ORIGINAL PAPER

A sparse matrix formulation of model-based ensemble Kalman filter

Håkon Gryvill[1] · **Håkon Tjelmeland**[1]

© The Author(s) 2023

Abstract

We introduce a computationally efficient variant of the model-based ensemble Kalman filter (EnKF). We propose two changes to the original formulation. First, we phrase the setup in terms of precision matrices instead of covariance matrices, and introduce a new prior for the precision matrix which ensures it to be sparse. Second, we propose to split the state vector into several blocks and formulate an approximate updating procedure for each of these blocks. We study in a simulation example the computational speedup and the approximation error resulting from using the proposed approach. The speedup is substantial for high dimensional state vectors, allowing the proposed filter to be run on much larger problems than can be done with the original formulation. In the simulation example the approximation error resulting from using the introduced block updating is negligible compared to the Monte Carlo variability inherent in both the original and the proposed procedures.

1 Introduction

State-space models are frequently used in many applications. Examples of application areas are economics (Creal [2011](); Chan and Strachan [2020]()), weather forecasting (Houtekamer and Zhang [2016](); Hotta and Ota [2021]()), signal processing (Loeliger et al. [2007]()) and neuroscience (Smith and Emery [2003]()). A state-space model consists of an unobserved latent $\{x_t\}$ discrete time Markov process and a related observed $\{y_t\}$ process, where y_t gives information about x_t, and the y_t's are assumed to be conditionally independent given the $\{x_t\}$ process. Given observed values $y_1, \ldots, y_t$ the goal is to do inference about one or more of the x_t's. In this article the focus is on the filtering problem, where the goal is to find the conditional distribution of x_t given observations $y_{1:t} = (y_1, \ldots, y_t)$. The filtering problem is also known as sequential data assimilation and online inference.

The Markov structure in the specification of the state-space model allows the filtering problem to be solved recursively. Having available a solution of the filtering problem at time $t-1$, i.e. the distribution $p(x_{t-1}|y_{1:t-1})$, one can first use this filtering distribution to find the one-step forecast distribution $p(x_t|y_{1:t-1})$, which one in turn can combine with the observed data at the next time step, y_t, to find $p(x_t|y_{1:t})$. The identification of the forecast distribution is often termed the forecast or prediction step, whereas the process of finding the filtering distribution is called the update or analysis step. If a linear-Gaussian model is assumed analytical solutions are available both for the forecast and update steps, and is known as the Kalman filter (Kalman [1960](); Kalman and Bucy [1961]()). Another situation where an analytical solution is available for the forecast and update steps is when x_t is a vector of categorical or discrete variables. In essentially all other situations, however, analytical solutions are not available and one has to resort to approximate solutions. Ensemble-based methods are here the most popular choice, where each distribution of interest is represented by a set of particles, or an ensemble of realisations, from the distribution of interest. In ensemble-based methods one also alternates between a forecast and an update step. The forecast step is straightforward to implement without approximations in ensemble methods, whereas the update step is challenging. In the update step an ensemble of realisations, $x_t^{(1)}, \ldots, x_t^{(\mathcal{M})}$, from the forecast distribution $p(x_t|y_{1:t-1})$ is available and the task is to update each realisation $x_t^{(m)}$ to a corresponding realisation $\widetilde{x}_t^{(m)}$ from

✉ Håkon Gryvill
  hakon.gryvill@ntnu.no

  Håkon Tjelmeland
  haakon.tjelmeland@ntnu.no

1 Department of Mathematical Sciences, Norwegian University of Science and Technology, Alfred Getz' vei 1, 7034 Trondheim, Norway

the filtering distribution $p(x_t|y_{1:t})$. The forecast distribution $p(x_t|y_{1:t-1})$ serves as a prior in the update step, and the filtering distribution $p(x_t|y_{1:t})$ is the resulting posterior. In the following we therefore refer to the ensemble of realisations from the forecast distribution as the prior ensemble and the ensemble of realisations from the filtering distribution as the posterior ensemble.

There are two popular classes of ensemble filtering methods, particle filters (Doucet et al. 2001; Doucet and Johansen 2011) and ensemble Kalman filters (EnKFs) (Burgers et al. 1998; Evensen 2009). Particle filters are based on importance sampling and resampling and has the advantage that it can be shown to converge to the correct filtering solution in the limit when the number of particles goes towards infinity. However, in applications with high dimensional state vectors, x_t, particle filters typically collapse when run with a finite number of particles. The updating procedure in EnKFs is based on a linear-Gaussian model, but experience in applications is that EnKFs produce good results also in situations where the linear-Gaussian assumptions are not fulfilled. However, except when the linear-Gaussian model is correct the EnKF is only approximate, even in the limit when the number of ensemble elements goes towards infinity. Most of the EnKF literature is in applied journals and has not much focus on the underlying statistical theory, but some publications has also appeared in statistical journals (Sætrom and Omre 2013; Katzfuss et al. 2016, 2020; Loe and Tjelmeland 2020, 2022).

In EnKF the update step consists of two parts. First a prior covariance matrix is established based on the prior ensemble, which thereafter is used to modify each prior ensemble element to a corresponding posterior element. Many variants of the original EnKF algorithm of Evensen (1994) have been proposed and studied in the literature, see for example Evensen (2009) and Houtekamer and Zhang (2016) and references therein. In particular the original algorithm of Evensen (1994) is known to severely underestimate the uncertainty and much attention has focused on how to correct for this. The most frequent approach to this problem is the rather ad hoc solution of variance inflation, see the discussion in Luo and Hoteit (1997). Houtekamer and Mitchell (1997) identified the source of the problem to be inbreeding, that the same prior ensemble elements that are first used to establish the prior covariance matrix are thereafter modified into posterior ensemble elements. To solve this problem Houtekamer and Mitchell (1997) proposed to split the prior ensemble in two, where the prior ensemble elements in one part is used to establish a prior covariance matrix which is used when modifying the prior ensemble elements in the other part into corresponding posterior elements. As discussed in Houtekamer and Zhang (2016) this cross validation setup is later generalised to a situation where the prior ensemble is split into more than two parts. Myrseth et al. (2013) propose to use a resampling approach to solve the inbreed-

ing problem. Another issue with the original EnKF setup of Evensen (2009) that has been focused in the literature is that the empirical covariance matrix of the prior ensemble is used as the prior covariance matrix, thereby effectively ignoring the associated estimation uncertainty. To cope with this problem Omre and Myrseth (2010) set the problem of identifying the prior covariance matrix in a Bayesian setting. To obtain an analytically tractable posterior distribution for the mean vector and the covariance matrix of the prior ensemble, the natural conjugate normal inverse Wishart prior is adopted. For each ensemble element, a sample from the resulting normal inverse Wishart posterior distribution is generated and used in the updating of that ensemble element. Corresponding Bayesian schemes for establishing the prior covariance matrix are later adopted in Bocquet (2011), Bocquet et al. (2015) and Tsyrulnikov and Rakitko (2017). Loe and Tjelmeland (2021) propose to base the updating step of the EnKF on an assumed Bayesian model for all variables involved. The updating step is defined by restricting it to be consistent with the assumed model at the same time as it should be robust against modeling error. Both the need for defining a prior for the covariance matrix and the cross validation approach discussed above comes as necessary consequences of this setup. In addition it entails that the new data y_t should be taken into account when generating the prior covariance matrix.

In the present article we adopt the model-based EnKF approach of Loe and Tjelmeland (2021), but our focus is to formulate a computationally efficient variant of this approach. To obtain this we do two important changes. First, in Loe and Tjelmeland (2021) the natural conjugate normal inverse Wishart prior is used for the mean vector and covariance matrix of the prior ensemble elements. When sampling from the corresponding posterior distribution this results in a covariance matrix that is full. To update a prior ensemble element to the corresponding posterior element the covariance matrices must be used in a series of matrix operations, such as various matrix decompositions and multiplications with vectors. We rephrase the approach of Loe and Tjelmeland (2021) to use precision matrices, or inverse covariance matrices, instead of covariance matrices and propose to use a Gaussian partially ordered Markov model (Cressie and Davidson 1998) to construct a prior for the mean vector and the precision matrix of the prior ensemble. This prior is also conjugate for the assumed normal likelihood for the ensemble elements, and the resulting posterior is therefore analytically tractable. Moreover, with this prior distribution we are able to ensure that the precision matrices sampled from the resulting posterior distribution are band matrices. With a band structure in the precision matrix some of the matrix operations in the updating of a prior ensemble element to the corresponding posterior element can be done more efficiently. However, some of the matrix operations, singular value decompositions, do not benefit from this band structure. To make also

this part of the procedure computationally more efficient we propose to do an approximation which allows us to replace the singular value decomposition of one large matrix with singular value decompositions of several smaller matrices.

Compared to the updating procedure in the original EnKF setup of Evensen (1994), the updating procedure resulting from adopting the Bayesian scheme introduced in Omre and Myrseth (2010) is clearly computationally much more expensive. The increased computational demands come for two reasons. First, the Bayesian setup requires one covariance matrix to be generated for each ensemble element, whereas the original EnKF setup uses the same covariance matrix for all ensemble elements. Second, the covariance matrices generated by the original Bayesian setup of Omre and Myrseth (2010) are full and without any special structure, whereas the covariance matrix used in the original EnKF setup has a low rank which can be used to speed up the necessary matrix computations. In the present paper we resolve the second issue by adopting a prior model producing sparse precision matrices. Regarding the first issue, Loe and Tjelmeland (2021) demonstrated that with their model-based EnKF approach the generated ensemble gave a realistic representation of the uncertainty, whereas the original EnKF setup produced ensembles which underestimated the uncertainty. For small ensemble sizes the underestimation of the uncertainty when adopting the original EnKF setup is severe, whereas with large enough ensemble sizes it becomes negligible. To obtain a realistic representation of the uncertainty we therefore have two possible strategies. One can either adopt the model-based EnKF approach used in Loe and Tjelmeland (2021) and in the present paper, or one can use the original EnKF setup with a very large number of ensemble elements. Which of the two approaches that are computationally more efficient depends on the computational cost of running the forward function. If the forward function is computationally expensive, so that this part dominates the total computing time of the filter, it is clearly computationally most efficient to adopt the model-based EnKF procedure introduced in the present paper. If the computation time is not dominated by the running of the forward function, the picture is less clear. Depending on how many ensemble elements that are necessary to get a sufficiently small underestimation of the uncertainty, the original EnKF with a large number of ensemble elements may be the computationally most efficient alternative.

This article is structured as follows. In Sect. 2 we first review some numerical algorithms for sparse matrices and some properties of the Gaussian density. In the same section we also present the state-space model. In Sect. 3 we rephrase the approach introduced in Loe and Tjelmeland (2021) to use precision matrices instead of covariance matrices. The new prior for the precision matrix is proposed in Sect. 4. In Sect. 5 we propose a computationally efficient approximation of the updating procedure presented in Sect. 3. We present results of

simulation examples in Sect. 6, and finally we provide some closing remarks in Sect. 7.

2 Preliminaries

This section introduces material necessary to understand the approach proposed in later sections. We start by discussing some numerical properties of sparse matrices, thereafter review how the multivariate Gaussian distribution can be formulated in terms of precision matrices. Lastly, we introduce the state-space model and provide a brief introduction on simulation-based techniques.

2.1 Numerical properties of sparse matrices

Suppose that $x, y \in \mathbb{R}^n$, that $Q \in \mathbb{R}^{n \times n}$ is a given symmetric positive definite band matrix with bandwidth p, where $n \gg p$, that y is given, and that we want to solve the matrix equation

$$Qx = y \tag{1}$$

with respect to x. Since we assume that Q is a symmetric matrix, we can solve the equation above using the Cholesky decomposition $Q = LL^T$, where $L \in \mathbb{R}^{n \times n}$ is a lower triangular matrix. Due to the band structure of Q we know that L has lower bandwidth p, which in turn enables us to compute L with complexity $\mathcal{O}(np^2)$, see Algorithm 2.9 in Rue and Held (2005). Since we now are able to compute L efficiently, we can make use of Algorithm 2.1 in Rue and Held (2005) to solve (1) efficiently as well. In the following, we describe the algorithm step by step.

First, we rewrite (1) as

$$LL^T x = y. \tag{2}$$

If we define $L^T x = v$, we can solve (1) by first solving $Lv = y$ for v and then solving $L^T x = v$ for x. We can exploit the band structure of L to solve these equations efficiently. Using that L is lower triangular, we can solve $Lv = y$ row by row, using "forward substitution" (Rue and Held 2005, p. 32). We denote the (i, j)th entry of L as $L^{i,j}$ and the ith element of y as y^i. The ith element of v, denoted v^i, is computed as follows

$$v^i = \frac{1}{L^{i,i}} \left(y^i - \sum_{j=\max\{1, i-p\}}^{i-1} L^{i,j} v^j \right). \tag{3}$$

Similarly, we can solve $v = L^T x$ for x using "backward substitution"

$$x^i = \frac{1}{L^{i,i}} \left(v^i - \sum_{j=i+1}^{\min\{i+p,n\}} L^{j,i} x^j \right). \tag{4}$$

Notice that we compute the entries of x "backwards"; we first compute x^n and then move our way backwards to x^1. If we again assume $n \gg p$, the computational complexity of forward and backward substitution is $\mathcal{O}(np)$. This means that the overall complexity of computing x using the presented approach is $\mathcal{O}(np^2)$. Note that solving (1) with the "brute force" approach, i.e. computing $x = Q^{-1}y$, has computational complexity $\mathcal{O}(n^3)$.

Backward substitution can also be used to sample efficiently from a Gaussian distribution when the precision matrix is a band matrix. Assume that z is a vector of standard normally distributed variables and that we want to simulate x from a Gaussian distribution with mean 0 and covariance matrix Q^{-1}. We can simulate x by solving

$$L^T x = z \tag{5}$$

using backward substitution, as specified in (4). When Q is a band matrix with bandwidth p, where $n \gg p$, the computational complexity is $\mathcal{O}(np^2)$. However, when Q is full the computational complexity of simulating x is $\mathcal{O}(n^3)$.

2.2 Gaussian distribution phrased with precision matrices

Let $x \in \mathbb{R}^n$ have a multivariate Gaussian distribution with mean $\mu \in \mathbb{R}^n$ and precision matrix $Q \in \mathbb{R}^{n \times n}$. We let $\mathcal{N}(x; \mu, Q)$ denote the density function of this distribution, i.e.

$$\mathcal{N}(x; \mu, Q) = (2\pi)^{-n/2} |Q|^{1/2}$$
$$\exp\left(-\frac{1}{2}(x-\mu)^T Q (x-\mu) \right). \tag{6}$$

Defining $A \subset \{1, \ldots n\}$ and $B = \{1, \ldots, n\} \backslash A$, we can partition x, μ and Q into blocks

$$x = \begin{pmatrix} x^A \\ x^B \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu^A \\ \mu^B \end{pmatrix}, \quad Q = \begin{pmatrix} Q^{AA} & Q^{AB} \\ Q^{BA} & Q^{BB} \end{pmatrix}. \tag{7}$$

According to Theorem 2.5 in Rue and Held (2005), we then have that

$$p(x^A | x^B) = \mathcal{N}(x^A; \mu^A - (Q^{AA})^{-1} Q^{AB} (x^B - \mu^B), Q^{AA}). \tag{8}$$

Similarly,

$$p(x^B) = \mathcal{N}(x^B; \mu^B, Q^{BB} - (Q^{AB})^T (Q^{AA})^{-1} Q^{AB}). \tag{9}$$

The last expression can be derived by picking out the parts of the mean vector μ and covariance matrix Q^{-1} that corresponds to x^B. When we have found the covariance matrix of x^B we find the precision matrix by inversion. From the first expression we see that computing the precision matrix for $x^A | x^B$ is particularly easy when the Gaussian distribution is formulated with precision matrices.

2.3 State-space model

A state-space model (Shumway and Stoffer 2016; Brockwell and Davis 1990) consists of a set of latent variables, denoted $\{x_t\}_{t=1}^T$, $x_t \in \mathbb{R}^{n_x}$, and a set of observations $\{y_t\}_{t=1}^T$, $y_t \in \mathbb{R}^{n_y}$. The latent variables follow a first order Markov chain with initial distribution $p(x_1)$ and transition probabilities $p(x_t | x_{t-1})$, $t \geq 2$. That is, the joint distribution for $x_{1:T} = (x_1, \ldots, x_T)$ can be written as

$$p(x_{1:T}) = p(x_1) \prod_{t=2}^T p(x_t | x_{t-1}). \tag{10}$$

In addition, each observation y_t is considered to be conditionally independent of the remaining observations, given x_t. The joint likelihood for the observations $y_{1:T} = (y_1, \ldots, y_T)$ can be formulated as

$$p(y_{1:T} | x_{1:T}) = \prod_{t=1}^T p(y_t | x_t). \tag{11}$$

A state-space model is illustrated by a directed acyclic graph (DAG) in Fig. 1, where each variable is represented with a node and the edges symbolise the dependencies between the variables.

The main interest of this article, and the reason we introduce the state-space model, is to assess the filtering problem. The objective of the filtering problem is to compute the filtering distribution, $p(x_t | y_{1:t})$, for $t = 1, \ldots, T$. That is, we want to find the distribution for the latent variable at time t, i.e. x_t, given all of the observations up to the same time step, $y_{1:t}$. Due to the Markov assumptions made in the state-space model, we are in principle able to assess this quantity sequentially. Each iteration is performed in two steps, namely

$$p(x_t | y_{1:t-1}) = \int p(x_t | x_{t-1}) p(x_{t-1} | y_{1:t-1}) dx_{t-1}, \tag{12}$$

$$p(x_t | y_{1:t}) = \frac{p(x_t | y_{1:t-1}) p(y_t | x_t)}{\int p(x_t | y_{1:t-1}) p(y_t | x_t) dx_t}. \tag{13}$$

The first expression is the prediction step and gives the forecast distribution $p(x_t | y_{1:t-1})$, while the second expression is called the update step and yields the filtering distribution $p(x_t | y_{1:t})$. Note that the filtering distribution is computed
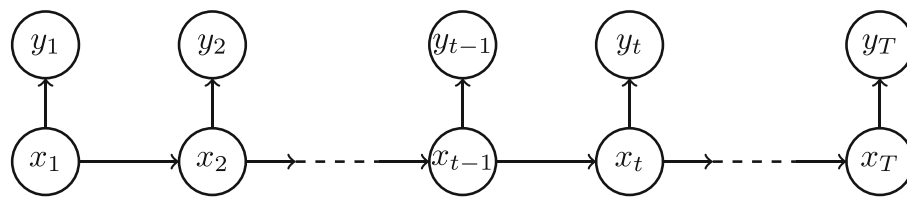
Fig. 1 DAG illustration of a state-space model. The edges illustrate the stochastic dependencies between the nodes. The latent variables x_t, $t = 1, \ldots, T$ are unobserved, while y_t, $t = 1, \ldots, T$ are observations

through Bayes' rule, where $p(x_t|y_{1:t-1})$ is the prior, $p(y_t|x_t)$ is the likelihood and $p(x_t|y_{1:t})$ is the posterior. In the following we therefore use the terms prior and forecast, and the terms posterior and filtering, interchangeably. In this article, our main focus is on the update step.

When the state-space model is linear and Gaussian, the expressions can be solved analytically through the Kalman filter (Kalman 1960). However, the prediction and update steps in (12) and (13) are generally speaking not feasible to solve analytically. Hence approximative methods are used. A common approach is to use simulation-based techniques, where a set of realisations, which is called an ensemble, are used to explore the state-space model by moving the realisations forward in time according to the state-space model. The filtering and forecast distributions are then represented by an ensemble at each time step, which is initially sampled from $p(x_1)$. The following paragraph provides an overview of how this is done for one iteration.

Assume that a set of $\mathcal{M}$ independent realisations $\{\widetilde{x}_{t-1}^{(1)}, \ldots, \widetilde{x}_{t-1}^{(\mathcal{M})}\}$ from $p(x_{t-1}|y_{1:t-1})$ is available at time t. If we are able to simulate from the forward model $p(x_t|x_{t-1})$, we can obtain $\mathcal{M}$ independent realisations from $p(x_t|y_{1:t-1})$ by simulating from $x_t^{(m)}|\widetilde{x}_{t-1}^{(m)} \sim p(x_t|\widetilde{x}_{t-1}^{(m)})$ independently for $m = 1, \ldots, \mathcal{M}$. This is the prediction step, and can usually be performed without any approximations. Next, we use the prediction, or prior, ensemble $\{x_{t-1}^{(1)}, \ldots, x_{t-1}^{(\mathcal{M})}\}$ to obtain samples from the filtering distribution $p(x_t|y_{1:t})$, which is often called a posterior ensemble. This can be done by conditioning the samples from the forecast distribution, $\{x_t^{(1)}, \ldots, x_t^{(\mathcal{M})}\}$, on the new observation y_t. This step is called the update step and is generally not analytically feasible. Hence approximative methods are necessary. In the following section, we present a procedure that enables us to carry out the update step.

3 Model-based EnKF

In this section we start by reviewing the model-based EnKF framework introduced in Loe and Tjelmeland (2021). The focus in our presentation is on the underlying model framework, the criterion used for selecting the particular chosen update, and on the resulting updating procedure. We do not include the mathematical derivations leading to the computational procedure. Moreover, we phrase the framework in terms of precision matrices, whereas Loe and Tjelmeland (2021) use covariance matrices.

The focus in the section is on how to use a prior ensemble $\{x_t^{(1)}, \ldots, x_t^{(\mathcal{M})}\}$ to update one of these prior ensemble elements, number m say, to a corresponding posterior ensemble element $\widetilde{x}_t^{(m)}$. All the variables involved in this operation are associated with the same time t. To simplify the notation we therefore omit the time subscript t in the following discussion. So we write $\{x^{(1)}, \ldots, x^{(\mathcal{M})}\}$ instead of $\{x_t^{(1)}, \ldots, x_t^{(\mathcal{M})}\}$ for the available prior ensemble, we write $\widetilde{x}^{(m)}$ instead of $\widetilde{x}_t^{(m)}$ for the generated posterior ensemble element number m, we write x instead of x_t for the latent variable at time t, and we write y instead of y_t for the new data that becomes available at time t.

3.1 Assumed Bayesian model

In model-based EnKF the updating of a prior ensemble element $x^{(m)}$ to the corresponding posterior ensemble element $\widetilde{x}^{(m)}$ is based on an assumed model. The dependence structure of the assumed model is illustrated in the DAG in Fig. 2. It should be noted that the assumed model is not supposed to be correct, it is just used as a mean to formulate a reasonable and consistent updating procedure. To stress this we follow the notation in Loe and Tjelmeland (2021) and use f as the generic symbol for all densities associated to the assumed model, whereas we continue to use p to denote the corresponding correct (and typically unknown) densities. We use subscripts on f to specify what stochastic variables the density relates to. For example $f_{y|x}(y|x)$ is the conditional density for the new observations y given the latent variable x.

In the assumed model we let the prior ensemble elements $x^{(1)}, \ldots, x^{(\mathcal{M})}$ and the unobserved latent variable x at time t be conditionally independent and identically Gaussian distributed with a mean vector μ and a precision matrix Q, i.e.

$$f_{x|\theta}(x|\theta) = \mathcal{N}(x; \mu, Q), \tag{14}$$

$$f_{x^{(i)}|\theta}(x^{(i)}|\theta) = \mathcal{N}(x^{(i)}; \mu, Q), \quad i = 1, \ldots, \mathcal{M}, \tag{15}$$
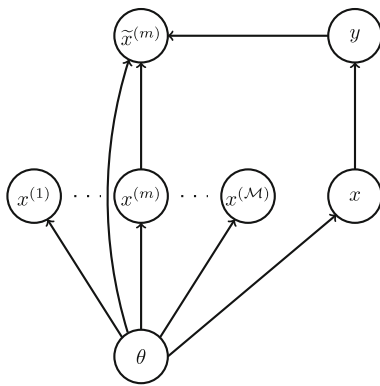
Fig. 2 DAG representation of the assumed Bayesian model for updating the mth realisation

where $\theta = (\mu, Q)$. For the parameter θ of this Gaussian density we assume some prior distribution $f_\theta(\theta)$. Loe and Tjelmeland (2021) assume a normal inverse Wishart prior for (μ, Q^{-1}), which implies that Q is full, whereas we want to adopt a prior which ensures that Q is a band matrix. In Sect. 4.1 we detail the prior we are using. The last element of the assumed model is to let the new data y be conditionally independent of $x^{(1)}, \ldots, x^{(\mathcal{M})}$ and θ given x, and

$$f_{y|x}(y|x) = \mathcal{N}(y; Hx, R). \tag{16}$$

Based on the assumed model the goal is to construct a procedure for generating an updated version $\widetilde{x}^{(m)}$ of $x^{(m)}$. One should first generate a $\theta = (\mu, Q)$ and thereafter $\widetilde{x}^{(m)}$. In addition to being a function of $x^{(m)}$ it is natural to allow $\widetilde{x}^{(m)}$ to depend on the generated θ and the new data y, as also indicated in the DAG in Fig. 2. The corresponding conditional distribution for $\widetilde{x}^{(m)}$ we denote by $q(\widetilde{x}^{(m)}|x^{(m)}, \theta, y)$. The updated $\widetilde{x}^{(m)}$ is to be used as an (approximate) sample from $p(x_t|y_{1:t})$, which in the ensemble based setting we are considering here is represented by $p(x_t|x_t^{(1)}, \ldots, x_t^{(\mathcal{M})}, y_t)$. However, as we want to use $x^{(m)}$ as a source for randomness when generating $\widetilde{x}^{(m)}$, the $x^{(m)}$ must be removed from the conditioning set so one should instead consider $p(x_t|x_t^{(1)}, \ldots, x_t^{(m-1)}, x_t^{(m+1)}, \ldots, x_t^{(\mathcal{M})}, y_t)$. Under the assumed model this density is equal to $f_{x|z^{(m)},y}(x|z^{(m)}, y)$ when using the shorthand notation $z^{(m)} = (x^{(1)}, \ldots, x^{(m-1)}, x^{(m+1)}, \ldots, x^{(\mathcal{M})})$. Thus, we should construct $q(\widetilde{x}^{(m)}|x^{(m)}, \theta, y)$ so that

$$f_{\widetilde{x}^{(m)}|z^{(m)},y}\left(x|z^{(m)}, y\right) = f_{x|z^{(m)},y}\left(x|z^{(m)}, y\right) \tag{17}$$

holds for all values of x, $z^{(m)}$ and y. Loe and Tjelmeland (2021) show that (17) is fulfilled if $\widetilde{x}^{(m)}$ is generated by first sampling $\theta = (\mu, Q)$ from $f_{\theta,z^{(m)},y}(\theta|z^{(m)}, y)$ and thereafter $\widetilde{x}^{(m)}$ is sampled according to a $q(\widetilde{x}^{(m)}|x^{(m)}, \theta, y)$ defined by

$$\widetilde{x}^{(m)} = B\left(x^{(m)} - \mu\right) + \mu + K(y - H\mu) + \widetilde{\epsilon}^{(m)}, \tag{18}$$

where $\widetilde{\epsilon}^{(m)}$ is independent of everything else and generated from a zero-mean Gaussian distribution with a covariance matrix S, B is a matrix connected to the positive semidefinite S by the relation

$$BQ^{-1}B^T + S = (I - KH)Q^{-1}, \tag{19}$$

and K is the Kalman gain matrix

$$K = (Q + H^T R H)^{-1} H^T R. \tag{20}$$

We note in passing that by inserting for K in (19) and thereafter applying the Woodbury identity (Woodbury 1950), one gets $(I - KH)Q^{-1} = (Q + H^T RH)^{-1}$, so (19) is equivalent to

$$BQ^{-1}B^T + S = (Q + H^T R H)^{-1}. \tag{21}$$

3.2 Optimality criterion

When applying the updating Eq. (18) we have a freedom in the choice of the matrices B and S. The restrictions are that S must be positive semidefinite and that B and S should be related as specified by (21).

Under the assumed model all choices of B and S fulfilling the given restrictions are equally good, as they are all generating an $\widetilde{x}^{(m)}$ from the same distribution. When recognising that the assumed model is wrong, however, all solutions are no longer equally good. So we should choose a solution which is robust against the assumptions made in the assumed model. Loe and Tjelmeland (2021) formulate a robust solution as one where the $x^{(m)}$ is changed as little as possible when forming $\widetilde{x}^{(m)}$, under the condition that B and S satisfy (21). The intuition is that this should allow non-Gaussian properties in $x^{(m)}$ to be transferred to $\widetilde{x}^{(m)}$. Mathematically the criterion is formulated as minimising the expected squared Euclidean distance between $x^{(m)}$ and $\widetilde{x}^{(m)}$. Thus, we should minimise

$$\mathrm{E}\left[\left(x^{(m)} - \widetilde{x}^{(m)}\right)^T\left(x^{(m)} - \widetilde{x}^{(m)}\right)\right], \tag{22}$$

with respect to B and S under the restriction (21), where the expectation is with respect to the joint distribution of $x^{(m)}$ and $\widetilde{x}^{(m)}$ under the assumed model. Note that Loe and Tjelmeland (2021) is considering a slightly more general solution by using a Mahalanobis distance. Loe and Tjelmeland (2021) show that (22) is minimised under the specified restrictions when $S = 0$ and

$$B = U\Lambda^{-\frac{1}{2}}PF^T D^{\frac{1}{2}}V^T, \tag{23}$$

Algorithm 1 Summary of the resulting updating procedure for computing the posterior ensemble $\widetilde{x}^{(1)}, \ldots, \widetilde{x}^{(\mathcal{M})}$ from the prior ensemble $x^{(1)}, \ldots, x^{(\mathcal{M})}$.

Given $x^{(1)}, \ldots, x^{(\mathcal{M})}$, y, H and R.
For $m = 1, \ldots, \mathcal{M}$:
1. Sample $\theta = (\mu, Q)$ from $f_{\theta|z^{(m)}, y}(\theta|z^{(m)}, y)$.
2. Do singular value decomposition in (24): $V D V^T = Q$.
3. Do singular value decomposition in (25): $U \Lambda U^T = Q + H^T R H$.
4. Evaluate Z in (26): $Z = \Lambda^{-\frac{1}{2}} U^T V D^{-\frac{1}{2}}$.
5. Do singular value decomposition in (26): $P G F^T = Z$.
6. Evaluate B in (23): $B = U \Lambda^{-\frac{1}{2}} P F^T D^{\frac{1}{2}} V^T$.
7. Evaluate $\widetilde{x}^{(m)}$: $\widetilde{x}^{(m)} = B(x^{(m)} - \mu) + \mu + K(y - H\mu)$

---

where U and V are orthogonal matrices and D and Λ are diagonal matrices given by the singular value decompositions

$$Q = V D V^T, \tag{24}$$

$$Q + H^T R H = U \Lambda U^T, \tag{25}$$

and P and F are orthogonal matrices given by the singular value decomposition

$$Z = \Lambda^{-\frac{1}{2}} U^T V D^{-\frac{1}{2}} = P G F^T. \tag{26}$$

3.3 Resulting updating procedure

The resulting procedure for updating a prior ensemble $x^{(1)}, \ldots, x^{(\mathcal{M})}$ to the corresponding posterior ensemble $\widetilde{x}^{(1)}, \ldots, \widetilde{x}^{(\mathcal{M})}$ is summarised by the pseudocode in Algorithm 1.

In the following sections our focus is on how to make this updating procedure computationally efficient when the dimensions of the state vector and corresponding observation vector are large. First, in Sect. 4, we propose a new prior for θ which enables efficient sampling of $\theta = (\mu, Q)$ from the corresponding posterior $f_{\theta|z^{(m)}, y}(\theta|z^{(m)}, y)$. The generated Q is then a sparse matrix, which also limits the memory usage necessary to store it, since we of course only need to store the non-zero elements of Q. However, sparsity of Q does not influence the computational efficiency of the singular value decompositions in Steps 2, 3 and 5 of Algorithm 1. One should note that we may rephrase Steps 2 and 3 to use Cholesky decompositions instead, since Q and $Q + H^T R H$ are symmetric and positive definite matrices. Under reasonable assumptions also $Q + H^T R H$ is a sparse matrix so thereby these two steps can be done computationally efficient. The Z in Step 5, however, is in general neither a symmetric positive definite matrix, nor sparse. To introduce Cholesky decompositions in Steps 2 and 3 will therefore not change the order of the computational complexity of the procedure. Instead of substituting the singular value decompositions in Steps 2 and 3

with Cholesky decompositions, we therefore in Sect. 5 propose an approximation of Steps 2–6 by splitting the state vector into a series of blocks and running Steps 2–6 for each of the blocks separately.

4 Prior model and sampling of $\theta = (\mu, Q)$

In this section we first formulate the new prior for $\theta = (\mu, Q)$, where Q is restricted to be a band matrix. Thereafter we formulate a computationally efficient procedure to generate a sample from the resulting $f_{\theta|z^{(m)}, y}(\theta|z^{(m)}, y)$. We start by formulating the prior.

4.1 Prior for $\theta = (\mu, Q)$

To obtain a band structure for the precision matrix Q we restrict the assumed Gaussian distributions in (14) and (15) to be a Gaussian partially ordered Markov model, a Gaussian POMM (Cressie and Davidson 1998). To be able to give a mathematically precise definition of the Gaussian POMM we first introduce some notation.

We let x^k denote the k'th element of x, so $x = (x^1, \ldots, x^{n_x})$. To each element x^k of x we associate a sequential neighbourhood $\Lambda_k \subseteq \{1, \ldots, k-1\}$, and use the notation introduced in Sect. 2.2 to denote the elements in x associated to Λ_k by x^{Λ_k}. The number of elements in Λ_k we denote by $|\Lambda_k|$. Moreover, we let $\Lambda_k(1)$ denote the smallest element in the set Λ_k, we let $\Lambda_k(2)$ be the second smallest element in Λ_k, and so on until $\Lambda_k(|\Lambda_k|)$, which is the largest element in Λ_k. We let the distribution of x be specified by the two parameter vectors $\eta = (\eta^1, \ldots, \eta^{n_x})$ and $\phi = (\phi^1, \ldots, \phi^{n_x})$, where for each $k = 1, \ldots, n_x$ we have that $\eta^k \in \mathbb{R}^{|\Lambda_k|+1}$ and $\phi^k > 0$ is a scalar. With this notation the Gaussian POMM is specified as

$$f_{x|\eta, \phi}(x|\eta, \phi) = \prod_{k=1}^{n_x} f_{x^k|\eta^k, \phi^k, x^{\Lambda_k}}\left(x^k|\eta^k, \phi^k, x^{\Lambda_k}\right), \tag{27}$$

where $x^k|\eta^k, \phi^k, x^{\Lambda_k}$ is Gaussian with mean

$$
\begin{aligned}
E\left[x^k|\eta^k, \phi^k, x^{\Lambda_k}\right] &= \eta^{k,1} + x^{\Lambda_k(1)}\eta^{k,2} + \cdots \\
&\quad + x^{\Lambda_k(|\Lambda_k|)}\eta^{k,|\Lambda_k|+1} \\
&= \left[1 \ \left(x^{\Lambda_k}\right)^T\right]\eta^k
\end{aligned}
\tag{28}
$$

and variance $\text{Var}[x^k|\eta^k, \phi^k, x^{\Lambda_k}] = \phi^k$.

It should be noted that $\theta = (\mu, Q)$ in $\mathcal{N}(x; \mu, Q)$ is uniquely specified by η and ϕ. If the number of sequential neighbours is small, the resulting precision matrix Q becomes sparse, see Appendices A and B for a detailed discussion. We can therefore specify a prior for θ by specifying a prior for η and ϕ, which we choose as conjugate to the Gaussian POMM just defined. More specifically, we first assume the elements in ϕ to be independent, and each element ϕ^k to be inverse Gamma distributed with parameters α^k and β^k. Next, we assume the elements of η to be conditionally independent and Gaussian distributed given ϕ,

$$
\eta^k|\phi \sim \mathcal{N}\left(\eta^k; \zeta^k, \left(\phi^k \Sigma_{\eta^k}\right)^{-1}\right),
\tag{29}
$$

where $\zeta^k \in \mathbb{R}^{|\Lambda_k|+1}$ and $\Sigma_{\eta^k} \in \mathbb{R}^{(|\Lambda_k|+1)\times(\Lambda_k|+1)}$ are hyperparameters that have to be set.

4.2 Sampling from $f_{\theta|z^{(m)}, y}(\theta|z^{(m)}, y)$

To sample from $f_{\theta|z^{(m)}, y}(\theta|z^{(m)}, y)$ we adopt the same general strategy as proposed in Loe and Tjelmeland (2021). We include the underlying state vector at time t, x, as an auxiliary variable and simulate (θ, x) from

$$
\begin{aligned}
&f_{\theta, x|z^{(m)}, y}(\theta, x|z^{(m)}, y) \\
&\quad \propto f_\theta(\theta) f_{x|\theta}(x|\theta) f_{y|x}(y|x) \prod_{i \neq m} f_{x^{(i)}|\theta}(x^{(i)}|\theta) \\
&\quad = f_\theta(\theta) \mathcal{N}(x; \mu, Q) \mathcal{N}(y; Hx, R) \prod_{i \neq m} \mathcal{N}(x^{(i)}; \mu, Q).
\end{aligned}
\tag{30}
$$

By thereafter ignoring the simulated x we have a sample of θ from the desired distribution. To simulate from the joint distribution $f_{\theta, x|z^{(m)}, y}(\theta, x|z^{(m)}, y)$ we adopt a two block Gibbs sampler, alternating between drawing x and θ from the full conditionals $f_{x|\theta, z^{(m)}, y}(x|\theta, z^{(m)}, y)$ and $f_{\theta|x, z^{(m)}, y}(\theta|x, z^{(m)}, y)$, respectively. We initialise the Gibbs sampler by setting $x = \frac{1}{\mathcal{M}-1}\sum_{i \neq m} x^{(i)}$. This initial value should be centrally located in $f_{x|z^{(m)}, y}(x|z^{(m)}, y)$, and since the Gibbs sampler we are using only consists of two blocks we should expect it to converge very fast. So just a few iterations should suffice.

From (30) we get the full conditional for x

$$
\begin{aligned}
f_{x|\theta, z^{(m)}, y}(x|\theta, z^{(m)}, y) &= f_{x|\theta, y}(x|\theta, y) \\
&\propto \mathcal{N}(x; \mu, Q)\,\mathcal{N}(y; Hx, R).
\end{aligned}
\tag{31}
$$

It is straightforward to show that this is a Gaussian distribution with mean $\widetilde{\mu}$ and covariance matrix $\widetilde{Q}$ given by

$$
\widetilde{\mu} = \mu + (Q + H^T R H)^{-1} H^T R(y - H\mu),
\tag{32}
$$

$$
\widetilde{Q} = Q + H^T R H.
\tag{33}
$$

As discussed in Appendices A and B, the Q becomes sparse whenever the number of sequential neighbours is small. If x represents values in a two dimensional lattice and the sequential neighbourhoods Λ_k are chosen as translations of each other, as we use in the simulation example in Sect. 6, the Q is a band matrix, again see the discussion in Appendix B. Assuming also R and H to be band matrices, the product $H^T R H$ can be efficiently computed and is also a band matrix. The $\widetilde{Q}$ is thereby also a band matrix, so the Cholesky decomposition of it can be computed efficiently as discussed in Sect. 2.1. The band structures of H and R can be used to evaluate the right hand side of (32) efficiently, and in addition computational efficiency can be gained by computing the product $H^T R(y - H\mu)$ in the right order. In general, multiplying two matrices $C \in \mathbb{R}^{u \times v}$ and $D \in \mathbb{R}^{v \times w}$ has computational complexity $\mathcal{O}(uvw)$. Hence, we should first compute $R(y - H\mu)$ before calculating $H^T R(y - H\mu)$. Having $\widetilde{Q}$ and the Cholesky decomposition of $\widetilde{Q}$ we can both get $\widetilde{\mu}$ and generate a sample from the Gaussian distribution efficiently as discussed in Sect. 2.1.

From (30) we also get the full conditional for θ,

$$
\begin{aligned}
f_{\theta|x, z^{(m)}, y}(\theta|x, z^{(m)}, y) &= f_{\theta|x, z^{(m)}}(\theta|x, z^{(m)}) \\
&\propto f_\theta(\theta)\,\mathcal{N}(x; \mu, Q) \\
&\quad \prod_{i \neq m} \mathcal{N}(x^{(i)}; \mu, Q).
\end{aligned}
\tag{34}
$$

To simulate from this full conditional we exploit that $\theta = (\mu, Q)$ is uniquely given by the parameters ϕ and η of the Gaussian POMM prior, which means that we can first simulate values for ϕ and η from $f_{\phi, \eta|x, z^{(m)}}(\phi, \eta|x, z^{(m)})$ and thereafter use the generated ϕ and η to compute the corresponding μ and Q. In Appendix C we study in detail the resulting $f_{\phi, \eta|x, z^{(m)}}(\phi, \eta|x, z^{(m)})$ and show that it has the same form as the corresponding prior $f_{\phi, \eta}(\phi, \eta)$, but with updated parameters. More specifically, the elements of ϕ are conditionally independent given x and $z^{(m)}$, with

$$
\phi^k|z^{(m)}, x \sim \text{InvGam}\left(\widetilde{\alpha}^k, \widetilde{\beta}^{(m),k}\right),
\tag{35}
$$

where

Algorithm 2 Summary of the block updating procedure for computing the posterior ensemble $\widetilde{x}^{(1),\text{Block}}, \ldots, \widetilde{x}^{(\mathcal{M}),\text{Block}}$ from the prior ensemble $x^{(1)}, \ldots, x^{(\mathcal{M})}$.

Given $x^{(1)}, \ldots, x^{(\mathcal{M})}$, y, H, R and C_b, D_b, E_b for $b = 1, \ldots, \mathcal{B}$.

For $m = 1, \ldots, \mathcal{M}$:

1.　　Sample $\theta = (\mu, Q)$ from $f_{\theta|z^{(m)},y}(\theta|z^{(m)}, y)$.

2.　　For $b = 1, \ldots, \mathcal{B}$:

　　　a.　　Form the set J_b in (47).

　　　b.　　Form $g(x^{E_b}, y^{J_b}|\theta)$ in (48).

　　　c.　　Form $g(x^{D_b}, y^{J_b}|\theta)$ by marginalising $g(x^{E_b}, y^{J_b}|\theta)$ over $x^{E_b \backslash D_b}$.

　　　d.　　Form $g(x^{D_b}|\theta)$ and $g(y^{J_b}|\theta, x^{D_B})$ from $g(x^{D_b}, y^{J_b}|\theta)$, and identify $\widetilde{\mu}^b$,　　$\widetilde{Q}^b, \widetilde{a}^b, \widetilde{H}^b$ and $\widetilde{R}^b$.

　　　e.　　Compute $\widetilde{x}^{(m),D_b}$ by Steps 2 to 7 in Algorithm 1 when $x^{(m)}, \widetilde{x}^{(m)}$,
　　　　　y, μ, Q, H and R are replaced with $x^{(m),D_b}, \widetilde{x}^{(m),D_b}, y^{J_b} - \widetilde{a}^b, \widetilde{\mu}^b$,
　　　　　$\widetilde{Q}^b, \widetilde{H}^b$ and $\widetilde{R}^b$, respectively.

3.　　Define $\widetilde{x}^{(m),\text{Block}}$ by setting $\widetilde{x}_\ell^{(m),\text{Block}} = \widetilde{x}_{\ell_b}^{(m),D_b}$ for $\ell \in C_b$,
　　　$b = 1, \ldots, \mathcal{B}$.

$$\widetilde{\alpha}^k = \alpha^k + \frac{\mathcal{M}}{2}, \tag{36}$$

$$\widetilde{\beta}^{(m),k} = \left(\frac{1}{\beta^k} + \frac{1}{2}\left(\gamma^{(m),k} - \left(\rho^{(m),k}\right)^T \left(\Theta^{(m),k}\right)^{-1} \rho^{(m),k}\right)\right)^{-1}, \tag{37}$$

$$\gamma^{(m),k} = \left(\zeta^k\right)^T \Sigma_{\eta^k}^{-1} \zeta^k + \left(\chi^{(m),k}\right)^T \cdot \chi^{(m),k}, \tag{38}$$

$$\rho^{(m),k} = \Sigma_{\eta^k}^{-1} \zeta^k + \left(1, \left(\chi^{(m),\Lambda_k}\right)^T\right)^T \cdot \left(\chi^{(m),k}\right)^T, \tag{39}$$

$$\Theta^{(m),k} = \Sigma_{\eta^k}^{-1} + \left(1, \left(\chi^{(m),\Lambda_k}\right)^T\right)^T \cdot \left(1, \left(\chi^{(m),\Lambda_k}\right)^T\right) \tag{40}$$

and

$$\chi^{(m),k} = \left(x^{(1),k}, \ldots, x^{(m-1),k}, x^{(m+1),k}, \ldots, x^{(\mathcal{M}),k}, x^k\right), \tag{41}$$

$$\chi^{(m),\Lambda_k} = \left(x^{(1),\Lambda_k}, \ldots, x^{(m-1),\Lambda_k}, x^{(m+1),\Lambda_k}, \ldots, \right.$$
$$\left. x^{(\mathcal{M}),\Lambda_k}, x^{\Lambda_k}\right). \tag{42}$$

The distribution for η given ϕ, x and $z^{(m)}$ becomes

$$f_{\eta|\phi,z^{(m)},x}\left(\eta|\phi, z^{(m)}, x\right) = \prod_{k=1}^{n_x} f_{\eta^k|\phi^k,z^{(m)},x}\left(\eta^k|\phi^k, z^{(m)}, x\right) \tag{43}$$

where

$$\eta^k|\phi^k, z^{(m)}, x \sim \mathcal{N}\left(\eta^k; \left(\Theta^{i,k}\right)^{-1} \rho^{i,k}, \left(\phi^k\right)^{-1} \Theta^{i,k}\right). \tag{44}$$

In particular we see that it is easy to sample from $f_{\phi,\eta|x,z^{(m)}}(\phi, \eta|x, z^{(m)})$ by first sampling the elements of ϕ independently according to (35) and thereafter generate the elements of η according to (44). Having samples of ϕ and η we can thereafter compute the corresponding μ and Q as detailed in Appendix A.

5 Block update

Section 3 presents a set of updating procedures that allows us to update a prior realisation into a posterior realisation, where the posterior realisation takes the observation in the current time step into account. In Sect. 3.2, we found the optimal filter according to our chosen criterion. However, as also discussed in Sect. 3.3, parts of this procedure is computationally demanding. In the following we introduce the approximate, but computationally more efficient procedure for Steps 2 to 7 in Algorithm 1 that we mentioned in Sect. 3.3. So the situation considered is that we want to update a prior realisation $x^{(m)}$ and that we already have generated a mean vector μ and a sparse precision matrix Q. The goal is now to define an approximation to the vector $\widetilde{x}^{(m)}$ given in Step 7 in Algorithm 1. The strategy is then to use the approximation to $\widetilde{x}^{(m)}$, which we denote by $\widetilde{x}^{(m),\text{Block}}$, instead of $\widetilde{x}^{(m)}$. In the following we assume the matrices H and R to be sparse.

The complete block updating procedure we propose is summarised in Algorithm 2. In the following we motivate and explain the various steps, and define the notation used.

Our construction of $\widetilde{x}^{(m),\text{Block}}$ is general, but to motivate our construction we consider a situation where the elements of the state vector are related to nodes in a two-dimensional (2D) lattice of nodes, with r rows and s columns say, and where the correlation between two elements of the state vector decay with distance between the corresponding nodes. We assume the nodes in the lattice to be numbered in the lexicographical order, so node (k, ℓ) in the lattice is related to the value of element number $(k - 1) \cdot s + \ell$ of the state vector. We use $\mathcal{S} = \{(k, \ell) : k = 1, \ldots, r, l = 1, \ldots, s\}$ to denote the set of all indices of the elements in the state vector, where we for the 2D lattice example have $n_x = rs$.

The first step in the construction of $\widetilde{x}^{(m),\text{Block}}$ is to define a partition $C_1, \ldots, C_{\mathcal{B}}$ of $\mathcal{S}$. The sets $C_b, b = 1, \ldots, \mathcal{B}$ should be chosen so that elements of the state vector corresponding to elements in the same set C_b are highly correlated. In the

Fig. 3 Example of how the sets C_b, D_b and E_b can be chosen when the elements of x are related to the nodes of a 2D lattice. The dots denote nodes. In **a** the nodes are partitioned in $\mathcal{B}$ different blocks, while **b** shows an example of how D_b and E_b might look like
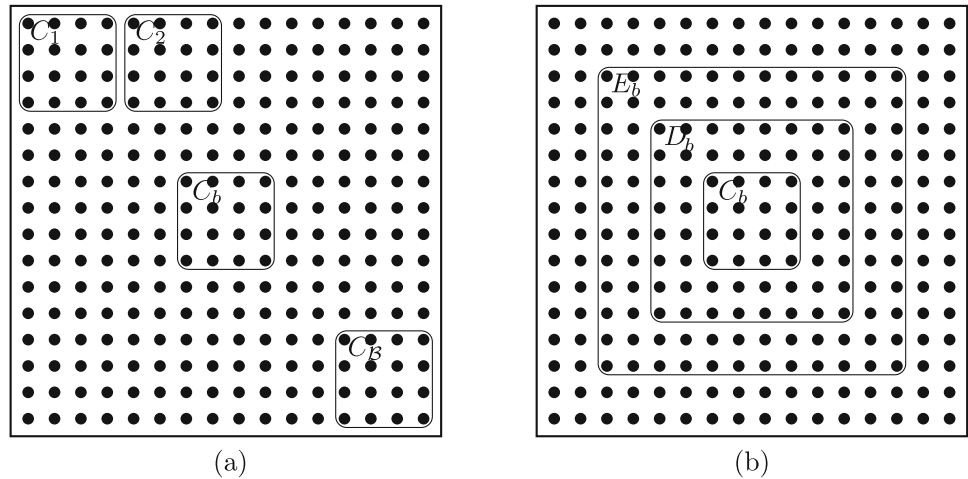
2D lattice example the most natural choice would be to let each C_b be a block of consecutive nodes in the lattice, for example a block of $r_C \times s_C$ nodes. Such a choice of the partition is visualised in Fig. 3a. One should note that a similar partition of $\mathcal{S}$ is also used in the domain localisation version of EnKF (Janjic et al. 2011). However, the motivation for the partition in domain localisation is mainly to eliminate or at least reduce the effect of spurious correlations (Haugen and Evensen 2002), whereas in our setup the motivation is merely to reduce the computational complexity of the updating procedure. In particular, if the dimension of the state vector, n_x, is sufficiently small we recommend not to use the block updating procedure at all, one should then rather use the procedure summarised in Sect. 4.

When forming $\widetilde{x}^{(m)}$ from $x^{(m)}$ in Step 7 of Algorithm 1 one would intuitively expect that element number k of $x^{(m)}$ has a negligible effect on element ℓ of $\widetilde{x}^{(m)}$ whenever the correlation between elements k and ℓ of the state vector is sufficiently weak. In numerical experiments where the dimension of the state vector is sufficiently small so that we can use the procedure in Sect. 3.3, this intuition has already been confirmed. This motivates our first approximation, which is to not allow an element of $x^{(m)}$ to influence an element of $\widetilde{x}^{(m),\text{Block}}$ when the two elements of the state vector have a very low correlation. More formally, for each $b = 1, \ldots, \mathcal{B}$ we define a set of nodes D_b so that $C_b \subseteq D_b$, where D_b should be chosen so that in the state vector, elements $j \in C_b$ are only weakly correlated with elements $i \in \mathcal{S} \backslash D_b$. In the approximation we only allow an element $\ell \in C_b$ of $\widetilde{x}^{(m),\text{Block}}$ to be a function of element k of $x^{(m)}$ if $k \in D_b$. In the 2D lattice example it is natural to define D_b by expanding the C_b block of nodes with u, say, nodes in each direction, see the illustration in Fig. 3b where $u = 2$ is used.

To decide how $\widetilde{x}_{\ell}^{(m),\text{Block}}$, $\ell \in C_b$ is given from $x_k^{(m)}$, $k \in D_b$, the most natural procedure would be the following. Start with the assumed joint Gaussian distribution for the state vector x and the observation vector y,

$$f_{x,y|\theta}(x, y|\theta) = \mathcal{N}(x; \mu, Q) \cdot \mathcal{N}(y; Hx, R). \tag{45}$$

Then we could have marginalised out all elements of the state vector that are not in D_b,

$$f_{x^{D_b}, y|\theta}(x^{D_b}, y|\theta) = \int f_{x,y|\theta}(x, y|\theta) dx^{-D_b}, \tag{46}$$

where we use the notation introduced in Sect. 2.2 and let x^{D_b} and x^{-D_b} denote vectors of the elements of x related to the sets D_b and $\mathcal{S} \backslash D_b$, respectively. From this, one could find the marginal distribution for the elements of the state vector x related to the block D_b, i.e. $f_{x^{D_b}|\theta}(x^{D_b}|\theta)$, and the conditional distribution for the observation vector y given the elements of the state vector related to D_b, $f_{y|x^{D_b},\theta}(y|x^{x_b},\theta)$. As $f_{x^{D_b}|\theta}(x^{D_b}|\theta)$ becomes a Gaussian distribution, and $f_{y|x^{D_b},\theta}(y|x^{x_b},\theta)$ becomes a Gaussian distribution where the mean vector that is a linear function of x^{D_b} and the precision matrix is constant, one could use the procedure specified in Sect. 3.3 to find the optimal update of the sub-vector $x^{(m),D_b}$, $\widetilde{x}^{(m),D_b}$ say. Finally, we could for each $\ell \in C_b$ have set $\widetilde{x}_{\ell}^{(m),\text{Block}}$ equal to the the value of the corresponding element in $\widetilde{x}^{(m),D_b}$. However, such a procedure is not computationally feasible when the dimension of the state vector is large, and for two reasons. First, the marginalisation in (45) is computationally expensive when dependence is represented by precision matrices. Second, when the dimension of the state vector is large the dimension of the observation vector y is typically also large, which makes such a marginalisation process even more computationally expensive. We therefore do one more approximation before following the procedure described above. Instead of starting out with (45), we start with a corresponding conditional distribution, where we condition on the value of elements of the state vector x and observation vector y that are only weakly correlated with the elements in x^{D_b}. The motivation for such a conditioning is twofold. First, to condition on elements that are only weakly correlated with the elements

in x^{D_b} should not change the distribution of x^{D_b} much. Second, when representing dependence by precision matrices the computation of conditional distributions is computationally very efficient. In the following, we define this last approximation more formally and discuss related computational aspects.

For each $b = 1, \ldots \mathcal{B}$, we define a set of nodes E_b such that $D_b \subseteq E_b$. The set E_b should be chosen so that the nodes in D_b are weakly correlated with the nodes that are not in E_b. For the 2D lattice example it is reasonable to define E_b by expanding the D_b block with, say, v nodes in each direction. Figure 3b displays an example of E_b where $v = 2$. Moreover, we let a set J_b contain the indices of the elements in y that by the likelihood are linked to one or more elements in x^{E_b}, i.e.

$$J_b = \{k \in \mathcal{S} : H^{k,\ell} \neq 0 \text{ for at least one } (k, \ell),$$
$$k \in \{1, \ldots, n_y\}, \ell \in E_b\}. \tag{47}$$

Starting from (45) we now find the corresponding conditional distribution when conditioning on x^k being equal to its mean value μ^k for all $k \notin E_b$, and conditioning on y_k being equal to its mean value $(Hx)^k$ for all $k \notin J_b$. The resulting conditional distribution we denote by

$$g\left(x^{E_b}, y^{J_b}|\theta\right) = f\left(x^{E_b}, y^{J_b}|\theta, x^{-E_b} = \mu^{-E_b},\right.$$
$$\left. y^{-J_b} = (Hx)^{-J_b}\right). \tag{48}$$

This (conditional) distribution is also multivariate Gaussian, and the mean vector and the precision matrix can be found by using the expressions in Sect. 2.2. One should note that the conditional precision matrix is immediately given, simply by clipping out the parts of the unconditional precision matrix that belong to x^{E_b} and y^{J_b}. The formula for the conditional mean includes a matrix inversion, where the size of the matrix that needs to be inverted equals the sum of the dimensions of the vectors x^{E_b} and y^{J_b}. The dimension of the matrix is thereby not problematically large. Moreover, one should note that in practice the conditional mean is computed using the techniques discussed in Sect. 2.1, thereby avoiding the matrix inversion. From $g(x^{E_b}, y^{J_b}|\theta)$ we marginalise over $x^{E_b \setminus D_b}$ to get $g(x^{D_b}, y^{J_b}|\theta)$, which is also Gaussian and where the mean vector and the precision matrix can be found as discussed in Sect. 2.2. The $g(x^{D_b}, y^{J_b}|\theta)$ should be thought of as a substitute for the $f(x^{D_b}, y^{J_b}|\theta)$ defined in (46). So following what we ideally would have liked to do with $f(x^{D_b}, y^{J_b}|\theta)$ we use $g(x^{D_b}, y^{J_b}|\theta)$ to form the marginal distribution for x^{D_b}, $g(x^{D_b}|\theta)$, and the conditional distribution for y^{J_b} given x^{D_b}, $g(y^{J_b}|\theta, x^{D_b})$. These two distributions are both Gaussian and can be found using the expressions in Sect. 2.2. The $g(x^{D_b}|\theta)$ should be thought of as a prior for x^{D_b}, and the $g(y^{J_b}|\theta, x^{D_b})$ represents the likelihood for y^{J_b} given x^{D_b}. In the following we let $\widetilde{\mu}^b$ and $\widetilde{Q}^b$ denote the resulting mean vector and precision matrix in the prior $g(x^{D_b}|\theta)$, respectively. From (8) we get that the Gaussian likelihood $g(y^{J_b}|\theta, x^{D_b})$ has a conditional mean in the form $\mathrm{E}[y^{J_b}|x^{D_b}, \theta] = \widetilde{a}^b + \widetilde{H}^b x^{D_b}$, where $\widetilde{a}^b$ is a column vector of size $|J_b|$ and $\widetilde{H}^b$ is a $|J_b| \times |D_b|$ matrix. The precision matrix of the likelihood $g(y^{J_b}|\theta, x^{D_b})$ we in the following denote by $\widetilde{R}^b$. Noting that observing y^{J_b} is equivalent to observing $y^{J_b} - \widetilde{a}^b$, we thereby have a Bayesian model corresponding to what we discussed in Sect. 3.1, except that now x is replaced with x^{D_b}, the ensemble element $x^{(m)}$ is similarly replaced with the sub-vector $x^{(m),D_b}$, and y, μ, Q, H and R are replaced with $y^{J_b} - \widetilde{a}^b, \widetilde{\mu}^b, \widetilde{Q}^b, \widetilde{H}^b$ and $\widetilde{R}^b$, respectively. To update $x^{(m),D_b}$ we can thereby use Steps 2 to 7 in Algorithm 1 to find the optimal update of $x^{(m),D_b}$, $\widetilde{x}^{(m),D_b}$ say. Finally, for each $\ell \in C_b$, we set $\widetilde{x}_\ell^{(m),\text{Block}} = \widetilde{x}_{\ell_b}^{(m),D_b}$, where ℓ_b is the index of the element in $\widetilde{x}^{(m),D_b}$ that corresponds to element ℓ in $\widetilde{x}^{(m),\text{Block}}$.

6 Simulation example

In the following we present a numerical experiment with the model-based EnKF using the proposed block update approximations. The purpose of the experiment is both to study the computational speedup we obtain by using the block updating, and to evaluate empirically the quality of approximation introduced by adopting the block updating procedure. To do this we do a simulation study where we run the model-based EnKF with block updates and the exact model-based EnKF on the same observed values, and compare the run time and filter outputs. As our focus is on the effect of the approximation introduced by using the block update, we use the Gaussian POMM prior distribution introduced in Sect. 4.1 for both filter variants and we also use the exact same observed data in both cases. We first define and study the results for a linear example in Sect. 6.1 and thereafter discuss a non-linear example in Sect. 6.2.

6.1 Linear example

In the simulation experiment we first generate a series of reference states $\{x_t\}_{t=1}^T$ and corresponding observed values $\{y_t\}_{t=1}^T$. The series of reference states we consider as the unknown true values of the latent x_t process and we use the corresponding generated observed values in the two versions of EnKF. We then compare the computational demands and the output of the two filtering procedures. Lastly, we compare the output of the block update procedure to the Kalman filter solution.

Fig. 4 Linear example: Reference states at times **a** $t = 1$ and **b** $t = 5$, and corresponding generated observed values at times **c** $t = 1$ and **d** $t = 5$

(a)

(b)

(c)

(d)
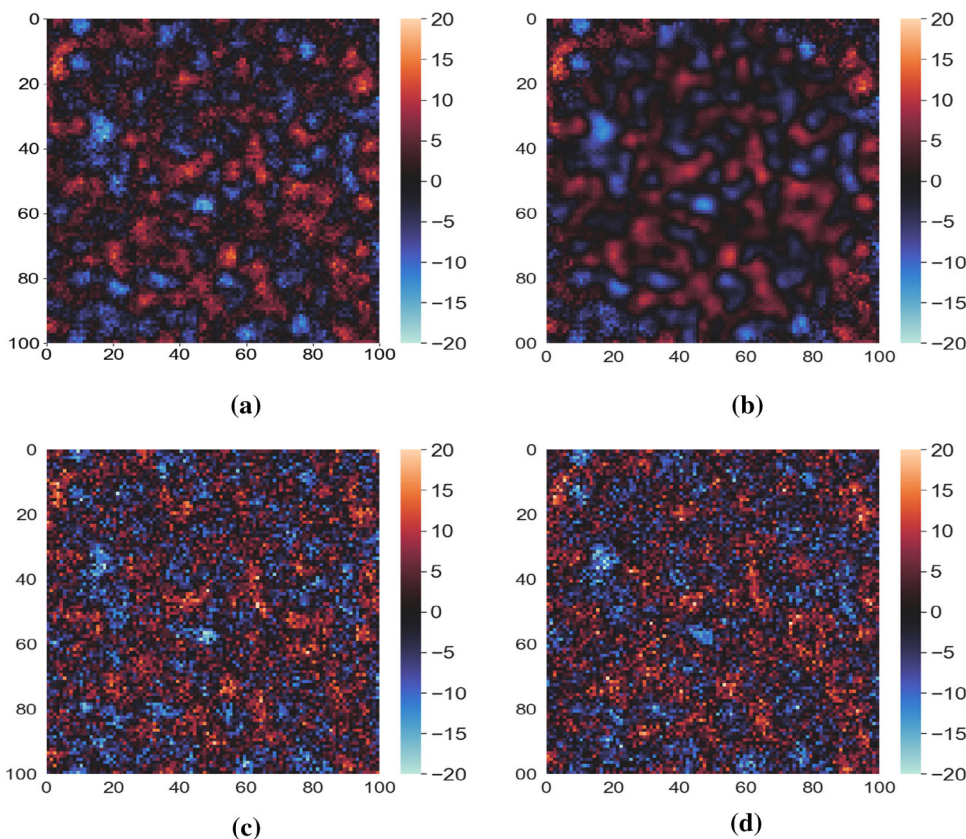
6.1.1 Reference time series and observations

To generate the series of reference states we adopt a two dimensional variant of the one dimensional setup used in Omre and Myrseth (2010). So for each time step $t = 1, \ldots, T$ the reference state x_t represents the values in a two dimensional $s \times s$ lattice. As described in Sect. 5 we number the nodes in the lexicographical order. The reference state at time $t = 1$ we generate as a moving average of a white noise field. More precisely, we first generate independent and standard normal variates $z^{(k,\ell)}$ for each node (k, ℓ), and to avoid boundary effects we do this for an extended lattice. The reference state at node (k, ℓ) and time $t = 1$ is then defined by

$$x_1^{(k-1)\cdot s+\ell} = \sqrt{\frac{20}{|\Gamma_{3,(k,\ell)}|}} \sum_{(i,j)\in\Gamma_{3,(k,\ell)}} z^{(i,j)}, \qquad (49)$$

where $\Gamma_{r,(k,\ell)}$ is the set of nodes in the extended lattice, that are less than or equal to a distance r from node (k, ℓ), and $|\Gamma_{r,(k,\ell)}|$ is the number of elements in that set. One should note that the factor $\sqrt{20/|\Gamma_{3,(k,\ell)}|}$ gives that the variance of each generated $x_1^{(k-1)\cdot s+\ell}$ is 20. Figure 4a shows a generated x_1 with $s = 100$, which is used in the numerical experiments discussed in Sects. 6.1.4 and 6.1.5.

Given the reference state at time $t = 1$, corresponding reference states at later time steps are generated sequentially. For $t = 2, \ldots, T$ the reference state at time $t - 1$ is used to generate the reference state at time t by performing a moving average operation on nodes that are inside an annulus defined by circles centred at the middle of the lattice. For $t = 2$ the radius of the inner circle defining the annulus is zero, and as t increases the annulus is gradually moved further away from the centre. More precisely, to generate x_t from x_{t-1} we define the annulus by the two radii $r_1 = \max\{0, \lfloor(\frac{s}{2} - 1)\frac{1}{T-1}(t - \frac{5}{2})\rfloor\}$ and $r_2 = \lfloor(\frac{s}{2} - 1)\frac{t-1}{T-1}\rfloor$, where $\lfloor v \rfloor$ denotes the largest integer less than or equal to v. For all nodes (k, ℓ) inside the annulus we define

$$x_t^{(k-1)\cdot s+\ell} = \frac{1}{|\Gamma_{1,(k,\ell)}|} \sum_{(i,j)\in\Gamma_{1,(k,\ell)}} x_{t-1}^{(i-1)\cdot s+j}. \qquad (50)$$

For nodes (k, ℓ) that are not inside the specified annulus the values are unchanged, i.e. for such nodes we have $x_t^{(k-1)\cdot s+\ell} = x_{t-1}^{(k-1)\cdot s+\ell}$. The reference solution at time $T = 5$ corresponding to the x_1 given in Fig. 4a is shown in Fig. 4b. By comparing the two we can see the effect of the smoothing operations.

For each time step $t = 1, \ldots, T$ we generate an observed value associated to each node in the lattice, so $n_y = n_x$. We generate the observed values at time t, y_t, by blurring x_t and
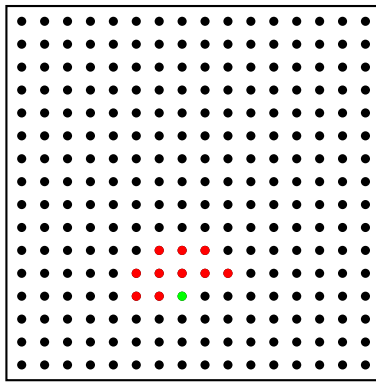
Fig. 5 The sequential neighbourhood used in the numerical examples. The red dots denote the sequential neighbours of the green node. (Color figure online)

adding independent Gaussian noise in each node. More precisely, the observed value in node (k, ℓ) at time t is generated as

$$y_t^{(k-1) \cdot s + \ell} = \frac{1}{|\Gamma_{\sqrt{2}, (k, \ell)} \cap \mathcal{S}|} \sum_{(i,j) \in \Gamma_{\sqrt{2}, (k, \ell)} \cap \mathcal{S}} x_t^{(i-1) \cdot s + j} + w^{(k-1) \cdot s + \ell}, \tag{51}$$

where $w^{(k-1) \cdot s + \ell}$ is a zero-mean normal variate with variance 20. Figure 4c and d show the generated observations corresponding to the x_1 and x_5 in Fig. 4a and b, respectively. These observations are used in the numerical experiments discussed in Sects. 6.1.4 and 6.1.5.

6.1.2 Assumed model and algorithmic parameters

In the numerical experiments with EnKFs we use the assumed model defined in Sect. 3.1. For nodes sufficiently far away from the lattice borders we let the sequential neighbourhood consist of ten nodes as shown in Fig. 5.

This set of sequential neighbours should be sufficient to be able to represent a large variety of spatial correlation structures at the same time as it induces sparsity in the resulting precision matrix. For nodes close to the lattice borders we reduce the number of sequential neighbours to consist of only the subset of the ten nodes shown in Fig. 5 that are within the lattice.

As priors for η and ϕ we use the parametric forms specified in Sect. 4.1. We want these priors to be vague so that the resulting posterior distributions are mainly influenced by the prior ensemble. We let the prior for ϕ be improper by setting $\alpha^k = 0$ and $\beta^k = \infty$ for all k. In the prior for $\eta_k | \phi_k$ we set $\zeta^k = 0$ and $\Sigma_{\eta^k} = 100 \cdot I_{|\Lambda_k|+1}$ for all k.

In preliminary simulation experiments we found that the Gibbs sampler for the sampling of θ converged very rapidly, consistent with our discussion in Sect. 4.2. In the numeri-

cal experiments we therefore only used five iterations of the Gibbs sampler. When using the block update procedure we let each C_b consist of a block of 20×20 nodes. To define the corresponding sets D_b and E_b we follow the procedure outlined in Sect. 5 with $u = v = 5$.

6.1.3 Comparison of computational demands

The main objective of the block update procedure is to provide a computationally efficient approximation to the optimal model-based EnKF procedure defined in Sect. 3. In this section we compare the computational demands of the two updating procedures as a function of the number of nodes in the lattice.

For an implementation in Python, we run the two EnKF updating procedures discussed above with $\mathcal{M} = 25$ ensemble elements for $T = 5$ time steps for different lattice sizes and monitor the computing time used in each case. The results are shown in Fig. 6. The observed computing times are shown as dots in the plot, in blue for the optimal model-based EnKF procedure and in orange for the block updating procedure. The lines between the dots are just included to make it easier to read the plot. As we should expect from the discussion in Sect. 5 we observe that the speedup resulting from adopting the block update increases quickly with the lattice size.

6.1.4 Approximation error by the block update

When adopting the block update in an EnKF procedure we will clearly get an approximation error in each update. When assessing this approximation error it is important to realise that the approximation error in one update may influence the behaviour of the filter also in later iterations. First, the effect of the approximation error emerged in one update may change when later iterations of the filter is run, the error may increase or decrease when run through later updates. Second, since the EnKF we are using is a stochastic filter, with the stochasticity lying in the sampling of θ, even a small change after one update may have a large impact on the resulting values in later iterations. In particular it is important to note that even if the approximation error in one update only marginally change the distribution of the values generated in later iterations, it may have a large impact on the actually sampled values. In this section our focus is first to assess the approximation error in one update when using the block update, whereas in the second part of this section we concentrate on the accumulated effect of the approximations after several iterations.

Adopting the reference time series and the data described in Sect. 6.1.1 and the assumed model and algorithmic parameters defined in Sect. 6.1.2 we isolate the approximation error in one update by the following procedure. We first run the optimal model-based EnKF procedure for all $T = 5$ steps.
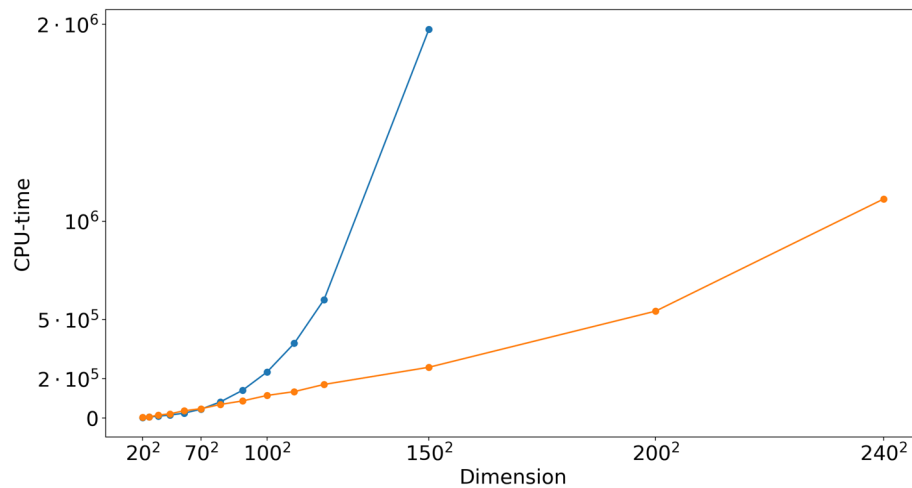
Fig. 6 Linear example: Computing time used for running each of the two model-based EnKF procedures with $\mathcal{M} = 25$ ensemble elements for $T = 5$ time steps as a function of the number of nodes $n_x = s^2$ in the lattice. Computing time for the optimal model-based EnKF procedure is shown in blue, and corresponding computing time for the block updating procedure is shown in orange. (Color figure online)
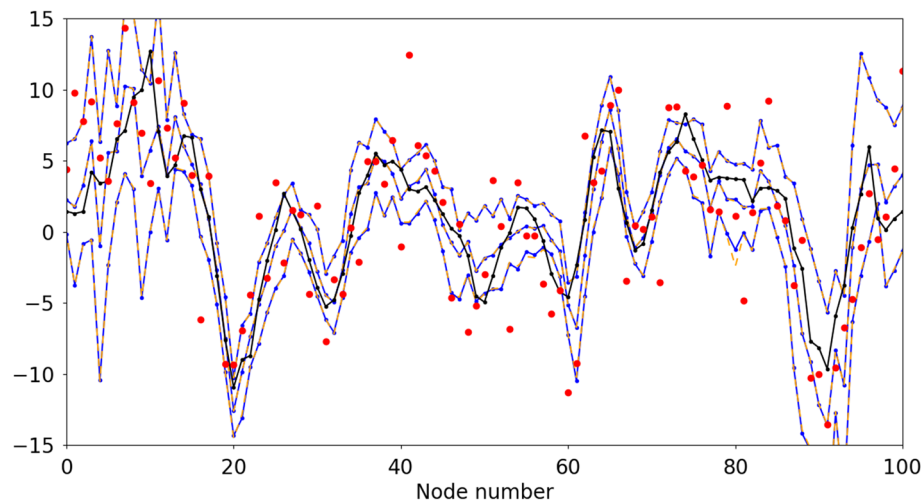

Fig. 7 Linear example: Assessment of the approximation error by using the block update at step $t = 4$. The figure shows values for nodes $(50, \ell)$, $\ell = 1, \ldots, 100$. The values of the reference are shown in black and the observations at $t = 4$ are shown in red. The ensemble averages and bounds for the empirical 90% credibility intervals when using the optimal model-based EnKF are shown in blue. The corresponding ensemble averages and bounds for the empirical 90% credibility intervals when using the block update are shown in dashed orange. (Color figure online)

Thereafter, for each $t = 1, \ldots, T$, we start with the prior ensemble generated for that time step in the optimal model-based EnKF and perform one step with the block update. One should note that since we are using the same prior ensemble for the block update as for the optimal model-based EnKF the generated parameters θ are identical for both updates, so the resulting difference in the updated values are only due to the difference in the B matrix used in the two procedures.

Figure 7 shows results for the nodes $(50, \ell)$, $\ell = 1, \ldots, 100$ at time step $t = 4$. The ensemble averages and bounds of the empirical 90% credibility intervals for the optimal model-based EnKF procedure are shown in blue, and the cor-

responding values when using the block update procedure for time $t = 4$ are shown in dashed orange. For comparison the reference is shown in black. One can observe that for most nodes the mean value and interval bounds when using the block update are visually indistinguishable from the corresponding values when using the block update. The largest difference can be observed for the lower bound of the credibility interval for $\ell = 80$. Remembering that we are using blocks of 20×20 nodes in the block update it should not come as a surprise that the largest errors are for values of ℓ that are multiples of 20.

Fig. 8 Linear example: Difference between the averages of the ensembles when using the optimal model-based EnKF and when using the block update. Results are shown for each $t = 1, \ldots, 5$


$t = 1$

$t = 2$

$t = 3$

$t = 4$

$t = 5$
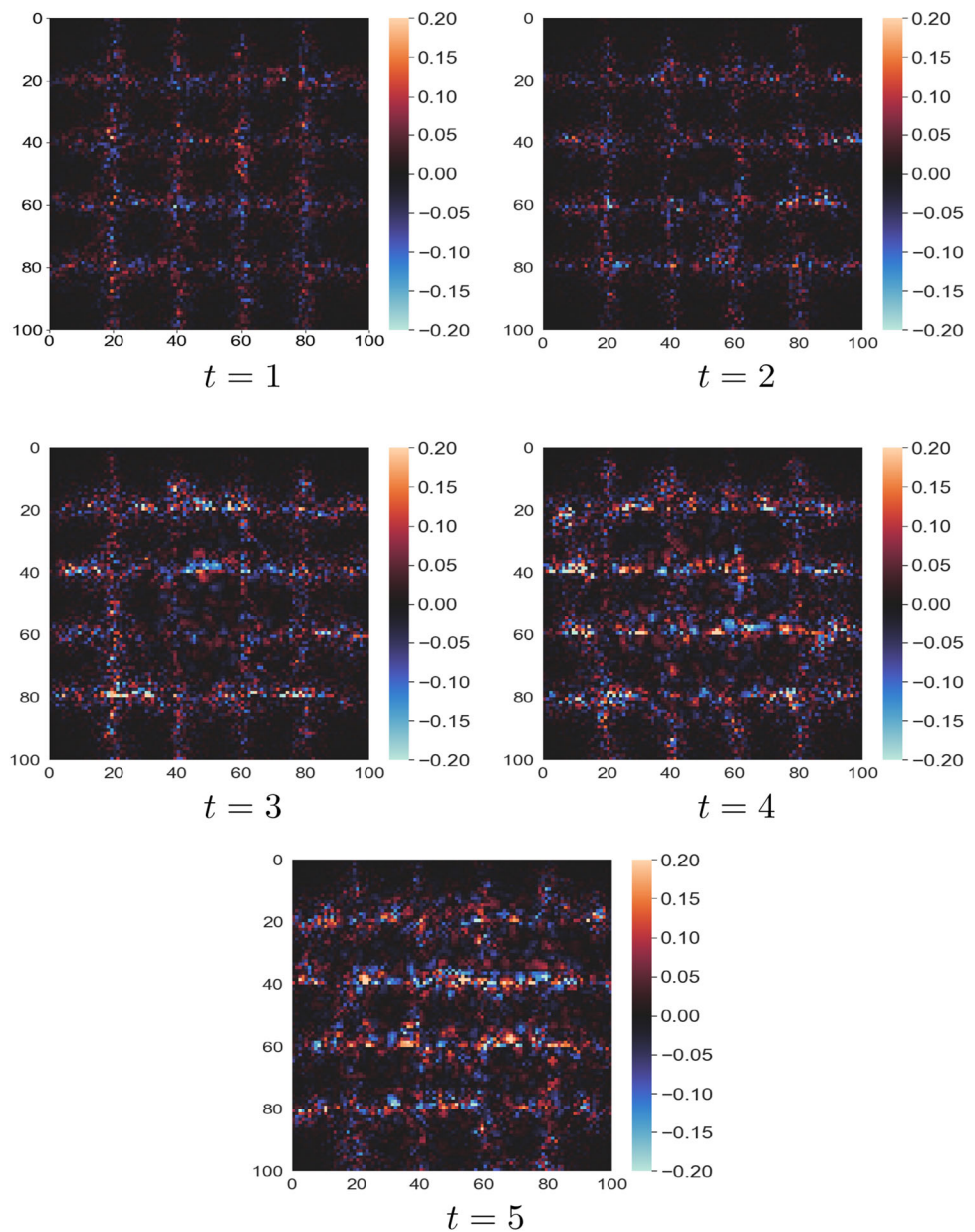
To study the approximation errors in more detail and in particular how the block structure used in the block update influence the approximation, we in Fig. 8 show, for each $t = 1, \ldots, 5$, the difference between the averages of the ensembles when using the optimal model-based EnKF and when using the block update procedure. As one should expect we see that the largest differences occur along the borders of the 20×20 blocks used in the block update. We can, however, also observe that the magnitude of the errors are all small compared with the spatial variability of the underlying x_t process, which explains why the approximation errors are almost invisible in Fig. 7.

We then turn our focus to the accumulated effect of the approximation error over several iterations. To do this we again adopt the reference time series and the data described in Sect. 6.1.1, and the assumed model and algorithmic parameters defined in Sect. 6.1.2. Based on this we run each of the optimal model-based EnKF and the block update procedures several times, each time with a different initial ensemble and using different random numbers in the updates. For two runs with each of the two filters, Fig. 9 shows in green the values of all the $\mathcal{M} = 25$ ensemble elements in nodes $(50, \ell)$, $\ell = 1, \ldots, 100$ at time $T = 5$. The reference state at $T = 5$ is shown in black and the observations at time $T = 5$ are again shown as red dots. Figure 9a and c show results for the optimal model-based EnKF, whereas Fig. 9b and d show results for the proposed block update procedure. Since all four runs are based on the same observations they

Fig. 9 Linear example: The reference solution (black) and the observations (red) along one cross section of the grid at time $T = 5$. The green lines are the ensemble elements along the same cross section. **a** and **c** show two simulations from the optimal update procedure, while **b** and **d** display two simulations from the block update. (Color figure online)
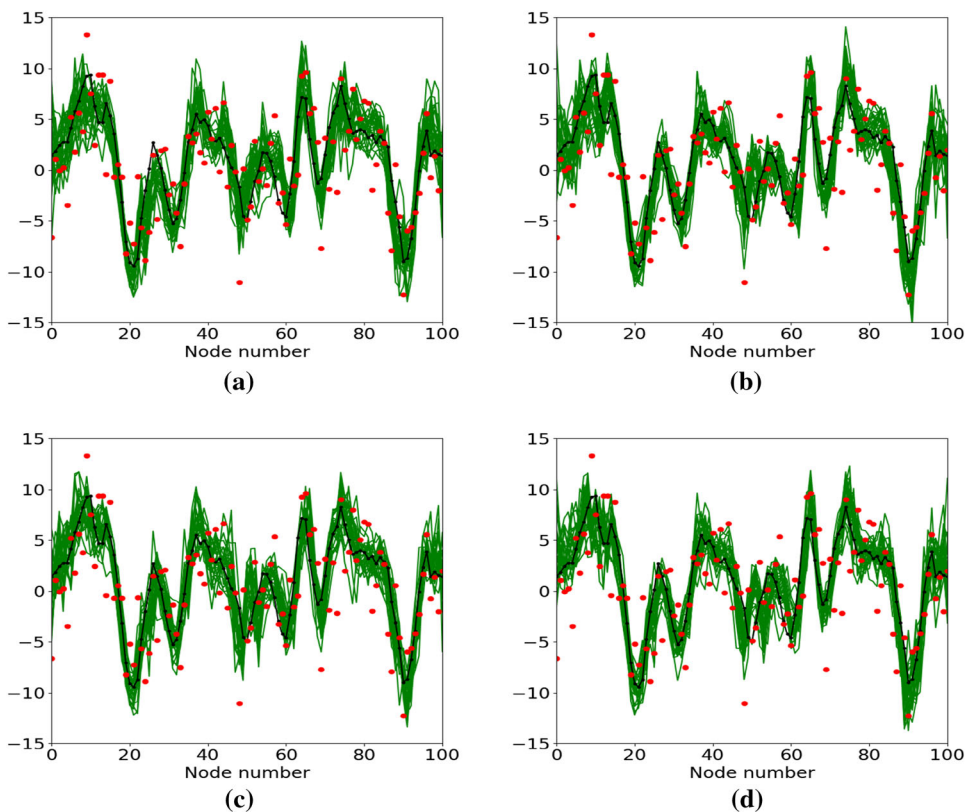
obviously have many similarities. However, all the ensembles are different. Ensembles updated with the same updating procedure differ because of Monte Carlo variability, i.e. because they used different initial ensembles and different random numbers in the updates. When comparing an ensemble updated with the optimal model-based EnKF procedure with an ensemble updated with the block update, they differ both because of Monte Carlo variability and because of the accumulated effect of the approximation errors up to time T. In Fig. 9, however, the differences between the ensembles in a and b and between the ensembles in c and d seem similar to the differences between the ensembles in a and c and between the ensembles in b and d. So this indicates that the accumulated effect of the approximation error is small compared to the Monte Carlo variability.

To compare the magnitude of the accumulated approximation error with the Monte Carlo variability in more detail we adopt the two-sample Kolmogorov–Smirnov statistic to measure the difference between the values of two ensembles in one node. More precisely, letting $\widehat{F}_1(x; k, \ell)$ and $\widehat{F}_2(x; k, \ell)$ denote the empirical distribution functions for the values in node (k, ℓ) in two ensembles we use

$$D(k, \ell) = \max_{x} |\widehat{F}_1(x; k, \ell) - \widehat{F}_2(x; k, \ell)| \tag{52}$$

to measure the difference between the two ensembles at node (k, ℓ). One should note that since we are using $\mathcal{M} = 25$ elements in each ensemble the $D(k, \ell)$ is a discrete variable with possible values $0.04d$; $d = 0, 1, \ldots, 25$. Using three runs based on the optimal model-based EnKF we compute $D(k, \ell)$ for each possible pair of ensembles, for each $t = 1, \ldots, 5$ and for each node in the 100×100 lattice. In Fig. 10 histogram of the resulting 30,000 values for each $t = 1, \ldots, 5$ are shown in blue. Correspondingly we estimate the distribution of $D(k, \ell)$ when one of the ensembles is based on the optimal model-based EnKF and the other is using the block update, and this is shown in red in Fig. 10. Comparing the blue and the red histograms for each $t = 1, \ldots, 5$ in Fig. 10 one may arguably see a tendency of the mass in the red histograms to be slightly moved to the right relative to the corresponding blue histograms. This is as one would expect, but we also observe that the effect of the approximation error is negligible compared to the Monte Carlo variability.

6.1.5 Comparison of the results with the Kalman filter output

Since the forward model in this numerical example is linear and deterministic and the observation model is Gauss-linear, we are able to compare the block update to the Kalman filter solution. Figure 11 displays the mean of the Kalman

Fig. 10 Linear example: Histograms of the two-sample Kolmogorov–Smirnov statistics. The statistics computed from two ensembles updated with the optimal ensemble are displayed in blue, while the statistics computed by two ensembles from different updating procedures are visualised in red. (Color figure online)
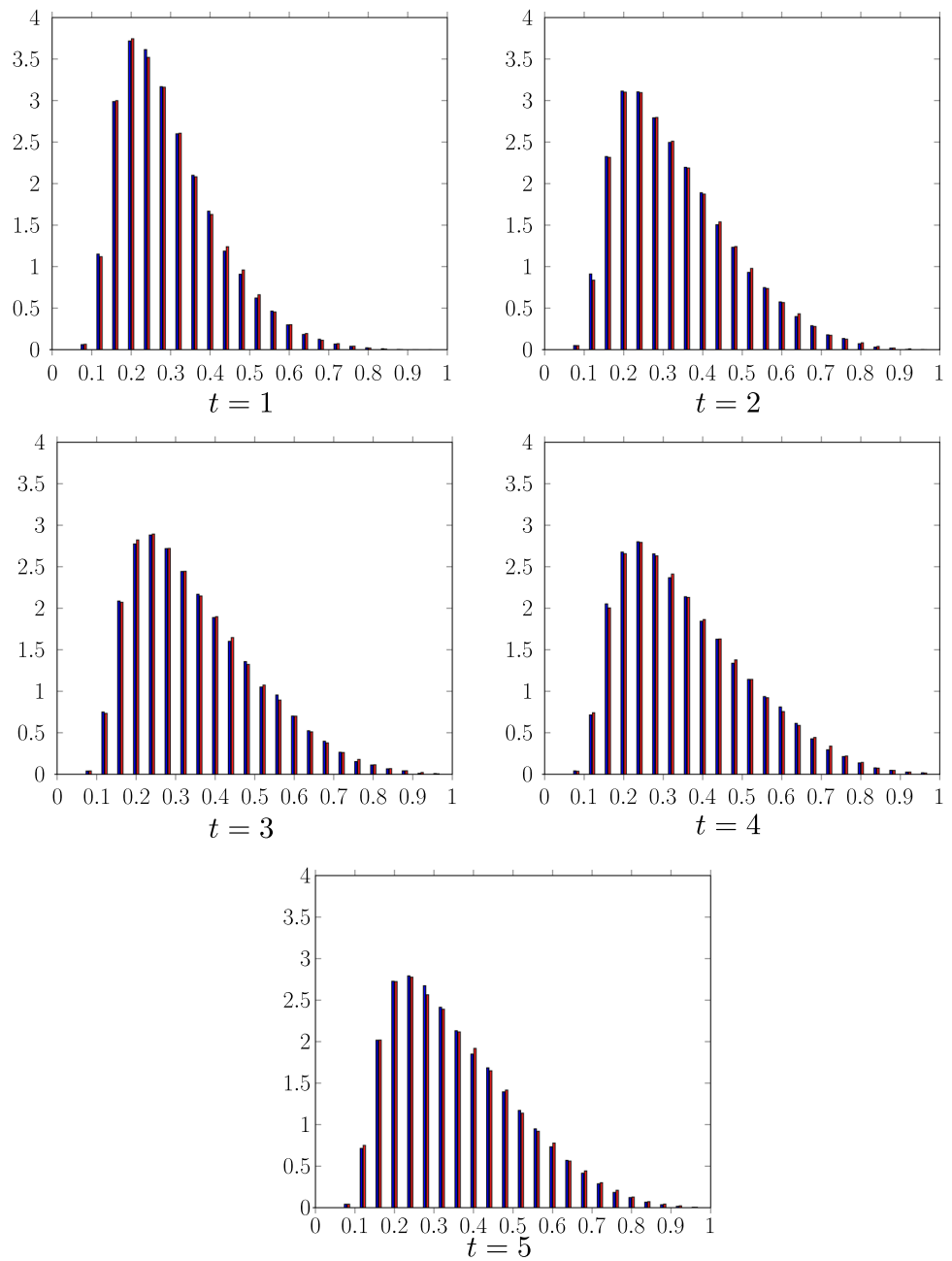
filter solution along with the 90% credibility interval for the nodes $(50, \ell)$, $l = 1, \ldots, 100$ in blue at $t = 4$. The red lines visualise the empirical mean and empirical 90% credibility interval as given by one simulation of the block update. When comparing the two set of curves one should remember that the EnKF output is stochastic, so for another EnKF run a different set of curves would be produced. In general, however, we notice that the mean given by the block update follows the mean of the Kalman filter quite well.

When comparing the credibility intervals we can observe that the intervals provided by the block update appears to be somewhat longer than the credibility intervals provided by the Kalman filter. This is as one should expect. At each step of the block update filter, as for any other ensemble based filter, we represent the credibility and filtering distribution by a rather small set of realisations. So compared to the Kalman filter, where the exact credibility and filtering distributions are represented by mean vectors and covariance matrices, we necessarily loose some information about the the underlying true state vector at each iteration of the block update filter, when representing the distributions by a set of realisations only.

The results for times $t = 1, 2, 3$ and 5 are similar to the results for time $t = 4$, except that the widths of the credibility

intervals decrease with time. The shrinkage of the widths of the credibility intervals is clearly visible for both the Kalman filter and the block update filter, but is stronger for the Kalman filter than for the block update filter. Such a shrinkage is as one should expect when the forward function is deterministic. That the shrinkage is strongest for the Kalman filter is reasonable since in each iteration of block update filter we loose some information about the underlying true state when just representing the distributions by ensembles.

6.2 Non-linear example

In the following we consider a non-linear example. We first describe how the reference solution and observations are generated, before we proceed to compare the results provided by the block update filter with the results of the optimal model-based EnKF.

6.2.1 Reference time series and observations

The reference state for the initial time step, x_1, is generated in the exact same manner as for the linear case. The forward model used is inspired by the non-linear forward function in Omre and Myrseth ([2010]). The forward function is acting on each element of the state vector separately. The reference state at time $t > 1$ for node (k, ℓ) is defined by

$$x_t^{(k-1) \cdot s + \ell} = x_{t-1}^{(k-1) \cdot s + \ell} + 0.5 \cdot \arctan\left(\frac{x_{t-1}^{(k-1) \cdot s + \ell}}{2}\right). \quad (53)$$

The term $0.5 \cdot \arctan(x/2)$ is chosen to make the forward function clearly non-linear over the interval in which the elements of the state vector vary. The term is displayed in Fig. 12. The observations are generated in the exact same manner as in the linear example. The assumed model and all the algorithmic parameters are also chosen identical to what we used in the linear example.

6.2.2 Approximation error in block update

As for the linear example, we first consider the approximation error obtained in one update, and thereafter study how the approximation error accumulates over multiple iterations.

To study the effect of the approximation error in one iteration we follow the same procedure as used in Sect. 6.1.4. In Fig. 13 we compare the credibility intervals provided by the optimal update and block update at time $t = 4$, similar to Fig. 7 for the linear case. The credibility interval for the nodes $(50, \ell)$, $\ell = 1, \ldots, 100$ provided by the optimal update is displayed in blue, while the credibility interval provided by the block update is visualised in orange. We notice that the credibility intervals provided by the two update pro-
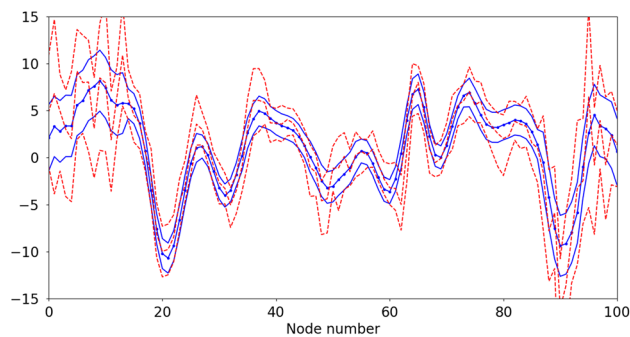

Fig. 11 Linear example: The blue lines display the mean of the Kalman filter solution along with the 90% credibility intervals for one cross section at time $t = 4$. The red lines visualise the empirical mean and empirical 90% credibility interval for the block update for the same cross section at the same time step. (Color figure online)
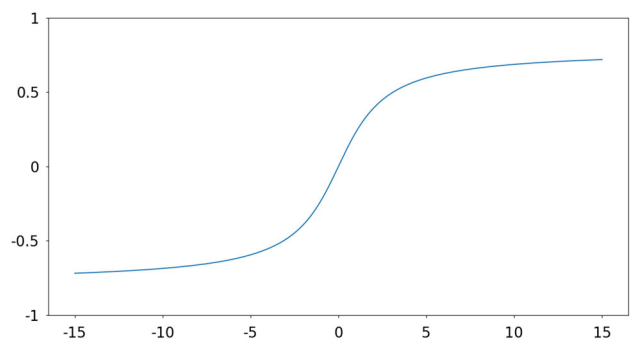

Fig. 12 Non-linear example: The term $0.5 \cdot \arctan(x/2)$ in the forward function of the non-linear example
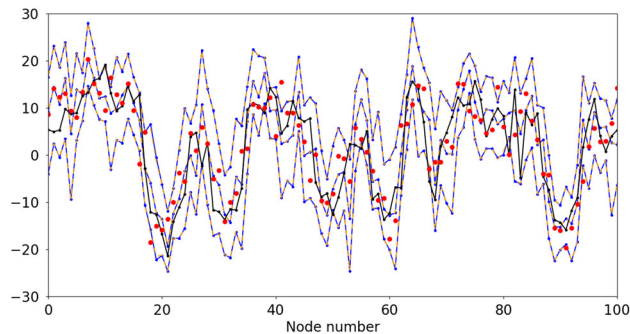

Fig. 13 Non-linear example: Assessment of the approximation error by using the block update at step $t = 4$. The figure shows values for nodes $(50, \ell)$, $\ell = 1, \ldots, 100$. The values of the reference are shown in black and the observations at $t = 4$ are shown in red. The ensemble averages and bounds for the empirical 90% credibility intervals when using the optimal model-based EnKF are shown in blue. The corresponding ensemble averages and bounds for the empirical 90% credibility intervals when using the block update are shown in dashed orange. (Color figure online)

cedures are practically speaking visually indistinguishable, similar to what we observe for the linear case. We have also studied the spatial variability of the approximation errors by inspecting plots of the difference of the means of the ensem-

Fig. 14 Non-linear example: Histograms of the two-sample Kolmogorov–Smirnov statistics. The statistics computed from two ensembles updated with the optimal ensemble are displayed in blue, while the statistics computed by two ensembles from different updating procedures are visualised in red. (Color figure online)
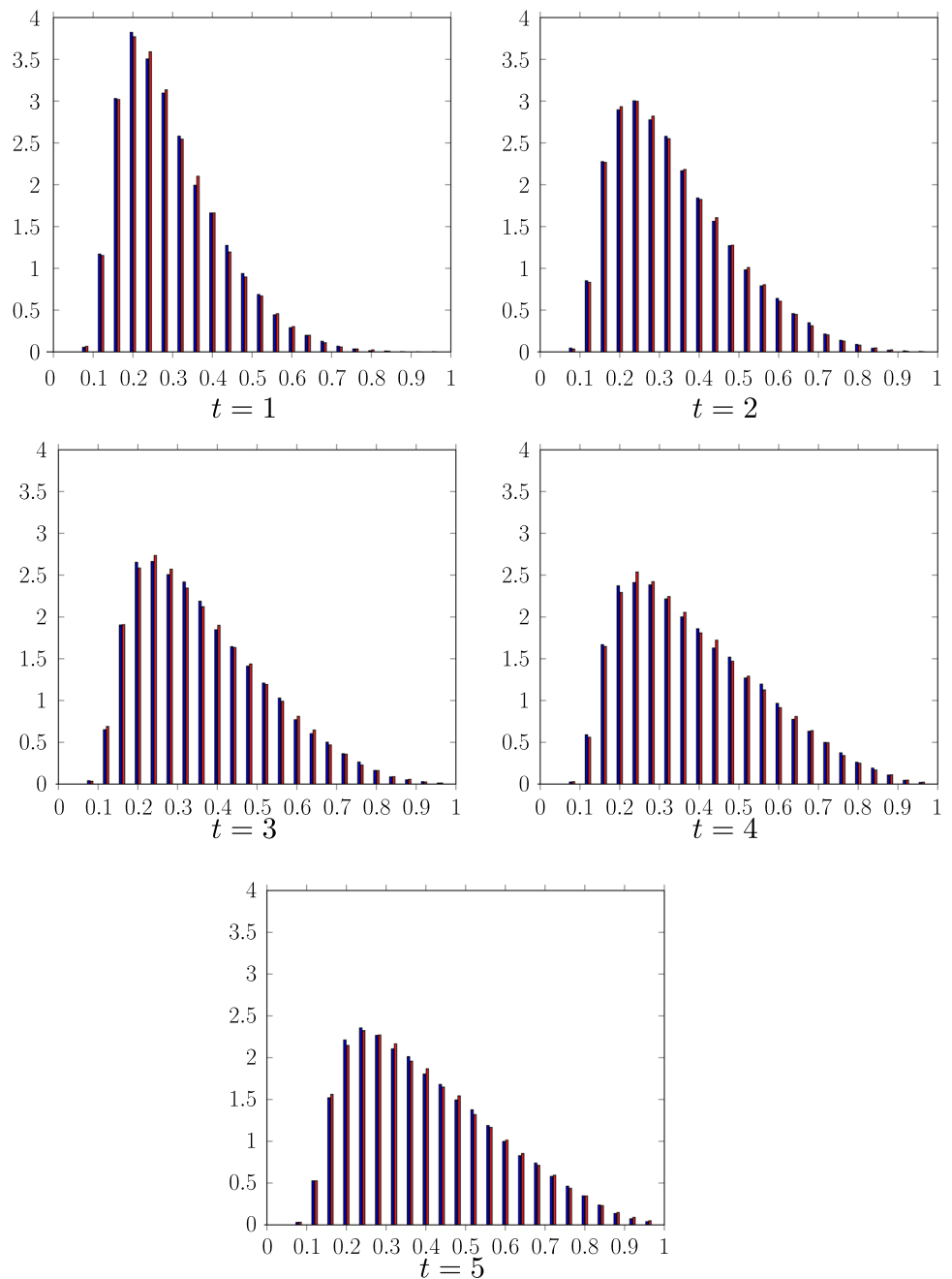
bles provided by the two update procedures, similar to Fig. 8. The results for the non-linear example are similar to what we observed in the linear case. The largest approximation errors are again concentrated around the block borders, and the relative approximation errors are similar to what we had with the linear forward function.

To study how the approximation errors accumulated over several iterations, we followed the same procedure as in the linear example. When comparing ensembles resulting from the two filters, corresponding to what is done in Fig. 9, the accumulated effect of the approximation error again seem to be small compared to the Monte Carlo variability. Figure 14 displays the histograms of the Kolmogorov–Smirnov statistics for $t = 1, \ldots, 5$, analogous to Fig. 10 for the linear case. We arguably can observe that the tendency of the mass in the red histograms to be moved to the right compared to the corresponding blue histograms, is slightly stronger in this non-linear example than in the linear case.

7 Closing remarks

In this paper we propose two changes in the model-based EnKF procedure introduced in Loe and Tjelmeland (2021). Our motivation is to get a procedure that is computationally faster than the original one, so that it is feasible to use it also in situations with high dimensional state vectors. The first change we introduce is to formulate the assumed model in terms of precision matrices instead of covariance matrices, and to adopt a prior for the precision matrix that ensures the sampled precision matrices to be sparse. The second change we propose is to adopt the block update, which allows us to do singular value decompositions of many smaller matrices instead of for one large one.

In a simulation example we have studied both the computational speedup and the associated approximation error resulting when adopting the proposed procedure. The computational speedup is substantial for high dimensional state vectors and this allows the proposed filter to be run on much larger problems than can be done with the original formulation. At the same time the approximation error resulting from using the introduced block updating is negligible compared to the Monte Carlo variability inherent in both the original and the proposed procedures.

In order to further investigate the strengths and weaknesses of the proposed approach, it should be applied on more examples. It is of interest to gain more experience with the proposed procedure both in other simulation examples and in real data situations. In particular it is of interest to experiment more with different sizes for the C_b, D_b and E_b blocks to try to find the best sizes for these when taking both the computational time and the approximation quality into account. As for all EnKFs the proposed procedure is ideal for parallel computation and with an implementation more tailored for parallellisation it should be possible get a code that is running much faster than our more basic implementation.

Declaration

Appendices

A Derivation of $\theta = (\mu, Q)$ from ϕ and η

To find how $\theta = (\mu, Q)$ is given from ϕ and η we restrict the Gaussian densities $f(x|\theta)$ and $f(x|\phi, \eta)$ to be identical. The density $f(x|\theta)$ is given by

$$
\begin{aligned}
f(x|\theta) &\propto \exp\left\{-\frac{1}{2}(x-\mu)^T Q(x-\mu)\right\} \\
&= \exp\left\{-\frac{1}{2}x^T Q x + \mu^T Q x\right\} \\
&= \exp\left\{\sum_{k=1}^{n_x}\left(-\frac{1}{2}Q^{k,k}\right)(x^k)^2 + \sum_{\ell<k}\left(-Q^{k,\ell}\right)x^k x^\ell \right. \\
&\quad \left. + \sum_{k=1}^{n_x}\left(\sum_{\ell=1}^{n_x}\mu^\ell Q^{\ell,k}\right)x^k\right\}.
\end{aligned}
\tag{54}
$$

Using the definition of $f(x|\phi, \eta)$ from Sect. 4.1 we have

$$
\begin{aligned}
f(x|\phi, \eta) &= \prod_{k=1}^{n_x} f\left(x^k | x^{\Lambda_k}, \eta^k, \phi^k\right) \\
&\propto \prod_{k=1}^{n_x} \exp\left\{-\frac{1}{2\phi^k}\left(x^k - (1, (x^{\Lambda_k})^T)\cdot\eta^k\right)^2\right\} \\
&= \exp\left\{-\frac{1}{2}\sum_{k=1}^{n_x}\frac{1}{\phi^k}\left(x^k - \eta^{k,1} - \sum_{\ell=2}^{|\Lambda_k|}\eta^{k,\ell}x^{\Lambda_k(\ell)}\right)^2\right\} \\
&\propto \exp\left\{\sum_{k=1}^{n_x}\left(-\frac{1}{2\phi^k} - \frac{1}{2}\sum_{\ell:k\in\Lambda_\ell}\frac{1}{\phi^\ell}\left(\eta^{\ell,\Lambda_\ell^{-1}(k)+1}\right)^2\right)(x^k)^2 \right. \\
&\quad + \sum_{\ell<k}\left(\frac{1}{\phi^k}\eta^{k,\Lambda_k^{-1}(\ell)+1}\mathbb{I}(\ell\in\Lambda_k)\right. \\
&\quad \left. - \sum_{t:k,\ell\in\Lambda_t}\frac{1}{\phi^t}\eta^{t,\Lambda_t^{-1}(k)+1}\eta^{t,\Lambda_t^{-1}(\ell)+1}\right)x_k x_\ell \\
&\quad \left. + \sum_{k=1}^{n_x}\left(\frac{\eta^{k,1}}{\phi^k} - \sum_{\ell:k\in\Lambda_\ell}\frac{1}{\phi^\ell}\eta^{\ell,1}\eta^{\ell,\Lambda_\ell^{-1}(k)+1}\right)x^k\right\},
\end{aligned}
\tag{55}
$$

where $\Lambda_\ell^{-1}(k)$ is the inverse function of $\Lambda_\ell(k)$, i.e. $\Lambda_\ell^{-1}(k) = t \Leftrightarrow \Lambda_\ell(t) = k$, and $\mathbb{I}(\ell \in \Lambda_k)$ is the indicator function returning one if $\ell \in \Lambda_k$ and zero otherwise. By setting equal corresponding coefficients in front of $(x^k)^2$, $x_k x_\ell$ and x_k in (54) and (55) we can identify the elements of μ and Q. Setting

equal the coefficients in front of $(x^k)^2$ we get the diagonal elements of Q,

$$Q^{k,k} = \frac{1}{\phi^k} + \sum_{\ell:k\in\Lambda_\ell} \frac{1}{\phi^\ell} \left(\eta^{\ell,\Lambda_\ell^{-1}(k)+1} \right)^2. \quad (56)$$

To get the non-diagonal elements we set equal the coefficients in front of $x_k x_\ell$. For $\ell < k$ we have

$$Q_{k,\ell} = Q_{\ell,k} = -\frac{1}{\phi^k} \eta^{k,\Lambda_k^{-1}(\ell)+1} \mathbb{I}(\ell \in \Lambda_k) + \sum_{t:k,\ell\in\Lambda_t} \frac{1}{\phi^t} \eta^{t,\Lambda_t^{-1}(k)+1} \eta^{t,\Lambda_t^{-1}(\ell)+1}. \quad (57)$$

In particular one should note that for $\ell < k$ we get $Q_{k,\ell} = 0$ if $\ell \notin \Lambda_k$ and there is no node t so that $k, \ell \in \Lambda_t$. Thus, if the sequential neighbourhoods are chosen to be sufficiently small, Q will be sparse. For a more detailed discussion see Appendix B. Finally, by equating corresponding coefficients in front of x_k we get

$$\sum_{\ell=1}^{n_x} \mu^\ell Q^{\ell,k} = \frac{\eta^{k,1}}{\phi^k} - \sum_{\ell:k\in\Lambda_\ell} \frac{1}{\phi^\ell} \eta^{\ell,1} \eta^{\ell,\Lambda_\ell^{-1}(k)+1}, \quad (58)$$

which can be used to find the elements of the mean vector μ. Since Q is sparse the strategy discussed in Sect. 2.1 can be used to compute μ efficiently.

B Sparseness of Q

In the following we discuss the relationship between the sequential neighbourhood and the structure of the precision matrix Q. We first identify a general expression for the set of non-zero elements in Q, and thereafter discuss the situation in more detail when the state vector is associated with the nodes in a two-dimensional lattice.

From (56) and (57) we see that element (k, ℓ) of the precision matrix, $Q_{k,\ell}$, may be non-zero only when

$$\ell \in \partial k = \{k\} \cup \Lambda_k \cup \{\ell \in S : \exists t \in S \text{ so that } k, \ell \in \Lambda_t\}, \quad (59)$$

where S is the set of all nodes in the lattice. A perhaps more instructive formulation of the set ∂k is as a union of sequential neighbourhoods,

$$\partial k = \bigcup_{t\in S:k\in\Lambda_t\cup\{t\}} (\Lambda_t \cup \{t\}). \quad (60)$$

Now consider a situation where the state vector is associated with the nodes in a two-dimensional lattice and assume
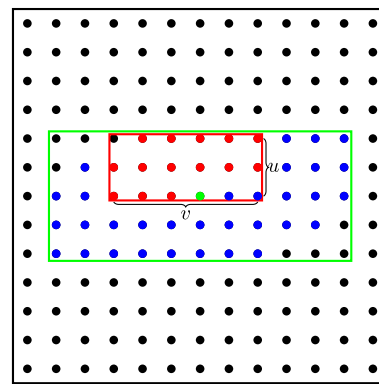

Fig. 15 The green node represents node k, while the red nodes represent Λ_k. The nodes inside the red rectangle visualises Ψ_k. The blue nodes represent the remaining nodes in ∂k, while the nodes inside the green rectangle represent Ω_k. (Color figure online)

all the sequential neighbourhoods to be of the same form, except for nodes close to the lattice boundaries where the sequential neighbourhood is necessarily smaller. In this situation (60) can be used to identify an upper bound on the number of non-zero elements in Q. The situation for this two-dimensional lattice case is illustrated in Fig. 15.

In the figure the red nodes are sequential neighbours of the green node. Assuming the green node to represent element k in the state vector, we let Ψ_k denote the set of nodes inside the smallest rectangle that contains node k and all the sequential neighbours of k. By construction we then have $\Lambda_k \cup \{k\} \subseteq \Psi_k$. In Fig. 15 the set Ψ_k consists of all nodes inside the red rectangle. Letting u and v denote the vertical and horizontal dimensions, respectively, of the rectangle defining Ψ_k, again as illustrated in Fig. 15, we get that the number of nodes in Ψ_k is $|\Psi_k| = (u+1)(v+1)$. In Fig. 15 we have $u = 2$ and $v = 5$ and $|\Psi_k| = 18$. As $\Lambda_k \cup \{k\} \subseteq \Psi_k$ it follows from (60) that

$$\partial k \subseteq \Omega_k = \bigcup_{t:k\in\Psi_t} \Psi_t. \quad (61)$$

Thereby an upper bound on the number of non-zero elements in row k of Q is the number of nodes in the set Ω_k. Since Ψ_k is defined by a rectangle, the set Ω_k can also be defined from a rectangle. In Fig. 15, the set Ω_k consists of all nodes inside the green rectangle, whereas the set ∂k consists of the green, red and blue nodes in the figure. The vertical and horizontal dimensions of the rectangle defining Ω_k become $2u$ and $2v$, respectively, and the number of elements in Ω_k becomes $|\Omega_k| = (2u+1)(2v+1)$. So the the number of non-zero elements in row k of Q is bounded by $(2u+1)(2v+1)$. Moreover, since we have n_x elements in the state vector, we will have at most $(2u+1)(2v+1)n_x$ non-zero elements in the precision matrix Q.

In the two-dimensional lattice case we can also provide an upper bound for the bandwidth of Q. Still assuming that each node in the lattice is assigned similar sequential neighbourhoods, an upper bound for the bandwidth is the maximum distance between node k and any node $\ell \in \Omega_k$, i.e. $su + v$, where s is the number of columns in the grid.

C Derivation of posterior for hyperpriors

In the following we derive the posterior distribution for (η, ϕ). In the derivation we simplify the notation by omitting the subscripts on the f's as it should be clear from the context what densities we are dealing with. We start by identifying $f(\phi|x, z^{(m)})$ and thereafter find $f(\eta|\phi, x, z^{(m)})$.

Using the definitions of $f(x|\eta, \phi)$, $f(\phi)$ and $f(\eta|\phi)$ given in Sect. 4.1, and that $z^{(m)}$ and x are conditionally independent given ϕ and η, we get

$$
\begin{aligned}
f\big(\phi|z^{(m)}, x\big) &\propto f(\phi) f\big(z^{(m)}, x|\phi\big) \\
&= f(\phi) \int f(\eta|\phi) f\big(z^{(m)}, x|\phi, \eta\big) d\eta \\
&= f(\phi) \int f(\eta|\phi) f(x|\phi, \eta) f\big(z^{(m)}|\phi, \eta\big) d\eta \\
&= f(\phi) \int f(\eta|\phi) f(x|\phi, \eta) \prod_{i \neq m} f\big(x^{(i)}|\phi, \eta\big) d\eta \\
&= \prod_{k=1}^{K} \bigg[f(\phi^k) \int f\big(\eta^k|\phi^k\big) f\big(x^k|\phi^k, \eta^k, x^{\Lambda_k}\big) \\
&\qquad \prod_{i \neq m} f\big(x^{(i),k}|\phi^k, \eta^k, x^{\Lambda_k}\big) d\eta^k \bigg] \\
&\propto \prod_{k=1}^{K} f\big(\phi^k|z^{(m)}, x\big),
\end{aligned}
$$

where

$$
\begin{aligned}
f\big(\phi^k|z^{(m)}, x\big) &\propto f(\phi^k) \int f\big(\eta^k|\phi^k\big) f\big(x^k|\phi^k, \eta^k, x^{\Lambda_k}\big) \\
&\quad \prod_{i \neq m} f\big(x^{(i),k}|\phi^k, \eta^k, x^{\Lambda_k}\big) d\eta^k \\
&= f(\phi^k) \cdot \frac{1}{(\phi^k)^{(|\Lambda_k|+\mathcal{M}+1)/2}} \\
&\quad \int \exp\bigg\{ -\frac{1}{2}\big(\eta^k - \zeta^k\big)^T \big(\phi^k \Sigma_{\eta^k}\big)^{-1}\big(\eta^k - \zeta^k\big) \\
&\quad -\frac{1}{2\phi^k}\big(x^k - \big(1, (x^{\Lambda_k})^T\big)\eta^k\big)^2 - \frac{1}{2\phi^k} \\
&\quad \sum_{i \neq m}\big(x^{(i),k} - \big(1, (x^{(i),\Lambda_k})^T\big)\eta^k\big)^2 \bigg\} d\eta^k. \quad (62)
\end{aligned}
$$

We note that the exponent in the integrand is a second order function in η^k, so by completing the square it is straightforward to evaluate the integral analytically. When having evaluated the integral and inserting for $f(\phi^k)$ this gives, using the notation defined in (38) to (40),

$$
\begin{aligned}
f\big(\phi^k|z^{(m)}, x\big) &\propto \frac{e^{-\frac{1}{\phi^k \beta^k}}}{(\phi^k)^{\alpha^k+1}} \cdot \\
&\quad \frac{1}{(\phi^k)^{\mathcal{M}/2}} e^{-\frac{1}{2\phi^k}\big(\gamma^{m,k} - (\rho^{m,k})^T (\Theta^{m,k})^{-1} \rho^{m,k}\big)} \\
&= \frac{e^{-\frac{1}{\phi^k \tilde{\beta}^{m,k}}}}{(\phi^k)^{\tilde{\alpha}^k+1}},
\end{aligned}
$$

where $\tilde{\alpha}^k$ and $\tilde{\beta}^{m,k}$ are as given in (36) and (37), respectively. We recognise this as the density of a inverse gamma distribution, so

$$
\phi^k|z^{(m)}, x \sim \text{InvGam}\big(\tilde{\alpha}^k, \tilde{\beta}^{m,k}\big). \quad (63)
$$

This concludes the derivation of the posterior distribution for ϕ, and we proceed to the derivation of the posterior for $\eta|\phi$. Using the definitions of $f(x|\eta, \phi)$ and $f(\eta|\phi)$ we get

$$
\begin{aligned}
f\big(\eta|x, z^{(m)}, \phi\big) &\propto f(\eta|\phi) f\big(x, z^{(m)}|\eta, \phi\big) \\
&= f(\eta|\phi) f(x|\eta, \phi) \prod_{i \neq m} f\big(x^{(m)}|\eta, \phi\big) \\
&= \prod_{k=1}^{K} \bigg[f\big(\eta^k|\phi^k\big) f\big(x^k|\eta^k, \phi^k, x^{\Lambda_k}\big) \\
&\qquad \prod_{i \neq m} f\big(x^{(i),k}|\eta^k, \phi^k, x^{\Lambda_k}\big) \bigg] \\
&\propto \prod_{i \neq m} f\big(\eta^k|x, z^{(m)}, \phi^k\big),
\end{aligned}
$$

where

$$
\begin{aligned}
f\big(\eta^k|x, z^{(m)}, \phi^k\big) &\propto f\big(\eta^k|\phi^k\big) f\big(x^k|\eta^k, \phi^k, x^{\Lambda_k}\big) \\
&\quad \prod_{i \neq m} f\big(x^{(i),k}|\eta^k, \phi^k, x^{\Lambda_k}\big) \\
&\propto \frac{1}{(\phi^k)^{(|\Lambda_k|+\mathcal{M}+1)/2}} \\
&\quad \exp\bigg\{ -\frac{1}{2}\big(\eta^k - \zeta^k\big)^T \big(\phi^k \Sigma_{\eta^k}\big)^{-1}\big(\eta^k - \zeta^k\big) \bigg\} \\
&\quad \times \exp\bigg\{ -\frac{1}{2\phi^k}\big(x^k - \big(1, (x^{\Lambda_k})^T\big)\eta^k\big)^2 - \frac{1}{2\phi^k} \\
&\quad \sum_{i \neq m}\big(x^{(i),k} - \big(1, (x^{(i),\Lambda_k})^T\big)\eta^k\big)^2 \bigg\}.
\end{aligned}
$$

Again introducing $\gamma^{i,k}$, $\rho^{i,k}$ and $\Theta^{i,k}$, as defined in (38) to (40) this can be simplified to

$$f\left(\eta^k | x, z^{(m)}, \phi^k\right)$$
$$\propto \exp\left\{-\frac{1}{2\phi^k}\left(\eta^k - \left(\Theta^{m,k}\right)^{-1}\rho^{m,k}\right)^T\right.$$
$$\left.\Theta^{m,k}\left(\eta^k - \left(\Theta^{m,k}\right)^{-1}\rho^{m,k}\right)\right\},$$

which we recognise as the density of a normal distribution, so

$$\eta^k | x, z^{(m)}, \phi^k \sim N\left(\left(\Theta^{m,k}\right)^{-1}\rho^{m,k}, \phi^k\left(\Theta^{m,k}\right)^{-1}\right). \qquad (64)$$

Thereby the derivation of $f\left(\eta, \phi | x, z^{(m)}\right)$ is complete.

References

Bocquet, M.: Ensemble Kalman filtering without the intrinsic need for inflation. Nonlinear Process. Geophys. **18**, 735–750 (2011)

Bocquet, M., Raanes, P.N., Hannart, A.: Expanding the validity of the ensemble Kalman filter without the intrinsic need for inflation. Nonlinear Process. Geophys. **22**, 645–662 (2015)

Brockwell, P.J., Davis, R.A.: Time Series: Theory and Methods, 2nd edn. Springer, Berlin (1990)

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

Chan, J.C.C., Strachan, R.W.: Bayesian state space models in macroeconometrics. J. Econ. Surv. (To appear) (2020)

Creal, D.: A survey of sequential Monte Carlo methods for economics and finance. Econom. Rev. **31**, 245–296 (2011)

Cressie, N., Davidson, J.: Image analysis with partially ordered Markov models. Comput. Stat. Data Anal. **29**, 1–26 (1998)

Doucet, A., Johansen, A.M.: A tutorial on particle filtering and smoothing: fifteen years later. Oxf. Handb. Nonlinear Filter. **12**, 3 (2011)

Doucet, A., de Freitas, N., Gordon, N.: Sequential Monte Carlo Methods in Practice. Springer, New York (2001)

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

Evensen, G.: Data Assimilation. The Ensemble Kalman Filter, 2nd edn. Springer, Dordrecht (2009)

Haugen, V.E.J., Evensen, G.: Assimilation of SLA and SST data into an OGCM for the Indian ocean. Ocean Dyn. **52**, 133–151 (2002)

Hotta, D., Ota, Y.: Why does EnKF suffer from analysis overconfidence? An insight into exploiting the ever-increasing volume of observations. Q. J. R. Meteorol. Soc. **147**, 1258–1277 (2021)

Houtekamer, P.L., Mitchell, H.L.: Data assimilation using an ensemble Kalman filter technique. Mon. Weather Rev. **126**, 796–811 (1997)

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

Janjic, T., Nerger, L., Albertella, A., et al.: On domain localization in ensemble-based Kalman filter algorithms. Mon. Weather Rev. **139**, 2046–2060 (2011)

Kalman, R.: A new approach to linear filtering and prediction problems. J. Basic Eng. Trans. ASME **82**, 35–46 (1960)

Kalman, R., Bucy, R.: New results in linear filtering and prediction theory. J. Basic Eng. Trans. ASME **83**, 95–108 (1961)

Katzfuss, M., Stroud, J.R., Wikle, C.K.: Understanding the ensemble Kalman filter. Am. Stat. **70**, 350–357 (2016)

Katzfuss, M., Stroud, J.R., Wikle, C.K.: Ensemble Kalman methods for high-dimensional hierarchical dynamic space-state models. J. Am. Stat. Assoc. **115**, 866–885 (2020)

Loe, M.K., Tjelmeland, H.: Ensemble updating of binary state vectors by maximizing the expected number of unchanged components. Scand. J. Stat. **48**, 1148–1185 (2020)

Loe, M.K., Tjelmeland, H.: A generalised and fully Bayesian framework for ensemble updating. Tech. rep., ArXiv e-prints (2021). arXiv:2103.14565

Loe, M.K., Tjelmeland, H.: Ensemble updating of categorical state vectors. Comput. Stat. **37**, 2363–2397 (2022)

Loeliger, H.A., Dauwels, J., Hu, J., et al.: The factor graph approach to model-based signal processing. Proc. IEEE **95**, 1295–1322 (2007)

Luo, X., Hoteit, I.: Covariance inflation in the ensemble Kalman filter: a residual nudging perspective and some implications. Mon. Weather Rev. **126**, 796–811 (1997)

Myrseth, I., Sætrom, J., Omre, H.: Resampling the ensemble Kalman filter. Comput. Geosci. **55**, 44–53 (2013)

Omre, H., Myrseth, I.: Hierarchical ensemble Kalman filter. SPE J. **15**, 569–580 (2010)

Rue, H., Held, L.: Gaussian Markov Random Fields. Chapman & Hall/CRC, London (2005)

Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications, 4th edn. Springer, Berlin (2016)

Smith, A.C., Emery, N.: Estimating a state-space model from point process observations. Neural Comput. **15**, 965–991 (2003)

Sætrom, J., Omre, H.: Uncertainty quantification in the ensemble Kalman filter. Scand. J. Stat. **40**, 868–885 (2013)

Tsyrulnikov, M., Rakitko, A.: A hierarchical Bayes ensemble Kalman filter. Physica D **338**, 1–16 (2017)

Woodbury, M.A.: Inverting Modified Matrices. Memorandum report 42, Statistical Research group, Princeton University, Princeton (1950)