# NEURAL LIKELIHOOD SURROGATES FOR PARAMETER INFERENCE VIA LOG-DENSITY PDE

**Kasper Bågmark***
Department of Mathematical Sciences
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden
bagmark@chalmers.se

**Filip Rydin***
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
filipry@chalmers.se

## ABSTRACT

We propose a two-stage method for parameter inference in discretely observed stochastic differential equations. First, we extend a deep log-density filter to learn parameter-dependent conditional log-densities. Second, we learn a neural surrogate for the Bayesian update's log-normalizing constants, yielding a fast differentiable approximation of the log-likelihood. On a 10-dimensional Ornstein–Uhlenbeck model and a nonlinear bistable equation, we obtain smooth posteriors and orders of magnitude faster likelihood evaluation than particle- and Kalman-type baselines at comparable accuracy.

## 1 INTRODUCTION

Nonlinear filtering concerns the inference of a latent stochastic state from noisy, partial observations, and plays a central role in statistics, control, and data assimilation (Särkkä & Svensson, 2023; Maybeck, 1979; Apte et al., 2008). In continuous time, filtering can be formulated as a coupled system consisting of a Fokker–Planck equation for prediction and a Bayesian update at discrete observation times. While this formulation is exact, it is computationally intractable in high dimensions due to the curse of dimensionality, which affects both classical Bayesian methods (e.g., particle filters (Chopin, 2004; Snyder et al., 2015)) and traditional numerical Partial Differential Equation (PDE) solvers such as finite differences and finite elements (Brenner & Scott, 2008).

Recent advances in scientific machine learning have enabled learning-based approaches to high-dimensional PDEs, including physics-informed neural networks (Raissi et al., 2019) and neural operator methods (Kovachki et al., 2023). In the context of filtering, stochastic representations of PDEs via deep Backward Stochastic Differential Equation (BSDE) methods are particularly attractive, as they scale naturally with dimension through simulation rather than spatial discretization (E et al., 2017; Han & Long, 2020).

A log-density reformulation of the filtering equations yields a stable parabolic PDE that can be solved via a deep BSDE scheme, enabling accurate state estimation in strongly nonlinear regimes (Bågmark & Rydin, 2025). This work extends this perspective to joint state-parameter inference by learning a parametric family of solutions to a conditional PDE. A two-stage learning pipeline is considered: first, a deep BSDE-based solver learns the conditional log-filter; second, a neural surrogate is trained for the log-normalizing constants in the Bayesian update. This provides a differentiable approximation of the log-likelihood, enabling scalable parameter inference via gradient-based optimization and Markov Chain Monte Carlo (MCMC).

**Problem formulation:** We consider the joint parameter-state estimation problem for applications in the online setting, where we continuously want to do new inference over new data sequences, corresponding to observations of an underlying signal. To set the notation we consider the underlying state space $\mathbb{R}^d$, and let a $p$-dimensional parameter space and $d' \times k$-dimensional space of observation sequences be defined by $\Theta = \mathbb{R}^p$ and $\mathbb{O}_k = \mathbb{R}^{d' \times k}$.

*equal contribution

Consider the $d$-dimensional signal $S = (S_t)_{t\in[0,T]}$ being a solution to the Stochastic Differential Equation (SDE) given by

$$S_t = S_0 + \int_0^t b(S_s, \theta)\,\mathrm{d}s + \int_0^t g(S_s, \theta)\,\mathrm{d}B_s, \tag{1}$$

where $B$ is a $d$-dimensional Brownian motion, $b\colon \mathbb{R}^d \times \Theta \to \mathbb{R}^d$, and $g\colon \mathbb{R}^d \times \Theta \to \mathbb{R}^{d\times d}$, and $S_0 \sim \pi_0(\theta)$. This time-homogeneous SDE has a unique strong solution under standard assumptions on the coefficients $b$ and $g$ with Lipschitz continuity and linear growth conditions (Øksendal, 2003).

The observation process $O = (O_k)_{k=1}^K$ is a $d'$-dimensional discrete random process depending on the measurement function $h\colon \mathbb{R}^d \times \Theta \times \mathbb{R}^q \to \mathbb{R}^{d'}$ defined by

$$O_k = h(S_{t_k}, \theta, \xi_k), \quad k = 1, \ldots, K, \tag{2}$$

where $\xi_k$ is a $q$-dimensional random variable inducing measurement noise. Assuming we know $h$, we can derive the likelihood $L$ of an observation given a state and a parameter $L\colon \mathbb{R}^{d'} \times \mathbb{R}^d \times \Theta \to \mathbb{R}$.

In this work we provide a novel method to estimate the probability density of the state $x$ conditioned on both a parameter and an observation sequence. The conditional object is denoted by $p_k(t_k, x \mid o_{1:k}, \theta)$. Equivalently, one may work with the negative log-density

$$v_k(t, x \mid o_{1:k}, \theta) = -\log p_k(t, x \mid o_{1:k}, \theta),$$

which satisfies a recursive system consisting of a parabolic PDE between observations, on $[t_k, t_{k+1}]$, and a Bayesian update at the observation times $t_k$, $k = 1, \ldots, K$. This system is stated precisely in Appendix A.

The goal of this work is to approximate this family of conditional PDE solutions for all $\theta \in \Theta$ and $o_{1:K} \in \mathbb{O}_K$, and thereby enable scalable joint state-parameter inference. Concretely, our contributions are:

- We introduce a novel deep parameter estimator for the joint state-parameter estimation problem by extending a deep density filtering method.
- We propose a neural surrogate for the normalizing constant that is differentiable and parallel, allowing for efficient parameter inference used online for new observation sequences.

## 2 Deep parameter estimation

In this section we outline the method consisting of a learning pipeline and an inference pipeline.

**Training in two steps:** In Bågmark & Rydin (2025) a deep density method is introduced that is designed for online state estimation for discretely observed SDEs with a fixed, known parameter $\theta$. Here we extend this framework to the joint state–parameter estimation problem. For each fixed $\theta$, the log-density PDE from Appendix A can be reformulated as a Forward Backward SDE (FBSDE) using a nonlinear Feynman–Kac representation (El Karoui et al., 1997; Pardoux & Peng, 1992). Following the deep BSDE methodology (E et al., 2017), this FBSDE is solved approximately by an Euler–Maruyama discretization and cast as a shooting-type optimization problem in which one learns an initial value function and its gradient. The filter is abbreviated LogBSDEF.

In this work, this formulation is generalized to functions of $(x, o_{1:k}, \theta) \in \mathbb{R}^d \times \mathbb{O}_k \times \Theta$. Parameterizing these functions with neural networks yields a trainable approximation of the conditional log-filter $v_k(t_k, x, o_{1:k}, \theta)$. The precise discrete BSDE optimization problem is given in Appendix B.

While this procedure yields an approximation of the conditional log-filter $\overline{v}_k(x, o_{1:k}, \theta) \approx v_k(t_k, x \mid o_{1:k}, \theta)$, it still requires repeated evaluation of the log-normalizing constant

$$C(o_{1:k}, \theta) = \log\left(\int_{\mathbb{R}^d} \exp\big(-v_{k-1}(t_k, z, o_{1:k-1}, \theta)\big) L(o_k, z, \theta)\,\mathrm{d}z\right). \tag{3}$$

which is computationally expensive. In Stage II we therefore train a neural surrogate $\Phi_k\colon \mathbb{O}_k \times \Theta \to \mathbb{R}$ such that $\Phi_k(o_{1:k}, \theta) \approx C(o_{1:k}, \theta)$. This is done by generating training data through direct, but

**Algorithm 1** Training pipeline

Input: $\pi_0(\theta)$, $b(\theta)$, $g(\theta)$, $h(\theta)$, and prior $p(\theta)$
Initialize $\overline{v}_0 = -\log \pi_0$
**for** $k = 0, \dots, K-1$ **do**
  **Stage I: learn conditional log-filter.**
  Initialize neural networks $\big(u, (\overline{w}_n)_{n=0}^{N-1}\big)$
  **for** $\ell = 0, \dots, L-1$ **do**
    Sample minibatch of parameters $\theta^{(m)} \sim p(\theta)$
    Sample $(\mathcal{X}_n^{k,(m)})_{n=0}^N$ and $O_{1:k}^{(m)}$ under $\theta^{(m)}$
    Evaluate prediction: $\mathcal{Y}^{k,(m)}$, and target: $\overline{v}_k\big(\mathcal{X}^{k,(m)}, O_{1:k}^{(m)}, \theta^{(m)}\big)$
    Evaluate loss: $\frac{1}{M}\sum_{m=1}^M \Big\| \mathcal{Y}^{k,(m)} - \overline{v}_k\big(\mathcal{X}^{k,(m)}, O_{1:k}^{(m)}, \theta^{(m)}\big) \Big\|$ and update $\big(u, (\overline{w}_n)_{n=0}^{N-1}\big)$
  **end for**
  Set $u_k^* = u$
  **Stage II: learn surrogate $\Phi$ for the log normalizer $C$.**
  **for** $\ell = 0, \dots, L-1$ **do**
    Sample $\theta^{(m)} \sim p(\theta)$ and $O_{1:k+1}^{(m)}$ from (2), and evaluate target $C\big(O_{1:k+1}^{(m)}, \theta^{(m)}\big)$
    Evaluate loss $\frac{1}{M}\sum_{m=1}^M \Big\| \Phi_{k+1}\big(O_{1:k+1}^{(m)}, \theta^{(m)}\big) - C\big(O_{1:k+1}^{(m)}, \theta^{(m)}\big) \Big\|$ and update $\Phi_{k+1}$
  **end for**
  Define $\overline{v}_{k+1}(x, o_{1:k+1}, \theta) = u_k^*(x, o_{1:k}, \theta) - \log(L(o_{k+1}, x, \theta)) + \Phi_{k+1}(o_{1:k+1}, \theta)$
**end for**
Output: $(\overline{v}, \Phi) = (\overline{v}_k, \Phi_k)_{k=1}^K$

costly, evaluations of $C$ and subsequently fitting $\Phi_k$ by supervised learning. Once trained, $\Phi_k$ provides a fast, differentiable approximation of the log-likelihood contributions needed for parameter inference.

**Inference:** The goal of parameter inference is to approximate the posterior density $p(\theta \mid o_{1:K})$ given a fixed observation sequence $o_{1:K}$. By learning the surrogates $\Phi_k$ in Algorithm 1, we obtain a fast, differentiable approximation of the log-likelihood contributions $\log p(o_k \mid o_{1:k-1}, \theta)$.

More precisely, using the identification of the log-normalizing constants with the incremental log-likelihood (Appendix C), we define the approximate log-posterior

$$\widehat{\ell}(\theta, o_{1:k}) = \log p(\theta) + \sum_{k=1}^{K} \Phi_k(o_{1:k}, \theta), \qquad \widehat{p}(\theta \mid o_{1:K}) \propto \exp(\widehat{\ell}(\theta)). \tag{4}$$

A Maximum A Posteriori (MAP) estimate can then be computed as $\hat{\theta}_{\mathrm{MAP}} \in \arg\max_{\theta \in \Theta} \widehat{\ell}(\theta)$, using gradient-based optimization, since $\widehat{\ell}$ is differentiable in $\theta$. Furthermore, samples from $\widehat{p}(\theta \mid o_{1:K})$ can be obtained efficiently using Metropolis–Hastings or Hamiltonian Monte Carlo.

## 3 Experimental results

In this section, we show the performance of our proposed method on two examples, compared to several classic algorithms. For further implementation details, we refer to Appendix E.

**Problems:** Our first example is linear with $d = 10$ state dimensions and the second is nonlinear with one state dimension. For both cases, the parameter is one-dimensional and sampled uniformly in the intervals $(-0.5, 0.5)$ and $(1/5, 3/5)$ respectively. In the first example, the drift for the SDE is set to $b(x, \theta) = 3\, I_{d\times d}\,(\theta\, \mathbf{1}_d - x)$ and in the second, the drift is set to $b(x, \theta) = \theta\,(5x - x^3)$. For both, we utilize a constant diffusion coefficient $g = 1$ and linear measurements $h(x, \xi) = x + \xi$, with $\xi \sim \mathcal{N}(0, 1)$. Note that for a fixed parameter, the former SDE is solved by an Ornstein–Uhlenbeck (OU) process with long-term expected value $\theta$. Moreover, the unconditional density of the second example has two modes in $\pm\sqrt{5}$, hence we refer to it as Bistable below. The strength of attraction to these modes is controlled by the parameter $\theta$.

**Benchmarks:** As the OU example is linear, the Kalman filter provides the exact filter solution. In the bistable case we instead utilize a bootstrap Particle Filter (PF) with a larger number of particles ($10^6$) to obtain a good reference solution. We also compare against particle filters with fewer number of samples (thereby altering the runtime-quality trade-off) and Ensemble Kalman Filters (EnKFs) with varying number of particles. In the nonlinear case, we also compare against the Extended Kalman Filter (EKF), being a straight-forward application of the Kalman filter to the linearized dynamics. Remark that all benchmarks require propagation of either particles (PF and EnKF) or state estimates and covariances (EKF) to approximate $p(o_{1:K} \mid \theta)$, yielding slow and non-differentiable evaluations.

**Metrics:** We report the following metrics, which are detailed in Appendix D: Log-Likelihood Error (LLE), MAP Error (MAPE) and Predictive Log-Likelihood (PLL). Additionally, we report the total inference time to evaluate the posterior log-likelihood for $10^4$ separate parameter values. This does not include training of the deep density method.

**Results:** The results are displayed in Table 1. In the high-dimensional linear example, only the PF with $10^6$ particles and the EnKF with $10^5$ particles outperform our method in terms of the LLE. On the other hand, our method has the lowest MAPE and a competitive PLL, while being several order of magnitude faster than all the benchmarked methods. In the nonlinear example, our method performs better than almost all the EnKFs and the EKF, as well as the PF with $10^3$ samples. The PFs with $10^4$ and $10^5$ samples perform better across all metrics, but only at a prohibitively long runtime.

Lastly, to show that our method provides sensible posterior densities, we show two examples for both the problems in Figure 1. Evidently, our method produces a smooth, noise-free approximation that matches the reference density quite well.

Table 1: Results in our two numerical examples. Arrows indicate whether lower is better (↓) or higher is better (↑).

| | OU | | | | Bistable | | | |
|---|---|---|---|---|---|---|---|---|
| | LLE (↓) | MAPE (↓) | PLL (↑) | Time | LLE (↓) | MAPE (↓) | PLL (↑) | Time |
| PF $10^3$ | 5.001 | 0.219 | -158.346 | (4.1m) | 0.017 | 0.042 | -16.017 | (1.7m) |
| PF $10^4$ | 0.772 | 0.157 | -157.347 | (15m) | 0.006 | 0.018 | **-16.006** | (5.3m) |
| PF $10^5$ | 0.146 | 0.099 | -157.047 | (1.2h) | **0.005** | **0.010** | -16.007 | (39m) |
| PF $10^6$ | 0.034 | 0.056 | **-156.983** | (16h) | - | - | - | (-) |
| EKF | - | - | - | - | 0.163 | 0.024 | -16.043 | (51s) |
| EnKF $10^3$ | 1.207 | 0.177 | -157.443 | (5.0m) | 0.071 | 0.058 | -16.031 | (1.2m) |
| EnKF $10^4$ | 0.104 | 0.100 | -157.068 | (16m) | 0.049 | 0.034 | -16.024 | (1.8m) |
| EnKF $10^5$ | **0.020** | 0.055 | -156.983 | (1.3h) | 0.047 | 0.023 | -16.024 | (10m) |
| Ours | 0.077 | **0.024** | -157.054 | (**0.3s**) | 0.009 | 0.027 | -16.017 | (**0.3s**) |

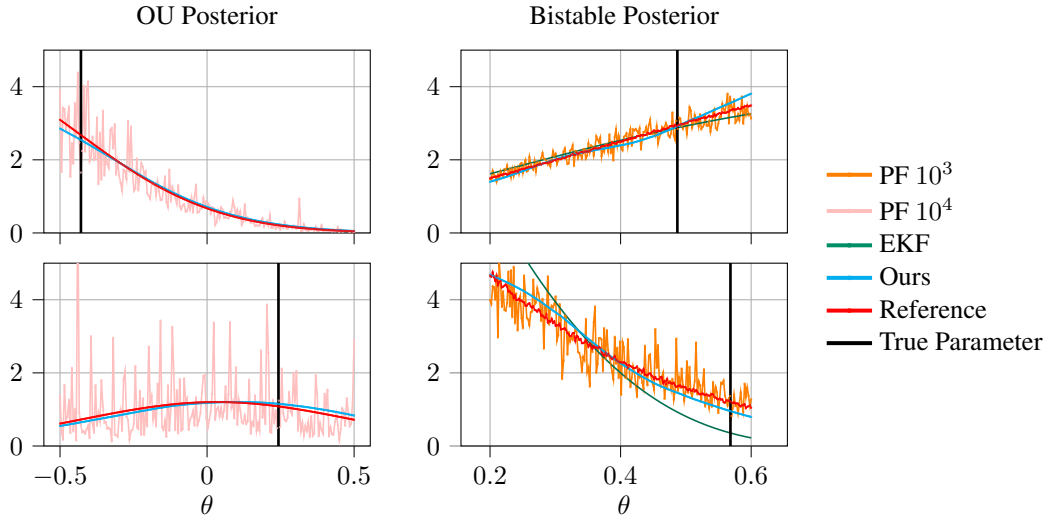


Figure 1: Posterior densities for two parameter samples and accompanying observation chains for both numerical examples.

## 4 CONCLUSION AND OUTLOOK

In this work, we extended a deep density method for filtering to joint parameter-state estimation. Additionally, we introduced neural surrogate models to learn the log normalizer in each step, allowing for efficient and effective evaluation of the log-posterior for the parameter. Through two numerical examples, we showed that the method is promising for several downstream tasks, such as obtaining MAP parameter estimates. In future work, we will investigate how well the method performs on nonlinear examples with higher state dimension and on high-dimensional parameter spaces.

### ACKNOWLEDGMENTS

The work of K.B. was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The work of F.R. was funded by the Swedish Electromobility Centre (SEC) and partially supported by WASP. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Chalmers e-Commons partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

## REFERENCES

A. Apte, C. K. R. T. Jones, A. M. Stuart, and J. Voss. Data assimilation: Mathematical and statistical perspectives. *Int. J. Numer. Methods Fluids*, 56:1033–1046, 2008.

Kasper Bågmark and Filip Rydin. High-dimensional Bayesian filtering through deep density approximation. *arXiv:2511.07261*, 2025.

Kasper Bågmark, Adam Andersson, and Stig Larsson. Nonlinear filtering based on density approximation and deep BSDE prediction. *arXiv:2508.10630*, 2025.

Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, third edition, 2008.

Nicolas Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *Ann. Statist.*, 32(6):2385–2411, 2004.

Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat*, 5:349–380, November 2017.

Nicole El Karoui, Shige Peng, and Marie Claire Quenez. Backward stochastic differential equations in finance. *Math. Finance*, 7(1):1–71, 1997.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Statist. Assoc.*, 102(477):359–378, 2007.

Jiequn Han and Jihao Long. Convergence of the deep BSDE method for coupled FBSDEs. *Probab. Uncertain. Quant. Risk*, 5:Paper No. 5, 33, 2020.

Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.*, 24(89), 2023.

Peter S. Maybeck. *Stochastic Models, Estimation, and Control, Volume 1*. Academic Press, 1979.

Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer Science & Business Media, 2003.

E. Pardoux and S. Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In Boris L. Rozovskii and Richard B. Sowers (eds.), *Stochastic Partial Differential Equations and Their Applications*, pp. 200–217. Springer, Berlin Heidelberg, 1992.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*, volume 17 of *Institute of Mathematical Statistics Textbooks*. Cambridge University Press, Cambridge, second edition, 2023.

Chris Snyder, Thomas Bengtsson, and Mathias Morzfeld. Performance bounds for particle filters using the optimal proposal. *Mon. Weather Rev.*, 143:4750–4761, 2015.

## A PDE SYSTEM

The SDE (1) gives rise to the infinitesimal generator $A$, and its adjoint $A^*$ which is directly related to the Fokker–Planck equation governing the evolution in time of the probability density of $S$. They are defined for a function $\varphi \in C^2(\mathbb{R}^d; \mathbb{R})$, $\theta \in \Theta$ and $x \in \mathbb{R}^d$, by

$$A_\theta \varphi(x) = \frac{1}{2} \sum_{i,j=1}^{d} a_{ij}(x,\theta) \, \frac{\partial^2 \varphi(x)}{\partial x_i \partial x_j} + \sum_{i=1}^{d} b_i(x,\theta) \, \frac{\partial \varphi(x)}{\partial x_i},$$

$$A_\theta^* \varphi(x) = \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}(a_{ij}\varphi)(x,\theta) - \sum_{i=1}^{d} \frac{\partial}{\partial x_i}(b_i \varphi)(x,\theta),$$

where $a = (a_{ij}) = gg^\top$ with entries $a_{ij}$. By considering a function $\varphi \in C^2(\mathbb{R}^d; \mathbb{R})$ we can define a residual function $f \colon \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \times \Theta \to \mathbb{R}$ so that $A_\theta^* \varphi(x) = A_\theta \varphi(x) + f(x, \varphi(x), \nabla\varphi(x), \theta)$. It is given by

$$f(x,u,v,\theta) = \sum_{i,j=1}^{d} \frac{\partial a_{ij}(x,\theta)}{\partial x_i} \, v_j + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 a_{ij}(x,\theta)}{\partial x_i \partial x_j} \, u - \sum_{i=1}^{d} \frac{\partial b_i(x,\theta)}{\partial x_i} \, u - 2 \sum_{i=1}^{d} b_i(x,\theta) \, v_i.$$

We denote the negative log probability density function, conditioned on parameters $\theta$ and observations $o_{1:k}$, by $v = (v_k)_{k=0}^K$ as maps $[t_k, t_{k+1}] \times \mathbb{R}^d \times \mathbb{O}_k \times \Theta \to \mathbb{R}$. Now, for $\theta \in \Theta$ and $o_{1:K} \in \mathbb{O}_K$, we have the following recursive system of PDEs, derived in Bågmark & Rydin (2025), initiated by $v_0(0, x, \cdot, \theta) = -\log \pi_0(x, \theta)$

$$\begin{aligned} v_k(t, x, o_{1:k}, \theta) &= v_k(t_k, x, o_{1:k}, \theta) \\ &\quad + \int_{t_k}^{t} \Big( A_\theta v_k(s, x, o_{1:k}, \theta) + f_{\log}\big(x, v_k(s, x, o_{1:k}, \theta), \nabla v_k(s, x, o_{1:k}, \theta), \theta\big) \Big) \, \mathrm{d}s, \end{aligned} \tag{5}$$

$$\begin{aligned} v_{k+1}(t_{k+1}, x, o_{1:k+1}, \theta) &= v_k(t_{k+1}, x, o_{1:k}, \theta) - \log\big(L(o_{k+1}, x, \theta)\big) \\ &\quad + \log\left( \int_{\mathbb{R}^d} \exp\big(-v_{k-1}(t_k, z, o_{1:k-1}, \theta)\big) L(o_k, z, \theta) \, \mathrm{d}z \right), \end{aligned} \tag{6}$$

where $f_{\log}$ is defined, for $x \in \mathbb{R}^d$, $u \in \mathbb{R}$, $w \in \mathbb{R}^d$, and $\theta \in \Theta$, by

$$f_{\log}(x, u, w, \theta) = -\frac{1}{2} \| g(x,\theta)^\top w \|^2 - f(x, 1, -w, \theta).$$

For each $\theta \in \Theta$ and observation sequence $o_{1:K} \in \mathbb{O}_K$ this system of PDE consists of a deterministic PDE (5) and a Bayes' update (6). This logarithmic formulation provides a more stable problem avoiding the risk of vanishing density values in increasing state dimension. In this work we consider the same reformulation but now for the system of parametric conditional PDEs.

## B BACKWARD STOCHASTIC DIFFERENTIAL EQUATION FILTER FORMULATION

We state the discrete deep BSDE optimization problem used to approximate the solution of the log-Fokker–Planck system from Appendix A. For each $k = 0, \ldots, K-1$, we consider the minimization

problem

$$\min_{\substack{u\in L^\infty(\mathbb{O}_k\times\Theta;C(\mathbb{R}^d;\mathbb{R}))\\(\overline{w}_n)_{n=0}^{N-1}\in\prod_{n=0}^{N-1}L^\infty(\mathbb{O}_k\times\Theta;C(\mathbb{R}^d;\mathbb{R}^d))}} \mathbb{E}\left[\left|\mathcal{Y}_N^k-\overline{v}_k(\mathcal{X}_N^k,O_{1:k},\theta)\right|^2\right]$$

$$\begin{aligned}
\mathcal{X}_{n+1}^k &= \mathcal{X}_n^k + b(\mathcal{X}_n^k,\theta)(t_{k,n+1}-t_{k,n}) + \sigma(\mathcal{X}_n^k,\theta)(W_{t_{k,n+1}}-W_{t_{k,n}}), \quad n=0,\dots,N-1,\\
\mathcal{Z}_n^k &= \overline{w}_n(\mathcal{X}_n^k,O_{1:k},\theta), \qquad n=0,\dots,N-1,\\
\mathcal{Y}_{n+1}^k &= u(\mathcal{X}_0^k,O_{1:k},\theta) - \sum_{\ell=0}^{n}\Big(f_{\log}\big(\mathcal{X}_\ell^k,\mathcal{Y}_\ell^k,\mathcal{Z}_\ell^k,\theta\big)(t_{k,\ell+1}-t_{k,\ell})\\
&\qquad - \big(\mathcal{Z}_\ell^k\big)^\top\sigma(\mathcal{X}_\ell^k,\theta)(W_{t_{k,\ell+1}}-W_{t_{k,\ell}})\Big), \qquad n=0,\dots,N-1,
\end{aligned}$$

where $\overline{v}_k$ is defined, for $x\in\mathbb{R}^d$, $o_{1:k}\in\mathbb{O}_k$, and $\theta\in\Theta$, from the previous optimum $u_{k-1}^*$

$$\overline{v}_k(x,o_{1:k},\theta) = \begin{cases} -\log(\pi_0(x,\theta)), & k=0,\\ u_{k-1}^*(x,o_{1:k-1},\theta) - \log\big(L(o_k,x,\theta)\big) + C(o_{1:k},\theta), & k=1,\dots,K,\end{cases}$$

and $C(o_{1:k},\theta)$ is the log-transformed normalizing constant given by

$$C(o_{1:k},\theta) = \log\left(\int_{\mathbb{R}^d}\exp\big(-u_{k-1}^*(z,o_{1:k-1},\theta)\big)L(o_k,z,\theta)\,\mathrm{d}z\right).$$

Parameterizing the continuous functions with neural networks and optimizing the weights with stochastic gradient descent yields the LogBSDEF method introduced in Bågmark & Rydin (2025). By recalling $p_k=-\log v_k$, we remark that in Bågmark et al. (2025) they study the corresponding FBSDE associated to $p$, where the approximation is denoted $\overline{p}_k=-\log v_k$. Here they prove a mixed a priori and a posteriori error estimate showing robustness of the method.

## C Inference details

For a fixed parameter $\theta$, the filtering recursion yields conditional densities $p_k(x\mid o_{1:k},\theta)$ with corresponding log-densities

$$v_k(t,x,o_{1:k},\theta) = -\log p_k(x\mid o_{1:k},\theta),$$

which satisfy the log-Fokker–Planck prediction and update equations stated in Appendix A. Starting from Bayes' formula, the (unnormalized) log-posterior can be decomposed as

$$\ell(\theta) = \log p(\theta) + \log p(o_{1:K}\mid\theta) = \log p(\theta) + \sum_{k=1}^{K}\log p(o_k\mid o_{1:k-1},\theta),$$

by the chain rule for conditional densities. The normalizing term in the log-density update (6) satisfies

$$C(o_{1:k},\theta) = \log\left(\int_{\mathbb{R}^d}\exp\big(-v_{k-1}(t_k,z,o_{1:k-1},\theta)\big)L(o_k,z,\theta)\;\mathrm{d}z\right) = \log p(o_k\mid o_{1:k-1},\theta),$$

since $\exp(-v_{k-1})$ is precisely the predictive density at time $t_k$. Replacing $C(o_{1:k},\theta)$ by its learned surrogate $\Phi_k(o_{1:k},\theta)$ yields the approximate log-likelihood (4).

Given the approximate log-posterior

$$\widehat{\ell}(\theta) = \log p(\theta) + \sum_{k=1}^{K}\Phi_k(o_{1:k},\theta), \qquad \widehat{p}(\theta\mid o_{1:K})\propto\exp\big(\widehat{\ell}(\theta)\big),$$

samples from $\widehat{p}(\theta\mid o_{1:K})$ can be generated using a Metropolis–Hastings (MH) algorithm.

Let $q(\theta'\mid\theta)$ denote a proposal density. Initiating $m=0$ with an initial value $\theta^{(0)}$, we continue by sampling

$$\theta'\sim q(\,\cdot\mid\theta^{(m)}),$$

and compute the log-acceptance ratio

$$r\big(\theta^{(m)}, \theta'\big) = \widehat{\ell}(\theta') - \widehat{\ell}(\theta^{(m)}) + \log q(\theta^{(m)} \mid \theta') - \log q(\theta' \mid \theta^{(m)}).$$

The proposal is accepted with probability

$$\alpha\big(\theta^{(m)}, \theta'\big) = \min\{1, \exp(r(\theta^{(m)}, \theta'))\}.$$

If accepted, set $\theta^{(m+1)} = \theta'$; otherwise $\theta^{(m+1)} = \theta^{(m)}$.

For a Gaussian proposal centered on the previous sample $\theta^{(m)}$ with covariance $\Sigma$, we have $\log q(\theta^{(m)} \mid \theta') = \log q(\theta' \mid \theta^{(m)})$, so that the acceptance probability reduces to

$$\alpha = \min\Big\{1, \exp\big(\widehat{\ell}(\theta') - \widehat{\ell}(\theta^{(m)})\big)\Big\}.$$

In practice one discards an initial burn-in period and tunes $\Sigma$ to obtain reasonable acceptance rates.

## D EVALUATION METRICS

Our first metric is the log-likelihood error. Given the ground truth log-likelihood $p$ (or a sufficiently close approximation thereof) as well as an approximation $\widehat{p}$, it is defined as

$$\mathbb{E}_{o\sim p(O|\theta'),\, \theta'\sim p(\theta')}\, \mathbb{E}_{\theta\sim p(\theta)}\Big[\Big(\log p\big(O_{1:K} \mid \theta\big) - \log \widehat{p}\big(O_{1:K} \mid \theta\big)\Big)^2\Big].$$

Consequently, the LLE measures how well an approximation captures the log-likelihood across the entire parameter space $\Theta$. Formally, it is a type of squared $L^2L^2$-error. In practice, we obtain it by sampling $M$ parameters $\theta'$ and corresponding observation chains $O_{1:K}$ as well as $M'$ new independent parameters $\theta$ for evaluation:

$$\text{LLE} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{M'}\sum_{j=1}^{M'}\Big(\log p\big(O_{1:K}^{(i)} \mid \theta^{(j)}\big) - \log \widehat{p}\big(O_{1:K}^{(i)} \mid \theta^{(j)}\big)\Big)^2.$$

The second metric is the MAP error. Given the MAP estimate $\theta_{\text{MAP}}(o_{1:K})$ from a reference method and the approximate MAP estimate $\widehat{\theta}_{\text{MAP}}(o_{1:K})$, the metric is defined as

$$\mathbb{E}_{O\sim p(O|\theta),\, \theta\sim p(\theta)}\, \big\|\theta_{\text{MAP}}(O_{1:K}) - \widehat{\theta}_{\text{MAP}}(O_{1:K})\big\|_1.$$

In practice, we sample over $M$ observation sequences and, since both our examples have one-dimensional parameter spaces, calculate the maximum over a uniformly spaced grid $G$ to obtain the MAP estimate:

$$\text{MAPE} = \frac{1}{M}\sum_{i=1}^{M}\Big\|\arg\max_{\theta\in G} p\big(\theta \mid O_{1:K}^{(i)}\big) - \arg\max_{\theta\in G}\widehat{p}\big(\theta \mid O_{1:K}^{(i)}\big)\Big\|_1.$$

Lastly, we utilize the predictive log-likelihood of the observation sequence. Given an approximation $\widehat{p}$ of the true likelihood, we sample $M$ parameters $\theta$ and corresponding observation sequences $O_{1:k}$ and calculate it according to

$$\text{PLL} = \frac{1}{M}\sum_{i=1}^{M}\sum_{k=1}^{K}\log \widehat{p}\big(O_k^{(i)} \mid O_{1:k-1}^{(i)}, \theta^{(i)}\big).$$

We can use this metric to measure solution quality, as the optimal method will maximize this quantity (Gneiting & Raftery, 2007).

For the numerical experiments, referencing the above equations, we set $M = 500$, $M' = 100$ and utilize 100 grid points in $G$.

## E MODELS, TRAINING AND HYPERPARAMETERS

Our setup closely resembles that in Bågmark & Rydin (2025), where both the Ornstein–Uhlenbeck example and bistable example with fixed parameter values are treated. More specifically, we utilize the same network architecture for the conditional log-filter, namely standard Fully Connected Networks (FCNs) with fixed hidden width, ReLU activation and constant input dimension across time steps. All hyperparameters are the same as reported in Table 1 in Appendix D in Bågmark & Rydin (2025), with the exception of the number of discretization steps for the bistable example, which we set to $N = 32$, and the number of normalized trajectories per batch, which we set equal to the batch size. The latter choice is due to the fact that the neural surrogate for the log normalization constant can be trained simultaneously as the filter, allowing for efficient normalization while training the subsequent filter model.

Regarding the normalization surrogate, it is also an FCN, with 3 hidden layers. The hidden width is set to $128$ in the bistable example and $512$ in the OU example. For training labels, we use quadrature and EKF-based importance sampling to evaluate the integral (3) in the bistable example and the OU example respectively. In both cases, we sample new training data per batch while training. The batch size is set to $256$, batches per epoch to $200$, number of epochs to $300$ and learning rate to $10^{-4}$.

For both the filter models and normalization models, we sample one unique parameter $\theta$ per batch with accompanying observation chains.

With this setup, the training takes approximately 79 hours in the linear example and 26 hours in the bistable example.