Tellus
SERIES A
DYANAMIC
METEOROLOGY
AND OCEANOGRAPHY
PUBLISHED BY THE INTERNATIONAL METEOROLOGICAL INSTITUTE IN STOCKHOLM

Check for updates

State-of-the-art stochastic data assimilation methods for high-dimensional non-Gaussian problems

By SANITA VETRA-CARVALHO[1]*, PETER JAN VAN LEEUWEN[1,2], LARS NERGER[3], ALEXANDER BARTH[4], M. UMER ALTAF[5], PIERRE BRASSEUR[6], PAUL KIRCHGESSNER[3] and JEAN-MARIE BECKERS[4], [1]*Department of Meteorology, University of Reading, Reading, UK;* [2]*National Centre for Earth Observation, Reading, UK;* [3]*Alfred Wegener Institute, Helmholtz Center for Polar and Marine Research, Bremerhaven, Germany;* [4]*GeoHydrodynamic and Environmental Research (GHER), University of Liège, Liège, Belgium;* [5]*King Abdullah University of Science and Technology, Thuwal, Saudi Arabia;* [6]*CNRS, IRD, Grenoble INP, IGE, University of Grenoble Alpes, Grenoble, France*

(Manuscript received 5 July 2017; in final form 19 February 2018)

ABSTRACT

This paper compares several commonly used state-of-the-art ensemble-based data assimilation methods in a coherent mathematical notation. The study encompasses different methods that are applicable to high-dimensional geophysical systems, like ocean and atmosphere and provide an uncertainty estimate. Most variants of Ensemble Kalman Filters, Particle Filters and second-order exact methods are discussed, including Gaussian Mixture Filters, while methods that require an adjoint model or a tangent linear formulation of the model are excluded. The detailed description of all the methods in a mathematically coherent way provides both novices and experienced researchers with a unique overview and new insight in the workings and relative advantages of each method, theoretically and algorithmically, even leading to new filters. Furthermore, the practical implementation details of all ensemble and particle filter methods are discussed to show similarities and differences in the filters aiding the users in what to use when. Finally, pseudo-codes are provided for all of the methods presented in this paper.

Keywords: ensemble Kalman filter, particle filter, data assimilation, high dimension, non Gaussian

1. Introduction

Data assimilation (DA) is the science of combining observations of a system, including their uncertainty, with estimates of that system from a dynamical model, including its uncertainty, to obtain a new and more accurate description of the system including an uncertainty estimate of that description. The uncertainty estimates point to an efficient description in terms of probability density functions, and in this paper we discuss methods that perform DA using an ensemble of model states to represent these probability density functions.

Ensemble Kalman filters are currently highly popular DA methods that are applied to a wide range of dynamical models including oceanic, atmospheric, and land surface models. The increasing popularity of Kalman-Filter-based ensemble (EnKF) methods in these fields is due to the relative ease of the filter implementation, increasing computational power and the natural forecast error evolution in EnKF schemes with the dynamical model in time. However, due to technological and scientific advances, three significant developments have occurred in the last decade that force us to look beyond standard Ensemble Kalman Filtering, which is based on linear and/or Gaussian assumptions. Firstly, continuous increase in computational capability has recently allowed to run operational models at high resolutions so that the dynamical models have become increasingly non-linear due to the direct resolution of small-scale non-linear processes in these models, e.g. small-scale turbulence. Secondly, in several geoscientific applications, such as atmosphere, ocean, land surface, hydrology and sea – i.e. it is of interest to estimate variables or parameters that are bounded requiring DA methods that can deal with non-Gaussian distributions. Thirdly, the observational network around the world has increased many-fold for weather, ocean and land surface areas providing more information about the real system with greater accuracy and higher spatial and temporal resolution. Often the so-called observation operators that connect model states to observations of these new observations are non-linear, again asking for non-Gaussian DA methods. Thus, the research in non-linear DA

*Corresponding author. e-mail: s.vetra-carvalho@reading.ac.uk

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
Citation: Tellus A: 2018, **70**, 1445364, https://doi.org/10.1080/16000870.2018.1445364

methods, which can be applied to high-resolution dynamical models and/or complex observation operators, has seen major developments in the last decade with the aim to understand how existing ensemble methods cope with non-linearity in the models, to develop new ensemble methods that are more suited to non-linear dynamical models, as well as to explore non-linear filters that are not limited to Gaussian distributions, such as particle filters or hybrids between particle and ensemble Kalman filters.

The origin of this paper lies in the EU-funded research project SANGOMA (Stochastic Assimilation for the Next Generation Ocean Model Applications). The project focused on generating a coherent and transparent database of the current ensemble-based data assimilation methods and development of data assimilation tools suitable for non-linear and high-dimensional systems concentrating on methods that do not require tangent linear approximations of the model or its adjoint. The methods described within this paper have been applied in operational oceanography, like the TOPAZ system (Sakov et al., 2012) or the FOAM system of the UK Met Office (see Blockley et al., 2014). While TOPAZ is already using an EnKF, FOAM applies an optimal interpolation scheme that takes in less dynamical information to estimate the error covariance matrix. That said this paper is aimed at a very broad audience and data assimilation methods discussed in this paper are not limited to applications to ocean or atmosphere models, hence, the methods are presented without the context of any specific dynamical model, allowing the reader to make the most of each technique for their specific application.

A number of reviews have been published recently, each collating parts of the development in data assimilation, e.g. Bannister (2017) gives a comprehensive review of operational methods of variational and ensemble-variational data assimilation, Houtekamer and Zhang (2016) review the ensemble Kalman filter with a focus on application to atmospheric data assimilation, Law and Stuart (2012) review variational and Kalman filter methods, and Bocquet et al. (2010) present a review of concepts and ideas of non-Gaussian data assimilation methods and discusses various sources of non-Gaussianity. The merits of this paper lie within:

- coherent mathematical description of the main methods that are used in the current data assimilation community for application to high-dimensional and non-Gaussian problems allowing the reader to easily oversee the differences between the methods and compare them;
- discussing ensemble Kalman Filters, particle filters, second-order exact filters and Gaussian Mixture Filters within the same paper using consistent notation;
- inclusion of practical application aspects of these methods, discussing computational cost, parallelising, localisation and inflation techniques;
- provision of pseudo-code algorithms for all of the presented methods;

along with inclusion of recent developments, such as the error-subspace transform Kalman filter (ESTKF) and recent particle filters, this paper goes beyond earlier reviews (e.g. Tippett et al., 2003; Hamill et al., 2006; van Leeuwen, 2009; Houtekamer and Zhang, 2016; Bannister, 2017).

The paper is organised as follows: in Section 2 the common ground through Bayes theorem is established, in Section 3 a historical overview is given for both ensemble Kalman and particle filter fields, and in Section 4 we define the basic problem solved by all of the methods presented in this paper. In Section 5 we discuss the most popular types of ensemble Kalman filter methods. Then, in Section 6, we discuss several particle filter methods that can be applied to high-dimensional problems. In Section 7, we describe ensemble filters with second-order accuracy, namely the particle filter with Gaussian resampling (PFGR), the non-linear ensemble transform filter (NETF), and the moment-matching ensemble filter. The Gaussian mixture filter is discussed in Section 8. The practical implementation of the filters including localisation, inflation, parallelisation and the computation cost as well as the aspect of non-linearity are discussed in Section 9. Finally, Appendix 1 provides pseudo codes for resampling techniques often used in particle filter methods, and Appendix 2 contains pseudo codes for all of the methods discussed in this paper.

We note that many of the filters discussed in this paper are available freely from *Sangoma* project website[1] along with many other tools valuable and/or necessary for data assimilation systems.

1.1. Notation

In the data assimilation community the currently most accepted notation is described in Ide et al. (1997). We adhere to this notation where possible while also making this paper acceptable and intuitive not only to data-assimilation experts but also to a wider audience including those who might like to explore data assimilation methods simply as tools for their specific needs. To this end, throughout this paper dimensions will always be described by capital N with an underscore indicating the space in question, that is

- N_x - dimension of state space;
- N_y - dimension of observation space;
- N_e - dimension of ensemble/particle space.

Further, the time index is always denoted in parentheses in the upper right corner of the variables, i.e. $(.)^{(m)}$, except for operators such as $\mathcal{M}$ (dynamical model) and $\mathcal{H}$ (observation operator) where it is in the lower right corner. However, we will omit the time index when possible to ease the notation. We will

refer to each ensemble member (or each particle) by $\mathbf{x}_j$ where the index $j = 1, \ldots, N_e$ and N_e is the total number of the ensemble members (or particles).

When discussing Bayesian theory in Sections 2 and 6 purely random variables will be denoted by capital letters, and fixed or deterministic or observed quantities will be denoted by lowercase letters. Probability density functions will be denoted by $p\,(..)$, and $q(..)$ and we will use lower case arguments in this context.

Throughout the paper, Greek letters will refer to various errors, e.g. observational or model errors. Finally, bold lowercase letters will denote vectors and bold uppercase letters will denote matrices.

2. Common ground through Bayes theorem

Various types of data assimilation methods, e.g. variational, ensemble Kalman filters, particle filters, etc., have originated from different fields and backgrounds, due to the needs of a particular community or application. However, all of these methods can be unified through Bayes theorem. In this section, we will give a summary of Bayes theorem showing how both ensemble Kalman filter (KF) methods and particle filter (PF) methods are linked in this context and what problems each of them solve. For an introduction to the Bayesian theory for data assimilation, the reader is referred to e.g. van Leeuwen and Evensen (1996) and Wikle and Berliner (2006).

Data assimilation is an approach for combining observations with model forecasts to obtain a more accurate estimate of the state and its uncertainty. In this context, we require

- *data*, that is observations $\mathbf{y}$ and a knowledge of their associated error distributions and
- a *prior*, that is a model forecast of the state, $\mathbf{x}^f$, and knowledge of the associated forecast and model errors;

to obtain the *posterior*, i.e. the analysis state $\mathbf{x}^a$, and its associated error. The posterior can be computed through Bayes theorem which states that

$$p\,(\mathbf{x}|\mathbf{y}) = \frac{p\,(\mathbf{y}|\mathbf{x})\;p\,(\mathbf{x})}{p\,(\mathbf{y})}, \tag{1}$$

where $p\,(\mathbf{x}|\mathbf{y})$ is the posterior or analysis probability density function, $p\,(\mathbf{y}|\mathbf{x})$ is the observation probability density function or also called the likelihood, $p\,(\mathbf{x})$ is the prior or forecast probability density function, and $p\,(\mathbf{y})$ is the marginal probability density function of the observations, which can be thought of as a normalising constant. From now on, for the ease of the readability, we will abbreviate 'probability density function' with 'pdf'.

Typically, data assimilation methods make the Markovian assumption for the dynamical model $\mathcal{M}$ and the conditional independence assumption of the observations. That is, we assume that the model state or the prior at time m, when conditioned on all previous states only depends on the state at the time $m-1$,

$$p\left(\mathbf{x}^{(0:T)}\right) = p\left(\mathbf{x}^{(0)}\right) \prod_{m=1}^{T} p\left(\mathbf{x}^{(m)}|\mathbf{x}^{(m-1)}\right). \tag{2}$$

Here, the superscript $0:T$ is to be read for time indices from initial time to time T, which is typically called *assimilation window* in data assimilation. Further, observations are also usually assumed to be conditionally independent, i.e. they are assumed to be independent in time,

$$p\left(\mathbf{y}^{(1:T)}|\mathbf{x}^{(0:T)}\right) = \prod_{m=1}^{T} p\left(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}\right). \tag{3}$$

Using Equations (2) and (3) we can rewrite Bayes theorem in Equation (1) as

$$p\left(\mathbf{x}^{(T)}|\mathbf{y}^{(T)}\right) \propto p(\mathbf{x}^0) \prod_{m=1}^{T} p\left(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}\right) p\left(\mathbf{x}^{(m)}|\mathbf{x}^{(m-1)}\right). \tag{4}$$

The Markovian assumption allows us to use new observations as they become available by updating the previous estimate of the state process without having to start the calculations from scratch. This is called *sequential* updating and the methods described in this paper all follow this approach.

Ensemble Kalman filter methods solve this problem using Gaussian assumptions for both prior and likelihood pdf's. Multiplying two Gaussian pdf's leads again to a Gaussian pdf, i.e. the posterior or analysis pdf will also be Gaussian. The posterior pdf will have only one global maximum, which will correspond to the ensemble mean (also mode and median since the pdf is Gaussian). In other words, the posterior pdf in ensemble Kalman filter methods described in Section 5 is found in terms of the first two moments (mean and covariance) of the prior and likelihood pdf's. This is also true when ensemble Kalman filters are applied to non-linear dynamical models or observation operators, in which case the information from higher moments in an ensemble KF analysis update is ignored. This is a shortcoming in ensemble Kalman filters when applied to non-Gaussian problems. However, in general ensemble Kalman filter methods are robust when applied to non-linear models and catastrophic filter divergence, where the filter deviates strongly from the observations while producing unrealistically small error estimates, occurs mainly due to sparse or inaccurate observations (Verlaan and Heemink, 2001; Tong et al., 2016). It should, of course, be realised that in non-linear settings the estimates of the posterior mean and covariance might be off.

In particle filter methods, the posterior is obtained using the prior and likelihood pdf's directly in Equation (1) without restricting them to being Gaussian. If both prior and likelihood are Gaussian the resulting posterior or analysis pdf is also Gaussian and has a global maximum corresponding to the mean state. However, if either or both prior and likelihood pdf's are non-Gaussian then the resulting posterior pdf will also not be Gaussian. In other words, if the dynamical model or mapping of the model variables to observation space are non-linear then particle filter methods will produce an analysis pdf which will provide knowledge of more than the first two statistical moments (mean and covariance), in contrast to ensemble Kalman filter methods. Thus, the analysis pdf could be skewed, multi-modal or of varying width in comparison to a Gaussian pdf. Hence, particle filters are, by design, able to produce analysis pdf's for non-Gaussian problems. While standard particle filter methods suffer from filter divergence for large problems recently several particle filter variants have been developed that avoid this divergence.

In what follows, we will describe numerous filtering methods in Sections 5, 6, 7, and 8 and discuss how each method attempts to produce an analysis pdf for non-Gaussian and high-dimensional problems. However, firstly we provide an overview of the historical development of both ensemble Kalman filters and particle filter methods to show how these fields have evolved and what has given rise in the development of each of the methods.

3. History of filtering for data assimilation

Before we precede to the main point of our paper – describing in unified notation current state-of-the-art ensemble and particle filter methods for non-linear and non-Gaussian applications, their implementation, and practical application, a short summary is in order on the historical development in both ensemble Kalman filter and particle filter areas.

3.1. Development history of ensemble Kalman filters

Ensemble data assimilation (EnDA) started in 1994 with the introduction of the Ensemble Kalman filter (EnKF, Evensen (1994)). The use of perturbed observations was introduced a few years later simultaneously by Burgers et al. (1998) and Houtekamer and Mitchell (1998) to correct the previously too low spread of the analysis ensemble. This filter formulation defines today the basic 'Ensemble Kalman filter', which we will denote as the Stochastic Ensemble Kalman Filter, with a slightly different interpretation and implementation, as will be described later. The first alternative variant of the original EnKF was introduced by Pham et al. (1998a) in the form of Singular 'Evolutive' Interpolated Kalman (SEIK) filter. The SEIK filter formulates the analysis step in the space spanned by the ensemble and hence is computationally particularly efficient. In contrast to the EnKF, which was formulated as a Monte Carlo method, the SEIK filter was designed to find the analysis ensemble by writing each posterior member as a linear combination of prior members without using perturbed observations. Another ensemble Kalman filter that uses the space spanned by the ensemble was introduced with the Error-Subspace Statistical Estimation (ESSE) method (Lermusiaux and Robinson, 1999).

The filters mentioned above were all introduced for data assimilation in oceanographic problems. A new set of filter methods was introduced during the years 2001 and 2002 for meteorological applications. The Ensemble Transform Kalman Filter (ETKF, Bishop et al., 2001) was first introduced in the context of meteorological adaptive sampling. Further, the Ensemble Adjustment Kalman Filter (EAKF, Anderson, 2001) and the Ensemble Square Root Filter (EnSRF, Whitaker and Hamill, 2002) were introduced. The motivation for these three filters was to avoid the use of perturbed observations, which were found to introduce additional sampling error into the filter solution, with the meteorological community apparently being unaware of the development of the SEIK filter. The new filters were classified as ensemble square root Kalman filters and presented in a uniform notation by Tippett et al. (2003). Nerger et al. (2005a) further classified the EnKF and SEIK filters as error-subspace Kalman filters because the filters compute the correction in the error-subspace spanned by the ensemble. This likewise holds for the ETKF, EAKF, and EnSRF, however, these filters do not explicitly use a basis in the error subspace but use the ensemble to represent the space. When the EAKF and EnSRF formulations are used to assimilate all observation at once, these filters exhibit a much larger computational cost compared to the ETKF. To reduce the cost, the original study on the EnSRF (Whitaker and Hamill, 2002) already introduced a variant in which observations are assimilated sequentially, which assumes that the observation errors are uncorrelated. A similar serial formulation of the EAKF was introduced by Anderson (2003). This sequential assimilation of observations was assessed by Nerger (2015) and it was shown that this formulation can destabilise the filtering process in cases when the observations have a strong influence.

With regard to the classification as an ensemble square root Kalman filter, the SEIK filter is the first filter method that was clearly formulated in square root form. The original EnKF uses the square root form only implicitly but an explicit square root formulation of the EnKF was presented by Evensen (2003).

The methods above all solve the original equations of the Kalman filter but use the sample covariance matrix of the ensemble to represent the state error covariance matrix. An alternative was introduced with the Maximum-Likelihood Kalman Filter (MLEF, Zupanski, 2005). This filter represents the first variant of the class of hybrid filters that were introduced in later years. The filter computes the maximum-a posteriori solution (in contrast to the minimum-variance solution of the Kalman filter) by an iterative scheme.[2]

While the EnKFs were very successful in making the application of the Kalman filter feasible for the high-dimensional

problems in oceanography and meteorology, the affordable ensemble size was always very limited. To counter the issue of sampling error in ensemble covariances (the ensemble-sampled covariance has a rank of not more than the ensemble size minus one while the applications were of very high state dimension) the method of covariance localisation was introduced by Houtekamer and Mitchell (1998) and Houtekamer and Mitchell (2001). Later, an alternative localisation was introduced for the ETKF (LETKF, Hunt et al., 2007) which uses a local analysis (also used previously, e.g. by Cohn et al. (1998)) where observations are down-weighted with increasing distance from the local analysis point through a tapering of the inverse observation covariances.

The relationship between the SEIK filter and the ETKF was investigated by Nerger et al. (2012a). The study leads to a new filter formulation, the Error-Subspace Transform Kalman Filter (ESTKF), which combined the advantages of both filter formulations.

The filters mentioned above represent main developments of the ensemble Kalman filters. However, there are many other developments, which are not included here. Some of them are discussed in the sections below, in particular with regard to localisation. Overall, while there are different reviews of selections of ensemble Kalman filters, a complete and coherent overview of the different methods is still missing.

3.2. *Development history of particle filters*

Particle filters, like ensemble Kalman filters, are variants of Monte Carlo methods in which the probability distribution of the model state given the observations is approximated by a number of particles; however, unlike ensemble Kalman filters, particle filters are fully non-linear data assimilation techniques. From a sampling point of view, Ensemble Kalman Filters draw samples directly from the posterior since the probability distribution function (pdf) is assumed to be a Gaussian. In a particle filter application, the shape of the posterior is not known, and hence one cannot sample directly from it. In its simplest form, samples are generated from the prior after which importance sampling is employed to turn them into samples from the posterior where each sample is weighted with its likelihood value.

Particle filters emerged before ensemble Kalman filters, and when Gordon et al. (1993) introduced resampling in the sequential scheme the method became mainstream in non-linear filtering. This basic scheme has been made more efficient for specific applications in numerous ways, like looking ahead, adding small perturbations to resampled particles to avoid that they are the same etc. (see Doucet et al., 2001 for a very useful review of the many methods available at that time). Attempts to apply the particle filter to geophysical systems are as old as 1996 (van Leeuwen and Evensen, 1996), with the first partially successful application by van Leeuwen (2003a). However, until recently, particle filters have been deemed to be computationally unfeasible for large-dimensional systems due to the filter degeneracy problem (Bengtsson et al., 2008; Snyder et al., 2008; van Leeuwen, 2009). This means that the likelihood weights vary substantially between the particles when the number of independent observations is large, such that one particle obtains a weight close to one, while all the others have weight very close to zero. New developments in the field generated particle filter variants that have been shown to work for large dimensional systems with a limited number of particles. These methods can be divided in two classes: those that use localisation (starting with van Leeuwen, 2003b; Bengtsson et al., 2003), followed more recently by local variants of the ensemble transform particle filter (ETPF, Reich, 2013) and the Local Particle Filter (Poterjoy, 2016a) and those that exploit the future observational information via proposal densities, such as the Implicit Particle Filter (Chorin and Tu, 2009), the Equivalent Weights Particle Filter (EWPF, van Leeuwen, 2010; van Leeuwen, 2011; Ades and van Leeuwen, 2013), and the Implicit Equal Weights Particle Filter (IEWPF, Zhu et al., 2016).

In another development, second-order exact filters have been developed that ensure that the first two moments of the posterior pdf are consistent with the particle filter, and higher-order moments are not considered. The first paper of this kind was the Particle Filter with Gaussian Resampling of Xiong et al. (2006), followed by the Merging Particle Filter (Nakano et al., 2007) and the Moment Matching Ensemble Filter (Lei and Bickel, 2011). All these filters seem to have been developed independently. The Non-linear Ensemble Transform Filter (Tödter and Ahrens, 2015) can be considered a local version of the filter by Xiong et al. (2006), ironically again developed independently.

A further approximation to particle filtering is the Gaussian Mixture Filter first introduced in the geosciences by Bengtsson et al. (2003), followed by the adaptive Gaussian mixture filter variants (Hoteit et al., 2008; Stordal et al., 2011). The advantage of these filters over the standard particle filter is that each particle is 'dressed' by a Gaussian such that the likelihood weights are calculated using a covariance that is broader than the pure observational covariance, leading to better behaving weights at the cost of reducing the influence of the observations on the posterior pdf (see e.g. van Leeuwen, 2009).

4. The problem

Consider the following non-linear stochastic discrete-time dynamical system at a time when observations are available:

$$\mathbf{x}^{(m)} = \mathcal{M}_m\left(\mathbf{x}^{(m-1)}\right) + \boldsymbol{\beta}^{(m)} \tag{5}$$

$$\mathbf{y}^{(m)} = \mathcal{H}_m\left(\mathbf{x}^{(m)}\right) + \boldsymbol{\beta}_o^{(m)}, \tag{6}$$

where $\mathbf{x}^{(m)} \in \mathcal{R}^{N_x}$ is the N_x dimensional state vector, $\mathbf{y}^{(m)} \in \mathcal{R}^{N_y}$ is the observation vector of size $N_y \ll N_x$, $\mathcal{M}_m : \mathcal{R}^{N_x} \rightarrow$

$\mathcal{R}^{N_x}$ is the forward model operator, $\mathcal{H}_m : \mathcal{R}^{N_x} \rightarrow \mathcal{R}^{N_y}$ is the observation operator, $\boldsymbol{\beta}^{(m)} \in \mathcal{R}^{N_x}$ is the model noise (or error) distributed Gaussian with a covariance matrix $\mathbf{Q}^{(m)}$, and $\boldsymbol{\beta}_o^{(m)} \in \mathcal{R}^{N_y}$ is the observation noise (or error) distributed Gaussian with covariance matrix $\mathbf{R}^{(m)}$.

Then we can define an ensemble of model forecasts obtained using Equation (5) for each ensemble or particle member as follows,

$$\mathbf{X}^{f,(m)} = \left[\mathbf{x}_1^{f,(m)}, \mathbf{x}_2^{f,(m)}, \ldots, \mathbf{x}_{N_e}^{f,(m)}\right] \in \mathcal{R}^{N_x \times N_e}, \quad (7)$$

where superscript $(.)^f$ stands for *forecast*.

The aim of the stochastic data assimilation methods is to produce a posterior pdf or analysis distribution of the state, $\mathbf{X}^a$, at the time of the observations through combining the ensemble model forecast $\mathbf{X}^f$ with observations $\mathbf{y}$. In Section 5, we will discuss ensemble Kalman filter based methods and in Sections 6–8 we will discuss particle, second-order exact, and adaptive Gaussian mixture filter methods all achieving this aim through different approaches.

5. Ensemble Kalman filters

Given an initial ensemble $\mathbf{X}^{(0)} \in \mathcal{R}^{N_x \times N_e}$, the different proposed variants of the ensemble Kalman filter have the following steps in common:

- **Forecast step:** the ensemble members at each time step between the observations $0 < k \leq m$ are propagated using the full non-linear dynamical model:

$$\mathbf{x}_j^{f,(k)} = \mathcal{M}_k\left(\mathbf{x}_j^{f,(k-1)}\right) + \boldsymbol{\beta}_j^{(k)}, \quad (8)$$

  starting at the previous analysis ensemble (if $k = 1$, then this would be $\mathbf{x}_j^{f,(1)} = \mathcal{M}_1\left(\mathbf{x}_j^{(0)}\right) + \boldsymbol{\beta}_j^{(1)}$), where $j = 1, \ldots, N_e$ is the ensemble member index.
- **Analysis step:** at the observation time $k = m$ the ensemble forecast mean and covariance are updated using the available observations to obtain a new analysis ensemble.

The various ensemble methods differ in the analysis step. Here we will discuss current methods applicable for large-dimensional systems, namely, the original ensemble Kalman filter (EnKF) (Evensen, 1994) with stochastic innovations (Burgers et al., 1998; Houtekamer and Mitchell, 1998), the singular evolutive interpolated Kalman filter (SEIK) (Pham et al., 1998a), the error-subspace statistical estimation (ESSE) (Lermusiaux and Robinson, 1999; Lermusiaux et al., 2002; Lermusiaux, 2007), the ensemble transform Kalman filter (ETKF) (Bishop et al., 2001), the ensemble adjustment Kalman filter (EAKF) (Anderson, 2001), the original ensemble square root filter (EnSRF) (Whitaker and Hamill, 2002) with synchronous and serial observation treatment, the square root formulation of the EnKF (Evensen, 2003), the error subspace transform Kalman filter (ESTKF) (Nerger et al., 2012a), and the maximum likelihood ensemble filter (MLEF) (Zupanski, 2005; Zupanski et al., 2008). We will present these methods in the square root form and point out the different ways the analysis ensemble is obtained in each of the methods. Tippett et al. (2003) gives a uniform framework for EnSRFs, which we follow closely here. In the rest of this section for ease of notation we omit the time index $(\cdot)^{(k)}$ since all of the analysis operations are done at time m.

The ensemble methods discussed in this section are based on the Kalman filter (Kalman, 1960) where the updated ensemble mean follows the Kalman update for the state, given by

$$\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^f + \mathbf{K}\left(\mathbf{y} - \mathcal{H}(\overline{\mathbf{x}}^f)\right) = \overline{\mathbf{x}}^f + \mathbf{K}\mathbf{d} \quad (9)$$

where $\mathbf{d} = \mathbf{y} - \mathcal{H}(\overline{\mathbf{x}}^f)$ is the *innovation*. The ensemble covariance update follows the covariance update equation in the Kalman Filter, given by

$$\mathbf{P}^a = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}^f, \quad (10)$$

where $\mathbf{K}$ is the Kalman gain given by

$$\mathbf{K} = \mathbf{P}^f\mathbf{H}^{\mathsf{T}}(\mathbf{H}\mathbf{P}^f\mathbf{H}^{\mathsf{T}} + \mathbf{R})^{-1}. \quad (11)$$

The matrix $\mathbf{H}$ is the linearised observation operator $\mathcal{H}(..)$ at the forecast mean $\overline{\mathbf{x}}^f$. Initially the Kalman filter was derived for a linear observation operator, but in the Extended Kalman Filter the non-linear observation operator is used as above.

Since for high-dimensional systems it is computationally not feasible to form the error covariance matrix $\mathbf{P}$, the analysis update of the covariance matrix in Equation (10) is formulated in a square root form by computing a transform matrix and applying it to the ensemble perturbation matrix, which is a scaled square root of $\mathbf{P}$. That is, the analysis ensemble is then given by

$$\mathbf{X}^a = \overline{\mathbf{X}}^a + \mathbf{X}'^a, \quad (12)$$

where $\overline{\mathbf{X}}^a = (\overline{\mathbf{x}}^a, \ldots, \overline{\mathbf{x}}^a) \in \mathcal{R}^{N_x \times N_e}$ is a matrix with the ensemble analysis mean in each column and the ensemble analysis perturbations are a scaled matrix square root of

$$\mathbf{P}^a = \frac{\mathbf{X}'^a\left(\mathbf{X}'^a\right)^{\mathsf{T}}}{N_e - 1}. \quad (13)$$

To obtain the general square root form we write, using (10)

$$\begin{aligned}\mathbf{X}'^{a}(\mathbf{X}'^{a})^{\mathsf{T}} &= \left(\mathbf{I}-\mathbf{P}^{f}\mathbf{H}^{\mathsf{T}}(\mathbf{H}\mathbf{P}^{f}\mathbf{H}^{\mathsf{T}}+\mathbf{R})^{-1}\mathbf{H}\right)\mathbf{X}'^{f}(\mathbf{X}'^{f})^{\mathsf{T}}\\ &= \mathbf{X}'^{f}\left(\mathbf{I}-\mathbf{S}^{\mathsf{T}}\mathbf{F}^{-1}\mathbf{S}\right)(\mathbf{X}'^{f})^{\mathsf{T}},\end{aligned} \tag{14}$$

where $\mathbf{S}=\mathbf{H}\mathbf{X}'^{f}$ is the ensemble perturbation matrix in observation space and

$$\mathbf{F} = \mathbf{S}\,\mathbf{S}^{\mathsf{T}} + (N_e - 1)\,\mathbf{R} \tag{15}$$

is the innovation covariance.

It is possible to use a slightly different way to calculate matrix $\mathbf{S}$ using the non-linear observation operator as $\mathbf{S}=\mathcal{H}(\mathbf{X}^{f})-\mathcal{H}(\overline{\mathbf{X}}^{f})$, in which, with a slight abuse of notation, $\mathcal{H}(\mathbf{X}^{f}) = \left(\mathcal{H}(\mathbf{x}_1^{f}),\cdots,\mathcal{H}(\mathbf{x}_{N_e}^{f})\right)$, and similarly for $\mathcal{H}(\overline{\mathbf{X}}^{f})$. This can be used in any of the ensemble Kalman filters discussed below.

To find the updated ensemble analysis perturbations $\mathbf{X}'^{a}$ we need to compute the square root $\mathbf{T}$ of the matrix

$$\mathbf{I}-\mathbf{S}^{\mathsf{T}}\mathbf{F}^{-1}\mathbf{S}=\mathbf{T}\mathbf{T}^{\mathsf{T}}, \tag{16}$$

where $\mathbf{T}$ is called a *transform matrix*. Different ways exist to compute the transform matrix $\mathbf{T}$ and here we will discuss the current methods applicable to large-dimensional systems.

For the ensemble-based Kalman filters presented in this paper we can write the analysis update as linear transformations using a weight vector $\overline{\mathbf{w}}$ for the ensemble mean and a weight matrix $\mathbf{W}'$ for the ensemble perturbations as

$$\overline{\mathbf{x}}^{a} = \overline{\mathbf{x}}^{f} + \mathbf{X}'^{f}\overline{\mathbf{w}}, \tag{17}$$

$$\mathbf{X}'^{a} = \mathbf{X}'^{f}\mathbf{W}'. \tag{18}$$

Notice, that the ensemble analysis perturbation matrix, $\mathbf{X}'^{a}$, in Equation (18) has a zero mean by construction. Further, we note that for most of the methods discussed in this section, matrix $\mathbf{W}'$ is the transformation matrix $\mathbf{T}$ in Equation (16). However, this is not the case for EnKF, SEnKF and MLEF. Further, we can compute the analysis ensemble directly by

$$\mathbf{X}^{a} = \overline{\mathbf{X}}^{f} + \mathbf{X}'^{f}\left(\overline{\mathbf{W}}+\mathbf{W}'\right), \tag{19}$$

where $\overline{\mathbf{W}} = \left(\overline{\mathbf{w}},\ldots,\overline{\mathbf{w}}\right)$. In the sections below we will derive the weight matrices for each of the ensemble-based Kalman filter methods we discuss. The updated ensemble can then be obtained using Equation (19).

To aid simplicity in discussing the different methods we use the same letter for the variables with the same meaning, i.e. $\mathbf{W}'$ is always the perturbation analysis transform matrix that transforms $\mathbf{X}'^{f}$ into $\mathbf{X}'^{a}$. Clearly, such variables do not necessarily have the same values for the various methods listed below. Thus, we subscript these variables common to all methods with a specific letter for each method. This letter is underlined in the title of each subsection that follows here, e.g. for EnKF we use $\mathbf{W}'_{N}$. Note that some of the variables can have the same values for different methods, though. At the end of this section we will provide a table of the common variables with their dimensions and whether they are equal to the same variable in a different method.

5.1. *The Stochastic Ensemble Kalman filter (EnKF)*

The Stochastic EnKF was introduced at the same time by Burgers et al. (1998) and Houtekamer and Mitchell (1998). It is a modified version of the original under-dispersive EnKF as introduced by Evensen (1994) by adding measurement noise to the innovations so that the filter maintains the correct spread in the analysis ensemble and prevents filter divergence.

Although the scheme was initially interpreted as perturbing observations, a more consistent interpretation is that the predicted observations are perturbed with the observation noise. The reason for this is that it doesn't make sense to perturb observations since they already contain measurement noise (errors), e.g. from measuring instruments, and thus have already departed from the true state of the system. Also, Bayes Theorem, see Section 2 tells us that we need the probabilities of the states given this set of observations, not a perturbed set. The idea is that each ensemble member is statistically equivalent to the true state of the system, and the true observation is a perturbed measurement of the true state. So to compare that observation with the predicted observations the latter have to be perturbed with the measurement noise too to make this comparison meaningful. This reasoning is identical to that used in rank histograms in which observations are ranked in the perturbed predicted observations from the ensemble to be statistically equivalent.

Each ensemble member individually is explicitly corrected using the Kalman filter equations, and hence the square root form is implicit only as the transform matrix and its square root are never explicitly computed. In contrast to the other filters, the stochastic EnKF perturbs the predicted observations by forming a matrix

$$\mathbf{Y}^{f} = \left(\mathcal{H}(\mathbf{x}_1^{f}),\mathcal{H}(\mathbf{x}_2^{f}),\ldots,\mathcal{H}(\mathbf{x}_{N_e}^{f})\right)+\mathbf{Y}' \in \mathcal{R}^{N_y\times N_e}, \tag{20}$$

where the observational noise (perturbation) matrix $\mathbf{Y}'$ is given by:

$$\mathbf{Y}' = \left(\boldsymbol{\epsilon}_1,\boldsymbol{\epsilon}_2,\ldots,\boldsymbol{\epsilon}_{N_e}\right) \in \mathcal{R}^{N_y\times N_e} \tag{21}$$

with the noise vectors $\boldsymbol{\epsilon}_j$ drawn from a Gaussian distribution with mean zero and covariance $\mathbf{R}$. We also introduce the

observation matrix $\mathbf{Y} = (\mathbf{y}, \mathbf{y}, \ldots, \mathbf{y}) \in \mathcal{R}^{N_y \times N_e}$ consisting of N_e identical copies of the observation vector.

The Stochastic EnKF uses the matrix $\mathbf{F}$ defined in Equation (15) with prescribed matrix $\mathbf{R}$ and proceeds by transforming all ensemble members according to

$$\mathbf{X}^a = \mathbf{X}^f + \frac{1}{N_e - 1}\mathbf{X'}^f \mathbf{S}^\mathsf{T} \mathbf{F}_N^{-1} \left(\mathbf{Y} - \mathbf{Y}^f\right), \tag{22}$$

Similar to the Equations (17)–(19), this can be written as

$$\mathbf{X}^a = \mathbf{X}^f + \mathbf{X'}^f \mathbf{W'}_N \tag{23}$$

with

$$\mathbf{W'}_N = \frac{1}{N_e - 1}\mathbf{S}^\mathsf{T} \mathbf{F}_N^{-1} \left(\mathbf{Y} - \mathbf{Y}^f\right). \tag{24}$$

Due to the use of the observation ensemble $\mathbf{Y}$ no explicit transformation of the ensemble mean needs to be performed.

Algorithm 4 in Appendix 2 gives a pseudo-algorithm of the EnKF method.

We note that while the above description of the stochastic EnKF is widely accepted and implemented, it does produce the correct posterior covariance only in a statistical sense due to extra sampling errors while the ensemble mean is not affected by the sampling error by ensuring that observation noise matrix, $\mathbf{Y'}$, has zero mean. However, in the limit of infinite ensemble size and when all sources of error (both observation and model) are correctly sampled, the stochastic EnKF does produce the correct posterior covariance (Whitaker and Hamill, 2002).

5.2. *The singular evolutive interpolated Kalman filter (SEIK)*

The SEIK filter (Pham et al., 1998b Pham, 2001) was the first filter method that allowed for non-linear model evolution and that was explicitly formulated in square root form. The filter uses the Sherman–Morrison–Woodbury identity (Golub and Van Loan, 1996) to rewrite $\mathbf{TT}^\mathsf{T}$ (Equation 16) as

$$\mathbf{TT}^\mathsf{T} = \mathbf{I} - \mathbf{S}^\mathsf{T}\mathbf{F}^{-1}\mathbf{S} = \left(\mathbf{I} + \frac{1}{N_e - 1}\mathbf{S}^\mathsf{T}\mathbf{R}^{-1}\mathbf{S}\right)^{-1}. \tag{25}$$

Note, that the performance of this scheme depends on whether the product of the inverse of the observation error matrix, $\mathbf{R}^{-1}$, and a given vector can be efficiently computed, which is for instance the case when we assume that the observation errors are uncorrelated.

The SEIK filter computes the analysis step in the ensemble error subspace. This is achieved by defining a matrix

$$\mathbf{L}_E = \mathbf{X}^f \mathbf{A}_E, \tag{26}$$

where $\mathbf{A}_E \in \mathcal{R}^{N_e \times (N_e - 1)}$ is a matrix with full rank and zero column sums. Commonly, matrix $\mathbf{A}_E$ is identified as

$$\mathbf{A}_E = \begin{bmatrix} \mathbf{I}_{N_e - 1 \times N_e - 1} \\ \mathbf{0}_{1 \times N_e - 1} \end{bmatrix} - \frac{1}{N_e}\left[\mathbf{1}_{N_e \times N_e - 1}\right], \tag{27}$$

where $\mathbf{0}$ is a matrix whose elements are equal to zero and $\mathbf{1}$ is a matrix whose elements are equal to one (Pham et al., 1998b). Matrix $\mathbf{A}_E$ implicitly subtracts the ensemble mean when the matrix $\mathbf{L}$ is computed. In addition, $\mathbf{A}_E$ removes the last column of $\mathbf{X'}^f$. Thus, $\mathbf{L}$ is an $N_e \times N_e - 1$ matrix that holds the first $N_e - 1$ ensemble perturbations. The product of the square root matrices in the ensemble error space becomes now

$$\mathbf{T}_E\mathbf{T}_E^\mathsf{T} = \left(\mathbf{A}_E^\mathsf{T}\mathbf{A}_E + \frac{1}{N_e - 1}(\mathbf{HL}_E)^\mathsf{T}\mathbf{R}^{-1}(\mathbf{HL}_E)\right)^{-1}. \tag{28}$$

The matrix $\mathbf{T}_E\mathbf{T}_E^\mathsf{T}$ is of size $N_e - 1 \times N_e - 1$. The square root $\mathbf{T}_E$ is obtained from the Cholesky decomposition of $(\mathbf{T}_E\mathbf{T}_E^\mathsf{T})^{-1}$. Then, the ensemble transformation weight matrices in Equations (17)–(19) are given by

$$\mathbf{W'}_E = \mathbf{A}_E\mathbf{T}_E\mathbf{\Omega}, \tag{29}$$

$$\overline{\mathbf{w}}_E = \frac{1}{\sqrt{N_e - 1}}\mathbf{A}_E\mathbf{T}_E\mathbf{T}_E^\mathsf{T}(\mathbf{HL}_E)^\mathsf{T}\mathbf{R}^{-1}\mathbf{d}. \tag{30}$$

Here, the columns of $\mathbf{\Omega} \in \mathcal{R}^{N_e - 1 \times N_e}$ are orthonormal and orthogonal to the vector $(1, \ldots, 1)^\mathsf{T}$. $\mathbf{\Omega}$ can be either random or a deterministic rotation matrix. However, if a deterministic $\mathbf{\Omega}$ is used then Nerger et al. (2012a) shows that a symmetric square root of $\mathbf{T}_E\mathbf{T}_E^\mathsf{T}$ should be used for a more stable ensemble.

Algorithm 5 in Appendix 2 gives a pseudo-algorithm of the SEIK method.

5.3. *The error-subspace statistical estimation (ESSE)*

The ESSE (Lermusiaux and Robinson, 1999 Lermusiaux et al., 2002 Lermusiaux, 2007) method is based on evolving an error subspace of variable size, that spans and tracks the scales and processes where the dominant errors occur (Lermusiaux et al., 2002). Here, we follow the formulation of Lermusiaux (2007) adapted to the unified notation used here.

The consideration of an evolving error subspace is analogous to the motivation of the SEIK filter. The main difference to other subspace filters mentioned here is how the ensemble matrix is truncated. That is, the full ensemble perturbation matrix $\mathbf{X'}^f$ at the current analysis time with columns

$$\mathbf{x'}_j^f = \mathbf{x}_j^f - \overline{\mathbf{x}}^f, \quad j = 1, \ldots, N_e$$

is approximated by the fastest growing singular vectors. The full ensemble perturbation matrix is decomposed using the reduced or thin singular value decomposition (SVD), (e.g. p. 72, Golub and Van Loan, 1996),

$$\mathbf{U}_S \mathbf{\Sigma}_S \mathbf{V}_S^\mathsf{T} = \mathbf{X'}^f, \tag{31}$$

where $\mathbf{U}_S \in \mathcal{R}^{N_x \times N_e}$ is an orthogonal matrix of left singular vectors, $\mathbf{\Sigma}_S \in \mathcal{R}^{N_e \times N_e}$ is a diagonal matrix with singular values on the diagonal, and $\mathbf{V}_S^\mathsf{T} \in \mathcal{R}^{N_e \times N_e}$ is an orthogonal matrix of right singular vectors of $\mathbf{X'}^f$. Next, normalised eigenvalues are computed via

$$\mathbf{E}_S = \frac{1}{N_e - 1} \mathbf{\Sigma}_S^2. \tag{32}$$

The matrices $\mathbf{U}_S$ and $\mathbf{E}_S$ are truncated to the leading eigenvalues. Using $\tilde{\mathbf{U}}_S, \tilde{\mathbf{E}}_S$ with rank $\tilde{N}_e \leq N_e$ and $\hat{\mathbf{U}}_S, \hat{\mathbf{E}}_S$ with rank $\hat{p} < \tilde{N}_e$ where the similarity coefficient ρ is computed via

$$\rho = \frac{\mathrm{Tr}\left(\hat{\mathbf{E}}_S^{\frac{1}{2}} \hat{\mathbf{U}}_S^\mathsf{T} \tilde{\mathbf{U}}_S \tilde{\mathbf{E}}_S^{\frac{1}{2}}\right)}{\mathrm{Tr}\left(\tilde{\mathbf{E}}_S\right)} \tag{33}$$

and Tr(.) is the trace of a matrix. ρ measures the similarity between two subspaces of different sizes. The process of reducing the subspace is repeated until ρ is close to one, i.e. $\rho > \alpha$ where $1 - \epsilon \leq \alpha \leq 1$ is a user selected scalar limit.[3] The dimension of the error subspace thus varies with time and in accord with model dynamics (Lermusiaux, 2007). Hence, in the following analysis update the reduced rank approximations

$$\tilde{\mathbf{U}}_S \approx \mathbf{U}_S, \tag{34}$$
$$\tilde{\mathbf{E}}_S \approx \mathbf{E}_S, \tag{35}$$
$$\tilde{\mathbf{V}}_S \approx \mathbf{V}_S \tag{36}$$

are used where the right singular vector matrix $\tilde{\mathbf{V}}_S$ is also truncated to have size $\tilde{N}_e \times \tilde{N}_e$.

The product of the square root matrices, using Equation (14), in the error subspace becomes

$$\mathbf{T}_S \mathbf{T}_S^\mathsf{T} = \mathbf{I} - \tilde{\mathbf{S}}^\mathsf{T} \tilde{\mathbf{F}}^{-1} \tilde{\mathbf{S}} \tag{37}$$

where ensemble errors in observation space are given by $\tilde{\mathbf{S}} = \mathbf{H}\tilde{\mathbf{U}}_S \tilde{\mathbf{E}}_S^{\frac{1}{2}} \tilde{\mathbf{V}}_S^T$ and innovation covariances by $\tilde{\mathbf{F}} = \mathbf{H}\tilde{\mathbf{U}}_S \tilde{\mathbf{E}} \tilde{\mathbf{U}}^\mathsf{T} \mathbf{H}^\mathsf{T} + \mathbf{R}$.

The inverse of the $N_y \times N_y$-matrix $\tilde{\mathbf{F}}$ is obtained by performing the eigenvalue decomposition (EVD)

$$\tilde{\mathbf{F}} = \mathbf{\Gamma} \mathbf{\Lambda} \mathbf{\Gamma}^\mathsf{T} \tag{38}$$

so that Equation (37) becomes

$$\mathbf{T}_S \mathbf{T}_S^\mathsf{T} = \mathbf{I} - \tilde{\mathbf{S}}^\mathsf{T} \mathbf{\Gamma} \mathbf{\Lambda}^{-1} \mathbf{\Gamma}^\mathsf{T} \tilde{\mathbf{S}}. \tag{39}$$

Performing another EVD in Equation (39),

$$\mathbf{T}_S \mathbf{T}_S^\mathsf{T} = \mathbf{Z} \mathbf{\Pi} \mathbf{Z}^\mathsf{T}, \tag{40}$$

the symmetric square root becomes

$$\mathbf{T}_S = \mathbf{Z} \mathbf{\Pi}^{\frac{1}{2}} \mathbf{Z}^T. \tag{41}$$

Hence, the ensemble transformation weight matrices needed to form the ensemble analysis mean and analysis perturbations in Equations (17)–(19) are given by

$$\mathbf{W'}_S^a = \mathbf{Z} \mathbf{\Pi}^{\frac{1}{2}} \mathbf{Z}^\mathsf{T} \tag{42}$$
$$\overline{\mathbf{w}}_S^a = \frac{1}{\sqrt{N_e - 1}} \tilde{\mathbf{S}} \tilde{\mathbf{F}}^{-1} \mathbf{d}. \tag{43}$$

Note, that when computing the analysis ensemble mean and perturbations, the truncated ensemble perturbation matrix $\tilde{\mathbf{X}}'^f$ is used in the pseudo-algorithm 6 in Appendix 2. The truncation to the rank $\tilde{N}_e$ will results in a reduction of the ensemble size. To avoid that the ensemble size shrinks, Lermusiaux (2007) described an optional adaptive method to generate new ensemble members.

5.4. *The ensemble transform Kalman filter (ETKF)*

The ETKF (Bishop et al., 2001) was derived to explicitly transform the ensemble in a way that results in the correct spread of the analysis ensemble. As the SEIK filter, the ETKF uses the Morrison-Woodbury identity to write

$$\mathbf{T}_T \mathbf{T}_T^\mathsf{T} = \left(\mathbf{I} + \frac{1}{N_e - 1} \mathbf{S}^\mathsf{T} \mathbf{R}^{-1} \mathbf{S}\right)^{-1}. \tag{44}$$

In contrast to the SEIK filter, $\mathbf{T}_T \mathbf{T}_T^\mathsf{T}$ is of size $N_e \times N_e$ and hence represents the error-subspace of dimension $N_e - 1$ indirectly by the full ensemble.

Currently, the most widespread method to compute the update in the ETKF appears to be the formulation of the LETKF by Hunt et al. (2007), which we describe here. By performing the EVD of the symmetric matrix $(\mathbf{T}_T \mathbf{T}_T^\mathsf{T})^{-1} = \mathbf{U}_T \mathbf{\Sigma}_T \mathbf{U}_T^\mathsf{T}$ we obtain the symmetric square root

$$\mathbf{T}_T = \mathbf{U}_T \mathbf{\Sigma}_T^{-\frac{1}{2}} \mathbf{U}_T^\mathsf{T}. \tag{45}$$

Using this decomposition, the ensemble transformation weight matrices needed to form the ensemble analysis mean and analysis perturbations in Equations (17)–(19) are given by

$$\mathbf{W}'_T = \mathbf{U}_T \mathbf{\Sigma}_T^{-\frac{1}{2}} \mathbf{U}_T^{\mathsf{T}}, \tag{46}$$

$$\overline{\mathbf{w}}_T = \frac{1}{\sqrt{(N_e - 1)}} \mathbf{U}_T \mathbf{\Sigma}_T^{-1} \mathbf{U}_T^{\mathsf{T}} (\mathbf{X}'^f)^{\mathsf{T}} \mathbf{H}^{\mathsf{T}} \mathbf{R}^{-1} \mathbf{d}. \tag{47}$$

Using the symmetric square root produces a transform matrix which is closest to the identity matrix in the Frobenius norm (Hunt et al., 2007). Thus, the ETKF results in a minimum transform in the ensemble space, which is different from the notion of 'optimal transportation' used in the ETPF (see Section 6.3).

The original publication introducing the ETKF (Bishop et al., 2001) did not specify the form of the matrix square root $\mathbf{T}_T$. There are different possibilities to compute it, and taking a simple single-sided square root could lead to implementations with a biased transformation, such that the transformation by $\mathbf{W}'$ would not preserve the ensemble mean. However, using the symmetric square root approach this bias is avoided. Livings (2005) proposed another variant normalising first the forecast observation ensemble perturbation matrix so that the observations are dimensionless with standard deviation one

$$\tilde{\mathbf{S}} = \frac{1}{\sqrt{N_e - 1}} \mathbf{R}^{-\frac{1}{2}} \mathbf{S}. \tag{48}$$

Substituting (48) into (44) gives

$$\mathbf{T}_T \mathbf{T}_T^{\mathsf{T}} = \left(\mathbf{I} + \tilde{\mathbf{S}}^{\mathsf{T}} \tilde{\mathbf{S}} \right)^{-1}. \tag{49}$$

To find the square root form next we perform the SVD

$$\tilde{\mathbf{S}}^{\mathsf{T}} = \mathbf{U}_T \tilde{\mathbf{\Sigma}}_T \tilde{\mathbf{V}}_T^{\mathsf{T}}. \tag{50}$$

In this case, the ensemble transformation weight matrices in Equations (17)–(19) become

$$\mathbf{W}'_T = \mathbf{U}_T \left(\mathbf{I} + \tilde{\mathbf{\Sigma}}_T \tilde{\mathbf{\Sigma}}_T^{\mathsf{T}} \right)^{-\frac{1}{2}} \mathbf{U}_T^{\mathsf{T}}, \tag{51}$$

$$\overline{\mathbf{w}}_T = \frac{1}{\sqrt{N_e - 1}} \mathbf{U}_T (\mathbf{I} + \tilde{\mathbf{\Sigma}}^{\mathsf{T}} \tilde{\mathbf{\Sigma}}_T^{\mathsf{T}})^{-1} \tilde{\mathbf{\Sigma}}_T \tilde{\mathbf{V}}_T^{\mathsf{T}} \mathbf{R}^{-\frac{1}{2}} \mathbf{d}.s \tag{52}$$

This formulation avoids the multiplication $\mathbf{S}^{\mathsf{T}} \mathbf{R}^{-1} \mathbf{S}$ and can hence prevent possible loss of accuracy due to rounding errors. However, this formulation also requires the computation of the square root of $\mathbf{R}$, which itself can result in rounding errors if $\mathbf{R}$ is not diagonal.

Algorithm 7 in Appendix 2 gives a pseudo-algorithm of the ETKF method.

5.5. *The ensemble adjustment Kalman filter (EAKF)*

The EAKF was introduced by Anderson (2001). Similarly to the SEIK filter and the ETKF we require here that the matrix $\mathbf{R}^{-1}$ is readily available. Using scaled ensemble perturbations as discussed for the ETKF-formulation by Livings (2005) in Equations (48)–(49) we can write

$$\mathbf{T}_A \mathbf{T}_A^{\mathsf{T}} = \left(\mathbf{I} + \tilde{\mathbf{S}}^{\mathsf{T}} \tilde{\mathbf{S}} \right)^{-1}. \tag{53}$$

We perform the SVD on the scaled forecast ensemble observation perturbation matrix

$$\tilde{\mathbf{S}}^{\mathsf{T}} = \mathbf{U}_A \mathbf{\Sigma}_A \mathbf{V}_A^{\mathsf{T}}. \tag{54}$$

Note that $\mathbf{U}_A = \mathbf{U}_T$, related to the similarity between the EAKF and the ETKF. We also use an EVD to obtain

$$\mathbf{P}^f = \mathbf{Z}_A \mathbf{\Gamma}_A \mathbf{Z}_A^{\mathsf{T}}. \tag{55}$$

The decomposition in Equation (55) is usually performed as an SVD of the ensemble perturbation matrix $\mathbf{X}'^f$, which approximates $\mathbf{P}^f$ using N_e ensemble members.

Due to the ranks of the matrices decomposed in Equations (54) and (55) there are at most $q = \min(N_e - 1, N_y)$ non-zero singular values in $\mathbf{\Sigma}_A$ and at most $N_e - 1$ non-zero eigenvalues in $\mathbf{\Gamma}_A$. Thus, the matrices in the equations below can be truncated as follows: $\mathbf{U}_A \in \mathcal{R}^{N_e \times q}$, $\mathbf{\Sigma}_A \in \mathcal{R}^{q \times q}$, $\mathbf{V}_A^{\mathsf{T}} \in \mathcal{R}^{q \times N_y}$ and $\mathbf{\Gamma}_A, \mathbf{Z}_A \in \mathcal{R}^{N_e - 1 \times N_e - 1}$. Then, the ensemble transformation weight matrices in Equations (17)–(19) are given by

$$\mathbf{W}'_A = \mathbf{U}_A \left(\mathbf{I} + \mathbf{\Sigma}_A \mathbf{\Sigma}_A^{\mathsf{T}} \right)^{-\frac{1}{2}} \mathbf{\Gamma}_A^{-\frac{1}{2}} \mathbf{Z}_A^{\mathsf{T}} \mathbf{X}'^f, \tag{56}$$

$$\overline{\mathbf{w}}_A = \frac{1}{\sqrt{N_e - 1}} \mathbf{U}_A (\mathbf{I} + \mathbf{\Sigma}_A^{\mathsf{T}} \mathbf{\Sigma}_A)^{-1} \mathbf{\Sigma}_T \mathbf{V}_A^{\mathsf{T}} \mathbf{R}^{-\frac{1}{2}} \mathbf{d}. \tag{57}$$

Note, that the EAKF perturbation weight matrix in Equation (56) is the same as applying the orthogonal matrix $\mathbf{\Gamma}_A^{-\frac{1}{2}} \mathbf{Z}_A^{\mathsf{T}} \mathbf{X}'^f$ instead of the orthogonal matrix $\mathbf{U}_T$ in the ETKF perturbation transform matrix given by Equation (51) (Tippett et al., 2003).

The decomposition in Equation (55) is costly due to the size of the matrix to be decomposed. For this reason, the EAKF is typically applied with serial observation processing as will be described for the EnSRF in Section 5.7.

Algorithm 8 in Appendix 2 gives a pseudo-algorithm of the EAKF method.

5.6. The ensemble square root filter (EnSRF)

The EnSRF was introduced by Whitaker and Hamill (2002) to avoid the use of perturbed observations by a square root formulation. In the EnSRF the transform matrix is given by

$$\mathbf{T}_R\mathbf{T}_R^{\mathsf{T}} = \mathbf{I} - \mathbf{S}^{\mathsf{T}}\mathbf{F}^{-1}\mathbf{S}.$$

We first perform an EVD of $\mathbf{F}$ to obtain its inverse

$$\mathbf{F}^{-1} = \mathbf{\Gamma}_R\mathbf{\Lambda}_R^{-1}\mathbf{\Gamma}_R^{\mathsf{T}}. \tag{58}$$

Then, we can write the ensemble analysis covariance as

$$\begin{aligned}\mathbf{X}'^a(\mathbf{X}'^a)^{\mathsf{T}} &= \mathbf{X}'^f\left(\mathbf{I} - \mathbf{S}^{\mathsf{T}}\mathbf{\Gamma}_R\mathbf{\Lambda}_R^{-1}\mathbf{\Gamma}_R^{\mathsf{T}}\mathbf{S}\right)(\mathbf{X}'^f)^{\mathsf{T}} \\ &= \mathbf{X}'^f\left(\mathbf{I} - \mathbf{G}_R\mathbf{G}_R^{\mathsf{T}}\right)(\mathbf{X}'^f)^{\mathsf{T}}\end{aligned} \tag{59}$$

where $\mathbf{G}_R = \mathbf{S}^{\mathsf{T}}\mathbf{\Gamma}_R\mathbf{\Lambda}_R^{-\frac{1}{2}}$. Decomposing $\mathbf{G}_R = \mathbf{U}_R\mathbf{\Sigma}_R\mathbf{V}_R^{\mathsf{T}}$ using an SVD we obtain

$$\begin{aligned}\mathbf{X}'^a(\mathbf{X}'^a)^{\mathsf{T}} &= \mathbf{X}'^f\left(\mathbf{I} - \left[\mathbf{U}_R\mathbf{\Sigma}_R\mathbf{V}_R^{\mathsf{T}}\right]\left[\mathbf{U}_R\mathbf{\Sigma}_R\mathbf{V}_R^{\mathsf{T}}\right]^{\mathsf{T}}\right)(\mathbf{X}'^f)^{\mathsf{T}} \\ &= \mathbf{X}'^f\mathbf{U}_R\left(\mathbf{I} - \mathbf{\Sigma}_R\mathbf{\Sigma}_R^{\mathsf{T}}\right)\mathbf{U}_R^{\mathsf{T}}(\mathbf{X}'^f)^{\mathsf{T}}.\end{aligned}$$

The diagonal matrix holding the singular values is of dimension $\mathbf{\Sigma}_R \in \mathcal{R}^{N_e\times N_y}$ and has thus at most $min(N_e, N_y)$ nonzero singular values. To reduce the computational cost for the case of high dimensional models with $N_e \ll N_y$, we can truncate to get the much smaller matrix $\mathbf{\Sigma}_R \in \mathcal{R}^{N_e\times min(N_e,N_y)}$ (see Table 1). The square root form for the ensemble analysis perturbations is given by

$$\mathbf{X}'^a = \mathbf{X}'^f\mathbf{U}_R\left(\mathbf{I} - \mathbf{\Sigma}_R\mathbf{\Sigma}_R^{\mathsf{T}}\right)^{\frac{1}{2}}, \tag{60}$$

and the ensemble transformation weight matrices needed to form the ensemble analysis mean and analysis perturbations in Equations (17)–(19) are given by

$$\mathbf{W}'_R = \mathbf{U}_R\left(\mathbf{I} - \mathbf{\Sigma}_R\mathbf{\Sigma}_R^{\mathsf{T}}\right)^{\frac{1}{2}}\mathbf{U}_R^{\mathsf{T}}, \tag{61}$$

$$\overline{\mathbf{w}}_R = \mathbf{S}^{\mathsf{T}}\mathbf{\Gamma}_R\mathbf{\Lambda}_R^{-1}\mathbf{\Gamma}_R^{\mathsf{T}}\mathbf{d}, \tag{62}$$

where in Equation (61) we have post-multiplied the ensemble analysis perturbations by the orthogonal matrix of the left singular vectors $\mathbf{U}_R^{\mathsf{T}}$ to ensure that the analysis ensemble is unbiased (Livings et al., 2008; Sakov and Oke, 2008).

Algorithm 9 in Appendix 2 gives a pseudo-algorithm of the EnSRF method.

5.7. EnSRF with serial observation treatment

The serial observation treatment in the EnSRF was introduced by Whitaker and Hamill (2002) together with the EnSRF assimilating all observations at once. The serial treatment reduces the computing cost. Hence, the EnSRF is typically not applied with the bulk update described above, but with serial treatment of observations, which is possible if $\mathbf{R}$ is diagonal. In this case, each single observation can be assimilated separately. Thus, $\mathbf{F}$ reduces to the scalar F and $\mathbf{S}\mathbf{S}^{\mathsf{T}}$ to the scalar S^2. For a single observation ($N_y = 1$), the matrix $\mathbf{G}_R$ becomes a vector given by

$$\mathbf{G}_R = \frac{1}{\sqrt{F}}\mathbf{S}^{\mathsf{T}}. \tag{63}$$

All singular values of $\mathbf{G}_R$ are zero except the first, which is its norm,

$$\mathbf{\Sigma}_R = \frac{S}{\sqrt{F}}\mathbf{e} \tag{64}$$

where $\mathbf{e}$ is a vector with N_e zero elements except the first, which is one. The first column of $\mathbf{U}_R$ corresponds to the normalised vector $\mathbf{S}^{\mathsf{T}}$

$$\mathbf{U}_R\mathbf{e} = \frac{1}{S}\mathbf{S}^{\mathsf{T}}. \tag{65}$$

The square root of the diagonal matrix in Equation (61) can be written as a sum of the identity matrix and a matrix proportional to $\mathbf{e}\mathbf{e}^{\mathsf{T}}$:

$$\left(\mathbf{I} - \mathbf{\Sigma}_R\mathbf{\Sigma}_R^{\mathsf{T}}\right)^{\frac{1}{2}} = \mathbf{I} - \left(1 - \sqrt{(N_e-1)R/F}\right)\mathbf{e}\mathbf{e}^{\mathsf{T}}. \tag{66}$$

Using Equation (65) and the fact that all columns of $\mathbf{U}$ are orthonormal, one obtains

$$\mathbf{W}'_R = \mathbf{I} - \frac{1 - \sqrt{(N_e-1)R/F}}{S^2}\mathbf{S}^{\mathsf{T}}\mathbf{S} \tag{67}$$

and the weight vector for the update of the ensemble mean is

$$\overline{\mathbf{w}}_R = \frac{1}{F}\mathbf{S}^{\mathsf{T}}\mathbf{d}. \tag{68}$$

The equations above are then applied in a series over each single observation. The equations are likewise valid when the EAKF is formulated with a serial observation treatment.

Algorithm 10 in Appendix 2 gives a pseudo-algorithm of the EnSRF method with serial observation treatment.

5.8. The square root formulation of the stochastic ensemble Kalman filter (SEnKF)

The SEnKF was introduced by Evensen (2003) as a square root formulation of the stochastic EnKF. Defining $\mathbf{Y}'$ as for the stochastic EnKF and using a matrix

$$\mathbf{F}_F = \mathbf{S}\mathbf{S}^\mathsf{T} + \mathbf{Y}'\mathbf{Y}'^\mathsf{T} \tag{69}$$

we obtain the matrix

$$\mathbf{T}_F\mathbf{T}_F^\mathsf{T} = \mathbf{I} - \mathbf{S}^\mathsf{T}\mathbf{F}_F^{-1}\mathbf{S}. \tag{70}$$

We could decompose $\mathbf{F}_F$ using an EVD but this is costly if $N_y \gg N_e$ (Evensen, 2003). Instead, we assume that forecast and observation errors are uncorrelated, i.e.

$$\mathbf{S}\mathbf{Y}'^\mathsf{T} \equiv 0, \tag{71}$$

so that

$$\mathbf{F}_F = \mathbf{S}\mathbf{S}^\mathsf{T} + \mathbf{Y}'\mathbf{Y}'^\mathsf{T} = (\mathbf{S}+\mathbf{Y}')(\mathbf{S}+\mathbf{Y}')^\mathsf{T}. \tag{72}$$

Now we can use an SVD to decompose $\mathbf{S}+\mathbf{Y}' = \mathbf{U}_F\mathbf{\Sigma}_F\mathbf{V}_F^\mathsf{T}$, giving

$$\mathbf{F}_F = \mathbf{U}_F\mathbf{\Sigma}_F\mathbf{\Sigma}_F^\mathsf{T}\mathbf{U}_F^\mathsf{T}, \tag{73}$$

which has a much smaller computational cost than decomposing $\mathbf{F}_F$ using an EVD when $N_y \gg N_e$.

The ensemble transformation is then computed according to Equation (23) with the weight matrix given by

$$\mathbf{W}'_F = \mathbf{S}^\mathsf{T}\mathbf{U}_F\mathbf{\Sigma}_F^{-1}(\mathbf{\Sigma}_F^{-1})^\mathsf{T}\mathbf{U}_F^\mathsf{T}(\mathbf{Y}-\mathbf{Y}^f). \tag{74}$$

Algorithm 11 in Appendix 2 gives a pseudo-algorithm of the EnKF in square root form.

5.9. The error-subspace transform Kalman filter (ESTKF)

The ESTKF has been derived from the SEIK filter (Nerger et al., 2012a) by combining the advantages of the SEIK filter and the ETKF. The ESTKF exhibits better properties than the SEIK filter, like a minimum ensemble transformation as the ETKF. However, unlike the ETKF, the ESTKF computes the ensemble transformation in the error subspace spanned by the ensemble rather than using the ensemble representation of it. That is, the error subspace of the dimension N_e-1 is represented directly in the ESTKF (similarly to the SEIK filter) while in the ETKF the error subspace is represented indirectly using the full ensemble of size N_e.

Similar to the SEIK filter, a projection matrix $\mathbf{A}_K \in \mathcal{R}^{N_e\times N_e-1}$ is used whose elements are defined by

$$\mathbf{A}_{K\{i,j\}} := \begin{cases} 1-\frac{1}{N_e}\frac{1}{\frac{1}{\sqrt{N_e}}+1} & \text{for } i=j, i<N_e \\ -\frac{1}{N_e}\frac{1}{\frac{1}{\sqrt{N_e}}+1} & \text{for } i\neq j, i<N_e \\ -\frac{1}{\sqrt{N_e}} & \text{for } i=N_e \end{cases} \tag{75}$$

With this projection, the basis vectors for the error subspace are given by

$$\mathbf{L}_K = \mathbf{X}^f\mathbf{A}_K. \tag{76}$$

As for the matrix $\mathbf{\Omega}$ in the SEIK filter, the columns of matrix $\mathbf{A}_K$ are orthonormal and orthogonal to the vector $(1,\ldots,1)^\mathsf{T}$. When the matrix $\mathbf{L}_K$ is computed, the multiplication with $\mathbf{A}_K$ implicitly subtracts the ensemble mean. Further, $\mathbf{A}_K$ subtracts a fraction of the last column of $\mathbf{X}'^f$ from all other columns. In this way, the last column of $\mathbf{X}'^f$ is not just dropped as in the SEIK filter, but its information is distributed over the other columns. The product of the square root matrices in the error subspace becomes now

$$\mathbf{T}_K\mathbf{T}_K^\mathsf{T} = \left(\mathbf{I} + \frac{1}{N_e-1}(\mathbf{H}\mathbf{L}_K)^\mathsf{T}\mathbf{R}^{-1}(\mathbf{H}\mathbf{L}_K)\right)^{-1}. \tag{77}$$

By performing the EVD of the symmetric matrix $(\mathbf{T}_K\mathbf{T}_K^\mathsf{T})^{-1} = \mathbf{U}_K\mathbf{\Sigma}_K\mathbf{U}_K^\mathsf{T}$ we obtain the symmetric square root

$$\mathbf{T}_K = \mathbf{U}_K\mathbf{\Sigma}_K^{-\frac{1}{2}}\mathbf{U}_K^\mathsf{T}. \tag{78}$$

Then, the ensemble transformation weight matrices needed to form the ensemble analysis mean and perturbations in Equations (17)–(19) are given by

$$\mathbf{W}'_K = \mathbf{A}_K\mathbf{T}_K\mathbf{A}_K^\mathsf{T}, \tag{79}$$

$$\overline{\mathbf{w}}_K = \frac{1}{\sqrt{N_e-1}}\mathbf{A}_K\mathbf{U}_K\mathbf{\Sigma}_K^{-1}\mathbf{U}_K^\mathsf{T}(\mathbf{H}\mathbf{L}_K)^\mathsf{T}\mathbf{R}^{-1}\mathbf{d}. \tag{80}$$

Compared to the SEIK filter, both the matrices $\mathbf{A}_E$ and $\mathbf{\Omega}$ are replaced by $\mathbf{A}_K$ in the ESTKF. In addition, the ESTKF uses the symmetric square root of $\mathbf{T}_K\mathbf{T}_K^\mathsf{T}$. The use of $\mathbf{A}_K$ leads to a consistent projection onto the error subspace and back onto the state space, while the symmetric square root ensures that the minimum transformation is obtained. It is also possible to apply the ESTKF with a random ensemble transformation. For this case, the rightmost matrix $\mathbf{A}_K$ in Equation (79) is replaced by a random matrix with the same properties as the deterministic $\mathbf{A}_K$.

Algorithm 12 in Appendix 2 gives a pseudo-algorithm of the ESTKF method.

5.10. The Maximum Likelihood Ensemble Filter (MLEF)

The MLEF (Zupanski, 2005; Zupanski et al., 2008) calculates the state estimate as the maximum of the posterior probability density (pdf) function. This is in contrast to the ensemble Kalman filter methods described in this paper, which are based on the minimum variance approach, so targeting the mean. The maximum of the pdf is found by an iterative minimisation of the cost function using a generalised non-linear conjugate-gradient method.

The original MLEF filter (Zupanski, 2005) uses a second-order Taylor approximation to the analysis increments, which requires that the cost function is twice differentiable. However, this requirement is not necessarily satisfied in many real life non-linear applications, for example where the parameterisation of some processes is used in models or for strongly non-linear observation operators. Here, we present the revised MLEF by Zupanski et al. (2008) that avoids this requirement by using a non-differentiable minimisation algorithm.

In contrast to all other ensemble filters discussed above, the MLEF maintains the actual state estimate separately from the ensemble, which is used to provide the measurement of estimation error. Thus, the analysis perturbations in the MLEF are computed for each ensemble member in a square root form without re-centring them onto the analysis ensemble mean. Hence, this filter does not follow the same square root form as the filters described above and is presented last in this section.

In the MLEF, the ensemble analysis perturbations are defined using the difference between analysis and forecast for each ensemble member, $\mathbf{x}'^a_j = \mathbf{x}^a_j - \mathbf{x}^f_j$ and not between ensemble analysis states and the analysis mean. They are found using generalised Hessian preconditioning in state space. A change of variable is performed as follows

$$\mathbf{x}^a_j = \mathbf{x}^f_j + \mathbf{x}'^a_j, \tag{81}$$

$$\mathbf{x}'^a_j = \mathbf{G}^{\frac{1}{2}} \boldsymbol{\xi}_j, \tag{82}$$

where the matrix $\mathbf{G}^{\frac{1}{2}} = \mathbf{X}'^f \left(\mathbf{I} + \mathbf{C}\right)^{-\frac{1}{2}} \in \mathcal{R}^{N_x \times N_e}$ represents the inverse square root of the generalised Hessian estimated at the initial point of minimisation, and $\boldsymbol{\xi}_j$ is a control variable defined in ensemble subspace. Matrix $\mathbf{C}$ is the covariance matrix

$$\mathbf{C} = (\mathbf{X}'^f)^{\mathsf{T}} \mathbf{H}^{\mathsf{T}} \mathbf{R}^{-1} \mathbf{H} \mathbf{X}'^f. \tag{83}$$

Equation (82) can be written as a transformation of ensemble perturbations by

$$\mathbf{x}'^a_j = \mathbf{X}'^f \mathbf{w}_{M,j} \tag{84}$$

where the elements of the weight vector $\mathbf{w}_{M,j} \in \mathcal{R}^{N_e}$ for ensemble member j are given by

$$\mathbf{w}_{M,j} = \left(\mathbf{I} + \mathbf{C}\right)^{-\frac{1}{2}} \boldsymbol{\xi}_j. \tag{85}$$

Now we use an EVD of $\mathbf{C} = \boldsymbol{\Gamma}_M \boldsymbol{\Lambda}_M \boldsymbol{\Gamma}_M^{\mathsf{T}}$ to write Equation (85) as

$$\mathbf{w}_{M,j} = \boldsymbol{\Gamma}_M \left(\mathbf{I} + \boldsymbol{\Lambda}_M\right)^{-\frac{1}{2}} \boldsymbol{\Gamma}_M^{\mathsf{T}} \boldsymbol{\xi}_j. \tag{86}$$

We note, that in a linear case matrix $\mathbf{G}^{\frac{1}{2}}$ is a square root of $\mathbf{P}^a$. Indeed the same decomposition and inversion was used to find the square root analysis perturbations for the ETKF, see Equation (45).

After successfully accomplishing the Hessian preconditioning, the next step in the iterative minimisation is to calculate the gradient in the ensemble-spanned subspace. The preconditioned generalised gradient at the k-th minimisation iteration is obtained by

$$\nabla_G J(\mathbf{x}_k) = \left[\mathbf{I} + \mathbf{C}\right]^{-1} \boldsymbol{\xi}_k - \mathbf{Z}(\mathbf{x}_k)^{\mathsf{T}} \mathbf{R}^{-\frac{1}{2}} \left[\mathbf{y} - \mathcal{H}(\mathbf{x}_k)\right], \tag{87}$$

where

$$\mathbf{Z}(\mathbf{x}) = [\mathbf{z}_1(\mathbf{x}), \mathbf{z}_2(\mathbf{x}), \ldots, \mathbf{z}_{N_e}(\mathbf{x})] \tag{88}$$

$$\mathbf{z}_j(\mathbf{x}) = \mathbf{R}^{-\frac{1}{2}} [\mathcal{H}(\mathbf{x} + \mathbf{x}'^f_j) - \mathcal{H}(\mathbf{x})]. \tag{89}$$

Upon convergence we have obtained an optimal state analysis

$$\mathbf{x}^a = \mathbf{x}_k.$$

To complete the non-differential formulation of the MLEF, ensemble analysis perturbations are computed as follows

$$\mathbf{X}'^a = \mathbf{X}'^f \left[\mathbf{I} + \left(\mathbf{Z}(\mathbf{x}^a)\right)^{\mathsf{T}} \mathbf{Z}(\mathbf{x}^a)\right]^{-\frac{1}{2}} \tag{90}$$

where $\mathbf{Z}(\mathbf{x}^a)$ is obtained by substituting $\mathbf{x} = \mathbf{x}^a$ into Equation (89).

Algorithm 13 in Appendix 2 gives a pseudo-algorithm of the MLEF method.

5.11. Summary of ensemble Kalman Filter methods

In this section, we have described ten most popular ensemble Kalman filter methods that are applicable to high-dimensional non-Gaussian problems.

This collection of methods could be categorised in different ways, for example in deterministic ensemble filters, where the

analysis is found through explicit mathematical transformations (SEIK, ETKF, EAKF, EnSRF, ESTKF, MLEF), and stochastic ensemble filters, where perturbed forecasted observations are used (EnKF, SEnKF). Burgers et al. (1998) and Houtekamer and Mitchell (1998) showed that in order to maintain sufficient spread in the ensemble and prevent filter divergence, the observations should be treated as random variables, i.e. perturbed, while our interpretation is slightly different, as described above. This stochasticity, of course, leads to extra sampling noise in the filters. On the other hand, Lawson and Hansen (2004) showed that for large ensemble sizes, stochastic filters can handle non-linearity better than the deterministic filters. This is due to the additional Gaussian observation spread normalising the ensemble update in the stochastic filter, which tends to erase the non-Gaussian higher moments non-linear error growth has generated. However, current computational power restricts us to small ensemble sizes for high-dimensional problems, in which case stochastic filters add another source of sampling error thus underestimating the analysis update (Whitaker and Hamill, 2002).

While all ensemble Kalman filter methods use low-rank approximations of the state error covariance matrix, some of the methods in this section are referred to as error-subspace ensemble filters because they directly operate in the error subspace spanned by ensemble rather than using the ensemble representation of it. Such filters are SEIK (see Section 5.2), ESTKF (see Section 5.9), and ESSE (see Section 5.3). Nerger et al. (2005b) compares the stochastic EnKF with the SEIK filter in an idealised high-dimensional shallow water model with non-linear evolution, showing that the main difference between the filters lies in the efficiency of the representation of the covariance matrix $\mathbf{P}$. In general, the EnKF filter will require a larger ensemble size N_e to achieve the same performance as the SEIK filter. The relation of the ETKF and SEIK methods has been studied by Nerger et al. (2012a), where also the ESTKF has been derived. Apart from computing the ensemble transformation in the error subspace in case of SEIK and ESKTF, the three filters are essentially equivalent. However, for the SEIK filter it has been found that the application of the Cholesky decomposition can lead to unevenly distributed variance in the ensemble members. To this end, the ESTKF and ETKF method are preferable unless a random matrix $\mathbf{\Omega}$ is used in the SEIK filter.

The methods described in this section each have nuances in which they differ one from another as well as underlying common ground. Writing these methods using unified mathematics notation allows us to see these algorithmic differences and commonalities more readily. Many filters described above have several common variables and while for some methods the variables have different sizes, others can not only have the same size but actually the same value, too. Table 1 summarises the sizes of the common variables between the methods and below we comment on whether they have the same value.

Apart from the matrices listed in Table 1, there are the finally resulting weight matrices $\mathbf{W}'$ and $\overline{\mathbf{W}}$. The matrix $\overline{\mathbf{W}}$ is identical for all filters. Thus, for all filters except EnKF, SEnKF and MLEF, which don't use this matrix, the mean of the analysis ensemble is identical. The EnKF and SEnKF are an exception because they do not explicitly transform the ensemble mean and introduce sampling error due to the perturbed observations. The maximum likelihood approach of the MLEF also results in a different analysis state estimate if the ensemble distribution is not Gaussian and the observation operator is non-linear. Further the square $\mathbf{W}'\mathbf{W}'^T$ is identical for all filters except the EnKF, SEnKF and MLEF. Thus, the analysis covariance matrix $\mathbf{P}^a$ will be identical.

In contrast to the equality of the matrix $\overline{\mathbf{W}}$, the matrix $\mathbf{W}'$ is different for almost all methods. Thus, while many methods yield the same analysis ensemble covariance matrix, their ensemble perturbations are distinct. In the ETKF and ESTKF methods, the dimensions of the matrices $\mathbf{T}$, $\mathbf{U}$, and $\mathbf{\Sigma}$ have distinct dimensions. However, the ensemble transformation weight matrices $\mathbf{W}'$ of both methods are identical (Nerger et al., 2012a).

In general, the choice of using a particular ensemble method depends on a number of the system's components: the dynamical model at hand, model error, number and types of observations, ensemble size. Given these degrees of freedom it is not possible to attribute one data assimilation method to be better suited for a general situation. In practise operational meteorological applications most widely use the LETKF and the serial formulations of the EnSRF and EAKF, while in oceanography there are many applications of the stochastic EnKF, the SEIK filter, and the ESTKF. There are less applications of the ESSE and MLEF, despite the fact that these filters are algorithmically interesting because of the ensemble-size adaptivity of ESSE and the maximum-likelihood solution MLEF. From the algorithmic viewpoint, the stochastic EnKF will be useful if the stochasticity can be an advantage and if large ensembles can be used. Further, the filters SEIK, ETKF, ESTKF differ from the EnKF, EnSRF and EAKF also in the application of distinct localisation methods (see Section 9.1 for the discussion of localisation). EnKF, EnSRF and EAKF allow for localisation in the state space, which could be advantageous for some observation types (Campbell et al., 2010). The serial formulation of the EnSRF and EAKF requires that the observation error covariance matrix is diagonal. Thus, these filters cannot directly be applied if the observation errors are correlated. A transformation into variables that are uncorrelated is possible in theory, but it is most likely not practical for large sets of observations.

6. Particle filters

In this section we will consider the standard particle filter followed by three efficient variants of the particle filters: the Equivalent Weights Particle Filter (EWPF, van Leeuwen, 2010;

Table 1: Overview of the sizes of matrices that are used in different filter methods.

Variable	EnKF	SEIK	ESSE	ETKF	EAKF	EnSRF	SEnKF	ESTKF
$\mathbf{T}$	$N_e \times N_e$	$N_e - 1 \times N_e - 1$	$\tilde{N}_e \times \tilde{N}_e$	$N_e \times N_e$	$N_e \times N_e$	$N_e \times N_e$	$N_e \times N_e$	$N_e - 1 \times N_e - 1$
$\mathbf{U}$	–	–	$N_x \times N_e$	$N_e \times N_e$	$N_e \times N_e$	$N_e \times N_e$	$N_e \times N_e$	$N_e - 1 \times N_e - 1$
$\mathbf{\Sigma}$	–	–	$N_e \times N_e$	$N_e \times N_e$	$N_e \times N_y$	$N_e \times min(N_e, N_y)$	$N_e \times N_y$	$N_e - 1 \times N_e - 1$
$\mathbf{V}$	$N_y \times N_y$	–	$N_e \times N_e$	–	$N_y \times N_y$	$\min(N_e, N_y) \times N_y$	–	–
$\mathbf{L}$	–	$N_x \times N_e - 1$	–	–	–	–	–	$N_x \times N_e - 1$
$\mathbf{A}$	–	$N_e \times N_e - 1$	–	–	–	–	–	$N_e \times N_e - 1$

van Leeuwen, 2011; Ades and van Leeuwen, 2013), the Implicit Equal-Weights Particle Filter (Zhu et al., 2016) and the Ensemble Transform Particle Filter (Reich, 2013). Other variants of local particle filters are discussed in Section 9 on practical implementation. Another interesting particle filter for high-dimensional systems, the so called implicit particle filter (Chorin and Tu, 2009; Chorin et al., 2010; Morzfeld et al., 2012; van Leeuwen et al., 2015), is not discussed here as it needs a 4D-Var-like minimisation for each particle. The Multivariate Rank Histogram Filter (MRHF, Metref et al., 2014b), based on the Rank-Histogram Filter of Anderson (2010) that performs well in highly non-Gaussian regimes, has been recently developed in the European project Sangoma.[4] However, it is still under development for high-dimensional systems and its idea is only shortly described in Section 9.1.5. Often particle filters are defined as providing approximations of $p\left(\mathbf{x}^{(0:m)}|\mathbf{y}^{(1:m)}\right)$, but we restrict ourselves to particle filters that are approximations of the marginal posterior pdf $p\left(\mathbf{x}^{(m)}|\mathbf{y}^{(1:m)}\right)$ as there are at present no efficient algorithms for the former for high-dimensional geophysical systems, and we have forecasting in mind. Furthermore, for ease of presentation we take all earlier observations for granted, leading to the marginal posterior at time m being denoted as $p\left(\mathbf{x}^{(m)}|\mathbf{y}^{(m)}\right)$.

6.1. *The standard particle filter*

This particle filter is also known as the bootstrap filter or Sequential Importance Resampling (SIR). The probability distribution function (pdf) in particle filtering, represented by N_e particles or ensemble members at time k, is given by

$$p\left(\mathbf{x}^{(m)}\right) = \frac{1}{N_e}\sum_{j=1}^{N_e}\delta\left(\mathbf{x}^{(m)} - \mathbf{x}_j^{(m)}\right), \tag{91}$$

where $\mathbf{x}^{(m)} \in \mathcal{R}^{N_x}$ is the N_x-dimensional state of the system that has been integrated forward in time using the stochastic forward model and $\delta(x)$ is a Dirac-delta function. We let time m to be the time of a current set of observations with the previous observation set at time 0. Then the stochastic forward model for times $0 < k < m$ for each particle $j = 1, \ldots, N_e$ is given by

$$\mathbf{x}_j^{(k)} = \mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right) + \boldsymbol{\beta}_j^{(k)}, \tag{92}$$

where $\boldsymbol{\beta}_j^{(k)} \in \mathcal{R}^{N_x}$ are random terms representing the Gaussian distributed model errors with mean zero and covariance matrix $\mathbf{Q}$, and $\mathcal{M}_k : \mathcal{R}^{N_x} \rightarrow \mathcal{R}^{N_x}$ is the deterministic model from time $k-1$ to k. Thus, the model state transition from time $k-1$ to k is fully described by the transition density given by

$$p\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}\right) = \mathcal{N}\left(\mathcal{M}_k\left(\mathbf{x}^{(k-1)}\right), \mathbf{Q}\right), \tag{93}$$

which will be of later use.

Using Bayes theorem

$$p\left(\mathbf{x}_j^{(m)}|\mathbf{y}^{(m)}\right) = \frac{p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m)}\right)}{p\left(\mathbf{y}^{(m)}\right)} p\left(\mathbf{x}_j^{(m)}\right) \tag{94}$$

and the Markovian property of the model, the full posterior at observation time m is written as

$$p\left(\mathbf{x}_j^{(m)}|\mathbf{y}^{(m)}\right) = \sum_{j=1}^{N_e} w_j^{(m)}\delta\left(\mathbf{x}^{(m)} - \mathbf{x}_j^{(m)}\right) \tag{95}$$

where the weights $w_j^{(m)}$ are given by

$$w_j^{(m)} \propto p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m)}\right) p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}\right) w_j^{(m-1)} \tag{96}$$

and each $w_j^{(m-1)}$ is the product of all the weights from all time steps $0 < k \leq m-1$. The conditional pdf $p\left(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}\right)$ is the pdf of the observations given the model state $\mathbf{x}^{(m)}$ which is often taken to be Gaussian

$$p\left(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}\right) \propto \exp\left[-\frac{1}{2}\left(\mathbf{y}^{(m)} - \mathcal{H}_m\left(\mathbf{x}^{(m)}\right)\right)^{\mathsf{T}} \mathbf{R}^{-1}\left(\mathbf{y}^{(m)} - \mathcal{H}_m\left(\mathbf{x}^{(m)}\right)\right)\right]. \tag{97}$$

To obtain equal-weight posterior particles one applies resampling, in which particles with high weights are duplicated, while particles with low weights are abandoned. Several schemes have been developed to perform resampling, and three of the most-used schemes are presented in Appendix 1.

The problem in high-dimensional spaces with a large number of independent observations is that these weights vary enormously over the particles, with one particle obtaining a weight close to one, while all the others have a weight very close to zero. This is the so-called degeneracy problem related to the 'curse of dimensionality': any resampling scheme will produce N_e copies of the particle with the highest weight, and all variation in the ensemble has disappeared.

Hence, as mentioned at the beginning of this section, to apply a particle filter to a high-dimensional system additional information is needed to limit the search space of the filter. One option is to use localisation directly on the standard particle filter. Local particle filters, like the so-called Local Particle Filter (Poterjoy, 2016a) will be discussed in the section on localisation in particle filters. We next discuss the proposal-density particle filters since this technique could be applied to all filters and permits us to achieve equal-weights for the particles in a different way.

6.2. *Proposal-density particle filters*

To avoid that the ensemble degenerates we aim at ensuring that equally significant particles are picked from the posterior density. To do this we have to ensure that all particles end up in the high-probability area of the posterior pdf, and that they have very similar, or even equal, weights. For the former we can use a scheme that pulls the particles towards the observations. Several methods can be used for this, including traditional methods like 4DVar, a variational method, and ensemble Kalman filters and smoothers. However, the main ingredient in efficient particle filters is the step that ensures that the weights of the different particles are close before any resampling step.

We start by writing the prior at time m as follows:

$$p\left(\mathbf{x}^{(m)}\right) = \int p\left(\mathbf{x}^{(m)}|\mathbf{x}^{(m-1)}\right) p\left(\mathbf{x}^{(m-1)}\right) d\mathbf{x}^{(m-1)}. \quad (98)$$

Without loss of generality but for simplicity we assume that the particle weights in the ensemble at the previous time step $m-1$ are equal, so

$$p\left(\mathbf{x}^{(m-1)}\right) = \frac{1}{N_e}\sum_{j=1}^{N_e} \delta\left(\mathbf{x}^{(m-1)} - \mathbf{x}_j^{(m-1)}\right). \quad (99)$$

Using Equation (99) in Equation (98) leads directly to:

$$p\left(\mathbf{x}^{(m)}\right) = \frac{1}{N_e}\sum_{j=1}^{N_e} p\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)}\right), \quad (100)$$

hence, from Equation (93) the prior can be seen as a mixture density, with each density centred around one of the forecast particles.

One can now multiply the numerator and denominator of Equation (100) by the same factor $q\left(x^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)}, \mathbf{y}^{(m)}\right)$, in which $\mathbf{x}_{1:N_e}^{(m-1)}$ is defined as the collection of all particles at time $m-1$, and the conditioning on j denotes that each particle does in general has a different parent to start from. This leads to

$$p\left(\mathbf{x}^{(m)}\right) = \frac{1}{N_e}\sum_{j=1}^{N_e} \frac{p\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)}\right)}{q\left(\mathbf{x}^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)}, \mathbf{y}^{(m)}\right)} q\left(\mathbf{x}^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)}, \mathbf{y}^{(m)}\right) \quad (101)$$

where $q\left(\mathbf{x}^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)}, \mathbf{y}^{(m)}\right)$ is the so-called *proposal transition density*, or *proposal* for short, whose support should be equal to or larger than that of $p\left(\mathbf{x}^{(m)}|\mathbf{x}^{(m-1)}\right)$. Note that the proposal density as formulated here is slightly more general than the usual $q\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{(m)}\right)$ through allowing for the explicit dependence on all particles at time $m-1$.

Drawing from this density we find for the posterior:

$$\begin{aligned} p\left(\mathbf{x}^{(m)}|\mathbf{y}^{(m)}\right) &= \frac{p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m)}\right)}{p\left(\mathbf{y}^{(m)}\right)} p\left(\mathbf{x}^{(m)}\right) \\ &= \frac{1}{N_e}\sum_{j=1}^{N_e} w_j \delta\left(\mathbf{x}^{(m)} - \mathbf{x}_j^{(m)}\right) \end{aligned} \quad (102)$$

where w_j are the particle weights given by

$$w_j = \frac{p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m)}\right)}{p\left(\mathbf{y}^{(m)}\right)} \frac{p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}\right)}{q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)}, \mathbf{y}^{(m)}\right)}. \quad (103)$$

Using Bayes' theorem, the numerator in the expression for the weights can be expressed as

$$\begin{aligned} &p\left(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}\right) p\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)}\right) \\ &\quad = p\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{(m)}\right) p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right) \end{aligned} \quad (104)$$

Therefore, the particle weight of ensemble member j can be written as:

$$w_j = \frac{p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right)}{p\left(\mathbf{y}^{(m)}\right)} \frac{p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{(m)}\right)}{q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)}, \mathbf{y}^{(m)}\right)}. \quad (105)$$

In the so-called optimal proposal density (Doucet et al., 2000) one chooses

$$q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)},\mathbf{y}^{(m)}\right)=p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)},\mathbf{y}^{(m)}\right),$$

leading to weights $w_j \propto p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right)$. For systems with a large number of independent observations these weights are again degenerate (see, e.g. Snyder et al., 2008; Ades and van Leeuwen, 2013; Snyder et al., 2015).

Several efficient particle filter schemes have been developed utilising the proposal density to avoid this degeneracy. Here we discuss the Equivalent-Weights Particle Filter (EWPF) and the Implicit Equal-Weights Particle Filter (IEWPF). As mentioned, the Implicit Particle Filter (Chorin et al., 2010), which allows for an extension of the one-time-step optimal proposal particle filter to a full time window explores a 4DVar-like method on each particle. Since it needs an adjoint of the underlying model, it is not discussed in this paper.

6.2.1. The equivalent-weights particle filter. The EWPF works as follows:

(1) Determine the optimal proposal weight $w_j \propto p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right)$ for each particle. Note that these weights vary enormously in high-dimensional systems.
(2) Choose a target weight w_{target} based on these weights that a certain percentage of particles can reach. For instance, if the target weight is set to the lowest of these weights we keep 100% of the particles. A choice of 50% will mean that the target weight is set to the medium value of these weights.
(3) Calculate the position in state space of each particle such that it has a weight exactly equal to the target weight. This is where the proposal density comes in. Note that some of the particles cannot reach this target weight no matter how we move them, and these are brought back into the ensemble via the resampling step in point 5.
(4) Add a small random perturbation to each particle and recalculate its weight.
(5) Resample *all* particles such that their weights are equal again.

It is in step 3 that we use the fact that the proposal density is dependent on all previous particles, and not just particle j. This step is the main reason for the efficiency of the filter.

As an example, when the error in the model equations is additive Gaussian and the observation operator is linear an analytical solution can be found for the maximum weight for each particle j, or actually, the minimum of minus the log of that weight called ϕ_j:

$$\phi_j=\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)^T\left(\mathbf{HQH}^T+\mathbf{R}\right)^{-1}$$
$$\times\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right). \quad (106)$$

Then a target weight is set from these ϕ_j's. The target weight splits the ensemble of particles in two groups: those particles that have a higher optimal proposal weight, and those with a lower optimal proposal weight. The latter are abandoned at this point, and will be regenerated in the resampling step 5.

For the retained particles, there is an infinite number of ways to move a particle in state space such that it reaches the target weight. In the EWPF that problem is solved by assuming

$$\hat{\mathbf{x}}_j^{(m)}=\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)+\alpha_j\boldsymbol{\Upsilon}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right) \quad (107)$$

in which α_j is a scalar, and $\boldsymbol{\Upsilon}$ is defined as

$$\boldsymbol{\Upsilon}=\mathbf{QH}^T\left(\mathbf{HQH}^T+\mathbf{R}\right)^{-1}. \quad (108)$$

Under this assumption the number of solutions is reduced to two, and the two values for α_j are given by

$$\alpha_j=1\pm\sqrt{1-b_j/a_j} \quad (109)$$

in which

$$a_j=\frac{1}{2}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)^T$$
$$\times\mathbf{R}^{-1}\mathbf{H}^T\boldsymbol{\Upsilon}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right) \quad (110)$$

and

$$b_j=\frac{1}{2}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)^T$$
$$\times\mathbf{R}^{-1}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)+$$
$$\log w_{\text{target}}-\log w_j^{(m-1)}, \quad (111)$$

in which $w_j^{(m-1)}$ is the weight of particle j accumulated over previous time steps, included here for completeness. Note that w_{target} is the target weight selected from ϕ's in Equation (106) (e.g. if we choose to keep 80% particles $-\log(w_{\text{target}}) = \{\tilde{\phi}_j\}_{j=\lceil N_e*0.8\rceil}$ where $\{\tilde{\phi}_j\}_{j=1,\ldots,N_e}$ is a sorted list of optimal-proposal weight of each particle) and that $\alpha_j = 1$ pushes the particle to its optimal-proposal weight position. The solution resembles the optimal proposal solution in which the deterministic part of the proposal is scaled to ensure equal weights. Also note the resemblance of the deterministic part with the shape of that used in a Kalman filter when we replace Q with the ensemble covariance of the state.

When the number of independent observations is large the optimal proposal density particle filter is degenerate, meaning that one particle gets a much larger weight than all the others. The EWPF is not degenerate because a set percentage of all particles has a similar weight (before the resampling step). The EWPF does not, however, converge for large N_e to the posterior pdf because of this equivalent-weights construction, in which high-weight particles are moved such that their weight becomes lower, equal to the target weight. So the scheme is biased. However, the large N_e limit is not that relevant in practise as the affordable number of particles will be low, below say 10,000, and typically of $O(20-100)$. In that setting, the Monte-Carlo error will be substantial, and the bias should be measured against the Monte-Carlo error. As long as the latter is larger than the former the scheme is a valid alternative in high-dimensional systems.

Algorithm 16 in Appendix 2 gives a pseudo-algorithm for the EWPF.

6.2.2. The implicit equal-weights particle filter. This scheme is very similar to that of the EWPF:

(1) Determine the optimal proposal weight $w_j \propto p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right)$ for each particle. Note that these weights vary enormously in high-dimensional systems.
(2) Choose a target weight based on these weights that a certain percentage of particles can reach. Typically the target weight is chosen as the minimum of the maximal weights, so that all particles are kept.
(3) Draw a random perturbation vector for each particle, and add this to the particle position that leads to maximal weight. So far the scheme is the same as that used in the optimal proposal density.
(4) Scale each random vector such that each particle will reach the target weight.
(5) Resample the particles such that their weights are equal in case the kept percentage is lower than 100%.

The main difference between this scheme and the EWPF is that in the EWPF we scale the deterministic part of the optimal proposal to reach a target weight, while here we scale the random part of the optimal proposal.

The implicit part of our scheme follows from drawing samples implicitly from a standard Gaussian distributed proposal density $q(\boldsymbol{\xi})$ instead of the original one $q\left(\mathbf{x}^{(m)}|\mathbf{x}^{(m-1)},\mathbf{y}^{(m)}\right)$, as in (Chorin and Tu, 2009). These two pdfs are related by:

$$q\left(\mathbf{x}^{(m)}|\mathbf{x}_{1:N_e}^{(m-1)},\mathbf{y}^{(m)}\right)=\frac{q\left(\boldsymbol{\xi}_j\right)}{\left\|\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}_j}\right\|} \tag{112}$$

where $\left\|\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}_j}\right\|$ denotes the absolute value of the determinant of the Jacobian matrix of the $\mathcal{R}^{N_x}\to\mathcal{R}^{N_x}$ transformation $\mathbf{x}_j=g_j(\boldsymbol{\xi}_j)$. The transformation $g_j(.)$ is now defined via the following implicit relation between variable $\mathbf{x}_j^{(m)}$ and $\boldsymbol{\xi}$ as

$$\mathbf{x}_j^{(m)}=\mathbf{x}_j^a+\alpha_j^{1/2}\mathbf{P}^{1/2}\boldsymbol{\xi}_j^{(m)} \tag{113}$$

where $\mathbf{x}_j^a$ is the mode of $p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)},\mathbf{y}^{(m)}\right)$, $\mathbf{P}$ a measure of the width of that pdf, and α_j a scalar that depends on $\boldsymbol{\xi}_j^{(m)}$.

The α_j are now chosen such that all particles get the same weight w_{target}, so the scalar α_j is determined for each particle from:

$$\begin{aligned}w_j&=\frac{p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)},\mathbf{y}^{(m)}\right)p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right)}{q\left(\boldsymbol{\xi}_j\right)}\left\|\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}_j}\right\|w_j^{(m-1)}\\&=w_{\text{target}}\end{aligned} \tag{114}$$

This ensures that the filter is not degenerate in systems with arbitrary dimensions and an arbitrary number of independent observations. Because of the target-weight construction the filter does not converge to the correct posterior pdf, and the same discussion as for the EWPF applies here, namely that as long as this bias is smaller than the Monte-Carlo error this filter is a valid candidate for high-dimensional non-linear filtering.

As an example we assume now that observation errors and model errors are Gaussian and that the observation operator $\mathbf{H}\in\mathcal{R}^{N_y\times N_x}$ is linear. Then we find that

$$\begin{aligned}&p\left(\mathbf{y}^{(m)}|\mathbf{x}^{(m)}\right)p\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)}\right)\\&=\frac{1}{A}\exp\left[-\frac{1}{2}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathbf{x}^{(m)}\right)^T\mathbf{R}^{-1}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathbf{x}^{(m)}\right)\right.\\&\left.-\frac{1}{2}\left(\mathbf{x}^{(m)}-\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)^T\mathbf{Q}^{-1}\left(\mathbf{x}^{(m)}-\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)\right]\\&=\frac{1}{A}\exp\left[-\frac{1}{2}\left(\mathbf{x}^{(m)}-\mathbf{x}_j^a\right)^T\mathbf{P}^{-1}\left(\mathbf{x}^{(m)}-\mathbf{x}_j^a\right)\right]\exp\left(-\frac{1}{2}\phi_j\right)\\&=p\left(\mathbf{x}^{(m)}|\mathbf{x}_j^{(m-1)},\mathbf{y}^{(m)}\right)p\left(\mathbf{y}^{(m)}|\mathbf{x}_j^{(m-1)}\right)\end{aligned} \tag{115}$$

where

$$\mathbf{P}=\left(\mathbf{Q}^{-1}+\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}, \tag{116}$$

$$\mathbf{x}_j^a=\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)+\boldsymbol{\Upsilon}\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right), \tag{117}$$

and

$$\begin{aligned}\phi_j=&\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right)^T\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T+\mathbf{R}\right)^{-1}\\&\times\left(\mathbf{y}^{(m)}-\mathbf{H}\mathcal{M}\left(\mathbf{x}_j^{(m-1)}\right)\right).\end{aligned} \tag{118}$$

This leads to a complicated non-linear differential equation for α_j that involves the determinant of $\mathbf{P}$. Since we are interested in high-dimensional problems we consider this equation in the limit of large state dimension N_x. In that limit it turns out that we can integrate this equation, leading to the much simpler equation (see Appendix in Zhu et al. (2016)):

$$(\alpha_j - 1)\gamma_j - N_x \log(\alpha_j) + \phi_j - \log w_j^{(m-1)} = \log w_{\text{target}}. \tag{119}$$

in which $\gamma_j = \boldsymbol{\xi}_j^T \boldsymbol{\xi}_j$. This equation could be approximated by using numerical methods, such as the Newton method, etc., but analytical solutions based on the so-called Lambert W function do exist. We do not elaborate on these here.

Algorithm 17 in Appendix 2 gives a pseudo-algorithm for the IEWPF.

6.2.3. Between observations: relaxation steps. If the system is not observed at every time step, the schemes mentioned above can be used over the time window between observations. No analytical solutions can be obtained in this case so that the solution has to be found iteratively. However, this procedure is rather expensive as it typically involves solving a problem similar to a 4DVar on each particle.[5] Thus, typically simpler schemes are employed between observation times. These schemes will be less efficient, although we can ensure that Bayes' theorem is fulfilled exactly for each particle.

In the following, we demonstrate the use of relaxation between observation times. We use the future observations to relax the particles at time k towards observations at next time $m > k$ by using instead of Equation (92) the modified forward model

$$\begin{aligned}\mathbf{x}_j^{(k)} = \mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right) + \tilde{\boldsymbol{\beta}}_j^{(k)} + \\ + \tilde{\boldsymbol{\Upsilon}}\left[\mathbf{y}^{(m)} - \mathcal{H}_k\left(\mathbf{x}_j^{(k-1)}\right)\right],\end{aligned} \tag{120}$$

where $\tilde{\boldsymbol{\beta}}_j^{(k)} \in \mathcal{R}^{N_x}$ are random terms representing the model error distributed according to a given covariance matrix $\tilde{\mathbf{Q}}$,[6] $\mathcal{M}_k$ is the same deterministic model as in Equation (92), $\tilde{\boldsymbol{\Upsilon}}$ is a relaxation matrix given by

$$\tilde{\boldsymbol{\Upsilon}} = \tau(k)\mathbf{Q}\mathbf{H}^T\mathbf{R}^{-1}. \tag{121}$$

Here, $\tau(k)$ is a time dependent scalar that determines the strength of the relaxation. $\mathbf{y}^{(m)} \in \mathcal{R}^{N_y}$ is the vector of N_y observations at time m and $\mathcal{H}_k : \mathcal{R}^{N_x} \to \mathcal{R}^{N_y}$ is the observation operator mapping model space into observation space. Note that the observations $\mathbf{y}^{(m)}$ exist at the later time $m > k$. The modified transition density is now given by

$$\begin{aligned}q\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}, \mathbf{y}^{(m)}\right) = \mathcal{N}\left(\mathcal{M}_k\left(\mathbf{x}^{(k-1)}\right)\right. \\ \left. + \tilde{\boldsymbol{\Upsilon}}\left[\mathbf{y}^{(m)} - \mathcal{H}_k\left(\mathbf{x}^{(k-1)}\right)\right], \mathbf{Q}\right),\end{aligned} \tag{122}$$

and the modified weights $w_j^{(k)}$ are accumulated as

$$\begin{aligned}w_j^{(k)} &\propto \frac{p\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}\right)}{q\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}, \mathbf{y}^{(m)}\right)} w_j^{(k-1)} \\ &\propto \prod_{t=1}^{k} \frac{p\left(\mathbf{x}_j^{(t)}|\mathbf{x}_j^{(t-1)}\right)}{q\left(\mathbf{x}_j^{(t)}|\mathbf{x}_j^{(t-1)}, \mathbf{y}^{(m)}\right)}.\end{aligned} \tag{123}$$

This simple modification of the forward model to include information about future observations using a relaxation term is only consistent with Bayes Theorem when the weights that are introduced by this modification are properly taken into account, and it leads to efficient schemes if it is used in combination with an equal-weight scheme, like the EWPF or the IEWPF. Algorithm 15 in Appendix 2 gives a pseudo-algorithm of the relaxation step used in the EWPF and the IEWPF.

Note that it would also be possible to use other methods like ensemble smoothers or ensemble 4Dvar-like methods to move particles between observations, but we will not elaborate on those here.

6.3. The Ensemble Transform Particle Filter (ETPF)

The idea of the Ensemble Transport Particle Filter (Reich, 2013) is to avoid resampling by finding a linear transportation map between the prior and the posterior ensemble such that the prior particles are minimally modified, while ensuring that the posterior particles have equal weight. We write each posterior particle as a linear combination of the prior particles as

$$\mathbf{x}_j^a = N_e \sum_{i=1}^{N_e} \mathbf{x}_i^f t_{ij} \tag{124}$$

in which we ensure that the particles have the correct mean via

$$\sum_{i=1}^{N_e} t_{ij} = \frac{1}{N_e}, \qquad \sum_{j=1}^{N_e} t_{ij} = w_i. \tag{125}$$

This still leads to $N_e^2 - 2$ undetermined elements t_{ij}. These are found by minimising the movement from old to new particles, by minimising

$$J(\mathbf{T}) = \sum_{i,j}^{N_e} t_{ij} \left\| \mathbf{x}_i^f - \mathbf{x}_j^f \right\|^2 \tag{126}$$

under the condition that $t_{ij} \geq 0$. The above formulation is an example of an optimal transportation algorithm, see e.g. the review by Chen and Reich in van Leeuwen et al. (2015). This scheme can be combined with any proposal density discussed in the previous section.

If the dynamical model is deterministic one needs to add some small random noise to the particles to avoid ensemble collapse. Typically this noise is assumed to be Gaussian with zero mean and covariance $\mathbf{P}^a = h^2\mathbf{P}^f$ with $0 < h < 1$ a free parameter. This term is an ad-hoc addition related to inflation in Ensemble Kalman Filters.

Algorithm 14 in Appendix 2 gives a pseudo-algorithm of the ETPF.

7. Second-order exact ensemble Kalman filters

Several extensions to ensemble Kalman filters have been proposed to overcome the linearity or Gaussianity assumptions. A large number of filters exists that try to bridge an ensemble Kalman filter and particle filter by defining smoothly varying parameters that move the filter between these two extremes based on the degeneracy of the particle filter. In high-dimensional systems, however, all of these filters become ensemble Kalman filters as any particle filter contribution results in complete degeneracy. These filters (not discussed here) will become useful when localisation is applied.

In a non-linear, non-Gaussian case the ensemble Kalman filters will necessarily produce an analysis where the mean and covariance are biased due to the assumption of a Gaussian prior pdf (Lei and Bickel, 2011). Here, we will discuss four ensemble filters that concentrate on getting the first two moments of the posterior distribution correct in non-linear situations. These are the Particle Filter with Gaussian Resampling of Xiong et al. (2006), the Non-linear Ensemble Transform Filter (Tödter and Ahrens, 2015), the Moment-Matching Ensemble Filter (Lei and Bickel, 2011), and the Merging Particle Filter (Nakano et al., 2007).

7.1. Particle Filter with Gaussian Resampling (PFGR) and Non-linear Ensemble Transform Kalman Filter (NETF)

The Particle Filter with Gaussian Resampling (PFGR, Xiong et al. (2006)) introduced an explicit ensemble transformation matching the mean and covariance matrix. The Non-linear Ensemble Transform Filter (NETF, Tödter and Ahrens, 2015) is a recent reinvention of this algorithm formulated to obtain an ensemble transformation that is analogous to that of the ETKF. In addition, the NETF was introduced with localisation, so that the filter can be applied to high-dimensional systems (see Section 9.1). The presentation here follows the more modern formulation of the NETF in analogy to the ETKF presented before. As a novel feature, the presented formulation avoids the explicit computation of the analysis state, that is given by the weighted ensemble mean.

The PFGR and the NETF are designed to exactly match the first two moments of the posterior pdf in Bayes theorem without assuming that the prior or likelihood are normally distributed. The forecast ensemble is transformed into an analysis ensemble by applying a weight vector to obtain the analysis mean state and a transform matrix to obtain analysis ensemble perturbations, analogous in form to a square root filter (Equations (17) to (19)).

As in most particle filters, the likelihood weights that arise from Bayes' theorem

$$w_j = \frac{p\left(\mathbf{y}|\mathbf{x}_j\right)}{\sum_{k=1}^{N_e} p\left(\mathbf{y}|\mathbf{x}_k\right)} \tag{127}$$

are used. For normally distributed observation errors, the weight of each member is at first given by

$$w_j \propto \exp\left[-\frac{1}{2}\left(\mathbf{y} - \mathcal{H}\left(\mathbf{x}_j^f\right)\right)^T \mathbf{R}^{-1}\left(\mathbf{y} - \mathcal{H}\left(\mathbf{x}_j^f\right)\right)\right] \tag{128}$$

and then normalised so that the weights sum up to one. Before the weights are computed, the ensemble perturbations should be inflated by an inflation factor $\gamma > 1$ as in the ensemble-based Kalman filters (for inflation see Section 9.2). Using the weight vector $\mathbf{w} = \left(w_1, \ldots, w_{N_e}\right)^T$ the transform matrix is

$$\mathbf{T}\mathbf{T}^T = N_e\left[\mathrm{diag}(\mathbf{w}) - \mathbf{w}\mathbf{w}^T\right]. \tag{129}$$

Here, $\mathrm{diag}(\mathbf{w})$ is a diagonal matrix that contains the weights w_j on the diagonal. The factor N_e was not present in the formulation by Xiong et al. (2006). It was introduced by Tödter and Ahrens (2015) to ensure that the ensemble has the correct analysis variance. As in the ensemble Kalman filters, the eigenvalue decomposition of $\mathbf{T}\mathbf{T}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ yields the ensemble transformation

$$\mathbf{T} = \mathbf{U}\mathbf{\Sigma}^{1/2}\mathbf{U}^T. \tag{130}$$

Combining the weight vector and transform matrix as in Equation (19), the analysis ensemble is given by

$$\mathbf{X}^a = \mathbf{X}^f\left(\mathbf{T}\mathbf{\Lambda} + [\mathbf{w}, \ldots, \mathbf{w}]\right). \tag{131}$$

Here, $\mathbf{\Lambda}$ is an random matrix. Xiong et al. (2006) use a random matrix sampled from a normal distribution with mean zero and standard deviation one. They use $\mathbf{\Lambda}$ because in Equation (130) they omit all eigenvalues that are very close to zero and need to restore an ensemble of full size. In contrast, Tödter and Ahrens (2015) use a mean-preserving orthogonal matrix (see Pham, 2001) analogous to that used in the SEIK filter. They motivate the use of $\mathbf{\Lambda}$ also by the reduction of ensemble outliers and showed experimentally that the random transformation with mean preserving properties leads to a more stable data assimilation process.

Note, that the transformation in Equation (131) is applied to the ensemble matrix $\mathbf{X}^f$ instead of the ensemble perturbation matrix $\mathbf{X'}^f$ without subsequent addition of the analysis mean state (see e.g. Equation 19). This is possible because of the property of $\mathbf{T}$ to implicitly subtract the ensemble mean, while the multiplication of $\mathbf{X}^f$ with the weight vector array adds the analysis mean state.

For high-dimensional systems, a localisation of the analysis step is required. It was introduced by Tödter and Ahrens (2015) in analogy to the localisation of the ETKF and SEIK filters (see Sec. 9.1). Algorithm 18 in Appendix 2 gives a pseudo-algorithm of the PFGR and NETF and Algorithm 22 shows the computation of the weights for Gaussian observation errors.

7.2. *Moment-Matching Ensemble Filter (MMEF)*

A stochastic algorithm that has second-order correct statistics was developed by Lei and Bickel (2011). In this moment-matching ensemble filter (MMEF) we generate an ensemble of perturbed pseudo-observations, $\mathbf{Y}^f$, as in the SEnKF (see Equations 20 and 21)

$$\mathbf{Y}_j^f \sim p_{\mathcal{H}(\mathbf{x}_j)}\left(\mathbf{y}|\mathbf{x}_j\right) \tag{132}$$

using $\mathcal{H}\left(\mathbf{x}_j\right)$ as variable in the density, so $\mathbf{y}$ is fixed. Then the analysis mean for each particle is generated using a corresponding pseudo-observation as follows

$$\bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right) = \sum_{k=1}^{N_e} w_k\left(\mathbf{y}_j^f\right)\mathbf{x}_k = \mathbf{X}^f\mathbf{w}\left(\mathbf{y}_j^f\right) \tag{133}$$

in which $w_k\left(\mathbf{y}_j^f\right)$ is given by

$$w_k\left(\mathbf{y}_j^f\right) = \frac{p\left(\mathbf{y}_j^f|\mathbf{x}_k\right)}{\sum_{l=1}^{N_e} p\left(\mathbf{y}_j^f|\mathbf{x}_l\right)}. \tag{134}$$

Similarly, the analysis mean for actual observations is computed via

$$\bar{\mathbf{x}}^a\left(\mathbf{y}\right) = \sum_{k=1}^{N_e} w_k\left(\mathbf{y}\right)\mathbf{x}_k = \mathbf{X}^f\mathbf{w}\left(\mathbf{y}\right). \tag{135}$$

Furthermore, we calculate equivalent expressions for covariances for perturbed and actual observations as follows

$$\begin{aligned}\mathbf{P}^a\left(\mathbf{y}_j^f\right) = &\sum_{k=1}^{N_e} w_k\left(\mathbf{y}_j^f\right)\left(\mathbf{x}_k - \bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right)\right)\\ &\times\left(\mathbf{x}_k - \bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right)\right)^T\end{aligned} \tag{136}$$

$$\begin{aligned}\mathbf{P}^a\left(\mathbf{y}\right) = &\sum_{k=1}^{N_e} w_k\left(\mathbf{y}\right)\left(\mathbf{x}_k - \bar{\mathbf{x}}^a\left(\mathbf{y}\right)\right)\\ &\times\left(\mathbf{x}_k - \bar{\mathbf{x}}^a\left(\mathbf{y}\right)\right)^T.\end{aligned} \tag{137}$$

Then each of the ensemble members or particles is updated via

$$\mathbf{x}_j^a = \bar{\mathbf{x}}^a\left(\mathbf{y}\right) + \mathbf{P}^a\left(\mathbf{y}\right)^{1/2}\mathbf{P}^a\left(\mathbf{y}_j^f\right)^{-1/2}\left(\mathbf{x}_j - \bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right)\right). \tag{138}$$

This filter gives the correct posterior mean and covariance in the large-ensemble limit (Lei and Bickel, 2011). To see this, note that $\mathbf{x}_j - \bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right)$ is distributed according to $N\left(0, \mathbf{P}^a\left(\mathbf{y}_j^f\right)\right)$, so $\mathbf{P}^a\left(\mathbf{y}_j^f\right)^{-1/2}\left(\mathbf{x}_j - \bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right)\right)$ is distributed $\mathcal{N}\left(0, \mathbf{I}\right)$, and hence the distribution of $\mathbf{x}_j^a$ is $\mathcal{N}\left(\bar{\mathbf{x}}^a\left(\mathbf{y}\right), \mathbf{P}^a\left(\mathbf{y}\right)\right)$.

This filter cannot be used in high-dimensional systems, even when localisation is applied, because it needs the evaluation of several full covariance matrices. However, we can explore ensemble perturbations that are used to calculate these covariances, as in all ensemble Kalman filter schemes. The following was not discussed by Lei and Bickel (2011), but is a practical way to make the filter useful in high-dimensional systems.

We can express each covariance matrix $\mathbf{P}^a\left(\mathbf{y}_j^f\right)$ directly in terms of the forecast ensemble as

$$\mathbf{P}^a\left(\mathbf{y}_j^f\right) = \mathbf{X}^f\mathbf{T}\left(\mathbf{y}_j^f\right)\mathbf{X}^{fT} \tag{139}$$

where matrix $\mathbf{T}\left(\mathbf{y}_j^f\right)$ is given by

$$\mathbf{T}\left(\mathbf{y}_j^f\right) = \mathrm{diag}\left(\mathbf{w}(\mathbf{y}_j^f)\right) - \mathbf{w}\left(\mathbf{y}_j^f\right)\mathbf{w}^T\left(\mathbf{y}_j^f\right). \tag{140}$$

The square root of this matrix is

$$\mathbf{P}^a\left(\mathbf{y}_j^f\right)^{1/2} = \mathbf{X}^f\mathbf{T}\left(\mathbf{y}_j^f\right)^{1/2}. \tag{141}$$

To find the inverse of this matrix we perform an SVD on the prior ensemble matrix

$$\mathbf{X}'^{f} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^{T} \tag{142}$$

and compute also the EVD on the much smaller square matrices $\mathbf{T}\left(\mathbf{y}_j^f\right)$

$$\mathbf{T}\left(\mathbf{y}_j^f\right) = \tilde{\mathbf{U}}_j \tilde{\mathbf{\Lambda}}_j \tilde{\mathbf{U}}_j^{T}. \tag{143}$$

Using Equations (142) and (143) we find

$$\mathbf{P}^{a}\left(\mathbf{y}_j^f\right)^{-\frac{1}{2}} = \tilde{\mathbf{U}}_j^{T} \tilde{\mathbf{\Lambda}}_j^{1/2} \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{U}^{T}. \tag{144}$$

Hence, we can write the update equation of the MMEF as

$$\mathbf{x}_j^a = \bar{\mathbf{x}}^a + \mathbf{X}'^{f}\mathbf{T}^{1/2}\tilde{\mathbf{U}}_j^{T}\tilde{\mathbf{\Lambda}}_j^{1/2}\mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{U}^{T}\left(\mathbf{x}_j - \bar{\mathbf{x}}^a\left(\mathbf{y}_j^f\right)\right). \tag{145}$$

This expression is suitable for high-dimensional applications when the matrices $\mathbf{T}(\mathbf{y}_j^f)$ are computed with localisation.

7.3. *Merging Particle Filter (MPF)*

The merging particle filter generates several sets of posterior ensembles and merges them via a weighted average to obtain a new set of particles that has the correct mean and covariance but is more robust than the standard particle filter. Specifically, the method draws a set of q ensembles each of size N_e from the weighted prior ensemble at the resampling step. Denote each ensemble member as $\mathbf{x}_{j,i}$ for ensemble member j in ensemble i. Then new merged ensemble members are generated via

$$\mathbf{x}_j^a = \sum_{i=1}^{q} \alpha_i \mathbf{x}_{j,i}. \tag{146}$$

To ensure that the new ensemble has the correct mean and covariance, the coefficients α_j need to fulfil the two conditions

$$\sum_{j=1}^{q} \alpha_j = 1; \quad \sum_{j=1}^{q} \alpha_j^2 = 1, \tag{147}$$

where each α_j also has to be a real number.

When $q > 3$ there is no unique solution for the α's, while for $q = 3$ we find:

$$\begin{aligned} \alpha_1 &= \frac{3}{4} \\ \alpha_2 &= \frac{\sqrt{13}+1}{8} \\ \alpha_3 &= -\frac{\sqrt{13}-1}{8}. \end{aligned} \tag{148}$$

Although not discussed by Nakano et al. (2007) this scheme will be degenerate for high-dimensional problems. However, we can make the α's space-dependent when $q > 3$ and then apply localisation.

8. Adaptive Gaussian mixture filter

Both ensemble Kalman and Monte Carlo-based techniques discussed in Sections 5 and 6, respectively, have their drawbacks. The Gaussian mixture filter (Anderson and Anderson, 1999; Bengtsson et al., 2003; Hoteit et al., 2008) attempts to avoid these by approximating an arbitrary form of the prior by combining multiple Gaussian priors. This gives it the advantage that both the local Kalman filter type correction step as well as the weighting and resampling step of a particle filter can be applied. This possibility also makes it applicable to highly non-linear and high dimensional systems. In this paper, we discuss the adaptive Gaussian mixture filter developed by Stordal et al. (2011) as a representative scheme, out of all Gaussian mixture filters that have been proposed.

In the Gaussian mixture filter, the prior distribution is approximated by a mixture density (Silverman, 1986) where each ensemble member forms the centre of a Gaussian density function

$$p(\mathbf{x}^f) = \sum_{j=1}^{N_e} \frac{1}{N_e} \mathcal{N}\left(\mathbf{x}_j^f, \tilde{\mathbf{P}}^f\right), \tag{149}$$

where $\mathcal{N}(\mathbf{x}_j, \tilde{\mathbf{P}})$ denotes a multivariate Gaussian kernel density with ensemble member $\mathbf{x}_j$ as mean and covariance matrix $\tilde{\mathbf{P}}^f = h^2\mathbf{P}^f$, in which $\mathbf{P}^f$ is the covariance of the whole forecast ensemble and h is a bandwidth parameter. Stordal et al. (2011) discuss that the optimal choice of the bandwidth h is $h_{opt} \sim N_e^{-1/5}$ if we are only interested in the marginal properties of the individual components of $\mathbf{x}$, but that it might be beneficial to choose $h > h_{opt}$ to reduce the risk of filter divergence, since the choice of the bandwidth parameter determines the magnitude of the Kalman filter update step. Thus, the parameter h is treated as the design parameter and is defined by the user. Note that each particle represents the mean of a Gaussian kernel and that the uncertainty associated with each particle is given by the covariance of that Gaussian kernel (Stordal et al., 2011).

If the likelihood is Gaussian, the posterior pdf is again a Gaussian mixture, now with pdf

$$p\left(\mathbf{x}^a|\mathbf{y}\right) = \sum_{j=1}^{N_e} w_j \mathcal{N}\left(\mathbf{x}_j^a, \tilde{\mathbf{P}}^a\right). \tag{150}$$

Here, the weights w_j are propotional to $\mathcal{N}\left(\mathbf{y} - \mathbf{H}\mathbf{x}_j^f, \mathbf{R}^a\right)$ with $\sum_j w_j = 1$ and $\mathbf{R}^a = \mathbf{H}\tilde{\mathbf{P}}^f\mathbf{H}^T + \mathbf{R}$. So, compared to the particle filter the covariance used in the weights is inflated with a term $\mathbf{H}\tilde{\mathbf{P}}^f\mathbf{H}^T$, leading to more equal weights. Each mean $\mathbf{x}_j^a$ and the covariance matrix $\tilde{\mathbf{P}}^a$ are obtained using one of the EnKF variants.

In high-dimensional systems, the covariance matrices are never formed explicitly, and the algorithm in Stordal et al. (2011) cannot be used. Hoteit et al. (2008) used an update based in the SEIK filter (see Section 5.2). For a more modern formulation, we provide here an algorithm based on Stordal et al. (2011) but explore an ETKF to avoid the explicit computation of $\tilde{\mathbf{P}}$. First, the matrix

$$\mathbf{T}_{GM}\mathbf{T}_{GM}^{\mathsf{T}} = \left(\mathbf{I} + \frac{h^2}{N_e - 1}\mathbf{S}^{\mathsf{T}}\mathbf{R}^{-1}\mathbf{S}\right)^{-1} \tag{151}$$

is generated with $\mathbf{S} = \mathbf{H}\mathbf{X}'^f$ similar to Equation (44), but including the factor h^2. Then, we perform an EVD of the symmetric matrix $(\mathbf{T}_{GM}\mathbf{T}_{GM}^{\mathsf{T}})^{-1} = \mathbf{U}_{GM}\mathbf{\Sigma}_{GM}\mathbf{U}_{GM}^{\mathsf{T}}$ and obtain the symmetric square root

$$\mathbf{T}_{GM} = \mathbf{U}_{GM}\mathbf{\Sigma}_{GM}^{-1/2}\mathbf{U}_{GM}^{\mathsf{T}}. \tag{152}$$

This is used to update the mean of each Gaussian kernel by calculating the ETKF update on each of the prior particles as

$$\overline{\mathbf{w}}_{j,GM} = \frac{1}{\sqrt{(N_e - 1)}}\mathbf{U}_{GM}\mathbf{\Sigma}_{GM}^{-1}\mathbf{U}_{GM}^{\mathsf{T}}(h\mathbf{X}'^f)^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}\mathbf{R}^{-1}\mathbf{d}_j \tag{153}$$

in which $\mathbf{d}_j = \mathbf{y} - \mathbf{H}\mathbf{x}_j^f$. The new centres of the Gaussian mixture densities are now found as

$$\mathbf{x}_j^a = \mathbf{x}_j^f + h\mathbf{X}'^f\overline{\mathbf{w}}_{j,GM}. \tag{154}$$

A square root of the posterior covariance of each Gaussian mixture density is found by

$$\mathbf{Z} = h\mathbf{X}'^f\mathbf{W}'_{GM} \tag{155}$$

in which

$$\mathbf{W}'_{GM} = \mathbf{U}_{GM}\mathbf{\Sigma}_{GM}^{-1/2}\mathbf{U}_{GM}^{\mathsf{T}}. \tag{156}$$

Thus, for Equation (150) we have $\tilde{\mathbf{P}}^a = (N_e - 1)^{-1}\mathbf{Z}\mathbf{Z}^T$, but to evaluate the equation one can use the square root $\mathbf{Z}$, so it is not required to compute $\tilde{\mathbf{P}}^a$ explicitly.

Until this point, the algorithm is the standard Gaussian mixture filter. The adaptive part of the filter was introduced by Stordal et al. (2011) and has been demonstrated to avoid filter divergence due to ensemble degeneration. Further, the adaptivity allows us to choose smaller values of the bandwidth parameter h. To stabilise the Gaussian mixture filter, we interpolate the original analysis weights with a uniform weight as

$$w_j^\alpha = \alpha w_j + (1 - \alpha)N_e^{-1}. \tag{157}$$

For the adaptivity, α is chosen to be

$$\alpha = N_{\text{eff}}N_e^{-1}, \tag{158}$$

where $N_{\text{eff}} = 1/(\sum_{l=1}^{N_e} w_l^2)$ is the effective ensemble size. To avoid ensemble degeneration one can further add a resampling step as in particle filters. It is performed if $N_{\text{eff}} < N_c$, with N_c a value that can be chosen freely, for instance $N_c = 0.5N_e$. The full scheme then becomes:

(a) When $N_{\text{eff}} \geq N_c$ no resampling is needed, so the weights are calculated as above and transported with each particle to the next set of observations.

(b) When $N_{\text{eff}} < N_c$ we will resample according to any of the resampling schemes in Appendix 1. This leads to a new set of states for the centres of the Gaussian mixtures denoted $\mathbf{x}_{j,(i)}$ in which j denotes the index of the state for resampling, and i its index after resampling. Note that several of the new states will coincide. To avoid identical samples we draw our final new ensemble from the Gaussian mixtures, as follows

$$\xi_i \sim \mathcal{N}(0, \mathbf{I}), \tag{159}$$

$$\mathbf{x}_i^a = \mathbf{x}_{j,(i)} + \mathbf{Z}\xi_i, \tag{160}$$

$$w_i = \frac{1}{N_e}. \tag{161}$$

Further, we set $\tilde{\mathbf{P}}^a = (N_e - 1)^{-1}\mathbf{X}^a(\mathbf{X}^a)^T$, but use this only in factorised form.

Note that in this scheme we never calculate a full state covariance.

It is important to realise what the adaptive part does. Indeed, by construction, the filter is not degenerate, but at the expense of strongly reducing the influence of the observations when α

is small. In high dimensional systems with a large number of independent observations localisation is essential to avoid using the scheme as a sum of ensemble Kalman filters only.

In the scheme by Bengtsson et al. (2003), the mean of each Gaussian pdf is chosen at random from the ensemble, and the covariance in each Gaussian pdf is estimated from the ensemble members which are local in state space, including a localisation and smoothing step. Since the scheme has not been applied to high-dimensional systems it will not be discussed here.

9. Practical implementation of the ensemble methods

This section is devoted to issues related to the practical implementation of the ensemble methods. In particular, we address the need for localisation and inflation in some of the presented ensemble methods to counteract the issues arising from ensemble undersampling in large scale problems such as ocean and atmosphere prediction. We also discuss the computational cost of each method as presented and the parallelisation of ensemble data assimilation methods. We will conclude this section with a discussion on the suitability of the ensemble data assimilation methods applied to non-linear dynamical models.

9.1. Localisation in EnDA

The success of the EnDA methods is highly dependent on the size of the ensemble being adequate for the system we apply these methods to. Thus, for large scale problems, where the number of state variables is many magnitudes larger than the number of ensemble members, ensemble undersampling can cause major problems in EnDA methods: underestimated ensemble variance, filter divergence, and errors in estimated correlations, in particular spurious long-range correlations. In such cases, spatial localisation is a necessary tool to minimise the effect of undersampling.

Localisation damps long-range correlations, e.g. in the ensemble covariance matrix ('covariance localisation', see Section 9.1.2). This damping can be applied to the extend to keep only correlations over limited distances and erase long-range correlations in the analysis step. Thus, localisation decouples the analysis update at distant locations in a model grid. The underlying assumption of localisation is that the assimilation problem has in fact a local structure. This means, that correlation length scales are much shorter than the extent of the model grid so that only correlations over short distances are relevant while for long distances the sampling error in the ensemble-estimated covariance matrices dominates (see, e.g. Morzfeld et al., 2017). This seems to be fulfilled for many oceanic and atmospheric applications. For example, Patil et al. (2001) described a locally low dimension for atmospheric dynamics. The success of localised filters in oceanic and atmospheric data assimilation applications also shows that this condition is dominantly fulfilled, even though it is known that long-range correlations (teleconnections) exist in the atmosphere and ocean. However, if a modelling problem does not have a local structure or if too little observations are available or the observations only represent long-range properties of the system, localisation cannot be applied.

Localisation is usually applied either explicitly by considering only observations from a region surrounding the location of the analysis or implicitly by modifying $\mathbf{P}$ or $\mathbf{R}$ so that observations from beyond a certain distance cannot affect the analysis state. The way in which such localisation is applied is still an active field of research and many variants of localisation schemes have emerged over the last decade. There are two main types of spatial localisation techniques (or simply localisation) that are widely used in ensemble data assimilation: Covariance localisation (also termed P- or B-localisation) and observation localisation (also denoted R-localisation). Both methods will be discussed here together with domain localisation, which is required for the application of observation localisation. In addition, a number of adaptive localisation schemes was developed over the recent years. A selection of these schemes is discussed in Section 9.1.4.

In general, all localisation schemes are empirical. While they improve the estimations by ensemble filters, they can disturb balances in the model state (Lorenc, 2003; Kepert, 2009). Further, the interaction of localisation with the serial observation processing usually applied with the EnSRF and EAKF methods can reduce the stability of these filters (Nerger, 2015).

9.1.1. Domain localisation. *Domain localisation* or *local analysis* is the oldest localisation technique. For ensemble Kalman filters it was first applied by Houtekamer and Mitchell (1998), but the method was also applied in earlier schemes of optimal interpolation (see Cohn et al., 1998). In domain localisation we only use the ensemble perturbations that belong to the domain D_γ in which the analysis correction of the state estimate is computed. For example, this domain can be a vertical column of grid points or a single grid point. Thus, we use a linear transformation $\mathbf{D}_\gamma$ to obtain

$$\mathbf{x}'_{j,\gamma} = \mathbf{D}_\gamma \mathbf{x}'^{f}_{j}, \tag{162}$$

where $j = 1, \ldots, N_e$ and $\gamma = 1, \ldots, \Gamma$ with Γ being the total number of subdomains. To localise, we now only use observations within a specified distance – the localization radius – around the local domain D_γ. This defines a local observation domain $\hat{D}_\gamma$. Using the corresponding linear transformation $\hat{\mathbf{D}}_\gamma$ we can transform the observation error covariance $\mathbf{R}$, the global observation vector $\mathbf{y}$, and the global observation operator $\mathbf{H}$ analogously to Equation (162) to their local parts

$$\mathbf{y}_\gamma = \hat{\mathbf{D}}_\gamma \mathbf{y}, \tag{163}$$

$$\mathbf{H}_\gamma = \hat{\mathbf{D}}_\gamma \mathbf{H}, \tag{164}$$

$$\mathbf{R}_\gamma = \hat{\mathbf{D}}_\gamma \mathbf{R} \mathbf{D}_\gamma^{\mathsf{T}}. \tag{165}$$

Thus, we neglect observations that are outside of the domain $\hat{D}_\gamma$. Then a general local analysis state is given by

$$\mathbf{X}_\gamma^a = \overline{\mathbf{X}}_\gamma^f + \mathbf{X'}_\gamma^f \left(\overline{\mathbf{W}}_\gamma + \mathbf{W'}_\gamma \right), \tag{166}$$

where $\overline{\mathbf{W}}_\gamma$ and $\mathbf{W'}_\gamma$ are computed using local ensemble forecast perturbations and local observations from the domain $\hat{D}_\gamma$. For a complete analysis update, a loop over all local analysis domains has to be performed with a local analysis update for each domain.

Applying domain localisation allows significant savings in computing time since solving for the analysis update is not performed globally but on much smaller local domains. Accordingly, updates on the smaller scale domains can be done independently and therefore parallel (Nerger et al., 2006) even if the observation domains overlap. In ensemble-based Kalman filters, domain localisation was used predominantly with filters that use the analysis error covariance matrix for the calculation of the gain like SEIK, ETKF, ESTKF, all discussed in detail in Section 5. In these algorithms, the forecast error covariance matrix is never explicitly computed. Examples of the application of domain localisation can be found, e.g. in Brusdal et al. (2003) and Testut et al. (2003).

Blindly using domain localisation can result in boxed analysis fields if neighbouring local domains are updated using significantly different observation sets. Thus, great care needs to be taken to choose domains so that they overlap sufficiently to produce smooth global analysis fields with minimal increase in computational cost. Today, domain localisation is typically applied with observation localisation (Hunt et al., 2007), which is discussed in Section 9.1.3.

9.1.2. Covariance localisation. *Covariance localisation* (also termed *P-localisation* or *B-localisation*, depending on whether the background covariance matrix is denoted $\mathbf{P}$ or $\mathbf{B}$ as in variational assimilation schemes) is a localisation method that is directly applied to the ensemble covariance matrix. The ensemble undersampling causes spurious cross-correlations between state variables. As realistic long-range correlations are typically small, the sampling errors are particularly pronounced for long distances. The direct covariance localisation can be used to reduce the long-range correlations in the forecast error covariance and hence damp the spurious correlations. In addition, the rank of the ensemble covariance is increased, giving more degrees of freedom to the analysis update (Hamill et al., 2001; Whitaker and Hamill, 2002).

Typically, covariance localisation is applied by first forming a correlation matrix $\mathbf{C}$ and then taking a Schur product (an element by element matrix multiplication) of this correlation matrix and the forecast error covariance. Thus, given some $\mathbf{P}^f$, our localised forecast error covariance will be

$$\mathbf{P}_L^f = \mathbf{C} \circ \mathbf{P}^f. \tag{167}$$

The localization matrix $\mathbf{C}$ is usually formed of correlation functions with compact support similar in shape to a Gaussian function (e.g. Gaspari and Cohn, 1999). Practically, the computation of the covariance matrix $\mathbf{P}^f$ can be avoided by applying the localisation matrix to the matrices $\mathbf{P}^f\mathbf{H}^T$ and $\mathbf{H}\mathbf{P}^f\mathbf{H}^T$ (see Equation (11)).

We note that, from all the ensemble-based Kalman filter methods presented in Section 5, covariance localisation can only be applied to the EnSRF and EAKF, since for these methods observations can be processed serially, and in the stochastic EnKF.

9.1.3. Observation localisation. In the case of square root filters, presented in Section 5, the full covariance matrix is never formed. Instead, only the ensemble perturbation matrix $\mathbf{X'}^f$ is calculated at each analysis step. Petrie and Dance (2010) showed that covariance-localisation for square root filters cannot be approximately decomposed into a square root of the correlation matrix $\boldsymbol{\rho}$,

$$\left(\boldsymbol{\rho}\boldsymbol{\rho}^{\mathsf{T}}\right) \circ \left(\mathbf{X'}\mathbf{X'}^{\mathsf{T}}\right) \neq \left(\boldsymbol{\rho} \circ \mathbf{X'}\right)\left(\boldsymbol{\rho} \circ \mathbf{X'}\right)^{\mathsf{T}} \tag{168}$$

thus covariance localisation cannot be applied. For such filters, e.g. SEIK, ETKF and ESTKF, the observation localisation is a more natural choice and is currently used instead of covariance localisation (Hunt et al., 2007; Miyoshi and Yamane, 2007; Janjić et al., 2011).

Observation localisation is applied by modifying the observation error covariance matrix $\mathbf{R}$. More specifically, one modifies its inverse $\mathbf{R}^{-1}$ so that the inverse observation variance decreases to zero with the distance of an observation from an analysis grid point. To be able to define the distance, it is necessary to perform the analysis with the domain localization method as described in Section 9.1.1. An abrupt cutoff could be obtained by setting observation variances to zero beyond a given distance. This would be equivalent to the simple domain localisation of Section 9.1.1 and could result in non-smooth analysis updates. For a smooth analysis, e.g. Brankart et al. (2003) described to increase the observation error variance with increasing distance from the analysis grid point. Hunt et al. (2007) proposed to use a gradual observation localisation in the LETKF acting on $\mathbf{R}^{-1}$, which is likewise applicable with the SEIK filter and the ESTKF. In this case, elements of $\mathbf{R}^{-1}$ are multiplied by a smoothly decreasing function of distance from the analysis grid point. This modification smoothly reduces the observation influence and excludes observations outside a defined radius by prescribing

their error to be infinitely large. As for covariance-localisation, the method uses a Schur product as

$$\tilde{\mathbf{R}} = \tilde{\mathbf{C}} \circ \mathbf{R} \,. \tag{169}$$

Here, the same correlation function (Gaspari and Cohn, 1999) as for covariance localisation can be used to construct the localisation matrix. However, in contrast to covariance localisation, $\tilde{\mathbf{C}}$ is not a correlation matrix as the values on the diagonal of this matrix vary with the distance between the observation and the local analysis domain. Then, the analysis update is computed as in the case of domain localisation, but using the weight-localised matrix $\tilde{\mathbf{R}}$. For computational savings we would in practise also discard any observations with zero weight from the analysis computations.

Both observation and covariance localisation can lead to similar assimilation results. In general, the optimal localisation has been found to be a bit larger for covariance localisation than for observation localisation (Greybush et al., 2011). The reason for this difference lies in the different effect of the localisations in the Kalman gain as was explained by Nerger et al. (2012b).

9.1.4. Adaptive localisation schemes. The localisation methods described above are widely used and can be applied without much additional computing cost. However, the optimal localisation radius is a priori unknown and needs to be tuned in numerical experiments. For the tuning one performs several data assimilation experiments with different localisation radii, perhaps over shorter time periods, and selects the radius that results in the smallest estimation errors. Regarding the theoretical understanding of localisation, Kirchgessner et al. (2014) showed for the case of observation localisation when each grid point is observed that the optimal localisation radius should be reached when the sum over the observation weights equals the ensemble size. This finding allows for a simple form of adaptivity or a starting point for further tuning. Further, Perianez et al. (2014) showed that both the sampling error in the ensemble covariance matrix and the observation error influence the optimal localisation radius. As the sampling error has a largest influence when the true correlations are small, the dynamically generated correlations also influence the optimal localisation radius (Zhen and Zhang, 2014; Flowerdew, 2015).

To avoid the need for numerical tuning and to better adapt the localisation to the dynamically created correlation structure, several adaptive localisation methods have been developed, which we shortly mention here. A common approach is to damp the spurious correlations that are caused by sampling errors due to the small ensemble size. Anderson (2007) developed a hierarchical localisation method, in which the ensemble is partitioned into sub-ensembles. Then, the sub-ensembles are used to estimate the sampling errors. Bishop and Hodyss (2009) proposed an adaptive localisation method that uses a power of the correlations to damp small correlations and pronounce those correlations that are significant. This method can find correlations even at longer distances. Further, methods have been developed to find empirical localisation functions. In these methods, one attempts to find for a single observation the weight factor that minimises the deviation from a true solution (Anderson, 2012; Lei and Anderson, 2014; Flowerdew, 2015). These methods are typically tuned once based on observation system simulation experiments (OSSEs), in which one knows the true state. When the OSSEs are configured realistically, the obtained localisation functions should be applicable for the assimilation of real observations after the tuning.

The major advantage of the methods proposed so far is that they are able to adaptively specify the localisation function or radius according to the dynamically generated covariance structure. However, the methods still need tuning, which can be even more costly than for the fixed covariance and observation localisation methods. For example, the method by Bishop and Hodyss (2009) requires the specification of an envelope function around the locations that are found by powering the correlations and the number of powers that are computed. Lei and Anderson (2014) also showed that the localisation function can change when it is applied iteratively such that a sufficient number of iterations have to be computed.

Apart from the adaptive localisation methods, further methods like spectral localisation (Buehner and Charron, 2007) and localisation in different variables (i.e. stream function, velocity potential, Kepert, 2006) have been developed. However, none of these methods are yet a standard for operational centres.

9.1.5. Localisation in particle filtering. Several variants of the Particle Filter that explore localisation have been developed recently, following its success in Ensemble Kalman Filters. An issue with directly localising $\mathbf{R}$ or using domain localisation is that the weight of each particle is a global property of the filter (van Leeuwen, 2009). That is, the same particle could have a high weight in one area and low weight in another making it ambiguous whether this particle should be resampled or not. Keeping parts of a number of particles that all perform well in a certain area of the domain and parts of other particles in other areas of the domain would lead to balance problems between variables and sharp gradients in the fields. In contrast, when performing parameter estimation a smooth variation of parameter values is less likely to cause imbalances in the model variables, and localisation is straightforward, as pioneered by Vossepoel and Van Leeuwen (2006).

Particle filters that use a proposal density, such as the EWPF discussed in Section 6.2.1 indirectly use localisation through the model error covariance matrix $\mathbf{Q}$. This localisation does not explicitly work on the weights but on how the states are updated, because a natural choice is to pre-multiply each update of a particle with that matrix. Since the model error covariance matrix will mainly contain short length-scale correlations related

to missing or inaccurate physics at the model grid scale, each point in the state space is only influenced by observations within the radius set by that covariance matrix. In fact, as noted in Section 6.2.1, we do have the freedom to choose this matrix differently from $\mathbf{Q}$, so other choices closer to our needs are possible. This is because the effects of this choice will be taken into account in the computation of the weights of each particle. This has not been explored in any detail in the literature.

Of the full particle filters, the ETPF (Reich, 2013) can easily be localised by taking for each grid point only observations close to that grid point into account and making the transformation matrix space-dependent to ensure smooth transitions between different regions. This can for example be achieved by calculating the transformation matrix in a limited number of grid points and interpolate that matrix between grid points. This would also reduce the number of computations, which would otherwise be prohibitive (see Section 9.4 on computational costs).

The PFGR and the NETF perform an ensemble transformation similar to the ETKF, but with a transform matrix $\mathbf{T}$ computed from particle filter weights. Accordingly, observation localisation can be applied to the NETF (Tödter and Ahrens, 2015) by smoothing the weight matrix over space. This can also be applied to the MMPF in the high-dimensional implementation. Also the MPF can be localised by making the weights local and using a systematic resampling method like Stochastic Universal Resampling (see Appendix 1). In practise, more might be needed, e.g. the extra averaging as advocated by Penny and Miyoshi (2016) described below.

Several localisation schemes have been proposed and discussed in the review van Leeuwen (2009) and those will not be repeated here. The most obvious thing to do is to weight and resample locally, and somehow glue the resampled particles together via averaging at the edges between resampled local particles (van Leeuwen, 2003b). Recently, Penny and Miyoshi (2016) used this idea with more extensive averaging, and their scheme runs as follows. First, for each grid point j the observations close to that grid point are found and the weight of each particle i is calculated based on the likelihood of only those observations:

$$w_{i,j} = \frac{p(\mathbf{y}_j|\mathbf{x}_{i,j})}{\sum_{k=1}^{N_e} p(\mathbf{y}_j|\mathbf{x}_{i,j})} \tag{170}$$

in which $\mathbf{y}_j$ denotes the set of observations within the localisation area. This is followed by resampling via Stochastic Universal Resampling to obtain ensemble members $x^a_{i,j}$ with $i = 1, \ldots, N_e$ for each grid point j. As mentioned before, the issue is that two neighbouring grid points can have different sets of particles, and smoothing is needed to ensure that the posterior ensemble consists of smooth particles. This smoothing is performed for each grid point j for each particle i by averaging over the N_p neighbouring points within the localisation area around grid point j:

$$x^a_{i,j} = \frac{1}{2}x^a_{i,j} + \frac{1}{N_p}\sum_{k=1}^{2N_p} x^a_{i,j_k} \tag{171}$$

in which j_k for $k = 1, \ldots, N_p$ denotes the grid point index for those points in the localisation area around grid point j. The resampling via Stochastic Universal Resampling is done such that the weights are sorted before resampling, so that high-weight particles are joined up to reduce spurious gradients.

While this scheme does solve the degeneracy problem in simple one-dimensional systems it is unclear if it will work well in complex systems such as the atmosphere in which fronts can easily be smoothed out, and non-linear balances broken, see e.g. the discussion in van Leeuwen (2009).

A new scheme has recently been proposed in Poterjoy (2016a), which involves a very careful process of ensuring smooth posterior particles and retaining non-linear relations. The filter processes each observation sequentially, as follows. First, adapted weights are calculated for the first element $\mathbf{y}_1$ of the observation vector, as

$$\tilde{w}_i = \alpha p(\mathbf{y}_1|\mathbf{x}_i) + 1 - \alpha \tag{172}$$

These weights are then normalised by their sum $\tilde{W}$. Then we resample the ensemble according to these normalised weights to form particles $\mathbf{x}_{k_i}$.

Here, α is an important parameter in this scheme, with $\alpha = 1$ leading to standard weighting, and $\alpha = 0$ leading to all weights being equal to 1. Its importance lies in the fact that the weights are always larger than $1-\alpha$, so even a value close to 1, say $\alpha = 0.99$, leads to a minimum weight of 0.01 that might seem small, but it means that particles that are more than 1.7 observational standard deviations away from the observations have their weights cut off to something close to $1 - \alpha$. This seriously limits the influence the observation can have on the ensemble. Furthermore, the influence of α does depend on the size of the observational error, which is perhaps not what one would like. It is included to avoid losing any particle.

Now, we do the following for each grid point j. For each member i we calculate a weight

$$\tilde{w}_i = \alpha\rho(\mathbf{y}_1, \mathbf{x}_j, r)p(\mathbf{y}_1|\mathbf{x}_i) + 1 - \alpha\rho(\mathbf{y}_1, \mathbf{x}_j, r) \tag{173}$$

in which $\rho(.)$ is the localisation function with localisation radius r. The normalised weights for this grid point, w_i, are obtained by dividing $\tilde{w}_i$ by the summed weights over all the particles. Note, again, the role played by α. Then, the posterior mean for this observation at this grid point is calculated as

$$\bar{\mathbf{x}}_j = \sum_{i=1}^{N_e} w_i \mathbf{x}_{i,j} \tag{174}$$

in which $\mathbf{x}_{i,j}$ is grid point j of particle i. Next, a number of scalars are calculated that ensure smooth posterior fields (Poterjoy, 2016a):

$$\begin{aligned}
\sigma_j^2 &= \sum_{i=1}^{N_e} \omega_i (\mathbf{x}_{i,j} - \bar{\mathbf{x}}_j)^2 \\
c_j &= \frac{N_e(1 - \alpha\rho(\mathbf{x}_j, \mathbf{y}_1, r))}{\alpha\rho(\mathbf{x}_j, \mathbf{y}_1, r)\tilde{W}} \\
r_{1,j} &= \sqrt{\frac{\sigma_j^2}{\frac{1}{N_e - 1}\sum_{i=1}^{N_e}(\mathbf{x}_{k_i,j} - \bar{\mathbf{x}}_j + c_j(\mathbf{x}_{i,j} - \bar{\mathbf{x}}_j))^2}} \\
r_{2,j} &= c_j r_{1j}
\end{aligned} \tag{175}$$

so that the final estimate becomes:

$$\mathbf{x}^a_{i,j} = \bar{\mathbf{x}}_j + r_{1,j}(\mathbf{x}_{k_i,j} - \bar{\mathbf{x}}_j) + r_{2,j}(\mathbf{x}_{i,j} - \bar{\mathbf{x}}_j). \tag{176}$$

This procedure is followed for each grid point so that at the end we have an updated set of particles that have incorporated the first observation. As a next step the whole process is repeated for the next observation, with a small change that $\tilde{w}_i$ is multiplied by $\tilde{w}_i$ from the previous observation, until all observations have been assimilated. In this way, the full weight of all observations is accumulated in the algorithm. Now the importance of α comes to full light: without α the ensemble would collapse because the $\tilde{w}$'s would be degenerate when observations are accumulated.

The final estimate shows that each particle at grid point j is the posterior mean at that point plus a contribution from the deviation of the posterior resampled particle from that mean and a contribution from the deviation of the prior particle from that mean. So each particle is a mixture of posterior and prior particles, and departures from the prior are suppressed. When $\alpha = 1$, so for a full particle filter, we find for grid points at the observation locations that $c_j = 0$ because it is $\rho(\mathbf{y}_1, \mathbf{x}_j, r) = 1$ here. Accordingly, it is $r_{2,j} = 0$ and $r_{1,j} \approx 1$ and indeed the scheme gives back the full particle filter.

Between observation locations it can be shown that the particles have the correct first and second order moments, but higher-order moments are not conserved. To remedy this a probabilistic correction is applied at each grid point as follows. The prior particles are dressed by Gaussians with width 1 and weighted by the likelihood weights to generate the correct posterior pdf. The posterior particles are dressed in the same way, each with weight $1/N_e$. Then the cumulative distribution functions (cdf's) for the two densities are calculated using a trapezoidal rule integration. A cubic spline is used to find the prior cdf values at each prior particle i, denoted by cdf(i). Then a cubic spline is fitted to the other cdf, and the posterior particle i is found as the inverse of its cdf at value cdf(i). See Poterjoy (2016a) for details. The result of this procedure is that higher order moments are brought back into the ensemble between observed points.

This scheme, although rather complicated, is the only local particle filter scheme that has been applied to high-dimensional geophysical systems based on primitive equations in Poterjoy and Anderson (2016b). (van Leeuwen, 2003b applied a local particle filter to a high-dimensional quasi-geostrophic system, but that system is quite robust to sharp gradients as it does not allow gravity waves.)

Another interesting local particle filter is the Multivariate Rank Histogram Filter (Metref et al., 2014a). The idea is to write the posterior pdf in terms of an observed marginal multiplied by a set of conditional pdfs. For example, for a 3-dimensional system in which variable x_1 is observed we have:

$$\begin{aligned}
p(x_1, x_2, x_3|y) &= \frac{p(y|x_1)}{p(y)} p(x_1, x_2, x_3) \\
&= \frac{p(y|x_1)}{p(y)} p(x_1, x_2) p(x_3|x_1, x_2) \\
&= \frac{p(y|x_1)}{p(y)} p(x_1) p(x_2|x_1) p(x_3|x_1, x_2).
\end{aligned} \tag{177}$$

The filter now uses the rank-histogram idea of Anderson (2010) on each component, resulting in a fully non-Gaussian update of each component. Localisation can be easily applied directly in this algorithm as it is a transformation algorithm and the transformation can be made local. Unfortunately, this procedure becomes too expensive when the system is high dimensional. However, via a so-called mean-field approximation we suppress the conditioning on non-observed variables, so that we find:

$$p(x_1, x_2, x_3|y) \approx \frac{p(y|x_1)}{p(y)} p(x_1) p(x_2|x_1) p(x_3|x_1). \tag{178}$$

This will make the algorithm parallelisable and suitable for high-dimensional applications, although that has not been explored yet.

9.2. *Ensemble covariance inflation*

In practice, an ensemble Kalman filter can diverge from the truth due to systematic underestimation of the error variances in the filter, possibly caused by model errors or ensemble undersampling as discussed in Section 9.1. In particular, estimating a too large amount of long range correlation will reduce the estimated variance too strongly. Regardless of the cause, underestimating the uncertainty leads to a filter that is overly confident in the state estimate. Thus, the analysis step of the filter puts increasingly

more weight on the ensemble background estimate than on the observations and, at some point, it disregards observations completely. Localisation is one method to reduce the undersampling. However, for high-dimensional systems, localisation alone is not sufficient to ensure a stable assimilation process and covariance inflation is applied to further increase the sampled variance and thus stabilise the filter. In addition, the inflation can partly account for model error in case of an imperfect model (Pham et al., 1998b; Hamill, 2001; Anderson, 2001; Whitaker and Hamill, 2002; Hunt et al., 2007).

Most common is a fixed multiplicative covariance inflation (Anderson and Anderson, 1999). The method uses the inflation factor r to perform a multiplicative inflation for each ensemble member $\mathbf{x}_j^{a,f}$. With $j = 1, \ldots, N_e$ being ensemble member indices, it is given by

$$\mathbf{x}_j^{a,f} = r\left(\mathbf{x}_j^{a,f} - \overline{\mathbf{x}}^{a,f}\right) + \overline{\mathbf{x}}^{a,f} \tag{179}$$

where r usually is chosen to be slightly greater than one. The specification of an optimal inflation factor may vary according to the size of the ensemble ((Hamill, 2001); Whitaker and Hamill, 2002) and the choice of r will depend on various factors, such as dynamics of the model, type of the ensemble filter used as well as the length scale of covariance localisation.

Related to covariance inflation is the so-called 'forgetting-factor' ρ introduced by Pham et al. (1998b). The forgetting factor is usually chosen to be slightly lower than one and is typically applied in the square root filters like the ETKF, SEIK and ESTKF. For example, in the ETKF it is applied to $\mathbf{T}\mathbf{T}^T$, e.g. in Equation (44) as

$$\mathbf{T}_T\mathbf{T}_T^{\mathsf{T}} = \left(\rho\mathbf{I} + \frac{1}{N_e - 1}\mathbf{S}^{\mathsf{T}}\mathbf{R}^{-1}\mathbf{S}\right)^{-1}. \tag{180}$$

In this way, the inflation and forgetting factors are related as $\rho = r^{-2}$. Equation (180) allows one to apply inflation in a computationally very efficient way because $\mathbf{T}\mathbf{T}^T$ is much smaller than the ensemble states to which the inflation is applied in Equation (179).

Next to the multiplicative inflation, an additive inflation has been proposed. The multiplicative inflation leads to an inflation that is relative to the variance level. Thus, large variances will be inflated much more than small variances. This behaviour can be avoided with additive inflation (Ott et al., 2004), which can also be applied in combination with the multiplicative inflation. In additive inflation, all variances are inflated by the same amount, rather than a relative factor. This difference can be useful if the variances vary strongly as in this case the additive inflation acts stronger on the very small variances.

The optimal strength of the inflation is usually determined by tuning experiments, i.e. running experiments with different inflation values and analysing which value results in the smallest estimation errors. Usually a single fixed value of r or ρ is chosen for all grid points. This situation is mainly motivated by the fact that a manual tuning of spatially varying inflations is not feasible for high-dimensional models. To avoid the tuning, several adaptive inflation methods have been proposed. Brankart et al. (2003) proposed to use the relation

$$\mathrm{tr}(\rho^{-1}\mathbf{S}\mathbf{S}^T + \mathbf{R}) = (\mathbf{y} - \mathcal{H}(\overline{\mathbf{x}}^f))^T(\mathbf{y} - \mathcal{H}(\overline{\mathbf{x}}^f)) \tag{181}$$

to estimate a temporally variable forgetting factor ρ for multiplicative inflation. This equation is one of the statistical consistency relations in observation space that Kalman filters should fulfil (Desroziers et al., 2005). Further, Anderson (2009) proposed a method to adaptively estimate spatially and temporally varying inflation factors. This method also aims to fulfil Equation (181) but uses Bayesian estimation to obtain the inflation values. All of these adaptive methods do assume that we have a very good knowledge of the error covariance of the observations. Apart from adaptively inflating the ensemble spread, adaptive inflation of observation errors has been proposed by Minamide and Zhang (2017) for assimilating all-sky satellite brightness temperatures.

An alternative to the inflation can be to explicitly account for the sampling error caused by the finite ensemble size as is done in the finite-size ensemble transform Kalman filter (Bocquet, 2011). This method, while still denoted 'Kalman filter', requires the iterative minimisation of a cost functional and is hence distinct from the Ensemble Kalman filter variants in Section 5, which compute a one-step analysis update.

9.3. *Parallelisation of EnDA*

The need to integrate an ensemble of model states leads to large computational costs, because instead of computing a single model integration as in normal modelling applications an ensemble of $\mathcal{O}(10–100)$ members has to be propagated. To reduce the time to perform the costly computations one can apply parallelisation of the data assimilation program and then use high-performance computers with a large number of processors to perform the computations. The ensemble integrations as the most costly part of the computations can be easily parallelised. In fact, the integration of each ensemble state is independent from the other states. Thus, this step could be parallelised by simply starting the numerical model N_e times. Each model state has to be initialised from a different restart file and one has to store the final state of each model integration to keep the information on the forecast ensemble. Subsequently to the ensemble forecasts, one starts the data assimilation program, which reads the ensemble information from the files, computes the analysis step, and writes a set of new restart files to prepare the next forecast phase. The computations of the analysis step can also be parallelised as is

outlined below. This implementation scheme of data assimilation can be termed 'offline coupling' (Nerger and Hiller, 2013). While being flexible, the frequent writing and reading of the large files holding the ensemble states can take a significant amount of time.

A more sophisticated parallelisation of the ensemble data assimilation problem with a high-dimensional ocean model was discussed by Keppenne and Rienecker (2002) and Keppenne and Rienecker (2003). This method applied a domain-decomposition to the model and then integrated several ensemble states concurrently. The forecast ensemble was then collected by the use of the parallelisation technique SHMEM, which was also used for exchanging data in between processors during the analysis step of the EnKF applied in this study. Keeping the analysis step and the ensemble forecasts within one program reduced the overall computing time because the writing and reading of model state files is reduced.

The analysis step of the ensemble filters can also be parallelised using parallelisation methods like the Message Passing Interface (MPI, Gropp et al., 1994). The parallelization differs depending on whether localisation is used and on which of the filters is used. For the filter methods that assimilate all observations at once (in contrast to the serial observation processing of the EAKF and EnSRF) using the domain-decomposition of a model was found to be more efficient than using ensembles which are distributed over several processors because the amount of data that has to be exchanged using MPI is smaller for domain-decomposition (Nerger et al., 2005a).

For the ensemble Kalman filters with domain localisation, the local analysis update is independent for each local domain. Thus, this part is naturally parallel and can be distributed with MPI, the shared-memory standard OpenMP, or a combination of both. However, because the observation domains have a larger spatial extent they can reach into the grid domain held by neighbouring processors. The local analysis step needs the difference (innovation) between the observation and the corresponding part of the observed state vector. These differences need to be first computed by the processor that holds the sub-domain and then exchanged in between the different processors computing the analysis step. This computation of the observation innovations and their exchange using MPI is only required once before the loop over all local analysis domains can be computed in parallel (Nerger and Hiller, 2013). The cost for these operations depends on the total number of observations and on their distribution over the model grid. For many observations this can limit the parallel speedup of the analysis update as was shown for the localised SEIK filter by Nerger and Hiller (2013).

The EAKF and EnSRF are typically applied with serial observation processing and covariance localisation. In this case, the parallelisation of the analysis step has to take into account that for each assimilated observation the full model state has to be updated. Hence, also the innovation differences between the not yet assimilated observations and the corresponding observed model state change after each update. Anderson (2007) proposed to let each processor separately update the innovations so that the required parallel communication is limited. This parallelisation does not take the localisation into account. Taking into account that the localisation results in a limited reach of the observation influence, Wang et al. (2013) proposed another parallelisation strategy.

The analysis step of the ensemble Kalman filters requires only the model states. This allows for a generic coupling between the model and the analysis step. In particular, one can implement filter algorithms such that they can be coupled in the same way with different models. This allows one to build generic frameworks for ensemble data assimilation (Nerger et al., 2005a; Nerger and Hiller, 2013; Browne and Wilson, 2015). In the generic form, the ensemble forecast can still be computed by concurrent parallel model forecasts. The transfer of the forecast state information can then be performed either directly in memory by subroutine calls (Nerger and Hiller, 2013) or by parallel communication using MPI (Nerger, 2004; Browne and Wilson, 2015). These strategies allow a tight 'online' coupling of the model and the data assimilation code that computes the analysis updates. The coupling can be achieved with minimal changes in the model code.

For the implementation of the EWPF and IEWPF different parallelisation schemes are applicable for the computations at each nudging step in between observations (Equations (120) and (123)) and at observation time for the EWPF between Equations (106) to (111) and for the IEWPF between Equations (112) to (114). Before the observation time, the computations for the random forcing $\tilde{\boldsymbol{\beta}}_j^{(m)}$ in Equation (120) are independent for each particle since a different forcing is drawn from the covariance for each of them. Similar, the nudging term in Equation (120) and the update of the weights are independent for each particle. Thus, these operations can be performed in parallel and there is no need to gather all particles on a single process. The computation of the matrix $\boldsymbol{\Upsilon}$ in Equations (108) and (117) is computationally the most expensive part. When the observation operator does not change over time, this matrix can be precomputed before beginning the assimilation. The downside of this approach is that this matrix can be huge and requires a lot of memory if the state dimension and number of observations are large. Otherwise, since the same matrix is used by all particles, it is possible to distribute the computation to all processes allocated for the particles, e.g. using a parallel matrix solver. At observation time, most of the computations are again independent for all particles. Only the maximum weight obtained from Equations (106) and (118) for EWPF and IEWPF, respectively, must be exchanged over all processes holding particles, so that the target weight w_{target} can be computed. Further parallelisation, e.g. to use the domain decomposition of the model, might also be possible. However, the matrix $\mathbf{Q}$ is frequently implemented in form of operators. As the parallelisation is always dependent on the par-

ticular implementation of the matrix $\mathbf{Q}$ it cannot be generalised for all models.

9.4. Computational cost

In Section 5, we presented various ensemble-base Kalman filter methods in a clean mathematical way for ease of comparison and clarity. In Appendix 2, we give a practical and precise pseudo-algorithms on how to implement each method. Providing detailed operation counts for all the ensemble methods presented in this paper would be too lengthy but more importantly the actual operation count would depend on many details such as operators $\mathcal{H}$ and $\mathbf{R}$, which are case specific for the model and observations. The operation counts provided here have been obtained by counting them in the pseudo codes in Appendix 2.

Generally, the leading order of operation counts in the different filters are those that scale with third order in any of the dimensions N_x, N_y, and N_e. For the SEIK, ETKF, ESTKF, and the EAKF methods, the leading order of operation count is $O\left(N_y N_e^2 + N_e^3 + N_x N_e^2\right)$ if the observation error covariance matrix is diagonal. The main cost is the update of the ensemble by multiplying with the weight matrix in Equation (19) which has a complexity of $O\left(N_x N_e^2\right)$. Computing the matrix $\mathbf{T}\mathbf{T}^T$, e.g. in Equation (25) involves multiplications of matrix $\mathbf{S}$ with $\mathbf{R}^{-1}$ which has a complexity of $O\left(N_y N_e^2\right)$. Finally computing the transform matrix $\mathbf{T}$ by a Cholesky decomposition or an EVD has a computational complexity of $O\left(N_y N_e^2\right)$. While the leading order of operation counts is identical for all four filters, the SEIK and ESTKF are in general computationally faster than the ETKF, or the bulk formulation of the EAKF, despite equal leading operation counts due to details in the algorithms. For the EAKF, computing the SVDs of the matrices $\mathbf{X}'^f$ and $\tilde{\mathbf{S}}$, whose costs scale with $O\left(N_x N_e^2\right)$ and $O\left(N_y N_e^2\right)$, respectively, increases the computing time without changing the leading order of operation counts. Thus, the leading order of operation count does not reflect the computing speed.

The serial observation handling that is usually applied in the EAKF and EnSRF leads to an operation count of $O\left(N_y N_x N_e\right)$ in the leading order. Because only the ensemble updates are of third order complexity in the serial update, it can be faster than the bulk updates that assimilate all observations at once. This is even the case when localisation is used. However, in combination with localisation, the stability of the serial formulations can be deteriorated (Nerger, 2015).

The leading order operation count of the stochastic EnKF with perturbed forecasted observations is $O(N_y N_e^2 + N_y^2 N_e + N_y^3 + N_x N_e^2)$. Here again the ensemble update, which scales as $O\left(N_x N_e^2\right)$, is usually the most costly operation. However, the EnKF is usually more costly than the filters mentioned before because of the inversion of the $N_y \times N_y$ matrix $\mathbf{F}_F$ (Equation (69)), which has a complexity of $O\left(N_y^3\right)$. Parallelising this inversion can help to reduce the computing time. A computing cost $O\left(N_y^3\right)$ also occurs for the bulk formulation of the EnSRF due to the EVD computed in Equation (58).

When localisation is used, the change in the cost compared to the global formulation depends on the localisation method used. For covariance localisation (Section 9.1.2), the cost for computing the weight in matrix $\mathbf{C}$ and to apply it to $\mathbf{P}^f$ or the matrices $\mathbf{P}^f\mathbf{H}^T$ and $\mathbf{H}\mathbf{P}^f\mathbf{H}^T$ is added. For observation localisation (Section 9.1.3), the cost to compute the analysis for a local analysis domain with the bulk update methods is $O(N_{y,\gamma} N_e^2 + N_e^3 + N_{x,\gamma} N_e^2)$, where $N_{y,\gamma}$ is the number of local observations and $N_{x,\gamma}$ is the size of a local state vector that is corrected. Because both $N_{y,\gamma}$ and $N_{x,\gamma}$ are usually much smaller than the global dimensions N_y and N_x, a single local analysis update is cheaper than the global update. However, the local analysis update has to be computed for each local analysis domain. Thus, the cost for the analysis with observation localisation is usually significantly higher than the global analysis. However, the local analysis can be easily parallelised to reduce the computing time as was described in Section 9.3.

The computing cost in the ETKF can be reduced using a projection matrix $\mathbf{A}$ analogous to the SEIK and ESKTF methods. For the ETKF, this projection is square-matrix with diagonal entries of $1 - 1/N_e$ and off-diagonal entries of $-1/N_e$. The advantage of using $\mathbf{A}$ is that one can avoid the explicit computation of $\mathbf{X}'$ in favour of applying $\mathbf{A}$ to smaller matrices when evaluating the analysis equations (Nerger et al., 2012a).

The computational cost for the particle-based non-linear filter NETF (Section 7.1) is similar to that of the ETKF since the analysis is performed in the N_e-dimensional subspace spanned by the ensemble members. In addition, the NETF does not compute an inverse matrix thus avoiding computational instabilities caused by small singular values, which are sometimes neglected in ETKF implementations for that reason (Sakov et al., 2012). If localisation is applied to the NETF, the local analysis computations are independent and can be evaluated in parallel as for the ETKF. The generation of random rotation matrices consumes additional resources; however, it is possible to resort to a collection of precalculated random matrices since they only depend on ensemble size N_e.

9.5. Ensemble data assimilation and non-linearity

The original EnKF was developed to overcome stability problems of the extended Kalman filter (see Jazwinski, 1970) that were discovered with ocean data assimilation applications (Evensen, 1993). Due to its use of an ensemble to propagate the state error covariance matrix, the EnKF is suited for non-linear models in this phase. However, the analysis step is based on the Kalman filter and is only optimal for Gaussian distributions. Obviously, a non-linear model forecast will transform a Gaussian distribution into a non-Gaussian distribution. Hence, the optimality of the Kalman filters is no longer preserved and the estimated analysis state and the error estimates will be sub-

Table 2: Overview of filter methods available from *Sangoma* project website.

Filter	Section	Comment
EnKF	5.1	includes covariance localisation
EnsRF	5.6	includes covariance localisation
ETKF/LETKF	5.4	ETKF is without localisation; LETKF includes domain/observation localisation
ESTKF/LESTKF	5.9	ESTKF is without localisation; LESTKF includes domain/observation localisation
NETF	7.1	without localisation

optimal. This is a common issue for all ensemble filters whose analysis step is based on the equations of the Kalman filter. Nonetheless, the many existing data assimilation studies with non-linear models, e.g. of the ocean or atmosphere, with different formulations of the ensemble Kalman Filters show that these filters are rather stable with regard to non-linearity.

Second-order accurate ensemble filters, like the NETF, MMEF and MPF in Section 7 as well as the adaptive Gaussian mixture filter described in Section 8 avoid the assumption that the forecast ensemble has a Gaussian distribution. Thus, they should be better suited for non-linear systems. When the methods are applied with localisation, they can also be applied with large systems (e.g. the NETF in Tödter et al., 2016). However, filters like the NETF are still approximations to the full non-linear analysis that is performed by particle filters.

Particle filters do not rely on any assumption on the error distribution of the state estimate. However, the observation errors are frequently assumed to be Gaussian as is for the particle filters presented in Section 6. Additionally, while the EWPF and IEWPF do not require knowledge of forecast errors, they both require good knowledge of model errors, i.e. $\mathbf{Q}$. Of course, good knowledge of model errors is always beneficial to forecasting irrespective of the data assimilation method used, but for the application EWPF and IEWPF model errors are essential. While standard particle filters suffer the curse of dimensionality when applied to large systems, the EWPF and IEWPF by construction are designed to work for high-dimensional models, including those which are highly non-linear, with a small number of particles, e.g. Ades and van Leeuwen (2013) and Zhu et al. (2016). However, when applying the relaxation scheme between observations in an EWPF it is important to keep in mind that one has to choose the relaxation term $\tilde{\boldsymbol{\Upsilon}}$ in Equation (120) very carefully. We can choose this term to suit the needs of our model and indeed we need to do so carefully by selecting an appropriate relaxation strength function ρ and covariance matrix. The relaxation term can be chosen to be constant between observation times, but that would not be a good idea if the system experiences oscillations between observations. In that case, the strength term can be chosen to be linearly increasing with the time lag to the next observation we are nudging particles to, or non-linear with maximum strength close to the observations.

In all local particle filters that we discussed the posterior particles are linear combinations of the prior particles. This has the potential to break non-linear balances between variables in the model. However, the linear combinations are typically formed such that only prior particles are added that are close to each other in state space, and hence quite similar. So this is not necessarily a disadvantage.

10. Summary and conclusion

This overview paper provides a coherent algorithmic summary and highlights differences between many currently used ensemble data assimilation methods that can be applied to high dimensional and non-linear problems such as ocean or weather prediction including well-known ensemble-based Kalman filters as well as recently developed particle filter methods and the Gaussian mixture filter.

We have presented these methods in a mathematically coherent way allowing the reader to compare many methods easily. In particular, we have presented all ensemble-based Kalman filter methods in form of a square root filter. In addition, we have included practical pseudo-algorithms for all methods since for computational reasons many of them would not be implemented in the form they are mathematically described. For some of the particle filters and for the Gaussian mixture filter we have presented the theory along with the step by step algorithm.

Finally, we have discussed important issues for practical implementation of the ensemble methods including various methods of localisation, inflation, parallelisation, computational cost and ensemble applicability to non-linear problems.

Concluding, a wealth of ensemble-based data-assimilation methods have been developed, and although they seem quite different in theory, the numerical implementations are quite similar. The implementations turn out to be quite similar to those for the particle filters, even those that explore a proposal density, where the state covariances that play an essential role in the Kalman filters are replaced by the covariance of the model errors. The main difference is that the state covariances are evolving over time and are always of low rank, while the model error covariance is given and of full rank but sparse. This means that different numerical algorithms need to be used to solve the equations when the system of interest has a high dimension.

11. Code availability

We note that many of the algorithms here have been efficiently implemented as a part of the Sangoma project and are freely available to everyone on the project website http://sourceforge.net/p/sangoma/ along with many other tools useful to data assimilation. Table 2 provides a list of available filters. Please note that

the filter implementation was done independently from this paper so that not all filters described here are available. For simplicity these filters have been implemented without parallelisation and are hence only usable for moderately large problems with a state dimension of $O(10^5)$.

Further, all of these analysis methods have been implemented in at least one of the toolboxes connected to the Sangoma project, these are: EMPIRE, OAK, SESAM, OPENDA, BELUGA/SEQUOIA, NERSC and PDAF. For example, the set of filters listed in Table 2, plus the SEIK filter (5.2) with localization, are available in a parallelised implementation for high-dimensional problems in the freely available data assimilation framework PDAF (Nerger et al., 2005a; Nerger and Hiller, 2013). Further, the EWPF (Section 6.2.1) and IEWPF (Section 6.2.2) are available in EMPIRE (Browne and Wilson, 2015).

Acknowledgements

PJvL thanks the European Research Council (ERC) for funding of the CUNDA project under the European Unions Horizon 2020 research and innovation programme.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the SANGOMA EU Project [grant number FP7-SPACE-2011-1-CT-283580-621 SANGOMA].

Notes

1. www.data-assimilation.net/.
2. The discussion of the increasingly growing developments in hybrid data assimilation methods is beyond the scope of this paper, instead we refer the reader to a very recent review article by Bannister (2017), and papers by Frei and Künsch (2013) and Chustagulprom et al. (2016) aiming to bridge particle and ensemble Kalman filter methods.
3. Note that $\rho = 1$ if $\hat{p} = \tilde{N}_e$.
4. Many of the analysis methods discussed in this paper including MRHF have been implemented in Sangoma and are available for free to download from www.data-assimilation.net, as well as many other data assimilation tools for diagnostics, utilities etc..
5. Interestingly, the ECMWF is using an ensemble of 4DVars for their weather forecasting scheme, and it is relatively easy to turn this into a set of particles using 4DVar as proposal (see e.g. van Leeuwen et al., 2015).
6. The model error covariance matrices are usually assumed to be equal, i.e. $\tilde{\mathbf{Q}} = \mathbf{Q}$. .

References

Ades, M. and van Leeuwen, P. J. 2013. An exploration of the equivalent weights particle filter. *Q. J. R. Meteorol. Soc.* **139**, 820–840.

Anderson, J. 2003. A local least squares framework for ensemble filtering. *Mon. Wea. Rev.* **131**, 634–642.

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

Anderson, J. L. 2007. Exploring the need for localization in ensemble data assimilation using a hierarchical ensemble filter. *Physica D* **230**, 99–111.

Anderson, J. L. 2009. Spatially and temporally varying adaptive covariance inflation for ensemble filters. *Tellus* **61A**, 72–83.

Anderson, J. L. 2010. A non-Gaussian ensemble filter update for data assimilation. *Mon. Wea. Rev.* **138**(11), 4186–4198.

Anderson, J. L. 2012. Localization and sampling error correction in ensemble Kalman filter data assimilation. *Mon. Wea. Rev.* **140**, 2359–2371.

Anderson, J. L. and Anderson, S. L. 1999. A Monte Carlo implementation of the non-linear filtering problem to produce ensemble assimilations and forecasts. *Mon. Wea. Rev.* **126**, 2741–2758.

Bannister, R. N. 2017. A review of operational methods of variational and ensemble-variational data assimilation. *Q. J. R. Meteorol. Soc.* **143**, 607–633.

Bengtsson, T., Snyder, C. and Nychka, D. 2003. Toward a nonlinear ensemble filter for high-dimensional systems. *J. Geophys. Res.* **108**, 8775–8785.

Bengtsson, T., Bickel, P. and Li, B. 2008. Curse-of-dimensionality revisited: collapse of the particle filter in very large scale systems. *IMS Collections: Prob. Stat. Essays Honor David A. Freedman* **2**, 316–334.

Bishop, C. H. and Hodyss, D. 2009. Ensemble covariances adaptively localized with ECO-RAP. Part 2: a strategy for the atmosphere. *Tellus* **61A**, 97–111.

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

Blockley, E. W., Martin, M. J., McLaren, A. J., Ryan, A. G., Waters, A., and co-authors. 2014. Recent development of the Met Office operational ocean forecasting system: an overview and assessment of the new global FOAM forecasts. *Geosci. Mod. Dev.* **7**, 2613–2638.

Bocquet, M. 2011. Ensemble Kalman filtering without the intrinsic need for inflation. *Nonl. Proc. Geophy.* **18**, 735–750.

Bocquet, M., Pires, C. A. and Wu, L. 2010. Beyond Gaussian statistical modelling in geophysical data assimilation. *Mon. Wea. Rev.* **138**, 2997–3023.

Bolić, M., Djurić, P. M. and Hong, S. 2003. New resampling algorithms for particle filters. *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference*, Vol. 2, IEEE, pp. II–589–592.

Brankart, J.-M., Testut, C.-E., Brasseur, P. and Verron, J. 2003. Implementation of a multivariate data assimilation scheme for isopycnic coordinate ocean models: application to a 1993–1996 hindcast of the North Atlantic ocean circulation. *J. Geophys. Res.* **108**(C3), 3074.

Browne, P. A. and Wilson, S. 2015. A simple method for integrating a complex model into an ensemble data assimilation system using MPI. *Env. Modell. Software* **68**, 122–128.

Brusdal, K., Brankart, J. M., Halberstadt, G., Evensen, G., Brasseur, P., and co-authors. 2003. A demonstration of ensemble based assimilation methods with a layered OGCM from the perspective of operational ocean forecasting systems. *J. Mar. Syst.* **40–41**, 253–289.

Buehner, M. and Charron, M. 2007. Spectral and spatial localization of background-error correlations for data assimilation. *Q. J. R. Meteorol. Soc.* **133**, 615–630.

Burgers, G., van Leeuwen, P. J. and Evensen, G. 1998. Analysis scheme in the ensemble Kalman filter. *Mon. Wea. Rev.* **126**(6), 1719–1724.

Campbell, W. F., Bishop, C. H. and Hodyss, D. 2010. Vertical covariance localization for satellite radiances in ensemble Kalman filters. *Mon. Wea. Rev.* **138**, 282–290.

Chorin, A. J. and Tu, X. 2009. Implicit sampling for particle filters. *PNAS* **106**, 17249–17254.

Chorin, A. J., Morzfeld, M. and Tu, X. 2010. Interpolation and iteration for nonlinear filters. *Commun. Appl. Math. Comput. Sci.* **5**, 221–240.

Chustagulprom, N., Reich, S. and Reinhardt, M. 2016. A hybrid ensemble transform filter for nonlinear and spatially extended dynamical systems. *SIAM/ASA J. Uncert. Quant.* **4**, 552–591.

Cohn, S. E., Da Silva, A., Guo, J., Sienkiewicz, M. and Lamich, D. 1998. Assessing the effects of data selection with the DAO physical-space statistical analysis system. *Mon. Wea. Rev.* **126**, 2913–2926.

Desroziers, G., Berre, L., Chapnik, B. and Poli, P. 2005. Diagnosis of observation, background and analysis-error statistics in observation space. *Q. J. R. Meteorol. Soc.* **131**, 3385–3396.

Doucet, A., Godsill, S. and Andrieu, C. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comput.* **10**, 197–208.

Doucet, A., de Freitas, N., Gordon, N. 2001. *Sequential Monte-Carlo Methods in Practice*. Springer-Verlag, New York.

Evensen, G. 1993. Open boundary conditions for the extended Kalman filter with a quasi-geostrophic ocean model. *J. Geophys. Res.* **98**(C9), 16529–16546.

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.

Evensen, G. 2003. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dyn.* **53**, 343–367.

Flowerdew, J. 2015. Towards a theory of optimal localisation. *Tellus A* **67**, 25257.

Frei, M. and Künsch, H. R. 2013. Bridging the ensemble Kalman and particle filters. *Biometrica* **100**, 781–800.

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

Golub, G. H. and Van Loan, C. F. 1996. *Matrix computations*, 3rd ed. The Johns Hopkins University Press, Baltimore and London.

Gordon, N. J., Salmond, D. J. and Smith, A. F. M. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proc.* **140**, 107–113.

Greybush, S. J., Kalnay, E., Miyoshi, T., Ide, K. and Hunt, B. R. 2011. Balance and ensemble Kalman filter localization techniques. *Mon. Wea. Rev.* **139**, 511–522.

Gropp, W., Lusk, E. and Skjellum, A. 1994. *Using MPI: Portable Parallel Programming with the Message-Passing Interface* The MIT Press, Cambridge, Massachusetts.

Hamill, T. M. 2001. Interpretation of rank histograms for verifying ensemble forecasts. *Mon. Wea. Rev.* **129**, 550–560.

Hamill, T. M. 2006. Ensemble-based atmospheric data assimilation. In: *Predictability of Weather and Climate* (eds. T. Palmer and R. Hagedorn). Cambridge University Press, New York, pp. 124–156. chapter 6.

Hamill, T. M., Whitaker, J. S. and Snyder, C. 2001. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. *Mon. Wea. Rev.* **129**, 2776–1790.

Hoteit, I., Pham, D.-T., Triantafyllou, G. and Korres, G. 2008. A new approximate solution of the optimal nonlinear filter for data assimilation in meteorology and oceanography. *Mon. Wea. Rev.* **136**, 317–334.

Houtekamer, P. L. and Mitchell, H. L. 1998. Data assimilation using an ensemble Kalman filter technique. *Mon. Wea. Rev.* **126**, 796–811.

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

Houtekamer, P. L. and Zhang, F. 2016. Review of the ensemble Kalman filter for atmospheric data assimilation. *Mon. Wea. Rev.* **144**, 4489–4532.

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

Ide, K., Courtier, P., Ghil, M. and Lorenc, A. C. 1997. Unified notation for data assimilation: pperational, l sequential and variational. *J. Meteor. Soc. Jpn.* **75**, 181–189.

Janjić, T., Nerger, L., Albertella, A., Schroeter, J. and Skachko, S. 2011. On domain localization in ensemble-based Kalman filter algorithms. *Mon. Wea. Rev.* **139**, 2046–2060.

Jazwinski, A. H. 1970. *Stochastic Processes and Filtering Theory* Academic Press, New York.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Trans. AMSE –J. Basic Eng.* **82**(D), 35–45.

Kepert, J. D. 2006. Localisation, balance and choice of analysis variable in an ensemble Kalman filter. *Q. J. R. Meteorol. Soc.* **135**(642), 1157–1176.

Kepert, J. D. 2009. Covariance localisation and balance in an ensemble Kalman filter. *Q. J. R. Meteorol. Soc.* **135**, 1157–1176.

Keppenne, C. L. and Rienecker, M. M. 2002. Initial testing of a massively parallel ensemble Kalman filter with the Poseidon isopycnal ocean circulation model. *Mon. Wea. Rev.* **130**, 2951–2965.

Keppenne, C. L. and Rienecker, M. M. 2003. Assimilation of temperature into an isopycnal ocean general circulation model using a parallel ensemble Kalman filter. *J. Mar. Syst.* **40–41**, 363–380.

Kirchgessner, P., Nerger, L. and Bunse-Gerstner, A. 2014. On the choice of an optimal localization radius in ensemble Kalman filter methods. *Mon. Wea. Rev.* **142**, 2165–2175.

Law, K. J. H. and Stuart, A. M. 2012. Evaluating data assimilation algorithms. *Mon. Wea. Rev.* **140**, 3757–3782.

Lawson, W. G. and Hansen, J. A. 2004. Implications of stochastic and deterministic filters as ensemble-based data assimilation methods in varying regimes of error growth. *Mon. Wea. Rev.* **132**, 1966–1989.

Lei, J. and Bickel, P. 2011. A moment matching ensemble filter for nonlinear non-Gaussian data assimilation. *Mon. Wea. Rev.* **139**, 3964–3973.

Lei, L. and Anderson, J. 2014. Empirical localization of observations for serial ensemble Kalman filter data assimilation in an atmospheric general circulation model. *Mon. Wea. Rev.* **142**, 1835–1851.

Lermusiaux, P. F. J. 2007. Adaptive modelling, adaptive data assimilation and adaptive sampling. *Physica D* **230**, 172–196.

Lermusiaux, P. F. J. and Robinson, A. R. 1999. Data assimilation via error subspaces statistical estimation, part I: theory and schemes. *Mon. Wea. Rev.* **127**, 1385–1407.

Lermusiaux, P. F. J., Robinson, A. R., Haley, P. J. and Leslie, W. G. 2002. Advanced interdisciplinary data assimilation: filtering and smoothing via error subspace statistical estimation. *Proceedings of "The OCEANS 2002 MTS/IEEE conference, Holland"*, Vol. 230, pp. 795–802.

Livings, D. 2005. *Aspects of the ensemble Kalman filter* [Master's thesis] Department of Mathematics, University of Reading, UK.

Livings, D., Dance, S. L. and Nichols, N. K. 2008. Unbiased ensemble square root filters. *Physica D* **237**, 1021–1028.

Lorenc, A. C. 2003. The potential of the ensemble Kalman filter for NWP - a comparison with 4D-Var. *Q. J. R. Meteorol. Soc.* **129**, 3183–3203.

Metref, S., Cosme, E., Snyder, C. and Brasseur, P. 2014a. A non-Gaussian analysis scheme using rank histograms for ensemble data assimilation. *Nonlin. Processes Geophys.* **21**, 869–885.

Metref, S., Cosme, E., Snyder, C. and Brasseur, P. 2014b. A non-Gaussian analysis scheme using rank histogram for ensemble data assimilation. *Nonlinear Proc. Geophys.* **21**, 869–885.

Minamide, M. and Zhang, F. 2017. Adaptive observation error inflation for assimilating all-sky satellite radiance. *Mon. Wea. Rev.* **145**, 1063–1081.

Miyoshi, T. and Yamane, S. 2007. Local ensemble transform Kalman filter with an AGCM at a T159/L48 resolution. *Mon. Wea. Rev.* **135**, 3841–3861.

Morzfeld, M., Tu, X., Atkins, E. and Chorin, A. J. 2012. A random map implementation of implicit filters. *J. Comput. Phys.* **231**, 2049–2066.

Morzfeld, M., Hodyss, D. and Snyder, C. 2017. What the collapse of the ensemble Kalman filter tells us about particle filters. *Tellus A* **69**, 1–14.

Nakano, S., Ueno, G. and Higuchi, T. 2007. Merging particle filter for sequential data assimilation. *Nonlinear Process. Geophys.* **14**, 395–408.

Nerger, L. 2004. *Parallel Filter Algorithms for Data Assimilation in Oceanography* Number 487 in Reports on Polar and Marine Research, Alfred Wegener Institute for Polar and Marine Research, Bremerhaven, Germany, [PhD Thesis]. University of Bremen, Germany.

Nerger, L. 2015. On serial observation processing on localized ensemble Kalman filters. *Mon. Wea. Rev.* **143**, 1554–1567.

Nerger, L. and Hiller, W. 2013. Software for ensemble-based data assimilation systems - implementation strategies and scalability. *Comput. Geosci.* **55**, 110–118.

Nerger, L., Hiller, W. and Schröter, J. 2005a. PDAF – the parallel data assimilation framework: experiences with Kalman filtering. In: *Use of High Performance Computing in Meteorology: Proceedings of the Eleventh ECMWF Workshop on the Use of High Performance Computing in Meteorology, Reading, UK, 25–29 October 2004* (eds. W. Zwieflhofer and G. Mozdzynski). World Scientific, Singapore, pp. 63–83.

Nerger, L., Hiller, W. and Schröter, J. 2005b. A comparison of error subspace Kalman filters. *Tellus* **57A**, 715–735.

Nerger, L., Danilov, S., Hiller, W. and Schröter, J. 2006. Using sea level data to constrain a finite-element primitive-equation ocean model with a local SEIK filter. *Ocean Dyn.* **56**, 634–649.

Nerger, L., Janjić, T., Schroeter, J. and Hiller, W. 2012a. A unification of ensemble square root filters. *Mon. Wea. Rev.* **140**, 2335–2345.

Nerger, L., Janjić, T., Schröter, J. and Hiller, W. 2012b. A regulated localization scheme for ensemble-based Kalman filters. *Q. J. Roy. Meteor. Soc.* **138**, 802–812.

Ott, E., Hunt, B. R., Szunyogh, I., Zimin, A. V., Kostelich, E. J., and co-authors. 2004. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus* **56A**, 415–428.

Patil, D. J., Hunt, B. R., Kalnay, E., Yorke, J. A. and Ott, E. 2001. Local low dimensionality of atmospheric dynamics. *Phys. Rev. Lett.* **86**(26), 5878–5881.

Penny, S. and Miyoshi, T. 2016. A local particle filter for high-dimensional geophysical systems. *Nonlinear Processes Geophys.* **23**, 391–405.

Perianez, A., Reich, H. and Potthast, R. 2014. Optimal localization for ensemble Kalman filter systems. *J. Meteorol. Soc. Jpn.* **92**, 585–597.

Petrie, R. E. and Dance, S. L. 2010. Ensemble-based data assimilation and the localisation problem. *Weather* **65**(3), 65–69.

Pham, D. T. 2001. Stochastic methods for sequential data assimilation in strongly nonlinear systems. *Mon. Wea. Rev.* **129**, 1194–1207.

Pham, D. T., Verron, J. and Gourdeau, L. 1998a. Singular evolutive Kalman filters for data assimilation in oceanography. *C. R. Acad. Sci. Ser. II* **326**(4), 255–260.

Pham, D. T., Verron, J. and Roubaud, M. C. 1998b. A singular evolutive extended Kalman filter for data assimilation in oceanography. *J. Mar. Syst.* **16**, 323–340.

Poterjoy, J. 2016a. A localized particle filter for high-dimensional nonlinear systems. *Mon. Wea. Rev.* **144**, 59–76.

Poterjoy, J. and Anderson, J. L. 2016b. Efficient sssimilation of simulated observations in a high-dimensional geophysical system using a localized particle filter. *Mon. Wea. Rev.* **144**, 2007–2020.

Reich, S. 2013. A nonparametric ensemble transform method for Bayesian inference. *SIAM Journal on Scientific Computing* **4**(35), 2013–2024.

Sakov, P. and Oke, P. R. 2008. Implications of the form of the ensemble transformation in the ensemble square root filters. *Mon. Wea. Rev.* **136**, 1042–1053.

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

Silverman, B. W. 1986. *Density estimation for statistics and data analysis* Chapman and Hall, New Work.

Snyder, C., Bengtsson, T., Bickel, P. and Anderson, J. 2008. Obstacles to high-dimensional particle filtering. *Mon. Wea. Rev.* **136**, 4629–4640.

Snyder, C., Bengtsson, T. and Morzfeld, M. 2015. Performance bounds for particle filters using the optimal proposal. *Mon. Wea. Rev.* **143**(11), 4750–4761.

Stordal, A. S., Karlsen, H. A., Nævdal, G., Skaug, H. J. and Valles, B. 2011. Bridging the ensemble Kalman filter and particle filters: the adaptive Gaussian mixture filter. *Comput. Geosci.* **15**, 293–305.

Testut, C.-E., Brasseur, P., Brankart, J.-M. and Verron, J. 2003. Assimilation of sea-surface temperature and altimetric observations

during 1992–1993 inot an eddy permitting primitive equation model of the North Atlantic ocean. *J. Mar. Sys.* **40–41**, 291–316.

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

Tödter, J. and Ahrens, B. 2015. A Second-Order Exact Ensemble Square Root Filter for Nonlinear Data Assimilation. *Mon. Wea. Rev.* **143**(4), 1347–1367.

Tödter, J., Kirchgessner, P., Nerger, L. and Ahrens, B. 2016. Assessment of a nonlinear ensemble transform filter for high-dimensional data assimilation. *Mon. Wea. Rev.* **144**, 409–427.

Tong, X. T., Majda, A. J. and Kelly, D. 2016. Nonlinear stability and ergodicity of ensemble based Kalman filters. *Nonlinearity* **29**, 657–691.

van Leeuwen, P. J. 2003a. A variance-minimizing filter for nonlinear dynamics. *Mon. Wea. Rev.* **131**, 2071–2084.

van Leeuwen, P. J. 2003b. Nonlinear ensemble data assimilation for the ocean. *Recent Developments in data assimilation for atmosphere and ocean, ECMWF Seminar 8–12 September 2003*, Reading, United Kingdom, pp. 265–286.

van Leeuwen, P. J. 2009. Particle filtering in geophysical systems. *Mon. Wea. Rev.* **137**, 4089–4114.

van Leeuwen, P. J. 2010. Nonlinear data assimilation in Geosciences: an extremely efficient particle filter. *Q. J. R. Meteorol. Soc.* **136**, 1991–1999.

van Leeuwen, P. J. 2011. Efficient nonlinear data-assimilation in geophysical fluid dynamics. *Comput. Fluids* **46**, 52–58.

van Leeuwen, P. J., Cheng, Y. and Reich, S. 2015. Nonlinear data assimilation. *Frontiers in Applied Dynamical Systems: Reviews and Tutorials 2* Springer.

van Leeuwen, P. J. and Evensen, G. 1996. Data assimilation and inverse methods in terms of a probabilistic formulation. *Mon. Wea. Rev.* **124**, 2898–2913.

Verlaan, M. and Heemink, A. W. 2001. Nonlinearity in data assimilation applications: a practical method for analysis. *Mon. Wea. Rev.* **129**(6), 1578–1589.

Vossepoel, F. C. and Van Leeuwen, P. J. 2006. Parameter estimation using a particle method: inferring mixing coefficients from sea-level observations. *Mon. Wea. Rev.* **135**, 1006–1020.

Wang, Y., Jung, Y., Supine, T. A. and Xue, M. 2013. A hybrid MPI-OpenMP parallel algorithm and performance analysis for an ensemble square root filter designed for multiscale observations. *J. Atm. and Oce. Tech.* **30**, 1382–1397.

Whitaker, J. S. and Hamill, T. M. 2002. Ensemble data assimilation without perturbed observations. *Mon. Wea. Rev.* **130**, 1913–1924.

Wikle, C. K. and Berliner, L. M. 2006. A Bayesian tutorial for data assimilation. *Physica D*, **230**(1):1–16.

Xiong, X., Navon, I. M. and Uzunoglu, B. 2006. A note on the particle filter with posterior Gaussian resampling. *Tellus A* **58**(4), 456–460.

Zhen, Y. and Zhang, F. 2014. A probabilistic approach to adaptive covariance localization for serial ensemble square root filters. *Mon. Wea. Rev.* **142**, 4499–4518.

Zhu, M., van Leeuwen, P. J. and Amezcua, J. 2016. Implicit equal-weights particle filter. *Q. J. R. Meteorol. Soc.* page personal communication.

Zupanski, M. 2005. Maximum likelihood ensemble filter: Theoretical aspects. *Mon. Wea. Rev.* **133**, 1710–1726.

Zupanski, M., Michael, N. I. and Zupanski, D. 2008. The maximum likelihood ensemble filter as a non-differentiable minimization algorithm. *Q. J. R. Meteorol. Soc.* **134**, 1039–1050.

Appendix 1. Resampling methods

In this section, we give descriptions of a number of resampling techniques that can be applied to the particle filter and Gaussian mixture filter methods to turn weighted particles into equal-weight particles. The resampling techniques included here are probabilistic resampling, stochastic universal resampling and residual resampling. However, they are by no means exclusive and other techniques could be used.

A.1. Probabilistic resampling (PR)

The probabilistic resampling or the basic random resampling is the most straightforward to implement as we sample directly from the density given by the weights.

Given the weights $\{w_j\}_{j=1}^{N_e}$ associated with the ensemble of particles, where the sum of weights is equal to one, the total number of particles N_e and the number of particles to be generated $\tilde{N}_e$, we generate an index of the sampled particles using the Algorithm 1.

Algorithm 1 Algorithm of probabilistic resampling

function PR(w, N_e, $\tilde{N}_e$)
  $\hat{w}_1 \leftarrow w_1$
  for $j \leftarrow 2$ to N_e **do** ▷ compute cumulative weights
    $\hat{w}_j = \sum_{i=1}^{j} w_j$
  end for
  $c \leftarrow 1$
  for $j \leftarrow 1$ to $\tilde{N}_e$ **do**
    $u \sim \mathcal{U}[0, \tilde{N}_e]$ ▷ generate a random number
    while $u > \hat{w}_c$ **do**
      $c \leftarrow c + 1$
    end while
    $I_j \leftarrow c$ ▷ assign an index of the sampled particle
    $c \leftarrow 1$
  end for
  return I
end function

The required input for the PR is: $w \in \mathcal{R}^{N_e}$ a vector of particle weights, N_e the total number of particles in the filter, and $\tilde{N}_e$ the number of particles to be sampled and the method returns an index $I \in \mathcal{R}^{\tilde{N}_e}$, which can then be used to select the sampled particles $\mathbf{x}_j^* = \mathbf{x}_{I(j)}$ for $j = 1 : \tilde{N}_e$.

Note that this scheme introduces sampling noise by drawing $\tilde{N}_e$ times from a uniform distribution.

A.2. Stochastic universal resampling (SUR)

Stochastic universal resampling is also known as systematic resampling. It performs resampling in the same way as the basic random resampling algorithm except instead of drawing each u_j independently from $\mathcal{U}(0,1)$ for $j = 1, \ldots, N_e$, it uses a uniform random number u according to $u \sim \mathcal{U}[0, 1/N_e]$ and $u_j = u + (j-1)/N_e$ (Bolić et al., 2003).

Given the weights $\{w_j\}_{j=1}^{N_e}$ associated with the ensemble of particles, where the sum of weights is equal to one, the total number of particles N_e and the number of particles to be generated $\tilde{N}_e$, we generate an index of the sampled particles using the Algorithm 2.

Algorithm 2 Algorithm of stochastic universal resampling

function SUR($w, N_e, \tilde{N}_e$)
  $\hat{w}_1 \leftarrow w_1$
  for $j \leftarrow 2$ to N_e **do** ▷ compute cumulative weights
    $\hat{w}_j = \sum_{i=1}^{j} w_j$
  end for
  $u \sim \mathcal{U}[0, 1/N_e]$ ▷ generate a random number
  $c \leftarrow 1$
  for $j \leftarrow 1$ to $\tilde{N}_e$ **do**
    while $u > \hat{w}_c$ **do**
      $c \leftarrow c + 1$
    end while
    $I_j \leftarrow c$ ▷ assign an index of the sampled particle
    $u \leftarrow u + 1/N_e$
    $c \leftarrow 1$
  end for
  return I
end function

The required input for the SUR is: $w \in \mathcal{R}^{N_e}$ a vector of particle weights, N_e the total number of particles in the filter, and $\tilde{N}_e$ the number of particles to be sampled and the method returns an index $I \in \mathcal{R}^{\tilde{N}_e}$ which can then be used to select the sampled particles $\mathbf{x}_j^* = \mathbf{x}_{I(j)}$ for $j = 1 : \tilde{N}_e$.

Note, that this method has a lower sampling noise than probabilistic resampling since only one random variable is drawn.

A.3. Residual resampling (RR)

The RR algorithm samples the particles in two parts. In the first part the number of replications of particles is calculated, but since the method does not guarantee that the number of resampled particles is N_e, the residual N_r is computed. The second step requires resampling, which produces N_r of the final $\tilde{N}_e$ particles. In Algorithm 3 this is done by PR, but other resampling technique can be used.

The required input for the RR is: $w \in \mathcal{R}^{N_e}$ a vector of particle weights, N_e the total number of particles in the filter, and $\tilde{N}_e$ the number of particles to be sampled and the method returns an index $I \in \mathcal{R}^{\tilde{N}_e}$ which can then be used to select the sampled particles $\mathbf{x}_j^* = \mathbf{x}_{I(j)}$ for $j = 1 : \tilde{N}_e$. Note, that we used the PR method to obtain an array $IR \in \mathcal{R}^{N_r}$ with the indices of the additional sampled particles, which we then stored in the remaining empty cells of the index array $I \in \mathcal{R}^{N_e}$.

Note, that this method reduces the sampling noise, but not as much as the SUR method.

Algorithm 3 Algorithm of residual resampling

function RR($w, N_e, \tilde{N}_e$)
  for $j \leftarrow 1$ to N_e **do**
    $\hat{w}_j \leftarrow \lfloor w_j \cdot \tilde{N}_e \rfloor$ ▷ the integer part of $w \cdot \tilde{N}_e$
  end for
  $N_r \leftarrow \tilde{N}_e$
  $c \leftarrow 1$ ▷ counter
  for $j \leftarrow 1$ to $\tilde{N}_e$ **do**
    if $\hat{w}_j > 0$ **then**
      $I_{c \text{ to } c+\hat{w}_j} \leftarrow j$ ▷ select copies of index to sample
      $c \leftarrow c + \hat{w}_j$
    end if
  end for

  if $N_r > 0$ **then**
    for $j \leftarrow 1$ to N_e **do**
      $\tilde{w}_j \leftarrow (w_j - \hat{w}_j)/N_r$ ▷ compute residual weights and normalise
    end for
    $IR \leftarrow PR(\hat{w}, N_e, N_r)$ ▷ sample additional indices
    $I_{c \text{ to } N_e} \leftarrow IR$ ▷ store extra indices at end of array I
  end if
  return I
end function

Appendix 2. Filter algorithms for practical implementation

This Appendix contains practical pseudo-algorithms of all the ensemble filter methods presented in Sections 5–7. To discuss the computational cost of each method in Section 9.4 we used the algorithms presented in this appendix because for some filter methods they are more computationally efficient or numerically stable than mathematically elegant versions given in Section 5. The algorithms are written in the way one would implement them efficiently in Fortran. For compactness, the algorithms don't show that the final step of the ensemble filters can usually be written in a blocked form, so that only the allocation one large ensemble array $\mathbf{X}$ is required. This is different for the MLEF, where two arrays of size $N_x \times N_e$ are required. If indices are given for matrices, the notation follows Fortran in that the first index defines the row, while the second index specifies the column.

Note, that in all the algorithms that follow in this appendix any variable with a number subscript is a temporary variable used to

Algorithm 4 EnKF (for $N_y > N_e/2$, see Section 5.1)

$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{HX}^f \leftarrow \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ Ens. predicted obs.; $(N_y \times N_e)$
$\overline{\mathbf{HX}}^f \leftarrow$ mean of $\mathbf{HX}^f$ ▷ $(N_y \times 1)$
$\mathbf{HX}'^f \leftarrow \mathbf{HX}^f - \overline{\mathbf{HX}}^f$ ▷ Ens. predicted pert.; $(N_y \times N_e)$
$\mathbf{HPH} \leftarrow \frac{1}{N-1}\mathbf{HX}'^f\left(\mathbf{HX}'^f\right)^{\mathsf{T}}$ ▷ $(N_y \times N_y)$
$\mathbf{A} \leftarrow \mathbf{HPH} + \mathbf{R}$ ▷ $(N_y \times N_y)$
Gen. obs. ensemble $\mathbf{Y}$ ▷ $(N_y \times N_e)$
$\mathbf{D} \leftarrow \mathbf{Y} - \mathbf{HX}^f$ ▷ $(N_y \times N_e)$
Solve $\mathbf{AC} = \mathbf{D}$ for $\mathbf{C}$ ▷ $(N_y \times N_e)$
$\mathbf{E} \leftarrow \left(\mathbf{HX}'^f\right)^{\mathsf{T}}\mathbf{C}$ ▷ $(N_y \times N_e)$
$\mathbf{X}' \leftarrow \mathbf{X}^f - \overline{\mathbf{x}}$ ▷ $(N_y \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + (N-1)\mathbf{X}'\mathbf{E}$ ▷ $(N_y \times N_e)$

Algorithm 5 SEIK, see Section 5.2

$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{H}\overline{\mathbf{x}}^f \leftarrow \mathcal{H}\left(\overline{\mathbf{x}}^f\right)$ ▷ Ens. predicted obs.; $(N_y \times 1)$
$\mathbf{d} \leftarrow \mathbf{y} - \mathbf{H}\overline{\mathbf{x}}^f$ ▷ $(N_y \times 1)$
$\mathbf{HX}^f \leftarrow \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ $(N_y \times N_e)$
$\mathbf{HL} \leftarrow (\mathbf{HX})\,\mathbf{L}$ ▷ $(N_y \times N_e - 1)$
$\mathbf{B}_1 \leftarrow \mathbf{R}^{-1}\mathbf{HL}$ ▷ $(N_y \times N_e - 1)$
Initialise $\mathbf{C}_1 \leftarrow (N-1)\mathbf{A}^{\mathsf{T}}\mathbf{A}$ ▷ $(N_e - 1 \times N_e - 1)$
$\mathbf{C}_2 \leftarrow \rho\mathbf{C}_1 + (\mathbf{HL})^{\mathsf{T}}\mathbf{B}_1$ ▷ $(N_e - 1 \times N_e - 1)$
$\mathbf{d}_1 \leftarrow \mathbf{B}_1\mathbf{d}$ ▷ $(N_e - 1 \times 1)$
Solve $\mathbf{C}_2\mathbf{e} = \mathbf{d}_1$ ▷ $(N_e - 1 \times 1)$
$\overline{\mathbf{w}} \leftarrow \mathbf{Ae}$ ▷ $(N_e \times 1)$
$\mathbf{TT}^{\mathsf{T}} \leftarrow \mathbf{C}_2$ ▷ Cholesky decomp.; $(N_e - 1 \times N_e - 1)$
Initialise $\boldsymbol{\Omega}$ ▷ $(N_e - 1 \times N_e)$
Solve $\mathbf{TI} = \mathbf{R}$ ▷ $(N_e - 1 \times N_e)$
$\mathbf{W}' \leftarrow \frac{1}{\sqrt{N-1}}\mathbf{AT}$ ▷ $(N_e \times N_e)$
$\mathbf{W} \leftarrow \mathbf{W}' + \overline{\mathbf{w}}$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + \mathbf{X}'^f\mathbf{W}$ ▷ $(N_x \times N_e)$

reduce the computational time and storage space needed for the algorithm. Further, for ease of reading these algorithms, we use abbreviations: SVD for singular value decomposition and EVD for eigenvalue decomposition. The values in the right column of each algorithm give the dimension of the resulting array, which helps to determine the computational cost of the operations.

Below, we use the notation $\mathcal{H}\left(\mathbf{X}^f\right)$ as a shorthand notation for applying the possibly non-linear observation operator $\mathcal{H}$ individually to each ensemble state in $\mathbf{X}^f$.

Algorithm 6 ESSE, see Section 5.3

$\mathbf{d} \leftarrow \mathbf{y} - \mathbf{H}(\overline{\mathbf{x}}^f)$ ▷ $(N_y \times 1)$
$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
for $j = 1$ to N_e **do** ▷ Form ens. pert. matr.
  $\mathbf{X}'^f_j \leftarrow \mathbf{x}^f_j - \overline{\mathbf{x}}^f$ ▷ $(N_x \times N_e)$
end for
$\mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{T}} \leftarrow \mathbf{X}'^f$ ▷ Compute SVD; $(N_x \times N_e)$
$\mathbf{E} \leftarrow \frac{1}{N_e-1}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T$ ▷ Normalised e.g.values; $(N_x \times N_x)$
$q_1 \leftarrow 2$ ▷ (1×1)
$\mathbf{U}_q \leftarrow \mathbf{U}(1 : q_1, :)$ ▷ $(N_x \times q_1)$
$\mathbf{E}_q \leftarrow \mathbf{E}(1 : q_1, 1 : q_1)$ ▷ $(q_1 \times q_1)$
$r \leftarrow 1 - \epsilon$ ▷ Set tolerance; (1×1)
repeat ▷ Find min. covar. matr.
  $q_1 \leftarrow q_1 + 1$ ▷ (1×1)
  $\mathbf{U}_{q1} \leftarrow \mathbf{U}(1 : q_1, :)$ ▷ $(N_x \times q_1)$
  $\mathbf{E}_{q1} \leftarrow \mathbf{E}(1 : q_1, 1 : q_1)$ ▷ $(q_1 \times q_1)$
  $\mathbf{A} \leftarrow \mathbf{E}_{q1}^{1/2}\mathbf{U}_{q1}^{\mathsf{T}}\mathbf{U}_q\mathbf{E}_q^{1/2}$ ▷ $(q \times q_1)$
  $\rho \leftarrow \mathrm{Tr}(A)/\mathrm{Tr}(E_q)$ ▷ (1×1)
  $\mathbf{U}_q \leftarrow \mathbf{U}_{q1}$ ▷ $(N_x \times q_1)$
  $\mathbf{E}_q \leftarrow \mathbf{E}_{q1}$ ▷ $(q_1 \times q_1)$
until $\rho > r$
$\mathbf{V}_q^{\mathsf{T}} \leftarrow \mathbf{V}(:, 1 : q_1)^{\mathsf{T}}$ ▷ $(q_1 \times N_e)$
$\mathbf{C}_1 \leftarrow \mathbf{HU}_q$ ▷ $(N_y \times q_1)$
$\boldsymbol{\Gamma}\boldsymbol{\Sigma}\boldsymbol{\Gamma}^{\mathsf{T}} \leftarrow \mathbf{C}_1\mathbf{E}_q\mathbf{C}_1^{\mathsf{T}} + \mathbf{R}$ ▷ Compute EVD; $(N_y \times N_y)$
$\mathbf{Finv} \leftarrow \boldsymbol{\Gamma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Gamma}^{\mathsf{T}}$ ▷ $(N_y \times N_y)$
$\mathbf{S} \leftarrow \mathbf{C}_1\mathbf{E}_q^{1/2}\mathbf{V}_q^{\mathsf{T}}$ ▷ $(q1 \times N_y)$
$\mathbf{C}_2 \leftarrow \mathbf{S}^T\mathbf{Finv}$ ▷ $(q1 \times N_y)$
$\mathbf{A} \leftarrow \mathbf{I} - \mathbf{C}_2\,\mathbf{S}$ ▷ $(q_1 \times q_1)$
$\mathbf{Z}\boldsymbol{\Pi}\mathbf{Z}^{\mathsf{T}} \leftarrow \mathbf{A}$ ▷ Compute EVD; $(q1 \times q1)$
$\overline{\mathbf{w}} \leftarrow \frac{1}{\sqrt{N_e-1}}\mathbf{S}_q\mathbf{F}^{-1}\mathbf{d}$ ▷ $(q_1 \times 1)$
$\mathbf{W}' \leftarrow \mathbf{Z}\boldsymbol{\Pi}^{1/2}\mathbf{Z}^T$ ▷ $(q_1 \times q_1)$
$\mathbf{W} \leftarrow \mathbf{W}' + \overline{\mathbf{w}}$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + \mathbf{X}'^f\mathbf{W}$ ▷ $(N_x \times N_e)$

Algorithm 7 ETKF, see Section 5.4

$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{H}\overline{\mathbf{x}}^f \leftarrow \mathcal{H}\left(\overline{\mathbf{x}}^f\right)$ ▷ Ens. predicted obs.; $(N_y \times 1)$
$\mathbf{d} \leftarrow \mathbf{y} - \mathbf{H}\overline{\mathbf{x}}^f$ ▷ $(N_y \times 1)$
$\mathbf{HX}^f \leftarrow \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ $(N_y \times N_e)$
$\mathbf{HX}'^f \leftarrow \mathbf{HX}^f - \mathbf{H}\overline{\mathbf{x}}^f$ ▷ Ens. predic. obs. perturb.;$(N_y \times N_e)$
$\mathbf{C} \leftarrow \mathbf{R}^{-1}\mathbf{HX}'^f$ ▷ $(N_y \times N_e)$
$\mathbf{A}_1 \leftarrow (N-1)\mathbf{I}$ ▷ $(N_e \times N_e)$
$\mathbf{A}_2 \leftarrow \mathbf{A}_1 + \left(\mathbf{HX}'^f\right)^\mathsf{T}\mathbf{C}$ ▷ $(N_e \times N_e)$
$\mathbf{X}'^f \leftarrow \mathbf{X}^f - \overline{\mathbf{X}}^f$ ▷ Ens. perturb.; $(N_x \times N_e)$
$\mathbf{D} \leftarrow \mathbf{C}^\mathsf{T}\mathbf{d}$ ▷ $(N_e \times 1)$
$\mathbf{U\Sigma U}^\mathsf{T} \leftarrow \mathbf{A}_2$ ▷ Compute EVD; $(N_e \times N_e)$
$\mathbf{w}_1 \leftarrow \mathbf{U}^\mathsf{T}\mathbf{D}$ ▷ $(N_e \times 1)$
for $j = 1$ to N_e **do** ▷ Scale for each ens. member
  $\mathbf{w}_2(j) \leftarrow \mathbf{w}_1(j)\mathbf{\Sigma}^{-1}(j,j)$ ▷ $(N_e \times 1)$
end for
$\overline{\mathbf{w}} \leftarrow \mathbf{U}\mathbf{w}_2$ ▷ $(N_e \times 1)$
for $j = 1$ to N_e **do** ▷ Scale for each ens. member
  $\mathbf{W}'_1(:,j) \leftarrow \sqrt{\mathbf{\Sigma}(j,j)}\mathbf{U}(:,j)$ ▷ $(N_e \times N_e)$
end for
$\mathbf{W}' \leftarrow \mathbf{W}'_1\mathbf{U}^\mathsf{T}$ ▷ $(N_e \times N_e)$
$\mathbf{W} \leftarrow \mathbf{W}' + \overline{\mathbf{w}}$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + \mathbf{X}'^f\mathbf{W}$ ▷ $(N_x \times N_e)$

Algorithm 8 EAKF, see Section 5.5

$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{H}\overline{\mathbf{x}}^f \leftarrow \mathcal{H}\left(\overline{\mathbf{x}}^f\right)$ ▷ Ens. predicted obs.; $(N_y \times 1)$
$\mathbf{d} \leftarrow \mathbf{y} - \mathbf{H}\overline{\mathbf{x}}^f$ ▷ $(N_y \times 1)$
$\mathbf{HX}^f \leftarrow \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ $(N_y \times N_e)$
$\mathbf{HX}'^f \leftarrow \mathbf{HX}^f - \mathbf{H}\overline{\mathbf{x}}^f$ ▷ Ens. predic. obs. perturb.;$(N_y \times N_e)$
$\mathbf{X}'^f \leftarrow \mathbf{X}^f - \overline{\mathbf{X}}^f$ ▷ Ens. perturb.; $(N_x \times N_e)$
$\tilde{\mathbf{S}} \leftarrow \frac{1}{\sqrt{N-1}}\mathbf{R}^{-1}\mathbf{HX}'^f$ ▷ $(N_y \times N_e)$
$\mathbf{U\Sigma V} \leftarrow \tilde{\mathbf{S}}$ ▷ Compute SVD; $(N_y \times N_e)$
$\mathbf{Z\Gamma G} \leftarrow \mathbf{X}'^f$ ▷ Compute SVD; $(N_x \times N_e)$
$\mathbf{W}'_1 \leftarrow \mathbf{ZX}'^f$ ▷ $(N_e \times N_e)$
for $j = 1$ to N_e **do** ▷ Scale for each ens. member
  $\mathbf{b}(j) \leftarrow 1 + \mathbf{\Sigma}^2(j,j)$ ▷ $(N_e \times 1)$
  $\mathbf{W}'_2(:,j) \leftarrow \frac{1}{\sqrt{\mathbf{\Gamma}(j,j)\mathbf{b}(j)}}\mathbf{W}'_1(:,j)$ ▷ $(N_e \times N_e)$
end for
$\mathbf{W}' \leftarrow \mathbf{UW}'_2$ ▷ $(N_e \times N_e)$
for $l = 1$ to N_y **do** ▷ Scale for each onbservation
  $\mathbf{c}_1(l) \leftarrow \frac{1}{\mathbf{b}(l)\sqrt{R(l,l)}}\mathbf{d}(l)$ ▷ $(N_y \times 1)$
end for
$\mathbf{c}_2 \leftarrow \mathbf{V}\mathbf{c}_1$ ▷ $(N_y \times 1)$
for $j = 1$ to N_e **do**
  $\mathbf{c}_3(j) \leftarrow \mathbf{\Sigma}(j,j)\mathbf{c}_2(j)$ ▷ $(N_e \times 1)$
end for
$\overline{\mathbf{w}} \leftarrow \frac{1}{\sqrt{N-1}}\mathbf{U}\mathbf{c}_3$ ▷ $(N_e \times 1)$
$\mathbf{W} \leftarrow \mathbf{W}' + \overline{\mathbf{w}}$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + \mathbf{X}'^f\mathbf{W}$ ▷ $(N_x \times N_e)$

Algorithm 9 EnSRF, see Section 5.6

$\overline{\mathbf{x}}^f \leftarrow$compute mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{X}'^f_j \leftarrow \mathbf{x}_j - \overline{\mathbf{x}}$, for $j = 1, \ldots, N_e$ ▷ $(N_x \times N_e)$
$\mathbf{S} \leftarrow \mathbf{HX}'$ ▷ $(N_y \times N_e)$
$\mathbf{I}_1 \leftarrow \mathbf{SS}^\mathsf{T}$ ▷ $(N_y \times N_y)$
$\mathbf{I}_2 \leftarrow \mathbf{I}_1 + (N-1)\mathbf{R}$ ▷ $(N_y \times N_y)$
$\mathbf{\Gamma\Lambda\Gamma}^\mathsf{T} \leftarrow \mathbf{I}_2$ ▷ Compute EVD; $(N_y \times N_y)$
$\mathbf{G}_1 \leftarrow \frac{1}{\mathbf{\Lambda}}\mathbf{\Gamma}$ ▷ $(N_y \times N_y)$
$\mathbf{G}_2 \leftarrow \mathbf{S}^\mathsf{T}\mathbf{G}_1$ ▷ $(N_e \times N_y)$
$\mathbf{U\Sigma V} \leftarrow \mathbf{G_2}$ ▷ Compute SVD; $(N_e \times N_y)$
$\mathbf{A} \leftarrow \sqrt{\mathbf{I} - \mathbf{\Sigma}^2}$ ▷ $(1 \times N_e)$
$\mathbf{W}'_1 \leftarrow \mathbf{UA}$ ▷ $(N_e \times N_y)$
$\mathbf{W}'_2 \leftarrow \mathbf{W}'_1\mathbf{U}^\mathsf{T}$ ▷ $(N_e \times N_e)$
$\mathbf{d} \leftarrow \mathbf{y} - \mathcal{H}\left(\overline{\mathbf{x}}\right)$ ▷ $(N_y \times 1)$
$\overline{\mathbf{w}}_1 \leftarrow \mathbf{\Gamma}^\mathsf{T}\mathbf{d}$ ▷ $(N_y \times 1)$
$\overline{\mathbf{w}}_2 \leftarrow \mathbf{\Lambda}^{-1}\overline{\mathbf{w}}_1$ ▷ $(N_y \times 1)$
$\overline{\mathbf{w}}_3 \leftarrow \mathbf{\Gamma}\overline{\mathbf{w}}_2$ ▷ $(N_y \times 1)$
$\overline{\mathbf{w}}_4 \leftarrow \mathbf{S}\overline{\mathbf{w}}_3$ ▷ $(N_e \times 1)$
$\mathbf{W} \leftarrow \mathbf{W}'_2 + [\mathbf{w}_4, \ldots, \mathbf{w}_4]$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a_1 \leftarrow \left[\overline{\mathbf{x}}^f, \ldots, \overline{\mathbf{x}}^f\right]$ ▷ form an $N_x \times N_y$ matrix with ens. forecast mean in each column
$\mathbf{X}^a \leftarrow \mathbf{X}^a_1 + \mathbf{X}'^f\mathbf{W}$ ▷ $(N_x \times N_e)$

Algorithm 10 Serial EnSRF, see Section 5.7

$\overline{\mathbf{x}}^f \leftarrow$ compute mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
for $i = 1$ to N_y **do** ▷ Loop over each single obs.
  $\mathbf{X}'_{io} \leftarrow \mathbf{X} - \overline{\mathbf{x}}$ ▷ Compute perturb.; $(N_x \times N_e)$
  $\mathbf{HX}_{io} \leftarrow \mathcal{H}(\mathbf{X})_{io}$ ▷ Ens. predicted obs; $(1 \times N_e)$
  $\overline{HX}_{io} \leftarrow$ mean of $\mathbf{HX}_{io}$ ▷ (1×1)
  $\mathbf{HX}'_{io} \leftarrow \mathbf{HX}_{io} - \overline{HX}_{io}$ ▷ $(1 \times N_e)$
  $\mathbf{HP} \leftarrow \frac{1}{N-1}\mathbf{HX}'_{io}\mathbf{X}'^\mathsf{T}_{io}$ ▷ $(1 \times N_x)$
  $HPH^\mathsf{T} \leftarrow \mathbf{HX}'_{io}\left(\mathbf{HX}'_{io}\right)^\mathsf{T}$ ▷ 1×1
  Localise $\mathbf{HP}$; optional
  $F \leftarrow HPH^\mathsf{T} + \sigma_{R,io}$ ▷ (1×1)
  $\mathbf{K} \leftarrow \frac{1}{F}\mathbf{HP}$ ▷ $(1 \times N_x)$
  $d \leftarrow y - \overline{HX}_{io}$ ▷ (1×1)
  $\alpha_1 \leftarrow 1 + \sqrt{\frac{\sigma_R}{F}}$ ▷ (1×1)
  $\alpha_2 \leftarrow \frac{1}{\alpha_1}$ ▷ (1×1)
  $\overline{\mathbf{x}}^a_{io} \leftarrow \overline{\mathbf{x}}^f_{io} + \mathbf{K}d$ ▷ $(N_x \times 1)$
  $\mathbf{X}'_{io} \leftarrow \mathbf{X}'_{io-1} - \alpha_2\mathbf{KHX}'$ ▷ $(N_x \times N_e)$
  $\mathbf{X}^a_{io} \leftarrow \mathbf{X}'_{io} + \overline{\mathbf{x}}^a_{io}$ ▷ $(N_x \times N_e)$
end for

Algorithm 11 SEnKF, see Section 5.8

$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{HX}^f \leftarrow \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ Ens. predicted obs.; $(N_y \times N_e)$
$\overline{\mathbf{HX}}^f \leftarrow$ mean of $\mathbf{HX}^f$ ▷ $(N_y \times 1)$
$\mathbf{HX}'^f \leftarrow \mathbf{HX}^f - \overline{\mathbf{HX}}^f$ ▷ Ens. predicted pert.; $(N_y \times N_e)$
Gen. obs. ensemble $\mathbf{Y}$ ▷ $(N_y \times N_e)$
$\mathbf{D} \leftarrow \mathbf{Y} - \mathbf{HX}^f$ ▷ $(N_y \times N_e)$
$\mathbf{A} \leftarrow \frac{1}{N-1}\mathbf{HX}'^f + \mathbf{D}$ ▷ $(N_y \times N_e)$
$\mathbf{U\Sigma V} \leftarrow \mathbf{A}$ ▷ Compute SVD; $(N_y \times N_e)$
$\mathbf{B}_1 \leftarrow \mathbf{U\Sigma}^{-1}$ ▷ $(N_y \times N_e)$
$\mathbf{I}_1 \leftarrow \mathbf{U}^\mathsf{T}\mathbf{D}$ ▷ $(N_e \times N_e)$
$\mathbf{I}_2 \leftarrow \mathbf{\Sigma}^{-2}\mathbf{I}_1$ ▷ $(N_e \times N_e)$
$\mathbf{I}_3 \leftarrow \frac{1}{N-1}\left(\mathbf{HX}'^f\right)^\mathsf{T}\mathbf{U}$ ▷ $(N_e \times N_e)$
$\overline{\mathbf{w}} \leftarrow \mathbf{I}_3\mathbf{I}_2$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + \frac{1}{N-1}\mathbf{X}'^f\overline{\mathbf{w}}$ ▷ $(N_x \times N_e)$

Algorithm 12 ESTKF, see Section 5.9

$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{H}\overline{\mathbf{x}}^f \leftarrow \mathcal{H}\left(\overline{\mathbf{x}}^f\right)$ ▷ Ens. predicted obs.; $(N_y \times 1)$
$\mathbf{d} \leftarrow \mathbf{y} - \mathbf{H}\overline{\mathbf{x}}^f$ ▷ $(N_y \times 1)$
$\mathbf{HX}^f \leftarrow \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ $(N_y \times N_e)$
$\mathbf{HL} \leftarrow (\mathbf{HX})\,\mathbf{L}$ ▷ $(N_y \times N_e - 1)$
$\mathbf{B}_1 \leftarrow \mathbf{R}^{-1}\mathbf{HL}$ ▷ $(N_y \times N_e - 1)$
Initialise $\mathbf{C}_1 \leftarrow (N-1)\mathbf{I}$ ▷ $(N_e - 1 \times N_e - 1)$
$\mathbf{C}_2 \leftarrow \rho\mathbf{C}_1 + (\mathbf{HL})^\mathsf{T}\mathbf{B}_1$ ▷ $(N_e - 1 \times N_e - 1)$
$\mathbf{d}_1 \leftarrow \mathbf{B}_1\mathsf{T}\mathbf{d}$ ▷ $(N_e - 1 \times 1)$
$\mathbf{U\Sigma U}^\mathsf{T} \leftarrow \mathbf{C}_2$ ▷ Compute EVD; $(N_e - 1 \times N_e - 1)$
$\mathbf{d}_2 \leftarrow \mathbf{U}^\mathsf{T}\mathbf{d}_1$ ▷ $(N_e - 1 \times 1)$
for $j = 1$ to $N-1$ **do**
  $\mathbf{d}_3(j) \leftarrow \mathbf{\Sigma}^{-1}(j,j)\mathbf{d}_2(j)$ ▷ $(N_e - 1 \times 1)$
  $\mathbf{T}_1(:\,j) \leftarrow \mathbf{\Sigma}^{-1/2}(j,j)\mathbf{U}(:,j)$ ▷ $(N_e - 1 \times N_e - 1)$
end for
$\overline{\mathbf{w}} \leftarrow \mathbf{U}\mathbf{d}_3$ ▷ $(N_e - 1 \times 1)$
$\mathbf{T}_2 \leftarrow \mathbf{T}_1\mathbf{U}^\mathsf{T}$ ▷ $(N_e - 1 \times N_e - 1)$
$\mathbf{W}' \leftarrow \mathbf{T}_2\mathbf{A}^\mathsf{T}$ ▷ $(N_e - 1 \times N_e)$
$\mathbf{W} \leftarrow \overline{\mathbf{w}} + \mathbf{W}'$ ▷ $(N_e - 1 \times N_e)$
$\mathbf{W}_A \leftarrow \mathbf{AW}$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f + \mathbf{X}'^f\mathbf{W}'$ ▷ $(N_x \times N_e)$

Algorithm 13 MLEF (using generalised non-linear conjugate-gradient), see Section 5.10

$\mathbf{C} \leftarrow (\mathbf{X}'^f)^\mathsf{T}\mathbf{H}^\mathsf{T}\mathbf{R}^{-1}\mathbf{HX}'^f$ ▷ $(N_e \times N_e)$
$\mathbf{F} \leftarrow \mathbf{I} + \mathbf{C}$ ▷ $(N_e \times N_e)$
$\mathbf{U\Lambda U}^\mathsf{T} \leftarrow \mathbf{F}$ ▷ Compute EVD; $(N_e \times N_e)$
$\mathbf{F_{inv}} \leftarrow \mathbf{U\Lambda}^{-1}\mathbf{U}^\mathsf{T}$ ▷ $(N_e \times N_e)$
$\mathbf{F_{inv2}} \leftarrow \mathbf{U\Lambda}^{-1/2}\mathbf{U}^\mathsf{T}$ ▷ $(N_e \times N_e)$
$\mathbf{Ghalf} \leftarrow \mathbf{X}'^f\mathbf{F_{inv2}}$ ▷ $(N_x \times N_e)$
$\mathbf{hx} \leftarrow \mathcal{H}(\mathbf{x}^f)$ ▷ $(N_y \times 1)$
for each particle $j = 1, \ldots, N_e$ **do**
  $\mathbf{X}^f(j) \leftarrow \mathbf{x}^f + \mathbf{X}'^f(j)$ ▷ $(N_x \times 1)$
  $\mathbf{HX}(j) \leftarrow \mathcal{H}(\mathbf{X}^f(j))$ ▷ $(N_y \times 1)$
  $\mathbf{Z}(j) \leftarrow \mathbf{HX}(j) - \mathbf{hx}$ ▷ $(N_y \times 1)$
end for
$\mathbf{d} \leftarrow \mathbf{y} - \mathbf{hx}$ ▷ $(N_y \times 1)$
$\mathbf{ZR} \leftarrow \mathbf{Z}^\mathsf{T}\mathbf{R}^{-1}$ ▷ $(N_e \times N_y)$
$\mathbf{b} \leftarrow \mathbf{ZR}\,\mathbf{d}$ ▷ $(N_e \times 1)$
$\beta_0 \leftarrow \mathbf{b}^\mathsf{T}\mathbf{b}$ ▷ (1×1)
$\boldsymbol{\xi} \leftarrow 0$ ▷ $(N_e \times 1)$
$\mathbf{x}^a \leftarrow \mathbf{x}^f$ ▷ $(N_x \times 1)$
repeat
  $\mathbf{hx} \leftarrow \mathcal{H}(\mathbf{x}^a)$ ▷ $(N_y \times 1)$
  for each particle $j = 1, \ldots, N_e$ **do**
    $\mathbf{X}(j) \leftarrow \mathbf{x}^a + \mathbf{X}'^f(j)$ ▷ $(N_x \times 1)$
    $\mathbf{HX}(j) \leftarrow \mathcal{H}(\mathbf{X}(j))$ ▷ $(N_y \times 1)$
    $\mathbf{Z}(j) \leftarrow \mathbf{HX}(j) - \mathbf{hx}$ ▷ $(N_y \times 1)$
  end for
  $\mathbf{d} \leftarrow \mathbf{y} - \mathbf{hx}$ ▷ $(N_y \times 1)$
  $\mathbf{ZR} \leftarrow \mathbf{Z}^\mathsf{T}\mathbf{R}^{-1}$ ▷ $(N_e \times p)$
  $\mathbf{db}_1 \leftarrow \mathbf{ZR}\,\mathbf{d}$ ▷ $(N_e \times 1)$
  $\mathbf{db}_2 \leftarrow \mathbf{F_{inv}}\,\boldsymbol{\xi}$ ▷ $(N_e \times 1)$
  $\mathbf{db}_2 \leftarrow \mathbf{db}_1 - \mathbf{db}_2$ ▷ $(N_e \times 1)$
  $\beta_1 \leftarrow \mathbf{db}_2^\mathsf{T}\mathbf{db}_2$ ▷ (1×1)
  $\beta = \beta_1/\beta_0$ ▷ (Fletcher-Reeves); (1×1)
    (Or use other formula for β, e.g. Polak-Ribiere.)
  $\mathbf{b} \leftarrow \mathbf{db}_2 + \beta\mathbf{b}$ ▷ $(N_e \times 1)$
  $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} + \alpha\mathbf{b}$ ▷ $(N_e \times 1)$
  $\mathbf{x}^a \leftarrow \mathbf{x}^f + \mathbf{Ghalf}\,\boldsymbol{\xi}$ ▷ $(N_x \times 1)$
  $\beta_0 \leftarrow \beta_1$ ▷ (1×1)
until Convergence
$\mathbf{hx} \leftarrow \mathcal{H}(\mathbf{x}^a)$ ▷ $(N_y \times 1)$
for each particle $j = 1, \ldots, N$ **do**
  $\mathbf{X}(j) \leftarrow \mathbf{x}^a + \mathbf{X}'^f(i)$ ▷ $(N_x \times 1)$
  $\mathbf{HX}(j) \leftarrow \mathcal{H}(\mathbf{X}(j))$ ▷ $(N_y \times 1)$
  $\mathbf{Z}(j) \leftarrow \mathbf{HX}(j) - \mathbf{hx}$ ▷ $(N_y \times 1)$
end for
$\mathbf{C} \leftarrow \mathbf{Z}^\mathsf{T}\mathbf{R}^{-1}\mathbf{Z}$ ▷ $(N_e \times N)$
$\mathbf{F} \leftarrow \mathbf{I} + \mathbf{C}$ ▷ $(N_e \times N_e)$
$\mathbf{U\Lambda U}^\mathsf{T} \leftarrow \mathbf{C}$ ▷ Compute EVD; $(N_e \times N_e)$
$\mathbf{F_{inv2}} \leftarrow \mathbf{U\Lambda}^{-1/2}\mathbf{U}^\mathsf{T}$ ▷ $(N_e \times N_e)$
$\mathbf{X}'^a \leftarrow \mathbf{X}'^f\mathbf{F_{inv2}}$ ▷ $(N_x \times 1)$

Algorithm 14 ETPF, see Section 6.3

$\mathbf{d}_i \leftarrow \mathbf{y} - \mathcal{H}\left(\mathbf{x}_i^f\right)$ ▷ For each particle; $(N_y \times N_e)$
$w_i = p(\mathbf{y}|\mathbf{x}_i^f)$ ▷ $(N_e \times 1)$
$J(T) \leftarrow \sum_{i,j}^{N_e} t_{ij} ||\mathbf{x}_i^f - \mathbf{x}_j^f||^2$
Solve for $\min_T J(T)$ with $t_{ij} \geq 0$ and $\sum_i^{N_e} t_{ij} = \frac{1}{N_e}$ and $\sum_j^{N_e} = w_i$
▷ $(N_e^2 \log N_e)$
$\overline{\mathbf{x}}^f \leftarrow$ mean of $\mathbf{X}^f$ ▷ $(N_x \times 1)$
$\mathbf{X}'^f = \mathbf{X}^f - \overline{\mathbf{x}}^f \mathbf{l}$ ▷ $(N_x \times N_e)$
$\mathbf{P} \leftarrow \mathbf{X}'^f \mathbf{X}'^{f\,T}$ ▷ $(N_x \times N_x)$
$\boldsymbol{\xi} \sim N(0, h^2\mathbf{P})$ ▷ $(N_x \times 1)$
$\mathbf{x}_j^a \leftarrow N \sum_i \mathbf{x}_i^f t_{ij}^* + \boldsymbol{\xi}_j$ ▷ $(N_x \times N_e)$

Algorithm 15 Relaxation step for EWPF and IEWPF algorithms (used in forecast step to nudge towards observations), see Section 6.2.3

$\mathbf{d}_j \leftarrow \mathbf{y} - \mathcal{H}\left(\mathbf{x}_j^f\right)$ ▷ For each particle; $(N_y \times N_e)$
Solve $\mathbf{R}\mathbf{e}_j = \mathbf{d}_j$ ▷ $(N_y \times N_e)$
$\mathbf{f}_j \leftarrow \tau(m)\mathbf{Q}^{1/2}\mathbf{H}^T\mathbf{e}_j$ ▷ $(N_x \times N_e)$
$\boldsymbol{\xi}_j \sim N(0, \mathbf{I})$ ▷ Random forcing $(N_x \times N_e)$
$\mathbf{x}_j^m \leftarrow \mathcal{M}\left(\mathbf{x}_j^{m-1}\right) + \mathbf{Q}^{1/2}(\mathbf{f}_j + \boldsymbol{\xi}_j)$ ▷ $(N_x \times N_e)$
$\log \mathbf{w}_j^m \leftarrow \log \mathbf{w}_j^{m-1} + \mathbf{f}_j^{\mathsf{T}}(\mathbf{f}_j + 2\boldsymbol{\xi}_j)$ ▷ $(N_x \times N_e)$

Algorithm 16 EWPF, see Section 6.2.1

$\mathbf{w}_{rest} \leftarrow -\log \mathbf{w}^{(m-1)}$ ▷ Weights from previous time steps $(1 \times N_e)$
$\epsilon \leftarrow 0.0001/N_e$ ▷ Choose parameter; (1×1)
$\gamma_U \leftarrow 10^{-6}$ ▷ has to be small; (1×1)
$\gamma_N \leftarrow \frac{2^{N_x/2}\epsilon\gamma_U^{N_x}}{\pi^{N_x/2}(1-\epsilon)}$ ▷ (1×1)
$k \leftarrow 0.8$ ▷ e.g. keep 80% particles; (1×1)
$N_k \leftarrow N_e k$ ▷ (1×1)
for $j = 1, \ldots, N_e$ **do** ▷ For each particle find max weights
  $\mathbf{d}_j \leftarrow \mathbf{y} - \mathcal{H}\left(\mathcal{M}(\mathbf{x}_j^{(m-1)})\right)$ ▷ $(N_y \times 1)$
  Solve $\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T + \mathbf{R}\right)\mathbf{e}_j = \mathbf{d}_j$ ▷ $(N_y \times 1)$
  $\phi_j \leftarrow \mathbf{d}_j^{\mathsf{T}}\mathbf{e}_j$ ▷ (1×1)
  $\mathbf{c}_j \leftarrow \mathbf{w}^{(m-1)} + 0.5\phi_j$ ▷ (1×1)
end for
$(\hat{\mathbf{c}}, \mathbf{idx}) \leftarrow sort(\mathbf{c})$ ▷ Sort max weights with **idx** holding sorted indices of **c**; $(N_e \times 1)$
$C_{max} \leftarrow \hat{\mathbf{c}}(N_k)$ ▷ Find abs. max weight; (1×1)
for $j = 1, \ldots, N_k$ **do** ▷ For each remaining particle
  $i \leftarrow \mathbf{idx}(j)$ ▷ go through each kept particle; (1×1)
  $\mathbf{K}_i \leftarrow \mathbf{Q}\mathbf{H}^T\mathbf{e}_i$ ▷ $(N_y \times 1)$
  $a_i \leftarrow \frac{1}{2}\mathbf{d}_i^T\mathbf{R}^{-1}\mathbf{H}\mathbf{K}_i$ ▷ (1×1)
  $r_i \leftarrow \mathbf{d}_i^T\mathbf{R}^{-1}\mathbf{d}_i$ ▷ (1×1)
  $b_i \leftarrow \frac{1}{2}r_i - C_{max} + w_{rest}(i)$ ▷ (1×1)
  $\alpha_i \leftarrow 1 + \sqrt{1 - b_i/a_i}$ ▷ (1×1)
  $\boldsymbol{\beta} \sim (1-\epsilon)\mathbf{Q}^{1/2}\mathcal{U}(-\gamma_U\mathbf{l}, +\gamma_U\mathbf{l}) + \epsilon\mathcal{N}\left(\gamma_N^2\mathbf{Q}\right)$ ▷ Random forcing $(N_x \times 1)$
  $\mathbf{x}_j^a \leftarrow \mathcal{M}\left(\mathbf{x}_i^{(m-1)}\right) + \alpha_i\mathbf{K}_i + \boldsymbol{\beta}$ ▷ $(N_x \times 1)$
  if $\boldsymbol{\beta}$ was from uniform distribution **then**
    $\mathbf{w}_j \leftarrow \mathbf{w}_{rest}(i) + (\alpha_i^2 - 2\alpha_i)a_i + \frac{1}{2}r_i$ ▷ (1×1)
  else
    $\mathbf{w}_1 \leftarrow \mathbf{w}_{rest}(i) + (\alpha_i^2 - 2\alpha_i)a_i$ ▷ (1×1)
    $\mathbf{w}_2 \leftarrow \mathbf{w}_1 + \frac{1}{2}r_i\left(2^{-N_x/2}\right)\left(\pi^{N_x/2}\right)$ ▷ (1×1)
    $\mathbf{w}_3 \leftarrow \mathbf{w}_2\gamma_N\gamma_U^{-N_x}(\frac{1-\epsilon}{\epsilon})$ ▷ (1×1)
    $\mathbf{w}_j \leftarrow \mathbf{w}_3 \exp\left(0.5\boldsymbol{\beta}_i^2\right)$ ▷ (1×1)
  end if
end for
Resample to have full ensemble, $\mathbf{X}^a$, of N_e particles using one of algorithms in Appendix 1. Resample from subset $\mathbf{x}^a \in \mathcal{R}^{N_x \times N_k}$ and $\mathbf{w} \in \mathcal{R}^{1 \times N_k}$. ▷ $(N_x \times N_e)$
$\mathbf{w} \leftarrow \frac{1}{N_e}$ ▷ $1 \times N_e$

Algorithm 17 IEWPF, see Section 6.2.2

$\mathbf{w}_{rest} \leftarrow -\log \mathbf{w}^{(m-1)}$ ▷ Weights from previous time steps $(1 \times N_e)$
for $j = 1, \ldots, N_e$ **do** ▷ For each particle find max weights
  $\mathbf{d}_j \leftarrow \mathbf{y} - \mathcal{H}\left(\mathcal{M}(\mathbf{x}_j^{(m-1)})\right)$ ▷ $(N_y \times 1)$
  Solve $\left(\mathbf{HQH}^T + \mathbf{R}\right)\mathbf{e}_j = \mathbf{d}_j$ ▷ $(N_y \times 1)$
  $\mathbf{db}_j \leftarrow \mathbf{d}_j^{\mathsf{T}}\mathbf{e}_j$ ▷ (1×1)
  $\mathbf{c}_j \leftarrow \mathbf{w}_{rest} + 0.5\mathbf{db}_j$ ▷ (1×1)
end for
$w_{target} \leftarrow \min(\mathbf{c})$ ▷ Keep all particles; (1×1)
$\mathbf{d} \leftarrow$ mean of $\mathbf{d}_j$ ▷ $(N_y \times 1)$
if $\mathbf{P}$ can be calculated directly **then**
  $\mathbf{P} = (\mathbf{Q}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}$ ▷ $(N_x \times N_x)$
  $\boldsymbol{\xi}_i \sim N(0, \mathbf{P})$ ▷ $(N_x \times N_e)$
else use e.g. ETKF algorithm:
  $\tilde{\Xi} \sim N(0, \mathbf{Q})$ ▷ Matrix of random vectors $(N_e \times N_x)$
  $\mathbf{C} \leftarrow \mathbf{R}^{-1}\mathbf{H}\tilde{\Xi}$ ▷ $(N_y \times N_e)$
  $\mathbf{A} \leftarrow (N-1)\mathbf{I} + \left(\mathbf{H}\mathbf{Ξ}'\right)^{\mathsf{T}}\mathbf{C}$ ▷ $(N_e \times N_e)$
  $\mathbf{D} \leftarrow \mathbf{C}^{\mathsf{T}}\mathbf{d}_j$ ▷ $(N_e \times 1)$
  $\mathbf{U\Sigma U}^{\mathsf{T}} \leftarrow \mathbf{A}$ ▷ Compute EVD; $(N_e \times N_e)$
  $\mathbf{w}_1 \leftarrow \mathbf{U}^{\mathsf{T}}\mathbf{D}$ ▷ $(N_e \times 1)$
  for $j = 1$ to N_e **do** ▷ Scale for each ens. member
    $\mathbf{w}_2(j) \leftarrow \mathbf{w}_1(j)\boldsymbol{\Sigma}^{-1}(j,j)$ ▷ $(N_e \times 1)$
  end for
  $\overline{\mathbf{w}} \leftarrow \mathbf{U}\mathbf{w}_2$ ▷ $(N_e \times 1)$
  for $j = 1$ to N_e **do** ▷ Scale for each ens. member
    $\mathbf{W}'_1(j,:) \leftarrow \sqrt{\boldsymbol{\Sigma}(j,j)}\mathbf{U}(j,:)$ ▷ $(N_e \times N_e)$
  end for
  $\mathbf{W}' \leftarrow \mathbf{W}'_1\mathbf{U}^{\mathsf{T}}$ ▷ $(N_e \times N_e)$
  $\mathbf{W} \leftarrow \mathbf{W}' + \overline{\mathbf{w}}$ ▷ $(N_e \times N_e)$
  $\Xi \leftarrow \tilde{\Xi} + \tilde{\Xi}\mathbf{W}$ ▷ $(N_x \times N_e)$
  $\boldsymbol{\xi}_j = \Xi_j$ ▷ $(N_x \times 1)$
end if
for $j = 1, \ldots, N_e$ **do** ▷ for each particle
  Solve $\left(\mathbf{HQH}^T + \mathbf{R}\right)\mathbf{e}_j = \mathbf{d}_j$ ▷ $(N_y \times 1)$
  $\mathbf{K}_j \leftarrow \mathbf{QH}^T\mathbf{e}_j$ ▷ $(N_x \times 1)$
  $\phi_j = \mathbf{d}_j^T\mathbf{e}_j$ ▷ $(N_x \times 1)$
  $\mathbf{x}_j^a \leftarrow \mathcal{M}\left(\mathbf{x}_j^{m-1}\right) + \mathbf{K}_j$ ▷ $(N_x \times 1)$
  $\gamma_j \leftarrow \boldsymbol{\xi}_j^{\mathsf{T}}\boldsymbol{\xi}_j$ ▷ $(N_x \times 1)$
  $a_j \leftarrow \phi_j - \mathbf{w}_j^{rest} + w_{target}$ ▷ (1×1)
  Solve $(\alpha_j - 1)\gamma_j - N_x \log \alpha_j + a_j = 0$ ▷ (1×1)
  $\mathbf{x}_j^n \leftarrow \mathbf{x}_j^a + \alpha_j\boldsymbol{\xi}_j$ ▷ $N_x \times 1$
end for

Algorithm 18 PFGR and NETF, see Section 7.1

for $j = 1, \ldots, N_e$ **do** ▷ for each particle
  $\mathbf{w}_j \leftarrow \frac{p(\mathbf{y}|\mathbf{x}_j)}{\sum_i p(\mathbf{y}|\mathbf{x}_i)}$ ▷ (1×1)
end for
$\mathbf{W} \leftarrow diag(\mathbf{w}) - \mathbf{ww}^T$ ▷ $(N_e \times N_e)$
$\mathbf{U\Sigma U}^T \leftarrow \mathbf{W}$ ▷ Compute EVD; $(N_e \times N_e)$
$\mathbf{T}' \leftarrow \sqrt{N}\mathbf{U}\boldsymbol{\Sigma}^{1/2}\mathbf{U}^T$ ▷ $(N_e \times N_e)$
Prepare $\boldsymbol{\Lambda}$ ▷ $(N_e \times N_e)$
$\mathbf{T} \leftarrow \mathbf{T}'\boldsymbol{\Lambda} + \mathbf{w}$ ▷ $(N_e \times N_e)$
$\mathbf{X}^a \leftarrow \mathbf{X}^f\mathbf{T}$ ▷ $(N_x \times N_e)$

Algorithm 19 MMPF, see Section 7.2

for $j = 1, \ldots, N_e$ **do** ▷ for each particle
  $w_j \leftarrow \frac{p(\mathbf{y}|\mathbf{x}_j)}{\sum_i p(\mathbf{y}|\mathbf{x}_i)}$ ▷ $(1 \times N_e)$
end for
$\tilde{\mathbf{x}}^a \leftarrow \mathbf{X}^f\mathbf{w}$ ▷ $(N_x \times 1)$
$\mathbf{TT}^T \leftarrow \mathrm{diag}(\mathbf{w}) - \mathbf{ww}^{\mathbf{T}}$ ▷ $(N_e \times N_e)$
$\mathbf{U\Sigma U}^T \leftarrow \mathbf{TT}^T$ ▷ Compute EVD; $(N_e \times N_e)$
$\mathbf{B\Lambda C}^T \leftarrow \mathbf{X}'^f$ ▷ Compute SVD; $(N_x \times N_e)$
$\mathbf{y}_k \sim p_{\mathcal{H}(\mathbf{x}_k)}(\mathbf{y}|\mathbf{x}_k)$ ▷ Sample observations; $(N_y \times N_e)$
for $k = 1, \ldots, N_e$ **do** ▷ Loop over sampled obs.
  for $j = 1, \ldots, N_e$ **do** ▷ for each particle
    $\tilde{w}_j \leftarrow \frac{p(\mathbf{y}_k|\mathbf{x}_j)}{\sum_{i=1}^N p(\mathbf{y}_k|\mathbf{x}_i)}$ ▷ $(1 \times N_e)$
  end for
  $\tilde{\tilde{\mathbf{x}}}^a \leftarrow \mathbf{X}^f\tilde{\mathbf{w}}$ ▷ $(N_x \times 1)$
  $\tilde{\mathbf{T}}\tilde{\mathbf{T}}^T \leftarrow \mathrm{diag}(\tilde{\mathbf{w}}) - \tilde{\mathbf{w}}\tilde{\mathbf{w}}^T$ ▷ $(N_e \times N_e)$
  $\tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{U}}^T \leftarrow \tilde{\mathbf{T}}\tilde{\mathbf{T}}^T$ ▷ Compute EVD; $(N_e \times N_e)$
  $\mathbf{d} \leftarrow \left(\mathbf{x}_k - \tilde{\tilde{\mathbf{x}}}^a\right)$ ▷ $(N_x \times 1)$
  $\mathbf{d}_1 \leftarrow \mathbf{B}^T\mathbf{d}$ ▷ $(1 \times N_e)$
  $\mathbf{d}'_1 \leftarrow \boldsymbol{\Lambda}^{-1}\mathbf{d}_1$ ▷ Diagonal $\boldsymbol{\Lambda}$; $(1 \times N_e)$
  $\mathbf{d}_2 \leftarrow \mathbf{C}\mathbf{d}'_1$ ▷ $(1 \times N_e)$
  $\mathbf{d}'_2 \leftarrow \tilde{\boldsymbol{\Sigma}}^{1/2}\mathbf{d}_2$ ▷ Diagonal $\tilde{\boldsymbol{\Sigma}}$; $(1 \times N_e)$
  $\mathbf{d}_3 \leftarrow \tilde{\mathbf{U}}^T\mathbf{d}'_2$ ▷ $(1 \times N_e)$
  $\mathbf{d}'_3 \leftarrow \boldsymbol{\Sigma}^{1/2}\mathbf{d}_3$ ▷ Diagonal $\boldsymbol{\Sigma}$; $(1 \times N_e)$
  $\mathbf{d}_4 \leftarrow \mathbf{U}\mathbf{d}'_3$ ▷ $(1 \times N_e)$
  $\mathbf{x}_k^a = \tilde{\mathbf{x}}^a + \mathbf{X}'^f\mathbf{d}_4$ ▷ $(N_x \times 1)$
end for

Algorithm 20 Merging Particle Filter, see Section 7.3

for $j = 1, \ldots, N_e$ **do** ▷ for each particle
  $w_j \leftarrow \frac{p(\mathbf{y}|\mathbf{x}_j)}{\sum_i p(\mathbf{y}|\mathbf{x}_i)}$ ▷ $(N_y \times 1)$
end for
$(X_1^a, \ldots X_q^a) \leftarrow q$ times resampled prior ensemble ▷ $(q \times N_e)$
Find α_i such that $\sum_i \alpha_i = 1$ and $\sum_i \alpha_i^2 = 1$ ▷ $(q \times q)$
$X^a \leftarrow \sum \alpha_i X_i^a$ ▷ $(q \times N_e \times N_x)$

Algorithm 21 Local PF Poterjoy, see Section 9

for Each observation **do**
  $\tilde{w_i} \leftarrow \alpha p(\mathbf{y}_1|x_i) + 1 - \alpha$ ▷ $(1 \times N_e)$
  Resample $\mathbf{x}_{k_i}$ ▷ $(N_x \times 1)$
  for Each grid point j **do**
    $\omega_i \leftarrow \alpha\rho(\mathbf{y}_1, x_j, r)\, p(\mathbf{y}_1|x_i) + 1 - \alpha\rho(\mathbf{y}_1, x_j, r)$ ▷ $(1 \times N_e)$
    $\overline{x}_i \leftarrow \sum \omega_i x_{i,j}$ ▷ (1×1)
    $\sigma^2 \leftarrow \sum \omega_i (x_{i,j} - \overline{x}_i)^2$ ▷ $(1 \times N_e)$
    $c_j \leftarrow \frac{N(1-\alpha\rho(x_j, \mathbf{y}_1, r))}{\alpha\rho(x_j, \mathbf{y}_1, r)\tilde{W}}$
    $r_{1j} \leftarrow \sqrt{\frac{\sigma_j^2}{\frac{1}{N-1}\sum_{i=1}^{N}(x_{k_i,j} - \bar{x}_j + c_j(x_{i,j} - \bar{x}_j))^2}}$ ▷ $(1 \times N_e)$
    $r_{2j} \leftarrow c_j r_{1j}$
    $x^a_{i,j} = \bar{x}_j + r_{1j}(x_{k_i,j} - \bar{x}_j) + r_{2j}(x_{i,j} - \bar{x}_j)$ ▷ $(1 \times N_e)$
  end for
end for

Algorithm 22 Weights $w_i = \frac{p(\mathbf{y}|\mathbf{x}_i)}{\sum_j p(\mathbf{y}|\mathbf{x}_j)}$ for Gaussian obs. errors

$\mathbf{D} \leftarrow [\mathbf{y}, \ldots, \mathbf{y}] - \mathcal{H}\left(\mathbf{X}^f\right)$ ▷ $(N_y \times N_e)$
$\mathbf{D}_1 \leftarrow \mathbf{R}^{-1}\mathbf{D}$ ▷ $(N_y \times N_e)$
$\mathbf{w}' \leftarrow \exp(-0.5\, \mathbf{D}^T\mathbf{D}_1)$ ▷ (N_y)
$\mathbf{w} \leftarrow \mathbf{w}'/\sum_j w'_j$ ▷ (N_y)