE` in the observation types parameter file (sec. 2.3.4).

The line with observation error can also have another token specifying the type of operation to be conducted: `EQUAL` ($\sigma_{tot} \leftarrow \sigma_{now}$, default), `PLUS` ($\sigma_{tot} \leftarrow \sqrt{\sigma_{tot}^2 + \sigma_{now}^2}$), `MULT` ($\sigma_{tot} \leftarrow \sigma_{tot}\sigma_{now}$), `MIN` ($\sigma_{tot} = \max(\sigma_{tot}, \sigma_{now})$), or `MAX` ($\sigma_{tot} = \min(\sigma_{tot}, \sigma_{now})$). There can be several error entries in a section in the observation parameter file.

The observation time only matters if the observation type is specified to be "asynchronous" (see sec. 2.5.2). In this case the model estimation for the observation is made by using model state at the appropriate time. Otherwise, observations are assumed to be made at the time of assimilation, regardless of the actual observation time.

Note that there can be multiple blocks with the same product. This enables custom treatment of some specific data. For example, the following entries override observation error for Geosat (files with prefix `g1_`) on 23 May 2006:

```
# set observation error for Geosat to 7cm
product == RADS
type = SLA
reader = standard2
file=/short/p93/pxs599/obs/RADS-IB/y2006/m05/g?_d23.nc
error_std = 0.07

# use default errors for other altimeters
product == RADS
type = SLA
reader = standard2
file=/short/p93/pxs599/obs/RADS-IB/y2006/m05/[!g]?_d23.nc
```

2.4 File name conventions

EnKF-C assumes that the ensemble and background file names have some predefined formats. The file name for member `mid` and model variable `varname` is assumed to be `sprintf("mem%03d_%s.nc", mid, varname)`. The background file (EnOI only) for variable `varname` is assumed to be `sprintf("bg_%s.nc", varname)`. The above names are used for reading forecast states for synchronous DA and for writing analyses, in the case if the analyses are appended to forecasts (true by default). For asynchronous DA the member and background file names for the time slot `t` are assumed to be `sprintf("mem%03d_%s_%d.nc", mid, varname, t)` and `sprintf("bg_%s_%d.nc", varname, t)`, correspondingly.

2.5 *prep*

prep is the first stage of data assimilation in EnKF-C. Its preprocesses observations by bringing them to a common form and merging close observations into so called superobservations.

By design, *prep* is supposed to be light-weight, so that it does not read either the ensemble or background, and the only model information it needs is the model grid. (Note that this may require some additional processing at later stages for models with dynamic grid, such as HYCOM.)

The name of the binary (executable) for *prep* is `enkf_prep`. It has the following usage and options:

```
>./bin/enkf_prep
  Usage: enkf_prep <prm file> [<options>]
  Options:
  --describe-prm-format [main|model|grid|obstypes|obsmeta]
      describe format of the parameter file and exit
  --describe-superob <sob #>
      print composition of this superobservation and exit
  --log-all-obs
      put all obs into observations.nc (default: obs within model domain only)
  --no-superobing
  --version
      print version and exit
```

`enkf_prep` writes the preprocessed observations into file `observatons.nc`. It also writes file `observatons-orig.nc`, which contains original (not superobed) observations. By default, only observations within the model grid are written to it, but the the option `--log-all-obs` changes this behaviour to writing all observations from the input files.

2.5.1 Observation types, products, instruments, batches

Types

Each observation has a number of attributes defined by the fields of the structure `observation`. One of them is observation type, which characterises the observation in a general way and relates it to the model state. For example, typical oceanographic observations may have tags SLA (for sea level anomalies), SST (sea surface temperature), TEM (subsurface temperature) and SAL (subsurface salinity). Different types can be related to the same model variable, as do SST and TEM in the above example. Observation types are described in the corresponding parameter file (sec. 2.3.4).

Products

An observation is also characterised by "product". It can be a tag for an organisation that provides data from certain observational platforms, e.g.:

```
PRODUCT == RADS
TYPE = SLA
READER = standard2
FILE = obs/RADS-IB/y2007/m12/??_d19.nc
FILE = obs/RADS-IB/y2007/m12/??_d20.nc
FILE = obs/RADS-IB/y2007/m12/??_d21.nc
FILE = obs/RADS-IB/y2007/m12/??_d22.nc
FILE = obs/RADS-IB/y2007/m12/??_d23.nc

PRODUCT == NAVO
TYPE = SST
READER = standard
```

```
FILE = obs/NAVO/navo_20071219.nc
FILE = obs/NAVO/navo_20071220.nc
FILE = obs/NAVO/navo_20071221.nc
FILE = obs/NAVO/navo_20071222.nc
FILE = obs/NAVO/navo_20071223.nc
```

Instruments

The observational data from a product can be collected by a number of instruments. The corresponding field in the `measurement` structure is supposed to be filled by the observation reader.

Batches

An observation can be attributed to one of the groups called "batches", such as altimeter passes, Argo profiles etc., to enable detection and discarding of bad batches. Programmatically, to switch on capabilities associated with observation batches for a particular kind of observations, the observation batch ID needs to be set by the corresponding observation reader.

A batch of observations is considered bad if either the magnitude of its innovation bias or the mean magnitude of innovation exceed specified thresholds. Specifications for bad batches can be set in the parameter file as follows:

```
BADBATCHES = SLA 0.05 0.12 200
BADBATCHES = TEM 5 5 1
```

The above entry means that any batch of observations of type SLA containing more than 200 observations and having either mean innovation greater than 0.05 (meter) in magnitude or mean absolute innovation greater than 0.12 is considered to be bad. Similarly, a TEM batch is considered bad if either the mean absolute innovation or mean innovation magnitude in the batch exceed 5 (degrees). The parameter file can have arbitrary number of such entries. Information about bad batches is written by `enkf_calc` to the file `badbatches.out`. When `enkf_prep` detects the presence of such file, it marks the corresponding observations as bad.

Therefore, the workflow for detecting and eliminating bad batches of observations is as follows:

1. specify bad batches in the parameter file;
2. make a pilot run of `enkf_prep`;
3. run `enkf_calc` with the flag `--forecast-stats-only`;
4. remove `observations.nc` and `observations-orig.nc`;
5. calculate analysis in a "normal" way by running `enkf_prep`, `enkf_calc` and `enkf_update`.

Note that bad batches are identified at stage 2 based on superobservations formed by observations from the same batch only. Consequently, detection of bad batches may become unreliable if there

are not enough such "clean" superobservations. It is possible to alleviate this problem by (1) switching off superobing by setting `SOBSTRIDE = 0` or specifying `--no-superobing` for `enkf_prep` and/or (2) by processing observations by parts, one type (or instrument) at a time, and repeating stages 1 and 2 as many times as necessary.

2.5.2 Asynchronous DA

An observation type can be specified as "asynchronous" by specifying entry `ASYNC` entry in the observation types parameter file (sec. 2.3.4), e.g.:

```
NAME = SLA
(...)
ASYNC = 1
(...)
```

The above means that SLA and SST observations are considered to be asynchronous, with the time quantification of 1 day. If, for example the assimilation time is specified as "6085.5 days since 1990-01-01", then the SLA and SST observations will be binned into 1-day time intervals centred at the time of assimilation, i.e. from day 6080.0 to day 6081.0, 6081.0 to 6082.0, and so on, and estimated versus the corresponding model states.

The model states used to calculate forecast observations are matched by file names, which are supposed to be of the form `mem<xxx>_<variable name>_<time shift>.nc` (for the EnKF) or `bg_<variable name>_<time shift>.nc` (for the EnOI). Here "time shift" is the number of the bin, with "0" corresponding to the bin centred at the time of assimilation, "-1" to the previous bin, "1" to the next bin, and so on. If the corresponding members (or the background files, in the case of EnOI) are found, the observations are assimilated asynchronously; if they are not found, then the observations are assimilated synchronously. This can be tracked from the *calc* log file, e.g.:

```
calculating ensemble observations:
2014-03-22 06:28:28
  ensemble size = 96
  distributing iterations:
    all processes get 6 iterations
    process 0: 0 - 5
  SST |aaaaaa|aaaaaa|aaaaaa|aaaaaa|aaaaaa
  SLA |aaaaaa|aaaaaa|aaaaaa|aaaaaa|aaaaaa
  TEM ......
  SAL ......
```

The entries "a" mean that the observations are assimilated asynchronously. They would be replaced by "s" if assimilated synchronously. The vertical lines indicate the time slots for asynchronous DA; in the above example the DAW has 5 time slots. The entries "." indicate calculating ensemble observations for synchronous observations. Note that only the master process is writing to the log here, which explains why there is only output from 6 members in the log above. For EnOI there also will be "+" at the end of the line for each observation type indicating reading of the corresponding background field.

2.5.3 Superobing

"Superobing" is the process of reduction of the number of observations by merging spatially close observations before their assimilation. EnKF-C merges observations if:

- they belong to the same model grid cell;
- are of the same type;
- for asynchronous observations – belong to the same time slot.

The horizontal size of superobing cells can be increased from the default of 1 model grid cell to $N \times N$ cells by setting `SOBSTRIDE = <N>` in the parameter file; the vertical size is always equal to 1 layer. Setting `SOBSTRIDE = 0` switches superobing off.

The observations are merged by averaging their values, coordinates and times with weights inversely proportional to the observation error variance. The observation error variance of a superobservation is set to the inverse of the sum of inverse observation error variances of the merged observations. The product and instrument fields of the superobservation are set either to those of the merged observations or to -1, depending on whether the merged observations have the same values for these fields or not.

2.6 *calc*

calc is the second stage of data assimilation in EnKF-C. It calculates 2D arrays of local ensemble transforms $\mathbf{X}_5$ (for EnKF) or coefficients $\mathbf{w}$ (for EnOI).

The name of the binary for *calc* is `enkf_calc`. It has the following usage and options:

```
>./bin/enkf_calc
  Usage: enkf_calc <prm file> [<options>]
  Options:
  --describe-prm-format [main|model|grid|obstypes]
      describe format of a parameter file and exit
  --forecast-stats-only
      calculate and print forecast observation stats only
  --ignore-no-obs
      proceed even if there are no observations
  --no-mean-update
      update ensemble anomalies only
  --print-batch-stats
       calculate and print global biases for each batch of observations
  --single-observation-xyz <lon> <lat> <depth> <type> <inn> <std>
      assimilate single observation with these parameters
  --single-observation-ijk <fi> <fj> <fk> <type> <inn> <std>
      assimilate single observation with these parameters
  --use-these-obs <obs file>
      assimilate observations from this file; the file format must be compatible
      with that of observations.nc produced by 'enkf_prep'
  --version
      print version and exit
```

The option `--forecast-stats-only` can be used for quick calculation of the innovation statistics for a given background (or ensemble). This can be used, for example, for obtaining the persistence statistics, that is, the innovation statistics for the previous analysis.

The options `--single-observation-xyz` and `--single-observation-ijk` provide an easy way to conduct the so called single observation experiments, with the observation coordinates provided either in spatial or grid coordinates, correspondingly. Parameter `<value>` defines innovation rather than the observation value. Normally, this experiments would be conducted in the EnOI mode, calculating increment (option `--output-increment` of `enkf_update`) rather than analysis. When run in the EnKF mode, the increment (or analysis, depending on options) for each member is calculated.

2.6.1 Multiple model grids

EnKF-C permits using multiple model grids, in which case the ensemble transforms are calculated sequentially for each of the grids. These transforms are then used for updating of the model variables defined on the corresponding grids.

2.6.2 Interpolation of ensemble transforms

Local ensemble transforms $\mathbf{X}_5$ (EnKF) or local ensemble weights $\mathbf{w}$ (EnOI) represent smooth fields with the characteristic spatial variability scale of the localisation radius. This smoothness allows one to reduce the computational load in *calc* by calculating local transforms or weights on a subgrid with a specified stride only, and using linearly interpolated transforms or weights in the intermediate grid cells. The value of the stride is defined by the `STRIDE` entry in the parameter file.

2.6.3 Observation functions

Model estimations for observations of each type are calculated using observation functions specified for this type by entry `HFUNCTIONS` in the observation types parameter file, e.g.:

```
NAME = SLA
...
HFUNCTION = standard
...
```

The available functions for each observation type are specified by the variable `allhentries` in `calc/allhs.c`. The "standard" functions do normally perform 2D or 3D linear interpolation from the corner model grid nodes for the cell containing the observation.

2.6.4 Innovation statistics

In its course *calc* calculates some basic innovation statistics: number of observations, mean absolute forecast innovation, mean absolute analysis innovation, mean forecast innovation, mean analysis

innovation, mean forecast ensemble spread, and mean analysis ensemble spread. This statistics is provided for each region defined in the parameter file, as well as for each time slot defined for asynchronous DA, and for each instrument. In addition, for 3D observations *calc* also calculates observation statistics for observations shallower or deeper than depth (height) specified by `DEPTH_SHALLOW` and `DEPTH_DEEP` in `calc/obsstats.c`. For example:

```
printing observation statistics:
  region obs.type   # obs.  |for.inn.| |an.inn.|   for.inn.   an.inn.  for.spread  an.spread
  ------------------------------------------------------------------------------------------
  Tasman
         SST         25928    0.363     0.244     -0.077     -0.048      0.352      0.251
          -4          7214    0.303     0.187     -0.077     -0.022      0.297      0.212
          -3          4727    0.339     0.205     -0.037      0.006      0.389      0.263
          -2          4862    0.370     0.283     -0.105     -0.073      0.378      0.268
          -1          4606    0.426     0.306     -0.160     -0.143      0.377      0.275
           0          4519    0.413     0.268     -0.007     -0.023      0.348      0.257
           AVHRR     10776    0.345     0.174     -0.053     -0.031      0.336      0.227
           WindSat   14332    0.381     0.301     -0.100     -0.064      0.362      0.269
           N/A         820    0.293     0.160     -0.009      0.009      0.377      0.247
         SLA          4637    0.063     0.035      0.014      0.003      0.047      0.029
          -4           728    0.049     0.032      0.006     -0.003      0.039      0.025
          -3          1417    0.055     0.032      0.008      0.004      0.050      0.031
          -2           520    0.063     0.038      0.047      0.025      0.035      0.023
          -1           612    0.074     0.043      0.013     -0.001      0.040      0.025
           0          1360    0.073     0.036      0.012     -0.000      0.056      0.034
           g1         1652    0.058     0.039     -0.031     -0.024      0.044      0.029
           j1         1581    0.061     0.028      0.035      0.014      0.043      0.026
           n1         1357    0.071     0.040      0.045      0.025      0.054      0.034
           N/A          47    0.060     0.021     -0.001      0.001      0.050      0.030
         TEM           284    0.421     0.253     -0.119     -0.023      0.410      0.266
           ARGO        284    0.421     0.253     -0.119     -0.023      0.410      0.266
           0-50m        58    0.409     0.201     -0.013      0.105      0.315      0.215
           >500m        70    0.272     0.214      0.093      0.064      0.293      0.201
```

This excerpt shows innovation statistics for the region "Tasman". It contains sections for SST, SLA and TEM observations. The summary statistics for each observation type is shown at the top of each section; then statistics for days -4, -3, -2, -1 and 0 of a 5-day DAW are shown for the two asynchronous types, SST and SLA. After that, statistics for particular instruments is shown; "N/A" corresponds to superobservations resulted from merging observations from two or more instruments. For subsurface temperature also statistics for shallow (0–50 m) and deep (0–500 m) observations is given.

The analysis innovation statistics is calculated from the updated (analysis) ensemble observations by *calc*, thus avoiding the need to access analysis files produced later by *update*. The update of ensemble observations is performed in the same way as that of any other element of the state vector: for the EnKF – by applying the appropriate local ensemble transforms to the forecast ensemble observations,

$$\mathcal{H}(\mathbf{E}^a) \leftarrow \mathcal{H}(\mathbf{E}^f)\,\mathbf{X}_5;$$

and for the EnOI – by applying the appropriate local linear combination of the ensemble observation anomalies:

$$\mathcal{H}(\mathbf{E}^a) \leftarrow \Big[\mathcal{H}(\mathbf{x}^f) + (\mathbf{H}\mathbf{A}^f)\mathbf{w}\Big]\,\mathbf{1}^{\mathrm{T}} + \mathbf{H}\mathbf{A}^f.$$

2.6.5 Impact of observations

In the course of its work *calc* routinely calculates two metrics for assessing the impact of observations, degrees of freedom of signal (DFS) and spread reduction factor (SRF):

$$\mathrm{DFS} = \mathrm{tr}(\mathbf{KH}) = \mathrm{tr}(\mathbf{GS}),$$
$$\mathrm{SRF} = \sqrt{\frac{\mathrm{tr}(\mathbf{HP}^f\mathbf{H}^\mathrm{T}\mathbf{R}^{-1})}{\mathrm{tr}(\mathbf{HP}^a\mathbf{H}^\mathrm{T}\mathbf{R}^{-1})}} - 1 = \sqrt{\frac{\mathrm{tr}(\mathbf{S}^\mathrm{T}\mathbf{S})}{\mathrm{tr}(\mathbf{GS})}} - 1,$$

where $\mathrm{tr}(\cdot)$ is the trace function. The values of these metrics for each local analysis, calculated both for all observations and for observations of each type only, are written to file `enkf_diagn.nc`. Note that the in EnKF-C DFS and SRF are calculated from the above expressions and represent theoretical values for the EnKF analysis; they coincide with the actual DFS and SRF values only for the ETKF, but not for the DEnKF, which is an approximation of the KF (and indeed not for the EnOI, which is not even an approximation).

In the EnKF context DFS is a useful indicator of potential rank problems. Normally, it should not exceed a fraction (a half, or better, a quarter) of the ensemble size. SRF shows the "strength" of DA. "Strong" DA implies a close to optimal system, which indeed never happens in practice. Therefore, ideally, SRF should be small (below 1, on average).

2.7 *update*

update is the third and final stage of data assimilation in EnKF-C. It updates the ensemble (EnKF) or the background (EnOI) by applying the transforms calculated by *calc*.

The name of the binary for *update* is `enkf_update`. It has the following usage and options:

```
>./bin/enkf_update
  Usage: enkf_update <prm file> [<options>]
  Options:
  --calculate-spread
      calculate ensemble spread and write to spread.nc
  --describe-prm-format [main|model|grid]
      describe format of a parameter file and exit
  --direct-write
      write fields directly to the output file (default: write to tiles first)
  --leave-tiles
      do not delete tiles
  --output-increment
      output analysis increment (default: output analysis)
  --separate-output
      write results to new files (default: append to forecast files)
  --version
      print version and exit
```

The option `--separate-output` tells *update* to write the updated ensemble (EnKF) or background (EnOI) to separate analysis files, rather than append them to the forecast files. When writing to

a separate file, the same variable names are used for the analysis as for the forecast; the new files have an extra suffix `.analysis` or `.increment`, depending on whether the analysis or increment is written. By default, *update* writes results to the forecast files by creating new variable names from old names using suffix `_an`.

By default, *update* first writes each updated horizontal field of the model to a separate file (referred to here as a tile), and then concatenates these fields into analysis files. The tiles are removed after writing the analysis files; one may save time for allocating them on disk in the next cycle by leaving them on disk by using option `--leave-tiles`. This approach is somewhat less effective than direct writing to analysis files (without intermediate tiles), but, unfortunately, the direct writing is generally not reliable due to parallel I/O issues with NetCDF. Note that in some cases it proved to be possible to obtain robust performance with direct write using "classic" or "64-bit-offset" NetCDF formats.

2.7.1 Capping of inflation

Applying spatially uniform ensemble inflation involves areas with no local observations, where no assimilation is conducted. It can gradually increase the model imbalance and deteriorate performance of the DAS over time. Similar problems may arise due to lack of correlation between some state elements updated with the same transforms, so that even in presence of local observations the ensemble spread for some elements may hardly reduce. To avoid this behaviour EnKF-C currently restricts inflation by half of the the spread reduction factor calculated directly for each element of the state vector during the update. This is a default mode of application of inflation in EnKF-C; to revert to the uniform inflation add qualifier `PLAIN` to the entry `INFLATION` in the main parameter file.

2.8 DA tuning

There are the following main factors for tuning a system on the DA side:

- impact of observations, including the relative impact of different types of observations;
- inflation;
- localisation.

The impact of observations can tuned via the so-called R-factor, which defines the multiple of the observation error variance. Increasing R-factor decreases the impact of observations. Specifying R-factor equal k produces the same increment as reducing the ensemble spread by $k^{1/2}$ times.

The main parameter file defines the base R-factor common for all observation types. It is possible to specify additional R-factors for observations of each type (sec. 2.3.4); the resulting R-factor for an observation is then given by multiplication of the common R-factor and the additional R-factor specified for observations of this type.

Multiplicative inflation can be seen as an additional forgetting factor in the KF. In EnKF-C one can specify the inflation multiple for analysed ensemble anomalies, e.g.:

```
> grep INFLATION main.prm
INFLATION = 1.05
> grep temp -A 1 model.prm
VAR = temp
INFLATION = 1.02
```

In this case all model variables except "temp" will have inflation of 5 %, while "temp" will have inflation of $1.05 \cdot 1.02 \approx 1.07$. The ability to define different inflation rates for different variables can be useful for non-dynamical variables, such as estimated biases, helping to avoid the ensemble collapse for them. In general, to retain dynamical balances one should rather avoid using different inflation magnitudes across model variables. Note that even small inflation can substantially affect the ensemble spread established in the course of evolution of the system.

Localisation radius is defined by the entry `LOCRAD` in the parameter file. Specifically, this entry defines the support localisation radius (in km). This is different to the "effective" localisation radius, which is defined sometimes as $e^{1/2} \approx 1.65$ - folding distance, and for the Gaspary and Cohn's taper function is approximately 3.5 times smaller than the support radius.

Increasing the localisation radius increases the number of local observations and hence the overall impact of observations. To compensate this in a system with horizontal localisation one has to change the R-factor as the square of the localisation radius.

2.9 Point logs

"Point logs" refer to a capability of EnKF-C to save DA related information for a number of horizontal locations specified in the parameter file, e.g.:

```
POINTLOG 94 134
POINTLOG 78 111
POINTLOG 57 51
POINTLOG 86 191
```

Here the information will be saved at each DA cycle for points with horizontal grid coordinates (94, 134), (78, 111), (67, 51), and (86, 191) in files `pointlog_94,134.nc`, `pointlog_78,111.nc`, and so on. The rationale behind this capability is that it is usually too expensive to save the whole ensemble, but is feasible to save the model states in a limited number of points. Following is an example of the header of the saved point log file in NetCDF format:

```
netcdf pointlog_57\,51 {
dimensions:
        m = 96 ;
        p = 2902 ;
        nk = 51 ;
variables:
        int obs_ids(p) ;
        float lcoeffs(p) ;
```

```
        float lon(p) ;
        float lat(p) ;
        float depth(p) ;
        float obs_val(p) ;
        float obs_std(p) ;
        float obs_fi(p) ;
        float obs_fj(p) ;
        float obs_fk(p) ;
        int obs_type(p) ;
                obs_type:SST = 0 ;
                obs_type:RFACTOR_SST = 2. ;
                obs_type:SLA = 1 ;
                obs_type:RFACTOR_SLA = 1. ;
                obs_type:TEM = 2 ;
                obs_type:RFACTOR_TEM = 1. ;
                obs_type:SAL = 3 ;
                obs_type:RFACTOR_SAL = 1. ;
        float obs_date(p) ;
                obs_date:units = "days from 8797.5 days since 1990-01-01" ;
        float s(p) ;
        float S(m, p) ;
        double X5(m, m) ;
        float eta_t(m) ;
        float temp(nk, m) ;
        float salt(nk, m) ;
        float u(nk, m) ;
        float v(nk, m) ;

// global attributes:
                :date = "8797.5 days since 1990-01-01" ;
                :i = 57 ;
                :j = 51 ;
                :lon = 95.75 ;
                :lat = -44.8499984741211 ;
                :depth = 3072. ;
```

This information makes it possible to check DA algorithms by reproducing the ensemble transforms calculated by EnKF-C from $\mathbf{S}$ and $\mathbf{s}$ according to section 1.3.3; restore observations from $\mathbf{s}$ by using the corresponding R-factors and localisation coefficients; to monitor the ensemble spread for each model variable; and so on.

2.10 Use of innovation statistics for model validation

EnKF-C can calculate innovation statistics for validating a model against observations only, without data assimilation. The pre-requisites are (i) observations and (ii) model dump readable by the code, and possibly (iii) auxiliary files for projecting the model state to observation space (e.g. grid specs and mean SSH). To get the innovation statistics one needs to:

- set up the parameter files in a normal way (`MODE = EnOI`), omitting the ensemble directory and assimilation related parameters;
- run `enkf_prep`;

- run `enkf_calc` with additional parameter `--forecast-stats-only`.

The results will be written to the log of `enkf_calc`. An example of using this functionality is available by running `make stats` in `examples/1` (see sec. 2.2).

2.11 Bias estimation

It is possible to estimate model bias for a variable with the EnKF by generating and using an ensemble of bias fields. These bias fields need to be subtracted from observation forecasts of the corresponding types. This is accomplished by specifying a secondary variable in the observation type specifications by the entry `VAR2` and by passing the name of this variable to the corresponding observation functions, which have to take care for subtracting the model and bias forecasts.

2.12 System issues

2.12.1 Memory footprint

To reduce the memory footprint, most of the potentially big arrays in EnKF-C use `float` data type.

The memory footprint of *prep* is defined by the size of the `measurement` structure and the number of observations. It is rarely a problem.

The memory footprint of *calc* is mainly defined by the size of ensemble observation anomalies that require $p \times m \times 4$ bytes for storage using `float` data type. For example, with $3 \cdot 10^6$ superobservations and 10^2 ensemble members the size of this array would be about 1.2 GB per CPU, which should be manageable on most contemporary systems. If the footprint becomes too big, one may consider reducing the number of observations by a coarser superobing (setting the parameter `SOBSTRIDE` to 2 or more) or reducing the number of cores per node used.

The memory footprint of *update* is mainly defined by the size of the array of simultaneously updated horizontal fields. For example, for a 1500×3600 horizontal field, 100 ensemble members and simultaneous update of 2 fields the size of this array would be about 4.3 GB per CPU. It could be reduced to 2.15 GB by reducing he parameter `FIELDBUFFERSIZE` from 2 to 1. Note that reducing the number of simultaneously updated fields defined by `FIELDBUFFERSIZE` increases I/O (the $\mathbf{X}_5$ array is read from disk N_f/N_b times, where N_f is the total number of horizontal fields, and N_b is the number of simultaneously updated fields) and reduces computational effectiveness (the $\mathbf{X}_5$ array needs to be interpolated horizontally N_f/N_b times).

2.12.2 Exit action

When exiting on an error, EnKF-C by default prints the stack trace, which allows to trace the exit location in the code. Another option – to generate a segmentation fault – can be activated by setting `EXITACTION = SEGFAULT` in the parameter file. Note that when run on multiple processors,

this can result in segmentation faults on more than one processor (but not necessarily on every engaged processor, as some processes can also be forced to exit by `MPI_abort()`). If the system is set to generate core dumps, they can indeed be used for investigating the final state of the program.

2.12.3 Dependencies

Compiling EnKF-C requires the following external packages:

- netcdf;
- lapack (or mkl_rt);
- openmpi;
- gridutils.

EnKF-C also relies on `qsort_r()`, which may be lacking in older systems. In such cases use compile flag `-DINTERNAL_QSORT_R` to activate the internal version of this procedure.

Notes:

1. Using Intel's version of Lapack library – Intel Math Kernel Library – can improve performance over Lapack compiled with gfortran.
2. Grid utilities (libgu) is necessary for handling curvilinear grids only. Add compile flag `-DNO_GRIDUTILS` to compile without it.

2.13 Possible problems

2.13.1 *calc* becomes too slow after increasing localisation radius

This is due to the increased number of local observations.

The local observations are sought by using so called k-d tree. Normally it works well, but can become a bottleneck when the number of local observations becomes very large. If this happens, one may use a coarser superobing to reduce the number of assimilated observations, by increasing parameter `SOBSTRIDE` from the default value of 1 to 2 or more.

Acknowledgements

EnKF-C has been developed during author's work with Bureau of Meteorology on Bluelink project. The author used his knowledge of TOPAZ (Sakov et al., 2012) and BODAS (Oke et al., 2008) codes and borrowed from them a number of design solutions and features. Paul Sandery was the first user of this code (apart from the author), and his enthusiastic support is cheerfully acknowledged.

References

Andrews, A., 1968: A square root formulation of the Kalman covariance equations. *AIAA J.*, **6**, 1165–1168.

Bishop, C. H., B. Etherton, and S. J. Majumdar, 2001: Adaptive sampling with the ensemble transform Kalman filter. part I: theoretical aspects. *Mon. Wea. Rev.*, **129**, 420–436.

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

— 2003: The Ensemble Kalman Filter: theoretical formulation and practical implementation. *Ocean Dynam.*, **53**, 343–367.

— 2004: Sampling strategies and square root analysis schemes for the EnKF. *Ocean Dynam.*, **54**, 539–560.

Evensen, G. and P. J. van Leeuwen, 2000: An ensemble Kalman smoother for nonlinear dynamics. *Mon. Wea. Rev.*, **128**, 1852–1867.

Gaspari, G. and S. E. Cohn, 1999: Construction of correlation functions in two and three dimensions. *Q. J. R. Meteorol. Soc.*, **125**, 723–757.

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

Houtekamer, P. L. and H. L. Mitchell, 2001: A sequential ensemble Kalman filter for atmospheric data assimilation. *Mon. Wea. Rev.*, **129**, 123–137.

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

Hunt, B. R., E. J. Kostelich, and I. Szunyogh, 2007: Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D*, **230**, 112–126.

Kalman, R. E., 1960: A new approach to linear filtering and prediction problems. *J. Basic. Eng.*, **82**, 35–45.

Oke, P. R., G. B. Brassington, D. A. Griffin, and A. Schiller, 2008: The Bluelink ocean data assimilation system (BODAS). *Ocean Model.*, **21**, 46–70.

Ott, E., B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corazza, E. Kalnay, D. J. Patil, and J. A. Yorke, 2003, rev. 2005: A local ensemble Kalman filter for atmospheric data assimilation. *http://arxiv.org/abs/physics/0203058*.

Sakov, P. and L. Bertino, 2011: Relation between two common localisation methods for the EnKF. *Comput. Geosci.*, **15**, 225–237.

Sakov, P., F. Counillon, L. Bertino, K. A. Lisæter, P. R. Oke, and A. Korablev, 2012: TOPAZ4: an ocean-sea ice data assimilation system for the North Atlantic and Arctic. *Ocean Science*, **8**, 633–656.

Sakov, P., G. Evensen, and L. Bertino, 2010: Asynchronous data assimilation with the EnKF. *Tellus*, **62A**, 24–29.

Sakov, P. and P. R. Oke, 2008a: A deterministic formulation of the ensemble Kalman filter: an alternative to ensemble square root filters. *Tellus*, **60A**, 361–371.

— 2008b: Implications of the form of the ensemble transformations in the ensemble square root filters. *Mon. Wea. Rev.*, **136**, 1042–1053.

Abbreviations

CL	-	covariance localisation
DEnKF	-	deterministic EnKF
DA	-	data assimilation
DAS	-	data assimilation system
DAW	-	data assimilation window
DFS	-	degrees of freedom of signal
EKF	-	extended Kalman filter
EnKF	-	ensemble Kalman filter
EnOI	-	ensemble optimal interpolation
ETKF	-	ensemble transform Kalman filter
ETM	-	ensemble transform matrix
FGAT	-	first guess at appropriate time
KF	-	Kalman filter
KS	-	Kalman smoother
LA	-	local analysis
SDAS	-	state of data assimilation system
SRF	-	spread reduction factor
SVD	-	singular value decomposition

Symbols

General symbols

$\mathbf{x}$ (small, bold)	-	a vector
$\mathbf{1}$	-	a vector with all elements equal to 1
$\mathbf{0}$	-	a vector with all elements equal to 0
$\mathbf{A}$ (capital, bold)	-	a matrix
$\mathbf{I}$	-	identity matrix
$\mathbf{U}$	-	a unitary matrix, $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$
$\mathbf{A}^{\mathrm{T}}$	-	transposed matrix $\mathbf{A}$
$\mathbf{A}^{1/2}$	-	unique positive definite square root of a positive definite matrix $\mathbf{A}$
$\mathrm{tr}(\mathbf{A})$	-	trace of $\mathbf{A}$
$\mathcal{H} \circ \mathcal{M}(\mathbf{x})$	-	$\mathcal{H}\left[\mathcal{M}(\mathbf{x})\right]$
$\mathbf{A} \circ \mathbf{B}$	-	by-element, or Hadamard, or Schur product of matrices

DA related symbols

m	-	ensemble size
n	-	state size
p	-	number of observations
$\mathbf{A}$	-	ensemble anomalies, $\mathbf{A} = \mathbf{E} - \mathbf{x}\mathbf{1}^{\mathrm{T}}$
$\mathbf{E}$	-	ensemble
$\mathcal{H}$	-	nonlinear observation operator; in linear case – affine observation operator
$\mathbf{G}$	-	an intermediate matrix in the EnKF analysis, $\mathbf{G} \equiv (\mathbf{I} + \mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathrm{T}} = \mathbf{S}^{\mathrm{T}}(\mathbf{I} + \mathbf{S}\mathbf{S}^{\mathrm{T}})^{-1}$
$\mathbf{H}$	-	linearised observation operator, $\mathbf{H} = \nabla\mathcal{H}(\mathbf{x})$
$\mathcal{M}$	-	nonlinear model operator; in linear case – affine model operator
$\mathbf{M}$	-	linearised model operator, $\mathbf{M} = \nabla\mathcal{M}(\mathbf{x})$
$\mathbf{P}$	-	state error covariance estimate; also used as abbreviation for $\mathbf{A}\mathbf{A}^{\mathrm{T}}/(m-1)$
$\mathbf{R}$	-	observation error covariance
$\mathbf{S}$	-	normalised ensemble observation anomalies, $\mathbf{S} = \mathbf{R}^{-1/2}\mathbf{H}\mathbf{A}/\sqrt{m-1}$
$\mathbf{T}_L$	-	left-multiplied ensemble transform matrix, $\mathbf{A}^a = \mathbf{T}_L\mathbf{A}^f$
$\mathbf{T}_R$	-	right-multiplied ensemble transform matrix, $\mathbf{A}^a = \mathbf{A}^f\mathbf{T}_R$
$\mathbf{U}^p$	-	a unitary mean-preserving matrix, $\mathbf{U}^p(\mathbf{U}^p)^{\mathrm{T}} = \mathbf{I}$, $\mathbf{U}^p\mathbf{1} = \mathbf{1}$
$\mathbf{X}_5$	-	historic symbol for the full ensemble transform matrix, $\mathbf{E}^a = \mathbf{E}^f\mathbf{X}_5$
$\mathbf{s}$	-	normalised innovation, $\mathbf{s} = \mathbf{R}^{-1/2}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}^f)\right]/\sqrt{m-1}$
$\mathbf{x}$	-	state estimate
$\mathbf{y}$	-	observation vector
$\mathbf{w}$	-	vector of linear coefficients for updating the mean, $\mathbf{x}^a = \mathbf{x}^f + \mathbf{A}^f\mathbf{w}$
$(\cdot)^f$	-	forecast expression
$(\cdot)^a$	-	analysis expression
$(\cdot)_i$	-	either expression at cycle i or ith element of a vector
$\overset{i}{(\cdot)}$	-	local expression for state element i
$\overset{\{o\}}{(\cdot)}$	-	local expression for observation o