SIAM J. SCI. COMPUT.
Vol. 40, No. 1, pp. A172–A198

NUMERICAL GAUSSIAN PROCESSES FOR TIME-DEPENDENT AND NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS*

MAZIAR RAISSI†, PARIS PERDIKARIS‡, AND GEORGE EM KARNIADAKIS†

Abstract. We introduce the concept of *numerical Gaussian processes*, which we define as Gaussian processes with covariance functions resulting from temporal discretization of time-dependent partial differential equations. Numerical Gaussian processes, by construction, are designed to deal with cases where (a) all we observe are noisy data on *black-box* initial conditions, and (b) we are interested in *quantifying the uncertainty* associated with such noisy data in our solutions to time-dependent partial differential equations. Our method circumvents the need for spatial discretization of the differential operators by proper placement of Gaussian process priors. This is an attempt to construct structured and data-efficient learning machines, which are explicitly informed by the underlying physics that possibly generated the observed data. The effectiveness of the proposed approach is demonstrated through several benchmark problems involving linear and nonlinear time-dependent operators. In all examples, we are able to recover accurate approximations of the latent solutions, and consistently propagate uncertainty, even in cases involving very long time integration.

Key words. probabilistic machine learning, linear multistep methods, Runge–Kutta methods, Bayesian modeling, uncertainty quantification

AMS subject classifications. 65C20, 68T05, 65M75

DOI. 10.1137/17M1120762

1. Introduction.

Data-driven methods are taking center stage across many disciplines of science, and machine learning techniques have achieved groundbreaking results across a diverse spectrum of pattern recognition tasks [10, 15, 16]. Despite their disruptive implications, many of these methods are blind to any underlying laws of physics that may have shaped the distribution of the observed data. A natural question would then be how one can construct efficient learning machines that explicitly leverage such structured prior information. To answer this question we have to turn our attention to the immense collective knowledge originating from centuries of research in applied mathematics and mathematical physics. Modeling the physical world through the lens of mathematics typically translates into deriving conservation laws from first principles, which often take the form of systems of partial differential equations. In many practical settings, the solution of such systems is only accessible by means of numerical algorithms that provide sensible approximations to given quantities of interest. In this work, we aim to capitalize on the long-standing developments of classical methods in numerical analysis and revisit partial differential equations from a *statistical inference* viewpoint. The merits of this approach are twofold. First, it enables the construction of data-efficient learning machines that can encode physical conservation laws as structured prior information. Second, it allows the design of novel numerical algorithms that can seamlessly blend equations and noisy data, infer

*Submitted to the journal's Methods and Algorithms for Scientific Computing section March 13, 2017; accepted for publication (in revised form) October 12, 2017; published electronically January 18, 2018.

http://www.siam.org/journals/sisc/40-1/M112076.html

Funding: This work was supported by DARPA EQUiPS grant N66001-15-2-4055, and by AFOSR grant FA9550-17-1-0013.

†Division of Applied Mathematics, Brown University, Providence, RI 02912 (maziar_raissi@brown.edu, george_karniadakis@brown.edu).

‡Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 (parisp@mit.edu).

latent quantities of interest (e.g., the solution to a partial differential equation), and naturally quantify uncertainty in computations. This approach is aligned in spirit with the emerging field of probabilistic numerics [1], which has roots all the way back to Poincaré's courses on probability theory [22] and has been recently revived by the pioneering works of [8, 9, 12, 20].

To illustrate the key ingredients of this study, let us start by considering linear[1] partial differential equations of the form

$$u_t = \mathcal{L}_x u, \quad x \in \Omega,\ t \in [0, T], \tag{1}$$

where $\mathcal{L}_x$ is a linear operator and $u(t, x)$ denotes the latent solution. As an example, the one dimensional heat equation corresponds to the case where $\mathcal{L}_x = \frac{\partial^2}{\partial x^2}$. Moreover, Ω is a subset of $\mathbb{R}^D$. All we observe are noisy data $\{\boldsymbol{x}^0, \boldsymbol{u}^0\}$ on the *black-box* initial function $u(0, x)$ as well as some information on the domain boundary $\partial\Omega$ to be specified later. Our goal is to predict the latent solution $u(t, x)$ at later times $t > 0$ and propagate the uncertainty due to noise in the initial data. In general, we envision that the proposed method could be most useful in cases where one would like to extrapolate from an initial condition obtained from noisy experimental data, and where a governing equation is known. Take, for example, the case of reconstructing a flow field from scattered measurements (e.g., particle image velocimetry data), and use the governing Navier–Stokes equations to extrapolate this initial condition in time. For starters, let us try to convey the main ideas of this work using the Euler time-stepping scheme

$$u^n = u^{n-1} + \Delta t\ \mathcal{L}_x u^{n-1}. \tag{2}$$

Here, $u^n(x) = u(t^n, x)$. Building upon the work of Raissi and coworkers [24, 25], we place a Gaussian process [26] prior on u^{n-1}, i.e.,

$$u^{n-1}(x) \sim \mathcal{GP}(0, k_{u,u}^{n-1,n-1}(x, x', \theta)). \tag{3}$$

Here, θ denotes the hyper-parameters of the covariance function $k_{u,u}^{n-1,n-1}$. Gaussian process regression (see [17, 26]) is a nonparametric Bayesian machine learning technique that provides a flexible prior distribution over functions, enjoys analytical tractability, and has a fully probabilistic work-flow 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 [31, 36, 37]) and are analogous to regularization approaches (see [21, 34, 35]). They can also be viewed as a prior on one-layer feed-forward Bayesian neural networks with an infinite number of hidden units [18]. The Gaussian process prior assumption (3) along with the Euler scheme (2) will allow us to capture the entire structure of the differential operator $\mathcal{L}_x$ as well as the Euler time-stepping rule in the resulting multioutput Gaussian process,

$$\begin{bmatrix} u^n \\ u^{n-1} \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k_{u,u}^{n,n} & k_{u,u}^{n,n-1} \\ & k_{u,u}^{n-1,n-1} \end{bmatrix}\right). \tag{4}$$

The specific forms of the kernels $k_{u,u}^{n,n}$ and $k_{u,u}^{n,n-1}$ are direct functions of the Euler scheme (2) as well as the prior assumption (3) and will be discussed in more detail later. The multioutput process (4) is an example of a *numerical Gaussian process*,

[1] Nonlinear equations have to be studied on a case-by-case basis (see, e.g., section 2.6).

TABLE 1
Some specific members of the family of linear multistep methods (5).

Forward Euler	$u^n = u^{n-1} + \Delta t \mathcal{L}_x u^{n-1}$
Backward Euler	$u^n = u^{n-1} + \Delta t \mathcal{L}_x u^n$
Trapezoidal rule	$u^n = u^{n-1} + \frac{1}{2}\Delta t \mathcal{L}_x u^{n-1} + \frac{1}{2}\Delta t \mathcal{L}_x u^n$

TABLE 2
Some special cases of (6).

Forward Euler	$u^n = \mathcal{Q}_x u^{n-1}$ $\mathcal{Q}_x u^{n-1} = u^{n-1} + \Delta t \mathcal{L}_x u^{n-1}$
Backward Euler	$\mathcal{P}_x u^n = u^{n-1}$ $\mathcal{P}_x u^n = u^n - \Delta t \mathcal{L}_x u^n$
Trapezoidal rule	$\mathcal{P}_x u^n = \mathcal{Q}_x u^{n-1}$ $\mathcal{P}_x u^n = u^n - \frac{1}{2}\Delta t \mathcal{L}_x u^n$ $\mathcal{Q}_x u^{n-1} = u^{n-1} + \frac{1}{2}\Delta t \mathcal{L}_x u^{n-1}$

because the covariance functions $k_{u,u}^{n,n}$ and $k_{u,u}^{n,n-1}$ result from a numerical scheme, in this case the Euler method. Essentially, this introduces a structured prior that explicitly encodes the physical law modeled by the partial differential equation (1). In the following, we will generalize the framework outlined above to arbitrary *linear multistep methods*, originally proposed by Bashforth and Adams [5], as well as *Runge–Kutta methods*, generally attributed to Runge [28]. The biggest challenge here is the proper placement of the Gaussian process prior (see, e.g., (3)) in order to avoid inversion of differential operators and to bypass the classical need for spatial discretization of such operators. For instance, in the above example (see (2) and (3)), it would have been an inappropriate choice to start by placing a Gaussian process prior on u^n rather than on u^{n-1}, as obtaining the *numerical Gaussian process* (4) would then involve inverting operators of the form $I + \Delta t \mathcal{L}_x$ corresponding to the Euler method. Moreover, propagating the uncertainty associated with the noisy initial observations $\{\boldsymbol{x}^0, \boldsymbol{u}^0\}$ through time is another major challenge addressed in what follows.

2. Linear multistep methods. Let us start with the most general form of the linear multistep methods [7] applied to (1), i.e.,

$$u^n = \sum_{i=1}^{m} \alpha_i u^{n-i} + \Delta t \sum_{i=0}^{m} \beta_i \mathcal{L}_x u^{n-i}. \tag{5}$$

Different choices for the parameters α_i and β_i result in specific schemes. For instance, in Table 1, we present some specific members of the family of linear multistep methods (5). We encourage the reader to keep these special cases in mind while reading the rest of this section. Linear multistep methods (5) can be equivalently written as

$$\mathcal{P}_x u^n = \sum_{i=1}^{m} \mathcal{Q}_x^i u^{n-i}, \tag{6}$$

where $\mathcal{P}_x u := u - \Delta t \beta_0 \mathcal{L}_x u$ and $\mathcal{Q}_x^i u := \alpha_i u + \Delta t \beta_i \mathcal{L}_x u$. Some special cases of (6) are given in Table 2. For every $j = 0, 1, \ldots, m$ and some $\tau \in [0, 1]$ which depends on the specific choices for the values of the parameters α_i and β_i, we define $u^{n-j+\tau}$ to

TABLE 3
Some special cases of (7).

Forward Euler	$u^n = \mathcal{Q}_x u^{n-1}$ $\tau = 0$
Backward Euler	$\mathcal{P}_x u^n = u^{n-1}$ $\tau = 1$
Trapezoidal rule	$\mathcal{P}_x u^n = u^{n-1/2} = \mathcal{Q}_x u^{n-1}$ $\tau = 1/2$

be given by

$$\mathcal{P}_x u^{n-j+1} =: u^{n-j+\tau} := \sum_{i=1}^{m} \mathcal{Q}_x^i u^{n-i-j+1}. \tag{7}$$

Definition (7) takes the specific forms given in Table 3 for some example schemes. Shifting every term involved in the above definition (7) by $-\tau$ yields

$$\mathcal{P}_x u^{n-j+1-\tau} = u^{n-j} = \sum_{i=1}^{m} \mathcal{Q}_x^i u^{n-i-j+1-\tau}. \tag{8}$$

To give an example, for the trapezoidal rule we obtain $\mathcal{P}_x u^{n+1/2} = u^n = \mathcal{Q}_x u^{n-1/2}$ and $\mathcal{P}_x u^{n-1/2} = u^{n-1} = \mathcal{Q}_x u^{n-3/2}$. Therefore, as a direct consequence of (8) we have

$$\begin{aligned} u^n &= \sum_{i=1}^{m} \mathcal{Q}_x^i u^{n-i+1-\tau} \quad \text{when } j = 0, \\ u^{n-j} &= \mathcal{P}_x u^{n-j+1-\tau} \qquad \text{when } j = 1, \dots, m. \end{aligned} \tag{9}$$

This, in the special case of the trapezoidal rule, translates to $u^n = \mathcal{Q}_x u^{n-1/2}$ and $u^{n-1} = \mathcal{P}_x u^{n-1/2}$. It is worth noting that by assuming $u^{n-1/2}(x) \sim \mathcal{GP}(0, k(x, x'; \theta))$, we can capture the entire structure of the trapezoidal rule in the resulting joint distribution of u^n and u^{n-1}. This proper placement of the Gaussian process prior is key to the proposed methodology, as it allows us to avoid any spatial discretization of differential operators since no inversion of such operators is necessary. We will capitalize on this idea in the following.

2.1. Prior. Assuming that

$$u^{n-j+1-\tau}(x) \sim \mathcal{GP}(0, k^{j,j}(x, x'; \theta_j)), \quad j = 1, \dots, m, \tag{10}$$

are m independent processes, we obtain the following *numerical Gaussian process:*

$$\begin{bmatrix} u^n \\ \vdots \\ u^{n-m} \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k_{u,u}^{n,n} & \cdots & k_{u,u}^{n,n-m} \\ & \ddots & \vdots \\ & & k_{u,u}^{n-m,n-m} \end{bmatrix}\right),$$

where

$$\begin{aligned} k_{u,u}^{n,n} &= \sum_{i=1}^{m} \mathcal{Q}_x^i \mathcal{Q}_{x'}^i k^{i,i}, & k_{u,u}^{n,n-j} &= \mathcal{Q}_x^j \mathcal{P}_{x'} k^{j,j}, \\ k_{u,u}^{n-i,n-j} &= 0, \quad i \neq j, & k_{u,u}^{n-j,n-j} &= \mathcal{P}_x \mathcal{P}_{x'} k^{j,j}, \quad j = 1, \dots, m. \end{aligned} \tag{11}$$

It is worth noting that the entire structure of linear multistep methods (5) is captured by the kernels given in (11). Note that although we start from an independence assumption in (10), the resulting *numerical Gaussian process* exhibits a fully correlated structure, as illustrated in equations (11). Moreover, the information on the boundary $\partial\Omega$ of the domain Ω can often be summarized by noisy observations $\{\boldsymbol{x}_b^n, \boldsymbol{u}_b^n\}$ of a linear transformation $\mathcal{B}_x$ of u^n, i.e., noisy data on

$$u_b^n := \mathcal{B}_x u^n.$$

Using this, we obtain the following covariance functions involving the boundary:

$$k_{b,u}^{n,n} = \mathcal{B}_x k_{u,u}^{n,n}, \quad k_{b,b}^{n,n} = \mathcal{B}_x \mathcal{B}_{x'} k_{u,u}^{n,n}, \quad k_{b,u}^{n,n-j} = \mathcal{B}_x k_{u,u}^{n,n-j}, \qquad j = 1, \ldots, m.$$

The numerical examples in the supplementary material accompanying this manuscript are designed to showcase different special treatments of boundary conditions, including Dirichlet, Neumann, mixed, and periodic boundary conditions.

2.2. Work flow and computational cost. The proposed work flow is summarized below:

1. Starting from the initial data $\{\boldsymbol{x}^0, \boldsymbol{u}^0\}$ and the boundary data $\{\boldsymbol{x}_b^1, \boldsymbol{u}_b^1\}$, we train the kernel hyper-parameters as outlined in section 2.3. This step carries the main computational burden, as it scales cubically with the total number of training points since it involves Cholesky factorization of full symmetric positive-definite covariance matrices [26].
2. Having identified the optimal set of kernel hyper-parameters, we utilize the conditional posterior distribution to predict the solution at the next time-step and generate the *artificial data* $\{\boldsymbol{x}^1, \boldsymbol{u}^1\}$. Note that $\boldsymbol{x}^1$ is randomly sampled in the spatial domain according to a uniform distribution, and $\boldsymbol{u}^1$ is a normally distributed random vector, as outlined in section 2.4.
3. Given the *artificial data* $\{\boldsymbol{x}^1, \boldsymbol{u}^1\}$ and boundary data $\{\boldsymbol{x}_b^2, \boldsymbol{u}_b^2\}$, we proceed with training the kernel hyper-parameters for the second time-step[2] (see section 2.3).
4. Having identified the optimal set of kernel hyper-parameters, we utilize the conditional posterior distribution to predict the solution at the next time-step and generate the *artificial data* $\{\boldsymbol{x}^2, \boldsymbol{u}^2\}$, where $\boldsymbol{x}^2$ is randomly sampled in the spatial domain according to a uniform distribution. However, since $\boldsymbol{u}^1$ is a random vector, we have to marginalize it out in order to obtain consistent uncertainty estimates for $\boldsymbol{u}^2$. This procedure is outlined in section 2.5.
5. Steps 3 and 4 are repeated until the final integration time is reached.

In summary, the proposed methodology boils down to a sequence of Gaussian process regressions at every time-step. To accelerate training, one can use the optimal set of hyper-parameters from the previous time-step as an initial guess for the current one.

2.3. Training. In the following, for notational convenience and without loss of generality,[3] we will operate under the assumption that $m = 1$ (see (5)). The hyper-parameters θ_i, $i = 1, \ldots, m$, can be trained by employing the negative log marginal

[2] To be precise, we are using the mean of the random vector $\boldsymbol{u}^1$ for training purposes.

[3] The reader should be able to figure out the details without much difficulty while generalizing to cases with $m > 1$. Moreover, for the examples in the supplementary material accompanying this manuscript, more details are also provided in the supplementary material (M112076_01.pdf [local/web 126KB]).

likelihood resulting from

$$\begin{bmatrix} \boldsymbol{u}_b^n \\ \boldsymbol{u}^{n-1} \end{bmatrix} \sim \mathcal{N}(0, \boldsymbol{K}), \tag{12}$$

where $\{\boldsymbol{x}_b^n, \boldsymbol{u}_b^n\}$ are the (noisy) data on the boundary, $\{\boldsymbol{x}^{n-1}, \boldsymbol{u}^{n-1}\}$ are *artificially generated data* to be explained later (see (14)), and

$$\boldsymbol{K} := \begin{bmatrix} k_{b,b}^{n,n}(\boldsymbol{x}_b^n, \boldsymbol{x}_b^n) + \sigma_n^2 I & k_{b,u}^{n,n-1}(\boldsymbol{x}_b^n, \boldsymbol{x}^{n-1}) \\ & k_{u,u}^{n-1,n-1}(\boldsymbol{x}^{n-1}, \boldsymbol{x}^{n-1}) + \sigma_{n-1}^2 I \end{bmatrix}.$$

It is worth mentioning that the marginal likelihood provides a natural regularization mechanism that balances the trade-off between data fit and model complexity. This effect is known as Occam's razor [27] after William of Occam (1285–1349), who encouraged simplicity in explanations by the principle: "Plurality should not be assumed without necessity." Although these properties are very attractive, they do come at a cost. The cost is primarily computational, and it is associated with computing the inverse and the determinant of the covariance matrices appearing in the computation of the negative log marginal likelihood. These matrices are symmetric, positive-definite, and typically dense. The computation of their inverse and determinant is commonly done using the Cholesky decomposition, which has an unfavorable cubic scaling with the total number of training points. A large body of recent literature has been devoted to overcoming this scalability shortcoming, and some prevalent approaches can be found in [13, 23, 32].

2.4. Posterior. In order to predict $u^n(x_*^n)$ at a new test point x_*^n, we use the following conditional distribution:

$$u^n(x_*^n) \left| \begin{bmatrix} \boldsymbol{u}_b^n \\ \boldsymbol{u}^{n-1} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{q}^T \boldsymbol{K}^{-1} \begin{bmatrix} \boldsymbol{u}_b^n \\ \boldsymbol{u}^{n-1} \end{bmatrix}, k_{u,u}^{n,n}(x_*^n, x_*^n) - \boldsymbol{q}^T \boldsymbol{K}^{-1} \boldsymbol{q} \right),$$

where

$$\boldsymbol{q}^T := \begin{bmatrix} k_{u,b}^{n,n}(x_*^n, \boldsymbol{x}_b^n) & k_{u,u}^{n,n-1}(x_*^n, \boldsymbol{x}^{n-1}) \end{bmatrix}.$$

2.5. Propagating uncertainty. However, to properly propagate the uncertainty associated with the initial data through time, one should not stop here. Since $\{\boldsymbol{x}^{n-1}, \boldsymbol{u}^{n-1}\}$ are *artificially generated data* (see (14)), we have to marginalize them out by employing

$$\boldsymbol{u}^{n-1} \sim \mathcal{N}\left(\boldsymbol{\mu}^{n-1}, \boldsymbol{\Sigma}^{n-1,n-1}\right)$$

to obtain

$$u^n(x_*^n) \mid \boldsymbol{u}_b^n \sim \mathcal{N}\left(\mu^n(x_*^n), \Sigma^{n,n}(x_*^n, x_*^n)\right), \tag{13}$$

where

$$\mu^n(x_*^n) = \boldsymbol{q}^T \boldsymbol{K}^{-1} \begin{bmatrix} \boldsymbol{u}_b^n \\ \boldsymbol{\mu}^{n-1} \end{bmatrix}$$

and

$$\Sigma^{n,n}(x_*^n, x_*^n) = k_{u,u}^{n,n}(x_*^n, x_*^n) - \boldsymbol{q}^T \boldsymbol{K}^{-1} \boldsymbol{q} + \boldsymbol{q}^T \boldsymbol{K}^{-1} \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{\Sigma}^{n-1,n-1} \end{bmatrix} \boldsymbol{K}^{-1} \boldsymbol{q}.$$

Now, one can use the resulting posterior distribution (13) to obtain the artificially generated data $\{\boldsymbol{x}^n, \boldsymbol{u}^n\}$ for the next time-step with

$$\boldsymbol{u}^n \sim \mathcal{N}\left(\boldsymbol{\mu}^n, \boldsymbol{\Sigma}^{n,n}\right). \tag{14}$$

Here, $\boldsymbol{\mu}^n = \mu^n(\boldsymbol{x}^n)$ and $\boldsymbol{\Sigma}^{n,n} = \Sigma^{n,n}(\boldsymbol{x}^n, \boldsymbol{x}^n)$. Throughout this work the location of the artificially generated data is chosen using a uniform sampling strategy. This simple choice is mainly motivated by the fact that it allows us to visit as many locations as possible in our spatial domain while we march in time. We should also note here that the method is still applicable if the spatial locations of the artificial points are fixed in time. More interestingly, one could construct more sophisticated sampling strategies to enable adaptive refinement, for example by tracking the curvature of the solution. This direction will be further investigated in future work.

2.6. Example: Burgers' equation (backward Euler). Burgers' equation is a fundamental partial differential equation arising in various areas of applied mathematics, including fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow [4]. In one space dimension the equation reads as

$$u_t + uu_x = \nu u_{xx}, \tag{15}$$

along with Dirichlet boundary conditions $u(t,-1) = u(t,1) = 0$, where $u(t,x)$ denotes the unknown solution and ν is a viscosity parameter. Let us assume that all we observe are noisy measurements $\{\boldsymbol{x}^0, \boldsymbol{u}^0\}$ of the *black-box* initial function $u(0,x) = -\sin(\pi x)$. Given such measurements, we would like to solve the Burgers' equation (15) while propagating through time the uncertainty associated with the noisy initial data (see Figure 1). This example is important because it involves solving a nonlinear partial differential equation.

To illustrate how one can encode the structure of the physical laws expressed by Burgers' equation in a *numerical Gaussian process* let us apply the backward Euler scheme to (15). This can be written as

$$u^n = u^{n-1} - \Delta t u^n \frac{d}{dx} u^n + \nu \Delta t \frac{d^2}{dx^2} u^n. \tag{16}$$

We would like to place a Gaussian process prior on u^n. However, the nonlinear term $u^n \frac{d}{dx} u^n$ causes problems simply because the product of two Gaussian processes is no longer Gaussian. Hence, we will approximate the nonlinear term with $\mu^{n-1} \frac{d}{dx} u^n$, where μ^{n-1} is the posterior mean of the previous time-step. Therefore, the backward Euler scheme (16) can be approximated by

$$u^n = u^{n-1} - \Delta t \mu^{n-1} \frac{d}{dx} u^n + \nu \Delta t \frac{d^2}{dx^2} u^n. \tag{17}$$

Rearranging the terms, we obtain

$$u^n + \Delta t \mu^{n-1} \frac{d}{dx} u^n - \nu \Delta t \frac{d^2}{dx^2} u^n = u^{n-1}. \tag{18}$$

2.6.1. Numerical Gaussian process. Let us make the prior assumption that

$$u^n(x) \sim \mathcal{GP}(0, k(x, x'; \theta)) \tag{19}$$

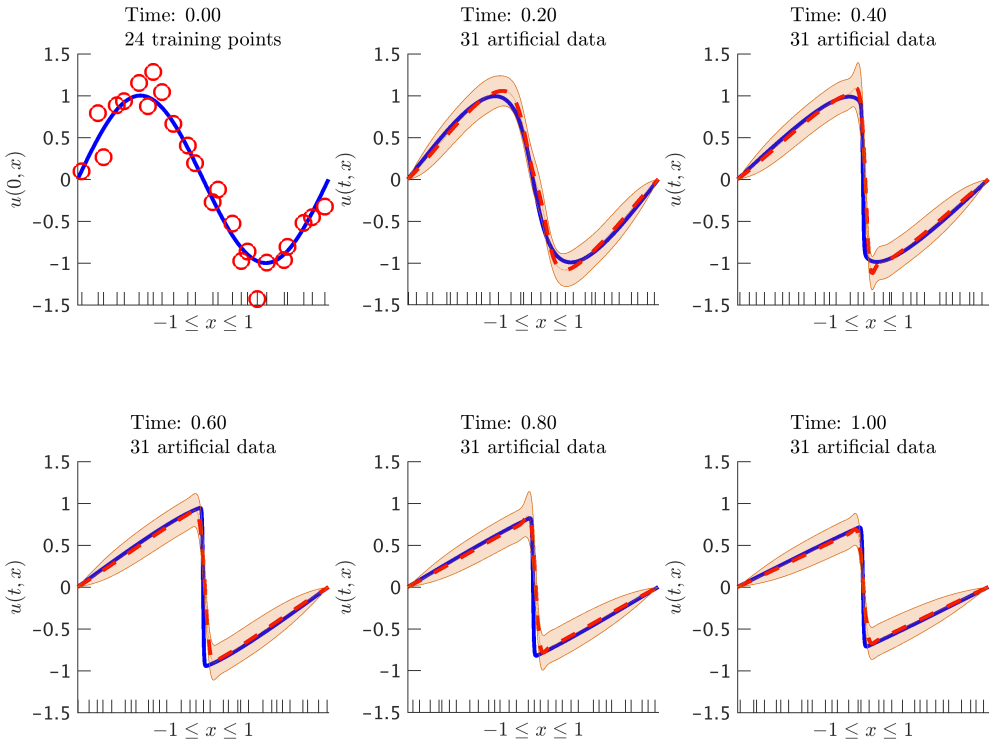

FIG. 1. Burgers' equation: *Initial data along with the posterior distribution of the solution at different time snapshots. The solid (blue) line represents the true data generating solution, while the dashed (red) line depicts the posterior mean. The shaded (orange) region illustrates the two-standard deviations band around the mean. We are employing the backward Euler scheme with time-step size* $\Delta t = 0.01$. *At each time-step we generate* 31 *artificial data points randomly located in the interval* $[-1, 1]$ *according to a uniform distribution. These locations are denoted by the ticks along the horizontal axis. Here, we set* $\nu = 0.01/\pi$—*a value leading to the development of a nonsingular thin internal layer at* $x = 0$ *that is notoriously hard to resolve by classical numerical methods* [4]. *(See the supplementary code:* http://bit.ly/2mnUiKT *and movie:* http://bit.ly/2m1sKHw.*)*

is a Gaussian process with a neural network [26] covariance function

$$k(x, x'; \theta) = \frac{2}{\pi} \sin^{-1}\left(\frac{2(\sigma_0^2 + \sigma^2 x x')}{\sqrt{(1 + 2\,(\sigma_0^2 + \sigma^2 x^2))\,(1 + 2\,(\sigma_0^2 + \sigma^2 x'^2))}}\right), \tag{20}$$

where $\theta = (\sigma_0^2, \sigma^2)$ denotes the hyper-parameters. Here we have chosen a nonstationary prior motivated by the fact that the solution to the Burgers' equation can develop discontinuities for small values of the viscosity parameter ν. This enables us to obtain the following *numerical Gaussian process:*

$$\begin{bmatrix} u^n \\ u^{n-1} \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k_{u,u}^{n,n} & k_{u,u}^{n,n-1} \\ & k_{u,u}^{n-1,n-1} \end{bmatrix}\right),$$

where the covariance functions $k_{u,u}^{n,n}$, $k_{u,u}^{n,n-1}$, and $k_{u,u}^{n-1,n-1}$ can be derived as

$$k_{u,u}^{n,n-1} = k + \Delta t \mu^{n-1}(x') \frac{d}{dx'} k - \nu \Delta t \frac{d^2}{dx'^2} k \tag{21}$$

and

$$
\begin{aligned}
k_{u,u}^{n-1,n-1} = k &+ \Delta t \mu^{n-1}(x') \frac{d}{dx'} k - \nu \Delta t \frac{d^2}{dx'^2} k \\
&+ \Delta t \mu^{n-1}(x) \frac{d}{dx} k + \Delta t^2 \mu^{n-1}(x) \mu^{n-1}(x') \frac{d}{dx} \frac{d}{dx'} k \\
&- \nu \Delta t^2 \mu^{n-1}(x) \frac{d}{dx} \frac{d^2}{dx'^2} k - \nu \Delta t \frac{d^2}{dx^2} k \\
&- \nu \Delta t^2 \mu^{n-1}(x') \frac{d^2}{dx^2} \frac{d}{dx'} k + \nu^2 \Delta t^2 \frac{d^2}{dx^2} \frac{d^2}{dx'^2} k.
\end{aligned} \tag{22}
$$

The only nontrivial operations in the aforementioned kernel computations are those involving derivatives of the kernels, which can be performed using any mathematical symbolic computation program, like Wolfram Mathematica. Training, prediction, and propagating the uncertainty associated with the noisy initial observations can be performed as in sections 2.3, 2.4, and 2.5, respectively. Figure 1 depicts the noisy initial data along with the posterior distribution of the solution to the Burgers' equation (15) at different time snapshots. It is remarkable that the proposed methodology can effectively propagate an infinite collection of correlated Gaussian random variables (i.e., a Gaussian process) through the complex nonlinear dynamics of the Burgers' equation.

2.6.2. Numerical study. It must be re-emphasized that *numerical Gaussian processes*, by construction, are designed to deal with cases where (a) all we observe is noisy data on *black-box* initial conditions, and (b) we are interested in *quantifying the uncertainty* associated with such noisy data in our solutions to time-dependent partial differential equations. In fact, we recommend resorting to other alternative classical numerical methods such as finite differences, finite elements, and spectral methods in cases where (a) the initial function is *not* a black-box function and we have access to *noiseless* data, or (b) we are *not* interested in quantifying the uncertainty in our solutions. However, in order to be able to perform a systematic numerical study of the proposed methodology, and despite the fact that this defeats the whole purpose of the current work, sometimes we will operate under the assumption that we have access to *noiseless* initial data. For instance, concerning the Burgers' equation, if we had access to such noiseless data, we would obtain results similar to those depicted in Figure 2.

Moreover, in order to make sure that the *numerical Gaussian process* resulting from the backward Euler scheme (18) applied to the Burgers' equation is indeed first-order accurate in time, we perform the numerical experiments depicted in Figures 3 and 4. Specifically, in Figure 3 we report the time-evolution of the relative spatial $\mathcal{L}^2$-error until the final integration time $T = 1.0$ is reached. We observe that the error indeed grows as $\mathcal{O}(\Delta t)$, and its resulting behavior reveals both the shock development region and the energy dissipation due to diffusion at later times. Moreover, in Figure 4 we fix the final integration time to $T = 0.1$ and the number of initial and artificial data to 50, and vary the time-step size Δt from 10^{-1} to 10^{-4}. As expected, we recover the first-order convergence properties of the backward Euler scheme, except for a saturation region arising when we further reduce the time-step size below approximately 10^{-3}. This behavior is not a result of the time-stepping scheme but is attributed to the underlying Gaussian process regression and the finite number of spatial data points used for training and prediction. To investigate the accuracy of the posterior mean in predicting the solution as the number of training points is increased, we per-

form the numerical experiment depicted in Figure 5. Here we have considered two cases for which we fix the time-step size to $\Delta t = 10^{-2}$ and $\Delta t = 10^{-3}$, respectively, and increase the number of initial as well as artificial data points. A similar accuracy saturation is also observed here as the number of training points is increased. In this case, it is attributed to the error accumulated due to time-stepping with the relatively large time-step sizes for the first-order accurate Euler scheme. If we keep decreasing the time-step further, this saturation behavior will occur for higher numbers of total training points.

The key point here is that, although Gaussian processes can yield satisfactory accuracy, they, by construction, cannot force the approximation error down to machine precision. This is due to the fact that Gaussian processes are suitable for solving regression problems. This is exactly the reason why we recommend other alternative classical numerical methods for solving partial differential equations in cases where one has access to noiseless data. In such cases, it is desirable to use numerical schemes that are capable of performing exact interpolation on the data rather than merely a regression.

2.7. Example: Wave equation (trapezoidal rule). The wave equation is an important second-order linear partial differential equation for the description of wave propagation phenomena, including sound waves, light waves, and water waves. It arises in many scientific fields such as acoustics, electromagnetics, and fluid dynamics. In one space dimension the wave equation reads as

$$u_{tt} = u_{xx}. \tag{23}$$

The function $u(t,x) = \frac{1}{2}\sin(\pi x)\cos(\pi t) + \frac{1}{3}\sin(3\pi x)\sin(3\pi t)$ solves this equation and satisfies the following initial and homogeneous Dirichlet boundary conditions:

$$\begin{aligned} u(0,x) &= u^0(x) := \frac{1}{2}\sin(\pi x), \\ u_t(0,x) &= v^0(x) := \pi \sin(3\pi x), \\ u(t,0) &= u(t,1) = 0. \end{aligned} \tag{24}$$

Now, let us assume that all we observe are noisy measurements $\{\boldsymbol{x}_u^0, \boldsymbol{u}^0\}$ and $\{\boldsymbol{x}_v^0, \boldsymbol{v}^0\}$ of the *black-box* initial functions u^0 and v^0, respectively. Given this data, we are interested in solving the wave equation (23) and quantifying the uncertainty in our solution associated with the noisy initial data (see Figure 6). To proceed, let us define $v := u_t$ and rewrite the wave equation as a system of equations given by

$$\begin{cases} u_t = v, \\ v_t = u_{xx}. \end{cases} \tag{25}$$

This example is important because it involves solving a *system* of partial differential equations. One could rewrite the system of equations (25) in matrix-vector notation and obtain

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ v \end{bmatrix} = \mathcal{L}_x \begin{bmatrix} u \\ v \end{bmatrix},$$

which takes the form of (1) with

$$\mathcal{L}_x = \begin{bmatrix} 0 & I \\ \frac{\partial^2}{\partial x^2} & 0 \end{bmatrix}.$$

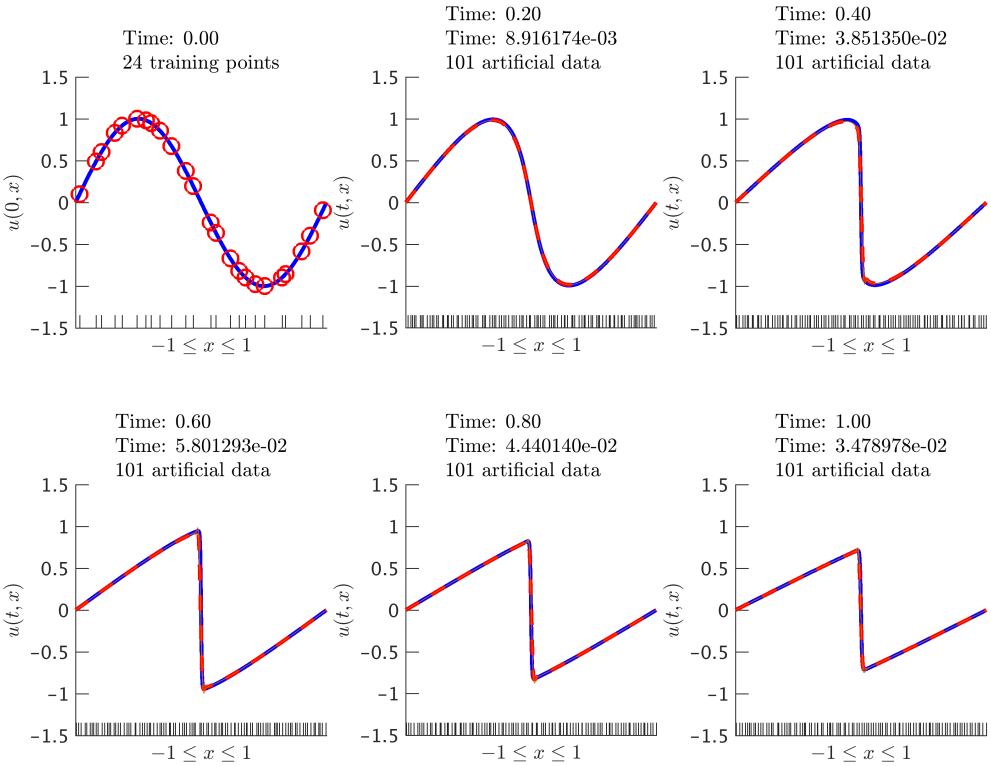

FIG. 2. Burgers' equation: *Initial data along with the posterior distribution of the solution at different time snapshots. The solid (blue) line represents the true data generating solution, while the dashed (red) line depicts the posterior mean. The shaded (orange) region illustrates the two-standard deviations band around the mean. We are employing the backward Euler scheme with time-step size* $\Delta t = 0.01$. *At each time-step we generate* 101 *artificial data points randomly located in the interval* $[-1, 1]$ *according to a uniform distribution. These locations are denoted by the ticks along the horizontal axis. Here, we set* $\nu = 0.01/\pi$—*a value leading to the development of a nonsingular thin internal layer at* $x = 0$ *that is notoriously hard to resolve by classical numerical methods* [4]. *We report here the relative* $\mathcal{L}^2$*-error between the posterior mean and the true solution. (See the supplementary code:* http://bit.ly/2mDKCwb*; Movie:* http://bit.ly/2mDOPA5.*)*

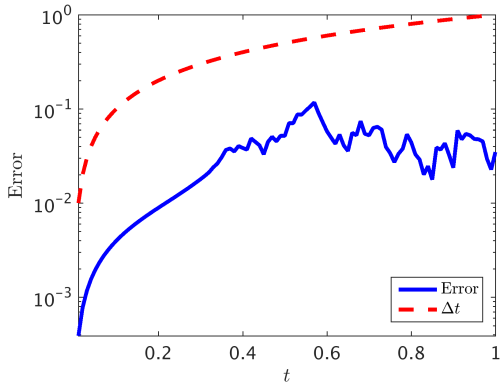

FIG. 3. Burgers' equation: *Time evolution of the relative spatial* $\mathcal{L}^2$*-error up to the final integration time* $T = 1.0$. *We are using the backward Euler scheme with a time-step size of* $\Delta t = 0.01$, *and the dashed (red) line illustrates the optimal first-order convergence rate. (See the supplementary code:* http://bit.ly/2mDY6It.*)*

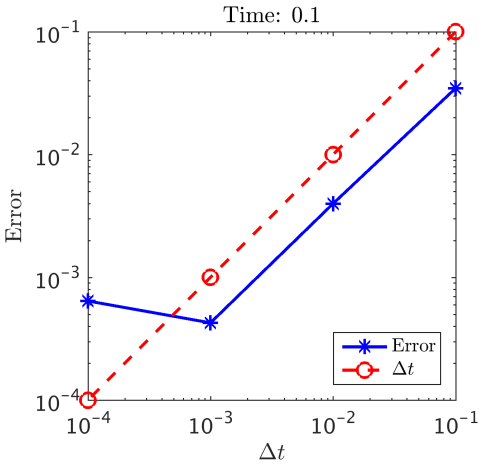

FIG. 4. Burgers' equation: *Relative spatial $\mathcal{L}^2$-error versus step-size for the backward Euler scheme at time $T = 0.1$. The number of noiseless initial and artificially generated data is set to be equal to 50. (See the supplementary code:* http://bit.ly/2mDY6It.*)*

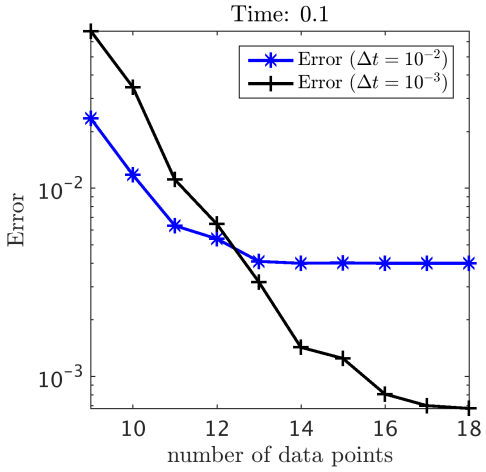

FIG. 5. Burgers' equation: *Relative spatial $\mathcal{L}^2$-error versus the number of noiseless initial as well as artificial data points used for the backward Euler scheme with time-step sizes of $\Delta t = 10^{-2}$ and $\Delta t = 10^{-3}$. (See the supplementary code:* http://bit.ly/2mDY6It.*)*

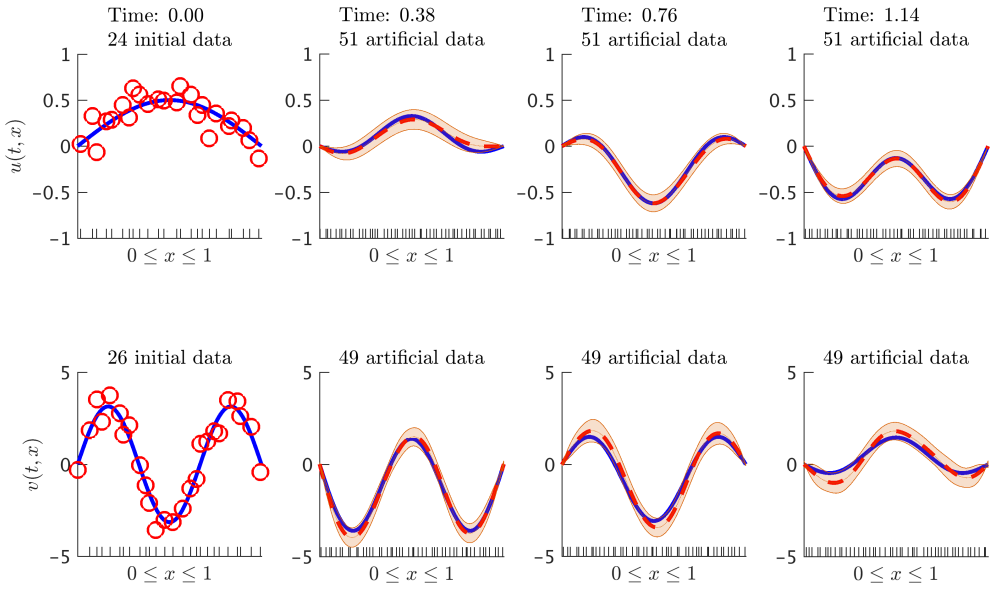

FIG. 6. Wave equation: *Initial data along with the posterior distribution of the solution at different time snapshots. Here, $v(t,x) = u_t(t,x)$. The solid (blue) line represents the true data generating solution, while the dashed (red) line depicts the posterior mean. The shaded (orange) region illustrates the two-standard deviations band around the mean. At each time-step we generate* 51 *artificial data points for u and* 49 *for v, all randomly located in the interval $[0,1]$ according to a uniform distribution. These locations are denoted by the ticks along the horizontal axis. We are employing the trapezoidal scheme with time-step size $\Delta t = 0.01$. (See the supplementary code:* http://bit.ly/2m3mfnA*; Movie:* http://bit.ly/2mpfhNi.*)*

This form is now amenable to the previous analysis provided for general linear multistep methods. However, for pedagogical purposes, let us slowly walk through the trapezoidal rule and apply it to the system of equations (25). This can be written as

$$u^n = u^{n-1} + \frac{1}{2}\Delta t v^{n-1} + \frac{1}{2}\Delta t v^n, \tag{26}$$
$$v^n = v^{n-1} + \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^{n-1} + \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^n.$$

Rearranging the terms yields

$$u^n - \frac{1}{2}\Delta t v^n = u^{n-1} + \frac{1}{2}\Delta t v^{n-1}, \qquad v^n - \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^n = v^{n-1} + \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^{n-1}.$$

Now, let us define $u^{n-1/2}$ and $v^{n-1/2}$ to be given by

$$u^n - \frac{1}{2}\Delta t v^n =: u^{n-1/2} := u^{n-1} + \frac{1}{2}\Delta t v^{n-1}, \tag{27}$$
$$v^n - \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^n =: v^{n-1/2} := v^{n-1} + \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^{n-1}.$$

As outlined in section 2, this is a key step in the proposed methodology, as it hints at the proper location at which to place the Gaussian process prior. Shifting the terms involved in the above equations by $-1/2$ and $+1/2$, we obtain

$$u^{n-1/2} - \frac{1}{2}\Delta t v^{n-1/2} = u^{n-1}, \qquad v^{n-1/2} - \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^{n-1/2} = v^{n-1}, \tag{28}$$

and

$$u^n = u^{n-1/2} + \frac{1}{2}\Delta t v^{n-1/2}, \qquad v^n = v^{n-1/2} + \frac{1}{2}\Delta t \frac{d^2}{dx^2}u^{n-1/2}, \tag{29}$$

respectively. Now we can proceed with encoding the structure of the wave equation into a *numerical Gaussian process* prior for performing Bayesian machine learning of the solution $\{u(t,x), v(t,x)\}$ at any $t > 0$.

2.7.1. Numerical Gaussian process. Let us make the prior assumption that

$$u^{n-1/2}(x) \sim \mathcal{GP}(0, k_u(x,x';\theta_u)), \qquad v^{n-1/2}(x) \sim \mathcal{GP}(0, k_v(x,x';\theta_v)), \tag{30}$$

are two independent Gaussian processes with squared exponential [26] covariance functions

$$k_u(x,x';\theta_u) = \gamma_u^2 \exp\left(-\frac{1}{2}w_u(x-x')^2\right), \tag{31}$$
$$k_v(x,x';\theta_v) = \gamma_v^2 \exp\left(-\frac{1}{2}w_v(x-x')^2\right),$$

where $\theta_u = \left(\gamma_u^2, w_u\right)$ and $\theta_v = \left(\gamma_v^2, w_v\right)$. From a theoretical point of view, each covariance function gives rise to a reproducing kernel Hilbert space [3, 6, 29] 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

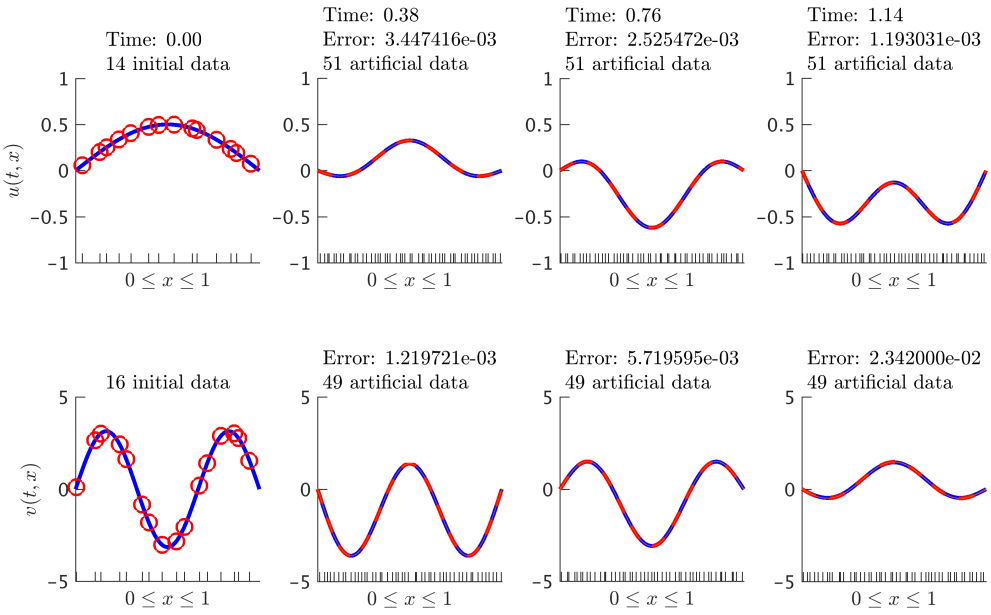

FIG. 7. Wave equation: *Initial data along with the posterior distribution of the solution at different time snapshots. Here,* $v(t,x) = u_t(t,x)$. *The solid (blue) line represents the true data generating solution, while the dashed (red) line depicts the posterior mean. The shaded (orange) region illustrates the two-standard deviations band around the mean. At each time-step we generate* 51 *artificial data points for* u *and* 49 *for* v, *all randomly located in the interval* $[0,1]$ *according to a uniform distribution. These locations are denoted by the ticks along the horizontal axis. We are employing the trapezoidal scheme with time-step size* $\Delta t = 0.01$. *We are reporting the relative* $\mathcal{L}^2$*-error between the posterior mean and the true solution. (See the supplementary code:* http://bit.ly/2m3mKhK*; Movie:* http://bit.ly/2mFalVg.*)*

(see, e.g., (20)). This enables us to obtain the following *numerical Gaussian process:*

$$\begin{bmatrix} u^n \\ v^n \\ u^{n-1} \\ v^{n-1} \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k_{u,u}^{n,n} & k_{u,v}^{n,n} & k_{u,u}^{n,n-1} & k_{u,v}^{n,n-1} \\ & k_{v,v}^{n,n} & k_{v,u}^{n,n-1} & k_{v,v}^{n,n-1} \\ & & k_{u,u}^{n-1,n-1} & k_{u,v}^{n-1,n-1} \\ & & & k_{v,v}^{n-1,n-1} \end{bmatrix}\right),$$

which captures the entire structure of the trapezoidal rule (26) when applied to the wave equation (23), in its covariance functions given in section SM1 of the supplementary material (M112076_01.pdf [local/web 126KB]). Training, prediction, and propagating the uncertainty associated with the noisy initial observations can be performed as in section SM1 of the supplementary material. Figure 6 depicts the noisy initial data along with the posterior distribution (SM4 of the supplementary material) of the solution to the wave equation (23) at different time snapshots.

2.7.2. Numerical study. In the case where we have access to noiseless initial data, we obtain the results depicted in Figure 7. Moreover, we perform a numerical study similar to the one reported in section 2.6.2. This is to verify that the *numerical Gaussian process* resulting from the trapezoidal rule (26) when applied to the wave equation is indeed second-order accurate in time. In particular, the numerical experiment shown in Figure 8 illustrates the time evolution of the relative spatial $\mathcal{L}^2$ up to the final integration time $T = 1.5$. The second-order convergence of the algo-

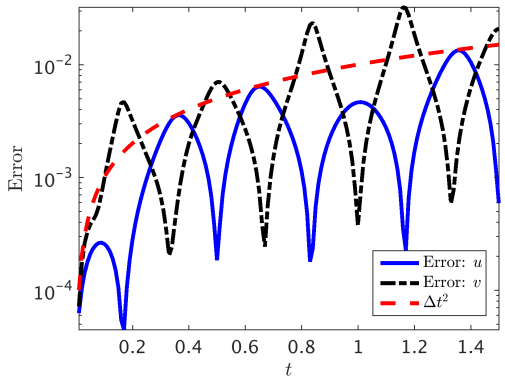

FIG. 8. Wave equation: *Time evolution of the relative spatial $\mathcal{L}^2$-error up to the final integration time $T = 1.5$. The solid (blue) line corresponds to the u component of the solution, while the dashed (black) line corresponds to the function v. We are using the trapezoidal rule with a time-step size of $\Delta t = 0.01$, and the dashed (red) line illustrates the optimal second-order convergence rate. (See the supplementary code:* http://bit.ly/2niW6lW*.)*

rithm is also demonstrated in Figure 9, where we have fixed the number of noiseless initial and artificially generated data while decreasing the time-step size. We also investigate the convergence behavior of the algorithm for a fixed time-step $\Delta t = 10^{-2}$ and as the number of training points is increased. The results are summarized in Figure 10. The analysis of both temporal and spatial convergence properties yields qualitatively similar conclusions to those reported in section 2.6.2. In Figures 9 and 10 (and similarly in all the other examples presented in this paper) the early signs of error saturation are visible. Indeed, increasing the total number of training points or decreasing Δt will eventually not affect the decrease of approximation error. This can be considered as a drawback of our method, and it is primarily attributed to spatial approximation errors. Specifically, Gaussian processes offer a regression framework and cannot practically decrease the approximation error to machine precision as the number of training points is increased.

3. Runge–Kutta methods. Let us now focus on the general form of Runge–Kutta methods [2] with q stages applied to (1); i.e.,

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{q} b_i \mathcal{L}_x u^{n+\tau_i}, \tag{32}$$

$$u^{n+\tau_i} = u^n + \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{L}_x u^{n+\tau_j}, \quad i = 1, \ldots, q.$$

Here, $u^{n+\tau_i}(x) = u(t^n + \tau_i \Delta t, x)$. This general form encapsulates both implicit and explicit time-stepping schemes, depending on the choice of the weights $\{a_{ij}, b_i\}$. An important feature of the proposed methodology is that it is oblivious to the choice of these parameters; hence the implicit or explicit nature of the time-stepping scheme is ultimately irrelevant. This is in sharp contrast to classical numerical methods in which implicit time-integration is burdensome due to the need for repeatedly solving linear or nonlinear systems. Here, for a fixed number of stages q, the cost of performing implicit or explicit time-marching is identical. This is attributed to the fact that the structure of the time-stepping scheme is encoded in the *numerical Gaussian process*

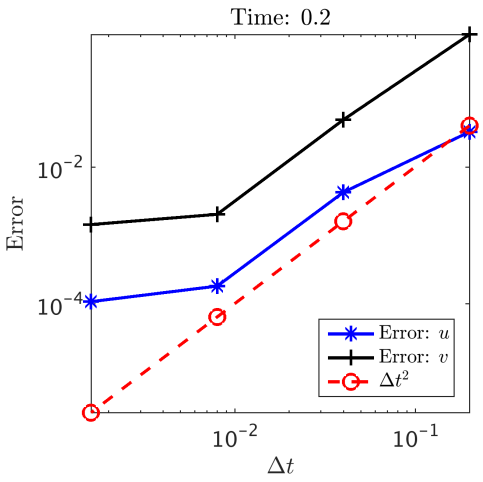

FIG. 9. Wave equation: *Relative spatial $\mathcal{L}^2$-error versus step-size for the trapezoidal rule. Here, the number of noiseless initial data as well as the artificially generated data is set to be equal to* 50. *We are running the time-stepping scheme up until time* 0.2 *is reached. (See the supplementary code:* http://bit.ly/2niW6lW.*)*

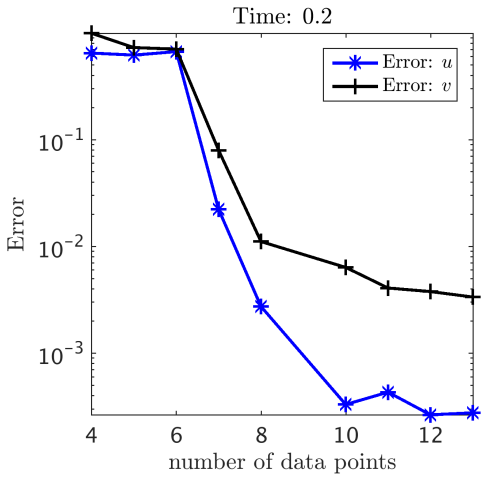

FIG. 10. Wave equation: *Relative spatial $\mathcal{L}^2$-error versus the number of noiseless initial as well as artificial data points used for the trapezoidal rule. Here, the time-step size is set to be* $\Delta t = 0.01$. *We are running the time-stepping scheme up until time* 0.2 *is reached. (See the supplementary code:* http://bit.ly/2niW6lW.*)*

prior, and the algorithm only involves solving a sequence of regression problems as outlined in section 2.2. This allows us to enjoy the favorable stability properties of fully implicit schemes at no extra cost and thus perform long-time integration using very large time-steps. Equations (32) can be equivalently written as

$$u^{n+1} - \Delta t \sum_{i=1}^{q} b_i \mathcal{L}_x u^{n+\tau_i} = u^n =: u^n_{q+1}, \tag{33}$$

$$u^{n+\tau_i} - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{L}_x u^{n+\tau_j} = u^n =: u^n_i, \quad i = 1, \dots, q.$$

Let us make the prior assumption that

$$u^{n+1}(x) \sim \mathcal{GP}(0, k^{n+1,n+1}_{u,u}(x, x'; \theta_{n+1})), \tag{34}$$

$$u^{n+\tau_i}(x) \sim \mathcal{GP}(0, k^{n+\tau_i,n+\tau_i}_{u,u}(x, x'; \theta_{n+\tau_i})), \quad i = 1, \dots, q,$$

are $q+1$ mutually independent Gaussian processes. Therefore, we can write the joint distribution of $u^{n+1}, u^{n+\tau_q}, \dots, u^{n+\tau_1}, u^n_{q+1}, \dots, u^n_1$ to capture the entire structure of the Runge–Kutta methods in the resulting *numerical Gaussian process*. However, rather than getting bogged down by heavy notation, and without sacrificing any generality, we will present the main ideas through the lens of an example.

3.1. Example: Advection equation (Gauss–Legendre method). We have chosen this classical pedagogical example as a prototype benchmark problem for testing the limits of long-time integration. This example also highlights the implementation of periodic constraints at the domain boundaries (36). The advection equation in one space dimension takes the form

$$u_t = -u_x. \tag{35}$$

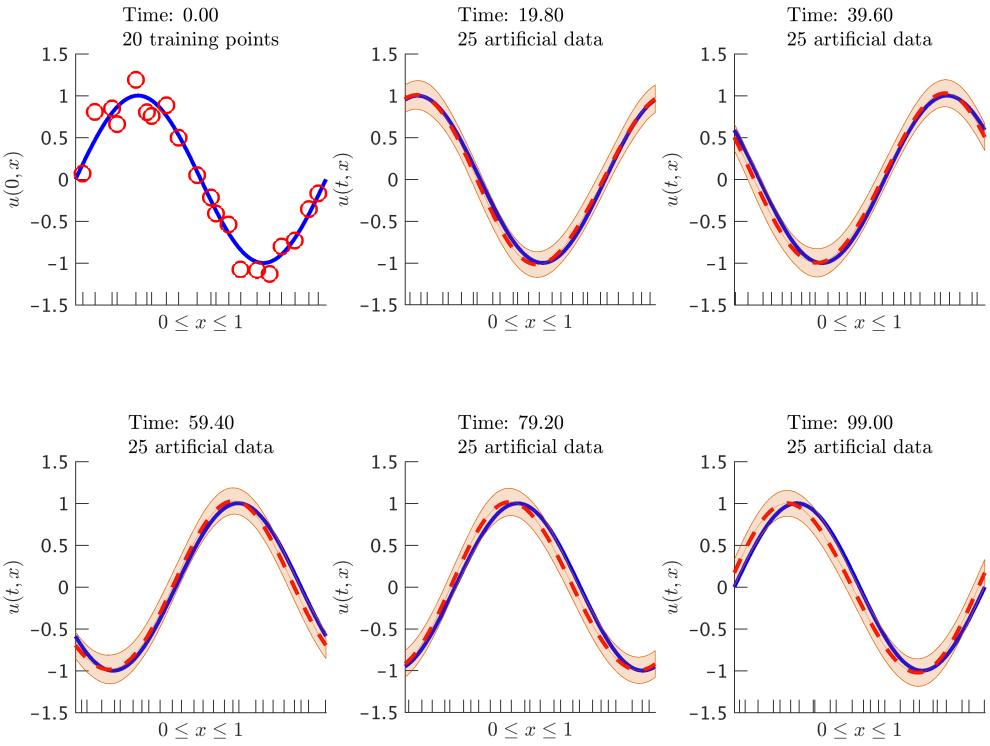

FIG. 11. Advection equation: *Initial data along with the posterior distribution of the solution at different time snapshots. The solid (blue) line represents the true data generating solution, while the dashed (red) line depicts the posterior mean. The shaded (orange) region illustrates the two-standard deviations band around the mean. At each time-step we generate* 25 *artificial data points randomly located in the interval* $[0, 1]$ *according to a uniform distribution. These locations are denoted by the ticks along the horizontal axis. We are employing the Gauss–Legendre time-stepping quadrature rule with time-step size* $\Delta t = 0.1$. *It is worth highlighting that we are running the time-stepping scheme for a very long time and with a relatively large time-step size. (See the supplementary code:* http://bit.ly/2m3JoXb*; Movie:* http://bit.ly/2mKHCP4.*)*

The function $u(t, x) = \sin(2\pi(x - t))$ solves this equation and satisfies the following initial and periodic boundary conditions:

$$\begin{aligned} u(0, x) &= u^0(x) := \sin(2\pi x), \\ u(t, 0) &= u(t, 1). \end{aligned} \tag{36}$$

However, let us assume that all we observe are noisy measurements $\{\boldsymbol{x}^0, \boldsymbol{u}^0\}$ of the *black-box* initial function u^0. Given this data, we are interested in encoding the structure of the advection operator in a *numerical Gaussian process* prior and using it to infer the solution $u(t, x)$ with quantified uncertainty for any $t > 0$ (see Figure 11). Let us apply the Gauss–Legendre time-stepping quadrature [14] with two stages (thus making it fourth-order accurate) to the advection equation (35). Referring to

equations (33), we obtain

$$
\begin{aligned}
u_3^n &:= u^n = u^{n+1} + b_1 \Delta t \frac{d}{dx} u^{n+\tau_1} + b_2 \Delta t \frac{d}{dx} u^{n+\tau_2}, \\
u_2^n &:= u^n = u^{n+\tau_2} + a_{21} \Delta t \frac{d}{dx} u^{n+\tau_1} + a_{22} \Delta t \frac{d}{dx} u^{n+\tau_2}, \\
u_1^n &:= u^n = u^{n+\tau_1} + a_{11} \Delta t \frac{d}{dx} u^{n+\tau_1} + a_{12} \Delta t \frac{d}{dx} u^{n+\tau_2}.
\end{aligned} \tag{37}
$$

Here, $\tau_1 = \frac{1}{2} - \frac{1}{6}\sqrt{3}$, $\tau_2 = \frac{1}{2} + \frac{1}{6}\sqrt{3}$, $b_1 = b_2 = \frac{1}{2}$, $a_{11} = a_{22} = \frac{1}{4}$, $a_{12} = \frac{1}{4} - \frac{1}{6}\sqrt{3}$, and $a_{21} = \frac{1}{4} + \frac{1}{6}\sqrt{3}$.

3.1.1. Prior. We make the prior assumption that

$$
\begin{aligned}
u^{n+1}(x) &\sim \mathcal{GP}(0, k_{u,u}^{n+1,n+1}(x, x'; \theta_{n+1})), \\
u^{n+\tau_2}(x) &\sim \mathcal{GP}(0, k_{u,u}^{n+\tau_2,n+\tau_2}(x, x'; \theta_{n+\tau_2})), \\
u^{n+\tau_1}(x) &\sim \mathcal{GP}(0, k_{u,u}^{n+\tau_1,n+\tau_1}(x, x'; \theta_{n+\tau_1}))
\end{aligned} \tag{38}
$$

are three independent Gaussian processes with squared exponential covariance functions similar to the kernels used in (31). This assumption yields the following *numerical Gaussian process:*

$$
\begin{bmatrix} u^{n+1} \\ u^{n+\tau_2} \\ u^{n+\tau_1} \\ u_3^n \\ u_2^n \\ u_1^n \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k_{u,u}^{n+1,n+1} & 0 & 0 & k_{u,3}^{n+1,n} & 0 & 0 \\ & k_{u,u}^{n+\tau_2,n+\tau_2} & 0 & k_{u,3}^{n+\tau_2,n} & k_{u,2}^{n+\tau_2,n} & k_{u,1}^{n+\tau_2,n} \\ & & k_{u,u}^{n+\tau_1,n+\tau_1} & k_{u,3}^{n+\tau_1,n} & k_{u,2}^{n+\tau_1,n} & k_{u,1}^{n+\tau_1,n} \\ & & & k_{3,3}^{n,n} & k_{3,2}^{n,n} & k_{3,1}^{n,n} \\ & & & & k_{2,2}^{n,n} & k_{2,1}^{n,n} \\ & & & & & k_{1,1}^{n,n} \end{bmatrix}\right),
$$

where the covariance functions are given in section SM2 of the supplementary material (M112076_01.pdf [local/web 126KB]).

3.1.2. Training. The hyper-parameters θ_{n+1}, $\theta_{n+\tau_2}$, and $\theta_{n+\tau_1}$ can be trained by minimizing the negative log marginal likelihood resulting from

$$
\begin{bmatrix} u^{n+1}(1) - u^{n+1}(0) \\ u^{n+\tau_2}(1) - u^{n+\tau_2}(0) \\ u^{n+\tau_1}(1) - u^{n+\tau_1}(0) \\ \boldsymbol{u}_3^n \\ \boldsymbol{u}_2^n \\ \boldsymbol{u}_1^n \end{bmatrix} \sim \mathcal{N}(0, \boldsymbol{K}). \tag{39}
$$

Here, $u^{n+1}(1) - u^{n+1}(0) = 0$, $u^{n+\tau_2}(1) - u^{n+\tau_2}(0) = 0$, and $u^{n+\tau_1}(1) - u^{n+\tau_1}(0) = 0$ correspond to the periodic boundary condition (36). Moreover, $\boldsymbol{u}_3^n = \boldsymbol{u}_2^n = \boldsymbol{u}_1^n = \boldsymbol{u}^n$ and $\{\boldsymbol{x}^n, \boldsymbol{u}^n\}$ are the artificially generated data. This last equality reveals a key feature of this Runge–Kutta *numerical Gaussian process*, namely the fact that it inspects the same data through the lens of different kernels. This observation directly results from applying the 2-stage implicit Runge–Kutta scheme (i.e., the 2-stage Gauss–Legendre method) described in (37) to the advection equation. Specifically, the joint distribution in (39) follows directly after applying the periodic boundary conditions on the two Runge–Kutta stages and the final solution u^{n+1}, as well as from the consistent Runge–Kutta discretization shown in (37). A detailed derivation of the covariance

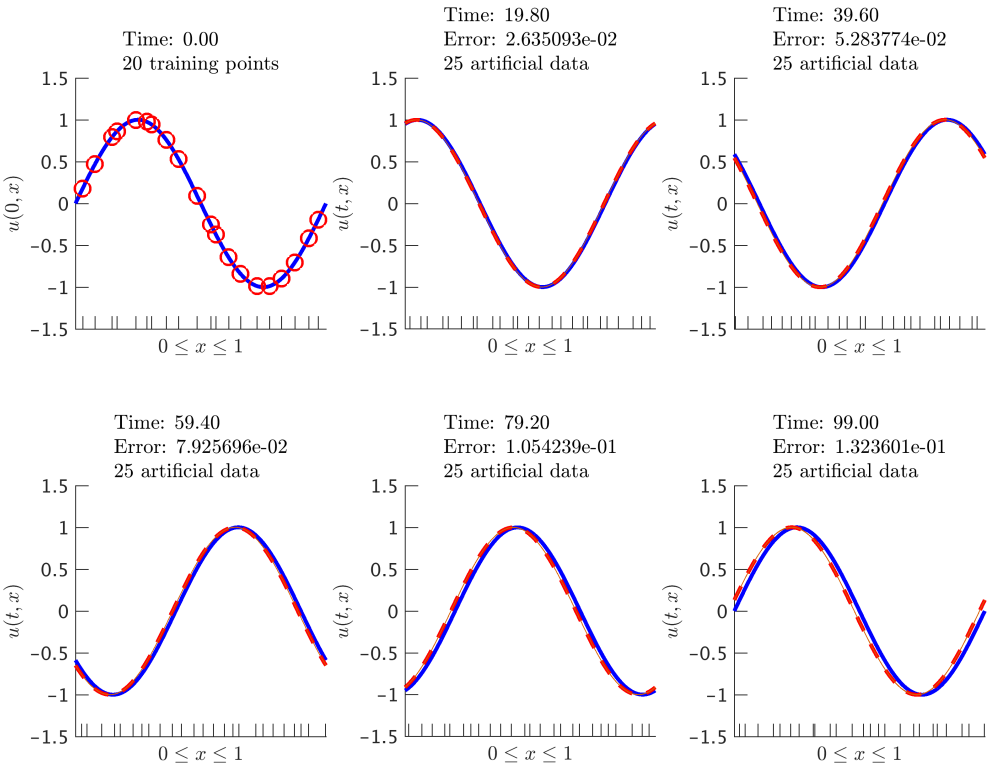

FIG. 12. Advection equation: *Initial data along with the posterior distribution of the solution at different time snapshots. The solid (blue) line represents the true data generating solution, while the dashed (red) line depicts the posterior mean. The shaded (orange) region illustrates the two-standard deviations band around the mean. At each time-step we generate* 25 *artificial data points randomly located in the interval* $[0, 1]$ *according to a uniform distribution. These locations are denoted by the ticks along the horizontal axis. We are employing the Gauss–Legendre time-stepping quadrature with time-step size* $\Delta t = 0.1$. *It is worth highlighting that we are running the time-stepping scheme for a very long time with a relatively large time-step size. We report here the relative spatial* $\mathcal{L}^2$*-error between the posterior mean and the true solution. (See the supplementary code:* http://bit.ly/2mpOtfQ*; Movie:* http://bit.ly/2m6XE2h.*)*

matrix $\boldsymbol{K}$ is given in section SM2 of the supplementary material (M112076_01.pdf [local/web 126KB]). Prediction and propagation of the uncertainty associated with the noisy initial observations can be performed as in section SM2. Figure 11 depicts the noisy initial data along with the posterior distribution (SM6 of the supplementary material) of the solution to the advection equation (35) at different time snapshots.

3.1.3. Numerical study. In the case when we have access to noiseless initial data we obtain the results depicted in Figure 12. Moreover, in order to make sure that the *numerical Gaussian process* resulting from the Gauss–Legendre method (37) applied to the advection equation is indeed fourth-order accurate in time, we perform the numerical experiment reported in Figures 13 and 14. The qualitative analysis of the temporal as well as the spatial convergence properties (as seen in Figure 15) closely follows the conclusions drawn in section 2.6.2.

3.2. Example: Heat equation (trapezoidal rule). Revisiting the trapezoidal rule, equipped with the machinery introduced for the Runge–Kutta methods,

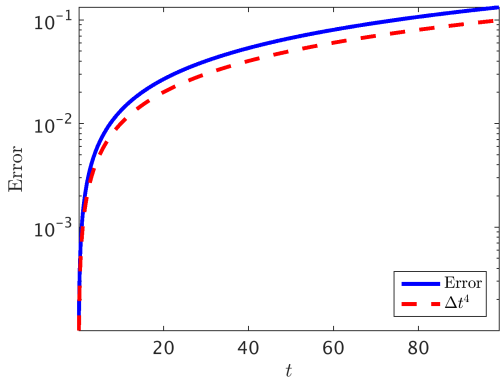

FIG. 13. Advection equation: *Time evolution of the relative spatial $\mathcal{L}^2$-error up to the final integration time $T = 99.0$. We are using the Gauss–Legendre implicit Runge–Kutta scheme with a time-step size of $\Delta t = 0.1$. The dashed (red) line illustrates the optimal fourth-order convergence rate. (See the supplementary code:* http://bit.ly/2mntVDh.*)*

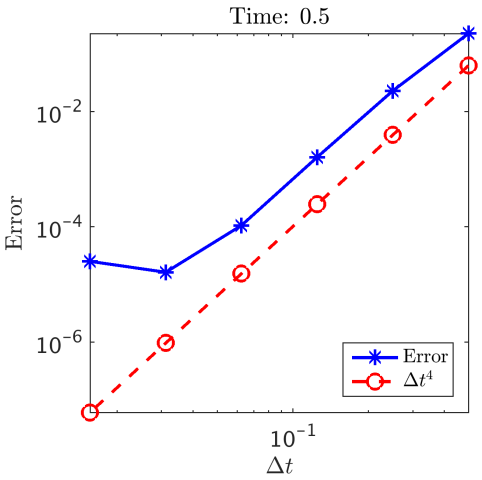

FIG. 14. Advection equation: *Relative spatial $\mathcal{L}^2$-error versus step-size for the Gauss–Legendre method. Here, the number of noiseless initial data as well as the artificially generated data is set to be equal to* 50. *We are running the time-stepping scheme up until time* 0.5 *is reached. (See the supplementary code:* http://bit.ly/2mntVDh.*)*

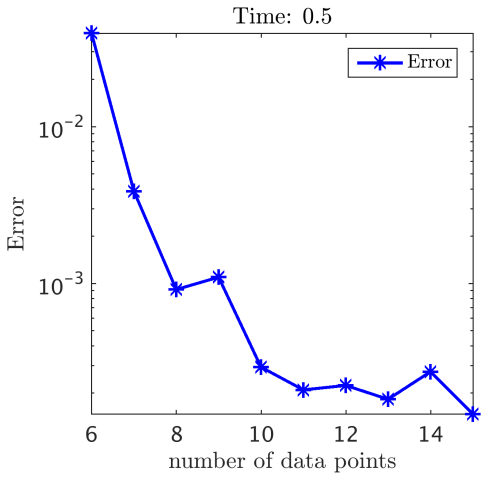

FIG. 15. Advection equation: *Relative spatial $\mathcal{L}^2$-error versus the number of noiseless initial as well as artificial data points used for the Gauss–Legendre method. Here, the time-step size is set to be $\Delta t = 0.1$. We are running the time-stepping scheme up until time* 0.5 *is reached. (See the supplementary code:* http://bit.ly/2mntVDh.*)*

we obtain an alternative *numerical Gaussian process* to that proposed in section 2. We will apply the resulting scheme to the heat equation in two space dimensions, i.e.,

$$u_t = u_{x_1x_1} + u_{x_2x_2}, \quad x_1 \in [0,1],\ x_2 \in [0,1]. \tag{40}$$

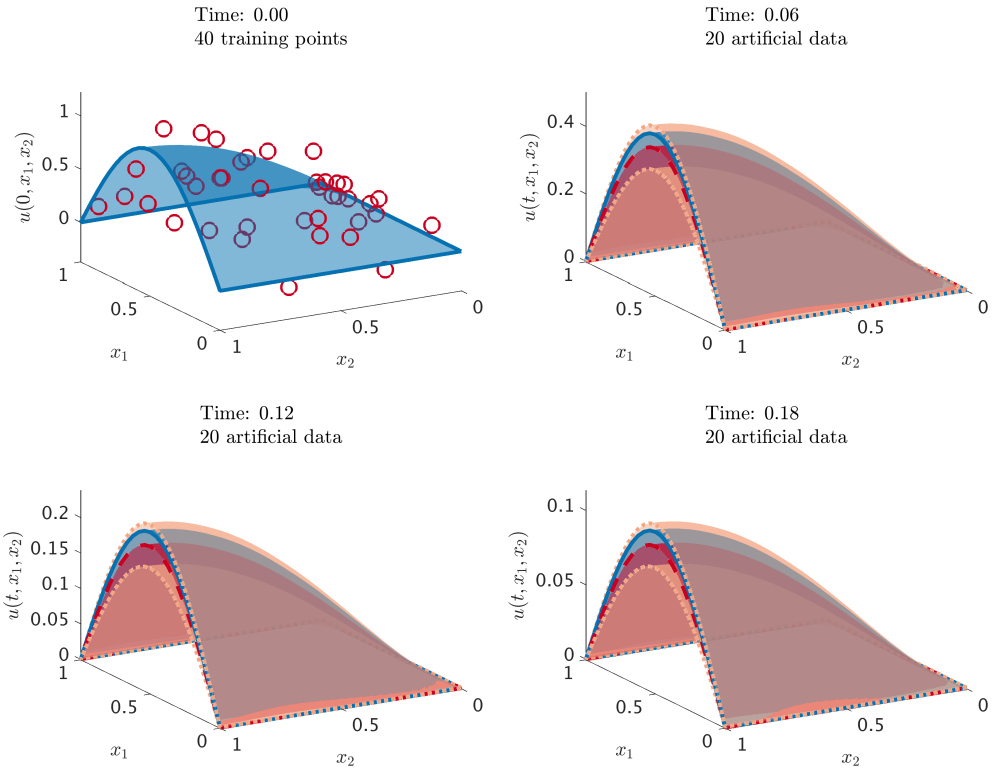

FIG. 16. Heat equation: *Initial data along with the posterior distribution of the solution at different time snapshots. The blue surface with solid lines represents the true data generating solution, while the red surface with dashed lines depicts the posterior mean. The two-standard deviations band around the mean is depicted using the orange surface with dotted boundary. We are employing the trapezoidal rule with time-step size* $\Delta t = 0.01$. *At each time-step we generate* 20 *artificial data points randomly located in the domain* $[0, 1] \times [0, 1]$ *according to a uniform distribution. We employ three noiseless data-points per boundary. (See the supplementary code:* http://bit.ly/2mnFpGS*; Movie:* http://bit.ly/2mq4UZt.*)*

The function $u(t, x_1, x_2) = e^{-\frac{5\pi^2 t}{4}} \sin(\pi x_1) \sin\left(\frac{\pi x_2}{2}\right)$ solves this equation and satisfies the following initial and boundary conditions:

$$u(0, x_1, x_2) = \sin(\pi x_1) \sin\left(\frac{\pi x_2}{2}\right),$$

$$u(t, 0, x_2) = u(t, 1, x_2) = 0, \qquad u(t, x_1, 0) = 0, \tag{41}$$

$$u_{x_2}(t, x_1, 1) = 0. \tag{42}$$

Equations (41) involve Dirichlet boundary conditions, while (42) corresponds to a Neumann-type boundary. Let us assume that all we observe are noisy measurements $\{(\boldsymbol{x}_1^0, \boldsymbol{x}_2^0), \boldsymbol{u}^0\}$ of the *black-box* initial function $u(0, x_1, x_2)$. Given such measurements, we would like to infer the latent scalar field $u(t, x_1, x_2)$ (i.e., the solution to the heat equation (40)), while quantifying the uncertainty associated with the noisy initial data (see Figure 16). This example showcases the ability of the proposed methods to handle multidimensional spatial domains and mixed boundary conditions (see (41) and (42)). Let us apply the trapezoidal scheme to the heat equation (40). The trapezoidal rule

for the heat equation is given by

$$u^{n+1} = u^n + \frac{1}{2}\Delta t \frac{d^2}{dx_1{}^2} u^n + \frac{1}{2}\Delta t \frac{d^2}{dx_1{}^2} u^{n+1} + \frac{1}{2}\Delta t \frac{d^2}{dx_2{}^2} u^n + \frac{1}{2}\Delta t \frac{d^2}{dx_2{}^2} u^{n+1}. \tag{43}$$

Rearranging the terms, we can write $u_1^n := u^n$ and

$$u_2^n := u_3^n := u^n = u^{n+1} - \frac{1}{2}\Delta t \frac{d^2}{dx_1^2} u^n - \frac{1}{2}\Delta t \frac{d^2}{dx_1^2} u^{n+1} - \frac{1}{2}\Delta t \frac{d^2}{dx_2^2} u^n - \frac{1}{2}\Delta t \frac{d^2}{dx_2^2} u^{n+1}. \tag{44}$$

In other words, we are just rewriting (37) for the heat equation (40) with $\tau_1 = 0$, $\tau_2 = 1$, $b_1 = b_2 = \frac{1}{2}$, $a_{11} = a_{12} = 0$, and $a_{21} = a_{22} = 1/2$.

3.2.1. Prior. Similar to the strategy (34) adopted for the Runge–Kutta methods, and as an alternative to the scheme used in section 2, we make the following prior assumptions:

$$\begin{aligned} u^{n+1}(x_1, x_2) &\sim \mathcal{GP}(0, k_{u,u}^{n+1,n+1}((x_1, x_2), (x_1', x_2'); \theta_{n+1})), \\ u^n(x_1, x_2) &\sim \mathcal{GP}(0, k_{u,u}^{n,n}((x_1, x_2), (x_1', x_2'); \theta_n)). \end{aligned} \tag{45}$$

Here, we employ anisotropic squared exponential covariance functions of the form

$$\begin{aligned} &k_{u,u}^{n+1,n+1}((x_1, x_2), (x_1', x_2'); \theta_{n+1}) \\ &\quad = \gamma_{n+1}^2 \exp\left(-\frac{1}{2} w_{n+1,1}(x_1 - x_1')^2 - \frac{1}{2} w_{n+1,2}(x_2 - x_2')^2\right), \\ &k_{u,u}^{n,n}((x_1, x_2), (x_1', x_2'); \theta_n) \\ &\quad = \gamma_n^2 \exp\left(-\frac{1}{2} w_{n,1}(x_1 - x_1')^2 - \frac{1}{2} w_{n,2}(x_2 - x_2')^2\right). \end{aligned}$$

The kernel hyper-parameters are given by $\theta_{n+1} = (\gamma_{n+1}^2, w_{n+1,1}, w_{n+1,2})$ and $\theta_n = (\gamma_n^2, w_{n,1}, w_{n,2})$. To deal with the mixed boundary conditions (41) and (42), let us define $v^{n+1} := \frac{d}{dx_2} u^{n+1}$ and $v^n := \frac{d}{dx_2} u^n$. We obtain the following *numerical Gaussian process:*

$$\begin{bmatrix} u^{n+1} \\ v^{n+1} \\ u^n \\ v^n \\ u_3^n \\ u_1^n \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k_{u,u}^{n+1,n+1} & k_{u,v}^{n+1,n+1} & 0 & 0 & k_{u,3}^{n+1,n} & 0 \\ & k_{v,v}^{n+1,n+1} & 0 & 0 & k_{v,3}^{n+1,n} & 0 \\ & & k_{u,u}^{n,n} & k_{u,v}^{n,n} & k_{u,3}^{n,n} & k_{u,1}^{n,n} \\ & & & k_{v,v}^{n,n} & k_{v,3}^{n,n} & k_{v,1}^{n,n} \\ & & & & k_{3,3}^{n,n} & k_{3,1}^{n,n} \\ & & & & & k_{1,1}^{n,n} \end{bmatrix}\right),$$

where the covariance functions are given in section SM3 of the supplementary material (M112076_01.pdf [local/web 126KB]).

3.2.2. Training. The hyper-parameters θ_{n+1} and θ_n can be trained by minimizing the negative log marginal likelihood resulting from

$$\begin{bmatrix} \boldsymbol{u}_D^{n+1} \\ \boldsymbol{v}_N^{n+1} \\ \boldsymbol{u}_D^{n} \\ \boldsymbol{v}_N^{n} \\ \boldsymbol{u}_3^{n} \\ \boldsymbol{u}_1^{n} \end{bmatrix} \sim \mathcal{N}(0, \boldsymbol{K}), \tag{46}$$

where $\{(\boldsymbol{x}_{1,D}^{n+1}, \boldsymbol{x}_{2,D}^{n+1}), \boldsymbol{u}_D^{n+1}\}$ and $\{(\boldsymbol{x}_{1,D}^{n}, \boldsymbol{x}_{2,D}^{n}), \boldsymbol{u}_D^{n}\}$ denote the data on the Dirichlet (41) portion of the boundary, while

$$\{(\boldsymbol{x}_{1,N}^{n+1}, \boldsymbol{x}_{2,N}^{n+1}), \boldsymbol{u}_N^{n+1}\} \quad \text{and} \quad \{(\boldsymbol{x}_{1,N}^{n}, \boldsymbol{x}_{2,N}^{n}), \boldsymbol{u}_N^{n}\}$$

correspond to the Neumann (42) boundary data. Moreover, $\boldsymbol{u}_1^n = \boldsymbol{u}_3^n = \boldsymbol{u}^n$ and $\{(\boldsymbol{x}_1^n, \boldsymbol{x}_2^n), \boldsymbol{u}^n\}$ are the artificially generated data. The exact form of the covariance matrix $\boldsymbol{K}$ is given in section SM3 of the supplementary material (M112076_01.pdf [local/web 126KB]). Prediction and propagation of uncertainty associated with the noisy initial observations can be performed as in section SM3 of the supplementary material. Figure 16 depicts the noisy initial data along with the posterior distribution (SM11 of the supplementary material) of the solution to the heat equation (40) at different time snapshots.

3.2.3. Numerical study. In order to be able to perform a systematic numerical study of the proposed methodology, we will operate under the assumption that we have access to *noiseless* initial data. The corresponding results are reported in Figure 17. Moreover, in order to make sure that the *numerical Gaussian process* resulting from the Runge–Kutta version of the trapezoidal rule (43) applied to the heat equation is indeed second-order accurate in time, we perform the numerical experiments reported in Figures 18 and 19. Again, the qualitative analysis of the temporal as well as the spatial convergence properties (as seen in Figure 20) closely follows the conclusions drawn in section 2.6.2.

4. Concluding remarks. We have presented a novel machine learning framework for encoding physical laws described by partial differential equations into Gaussian process priors for nonparametric Bayesian regression. The proposed algorithms can be used to infer solutions to time-dependent and nonlinear partial differential equations, and effectively quantify and propagate uncertainty due to noisy initial or boundary data. Moreover, to the best of our knowledge, this is the first attempt to construct structured learning machines which are explicitly informed by the underlying physics that possibly generated the observed data. Exploiting this structure is critical for constructing data-efficient learning algorithms that can effectively distill information in the data-scarce scenarios appearing routinely when we study complex physical systems. In contrast to classical deterministic numerical methods for solving partial differential equations (e.g., finite difference and finite-element methods), the proposed approach is by construction capable of propagating entire probability distributions in time. Although popular methods such as Monte Carlo sampling or probabilistic collocation could be employed for propagating uncertainty, their applicability depends on the cost of repeatedly sampling a forward model for solving the PDE for a given initial condition. However, all such sampling-based methods are well known to suffer from the curse of dimensionality; i.e., their convergence either

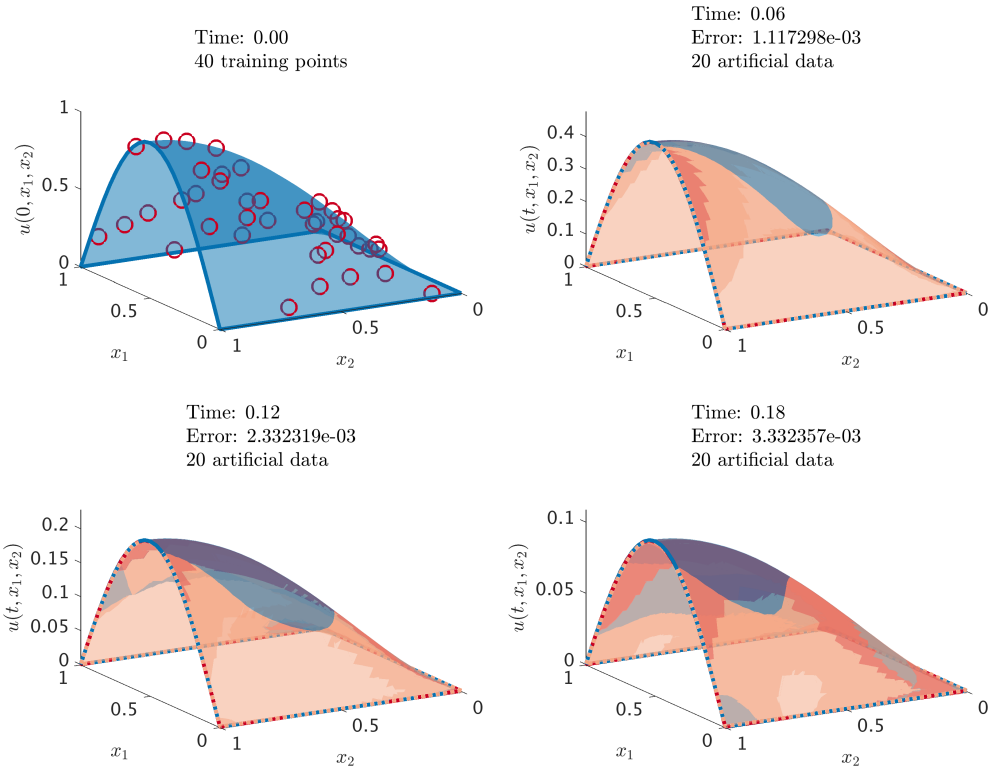

FIG. 17. Heat equation: *Initial data along with the posterior distribution of the solution at different time snapshots. The blue surface with solid lines represents the true data generating solution, while the red surface with dashed lines depicts the posterior mean. The two-standard deviations band around the mean is depicted using the orange surface with dotted boundary. We are employing the trapezoidal rule with time-step size* $\Delta t = 0.01$. *At each time-step we generate* 20 *artificial data points randomly located in the domain* $[0, 1] \times [0, 1]$ *according to a uniform distribution. We employ three noiseless data-points per boundary. We are reporting the relative* $\mathcal{L}^2$*-error between the posterior mean and the true solution. (See the supplementary code:* http://bit.ly/2mLwyB6*; Movie:* http://bit.ly/2mnFRod.*)*

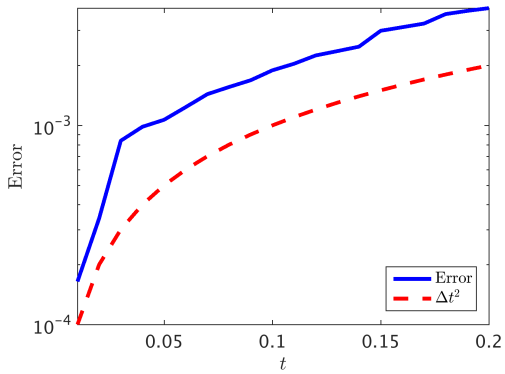

FIG. 18. Heat equation: *Time evolution of the relative spatial* $\mathcal{L}^2$*-error up to the final integration time* $T = 0.2$. *We are using the trapezoidal rule with a time-step size of* $\Delta t = 0.01$, *and the red dashed line illustrates the optimal second-order convergence rate. (See the supplementary code:* http://bit.ly/2m7aoG9.*)*

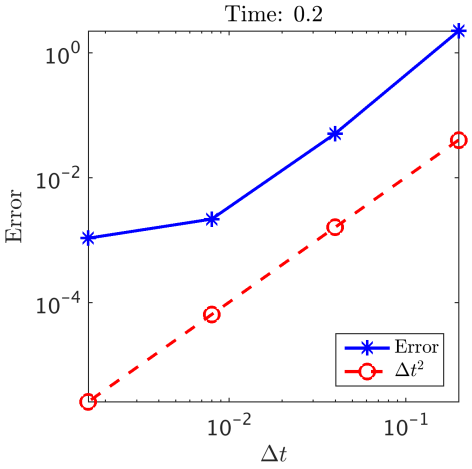

FIG. 19. Heat equation: *Relative spatial $\mathcal{L}^2$-error versus step-size for the Runge–Kutta version of the trapezoidal rule at time $T = 0.2$. Here, the number of noiseless initial data as well as the artificially generated data is set to be equal to* 50. *We are running the time-stepping scheme up until time* 0.2 *is reached. We employ* 10 *noiseless data per boundary. (See the supplementary code:* http://bit.ly/2m7aoG9.*)*

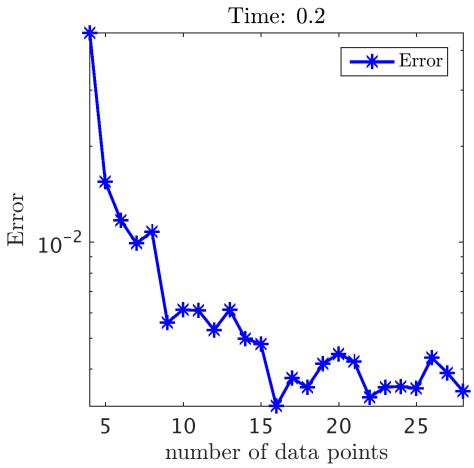

FIG. 20. Heat equation: *Relative spatial $\mathcal{L}^2$-error versus the number of noiseless initial as well as artificial data points used for the Runge–Kutta version of the trapezoidal rule at time $T = 0.2$. Here, the time-step size is set to be $\Delta t = 0.01$. We are running the time-stepping scheme up until time* 0.2 *is reached. We employ* 10 *noiseless data per boundary. (See the supplementary code:* http://bit.ly/2m7aoG9.*)*

becomes slower or requires an excessive number of samples as the problem dimension increases. This shortcoming is severely pronounced in our case as we are trying to propagate an infinite collection of jointly Gaussian correlated random variables (i.e., a Gaussian process) through the partial differential equation. Therefore, all these methods are susceptible to returning inaccurate uncertainty estimates or attain a very slow convergence rate. In contrast, the proposed approach leverages the properties of Gaussian processes and offers an exact treatment of this particular infinite dimensional uncertainty propagation problem. Although the proposed methodology provides a natural platform for learning from noisy data and computing under uncertainty, it comes with a nonnegligible computational cost. Specifically, a limitation of this work in its present form stems from the cubic scaling with respect to the total number of training data points.

In future work we plan to design more computationally efficient algorithms by exploring ideas including recursive Kalman updates [11] and variational inference [13]. From a classical numerical analysis standpoint, it also becomes natural to ask questions about convergence, derivation of dispersion relations, quantification of truncation errors, comparison against classical schemes, etc. We must underline that these questions become obsolete in the presence of noisy data and cannot be straightforwardly tackled using standard techniques from numerical analysis due to the probabilistic nature of the proposed work flow. In the realm of *numerical Gaussian processes* such questions translate into investigating theoretical concepts like prior consistency [26], posterior robustness [19], and posterior contraction rates [33]. These define a vast territory for analysis and future developments that currently remains unexplored. In terms of future work, we plan to leverage the proposed framework to study more complex physical systems (e.g., fluid flows via the Navier–Stokes prior); propose ex-

tensions that can accommodate parameter inference, inverse, and model discovery problems [24]; and incorporate probabilistic time integration schemes that allow for a natural quantification of uncertainty due to time-stepping errors [30].

REFERENCES

[1] *Probabilistic Numerics*, discussion forum, http://probabilistic-numerics.org/index.html (2017).
[2] R. Alexander, *Diagonally implicit Runge–Kutta methods for stiff ODE's*, SIAM J. Numer. Anal., 14 (1977), pp. 1006–1021, https://doi.org/10.1137/0714068.
[3] N. Aronszajn, *Theory of reproducing kernels*, Trans. Amer. Math. Soc., 68 (1950), pp. 337–404.
[4] C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. Patera, *Spectral and finite difference solutions of the Burgers equation*, Comput. & Fluids, 14 (1986), pp. 23–41.
[5] F. Bashforth and J. C. Adams, *An Attempt to Test the Theories of Capillary Action: By Comparing the Theoretical and Measured Forms of Drops of Fluid. With an Explanation of the Method of Integration Employed in Constructing the Tables Which Give the Theoretical Forms of Such Drops*, University Press, 1883.
[6] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, Springer Science & Business Media, 2011.
[7] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, 2016.
[8] P. R. Conrad, M. Girolami, S. Särkkä, A. Stuart, and K. Zygalakis, *Statistical analysis of differential equations: Introducing probability measures on numerical solutions*, Stat. Comput., 27 (2017), pp. 1065–1082.
[9] P. Diaconis, *Bayesian numerical analysis*, Statist. Decision Theory Rel. Topics IV, 1 (1988), pp. 163–175.
[10] Z. Ghahramani, *Probabilistic machine learning and artificial intelligence*, Nature, 521 (2015), pp. 452–459.
[11] J. Hartikainen and S. Särkkä, *Kalman filtering and smoothing solutions to temporal Gaussian process regression models*, in Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2010, pp. 379–384.
[12] P. Hennig, M. A. Osborne, and M. Girolami, *Probabilistic numerics and uncertainty in computations*, Proc. Roy. Soc. London A, 471 (2015), 20150142, https://doi.org/10.1098/rspa.2015.0142.
[13] J. Hensman, N. Fusi, and N. D. Lawrence, *Gaussian processes for big data*, Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2013, pp. 282–290.
[14] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, Cambridge, UK, 1996, Chinese translation: Springer & Tsing-hua University Press, 2005, 2nd ed.: 2008..
[15] M. Jordan and T. Mitchell, *Machine learning: Trends, perspectives, and prospects*, Science, 349 (2015), pp. 255–260.
[16] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.
[17] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.
[18] R. M. Neal, *Bayesian Learning for Neural Networks*, Springer Science & Business Media 118, 2012.
[19] H. Owhadi, C. Scovel, and T. Sullivan, *Brittleness of Bayesian inference under finite information in a continuous world*, Electron. J. Statist., 9 (2015), pp. 1–79.
[20] A. O'Hagan, *Some Bayesian numerical analysis*, Bayesian Statist., 4 (1992), pp. 345–363.
[21] T. Poggio and F. Girosi, *Networks for approximation and learning*, Proc. IEEE, 78 (1990), pp. 1481–1497.
[22] H. Poincaré, *Calcul des Probabilités*, Gauthier-Villars, Paris, 1896.
[23] M. Raissi, *Parametric Gaussian Process Regression for Big Data*, preprint, arXiv:1704.03144, 2017.
[24] M. Raissi and G. E. Karniadakis, *Machine Learning of Linear Differential Equations Using Gaussian Processes*, preprint, arXiv:1701.02440, 2017.
[25] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Inferring solutions of differential equations using noisy multi-fidelity data*, J. Comput. Phys., 335 (2017), pp. 736–746.
[26] C. E. Rasmussen, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[27] C. E. Rasmussen and Z. Ghahramani, *Occam's razor*, in Advances in Neural Information Processing Systems, Morgan Kaufmann, 2001, pp. 294–300.
[28] C. Runge, *Über die numerische Auflösung von Differentialgleichungen*, Math. Annal., 46 (1895), pp. 167–178.
[29] S. Saitoh, *Theory of Reproducing Kernels and Its Applications*, Developments in Mathematics 44, Springer Science+Business Media, Singapore, 2016, https://doi.org/10.1007/978-981-10-0530-5.
[30] M. Schober, D. K. Duvenaud, and P. Hennig, *Probabilistic ODE solvers with Runge-Kutta means*, in Advances in Neural Information Processing Systems, Morgan Kaufmann, 2014, pp. 739–747.
[31] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.
[32] E. Snelson and Z. Ghahramani, *Sparse Gaussian processes using pseudo-inputs*, in Advances in Neural Information Processing Systems, Morgan Kaufmann, 2006, pp. 1257–1264.
[33] A. M. Stuart and A. L. Teckentrup, *Posterior consistency for Gaussian process approximations of Bayesian posterior distributions*, Math. Comp., 87 (2018), pp. 721–753, https://doi.org/10.1090/mcom/3244.
[34] A. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Soviet Math. Dokl., 5 (1963), pp. 1035–1038.
[35] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, W.H. Winston, 1977.
[36] M. E. Tipping, *Sparse Bayesian learning and the relevance vector machine*, J. Machine Learning Res., 1 (2001), pp. 211–244.
[37] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer Science & Business Media, 2013.