AN ENERGY-BASED DEEP SPLITTING METHOD FOR THE NONLINEAR FILTERING PROBLEM

KASPER BÅGMARK, ADAM ANDERSSON, AND STIG LARSSON

Abstract. The purpose of this paper is to explore the use of deep learning for the solution of the nonlinear filtering problem. This is achieved by solving the Zakai equation by a deep splitting method, previously developed for approximate solution of (stochastic) partial differential equations. This is combined with an energy-based model for the approximation of functions by a deep neural network. This results in a computationally fast filter that takes observations as input and that does not require re-training when new observations are received. The method is tested on four examples, two linear in one and twenty dimensions and two nonlinear in one dimension. The method shows promising performance when benchmarked against the Kalman filter and the bootstrap particle filter.

1. Introduction

Nonlinear filtering is a topic intertwined between Bayesian statistics, stochastic analysis and numerical analysis. It concerns finding the conditional distribution of an unknown state, given noisy observations. There are many domains of applications for the filtering problem, e.g., finance [8, 12], chemical engineering [41], weather forecasts [10, 14], and target tracking [7, 19]. The most common approaches to approximating the nonlinear filtering problem involves extended and unscented Kalman filters as well as particle filters [27, 40, 43]. Extended and unscented Kalman filters are useful in settings with unimodal, symmetric and approximately Gaussian densities but perform poorly when dealing with multimodal models. Particle filters are useful when dealing with more complex distributions but the number of required particles, and thus the computational complexity, scales poorly in the number of dimensions of the state space [40]. The normalized optimal filter is known from theory to solve the Kushner–Stratonovich equation, a nonlinear Stochastic Partial Differential Equation (SPDE) [32]. The unnormalized optimal filter solves the Zakai equation which is a linear SPDE. These theoretically appealing facts have not been extensively used to derive practical algorithms, mostly because of the computational load associated with approximating SPDEs with classical methods such as finite element or finite difference methods. Following the recent years' extensive developments in solving PDEs and SPDEs with deep neural networks [3, 4, 11, 47], new opportunities arise. In [4] the authors present a deep splitting method for high-dimensional PDEs. It relies on operator splitting and deep learning. This grid-free method was demonstrated by approximating solutions to PDEs in up to 10 000 dimensions. In the follow up paper [3] the method was applied successfully to SPDEs in up to 50 dimensions. In particular, it was applied to the Zakai equation for nonlinear filtering with a fixed observation sequence and a quite peculiar dynamics, chosen to admit an analytic benchmark solution.

Other deep learning based approaches that can be used to estimate the filtering density involve parameter optimization for a family of distributions, which is a common technique for estimating probability densities in regression problems [28, 33]. A limitation of this approach is that it is sometimes difficult to find parameterized families of distributions that are flexible enough to fit the data well. A successful alternative to parameterizing the distributions is given by energy-based methods. This family of methods has shown excellent performance on regression problems [20, 34, 46]. The energy-based models are commonly trained by minimizing the negative log-likelihood. Other

2020 *Mathematics Subject Classification.* 60G35, 62F15, 62G07, 62M20, 65C30, 65M75, 68T07.

Key words and phrases. Filtering problem, Zakai equation, stochastic partial differential equation, splitting scheme, deep learning, energy-based method.

common techniques are based on the Kullback–Leibler divergence, noise contrastive estimation and score-matching, see [20, 21, 25, 46].

In the present exploratory work we take a step towards fast and scalable nonlinear filters. We do this by combining an energy-based approach with the deep splitting method of [3] applied to the Zakai equation. In this way we use a sound probabilistic model that is flexible and does not allow negative values. A main contribution is that we allow the observation sequences as input to the model, avoiding re-training for every new sequence. We demonstrate our model on two linear and Gaussian examples and on two nonlinear examples. The performance is measured with the mean absolute error, a first moment error, and the Kullback–Leibler divergence. Our method shows promising performance when benchmarked against the Kalman filter and the bootstrap particle filter. There are no contributions to the error analysis in the present paper, however, in a future work we investigate the convergence order of this method theoretically.

The paper is structured as follows. In Section 2 we present a background on the filtering problem and its solution by the Zakai equation. The deep splitting method and our extension of it is derived in Section 3. The energy-based approach and the full algorithm is presented in Section 4. It also contains a discussion of two previous approaches to solving the optimal filtering problem with deep splitting methods [3, 11]. In Section 5 we present our numerical results.

2. Preliminaries

In this section we present the notation that we use, the filtering problem, and the Zakai equation that solves it. The presentation is formal and is valid under suitable conditions. Details are omitted.

2.1. **Notation.** We denote by $\langle \cdot\,, \cdot \rangle$ and $\| \cdot \|$ the inner product and norm on the Euclidean space $\mathbb{R}^d$. The space of functions on $[0,T] \times \mathbb{R}^d \to \mathbb{R}$, that are once continuously differentiable in the first variable and twice continuously differentiable in the second variable with no cross derivatives between the first variable and the second variable, is denoted $C^{1,2}([0,T] \times \mathbb{R}^d; \mathbb{R})$. Probability distributions over $\mathbb{R}^d$ are all assumed to have a probability density function with respect to Lebesgue measure. We follow the convention in Bayesian modelling to denote densities p and let their arguments specify which one is meant. For a stochastic process $Y\colon \mathbb{N} \times \Omega \to \mathbb{R}^{d'}$, we denote by $Y_{k:n}$ the $d' \times (n-k+1)$-matrix $(Y_k, Y_{k+1}, \dots, Y_n)$, where $k < n$.

2.2. **The filtering problem.** Filtering aims at finding the conditional distribution of an unobserved state variable, given noisy measurement of the state. In the setting of this paper both state and measurements are modeled by Stochastic Differential Equations (SDE). We denote by $(\Omega, \mathcal{A}, (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$ a complete filtered probability space. The filtration $\mathcal{F} := (\mathcal{F}_t)_{0\leq t \leq T}$ is defined with respect to W and V, which are two d- respectively d'-dimensional independent Brownian motions. For a drift coefficient $\mu\colon \mathbb{R}^d \to \mathbb{R}^d$ and a diffusion coefficient $\sigma\colon \mathbb{R}^d \to \mathbb{R}^{d\times d}$, regular enough, the process $X\colon [0,T] \times \Omega \to \mathbb{R}^d$, commonly referred to as the signal process or latent state process, is the process that for all $t \in [0,T]$, $\mathbb{P}$-a.s., satisfies

$$X_t = X_0 + \int_0^t \mu(X_s)\,\mathrm{d}s + \int_0^t \sigma(X_s)\,\mathrm{d}W_s. \tag{1}$$

The initial condition X_0 is $\mathcal{F}_0$-measurable, independent of W and V, with a distribution p_0. In addition, for a sufficiently regular measurement function $h\colon \mathbb{R}^d \to \mathbb{R}^{d'}$, we define the observation process $Y\colon [0,T] \times \Omega \to \mathbb{R}^{d'}$ satisfying for all $t \in [0,T]$, $\mathbb{P}$-a.s.,

$$Y_t = \int_0^t h(X_s)\,\mathrm{d}s + V_t. \tag{2}$$

The filtering problem consists of finding the conditional probability density of the state X_t given the observations $(Y_s)_{0\leq s\leq t}$. This is commonly referred to as the filtering density. More precisely, the filtering density π is at time t and observation $(Y_s)_{0\leq s\leq t}$ the function, that for all measurable sets $B \subset \mathbb{R}^d$, satisfies

$$\mathbb{P}(X_t \in B \mid (Y_s)_{0\leq s\leq t}) = \int_B \pi_t(x \mid (Y_s)_{0\leq s\leq t})\,\mathrm{d}x.$$

Since Y is stochastic, in fact, π_t is a density-valued stochastic process (under suitable regularity assumptions). On the basis of the Kallianpur–Striebel formula, one can derive an associated SPDE whose solution is an unnormalized version of the filtering distribution. This equation is known as the Zakai equation and is first derived in weak form. Additional conditions guarantee that the distribution has a density with respect to Lebesgue measure, in which case the density, p, is a solution to the strong form of the Zakai equation. These derivations together with rigorous assumptions can be found in [2].

Next, we present the strong form of the Zakai equation [49]. We denote an unnormalized filtering density by $(p_t)_{0\le t\le T}$. To introduce the equation, we recall the differential operator A associated to the process X as well as its formal adjoint A^*, which are defined, with $a := \sigma\sigma^\top$, for $\varphi \in C_0^\infty(\mathbb{R}^d;\mathbb{R})$, as

$$A\varphi = \frac{1}{2}\sum_{i,j=1}^{d} a_{ij}\,\frac{\partial^2\varphi}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \mu_i\,\frac{\partial \varphi}{\partial x_i} \quad \text{and} \quad A^*\varphi = \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}(a_{ij}\varphi) - \sum_{i=1}^{d} \frac{\partial}{\partial x_i}(\mu_i\varphi).$$

The operator A is associated to the deterministic PDE known as the Kolmogorov backward equation and the adjoint operator A^* is associated to the Kolmogorov forward equation. The latter is also called the Fokker–Planck equation, which models the (unconditional) density of the signal process X [39], while the Zakai equation models the (conditional) filtering density.

The drift coefficient μ, the diffusion coefficient σ, and the measurement function h are assumed to be regular enough so that a solution to the Zakai equation exists for any initial density p_0. The strong form of the Zakai equation is to find a function p such that for $t \in [0,T]$, $\mathbb{P}$-a.s.

$$p_t(x) = p_0(x) + \int_0^t A^* p_s(x)\,\mathrm{d}s + \int_0^t p_s(x)h(x)^\top\,\mathrm{d}Y_s, \quad x \in \mathbb{R}^d. \tag{3}$$

See [2] for details on the derivation of the equation. Following [3], we consider a more general equation of the form

$$\begin{aligned} p_t(x) = {}& p_0(x) + \int_0^t A p_s(x)\,\mathrm{d}s + \int_0^t f(x, p_s(x), \nabla p_s(x))\,\mathrm{d}s \\ &+ \int_0^t b(x, p_s(x), \nabla p_s(x))\,\mathrm{d}Y_s, \quad x \in \mathbb{R}^d, \end{aligned} \tag{4}$$

with coefficients $f\colon \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ and $b\colon \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^{d'}$. By expanding the derivative A^* we see that the Zakai equation (3) is of this form with

$$\begin{aligned} f(x,u,v) &= \sum_{i,j=1}^{d} \frac{\partial a_{ij}(x)}{\partial x_i}\,v_j + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 a_{ij}(x)}{\partial x_i \partial x_j}\,u - \sum_{i=1}^{d} \frac{\partial \mu_i(x)}{\partial x_i}\,u - 2\sum_{i=1}^{d} \mu_i(x)\,v_i, \\ b(x,u,v) &= u h(x)^\top. \end{aligned}$$

In addition to letting (4) represent more a general SPDE, the main reason for rewriting (3) in this way is that equations with leading operator A admit Feynman–Kac type representations of solutions. This is used in the next section in the derivation of the deep splitting scheme.

It is important to note that if the system of SDEs (1)–(2) have a linear drift coefficient μ, constant diffusion coefficient σ, as well as a linear measurement function h, then the exact density of the filter is tractable. The problem is then solved by an evolution of Gaussian densities known as the Kalman–Bucy filter, see [43] for details. Otherwise there are few closed form filters, one such exception is the Beneš filter which solves a bistable model with a tanh drift. See [2, 6] for details.

3. Deep splitting method

The deep splitting method was originally derived in [4] for a class of parabolic PDEs and it was extended in [3] to SPDEs on the form of (4). The aim of this section is to provide a derivation of the proposed modified recursive optimization problem. In this section, we make the tacit assumption that both f and b are regular enough so that (4) has a unique strong solution. We start by deriving the original method, contributing with a complement to the presentation in [3], and continue by deriving an extension. The section begins by applying a splitting scheme and continues by deriving a Feynman–Kac representation. Finally, we obtain a modified version of the method derived in [3], which is later used in the numerical examples.

3.1. **A splitting scheme for the SPDE.** We introduce a partition of $[0,T]$ into $0 = t_0 < t_1 < \cdots < t_M = T$ and note that (4) can be written for $n = 0, \ldots, M-1$, $\mathbb{P}$-a.s.,

$$\begin{aligned} p_t(x) = p_{t_n}(x) + \int_{t_n}^{t} A p_s(x)\,\mathrm{d}s + \int_{t_n}^{t} f(x, p_s(x), \nabla p_s(x))\,\mathrm{d}s \\ + \int_{t_n}^{t} b(x, p_s(x), \nabla p_s(x))\,\mathrm{d}Y_s, \quad t \in (t_n, t_{n+1}]. \end{aligned} \tag{5}$$

In the next step we split (5) into two equations. The idea behind this splitting is to treat the operator term differently than the terms involving f and b. We define q and $\widehat{q}$ recursively on $(t_n, t_{n+1}]$, for $n = 0, \ldots, M-1$, as the solutions to

$$q_t(x) = \widehat{q}_{t_n}(x) + \int_{t_n}^{t} f(x, q_s(x), \nabla q_s(x))\,\mathrm{d}s + \int_{t_n}^{t} b(x, q_s(x), \nabla q_s(x))\,\mathrm{d}Y_s, \quad t \in (t_n, t_{n+1}], \tag{6}$$

$$\widehat{q}_t(x) = q_{t_{n+1}}(x) + \int_{t_n}^{t} A\widehat{q}_s(x)\,\mathrm{d}s, \quad t \in (t_n, t_{n+1}], \tag{7}$$

$$\widehat{q}_0(x) = p_0(x). \tag{8}$$

Note that q and $\widehat{q}$ are piecewise smooth with respect to t. This splitting method is constructed such that $\widehat{q}_{t_{n+1}} \approx p_{t_{n+1}}$ for $n = 0, \ldots, M-1$ and is investigated in, e.g., [22, 23] where strong convergence order 1 in time is shown. We further approximate (6) with an Euler–Maruyama scheme, which in general is of strong order 0.5 in time. Merging the equations (6)–(7), after Euler–Maruyama approximation, into one formula we obtain the following approximation of (5), where we keep the notation of $\widehat{q}$,

$$\begin{aligned} \widehat{q}_t(x) &= \widehat{q}_{t_n}(x) + \int_{t_n}^{t} A\widehat{q}_s(x)\,\mathrm{d}s + f(x, \widehat{q}_{t_n}(x), \nabla \widehat{q}_{t_n}(x))(t_{n+1} - t_n) \\ &\quad + b(x, \widehat{q}_{t_n}(x), \nabla \widehat{q}_{t_n}(x))(Y_{t_{n+1}} - Y_{t_n}), \quad t \in (t_n, t_{n+1}], \\ \widehat{q}_0(x) &= p_0(x). \end{aligned}$$

In preparation for the final algorithm we consider, for fixed $\omega \in \Omega$, deterministic input $y = (Y_{t_n}(\omega))_{n=0}^{M} \in \mathbb{R}^{d \times (M+1)}$ and define an approximation $\widehat{p}_t(x) = \widehat{p}_t(x, y_{0:n+1})$ recursively on $(t_n, t_{n+1}]$ for $n = 0, \ldots, M-1$, by

$$\begin{aligned} \widehat{p}_t(x) &= \widehat{p}_{t_n}(x) + \int_{t_n}^{t} A\widehat{p}_s(x)\,\mathrm{d}s + f(x, \widehat{p}_{t_n}(x), \nabla \widehat{p}_{t_n}(x))(t_{n+1} - t_n) \\ &\quad + b(x, \widehat{p}_{t_n}(x), \nabla \widehat{p}_{t_n}(x))(y_{n+1} - y_n), \quad t \in (t_n, t_{n+1}], \\ \widehat{p}_0(x) &= p_0(x). \end{aligned} \tag{9}$$

This is a splitting approximation of p, where the idea is to first approximate (6) with the Euler–Maruyama scheme and then solve the equation with respect to the generator A exactly in the second step in (7). There exists a unique solution $\widehat{p}$ to the Cauchy problem (9), which belongs to $C^{1,2}((t_n, t_{n+1}] \times \mathbb{R}^d; \mathbb{R})$ [16, Chapter 1, Theorem 10]. In Subsection 3.4 this method is extended to a Milstein scheme in the special case $d = 1$. In the next subsection we derive an optimization problem based on this splitting scheme.

3.2. **Derivation of a local optimization problem.** In this subsection we fix $n \in \{0, \ldots, M-1\}$ and let $N \in \{n+1, \ldots, M\}$ be arbitrary. We will derive a Feynman–Kac formula for $\widehat{p}|_{(t_n, t_{n+1}]}$. We begin by noting that, from (9), it is clear that

$$\frac{\partial}{\partial t}\widehat{p}_t(x) = A\widehat{p}_t(x), \quad t \in (t_n, t_{n+1}].$$

Next, we reparameterize with time $t \mapsto t_N - t$, so that $t_N - t \in (t_n, t_{n+1}]$, which yields

$$\frac{\partial}{\partial t}\widehat{p}_{t_N - t}(x) + A\widehat{p}_{t_N - t}(x) = 0, \quad t \in [t_N - t_{n+1}, t_N - t_n), \tag{10}$$

with final condition

$$\begin{aligned}\widehat{p}_{t_N-(t_N-t_n)^-}(x) = \widehat{p}_{t_n^+}(x) = \widehat{p}_{t_n}(x) + f(x, \widehat{p}_{t_n}(x), \nabla \widehat{p}_{t_n}(x))(t_{n+1}-t_n) \\ + b(x, \widehat{p}_{t_n}(x), \nabla \widehat{p}_{t_n}(x))(y_{n+1}-y_n),\end{aligned} \tag{11}$$

where $\widehat{p}_{t_n}$ is defined on the previous interval $(t_{n-1}, t_n]$, and we note that t_n^+ and $(t_N - t_n)^-$ denote a right and left limit, respectively. We see that the approximation $\widehat{p}$ satisfies a Kolmogorov backward equation (10). Such equations are studied in [16, 17], where it is shown that there exists a unique solution $\widehat{p} \in C^{1,2}((t_n, t_{n+1}] \times \mathbb{R}^d; \mathbb{R})$, which together with its first spatial derivatives satisfies a polynomial growth bound in space. We refer to the material in Sections 4 and 5 leading up to Theorem 6.5.3 in Chapter 6 of [17].

We introduce a new Itô process $\widetilde{X}$ defined with respect to a d-dimensional Brownian motion $\widetilde{W}$ independent of W and V. The Brownian motion $\widetilde{W}$ is adapted with respect to the filtration $\widetilde{\mathcal{F}} := (\widetilde{\mathcal{F}}_t)_{0 \le t \le T}$. The process $\widetilde{X}$ satisfies, $\mathbb{P}$-a.s.,

$$\widetilde{X}_t = \widetilde{X}_0 + \int_0^t \mu(\widetilde{X}_s)\,\mathrm{d}s + \int_0^t \sigma(\widetilde{X}_s)\,\mathrm{d}\widetilde{W}_s, \quad t \in [0,T],$$

where $\widetilde{X}_0$ is $\widetilde{\mathcal{F}}_0$-measurable, independent of W, V and $\widetilde{W}$, with distribution p_0. Since $\widetilde{X}$ is defined with respect to the generator A, we obtain A in the integrand when applying Itô's formula. Using the fact that $\widehat{p} \in C^{1,2}((t_n, t_{n+1}] \times \mathbb{R}^d; \mathbb{R})$, Itô's formula can be applied to $\widehat{p}_{t_N-t}(\widetilde{X}_t)$ which gives $\mathbb{P}$-a.s.

$$\begin{aligned}\widehat{p}_{t_N-t}(\widetilde{X}_t) = \widehat{p}_{t_{n+1}}(\widetilde{X}_{t_N-t_{n+1}}) + \int_{t_N-t_{n+1}}^t \langle \nabla \widehat{p}_{t_N-s}(\widetilde{X}_s), \sigma(\widetilde{X}_s)\,\mathrm{d}\widetilde{W}_s\rangle \\ + \int_{t_N-t_{n+1}}^t \Big(\frac{\partial}{\partial s}\widehat{p}_{t_N-s}(\widetilde{X}_s) + A\widehat{p}_{t_N-s}(\widetilde{X}_s)\Big)\,\mathrm{d}s, \quad t \in [t_N - t_{n+1}, t_N - t_n).\end{aligned}$$

Inserting (10) into the right hand side yields

$$\widehat{p}_{t_N-t}(\widetilde{X}_t) = \widehat{p}_{t_{n+1}}(\widetilde{X}_{t_N-t_{n+1}}) + \int_{t_N-t_{n+1}}^t \langle \nabla \widehat{p}_{t_N-s}(\widetilde{X}_s), \sigma(\widetilde{X}_s)\,\mathrm{d}\widetilde{W}_s\rangle. \tag{12}$$

The polynomial growth bounds on $\widehat{p}$, $\nabla\widehat{p}$ and the assumptions on σ guarantee that

$$\int_{t_N-t_{n+1}}^{t_N-t_n} \mathbb{E}\Big[\big\|\sigma(\widetilde{X}_s)^\top \nabla \widehat{p}_{t_N-s}(\widetilde{X}_s)\big\|^2\Big]\,\mathrm{d}s < \infty$$

and hence the Itô integral in (12) is a square integrable martingale with respect to $\widetilde{\mathcal{F}}$. The conditional expectation with respect to the filtration $\widetilde{\mathcal{F}}$ gives

$$\mathbb{E}\bigg[\int_{t_N-t_{n+1}}^t \langle \nabla \widehat{p}_{t_N-s}(\widetilde{X}_s), \sigma(\widetilde{X}_s)\,\mathrm{d}\widetilde{W}_s\rangle \,\bigg|\, \widetilde{\mathcal{F}}_{t_N-t_{n+1}}\bigg] = 0. \tag{13}$$

Now, as $\widetilde{X}$ is $\widetilde{\mathcal{F}}$-adapted, we can combine (12) and (13) to get, for $t \in [t_N - t_{n+1}, t_N - t_n)$,

$$\mathbb{E}\big[\widehat{p}_{t_N-t}(\widetilde{X}_t) \,\big|\, \widetilde{\mathcal{F}}_{t_N-t_{n+1}}\big] = \mathbb{E}\big[\widehat{p}_{t_{n+1}}(\widetilde{X}_{t_N-t_{n+1}}) \,\big|\, \widetilde{\mathcal{F}}_{t_N-t_{n+1}}\big] = \widehat{p}_{t_{n+1}}(\widetilde{X}_{t_N-t_{n+1}}). \tag{14}$$

We recall from the final condition (11) that, for all $x \in \mathbb{R}^d$, we have,

$$\widehat{p}_{t_N-t}(x) \to \widehat{p}_{t_n}(x), \quad \text{as } t \uparrow (t_N - t_n),$$

and $\mathbb{P}$-a.s.

$$\widetilde{X}_t \to \widetilde{X}_{t_N-t_n}, \quad \text{as } t \uparrow (t_N - t_n).$$

Combining these, we get, by the polynomial growth bounds via dominated convergence, that

$$\begin{aligned}\lim_{t\uparrow(t_N-t_n)} \mathbb{E}\Big[\Big|\widehat{p}_{t_N-t}(\widetilde{X}_t) - \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}) + f(\widetilde{X}_{t_N-t_n}, \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}), \nabla\widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}))(t_{n+1}-t_n) \\ + b(\widetilde{X}_{t_N-t_n}, \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}), \nabla\widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}))(y_{n+1}-y_n)\Big|^2\Big] = 0.\end{aligned}$$

Now, we take the left limit $t \uparrow (t_N - t_n)$ in (14) to obtain the $L^2(\Omega)$-limit

$$\begin{aligned}
&\widehat{p}_{t_{n+1}}(\widetilde{X}_{t_N - t_{n+1}}) \\
&\quad = \mathbb{E}\Big[\widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}) + f(\widetilde{X}_{t_N-t_n}, \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}), \nabla \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}))(t_{n+1}-t_n) \\
&\qquad + b(\widetilde{X}_{t_N-t_n}, \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}), \nabla \widehat{p}_{t_n}(\widetilde{X}_{t_N-t_n}))(y_{n+1}-y_n) \,\big|\, \widetilde{\mathcal{F}}_{t_N-t_{n+1}}\Big].
\end{aligned} \tag{15}$$

To evaluate this recursion numerically we approximate the underlying stochastic $\widetilde{X}$ on the time grid $0 = t_0 < t_1 < \cdots < t_M$. The approximation, denoted $(\widetilde{X}_n)_{n=0}^M$, with initial value $\widetilde{X}_0 \sim p_0$, is given by the Euler–Maruyama method:

$$\widetilde{X}_{n+1} = \widetilde{X}_n + \mu(\widetilde{X}_n)(t_{n+1} - t_n) + \sigma(\widetilde{X}_n)(\widetilde{W}_{t_{n+1}} - \widetilde{W}_{t_n}). \tag{16}$$

For simplicity of notation, we use a uniform mesh with $t_n = n\Delta t$. This allows us to use the same mesh for $\widetilde{X}$, as in the splitting method, because $t_{N-n} = t_N - t_n$ and hence $\widetilde{X}_{N-n} \approx \widetilde{X}_{t_N - t_n}$.

We exchange $\widetilde{\mathcal{F}}_{t_N-t_{n+1}}$ for $\mathfrak{S}(\widetilde{X}_{N-(n+1)}) \subset \widetilde{\mathcal{F}}_{t_N-t_{n+1}}$, with the tower property, substitute (16) for $\widetilde{X}$, and obtain an approximation $\overline{p}_n$ of $\widehat{p}_{t_n}$ defined by

$$\begin{aligned}
&\overline{p}_{n+1}(\widetilde{X}_{N-(n+1)}) \\
&\quad = \mathbb{E}\Big[\overline{p}_n(\widetilde{X}_{N-n}) + f(\widetilde{X}_{N-n}, \overline{p}_n(\widetilde{X}_{N-n}), \nabla \overline{p}_n(\widetilde{X}_{N-n}))(t_{n+1}-t_n) \\
&\qquad + b(\widetilde{X}_{N-n}, \overline{p}_n(\widetilde{X}_{N-n}), \nabla \overline{p}_n(\widetilde{X}_{N-n}))(y_{n+1}-y_n) \,\Big|\, \mathfrak{S}(\widetilde{X}_{N-(n+1)})\Big].
\end{aligned} \tag{17}$$

We thus introduced an additional Euler–Maruyama approximation with strong convergence order 0.5. This representation is an approximate Feynman–Kac formula for the solution of (10)–(11) for a fixed $y = Y(\omega)$, $\omega \in \Omega$. See [39] for more details on the Feynman–Kac formula and [3] for more details on the derivation of this numerical scheme.

The formula in (17) expresses the approximation $\overline{p}(\widetilde{X}_{N-(n+1)})$ as a conditional expectation with respect to $\mathfrak{S}(\widetilde{X}_{N-(n+1)})$. On the other hand, the conditional expectation can be computed using the L^2-minimality property [30, Corollary 8.17] as a minimization problem over $L^2(\Omega, \mathfrak{S}(\widetilde{X}_{N-(n+1)}))$. With an additional argument, see [5, Proposition 2.7], this can be expressed as a minimization over $C(\mathbb{R}^d;\mathbb{R})$. In the context of (17) we have that, for $N \leq M$ and $n = 0, \ldots, N-1$,

$$\begin{aligned}
(\overline{p}_{n+1}(x))_{x\in\mathbb{R}^d} &= \operatorname*{arg\,min}_{u \in C(\mathbb{R}^d;\mathbb{R})} \mathbb{E}\bigg[\Big|u(\widetilde{X}_{N-(n+1)}) - \Big(\overline{p}_n(\widetilde{X}_{N-n}) \\
&\qquad + f(\widetilde{X}_{N-n}, \overline{p}_n(\widetilde{X}_{N-n}), \nabla \overline{p}_n(\widetilde{X}_{N-n}))(t_{n+1}-t_n) \\
&\qquad + b(\widetilde{X}_{N-n}, \overline{p}_n(\widetilde{X}_{N-n}), \nabla \overline{p}_n(\widetilde{X}_{N-n}))(y_{n+1}-y_n)\Big)\Big|^2\bigg], \\
\overline{p}_0(x) &= p_0(x).
\end{aligned} \tag{18}$$

We recall that the solution to (18) gives an approximation to (4) for one fixed realization y. This is the final form of the original recursive optimization problem introduced in [3]. To solve and find good approximators $u \in C(\mathbb{R}^d;\mathbb{R})$ the authors of [3] employ a deep learning framework to (18), hence it is called a "deep splitting method". This optimization is done in [3] for each specific realization $y = Y(\omega)$.

3.3. **Extension to non-fixed observation sequence.** Up to this point we have derived an optimization problem for fixed observation sequence. This requires a new training for each new observation sequence and this is very limiting in applications. Instead of solving the optimization problem in (18) for a fixed input sequence $y = y_{0:n+1}$, we now let the input vary over a relevant set of inputs. To achieve this, we could in principle integrate the objective in (18) with respect to y over some probability measure ν. In practice, Monte Carlo approximation is required and for this reason it is important that ν is chosen so that relevant observation sequences are sampled. As we are interested in approximating the filtering density, the natural choice for our setting is letting y be distributed according to (2). Since $\widetilde{X}$ and Y are independent, the measure $\mathbb{P} \times \nu$ can be replaced by $\mathbb{P}$ when replacing y with Y. In this way we obtain a single approximator of the filtering density that can be used for any observation sequence, which implies that the network can be applied to new data without re-training.

We approximate (X, Y) defined in (1)–(2), on the same time grid as for $\widetilde{X}$ and $\overline{p}$, with an Euler–Maruyama method. The approximations $(X_n, Y_n)_{n=0}^{M}$, with $X_0 \sim p_0$ and $Y_0 = 0$, are defined by

$$\begin{aligned} X_{n+1} &= X_n + \mu(X_n)(t_{n+1} - t_n) + \sigma(X_n)(W_{t_{n+1}} - W_{t_n}), \\ Y_{n+1} &= Y_n + h(X_{n+1})(t_{n+1} - t_n) + (V_{t_{n+1}} - V_{t_n}). \end{aligned} \tag{19}$$

To formalize the extended optimization problem, we want to find functions $\widetilde{p}_{n+1} \colon \mathbb{R}^d \times \mathbb{R}^{d\times(n+2)} \to \mathbb{R}$, for $N \leq M$ and $n = 0, \dots, N-1$, satisfying

$$\begin{aligned} &(\widetilde{p}_{n+1}(x,y))_{(x,y)\in\mathbb{R}^d\times\mathbb{R}^{d\times(n+2)}} \\ &\quad = \underset{u\in C(\mathbb{R}^d\times\mathbb{R}^{d\times(n+2)};\mathbb{R})}{\arg\min} \mathbb{E}\bigg[\Big|u(\widetilde{X}_{N-(n+1)}, Y_{0:n+1}) - \Big(\widetilde{p}_n(\widetilde{X}_{N-n}, Y_{0:n}) \\ &\qquad + f(\widetilde{X}_{N-n}, \widetilde{p}_n(\widetilde{X}_{N-n}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{N-n}, Y_{0:n}))(t_{n+1} - t_n) \\ &\qquad + b(\widetilde{X}_{N-n}, \widetilde{p}_n(\widetilde{X}_{N-n}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{N-n}, Y_{0:n}))(Y_{n+1} - Y_n)\Big)\Big|^2\bigg], \\ &\widetilde{p}_0(x,y) = p_0(x). \end{aligned} \tag{20}$$

Solving (20) implies solving (18) for almost every observation sequence y. This can be seen by the following argument. The objective function in (20) can be written $u \mapsto \mathbb{E}[\ell(u, \widetilde{X}, Y)]$, where the reader can identify ℓ. The minimum is attainable and the objective is zero at the optimum. Denoting the minimum by u^* we thus have $\mathbb{E}[L(\widetilde{X}, Y)] := \mathbb{E}[\ell(u^*, \widetilde{X}, Y)] = 0$. The discrete processes $\widetilde{X}$ and Y are independent and assuming they have densities $p_{\widetilde{X}}$ and p_Y with respect to Lebesgue measure in the suitable dimensions, we have

$$\mathbb{E}[L(\widetilde{X}, Y)] = \int\int L(x,y)p_{\widetilde{X}}(x)p_Y(y)\,\mathrm{d}x\,\mathrm{d}y = 0.$$

This implies that

$$\mathbb{E}[L(\widetilde{X}, y)] = \int L(x,y)p_{\widetilde{X}}(x)\,\mathrm{d}x = 0$$

for almost all y. Thus a solution to (20) solves (18) for almost every y.

3.4. **Accounting for a limited number of samples.** The next step is to consider Monte Carlo samples of $(Y_n, \widetilde{X}_n)_{n=0}^{N}$ to approximate the expectation in (20). The limitation of having a finite number of samples can, besides classical Monte Carlo issues, in this context provide an additional problem stemming from distribution mismatch. To exemplify this, consider a one dimensional example with $p_0 = \mathcal{N}(0,1)$, $t_N = 1$, $N = 100$ and a system of SDEs $(X, Y, \widetilde{X})$, in the same form as (1)–(2), given, for $t \in [0,1]$, $\mathbb{P}$-a.s., by

$$\begin{aligned} X_t &= X_0 + 10t + W_t, \\ Y_t &= V_t, \\ \widetilde{X}_t &= \widetilde{X}_0 + 10t + \widetilde{W}_t. \end{aligned}$$

In the first optimization step, $n = 0$, the aim is to find $\widetilde{p}_1$ from (20). This step consists of evaluating $p_0(\widetilde{X}_N)$ and as can be seen in Figure 1a the distribution p_0 and the distribution of $\widetilde{X}_N$ essentially lack overlapping probability mass, being a problem in the finite sample situation of practical algorithms. To account for this we opt for a modified approach of (20). The setup presented so far is structured with a final time t_N and having N recursive minimization problems that are intertwined in the sense that the optimized models depend on training samples from time points potentially far in the future or likewise far back in time. The idea is now to use the fact that $t_N \leq t_M$ is at our disposal in (20). What we aim to do instead is making sure each optimization problem only depends on the closest neighbours of $\widetilde{X}$ in time. Instead of considering all time intervals of the partition simultaneously, we begin by considering $(0, t_1]$. Consider (20) with $N = 1$ and $n = 0$. The optimization problem is then for $\widetilde{p}_1$ defined by

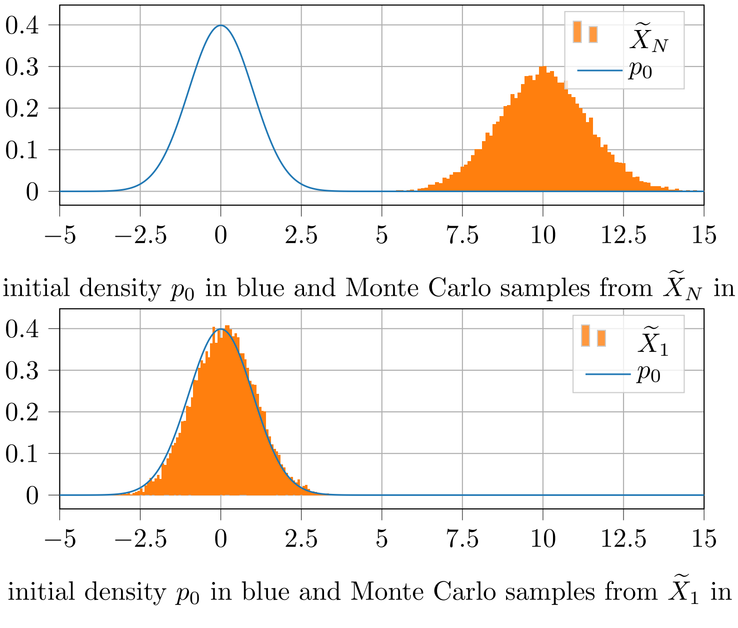
$$\begin{aligned}(\widetilde{p}_1(x,y))_{(x,y)\in\mathbb{R}^d\times\mathbb{R}^{d\times 2}} = \underset{u\in C(\mathbb{R}^d\times\mathbb{R}^{d\times 2};\mathbb{R})}{\arg\min}\ \mathbb{E}\Big[\Big|u(\widetilde{X}_0, Y_{0:1}) - \Big(p_0(\widetilde{X}_1)\\ + f(\widetilde{X}_1, p_0(\widetilde{X}_1), \nabla p_0(\widetilde{X}_1))(t_1 - t_0) + b(\widetilde{X}_1, p_0(\widetilde{X}_1), \nabla p_0(\widetilde{X}_1))(Y_1 - Y_0)\Big)\Big|^2\Big].\end{aligned}$$

By this construction we only need X_0 and X_1 from the local time interval $(0, t_1]$. In Figure 1b we see how the problem of the mismatching distributions is solved, assuming our time step is sufficiently small. In this illustration the time step is 0.01.

For the second step we consider the time interval $(t_1, t_2]$. In this step, with $n = 1$, we optimize $\widetilde{p}_2$ on the samples $\widetilde{X}_1$ and $\widetilde{X}_2$ in a similar way. This is achieved by considering $N = 3$ and $n = 1$ in (20). We do this at every time interval to obtain a local optimization problem. For each time step n, we let $N = 2n + 1$ to obtain the problem on a local interval. Notice that we can choose M so that we never evaluate $\widetilde{p}$ in $\widetilde{X}_m$ for $m > M$. By this construction, the obtained local minimization problem becomes for $N \leq M$ and $n = 0, \ldots, N - 1$

$$\begin{aligned}(\widetilde{p}_{n+1}(x,y))_{(x,y)\in\mathbb{R}^d\times\mathbb{R}^{d\times(n+2)}} = \underset{u\in C(\mathbb{R}^d\times\mathbb{R}^{d\times(n+2)})}{\arg\min}\ \mathbb{E}\Big[\Big|u(\widetilde{X}_n, Y_{0:n+1}) - \Big(\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n})\\ + f(\widetilde{X}_{n+1}, \widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}))(t_{n+1} - t_n)\\ + b(\widetilde{X}_{n+1}, \widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}))(Y_{n+1} - Y_n)\Big)\Big|^2\Big].\end{aligned} \tag{21}$$

By this setup we obtain the same conditions for the filtering problem and we are simply solving the optimization problems independently. This eliminates the problem previously mentioned assuming we have continuous process and a small enough time step Δt.



(a) The initial density p_0 in blue and Monte Carlo samples from $\widetilde{X}_N$ in orange.

(b) The initial density p_0 in blue and Monte Carlo samples from $\widetilde{X}_1$ in orange.

Figure 1

In (21) we use the explicit Euler–Maruyama scheme. In the case $d = 1$ we may instead use the Milstein scheme [31]. Introducing the notation

$$\begin{aligned} f_n &:= f(\widetilde{X}_{n+1}, \widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n})), \\ b_n &:= b(\widetilde{X}_{n+1}, \widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n})), \\ b'_n &:= \frac{\partial}{\partial\widetilde{p}} b(\widetilde{X}_{n+1}, \widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}), \nabla\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n})) \end{aligned}$$

the modified recursive optimization scheme reads

$$\begin{aligned} (\widetilde{p}_{n+1}(x,y))_{(x,y)\in\mathbb{R}\times\mathbb{R}^{n+2}} = \underset{u\in C(\mathbb{R}\times\mathbb{R}^{n+2};\mathbb{R})}{\arg\min}\ \mathbb{E}\Bigg[\Big|u(\widetilde{X}_n, Y_{0:n+1}) - \Big(\widetilde{p}_n(\widetilde{X}_{n+1}, Y_{0:n}) \\ + f_n(t_{n+1}-t_n) + b_n(Y_{n+1}-Y_n) + \frac{1}{2}b_n b'_n((Y_{n+1}-Y_n)^2 - (t_{n+1}-t_n))\Big)\Big|^2\Bigg]. \end{aligned} \tag{22}$$

4. The energy-based approximation scheme

This section defines the proposed method for finding $\widetilde{p}_{n+1}$ given by the minimization problem in (21) or (22). We also make a comparison to two related approaches concerned with the nonlinear filtering problem. In particular, we consider a deep neural network $\Phi_n^{\theta_n} : \mathbb{R}^d \times \mathbb{R}^{d\times(n+1)} \to \mathbb{R}$, parameterized by θ_n, to approximate the solution of the problem at each time step t_n. The optimization problem (22) can be reformulated as finding θ_n^* for all $n = 1, \dots, N$ satisfying

$$\begin{aligned} \theta_1^* &= \arg\min_\theta \mathbb{E}\Bigg[\Big|\Phi_1^\theta(\widetilde{X}_0, Y_{0:1}) - \Big(p_0(\widetilde{X}_1) + f_0(t_1 - t_0) \\ &\quad + b_0(Y_1 - Y_0) + \frac{1}{2}b_0 b'_0((Y_1 - Y_0)^2 - (t_1 - t_0))\Big)\Big|^2\Bigg], \\ \theta_n^* &= \arg\min_\theta \mathbb{E}\Bigg[\Big|\Phi_n^\theta(\widetilde{X}_{n-1}, Y_{0:n}) - \Big(\Phi_{n-1}^{\theta_{n-1}^*}(\widetilde{X}_n, Y_{0:n-1}) + f_{n-1}(t_n - t_{n-1}) \\ &\quad + b_{n-1}(Y_n - Y_{n-1}) + \frac{1}{2}b_{n-1}b'_{n-1}((Y_n - Y_{n-1})^2 - (t_n - t_{n-1}))\Big)\Big|^2\Bigg], \quad n = 2, \dots, N. \end{aligned} \tag{23}$$

Analogously, we can define this optimization scheme with the Euler–Maruyama method (21), which is used when $d \neq 1$. Deep neural networks have a strong ability to approximate nonlinearities and are chosen as function approximators for their ability to scale well in increasing state dimension. In this paper we construct networks with fairly simple but effective architecture. The goal of this paper and method is not necessarily to find the most effective architecture for a regression task. Instead we aim to find an effective solver and approximator of the filtering density by applying the deep learning framework to a well constructed minimization problem.

We consider an energy-based method by letting the normalized conditional density be approximated, for each pair $(x_n, y_{0:n}) \in \mathbb{R}^d \times \mathbb{R}^{d\times(n+1)}$, by

$$p(x_n \mid y_{0:n}) \approx \frac{\Phi_n^{\theta_n}(x_n, y_{0:n})}{Z_n^{\theta_n}(y_{0:n})},$$

where

$$\Phi_n^{\theta_n}(x_n, y_{0:n}) = e^{-f_n^{\theta_n}(x_n, y_{0:n})}, \quad Z_n^{\theta_n}(y_{0:n}) = \int_{\mathbb{R}^d} e^{-f_n^{\theta_n}(x, y_{0:n})}\,\mathrm{d}x.$$

The main idea behind energy-based methods is to let the model output a scalar $f_n^{\theta_n}$, commonly called energy, for each input pair $(x_n, y_{0:n})$. It assigns low energy to input that is likely to occur (accurate) and high energy to unlikely input. The normalizing constant, $Z_n^{\theta_n}$, is evaluated after training to obtain the normalized density. We assume that $f_n^{\theta_n}$ is defined in a way that guarantees that $Z_n^{\theta_n}$ is finite.

4.1. **Comparison to related approaches.** Deep learning for the filtering problem is not an extensively studied topic. To the best of our knowledge, there are only two papers, [3, 11], that are based on partial differential equations. In [3] a deep splitting method is used to solve the Zakai equation, while in [11] the Fokker–Planck equation is solved by deep splitting.

4.1.1. *The original approach based on the Zakai equation.* In [3] the function $\overline{p}$ is approximated, for a fixed observation sequence $y_{0:N}$, by performing the minimization in (18) over the parameters of a deep neural network. The model is a fully connected neural network that can take negative values which would violate the density property. The performance, after training, is measured by the error of the filtering density in one specific point $x \in \mathbb{R}^d$. This spatial point is selected to be close to the mode of the unnormalized density. It is shown that the model learns the value of the filter at this point with small errors at the final time step. The behavior at the other time steps is not demonstrated but in the previous work [4], in which this deep splitting method is applied to PDEs, there is a tendency towards accumulation of error. This is inherent in the recursive optimization procedure as in each step there is remaining error after optimization and in every subsequent step one optimizes with respect to a non-optimal approximation.

In addition to allowing negative values, a problem is that the unnormalized density is only trained around typical trajectories of $\widetilde{X}$. Elsewhere, the neural network cannot be expected to extrapolate or even have finite integral. Therefore, normalizing the approximate solution becomes meaningless and also impossible. Finally, the networks are trained for a fixed observation sequence, making filtering in a real-time setting, often important in applications, impossible.

4.1.2. *An approach based on the Fokker–Planck equation.* In [11] the authors present a similar approach as in [3]. It also involves a deep splitting scheme on an underlying equation. But in [11] this is done directly on the Fokker–Planck equation which gives the unconditional density of X instead. To obtain a corresponding filter estimate, a likelihood normalization is applied after each update of the Fokker–Planck approximation. This likelihood normalization, consists of an update based on the observation Y at the current time point and a normalization of the density, derived from Bayes' formula.

Similarly to the approach of the present paper and [3], the authors in [11] use a grid-free algorithm based on a Feynman–Kac type formulation, approximated by Monte Carlo simulations. The training procedure in [11] is similar to that of [3] but in addition to the L^2-loss at each time step another term is added to encourage positive outputs of the network in the following way

$$\widehat{L}(\theta) = L(\theta) + \lambda \max\left(0, -\Phi^{\theta}\right).$$

In this context $L(\theta)$ represents another version of the right hand side of (23), derived from another splitting scheme than the one in the present paper. With weight λ, negative values of Φ^{θ} are penalized in order to avoid violation of non-negativity of the approximated density. This is an improvement over the use of only a scalar output as in [3], but offers no guarantees in retrieving a non-negative function.

In [11] the authors demonstrate the method on three one-dimensional examples, two of which are solved by the Kalman filter and a third one solved by the Beneš filter. The model manages to approximate non-negative densities and the paper demonstrates corresponding errors for the means of the approximation compared to the true filters. These errors show an oscillating pattern for the more advanced examples. This means that the method is not fully consistent in its predictions of the mean. There are no numerical metrics evaluated with respect to how well the approximation captures the overall distribution, e.g., the tails and shape of the distribution. Furthermore, this work was extended in [36], where the method was demonstrated on the Beneš filter with improved accuracy. The main drawback is that the model is trained for a specific observation sequence and thus would have to be trained again for new observations, similarly to the model in [3].

5. Numerical examples

In this section we employ the proposed method on four different underlying SDEs. Two are linear and Gaussian with the Kalman filter as benchmark. The other two examples are nonlinear SDEs, where we use a particle filter as benchmark. Three of the examples are one-dimensional and allow us to investigate the scheme for different degrees of non-linearity for the drift, one linear and two cubic with uni-modal and bi-modal distributions, respectively. To demonstrate that the method can scale to higher dimensions we finally consider filtering of a 20 dimensional linear spring-mass system. This should be compared to [3] and [4] in which 50 and 10 000 dimensional SPDEs and PDEs were solved with the deep splitting method. We stress that our example does not exhibit symmetry (the variables of solution are not permutation invariant) as in [3, 4] and is therefore challenging although it has fewer dimensions.

In Subsection 5.1 we describe the reference solutions that we use as benchmarks. In Subsection 5.2 we describe how the model is designed and trained. Performance metrics are presented in Subsection 5.3 and Subsection 5.4 contains our numerical results. In the final part, Subsection 5.5, we briefly discuss the model.

5.1. **Reference solution.** In our four examples we measure performance versus a benchmark solution. All our benchmarks are in discrete time, since for fixed time discretization we have a discrete system, see [35]. In the linear Gaussian case we use the Kalman Filter (KF), see, e.g., [27, 43]. This gives a recursive closed formula for the mean and covariance matrix of the Gaussian distribution that solves the filtering problem. Below we denote by μ_{KF} and p_{KF} the mean and the density of the Kalman filter.

In the nonlinear examples we employ a Particle Filter (PF) as the benchmark. More specifically, we use a bootstrap particle filter to estimate the filtering density at each time step [43]. To find a sufficient number of particles to run, we made an experiment where we looked at the variance of the estimated mean from the particle filter. This suggested using 100 000 particles to keep the standard deviation at approximately 0.01. In this section we denote by μ_{PF} and p_{PF} the mean and the density of the particle filter.

Both the Kalman filter and the particle filter require knowledge of μ, σ, h and p_0 similarly to our method, making it a fair comparison. If the problem consisted of unknown coefficients one would have to do inference over possible coefficients μ, σ, and h, which is out of the scope of this work. In such parameter estimation, filtering is an integral part.

5.2. **Model architecture, training and evaluation.** The regression task is one of the most common application of neural networks. In this paper we use a simple architecture consisting of a feed-forward fully connected network with 4 hidden layers with 100 neurons each. Each layer uses batch normalization and ReLU (Rectified Linear Units) activation functions. A general overview of deep learning is found in [44]. For details on batch normalization, see the original paper [26] and an investigation of why this is effective in [42]. In our energy-based approach we adapt the energy $f_n^{\theta_n}$ to each of our examples, see the corresponding Subsections 5.4.1–5.4.4.

The optimization is performed with the ADAM optimizer together with minibatches. The latter consists of using subsets of the training data to introduce randomness of the loss function; this is a small generalization of the true Stochastic Gradient Descent method (SGD) but with more efficient iterations. Details on SGD and minibatches can be found in [18]. The ADAM optimizer makes use of the momentum of the gradient from the previous iterations; details on this can be found in [29]. We use the suggested hyperparameters ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) from the original paper, except for the learning rate which is set to $\alpha = 10^{-5}$.

In this paper we have access to the parameters of the underlying processes and we can generate as many samples of $(Y_n, \widetilde{X}_n)_{n=0}^N$ as we desire. Recall that $\widetilde{X}$ and Y are independent. In practice it speeds up the training to limit the number of samples and reuse them in different epochs. We generate 1 million samples of each process, with the Euler–Maruyama method, and use these in minibatches to train the network. We use a rotation between different sizes of minibatches in the following order $(2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14})$ until the loss ceases to decrease. The rotation of sizes in minibatches creates different stochasticity in the optimizer, the larger minibatch size the less randomness. We train the network for 5-10 epochs on each batchsize before switching. The training goes on until validation error increases for 5 epochs in a row.

After the model is trained appropriately and we have a sequence $(\widehat{\theta}_n)_{n=1}^N$, approximating $(\theta_n^*)_{n=1}^N$, we want to estimate the normalized density and the mean of the density. In the low-dimensional case we can approximate the normalizing constant, for each observation sequence $Y_{0:n}$ and every $\widehat{\theta}_n$,

$$Z_n^{\widehat{\theta}_n}(Y_{0:n}) = \int_{\mathbb{R}^d} \Phi_n^{\widehat{\theta}_n}(x, Y_{0:n}) \,\mathrm{d}x$$

by the use of quadrature. This is not suitable in higher dimensions since these methods do not scale very well. In the high dimensional case it is more appropriate to use a Monte Carlo sampler, e.g., a Markov chain Monte Carlo such as the Metropolis–Hastings algorithm [24, 38] or a Hamiltonian Monte Carlo (HMC) method [9, 13]. Given the normalization constants we can evaluate the normalized density and the corresponding mean by

$$\begin{aligned} p_n^{\widehat{\theta}_n}(x, Y_{0:n}) &= \frac{\Phi_n^{\widehat{\theta}_n}(x, Y_{0:n})}{Z_n^{\widehat{\theta}_n}(Y_{0:n})}, \quad x \in \mathbb{R}^d, \\ \mu_n^{\widehat{\theta}_n}(Y_{0:n}) &= \int_{\mathbb{R}^d} x \, p_n^{\widehat{\theta}_n}(x, Y_{0:n}) \, \mathrm{d}x. \end{aligned} \tag{24}$$

In the high-dimensional case we make use of an HMC sampler to find the mean and normalizing constant of the approximation. Specifically, we use a step size of 0.1 and a final time of 1.0 in the leapfrog step of the sampler. More details on this method can be found in [9].

5.3. **Metrics.** In the previous section, finding the (unnormalized) density was emphasized. However, the most common estimates in filtering problems are the mode and the mean. In cases where the filtering density is symmetric and unimodal, or even Gaussian, these two coincide. In other applications one can argue about which is most relevant but in this context we opt to primarily measure the mean of the distribution.

In order to approximate the expectation of different metrics that we want to evaluate, we simulate M coupled state-observation sequences $(X, Y)^{(m)} = (X_n^{(m)}, Y_n^{(m)})_{n=0}^N$ for $m = 1, \ldots, M$. We calculate the mean $\mu_n^{(m)}$ and the density $p_n^{(m)}$ of the reference solution for each time step t_n, and sample index m. Similarly, we denote the normalized density and mean of a generic approximation by

$$\begin{aligned} \widehat{p}_n^{(m)}(x) &= \widehat{p}_n(x, Y_{0:n}^{(m)}), \quad x \in \mathbb{R}^d, \\ \widehat{\mu}_n^{(m)} &= \widehat{\mu}_n(Y_{0:n}^{(m)}). \end{aligned}$$

In the examples we evaluate these with our method and also with two standard methods that we consider as baselines for comparison. The first way of using the mean is to directly measure the Euclidean distance between the empirical mean of the approximation and the mean from the reference solution. We call this error the First Moment Error (FME). This metric is defined by

$$\mathrm{FME} = \frac{1}{M} \sum_{m=1}^{M} \big\| \mu_n^{(m)} - \widehat{\mu}_n^{(m)} \big\|, \quad \text{for } n = 1, \ldots, N. \tag{25}$$

The second way is to compare the mean directly with the true state X_n at each time step. This is a common measurement of how well an approximation performs [1, 15, 25, 37]. Let $\mathfrak{m}_n^{(m)}$ denote the mean, either from the true filter or from an approximation. This measurement is known in the literature as the Mean Absolute Error (MAE). However, it is not an actual error since we do not expect it to converge to zero. More precisely, in the evaluation we aim to achieve almost the same MAE value from our approximation as from the true filter. The metric is defined by

$$\mathrm{MAE} = \frac{1}{M} \sum_{m=1}^{M} \big\| X_n^{(m)} - \mathfrak{m}_n^{(m)} \big\|, \quad \text{for } n = 1, \ldots, N. \tag{26}$$

The final, and perhaps the most interesting measurement, is the Kullback–Leibler divergence between the entire filtering density and the one generated with our method. This shows how well we manage to capture the density in the whole domain. The distance consists of taking the expectation of the difference between the logarithms of the two distributions, with respect to one of the distributions. It is important to note that the divergence is not a metric since it is not symmetric. One can consider both the forward divergence, in which one takes the expectation with respect to the true distribution p, and the reverse divergence by taking the expectation with respect to the approximation $\widehat{p}$. The forward divergence is considered mean seeking and the reverse divergence is considered mode seeking [50]. In this paper we consider the forward divergence averaged over M

samples. We sample $x_n^{(k,m)}$ from $p_n^{(m)}$ for $k = 1, \dots, K$ and evaluate the averaged Kullback–Leibler Divergence (KLD) D_{KL} according to

$$\mathrm{KLD} = D_{\mathrm{KL}}(p_n \| \widehat{p}_n) = \sum_{m=1}^{M} \sum_{k=1}^{K} \log \left(\frac{p_n^{(m)}\big(x_n^{(k,m)}\big)}{\widehat{p}_n^{(m)}\big(x_n^{(k,m)}\big)} \right), \quad \text{for } n = 1, \dots, N. \tag{27}$$

In [48] this distance is used to measure how well the distributions match. It is also common to use the Kullback–Leibler divergence to calculate likelihood ratios in particle filters [37]. It can also be used directly as a loss function during training, such as in [21], where it is used for unconditional densities in a data driven manner.

In the examples in the next subsection we evaluate these metrics between our proposed approximation, which we denote by the Energy-Based Deep Splitting (EBDS), and the reference solution. For comparison we also evaluate these metrics on other approximate solutions. For the nonlinear examples we employ an Extended Kalman Filter (EKF) as a baseline [43]. We expect the EKF to yield decent estimates when the filtering density is unimodal. In the linear examples we use particle filters with fewer particles as baseline. These are less exact but are the most commonly used tool and thus an interesting comparison to our model.

5.4. **Examples.** In this subsection we test the performance of our model on three different underlying SDEs with $d = 1$ in Sections 5.4.1–5.4.3, and one with $d = 20$ in Section 5.4.4. For the linear examples we benchmark the model against the Kalman filter which provides the true solution. In the nonlinear examples we use the bootstrap particle filter. To simplify the comparison between the four examples we use the same constant time step $\Delta t = 0.01$ in all examples. In the first two we have $N = 100$ and final time $t_N = 1$, while in the last two, we have $N = 50$ and $t_N = 0.5$. In all one-dimensional examples the measurement function in (2) is linear and defined by

$$h(x) = \beta x, \quad x \in \mathbb{R}.$$

We set $\beta = 1$ and it is worth noting that this results in a very large observation noise. In [11] they have two linear examples with a factor $\beta = 90$ in the measurement function h, resulting in much smaller observation noise. For a similar setting, in [15] they opt for a factor $\beta = 5.5$ in most of their examples. Furthermore, we consider the same diffusion coefficient for X in all one-dimensional examples, given by

$$\sigma(x) = 1, \quad x \in \mathbb{R}.$$

Finally, we consider an initial density $p_0 = \mathcal{N}(0, 1)$ in all one-dimensional examples.

5.4.1. *Mean-reverting linear state equation.* Here we consider an underlying process (1) with drift coefficient defined by

$$\mu(x) = -x, \quad x \in \mathbb{R}.$$

The solution to (1) is an Ornstein–Uhlenbeck process. This process is mean reverting towards 0 and this can be seen in the underlying density. The density at time t, conditioned on $X_0 = x_0$, is given by $\mathcal{N}(x_0 e^{-t}, \frac{1}{2}(1 - e^{-2t}))$. With the prior $x_0 \sim \mathcal{N}(0, 1)$, the posterior density is given by $\mathcal{N}(0, \frac{1}{2}(1 + e^{-2t}))$.

In this linear example we use a neural network $\widetilde{f}_n^{\widehat{\theta}_n} \colon (x_n, y_{0:n}) \mapsto (x_n, \xi_1, \xi_2) \in \mathbb{R}^3$ and concatenate it with a layer specifically designed for the problem,

$$g(x_n, \xi_1, \xi_2) = \xi_1 + (x_n - \xi_2)^2 \mathbb{1}_{|x_n| > \xi_2}. \tag{28}$$

The energy function is then defined as $f_n^{\widehat{\theta}_n} = g \circ \widetilde{f}_n^{\widehat{\theta}_n}$. The second term is added to guarantee that the density is integrable with essentially Gaussian tails. In this way we build structure into the model, based on prior knowledge of the problem. This is further discussed in Section 5.5.

We employ our method and present the different metrics in the first column of Figure 2 over the 100 time steps. In Figure 2a and Figure 2d, we see the (unconditional) density of X and an illustration of a trajectory as well as the corresponding filter mean and estimated mean. In this example we also demonstrate the performance of a particle filter with 1000 (PF-1000) particles as a baseline for comparison. In Figure 2g we present the MAE (26) with $\mathfrak{m}$ given by the Kalman filter as well as with our method and PF-1000. In Figure 2j we see the FME, the difference between the true mean of the Kalman filter and the empirical mean of our model, measured as in (25) with $\mu = \mu_{\text{KF}}$. We also present the FME between the mean from PF-1000 and the true mean. Finally in Figure 2m the KLD (27) is presented. We can see that the model performs well with respect to the MAE but slowly lose accuracy over time. Similarly the FME and KLD show increasing error over time but decent results compared to PF-1000. In the example trajectory in Figure 2d we can see that the mean of our method follows the true mean very closely in the beginning but loses accuracy toward the final time.

We present our approximation for one arbitrarily chosen observation sequence in Figure 3. This illustration was inspired by the figures in [11]. In Figure 3a we see the time evolution of the density in blue. On the right, in Figure 3b–3d, we see snapshots at three different times of the density compared to the true density given by the Kalman filter.

5.4.2. *Mean-reverting nonlinear state equation.* Here we consider an underlying SDE (1), where the drift coefficient is defined by

$$\mu(x) = -x - x^3, \quad x \in \mathbb{R}.$$

This process has roughly a similar statistics as the linear process of Subsection 5.4.1. More precisely, we have a process that is mean reverting towards the long term mean 0. The cubic term increases the strength of the mean reversion, compared to the Ornstein–Uhlenbeck process. This results in a more narrow underlying distribution with smaller tails. One could argue that this problem constitutes a setting for which the EKF is suitable since the distribution is unimodal and symmetric, and hence we use it as a baseline comparison. We use a neural network with the same structure as for the previous example, defined in (28).

We demonstrate our method in the second column of Figure 2. The figures follows the same format as for the linear example. In Figure 2b wee see the density of X at the final time $T = 1$. In Figure 2e we see one trajectory of X together with the corresponding filter estimates. For the metrics MAE, FME and KLD we use a bootstrap particle filter with 100 000 particles as our benchmark. In addition to evaluating the metrics for our method, we also demonstrate the performance of the EKF. These errors are presented in Figure 2h, 2k and 2n. Clearly, in all three metrics our model performs better than the EKF. It is also interesting to note that in this example we reach a steady error level over time in both FME and KLD.

Similarly to the previous example we present the approximated density in the whole domain for a single observation sequence in Figure 4. In these figures the reference solution is given by a particle filter, presented in the snapshots of Figure 4b–4d, in red.

5.4.3. *Bistable nonlinear state equation.* In our final example we consider an underlying SDE (1) with the drift coefficient

$$\mu(x) = \frac{2}{5}(5x - x^3), \quad x \in \mathbb{R}.$$

Compared to the mean reverting setting, the two terms have different signs here. This results in two attracting equilibria, positioned symmetrically around 0, creating a bimodal underlying distribution. Compared to the mean reverting example we expect the extended Kalman filter to perform poorly in this setting.

In this bimodal example we instead let the neural network be defined by $\widetilde{f_n^{\theta_n}} \colon (x_n, y_{0:n}) \mapsto (x_n, \xi_1, \xi_2, \xi_3, \xi_4, \xi_5) \in \mathbb{R}^6$ and concatenate it with a different layer g defined as

$$g(x_n, \xi_{1:5}) = \xi_1 + \xi_2(x_n - \xi_3)^2 \mathbb{1}_{x_n < \xi_3} + \xi_4(x_n - \xi_5)^2 \mathbb{1}_{x_n > \xi_5}.$$

The energy function is defined as before by the concatenation $\widehat{f_n^{\theta_n}} = g \circ \widetilde{f_n^{\theta_n}}$. The second and third term guarantee that the density is integrable. Compared to the architecture for the linear example (28), we let the exponential decay from these extra terms be non-symmetrical around 0. We have also introduced ξ_2 and ξ_4 to increase the flexibility of the model; both of these are defined as non-negative outputs to guarantee that the model does not explode.

In the right column of Figure 2 we present the metrics of our method employed on this bistable SDE. We compare the performance of our model to that of the EKF which we believe to yield poorer approximations in this setting when the filtering density might not be unimodal. In Figure 2c, we see the density of X at the final time $T = 0.5$. In Figure 2i, 2l and 2o we see that our method outperforms the EKF in all metrics. In the MAE we see a similar performance for our model as for the PF. We also note that the FME and KLD increase almost linearly with time.

Similarly to earlier examples we illustrate the approximation of the filtering density over time for a single observation sequence. This is seen in Figure 5, where on the right we see snapshots of the approximation as well as the reference given by the PF. In the last snapshot we can see how our method starts to lose accuracy when compared to the PF but still achieves favorable result to the EKF.

5.4.4. *Linear spring-mass.* In this example we consider filtering of a high-dimensional equation. Consider a mechanical system with M masses connected in series with springs and dampers. They move friction-less and without gravity. The state of the system consists of displacements (from equilibrium points) and velocities. The spring forces are linear and the resulting system of ordinary differential equations is linear. We denote the masses m_i, $i = 1, \dots, M$, the stiffness and damping constants, k_i and c_i, $i = 1, \dots, M+1$, respectively. In this example we consider a system perturbed by noise, e.g., stemming from vibrations. This results in a $2 \times M$ dimensional SDE where the first M dimensions represent the displacements and the last M dimensions represent the velocities. The equation is given by

$$X_t = X_0 + \int_0^t \begin{bmatrix} 0_{M\times M} & I_{M\times M} \\ A_{21} & A_{22} \end{bmatrix} X_s \,\mathrm{d}s + \int_0^t \begin{bmatrix} \sigma_1 I_{M\times M} & 0_{M\times M} \\ 0_{M\times M} & \sigma_2 I_{M\times M} \end{bmatrix} \mathrm{d}W_s,$$

where

$$A_{21} = \begin{pmatrix} -\frac{k_1+k_2}{m_1} & \frac{k_2}{m_1} & & \\ \frac{k_2}{m_2} & \ddots & \ddots & \\ & \ddots & \ddots & \frac{k_M}{m_{M-1}} \\ & & \frac{k_M}{m_M} & -\frac{k_M+k_{M+1}}{m_M} \end{pmatrix}, \quad A_{22} \quad = \begin{pmatrix} -\frac{c_1+c_2}{m_1} & & \\ & \ddots & \\ & & -\frac{c_M+c_{M+1}}{m_M} \end{pmatrix}.$$

In our example we consider identical masses, $m_i = 1$, stiffness constants $k_i = 5$ and damping constants $c_i = 0.1$. We also choose the diffusion constants $\sigma_1 = \sigma_2 = 1$. Thus we have the same level of noise here as in the previous examples. Note that in the setting of this paper we have $d = 2M$ and we consider a selection of linear measurements of displacements and velocities. We let the number of masses be $M = 10$, and we define an observation process Y in (2), with $d' = 10$, by a measurement function $h : \mathbb{R}^{20} \to \mathbb{R}^{10}$ with $h(x) = Hx$, where H is the 10×20 matrix with zero entries except for $H_{i,2i-1} = H_{j,2j} = 1$ for $i = 1, 2, 3, 4, 5$ and $j = 6, 7, 8, 9, 10$. In other words, we measure five positions and five velocities.

For this example we used a different architecture for the model. Let the input x_n to the network be split as $x_n = (x_n^{\mathrm{pos}}, x_n^{\mathrm{vel}})$, where the two parts represent the 10-dimensional vectors of displacements and velocities, respectivly. Let $\widetilde{f_n^{\widehat{\theta}_n}} : (x_n, y_{0:n}) \mapsto (x_n^{\mathrm{pos}}, x_n^{\mathrm{vel}}, \alpha, \xi_1, \xi_2, \beta_1, \beta_2, \lambda_1, \lambda_2)$ define the model, where $(\alpha, \xi_1, \xi_2, \beta_1, \beta_2, \lambda_1, \lambda_2) \in \mathbb{R}^{1\times M\times M\times 1\times 1\times 1\times 1}$. The intuition behind this model is that α fits the training data with high flexibility. The other outputs are defined to handle domains outside of the support of the training data similarly to the previous examples. More precisely (ξ_1, ξ_2) define means and (β_1, β_2) variances for the Gaussian tails outside some domain defined by λ_1 and λ_2. The network $\widetilde{f_n^{\widehat{\theta}_n}}$ is concatenated with

$$\begin{aligned} g(x_n^{\mathrm{pos}}, x_n^{\mathrm{vel}}, &\alpha, \xi_1, \xi_2, \beta_1, \beta_2, \lambda_1, \lambda_2) = \\ &\alpha + \beta_1 \|x_n^{\mathrm{pos}} - \xi_1\|^2 \mathbb{1}_{\|x_n^{\mathrm{pos}} - \xi_1\|^2 > \lambda_1} + \beta_2 \|x_n^{\mathrm{vel}} - \xi_2\|^2 \mathbb{1}_{\|x_n^{\mathrm{vel}} - \xi_2\|^2 > \lambda_2}. \end{aligned}$$

The energy function is then defined as $f_n^{\widehat{\theta}_n} = g \circ \widetilde{f_n^{\widehat{\theta}_n}}$. In this example we construct the network based on knowledge of the problem and let the model learn different tail properties for the densities of the positions and of the velocities.

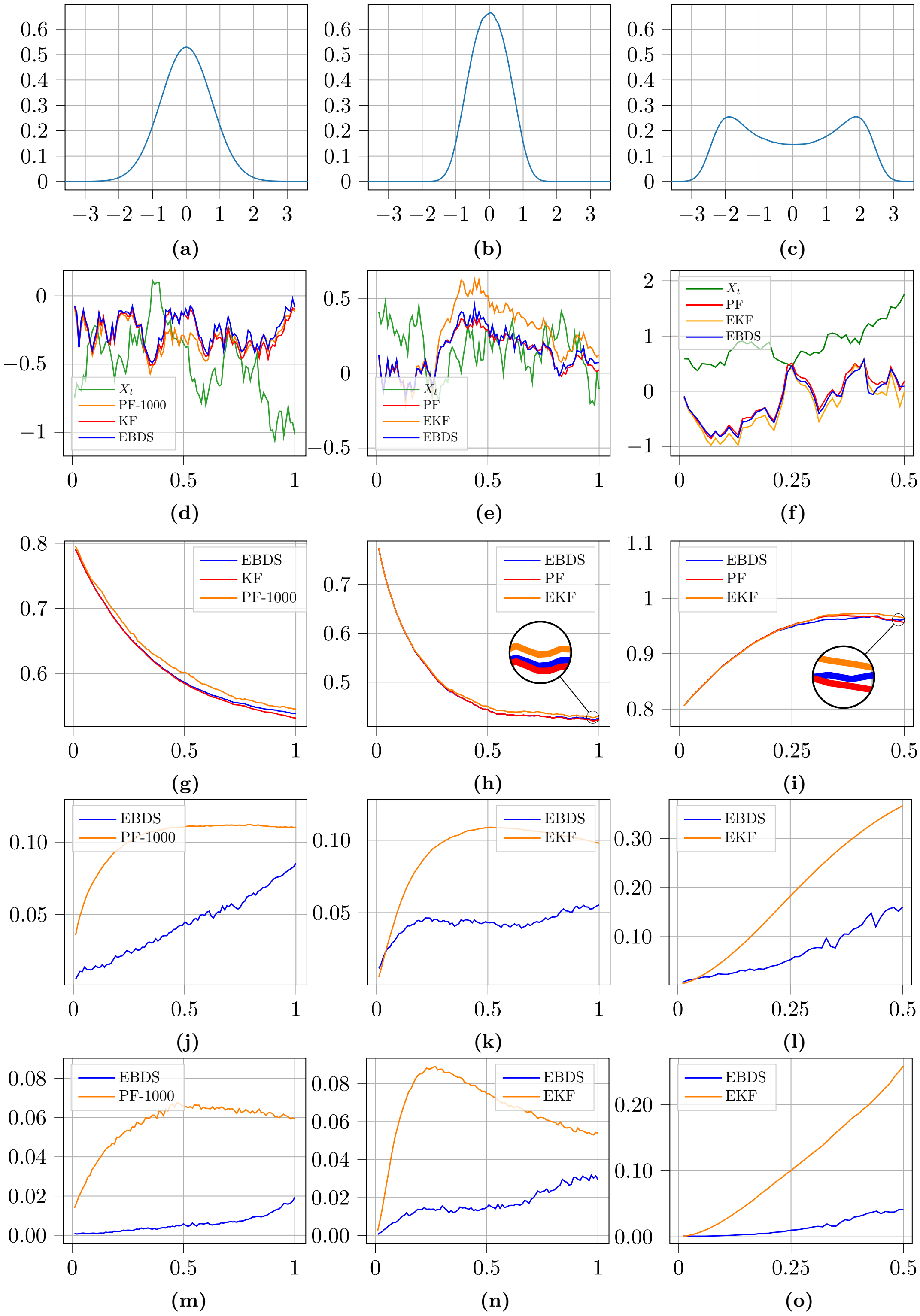
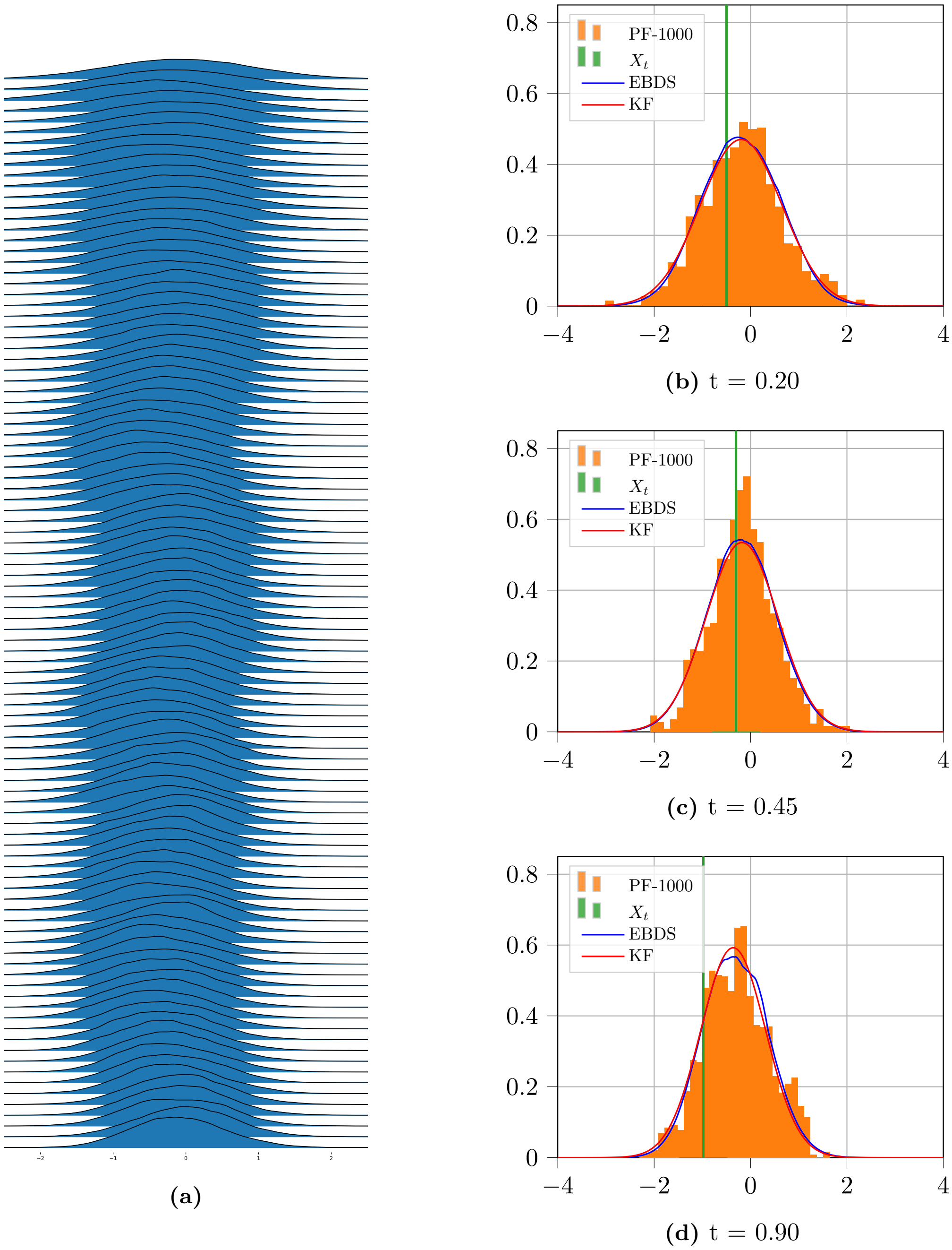


Figure 2. The figure presents numerical results for the three examples with respect to time. Left to right: Linear, mean reverting and bistable example. Top to bottom: Underlying densities of X_T, example trajectories, MAE, FME and KLD. Our method (EBDS) is illustrated in blue, the reference solution in red and a baseline in orange.



(a)

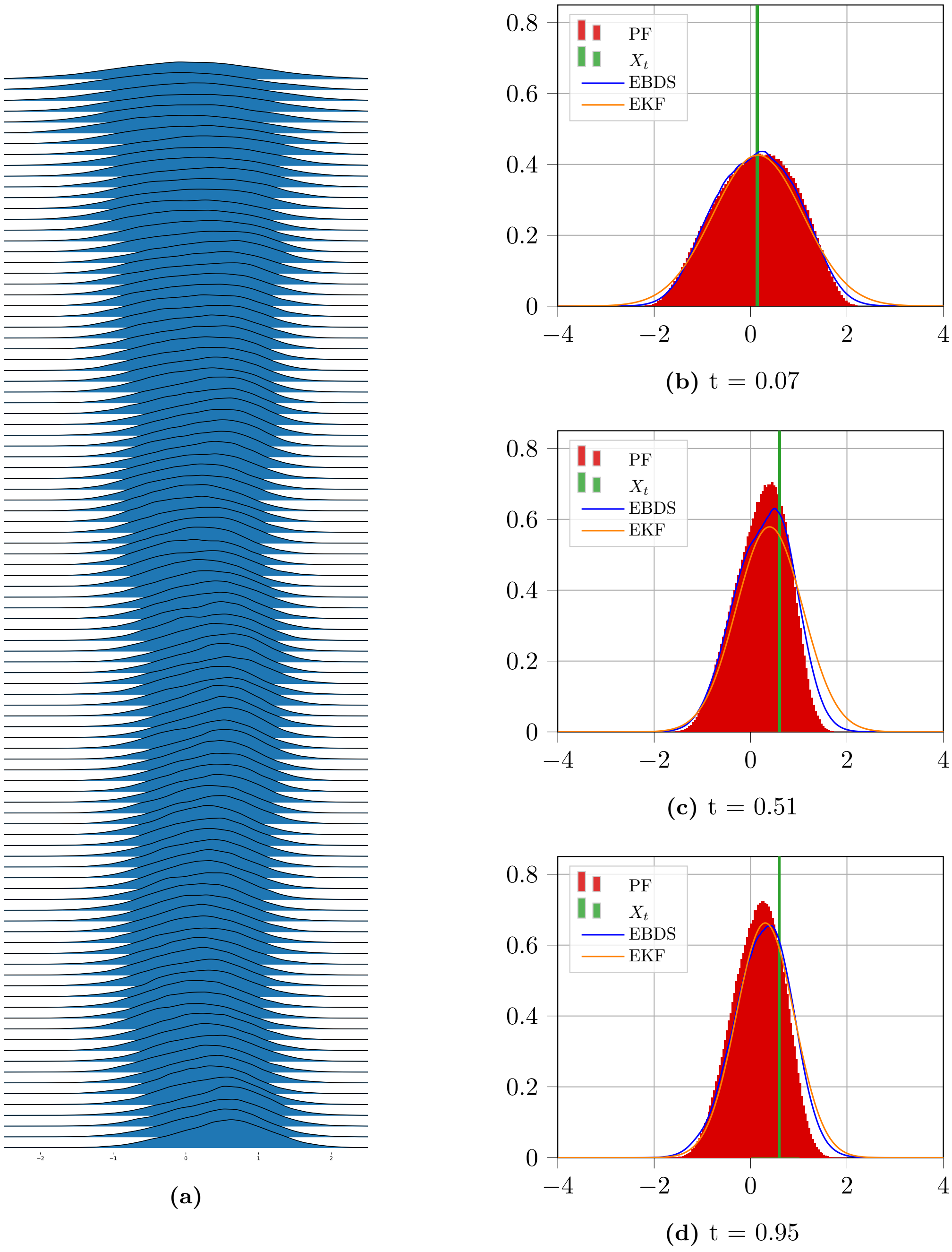
(b) t = 0.20

(c) t = 0.45

(d) t = 0.90

Figure 3. In (a) we see the time evolution of the density given by our model in blue, from time $t = 0.01$ at the top to $t = 1.00$ at the bottom. (b)–(d) are snapshots of the true filtering density given by the KF in red, the density from our model (EBDS) in blue, the PF-1000 in orange as well as the true state X_t in green.

In Figure 6 we present the metrics of our method employed on this linear spring-mass example. We compare it directly to the Kalman filter which provides the true solution. As in the previous examples we demonstrate the performance with particle filters with fewer particles. This time we employ three different accuracies to understand the performance by comparisons. By doing this we encapsulate the performance of our method for each metric. The performances in MAE and FME are, considering the low number of particles in the comparisons, less satisfactory than for the



(a)

(b) t = 0.07

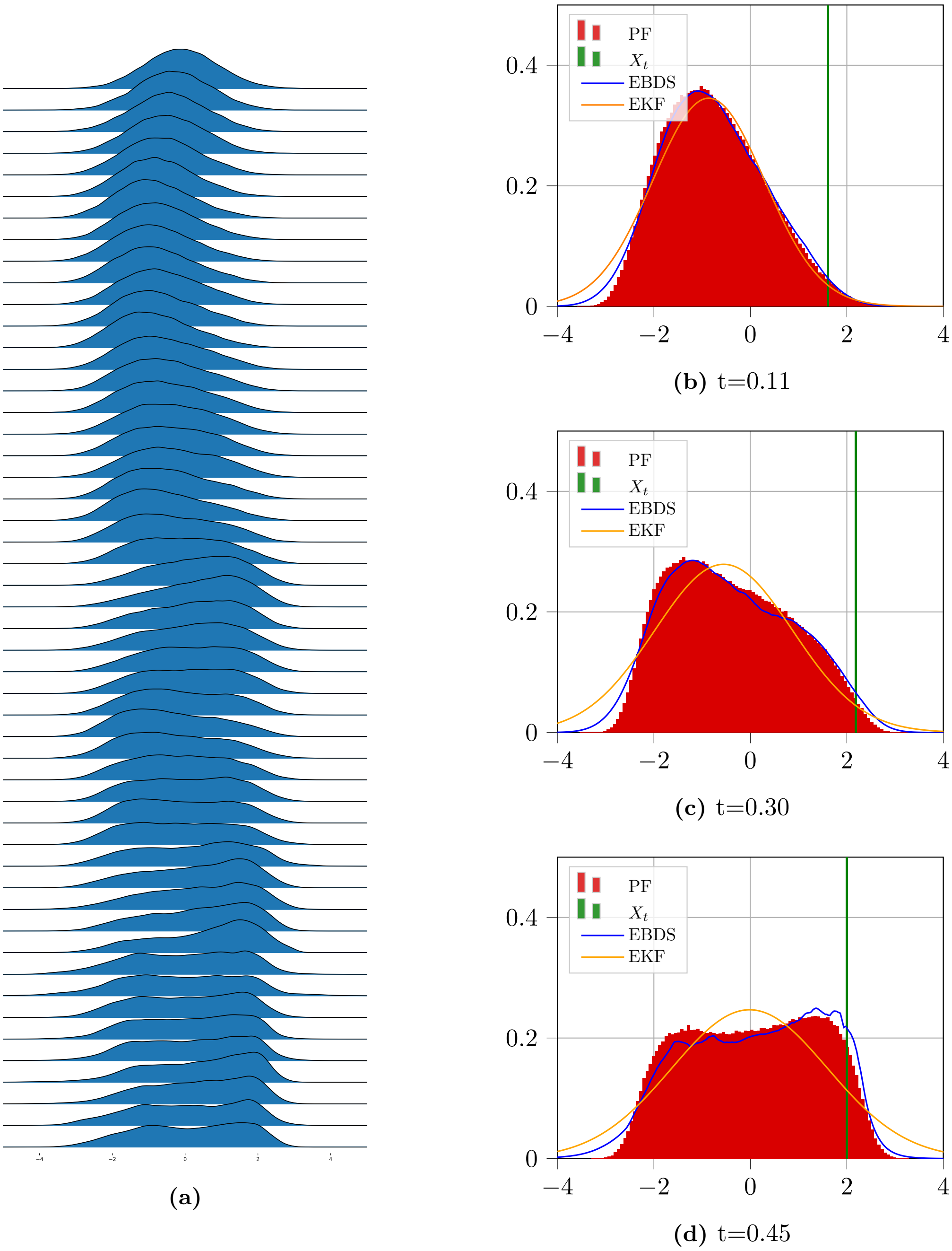
(c) t = 0.51

(d) t = 0.95

Figure 4. In (a) we see the time evolution of the density given by our model in blue, from time $t = 0.01$ at the top to $t = 1.00$ at the bottom. (b)–(d) are snapshots of the true filtering density given the PF in red, the density from our model (EBDS) in blue and the EKF in orange as well as the true state X_t in green.

one-dimensional examples. The performance KLD on the other hand is much better, in particular for the final time.

To also evaluate the method qualitatively, we compare the marginal densities for a single observation sequence for the different methods. In Figure 7 we illustrate four different marginal densities of our method compared to the KF and a PF with 1000 particles. This is to demonstrate some of the qualities of the model and should not be used as a reference of how well the method performs more generally. Clearly the model manages to learn Gaussian tails outside of the domain of the training data. The model is also flexible enough to match the variance of the Kalman filter. In



(a)

(b) t=0.11

(c) t=0.30

(d) t=0.45

Figure 5. In (a) we see the time evolution of the density given by our model in blue, from time $t = 0.01$ at the top to $t = 0.50$ at the bottom. (b)–(d) are snapshots of the true filtering density given by the PF in red, the density from our model (EBDS) in blue and the EKF in orange as well as the true state X_t in green.

this figure one can also observe that the model fits the velocities slightly better than the positions. This behaviour was observed for most observation trajectories, which is not shown here.

5.5. **Discussion.** The approximation method depends on the splitting of the SPDE (5). This is a crucial step in the derivation and we have not analysed the error caused by this approximation. Furthermore, the approximation of q in (6) and the stochastic processes $(X, Y, \widetilde{X})$ yield discretization errors. This means that in each local optimization step we commit an error, even if a global optimum was to be found. Looking at the Kullback–Leibler divergence for the three one

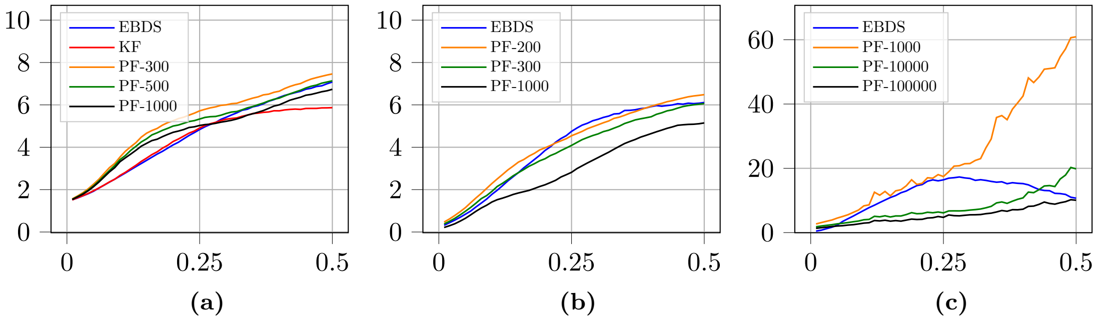


Figure 6. The figure presents numerical results for the LSM example. Left to right: MAE, FME and KLD. Our method (EBDS) is shown in blue, the reference solution (KF) in red, and particle filters with different accuracies in orange, green and black.

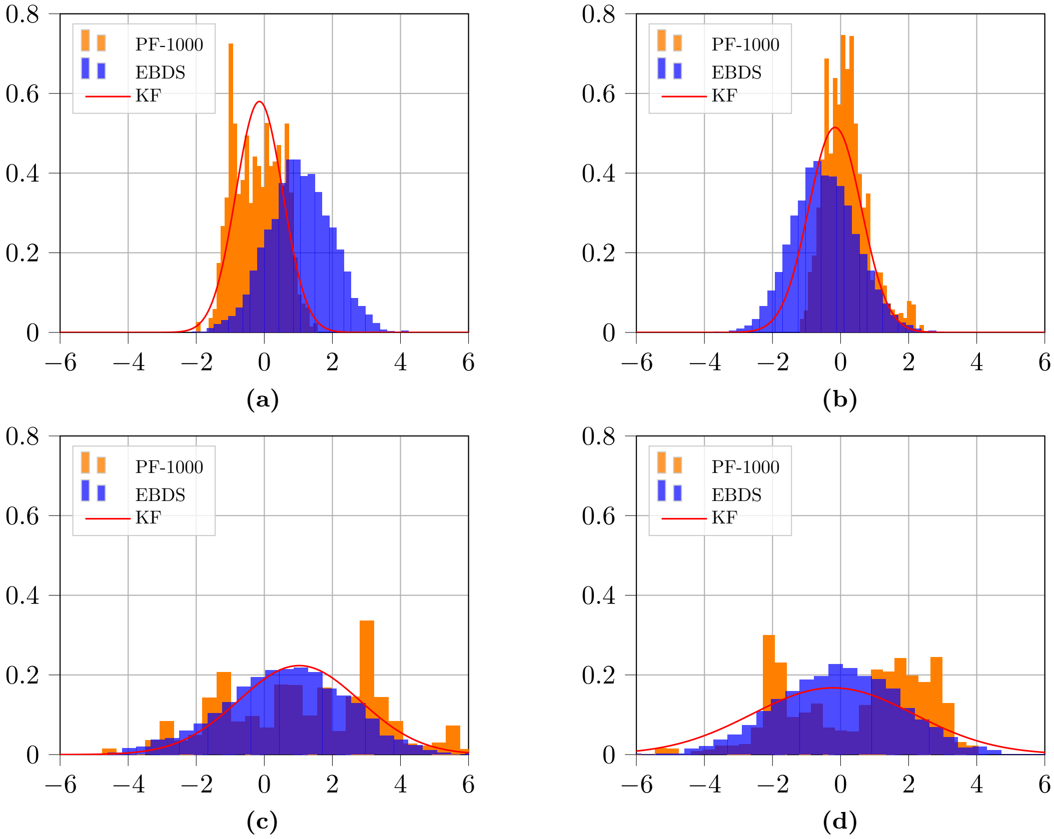


Figure 7. The figure shows the marginal densities, for a single observation sequence, of the Kalman filter in red, our method (EBDS) in blue and a particle filter with 1000 particles in orange. The left column shows dimensions that have been observed and the right column shows dimensions which are not observed. The top row contains positions and the bottom row shows velocities.

dimensional examples, we see roughly a linear increase in the error over time. Similar behaviour was observed in [4] for the approximation of PDEs. We believe that this is mostly due to the accumulation of local errors from each time step t_n. In a second part of this study we will do an error analysis of the different approximations employed here.

For the first few time steps the training was a simple task and convergence was fast. This is likely due to the simplicity of the problem, i.e., the filtering distribution has not yet deviated much from the given initial distribution, and also the observations are fewer. At later time steps, especially for the bistable example, the sought density looks very different for each observation sequence. For some sequences the density is unimodal throughout every time step, while for others it might create a bimodal density such as the one seen toward the final time in Figure 5a. In the bistable example we chose to stop at $N = 50$ time steps because of the increased difficulty of training the models. Similar difficulty was found in the linear spring-mass example. In particular, the networks were repeatedly retrained until a satisfactory solution was found.

In the different examples we also incorporated additional architecture into the model adapted to the dynamics of the particular example. The main idea with this is to guarantee that we obtain a function that can be normalized by letting the function go to 0 outside of the support of the training data. When comparing to a standard $\mathbb{R}$-valued output, which potentially could yield satisfactory results with respect to the mean but most likely not in the KLD, one can expect better results with respect to the KLD with this additional architecture while keeping a good approximation of the mean. Researchers who seek a problem-agnostic method might find this unsatisfactory, but we find it a sound approach to understand the problem at hand and utilize structure from it, known from theory or gained from simulations.

The purpose of our development of the method is to get a filter that scales better than particle filters. While the latter are performing a better inference with sufficiently many particles, the neural network of EBDS, after training, are orders of magnitudes faster and there is a marginal difference in computational time for 1 or 20 dimensions. This was also demonstrated in [3, 4]. In fact, the number of particles required for a particle filter scales exponentially in the state dimension [45]. Since we have not optimized our code, or have a machine for which we have control of the background processes, a direct comparison of computational times would not do the methods justice. Our work is a first small step towards non-linear filters in high dimensions, based on partial differential equations and applicable in a real time setting.

Acknowledgements. We are grateful to Oskar Eklund, Moritz Schauer and Kristoffer Andersson for useful input during the writing of this paper. We also want to thank the anonymous reviewers for very insightful suggestions on both theoretical and practical matters. The work of K.B. and S.L. was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Y. T. Bai, X. Y. Wang, X. B. Jin, Z. Y. Zhao, and B. H. Zhang. A neuron-based Kalman filter with nonlinear autoregressive model. *Sensors*, 20(1):299, 2020.

[2] A. Bain and D. Crisan. *Fundamentals of Stochastic Filtering.* Springer, 2009.

[3] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld. Deep learning based numerical approximation algorithms for stochastic partial differential equations and high-dimensional nonlinear filtering problems. *arXiv:2012.01194*, 2020.

[4] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld. Deep splitting method for parabolic PDEs. *SIAM J. Sci. Comput.*, 43(5):A3135–A3154, 2021.

[5] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving the Kolmogorov PDE by means of deep learning. *arXiv:1806.00421v2*, 2021.

[6] V. E. Beneš. Exact finite-dimensional filters for certain diffusions with nonlinear drift. *Stochastics*, 5(1-2):65–92, 1981.

[7] S. S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Artech House Publishers, 1999.

[8] D. Brigo and B. Hanzon. On some filtering problems arising in mathematical finance. *Insurance: Mathematics and Economics*, 22(1):53–64, 1998.

[9] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo.* CRC press, 2011.

[10] F. Cassola and M. Burlando. Wind speed and wind energy forecast through Kalman filtering of numerical weather prediction model output. *Applied Energy*, 99:154–166, 2012.

[11] D. Crisan, A. Lobbe, and S. Ortiz-Latorre. An application of the splitting-up method for the computation of a neural network representation for the solution for the filtering equations. *arXiv:2201.03283*, 2022.

[12] P. Date and K. Ponomareva. Linear and non-linear filtering in mathematical finance: a review. *IMA J. Manag. Math.*, 22(3):195–211, 2011.

[13] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

[14] L. Duc, T. Kuroda, K. Saito, and T. Fujita. Ensemble Kalman filter data assimilation and storm surge experiments of tropical cyclone nargis. *Tellus A: Dynamic Meteorology and Oceanography*, 67(1):25941, 2015.

[15] R. Frey, T. Schmidt, and L. Xu. On Galerkin approximations for the Zakai equation with diffusive and point process observations. *SIAM J. Numer. Anal.*, 51(4):2036–2062, 2013.

[16] A. Friedman. *Partial Differential Equations of Parabolic Type.* Prentice Hall, Inc., 1964.

[17] A. Friedman. *Stochastic Differential Equations and Applications : vol.I.* Academic Press, 1975.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT press, 2016.

[19] I. Goodman, R. Mahler, and H. T. Nguyen. *Mathematics of Data Fusion*, volume 37. Springer Science & Business Media, 1997.

[20] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. Energy-based models for deep probabilistic regression. In *European Conference on Computer Vision*, pages 325–343. Springer, 2020.

[21] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. How to train your energy-based model for regression. *arXiv:2005.01698*, 2020.

[22] I. Gyöngy and N. Krylov. On the rate of convergence of splitting-up approximations for spdes. In *Stochastic inequalities and applications*, pages 301–321. Springer, 2003.

[23] I. Gyöngy and N. Krylov. On the splitting-up method and stochastic partial differential equations. *The Annals of Probability*, 31(2):564–591, 2003.

[24] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 1970.

[25] J. N. Hendriks, F. K. Gustafsson, A. H. Ribeiro, A. G. Wills, and T. B. Schön. Deep energy-based narx models. *IFAC-PapersOnLine*, 54(7):505–510, 2021.

[26] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

[27] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *J. Basic Eng.*, 1961.

[28] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Adv. Neural Inf. Process. Syst.*, 30, 2017.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[30] A. Klenke. *Probability Theory: A Comprehensive Course.* Springer Science & Business Media, 2013.

[31] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Springer, Berlin, Heidelberg, 1992.

[32] H. J. Kushner. On the differential equations satisfied by conditional probablitity densities of Markov processes, with applications. *J. Soc. Industrial Appl. Math., Series A: Control*, 2(1):106–119, 1964.

[33] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Adv. Neural Inf. Process. Syst.*, 30, 2017.

[34] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 1(0), 2006.

[35] F. L. Lewis, L. Xie, and D. Popa. *Optimal and robust estimation: with an introduction to stochastic control theory.* CRC press, 2017.

[36] A. Lobbe. Deep Learning for the Beneš filter. *arXiv:2203.05561*, 2022.

[37] M. Mansouri, H. Nounou, and M. Nounou. Kullback-Leibler divergence-based improved particle filter. In *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, pages 1–6. IEEE, 2014.

[38] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[39] B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications.* Springer Science & Business Media, 2003.

[40] J. Quinn. A high-dimensional particle filter algorithm. *arXiv:1901.10543*, 2019.

[41] W. Rutzler. Nonlinear and adaptive parameter estimation methods for tubular reactors. *Ind. Eng. Chem. Res.*, 26(2):325–333, 1987.

[42] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? *Adv. Neural Inf. Process. Syst.*, 31, 2018.

[43] S. Särkkä. *Bayesian Filtering and Smoothing.* Cambridge University Press, 2013.

[44] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[45] C. Snyder, T. Bengtsson, and M. Morzfeld. Performance bounds for particle filters using the optimal proposal. *Monthly Weather Review*, 143(11):4750–4761, 2015.

[46] Y. Song and D. P. Kingma. How to train your energy-based models. *arXiv:2101.03288*, 2021.

[47] Y. Xu, H. Zhang, Y. Li, K. Zhou, Q. Liu, and J. Kurths. Solving Fokker–Planck equation using deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013133, 2020.

[48] K. Yeo and I. Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *J. Comput. Phys.*, 376:1212–1231, 2019.

[49] M. Zakai. On the optimal filtering of diffusion processes. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 11(3):230–243, 1969.

[50] M. Zhang, T. Bird, R. Habib, T. Xu, and D. Barber. Variational f-divergence minimization. *arXiv:1907.11891*, 2019.

Kasper Bågmark, Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE–412 96 Gothenburg, Sweden

Email address: bagmark@chalmers.se

Adam Andersson, Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, S–412 96 Gothenburg, Sweden, and Saab AB Radar Solutions, S–412 76 Gothenburg, Sweden

Email address: adam.andersson@chalmers.se

Stig Larsson, Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE–412 96 Gothenburg, Sweden

Email address: stig@chalmers.se