Version of Record: https://www.sciencedirect.com/science/article/pii/S0021999117305582
Manuscript_47f0faafc9a5fe057dff816139291987

Machine Learning of Linear Differential Equations using Gaussian Processes

Maziar Raissi[a], Paris Perdikaris[b], George Em Karniadakis[a]

[a] *Division of Applied Mathematics, Brown University, Providence, RI, USA*
[b] *Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA*

Abstract

This work leverages recent advances in probabilistic machine learning to discover governing equations expressed by parametric linear operators. Such equations involve, but are not limited to, ordinary and partial differential, integro-differential, and fractional order operators. Here, Gaussian process priors are modified according to the particular form of such operators and are employed to infer parameters of the linear equations from scarce and possibly noisy observations. Such observations may come from experiments or "black-box" computer simulations, as demonstrated in several synthetic examples and a realistic application in functional genomics.

Keywords: Probabilistic machine learning, Inverse problems, Fractional differential equations, Uncertainty quantification, Functional genomics

1. Introduction

A grand challenge with great opportunities facing researchers is to develop a coherent framework that enables them to blend differential equations with the vast data sets available in many fields of science and engineering. In particular, this article investigates governing equations of the form

$$u(x) \longrightarrow \boxed{\mathcal{L}_x^{\phi} : \phi = ?} \longrightarrow f(x),$$

where $u(x)$ is the unknown solution to a differential equation defined by the operator $\mathcal{L}_x^{\phi}$, $f(x)$ is a black-box forcing term, and x is a vector that can include space, time, or parameter coordinates. In other words, the relationship

Preprint submitted to Journal of Computational Physics *October 18, 2020*

© 2017 published by Elsevier. This manuscript is made available under the Elsevier user license
https://www.elsevier.com/open-access/userlicense/1.0/

between $u(x)$ and $f(x)$ can be expressed as

$$\mathcal{L}_x^{\phi} u(x) = f(x). \tag{1}$$

Throughout this work we will consider the case where $\mathcal{L}_x^{\phi}$ is a linear differential operator with a form that is known up to a set of parameters ϕ that are a-priori unknown. Take, for instance, the classical problem of heat conduction in a medium with unknown conductivity properties. The spatio-temporal distribution of temperature is governed by a linear time-dependent parabolic partial differential equation, albeit with an unknown thermal diffusivity coefficient α. We will revisit this problem is section 3.3 with the goal of estimating α and reconstructing the unknown temperature field directly from a finite set of noisy measurements.

In general, given noisy data $\{\boldsymbol{X}_u, \boldsymbol{y}_u\}$, $\{\boldsymbol{X}_f, \boldsymbol{y}_f\}$ of the unknown solution $u(x)$, and the forcing term $f(x)$, respectively, the aim is to learn the parameters ϕ and hence the governing equation which best describes the data. This setup defines a broad class of problems that are often referred to as "inverse problems" (see [21, 48]) and they permeate applications across a wide range of scientific disciplines. To provide a unified framework for resolving such problems, this work employs and modifies Gaussian processes (GPs) (see [54, 32]), which is a non-parametric Bayesian machine learning technique. Quoting Diaconis [13], "once we allow that we don't know f (and u), but do know some things, it becomes natural to take a Bayesian approach". The Bayesian procedure adopted here, namely Gaussian processes, provides a flexible prior distribution over functions, enjoys analytical tractability, and has a fully probabilistic workflow that returns robust posterior variance estimates which quantify uncertainty in a natural way. Moreover, Gaussian processes are among a class of methods known as kernel machines (see [53, 46, 51]) and have analogies with regularization approaches (see [49, 50, 38]). They can also be viewed as a prior on one-layer feed-forward Bayesian neural networks with an infinite number of hidden units [33]. However, they are distinguished by their probabilistic viewpoint and their powerful training procedure. Along exactly the same lines, the methodology outlined in this work is related to and yet fundamentally differentiated from the so-called "meshless" methods (see [17, 16, 35, 10, 11]) for differential equations. A key difference of our work in comparison with the aforementioned approaches is that the optimal model parameters and hyper-parameters are all *learned* directly from the data by maximizing the *joint marginal log-likelihood* of the probabilistic model instead of being guessed or tuned manually by the user.

Furthermore, differential equations are the cornerstone of a diverse range of applied sciences and engineering fields. However, their use within statistics and machine learning, and combination with probabilistic models is less explored. Perhaps the most significant related work in this direction is latent force models [3, 2]. Such models generalize latent variable models [25, 26, 52] using differential equations. In contrast to latent force models, the current work bypasses the need for solving differential equations either analytically or numerically by placing the Gaussian process prior on $u(x)$ rather than on $f(x)$.

2. Methodology

The proposed data-driven algorithm for learning general parametric linear equations of the form presented in Eq. 1, employs Gaussian process priors that are tailored to the corresponding differential operators.

2.1. Prior

Specifically, the algorithm starts by making the assumption that $u(x)$ is Gaussian process [54] with mean 0 and covariance function $k_{uu}(x, x'; \theta)$, i.e.,

$$u(x) \sim \mathcal{GP}(0, k_{uu}(x, x'; \theta)),$$

where θ denotes the hyper-parameters of the kernel k_{uu}. The choice of the kernel k_{uu} allows us to encode any prior knowledge we may have about $u(x)$ (e.g., periodicity, monotonicity, smoothness, etc.), and can accommodate the approximation of arbitrarily complex functions [54]. The key observation here is that any linear transformation of a Gaussian process such as differentiation and integration is still a Gaussian process. Consequently,

$$\mathcal{L}_x^\phi u(x) = f(x) \sim \mathcal{GP}(0, k_{ff}(x, x'; \theta, \phi)),$$

with the following fundamental relationship between the kernels k_{uu} and k_{ff},

$$k_{ff}(x, x'; \theta, \phi) = \mathcal{L}_x^\phi \mathcal{L}_{x'}^\phi k_{uu}(x, x'; \theta). \tag{2}$$

Moreover, the covariance between $u(x)$ and $f(x')$, and similarly the one between $f(x)$ and $u(x')$, are given by $k_{uf}(x, x'; \theta, \phi) = \mathcal{L}_{x'}^\phi k_{uu}(x, x'; \theta)$, and $k_{fu}(x, x'; \theta, \phi) = \mathcal{L}_x^\phi k_{uu}(x, x'; \theta)$, respectively. The reasoning leading to Eq.

2 which is at the heart of our proposed methodology, dates back to the original work of [18, 45] and has been again recently revived among others in [10, 11, 31, 14, 40].

A major contribution of the current work can be best recognized by noticing how the parameters ϕ of the operator $\mathcal{L}_x^\phi$ are turned into hyper-parameters of the kernels k_{ff}, k_{uf}, and k_{fu}. This enables us to infer these parameters directly from data by maximizing the marginal log-likelihood of the Gaussian process model. A similar approach has been adopted in [27], and further developed in the later work of [3]. An important difference between our methodology and these approaches stems from the placement of the GP prior. Unlike [3, 2, 27], here we place the GP prior on $u(x)$ (see [18, 31, 14]) instead of $f(x)$ (see [3, 2, 6, 27]). As will be shown later in section 3, this enables us to jointly infer $u(x), f(x)$, and the unknown parameters ϕ of the linear operator $\mathcal{L}_x^\phi$, bypassing the need for inverting the operator either analytically or numerically.

2.1.1. Kernels [54]

Without loss of generality, all Gaussian process priors used in this work are assumed to have a squared exponential covariance function, i.e.,

$$k_{uu}(x, x'; \theta) = \sigma_u^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D} w_d (x_d - x'_d)^2\right),$$

where σ_u^2 is a variance parameter, x is a D-dimensional vector that includes spatial or temporal coordinates, and $\theta = \left(\sigma_u^2, (w_d)_{d=1}^D\right)$. Moreover, anisotropy across input dimensions is handled by Automatic Relevance Determination (ARD) weights w_d. From a theoretical point of view, each kernel gives rise to a Reproducing Kernel Hilbert Space [4, 44, 5] that defines a class of functions that can be represented by this kernel. In particular, the squared exponential covariance function chosen above implies smooth approximations. More complex function classes can be accommodated by appropriately choosing kernels. For example, non-stationary kernels employing nonlinear warpings of the input space can be constructed to capture discontinuous response [8]. In general, the choice of kernels is crucial and in many cases still remains an art that relies on one's ability to encode any prior information (such as known symmetries, invariances, etc.) into the regression scheme. In this work, this prior information is the knowldge of the parametric linear operator $\mathcal{L}_x^\phi$ itself. Starting, for example, from a base square exponential kernel and applying $\mathcal{L}_x^\phi$

as descibed above, one obtains a prior that is adapted to the linear operator and inherits its underlying structure. Our empirical findings so far indicate that this procedure is quite robust and insensitive to the choice of the base kernel, although this observation should be interpreted as a conjecture rather as a firm result.

2.2. Training

The hyper-parameters θ and more importantly the parameters ϕ of the linear operator $\mathcal{L}_x^\phi$ can be trained by employing a Quasi-Newton optimizer L-BFGS [29] to minimize the negative log marginal likelihood [54]

$$-\log p(\boldsymbol{y}|\phi, \theta, \sigma^2_{n_u}, \sigma^2_{n_f}) = \frac{1}{2}\log|\boldsymbol{K}| + \frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}^{-1}\boldsymbol{y} + \frac{N}{2}\log 2\pi, \tag{3}$$

where $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_u \\ \boldsymbol{y}_f \end{bmatrix}$, $p(\boldsymbol{y}|\phi, \theta, \sigma^2_{n_u}, \sigma^2_{n_f}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, and $\boldsymbol{K}$ is given by

$$\boldsymbol{K} = \begin{bmatrix} k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \theta) + \sigma^2_{n_u}\boldsymbol{I}_{n_u} & k_{uf}(\boldsymbol{X}_u, \boldsymbol{X}_f; \theta, \phi) \\ k_{fu}(\boldsymbol{X}_f, \boldsymbol{X}_u; \theta, \phi) & k_{ff}(\boldsymbol{X}_f, \boldsymbol{X}_f; \theta, \phi) + \sigma^2_{n_f}\boldsymbol{I}_{n_f} \end{bmatrix}. \tag{4}$$

Here, $\sigma^2_{n_u}$ and $\sigma^2_{n_f}$ are included to capture noise in the data and are also inferred from the data. The implicit underlying assumption is that $\boldsymbol{y}_u = u(\boldsymbol{X}_u) + \boldsymbol{\epsilon}_u$, $\boldsymbol{y}_f = f(\boldsymbol{X}_f) + \boldsymbol{\epsilon}_f$ with $\boldsymbol{\epsilon}_u \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{n_u}\boldsymbol{I}_{n_u})$, $\boldsymbol{\epsilon}_f \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{n_f}\boldsymbol{I}_{n_f})$. Moreover, $\boldsymbol{\epsilon}_u$ and $\boldsymbol{\epsilon}_f$ are assumed to be independent. It is worth mentioning that the marginal likelihood does not simply favor the models that fit the training data best. In fact, it induces an automatic trade-off between data-fit and model complexity. This effect is called Occam's razor [42] after William of Occam (1285–1349), who encouraged simplicity in explanations by the principle: "plurality should not be assumed without necessity". Specifically, minimizing the $\boldsymbol{y}^T\boldsymbol{K}^{-1}\boldsymbol{y}$ term in Eq. 3 targets fitting the training data, while the log-determinant term $\log|\boldsymbol{K}|$ penalizes model complexity. This mechanism automatically meets the Occam's razor principle, albeit at the computational cost of obtaining the Cholesky factors of $\boldsymbol{K}$ that are used to compute both the inverse and the determinant. This natural regularization mechanism is a key property of Gaussian process regression and it enables inferring the unknown model parameters from very few data while effectively guarding against overfitting. However, regularization still remains an important factor even in cases where data is abundant as seen in the recently growing literature on discovering ordinary and partial differential equations from data using sparse regression techniques [7, 43].

Model training practically consists of minimizing Eq. 3 with respect to the model paramters and hyper-parameters, namely $\{\phi, \theta, \sigma_{n_u}^2, \sigma_{n_f}^2\}$. This defines a non-convex optimization problem, and the common practice of employing classical gradient descent algorithms may lead to a local minimum. Therefore, the proposed method is also susceptible to converging to a set of hyper-parameters that corresponds to a local minimum of the negative marginal log likelihood. This behavior is standard for many machine learning algorithms (e.g., Gaussian processes, neural networks) [54], and the efficient *global* optimization still remains an open problem. In practice, this issue is addressed by solving the optimization problem from different hyper-parameter initializations, and returning the solution that yields the smallest negative marginal log likelihood. Although this still does not guarantee convergence to a global optimum, it is usually sufficient for obtaining a good solution.

The most computationally intensive part of training is associated with inverting dense covariance matrices $\boldsymbol{K}$. This scales cubically with the number of training data in $\boldsymbol{y}$. Although this scaling is a well-known limitation of Gaussian process regression, it must be emphasized that it has been effectively addressed by the recent works of [47, 19]. Furthermore, it should be emphasized that, although not pursued here, a fully Bayesian [48] and more robust estimate of the linear operator parameters ϕ can be obtained by assigning priors on $\{\theta, \phi, \sigma_{n_u}^2, \sigma_{n_f}^2\}$. However, this would require more costly sampling procedures such as Markov Chain Monte Carlo integration (see [54], chapter 5) to train the model.

2.3. Prediction

Having trained the model, one can predict the values $u(x)$ and $f(x)$ at a new test point x by writing the posterior distributions

$$\begin{aligned} p(u(x)|\boldsymbol{y}) &= \mathcal{N}\left(\overline{u}(x), s_u^2(x)\right), & (5)\\ p(f(x)|\boldsymbol{y}) &= \mathcal{N}\left(\overline{f}(x), s_f^2(x)\right), & (6) \end{aligned}$$

with

$$\overline{u}(x) = \boldsymbol{q}_u^T \boldsymbol{K}^{-1} \boldsymbol{y},\ \ s_u^2(x) = k_{uu}(x,x) - \boldsymbol{q}_u^T \boldsymbol{K}^{-1} \boldsymbol{q}_u,$$
$$\overline{f}(x) = \boldsymbol{q}_f^T \boldsymbol{K}^{-1} \boldsymbol{y},\ \ s_f^2(x) = k_{ff}(x,x) - \boldsymbol{q}_f^T \boldsymbol{K}^{-1} \boldsymbol{q}_f,$$

where

$$\boldsymbol{q}_u^T = [k_{uu}(x, \boldsymbol{X}_u) \ k_{uf}(x, \boldsymbol{X}_f)] \, ,$$
$$\boldsymbol{q}_f^T = [k_{fu}(x, \boldsymbol{X}_u) \ k_{ff}(x, \boldsymbol{X}_f)] \, .$$

Note that, for notational convenience, the dependence of kernels on hyperparameters and other parameters is dropped. The posterior variances $s_u^2(x)$ and $s_f^2(x)$ can be used as good indicators of how confident one could be about predictions made based on the learned parameters ϕ. Furthermore, such built-in quantification of uncertainty encoded in the posterior variances is a direct consequence of the Bayesian approach adopted in this work. Although not pursued here, this information is very useful in designing a data acquisition plan, often referred to as *active learning* [12, 24, 30], that can be used to optimally enhance our knowledge about the parametric linear equation under consideration.

3. Results

The proposed algorithm provides a general treatment of linear operators, which can be of fundamentally different nature. For example, one can seamlessly learn parametric integro-differential, time-dependent, or even fractional equations. This generality will be demonstrated using three benchmark problems with simulated data. Moreover, the methodology will be applied to realistic problem in functional genomics, namely determining the structure and dynamics of genetic networks based on real expression data [36] on early *Drosophila melanogaster* development.

3.1. Fractional Equation

Consider the one dimensional fractional equation

$$\mathcal{L}_x^{\alpha} u(x) = {}_{-\infty}D_x^{\alpha} u(x) - u(x) = f(x),$$

where $\alpha \in \mathbb{R}$ is the fractional order of the operator that is defined in the Riemann-Liouville sense [37]. Fractional operators often arise in modeling anomalous diffusion processes and other non-local interactions. Their non-local behavior poses serious computational challenges as it involves costly convolution operations for resolving the underlying non-Markovian dynamics [37]. However, the machine leaning approach pursued in this work bypasses the need for numerical discretization, hence, overcomes these computational bottlenecks, and can seamlessly handle all such linear cases without any modifications. The only technicality induced by fractional operators has to do with deriving the kernel $k_{ff}(x, x'; \theta, \alpha)$ in Eq. 2. Here, $k_{ff}(x, x'; \theta, \alpha)$ was obtained by taking the inverse Fourier transform [37] of

$$[(-iw)^{\alpha}(-iw')^{\alpha} - (-iw)^{\alpha} - (-iw')^{\alpha} + 1]\widehat{k}_{uu}(w, w'; \theta),$$

where $\widehat{k}_{uu}(w, w'; \theta)$ is the Fourier transform of the kernel $k_{uu}(x, x'; \theta)$. Similarly, one can obtain $k_{uf}(x, x'; \theta, \alpha)$ and $k_{fu}(x, x'; \theta, \alpha)$.

Note that for $\alpha = \sqrt{2}$, the functions $u(x) = \frac{1}{2}e^{-2i\pi x}\left(\frac{(2\pi+i)e^{4i\pi x}}{-1+(2i\pi)^{\sqrt{2}}} + \frac{2\pi-i}{-1+(-2i\pi)^{\sqrt{2}}}\right)$ and $f(x) = 2\pi\cos(2\pi x) - \sin(2\pi x)$ satisfy the fractional equation. Assume that the noise-free data $\{\boldsymbol{x}_u, \boldsymbol{y}_u\}$, $\{\boldsymbol{x}_f, \boldsymbol{y}_f\}$ on $u(x)$, $f(x)$ are generated according to $\boldsymbol{y}_u = u(\boldsymbol{x}_u)$, $\boldsymbol{y}_f = f(\boldsymbol{x}_f)$ with $\boldsymbol{x}_u$, $\boldsymbol{x}_f$ constituting of $n_u = 5$, $n_f = 4$ data points chosen at random in the interval $[0, 1]$, respectively. Given these noise-free training data, the algorithm learns the parameter α to have value 1.412104. The resulting posterior distributions for $u(x)$ and $f(x)$ are depicted in Figure 1.

In order to asses the sensitivity of the proposed algorithm on the size of the training data, we have repeated this numerical study using different training sets. These sets are constructed by randomly sampling data for $u(x)$ and $f(x)$ using a total number of training points in the range between 2 and 10. Figure 2(A) depicts the absolute error in logarithmic scale between the true fractional order α and the predicted one as a function of the total number of training points for $u(x)$ and $f(x)$, denoted by n_u and n_f, respectively. We can see that the algorithm converges to the true parameter value as the number of training points is increased. This is further confirmed by Figure 2(B), where we plot the $\mathbb{L}_2$ error between the true and the prediction solution $u(x)$. Evidently, as the number of training points is increased we obtain a more accurate solution, and, consequently, a more accurate estimation of the unknown fractional order α.

This blending of fractional calculus and machine learning is a major contribution of this work as it leads to an important observation that underpins the ability of the proposed framework to *learn* general linear differential operators from data. For example, integer values such as $\alpha = 1$ and $\alpha = 2$ can model classical advection and diffusion phenomena, respectively. However, under the fractional calculus setting, α can assume real values and thus continuously interpolate between inherently different model behaviors.

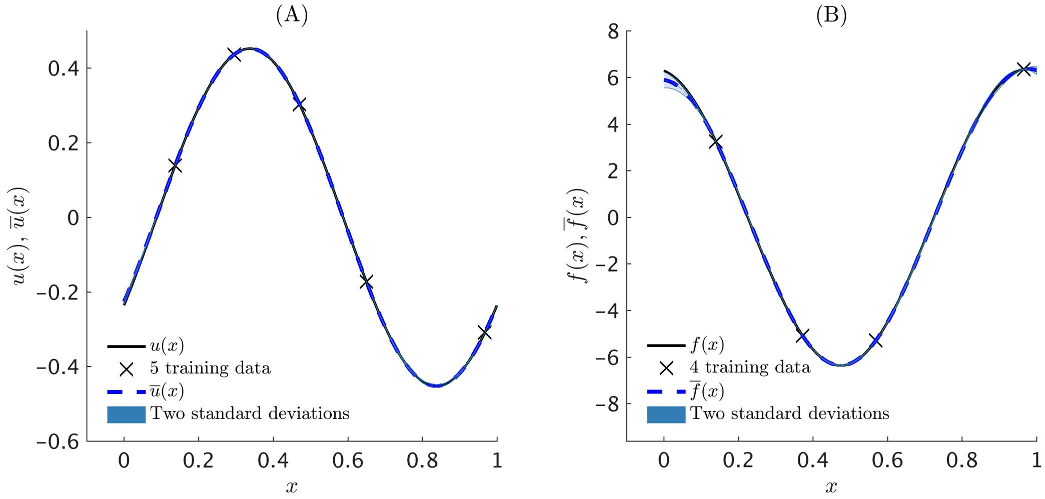

Figure 1: **Fractional equation in 1D: (A)** Exact left-hand-side function $u(x)$, training data $\{\boldsymbol{x}_u, \boldsymbol{y}_u\}$, predictive mean $\overline{u}(x)$, and two-standard-deviation band around the mean. **(B)** Exact right-hand-side function $f(x)$, training data $\{\boldsymbol{x}_f, \boldsymbol{y}_f\}$, predictive mean $\overline{f}(x)$, and two-standard-deviation band around the mean.

Even more generally, α can be variable in space-time or even governed by an unknown distribution [23], allowing one to model complex systems with memory and intricate transition dynamics. The proposed framework allows α to be directly inferred from noisy data, and opens the path to a flexible formalism for model discovery and calibration.

3.2. Integro-differential equation in 1D

Consider the following integro-differential equation,

$$\mathcal{L}_x^{(\alpha,\beta)} u(x) := \frac{d}{dx} u(x) + \alpha u(x) + \beta \int_0^x u(\xi) d\xi = f(x). \tag{7}$$

Note that for $(\alpha, \beta) = (2, 5)$, the functions $u(x) = \sin(2\pi x)$ and $f(x) = 2\pi \cos(2\pi x) + \frac{5}{\pi} \sin(\pi x)^2 + 2\sin(2\pi x)$ satisfy this equation. In the following, the parameters (α, β) will be inferred from two types of data, namely, noise-free and noisy observations.

Noise-free data

Assume that the noise-free data $\{\boldsymbol{x}_u, \boldsymbol{y}_u\}$, $\{\boldsymbol{x}_f, \boldsymbol{y}_f\}$ on $u(x)$, $f(x)$ are generated according to $\boldsymbol{y}_u = u(\boldsymbol{x}_u)$, $\boldsymbol{y}_f = f(\boldsymbol{x}_f)$ with $\boldsymbol{x}_u$, $\boldsymbol{x}_f$ constituting

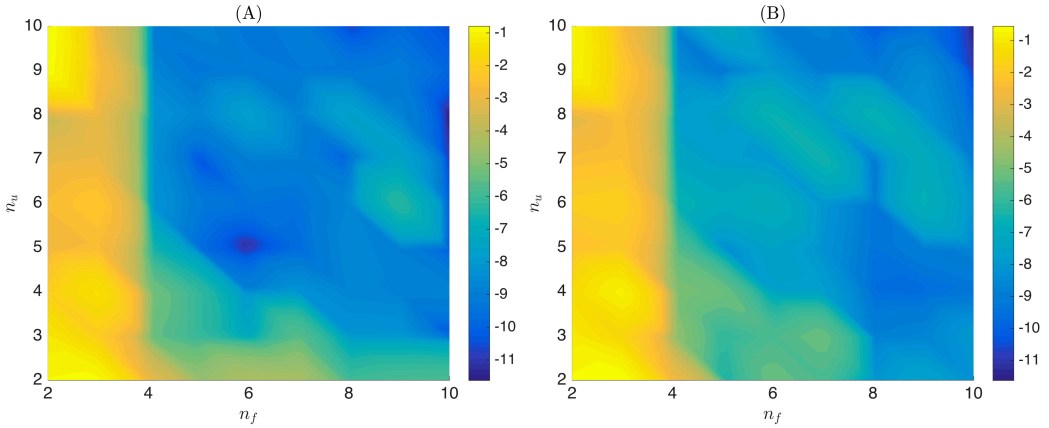

Figure 2: **Fractional equation in 1D: (A)** Absolute error in logarithmic scale between the true fractional order α and the predicted one as a function of the total number of training points for $u(x)$ and $f(x)$, denoted by n_u and n_f, respectively. **(B)** $\mathbb{L}_2$ error between the true and the prediction solution $u(x)$.

of $n_u = 4$, $n_f = 3$ data points chosen at random in the interval $[0, 1]$, respectively. Given these noise-free training data, the algorithm learns the parameters (α, β) to have values $(2.012627,\ 4.977879)$. It should be emphasized that the algorithm is able to learn the parameters of the operator using only 7 training data. Moreover, the resulting posterior distributions for $u(x)$ and $f(x)$ are depicted in Figure 3(A, B). The posterior variances could be used an indicator of how uncertain one should be about the estimated parameters and predictions made based on these parameters. As expected, the posterior variance grows in regions of the space where we don't have data, e.g., near the domain boundaries (see Figure 3(B, D)).

Noisy data

Consider the case where the noisy data $\{\boldsymbol{x}_u, \boldsymbol{y}_u\}$, $\{\boldsymbol{x}_f, \boldsymbol{y}_f\}$ on $u(x)$, $f(x)$ are generated according to $\boldsymbol{y}_u = u(\boldsymbol{x}_u) + \boldsymbol{\epsilon}_u$, $\boldsymbol{y}_f = f(\boldsymbol{x}_f) + \boldsymbol{\epsilon}_f$ with $\boldsymbol{x}_u$, $\boldsymbol{x}_f$ constituting of $n_u = 14$, $n_f = 10$ data points chosen at random in the interval $[0, 1]$, respectively. Here, the noise $\boldsymbol{\epsilon}_u$ and $\boldsymbol{\epsilon}_f$ are randomly generated according to the normal distributions $\mathcal{N}(\mathbf{0}, 0.1^2\ \boldsymbol{I}_{n_u})$ and $\mathcal{N}(\mathbf{0}, 0.5^2\ \boldsymbol{I}_{n_f})$, respectively. Given these noisy training data, the algorithm learns the parameters (α, β) to have values $(2.073054,\ 5.627249)$. It should be emphasized that for this example the data is deliberately chosen to have a sizable noise. This highlights the ability of the method to handle highly noisy observations

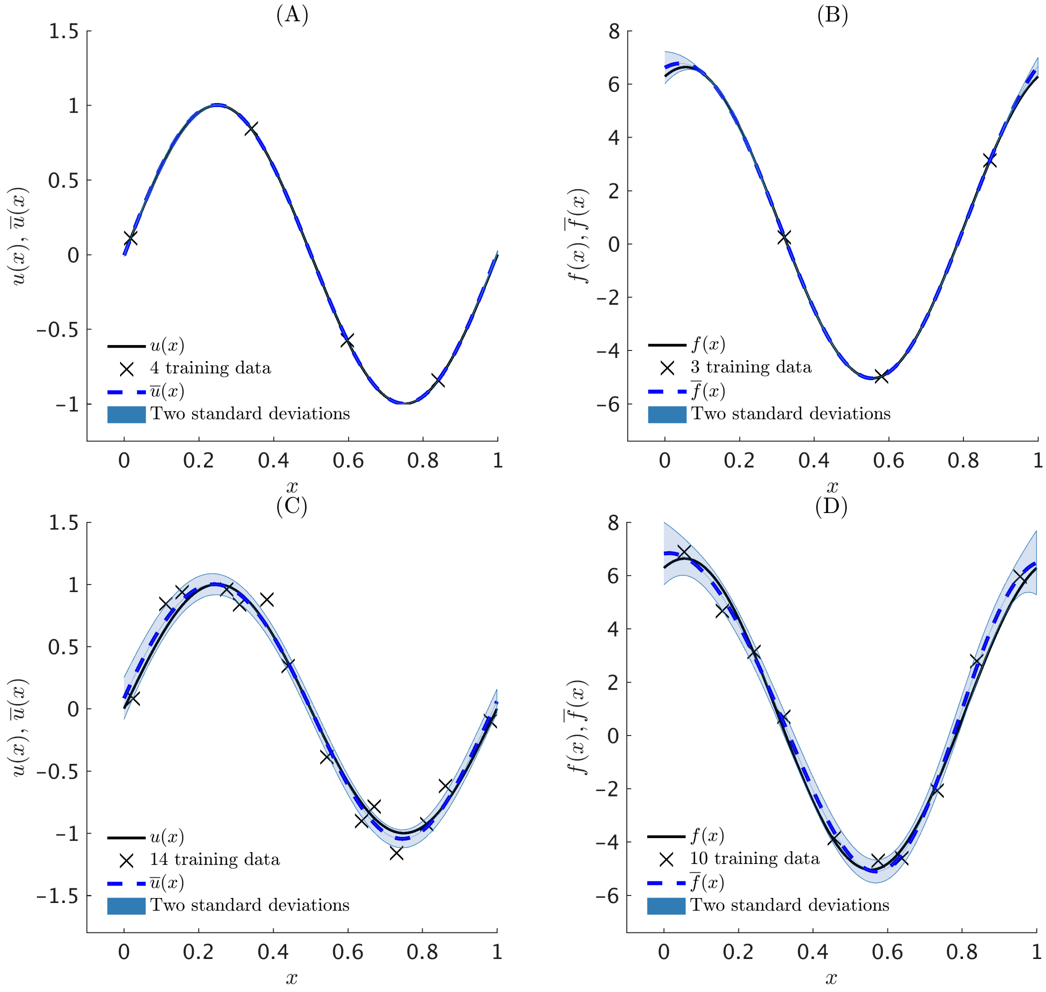

Figure 3: **Integro-differential equation in 1D: (A)** Exact left-hand-side function $u(x)$, "noise-free" training data $\{\boldsymbol{x}_u, \boldsymbol{y}_u\}$, predictive mean $\overline{u}(x)$, and two-standard-deviation band around the mean. **(B)** Exact right-hand-side function $f(x)$, "noise-free" training data $\{\boldsymbol{x}_f, \boldsymbol{y}_f\}$, predictive mean $\overline{f}(x)$, and two-standard-deviation band around the mean. **(C)** Exact left-hand-side function $u(x)$, "noisy" training data $\{\boldsymbol{x}_u, \boldsymbol{y}_u\}$, predictive mean $\overline{u}(x)$, and two-standard-deviation band around the mean. **(D)** Exact right-hand-side function $f(x)$, "noisy" training data $\{\boldsymbol{x}_f, \boldsymbol{y}_f\}$, predictive mean $\overline{f}(x)$, and two-standard-deviation band around the mean.

without any modifications. The resulting posterior distributions for $u(x)$ and $f(x)$ are depicted in Figure 3(C, D). By construction, the posterior variances are able to quantify scarcity in observations but also account for noise in the training data.

3.3. Heat Equation

This example is chosen to highlight the capability of the proposed framework to handle time-dependent problems using only scattered space-time observations. To this end, consider the heat equation

$$\mathcal{L}^{\alpha}_{(t,x)} u(t,x) := \frac{\partial}{\partial t} u(t,x) - \alpha \frac{\partial^2}{\partial x^2} u(t,x) = f(t,x).$$

Note that for $\alpha = 1$, the functions $f(t,x) = e^{-t}(4\pi^2 - 1)\sin(2\pi x)$ and $u(t,x) = e^{-t}\sin(2\pi x)$ satisfy this equation. Assume that the noise-free data $\{(\boldsymbol{t}_u, \boldsymbol{x}_u), \boldsymbol{y}_u\}$, $\{(\boldsymbol{t}_f, \boldsymbol{x}_f), \boldsymbol{y}_f\}$ on $u(t,x)$, $f(t,x)$ are generated according to $\boldsymbol{y}_u = u(\boldsymbol{t}_u, \boldsymbol{x}_u)$, $\boldsymbol{y}_f = f(\boldsymbol{t}_f, \boldsymbol{x}_f)$ with $(\boldsymbol{t}_u, \boldsymbol{x}_u)$, $(\boldsymbol{t}_f, \boldsymbol{x}_f)$ constituting of $n_u = n_f = 20$ data points chosen at random in the domain $[0,1]^2$, respectively. Given these training data, the algorithm learns the parameter α to have value 0.999943. The resulting posterior distributions for $u(t,x)$ and $f(t,x)$ are depicted in Figure 4. A visual inspection of this figure illustrates how closely uncertainty in predictions measured by posterior variances (see Figure 4(E, F)) correlate with prediction errors (see Figure 4(C, D)). Remarkably, the proposed methodology circumvents the need for temporal discretization, and is essentially immune to any restrictions arising due to time-stepping, e.g., the fundamental consistency and stability issues in classical numerical analysis.

3.4. Drosophila melanogaster gap gene dynamics [36, 3]

The gap gene dynamics of protein $a \in \{Hb, Kr, Gt, Kni\}$ (see figure 5) can be modeled using a reaction-diffusion partial differential equation

$$\mathcal{L}^{(\lambda^a, D^a)}_{(t,x)} u^a(t,x) = \frac{\partial}{\partial t} u^a(t,x) + \lambda^a u^a(t,x) - D^a \frac{\partial^2}{\partial x^2} u^a(t,x) = f^a(t,x),$$

where $u^a(t,x)$ denotes the relative concentration of gap protein a (unitless, ranging from 0 to 255) at space point x (from 35% to 92% of embryo length) and time t (0 min to 68 min after the start of cleavage cycle 13). Here, λ^a

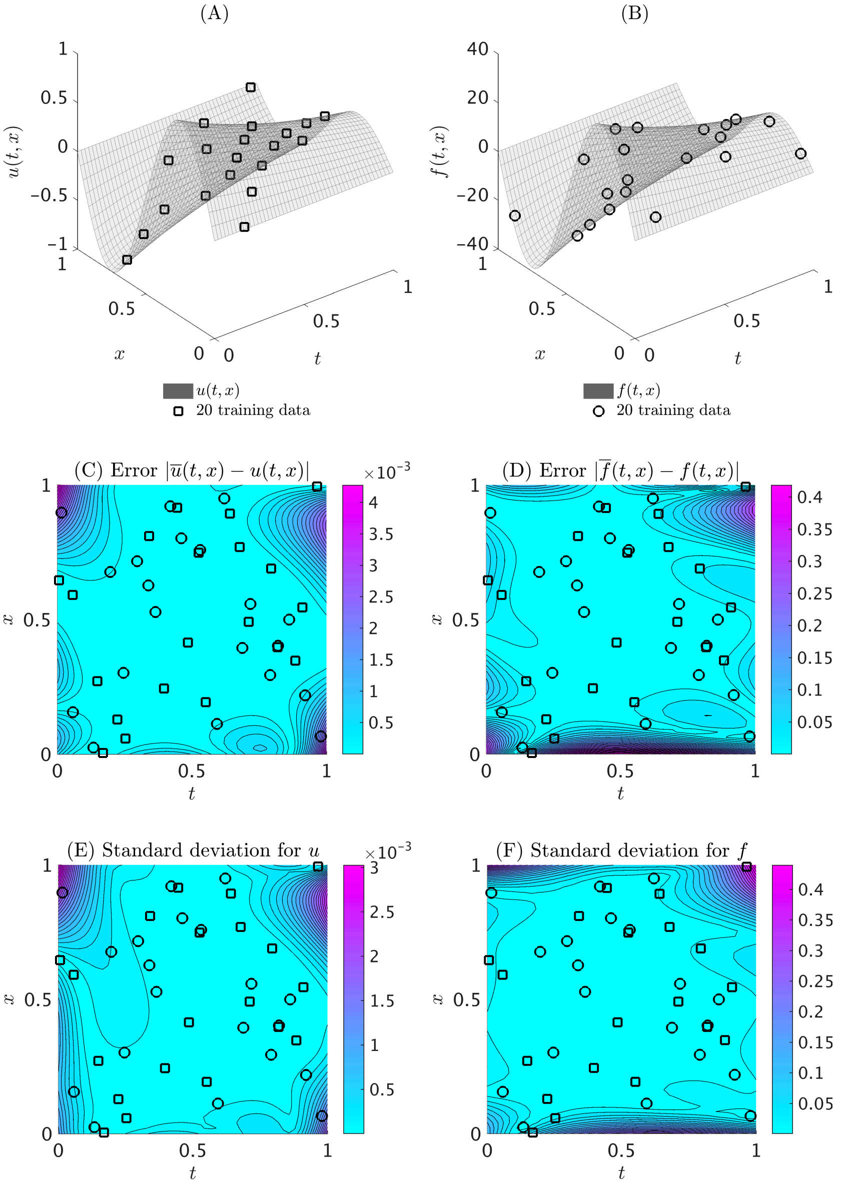

Figure 4: **Heat equation:** **(A)** Exact left-hand-side function $u(t,x)$ and training data $\{(\boldsymbol{t}_u, \boldsymbol{x}_u), \boldsymbol{y}_u\}$. **(B)** Exact right-hand-side function $f(t,x)$ and training data $\{(\boldsymbol{t}_f, \boldsymbol{x}_f), \boldsymbol{y}_f\}$. **(C)** Absolute point-wise error between the predictive mean $\overline{u}(t,x)$ and the exact function $u(t,x)$. The relative L_2 error for the left-hand-side function is 1.250278e−03. **(D)** Absolute point-wise error between the predictive mean $\overline{f}(t,x)$ and the exact function $f(t,x)$. The relative L_2 error for the right-hand-side function is 4.167404e−03. **(E)**, **(F)** Standard deviations $s_u(t,x)$ and $s_f(t,x)$ for u and f, respectively.

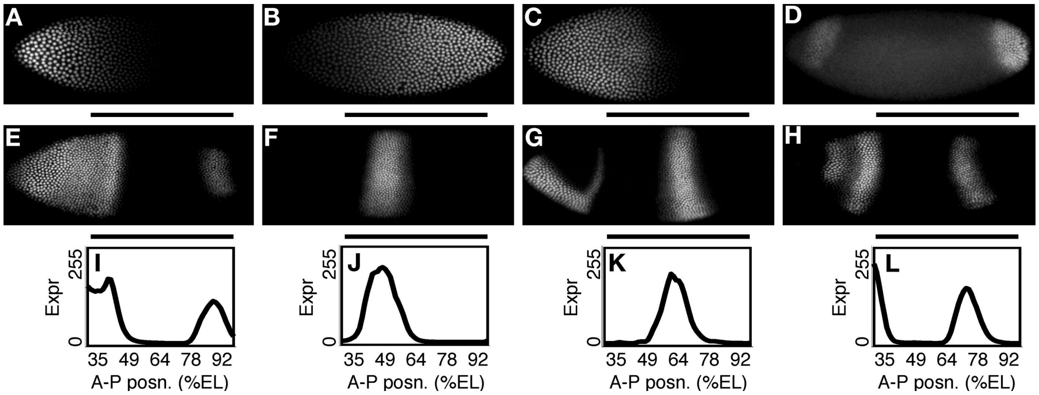

Figure 5: **Maternal and Gap Gene Expression (see [36])**: **(A–C)** Drosophila embryos at early blastoderm stage (cleavage cycle 13) fluorescently stained for Bcd **(A)**, Cad **(B)**, and Hb **(C)** protein. **(D–H)** Drosophila embryos at late blastoderm stage (late cleavage cycle 14A) fluorescently stained for Tll **(D)**, Hb **(E)**, Kr **(F)**, Kni **(G)**, and Gt **(H)** protein. Anterior is to the left, dorsal is up. Black bars indicate the modeled A-P extent. **(I–L)** Mean relative gap protein concentration as a function of A-P position (measured in percent embryo length) for Hb **(I)**, Kr **(J)**, Kni **(K)**, and Gt **(L)**. Expression levels are from images and are unitless, ranging from 0 to 255. Images and expression profiles are from the FlyEx database [39]. **Embryo IDs:** bd3 (A,B), hz30 (C), tb6 (D), kf9 (E), kd17 (F), fq1 (G), nk5 (H). **Abbreviations:** A-P, anterior-posterior; Bcd, Bicoid; Cad, Caudal; Gt, Giant; Hb, hunchback; Kni, Knirps; Kr, Krüppel; Tll, tailless.

and D^a are decay and diffusion rates of protein a, respectively. Moreover, the right-hand-side is given by

$$f^a(t,x) := \zeta(t)P^a(t,x),$$

where the term

$$\zeta(t) = \begin{cases} 0.5 & 0 \text{ min } \leq t < 16 \text{ min} \\ 0.0 & 16 \text{ min } \leq t < 21 \text{ min} \\ 1.0 & 21 \text{ min } \leq t \end{cases}$$

models the doubling of nuclei and shutdown of transcription during mitosis and

$$P^a(t,x) = R^a g\left(\sum_b T^{ab}u^b(t,x) + h^a\right)$$

specifies the production rate of protein a. The model combines the processes of transcription and translation into a single production process $P^a(t,x)$.

Here, R^a is the maximum production rate,

$$g(u) = \frac{1}{2}\left(\frac{u}{\sqrt{u^2+1}} + 1\right),$$

and $b \in \{Bcd, Cad, Hb, Kr, Gt, Kni, Tll\}$ ranges over the seven genes (see figure 5). The regulatory weights T^{ab}, encode the effect protein b has on the production rate of protein a. If $T^{ab}\hat{a} > 0$ (or $T^{ab}\hat{a} < 0$), then gene b is interpreted as being an activator (or a repressor) of gene a.

	Max prod.	Regulatory weights (T^{ab})							Bias
Gene	rate (R^a)	Bcd	Cad	Hb	Kr	Gt	Kni	Tll	(h^a)
Hb	32.03	0.1114	-0.0054	0.0293	-0.0124	0.0553	-0.3903	0.0144	-3.5
Kr	16.70	0.1173	0.0215	-0.0498	0.0755	-0.0141	-0.0666	-1.2036	-3.5
Gt	25.15	0.0738	0.0180	-0.0008	-0.0758	0.0157	0.0056	-0.0031	-3.5
Kni	16.12	0.2146	0.0210	-0.1891	-0.0458	-0.1458	0.0887	-0.3028	-3.5

Table 1: Parameters R^a, T^{ab}, and h^a are assumed to be exogenously given and their values are taken from [36].

In the current work, we assume the maximum production rate R^a, the regulatory weights T^{ab} and the bias or offset h^a to be specified as in table 1, and we seek to learn the decay λ^a and diffusion D^a rates of protein a. In fact, table 2 summarizes the values learned by the algorithm for these parameters and figure 6 depicts the corresponding posterior distributions for $u^a(t,x)$ and $f^a(t,x)$. Indeed, figure 6 gives a good indication of how certain one could be about the estimated parameters and the predictions made based on them.

	Decay	Diff.
Gene	(λ^a)	(D^a)
Hb	0.1606	0.3669
Kr	0.0797	0.4490
Gt	0.1084	0.4543
Kni	0.0807	0.2683

Table 2: Inferred parameter values for the decay λ^a and diffusion D^a rates of protein a.

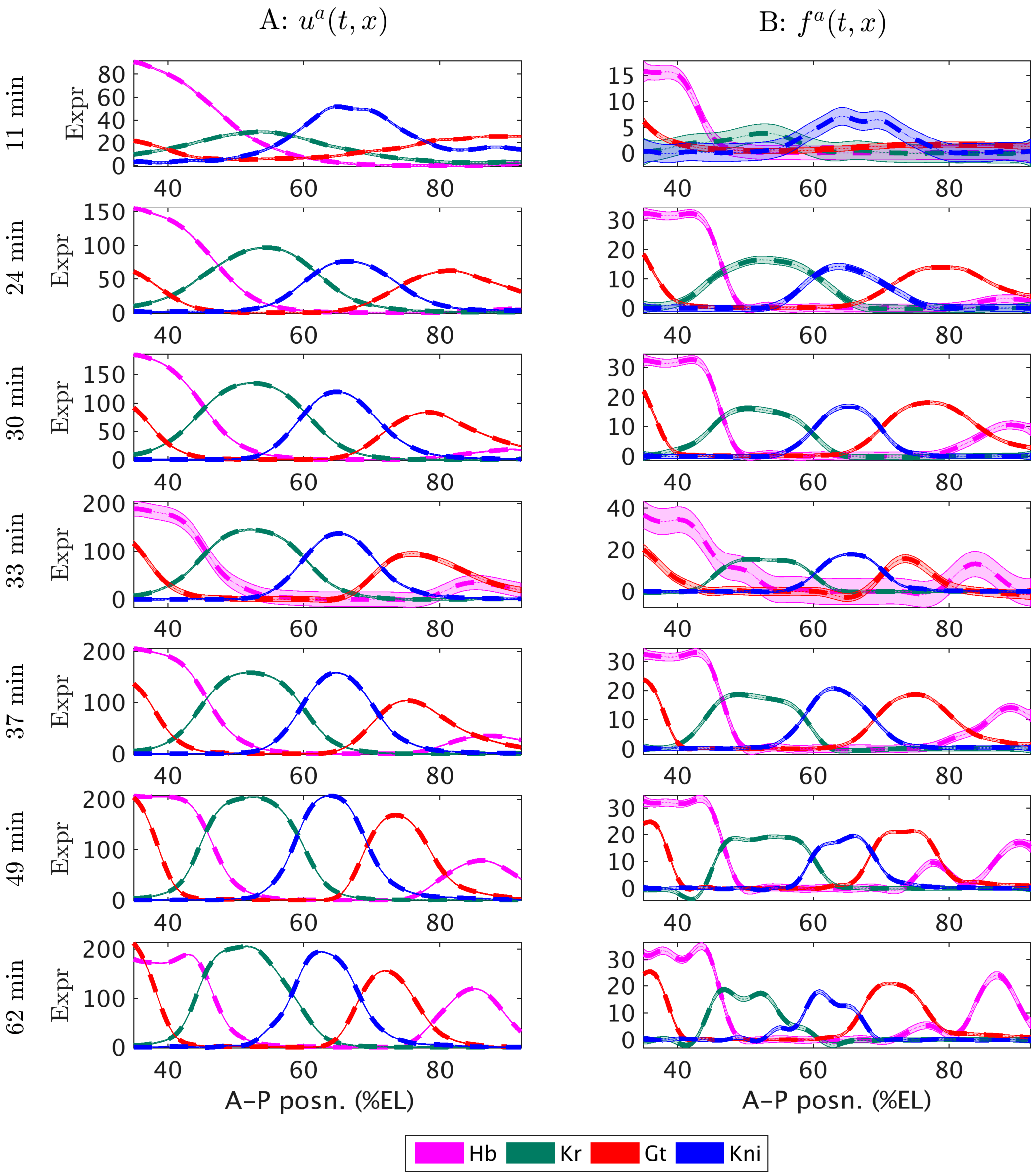

Figure 6: **Predictive expression at seven points in time**: **(A)** Predictive mean expression along with the two-standard-deviations band around the mean for Hb, Kr, Gt, and Kni. The vertical axis represents relative protein concentration corresponding to fluorescence intensity from quantitative gene expression data [36, 39]. **(B)** Predictive mean along with the two-standard-deviations band around the mean for the right-hand-side function corresponding to Hb, Kr, Gt, and Kni. The horizontal axis in each plot is A-P position, ranging from 35% to 92% of embryo length. No data points are available at time $t = 33$ min.

4. Discussion

In summary, this work introduced a computational framework for learning general parametric linear equations from noisy data. This generality was demonstrated using various benchmark problems with different attributes along with an example application in functional genomics. The methodology can be straightforwardly generalized to address data with multiple levels of fidelity by modeling the joint distribution of all observed data and modeling their cross-correlation using the auto-regressive structure first put forth by [22]. This is a simple extension of the recent work by Raissi *et. al.* [40] on inferring solutions of linear diffrential equations from noisy multi-fidelity data. The proposed methodology can also be extended to equations with variable coefficients, in which case, similarly to this work, the variability in the coeffcients will be absorbed in the kernels. Non-Gaussian and input-dependent noise models (e.g., student-t, heteroscedastic, etc.) [54] can also be accommodated. Moreover, systems of linear integro-differential equations can be addressed using multi-output Gaussian process regression [6, 1, 34]. Although in this study the model form was assumed to be known, another potential extension could pursue learning the model form itself using ideas from compositional kernel search [15]. All these scenarios are feasible because they do not affect the key observation that any linear transformation of a Gaussian process is still a Gaussian process. In its current form, despite its generality regarding linear equations, the proposed framework cannot deal with non-linear equations. However, some specific non-linear operators can be addressed with extensions of the current framework by transforming such equations into systems of linear equations [55, 9] – albeit in high dimensions. In the end, the proposed methodology in this work, being essentially a regression technology, is suitable for resolving such high-dimensional problems. Lastly, for general non-linear equations we can employ a sequential inference approach following the ideas recently put forth in [41].

Acknowledgements

This works received support by the DARPA EQUiPS grant N66001-15-2-4055, the MURI/ARO grant W911NF-15-1-0562, and the AFOSR grant FA9550-17-1-0013.

References

[1] Alvarez, M. & Lawrence, N. D. 2009 Sparse convolved Gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pp. 57–64.

[2] Alvarez, M. A., Luengo, D. & Lawrence, N. D. 2009 Latent force models. In *Aistats*, vol. 12, pp. 9–16.

[3] Álvarez, M. A., Luengo, D. & Lawrence, N. D. 2013 Linear latent force models using Gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, **35**(11), 2693–2705.

[4] Aronszajn, N. 1950 Theory of reproducing kernels. *Transactions of the American mathematical society*, **68**(3), 337–404.

[5] Berlinet, A. & Thomas-Agnan, C. 2011 *Reproducing kernel Hilbert spaces in probability and statistics.* Springer Science & Business Media.

[6] Boyle, P. & Frean, M. 2004 Dependent Gaussian processes. In *Advances in neural information processing systems*, pp. 217–224.

[7] Brunton, S. L., Proctor, J. L., & Kutz, J. N. 2016 Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, **113**(15), 3932–3937.

[8] Calandra, R., Peters, J., Rasmussen, C. E. & Deisenroth, M. P. 2016 Manifold Gaussian processes for regression. *2016 International Joint Conference on Neural Networks*, 3338–3345.

[9] Chorin, A. J., Hald, O. H. & Kupferman, R. 2000 Optimal prediction and the Mori–Zwanzig representation of irreversible processes. *Proceedings of the National Academy of Sciences*, **97**(7), 2968–2973.

[10] Cockayne, J., Oates, C., Sullivan, T. & Girolami, M. 2016 Probabilistic meshless methods for partial differential equations and bayesian inverse problems. *arXiv preprint arXiv:1605.07811.*

[11] Cockayne, J., Oates, C., Sullivan, T. & Girolami, M. 2016 Probabilistic numerical methods for PDE-constrained bayesian inverse problems. *Proceedings of the 36th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering.*

[12] Cohn, D. A., Ghahramani, Z. & Jordan, M. I. 1996 Active learning with statistical models. *Journal of artificial intelligence research.*

[13] Diaconis, P. 1988 Bayesian numerical analysis. *Statistical decision theory and related topics IV*, **1**, 163–175.

[14] Dondelinger, F., Husmeier, D., Rogers, S. & Filippone, M. 2013 ODE parameter inference using adaptive gradient matching with Gaussian processes. In *Aistats*, pp. 216–228.

[15] Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B. & Ghahramani, Z. 2013 Structure discovery in nonparametric regression through compositional kernel search In *arXiv preprint arXiv:1302.4922.*

[16] Fasshauer, G. E. & Ye, Q. 2013 A kernel-based collocation method for elliptic partial differential equations with random coefficients. In *Monte carlo and quasi-monte carlo methods 2012*, pp. 331–347. Springer.

[17] Franke, C. & Schaback, R. 1998 Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computation*, **93**(1), 73–82.

[18] Graepel, T. 2003 Solving noisy linear operator equations by gaussian processes: Application to ordinary and partial differential equations. In *Icml*, pp. 234–241.

[19] Hensman, J., Fusi, N. & Lawrence, N. D. 2013 Gaussian processes for big data. *arXiv preprint arXiv:1309.6835.*

[20] Higdon, D. 2002 Space and space-time modeling using process convolutions. In *Quantitative methods for current environmental issues*, pp. 37–56. Springer.

[21] Kaipio, J. & Somersalo, E. 2006 *Statistical and computational inverse problems*, vol. 160. Springer Science & Business Media.

[22] Kennedy, M. C. & O'Hagan, A. 2000 Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, **87**(1), 1–13.

[23] Kharazmi, E., Zayernouri, M. & Karniadakis, G. E. 2016 Petrov-galerkin and spectral collocation methods for distributed order differential equations. *arXiv preprint arXiv:1604.08650.*

[24] Krause, A. & Guestrin, C. 2007 Nonmyopic active learning of Gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th international conference on machine learning*, pp. 449–456. ACM.

[25] Lawrence, N. 2005 Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, **6**(Nov), 1783–1816.

[26] Lawrence, N. D. 2004 Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, **16**(3), 329–336.

[27] Lawrence, N. D., Sanguinetti, G. & Rattray, M. 2006 Modelling transcriptional regulation using Gaussian processes. In *Advances in neural information processing systems*, pp. 785–792.

[28] Le Gratiet, L. 2013 Multi-fidelity Gaussian process regression for computer experiments. Ph.D. thesis, Université Paris-Diderot-Paris VII.

[29] Liu, D. C. & Nocedal, J. 1989 On the limited memory BFGS method for large scale optimization. *Mathematical programming*, **45**(1-3), 503–528.

[30] MacKay, D. J. 1992 Information-based objective functions for active data selection. *Neural computation*, **4**(4), 590–604.

[31] Melkumyan, A. 2012 Operator induced multi-task Gaussian processes for solving differential equations. In *Neural information processing systems (nips) workshop" new directions in multiple kernel learning.*

[32] Murphy, K. P. 2012 *Machine learning: a probabilistic perspective.* MIT press.

[33] Neal, R. M. 2012 *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media.

[34] Osborne, M. A., Roberts, S. J., Rogers, A., Ramchurn, S. D. & Jennings, N. R. 2008 Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the 7th international conference on information processing in sensor networks*, pp. 109–120. IEEE Computer Society.

[35] Owhadi, H. 2015 Bayesian numerical homogenization. *Multiscale Modeling & Simulation*, **13**(3), 812–828.

[36] Perkins, T. J., Jaeger, J., Reinitz, J. & Glass, L. 2006 Reverse engineering the gap gene network of drosophila melanogaster. *PLoS Comput Biol*, **2**(5), e51.

[37] Podlubny, I. 1998 *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, vol. 198. Academic press.

[38] Poggio, T. & Girosi, F. 1990 Networks for approximation and learning. *Proceedings of the IEEE*, **78**(9), 1481–1497.

[39] Poustelnikova, E., Pisarev, A., Blagov, M., Samsonova, M. & Reinitz, J. 2004 A database for management of gene expression data in situ. *Bioinformatics*, **20**(14), 2212–2221.

[40] Raissi, M., Perdikaris, P. & Karniadakis, G. E. 2017 Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, **335**, 736–746.

[41] Raissi, M., Perdikaris, P. & Karniadakis, G. E. 2017 Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations *arXiv preprint arXiv:1703.10230*.

[42] Rasmussen, C. E. & Ghahramani, Z. 2001 Occam's razor. *Advances in neural information processing systems*, pp. 294–300.

[43] Rudy, S. H., Brunton, S. L., Proctor, J. L. & Kutz, J. N. 2017 Data-driven discovery of partial differential equations. *Science Advances*, **3**(4), e1602614.

[44] Saitoh, S. 1988 *Theory of reproducing kernels and its applications*, vol. 189. Longman.

[45] Särkkä, S. 2011 Linear operators and stochastic partial differential equations in Gaussian process regression. In *International conference on artificial neural networks*, pp. 151–158. Springer.

[46] Schölkopf, B. & Smola, A. J. 2002 *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press.

[47] Snelson, E. & Ghahramani, Z. 2005 Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pp. 1257–1264.

[48] Stuart, A. M. 2010 Inverse problems: a Bayesian perspective. *Acta Numerica*, **19**, 451–559.

[49] Tikhonov, A. 1963 Solution of incorrectly formulated problems and the regularization method. In *Soviet math. dokl.*, vol. 5, pp. 1035–1038.

[50] Tikhonov, A. N. & Arsenin, V. Y. 1977 *Solutions of ill-posed problems.* W.H. Winston.

[51] Tipping, M. E. 2001 Sparse Bayesian learning and the relevance vector machine. *The journal of machine learning research*, **1**, 211–244.

[52] Titsias, M. K. & Lawrence, N. D. 2010 Bayesian Gaussian process latent variable model. In *Aistats*, pp. 844–851.

[53] Vapnik, V. 2013 *The nature of statistical learning theory.* Springer Science & Business Media.

[54] Williams, C. K. & Rasmussen, C. E. 2006 Gaussian processes for machine learning. *the MIT Press*, **2**(3), 4.

[55] Zwanzig, R. 1960 Ensemble method in the theory of irreversibility. *The Journal of Chemical Physics*, **33**(5), 1338–1341.