SANGOMA: Stochastic Assimilation for the Next Generation Ocean Model Applications EU FP7 SPACE-2011-1 project 283580

Deliverable 3.5:
Final Document
Due date: 01/11/2015
Delivery date: 01/11/2015
Delivery type: Report , public

Peter Jan Van Leeuwen   Sanita Vetra-Carvalho
University of Reading, UK

Jean-Marie Beckers   Alexander Barth
University of Liège, BELGIUM

Arnold Heemink   Nils van Velzen
Martin Verlaan   Umer Altaf
Delft University of Technology, NETHERLANDS

Lars Nerger   Paul Kirchgessner
Alfred-Wegener-Institut, GERMANY

Pierre Brasseur Kirchgessner Jean-Michel Brankart
CNRS-LEGI, FRANCE

Pierre de Mey
CNRS-LEGOS, FRANCE

Laurent Bertino
NERSC, NORWAY

Contents

Introduction

MyOcean is the E.U. project dedicated to the implementation of the GMES Marine Core Service for ocean monitoring and forecasting. MyOcean aims at accurately delivering regular and systematic information on the state of global oceans and European regional seas at the required resolution. This information includes hindcasts, nowcasts and forecasts describing the physical state of the ocean and its primary ecosystem. The project also contributes to climate research by providing time series of analysed parameters. A new FP7 project (R&D to enhance future GMES applications in the Marine and Atmosphere areas) by the MyOcean consortium aims at expansion to the MyOcean project is recently initiated. As part of this FP7 project a proposal "Stochastic assimilation for the next generation ocean model application (SANGOMA)" to prepare an assimilation component of the next generation operational system of the GMES marine core service is funded.

Data assimilation (DA) is a group of methods in which the observations of the state of a system are combined with the results from numerical model to produce accurate estimates of all the current (and future) state variables of the system. A data assimilation system consists of three components: a set of observations, a dynamical model, and a data assimilation scheme.

The central concept of the data assimilation is the concept of errors, error estimation and error modelling. The observations have errors arising from various sources: e.g. instrumental noise and the representativeness errors. All dynamical models are imperfect with errors arising from: the approximate physics (or biology or chemistry), that parametrises the interaction of the state variables and the discretisation of continuum dynamics into a numerical model. In its most general form these uncertainties are described by probability density functions, or *pdfs*. Bayes Theorem tells us how to combine the information from model, the *prior*, and observations, the *likelihood* into the so-called posterior pdf. This pdf describes a best estimate of the state and its uncertainty, and is the solution to the data-assimilation problem.

The most well-known application of DA is in weather forecasting problems in which it was applied in real life for the first time in $1950's$ and $1960's$ to improve the weather forecasts. A good description of the development of DA in meteorology can be found in Delay [1991]. The DA has already proved to be useful in other fields of application like tidal models Heemink and Kloosterhuis [1990], oceanography Evensen [1994b], nonlinear shallow-water storm surge models Verlaan and Heemink [1996] and atmospheric chemistry and transport modelling (e.g. Elbern et al. [1997], Segers et al. [2000]). Among all the DA methods, four dimensional variational data assimilation (4DVAR) called as adjoint method is the one of the most effective and powerful approaches. The method has an advantage of directly assimilating all the available observations distributed in time and space into the numerical model while maintaining dynamical and physical consistency with the model Talagrand [1997]. On the other hand since the adjoint of the numerical model needs to be determined, which is usually complicated and time consuming effort for a nonlinear model, the use of 4DVAR is still limited in various fields.

The Kalman filter (KF) Kalman [1960] is a sequential data assimilation algorithm. For linear stochastic systems, it can be shown that the KF is an optimal linear estimator that minimises the variance of the estimation error Simon [2006]. Because of its relative simplicity in implementation, the KF is suitable for many data assimilation problems. However, for high dimensional systems such as weather forecasting models, direct application of the KF is prohibitively expensive as it involves manipulating covariance matrices of the system states. For this reason, different modifications of the KF were proposed to reduce the computational cost. These include various ensemble Kalman filters (EnKF) Anderson [2001], Bishop et al. [2001], Burgers et al. [1998], Evensen [1994a], Evensen and van Leeuwen [1996], Houtekamer and Mitchell [1998], Whitaker and Hamill [2002], the error subspace-based filters Cohn and Todling [1996], Hoteit et al. [2001, 2002], Pham et al. [1998], Verlaan and Heemink [1997], to name but a few. For a detailed description of the above filters, readers are referred to Evensen [2003], Nerger et al. [2005] for reviews of some of the aforementioned filters. Roughly speaking, these modifications exploit the information of a subset in the state space of a dynamical system, while the information of the complement set is considered less influential, and thus ignored. Consequently, the computations of these modified filters are normally conducted on the chosen subsets, instead of the whole state space, so that their computational costs are reduced. For simplicity, we may sometimes abuse the terminology by referring to all the aforementioned filters as the EnKF-based methods (EnKF methods for short).

Both variational and (en)KF methods assume the prior, so the model state, or parameters, to be Gaussian distributed. This assumption is relaxed in particle filters, that try to solve the full nonlinear data-assimilation problem, directly via Bayes Theorem. While particle filters were not considered efficient for high-dimensional systems (Bengtsson et al. [2008]), recent developments allow application of particle filters to these systems too (e.g. [Van Leeuwen, 2010]). Simultaneously methods are developed that are combinations of EnKF and PF (e.g., Van Leeuwen [2009]). Similarly, efforts are made to develop robust filters that emphasise on the robustness of their error estimates, so that they may have better tolerances to possible uncertainties in assimilation. As an example, the H_∞ filter (HF) Francis [1987], Simon [2006]. An Ensemble time-local H_∞ filter (EnTLHF) is proposed recently in Luo and Hoteit [2011] as an analogy to the EnKF for high dimensional data assimilation problems.

The main aim of the SANGOMA project is to accelerate the implementation of flexible DA toolboxes to strengthen the connection between academics and oceanographic community. This will also allow fast implementation and evaluation of the new DA techniques. Present day high resolution ocean models are very nonlinear and require a strong need for data assimilation methods that can handle these non-linearities.

This document gives a detailed description of the DA methods used in SANGOMA that include uncertainty estimation and that can be implemented for large dimensional ocean models. The next section gives a detail description of each individual method. This document is the final report on non-linear data-assimilation methods for high-dimensional ocean models.

Shared Software Modules

This section gives a detailed description of the shared modules. We start by giving common symbol notations. These main symbol notations will be used for all the modules throughout this document.

M	non-linear model operator
$\mathbf{M}$	Tangent linear model
X	model state vector
X_t	true value of the model state vector
X_b	background model state
X_a	analyzed model state
Y	observation vector
H	observation operator
$\mathbf{H}$	linearized observation operator
$\mathbf{B}$	background error covariances
$\mathbf{A}$	analysis error covariances
$\mathbf{R}$	observation error covariances
$\mathbf{K}$	analysis gain Matrix
$\mathbf{I}$	identity Matrix
J	cost function
J_b	background term of the cost function
J_o	observation term of the cost function
J_p	penalty term of the cost function
γ	parameter vector
n	size of state vector
n^q	size of observation vector

2.1 POD Calibration Method

The adjoint method is a well-known approach to inverse modelling. The method aims at adjusting a number of unknown control parameters on the basis of given data. The control parameters might be model initial conditions or model parameters Le Dimet and Talagrand [1986], Thacker and Long [1988]. An objective function is defined which measures the misfit between the solution and the available data for any model solution over the assimilation interval. To obtain a computationally efficient procedure this objective function is minimized with a gradient-based algorithm where the gradient is determined by solving the adjoint problem. The adjoint approach is computationally very efficient because one gradient calculation requires just a single simulation of the forward model and a single simulation of the adjoint model backward in time, irrespective of the number of parameters. The adjoint method has been used and applied successfully to many types of inverse problems in ground water flow studies (e.g. Carrera and Neuman [1986]), in meteorology (e.g. Courtier and Talagrand [1990]), in oceanography (e.g. Tziperman et al. [1992]) and in shallow water flow models (e.g. Ten-Brummelhuis et al. [1993], Lardner et al. [1993], Ulman and Wilson [1998], Heemink et al. [2002]). One of the drawbacks of the adjoint method is the programming effort required for the implementation of the adjoint model. Research has recently been carried out on automatic generation of computer codes for the adjoint, and adjoint compilers have now become available (see Kaminski et al. [2003]). Even with the use of these adjoint compilers developing an adjoint model is often a significant programming effort that hampers new applications of the method.

Proper orthogonal decomposition (POD) is a model reduction method considered as an application of the singular value decomposition (SVD) to the approximation of general dynamical systems Antoulas [2005]. It is a data driven projection based method originally developed by Karl Pearson Pearson [1901]. Karhunen Karhunen [1946] and Loeve Loeve [1946] had used it as statistical tool to analyze random process data. Lumley [1967] gave the name POD, and used the method to study turbulent flow. The POD method has application in many fields like image processing, signal processing, data compression, oceanography, chemical engineering and fluid mechanics (see Gunzburger [2004]). In the POD method the projection subspace is determined by processing data obtained from numerical simulations of high dimensional model which is expected to provide information about the dynamical behaviour of the system. The high dimensional equations are projected onto the low dimensional subspace resulting in a low dimensional model and thus reduces the CPU time of model simulation.

Vermeulen and Heemink Vermeulen and Heemink [2006] proposed a method based on POD which shifts the minimisation into lower dimensional space and avoids the implementation of the adjoint of the tangent linear approximation of the original nonlinear model. In their approach, an ensemble of snapshot vectors of forward model simulations is used to determine an approximation of the covariance matrix of the model variability and a small number of leading eigenvectors of this matrix is used to define a model subspace. By projecting the original model onto this subspace an approximate linear reduced model is obtained. Due to the linear character of the reduced model its adjoint can be implemented easily and the minimisation problem is solved completely in reduced space with very low computational cost. The method has recently been applied successfully to the Dutch continental shelf model to estimate water depth Altaf et al. [2012].

2.1.1 Background

The discrete model for the evaluation of shallow water system from time t_i to time t_{i+1} can be described by an equation of the form

$$X(t_{i+1}) = M_i[X(t_i), \gamma], \tag{2.1}$$

where state vector $X(t_{i+1}) \in \Re^n$ denotes the state vector at time t_{i+1} and γ is the vector of the uncertain parameters which needs to be determined. M_i is nonlinear and deterministic dynamics operator that includes inputs. Suppose now that we have imperfect observations $Y(t_i) \in \Re^{n^q}$ of the dynamical system (2.1) that are related to the model state at time t_i through

$$Y(t_i) = \mathbf{H}X(t_i) + \eta(t_i), \tag{2.2}$$

where $\mathbf{H} : \Re^n \to \Re^{n^q}$ is a linear observation operator that maps the model fields on observation space and $\eta(t_i)$ is an unbiased, random Gaussian error vector with covariance matrix $\mathbf{R}_i$. The aim of data assimilation is to find the best estimate of the system initial parameters γ based on given observations. We assume that the difference between data and simulation results is only due to measurement errors and incorrectly prescribed model parameters. The problem of the estimation is then solved by directly minimising the objective function J

$$J(\gamma) = \sum_i [Y(t_i) - \mathbf{H}(X(t_i))]^T \mathbf{R}_i^{-1} [Y(t_i) - \mathbf{H}(X(t_i))] \tag{2.3}$$

with respect to the parameters γ satisfying the discrete nonlinear forecast model (2.1). The efficient minimisation of the objective function requires the computation of the gradient of the objective function (2.3). The adjoint method computes the exact gradient efficiently. Regardless of the number of parameters, the time required to compute the gradient using adjoint technique is more or less identical and is comparable with the computational time needed for a single simulation run of the nonlinear model (2.1). It requires one forward simulation with the original nonlinear model (2.1) and a second additional simulation backward in time with the adjoint model

$$\nu(t_i) = \mathbf{M}_{i+1}^T \nu(t_{i+1}) - 2\mathbf{H}\mathbf{R}_i^{-1}[Y(t_i) - \mathbf{H}(X(t_i))], \tag{2.4}$$

where $\nu(t_i)$ represents the solution of the adjoint model. The gradient ∇J of the objective function J with respect to each component γ_k of the uncertain parameters vector γ is given by

$$\nabla J_k = \sum_i -[\nu(t_{i+1})]^T [\frac{\partial M_i[X(t_i), \gamma]}{\partial \gamma_k}], k = \{1, \cdots, n^p\}. \tag{2.5}$$

The main hurdle in the use of adjoint method is its implementation, especially when the forward model contains non-linearities.

2.1.2 Linearisation and reduced model formulation

The classical adjoint problem can be simplified with the hypothesis that the objective function J can be made quadratic by assuming that the nonlinear dynamics operator M_i can be linearised. The linearisation of nonlinear high-order model (2.1) using the first order Taylor's formula around the background parameter γ_k^b gives

$$\triangle\overline{X}(t_{i+1}) = \frac{\partial M_i[X_b(t_i),\gamma^b]}{\partial X_b(t_i)}\triangle\overline{X}(t_i) + \sum_k \frac{\partial M_i[X_b(t_i),\gamma^b]}{\partial \gamma_k}\Delta\gamma_k \tag{2.6}$$

where $\overline{X}$ is linearised state vector, X_b is the background state vector with the prior estimated parameters vector γ^b and $\triangle\overline{X}$ is a deviation of the model from background trajectory.

A model can be reduced if the incremental state $\triangle\overline{X}(t_{i+1})$ can be written as linear combination

$$\triangle\overline{X}(t_i) = P\xi(t_{i+1}), \tag{2.7}$$

where $P = [p_1, p_2, \cdots, p_r]$ is a projection matrix such that $P^TP = I_r$ and ξ is a reduced state vector given by

$$\xi(t_{i+1}) = \widetilde{\mathbf{M}}_i\xi(t_i) + \sum_k \frac{\partial\widetilde{M}_i}{\partial\gamma_k}\Delta\gamma_k \tag{2.8}$$

or in matrix form

$$\begin{pmatrix} \xi(t_{i+1}) \\ \Delta\gamma \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{M}}_i & \widetilde{\mathbf{M}}_i^\gamma \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \xi(t_i) \\ \Delta\gamma \end{pmatrix} \tag{2.9}$$

Here $\Delta\gamma$ is the control parameter vector, $\widetilde{\mathbf{M}}_i$ and $\widetilde{\mathbf{M}}_i^\gamma$ are simplified dynamics operators which approximate the full Jacobians $\mathbf{M}_i$ and $\frac{\partial M_i}{\partial\gamma_k}$ respectively:

$$\widetilde{\mathbf{M}}_i = P^T\mathbf{M}_iP, \tag{2.10}$$

$$\widetilde{\mathbf{M}}_i^\gamma = P^T(\frac{\partial M_i}{\partial\gamma_1}, \cdots, \frac{\partial M_i}{\partial\gamma_{n^p}}), \tag{2.11}$$

The Jacobian $\mathbf{M}_i$, is obtained by approximating the nonlinear dynamics operator M_i by linearising it with respect to background state X^b. Instead of computing this huge Jacobian by approximating the partial differential with finite difference by perturbing the nonlinear operator M_i in the direction of each node, we perturb along the direction of p_h only:

$$\mathbf{M}_ip_h = \frac{M_i[X_b(t_i) + \varepsilon p_h, \gamma^b] - M_i[X_b(t_i),\gamma^b]}{\varepsilon}, h = \{1, \cdots, r\}, \tag{2.12}$$

with ε being the size of the perturbation. The reduced dynamics operator $\widetilde{M}_i$ can now be computed by premultiplying the above formulae by P^T:

$$\widetilde{\mathbf{M}}_i = P^T(\frac{\partial M_i}{\partial X_b(t_i)}p_1, \cdots, \frac{\partial M_i}{\partial X_b(t_i)}p_r). \tag{2.13}$$

Notice also that only the original model simulations are needed here.The reduced model requires less computational time as it simulates a reduced state within the dimension r instead of the original dimension n where $r < n$. The dimension on which the reduced model operates is $(r+n^p)\times(r+n^p)$ with n^p being the number of estimated parameters.

2.1.3 Collection of the snapshots and POD basis

The POD method is used here to obtain an approximate low-order formulation of the original tangent linear model. POD is an optimal technique of finding a basis which spans an ensemble of data (snapshots) collected from an experiment or a numerical simulation of a dynamical system. The POD modes are optimal at approximating a given dataset. Since the reduced model is used here to estimate uncertain parameters (depth D and manning coefficient c_m), the snapshots should be able to represent the behaviour of the system for these parameters. Therefore the snapshot vectors $e_i \in \Re^s$ are obtained from the perturbations $\frac{\partial M_i}{\partial \gamma_k}$ along each estimated parameter γ_k to get a matrix

$$E = \{e_1, \cdots, e_s\}; i = \{1, 2, \cdots, s\}. \tag{2.14}$$

The dimension of this ensemble matrix E is $s = n^p \times n^s$, where n^s is the number of snapshot collected for each individual parameter γ_k. The covariance matrix Q can be constructed from the ensemble E of the snapshots by taking the outer product

$$Q = EE^T \tag{2.15}$$

The projection matrix P used in the previous section is based on the dominant eigenvectors (POD modes) of this covariance matrix. This covariance matrix is usually huge as in the current application with state vector of dimension $\sim 3 \times 10^6$, so direct solution of eigenvalue problem is not feasible. To shorten the calculation time necessary for solving the eigenvalue problem for this high dimensional covariance matrix, we define a covariance matrix G as an inner product

$$G = E^t E \tag{2.16}$$

In the method of snapshots (Sirovich [1987]), one then solves the $s \times s$ eigenvalue problem

$$G\mathbf{z}_i = E^t E\mathbf{z}_i = \lambda_i \mathbf{z}_i, i \in \{1, 2, \cdots, s\} \tag{2.17}$$

where λ_i are the eigenvalues of the above eigenvalue problem. The eigenvectors $\mathbf{z}_i$ may be chosen to be orthonormal and the POD modes P are then given by:

$$\mathbf{p}_i = E\mathbf{z}_i / \sqrt{\lambda_i} \tag{2.18}$$

A physical explanation of POD modes is that they maximise the average energy in the projection of data onto subspace spanned by the modes. We define a measure ψ_i for the relative information to choose a low dimensional basis by neglecting modes corresponding to the small eigenvalues:

$$\psi_i = \frac{\lambda_i}{\sum_{l=1}^{s} \lambda_l} 100\%, i = \{1, 2, \cdots, s\} \tag{2.19}$$

We collect p_r $(r < s)$ modes such that $\psi_1 > \psi_2 > \ldots > \psi_r$ and they totally explain at least the required variance ψ^e,

$$\psi^e = \sum_{l=1}^{r} \psi_l \tag{2.20}$$

The total number of eigenmodes r in the POD basis P depends on the required accuracy of the reduced model.

2.1.4 Approximate objective function and its adjoint

In reduced model approach, we look for an optimal solution of the (2.1) to minimise the approximate objective function $\hat{J}$ in an incremental way:

$$\hat{J}(\Delta\gamma) = \sum_i [\{Y(t_i) - \mathbf{H}(X_b(t_i))\} - \hat{\mathbf{H}}\xi(t_i, \Delta\gamma)]^T \mathbf{R}_i^{-1} [\{Y(t_i) - \mathbf{H}(X_b(t_i))\} - \hat{\mathbf{H}}\xi(t_i, \Delta\gamma)]. \tag{2.21}$$

The value of the approximate objective function $\hat{J}$ is obtained by correcting the observations $Y(t_i)$ for background state $X_b(t_i)$ which is mapped on the observational space through a mapping $\mathbf{H}$ and for the reduced model state $\xi(t_i, \Delta\gamma)$ which is mapped to the observational space through mapping $\hat{\mathbf{H}}$ with $\hat{\mathbf{H}} = \mathbf{H}P$.

Since the reduced model has linear characteristics, it is easy to build an approximate adjoint model for the computation of gradient of the approximate objective function (2.21). The gradient of $\hat{J}$ with respect to $\Delta\gamma$ is given by

$$\frac{\partial \hat{J}}{\partial(\Delta\gamma)} = \sum_i -[\hat{\nu}(t_{i+1})]^T \frac{\partial \xi(t_{i+1})}{\partial(\Delta\gamma)}, \tag{2.22}$$

where $\hat{\nu}(t_{i+1})$ is the reduced adjoint state variable (see Appendix A). Once the gradient has been computed, the process of minimizing the approximate objective function $\hat{J}$ is done along the direction of the gradient vector in the reduced space.

After the minimization process the initial parameters γ are updated and new set of updated parameters γ^{up} is obtained:

$$\gamma^{up} = \gamma + \Delta\gamma. \tag{2.23}$$

This process of minimization is repeated several times by constructing new POD model with new set of updated parameters γ^{up} to get optimal parameters.

2.1.5 Workflow with POD algorithm

In order to perform the whole parameter estimation process, the following steps are executed.

1. Outer Iteration:
   (a) Generate an ensemble of forward model simulations using initial parameters γ^b.
   (b) Solve eigenvalue problem to get dominant eigenmodes p_i
   (c) Establish a POD reduced model and its adjoint model using eigenmodes p_i.
2. Inner Iteration:
   (a) Perform optimisation iterations in the reduced space to obtain the optimal solution of the approximate objective function $\hat{J}$.
   (b) Update the initial parameters γ^b after the minimisation process obtain new set of updated parameters γ^{up}.
3. Return to step 1 with new set of updated parameters γ^{up} until optimality condition is achieved.

2.1.6 Convergence criterion for inner and outer iterations

The minimisation is performed using a quasi-Newton optimisation algorithm where the Hessian of the objective function is updated using the limited Broyden-Fletcher-Goldfarb-Shanno (LBFGS) method. The minimisation algorithm requires convergence criteria to terminate. We have defined two criterions, one is for inner iterations and one is for outer iterations of the optimisation process. We stop the present inner iteration α and switch to a new outer iteration β with updated parameters γ^{up} by criterion μ, which is defined as

$$\mu = \frac{\mid \hat{J}_{\alpha_{i+1}} - \hat{J}_{\alpha_i} \mid}{max\{\mid \hat{J}_{\alpha_{i+1}} \mid, 1\}} < \epsilon, \tag{2.24}$$

where α_i represents the i^{th} inner iteration. The value of the ϵ is chosen such that the approximate objective function $\hat{J}$ stops to change, i.e. $\epsilon = 10^{-4}$ (see Oliver et al. [2008]). The outer iteration cycle is aborted when the terminal value of ρ is obtained

$$\rho = \frac{\mid J_{\beta_i} - J_{\beta_{i-1}} \mid}{\mid J_{\beta_i} \mid} \leq \kappa, \tag{2.25}$$

where β_i stands for the i^{th} outer iteration, κ is the terminal value.

2.1.7 Computational efficiency of the algorithm

The computational efficiency of the model-reduced approach is influenced by three factors.

1. Ensemble generation: The computational costs of the reduced model approach are dominated by the generation of the ensemble of forward model simulations. If the dynamics of the system does not change significantly during the course of simulation then a smaller simulation period can be chosen for the generation of ensemble Altaf et al. [2009]. Using this ensemble the optimisation problem can then be solved over the whole period of model simulation.

   To achieve convergence, the POD method needs to be updated in each outer iteration β, so the ensemble E of snapshot vectors is required in each β. Instead of defining a new model subspace of the leading eigenvectors in each β by generating a new ensemble of the forward model simulations, it is possible to obtain the reduced model by projecting the original model with updated parameters onto the same subspace.

2. Ensemble size: The efficiency of optimisation process is also influenced by the ensemble size. A large ensemble size leads to a huge eigenvalue problem. On the other hand, since the ensemble gives the representation of the model behaviour with respect to each γ_k, it is important that the number of snapshot vectors included in the ensemble must give this representation. So the quality of ensemble is crucial for a reduced-order procedure to be effective. It is possible to include only those snapshots in the ensemble for the period where data is available.

3. Outer iteration: The convergence criterion ρ should be carefully chosen. It should not be chosen too small as this causes jumping of the updated parameters γ^{up} around the optimal global solution Vermeulen and Heemink [2006].

2.2 Efficient Particle Filters

Particle filters are ensemble data-assimilation methods that explore Bayes Theorem directly at observation times, without assumptions of linearity and/or Gaussianity. An ensemble member is called a particle. In short, at observation times each particle is given a weight dependent on its distance to all observations present. This results in a weighted ensemble. Then the ensemble is resampled such that high-weight particles are duplicated and small-weight particles are abandoned such that the total number of particles is unchanged, and each of the particles has the same weight. Several resampling schemes are available in the literature (see e.g. Doucet et al. [2001], Van Leeuwen [2009]).

It is well known that when the observation space is of order 10 or larger these particle filters need a prohibitively large number of particles to avoid collapse of the whole ensemble on the one particle with the highest weight. This is because the weights tend to vary widely, related to the relatively small part of state space where the observations are. Bengtsson et al. [2008] provide a firm proof that particle filters of this kind will not work in the high-dimensional spaces we are interested in.

However, importance sampling can be explored, in which the model equations are slightly altered such that the particles end up closer to the observations at observation times. This allows the weights of the particles to be closer to each other, avoiding the collapse mentioned above. One has to take into account the fact that a different model is used to get there, resulting in an extra weight on each particle. The more a particle deviates from the original model, the smaller this weight. The performance of the particle filters can be drastically improved in this way, and there is even an optimal, called the optimal proposal density particle filter. However, even this method does not work in our high-dimensional systems, as shown convincingly by [Snyder et al., 2015].

It is possible, however, to construct particle filters that do not have this limitation to low-dimensional systems. They are based on the notion that one can set a target weights and move all particles such in state space that they obtain a weight that is equivalent to, or exactly equal to the target weight. This is possible because the two weights related to observations and to original-model deviations are competing: the closer the model is pulled towards the observations the higher the observational weight, but the larger the deviation from the original model, so the lower that part of the weight, and vice-versa. Examples of these particle filters are the Equivalent-Weights Particle Filter and the Implicit Equal-Weight Particle Filter. These methods have now been explored in very high-dimensional applications, like climate models and the NEMO ocean circulation model.

2.2.1 Description

For a proper derivation of the equations see Ades and van Leeuwen [2013], and Zhu et al. [2015].

2.2.2 The basics

Particle filters, like ensemble Kalman filters, are variants of Monte Carlo methods in which the probability distribution of the model state given some observations is approximated by a number of particles; however, unlike ensemble Kalman filters, particle filters are fully non-linear data assimilation techniques. While particle filters are not a new concept, until very recently they have been deemed to be computationally unfeasible for large-dimensional systems due to the filter degeneracy problem [Snyder et al., 2008, Van Leeuwen, 2009]. However, recently there have been new developments in the field and particle filter variants have emerged that have been shown to work for large dimensional systems with a limited number of particles. These methods exploit the future observational information by relaxing particles towards the future observations. In this document we will consider two such variants of the particle filters: the equivalent weights particle filter [EWPF, Van Leeuwen, 2010, 2011, Ades and van Leeuwen, 2015] and the auxiliary particle filter [Pitt and Shephard, 1999]. 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, 2009], is not discussed here as it needs a 4D-Var minimisation in each particle.

The probability distribution function (pdf) in particle filtering, represented by N particles or ensemble members at time k, is given by

$$p\left(\mathbf{x}^{(k)}\right) = \frac{1}{N}\sum_{j=1}^{N}\delta\left(\mathbf{x}^{(k)} - \mathbf{x}_j^{(k)}\right), \tag{2.26}$$

where $\mathbf{x}^{(k)} \in \mathcal{R}^n$ is the n-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 k to be time of a current set of observations with the previous observation set at time 0. Then the stochastic forward model for times $0 < m < k$ for each particle $j = 1, ..., N$ is given by

$$\mathbf{x}_j^{(m)} = \mathcal{M}_m\left(\mathbf{x}_j^{(m-1)}\right) + \boldsymbol{\epsilon}_j^{(m)}, \tag{2.27}$$

where $\boldsymbol{\epsilon}_j^{(m)} \in \mathcal{R}^n$ are random terms representing the model error distributed according to a given covariance matrix **Q** and $\mathcal{M}_m : \mathcal{R}^n \to \mathcal{R}^n$ is the deterministic model from time $m-1$ to m. Thus, the model state transition from time $m-1$ to m is fully described by the transition density given by

$$p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}\right) = \mathcal{N}\left(\mathcal{M}_m\left(\mathbf{x}^{(m-1)}\right), \mathbf{Q}\right). \tag{2.28}$$

Using Bayes theorem and the Markovian property of the model, the full posterior at observation time k is written as

$$p\left(\mathbf{x}_j^{(k)}|\mathbf{y}^{(k)}\right) = \sum_{j=1}^{N} w_j^{(k)}\delta\left(\mathbf{x}^{(k)} - \mathbf{x}_j^{(k)}\right) \tag{2.29}$$

where $\delta(x)$ is a Dirac-delta function and weights $w_j^{(k)}$ are given by

$$w_j^{(k)} \propto p\left(\mathbf{y}^{(k)}|\mathbf{x}_j^{(k)}\right) p\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}\right) w_j^{(k-1)} \tag{2.30}$$

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

$\mathrm{x}^{(k)}$ which is often taken to be Gaussian

$$p\left(\mathbf{y}^{(k)}|\mathbf{x}^{(k)}\right) \propto \exp\left[-\frac{1}{2}\left(\mathbf{y}^{(k)}-\mathcal{H}_k\left(\mathbf{x}^{(k)}\right)\right)^{\mathsf{T}} \mathbf{R}^{-1}\left(\mathbf{y}^{(k)}-\mathcal{H}_k\left(\mathbf{x}^{(k)}\right)\right)\right]. \tag{2.31}$$

However, 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. Next we discuss two particular particle filters which have been shown to work in high-dimensional systems with a very limited number of particles, namely the Equivalent-Weights Particle Filter, and the Implicit Equal-Weight Particle Filter

2.2.3 Making Particle Filters efficient

We aim to ensure 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 need a scheme that pulls the particles towards the observations. Several methods can be used for this, including traditional methods like 4Dvar 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.

For a Markovian system with observational errors that are independent from one time to another, the posterior pdf can be written as

$$p(\mathbf{x}^n|\mathbf{y}^{1:n}) = \frac{p(\mathbf{y}^n|\mathbf{x}^n)}{p(\mathbf{y}^n)}\int p(\mathbf{x}^n|\mathbf{x}^{n-1})p(\mathbf{x}^{n-1}|\mathbf{y}^{1:n-1})d\mathbf{x}^{n-1} \tag{2.32}$$

It is assumed that the particle weights in the ensemble at previous time-step $n-1$ are equal:

$$p(\mathbf{x}^{n-1}|\mathbf{y}^{1:n-1}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}^{n-1}-\mathbf{x}_i^{n-1}) \tag{2.33}$$

As a consequence of plugging equation (2.33) into equation (2.32), it is clear that

$$p(\mathbf{x}^n|\mathbf{y}^{1:n}) = \frac{1}{N}\sum_{i=1}^{N}\frac{p(\mathbf{y}^n|\mathbf{x}^n)p(\mathbf{x}^n|\mathbf{x}_i^{n-1})}{p(\mathbf{y}^n)} \tag{2.34}$$

One can now multiply the numerator and denominator of equation (2.34) by the same factor $q(\mathbf{x}^n|\mathrm{x}^{\mathbf{n-1}},\mathbf{y}^n)$, in which $\mathrm{x}^{\mathbf{n-1}}_{\mathbf{1:N}}$ is defined as the collection of all particles at time $n-1$.

$$p(\mathbf{x}^n|\mathbf{y}^{1:n}) = \frac{1}{N}\sum_{i=1}^{N}\frac{p(\mathbf{y}^n|\mathbf{x}^n)}{p(\mathbf{y}^n)}\frac{p(\mathbf{x}^n|\mathbf{x}_i^{n-1})}{q(\mathbf{x}^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}},\mathbf{y}^n)}q(\mathbf{x}^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}},\mathbf{y}^n) \tag{2.35}$$

where the support of $q(\mathbf{x}^n|\mathrm{x}^{\mathbf{n-1}}_{\mathbf{1:N}},\mathbf{y}^n)$ should be equal to or larger than that of $p(\mathbf{x}^n|\mathbf{x}^{n-1})$. $q(\mathbf{x}^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}},\mathbf{y}^n)$ is the so-called *proposal transition density*.

Observations every time-step is the simplest setup to analyse the effect of drawing samples from the proposal transition density $q(\mathbf{x}^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n)$, instead of the original transition density $p(\mathbf{x}^n|\mathbf{x}_i^{n-1})$. This leads the posterior pdf to be epressed as:

$$p(\mathbf{x}^n|\mathbf{y}^{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \frac{p(\mathbf{y}^n|\mathbf{x}_i^n)}{p(\mathbf{y}^n)} \frac{p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1})}{q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n)} \delta(\mathbf{x}^n - \mathbf{x}_i^n) \tag{2.36}$$

Consequently the posterior pdf of model state at time-step n can be written as

$$p(\mathbf{x}^n|\mathbf{y}^{1:n}) = \frac{1}{N}\sum_{i=1}^{N} w_i \delta(\mathbf{x}^n - \mathbf{x}_i^n) \tag{2.37}$$

where w_i is the particle weights given by

$$w_i = \frac{p(\mathbf{y}^n|\mathbf{x}_i^n)}{p(\mathbf{y}^n)} \frac{p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1})}{q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n)} \tag{2.38}$$

Now assuming that the model system is Markovian and using Bayes' theorem, the numerator in the expression for the weights can be expressed as

$$p(\mathbf{y}^n|\mathbf{x}^n)p(\mathbf{x}^n|\mathbf{x}_i^{n-1}) = p(\mathbf{x}^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n)p(\mathbf{y}^n|\mathbf{x}_i^{n-1}) \tag{2.39}$$

Therefore the particle weights of ensemble i at observed time-step becomes

$$w_i = \frac{p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n)p(\mathbf{y}^n|\mathbf{x}_i^{n-1})}{p(\mathbf{y}^n)q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n)} \tag{2.40}$$

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

$$q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n) = p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n)$$

, leading to weights $w_i \propto p(\mathbf{y}^n|\mathbf{x}_i^{n-1})$. For systems with a large number of independent observations these weights are degenerate, see e.g. [Ades and van Leeuwen, 2013].

Two efficient particle filter schemes have been developed using this idea. The first one, the Equivalent-Weights Particle Filter (EWPF), has been implemented in the majority of the software systems developed in SANGOMA. The second one has recently been developed and turns out to be much more robust than the EWPF. It is called the Implicit Equal-Weights Particle Filter (IEWPF). Both are described below.

The Equivalent-Weights Particle Filter

The EWPF works as follows

1) Determine the maximum weight each particle can reach using a deterministic step.

2) Choose a target weight based on these maximal weights that a certain percentage of particles can reach. For instance, if the target weight is set to the lowest of the maximal weights we keep 100% of the particles. A choice of 50% will mean that the target weight is set to the medium value of the maximal weights.

3) Calculate the position in state space of each particle such that it has a weight exactly equal to the target weight.

4) Add a small random perturbation to each particle and recalculate its weight.

5) Resample the particles such that their weights are equal again.

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 I, or actually, the minimum of minus the $\log$ of that weight:

$$-\log(w_i) \propto \phi_i = (\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1}))^T(HQH^T + R)^{-1}(\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1})) \quad (2.41)$$

Then a target weights is set from these ϕ_i's. The target weights splits the ensemble of particles in two: those that have a higher maximal weight, and those with a lower maximal weight. The latter are abandoned at this point, and will come back in the resampling step 5).

For the others, 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:

$$\hat{\mathbf{x}_i^n} = \mathcal{M}(\mathbf{x}_i^{n-1}) + \alpha_i(Q^{-1} + H^TR^{-1}H)^{-1}H^TR^{-1}(\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1})) \quad (2.42)$$

in which α_i is a scalar. Doing this the number of solutions is reduced to two, so two values for α_i. Note that $alpha_i = 1$ pushes the particle to its maximum weight position. Also note the resemblance of this solution to that of the Kalman Filter, replacing Q with the ensemble covariance.

In the workflow below the solutions for α_i are presented, as is the shape of the random forcing.

The Implicit Equal-Weights Particle Filter

This scheme is very similar to that of the EWPF:

1) Determine the maximum weight each particle can reach using a deterministic step.

2) Choose a target weight based on these maximal 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 ensures maximal weight.

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

$$q(\mathbf{x}^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n) = \frac{q(\xi)}{||\frac{dx}{d\xi}||} \tag{2.43}$$

where $||\frac{dx}{d\xi}||$ denotes the absolute value of the determinant of the Jacobian matrix of the $\mathcal{R}^{N_x} \rightarrow \mathcal{R}^{N_x}$ transformation $\mathbf{x}_i = g(\xi_i)$. To find a solution of this underdetermined situation, the implicit relation between variable $\mathbf{x}_i^n$ and ξ is defined as

$$\mathbf{x}_i^n = \mathbf{x}_i^a + \alpha_i^{1/2} P^{1/2} \xi_i^n \tag{2.44}$$

with $\mathbf{x}_i^a$ the mode of $q(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n)$, P a measure of the width of that pdf, and α_i a scalar. In the implicit particle filter of Chorin et al. [2010] α_i is determined by choosing the proposal density as the optimal proposal density, so again

$$q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n) = p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n),$$

and using the expression for $\mathbf{x}_i^n$ directly in

$$p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n) = \frac{q(\xi)}{||\frac{dx}{d\xi}||} \tag{2.45}$$

leading to a nonlinear scalar equation for α_i. Therefore, when observations are present every time step the implicit particle filter is the optimal proposal particle filter with a smart sampling scheme.

Our scheme is different in that we choose the α_i such that all particles get the same weight w_{target}, so we determine the scalar α_i for each particle from:

$$w_i = \frac{p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n) p(\mathbf{y}^n|\mathbf{x}_i^{n-1})}{N p(\mathbf{y}^n) q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n)} = w_{target} \tag{2.46}$$

in which $x^{n-1}_{1:N}$ denotes the ensemble of particles at time $n-1$. So we allow the proposal density to depend on all previous particles, and not just on one of them. This ensures that the filter is not degenerate in systems with arbitrary dimensions and an arbitrary number of independent observations.

We can expand this as follows. Sampling implicitly from $q(\xi)$ instead of

$$q(\mathbf{x}_i^n|\mathbf{x}^{\mathbf{n-1}}_{\mathbf{1:N}}, \mathbf{y}^n),$$

the particle weights are now given by

$$w_i = \frac{p(\mathbf{x}_i^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n) p(\mathbf{y}^n|\mathbf{x}_i^{n-1})}{q(\xi)} \left|\left| \frac{dx}{d\xi} \right|\right| \cdot w_i^{prev} \tag{2.47}$$

where $q(\xi)$ is the standard Gaussian distribution and w_i^{prev} introduces the weight from previous time-steps. This equation is the basis of our new scheme.

Assuming now that observation errors and model errors are Gaussian, and the observation operator $H \in \mathcal{R}^{N_y \times N_x}$ is assumed to be linear, we find:

$$\begin{aligned}
&p(\mathbf{y}^n|\mathbf{x}^n)p(\mathbf{x}^n|\mathbf{x}_i^{n-1}) \\
&= \frac{1}{A}\exp\Big[-\frac{1}{2}(\mathbf{y}^n - H\mathbf{x}^n)^T R^{-1}(\mathbf{y}^n - H\mathbf{x}^n) \\
&- \frac{1}{2}(\mathbf{x}^n - f(\mathbf{x}_i^{n-1}))^T Q^{-1}(\mathbf{x}^n - f(\mathbf{x}_i^{n-1}))\Big] \\
&= \frac{1}{A}\exp\left(-\frac{1}{2}(\mathbf{x}^n - \hat{\mathbf{x}_i^n})^T P^{-1}(\mathbf{x}^n - \hat{\mathbf{x}_i^n})\right)\exp(-\frac{1}{2}\phi_i) \\
&= p(\mathbf{x}^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n)p(\mathbf{y}^n|\mathbf{x}_i^{n-1}) \qquad (2.48)
\end{aligned}$$

where

$$P = (Q^{-1} + H^T R^{-1} H)^{-1} \qquad (2.49)$$

$$\hat{\mathbf{x}_i^n} = \mathcal{M}(\mathbf{x}_i^{n-1}) + (Q^{-1} + H^T R^{-1} H)^{-1} H^T R^{-1}(\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1})) \qquad (2.50)$$

$$\phi_i = (\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1}))^T (HQH^T + R)^{-1}(\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1})) \qquad (2.51)$$

$\mathbf{x}_i^a$ in equation (2.75) is the mode of $p(\mathbf{x}^n|\mathbf{x}_i^{n-1}, \mathbf{y}^n)$, given by

$$\mathbf{x}_i^a = \hat{\mathbf{x}_i^n} = \mathcal{M}(\mathbf{x}_i^{n-1}) + QH^T(HQH^T + R)^{-1}(\mathbf{y}^n - H\mathcal{M}(\mathbf{x}_i^{n-1})) \qquad (2.52)$$

This leads to a complicated nonlinear differential equation for α_i that involves the determinant of 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:

$$(\alpha_i - 1)\gamma_i - N_x \log(\alpha_i) + \phi_i - \log w_i^{prev} - C = 0. \qquad (2.53)$$

in which $C = -\log w_{target}$. This equation could be approximated by using numerical methods, such as Newton method, etc., but interestingly analytical solutions based on the so-called Lambert W function do exist. We do not elaborate on these here.

Between observations: Relaxation steps

If the system is not observed every time step the schemes mentioned above can be used over the time window between observations. No analytical solutions can be obtained in that case, and the solution has to be found by iterations. However, this procedure is rather expensive as it typically involves solving a problem similar to a 4DVar on each particle. Interestingly, 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.

However, in general this is a rather expensive procedure, and typically simpler methods are employed between observations. Although we can ensure that Bayes theorem is fulfilled exactly for each particle, the schemes will be less efficient. In the following we demonstrate the use of relaxation between observation times. We use the future observations to guide or nudge the particles at time

m towards observations at next time $k > m$ by using a modified forward model instead of (2.27)

$$\begin{aligned} \mathbf{x}_j^{(m)} &= \mathcal{M}_m\left(\mathbf{x}_j^{(m-1)}\right) + \tilde{\boldsymbol{\epsilon}}_j^{(m)} + \\ &\quad + \boldsymbol{\Upsilon}\left[\mathbf{y}^{(k)} - \mathcal{H}_k\left(\mathbf{x}_j^{(m-1)}\right)\right], \end{aligned} \tag{2.54}$$

where $\tilde{\boldsymbol{\epsilon}}_j^{(m)} \in \mathcal{R}^n$ are random terms representing the model error distributed according to a given covariance matrix $\tilde{\mathbf{Q}}$[1], $\mathcal{M}_m$ is the same deterministic model as in equation (2.27), $\boldsymbol{\Upsilon}$ is a relaxation matrix which we will chose later, $\mathbf{y}^{(k)} \in \mathcal{R}^{p_k}$ is the vector of p_k observations at time k and $\mathcal{H}_k : \mathcal{R}^n \to \mathcal{R}^{p_k}$ is the observation operator mapping model space in to observation space. Note that the observations $\mathbf{y}^{(k)}$ are at later time $k > m$. Then the modified transition density is given by

$$\begin{aligned} q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{(k)}\right) &= \mathcal{N}\left(\mathcal{M}_m\left(\mathbf{x}^{(m-1)}\right) + \right. \\ &\quad \left. + \boldsymbol{\Upsilon}\left[\mathbf{y}^{(k)} - \mathcal{H}_k\left(\mathbf{x}^{(m-1)}\right)\right], \mathbf{Q}\right), \end{aligned} \tag{2.55}$$

and modified weights $w_j^{(k)}$ are given by

$$w_j^{(k)} \propto p\left(\mathbf{y}^{(k)}|\mathbf{x}_j^{(k)}\right) \frac{p\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}\right)}{q\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}, \mathbf{y}^{(k)}\right)} w_j^{(k-1)} \tag{2.56}$$

and $w_j^{(k-1)}$ are the product of all the weights from all time steps up to time $k-1$, i.e. fractions $\frac{p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}\right)}{q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{(k)}\right)}$ for each time $0 < m < k$. The conditional pdf $p\left(\mathbf{y}^{(k)}|\mathbf{x}^{(k)}\right)$ remains the same as in Eqn. (2.31).

This simple modification of the forward model to include information about future observations along with the relaxation matrix $\boldsymbol{\Upsilon}$ limits the search space of the particles to high probability space thus making it possible to use EWPF method for high-dimensional systems with only a very limited number of particles, i.e. where number of particles are much less than the size of the state space $(N \ll n)$.

2.2.4 Workflow

In this scheme we perform the following steps:

1. Before the observation time k for each time step $0 < m < k$ and for each particle $j = 1, ..., N$:

   (a) Advect the model state in time using Eqn. (2.54) with

$$\boldsymbol{\Upsilon}^{(m)} = \rho^{(m)}\mathbf{Q}\left(\mathbf{H}^{(k)}\right)^{\mathsf{T}}\left[\mathbf{H}^{(k)}\mathbf{Q}\left(\mathbf{H}^{(k)}\right)^{\mathsf{T}} + \mathbf{R}^{(k)}\right]^{-1} \tag{2.57}$$

[1]The model error covariance matrices are usually assumed to be equal, i.e. $\tilde{\mathbf{Q}} = \mathbf{Q}$.

with $\mathbf{Q}$ being the covariance matrix of model errors, $\mathbf{R}$ the covariance matrix of observation errors, and $\mathbf{H}^{(k)}$ the linearised version of the observation operator $\mathcal{H}_k$.

The scalar $\rho^{(m)}$ determines the strength of the nudging at each model time step and it is crucial to get the strength of the nudging right for each forward model. For example, for strongly non-linear models nudging should only be significant if current time step m is very close to the next observation time k and negligible otherwise. This is because in non-linear models nudging particles towards a future observation that are far enough in time would result in particles entering wrong or unrealistic attractors for time m and most likely resulting in filter divergence. However, for forward models that are linear or close to linear (also if time between observations is small enough for a non-linear model) nudging term $\rho^{(m)}$ could be increased linearly from the time of the previous observation to next one.

(b) Compute weights

$$w_j^{(m)} = w_j^{(m-1)} \frac{p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}\right)}{q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{(k)}\right)}, \tag{2.58}$$

where both transition densities are assumed to be Gaussian and are calculated according to

$$\begin{aligned} p\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}\right) &= \exp\left[-\frac{1}{2}\left(\boldsymbol{\Upsilon}\left[\mathbf{y}^{(k)} - \mathbf{H}^{(k)}\mathbf{x}_j^{(m)}\right] + \tilde{\boldsymbol{\epsilon}}_j^{(m)}\right)^{\mathsf{T}} \mathbf{Q}^{-1}\right. \\ &\qquad \left.\left(\boldsymbol{\Upsilon}\left[\mathbf{y}^{(k)} - \mathbf{H}^{(k)}\mathbf{x}_j^{(m)}\right] + \tilde{\boldsymbol{\epsilon}}_j^{(m)}\right)\right] && (2.59) \\ q\left(\mathbf{x}_j^{(m)}|\mathbf{x}_j^{(m-1)}, \mathbf{y}^{k}\right) &= \exp\left[-\frac{1}{2}\left(\tilde{\boldsymbol{\epsilon}}_j^{(m)}\right)^{\mathsf{T}} \mathbf{Q}^{-1}\tilde{\boldsymbol{\epsilon}}_j^{(m)}\right]. && (2.60) \end{aligned}$$

2. At observation time k the EWPF proceeds with:

(a) Calculate the maximum weight value for each particle

$$\begin{aligned} C_j = -\log w_j^{(k-1)} + \frac{1}{2}\left[\mathbf{y}^{(k)} - \mathcal{H}_k\left(\mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right)\right)\right]^{\mathsf{T}} \cdot \\ \left[\mathbf{H}^{(k)}\mathbf{Q}\left(\mathbf{H}^{(k)}\right)^{\mathsf{T}} + \mathbf{R}\right]^{-1}\left[\mathbf{y}^{(k)} - \mathcal{H}_k\left(\mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right)\right)\right]. \end{aligned} \tag{2.61}$$

Then choose a target weight C such that 80% (or any other suitable percentage) of particles can reach this weight, i.e. that 80% of C_j are less than C.

(b) Find the deterministic particle analysis update (for the particles which can reach the target weight C) via

$$\check{\mathbf{x}}_j^{(k)} = \mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right) + \alpha_j \boldsymbol{\Upsilon}\mathbf{d}_j^{(k)}, \tag{2.62}$$

where

$$\alpha_j = 1 - \sqrt{1 - \frac{b_j}{a_j}} \tag{2.63}$$

$$a_j = \frac{1}{2}\left(\mathbf{d}_j^{(k)}\right)^{\mathsf{T}} \mathbf{R}^{-1}\mathbf{H}^{(k)}\boldsymbol{\Upsilon}\mathbf{d}_j^{(k)} \tag{2.64}$$

$$b_j = \frac{1}{2}\left(\mathbf{d}_j^{(k)}\right)^{\mathsf{T}} \mathbf{R}^{-1}\mathbf{d}_j^{(k)} - C - \log w_j^{(k-1)} \tag{2.65}$$

$$\mathbf{d}_j^{(k)} = \mathbf{y}^{(k)} - \mathcal{H}^{(k)}\left(\mathbf{x}_j^{(k)}\right). \tag{2.66}$$

(c) Perturb each particle with random perturbations

$$\mathbf{x}_j^{(k)} = \check{\mathbf{x}}_j^{(k)} + \mathbf{d}\boldsymbol{\epsilon}_j^{(k)} \tag{2.67}$$

where the perturbation $\mathbf{d}\boldsymbol{\epsilon}_j^{(k)}$ is drawn from a mixture of uniform and Gaussian distributions, given by

$$\mathbf{d}\boldsymbol{\epsilon} \sim (1-\epsilon)\mathbf{Q}^{1/2}\mathcal{U}(-\gamma_U\mathbf{I}, +\gamma_U\mathbf{I}) + \epsilon\mathcal{N}(0, \gamma_N^2\mathbf{Q}). \tag{2.68}$$

Choosing $\epsilon = 0.001/N$ ensures that we mainly sample from the uniform distribution, but the possibility to sample from the Gaussian distribution ensures the support of the proposal density is at least equal to the support of the model prior. Other parameters are chosen to be as follows:

$$\gamma_U = 10^{-5} \tag{2.69}$$

$$\gamma_N = \frac{2^{n/2}\epsilon\gamma_U^n}{\pi^{n/2}(1-\epsilon)}. \tag{2.70}$$

(d) Calculate the full weights at time k

$$w_j^{(k)} = w_j^{(k-1)} \frac{p\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}\right) p\left(\mathbf{y}^{(k)}|\mathbf{x}_j^{(k)}\right)}{q\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}, \mathbf{y}^{(k)}\right)}. \tag{2.71}$$

taking the final perturbation into the account using the transition density

$$q\left(\mathbf{x}_j^{(k)}|\mathbf{x}_j^{(k-1)}, \mathbf{y}^{(k)}\right) = (1-\epsilon)\mathbf{Q}^{1/2}\mathcal{U}(-\gamma_U\mathbf{I}, +\gamma_U\mathbf{I}) + +\epsilon\mathcal{N}(0, \gamma_N^2\mathbf{Q}). \tag{2.72}$$

(e) Resample to obtain a full ensemble again, e.g. using universal resampling (see Appendix). After resampling the weights of the resampled particles are set to be equal, i.e. $w_j^{(k)} = 1/N$.

3. For the IEWPF we take the following steps at observation times k:

(a) Calculate the maximum weight value for each particle

$$C_j = -\log w_j^{(k-1)} + \frac{1}{2}\left[\mathbf{y}^{(k)} - \mathcal{H}_k\left(\mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right)\right)\right]^{\mathsf{T}} \cdot \left[\mathbf{H}^{(k)}\mathbf{Q}\left(\mathbf{H}^{(k)}\right)^{\mathsf{T}} + \mathbf{R}\right]^{-1}\left[\mathbf{y}^{(k)} - \mathcal{H}_k\left(\mathcal{M}_k\left(\mathbf{x}_j^{(k-1)}\right)\right)\right]. \quad (2.73)$$

Then choose a target weight C such that a certain percentage of particles can reach this weight. In the IEWPF we typically use 100%.

(b) Draw a random vector ξ_i of size N_x from $N(0, I)$, one for each particle.

(c) For each particle i calculate the scalar α_i from

$$(\alpha_i - 1)\gamma_i - N_x \log(\alpha_i) + \phi_i - \log w_i^{prev} - C = 0. \quad (2.74)$$

in which $C = -\log w_{target}$, and $\gamma_i = \xi_i^T \xi$. There are two solutions to this equation that are equally likely, so we choose one of them with a 50% change, and do this for each particle. The new particles are now given by:

$$\mathbf{x}_i^n = \mathbf{x}_i^a + \alpha_i^{1/2} P^{1/2} \xi_i^n \quad (2.75)$$

2.2.5 Computational Costs

The computational costs are as follows:

1. Propagate the ensemble forward in time to the next observation. It is noted that one has to include model errors, which calls for drawing a random vector of size N_x from a standard Gaussian, and an extra matrix-vector multiplication of size N_x^2. The matrix is $Q^{1/2}$.

2. At each time step calculate the weight related to the relaxation term. All factors are available from the model time step, so we only need to calculate a vector-matrix-vector product of order N_x^2 multiplications. This is negligible compared to the calculation of the model time step itself. Interestingly, we don't need the inverse of matrix Q as the $\boldsymbol{\Upsilon}$ factor contains Q too, and $\tilde{\boldsymbol{\epsilon}}_j^{(m)}$ is proportional to $Q^{1/2}$. The result is that we need $Q^{1/2}$, which we already need for the stochastic term in the model equation.

3. At observation terms calculate the maximal weight for each particle. This consists of a vector-matrix-vector product of order N_y^2 multiplications, and the inverse of matrix $HQH^T + R$. If N_y is large the procedure is to or solve for a in $(HQHT + R)a = b$ and multiply the result by a vector. This is typically of order N_y^2 multiple-add operations.

4. In the EWPF we need to calculate the scalar α_i for each particle, which means a vector-matrix-vector multiplication of order N_y^2. This is than followed by a draw of an N_x dimensional vector from a uniform density, followed by a matrix-vector multiplication of size N_x^2.

5. In the IEWPF we need to draw draw random vectors from $N(0, P)$ in which $P^{-1} = Q^{-1} + H^T R^{-1} H$. This looks complicated, but in fact we can draw random vectors from $N(0, Q)$ and transform them as in the ETKF.

2.3 Nonlinear Ensemble Transform Filter

2.3.1 Description

The nonlinear ensemble transform filter (NETF), which was recently introduced in [Tödter and Ahrens, 2015] was implemented in PDAF and further tested using the medium case benchmark.

While the EnKFs are based on a Gaussian assumption for the prior ensemble, the transformation in the NETF is designed to exactly match the first two moments of Bayes theorem [see Tödter and Ahrens, 2015, for a derivation of the NETF].

The ensemble mean at time t_k is computed as $\overline{\mathbf{x}}_k = \frac{1}{m}\mathbf{X}_k\mathbf{1}$, using the vector $\mathbf{1} = (1, \dots, 1)^T$ of length m. Defining the following matrix,

$$\overline{\mathbf{X}}_k = \left[\overline{\mathbf{x}}_k, \dots, \overline{\mathbf{x}}_k\right] = \frac{1}{m}\mathbf{X}_k\mathbf{1}\mathbf{1}^T, \tag{2.76}$$

the ensemble perturbation matrix $\mathbf{X}'_k$ is given by

$$\mathbf{X}'_k = \mathbf{X}_k - \overline{\mathbf{X}}_k = \mathbf{X}_k\mathbf{S}, \qquad \text{where} \qquad \mathbf{S} = \mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T. \tag{2.77}$$

Here, $\mathbf{I}$ denotes the identity matrix. Hence, the matrix $\mathbf{S}$ subtracts the ensemble mean from $\mathbf{X}_k$.

The NETF transforms the forecast ensemble into an analysis ensemble by applying a weight vector and a transform matrix to the forecast mean and perturbations, respectively. As most particle filters, the NETF uses the likelihood weights that arise from Bayes theorem. For normally distributed observation errors, the weight of each member is given by

$$w^i \propto \mathcal{N}\left(\mathbf{y}; \mathcal{H}\left(\mathbf{x}_k^{f(i)}\right), \mathbf{R}_k\right) \tag{2.78}$$

$$\propto \exp\left[-\frac{1}{2}\left(\mathbf{y} - \mathbf{y}_k^{f(i)}\right)^T \mathbf{R}_k^{-1}\left(\mathbf{y} - \mathbf{y}_k^{f(i)}\right)\right], \tag{2.79}$$

where $\mathbf{y}_k^{f(i)} = \mathcal{H}_k(\mathbf{x}_k^{f(i)})$. The weights are normalized so that they sum up to one. Before the weights are computed, the ensemble perturbations should be inflated by an inflation factor $\gamma > 1$. The weight vector and transform matrix of the NETF are computed from these weights as follows [Tödter and Ahrens, 2015]:

$$\mathbf{w} = (w^1, \dots, w^m)^T \tag{2.80}$$

$$\mathbf{T} = \sqrt{m}\left[\mathbf{Diag}(w) - \mathbf{w}\mathbf{w}^T\right]^{1/2}. \tag{2.81}$$

Here, $\mathbf{Diag}(w)$ is a diagonal matrix that contains the weights w^i on the diagonal. To complete the algorithm, the analysis perturbations are computed as

$$\mathbf{X}_k^{a'} = \mathbf{X}_k^{f'}\mathbf{T}\mathbf{\Lambda}, \tag{2.82}$$

where $\mathbf{\Lambda}$ is an random orthogonal matrix [see Pham, 2001]. Finally, the mean is updated via

$$\overline{\mathbf{X}}_k^a = \overline{\mathbf{X}}_k^f + \mathbf{X}_k^{f'}\mathbf{w}\mathbf{1}^{\mathbf{T}}, \tag{2.83}$$

Thus, the new analysis ensemble is given by

$$\mathbf{X}_k^a = \overline{\mathbf{X}}_k^a + \mathbf{X}_k^{a'}. \tag{2.84}$$

The NETF has to be applied in combination with localization as described in [Tödter and Ahrens, 2015] for high-dimensional systems.

2.3.2 Workflow

The workflow of the NETF is presented here as given in Tödter and Ahrens [2015] paper. This can be used with or without localisation (skip first three steps if localisation not needed) and to distinguish between global quantities from local we use subscript g for global and nothing for local variables. that is, we begin with the prior ensemble $\{\mathbf{x}_g^f\}$ of size m, stored in the $d \times m$ ensemble matrix $\mathbf{X}_g^f$, and the k-dimensional observation $\mathbf{y}_g$ with error covariance $\mathbf{R}_g$.

1. Compute the predicted observations by applying the observation operator to the prior ensemble, $\mathbf{y}_g^{f(i)} = \mathcal{H}\left(\mathbf{x}^{f(i)_g}\right)$, and put the resulting vectors into a $k \times m$ ensemble matrix $\mathbf{Y}^{f(i)}$.

2. Prepare an appropriate orthogonal, mean-preserving random rotation matrix $\mathbf{Lambda}_g$.

3. This step selects the data for the local analysis: extract all rows of $\mathbf{X}^f$ that belong to the current local domain to obtain the local ensemble states, $\mathbf{X}_g^f$. Select the entries of $\mathbf{y}_g$ that are to be considered for the local analysis (e.g. observations within a specified localisation radius) and the corresponding rows and columns of $\mathbf{R}_g$ which forms the local observation vector $\mathbf{y}$ and covariance matrix $\mathbf{R}$. Accordingly, choose the same rows from $\mathbf{Y}_g^f$, forming $\mathbf{Y}^f$, which contains the ensemble's counterparts $\mathbf{y}^{f(i)}$ of the localised observations $\mathbf{y}$.

4. If desired, multiply the entries of $\mathbf{R}^{-1}$ with an appropriate weight function to reduce the influence of more distant observations.

5. Calculate the Bayesian weights of the ensemble states using the observational likelihood density, collect them in the weight vector $\mathbf{w}$, and normalise it:

$$w_i \propto p\left(\mathbf{y}|\mathbf{x}^{f(i)}\right), \qquad \sum w_i = 1.$$

   For example, if Gaussian observation errors are used, evaluate

$$w_i \propto \exp\left[-\frac{1}{2}(\mathbf{y}-\mathbf{y}^{f(i)})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{y}^{f(i)})\right].$$

6. Compute the (local) analysis mean by

$$\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^f + \mathbf{X'}^f\mathbf{w}.$$

7. Form the matrix **A**, with $\mathbf{W} = \mathrm{diag}(\mathbf{w})$, and compute the transform matrix:
$$\mathbf{T} = \mathbf{A}^{1/2} = (\mathbf{W} - \mathbf{w}\mathbf{w}^T)^{1/2}.$$
Use the symmetric square root by applying a singular value decomposition to **A**.

8. Transform the prior ensemble perturbations into analysis perturbations by applying the transform matrix and random rotation:
$$\mathbf{X'}^a = \sqrt{m}\mathbf{X'}^f\mathbf{W}\mathbf{\Lambda}_g.$$

9. Recentre the ensemble perturbations to obtain the (local) analysis ensemble:
$$\mathbf{X}^a = \overline{\mathbf{X}}^a + \mathbf{X'}^f.$$

10. Having performed steps 3 to 9 for all local domains, form the global analysis ensemble: Reverse the first procedure of step 3 by inserting the local analysis ensemble matrices (the outputs of step 9) into the corresponding rows of the $d \times m$ matrix $\mathbf{X}_g^a$.

2.3.3 Computational Cost

In the original paper of NETF Tödter and Ahrens [2015] summarises the computational cost of the NETF stating that the computational expense of NETF is similar to the Ensemble Transform Kalman Filter (ETKF) [Lawson and Hansen, 2004, Sakov et al., 2012] for a given ensemble size since analysis is performed in the m-dimensional subspace spanned by the ensemble members. In addition the NETF filter does not involve the computation of an inverse matrix thus preventing computational instabilities caused by considerably small singular values, which are sometimes neglected in ETKF implementations for that reason [Sakov et al., 2012]. If localisation is applied to NETF, the local analysis are independent and can be computed 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 pre-calculated random matrices since they only depend on ensemble size m.

2.4 Multivariate Rank Histogram Filter

Most data assimilation methods are *transformation methods*: At analysis times, each particle is *transformed*, i.e. modified by a correction designed to take it closer to the observations. Transform methods typically include the numerous variants of the Ensemble Kalman Filter. Only transformation methods have been successfully used for high-dimensional problems so far. However, they do generally not resolve the full non-Gaussian data assimilation problem (typically, where the particles form multimodal probability densities). As illustrated by Metref et al.

[2014], this is because they always rely on linear considerations to correct unobserved variables after the correction of observed variables.

By contrast, *sampling methods* do not modify particles, but rather select the particles closest to the observations. The original particle filter and the Sequential Importance Resampling filter are the best-known sampling methods. Sampling methods are attractive because they theoretically solve the full non-Gaussian data assimilation problem. However, they are well known to fail in large dimensions problems with large numbers of observations.

A *Holy Graal* of data assimilation would be a transformation method, applicable to high-dimensional problems, which solves the full non-Gaussian problem. This is the reason for developing the EWPF, presented previously, and the Multivariate Rank Histogram Filter (MRHF). A detailed description of the MRHF is provided in Metref et al. [2014].

2.4.1 Description

For the present exposition, it is assumed that the state vector is composed of only 3 scalar variables: $\mathbf{X} = (x, y, z)$, and that only the last one z is observed with z^o, characterised by the likelihood $p(z^o|z)$. Extension to larger state vectors and more independent observations is straightforward. Given the prior probability density $p(x, y, z)$ and the likelihood $p(z^o|z)$, the assimilation of z^o is meant to find the posterior, conditional density $p(x, y, z|z^o)$. The latter is given by Bayes' rule. After introducing the Knothe-Rosenblalt rearrangement ($p(x, y, z) = p(z)p(x|z)p(y|x, z)$), the posterior density writes:

$$p(x, y, z|z^o) \quad = \quad p(z|z^o)p(x|z, z^o)p(y|x, z, z^o). \tag{2.85}$$

This density can be calculated sequentially by first calculate $p(z|z^o)$, then $p(x|z, z^o)$, and $p(y|x, z, z^o)$. This is actually what some implementations of the EnKF do, see for example [Anderson, 2003]. The correction on z for each particle is computed with

$$\delta z_i = \frac{\text{Var}(z)}{\text{Var}(z) + \text{Var}(\epsilon)}(z^o - z_i - \epsilon_i), \tag{2.86}$$

where i is the particle index, Var(z) is the variance of z computed from the ensemble, ϵ is the observation error, ϵ_i a realisation of this error. Then, non-observed variables are corrected using a linear regression of the z corrections. For x, member i, this writes:

$$\delta x_i = \frac{\text{Cov}(x, z)}{\text{Var}(z)}\,\delta z_i. \tag{2.87}$$

Here, Cov(x, z) represents the covariance between x and z computed from the ensemble. Corrections for other non-observed variables (y in our case) take a similar shape. In particular, the conditioning to x for y (Eq. 2.85) does not affect the nature of the correction. This is due to the Gaussian assumption hidden behind Eq. 2.86 and 2.87, which are nothing less than the Best Linear Unbiased Estimate (BLUE) equations in a developed format.

It is well known that Eqns. 2.86 and 2.87 provide an optimal correction when the prior ensemble and the likelihood are Gaussian. It is not optimal, and it can even be detrimental to use these formulas for non-Gaussian priors. Equation 2.86

is typically inadequate if the prior density of z is bimodal, because the analysis can create non-physical estimates between the two modes. Similar problems can arise with Eqn. 2.87, when x and z exhibits a strongly nonlinear statistical relationship. This is illustrated by Fig. 2.1 with a simple example.

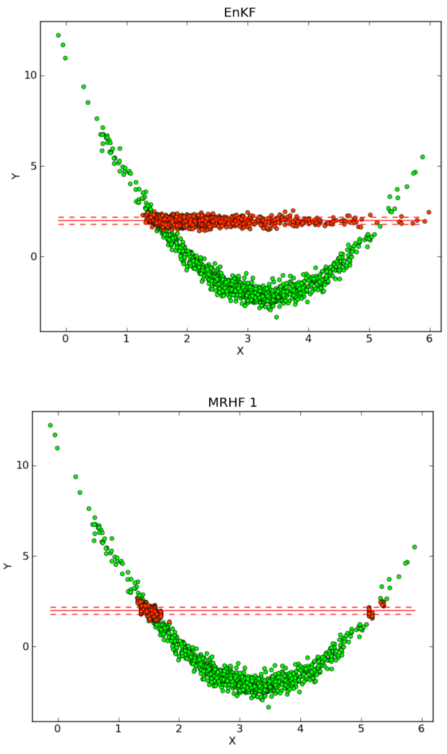

Figure 2.1: Illustration of an ensemble analysis in a strongly nonlinear case: EnKF on the left, MRHF on the right. The scatterplots show ensembles in the space of 2 variables X and Y. The green ensembles are the priors. Only Y is observed; The observation (along with error standard deviation) is materialised by the solid (resp. dashed) lines. The red dots represents the analysed ensembles. By creates particles where there was no particle in the prior, the EnKF does not comply with the prior and possibly results in unrealistic physical states.

The MRHF is designed to solve Eq. 2.85 sequentially, as the EnKF in the form of Anderson [2003], but using general transformations for z and x, in contrast with Eq. 2.86 and 2.87 that are very specific to the linear-Gaussian framework. Following the method proposed by Anderson [2010] under the name of *Rank Histogram Filter* (RHF), the continuous prior density for z is represented as a rank histogram. The histogram is composed of $N-1$ bounded regions partitioned by the sorted ensemble members (the order statistics of the problem) and two unbounded regions on the edge. In each inner region, a density value is assigned so that the region contains a probability mass of $\frac{1}{N+1}$. The two outer regions are covered by tails of probability mass $\frac{1}{N+1}$ as well; Their shape may be chosen freely, and this may actually be a key element for the success of the RHF Anderson [2010]. The likelihood $p(z^o|z)$, known analytically from the observation error density, is discretised on the same grid as $p(z)$, and the two functions are multiplied point-wise to provide a constant piecewise expression (after normalisation) of the posterior density $p(z|z^o)$. The analysis ensemble is finally obtained using a (deterministic) procedure of inversion of the cumulative distribution function.

In the original form of RHF, proposed by Anderson [2010], the corrections on z particles are calculated by the difference between the posterior and the prior values of z, and used with Eq. 2.87 to correct non-observed variables. Corrections are still performed using a linear regression. In contrast, the MRHF extends the nonlinear, Rank Histogram-based analysis to non-observed variables. Let $\{z_i^a\}_{i=1,...,N}$ be the posterior ensemble of the observed variable z, i.e. a sample of $p(z|z^o)$. Consider the first unobserved variable, x in Eq. 2.85. The MRHF analysis determines x_i^a, the x analysis value for particle i, by deterministically sampling the conditional density $p_i^a(x) \equiv p(x|z = z_i^a)$. This density must first be formed. Some steps of the procedure are illustrated on Fig. 2.2. The green dots represent the prior ensemble in the $X-Z$ plane; the blue dots at $X = 0$ represent the z analysis ensemble $\{z_i^a\}_{i=1,...,N}$. The red line is the observation realisation. The following process is repeated for $i = 1, ..., N$. For a given i, a few particles are selected in the prior ensemble (green dots with blue circles), whose z values lie in the neighbourhood of z_i^a (blue dot with red triangle) along the z direction. The details of the selection process, based on simple Euclidean distances, are given in Metref et al. [2014]. Applying the rank histogram approach to the x values of the selected particles, a one-dimensional density is then formed to represent $p_i^a(x)$.

To follow Eq. 2.85, one approach would be to draw a random realisation from this density to provide x_i^a. This, however, is far from optimal from the physical viewpoint, because it can generate large corrections resulting in physical instabilities and imbalances. In Fig. 2.2 for instance, the prior particle (green dot with red triangle) is in the right hand side mode of the distribution. Since the observation does not enable one to know in which mode the truth (red dot) actually is, it makes sense to try to keep this particle in its mode of origin, thus minimising its modification.

Instead of a random draw in $p_i^a(x)$ which could arbitrarily move the particle to the left hand side mode, the following steps are proposed:

- With a similar selection and a rank histogram process, form the density of x conditioned to the *background* value of z: $p_i^b(x) \equiv p(x|z = z_i^b)$;
- Compute the cumulative distribution functions $C_i^b(x)$ and $C_i^a(x)$ from $p_i^b(x)$ and $p_i^a(x)$, respectively;
- Compute the position of the prior particle in the prior density: $c_i = C_i^b(x_i^b)$;
- Preserve the rank of the particle in the posterior density by taking $x_i^a = C_i^{a\,(-1)}(c_i)$ as analysis value for x and particle i. This is illustrated on Fig. 2.2.

Although this method does not prevent a particle shifting from one mode to another, two neighbouring particles in the prior ensemble are likely to remain neighbours in the posterior ensemble.

Once the z and x analysis values are computed for each particle, the analysis values for the third variable y can be computed. The process is strictly similar to the one described above, but for the variable y, and with an additional $x = x_i^a$ term in the conditional statement. This reduces to selecting particles from the prior distribution in the neighbourhood of (x_i^a, z_i^a) in the two-dimensional plane (x, z), to form the density $p(y|x, z, z^o)$. In practice though, the conditioning to the previously analysed non-observed variables is worth being omitted for computational issues.

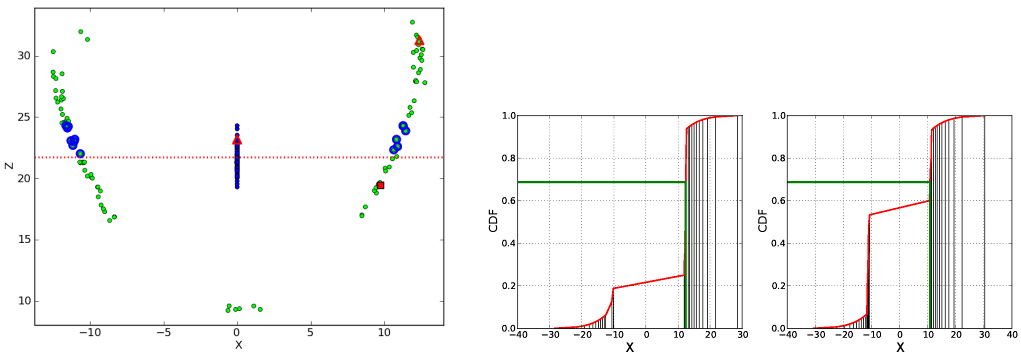

Figure 2.2: Left: Illustration of the particle-by-particle MRHF analysis step for the first unobserved variable: Green dots represent the prior ensemble; blue dots vertically aligned at $X = 0$ represent the posterior z ensemble. The red dotted line is the observation of z, the red square is the true state (not used for the analysis). The red empty triangles show the particle being processed and its corresponding z analysis value. Blue circles show the selected particles to form the posterior density $p_i^a(x)$. Right: Illustration of the particle-by-particle MRHF analysis step for the first unobserved variable (one particle): Red lines represent the cumulative distribution functions (cdf) of the prior density $p_i^b(x)$ (left panel) and the posterior density $p_i^a(x)$ (right panel). The vertical black lines show the selected particles used to build these densities. To compute the x analysis value for the prior x particle near 11 on the left panel, the green line must be followed: The cdf for the prior x is computed with the prior cdf (Result is near 0.68); from this cdf value, the x analysis value is obtained on the right panel.

2.4.2 Workflow

Forecast steps are standard ensemble integrations. The MRHF is a serial analysis scheme: Observations must be processed one by one. It proceeds as follows:

Loop on observations:

1. Isolate variables affected by the analysis (for localisation);
2. Correct observed variable z using the RHF approach:
   - Sort the prior ensemble z_i^b;
   - Create prior density $p(z)$ from the histogram;
   - Create likelihood $p(z|z^o)$ on the same "grid" than the histogram;
   - Compute posterior density by multiplying point-wise the prior density and the likelihood;

- Resample the posterior using an inversion of the Cumulative Distribution Function (CDF);
- Apply the inverse sorting process to find the posterior ensemble z_i^a;

3. Correct non-observed variables.
   Loop on non-observed variables (x):
   Loop on ensemble particles (i):

   - Select particles of the prior ensemble which observed value z lies in the neighbourhood of the **prior** observed value of the processed particle, z_i^b;
   - Form the **prior** density $p(x|z = z_i^b)$ from these particles and a rank histogram approach;
   - Compute the corresponding CDF, $C_i^b(x)$;
   - Select particles of the prior ensemble which observed value z lies in the neighbourhood of the **posterior** observed value of the processed particle, z_i^a;
   - Form the **posterior** density $p(x|z = z_i^a)$ from these particles and a rank histogram approach;
   - Compute the corresponding CDF, $C_i^a(x)$;
   - Compute the position of the prior particle in the prior density: $c_i = C_i^b(x_i^b)$;
   - Preserve the rank of the particle in the posterior density by computing $x_i^a = C_i^{a\,(-1)}(c_i)$;

4. From prior and posterior ensembles, compute corrections and apply tapering function for localisation.

2.4.3 Computational cost

The computational cost has not been characterised in detail so far. The algorithm detailed above is expensive because it involves sorting operations inside intricated loops on the particles and the variables. A computationally more efficient algorithm is presently under study, where, basically, the loop on ensemble particles would be removed. Note, however, that the workflow follows the steps of an EnKF (in Anderson's form), with the correction of the observed variable followed by the correction of non-observed variables, imbedded in the serial processing of observations. The MRHF is quite flexible then: It is possible to process only a few observations, or even a few variables, with the MRHF, the rest being processed with an EnKF. Note also that the MRHF formulation naturally complies with localisation.

2.5 Local anamorphosis transformation (CNRS/LGGE)

2.6 Description

The basic problem of the algorithm is to look for a nonlinear change of variable transforming a random variable X with known cumulative distribution function (cdf) $F(x) = p(X \leq x)$ into a new random variable Z with the target cdf $G(z) = p(Z \leq z)$. Elementary probability calculus provides a general solution for the forward and backward transformations:

$$Z = G^{-1}\left[F(X)\right] \quad \text{and} \quad X = F^{-1}\left[G(Z)\right] \tag{2.88}$$

providing that F and G are invertible. In particular, if $Z \sim \mathcal{U}(0,1)$ is uniformly distributed on the interval $[0,1]$, with $G(z) = z$, then $x = F^{-1}(k/q)$ is the kth q-quantile of X; and if $Z \sim \mathcal{N}(0,1)$ is normally distributed, with $G(z) = \frac{1}{2}[1 + \text{erf}(z/\sqrt{2})]$, then Eq. (2.88) defines the forward and backward Gaussian anamorphosis transformation of the random variable X [Wackernagel, 2003, chapter 33].

2.6.1 Efficient approximate algorithm

In the Monte Carlo estimation methods (like the ensemble Kalman filter), the prior probability distribution for the control variables is only approximately described by a finite-size sample. The anamorphosis transformation in Eq. (2.88) for each control variable can thus only be approximately computed from the available sample using a nonparametric estimate $\tilde{F}(x)$ of the exact marginal cdf $F(x)$. The most simple nonparametric estimate of a probability density function (pdf) $\tilde{f}(x) = d\tilde{F}(x)/dx$ is the histogram: a piecewise constant pdf $\tilde{f}(x)$, or a piecewise linear cdf $\tilde{F}(x)$. As a simple choice for the classes of the histogram, we may use prescribed quantiles $\tilde{x}_k,\ k = 1,\ldots,q$ of the input sample, i.e. such that $\tilde{F}(\tilde{x}_k) = r_k$, for a given set of r_k $(0 \leq r_k \leq 1,\ r_k < r_{k+1})$. In this way, we can control explicitly the fraction of ensemble members $(r_{k+1} - r_k)$ in each class of the histogram.

Then, with the same level of approximation, we can use the same histogram representation of the Gaussian distribution, i.e. a piecewise linear $\tilde{G}(z)$ interpolating the true Gaussian cdf between $G(z_k) = r_k,\ k = 1,\ldots,q$, so that the anamorphosis transformation in Eq. (2.88) is also piecewise linear:

$$\varphi_{\text{forw}}(x) = \tilde{G}^{-1}\left[\tilde{F}(x)\right] = z_k + \frac{z_{k+1} - z_k}{\tilde{x}_{k+1} - \tilde{x}_k}(x - \tilde{x}_k) \quad \text{for} \quad x \in [\tilde{x}_k, \tilde{x}_{k+1}] \tag{2.89}$$

$$\varphi_{\text{back}}(z) = \tilde{F}^{-1}\left[\tilde{G}(z)\right] = \tilde{x}_k + \frac{\tilde{x}_{k+1} - \tilde{x}_k}{z_{k+1} - z_k}(z - z_k) \quad \text{for} \quad z \in [\tilde{z}_k, \tilde{z}_{k+1}] \tag{2.90}$$

This approximate transformation remaps the quantiles $\tilde{x}_k,\ k = 1,\ldots,q$ of the input sample on the corresponding Gaussian quantiles $z_k,\ k = 1,\ldots,q$, and interpolates linearly between them. It is bijective between the interval $[\tilde{x}_1, \tilde{x}_q]$ and $[z_1, z_q]$, providing that the quantiles $\tilde{x}_k$ are distinct: $\tilde{x}_k \neq \tilde{x}_{k+1}\ \forall k$.

A full description of this algorithm, with many examples, can be found in Brankart et al. [2012].

2.6.2 Example

Figure 2.3 shows for instance the approximate Gaussian anamorphosis transformation that is obtained with Eq. (2.89) using a 200-member random sample

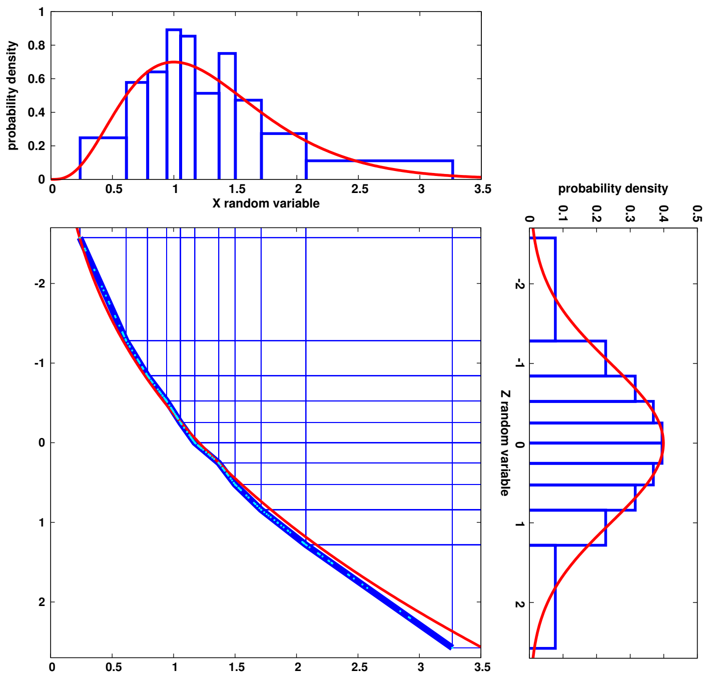

Figure 2.3: Approximate piecewise linear Gaussian anamorphosis transformation (thick blue curve), remapping the deciles $\tilde{x}_k$ of a 200-member random sample of the Gamma distribution $\Gamma(k, \theta)$ (top histogram) on the Gaussian deciles z_k (left histogram), as compared to the exact transformation (in red) transforming the exact $\Gamma(k, \theta)$ (red curve superposed to the top histogram) into $\mathcal{N}(0, 1)$ (red curve superposed to the left histogram).

of the Gamma distribution $X \sim \Gamma(k, \theta)$, with $k = 4.236$ and $\theta = 0.309$ (chosen so that the mode is equal to 1, and the 95% percentile is equal to 2.5). The classes of the histogram for X are defined using the 10-quantiles (or deciles) of the random sample: $r_k = k/q$, with $q = 10$. They are remapped on the Gaussian deciles z_k (histogram on the left) using the piecewise linear transformation (blue curve), which is here not far from the exact transformation (red curve), given by Eq. (2.88). With this definition of r_k, there is the same number of random draws in each class of the histogram.

2.6.3 Workflow

The workflow of the algorithm is very simple since it only requires inserting 3 additional steps in any Gaussian observational update algorithm: three additional steps before the classic Gaussian observational update, and one additional step after the classic Gaussian observational update:

1. Compute quantiles of the input ensemble, in state space and in observation space;
2. Perform forward anamorphosis transformation of the input ensemble, in state space and in observation space;
3. Perform forward anamorphosis transformation of the observation vector;
4. Apply the classic Gaussian observational update to the transformed ensemble and transformed observation; and
5. Perform backward anamorphosis transformation of the updated ensemble.

2.6.4 Computational cost

The first reason why such a simple approximation of the Gaussian anamorphosis may be useful in practical ocean applications is that it can be performed at a numerical cost that is usually much smaller than the numerical cost of a Gaussian observational update (e.g. the analysis step of the ensemble Kalman filter). In the identification of the approximate transformation in (2.89), the main cost is associated to the computation of the quantiles $\tilde{x}_k$ of the input sample. If m is the size of the sample, this cost is proportional to $m \log m$, to sort the sample values. Then, if n is the size of the control vector (i.e. the number of random variables to transform), the total computational complexity to identify the functions φ_{forw} and φ_{back} in Eqs. (2.89) and (2.90) is:

$$C_{\text{quantiles}} \sim nm \log m. \tag{2.91}$$

In addition, in order to perform the observational update, one must apply the transformation to the ensemble forecast and to the observations. Each transformation requires localising the input value among the quantiles $\tilde{x}_k$ (with complexity proportional to $\log_2 q$ with a bisection method), and then applying the corresponding linear transformation in Eq. (2.89) (i.e. about 3 operations). To transform the ensemble of m control vectors, together with the p observations values, and then the updated ensemble back in the original control space, this corresponds to a computational complexity of:

$$C_{\text{anamorphosis}} \sim (2mn + p)(3 + \alpha \log_2 q), \tag{2.92}$$

where α stands for the relative numerical cost between numerical comparisons (needed to localize values in the list of quantiles) and algebraic operations (needed to compute the linear transformations). Transforming the observations simply requires applying the observation operator to the quantiles of the control vector, but if some observations are nonlinearly linked to the control vector, it may be better to augment the control vector with these observations (thus producing a problem with larger n) and transform them using their own anamorphosis transformation.

On the other hand, this simple algorithm does not require a lot of memory or disk space to store the approximate functions φ_{forw} and φ_{back}: only the quantiles of the input ensemble $\tilde{x}_k,\ k = 1, \ldots, q$ need to be stored, for a total storage of qn real values (i.e. less than the storage of the forecast ensemble itself, which requires storing mn real values). See the appendix for more details about the practical implementation of the algorithm.

Conclusions

In this deliverable the filters and smoothers that have been newly implemented in the different toolboxes during the SANGOMA have been described. Specific emphasis has been placed on the workflow of each new method and its computational costs. All methods described explore nonlinearity in some form, from extensions to existing linear filters to fully nonlinear particle filters. The methods chosen are Proper Orthogonal Decomposition (POD), two particle filters, the Equivalent-Weights Particle Filter and the Implicit Equal-Weights Particle Filter, the second-order exact Nonlinear Ensemble Transform Filter and Multivariate Rank Histogram Filter.

All of these methods are applicable to high-dimensional ocean models, and as such should be a valuable extension to present-day practise in academia and operational oceanography centres. Deliverable 3.4 describes the implementation of the methods in the toolboxes in more detail, including examples of application on small and sometimes medium-size benchmark cases.

Within Sangoma project we have implemented these new nonlinear methods in the various data assimilation toolboxes and hence, they are easily usable with any model that is connected to the data assimilation toolbox. Further, since all partners have complied with data model specifications in work package 1 these methods are easily transferable from their current toolboxes to any another toolbox. The ease of using code implemented in Sangoma was seen in Deliverable 3.3 were EWPF was implemented within most of Sangoma toolboxes (of which most use very different languages and data models).

Bibliography

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

M. Ades and P. J. van Leeuwen. The equivalent-weights particle filter in a high-dimensional system. *Q. J. R. Meteorol. Soc.*, 141:484–503, 2015.

M. U. Altaf, A. W. Heemink, and M. Verlaan. Inverse shallow-water flow modelling using model reduction. *International Journal for Multiscale Computational Engineering*, 7:577–596, 2009.

M. U. Altaf, M. Verlaan, and A. W. Heemink. Efficient identification of uncertain parameters in a large scale tidal model of european continental shelf by proper orthogonal decomposition. *Int. J. Numer. Meth. Fluids*, 68:422–450, 2012.

J. Anderson. A local least squares framework for ensemble filtering. *Monthly Weather Review*, 131:634–642, 2003.

J. Anderson. A non-gaussian ensemble filter update for data assimilation. *Monthly Weather Review*, 138:4186–4198, 2010.

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

A. C. Antoulas. *Approximation of large-scale Dynamical Systems*. USA: SIAM, 2005.

Thomas Bengtsson, Peter Bickel, and Bo Li. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *IMS Collections Probability and Statistics: Essays in Honor of David A. Freedman*, 2:316–334, 2008.

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

J.-M. Brankart, C.-E. Testut, D. Bèal, M. Doron, C. Fontana, M. Meinvielle, P. Brasseur, and J. Verron. Towards an improved description of ocean uncertainties: effect of local anamorphic transformations on spatial correlations. *Ocean Science*, 8:121–142, 2012.

G. Burgers, P. J. van Leeuwen, and G. Evensen. On the analysis scheme in the ensemble Kalman filter. *Mon. Wea. Rev.*, 126:1719–1724, 1998.

J. Carrera and S. P. Neuman. Estimation of aquifer parameters under transient and steady state conditions, part 1: Maximum likelihood method incorporating prior information. *Wat. Resourc. Res.*, 22:199–210, 1986.

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

A. J. Chorin, M. Morzfeld, and X. Tu. Interpolation and iteration for nonlinear filters. *Communications in Applied Mathematics and Computational Science*, 5:221–240, 2010.

SE Cohn and R Todling. Approximate data assimilation schemes for stable and unstable dynamics. *J. Meteor. Soc. Japan*, 74:63–75, 1996.

P. Courtier and O. Talagrand. Variational assimilation of meteorological observations with direct and adjoint shallow water equations. *Tellus*, 42A:531–549, 1990.

R. Delay. *Atmospheric data analysis*. Cambridge Unicersity Press, 1991.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.

A. Doucet, N. de Freitas, and N. Gordon. Sequential monte-carlo methods in practice. *Springer-Verlag: Berlin*, , 2001.

H. Elbern, H. Schmidt, and A. Ebel. Variational data assimilation for tropospheric chemistry modeling. *J. Goephys. Res.*, 102:15967–15985, 1997.

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

G. Evensen. Sequential data assimilation with a nonlinear quasi-geostropic model using monte carlo methods to forecast error statistics. *J. Geophys. Res.*, 99:10,143–10,162, 1994b.

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

G. Evensen and P. J. van Leeuwen. Assimilation of geosat altimeter data for the aghulas current using the ensemble Kalman filter with a quasi-geostrophic model. *Mon. Wea. Rev.*, 124:85–96, 1996.

B. A. Francis. *A Course in H_∞ Control Theory*. Springer-Verlag, 1987.

Max D. Gunzburger. Reduced-order modeling, data compression and the design of experiments. In *Second DOE workshop on multiscale Mathematics*, Broomfield, Colorado, July 20–22 2004.

A. W. Heemink and H. Kloosterhuis. Data assimilation for non-linear tidal models. *International Journal for Numerical Methods in Fluids*, 11:1097–1112, 1990.

A. W. Heemink, E. E. A. Mouthaan, and M. R. T. Roest. Inverse 3D shallow water flow modeling of the continental shelf. *Continental Shelf Research*, 22: 465–484, 2002.

I. Hoteit, D. T. Pham, and J. Blum. A semi-evolutive partially local filer for data assimilation. *Marine Pollution Bulletin*, 43:164–174, 2001.

I. Hoteit, D. T. Pham, and J. Blum. A simplified reduced order Kalman filtering and application to altimetric data assimilation in Tropical Pacific. *Journal of Marine Systems*, 36:101–127, 2002.

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

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

T. Kaminski, R. Giering, and M. Scholze. An example of an automatic differentiation-based modeling system. *Lecture Notes Comput. Sci.*, 2668:5–104, 2003.

K. Karhunen. Zur spektral theorie stochasticher prozsee. *Ann. Acad. Sci. Fenicae*, 34 (A1):1–7, 1946.

R. W. Lardner, A. H. Al-Rabeh, and N. Gunay. Optimal estimation of parameters for a two dimensional hydrodynamical model of the arabian gulf. *J. Geophys. Res. Oceans*, 98:229–242, 1993.

W. G. Lawson and J. A. Hansen. Implications of stochastic and deterministic filters as ensemble-based data assimilation methods in varying regimes of error growth. *Monthly Weather Review*, 132:1966–1981, 2004.

M. Loeve. Functions aleatoire de second ordre. *Revue Science*, 84:195–206, 1946.

E. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20:130–141, 1963.

J. L. Lumley. The structure of inhomogeneous turbulence. In *A. M. Yaglom and V. I. Tatarski (eds)*, pages 166–178, Nauka, Moscow, 1967.

X. Luo and I. Hoteit. Robust ensemble filtering and its relation to covariance inflation in the ensemble Kalman filter. *Mon. Wea. Rev.*, 139:3938–3953, 2011.

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

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

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

D. S. Oliver, A. C. Reynolds, and N. Liu. *Inverse theory for petroleum reservoir characterization and history matching*. UK: Cambridge, 2008.

K. Pearson. On lines and planes of closest fit to points in space. *Phil. Mag.*, 2(6): 559–572, 1901.

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

D. T. Pham, J. Verron, and M. C. Roubaud. A singular evolutive extended Kalman filter for data assimilation in oceanography. *Journal of Marine Systems*, 16: 323–340, 1998.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, **94**(446):590–599, 1999.

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

A. Segers, A. W. Heemink, M. Verlaan, and M. van Loon. A modified rrsqrt-filter for assimilating data in atmospheric chemistry models. *Environmental Modeling and Software*, 15:663–671, 2000.

D. Simon. *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006.

L. Sirovich. choatic dynamics of coherent structures. *Physica D*, 37:126–145, 1987.

C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136:4629–4640, 2008.

C. Snyder, T. Bengtsson, and M. Morzfeld. Performance bounds for particle filters using the optimal proposal. *Monthly Weather Review*, **143**:4750–4761, 2015.

O. Talagrand. Assimilation of observations, an introduction. *Journal of the Meteorological Society of Japan*, 75:191–209, 1997.

P. G. J. Ten-Brummelhuis, A. W. Heemink, and H. F. P. van den Boogard. Identification of shallow sea models. *Int. J. for Num. Met. Fluids*, 17:637–665, 1993.

F. X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: Theoratical aspects. *Tellus*, 38:97–110, 1986.

W. C. Thacker and R. B. Long. Fitting models to inadequate data by enforcing spatial and temporal smoothness. *J. Geophys. Res.*, 93:10655–10664, 1988.

Julian Tödter and Bodo Ahrens. A Second-Order Exact Ensemble Square Root Filter for Nonlinear Data Assimilation. *Mon. Wea. Rev.*, 143(4):1347–1367, 2015. ISSN 0027-0644.

E. Tziperman, W. C. Thacker, and R. B. Long. Oceanic data analysis using a general circulation model, part2: A north atlantic model. *J. Phys. Oceanography*, 22:1458–1485, 1992.

D. S. Ulman and R. E. Wilson. Model parameter estimation for data assimilation modeling: Temporal and spatial variability of the bottom drag coefficient. *J. Geophys. Res. Oceans*, 103:5531–5549, 1998.

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

P. J. Van Leeuwen. Nonlinear data assimilation in qeosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, **136**:1991–1999, 2010.

P. J. Van Leeuwen. Efficient nonlinear data-assimilation in geophysical fluid dynamics. *Computers and Fluids*, **46**:52–58, 2011.

M. Verlaan and A. W. Heemink. Data assimilation schemes for non-linear shallow water flow models. *Adv. Fluid Mechanics*, 96:277–286, 1996.

M. Verlaan and A. W. Heemink. Tidal flow forecasting using reduced rank square root filters. *Stochastic Hydrology and Hydraulics*, 11:349 – 368, 1997.

P. T. M. Vermeulen and A. W. Heemink. Model-reduced variational data assimilation. *Mon. Wea. Rev.*, 134:2888–2899, 2006.

H. Wackernagel. *Multivariate Geostatistics: An Introduction with Applications*. Springer, 2003.

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

M. Zhu, P. J. van Leeuwen, and J. Amezcua. Implicit equal-weights particle filter. *Q. J. R. Meteorol. Soc.*, page personal communication, 2015.