Leveraging Viscous Hamilton–Jacobi PDEs for Uncertainty Quantification in Scientific Machine Learning[*]

Zongren Zou[†], Tingwei Meng[‡], Paula Chen[§], Jérôme Darbon[¶], and George Em Karniadakis[∥]

Abstract. Uncertainty quantification (UQ) in scientific machine learning (SciML) combines the powerful predictive power of SciML with methods for quantifying the reliability of the learned models. However, two major challenges remain: limited interpretability and expensive training procedures. We provide a new interpretation for UQ problems by establishing a new theoretical connection between some Bayesian inference problems arising in SciML and viscous Hamilton–Jacobi partial differential equations (HJ PDEs). Namely, we show that the posterior mean and covariance can be recovered from the spatial gradient and Hessian of the solution to a viscous HJ PDE. As a first exploration of this connection, we specialize in Bayesian inference problems with linear models, Gaussian likelihoods, and Gaussian priors. In this case, the associated viscous HJ PDEs can be solved using Riccati ODEs, and we develop a new Riccati-based methodology that provides computational advantages when continuously updating the model predictions. Specifically, our Riccati-based approach can efficiently add or remove data points to the training set invariant to the order of the data and continuously tune hyperparameters. Moreover, neither update requires retraining on or access to previously incorporated data. We provide several examples from SciML involving noisy data and *epistemic uncertainty* to illustrate the potential advantages of our approach. In particular, this approach's amenability to data streaming applications demonstrates its potential for real-time inferences, which, in turn, allows for applications in which the predicted uncertainty is used to dynamically alter the learning process.

Key words. multi-time Hamilton–Jacobi PDEs, scientific machine learning, uncertainty quantification, Bayesian inference, Riccati equation

MSC codes. 35F21, 62F15, 65L99, 65N99, 68T05, 35B37

DOI. 10.1137/24M1646455

[†]Division of Applied Mathematics, Brown University, Providence, RI 02912 USA (zongren_zou@brown.edu).
[‡]Department of Mathematics, UCLA, Los Angeles, CA 90025 USA (tingwei@math.ucla.edu).
[§]Naval Air Warfare Center Weapons Division China Lake, Ridgecrest, CA 93555 USA (paula.x.chen.civ@us.navy.mil).
[¶]Corresponding author. Division of Applied Mathematics, Brown University, Providence, RI 02912 USA (jerome_darbon@brown.edu).
[∥]Division of Applied Mathematics and School of Engineering, Brown University, Providence, RI 02912 USA, and Pacific Northwest National Laboratory, Richland, WA 99354 USA (george_karniadakis@brown.edu).

1. Introduction. Scientific machine learning (SciML) is a recent and evolving field consisting of data-driven techniques for learning physics-based models [42, 21, 50, 11, 34, 40, 41, 56, 7, 47, 8, 58, 52, 30, 57] that finds applications in diverse areas [26, 3, 2, 46, 31, 24, 53, 55]. While deterministic SciML models demonstrate impressive predictive power, real-life applications often require quantitative metrics for the trustworthiness of learned models. To this end, uncertainty quantification (UQ) [20, 38, 59] has been integrated with SciML to create new learning approaches that produce quantifiably high-confidence models in the absence of a ground truth and use these confidence metrics to help inform the learning process.

In this paper, we reinterpret certain Bayesian inference problems arising from SciML and UQ using the framework of viscous Hamilton–Jacobi partial differential equations (HJ PDEs) [13, 54, 14]. Specifically, we develop a new theoretical connection between these Bayesian inference problems and multi-time viscous HJ PDEs that shows that the posterior mean and covariance of the Bayesian model can be recovered from the spatial gradient and Hessian of the solution to an associated viscous HJ PDE (section 2). This connection is summarized in Figure 1. Our previous work in [6, 5] establishes similar theoretical connections between deterministic SciML problems and inviscid HJ PDEs. As such, the methodology in [6, 5] does not automatically extend to UQ. Meanwhile, the work in [12] establishes a similar link between Bayesian inference and viscous HJ PDEs but develops this connection via the Cole–Hopf transformation [13] and only considers the single-time case. In contrast, we do not rely on the Cole–Hopf transformation and consider multi-time HJ PDEs to enable our connection to SciML.

As a first exploration of this connection, we specialize in Bayesian inference problems with linear models, Gaussian likelihoods, and Gaussian priors (section 3). In this case, the corresponding viscous HJ PDE (and hence the Bayesian inference problem) can be solved using Riccati ODEs [28]. The resulting Riccati-based methodology can then be leveraged to efficiently update the model predictions (section 3.2) and continuously tune hyperparameters (section 3.3). To illustrate the potential advantages of this approach, we apply this Riccati-based methodology to several UQ-based examples from SciML (section 4). In particular, we focus on noisy data cases and quantify *epistemic uncertainty* [20, 38, 59], which refers to the
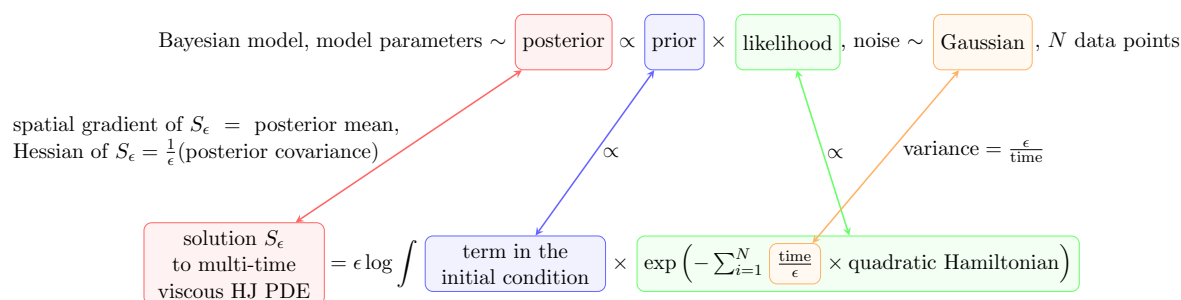
Figure 1. *(See section 2) Illustration of a connection between a Bayesian inference problem in scientific machine learning (top) and the solution to a multi-time viscous HJ PDE (bottom). The colors indicate the associated quantities between problems. This color scheme is reused in the subsequent illustrations of our connection. The arrow labels describe how the boxed quantities are related. For example, the posterior mean in the learning problem is equivalent to the spatial gradient of the solution to the multi-time viscous HJ PDE (red).*

uncertainty associated with the model parameters. In section 4.1, we demonstrate the potential computational advantages of our approach over more standard SciML/UQ techniques when incrementally updating the model predictions. Namely, we show how our Riccati-based approach naturally coincides with continual learning while inherently avoiding catastrophic forgetting despite updating the model predictions without accessing the historical data (section 4.1.1). We also show how solving the Riccati ODEs provides a continuous flow of solutions with respect to the hyperparameters of the model, which allows the hyperparameters to be tuned continuously (section 4.1.2). In section 4.2, we demonstrate the versatility of our Riccati-based approach in handling different learning scenarios, e.g., big data (section 4.2.1) and active learning (section 4.2.2) settings. In particular, these examples illustrate how uncertainty metrics can be leveraged to inform how we learn. In section 4.3, we highlight the invariance of our Riccati-based approach to the order of the data, which provides flexibility in how the data is sampled.

The contributions of this work can be summarized as follows:
- new mathematical theory connecting certain Bayesian inference problems arising from SciML to viscous HJ PDEs,
- new Riccati-based methodology for solving Bayesian inference problems with linear models, Gaussian likelihoods, and Gaussian priors, and
- detailed experimental demonstrations of the potential computational advantages of this Riccati-based approach across a variety of learning applications.

This work presents exciting opportunities to advance the theoretical foundations of SciML and thus to create new interpretable SciML methods. While we demonstrate promising results, further research is needed to extend the work presented here to more general learning settings. We discuss some possible future research directions of this work in section 5. Some additional technical details and numerical results are provided in the appendices.

2. Connecting viscous HJ PDEs to Bayesian inference in machine learning. In this section, we establish a new theoretical connection between viscous HJ PDEs and Bayesian inference in machine learning (ML). Specifically, we formulate a general regression task in the Bayesian framework and then connect it to a viscous HJ PDE with quadratic Hamiltonian.

2.1. Bayesian inference in machine learning. Bayesian inference is often adopted in ML due to its effectiveness in integrating prior information, handling noisy data, providing UQ, and hence in making quantifiably trustworthy predictions. Consider the following model [50]:

$$\boldsymbol{y} = u_{\boldsymbol{\theta}}(\boldsymbol{z}) + \xi, \tag{2.1}$$

where $\boldsymbol{z} \in \mathbb{R}^\ell$ is the input, $\boldsymbol{y} \in \mathbb{R}^m$ is the observed output, $u_{\boldsymbol{\theta}} : \mathbb{R}^\ell \to \mathbb{R}^m$ is the model parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$, and $\xi \in \mathbb{R}^m$ is additive noise. The goal is to learn $u_{\boldsymbol{\theta}}$ from data $\mathcal{D} = \{(\boldsymbol{z}_i, \boldsymbol{y}_i)\}_{i=1}^N$, where $\boldsymbol{z}_i, \boldsymbol{y}_i$ denote measurements of $\boldsymbol{z}, \boldsymbol{y}$, respectively. Specifically, we learn the model $u_{\boldsymbol{\theta}}$ by estimating $\boldsymbol{\theta}$ from the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ (e.g., $\boldsymbol{\theta}$ could be the posterior mean estimator $\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$ or the maximum a posteriori (MAP) estimator $\arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D})$). By Bayes' theorem, $p(\boldsymbol{\theta}|\mathcal{D})$ can be computed as

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{2.2}$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ is the prior, and $p(\mathcal{D}) = \int_{\mathbb{R}^n} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ is the marginal likelihood, which is independent of $\boldsymbol{\theta}$ and often treated as a normalizing constant. From these distributions, we obtain quantities that can be used to statistically analyze and quantify the uncertainty in the learned model arising from the noisy data and/or the prior on $\boldsymbol{\theta}$.

The posterior $p(\boldsymbol{\theta}|\mathcal{D})$ often becomes analytically intractable [38, 59], particularly when either the likelihood or the prior is complicated. In these scenarios, estimation techniques, such as Markov chain Monte Carlo (MCMC) [32], are typically employed. Here we create a new theoretical connection between HJ PDEs and Bayesian inference in ML that yields opportunities for new HJ PDE-based methods for estimating $\boldsymbol{\theta}$ and $p(\boldsymbol{\theta}|\mathcal{D})$. As a first exploration of this connection, we consider the case where the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ follows a Gaussian distribution and the model $u_{\boldsymbol{\theta}}$ depends linearly on $\boldsymbol{\theta}$. In other words, we consider the case where the data is perturbed by additive white noise (i.e., ξ is Gaussian), and we learn the model $u_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Phi(\boldsymbol{z})\boldsymbol{\theta}$, where $\Phi : \mathbb{R}^\ell \to \mathbb{R}^{m \times n}$. With some assumptions on the prior, this scenario yields a connection to a viscous HJ PDE with a quadratic Hamiltonian.

2.2. Connection to viscous HJ PDEs. In this section, we detail the connection between certain Bayesian inference problems with Gaussian likelihood and viscous HJ PDEs with a quadratic Hamiltonian. Specifically, we consider the case where we learn the model $u_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Phi(\boldsymbol{z})\boldsymbol{\theta}$, where $\Phi : \mathbb{R}^\ell \to \mathbb{R}^{m \times n}$ and the posterior of the parameter $\boldsymbol{\theta}$ is estimated using the Bayesian framework described in section 2.1. We assume that the likelihood is Gaussian given by $p(\mathcal{D}|\boldsymbol{\theta}) \propto \prod_{i=1}^N \exp(-\frac{1}{2\sigma_i^2}\|\Phi(\boldsymbol{z}_i)\boldsymbol{\theta} - \boldsymbol{y}_i\|^2)$, where σ_i^2 is the variance of the ith data point $(\boldsymbol{z}_i, \boldsymbol{y}_i)$. We assume that the prior distribution is given by $p(\boldsymbol{\theta}) \propto \exp(-\frac{1}{\epsilon}(-\langle\mathbf{x}, \boldsymbol{\theta}\rangle + g(\boldsymbol{\theta})))$, where $\epsilon > 0$ is a hyperparameter and $g : \mathbb{R}^n \to \mathbb{R}$ is such that $\boldsymbol{\theta} \mapsto \exp(-\frac{1}{\epsilon}(-\langle\mathbf{x}, \boldsymbol{\theta}\rangle + g(\boldsymbol{\theta})))$ is integrable for every $\mathbf{x} \in \mathbb{R}^n$. Hence, if g is quadratic, then the prior distribution is also Gaussian. Note that we use $p(\boldsymbol{\theta}) \propto f(\boldsymbol{\theta})$ to denote that the probability density function $p(\boldsymbol{\theta})$ is proportional to $f(\boldsymbol{\theta})$ or, in other words, $p(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta})}{\int_{\mathbb{R}^n} f(w)dw}$. Similarly, for conditional probabilities, $p(\boldsymbol{\theta}|\mathcal{D}) \propto f(\boldsymbol{\theta}, \mathcal{D})$ means that $p(\boldsymbol{\theta}|\mathcal{D}) = \frac{f(\boldsymbol{\theta}, \mathcal{D})}{\int_{\mathbb{R}^n} f(w, \mathcal{D})dw}$.

Remark 2.1. The particular form of the likelihood that we consider corresponds to the data $\mathcal{D}$ being produced by $\boldsymbol{y}_i = u_{\boldsymbol{\theta}}(\boldsymbol{z}_i) + \xi_i$, where $\xi_1, \ldots, \xi_N$ are independent, each ξ_i represents Gaussian noise with zero mean and covariance matrix $\sigma_i^2 I$ (I is the $m \times m$ identity matrix), and $u_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Phi(\boldsymbol{z})\boldsymbol{\theta}$. The results in this paper have a straightforward generalization to the case where $\xi_1, \ldots, \xi_N$ are independent, each ξ_i is a general Gaussian, and $u_{\boldsymbol{\theta}}(\boldsymbol{z}) = \Phi(\boldsymbol{z})\boldsymbol{\theta} + c(\boldsymbol{z})$, where $c : \mathbb{R}^\ell \to \mathbb{R}^m$ is a function independent of $\boldsymbol{\theta}$. However, for simplicity, we only present the case where $c(\boldsymbol{z}) = 0$.

First consider the case where we have $N = 1$ data point $(\boldsymbol{z}_1, \boldsymbol{y}_1)$. Denote $\Phi_1 := \Phi(\boldsymbol{z}_1)$. Define a Hamiltonian $H : \mathbb{R}^n \to \mathbb{R}$ by $H(\boldsymbol{\theta}) = \frac{1}{2}\|\Phi_1\boldsymbol{\theta} - \boldsymbol{y}_1\|^2$ and a function $S_\epsilon : \mathbb{R}^n \times [0, \infty) \to \mathbb{R}$ by

$$(2.3) \qquad S_\epsilon(\mathbf{x}, t) = \epsilon \log \int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle\mathbf{x}, \boldsymbol{\theta}\rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}.$$

Then S_ϵ satisfies the following viscous HJ PDE:

$$(2.4) \qquad \begin{cases} \partial_t S_\epsilon(\mathbf{x}, t) + \frac{1}{2}\|\Phi_1 \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t) - \boldsymbol{y}_1\|^2 + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \cdot (\Phi_1^T \Phi_1 \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t)) = 0, & \mathbf{x} \in \mathbb{R}^n, \ t > 0, \\ S_\epsilon(\mathbf{x}, 0) = J(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^n, \end{cases}$$

where the initial condition J is defined by

$$(2.5) \qquad J(\mathbf{x}) = \epsilon \log \int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}.$$

Using this HJ PDE framework, the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ can be characterized by $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right)$ by setting the variance of the noise to $\sigma_1^2 = \frac{\epsilon}{t}$. By straightforward computation, the derivatives of S_ϵ are given by

$$\partial_t S_\epsilon(\mathbf{x}, t) = -\frac{\int_{\mathbb{R}^n} H(\boldsymbol{\theta}) \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}} = -\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[H(\boldsymbol{\theta})],$$

$$(2.6) \qquad \nabla_\mathbf{x} S_\epsilon(\mathbf{x}, t) = \frac{\int_{\mathbb{R}^n} \boldsymbol{\theta} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}],$$

$$\nabla_\mathbf{x}^2 S_\epsilon(\mathbf{x}, t) = \frac{1}{\epsilon}\left(\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}\left[\boldsymbol{\theta}\boldsymbol{\theta}^T\right] - \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]^T\right) = \frac{1}{\epsilon}\text{Cov}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}],$$

all of which are related to the posterior expectation and posterior covariance. Namely, computing the first and second derivatives of the solution to the viscous HJ PDE (2.4) is equivalent (up to a multiplicative constant) to computing the first and second moments of the posterior distribution, respectively, and the posterior mean estimator for $\boldsymbol{\theta}$ can be computed as $\nabla_\mathbf{x} S_\epsilon(\mathbf{x}, t)$.

These connections can be easily generalized to the case with multiple data points ($N > 1$). Denote $\Phi_i = \Phi(\mathbf{z}_i)$, and set $\frac{\epsilon}{t_i} = \sigma_i^2$ for each $i = 1, \ldots, N$. The posterior distribution then becomes $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N \frac{t_i}{2}\|\Phi_i\boldsymbol{\theta} - \mathbf{y}_i\|^2 - g(\boldsymbol{\theta})))$. This learning problem is related to a multi-time viscous HJ PDE as follows. Define the ith Hamiltonian $H_i : \mathbb{R}^n \to \mathbb{R}$ by $H_i(\boldsymbol{\theta}) = \frac{1}{2}\|\Phi_i\boldsymbol{\theta} - \mathbf{y}_i\|^2$ and the function $S_\epsilon : \mathbb{R}^n \times [0, \infty)^N \to \mathbb{R}$ by
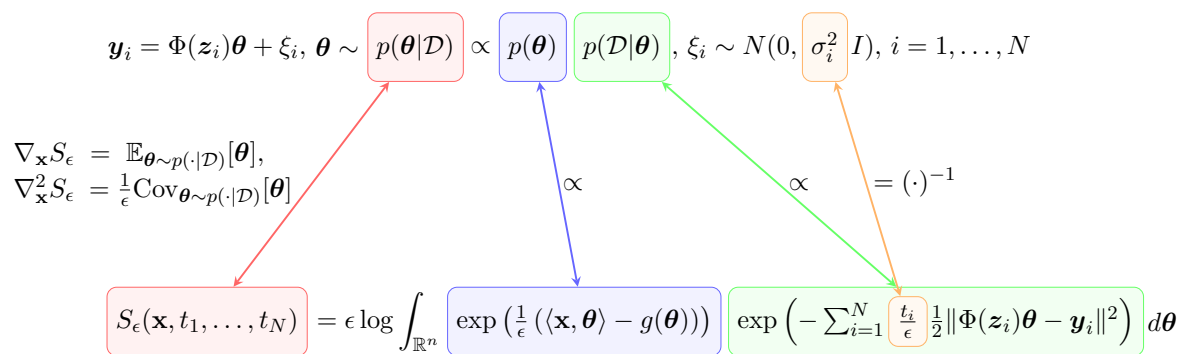
$$(2.7) \qquad S_\epsilon(\mathbf{x}, t_1, \ldots, t_N) = \epsilon \log \int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}\left(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N \frac{t_i}{2}\|\Phi_i\boldsymbol{\theta} - \mathbf{y}_i\|^2 - g(\boldsymbol{\theta})\right)\right) d\boldsymbol{\theta}.$$

Then S_ϵ satisfies the following multi-time viscous HJ PDE:

$$(2.8) \qquad \begin{cases} \partial_{t_i} S_\epsilon(\mathbf{x}, t_1, \ldots, t_N) + \dfrac{1}{2}\|\Phi_i \nabla_\mathbf{x} S_\epsilon(\mathbf{x}, t_1, \ldots, t_N) - \mathbf{y}_i\|^2 \\ \qquad + \dfrac{\epsilon}{2}\nabla_\mathbf{x} \cdot (\Phi_i^T \Phi_i \nabla_\mathbf{x} S_\epsilon(\mathbf{x}, t_1, \ldots, t_N)) = 0, & \mathbf{x} \in \mathbb{R}^n, \, t_i > 0, \, i = 1, \ldots, N, \\ S_\epsilon(\mathbf{x}, 0, \ldots, 0) = J(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^n, \end{cases}$$

where the initial condition J is defined in (2.5). Using the notation of the Hamiltonians H_i, the posterior distribution is also given by $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta})))$. With straightforward computation, the derivatives of S_ϵ are given by

$$\partial_{t_i} S_\epsilon(\mathbf{x}, t_1, \ldots, t_N) = -\frac{\int_{\mathbb{R}^n} H_i(\boldsymbol{\theta}) \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}$$
$$= -\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[H_i(\boldsymbol{\theta})],$$

$$\boldsymbol{y}_i = \Phi(\boldsymbol{z}_i)\boldsymbol{\theta} + \xi_i, \ \boldsymbol{\theta} \sim \boxed{p(\boldsymbol{\theta}|\mathcal{D})} \propto \boxed{p(\boldsymbol{\theta})} \ \boxed{p(\mathcal{D}|\boldsymbol{\theta})}, \ \xi_i \sim N(0, \boxed{\sigma_i^2} I), \ i = 1, \dots, N$$

$$\nabla_{\mathbf{x}}S_\epsilon = \mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}],$$
$$\nabla_{\mathbf{x}}^2 S_\epsilon = \tfrac{1}{\epsilon}\mathrm{Cov}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$$

$$\propto \qquad \propto \qquad = (\cdot)^{-1}$$

$$\boxed{S_\epsilon(\mathbf{x}, t_1, \dots, t_N)} = \epsilon \log \int_{\mathbb{R}^n} \boxed{\exp\left(\tfrac{1}{\epsilon}\left(\langle \mathbf{x}, \boldsymbol{\theta}\rangle - g(\boldsymbol{\theta})\right)\right)} \boxed{\exp\left(-\sum_{i=1}^{N} \boxed{\tfrac{t_i}{\epsilon}} \tfrac{1}{2}\|\Phi(\boldsymbol{z}_i)\boldsymbol{\theta} - \boldsymbol{y}_i\|^2\right)} d\boldsymbol{\theta}$$

Figure 2. *(See section* 2*) Mathematical formulation of the connection between a Bayesian inference problem with a linear model and Gaussian likelihood (*top*) and the solution to a multi-time viscous HJ PDE with a quadratic Hamiltonian (*bottom*). The content of this illustration matches that of Figure* 1 *by replacing each term in Figure* 1 *with its corresponding mathematical expression. The colors indicate the associated quantities. The arrow labels describe how the boxed quantities are related.*

$$\nabla_{\mathbf{x}}S_\epsilon(\mathbf{x}, t_1, \dots, t_N) = \frac{\int_{\mathbb{R}^n} \boldsymbol{\theta} \exp\left(\tfrac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta}\rangle - \sum_{i=1}^{N} t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp\left(\tfrac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta}\rangle - \sum_{i=1}^{N} t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}],$$

(2.9)

$$\nabla_{\mathbf{x}}^2 S_\epsilon(\mathbf{x}, t_1, \dots, t_N) = \frac{1}{\epsilon}\left(\mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}\left[\boldsymbol{\theta}\boldsymbol{\theta}^T\right] - \mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]\mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]^T\right) = \frac{1}{\epsilon}\mathrm{Cov}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}].$$

As in the $N = 1$ case, $\nabla_{\mathbf{x}}S_\epsilon(\mathbf{x}, t_1, \dots, t_N)$ and $\epsilon\nabla_{\mathbf{x}}^2 S_\epsilon(\mathbf{x}, t_1, \dots, t_N)$ are equivalent to the posterior expectation and posterior covariance matrix, respectively. These connections can then be leveraged to compute the posterior mean estimator and to estimate the uncertainty in the learned model. Figure 2 summarizes the connections established above between Bayesian inference problems and multi-time viscous HJ PDEs.

3. Riccati-based methodology. In the previous section, we established a new theoretical connection between Bayesian inference problems with linear models and additive Gaussian noise and viscous HJ PDEs with quadratic Hamiltonians. In this section, we show how this connection can be leveraged to reuse existing HJ PDE solvers to create new efficient training methods for Bayesian inference. While the theoretical connection holds more generally in this section, we focus on the case where the prior distribution is also Gaussian. In this case, the corresponding viscous HJ PDEs can be solved using Riccati ODEs, which in turn can be used to solve the original Bayesian inference problem of interest. Specifically, we develop a new Riccati-based approach for solving these Bayesian inference problems that

1. can add and remove data points from the training dataset invariant to the order of the data;
2. does not require retraining on or access to previously incorporated data to update the learned model, which may provide computational benefits, particularly when the posterior distribution must be recomputed many times; and
3. yields a continuous flow of solutions, which allows the hyperparameters in the learning problem to be tuned continuously.

3.1. Connection using Gaussian prior to Riccati ODEs. In this section, we discuss the special case where the prior distribution is Gaussian with mean $\boldsymbol{\theta}^0 = \Lambda\mathbf{x}$ and covariance matrix $\epsilon\Lambda$, where $\Lambda \in \mathbb{R}^{n\times n}$ is symmetric positive definite. Note that this choice of mean and covariance is not as restrictive as the notation suggests since we have freedom in picking $\mathbf{x} \in \mathbb{R}^n$ (picking a different $\mathbf{x}$ corresponds to evaluating the associated viscous HJ PDE at a different point in space); as long as Λ is invertible (i.e., the prior is a nondegenerate Gaussian), $\boldsymbol{\theta}^0$ can be chosen to be any point in $\mathbb{R}^n$. This particular prior corresponds to setting $g(\boldsymbol{\theta}) = \frac{1}{2}\langle\boldsymbol{\theta}, \Lambda^{-1}\boldsymbol{\theta}\rangle$, and the initial condition (2.5) of the associated viscous HJ PDE becomes $J(\mathbf{x}) = \frac{1}{2}\langle\mathbf{x}, \Lambda\mathbf{x}\rangle + \frac{\epsilon}{2}\log\det(\Lambda) + \frac{\epsilon n}{2}\log(2\pi\epsilon)$. Since the viscous HJ PDE has a quadratic Hamiltonian and a quadratic initial condition, its solution S_ϵ can be computed using Riccati ODEs as follows.

We first consider the single data point case ($N = 1$). In this case, the solution S_ϵ to the viscous HJ PDE (2.4) is given by $S_\epsilon(\mathbf{x}, t) = \frac{1}{2}\mathbf{x}^T P(t)\mathbf{x} + \mathbf{q}(t)^T\mathbf{x} + r(t)$, where $P: [0, +\infty) \to \mathbb{R}^{n\times n}$, $\mathbf{q}: [0, +\infty) \to \mathbb{R}^n$, and $r: [0, +\infty) \to \mathbb{R}$ satisfy the following Riccati ODEs:

(3.1)
$$\begin{cases} \dot{P}(s) + P(s)\Phi_1^T\Phi_1 P(s) = 0, \\ \dot{\mathbf{q}}(s) + P(s)\Phi_1^T(\Phi_1\mathbf{q}(s) - \boldsymbol{y}_1) = 0, \\ \dot{r}(s) + \frac{1}{2}\|\Phi_1\mathbf{q}(s) - \boldsymbol{y}_1\|^2 + \frac{\epsilon}{2}Tr(\Phi_1^T\Phi_1 P(s)) = 0, \end{cases} \quad \begin{cases} P(0) = \Lambda, \\ \mathbf{q}(0) = 0, \\ r(0) = \frac{\epsilon}{2}\log\det(\Lambda) + \frac{\epsilon n}{2}\log(2\pi\epsilon). \end{cases}$$

The mean and covariance matrix of the posterior distribution are given by $\mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}] = \nabla_{\mathbf{x}}S_\epsilon(\mathbf{x}, t) = P(t)\mathbf{x} + \mathbf{q}(t)$ and $\mathrm{Cov}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}] = \epsilon\nabla_{\mathbf{x}}^2 S_\epsilon(\mathbf{x}, t) = \epsilon P(t)$, respectively. Moreover, the MAP estimator and posterior mean estimator are equivalent in this case (since the posterior is Gaussian) and can be computed as $\boldsymbol{\theta} = \nabla_{\mathbf{x}}S_\epsilon(\mathbf{x}, t) = P(t)\mathbf{x} + \mathbf{q}(t)$. Note that none of these formulas involve r, and hence if we only care about solving the associated learning problem, the ODE for r may be ignored. The ODE system (3.1) also has an analytical solution given by $P(s) = (\Lambda^{-1} + s\Phi_1^T\Phi_1)^{-1}$, $\mathbf{q}(s) = sP(s)\Phi_1^T\boldsymbol{y}_1$, which can be computed directly when the dimension n is small. However, when n is large, inverting an $n \times n$ matrix becomes inefficient, and using other numerical methods (e.g., the Runge–Kutta method, recursive least squares) to solve the Riccati ODEs (3.1) may be preferable.

Now consider the case where we have $N > 1$ data points. Then the solution S_ϵ to the multi-time viscous HJ PDE (2.8) satisfies $S_\epsilon(\mathbf{x}, t_1, \ldots, t_N) = \frac{1}{2}\mathbf{x}^T P(\sum_{i=1}^N t_i)\mathbf{x} + \mathbf{q}(\sum_{i=1}^N t_i)^T\mathbf{x} + r(\sum_{i=1}^N t_i)$. Here $P: [0, +\infty) \to \mathbb{R}^{n\times n}$, $\mathbf{q}: [0, +\infty) \to \mathbb{R}^n$, and $r: [0, +\infty) \to \mathbb{R}$ are three continuous functions that satisfy the following ODE system over the interval $s \in [\sum_{i=1}^{j-1} t_i, \sum_{i=1}^j t_i]$ for each $j = 1, \ldots, N$:

(3.2)
$$\begin{cases} \dot{P}(s) + P(s)\Phi_j^T\Phi_j P(s) = 0, \\ \dot{\mathbf{q}}(s) + P(s)\Phi_j^T(\Phi_j\mathbf{q}(s) - \boldsymbol{y}_j) = 0, \\ \dot{r}(s) + \frac{1}{2}\|\Phi_j\mathbf{q}(s) - \boldsymbol{y}_j\|^2 + \frac{\epsilon}{2}Tr(\Phi_j^T\Phi_j P(s)) = 0, \end{cases}$$
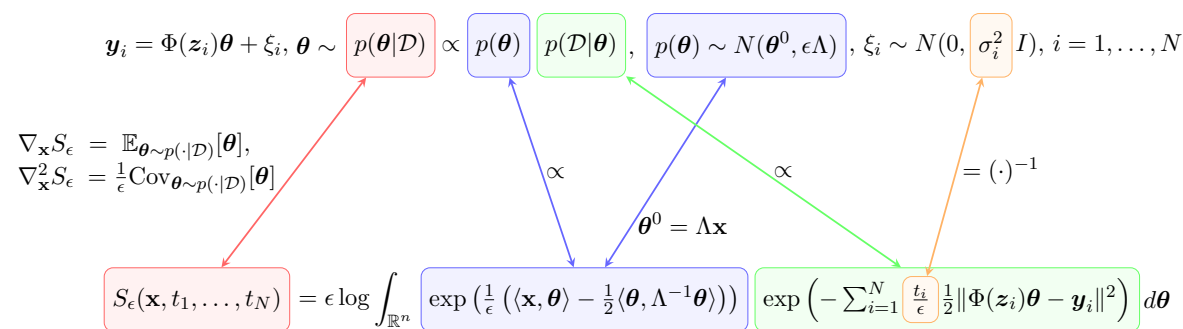
where the initial condition is the same as in (3.1). The posterior mean, posterior covariance, and MAP/posterior mean estimator are computed identically as in the $N = 1$ case but with $t = \sum_{i=1}^N t_i$. This ODE system also has an analytical solution: $P(s) = (\Lambda^{-1} + \sum_{i=1}^{j-1} t_i\Phi_i^T\Phi_i + (s - \sum_{i=1}^{j-1} t_i)\Phi_j^T\Phi_j)^{-1}$, $\mathbf{q}(s) = P(s)(\sum_{i=1}^{j-1} t_i\Phi_i^T\boldsymbol{y}_i + (s - \sum_{i=1}^{j-1} t_i)\Phi_j^T\boldsymbol{y}_j)$ for $s \in [\sum_{i=1}^{j-1} t_i, \sum_{i=1}^j t_i]$,

and $j = 1, \ldots, N$. When n is relatively small and there are no computational restrictions on accessing all N data points at once, these analytical formulas can be used to compute the posterior mean and covariance. However, when the dimension n is large or there are computational restrictions on the storage of or access to the entire dataset, using numerical methods to solve the Riccati ODEs (3.2) may be more advantageous than using the analytical formulas. In particular, our Riccati-based approach provides potential benefits when incrementally updating the learned models. Note that while incremental, online, or recursive Gaussian process regression (GPR) techniques [10, 43, 19, 29, 9, 4, 33, 37, 45] could also be employed to sequentially update the learned models, such techniques only provide a discrete perspective for these model updates. In contrast, our Riccati-based approach provides a continuous perspective that gives us access to a continuous flow of solutions that allows for the addition and removal of data points as well as continuous hyperparameter tuning. Detailed discussions of some potential computational advantages are presented in the remainder of section 3. The theoretical connection using a Gaussian prior is summarized in Figure 3.

3.2. Updating the likelihood. In this section, we discuss how the likelihood can be updated efficiently using our Riccati-based methodology. Namely, we show how data points can be added or removed from the learned model without retraining on or accessing the entire previous training set and in a manner that is invariant to the order of the data points. We also show how this Riccati-based framework naturally yields a continuous flow of solutions that allows hyperparameters to be tuned continuously.

3.2.1. Adding and removing data points. First, consider adding a data point $(\boldsymbol{z}_{N+1}, \boldsymbol{y}_{N+1})$ to the training set $\mathcal{D}$. Recalling that we assume each data point is independent, the posterior distribution after adding the $(N+1)$th data point becomes

$$(3.3) \qquad p(\boldsymbol{\theta}|\mathcal{D}, (\boldsymbol{z}_{N+1}, \boldsymbol{y}_{N+1})) \propto p(\boldsymbol{\theta}|\mathcal{D}) \exp\left(-\frac{1}{2\sigma_{N+1}^2}\|\Phi_{N+1}\boldsymbol{\theta} - \boldsymbol{y}_{N+1}\|^2\right),$$

Figure 3. *(See section 3.1) Mathematical formulation of the connection between a Bayesian inference problem with a linear model, Gaussian likelihood, and a Gaussian prior (top) and the solution to a multi-time viscous HJ PDE with a quadratic Hamiltonian and a quadratic initial condition (bottom). The content of this illustration is a special case of the connection in Figure 2, where the prior is Gaussian (set $g(\boldsymbol{\theta}) = \frac{1}{2}\langle\boldsymbol{\theta}, \Lambda^{-1}\boldsymbol{\theta}\rangle$ in Figure 2). The colors indicate the associated quantities. The arrow labels describe how the boxed quantities are related.*

where $\Phi_{N+1} = \Phi(\boldsymbol{z}_{N+1})$. In the HJ PDE framework, this corresponds to augmenting the multi-time system (2.8) with the following HJ PDE:

(3.4)

$$\partial_{t_{N+1}} S_\epsilon(\mathbf{x}, t_1, \ldots, t_{N+1}) + \frac{1}{2}\|\Phi_{N+1}\nabla_\mathbf{x} S_\epsilon(\mathbf{x}, t_1, \ldots, t_{N+1}) - \boldsymbol{y}_{N+1}\|^2 + \frac{\epsilon}{2}\nabla_\mathbf{x} \cdot (\Phi_{N+1}^T \Phi_{N+1} \nabla_\mathbf{x} S_\epsilon(\mathbf{x}, t_1, \ldots, t_{N+1})) = 0.$$

Thus, adding an additional data point can be interpreted as evolving a new time variable t_{N+1} from 0 to $\frac{\epsilon}{\sigma_{N+1}^2}$, where σ_{N+1}^2 denotes the variance of the $(N+1)$th data point. As such, the new posterior mean

(3.5)
$$\mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D},(\boldsymbol{z}_{N+1},\boldsymbol{y}_{N+1}))}[\boldsymbol{\theta}] = P\left(\sum_{i=1}^{N+1} t_i\right)\mathbf{x} + \mathbf{q}\left(\sum_{i=1}^{N+1} t_i\right)$$

and posterior covariance

(3.6)
$$\mathrm{Cov}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D},(\boldsymbol{z}_{N+1},\boldsymbol{y}_{N+1}))}[\boldsymbol{\theta}] = \epsilon P\left(\sum_{i=1}^{N+1} t_i\right)$$

can be computed by solving the Riccati ODE (3.2) (with index $j = N+1$) on $(\sum_{i=1}^N t_i, \sum_{i=1}^{N+1} t_i)$ with initial conditions $P(\sum_{i=1}^N t_i) = \frac{\Sigma_N}{\epsilon}$ and $\mathbf{q}(\sum_{i=1}^N t_i) = \mu_N - \frac{1}{\epsilon}\Sigma_N\mathbf{x}$, where $\mu_N := \mathbb{E}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$ and $\Sigma_N := \mathrm{Cov}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$ denote the original posterior mean and covariance, respectively.

Next, we discuss removing one data point $(\boldsymbol{z}_k, \boldsymbol{y}_k)$, $k \in \{1, \ldots, N\}$, from $\mathcal{D}$. In the HJ PDE framework, removing the kth data point corresponds to evolving the time variable t_k from $\frac{\epsilon}{\sigma_k^2}$ to 0. Thus, the new posterior mean and covariance resulting from deleting the kth data point can be computed as $P(\sum_{i=1}^N t_i - t_k)\mathbf{x} + \mathbf{q}(\sum_{i=1}^N t_i - t_k)$ and $\epsilon P(\sum_{i=1}^N t_i - t_k)$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.2) (with index $j = k$) backwards on $(\sum_{i=1}^N t_i - t_k, \sum_{i=1}^N t_i)$ with terminal conditions $P(\sum_{i=1}^N t_i) = \frac{\Sigma_N}{\epsilon}$ and $\mathbf{q}(\sum_{i=1}^N t_i) = \mu_N - \frac{1}{\epsilon}\Sigma_N\mathbf{x}$, where we again denote by μ_N, Σ_N the original posterior mean and covariance using all N data points.

Note that when adding or removing a data point, we do not require access to the entire previous dataset $\mathcal{D}$. Instead, we only require access to the previous posterior mean μ_N, the previous posterior covariance Σ_N, and the data point to be added or removed; all of the information about the rest of the training data remains encoded in the solution to the HJ PDE. As such, our Riccati-based approach provides potential computational and memory advantages when adding or removing data points by allowing the posterior distribution to be updated without having to retrain on the entire dataset. Additionally, note that, due to our assumption of independent data, the posterior mean and covariance are invariant to the order of the data points. As a result, adding or removing data points using our Riccati-based approach is also invariant to the order of the data; i.e., the final learned model will be the same regardless of the order in which data points are added or removed.

3.2.2. Tuning the variance. Tuning the variance σ_k^2 of the kth data point can be done using a similar methodology similar to that for adding or removing a data point. If we decrease the hyperparameter from σ_k^2 to $\tilde{\sigma}_k^2$, the new posterior mean and covariance can be computed as $P(\frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2})\mathbf{x} + \mathbf{q}(\frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2})$ and $\epsilon P(\frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2})$, respectively, where $P, \mathbf{q}$ are obtained by solving

the Riccati ODE (3.2) (with index $j = k$) forward from $s = 0$ to $s = \frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}$ with initial conditions $P(0) = \frac{\Sigma}{\epsilon}$ and $\mathbf{q}(0) = \mu - \frac{1}{\epsilon}\Sigma\mathbf{x}$, where we denote by μ, Σ the original posterior mean and covariance computed using the original value of σ_k^2. Note that every $s \in [0, \frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}]$ can be written as $\frac{\epsilon}{\hat{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}$ for some $\hat{\sigma}_k^2 \in [\tilde{\sigma}_k^2, \sigma_k^2]$. Hence, as we evolve the Riccati ODE from $s = 0$ to $s = \frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}$, we simultaneously obtain a flow of solutions corresponding to each $\hat{\sigma}_k^2 \in [\tilde{\sigma}_k^2, \sigma_k^2]$, or, in other words, we tune the variance continuously from σ_k^2 to $\tilde{\sigma}_k^2$.

Similarly, if we increase the hyperparameter from σ_k^2 to $\tilde{\sigma}_k^2$, the new posterior mean and covariance can be computed as $P(0)\mathbf{x} + \mathbf{q}(0)$ and $\epsilon P(0)$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.2) (with index $j = k$) backward from $s = \frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}$ to $s = 0$ with terminal conditions $P(\frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}) = \frac{\Sigma}{\epsilon}$ and $\mathbf{q}(\frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}) = \mu - \frac{1}{\epsilon}\Sigma\mathbf{x}$. As in the previous case, evolving the Riccati ODE from $s = \frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}$ to $s = 0$ again corresponds to continuously tuning the variance from σ_k^2 to $\tilde{\sigma}_k^2$, and hence every value of $P(s)\mathbf{x} + \mathbf{q}(s)$ and $\epsilon P(s)$, $s \in [0, \frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}]$ corresponds to the posterior mean and covariance using a different variance for the kth data point.

3.3. Updating the prior: Tuning hyperparameters. In this section, we discuss how our Riccati-based approach can be used to update the mean and covariance of the Gaussian prior. Tuning these hyperparameters may improve the inference accuracy of the learned model by updating the prior as new information becomes available.

Changing the prior mean $\boldsymbol{\theta}^0$ simply requires evaluating the solution to the corresponding HJ PDE at a different point $\mathbf{x} = \Lambda^{-1}\boldsymbol{\theta}^0$ in space, which, in turn, only requires the posterior mean to be updated with the new $\mathbf{x}$-value using some matrix-vector multiplications and addition, while the posterior covariance remains unchanged. Hence, tuning the prior mean is relatively cheap computationally.

Consider changing the prior covariance from $\epsilon\Lambda$ to $\epsilon\tilde{\Lambda}$. Originally, the prior covariance appears in the initial condition of the sequence of Riccati ODEs (3.2). However, we do not have to re-solve the entire sequence of Riccati ODEs in order to update the prior covariance. Instead, we reinterpret the prior covariance as part of a Hamiltonian of the corresponding multi-time HJ PDE, which allows it to be tuned using only the results of training with the original prior. Denote the original posterior mean and covariance (computed using the original prior) by μ_{pos} and Σ_{pos}, respectively. Then the new posterior mean $\tilde{\mu}_{pos}$ and covariance $\tilde{\Sigma}_{pos}$ using the new prior covariance $\epsilon\tilde{\Lambda}$ can be obtained using Riccati ODEs as follows. First, solve the Riccati ODE (3.2) forward from $s = 0$ to $s = 1$ with $\Phi_j = \tilde{\Lambda}^{-1/2}$, $\boldsymbol{y}_j = 0$, and initial conditions $P(0) = \frac{\Sigma_{pos}}{\epsilon}$ and $\mathbf{q}(0) = \mu_{pos} - \frac{1}{\epsilon}\Sigma_{pos}\mathbf{x}$. This step incorporates information about the new prior covariance into the model and yields solutions $P(1)$ and $\mathbf{q}(1)$. Next, solve the Riccati ODE (3.2) backward from $s = 1$ to $s = 0$ with $\Phi_j = \Lambda^{-1/2}$, $\boldsymbol{y}_j = 0$, and terminal conditions $\tilde{P}(1) = P(1)$ and $\tilde{\mathbf{q}}(1) = \mathbf{q}(1)$, where $P(1), \mathbf{q}(1)$ are the solutions from the previous step. This step removes information about the old prior covariance from the model and yields the solution $\tilde{P}(0)$ and $\tilde{\mathbf{q}}(0)$. Finally, the new posterior mean and covariance are computed as $\tilde{\mu}_{pos} = \tilde{P}(0)\mathbf{x} + \tilde{\mathbf{q}}(0)$ and $\tilde{\Sigma}_{pos} = \epsilon\tilde{P}(0)$, respectively.

When the new prior covariance $\epsilon\tilde{\Lambda}$ is a scaling of the original prior covariance $\epsilon\Lambda$ (i.e., $\tilde{\Lambda} = \alpha\Lambda$ for some $\alpha > 0$), the posterior can be updated by solving only one Riccati ODE instead of two. Again denote the original posterior mean and covariance computed using the original prior by μ_{pos} and Σ_{pos}. If $\alpha > 1$, then the new posterior mean and covariance can

be computed as $P(\frac{1}{\alpha}) + \mathbf{q}(\frac{1}{\alpha})\mathbf{x}$ and $\epsilon P(\frac{1}{\alpha})$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.2) backward from $s = 1$ to $s = \frac{1}{\alpha}$ with $\Phi_j = \Lambda^{-1/2}$, $\mathbf{y}_j = 0$, and terminal conditions $P(1) = \frac{\Sigma_{pos}}{\epsilon}$ and $\mathbf{q}(1) = \mu_{pos} - \frac{1}{\epsilon}\Sigma_{pos}\mathbf{x}$. Similarly, if $\alpha \in (0, 1)$, then the new posterior mean and covariance can be computed as $P(\frac{1}{\alpha}) + \mathbf{q}(\frac{1}{\alpha})\mathbf{x}$ and $\epsilon P(\frac{1}{\alpha})$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.2) forward from $s = 1$ to $s = \frac{1}{\alpha}$ with $\Phi_j = \Lambda^{-1/2}$, $\mathbf{y}_j = 0$, and initial conditions $P(1) = \frac{\Sigma_{pos}}{\epsilon}$ and $\mathbf{q}(1) = \mu_{pos} - \frac{1}{\epsilon}\Sigma_{pos}\mathbf{x}$. In both cases, evolving the Riccati ODE yields a continuous flow of solutions corresponding to tuning the hyperparameter α continuously; namely, every s between 1 and $1/\alpha$ corresponds to another solution with a different prior covariance $\frac{\epsilon}{s}\Lambda$.

4. Examples in scientific machine learning. In this section, we apply our methodology to three examples from SciML, in which ML tools are employed to solve ODEs or PDEs [42, 21]. In each example, we learn the solution $u : \mathbb{R}^\ell \to \mathbb{R}^m$ and the source term $f : \mathbb{R}^\ell \to \mathbb{R}^m$ of a differential equation

$$(4.1) \qquad \begin{cases} \mathcal{F}[u] = f & \text{in } \Omega, \\ \mathcal{B}[u] = b & \text{on } \partial\Omega \end{cases}$$

using noisy data and quantify corresponding epistemic uncertainties [20, 38, 59]. In the differential equation above, $\mathcal{F}$ and $\mathcal{B}$ represent linear differential and boundary operators, respectively. Following the framework in section 2.1, we estimate u using the linear model $u_{\boldsymbol{\theta}} = \Phi(\cdot)\boldsymbol{\theta}$, where $\Phi : \mathbb{R}^\ell \to \mathbb{R}^{m \times n}$ is the matrix whose columns are the basis functions $\phi_i : \mathbb{R}^\ell \to \mathbb{R}^m$, $i = 1, \ldots, n$. As a result, we also estimate f using a linear model $f_{\boldsymbol{\theta}} = \mathcal{F}[u_{\boldsymbol{\theta}}](\cdot) = \mathcal{F}[\Phi](\cdot)\boldsymbol{\theta}$.

We use the predicted mean associated with our learned models as our prediction of the quantities of interest; i.e., our prediction for u is $\mathbb{E}[u_{\boldsymbol{\theta}}|\mathcal{D}] = \Phi\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}]$ and our prediction for f is $\mathbb{E}[f_{\boldsymbol{\theta}}|\mathcal{D}] = \mathcal{F}[\Phi]\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}]$. Note that considering the predicted mean of our learned models is equivalent to taking $\boldsymbol{\theta}$ to be the posterior mean estimate. We compute the predicted uncertainty of our learned models as their covariances: $\text{Cov}[u_{\boldsymbol{\theta}}|\mathcal{D}] = \Phi\text{Cov}[\boldsymbol{\theta}|\mathcal{D}]\Phi^T$ and $\text{Cov}[f_{\boldsymbol{\theta}}|\mathcal{D}] = \mathcal{F}[\Phi]\text{Cov}[\boldsymbol{\theta}|\mathcal{D}]\mathcal{F}[\Phi]^T$. The predicted uncertainty can then be used to help inform the learning process (e.g., by providing a metric for determining when to stop learning or where more data may be needed). When $m = 1$, the covariance is scalar-valued and we can take twice its square root (i.e., twice the posterior standard deviation) as the predicted uncertainty. In each example, we assume Gaussian likelihood and a Gaussian prior and apply the Riccati-based methodology from section 3 to compute the posterior mean and covariance. For demonstration purposes, we use the 4th-order Runge–Kutta (RK4) method to numerically solve the corresponding Riccati ODEs, although any other appropriate numerical method could be used instead. Details for the choice of hyperparameters in each example can be found in Appendix A. Code for all of the examples will be made publicly available at https://github.com/ZongrenZou/HJPDE4UQSciML after the paper is published.

4.1. Solving a boundary-value ODE problem. Consider the following boundary value problem:

$$(4.2) \qquad \begin{cases} \kappa\dfrac{d^4u}{dt^4}(\tau) + \beta\dfrac{d^2u}{dt^2}(\tau) + u(\tau) = f(\tau), \tau \in (0, 1), \\ u(0) = u_0, u(T) = u_T, u'(0) = u'_0, u'(T) = u'_T, \end{cases}$$

where $\kappa = 0.0001, \beta = 0.01$ are known constants. In this example, we solve (4.2) using noisy measurements of u_0, u_T, u_0', u_T' and noisy measurements of the source term f at different times τ, which we denote by $\{(\tau_i, f_i)\}_{i=1}^{N_f}$. To demonstrate the capabilities and potential computational advantages of our Riccati-based approach, we consider the following three learning scenarios:

    A. Continual learning: stream data of f sequentially in τ (section 4.1.1),

    B. Hyperparameter tuning: continuously tune the standard deviation of the prior (section 4.1.2),

    C. Outlier removal: remove excessively noisy data from the learned model (Appendix B).

In each scenario, we use the truncated Karhunen-Loève (KL) expansion of a Gaussian process as our linear model. Specifically, we use the leading $n = 30$ terms of the KL expansion:

$$(4.3) \qquad u_{\boldsymbol{\theta}}(\tau) = \sum_{k=1}^{n} \theta_k \sqrt{\alpha_k} \tilde{\phi}_k(\tau),$$

where $\tilde{\phi}_k, \alpha_k$ are the eigenfunctions and eigenvalues of the kernel

$$(4.4) \qquad k(x_1, x_2) = \exp\left(-\frac{|x_1 - x_2|}{0.05}\right) \quad \forall x_1, x_2 \in [-10, 10].$$

In other words, we use the basis functions $\{\phi_k(\tau) = \sqrt{\alpha_k} \tilde{\phi}_k(\tau), k = 1, \ldots, n\}$ [52]. For more details, the analytic expansion can be found in [15, 51]. Unless otherwise specified, we also assume that the prior on $\boldsymbol{\theta}$ is standard independent Gaussian, i.e., $\boldsymbol{\theta} \sim N(0, I)$, where I is the $n \times n$ identity matrix. The exact solution is assumed to be $u(\tau) = \exp(-2\tau)\sin(15\tau)$ (but treated as unknown a priori), and we set $T = 1$. The data of u_0, u_T (u_0', u_T', respectively) are corrupted by additive Gaussian noise with mean zero and standard deviation 0.01 (0.001, respectively).

4.1.1. Case A: Continual learning. In this scenario, we learn u and f using continual learning [35, 49] to demonstrate the potential of our Riccati-based approach for real-time inferences. Continual learning refers to learning scenarios in which data is accessed in a stream and learned models are updated incrementally as new data becomes available. In some cases, the historical data is also assumed to become inaccessible after it is incorporated into the learned model, which often leads to catastrophic forgetting [23, 35], i.e., the abrupt degradation in the performance of learned models on previous tasks upon training on new tasks. Here we assume that measurements of f are streamed sequentially in τ (assume that one measurement of f becomes available every $\Delta\tau = \frac{T}{200}$) and that the previous data cannot be accessed again after a new measurement arrives. As such, we want to update our learned models as soon as new data becomes available. We also assume that the measurements of f are corrupted by additive Gaussian noise with mean zero and standard deviation 0.2, so that a Bayesian approach may be useful. We apply the Riccati-based methodology from section 3.2.1, which is naturally amenable to data streaming. In particular, this Riccati-based approach provides computational advantages over more standard SciML and UQ techniques in continual learning settings by using only information of the newly available data to update the learned model instead of retraining on the entire dataset. These computational advantages become more significant in long-term integration problems, in which retraining on and storage of

(a) Inferences of u with UQ.

(b) Inferences/fitting of f with UQ.
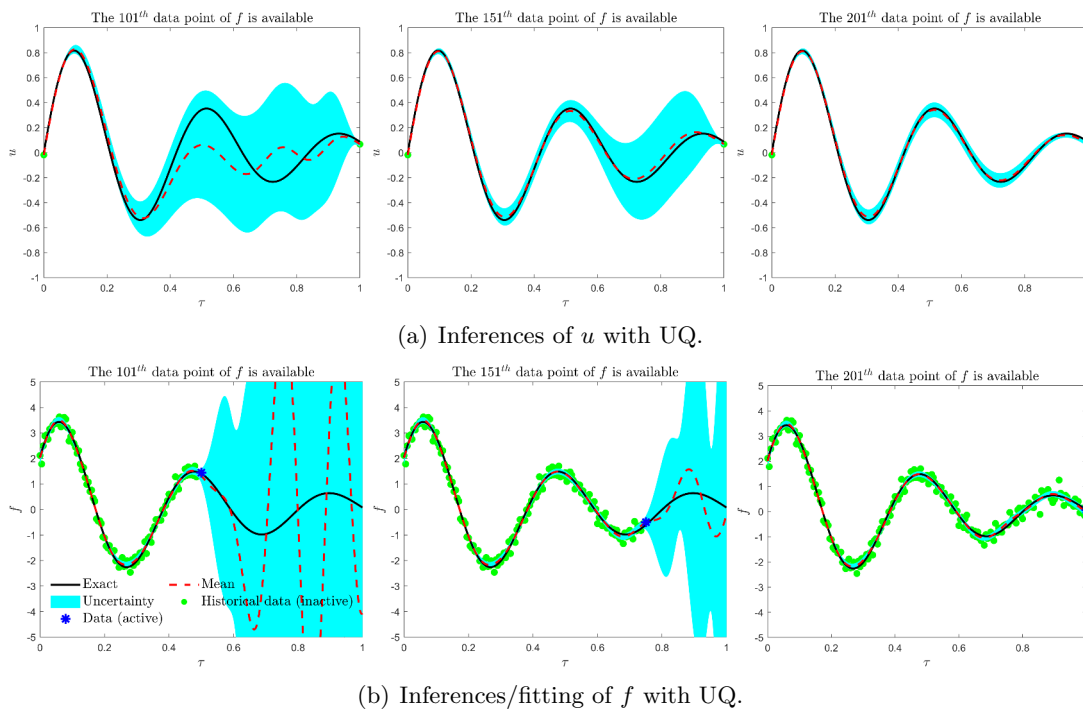
Figure 4. *Results of solving* (4.2) *using continual learning and our Riccati-based approach.* (a) *and* (b) *show the predicted mean and uncertainty of u and f, respectively, after the 101th, 151th, and 201th noisy data points of f become available. Our Riccati-based approach naturally coincides with the continual learning framework while inherently avoiding catastrophic forgetting. Our approach allows us to incrementally update the learned models without accessing the historical data (●), while also providing a quantitative metric for our confidence in the learned models in the form of the predicted uncertainties (■). We observe that regions of low predicted uncertainty generally coincide with regions of high inference accuracy, which implies that this confidence metric is a good indicator of the reliability of the model.*

the historical data becomes considerably more expensive. Furthermore, although our Riccati-based approach does not rely on historical data, it does not suffer from catastrophic forgetting, as all of the information from the previous data remains intrinsically encoded in the solution to the corresponding HJ PDE.

Figure 4 displays the predicted means and uncertainties of u and f after the 101th, 151th, and 201th noisy measurements of f become available. We see that as more data points of f are incorporated into the learned models, the predicted means of both u and f more closely match their exact values and the predicted uncertainties shrink. Moreover, we see that regions of high predicted uncertainty generally coincide with low inference accuracy, which shows that our confidence in the learned models is correlated with their reliability. We also observe that the predicted uncertainty of u does not develop in the same way as that of f. While the predicted uncertainty of f is generally uniformly small on $(0, \tau^*)$ and large on $[\tau^*, T]$, where τ^* denotes the τ-coordinate of the currently available data point, u can often still have relatively large predicted uncertainty on $(0, \tau^*)$ since we do not learn u directly. Note that u always has low predicted uncertainty on the boundaries as the boundary data were incorporated into the

Table 1

Relative L_2 errors of the predicted means of u and f when using continual learning and our Riccati-based approach to solve (4.2) *after different amounts of noisy measurements of f are incorporated into the learned models. We achieve high accuracy inferences after all the data is incorporated.*

	After the 101th data point	After the 151th data point	After the 201th data point
Error of u	41.77%	6.79%	3.88%
Error of f	217.27%	24.48%	3.39%

(a) Validation error.

(b) Predicted mean of u at different τ as the function of σ.
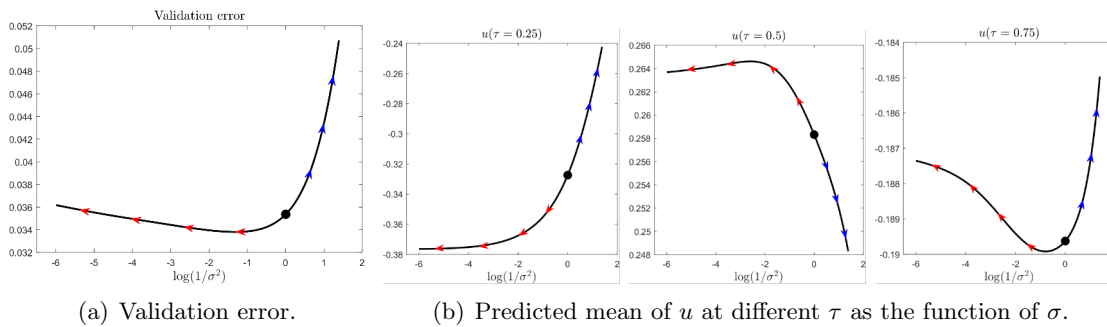
Figure 5. *Results of tuning the standard deviation σ of the prior using our Riccati-based approach to solve* (4.2). *(a) shows how the validation error changes as σ is tuned. (b) shows the flow with respect to σ of the predicted mean of u at different times ($\tau = 0.25, 0.5, 0.75$). The black dot represents the value using the original choice of $\sigma = 1$, and the arrows represent the direction of the flow of solutions as we increase σ from 1 (red; i.e., we evolve the corresponding Riccati ODE backward) or decrease σ from 1 (blue; i.e., we evolve the corresponding Riccati ODE forward). In contrast to conventional ML approaches, our Riccati-based approach allows the hyperparameter to be tuned continuously without retraining on or access to the training data.*

learned models at the beginning of the training. Table 1 presents the relative L_2 errors of the predicted means, which show that we achieve high inference accuracy after all the data has been incorporated. We compute the L_2 errors using the trapezoidal rule with a uniform grid of size 1001 over the whole domain $[0, T]$.

4.1.2. Case B: Tuning the standard deviation of the prior. In the previous case, the prior distribution was set to be independent standard Gaussian, i.e., $\boldsymbol{\theta} \sim N(0, \sigma^2 I)$, which is the usual choice for KL expansions of Gaussian processes [50]. However, this choice may not be appropriate for every problem, and tuning the hyperparameters of the prior may yield better results. In this section, we demonstrate how the Riccati-based approach from section 3.3 can be used to tune the standard deviation σ of the prior continuously. In this case, the training data consists of the same measurements of u and u' used in Case A and 41 measurements of f sampled equidistantly on $[0, T]$. The validation data consists of 10 measurements of f sampled randomly on $[0, T]$ following a uniform distribution. Both the training and the validation data are corrupted by additive Gaussian noise with mean zero and standard deviation 0.2.

Figure 5 shows how the validation error and predicted mean of u develop as σ is tuned. Figure 5(b) shows that the predicted mean of u varies as σ changes, and Figure 5(a) shows that the lowest validation error is not achieved with the original choice of $\sigma = 1$, which highlights the importance of hyperparameter tuning. Recall that tuning σ corresponds to evolving a Riccati ODE in time. Hence, tuning σ can be done continuously and results in a continuous

flow of solutions, where each solution corresponds to a different choice of σ. This flow of solutions is obtained numerically via the intermediary steps of RK4 and is represented by the arrows in Figure 5. In Figure 5(a), tuning σ results in a continuous flow of validation errors, and in Figure 5(b), it results in a continuous flow of inferences of u. Our Riccati-based approach allows hyperparameter values to be explored flexibly and continuously and, in contrast to more standard ML techniques, does not require retraining on or access to the training dataset during the tuning process.

4.2. Solving the 1D steady-state advection-diffusion equation. Consider the following steady-state advection-diffusion equation with Dirichlet boundary conditions:

$$(4.5) \qquad \begin{cases} D\dfrac{\partial^2 u}{\partial x^2}(x) + \kappa\dfrac{\partial u}{\partial x}(x) = f(x), & x \in (0,1), \\ u(0) = u(1) = 0, \end{cases}$$

where $D = 0.001$ and $\kappa = 0.1$. The exact solution to this equation is chosen as $u(x) = \sin(6\pi x)\cos^2(4\pi x)$. We solve (4.5) using noisy measurements of f at different x, which we denote by $\{(x_i, f_i)\}_{i=1}^N$. To demonstrate the versatility of our Riccati-based approach in handling different learning settings, we consider the following two cases:

    A. The data of f are corrupted by large-scale noise but are cheap to obtain (section 4.2.1),

    B. The data of f are corrupted by small-scale noise but are expensive to obtain (section 4.2.2).

Case A uses a large-scale problem with a huge amount of data to show that we can achieve high accuracy despite a large amount of noise, while case B uses active learning [44] to search for where new measurements are needed. In both cases, we use the truncated KL expansion of the Brownian bridge to model u. Specifically, we use the leading $n = 50$ terms of the KL expansion [15]:

$$(4.6) \qquad u_{\boldsymbol{\theta}}(x) = \sum_{k=1}^{n} \theta_k \frac{\sqrt{2}\sin(k\pi x)}{k\pi}.$$

Note that this choice of basis functions automatically enforces the Dirichlet boundary conditions. We take the prior to be independent standard Gaussian, i.e., $\boldsymbol{\theta} \sim N(0, I)$.

4.2.1. Case A: Large-scale data with high noise level. In this case, we demonstrate the computational efficiency of our Riccati-based approach on a large-scale problem. Specifically, we assume that measurements of f are corrupted by a large amount of noise but are cheap to acquire. Thus, to obtain accurate predictions and compensate for the high noise level, we need a lot of data. Working with such a large dataset in its entirety would be extremely expensive in terms of both computations and memory. Instead, we apply our Riccati-based approach from section 3.2.1 to continuously learn our model without have to train on or store the entire dataset. In this case, the measurements of f are sampled uniformly randomly on $[0,1]$ and corrupted by additive Gaussian noise with mean zero and standard deviation 2 (e.g., see Figure 6(b)). We keep sampling f until a satisfactory result is obtained (i.e., until the predicted uncertainty is sufficiently low).

Figure 6 shows inferences of u and f after $1,000$, $5,000$, and $100,000$ data points of f are incorporated into the learned models. Note that the predicted uncertainty at the boundary

(a) Inferences of u with UQ.

(b) Inferences/fitting of f.
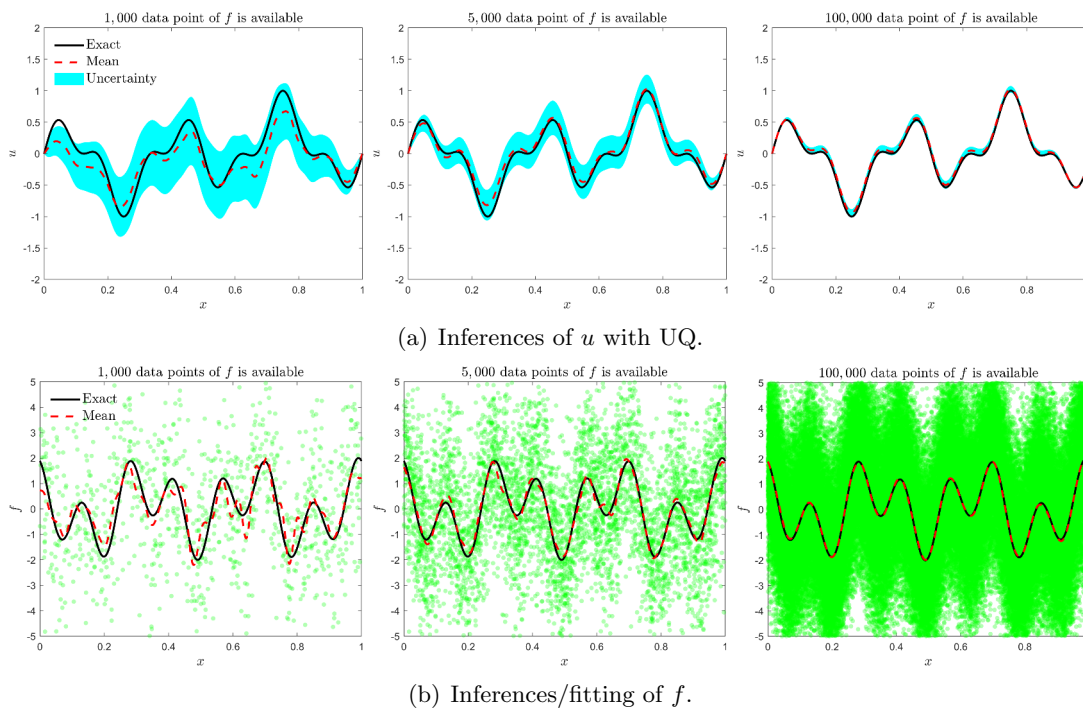
Figure 6. *Results of solving* (4.5) *using large-scale data with a high noise level and our Riccati-based approach.* (a) *shows the corresponding predicted mean and uncertainty* (■) *of* u. (b) *shows the inference/fitting of* f *using* 1,000, 5,000, *and* 100,000 *noisy data points. Note that we do not show the predicted uncertainty of* f *for clarity of presentation. Our Riccati-based approach updates the learned models incrementally using information from one data point* (•) *at a time, which provides significant computational and memory savings in big data regimes.*

points is always zero because our choice of basis functions automatically enforces the boundary conditions. Due to the high noise level, the inferences of u and f are inaccurate when only 1,000 data points are used. Additionally, the predicted uncertainty of u is relatively large, which indicates that the learned models are low confidence. As we incorporate more measurements of f into the learning process, the inferences become more accurate and the predicted uncertainty of u decreases. Once the predicted uncertainty is small enough, we determine that we are confident in our learned model and can stop collecting data. For example, in Figure 6(a), we see that the predicted uncertainty of u is significantly and sufficiently reduced after using 100,000 data points, and we stop learning. In Table 2, we observe that the relative L_2 errors of our inferences are also significantly reduced, confirming that our confidence in the model is a reasonable indicator of its accuracy. The L_2 errors are computed using the trapezoidal rule with a uniform grid of size 1001 over the domain $[0, 1]$. We again note that by using the Riccati-based methodology from section 3.2.1, we avoid having to retrain every time we acquire new data points and never have to work with all 100,000 data points at once. Instead, our Riccati-based approach only requires information about one data point at a time, which provides significant computational and memory advantages in these big data regimes.

Table 2

Relative L_2 errors of the predicted means of u and f using different amounts of noisy measurements of f to solve (4.5). Our Riccati-based approach updates the learned models without requiring retraining when new data becomes available or storing and processing large datasets in their entirety.

	1,000 data points	5,000 data points	100,000 data points
Error of u	44.44%	17.19%	9.49%
Error of f	41.89%	16.72%	3.81%

(a) Inferences/fitting of f with UQ.

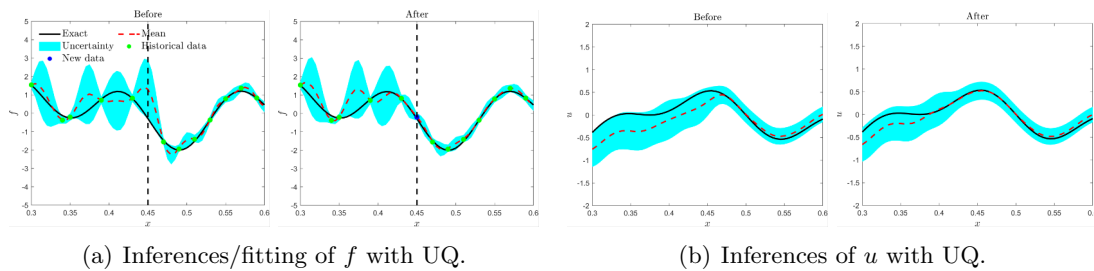(b) Inferences of u with UQ.

Figure 7. *Intermediate result for adding the 46th data point when using active learning and our Riccati-based approach to solve (4.5). The left side of (a) and (b) shows the inferences of f and u, respectively, before the placement of the new sensor and incorporation of the 46th data point. The right side of (a) and (b) shows the updates to f and u after these actions. The vertical black dashed line represents the location where the 46th sensor will be placed. Note that we zoom in on the region around the new sensor instead of showing the full domain for clarity of comparison. Following an active learning framework, we repeat the following steps: compute the predicted mean and uncertainty of u and f, place a new sensor at the location of the largest predicted uncertainty (■) of f, and obtain a new measurement of f (*) at the new sensor location. Our Riccati-based approach naturally complements this framework, as it can efficiently perform repeated updates of the predicted mean and uncertainty.*

4.2.2. Case B: Expensive data with low noise level. In this case, we demonstrate the versatility of our Riccati-based methodology by showing how it naturally complements an active learning approach. Active learning refers to the ML paradigm where the training data are sampled dynamically until a desired result is achieved (e.g., low generalization error) [48, 44, 25]. In this section, we focus on uncertainty-based active learning [25, 16, 36], in which we choose the new data point to be sampled from the location where the predicted uncertainty is largest. Assume that we have access to high quality data with low-level noise (here we use additive Gaussian noise with mean zero and standard deviation 0.1) but that the data are expensive to collect. For example, consider the scenario where high precision but expensive sensors are deployed to measure the value of f at particular locations x. As a result, we have limited access to data and must carefully select where sensors should be placed in order to obtain reliable inferences. Assume the potential locations for sensors form a 101-point uniform grid on $[0, 1]$. We use the predicted uncertainty of f to determine where the next sensor should be placed. Note that although we could have instead used the predicted uncertainty of u to choose the sensor location, using the predicted uncertainty of f is more consistent with the fact that we collect measurements of f. Following the active learning framework [44, 39, 36], we start with an empty dataset $\mathcal{D}$ and index $i = 0$ and then do the following:

1. Compute the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ based on the current dataset $\mathcal{D}$ as well as the predicted mean and uncertainty of u and f at all potential sensor locations. If the predicted uncertainty is sufficiently low, then terminate the procedure.

2. Identify the location where the predicted uncertainty of f is highest out of all remaining potential locations. Label this location as x_i, and place a sensor there to measure f (denote these measurements by f_i).

3. Update the current dataset $\mathcal{D}$ to $\mathcal{D} \cup \{(x_i, f_i)\}$ and the index i to $i+1$. Go back to step 1.

To perform steps 1 and 2, we employ the Riccati-based approach from section 3.2.1, in which the models and their predicted uncertainty are updated incrementally as new data become available. Note that the Riccati-based approach performs these updates without using historical data.

Figure 7 displays a zoom in of the intermediate result when adding the 46th measurement of f. The vertical black dashed line indicates where the predicted uncertainty of f is the highest (i.e., where the new sensor is placed). While the addition of the sensor significantly reduces the predicted uncertainty of f around that point, it only mildly reduces the predicted uncertainty of u since we do not learn u directly. Figure 13 in Appendix C displays intermediate results when adding the 47th sensor, which more dramatically reduces the predicted uncertainty of u. These results demonstrate how our Riccati-based approach can be seamlessly integrated into an active learning framework by allowing for continual updates of the models and uncertainty metrics that can be leveraged to interactively improve the learning process.
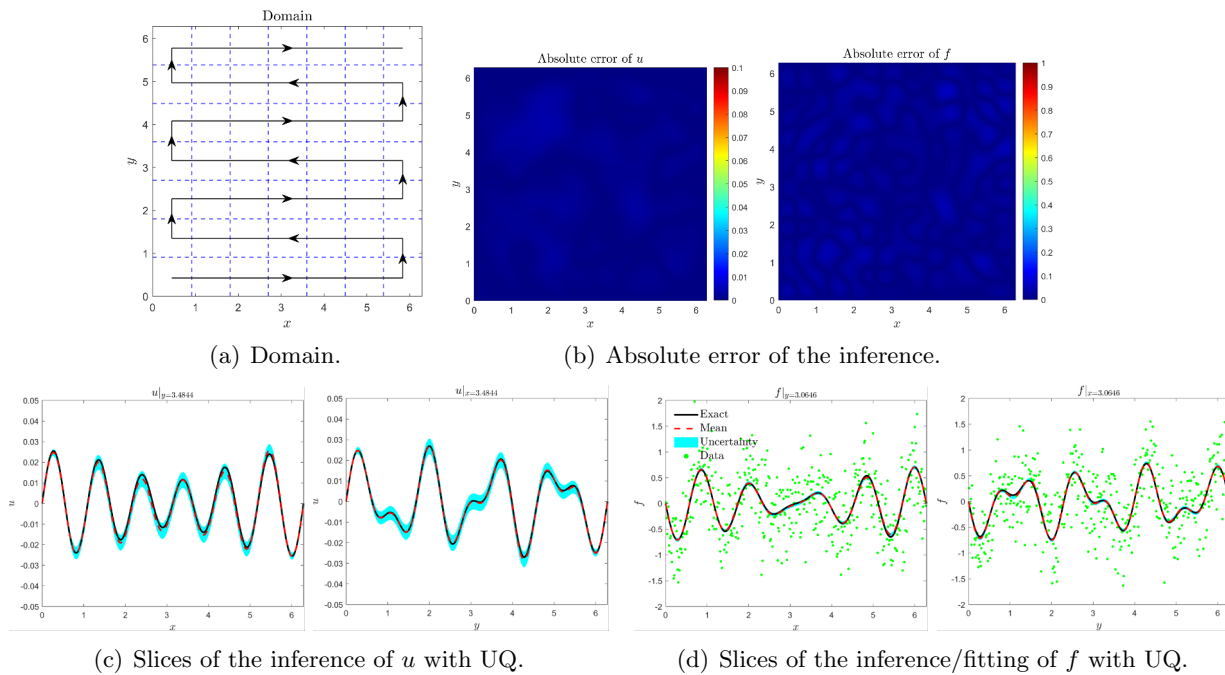
4.3. Solving the 2D Helmholtz equation. In this example, we solve the 2D Helmholtz equation with Dirichlet boundary conditions

$$(4.7) \quad \begin{cases} (\kappa^2 - \Delta)u(x,y) = f(x,y), & x,y \in [0, 2\pi], \\ u(x,0) = u(x,2\pi) = u(0,y) = u(2\pi,y) = 0, & x,y \in [0, 2\pi], \end{cases}$$

where $\kappa^2 = 1$ is the Helmholtz constant and Δ is the Laplacian operator. We consider the scenario where we have large-scale, noisy data of f but assume that computational limitations prevent us from being able to process or store all of the data at once. We use this example to show how our Riccati-based approach naturally overcomes these computational limitations. Specifically, we decompose the domain into multiple smaller subdomains and update our learned model on each piece one-by-one using continual learning and the Riccati-based methodology from section 3.2. Note that since the Riccati-based approach is invariant to the order of the data points, it does not require a specific method for decomposing the domain. To illustrate the flexibility of our approach, we decompose the domain $[0, 2\pi]^2$ uniformly into 7×7 equal subdomains and consider the following two patterns for traversing the subdomains:

A. A sequential pattern (section 4.3.1; e.g., see Figure 8(a)),

B. A multilevel order of traversal (section 4.3.2; e.g., see Figure 10).

In both cases, we employ a linear model $u_{\boldsymbol{\theta}}(x,y) = \sum_{k=1}^{n} \theta_k \phi_k(x,y)$, where $\{(x,y) \mapsto \phi_k(x,y)\}_{k=1}^{n} = \{(x,y) \mapsto \frac{\sqrt{2L}\sin(\frac{j\pi x}{L})}{j\pi} \frac{\sqrt{2L}\sin(\frac{k\pi y}{L})}{k\pi}\}_{j,k=1}^{\sqrt{n}}$, $n = 225$, and $L = 2\pi$. Note that these basis functions automatically enforce the boundary conditions. We learn the model using

(a) Domain.

(b) Absolute error of the inference.

(c) Slices of the inference of u with UQ.

(d) Slices of the inference/fitting of f with UQ.

Figure 8. *Results of solving (4.7) with noisy data of f using our Riccati-based approach while traversing the domain sequentially. (a) displays the 7×7 uniform domain decomposition and the sequential order of integration of the domain. (b) shows the absolute errors of the predicted means of u and f after all 49 subdomains have been visited. (c) and (d) show 1D slices of the inferences and predicted uncertainties ($\blacksquare$) of u and f, respectively. These results show how our Riccati-based approach can naturally be extended to solve higher-dimensional problems. Namely, by leveraging data streaming, our Riccati-based approach is able to overcome the computational and memory challenges related to the higher dimension of the PDE and subsequently increased dataset size.*

measurements of f that are corrupted by additive Gaussian noise with mean 0 and standard deviation 0.5. We define f by

$$(4.8) \qquad\qquad f(x, y) = \sin(6x)\sin(4y) - 0.8\sin(5x)\sin(7y),$$

and the solution to (4.7) can be analytically derived accordingly. To compute the accuracy of learned models, we evaluate our inferences of u and f on a uniform grid of size 450×450 over the domain. To learn $\boldsymbol{\theta}$, we take noisy measurements of f at each of the $200,704$ interior points of that grid. Hence, each subdomain contains $4,096$ measurements of f.

4.3.1. Case A: Sequential domain decomposition. In this case, we traverse the subdomains sequentially using the snaking pattern shown in Figure 8(a). This decomposition pattern naturally extends the sequential data streaming pattern used for the 1D problem in section 4.2.1 to the 2D problem considered here. As such, this traversal pattern intuitively coincides with the Riccati-based methodology for adding data points in section 3.2.1.

In Figure 8, we see that we learn u and f fairly well, but the accuracy of the predicted means of u and f remains compromised by the large amount of noise in the data. In particular, Figures 8(c)–(d) show that the exact values are still within the predicted uncertainty intervals

Table 3

Relative L_2 errors of the predicted means of u and f when solving (4.7) *using our Riccati-based approach and the sequential order of integration shown in Figure* 8(a). *The errors significantly reduce as more subdomains are visited but remain slightly high after all of the data is incorporated due to the high level of noise in the data.*

	25 subdomains	40 subdomains	44 subdomains	49 subdomains (all)
Error of u	897.75%	231.56%	93.19%	9.73%
Error of f	178.51%	62.74%	20.38%	2.32%
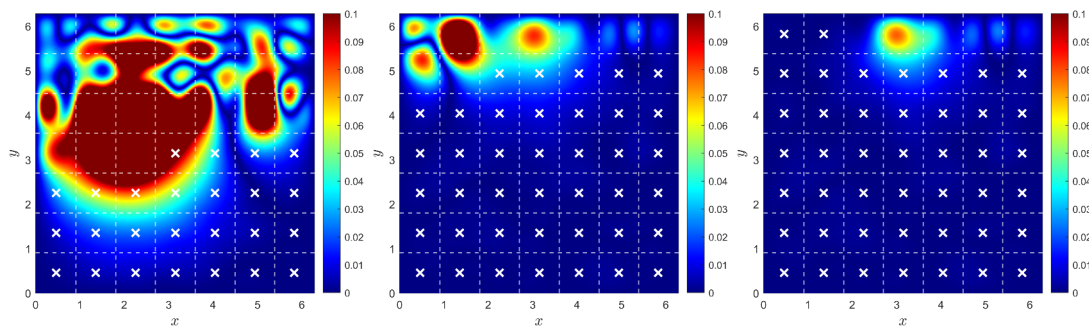
(a) Absolute error of the inference of u.
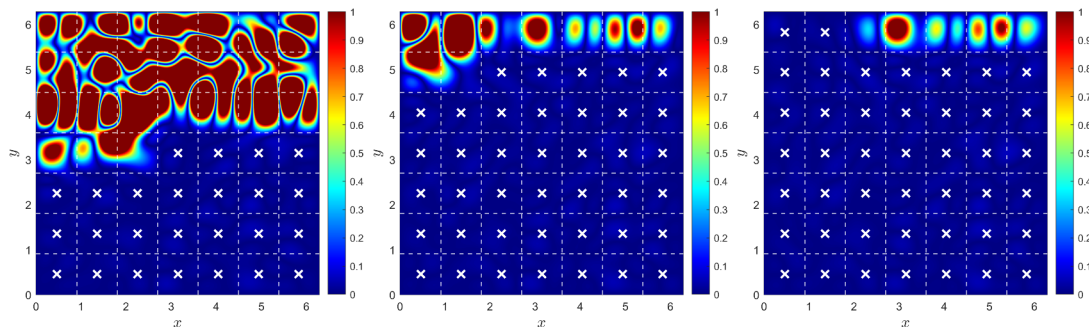
(b) Absolute error of the inference/fitting of f.

Figure 9. *Intermediate results of solving* (4.7) *using our Riccati-based approach while traversing the domain sequentially. We collect data from each subdomain sequentially following the pattern in Figure* 8(a). *(a) and (b) shows the absolute errors of the inferences of u and f, respectively. The white crosses indicate the subdomains that have already been visited. Our Riccati-based approach incrementally updates our predictions as we traverse the domain, which provides computational and memory advantages over having to process or store all of the data from the entire domain at once.*

around our predicted means, which indicates that our inferences are still reliable. In Figure 9 and Table 3, we present intermediate results for the errors of our inferences after the domain has been partially traversed. In Figure 9, we observe that the absolute error of the inference of f vanishes on each of the visited subdomains, whereas the absolute error of the inference of u is reduced on the visited subdomains but not necessarily eliminated. This behavior is due to the fact that we do not learn u directly and instead learn u from data of f and information from the PDE. In Table 3, we again see that the errors significantly reduce as we traverse domain, but the final error of our inference of u remains somewhat high due to the high noise
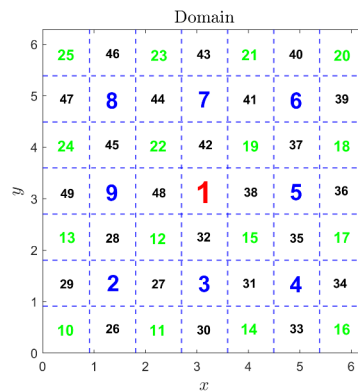
Figure 10. *Domain decomposition with a multilevel subdomain traversal. The level one subdomain is marked in* red*, the level two subdomains are marked in* blue *numbers, the level three subdomains are marked in* green *numbers, and the level four subdomains are marked in* black*.*

level of the data. Since our Riccati-based approach incrementally updates our inferences with one data point at a time, it does not suffer from any computational or memory scaling issues related to the large dataset size but does still depend on the numerical precision of the ODE solver.

4.3.2. Case B: Multilevel domain decomposition. In this section, we follow a multilevel order of a subdomain traversal inspired by multigrid methods [17, 27] to highlight the invariance of our Riccati-based approach to the order of the data. In the previous case, we incorporated the data sequentially. However, this order is not required, as our Riccati-based approach can incrementally incorporate the data into the learned models in any order. Specifically, we consider the multilevel traversal order shown in Figure 10. Level one consists of the center subdomain. Level two consists of the eight subdomains roughly centered around the level one subdomain. Level three consists of 16 subdomains, each corresponding to one corner of each of the level two subdomains. Level four consists of the remaining 24 subdomains. Using this type of multilevel order has the potential to allow the learning process to stop earlier (and hence use less data) because each level captures information across the entire domain. As a result, the error of the inferences is more likely to reduce uniformly across the domain. In contrast, the sequential traversal pattern used in section 4.3.1 is more likely to learn accurate predictions on the bottom half of the domain before learning any reasonably reliable prediction on the top half of the domain.

Figure 11 shows the intermediate results using this multilevel approach. As in the previous case, we see that the errors generally improve in/near the subdomains we have visited. The end result (not shown) after all of the subdomains have been visited is identical to Figures 8(b)–(d) with 9.73% relative L_2 error for u and 2.32% relative L_2 error for f, which demonstrates the invariance of our Riccati-based approach to the order of the data points.

5. Summary. In this paper, we established a new theoretical connection between Bayesian inference problems with linear models and Gaussian likelihoods and viscous HJ PDEs with quadratic Hamiltonians. As a first exploration of this connection, we specialized in Gaussian

(a) Absolute error of the inference of u.

(b) Absolute error of the inference/fitting of f.
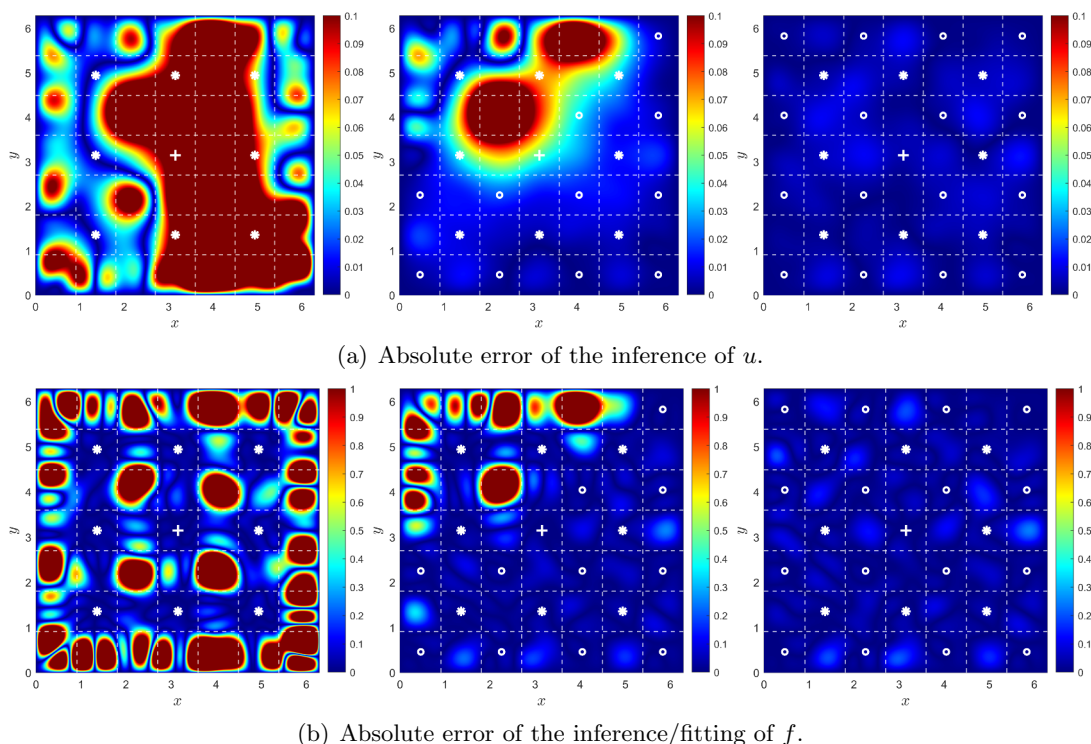
Figure 11. *Intermediate results of solving* (4.7) *using our Riccati-based approach while traversing the domain using the multilevel order in Figure* 10. (a) *and* (b) *show the absolute errors of the inferences of u and f, respectively. The level one subdomain is marked by the white cross, the level two subdomains are marked by white stars, and the level three subdomains are marked by white circles. Our Riccati-based approach incrementally incorporates data points into the predictions invariant to the order of the data. As such, the end result after all the data is incorporated is identical to Figures* 8(b)–(d), *despite the intermediate results differing.*

priors and leveraged the result to develop a new Riccati-based methodology for efficiently updating the predicted models. We then demonstrated some potential computational advantages of this Riccati-based approach by applying it to several UQ-based examples from SciML [21, 38, 59]. In particular, these examples illustrate that this approach naturally coincides with data streaming applications [35, 49, 48, 44]. As such, this Riccati-based approach is amenable to memory-limited hardware architectures, such as FPGAs [22], which opens possibilities for real-time applications requiring embedded implementations.

Some other possible future directions for this work are as follows. Currently, our theoretical connection requires a linear model, Gaussian likelihood, and particular assumptions on the prior. In the Gaussian setting, it would be interesting to explore how incremental or online GPR techniques [10, 43, 19, 29, 9, 4, 33, 37, 45] could be integrated into our Riccati-based framework to approximate the solutions when the Riccati ODEs are difficult to solve directly using numerical ODE solvers, particularly in big data regimes. To consider a wider range of applications, more research is needed to extend this connection to nonlinear models, non-Gaussian likelihoods, and more general priors, including the development of new and efficient numerical algorithms for these settings in high dimensions. Specifically, the link

Table 4

Step size h of RK4 when tuning σ in the hyperparameter tuning example in section 4.1.2. $\sigma_0 \to \sigma_1$ denotes the case where σ is tuned from σ_0 to σ_1.

| σ	$1 \to 0.5$	$1 \to 2$	$2 \to 5$	$5 \to 10$	$10 \to 20$
h	10^{-5}	10^{-5}	10^{-6}	10^{-7}	10^{-7}

between Bayesian inference and viscous HJ PDEs no longer holds when the Hamiltonian is nonquadratic. Moreover, the Riccati-based methodology in section 3 further restricts the prior to be Gaussian. As such, more work is needed to explore how this connection could be leveraged to create efficient numerical methods for more general learning settings. Another natural extension of this work would be to investigate how this connection could be leveraged to reuse existing efficient ML algorithms to solve high-dimensional HJ PDEs. So far, this work only considers the opposite direction but opens opportunities for new ML-based solvers for fields related to HJ PDEs (e.g., stochastic optimal control [54, 14]).

Appendix A. Details of the hyperparameters in the numerical examples. In section 4, we use RK4 to numerically solve the Riccati ODEs, where the step size h of RK4 is chosen to achieve a balance between high accuracy and stable, efficient computations. We use Python, the NumPy library [18], the JAX library [1], and double precision in all numerical examples. An NVIDIA A100 GPU is used to accelerate computations.

In section 4.1.1, we use $h = 5 \times 10^{-5}$. In section 4.1.2, the baseline solution using $\sigma = 1$ is computed using the method of least squares. Then σ is tuned using the values of σ and h shown in Table 4. In Appendix B, the baseline solution using the entire training set is computed using the method of least squares, and we use RK4 with step size $h = 1 \times 10^{-5}$ to delete each outlier. In section 4.2.1, we use $h = 1 \times 10^{-1}$, and in section 4.2.2 we use $h = 1 \times 10^{-2}$. In section 4.3, we use $h = 2 \times 10^{-6}$ for both domain decomposition patterns.

Appendix B. Additional results for example 1. In this section, we present the results of removing outliers from the trained model when solving (4.2) using our Riccati-based methodology from section 3.2.1. Specifically, we first train the model with the entire noisy training set and then remove the effect of identified outliers from the learned model. Figure 12 shows the results of the initial training and the subsequent sequential removal of two outliers. As expected, removing the outliers improves the model prediction accuracy. After both outliers are removed, the exact values now lie within the uncertainty band around the predicted means, which indicates that our models have become more reliable. Recall that the outliers are removed by solving one step of Riccati ODEs backward in time. As such, we do not require retraining on or access to the entire remaining dataset, which provides potential computational savings.

Appendix C. Additional results for example 2. In this section, we present additional results for Case B of example 2. Recall that in section 4.2.2, we consider the scenario where we have access to high quality data with low-level noise but sensors are expensive to deploy. As such, the location of new data measurements must be chosen carefully. Following an active

(a) Inference of u.

(b) Inference/fitting of f.
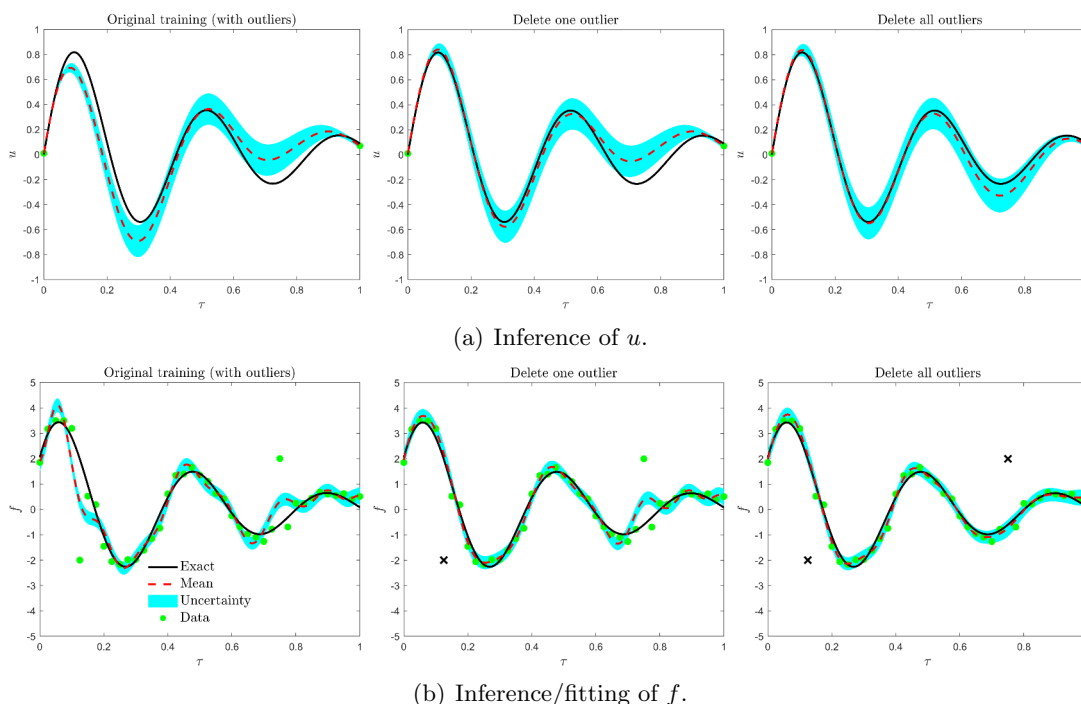
Figure 12. *Deleting two outliers ($\times$) sequentially from the trained model using our Riccati-based approach to solve* (4.2). *The* left *column of* (a) *and* (b) *shows the inferences of u and f, respectively, after the initial training with all data points. The* middle *column shows the inferences after the first outlier is deleted, and the* right *column shows the inferences after two outliers are deleted. The outliers are deleted by solving the associated Riccati ODEs backward in time. This process only uses the results of the previous training step and information of the point to be deleted, which provides potential computational advantages over more standard SciML approaches that would otherwise require retraining on the entire remaining dataset.*

(a) Inferences of f. (b) Inferences of u.
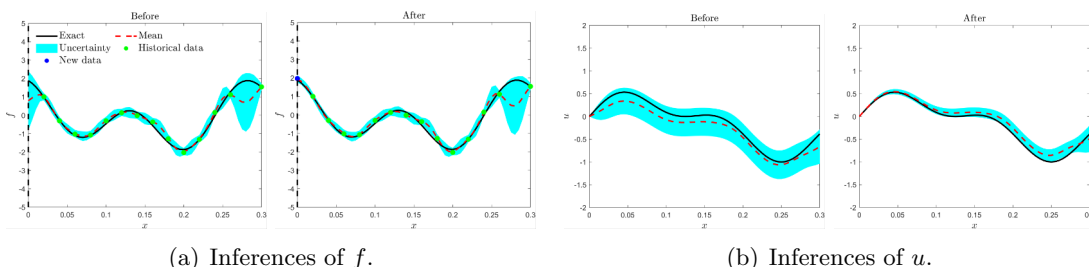
Figure 13. *Intermediate result for adding the 47th data point when using active learning and our Riccati-based approach to solve* (4.5). *The* left *side of* (a) *and* (b) *shows the inferences of f and u, respectively, before the placement of the new sensor and incorporation of the 47th data point. The* right *side of* (a) *and* (b) *shows the updates to f and u after these actions. The vertical black dashed line represents the location ($x = 0$) where the 47th sensor will be placed. Note that we zoom in on the region around the new sensor instead of showing the full domain for clarity of comparison. Our Riccati-based approach naturally complements this active learning framework, as it can efficiently perform repeated updates of the predicted mean and uncertainty.*

learning framework [44], we choose the location of the next sensor to be where the predicted uncertainty is the highest. In Figure 13, we show the intermediate results when adding the 47th sensor. In this case, the predicted uncertainty of u more dramatically reduces. This example

demonstrates how the potential of our Riccati-based approach for real-time inferences allows it to be seamlessly integrated into active learning applications.

REFERENCES

[1] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: Composable Transformations of Python+NumPy Programs*, http://github.com/google/jax, 2018.

[2] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, *Physics-informed neural networks (PINNs) for fluid mechanics: A review*, Acta Mech. Sinica, 37 (2021), pp. 1727–1738.

[3] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks for heat transfer problems*, J. Heat Transf., 143 (2021), 060801.

[4] J. Q. Candela and O. Winther, *Incremental Gaussian processes*, Adv. Neural Inf. Process. Syst., 15 (2002).

[5] P. Chen, T. Meng, Z. Zou, J. Darbon, and G. E. Karniadakis, *Leveraging Hamilton–Jacobi PDEs with time-dependent Hamiltonians for continual scientific machine learning*, in Proceedings of the 6th Annual Learning for Dynamics & Control Conference, PMLR, 2024, pp. 1–12.

[6] P. Chen, T. Meng, Z. Zou, J. Darbon, and G. E. Karniadakis, *Leveraging multitime Hamilton–Jacobi PDEs for certain scientific machine learning problems*, SIAM J. Sci. Comput., 46 (2024), pp. C216–C248, https://doi.org/10.1137/23M1561397.

[7] Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart, *Solving and learning nonlinear PDEs with Gaussian processes*, J. Comput. Phys., 447 (2021), 110668.

[8] Y. Chen, H. Owhadi, and F. Schäfer, *Sparse Cholesky factorization for solving nonlinear PDEs via Gaussian processes*, Math. Comp., (2024).

[9] C.-A. Cheng and B. Boots, *Incremental variational sparse Gaussian process regression*, Adv. Neural Inf. Process. Syst., 29 (2016).

[10] L. Csató and M. Opper, *Sparse on-line Gaussian processes*, Neural Comput., 14 (2002), pp. 641–668.

[11] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, *Scientific machine learning through physics–informed neural networks: Where we are and what's next*, J. Sci. Comput., 92 (2022), 88.

[12] J. Darbon and G. P. Langlois, *On Bayesian posterior mean estimators in imaging sciences and Hamilton–Jacobi partial differential equations*, J. Math. Imaging Vision, 63 (2021), pp. 821–854.

[13] L. C. Evans, *Partial Differential Equations*, 2nd ed., Grad. Stud. Math. 19, American Mathematical Society, Providence, RI, 2010.

[14] G. Fabbri, F. Gozzi, and A. Swiech, *Stochastic Optimal Control in Infinite Dimension*, Probab. Theory Stoch. Model. 82, Springer, Cham, 2017.

[15] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Courier Corporation, Chelmsford, MA, 2003.

[16] R. B. Gramacy and H. K. Lee, *Adaptive design and analysis of supercomputer experiments*, Technometrics, 51 (2009), pp. 130–145.

[17] W. Hackbusch, *Multi-grid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer, Berlin, Heidelberg, 2013.

[18] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.

[19] M. F. Huber, *Recursive Gaussian process: On-line regression and learning*, Pattern Recognit. Lett., 45 (2014), pp. 85–91.

[20] E. Hüllermeier and W. Waegeman, *Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods*, Mach. Learn., 110 (2021), pp. 457–506.

[21] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nature Rev. Phys., 3 (2021), pp. 422–440.

[22] R. KASTNER, J. MATAI, AND S. NEUENDORFFER, *Parallel Programming for FPGAs*, preprint, https://arxiv.org/abs/1805.03648, 2018.

[23] J. KIRKPATRICK, R. PASCANU, N. RABINOWITZ, J. VENESS, G. DESJARDINS, A. A. RUSU, K. MILAN, J. QUAN, T. RAMALHO, A. GRABSKA-BARWINSKA, ET AL., *Overcoming catastrophic forgetting in neural networks*, Proc. Natl. Acad. Sci. USA, 114 (2017), pp. 3521–3526.

[24] K. LINKA, A. SCHÄFER, X. MENG, Z. ZOU, G. E. KARNIADAKIS, AND E. KUHL, *Bayesian physics informed neural networks for real-world nonlinear dynamical systems*, Comput. Methods Appl. Mech. Engrg., 402 (2022), 115346.

[25] D. J. MACKAY, *Information-based objective functions for active data selection*, Neural Comput., 4 (1992), pp. 590–604.

[26] Z. MAO, A. D. JAGTAP, AND G. E. KARNIADAKIS, *Physics-informed neural networks for high-speed flows*, Comput. Methods Appl. Mech. Engrg., 360 (2020), 112789.

[27] S. F. McCORMICK, *Multigrid Methods*, SIAM, Philadelphia, 1987, https://doi.org/10.1137/1.9781611971057.

[28] W. McENEANEY, *Max-plus Methods for Nonlinear Control and Estimation*, Birkhäuser, Boston, 2006.

[29] F. MEIER, P. HENNIG, AND S. SCHAAL, *Incremental local Gaussian regression*, Neural Inf. Process. Syst., 27 (2014).

[30] X. MENG, L. YANG, Z. MAO, J. DEL Á GUILA FERRANDIS, AND G. E. KARNIADAKIS, *Learning functional priors and posteriors from data and physics*, J. Comput. Phys., 457 (2022), 111073.

[31] G. S. MISYRIS, A. VENZKE, AND S. CHATZIVASILEIADIS, *Physics-informed neural networks for power systems*, in 2020 IEEE Power & Energy Society General Meeting (PESGM), 2020, pp. 1–5.

[32] R. M. NEAL, *MCMC using Hamiltonian dynamics*, in Handbook of Markov Chain Monte Carlo, CRC Press, Boca Raton, FL, 2011, pp. 113–162.

[33] D. NGUYEN-TUONG, J. PETERS, AND M. SEEGER, *Local Gaussian process regression for real time online model learning*, Adv. Neural Inf. Process. Syst., 21 (2008).

[34] G. PANG, L. LU, AND G. E. KARNIADAKIS, *fPINNs: Fractional physics-informed neural networks*, SIAM J. Sci. Comput., 41 (2019), pp. A2603–A2626, https://doi.org/10.1137/18M1229845.

[35] G. I. PARISI, R. KEMKER, J. L. PART, C. KANAN, AND S. WERMTER, *Continual lifelong learning with neural networks: A review*, Neural Netw., 113 (2019), pp. 54–71.

[36] E. PICKERING, S. GUTH, G. E. KARNIADAKIS, AND T. P. SAPSIS, *Discovering and forecasting extreme events via active learning in neural operators*, Nature Comput. Sci., 2 (2022), pp. 823–833.

[37] L. POLOK, V. ILA, M. SOLONY, P. SMRZ, AND P. ZEMCIK, *Incremental block Cholesky factorization for nonlinear least squares in robotics*, in Robotics: Science and Systems, 2013, pp. 328–336.

[38] A. F. PSAROS, X. MENG, Z. ZOU, L. GUO, AND G. E. KARNIADAKIS, *Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons*, J. Comput. Phys., 477 (2023), 111902.

[39] J. QUIÑONERO-CANDELA, M. SUGIYAMA, A. SCHWAIGHOFER, AND N. D. LAWRENCE, *Dataset Shift in Machine Learning*, MIT Press, Cambridge, MA, 2008.

[40] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Machine learning of linear differential equations using Gaussian processes*, J. Comput. Phys., 348 (2017), pp. 683–693.

[41] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Numerical Gaussian processes for time-dependent and nonlinear partial differential equations*, SIAM J. Sci. Comput., 40 (2018), pp. A172–A198, https://doi.org/10.1137/17M1120762.

[42] M. RAISSI, P. PERDIKARIS, AND G. 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 (2019), pp. 686–707.

[43] A. RANGANATHAN, M.-H. YANG, AND J. HO, *Online sparse Gaussian process regression and its applications*, IEEE Trans. Image Process., 20 (2010), pp. 391–404.

[44] P. REN, Y. XIAO, X. CHANG, P.-Y. HUANG, Z. LI, B. B. GUPTA, X. CHEN, AND X. WANG, *A survey of deep active learning*, ACM Comput. Surveys (CSUR), 54 (2021), pp. 1–40.

[45] D. RULLIÈRE, N. DURRANDE, F. BACHOC, AND C. CHEVALIER, *Nested kriging predictions for datasets with a large number of observations*, Stat. Comput., 28 (2018), pp. 849–867.

[46] F. SAHLI COSTABAL, Y. YANG, P. PERDIKARIS, D. E. HURTADO, AND E. KUHL, *Physics-informed neural networks for cardiac activation mapping*, Front. Phys., 8 (2020), 42.

[47] F. Schäfer, M. Katzfuss, and H. Owhadi, *Sparse Cholesky factorization by Kullback–Leibler minimization*, SIAM J. Sci. Comput., 43 (2021), pp. A2019–A2046, https://doi.org/10.1137/20M1336254.

[48] B. Settles, *Active Learning Literature Survey*, Technical report, Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, 2009.

[49] G. M. Van de Ven and A. S. Tolias, *Three Scenarios for Continual Learning*, preprint, arXiv:1904.07734, 2019.

[50] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, Vol. 2, MIT Press, Cambridge, MA, 2006.

[51] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*, Princeton University Press, Princeton, NJ, 2010.

[52] L. Yang, X. Meng, and G. E. Karniadakis, *B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data*, J. Comput. Phys., 425 (2021), 109913.

[53] M. Yin, Z. Zou, E. Zhang, C. Cavinato, J. D. Humphrey, and G. E. Karniadakis, *A generative modeling framework for inferring families of biomechanical constitutive laws in data-sparse regimes*, J. Mech. Phys. Solids, 181 (2023), 105424.

[54] J. Yong and X. Y. Zhou, *Stochastic Controls: Hamiltonian Systems and HJB Equations*, Stoch. Model. Appl. Probab. 43, Springer, New York, 2012.

[55] Z. Zhang, Z. Zou, E. Kuhl, and G. E. Karniadakis, *Discovering a reaction–diffusion model for Alzheimer's disease by combining PINNs with symbolic regression*, Comput. Methods Appl. Mech. Engrg., 419 (2024), 116647.

[56] Z. Zou and G. E. Karniadakis, *L-HYDRA: Multi-head Physics-Informed Neural Networks*, preprint, arXiv:2301.02152, 2023.

[57] Z. Zou, X. Meng, and G. E. Karniadakis, *Uncertainty Quantification for Noisy Inputs-Outputs in Physics-Informed Neural Networks and Neural Operators*, preprint, arXiv:2311.11262, 2023.

[58] Z. Zou, X. Meng, and G. E. Karniadakis, *Correcting model misspecification in physics-informed neural networks (PINNs)*, J. Comput. Phys., 505 (2024), 112918, https://www.sciencedirect.com/science/article/pii/S0021999124001670.

[59] Z. Zou, X. Meng, A. F. Psaros, and G. E. Karniadakis, *NeuralUQ: A comprehensive library for uncertainty quantification in neural differential equations and operators*, SIAM Rev., 66 (2024), pp. 161–190, https://doi.org/10.1137/22M1518189.