Scaling up Probabilistic PDE Simulators with Structured Volumetric Information

Tim Weiland — *University of Tübingen, Tübingen AI Center, Germany*
Marvin Pförtner — *University of Tübingen, Tübingen AI Center, Germany*
Philipp Hennig — *University of Tübingen, Tübingen AI Center, Germany*

Abstract

Modeling real-world problems with partial differential equations (PDEs) is a prominent topic in scientific machine learning. Classic solvers for this task continue to play a central role, e.g. to generate training data for deep learning analogues. Any such numerical solution is subject to multiple sources of uncertainty, both from limited computational resources and limited data (including unknown parameters). Gaussian process analogues to classic PDE simulation methods have recently emerged as a framework to construct fully probabilistic estimates of all these types of uncertainty. So far, much of this work focused on theoretical foundations, and as such is not particularly data efficient or scalable. Here we propose a framework combining a discretization scheme based on the well-known Finite Volume Method with complementary numerical linear algebra techniques. Practical experiments, including a spatiotemporal tsunami simulation, demonstrate substantially improved scaling behavior of this approach over previous collocation-based techniques.

Proceedings of the 2nd International Conference on Probabilistic Numerics (ProbNum) 2026, Lappeenranta, Finland. PMLR Volume TBA.

1 Introduction

Partial differential equations (PDEs) are a powerful language for expressing mechanistic knowledge about the world. Important applications include continuum mechanics (Slaughter, 2012), medical imaging (Holder, 2004), weather prediction (Richardson, 2007), and tsunami simulation (Shuto, 1991). In the following, we focus on linear PDEs. Let $\mathbb{D} \subset \mathbb{R}^n$ be open and bounded. A linear PDE is expressed by an equation of the form

$$\mathcal{D}[u] = f, \tag{1}$$

where $f : \mathbb{D} \to \mathbb{R}$ and $\mathcal{D} : U \subset \mathbb{R}^{\mathbb{D}} \to \mathbb{R}^{\mathbb{D}}$ is a linear differential operator. We consider spatiotemporal domains in the form of $\mathbb{D} := [0, T] \times \mathbb{X}$, $T \in \mathbb{R}$, and call $\mathbb{X}$ the spatial domain. Additionally, we write $u(t, \boldsymbol{x})$ for functions on these domains. Spatiotemporal applications impose additional constraints in the form of initial and boundary conditions, restricting the solution at time $t = 0$ and at the spatial domain boundaries, respectively. A system of PDEs together with such conditions is called an initial boundary value problem (IBVP); we seek $u \in U$ satisfying a given IBVP.

In real-world applications, numerous sources of uncertainty affect the solution of an IBVP: initial and boundary conditions are often uncertain, e.g. due to missing data or measurement noise, and discretizations of the PDE induce approximation error. A solver that quantifies these uncertainties enables more informed downstream decisions. Fig. 1 shows an example scenario of a wave caused by an earthquake. If we assume that sensor inaccuracies cause uncertainty about the initial displacement of the water column, this implies several different potential trajectories of a tsunami, which may be modeled by a probability distribution.

In the remainder of this work, we thus view the task of finding a solution u as a machine learning problem in the sense of probabilistic numerics (Hennig et al., 2015). Probabilistic numerical methods frame computation as Bayesian inference. In our case, we impose a Gaussian process (GP) prior on u. Then we can condition on PDE information to obtain a posterior distribution over the solution.

The theory underlying GP-based solvers for both linear and non-linear PDEs has been well-studied (Cockayne et al., 2017; Raissi et al., 2018; Chen et al., 2021; Krämer et al., 2022; Pförtner et al., 2022). Various techniques have been proposed to enable inference for larger scale problems (Chen et al., 2023; Yang and Owhadi, 2023). These formulations are typically stated for general linear functionals, but in practice most of the existing literature instantiates them with point-wise observations of differential operators (which is called **collocation**). Simulating high-frequency physical phenomena using priors based on stationary covariance functions may require small length scales. Our work is motivated by the intuition that highly local point-wise differential observations may pose difficulties in these regimes. Instead, we propose the use of non-local observations obtained by integrating the differential operator over a subdomain, which is equivalent to the widely used Finite Volume Method (FVM).

Our central contribution is a framework for efficient inference on FVM observations. We present a class of GPs in Section 3 that lends itself to particularly efficient application of FVM operators to covariance functions. The key idea is to use box volumes and tensor product covariance functions, reducing multidimensional integrals to efficient one-dimensional integrals. We then explore how to deal with a large number of observations in Section 4, leveraging structured volume layouts to induce Kronecker structure, and combining a Cholesky

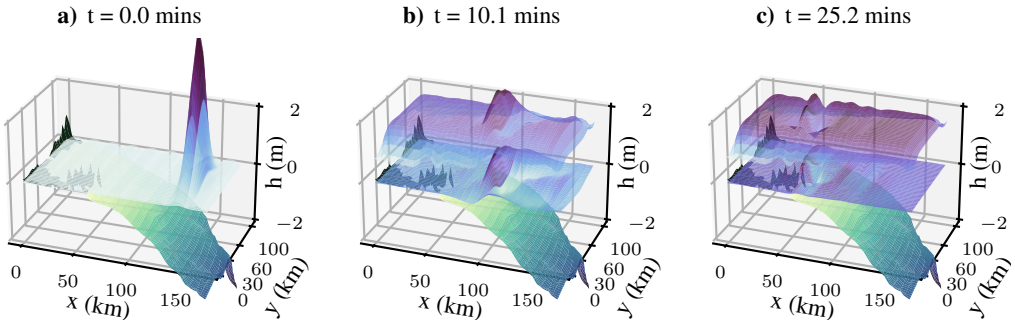

Figure 1: **A physics-informed GP models the propagation of a tsunami near a coast.** For visualization purposes, the seabed topography (the bottom surface) is *not* true to scale. The middle surface (in shades of blue) depicts the posterior mean over the deviation of the water from its mean height. The transparent surface at the top depicts the 95% confidence upper bound, including computational uncertainty from early termination of the iterative solver. An animated version is available at https://github.com/timweiland/gp-fvm.

decomposition with IterGP (Wenger et al., 2022), a probabilistic linear solver that quantifies the computational uncertainty induced by early stopping. We validate the advantage of FVM observations in Section 6.1 by comparing their scaling behavior to collocation observations. Finally, we demonstrate real-world applicability in Section 6.2 by computing a fully probabilistic simulation of a tsunami, and show in Section 6.3 that the same framework extends to PDE-constrained inverse problems.

2 Background

2.1 Affine Gaussian Process Inference

The key idea underlying GP-based PDE solvers is that GPs are closed under affine transformations. Consider transforming both functions in (1) through a linear functional $l : U \to \mathbb{R}$, which yields

$$l[\mathcal{D}[u]] = l[f]. \tag{2}$$

If we model u with a GP, then the posterior $u \mid (l[\mathcal{D}[u]] = l[f])$ is again a GP with closed-form mean and covariance, provided $l \circ \mathcal{D}$ is a bounded linear functional on the sample space of the prior (Section 2.2). In the following, we introduce the necessary notation to formalize this statement. The underlying assumptions and theory are in Appendix A.

Definition 2.1. Let k be the covariance function of a GP, and let $\boldsymbol{\mathcal{L}_1} : U \to \mathbb{R}^{n_1}, \boldsymbol{\mathcal{L}_2} : U \to \mathbb{R}^{n_2}$ be linear operators. Define $\boldsymbol{\mathcal{L}_1} k : \mathbb{D} \to \mathbb{R}^{n_1}$ by

$$\boldsymbol{\mathcal{L}_1} k(\boldsymbol{x}) := \boldsymbol{\mathcal{L}_1}[k(\cdot, \boldsymbol{x})]. \tag{3}$$

Define further $\boldsymbol{\mathcal{L}_1} k \boldsymbol{\mathcal{L}_2'} \in \mathbb{R}^{n_1 \times n_2}$ by

$$(\boldsymbol{\mathcal{L}_1} k \boldsymbol{\mathcal{L}_2'})_{ij} := \boldsymbol{\mathcal{L}_1}[\boldsymbol{x} \mapsto \boldsymbol{\mathcal{L}_2}[k(\boldsymbol{x}, \cdot)]_j]_i. \tag{4}$$

Similar notation may be defined for Banach space-valued linear operators (Pförtner et al., 2022, Notation 2).

Now consider the setting where we know that the transformation of $f \sim \mathcal{GP}(m, k)$ under the linear operator $\boldsymbol{\mathcal{L}} : U \to \mathbb{R}^n$ yields observations $\boldsymbol{y} \in \mathbb{R}^n$ with noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} \neq \boldsymbol{0}$ is permitted for generality, though all our experiments use $\boldsymbol{\mu} = \boldsymbol{0}$. Define $\boldsymbol{G} := \boldsymbol{\mathcal{L}} k \boldsymbol{\mathcal{L}'} + \boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ and denote by $\boldsymbol{G}^\dagger$ its Moore–Penrose pseudoinverse. Then:

$$f \mid \boldsymbol{\mathcal{L}}[f] + \boldsymbol{\varepsilon} = \boldsymbol{y} \sim \mathcal{GP}(m^*, k^*), \tag{5}$$

$$m^*(\boldsymbol{x}) := m(\boldsymbol{x}) + \boldsymbol{\mathcal{L}} k(\boldsymbol{x})^\intercal \boldsymbol{G}^\dagger (\boldsymbol{y} - (\boldsymbol{\mathcal{L}}[m] + \boldsymbol{\mu})), \tag{6}$$

$$k^*(\boldsymbol{x_1}, \boldsymbol{x_2}) := k(\boldsymbol{x_1}, \boldsymbol{x_2}) - \boldsymbol{\mathcal{L}} k(\boldsymbol{x_1})^\intercal \boldsymbol{G}^\dagger \boldsymbol{\mathcal{L}} k(\boldsymbol{x_2}). \tag{7}$$

The operator $\boldsymbol{\mathcal{I}}[u] := \boldsymbol{\mathcal{L}}[u] - \boldsymbol{y}$ is often referred to as an **information operator** (Cockayne et al., 2019b). In the following section, we show that certain PDE discretizations can be formulated through information operators. Thus, Eqs. (6) to (7) enable closed-form inference from affine information.

2.2 PDE Solving as a Learning Problem

Instead of solving Eq. (1) directly, we define n linear **test functionals** $l^{(i)} : U \to \mathbb{R}$ $(i \in \{1, \dots, n\})$ to obtain:

$$l^{(i)}[\mathcal{D}[u]] = l^{(i)}[f] \quad (i \in \{1, \dots, n\}). \tag{8}$$

This reduces an infinite-dimensional equation system to a linear system on $\mathbb{R}^n$. The constraints on u are strictly weaker, but the approximation error decreases as the number of test functionals grows. Methods based on this approach are called **methods of weighted residuals** (MWRs), a class that includes e.g. finite element and Galerkin methods.

Choose test functionals $l^{(i)}$ $(i \in \{1, \dots, n\})$ and a GP prior for the solution of the PDE $u \sim \mathcal{GP}(m, k)$ such that Assumption A.2 is fulfilled and $\left(l^{(i)} \circ \mathcal{D}\right)$ is a bounded linear functional on the reproducing kernel Banach space (RKBS) corresponding to u for all $i \in \{1, \dots, n\}$. We work with RKBSs rather than RKHSs because our priors have sample paths in the Banach spaces $C^{\boldsymbol{\beta}}(\bar{\mathbb{X}})$; in the RKHS case, this boundedness is equivalent to existence of the representer $\left(l^{(i)} \circ \mathcal{D}\right) k(\cdot)$ in the RKHS (Kanagawa et al., 2018, Sec. 4.1). Then according to Theorem A.3,

$$\left(u \mid \left(l^{(i)} \circ \mathcal{D}\right)[u] = l^{(i)}[f]\right) \sim \mathcal{GP}(m^*, k^*), \tag{9}$$

with closed-form expressions for m^* and k^*. We may also combine all n test functionals into one $\mathbb{R}^n$-valued linear operator $\boldsymbol{\mathcal{L}}_{\text{PDE}}$ instead. The sample paths of the resulting posterior fulfill Eq. (8) for all $i \in \{1, \dots, n\}$.

This is a probabilistic numerical solution of the PDE. Pförtner et al. (2022, Sec 3) show that this construction yields a GP with posterior mean matching MWR.

For an IBVP, the initial and boundary conditions may also be expressed through affine GP inference. We write $\boldsymbol{\mathcal{L}}_{\text{IC, BC}}$ for the linear operator that computes all corresponding observations. If we further write $\boldsymbol{\mathcal{L}}_{\text{IBVP}}[u] := (\boldsymbol{\mathcal{L}}_{\text{IC, BC}}[u] \quad \boldsymbol{\mathcal{L}}_{\text{PDE}}[u])^\intercal$, then $\boldsymbol{G} = \boldsymbol{\mathcal{L}}_{\text{IBVP}} k \boldsymbol{\mathcal{L}}'_{\text{IBVP}} + \boldsymbol{\Sigma}$ is the Gram matrix of interest, and GP inference serves as a full probabilistic pipeline to numerically solve an IBVP.

3 Volumetric Information Operators

Most GP-based PDE solvers are based on test functionals $l_{\boldsymbol{x}}[u] = u(\boldsymbol{x})$, inducing the collocation linear operator $\mathcal{L}_{\boldsymbol{x}}[u] = \mathcal{D}[u](\boldsymbol{x})$. In this work, we instead consider test functionals of the form $l_V[u] = \int_V u(\boldsymbol{x}) d\boldsymbol{x}$, where $V \subset \mathbb{D}$ is some chosen volume. Affine inference involves transforming the covariance function under $l_V \circ \mathcal{D}$, which we call the Finite Volume Method (FVM) operator:

$$\mathcal{L}_V[u] := \int_V \mathcal{D}[u](\boldsymbol{x}) d\boldsymbol{x}. \tag{10}$$

Mild assumptions on $u \sim \mathcal{GP}(m, k)$ yield:

$$\left(u \mid \int_V \mathcal{D}[u](\boldsymbol{x}) d\boldsymbol{x} = \int_V f(\boldsymbol{x}) d\boldsymbol{x}\right) \sim \mathcal{GP}(m^*, k^*), \tag{11}$$

with closed-form expressions for m^* and k^* (Eqs. (6) to (7)). The samples of the resulting posterior fulfill the PDE on average across the chosen volume V. The method resulting from this choice of test functional is classically called the subdomain method. As stated in Fletcher (1984), the subdomain method is equivalent to the Finite Volume Method (FVM) due to the divergence theorem. Hence, we use the name **GP-FVM** for GPs conditioned on FVM observations in the remainder of this work. Fig. 2 shows the difference between collocation and GP-FVM on an example problem.

GP-FVM requires transforming the covariance function under $\mathcal{L}_V$. In the following, we explore a framework to enable efficient inference from FVM observations. We start with functions $u : \mathbb{D} \subset \mathbb{R} \to \mathbb{R}$ and build up towards functions $\boldsymbol{u} : \mathbb{D} \subset \mathbb{R}^d \to \mathbb{R}^{d'}$. Appendix B justifies that the constructions of this section satisfy the assumptions underlying Eqs. (5) to (7).

3.1 One-dimensional domains

For $\mathbb{D} \subset \mathbb{R}$, we consider intervals of the form $V := [a, b]$. Thus, the FVM operator has the form

$$\mathcal{L}_{[a,b]}[u] = \int_a^b \mathcal{D}[u](x) dx. \tag{12}$$

A linear differential operator $\mathcal{D}$ is a linear combination of partial derivatives up to order $k \in \mathbb{N}$:

$$\mathcal{D}[u] = \sum_{i=0}^{k} c_i \frac{d^i u}{dx^i}. \tag{13}$$

Hence, Eq. (12) is equivalent to

$$\begin{aligned}\mathcal{L}_{[a,b]}[u] &= \sum_{i=0}^{k} c_i \int_a^b \frac{d^i}{dx^i} u(x) dx \\ &= c_0 \int_a^b u(x) dx + \sum_{i=0}^{k-1} c_{i+1} \left(u^{(i)}(b) - u^{(i)}(a)\right).\end{aligned} \tag{14}$$

The fundamental theorem of calculus simplifies the terms where integrals are mixed with derivatives to differences of derivatives. GP-FVM thus requires efficient access to the integral (for $i = 0$) and arbitrary derivatives $u^{(i)}$ of the covariance function. The widely used Matérn covariance function is isotropic, of the form $k_{\nu,\ell}(x_1, x_2) = k_\nu(|x_1 - x_2|/\ell)$ ($\nu \in \mathbb{R}_+$). For $\nu = q + \frac{1}{2}$ ($q \in \mathbb{N}$), it simplifies to

$$k_{\nu=q+\frac{1}{2}}(r) = \exp\left(-\sqrt{2q+1}r\right) p_q\left(\sqrt{2q+1}r\right), \tag{15}$$

where p_q is a polynomial of order q with known coefficients (Rasmussen and Williams, 2005). By the product rule, derivatives again have the form of an exponential times a polynomial, so evaluations of the Matérn covariance function and arbitrary derivatives thereof are efficient. One can also derive efficient closed-form integrals of the Matérn covariance function; Appendix H gives the derivation.

3.2 Multi-dimensional domains

To transfer the efficiency of the one-dimensional case to a multi-dimensional input domain $\mathbb{D} \subset \mathbb{R}^d$, we reduce multi-dimensional integrals to one-dimensional ones. To achieve this, we restrict ourselves to box volumes of the form $V := [a_1, b_1] \times \cdots \times [a_d, b_d]$. More complex shapes can be approximated by unions of box volumes. Due to Fubini's theorem, Eq. (10) can then be expressed in terms of one-dimensional integrals. Furthermore, we write the order-k linear differential operator $\mathcal{D}$ as a linear combination of partial derivatives $D^{\boldsymbol{\alpha}} := \partial^{|\boldsymbol{\alpha}|} / \partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}$, indexed by multi-indices $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ with $|\boldsymbol{\alpha}| := \sum_i \alpha_i$ and scalar coefficients $c_{\boldsymbol{\alpha}}$. This yields:

$$\begin{aligned}\mathcal{L}_V[u] &= \int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} \mathcal{D}u(\boldsymbol{x}) \, dx_d \dots dx_1 \\ &= \sum_{|\boldsymbol{\alpha}| \le k} c_{\boldsymbol{\alpha}} \int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} D^{\boldsymbol{\alpha}} u(\boldsymbol{x}) \, dx_1 \dots dx_d.\end{aligned} \tag{16}$$

To harness this structure, we use a tensor product of one-dimensional Matérn covariance functions:

$$k(\mathbf{x}_1, \mathbf{x}_2) := \prod_{i=1}^{d} k^{(i)}_{\nu_i, \ell_i}(x_{1i}, x_{2i}). \tag{17}$$

This allows us to distribute the partial derivatives and the one-dimensional integrals to the individual one-dimensional covariance functions. For two different box

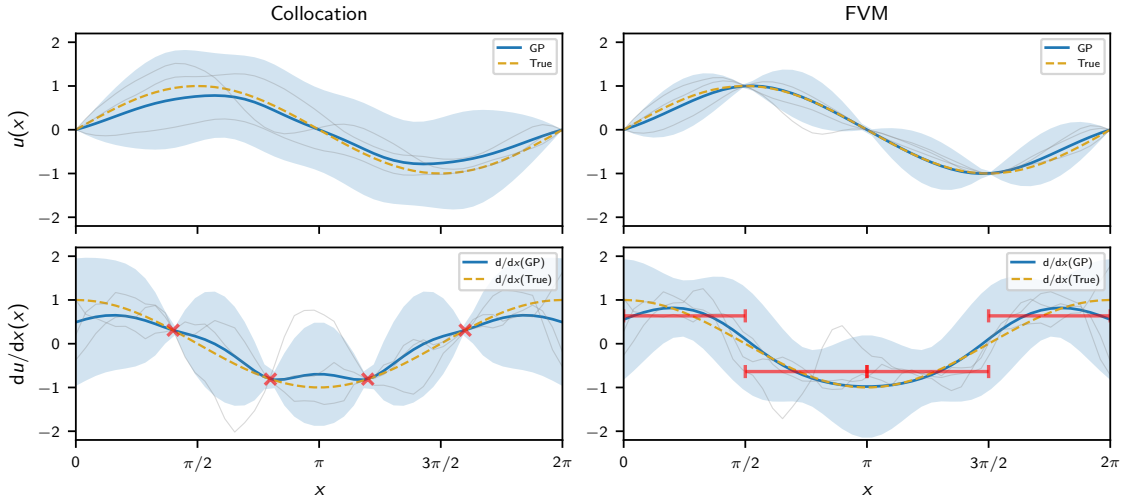

Figure 2: **Learning to solve $^{du}/_{dx}(x) = \cos(x)$ through collocation and FVM.** Both approaches condition a GP prior on i) the boundary conditions $u(0) = u(2\pi) = 0$ and ii) a discretization of the differential equation, via point observations ✕ for collocation and integral observations ⟼ for FVM. Plotted are the resulting posteriors over $u(x)$ and $^{du}/_{dx}(x)$, the ground truths, and three samples per posterior.

volumes $V, V' \subset \mathbb{D}$, we get

$$(\mathcal{L}_V k)(\mathbf{x}_2) = \sum_{|\boldsymbol{\alpha}| \leq k} c_{\boldsymbol{\alpha}} \prod_{i=1}^{d} \mathcal{L}_{[a_i,b_i];\alpha_i}[k^{(i)}_{\nu_i,\ell_i}](x_{2i}), \tag{18}$$

$$\mathcal{L}_V k \mathcal{L}'_{V'} = \sum_{\substack{|\boldsymbol{\alpha}| \leq k \\ |\boldsymbol{\alpha}'| \leq k'}} c_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}'} \prod_{i=1}^{d} \mathcal{L}_{[a_i,b_i];\alpha_i} k^{(i)}_{\nu_i,\ell_i} \mathcal{L}'_{[a'_i,b'_i];\alpha'_i}, \tag{19}$$

with $\mathcal{L}_{[a,b];k}[u] := \int_a^b \frac{d^k}{dx^k} u(x)\, dx$. In this way, the multi-dimensional integrals reduce to linear combinations of products of one-dimensional integrals of the type introduced in Section 3.1. The smoothness properties of the one-dimensional covariance functions are transferred to the tensor product: If $\nu_i = \beta_i + \frac{1}{2}$ ($\beta_i \in \mathbb{N}_0$) and $m \in C^{\boldsymbol{\beta}}(\bar{\mathbb{D}})$ (prior mean), then the sample paths (almost surely) lie in $C^{\boldsymbol{\beta}}(\bar{\mathbb{D}})$ (Wang et al., 2021; Da Costa et al., 2023). Thus, this prior ensures manageable smoothness and enables efficient inference from box volume FVM observations, eliminating the requirement for costly multi-dimensional numerical quadrature.

3.3 Multi-dimensional codomains

So far we assumed a one-dimensional codomain. As a final step, we extend to multi-dimensional codomains to model functions $\boldsymbol{u} : \mathbb{D} \subset \mathbb{R}^d \to \mathbb{R}^{d'}$. A multi-output GP prior for this situation uses a covariance function $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d' \times d'}$. The off-diagonal entries of the output matrix are the cross-covariances between different outputs (e.g. Alvarez et al., 2012). To extend our framework to the multi-output case, we simply set the cross-covariances between the outputs to zero, i.e.

$$\boldsymbol{k}(\boldsymbol{x_1}, \boldsymbol{x_2}) = \operatorname{diag}\left(k_{11}(\boldsymbol{x_1}, \boldsymbol{x_2}) \dots k_{d'd'}(\boldsymbol{x_1}, \boldsymbol{x_2})\right), \tag{20}$$

where the $k_{ii} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ are tensor product covariance functions as in Eq. (17). This prior assumes that the outputs are *a priori* uncorrelated, which is clearly not correct for realistic PDEs with several output variables. The dependencies that matter are nonetheless recovered in the posterior, where the PDE constraint couples the outputs. Alvarez et al. (2012) instead model output correlations in the prior; we do not, both for simplicity and because this keeps the cross-output Kronecker structure intact. The k_{ii} may share hyperparameters or not; Appendix C.4 shares lengthscales across the three tsunami outputs, whereas Section 6.3 uses separate kernels for u and f. A multi-dimensional codomain yields functions $\boldsymbol{u}(\boldsymbol{x}) = (u_1(\boldsymbol{x}) \dots u_{d'}(\boldsymbol{x}))^\intercal$. In this case, linear differential operators take the form

$$\mathcal{D}[\boldsymbol{u}] := \sum_{|\boldsymbol{\alpha}| \leq k} \sum_{i=1}^{d'} c_{\boldsymbol{\alpha},i} D^{\boldsymbol{\alpha}}[u_i]. \tag{21}$$

$\mathcal{D}\boldsymbol{k}(\boldsymbol{x_1}, \boldsymbol{x_2})$ is then (Pförtner et al., 2022, Remark 4.1) a linear combination of vectors $D^{\boldsymbol{\alpha}}\boldsymbol{k}_{\boldsymbol{i},:}(\boldsymbol{x_1}, \boldsymbol{x_2}) \in \mathbb{R}^{d'}$ with:

$$(D^{\boldsymbol{\alpha}}\boldsymbol{k}_{\boldsymbol{i},:}(\boldsymbol{x_1}, \boldsymbol{x_2}))_j := \delta_{ij} \cdot (D^{\alpha} k_{jj})(\boldsymbol{x_1}, \boldsymbol{x_2}). \tag{22}$$

Similarly, $\mathcal{D}\boldsymbol{k}\mathcal{D}'(\boldsymbol{x_1}, \boldsymbol{x_2})$ is a linear combination of scalars $D^{\boldsymbol{\alpha_1}} k_{i_1 i_2} D^{\boldsymbol{\alpha_2}'}(\boldsymbol{x_1}, \boldsymbol{x_2})$ with:

$$D^{\boldsymbol{\alpha_1}} k_{i_1 i_2} D^{\boldsymbol{\alpha_2}'}(\boldsymbol{x_1}, \boldsymbol{x_2}) := \delta_{i_1 i_2} \cdot D^{\boldsymbol{\alpha_1}} k_{i_1 i_1} D^{\boldsymbol{\alpha_2}}(\boldsymbol{x_1}, \boldsymbol{x_2}). \tag{23}$$

Hence, there is no added complexity over the computations from Section 3.2.

4 Large-Scale Inference

Conditioning on FVM observations in the framework described in Section 3 produces a GP posterior with closed-form expressions for the posterior mean and covariance given by Eqs. (6) to (7). The challenge considered in this section is the computation of the quantities involved in these expressions. The computation of the posterior mean (Eq. (6)) requires one linear system solve involving the Gram matrix $\boldsymbol{G} \in \mathbb{R}^{N \times N}$ to obtain the **representer weights**:

$$\boldsymbol{\alpha} := \boldsymbol{G}^{\dagger}(\boldsymbol{y} - (\boldsymbol{\mathcal{L}}[m] + \boldsymbol{\mu})). \tag{24}$$

Evaluating the posterior covariance (Eq. (7)) at N^* test points requires N^* solves with $\boldsymbol{G}$. The classic approach to solve these linear systems is to compute the Cholesky decomposition $\boldsymbol{G} = \boldsymbol{L}\boldsymbol{L}^\intercal$, where $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ is lower triangular with positive diagonal entries. This requires $\frac{1}{3}N^3$ FLOPs to leading order. Afterwards, linear system solves involving $\boldsymbol{G}$ require $\mathcal{O}(N^2)$ operations using forward and backward substitution. This presumes a nonsingular $\boldsymbol{G}$: our formal results use the pseudoinverse $\boldsymbol{G}^\dagger$ to also cover the singular case, but all numerical methods here assume $\boldsymbol{G}$ symmetric positive definite. However, for the use case of PDE solving, naive computation of the Cholesky decomposition quickly becomes infeasible. An example resolution of 100^3 finite volumes for a 3D domain already induces Gram matrices with $N \geq 10^6$. The corresponding Cholesky decomposition requires terabytes of RAM and a leading order of $\frac{1}{3}\cdot 10^{18}$ FLOPs to compute.

4.1 Iterative techniques

Iterative methods such as the conjugate gradients (CG) method address this problem. CG approximates a solution to $\boldsymbol{G}\boldsymbol{x} = \boldsymbol{b}$ by iteratively searching in $\boldsymbol{G}$-orthogonal directions formed from the gradient of $\boldsymbol{x} \mapsto \frac{1}{2}\boldsymbol{x}^\intercal\boldsymbol{G}\boldsymbol{x} - \boldsymbol{x}^\intercal\boldsymbol{b}$, and is matrix-free: it only requires matrix-vector products with $\boldsymbol{G}$. We use an iterative method called IterGP (Wenger et al., 2022). IterGP generalizes CG (among other methods) and produces a combined posterior. The combined posterior adds **computational uncertainty** to the mathematical uncertainty from Eq. (7). As CG obtains a better estimate of the representer weights, the computational uncertainty decreases. Thus, even after $k \ll N$ iterations, IterGP produces a posterior that is exact in the **combined** sense: it does not approximate the exact GP posterior, but quantifies the uncertainty induced by early termination. Section 6.2 reports a case where this combined posterior is underconfident.

Iterative methods are especially effective when fast matrix-vector products with $\boldsymbol{G}$ are available. To achieve this, we use structured discretization schemes.

Definition 4.1. A box FVM discretization scheme is a non-empty set $\mathcal{V}$ of box volumes. We say that $\mathcal{V}$ has factorized structure if

$$\mathcal{V} = \{I_1 \times \cdots \times I_d \mid I_i \in \mathcal{G}_i \text{ for } i = 1, \ldots, d\}, \tag{25}$$

where the one-dimensional partitions $\mathcal{G}_i$ (distinct from the Gram matrix $\boldsymbol{G}$) are, for $i \in \{1, \ldots, d\}$ and $N_i \in \mathbb{N}$:

$$\mathcal{G}_i = \{[a_{i,1}, b_{i,1}], \ldots, [a_{i,N_i}, b_{i,N_i}]\}. \tag{26}$$

Write V_j for the j-th volume in $\mathcal{V}$ and $N_i = |\mathcal{G}_i|$. Define "stacked" linear operators $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{V}}}, \boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{G}_i}}$ through $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{V}}}[u] \in \mathbb{R}^{|\mathcal{V}|}, (\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{V}}}[u])_j = \mathcal{L}_{V_j}[u]$ and $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{G}_i}}[u] \in \mathbb{R}^{N_i}, (\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{G}_i}}[u])_j = \mathcal{L}_{[a_{i,j}, b_{i,j}]}[u]$. Then $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{V}}} k \boldsymbol{\mathcal{L}}'_{\boldsymbol{\mathcal{V}}} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ contains the cross-covariances between the volumes. If $\mathcal{V}$ has factorized structure, Eq. (19) shows that $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{V}}} k \boldsymbol{\mathcal{L}}'_{\boldsymbol{\mathcal{V}}}$ has Kronecker structure:

$$\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{V}}} k \boldsymbol{\mathcal{L}}'_{\boldsymbol{\mathcal{V}}} = \sum_{|\boldsymbol{\alpha}|\leq k, |\boldsymbol{\alpha}'|\leq k} c_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}'} \bigotimes_{i=1}^{d} \boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{G}_i}} k^{(i)}_{\nu_i,\ell_i} \boldsymbol{\mathcal{L}}'_{\boldsymbol{\mathcal{G}_i}}. \tag{27}$$

Matrix-vector products with Kronecker products are cheap and only require access to the individual factors (Van Loan, 2000). Concretely, with $N = \prod_i N_i$, a dense Gram matrix needs $\mathcal{O}(N^2)$ storage and $\mathcal{O}(N^2)$ per matrix-vector product, whereas the factorized form stores $\mathcal{O}(\sum_i N_i^2)$ and applies at $\mathcal{O}(N\sum_i N_i)$, leaving the CG iteration count unchanged. For reasonable values of $|\mathcal{G}_i|$, we can precompute $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{G}_i}} k^{(i)}_{\nu_i,\ell_i} \boldsymbol{\mathcal{L}}'_{\boldsymbol{\mathcal{G}_i}}$. In this way, even discretization schemes with hundreds of thousands of volumes remain tractable.

4.2 Cholesky preconditioners

The solution of an IBVP needs to satisfy initial and boundary conditions in addition to the PDE. Thus, the Gram matrix is constructed from different types of observations. Applying CG directly to such an inhomogeneous matrix may require many iterations to obtain accurate representer weights. We consider situations where a reasonable amount of observations is sufficient to cover the initial and boundary conditions, making a Cholesky decomposition feasible. This is possible even for large-scale problems, exemplified by the experiment in Section 6.2. In these cases, we can separate the initial and boundary observations from the PDE observations. First, compute the Cholesky decomposition for $u_{\text{IC, BC}} := (u \mid \boldsymbol{\mathcal{L}}_{\text{IC, BC}}[u] + \boldsymbol{\varepsilon}_{\text{IC, BC}} = \boldsymbol{y}_{\text{IC, BC}})$. Then, we use $u_{\text{IC, BC}}$ as a new prior for which we use IterGP to integrate the PDE observations. This new prior is a full distribution, not merely a mean; its covariance is a low-rank downdate of k, so matrix-vector products still reduce to the Kronecker factors plus a low-rank correction. In other words, we apply IterGP with CG actions to compute $u_{\text{IBVP}} := (u_{\text{IC, BC}} \mid \boldsymbol{\mathcal{L}}_{\text{PDE}}[u] + \boldsymbol{\varepsilon}_{\text{PDE}} = \boldsymbol{y}_{\text{PDE}})$. Wenger et al. (2022) show that IterGP generalizes the Cholesky decomposition through unit vector actions. Thus, in the sense of IterGP, this strategy constructs a deflation-based preconditioner which projects onto the subspace of the search space that satisfies the initial and boundary observations. All further CG iterations are then isolated to the PDE and cannot violate initial or boundary observations. This combines exact computations *where they are affordable* with approximate computations *where they are necessary*. As a consequence, there is no computational uncertainty for the initial and boundary conditions. This deflation is one instance of the preconditioners and policies available within IterGP; Wenger et al. (2022) discuss further options.

5 Related work

Physics-Informed Machine Learning Popular machine learning based approaches to PDE solving include physics-informed neural networks (Raissi et al., 2019), (Fourier) neural operators (Li et al., 2020; Lu et al., 2021), and neural ODEs (Chen et al., 2018). The latter two rely on internal calls to low-level numerical methods, either to produce training data (neural operators) or to solve ODEs (NODEs), whereas our method is a classic stand-alone numerical PDE solver. It is thus closest to PINNs, but its analytic nature allows for the significant

computational savings outlined above.

Probabilistic Numerics (PN) Our work fits into the framework of PN methods (Hennig et al., 2015), where several variations of GP-based PDE solvers have been considered. Cockayne et al. (2017) propose a solver based on collocation and also consider inverse problems. A generalization of collocation to nonlinear PDEs is given by Chen et al. (2021) through a quadratic optimization problem with nonlinear constraints. Other variants include multistep methods (Raissi et al., 2018) or the PN method of lines (Krämer et al., 2022).

MWRs Pförtner et al. (2022) prove more generally that affine GP inference generalizes all methods of weighted residuals (MWRs), of which FVM is a special case. We leverage this theoretical result to provide a practical framework for efficient inference on this observation class. To the best of our knowledge, our work is the first to focus on practical aspects of FVM observations for GP inference.

Scalability Other approaches towards scalable GP-based PDE solvers include approximations through Fourier features (Mou et al., 2022), sparse Cholesky decompositions (Chen et al., 2023), inducing points (Meng and Yang, 2023), and a stochastic proximal mini-batch approach (Yang and Owhadi, 2023). The use of Kronecker structure has been proposed previously in the context of GPs (Wilson et al., 2015), PDE solvers (Gavrilyuk and Khoromskij, 2019), and GP-based PDE solvers (Krämer et al., 2022; Fang et al., 2023). Our work complements this view, using a new observation class and an efficient iterative solver that quantifies its computational uncertainty.

6 Experiments

Here we investigate the utility and scaling behavior of our GP-FVM framework. First, we run scaling benchmarks comparing GP-FVM to collocation (Section 6.1). Then we test our methods on a real-world tsunami simulation (Section 6.2), demonstrate an inverse problem (Section 6.3), and propose a calibration method for the output scale (Section 6.4).

Implementation An efficient implementation of GP-FVM is available as open-source Python code at `https://github.com/timweiland/gp-fvm`. We build on source code from Pförtner et al. (2022), which in turn leverage the ProbNum package (Wenger et al., 2021). The iterative solvers in particular are designed for execution on a GPU using PyTorch (Paszke et al., 2019). The benchmarks in Section 6.1 were run on one Intel Xeon Gold 6240 CPU. The tsunami simulation in Section 6.2 was run on one NVIDIA A100 GPU. The inverse problem (Section 6.3) and calibration (Section 6.4) experiments were run on an Apple M1 Max, 32 GB RAM. Only the tsunami simulation uses a GPU; all other experiments run on CPU. All experiments use double precision (`float64`) throughout.

6.1 Comparative Benchmarks

Setup We evaluate GP-FVM on three IBVP classes based on manufactured solutions with known analytical ground truth. **a)** 1D Advection ($\partial_t u + \beta \partial_x u = 0$, $\beta = 0.4$) is a spatiotemporal first-order problem with periodic boundary conditions. **b)** 2D Poisson ($\Delta u = f$) is a purely spatial second-order problem with homogeneous Dirichlet conditions. **c)** 2D Wave ($\partial_{tt} u = c^2 \Delta u$, $c = 1$) is second-order in both space and time. Initial/boundary conditions are random superpositions of sine modes, enabling evaluation across many IBVPs. Appendices C.1 to C.3 contain further details.

Evaluation For each IBVP we compute the RMSE or MAE between the GP posterior mean and the analytical solution on a grid of test points, reporting the mean across all IBVPs. For each benchmark we evaluate collocation and GP-FVM at increasing discretization resolution N_{PDE}, uniformly distributed across the domain in both cases. Since f is known analytically, neither method optimizes its observed values; both condition on known data, pointwise for collocation and volume-integrated for GP-FVM. Only lengthscales are tuned, symmetrically for both: 30 Optuna trials (Akiba et al., 2019) minimizing mean MSE on 5 validation IBVPs, per method and resolution.

Results In all cases, GP-FVM outperforms collocation (Fig. 3). The shaded bands show the 25th–75th percentile range across IBVPs. The gap widens with increasing resolution: for **a)** advection, GP-FVM achieves $\sim 10\times$ lower RMSE at 64×64; for **b)** Poisson, the gap exceeds $800\times$ at the same resolution. For all benchmarks, the log-log plot implies different power-law relationships between GP-FVM and collocation. Fig. 2 provides some intuition on why this might be the case: Whereas for collocation the uncertainty collapses in the derivative space, for GP-FVM it collapses directly in the original space. The idea in general is that the integral yields observations in a space that is closer to the solution space than for collocation. This is the regime that motivated our approach: at the small lengthscales needed for finer features, highly local point observations constrain the solution poorly, whereas each volumetric observation aggregates information over a whole cell.

Comparability Collocation can leverage Kronecker structure in the same way as GP-FVM, so for equal N_{PDE} both methods have comparable compute and equal memory requirements.

6.2 Real-world example: Tsunami propagation

Having established the improved scaling behavior of GP-FVM over collocation, we now investigate to what extent it is a feasible choice for less trivial real-world physical processes, using a fluid dynamics problem as an example.

Setup We use the linearized 2D shallow-water equations (Eqs. (40) to (42), Appendix C.4) to model a fictional tsunami propagating towards a coast, in a scenario where

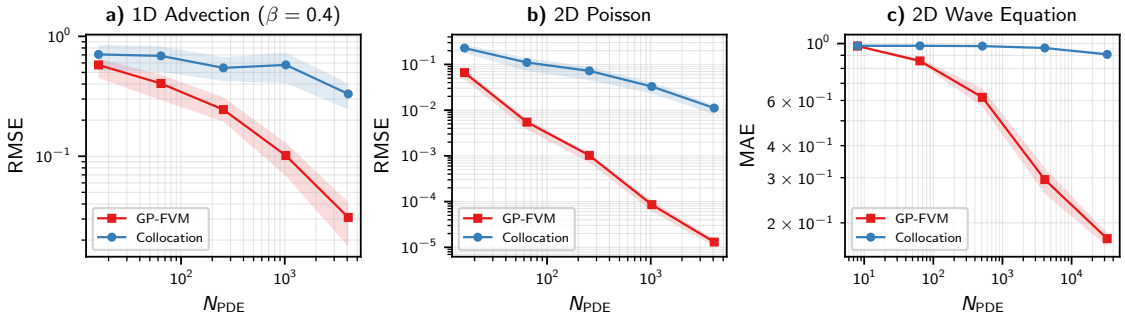

Figure 3: **Scaling behavior of GP-FVM compared to collocation for various problem classes.** The plots show the mean error across all IBVPs in each problem class.

an earthquake causes an anisotropic Gaussian-shaped displacement of water (Fig. 1). An additional challenge is that Eq. (40) has a variable coefficient $H(t, x, y)$ due to the seabed topography. Since Section 3 assumes constant coefficients, we approximate H as piecewise constant on each finite volume. With this and the linearization, which drops the nonlinear mass- and momentum-conservation terms, the example is a proof of concept rather than a full shallow-water simulation. For details on these topics, refer to Appendix C.4.

The results (Fig. 1) show a plausible posterior mean; the arrival time of the synthetic tsunami is similar to that of a real tsunami observed in this coastal area.

Linear system scale The simulation is based on 918000 FVM observations and a Gram matrix with 936672^2 entries; by Section 6.1, collocation would likely have required prohibitively many more observations for this solution quality. We first solve for the initial and boundary observations by Cholesky decomposition (Section 4.2), then run IterGP with CG actions for a budget of 1250 iterations.

Uncertainty The posterior uncertainty (with dedicated plots in Appendix C.4) reflects the computational uncertainty induced by early termination of the solver. It is underconfident relative to the posterior mean quality, a well-known issue with probabilistic linear solvers (Cockayne et al., 2019a): the computational uncertainty cannot match the fast convergence of CG. To alleviate this, we perform 500 additional iterations with actions targeted to uncertainty reduction around the shore (Appendix D); this is a practical budget rather than a tuned hyperparameter. Uncertainty also collapses at $t = 0$ and at the boundaries, whose observations are solved by Cholesky and thus carry no computational uncertainty.

Including the post-iterations, the iterative linear solve requires about 15 minutes on a single GPU, demonstrating the **feasibility of large-scale probabilistic PDE simulations** in our framework. Feasibility is meant relative to Section 4: at this size, a naive Cholesky would require terabytes of RAM and $\sim 10^{18}$ FLOPs. Appendix E visualizes how the CG solver progresses through the time dimension.

6.3 Inverse problem: Source identification

The GP-FVM framework naturally extends to inverse problems. Consider the Helmholtz equation $\nabla^2 u + \kappa^2 u = f$ on $[0, 1]^2$ with homogeneous Dirichlet boundary conditions, where we seek to infer the unknown source f from sparse, noisy observations of the solution u.

Setup We model (u, f) jointly with an independent multi-output GP prior (Eq. (20)), coupling the outputs through the PDE constraint via FVM observations on a 32×32 grid. The ground truth source consists of two localized Gaussian bumps with opposite signs. We observe u at 100 random interior points with additive Gaussian noise ($\sigma_\varepsilon = 0.005$) and condition on $u = 0$ at the boundary.

Results Fig. 4 shows the posterior mean and standard deviation for both u and f. The solution u is reconstructed accurately (maximum error ≈ 0.03 against values of ± 0.7), and the source f, for which only boundary values are imposed, is recovered in the interior with the correct spatial structure: both Gaussian bumps are localized with the right sign and amplitude. The posterior standard deviation is largest for f in regions between the observation points, and smallest for u near the observations, which is qualitatively consistent with well-calibrated uncertainty. We do not verify this quantitatively: the post-hoc argument of Section 6.4 assumes noiseless observations throughout, which does not hold here. This demonstrates that GP-FVM can propagate information bidirectionally through the PDE: from observations of u to inference about f.

6.4 Towards calibration of the output scale

A GP prior $u \sim \mathcal{GP}(0, \sigma^2 \tilde{k})$ has an output scale σ^2 controlling the amplitude of the posterior uncertainty, normally set via marginal likelihood optimization. For GP-FVM the "observations" are a mix of genuine data (initial/boundary conditions) and PDE constraints with known right-hand side, which makes calibration less straightforward: calibrating σ^2 from PDE residuals alone does not yield well-calibrated posteriors in the solution space, because the differential operator compresses uncertainty differently than pointwise evaluation.

Key property If *all* observations are noiseless, the posterior mean is independent of σ^2: it cancels in $m^*(\boldsymbol{x}) = \sigma^2 \tilde{k}_{\boldsymbol{\mathcal{L}}}(\boldsymbol{x})^\intercal (\sigma^2 \boldsymbol{\mathcal{L}} \tilde{k} \boldsymbol{\mathcal{L}'})^{-1} \boldsymbol{y}$. The posterior covariance scales linearly: $k^* = \sigma^2 v$, where v does not depend on σ^2. Thus, σ^2 can be estimated post-hoc as a pure calibration parameter without re-solving. Noiselessness is essential: if observations carry noise that does not scale with σ^2,

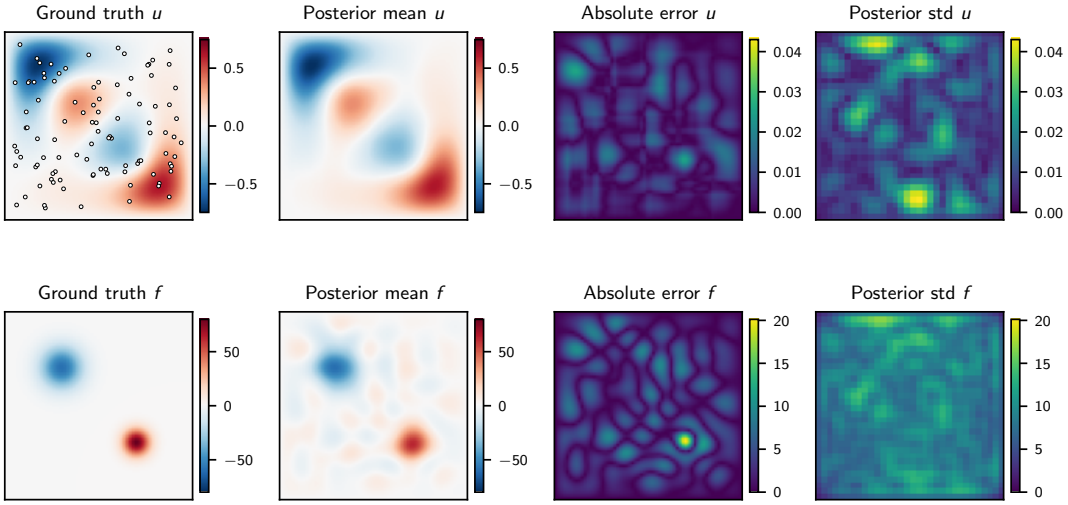

Figure 4: **Helmholtz inverse problem** ($\kappa = 12$). Top row: solution u. Bottom row: inferred source f. From left to right: ground truth, posterior mean, absolute error, and posterior standard deviation. White dots in the top-left panel indicate the 100 noisy observations of u.

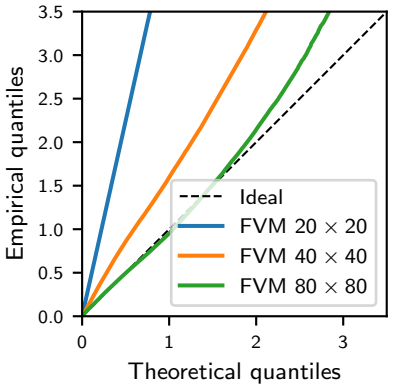

Figure 5: **Calibration quality** for the 1D advection benchmark. QQ plot of pointwise $|z|$-scores against a half-normal reference, pooled over 25 IBVPs. Calibration improves with resolution but remains overconfident at coarser discretizations.

as in Section 6.3, then σ^2 no longer cancels from the mean.

LOO calibration We estimate σ^2 via the closed-form leave-one-out (LOO) predictive distribution of the IC observations within the full Gram matrix $\boldsymbol{G}$:

$$\hat{\sigma}^2 = \frac{1}{N_{\mathrm{IC}}} \sum_{i \in \mathrm{IC}} \frac{\alpha_i^2}{[\boldsymbol{G}^{-1}]_{ii}}, \tag{28}$$

where $\boldsymbol{\alpha} = \boldsymbol{G}^{-1}\boldsymbol{y}$ are the representer weights. Restricting the LOO to IC observations, but using the *full* Gram matrix inverse, separates *data* from *constraints* while still accounting for how the PDE reshapes the covariance; Appendix G gives the details and the cost, which is negligible relative to the solve.

Results Fig. 5 shows a QQ plot of pointwise z-scores for the 1D advection benchmark, where $z(\boldsymbol{x}) = |m^*(\boldsymbol{x}) - u_{\mathrm{true}}(\boldsymbol{x})| / \sqrt{\hat{\sigma}^2\, v(\boldsymbol{x}, \boldsymbol{x})}$. Calibration improves with resolution: at 80×80 the bulk of the distribution ($\lesssim 1.5\sigma$) is well-calibrated, though the tails remain slightly overconfident, while at 20×20 and 40×40 the posterior is more noticeably overconfident. This is expected: a scalar σ^2 rescales the posterior covariance uniformly, but cannot correct errors in its *shape* caused by suboptimal length-scales or insufficient resolution. Principled calibration of probabilistic PDE solvers remains an open problem; we view the LOO approach as a practical step in this direction.

7 Limitations

Like most classic PDE solvers, our approach requires iterative numerical linear algebra (here its probabilistic version, IterGP), which needs careful stabilization; we currently rely on full reorthogonalization, which is effective but costly. IterGP and other probabilistic linear solvers also tend towards overly conservative uncertainty under the purely exploratory CG policy; Appendix D suggests an approach, but further treatment remains beyond the present scope. On a lower level, Kronecker structure requires factorized discretization schemes; non-rectangular domains can be covered, but at a computational overhead. Finally, our framework relies on the linearity of $\mathcal{D}$ for closed-form integral observations; nonlinear PDEs would require linearization, as in Section 6.2, or a different conditioning scheme.

8 Conclusion

We introduced a framework for efficient probabilistic inference on PDE solutions from volumetric (FVM) observations. FVM observations scale favorably: GP-FVM needs several orders of magnitude fewer observations than collocation for the same solution quality, and the matrix-free probabilistic linear algebra we proposed solves the arising system efficiently. This scales probabilistic solutions to nontrivial simulation problems, a key step towards real-world applicability of probabilistic PDE simulators, which remain important also and especially in machine learning: describing simulation as an inference scheme opens avenues, such as deep emulator design, that are hard to reach if simulation is left as a black box.

Acknowledgements

The authors gratefully acknowledge co-funding by the European Union (ERC, ANUBIS, 101123955, Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them); the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); the DFG SPP 2298 (Project HE 7114/5-1), and the Carl Zeiss Foundation, (project "Certification and Foundations of Safe Machine Learning Systems in Healthcare"), as well as funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting TW and MP.

References

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

M. A. Alvarez, L. Rosasco, N. D. Lawrence, et al. Kernels for Vector-Valued Functions: A Review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Solving and learning nonlinear PDEs with Gaussian processes. *Journal of Computational Physics*, 447: 110668, 2021.

Y. Chen, H. Owhadi, and F. Schäfer. Sparse Cholesky factorization for solving nonlinear PDEs via Gaussian processes. *arXiv preprint arXiv:2304.01294*, 2023.

J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. Probabilistic Numerical Methods for PDE-constrained Bayesian Inverse Problems. In *AIP Conference Proceedings*, volume 1853. AIP Publishing, 2017.

J. Cockayne, C. J. Oates, I. C. Ipsen, and M. Girolami. A Bayesian Conjugate Gradient Method (with Discussion). *Bayesian Analysis*, 14(3):937 – 1012, 2019a. doi: 10.1214/19-BA1145.

J. Cockayne, C. J. Oates, T. J. Sullivan, and M. Girolami. Bayesian Probabilistic Numerical Methods. *SIAM review*, 61(4):756–789, 2019b.

N. Da Costa, M. Pförtner, L. Da Costa, and P. Hennig. Sample Path Regularity of Gaussian Processes from the Covariance Kernel. *arXiv preprint arXiv:2312.14886*, 2023.

S. Fang, M. Cooley, D. Long, S. Li, R. Kirby, and S. Zhe. Solving High Frequency and Multi-Scale PDEs with Gaussian Processes. *arXiv preprint arXiv:2311.04465*, 2023.

C. A. J. Fletcher. *Computational Galerkin Methods*. Springer, Berlin, Heidelberg, 1984.

I. Gavrilyuk and B. N. Khoromskij. Tensor Numerical Methods: Actual Theory and Recent Applications. *Computational Methods in Applied Mathematics*, 19 (1):1–4, 2019.

GEBCO Compilation Group (2023). GEBCO 2023 Grid. DOI: 10.5285/f98b053b-0cbc-6c23-e053-6c86abc0af7b.

P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

D. S. Holder, editor. *Electrical Impedance Tomography: Methods, History and Applications*. CRC Press, Dec. 2004.

M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

N. Krämer, N. Bosch, J. Schmidt, and P. Hennig. Probabilistic ODE Solutions in Millions of Dimensions. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11634–11649. PMLR, 17–23 Jul 2022.

N. Krämer, J. Schmidt, and P. Hennig. Probabilistic Numerical Method of Lines for Time-Dependent Partial Differential Equations. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 625–639. PMLR, 28–30 Mar 2022.

P. K. Kundu, I. M. Cohen, and D. R. Dowling. *Fluid Mechanics*. Academic Press, Mar. 2015.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv preprint arXiv:2010.08895*, 2020.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

R. Meng and X. Yang. Sparse Gaussian processes for solving nonlinear PDEs. *Journal of Computational Physics*, 490:112340, 2023.

C. Mou, X. Yang, and C. Zhou. Numerical methods for mean field games based on Gaussian processes and Fourier features. *Journal of Computational Physics*, 460:111188, 2022.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

M. Pförtner, I. Steinwart, P. Hennig, and J. Wenger. Physics-Informed Gaussian Process Regression Generalizes Linear PDE Solvers. *arXiv preprint arXiv:2212.12474*, 2022.

G. Pleiss, J. Gardner, K. Weinberger, and A. G. Wilson. Constant-Time Predictive Distributions for Gaussian Processes. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4114–4123. PMLR, 10–15 Jul 2018.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, 2018. doi: 10.1137/17M1120762.

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. *Journal of Computational physics*, 378:686–707, 2019.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005.

L. F. Richardson. *Weather Prediction by Numerical Process*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 2 edition, 2007.

N. Shuto. Numerical simulation of tsunamis — Its present and near future. *Natural Hazards*, 4(2):171–191, June 1991.

W. S. Slaughter. *The Linearized Theory of Elasticity*. Springer Science & Business Media, Dec. 2012.

L. N. Trefethen. Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations. Unpublished text, 1996. URL http://people.maths.ox.ac.uk/trefethen/pdetext.html.

G. K. Vallis. *Atmospheric and Oceanic Fluid Dynamics*. Cambridge University Press, 2017.

C. F. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, 2000.

J. Wang, J. Cockayne, O. Chkrebtii, T. J. Sullivan, and C. J. Oates. Bayesian Numerical Methods for Nonlinear Partial Differential Equations. *Statistics and Computing*, 31(5):55, Sept. 2021.

J. Wenger, N. Krämer, M. Pförtner, J. Schmidt, N. Bosch, N. Effenberger, J. Zenn, A. Gessner, T. Karvonen, F.-X. Briol, et al. ProbNum: Probabilistic Numerics in Python. *arXiv preprint arXiv:2112.02100*, 2021.

J. Wenger, G. Pleiss, M. Pförtner, P. Hennig, and J. P. Cunningham. Posterior and Computational Uncertainty in Gaussian Processes. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:10876–10890, 2022.

A. G. Wilson, C. Dann, and H. Nickisch. Thoughts on Massively Scalable Gaussian Processes. *arXiv preprint arXiv:1511.01870*, 2015.

X. Yang and H. Owhadi. A Mini-Batch Method for Solving Nonlinear PDEs with Gaussian Processes. *arXiv preprint arXiv:2306.00307*, 2023.

A Affine Gaussian Process Inference, in Detail

In the following, we elaborate on the assumptions required for the results in Section 2.1 to hold. The following theory is based on Section 4 of Pförtner et al. (2022).

First, some assumptions on the prior are necessary. This requires the following definition:

Definition A.1. A **reproducing kernel Banach space** (RKBS) is a Banach space of real-valued functions on a non-empty set $\mathbb{X}$ on which all point evaluation functionals are continuous.

Assumption A.2. $f \sim \mathcal{GP}(m, k)$ is a Gaussian process with index set $\mathbb{X}$ on $(\Omega, \mathcal{F}, P)$, mean function $m : \mathbb{X} \to \mathbb{R}$ and covariance function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. The sample paths of f lie in a separable RKBS $\mathbb{B}$ and $\omega \mapsto f(\cdot, \omega)$ is a $\mathbb{B}$-valued Gaussian random variable.

Under Assumption A.2, we can condition a GP on observations obtained through linear operators. The following theorem uses the notation from Eqs. (3) to (4):

Theorem A.3. *Let $\boldsymbol{\mathcal{L}} : \mathbb{B} \to \mathbb{R}^n$ be a bounded linear operator. Under Assumption A.2, the following results hold.*
Viewing f as a $\mathbb{B}$-valued Gaussian random variable, the application of $\mathcal{L}$ yields

$$\boldsymbol{\mathcal{L}}[f] \sim \mathcal{N}(\boldsymbol{\mathcal{L}}[m], \boldsymbol{\mathcal{L}} k \boldsymbol{\mathcal{L}}'). \tag{29}$$

Let $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be an $\mathbb{R}^n$-valued Gaussian random variable such that $\boldsymbol{\varepsilon} \perp f$, and let $\boldsymbol{y} \in \mathbb{R}^n$. Define $\boldsymbol{G} := \boldsymbol{\mathcal{L}} k \boldsymbol{\mathcal{L}}' + \boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$. Then:

$$f \mid \boldsymbol{\mathcal{L}}[f] + \boldsymbol{\varepsilon} = \boldsymbol{y} \sim \mathcal{GP}(m^*, k^*), \tag{30}$$

with

$$m^*(\boldsymbol{x}) := m(\boldsymbol{x}) + \boldsymbol{\mathcal{L}} k(\boldsymbol{x})^\intercal \boldsymbol{G}^\dagger (\boldsymbol{y} - (\boldsymbol{\mathcal{L}}[m] + \boldsymbol{\mu})), \tag{31}$$

$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) := k(\boldsymbol{x_1}, \boldsymbol{x_2}) - \boldsymbol{\mathcal{L}} k(\boldsymbol{x_1})^\intercal \boldsymbol{G}^\dagger \boldsymbol{\mathcal{L}} k(\boldsymbol{x_2}). \tag{32}$$

Theorem A.3 gives closed-form expressions for the posterior obtained by conditioning a GP on linear operator observations of its paths. Pförtner et al. (2022) also state the equivalent result for when the codomain of $\mathcal{L}$ is another real separable RKBS. Classic GP regression on point evaluations then corresponds to the special case $\mathcal{L} = \mathrm{id}_{\mathbb{B}}$.

B Theoretical justification

For affine GP inference based on these ideas to be theoretically justified, the introduced priors must satisfy Assumption A.2 and the linear operators must be bounded on their sample spaces.

Pförtner et al. (2022) show that if f is a GP with paths$(f) \subset C^{\boldsymbol{\beta}}(\bar{\mathbb{X}})$ P-almost surely ($\boldsymbol{\beta} \in \mathbb{N}_0^d$), then f fulfills Assumption A.2. As mentioned in Section 3.2, this is the case for the tensor product covariance function.

For the multi-dimensional codomain case, our choice of covariance function amounts to modelling d' independent single-output GPs. Each of these single-output GPs has a sample space in $C^{\boldsymbol{\beta}}(\bar{\mathbb{X}})$ for some $\boldsymbol{\beta} \in \mathbb{N}_0^d$. As a consequence, the overall sample space $\mathbb{B}$ is isometrically isomorphic to a subset of the Cartesian product $\times_{i=1}^{d'} C^{\beta_i}(\bar{\mathbb{X}})$, which is again a separable RKBS under an appropriate norm. Thus, f is a Gaussian random variable whose values P-almost surely lie in $\mathbb{B}$. Hence, Assumption A.2 is fulfilled. For more details on the multi-dimensional codomain case, refer to Pförtner et al. (2022, Remark 4.2).

We also need to prove that the linear operator we described is bounded on the sample space. The linear operator is the composition of an integral over a finite box volume with a linear differential operator. Because the sample space consists of functions in $C^{\boldsymbol{\beta}}(\bar{\mathbb{X}})$, we know that all partial derivatives appearing in the PDE are bounded and uniformly continuous (assuming $\boldsymbol{\beta}$ is sufficiently large). It immediately follows that $\mathcal{D}$ is bounded as a sum of bounded linear operators.

Furthermore, it is trivial to show that the integral of bounded and uniformly continuous functions over a finite box volume is bounded:

$$\begin{aligned}
\left| \int_V f(\mathbf{x}) d\mathbf{x} \right| &\leq \int_V |f(x)| d\mathbf{x} && (33)\\
&\leq \int_V \|f\|_{C^0(\bar{\mathbf{X}})} d\mathbf{x} && (34)\\
&= \int_{a_1}^{b_1} \cdots \int_{a_d}^{b_d} \|f\|_{C^0(\bar{\mathbf{X}})} dx_d \ldots dx_1 && (35)\\
&= \|f\|_{C^0(\bar{\mathbf{X}})} \prod_{i=1}^{d} (b_i - a_i) && (36)\\
&\leq \|f\|_{C^{\boldsymbol{\beta}}(\bar{\mathbf{X}})} \prod_{i=1}^{d} (b_i - a_i). && (37)
\end{aligned}$$

Clearly, the integral is also linear. Hence, the operator that generates our observations is a bounded linear operator as a composition of bounded linear operators. This proves that our framework is valid for both the single-input and the multi-input cases.

C Experiment details

All benchmarks use manufactured solutions with known analytical ground truth, enabling exact error evaluation without reliance on external datasets. For each problem class, initial/boundary conditions are random superpositions of sine modes, and we evaluate across 50 random IBVPs. The shaded bands in Fig. 3 show the 25th–75th percentile range across IBVPs.

C.1 1D Advection

PDE $\partial_t u + \beta \partial_x u = 0$ on $[0, 2] \times [0, 1]$ with $\beta = 0.4$ and periodic boundary conditions. The initial condition is $u_0(x) = \sum_{j=1}^{5} a_j \sin(2\pi j x)$ with $a_j \sim \mathcal{N}(0, 1/j^2)$. The exact solution is $u(t, x) = u_0((x - \beta t) \bmod 1)$.

Observation Layout For the initial condition, we use ∼100 uniformly spaced observations. For the periodic boundary, we use 50 observations uniformly spaced across the time dimension, expressed as $\mathcal{L}_{\text{periodic},t}[u] = u(t,0) - u(t,1) = 0$. PDE observations are uniformly spread across the domain.

Prior Zero mean, tensor product of $^3/_2$-Matérn covariance functions.

Hyperparameter Optimization Lengthscales are optimized per method and resolution via Bayesian optimization (Akiba et al., 2019) with 30 trials, minimizing mean MSE on 5 random IBVPs.

C.2 2D Poisson

PDE $\Delta u = f$ on $[0,1]^2$ with homogeneous Dirichlet boundary conditions. The manufactured solution is $u(x,y) = \sum_{i,j=1}^{5} a_{ij} \sin(i\pi x)\sin(j\pi y)$ with $a_{ij} \sim \mathcal{N}(0, 1/(i+j))$, which automatically satisfies the boundary conditions. The right-hand side $f = \Delta u$ and its FVM integrals are computed analytically.

Observation Layout 30 boundary observations per edge. PDE observations uniformly distributed.

Prior Zero mean, tensor product of $^5/_2$-Matérn covariance functions with equal lengthscale in x and y.

Hyperparameter Optimization Same as advection: 30 Optuna trials on 5 random IBVPs.

C.3 2D Wave Equation

PDE $\partial_{tt} u = c^2 \Delta u$ on $[0,2] \times [0,1]^2$ with $c = 1$ and homogeneous Dirichlet boundary conditions. The initial condition is $u(0,x,y) = \sum_{i,j=1}^{2} C_{ij} \sin(i\pi x)\sin(j\pi y)$ with zero initial velocity, where the coefficients $\boldsymbol{C}$ are sampled from a Gaussian (see Eq. (39)). The exact solution is obtained by separation of variables:

$$u(t,x,y) = \sum_{i,j} C_{ij} \cos(\omega_{ij} t)\sin(i\pi x)\sin(j\pi y), \quad (38)$$
$$\omega_{ij} = c\pi\sqrt{i^2 + j^2}.$$

$$\mathbf{C} = \begin{pmatrix} \tilde{c}_1 & \tilde{c}_2 \\ \tilde{c}_3 & \tilde{c}_4 \end{pmatrix}, \quad (39)$$
$$\tilde{\mathbf{c}} \sim \mathcal{N}\left(\begin{pmatrix} 1 \\ 0.5 \\ 0.5 \\ 0 \end{pmatrix}, \operatorname{diag} \begin{pmatrix} 0.1^2 \\ 0.2^2 \\ 0.2^2 \\ 0.3^2 \end{pmatrix} \right).$$

Observation Layout 8×8 initial condition observations, 20 temporal $\times$ 10 spatial boundary observations per edge. PDE observations uniformly distributed.

Prior Zero mean, tensor product of $^5/_2$-Matérn covariance functions with shared spatial lengthscale.

Hyperparameter Optimization Grid search over 10 temporal and 10 spatial lengthscales, selecting the combination with lowest error per IBVP.

C.4 Tsunami propagation

PDEs We model the propagation of a tsunami using the linearized 2D shallow-water equations (Vallis, 2017):

$$\frac{\partial}{\partial t} h + H\left(\frac{\partial}{\partial x} u + \frac{\partial}{\partial y} v\right) = 0, \quad (40)$$
$$\frac{\partial}{\partial t} u - fv + g\frac{\partial}{\partial x} h + ku = 0, \quad (41)$$
$$\frac{\partial}{\partial t} v + fu + g\frac{\partial}{\partial y} h + kv = 0. \quad (42)$$

The vertical deviation of a body of water from its mean height $H(t,x,y)$ is modelled by $h(t,x,y)$. The water velocities in the x- and y-directions are modeled by $u(t,x,y)$ and $v(t,x,y)$, respectively. The scalar parameters are the gravitational acceleration g, the Coriolis coefficient f and the viscous drag coefficient k.

Boundary condition At the boundary, we use radiation boundary conditions with wave speed $c(x,y) = \sqrt{gH(x,y)}$. For example, at $x = 0$, we use the condition

$$\frac{\partial h}{\partial t} = c(0,y)\frac{\partial h}{\partial x}, \quad (43)$$

and similarly for the other boundaries. This condition effectively allows outward flow and rejects inward flow.

Bathymetry For the seabed topography (also known as bathymetry), we use data from the GEBCO 2023 grid (GEBCO Compilation Group (2023)) of a rectangular crop of Japan's coastal area. The data is included in our code repository.

Finite volume layout We use finite volumes of equal volume that disjointly span the entire domain. Each volume has approximately 1km length along the x axis, 5km length along the y axis and 20 seconds span across the temporal dimension. This results in $90 \cdot 170 \cdot 20 = 306000$ volumes, and each volume yields three observations for the three PDEs. We chose this imbalance between the x and y axes because the initial condition exhibits higher frequencies along the x axis.

IBVP scale We consider a simulation length of 30 minutes. For the initial condition, we use uniformly spaced observations with a spacing of 3km per spatial dimension. We also use uniformly spaced observations for the boundary conditions with a spatial spacing of 5km and a temporal spacing of 30 seconds.

Prior We use an independent multi-output covariance function consisting of three tensor products of $^5/_2$-Matérn covariance functions with lengthscales 60, 6 and 10 for the t, x and y dimensions.

Variable bathymetry Eq. (40) uses a variable coefficient $H(t,x,y)$. It is constant with respect to time, but not with respect to space due to the variable bathymetry. To handle this challenge, we assume that the bathymetry is piecewise constant within each subdomain used to discretize the PDE. This amounts to forming Hadamard products of the Kronecker products in Eq. (27) with rank-one matrices formed by the seabed topography. At the resolution used for the subdomain observations, the piecewise constant approximation is nearly exact for the x dimension and not too inaccurate for the y dimension.

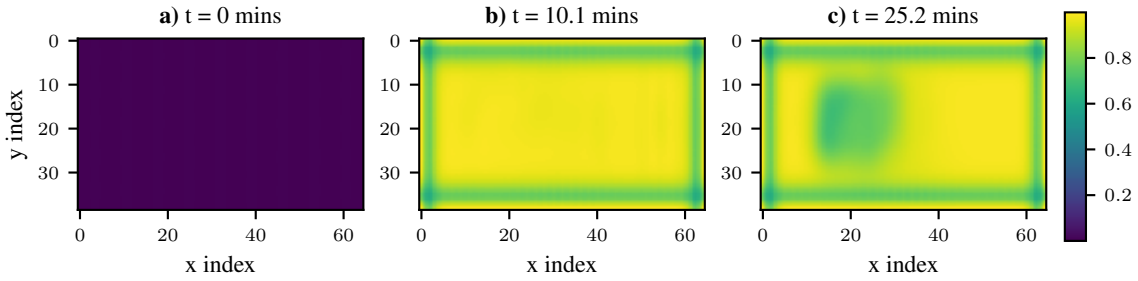

Figure 6: **Posterior covariance of the tsunami simulation.** The heatmaps depict the marginal posterior covariance at the same discrete locations used to create Fig. 1.

Uncertainty Fig. 6 shows the posterior marginal uncertainty from Fig. 1. The uncertainty reduction in a rectangular area between Fig. 1b) and Fig. 1c) is caused by post-iterations using the policy described in Appendix D. This policy is based on $10 \cdot 5 \cdot 5$ uniformly spaced points in the subdomain $[1200, 1800] \times [40, 75] \times [33.3, 66.6]$.

Compute The experiment was run on one NVIDIA A100 GPU on an internal cluster. One full run of the experiment requires about 1 hour of runtime, which includes data loading and writing as well as a high-resolution evaluation of the posterior. The actual iterative linear solve requires about 15 minutes. We estimate that we spent about 15 runs in total to fix bugs and tweak parameters during our research until we converged at the run presented in the paper.

D Targeted uncertainty reduction

As described in Section 6.2, the posterior uncertainty visualized in Fig. 1 and Fig. 6 is underconfident. In this section, we derive a simple technique to obtain better uncertainty quantification locally in targeted areas of the domain.

Consider Eq. (7). Computing the marginal posterior covariance exactly at N^* points given by $\boldsymbol{X^*} \in \mathbb{R}^{N^* \times d}$ would require N^* linear system solves, encapsulated by $\boldsymbol{G}^\dagger \underbrace{\boldsymbol{\mathcal{L}} k(\boldsymbol{X^*})}_{\in \mathbb{R}^{N \times N^*}}$.

Instead, we focus on a subsample $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\tilde{N} \times d}$ with $\tilde{N} \ll N^*$. In the tsunami example, we chose uniformly spaced points towards the end of the simulation, close to the shore. Even for a subsample we cannot afford $\tilde{N}$ accurate linear system solves. Instead, we leverage similarities between related linear systems by sampling linear combinations of the columns of $\boldsymbol{\mathcal{L}} k(\tilde{\boldsymbol{X}}) \in \mathbb{R}^{N \times \tilde{N}}$.

Concretely, we sample a weight vector $\boldsymbol{\beta} \sim \mathrm{Dir}(1, \ldots, 1)$ from a Dirichlet distribution and consider the linear combination $\boldsymbol{\mathcal{L}} k(\tilde{\boldsymbol{X}})\boldsymbol{\beta} \in \mathbb{R}^N$. IterGP iteratively builds up an inverse approximation, which we denote by $\boldsymbol{C_i}$ at iteration i. Then the residual with respect to the inverse approximation[1] is given by

$$\boldsymbol{r}_i := \boldsymbol{\mathcal{L}} k(\tilde{\boldsymbol{x}})\boldsymbol{\beta} - \boldsymbol{G}\boldsymbol{C_i}\boldsymbol{\mathcal{L}} k(\tilde{\boldsymbol{x}})\boldsymbol{\beta}. \tag{44}$$

Using these residual vectors as IterGP actions implements CG with respect to the right-hand side $\boldsymbol{\mathcal{L}} k(\tilde{\boldsymbol{X}})\boldsymbol{\beta}$.

[1] Refer to Wenger et al. (2022) for details.

Thus, we iteratively solve $\boldsymbol{G}^\dagger \boldsymbol{\mathcal{L}} k(\tilde{\boldsymbol{X}})\boldsymbol{\beta}$. If we sample a different $\boldsymbol{\beta}$ in each iteration, we effectively spread the solve across the entire area covered by $\tilde{\boldsymbol{X}}$.

The idea of using a Lanczos process on the right-hand side columns in Eq. (7) to build up a low-rank approximation effective at uncertainty quantification has been considered previously by Pleiss et al. (2018), which directly inspired our approach described here. We start the Lanczos process indirectly through CG, and the low-rank approximation is automatically constructed by IterGP's inverse approximation.

E CG action visualization

Fig. 7 depicts the behavior of CG for the tsunami problem. CG weighs the subdomain observations by their current residual. Initially, the residual is the largest for the volumes near $t = 0$. Then, the solver gradually progresses through time. The repeated diagonal lines are effectively caused by early observations requiring multiple iterations to sufficiently reduce the residual. Meanwhile the progress from the previous iterations already propagates through time.

F Conservation Laws

PDEs are often derived from conservation laws in integral form. As an example, consider a fluid with density $\rho(t, \boldsymbol{x})$ and velocity $u(t, \boldsymbol{x})$. Then conservation of mass with respect to a fixed control volume V with surface A is expressed by the equality (Kundu et al., 2015):

$$\frac{d}{dt} \int_V \rho(t, \boldsymbol{x}) dV = - \int_A \rho(t, \boldsymbol{x}) u(t, \boldsymbol{x}) \boldsymbol{n} dA. \tag{45}$$

Intuitively, this means that the amount of mass contained inside a static volume may only change through inflow or outflow through its boundaries.

In the finite volume method, the domain is divided into a fixed number of control volumes. Then, conservation laws are solved by balancing fluxes between neighboring volumes. As a result, the finite volume method is inherently conservative.

Similarly, our method also divides the domain into volumes. The difference is that our method is based on fulfillment of some PDE, on average, across the volume. This PDE may be the differential form of a conservation

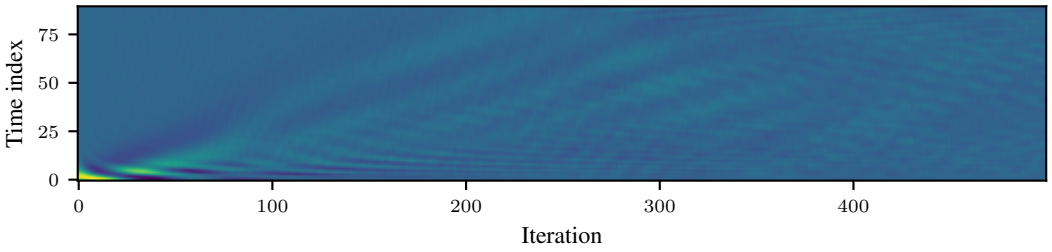

Figure 7: **IterGP actions show how the solver progresses through the time dimension.** Depicted are the first 500 actions of IterGP with a CG policy for the tsunami example. Each action is reduced to the FVM components, and then summed up over the spatial dimensions. The result is a vector containing the weight by which each time slice is targeted.

law. But there is no explicit balancing of fluxes, and thus, our method is not inherently conservative.

Nevertheless, there are situations where our method can directly express conservation laws. Assume we model $\rho(t, \boldsymbol{x})$ by a GP as described in Sections 3.1 to 3.2. Then the left-hand side of Eq. (45) can be computed in closed-form. Due to our use of box volumes, the surface integral in Eq. (45) is a one-dimensional integral. For fluids with constant velocity at the surface, the right-hand side is thus also tractable. A special case is a volume with closed boundaries that allow no outward flow, i.e. $u(t, \boldsymbol{x}) = 0$ at the boundaries. In these situations, the fulfillment of a conservation law at discrete points in time may be expressed by subdomain observations.

In the spirit of probabilistic numerics, a conservation law can thus be considered as yet another type of information about the function we want to infer. It may be used, for example, to construct priors with samples that are all (approximately) globally conservative of some quantity. This prior may then be conditioned on PDE information to obtain a simulation in the posterior that respects the conservation law.

Even if the PDE implicitly already contains the corresponding conservation law this may be useful. If the resolution of the PDE observations is too coarse, numerical dissipation (Trefethen, 1996) may deteriorate the quality of the solution. Explicitly "injecting" the conservation law into the prior may counteract numerical dissipation.

Overlapping volumes. Nothing in the inference formalism requires the volumes to be disjoint: the test functionals $\mathcal{L}_V$ of Section 3 are well defined for overlapping V, and the Kronecker structure of Section 4 is preserved as long as the layout remains a Cartesian product of one-dimensional partitions, which permits overlapping intervals within each axis. What is lost is the identification with the Finite Volume Method proper. That identification runs through the divergence theorem applied to disjoint control volumes with shared faces (Fletcher, 1984), so the conservation-law reading above no longer applies to an overlapping layout, even though the GP inference itself remains valid.

G LOO output-scale calibration

The estimator in Eq. (28) rests on a separation between two kinds of "observation" in the Gram matrix. The initial-condition observations are genuine *data*: they carry information about the amplitude of u, and are therefore informative about σ^2. The PDE and boundary observations are *constraints*: their right-hand sides are known, and they shape the posterior without saying anything about its amplitude. Calibrating σ^2 from PDE residuals alone therefore fails, since the differential operator compresses uncertainty differently than pointwise evaluation.

Restricting the leave-one-out sum to IC observations isolates the informative subset, while evaluating it against the *full* Gram matrix inverse retains the effect of the PDE and boundary constraints on the covariance structure. This is what distinguishes Eq. (28) from a LOO estimate computed on the IC block alone, which ignores how the constraints reshape the posterior.

The additional cost is small. For IterGP, the diagonal of $\boldsymbol{G}^{-1}$ is available from the low-rank inverse approximation the solver already maintains, at $\mathcal{O}(N_{\mathrm{IC}}k)$ for k iterations. For the Cholesky solver, it requires N_{IC} back-substitutions at $\mathcal{O}(N_{\mathrm{IC}}N^2)$. In both cases this is negligible relative to the solve itself, and no re-solve is needed: by the scaling property of Section 6.4, σ^2 enters only as a multiplicative factor on the posterior covariance.

H Closed-form Matérn integrals

For $\nu = q + \frac{1}{2}$ ($q \in \mathbb{N}$), the one-dimensional Matérn covariance function is

$$k_\nu(r) = \exp\left(-\sqrt{2q+1}\, r\right) p_q\left(\sqrt{2q+1}\, r\right), \qquad (46)$$

with p_q a polynomial of degree q (Rasmussen and Williams, 2005). The integrals required by Eq. (14) are therefore of the form $\int e^{-\lambda r} r^j \, dr$ with $\lambda = \sqrt{2q+1}/\ell$ and $j \le q$. Repeated integration by parts gives the antiderivative in closed form,

$$\int e^{-\lambda r} r^j \, dr = -e^{-\lambda r} \sum_{m=0}^{j} \frac{j!}{(j-m)!} \frac{r^{j-m}}{\lambda^{m+1}}, \qquad (47)$$

so every term is again an exponential times a polynomial. Definite integrals over $[a, b]$ follow by evaluation at the

endpoints, taking care to split the domain at $r = 0$, where $|x_1 - x_2|$ is not differentiable. The same argument applies to the mixed integral-derivative terms, since differentiating an exponential-times-polynomial again yields an exponential times a polynomial. Consequently all entries of $\boldsymbol{\mathcal{L}}_{\boldsymbol{\mathcal{G}}_i} k^{(i)}_{\nu_i,\ell_i} \boldsymbol{\mathcal{L}}'_{\boldsymbol{\mathcal{G}}_i}$ are available in closed form at $\mathcal{O}(1)$ cost per entry, with no numerical quadrature.